# Arria GX Device Handbook,
## Volume 1

**nsai**

**I.S. EN ISO 9001**

# Contents

## Chapter 5. Reference and Ordering Information

## Additional Information

The chapters in this book, *Arria GX Device Handbook, Volume 1*, were revised on the following dates. Where chapters or groups of chapters are available separately, part numbers are listed.

Chapter 1   Arria GX Device Family Overview
Revised:        *December 2009*
Part Number: *AGX51001-2.0*

Chapter 2   Arria GX Architecture
Revised:        *December 2009*
Part Number: *AGX51002-2.0*

Chapter 3   Configuration and Testing
Revised:        *December  2009*
Part Number: *AGX51003-2.0*

Chapter 4   DC and Switching Characteristics
Revised:        *December 2009*
Part Number: *AGX51004-2.0*

Chapter 5    Reference and Ordering Information
Revised:        *December 2009*
Part Number: *AGX51005-2.0*

# Section I. Arria GX Device Data Sheet

This section provides designers with the data sheet specifications for Arria® GX devices. They contain feature definitions of the transceivers, internal architecture, configuration, and JTAG boundary-scan testing information, DC operating conditions, AC timing parameters, a reference to power consumption, and ordering information for Arria GX devices.

This section includes the following chapters:

- Chapter 1, Arria GX Device Family Overview
- Chapter 2, Arria GX Architecture
- Chapter 3, Configuration and Testing
- Chapter 4, DC and Switching Characteristics
- Chapter 5, Reference and Ordering Information

## Revision History

Refer to each chapter for its own specific revision history. For information about when each chapter was updated, refer to the Chapter Revision Dates section, which appears in the full handbook.

# 1. Arria GX Device Family Overview

## Introduction

The Arria® GX family of devices combines 3.125 Gbps serial transceivers with reliable packaging technology and a proven logic array. Arria GX devices include 4 to 12 high-speed transceiver channels, each incorporating clock data recovery (CDR) technology and embedded SERDES circuitry designed to support PCI-Express, Gigabit Ethernet, SDI, SerialLite II, XAUI, and Serial RapidIO protocols, along with the ability to develop proprietary, serial-based IP using its Basic mode. The transceivers build upon the success of the Stratix® II GX family. The Arria GX FPGA technology offers a 1.2-V logic array with the right level of performance and dependability needed to support these mainstream protocols.

## Features

The key features of Arria GX devices include:

- Transceiver block features

    - High-speed serial transceiver channels with CDR support up to 3.125 Gbps.

    - Devices available with 4, 8, or 12 high-speed full-duplex serial transceiver channels

    - Support for the following CDR-based bus standards—PCI Express, Gigabit Ethernet, SDI, SerialLite II, XAUI, and Serial RapidIO, along with the ability to develop proprietary, serial-based IP using its Basic mode

    - Individual transmitter and receiver channel power-down capability for reduced power consumption during non-operation

    - 1.2- and 1.5-V pseudo current mode logic (PCML) support on transmitter output buffers

    - Receiver indicator for loss of signal (available only in PCI Express [PIPE] mode)

    - Hot socketing feature for hot plug-in or hot swap and power sequencing support without the use of external devices

    - Dedicated circuitry that is compliant with PIPE, XAUI, Gigabit Ethernet, Serial Digital Interface (SDI), and Serial RapidIO

    - 8B/10B encoder/decoder performs 8-bit to 10-bit encoding and 10-bit to 8-bit decoding

    - Phase compensation FIFO buffer performs clock domain translation between the transceiver block and the logic array

    - Channel aligner compliant with XAUI

■ Main device features:

   ■ TriMatrix memory consisting of three RAM block sizes to implement true dual-port memory and first-in first-out (FIFO) buffers with performance up to 380 MHz

   ■ Up to 16 global clock networks with up to 32 regional clock networks per device

   ■ High-speed DSP blocks provide dedicated implementation of multipliers, multiply-accumulate functions, and finite impulse response (FIR) filters

   ■ Up to four enhanced phase-locked loops (PLLs) per device provide spread spectrum, programmable bandwidth, clock switch-over, and advanced multiplication and phase shifting

   ■ Support for numerous single-ended and differential I/O standards

   ■ High-speed source-synchronous differential I/O support on up to 47 channels

   ■ Support for source-synchronous bus standards, including SPI-4 Phase 2 (POS-PHY Level 4), SFI-4.1, XSBI, UTOPIA IV, NPSI, and CSIX-L1

   ■ Support for high-speed external memory including DDR and DDR2 SDRAM, and SDR SDRAM

   ■ Support for multiple intellectual property megafunctions from Altera® MegaCore® functions and Altera Megafunction Partners Program (AMPPᔆᴹ)

   ■ Support for remote configuration updates

Table 1–1 lists Arria GX device features for FineLine BGA (FBGA) with flip chip packages.

**Table 1–1.** Arria GX Device Features  (Part 1 of 2)

| Feature | EP1AGX20C | EP1AGX35C/D | | EP1AGX50C/D | | EP1AGX60C/D/E | | | EP1AGX90E |
|---|---|---|---|---|---|---|---|---|---|
| | C | C | D | C | D | C | D | E | E |
| Package | 484-pin, 780-pin (Flip chip) | 484-pin (Flip chip) | 780-pin (Flip chip) | 484-pin (Flip chip) | 780-pin, 1152-pin (Flip chip) | 484-pin (Flip chip) | 780-pin (Flip chip) | 1152-pin (Flip chip) | 1152-pin (Flip chip) |
| ALMs | 8,632 | 13,408 | | 20,064 | | 24,040 | | | 36,088 |
| Equivalent logic elements (LEs) | 21,580 | 33,520 | | 50,160 | | 60,100 | | | 90,220 |
| Transceiver channels | 4 | 4 | 8 | 4 | 8 | 4 | 8 | 12 | 12 |
| Transceiver data rate | 600 Mbps to 3.125 Gbps | 600 Mbps to 3.125 Gbps | | 600 Mbps to 3.125 Gbps | | 600 Mbps to 3.125 Gbps | | | 600 Mbps to 3.125 Gbps |
| Source-synchronous receive channels | 31 | 31 | 31 | 31 | 31, 42 | 31 | 31 | 42 | 47 |

**Table 1–1.** Arria GX Device Features  (Part 2 of 2)

| Feature | EP1AGX20C | EP1AGX35C/D | | EP1AGX50C/D | | EP1AGX60C/D/E | | | EP1AGX90E |
|---|---|---|---|---|---|---|---|---|---|
| | C | C | D | C | D | C | D | E | E |
| Source-synchronous transmit channels | 29 | 29 | 29 | 29 | 29, 42 | 29 | 29 | 42 | 45 |
| M512 RAM blocks (32 × 18 bits) | 166 | 197 | | 313 | | 326 | | | 478 |
| M4K RAM blocks (128 × 36 bits) | 118 | 140 | | 242 | | 252 | | | 400 |
| M-RAM blocks (4096 × 144 bits) | 1 | 1 | | 2 | | 2 | | | 4 |
| Total RAM bits | 1,229,184 | 1,348,416 | | 2,475,072 | | 2,528,640 | | | 4,477,824 |
| Embedded multipliers (18 × 18) | 40 | 56 | | 104 | | 128 | | | 176 |
| DSP blocks | 10 | 14 | | 26 | | 32 | | | 44 |
| PLLs | 4 | 4 | | 4 | 4, 8 | 4 | | 8 | 8 |
| Maximum user I/O pins | 230, 341 | 230 | 341 | 229 | 350, 514 | 229 | 350 | 514 | 538 |

Arria GX devices are available in space-saving FBGA packages (refer to Table 1–2). All Arria GX devices support vertical migration within the same package. With vertical migration support, designers can migrate to devices whose dedicated pins, configuration pins, and power pins are the same for a given package across device densities. For I/O pin migration across densities, the designer must cross-reference the available I/O pins with the device pin-outs for all planned densities of a given package type to identify which I/O pins are migratable.

**Table 1–2.** Arria GX Package Options (Pin Counts and Transceiver Channels)  (Part 1 of 2)

| Device | Transceiver Channels | Source-Synchronous Channels | | Maximum User I/O Pin Count | | |
|---|---|---|---|---|---|---|
| | | Receive | Transmit | 484-Pin FBGA (23 mm) | 780-Pin FBGA (29 mm) | 1152-Pin FBGA (35 mm) |
| EP1AGX20C | 4 | 31 | 29 | 230 | 341 | — |
| EP1AGX35C | 4 | 31 | 29 | 230 | — | — |
| EP1AGX50C | 4 | 31 | 29 | 229 | — | — |
| EP1AGX60C | 4 | 31 | 29 | 229 | — | — |
| EP1AGX35D | 8 | 31 | 29 | — | 341 | — |
| EP1AGX50D | 8 | 31, 42 | 29, 42 | — | 350 | 514 |

**Table 1–2.** Arria GX Package Options (Pin Counts and Transceiver Channels)  (Part 2 of 2)

| Device | Transceiver Channels | Source-Synchronous Channels | | Maximum User I/O Pin Count | | |
| --- | --- | --- | --- | --- | --- | --- |
| | | Receive | Transmit | 484-Pin FBGA (23 mm) | 780-Pin FBGA (29 mm) | 1152-Pin FBGA (35 mm) |
| EP1AGX60D | 8 | 31 | 29 | — | 350 | — |
| EP1AGX60E | 12 | 42 | 42 | — | — | 514 |
| EP1AGX90E | 12 | 47 | 45 | — | — | 538 |

Table 1–3 lists the Arria GX device package sizes.

**Table 1–3.** Arria GX FBGA Package Sizes

| Dimension | 484 Pins | 780 Pins | 1152 Pins |
| --- | --- | --- | --- |
| Pitch (mm) | 1.00 | 1.00 | 1.00 |
| Area (mm$^2$) | 529 | 841 | 1225 |
| Length × width (mm × mm) | 23 × 23 | 29 × 29 | 35 × 35 |

# Document Revision History

Table 1–4 lists the revision history for this chapter.

**Table 1–4.** Document Revision History

| Date and Document Version | Changes Made | Summary of Changes |
| --- | --- | --- |
| December 2009, v2.0 | ■ Document template update.  ■ Minor text edits. | — |
| May 2008, v1.2 | Included support for SDI, SerialLite II, and XAUI. | — |
| June 2007, v1.1 | Included GIGE information. | — |
| May 2007, v1.0 | Initial Release | — |

# Transceivers

Arria® GX devices incorporate up to 12 high-speed serial transceiver channels that build on the success of the Stratix® II GX device family. Arria GX transceivers are structured into full-duplex (transmitter and receiver) four-channel groups called transceiver blocks located on the right side of the device. You can configure the transceiver blocks to support the following serial connectivity protocols (functional modes):

■ PCI Express (PIPE)

■ Gigabit Ethernet (GIGE)

■ XAUI

■ Basic (600 Mbps to 3.125 Gbps)

■ SDI (HD, 3G)

■ Serial RapidIO (1.25 Gbps, 2.5 Gbps, 3.125 Gbps)

Transceivers within each block are independent and have their own set of dividers. Therefore, each transceiver can operate at different frequencies. Each block can select from two reference clocks to provide two clock domains that each transceiver can select from.

Table 2–1 lists the number of transceiver channels for each member of the Arria GX family.

**Table 2–1.** Arria GX Transceiver Channels

| Device | Number of Transceiver Channels |
|--------|--------------------------------|
| EP1AGX20C | 4 |
| EP1AGX35C | 4 |
| EP1AGX35D | 8 |
| EP1AGX50C | 4 |
| EP1AGX50D | 8 |
| EP1AGX60C | 4 |
| EP1AGX60D | 8 |
| EP1AGX60E | 12 |
| EP1AGX90E | 12 |

Figure 2–1 shows a high-level diagram of the transceiver block architecture divided into four channels.

**Figure 2–1.** Transceiver Block
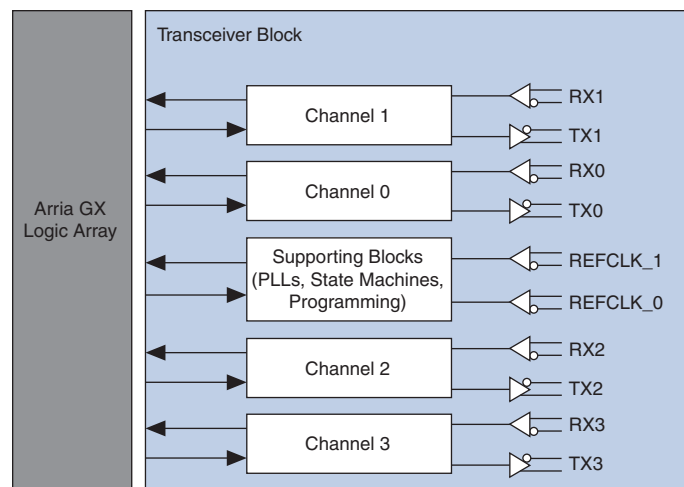


Each transceiver block has:

■ Four transceiver channels with dedicated physical coding sublayer (PCS) and physical media attachment (PMA) circuitry

■ One transmitter PLL that takes in a reference clock and generates high-speed serial clock depending on the functional mode

■ Four receiver PLLs and clock recovery unit (CRU) to recover clock and data from the received serial data stream

■ State machines and other logic to implement special features required to support each protocol

Figure 2–2 shows functional blocks that make up a transceiver channel.

**Figure 2–2.** Arria GX Transceiver Channel Block Diagram
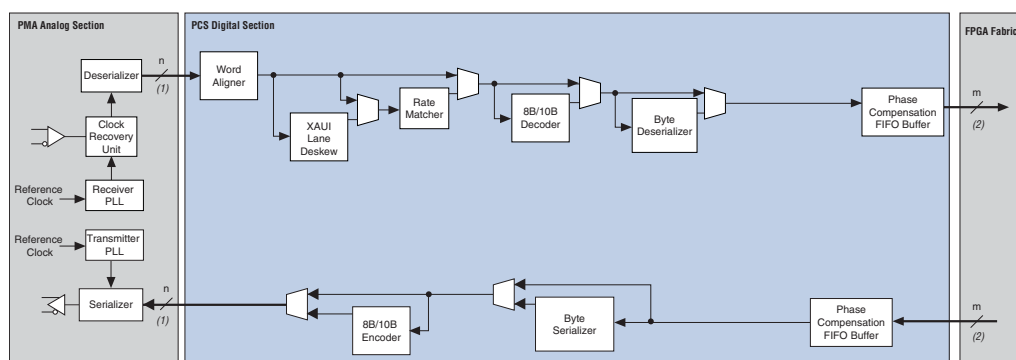


**Notes to Figure 2–2:**

(1) "n" represents the number of bits in each word that must be serialized by the transmitter portion of the PMA. n = 8 or 10.

(2) "m" represents the number of bits in the word that passes between the FPGA logic and the PCS portion of the transceiver. m = 8, 10, 16, or 20.

Each transceiver channel is full-duplex and consists of a transmitter channel and a receiver channel.

The transmitter channel contains the following sub-blocks:

■ Transmitter phase compensation first-in first-out (FIFO) buffer

■ Byte serializer (optional)

■ 8B/10B encoder (optional)

■ Serializer (parallel-to-serial converter)

■ Transmitter differential output buffer

The receiver channel contains the following:

■ Receiver differential input buffer

■ Receiver lock detector and run length checker

■ CRU

■ Deserializer

■ Pattern detector

■ Word aligner

■ Lane deskew

■ Rate matcher (optional)

■ 8B/10B decoder (optional)

■ Byte deserializer (optional)

■ Receiver phase compensation FIFO buffer

You can configure the transceiver channels to the desired functional modes using the ALT2GXB MegaCore instance in the Quartus® II MegaWizard™ Plug-in Manager for the Arria GX device family. Depending on the selected functional mode, the Quartus II software automatically configures the transceiver channels to employ a subset of the sub-blocks listed above.

## Transmitter Path

This section describes the data path through the Arria GX transmitter. The sub-blocks are described in order from the PLD-transmitter parallel interface to the serial transmitter buffer.

### Clock Multiplier Unit

Each transceiver block has a clock multiplier unit (CMU) that takes in a reference clock and synthesizes two clocks: a high-speed serial clock to serialize the data and a low-speed parallel clock to clock the transmitter digital logic (PCS).

The CMU is further divided into three sub-blocks:

■ One transmitter PLL

■ One central clock divider block

■ Four local clock divider blocks (one per channel)

Figure 2–3 shows the block diagram of the clock multiplier unit.

**Figure 2–3.** Clock Multiplier Unit
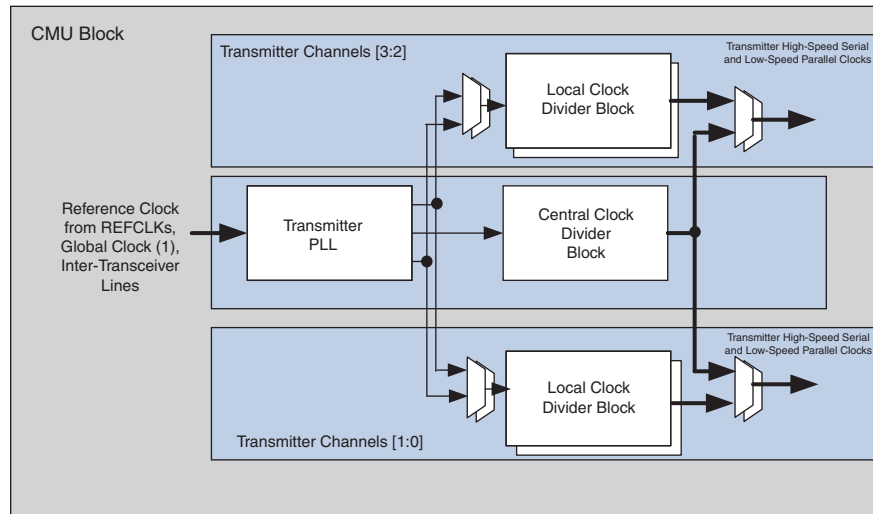


The transmitter PLL multiplies the input reference clock to generate the high-speed serial clock required to support the intended protocol. It implements a half-rate voltage controlled oscillator (VCO) that generates a clock at half the frequency of the serial data rate for which it is configured.

Figure 2–4 shows the block diagram of the transmitter PLL.

**Figure 2–4.** Transmitter PLL
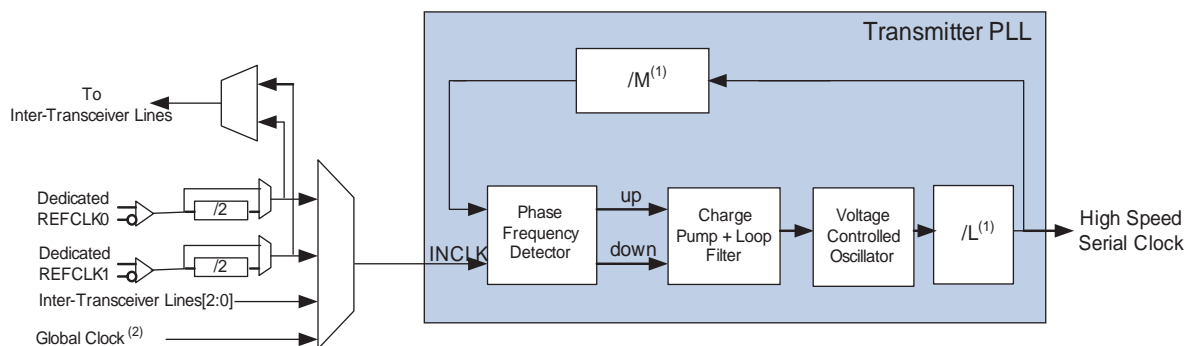


**Notes to Figure 2–4:**

(1)  You only need to select the protocol and the available input reference clock frequency in the ALTGXB MegaWizard Plug-In Manager. Based on your selections, the MegaWizard Plug-In Manager automatically selects the necessary /M and /L dividers (clock multiplication factors).

(2)  The global clock line must be driven from an input pin only.

The reference clock input to the transmitter PLL can be derived from:

■ One of two available dedicated reference clock input pins (REFCLK0 or REFCLK1) of the associated transceiver block

■ PLD global clock network (must be driven directly from an input clock pin and cannot be driven by user logic or enhanced PLL)

■ Inter-transceiver block lines driven by reference clock input pins of other transceiver blocks

☞ Altera® recommends using the dedicated reference clock input pins (REFCLK0 or REFCLK1) to provide reference clock for the transmitter PLL.

Table 2–2 lists the adjustable parameters in the transmitter PLL.

**Table 2–2.** Transmitter PLL Specifications

| Parameter | Specifications |
| --- | --- |
| Input reference frequency range | 50 MHz to 622.08 MHz |
| Data rate support | 600 Mbps to 3.125 Gbps |
| Bandwidth | Low, medium, or high |

The transmitter PLL output feeds the central clock divider block and the local clock divider blocks. These clock divider blocks divide the high-speed serial clock to generate the low-speed parallel clock for the transceiver PCS logic and PLD-transceiver interface clock.

### Transmitter Phase Compensation FIFO Buffer

A transmitter phase compensation FIFO is located at each transmitter channel's logic array interface. It compensates for the phase difference between the transmitter PCS clock and the local PLD clock. The transmitter phase compensation FIFO is used in all supported functional modes. The transmitter phase compensation FIFO buffer is eight words deep in PCI Express (PIPE) mode and four words deep in all other modes.

✎ For more information about architecture and clocking, refer to the *Arria GX Transceiver Architecture* chapter.

### Byte Serializer

The byte serializer takes in two-byte wide data from the transmitter phase compensation FIFO buffer and serializes it into a one-byte wide data at twice the speed. The transmit data path after the byte serializer is 8 or 10 bits. This allows clocking the PLD-transceiver interface at half the speed when compared with the transmitter PCS logic. The byte serializer is bypassed in GIGE mode. After serialization, the byte serializer transmits the least significant byte (LSByte) first and the most significant byte (MSByte) last.

Figure 2–5 shows byte serializer input and output. `datain[15:0]` is the input to the byte serializer from the transmitter phase compensation FIFO; `dataout[7:0]` is the output of the byte serializer.

**Figure 2–5.** Byte Serializer Operation *(Note 1)*
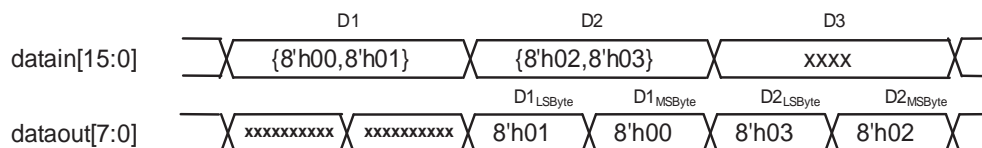


**Note to Figure 2–5:**

(1)  `datain` may be 16 or 20 bits. `dataout` may be 8 or 10 bits.

## 8B/10B Encoder

The 8B/10B encoder block is used in all supported functional modes. The 8B/10B encoder block takes in 8-bit data from the byte serializer or the transmitter phase compensation FIFO buffer. It generates a 10-bit code group with proper running disparity from the 8-bit character and a 1-bit control identifier (`tx_ctrlenable`). When `tx_ctrlenable` is low, the 8-bit character is encoded as data code group (`Dx.y`). When `tx_ctrlenable` is high, the 8-bit character is encoded as a control code group (`Kx.y`). The 10-bit code group is fed to the serializer. The 8B/10B encoder conforms to the IEEE 802.3 1998 edition standard.

For additional information regarding 8B/10B encoding rules, refer to the *Specifications and Additional Information* chapter.

Figure 2–6 shows the 8B/10B conversion format.

**Figure 2–6.** 8B/10B Encoder
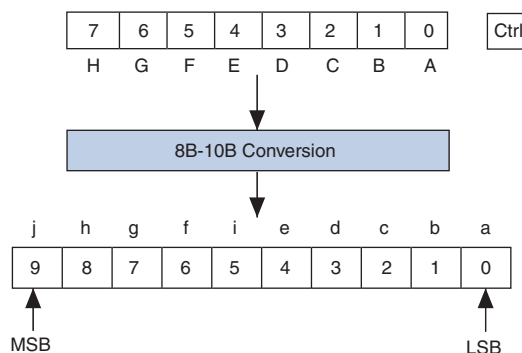


During reset (`tx_digitalreset`), the running disparity and data registers are cleared and the 8B/10B encoder continously outputs a K28.5 pattern from the RD-column. After out of reset, the 8B/10B encoder starts with a negative disparity (RD-) and transmits three K28.5 code groups for synchronizing before it starts encoding the input data or control character.

## Transmit State Machine

The transmit state machine operates in either PCI Express (PIPE) mode, XAUI mode, or GIGE mode, depending on the protocol used.

### GIGE Mode

In GIGE mode, the transmit state machine converts all idle ordered sets (/K28.5/, /Dx.y/) to either /I1/ or /I2/ ordered sets. The /I1/ set consists of a negative-ending disparity /K28.5/ (denoted by /K28.5/-), followed by a neutral /D5.6/. The /I2/ set consists of a positive-ending disparity /K28.5/ (denoted by /K28.5/+) and a negative-ending disparity /D16.2/ (denoted by /D16.2/-). The transmit state machines do not convert any of the ordered sets to match /C1/ or /C2/, which are the configuration ordered sets. (/C1/ and /C2/ are defined by [/K28.5/, /D21.5/] and [/K28.5/, /D2.2/], respectively). Both the /I1/ and /I2/ ordered sets guarantee a negative-ending disparity after each ordered set.

### XAUI Mode

The transmit state machine translates the XAUI XGMII code group to the XAUI PCS code group. Table 2–3 lists the code conversion.

**Table 2–3.** On-Chip Termination Support by I/O Banks

| XGMII TXC | XGMII TXD | PCS Code-Group | Description |
|---|---|---|---|
| 0 | 00 through FF | Dxx.y | Normal data |
| 1 | 07 | K28.0 or K28.3 or K28.5 | Idle in ||I|| |
| 1 | 07 | K28.5 | Idle in ||T|| |
| 1 | 9C | K28.4 | Sequence |
| 1 | FB | K27.7 | Start |
| 1 | FD | K29.7 | Terminate |
| 1 | FE | K30.7 | Error |
| 1 | Refer to IEEE 802.3 reserved code groups | Refer to IEEE 802.3 reserved code groups | Reserved code groups |
| 1 | Other value | K30.7 | Invalid XGMII character |

The XAUI PCS idle code groups, /K28.0/ (/R/) and /K28.5/ (/K/), are automatically randomized based on a PRBS7 pattern with an ×7 + ×6 + 1 polynomial. The /K28.3/ (/A/) code group is automatically generated between 16 and 31 idle code groups. The idle randomization on the /A/, /K/, and /R/ code groups is automatically done by the transmit state machine.

## Serializer (Parallel-to-Serial Converter)

The serializer block clocks in 8- or 10-bit encoded data from the 8B/10B encoder using the low-speed parallel clock and clocks out serial data using the high-speed serial clock from the central or local clock divider blocks. The serializer feeds the data LSB to MSB to the transmitter output buffer.

Figure 2–7 shows the serializer block diagram.

**Figure 2–7.** Serializer
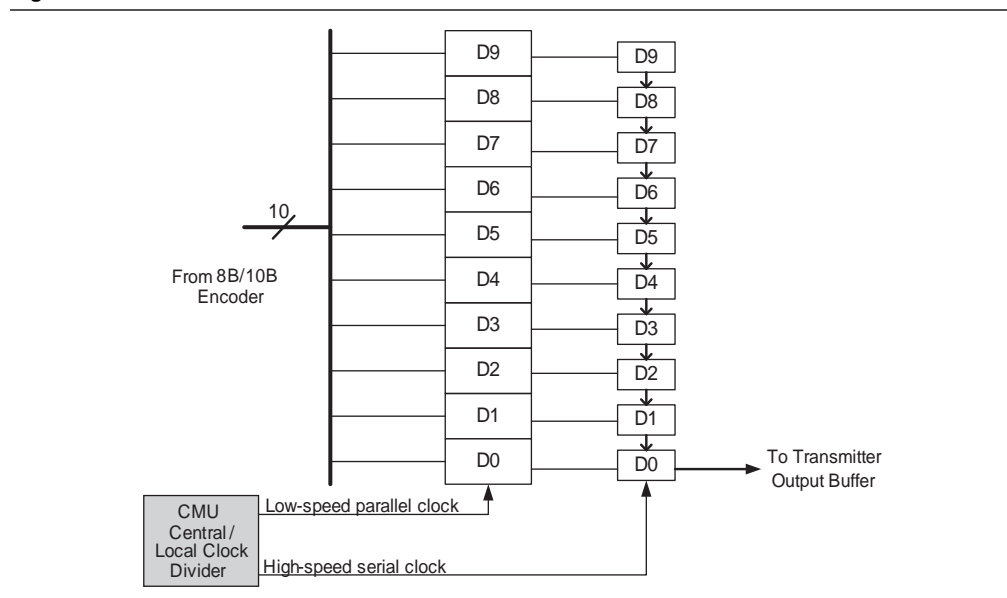


## Transmitter Buffer

The Arria GX transceiver buffers support the 1.2- and 1.5-V PCML I/O standard at rates up to 3.125 Gbps. The common mode voltage ($V_{CM}$) of the output driver may be set to 600 or 700 mV.

For more information about the Arria GX transceiver buffers, refer to the *Arria GX Transceiver Architecture* chapter.

The output buffer, as shown in Figure 2–8, is directly driven by the high-speed data serializer and consists of a programmable output driver, a programmable pre-emphasis circuit, and OCT circuitry.

**Figure 2–8.** Output Buffer
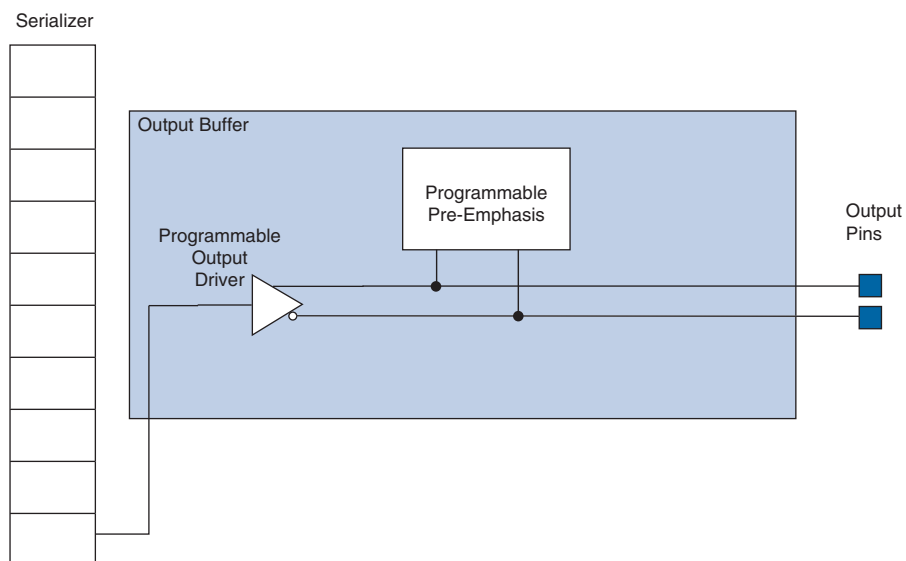


### Programmable Output Driver

The programmable output driver can be set to drive out differentially from 400 to 1200 mV. The differential output voltage ($V_{OD}$) can be statically set by using the ALTGXB megafunction.

You can configure the output driver with 100-Ω OCT or external OCT.

Differential signaling conventions are shown in Figure 2–9. The differential amplitude represents the value of the voltage between the true and complement signals. Peak-to-peak differential voltage is defined as 2 ($V_{HIGH} - V_{LOW}$) = 2 single-ended voltage swing. The common mode voltage is the average of $V_{HIGH}$ and $V_{LOW}$.

**Figure 2–9.** Differential Signaling
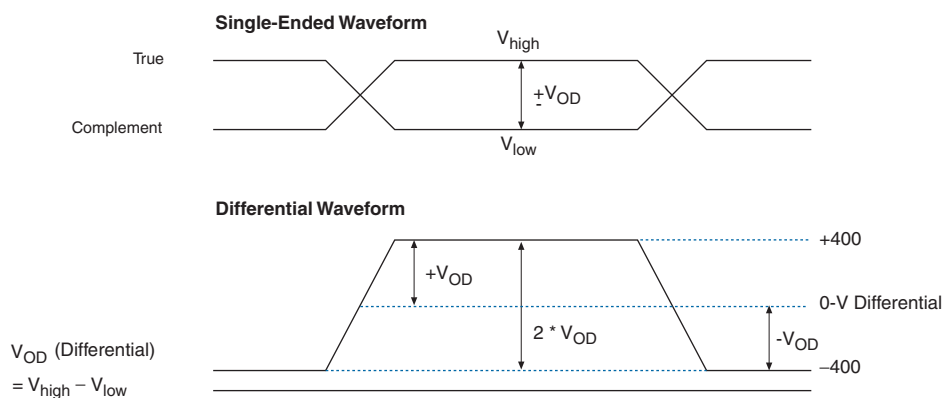
### Programmable Pre-Emphasis

The programmable pre-emphasis module controls the output driver to boost high frequency components and compensate for losses in the transmission medium, as shown in Figure 2–10. Pre-emphasis is set statically using the ALTGXB megafunction.

**Figure 2–10.** Pre-Emphasis Signaling



$$\text{Pre-Emphasis \%} = \left( \frac{V_{MAX}}{V_{MIN}} - 1 \right) \times 100$$
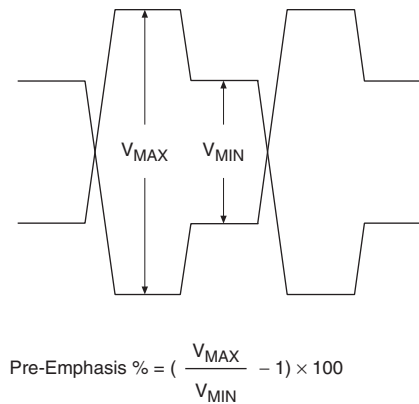
Pre-emphasis percentage is defined as $(V_{MAX}/V_{MIN} - 1) \times 100$, where $V_{MAX}$ is the differential emphasized voltage (peak-to-peak) and $V_{MIN}$ is the differential steady-state voltage (peak-to-peak).

### PCI Express (PIPE) Receiver Detect

The Arria GX transmitter buffer has a built-in receiver detection circuit for use in PCI Express (PIPE) mode. This circuit provides the ability to detect if there is a receiver downstream by sending out a pulse on the channel and monitoring the reflection. This mode requires a tri-stated transmitter buffer (in electrical idle mode).

### PCI Express (PIPE) Electric Idles (or Individual Transmitter Tri-State)

The Arria GX transmitter buffer supports PCI Express (PIPE) electrical idles. This feature is only active in PCI Express (PIPE) mode. The `tx_forceelecidle` port puts the transmitter buffer in electrical idle mode. This port is available in all PCI Express (PIPE) power-down modes and has specific usage in each mode.

## Receiver Path

This section describes the data path through the Arria GX receiver. The sub-blocks are described in order from the receiver buffer to the PLD-receiver parallel interface.

### Receiver Buffer

The Arria GX receiver input buffer supports the 1.2-V and 1.5-V PCML I/O standards at rates up to 3.125 Gbps. The common mode voltage of the receiver input buffer is programmable between 0.85 V and 1.2 V. You must select the 0.85 V common mode voltage for AC- and DC-coupled PCML links and 1.2 V common mode voltage for DC-coupled LVDS links.

The receiver has 100-$\Omega$ on-chip differential termination (R$_D$ OCT) for different protocols, as shown in Figure 2–11. You can disable the receiver's internal termination if external terminations and biasing are provided. The receiver and transmitter differential termination method can be set independently of each other.

**Figure 2–11.** Receiver Input Buffer
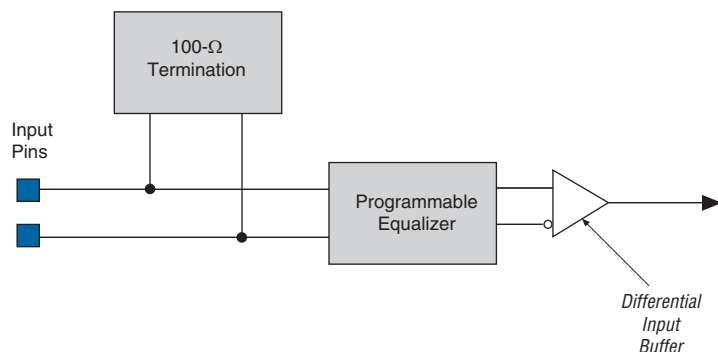


If a design uses external termination, the receiver must be externally terminated and biased to 0.85 V or 1.2 V. Figure 2–12 shows an example of an external termination and biasing circuit.

**Figure 2–12.** External Termination and Biasing Circuit
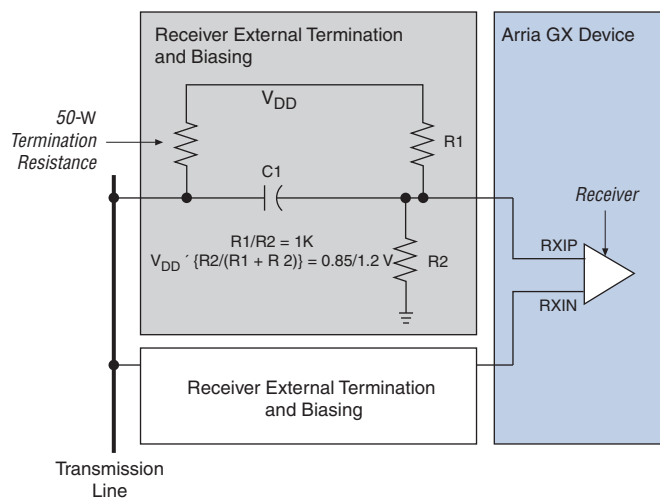


## Programmable Equalizer

The Arria GX receivers provide a programmable receiver equalization feature to compensate for the effects of channel attenuation for high-speed signaling. PCB traces carrying these high-speed signals have low-pass filter characteristics. Impedance mismatch boundaries can also cause signal degradation. Equalization in the receiver diminishes the lossy attenuation effects of the PCB at high frequencies.

The receiver equalization circuit is comprised of a programmable amplifier. Each stage is a peaking equalizer with a different center frequency and programmable gain. This allows varying amounts of gain to be applied, depending on the overall frequency response of the channel loss. Channel loss is defined as the summation of all losses through the PCB traces, vias, connectors, and cables present in the physical link. The Quartus II software allows five equalization settings for Arria GX devices.

### Receiver PLL and Clock Recovery Unit (CRU)

Each transceiver block has four receiver PLLs and CRU units, each of which is dedicated to a receiver channel. The receiver PLL is fed by an input reference clock. The receiver PLL, in conjunction with the CRU, generates two clocks: a high-speed serial recovered clock that clocks the deserializer and a low-speed parallel recovered clock that clocks the receiver's digital logic.

Figure 2–13 shows a block diagram of the receiver PLL and CRU circuits.

**Figure 2–13.** Receiver PLL and Clock Recovery Unit
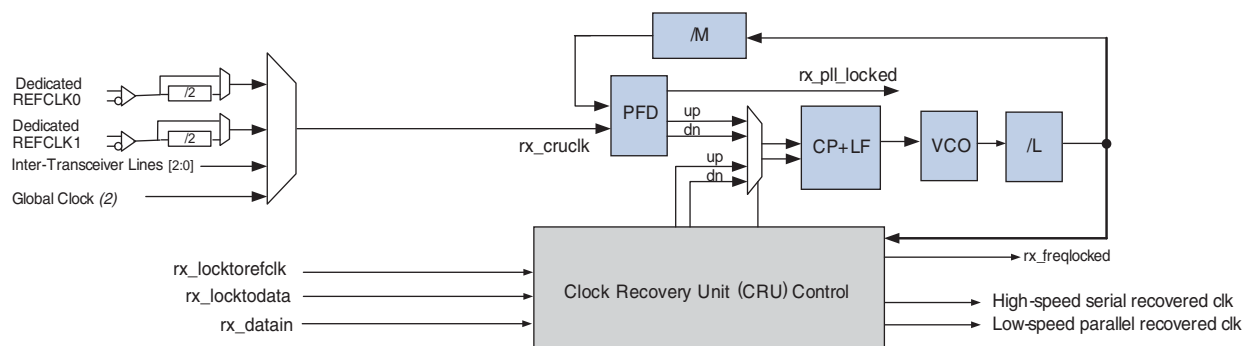


**Notes to Figure 2–13:**

(1) You only need to select the protocol and the available input reference clock frequency in the ALTGXB MegaWizard Plug-In Manager. Based on your selections, the ALTGXB MegaWizard Plug-In Manager automatically selects the necessary /M and /L dividers.

(2) The global clock line must be driven from an input pin only.

The reference clock input to the receiver PLL can be derived from:

- One of the two available dedicated reference clock input pins (REFCLK0 or REFCLK1) of the associated transceiver block

- PLD global clock network (must be driven directly from an input clock pin and cannot be driven by user logic or enhanced PLL)

- Inter-transceiver block lines driven by reference clock input pins of other transceiver blocks

All the parameters listed are programmable in the Quartus II software. The receiver PLL has the following features:

- Operates from 600 Mbps to 3.125 Gbps.

- Uses a reference clock between 50 MHz and 622.08 MHz.

- Programmable bandwidth settings: low, medium, and high.

- Programmable rx_locktorefclk (forces the receiver PLL to lock to reference clock) and rx_locktodata (forces the receiver PLL to lock to data).

■ The voltage-controlled oscillator ($V_{CO}$) operates at half rate.

■ Programmable frequency multiplication W of 1, 4, 5, 8, 10, 16, 20, and 25. Not all settings are supported for any particular frequency.

■ Two lock indication signals are provided. They are found in PFD mode (lock-to-reference clock), and PD (lock-to-data).

The CRU controls whether the receiver PLL locks to the input reference clock (lock-to-reference mode) or the incoming serial data (lock-to data mode). You can set the CRU to switch between lock-to-data and lock-to-reference modes automatically or manually. In automatic lock mode, the phase detector and dedicated parts per million (PPM) detector within each receiver channel control the switch between lock-to-data and lock-to-reference modes based on some pre-set conditions. In manual lock mode, you can control the switch manually using the rx_locktorefclk and rx_locktodata signals.

For more information, refer to the "Clock Recovery Unit" section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter.

Table 2–4 lists the behavior of the CRU block with respect to the rx_locktorefclk and rx_locktodata signals.

**Table 2–4.** CRU Manual Lock Signals

| rx_locktorefclk | rx_locktodata | CRU Mode |
|---|---|---|
| 1 | 0 | Lock-to-reference clock |
| x | 1 | Lock-to-data |
| 0 | 0 | Automatic |

If the rx_locktorefclk and rx_locktodata ports are not used, the default setting is automatic lock mode.

## Deserializer

The deserializer block clocks in serial input data from the receiver buffer using the high-speed serial recovered clock and deserializes into 8- or 10-bit parallel data using the low-speed parallel recovered clock. The serial data is assumed to be received with LSB first, followed by MSB. It feeds the deserialized 8- or 10-bit data to the word aligner, as shown in Figure 2–14.

**Figure 2–14.** Deserializer    *(Note 1)*
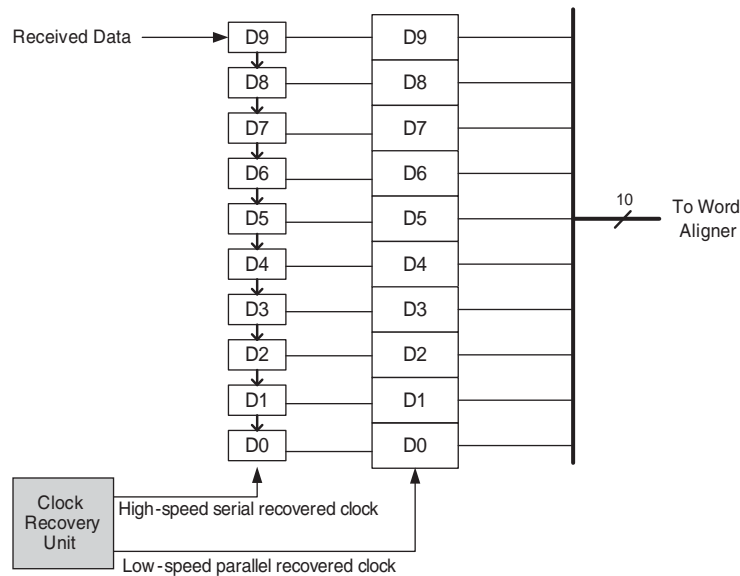


**Note to Figure 2–14:**

(1)    This is a 10-bit deserializer. The deserializer can also convert 8 bits of data.

## Word Aligner

The deserializer block creates 8- or 10-bit parallel data. The deserializer ignores protocol symbol boundaries when converting this data. Therefore, the boundaries of the transferred words are arbitrary. The word aligner aligns the incoming data based on specific byte or word boundaries. The word alignment module is clocked by the local receiver recovered clock during normal operation. All the data and programmed patterns are defined as "big-endian" (most significant word followed by least significant word). Most-significant-bit-first protocols should reverse the bit order of word align patterns programmed.

This module detects word boundaries for 8B/10B-based protocols. This module is also used to align to specific programmable patterns in PRBS7/23 test mode.

## Pattern Detection

The programmable pattern detection logic can be programmed to align word boundaries using a single 7- or 10-bit pattern. The pattern detector can either do an exact match, or match the exact pattern and the complement of a given pattern. Once the programmed pattern is found, the data stream is aligned to have the pattern on the LSB portion of the data output bus.

XAUI, GIGE, PCI Express (PIPE), and Serial RapidIO standards have embedded state machines for symbol boundary synchronization. These standards use K28.5 as their 10-bit programmed comma pattern. Each of these standards uses different algorithms before signaling symbol boundary acquisition to the FPGA.

Pattern detection logic searches from the LSB to the MSB. If multiple patterns are found within the search window, the pattern in the lower portion of the data stream (corresponding to the pattern received earlier) is aligned and the rest of the matching patterns are ignored.

Once a pattern is detected and the data bus is aligned, the word boundary is locked. The two detection status signals (`rx_syncstatus` and `rx_patterndetect`) indicate that an alignment is complete.

Figure 2–15 is a block diagram of the word aligner.

**Figure 2–15.** Word Aligner
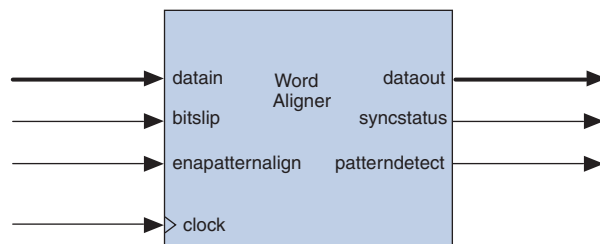


### Control and Status Signals

The `rx_enapatternalign` signal is the FPGA control signal that enables word alignment in non-automatic modes. The `rx_enapatternalign` signal is not used in automatic modes (PCI Express [PIPE], XAUI, GIGE, and Serial RapidIO).

In manual alignment mode, after the `rx_enapatternalign` signal is activated, the `rx_syncstatus` signal goes high for one parallel clock cycle to indicate that the alignment pattern has been detected and the word boundary has been locked. If `rx_enapatternalign` is deactivated, the `rx_syncstatus` signal acts as a re-synchronization signal to signify that the alignment pattern has been detected but not locked on a different word boundary.

When using the synchronization state machine, the `rx_syncstatus` signal indicates the link status. If the `rx_syncstatus` signal is high, link synchronization is achieved. If the `rx_syncstatus` signal is low, link synchronization has not yet been achieved, or there were enough code group errors to lose synchronization.

For more information about manual alignment modes, refer to the *Arria GX Device Handbook*.

The `rx_patterndetect` signal pulses high during a new alignment and whenever the alignment pattern occurs on the current word boundary.

### Programmable Run Length Violation

The word aligner supports a programmable run length violation counter. Whenever the number of the continuous '0' (or '1') exceeds a user programmable value, the `rx_rlv` signal goes high for a minimum pulse width of two recovered clock cycles. The maximum run values supported are 128 UI for 8-bit serialization or 160 UI for 10-bit serialization.

### Running Disparity Check

The running disparity error `rx_disperr` and running disparity value `rx_runningdisp` are sent along with aligned data from the 8B/10B decoder to the FPGA. You can ignore or act on the reported running disparity value and running disparity error signals.

### Bit-Slip Mode

The word aligner can operate in either pattern detection mode or in bit-slip mode.

The bit-slip mode provides the option to manually shift the word boundary through the FPGA. This feature is useful for:

■ Longer synchronization patterns than the pattern detector can accommodate

■ Scrambled data stream

■ Input stream consisting of over-sampled data

The word aligner outputs a word boundary as it is received from the analog receiver after reset. You can examine the word and search its boundary in the FPGA. To do so, assert the rx_bitslip signal. The rx_bitslip signal should be toggled and held constant for at least two FPGA clock cycles.

For every rising edge of the rx_bitslip signal, the current word boundary is slipped by one bit. Every time a bit is slipped, the bit received earliest is lost. If bit slipping shifts a complete round of bus width, the word boundary is back to the original boundary.

The rx_syncstatus signal is not available in bit-slipping mode.

### Channel Aligner

The channel aligner is available only in XAUI mode and aligns the signals of all four channels within a transceiver. The channel aligner follows the IEEE 802.3ae, clause 48 specification for channel bonding.

The channel aligner is a 16-word FIFO buffer with a state machine controlling the channel bonding process. The state machine looks for an /A/ (/K28.3/) in each channel and aligns all the /A/ code groups in the transceiver. When four columns of /A/ (denoted by //A//) are detected, the rx_channelaligned signal goes high, signifying that all the channels in the transceiver have been aligned. The reception of four consecutive misaligned /A/ code groups restarts the channel alignment sequence and sends the rx_channelaligned signal low.

Figure 2–16 shows misaligned channels before the channel aligner and the aligned channels after the channel aligner.

**Figure 2–16.** Before and After the Channel Aligner
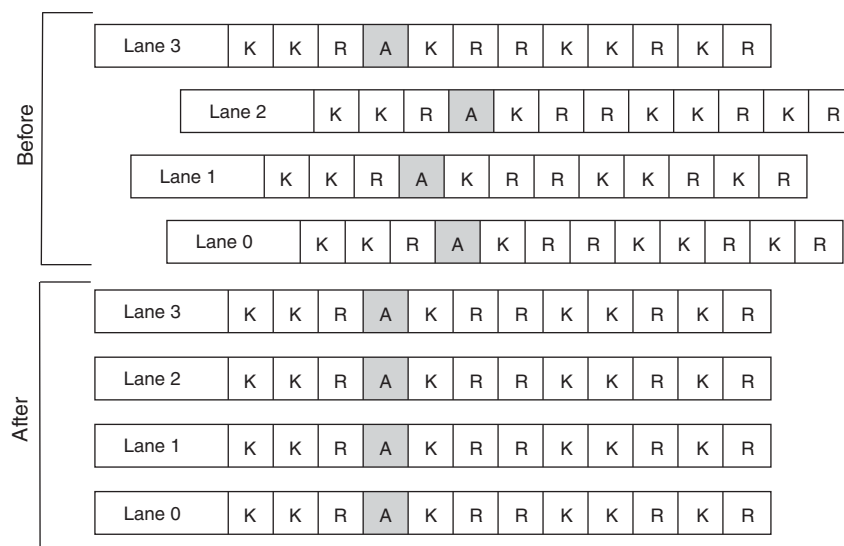


## Rate Matcher

In asynchronous systems, the upstream transmitter and local receiver can be clocked with independent reference clock sources. Frequency differences in the order of a few hundred PPM can potentially corrupt the data at the receiver.

The rate matcher compensates for small clock frequency differences between the upstream transmitter and the local receiver clocks by inserting or removing skip characters from the inter packet gap (IPG) or idle streams. It inserts a skip character if the local receiver is running a faster clock than the upstream transmitter. It deletes a skip character if the local receiver is running a slower clock than the upstream transmitter. The Quartus II software automatically configures the appropriate skip character as specified in the IEEE 802.3 for GIGE mode and PCI-Express Base Specification for PCI Express (PIPE) mode. The rate matcher is bypassed in Serial RapidIO and must be implemented in the PLD logic array or external circuits depending on your system design.

Table 2–5 lists the maximum frequency difference that the rate matcher can tolerate in XAUI, PCI Express (PIPE), GIGE, and Basic functional modes.

**Table 2–5.** Rate Matcher PPM Tolerance

| Function Mode | PPM |
|---|---|
| XAUI | ± 100 |
| PCI Express (PIPE) | ± 300 |
| GIGE | ± 100 |
| Basic | ± 300 |

### XAUI Mode

In XAUI mode, the rate matcher adheres to clause 48 of the IEEE 802.3ae specification for clock rate compensation. The rate matcher performs clock compensation on columns of /R/ (/K28.0/), denoted by //R//. An //R// is added or deleted automatically based on the number of words in the FIFO buffer.

### PCI Express (PIPE) Mode Rate Matcher

In PCI Express (PIPE) mode, the rate matcher can compensate up to ± 300 PPM (600 PPM total) frequency difference between the upstream transmitter and the receiver. The rate matcher logic looks for skip ordered sets (SOS), which contains a /K28.5/ comma followed by three /K28.0/ skip characters. The rate matcher logic deletes or inserts /K28.0/ skip characters as necessary from/to the rate matcher FIFO.

The rate matcher in PCI Express (PIPE) mode has a FIFO buffer overflow and underflow protection. In the event of a FIFO buffer overflow, the rate matcher deletes any data after detecting the overflow condition to prevent FIFO pointer corruption until the rate matcher is not full. In an underflow condition, the rate matcher inserts 9'h1FE (/K30.7/) until the FIFO buffer is not empty. These measures ensure that the FIFO buffer can gracefully exit the overflow and underflow condition without requiring a FIFO reset. The rate matcher FIFO overflow and underflow condition is indicated on the `pipestatus` port.

You can bypass the rate matcher in PCI Express (PIPE) mode if you have a synchronous system where the upstream transmitter and local receiver derive their reference clocks from the same source.

### GIGE Mode Rate Matcher

In GIGE mode, the rate matcher can compensate up to ± 100 PPM (200 PPM total) frequency difference between the upstream transmitter and the receiver. The rate matcher logic inserts or deletes /I2/ idle ordered sets to/from the rate matcher FIFO during the inter-frame or inter-packet gap (IFG or IPG). /I2/ is selected as the rate matching ordered set because it maintains the running disparity, unlike /I1/ that alters the running disparity. Because the /I2/ ordered-set contains two 10-bit code groups (/K28.5/, /D16.2/), 20 bits are inserted or deleted at a time for rate matching.

☞ The rate matcher logic has the capability to insert or delete /C1/ or /C2/ configuration ordered sets when 'GIGE Enhanced' mode is chosen as the sub-protocol in the MegaWizard Plug-In Manager.

If the frequency PPM difference between the upstream transmitter and the local receiver is high, or if the packet size is too large, the rate matcher FIFO buffer can face an overflow or underflow situation.

### Basic Mode

In basic mode, you can program the skip and control pattern for rate matching. There is no restriction on the deletion of a skip character in a cluster. The rate matcher deletes the skip characters as long as they are available. For insertion, the rate matcher inserts skip characters such that the number of skip characters at the output of rate matcher does not exceed five.

## 8B/10B Decoder

The 8B/10B decoder is used in all supported functional modes. The 8B/10B decoder takes in 10-bit data from the rate matcher and decodes it into 8-bit data + 1-bit control identifier, thereby restoring the original transmitted data at the receiver. The 8B/10B decoder indicates whether the received 10-bit character is a data or control code through the `rx_ctrldetect` port. If the received 10-bit code group is a control character (`Kx.y`), the `rx_ctrldetect` signal is driven high and if it is a data character (`Dx.y`), the `rx_ctrldetect` signal is driven low.

Figure 2–17 shows a 10-bit code group decoded to an 8-bit data and a 1-bit control indicator.

**Figure 2–17.** 10-Bit to 8-Bit Conversion
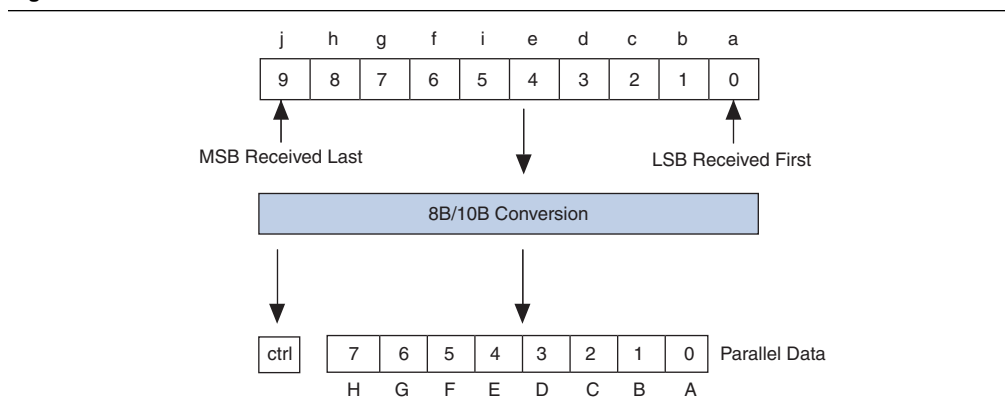


If the received 10-bit code is not a part of valid Dx.y or Kx.y code groups, the 8B/10B decoder block asserts an error flag on the `rx_errdetect` port. If the received 10-bit code is detected with incorrect running disparity, the 8B/10B decoder block asserts an error flag on the `rx_disperr` and `rx_errdetect` ports. The error flag signals (`rx_errdetect` and `rx_disperr`) have the same data path delay from the 8B/10B decoder to the PLD-transceiver interface as the bad code group.

## Receiver State Machine

The receiver state machine operates in Basic, GIGE, PCI Express (PIPE), and XAUI modes. In GIGE mode, the receiver state machine replaces invalid code groups with K30.7. In XAUI mode, the receiver state machine translates the XAUI PCS code group to the XAUI XGMII code group.

### Byte Deserializer

Byte deserializer takes in one-byte wide data from the 8B/10B decoder and deserializes it into a two-byte wide data at half the speed. This allows clocking the PLD-receiver interface at half the speed as compared to the receiver PCS logic. The byte deserializer is bypassed in GIGE mode.

The byte ordering at the receiver output might be different than what was transmitted. This is a non-deterministic swap, because it depends on PLL lock times and link delay. If required, you must implement byte ordering logic in the PLD to correct this situation.

For more information about byte serializer, refer to the *Arria GX Transceiver Architecture* chapter.

### Receiver Phase Compensation FIFO Buffer

A receiver phase compensation FIFO buffer is located at each receiver channel's logic array interface. It compensates for the phase difference between the receiver PCS clock and the local PLD receiver clock. The receiver phase compensation FIFO is used in all supported functional modes. The receiver phase compensation FIFO buffer is eight words deep in PCI Express (PIPE) mode and four words deep in all other modes.

For more information about architecture and clocking, refer to the *Arria GX Transceiver Architecture* chapter.

## Loopback Modes

Arria GX transceivers support the following loopback configurations for diagnostic purposes:

■ Serial loopback

■ Reverse serial loopback

■ Reverse serial loopback (pre-CDR)

■ PCI Express (PIPE) reverse parallel loopback (available only in [PIPE] mode)

### Serial Loopback

Figure 2–18 shows the transceiver data path in serial loopback.

**Figure 2–18.** Transceiver Data Path in Serial Loopback
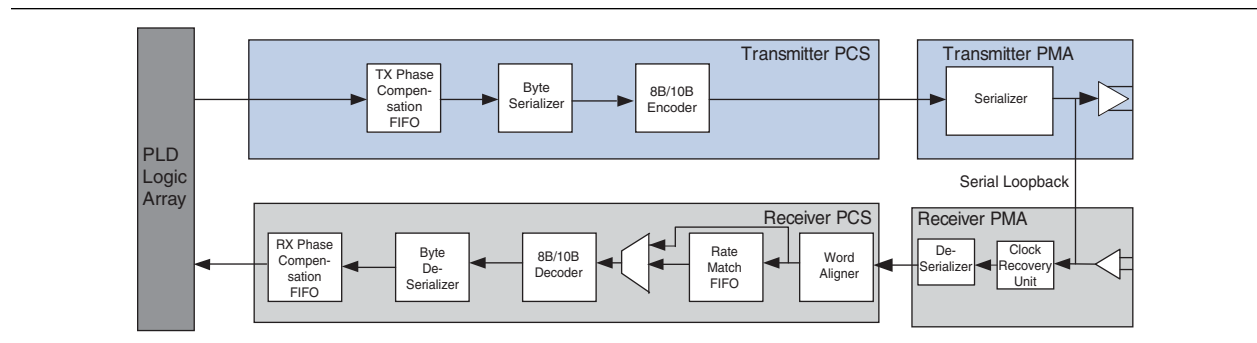
In GIGE and Serial RapidIO modes, you can dynamically put each transceiver channel individually in serial loopback by controlling the rx_seriallpbken port. A high on the rx_seriallpbken port puts the transceiver into serial loopback and a low takes the transceiver out of serial loopback.

As seen in Figure 2–18, the serial data output from the transmitter serializer is looped back to the receiver CRU in serial loopback. The transmitter data path from the PLD interface to the serializer in serial loopback is the same as in non-loopback mode. The receiver data path from the clock recovery unit to the PLD interface in serial loopback is the same as in non-loopback mode. Because the entire transceiver data path is available in serial loopback, this option is often used to diagnose the data path as a probable cause of link errors.

☞ When serial loopback is enabled, the transmitter output buffer is still active and drives the serial data out on the tx_dataout port.

### Reverse Serial Loopback

Reverse serial loopback mode uses the analog portion of the transceiver. An external source (pattern generator or transceiver) generates the source data. The high-speed serial source data arrives at the high-speed differential receiver input buffer, passes through the CRU unit and the retimed serial data is looped back, and is transmitted though the high-speed differential transmitter output buffer.

Figure 2–19 shows the data path in reverse serial loopback mode.

**Figure 2–19.** Arria GX Block in Reverse Serial Loopback Mode
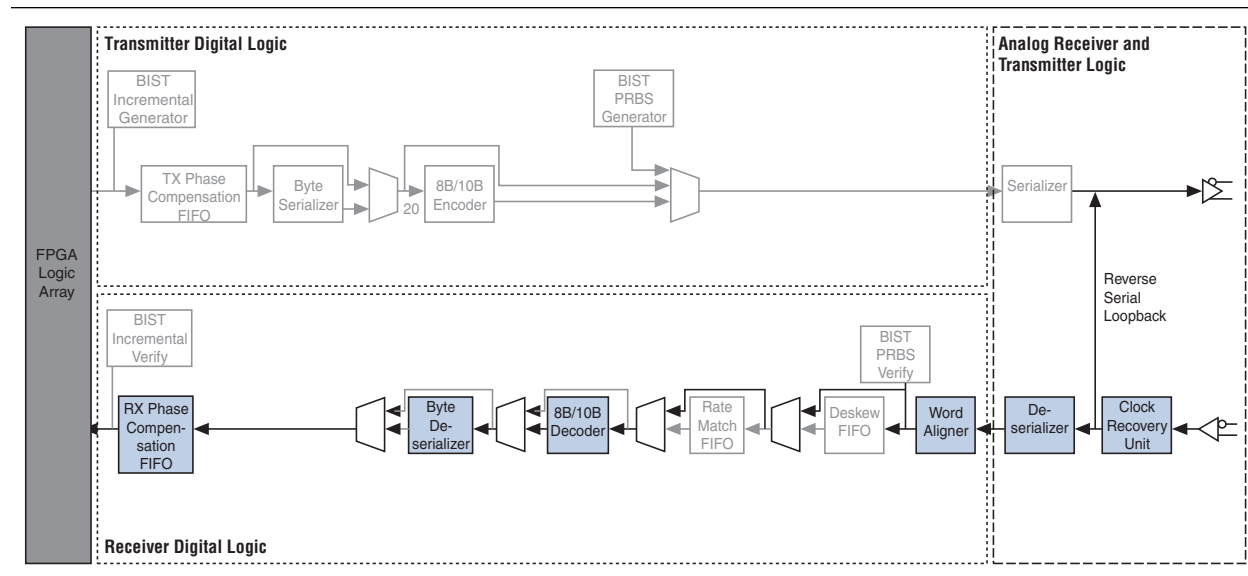
## Reverse Serial Pre-CDR Loopback

Reverse serial pre-CDR loopback mode uses the analog portion of the transceiver. An external source (pattern generator or transceiver) generates the source data. The high-speed serial source data arrives at the high-speed differential receiver input buffer, loops back before the CRU unit, and is transmitted though the high-speed differential transmitter output buffer. It is for test or verification use only to verify the signal being received after the gain and equalization improvements of the input buffer. The signal at the output is not exactly what is received because the signal goes through the output buffer and the $V_{OD}$ is changed to the $V_{OD}$ setting level. Pre-emphasis settings have no effect.

Figure 2–20 shows the Arria GX block in reverse serial pre-CDR loopback mode.

**Figure 2–20.** Arria GX Block in Reverse Serial Pre-CDR Loopback Mode
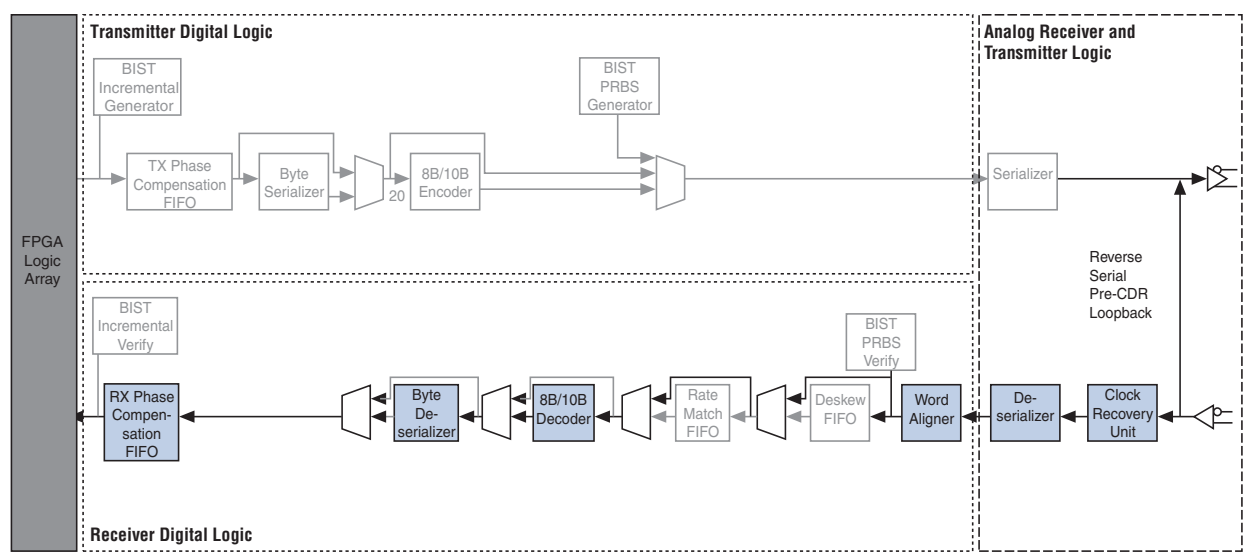


### PCI Express (PIPE) Reverse Parallel Loopback

Figure 2–21 shows the data path for PCI Express (PIPE) reverse parallel loopback. The reverse parallel loopback configuration is compliant with the PCI Express (PIPE) specification and is available only on PCI Express (PIPE) mode.

**Figure 2–21.** PCI Express (PIPE) Reverse Parallel Loopback
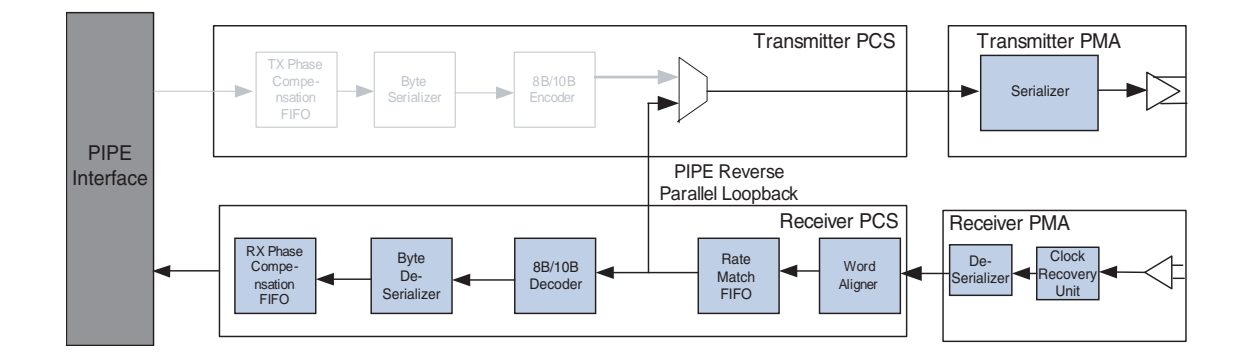
You can dynamically put the PCI Express (PIPE) mode transceiver in reverse parallel loopback by controlling the `tx_detectrxloopback` port instantiated in the MegaWizard Plug-In Manager. A high on the `tx_detectrxloopback` port in P0 power state puts the transceiver in reverse parallel loopback. A high on the `tx_detectrxloopback` port in any other power state does not put the transceiver in reverse parallel loopback.

As seen in Figure 2–21, the serial data received on the `rx_datain` port in reverse parallel loopback goes through the CRU, deserializer, word aligner, and the rate matcher blocks. The parallel data at the output of the receiver rate matcher block is looped back to the input of the transmitter serializer block. The serializer converts the parallel data to serial data and feeds it to the transmitter output buffer that drives the data out on the `tx_dataout` port. The data at the output of the rate matcher also goes through the 8B/10B decoder, byte deserializer, and receiver phase compensation FIFO before being fed to the PLD on the `rx_dataout` port.

## Reset and Powerdown

Arria GX transceivers offer a power saving advantage with their ability to shut off functions that are not needed.

The following three reset signals are available per transceiver channel and can be used to individually reset the digital and analog portions within each channel:

■ `tx_digitalreset`

■ `rx_analogreset`

■ `rx_digitalreset`

The following two powerdown signals are available per transceiver block and can be used to shut down an entire transceiver block that is not being used:

■ `gxb_powerdown`

■ `gxb_enable`

Table 2–6 lists the reset signals available in Arria GX devices and the transceiver circuitry affected by each signal.

**Table 2–6.** Reset Signal Map to Arria GX Blocks

| Reset Signal | Transmitter Phase Compensation FIFO Module/ Byte Serializer | Transmitter 8B/10B Encoder | Transmitter Serializer | Transmitter Analog Circuits | Transmitter PLL | Transmitter XAUI State Machine | BIST Generators | Receiver Deserializer | Receiver Word Aligner | Receiver Deskew FIFO Module | Receiver Rate Matcher | Receiver 8B/10B Decoder | Receiver Phase Comp FIFO Module/ Byte Deserializer | Receiver PLL / CRU | Receiver XAUI State Machine | BIST Verifiers | Receiver Analog Circuits |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| rx_digitalreset | — | — | — | — | — | — | — | — | ✓ | — | ✓ | ✓ | ✓ | — | ✓ | ✓ | — |
| rx_analogreset | — | — | — | — | — | — | — | ✓ | — | — | — | — | — | ✓ | — | — | ✓ |
| tx_digitalreset | ✓ | ✓ | — | — | — | ✓ | ✓ | — | — | — | — | — | — | — | — | — | — |
| gxb_powerdown | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | — | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| gxb_enable | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | — | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

## Calibration Block

Arria GX devices use the calibration block to calibrate OCT for the PLLs, and their associated output buffers, and the terminating resistors on the transceivers. The calibration block counters the effects of process, voltage, and temperature (PVT). The calibration block references a derived voltage across an external reference resistor to calibrate the OCT resistors on Arria GX devices. You can power down the calibration block. However, powering down the calibration block during operations can yield transmit and receive data errors.

## Transceiver Clocking

This section describes the clock distribution in an Arria GX transceiver channel and the PLD clock resource utilization by the transceiver blocks.

### Transceiver Channel Clock Distribution

Each transceiver block has one transmitter PLL and four receiver PLLs.

The transmitter PLL multiplies the input reference clock to generate a high-speed serial clock at a frequency that is half the data rate of the configured functional mode. This high-speed serial clock (or its divide-by-two version if the functional mode uses byte serializer) is fed to the CMU clock divider block. Depending on the configured functional mode, the CMU clock divider block divides the high-speed serial clock to generate the low-speed parallel clock that clocks the transceiver PCS logic in the associated channel. The low-speed parallel clock is also forwarded to the PLD logic array on the `tx_clkout` or `coreclkout` ports.

The receiver PLL in each channel is also fed by an input reference clock. The receiver PLL along with the clock recovery unit generates a high-speed serial recovered clock and a low-speed parallel recovered clock. The low-speed parallel recovered clock feeds the receiver PCS logic until the rate matcher. The CMU low-speed parallel clock clocks the rest of the logic from the rate matcher until the receiver phase compensation FIFO. In modes that do not use a rate matcher, the receiver PCS logic is clocked by the recovered clock until the receiver phase compensation FIFO.

The input reference clock to the transmitter and receiver PLLs can be derived from:

■ One of two available dedicated reference clock input pins (`REFCLK0` or `REFCLK1`) of the associated transceiver block

■ PLD clock network (must be driven directly from an input clock pin and cannot be driven by user logic or enhanced PLL)

■ Inter-transceiver block lines driven by reference clock input pins of other transceiver blocks

Figure 2–22 shows the input reference clock sources for the transmitter and receiver PLL.

**Figure 2–22.** Input Reference Clock Sources
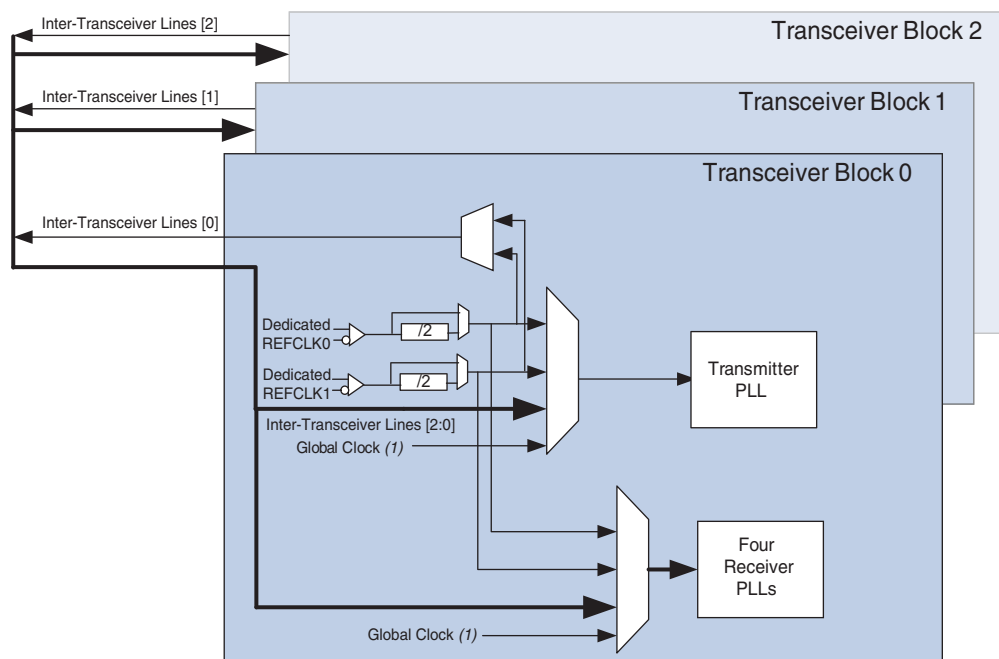
For more information about transceiver clocking in all supported functional modes, refer to the *Arria GX Transceiver Architecture* chapter.

## PLD Clock Utilization by Transceiver Blocks

Arria GX devices have up to 16 global clock (GCLK) lines and 16 regional clock (RCLK) lines that are used to route the transceiver clocks. The following transceiver clocks use the available global and regional clock resources:

- `pll_inclk` (if driven from an FPGA input pin)
- `rx_cruclk` (if driven from an FPGA input pin)
- `tx_clkout`/`coreclkout` (CMU low-speed parallel clock forwarded to the PLD)
- Recovered clock from each channel (`rx_clkout`) in non-rate matcher mode
- Calibration clock (`cal_blk_clk`)
- Fixed clock (`fixedclk` used for receiver detect circuitry in PCI Express [PIPE] mode only)

Figure 2–23 and Figure 2–24 show the available GCLK and RCLK resources in Arria GX devices.

**Figure 2–23.** Global Clock Resources in Arria GX Devices
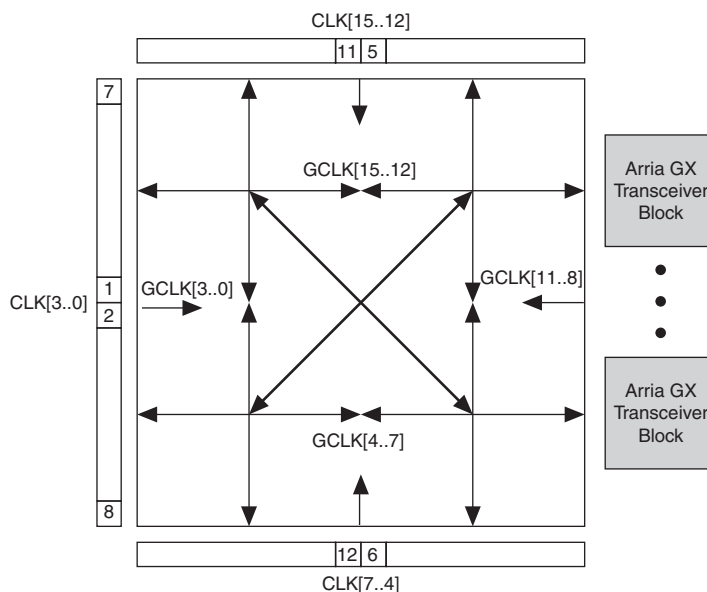
**Figure 2–24.** Regional Clock Resources in Arria GX Devices
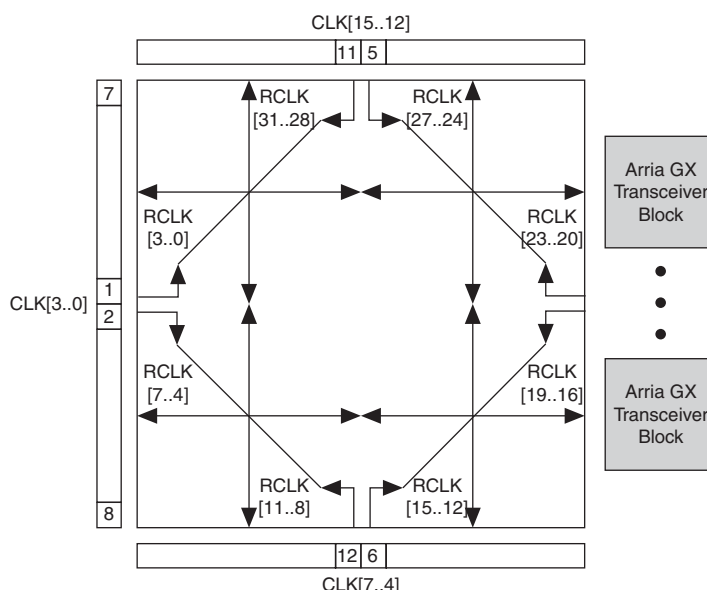


For the RCLK or GCLK network to route into the transceiver, a local route input output (LRIO) channel is required. Each LRIO clock region has up to eight clock paths and each transceiver block has a maximum of eight clock paths for connecting with LRIO clocks. These resources are limited and determine the number of clocks that can be used between the PLD and transceiver blocks. Table 2–7 and Table 2–8 list the number of LRIO resources available for Arria GX devices with different numbers of transceiver blocks.

**Table 2–7.** Available Clocking Connections for Transceivers in EP1AGX35D, EP1AGX50D, and EP1AGX60D

| Source | Clock Resource | | Transceiver | |
| --- | --- | --- | --- | --- |
| | Global Clock | Regional Clock | Bank13 8 Clock I/O | Bank14 8 Clock I/O |
| Region0 8 LRIO clock | ✔ | RCLK 20-27 | ✔ | — |
| Region1 8 LRIO clock | ✔ | RCLK 12-19 | — | ✔ |

**Table 2–8.** Available Clocking Connections for Transceivers in EP1AGX60E and EP1AGX90E

| Source | Clock Resource | | Transceiver | | |
| --- | --- | --- | --- | --- | --- |
| | Global Clock | Regional Clock | Bank13 8 Clock I/O | Bank14 8 Clock I/O | Bank15 8 Clock I/O |
| Region0 8 LRIO clock | ✔ | RCLK 20-27 | ✔ | — | — |
| Region1 8 LRIO clock | ✔ | RCLK 20-27 | ✔ | ✔ | — |
| Region2 8 LRIO clock | ✔ | RCLK 12-19 | — | ✔ | ✔ |
| Region3 8 LRIO clock | ✔ | RCLK 12-19 | — | — | ✔ |

# Logic Array Blocks

Each logic array block (LAB) consists of eight adaptive logic modules (ALMs), carry chains, shared arithmetic chains, LAB control signals, local interconnects, and register chain connection lines. The local interconnect transfers signals between ALMs in the same LAB. Register chain connections transfer the output of an ALM register to the adjacent ALM register in a LAB. The Quartus II Compiler places associated logic in a LAB or adjacent LABs, allowing the use of local, shared arithmetic chain, and register chain connections for performance and area efficiency. Table 2–9 lists Arria GX device resources. Figure 2–25 shows the Arria GX LAB structure.

**Table 2–9.** Arria GX Device Resources

| Device | M512 RAM Columns/Blocks | M4K RAM Columns/Blocks | M-RAM Blocks | DSP Block Columns/Blocks |
|--------|-------------------------|------------------------|--------------|--------------------------|
| EP1AGX20 | 166 | 118 | 1 | 10 |
| EP1AGX35 | 197 | 140 | 1 | 14 |
| EP1AGX50 | 313 | 242 | 2 | 26 |
| EP1AGX60 | 326 | 252 | 2 | 32 |
| EP1AGX90 | 478 | 400 | 4 | 44 |

**Figure 2–25.** Arria GX LAB Structure
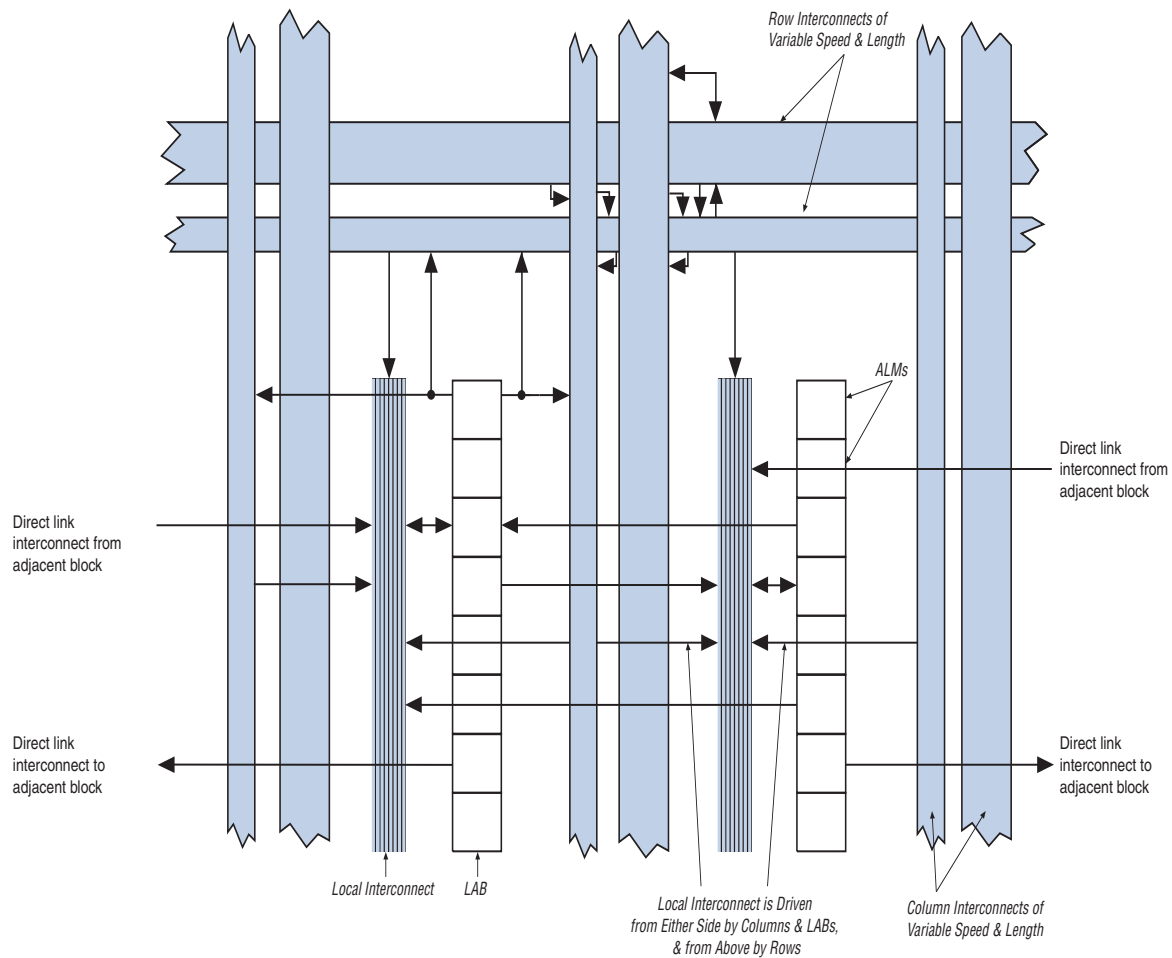


## LAB Interconnects

The LAB local interconnect can drive all eight ALMs in the same LAB. It is driven by column and row interconnects and ALM outputs in the same LAB. Neighboring LABs, M512 RAM blocks, M4K RAM blocks, M-RAM blocks, or digital signal processing (DSP) blocks from the left and right can also drive the local interconnect of a LAB through the direct link connection. The direct link connection feature minimizes the use of row and column interconnects, providing higher performance and flexibility. Each ALM can drive 24 ALMs through fast local and direct link interconnects.

Figure 2–26 shows the direct link connection.

**Figure 2–26.** Direct Link Connection
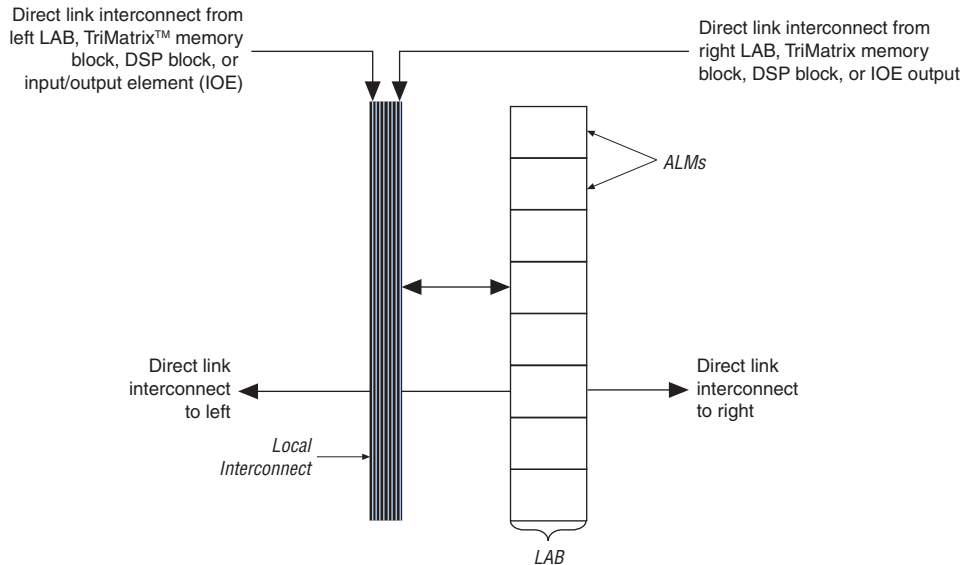


## LAB Control Signals

Each LAB contains dedicated logic for driving control signals to its ALMs. The control signals include three clocks, three clock enables, two asynchronous clears, synchronous clear, asynchronous preset or load, and synchronous load control signals, providing a maximum of 11 control signals at a time. Although synchronous load and clear signals are generally used when implementing counters, they can also be used with other functions.

Each LAB can use three clocks and three clock enable signals. However, there can only be up to two unique clocks per LAB, as shown in the LAB control signal generation circuit in Figure 2–27. Each LAB's clock and clock enable signals are linked. For example, any ALM in a particular LAB using the labclk1 signal also uses labclkena1. If the LAB uses both the rising and falling edges of a clock, it also uses two LAB-wide clock signals. De-asserting the clock enable signal turns off the corresponding LAB-wide clock. Each LAB can use two asynchronous clear signals and an asynchronous load/preset signal. The asynchronous load acts as a preset when the asynchronous load data input is tied high. When the asynchronous load/preset signal is used, the labclkena0 signal is no longer available.

The LAB row clocks [5..0] and LAB local interconnect generate the LAB-wide control signals. The MultiTrack interconnects have inherently low skew. This low skew allows the MultiTrack interconnects to distribute clock and control signals in addition to data.

Figure 2–27 shows the LAB control signal generation circuit.

**Figure 2–27.** LAB-Wide Control Signals
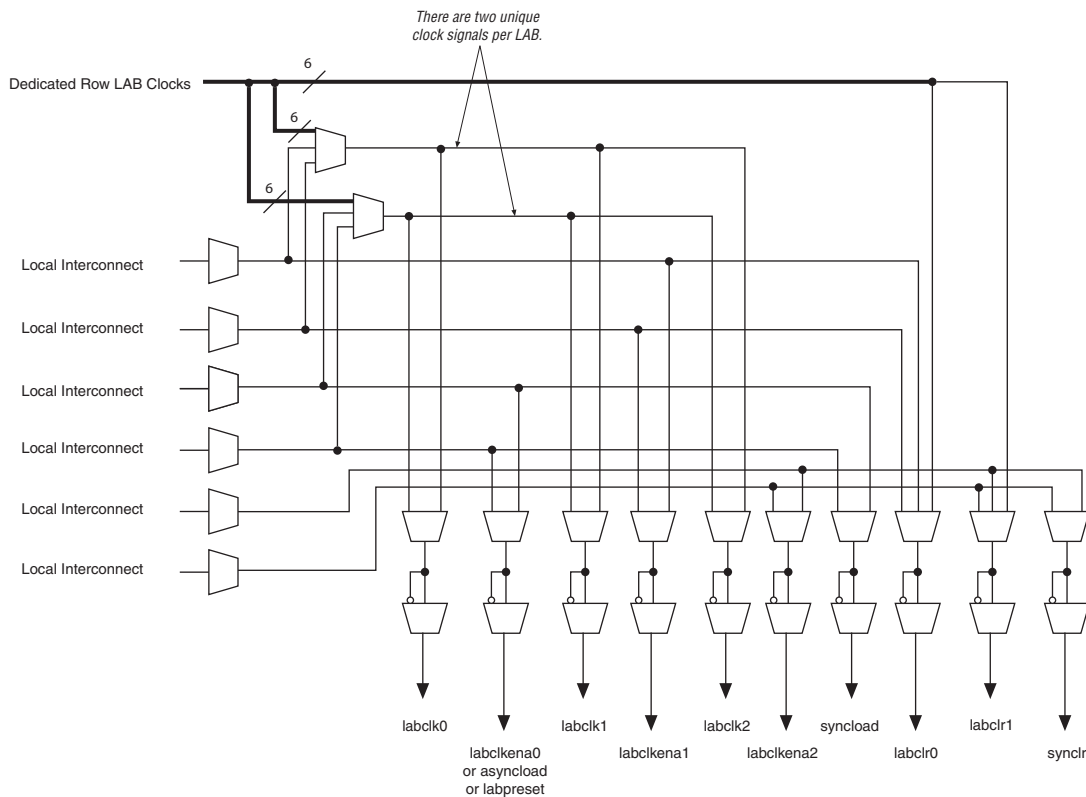


# Adaptive Logic Modules

The basic building block of logic in the Arria GX architecture is the ALM. The ALM provides advanced features with efficient logic utilization. Each ALM contains a variety of look-up table (LUT)-based resources that can be divided between two adaptive LUTs (ALUTs). With up to eight inputs to the two ALUTs, one ALM can implement various combinations of two functions. This adaptability allows the ALM to be completely backward-compatible with four-input LUT architectures. One ALM can also implement any function of up to six inputs and certain seven-input functions.

In addition to the adaptive LUT-based resources, each ALM contains two programmable registers, two dedicated full adders, a carry chain, a shared arithmetic chain, and a register chain. Through these dedicated resources, the ALM can efficiently implement various arithmetic functions and shift registers. Each ALM drives all types of interconnects: local, row, column, carry chain, shared arithmetic chain, register chain, and direct link interconnects. Figure 2–28 shows a high-level block diagram of the Arria GX ALM while Figure 2–29 shows a detailed view of all the connections in the ALM.

**Figure 2–28.** High-Level Block Diagram of the Arria GX ALM

**Figure 2–29.** Arria GX ALM Details
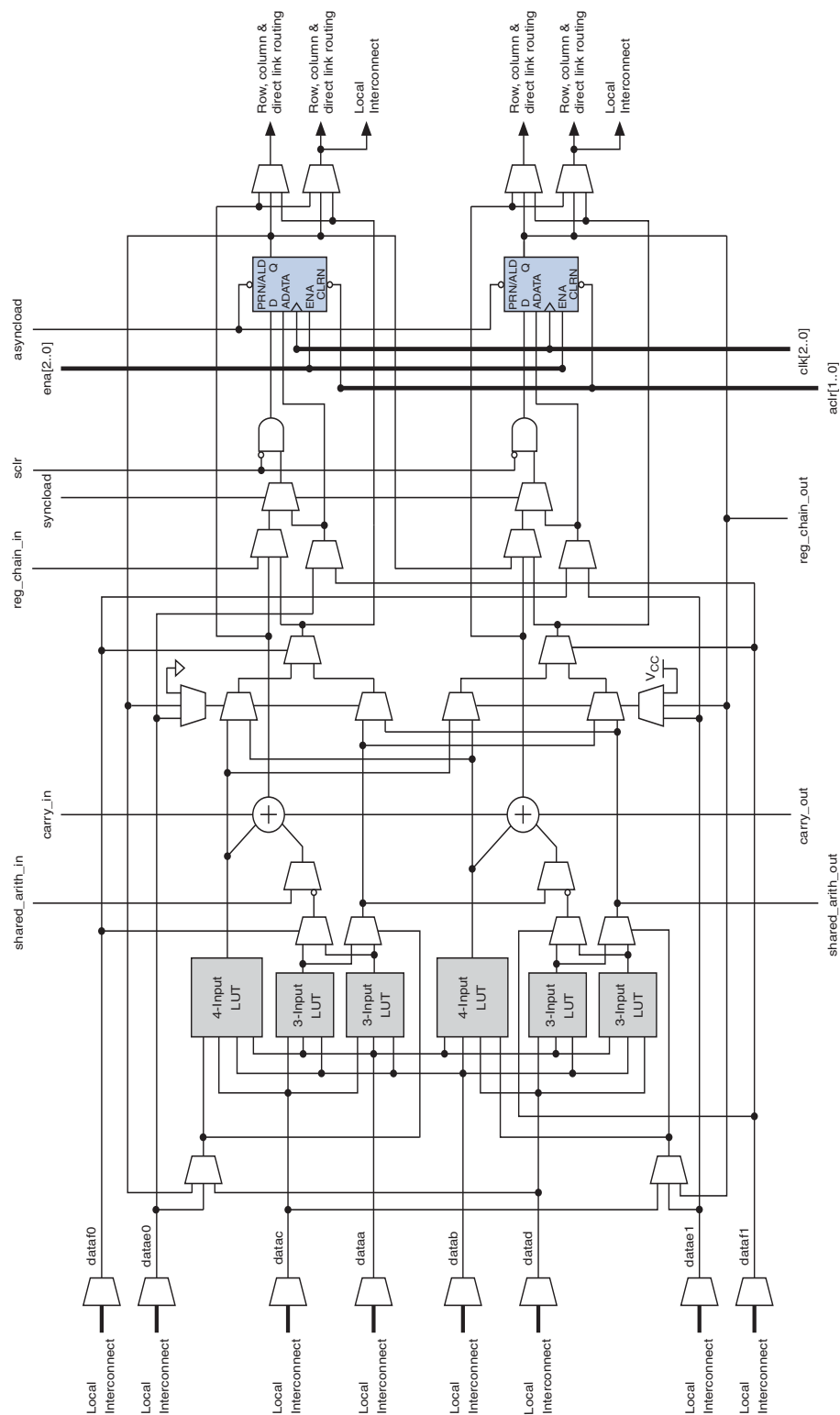
One ALM contains two programmable registers. Each register has data, clock, clock enable, synchronous and asynchronous clear, asynchronous load data, and synchronous and asynchronous load/preset inputs.

Global signals, general-purpose I/O pins, or any internal logic can drive the register's clock and clear control signals. Either general-purpose I/O pins or internal logic can drive the clock enable, preset, asynchronous load, and asynchronous load data. The asynchronous load data input comes from the datae or dataf input of the ALM, which are the same inputs that can be used for register packing. For combinational functions, the register is bypassed and the output of the LUT drives directly to the outputs of the ALM.

Each ALM has two sets of outputs that drive the local, row, and column routing resources. The LUT, adder, or register output can drive these output drivers independently (refer to Figure 2–29). For each set of output drivers, two ALM outputs can drive column, row, or direct link routing connections. One of these ALM outputs can also drive local interconnect resources. This allows the LUT or adder to drive one output while the register drives another output. This feature, called register packing, improves device utilization because the device can use the register and combinational logic for unrelated functions. Another special packing mode allows the register output to feed back into the LUT of the same ALM so that the register is packed with its own fan-out LUT. This feature provides another mechanism for improved fitting. The ALM can also drive out registered and unregistered versions of the LUT or adder output.

## ALM Operating Modes

The Arria GX ALM can operate in one of the following modes:

- Normal mode

- Extended LUT mode

- Arithmetic mode

- Shared arithmetic mode

Each mode uses ALM resources differently. Each mode has 11 available inputs to the ALM (refer to Figure 2–28)—the eight data inputs from the LAB local interconnect; carry-in from the previous ALM or LAB; the shared arithmetic chain connection from the previous ALM or LAB; and the register chain connection—are directed to different destinations to implement the desired logic function. LAB-wide signals provide clock, asynchronous clear, asynchronous preset/load, synchronous clear, synchronous load, and clock enable control for the register. These LAB-wide signals are available in all ALM modes. For more information about LAB-wide control signals, refer to "LAB Control Signals" on page 2–30.

The Quartus II software and supported third-party synthesis tools, in conjunction with parameterized functions such as library of parameterized modules (LPM) functions, automatically choose the appropriate mode for common functions such as counters, adders, subtractors, and arithmetic functions. If required, you can also create special-purpose functions that specify which ALM operating mode to use for optimal performance.

## Normal Mode

Normal mode is suitable for general logic applications and combinational functions. In this mode, up to eight data inputs from the LAB local interconnect are inputs to the combinational logic. Normal mode allows two functions to be implemented in one Arria GX ALM, or an ALM to implement a single function of up to six inputs. The ALM can support certain combinations of completely independent functions and various combinations of functions which have common inputs. Figure 2–30 shows the supported LUT combinations in normal mode.

**Figure 2–30.** ALM in Normal Mode    *(Note 1)*
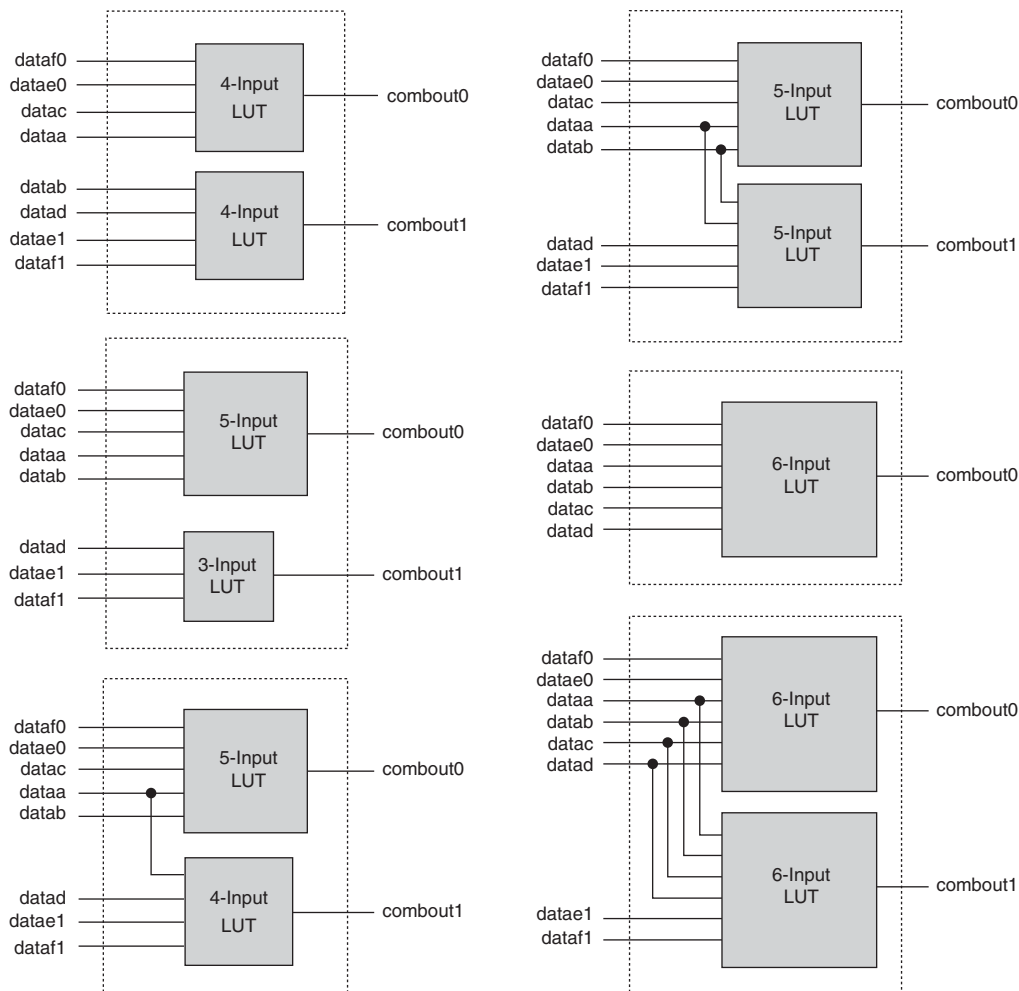


**Note to Figure 2–30:**

(1) Combinations of functions with less inputs than those shown are also supported. For example, combinations of functions with the following number of inputs are supported: 4 and 3, 3 and 3, 3 and 2, 5 and 2, and so on.

Normal mode provides complete backward compatibility with four-input LUT architectures. Two independent functions of four inputs or less can be implemented in one Arria GX ALM. In addition, a five-input function and an independent three-input function can be implemented without sharing inputs.

To pack two five-input functions into one ALM, the functions must have at least two common inputs. The common inputs are dataa and datab. The combination of a four-input function with a five-input function requires one common input (either dataa or datab).

To implement two six-input functions in one ALM, four inputs must be shared and the combinational function must be the same. For example, a 4 × 2 crossbar switch (two 4-to-1 multiplexers with common inputs and unique select lines) can be implemented in one ALM, as shown in Figure 2–31. The shared inputs are dataa, datab, datac, and datad, while the unique select lines are datae0 and dataf0 for function0, and datae1 and dataf1 for function1. This crossbar switch consumes four LUTs in a four-input LUT-based architecture.

**Figure 2–31.** 4 × 2 Crossbar Switch Example
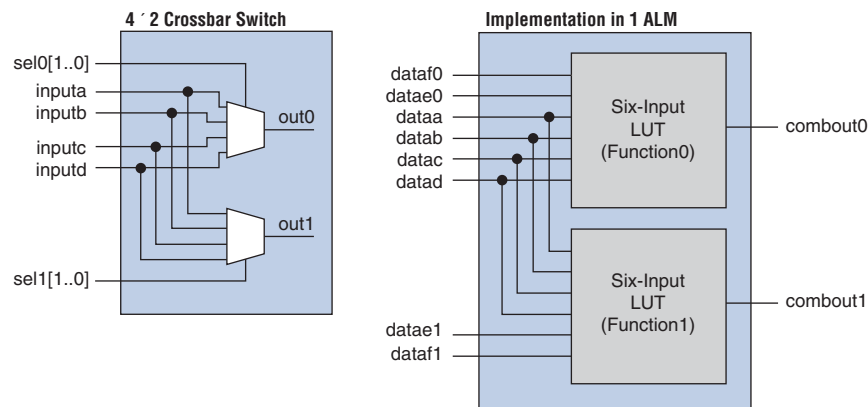


In a sparsely used device, functions that can be placed into one ALM can be implemented in separate ALMs. The Quartus II Compiler spreads a design out to achieve the best possible performance. As a device begins to fill up, the Quartus II software automatically uses the full potential of the Arria GX ALM. The Quartus II Compiler automatically searches for functions of common inputs or completely independent functions to be placed into one ALM and to make efficient use of the device resources. In addition, you can manually control resource usage by setting location assignments. Any six-input function can be implemented utilizing inputs dataa, datab, datac, datad, and either datae0 and dataf0 or datae1 and dataf1. If datae0 and dataf0 are used, the output is driven to register0, and/or register0 is bypassed and the data drives out to the interconnect using the top set of output drivers (refer to Figure 2–32). If datae1 and dataf1 are used, the output drives to register1 and/or bypasses register1 and drives to the interconnect using the bottom set of output drivers. The Quartus II Compiler automatically selects the inputs to the LUT. Asynchronous load data for the register comes from the datae or dataf input of the ALM. ALMs in normal mode support register packing.

**Figure 2–32.** Six-Input Function in Normal Mode *Note (1), (2)*



**Notes to Figure 2–32:**

(1) If datae1 and dataf1 are used as inputs to the six-input function, datae0 and dataf0 are available for register packing.

(2) The dataf1 input is available for register packing only if the six-input function is un-registered.

## Extended LUT Mode

Extended LUT mode is used to implement a specific set of seven-input functions. The set must be a 2-to-1 multiplexer fed by two arbitrary five-input functions sharing four inputs. Figure 2–33 shows the template of supported seven-input functions utilizing extended LUT mode. In this mode, if the seven-input function is unregistered, the unused eighth input is available for register packing. Functions that fit into the template shown in Figure 2–33 occur naturally in designs. These functions often appear in designs as "if-else" statements in Verilog HDL or VHDL code.

**Figure 2–33.** Template for Supported Seven-Input Functions in Extended LUT Mode
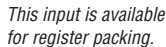


**Note to Figure 2–33:**

(1) If the seven-input function is unregistered, the unused eighth input is available for register packing. The second register, reg1, is not available.

## Arithmetic Mode

Arithmetic mode is ideal for implementing adders, counters, accumulators, wide parity functions, and comparators. An ALM in arithmetic mode uses two sets of 2 four-input LUTs along with two dedicated full adders. The dedicated adders allow the LUTs to be available to perform pre-adder logic; therefore, each adder can add the output of two four-input functions. The four LUTs share the `dataa` and `datab` inputs. As shown in Figure 2–34, the carry-in signal feeds to `adder0`, and the carry-out from `adder0` feeds to carry-in of `adder1`. The carry-out from `adder1` drives to `adder0` of the next ALM in the LAB. ALMs in arithmetic mode can drive out registered and/or unregistered versions of the adder outputs.

**Figure 2–34.** ALM in Arithmetic Mode
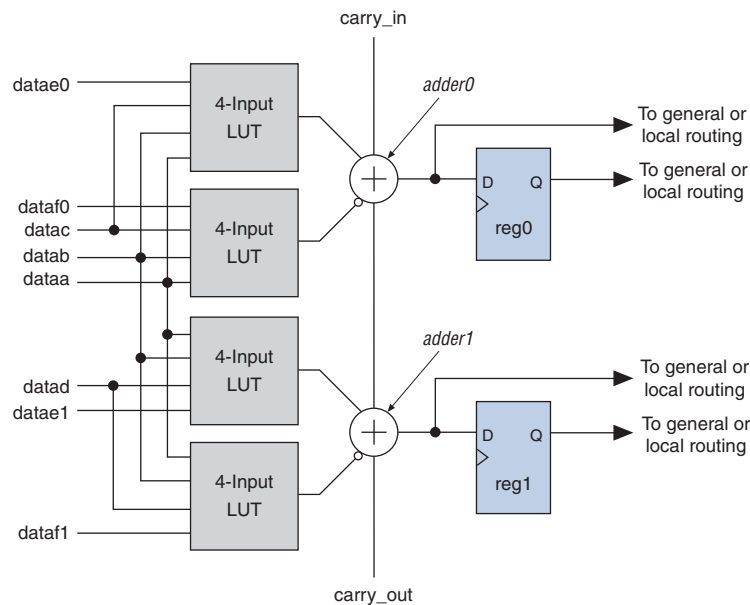


While operating in arithmetic mode, the ALM can support simultaneous use of the adder's carry output along with combinational logic outputs. In this operation, adder output is ignored. This usage of the adder with the combinational logic output provides resource savings of up to 50% for functions that can use this ability. An example of such functionality is a conditional operation, such as the one shown in Figure 2–35. The equation for this example is:

**Equation 2–1.**

$$R = (X < Y) ? Y : X$$

To implement this function, the adder is used to subtract 'Y' from 'X.' If 'X' is less than 'Y,' the `carry_out` signal is '1.' The `carry_out` signal is fed to an adder where it drives out to the LAB local interconnect. It then feeds to the LAB-wide `syncload` signal. When asserted, `syncload` selects the `syncdata` input. In this case, the data 'Y' drives the `syncdata` inputs to the registers. If 'X' is greater than or equal to 'Y,' the `syncload` signal is deasserted and 'X' drives the data port of the registers.

**Figure 2–35.** Conditional Operation Example
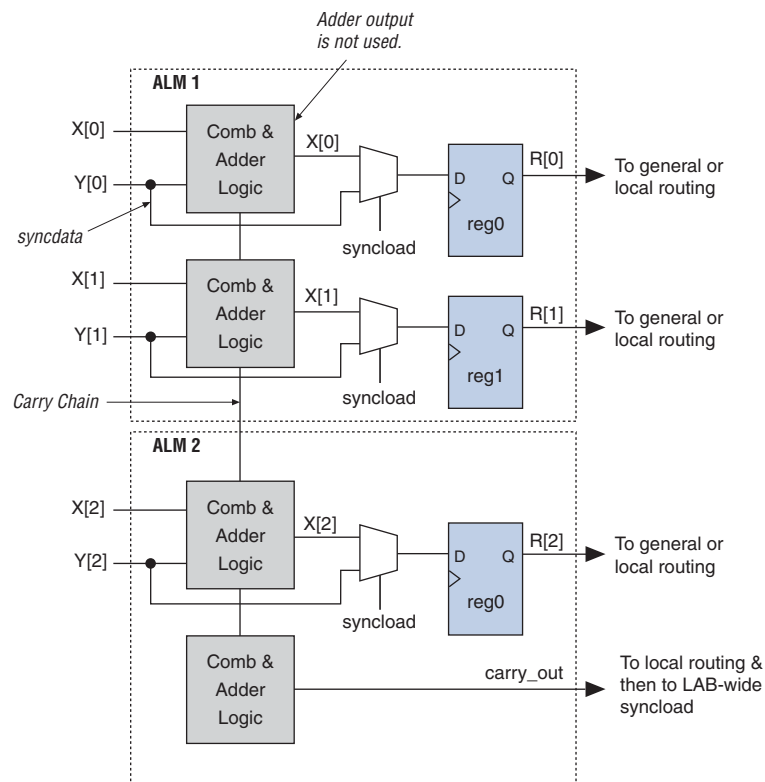


Arithmetic mode also offers clock enable, counter enable, synchronous up/down control, add/subtract control, synchronous clear, and synchronous load. The LAB local interconnect data inputs generate the clock enable, counter enable, synchronous up/down and add/subtract control signals. These control signals can be used for the inputs that are shared between the four LUTs in the ALM. The synchronous clear and synchronous load options are LAB-wide signals that affect all registers in the LAB. The Quartus II software automatically places any registers that are not used by the counter into other LABs.

### Carry Chain

Carry chain provides a fast carry function between the dedicated adders in arithmetic or shared arithmetic mode. Carry chains can begin in either the first ALM or the fifth ALM in a LAB. The final carry-out signal is routed to an ALM, where it is fed to local, row, or column interconnects.

The Quartus II Compiler automatically creates carry chain logic during compilation, or you can create it manually during design entry. Parameterized functions such as LPM functions automatically take advantage of carry chains for the appropriate functions. The Quartus II Compiler creates carry chains longer than 16 (8 ALMs in arithmetic or shared arithmetic mode) by linking LABs together automatically. For enhanced fitting, a long carry chain runs vertically allowing fast horizontal connections to TriMatrix memory and DSP blocks. A carry chain can continue as far as a full column. To avoid routing congestion in one small area of the device when a high fan-in arithmetic function is implemented, the LAB can support carry chains that only use either the top half or bottom half of the LAB before connecting to the next LAB.

The other half of the ALMs in the LAB is available for implementing narrower fan-in functions in normal mode. Carry chains that use the top four ALMs in the first LAB carries into the top half of the ALMs in the next LAB within the column. Carry chains that use the bottom four ALMs in the first LAB carries into the bottom half of the ALMs in the next LAB within the column. Every other column of the LABs are top-half bypassable, while the other LAB columns are bottom-half bypassable. For more information about carry chain interconnect, refer to .

## Shared Arithmetic Mode

In shared arithmetic mode, the ALM can implement a three-input add. In this mode, the ALM is configured with four 4-input LUTs. Each LUT either computes the sum of three inputs or the carry of three inputs. The output of the carry computation is fed to the next adder (either to `adder1` in the same ALM or to `adder0` of the next ALM in the LAB) using a dedicated connection called the shared arithmetic chain. This shared arithmetic chain can significantly improve the performance of an adder tree by reducing the number of summation stages required to implement an adder tree. Figure 2–36 shows the ALM in shared arithmetic mode.

**Figure 2–36.** ALM in Shared Arithmetic Mode
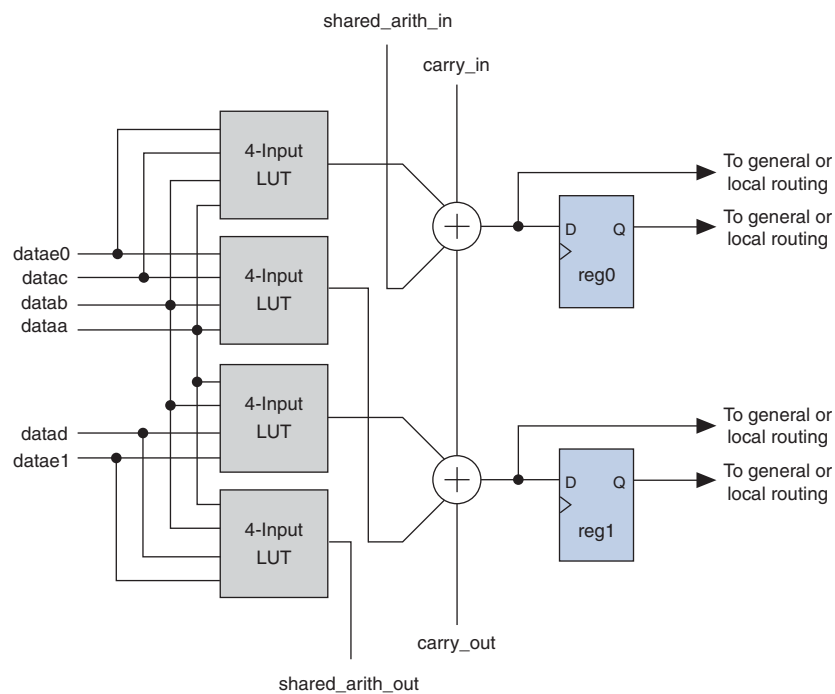


**Note to Figure 2–36:**

(1) Inputs `dataf0` and `dataf1` are available for register packing in shared arithmetic mode.

Adder trees are used in many different applications. For example, the summation of partial products in a logic-based multiplier can be implemented in a tree structure. Another example is a correlator function that can use a large adder tree to sum filtered data samples in a given time frame to recover or to de-spread data which was transmitted utilizing spread spectrum technology. An example of a three-bit add operation utilizing the shared arithmetic mode is shown in Figure 2–37. The partial sum (S[2..0]) and the partial carry (C[2..0]) is obtained using LUTs, while the result (R[2..0]) is computed using dedicated adders.

**Figure 2–37.** Example of a 3-Bit Add Utilizing Shared Arithmetic Mode
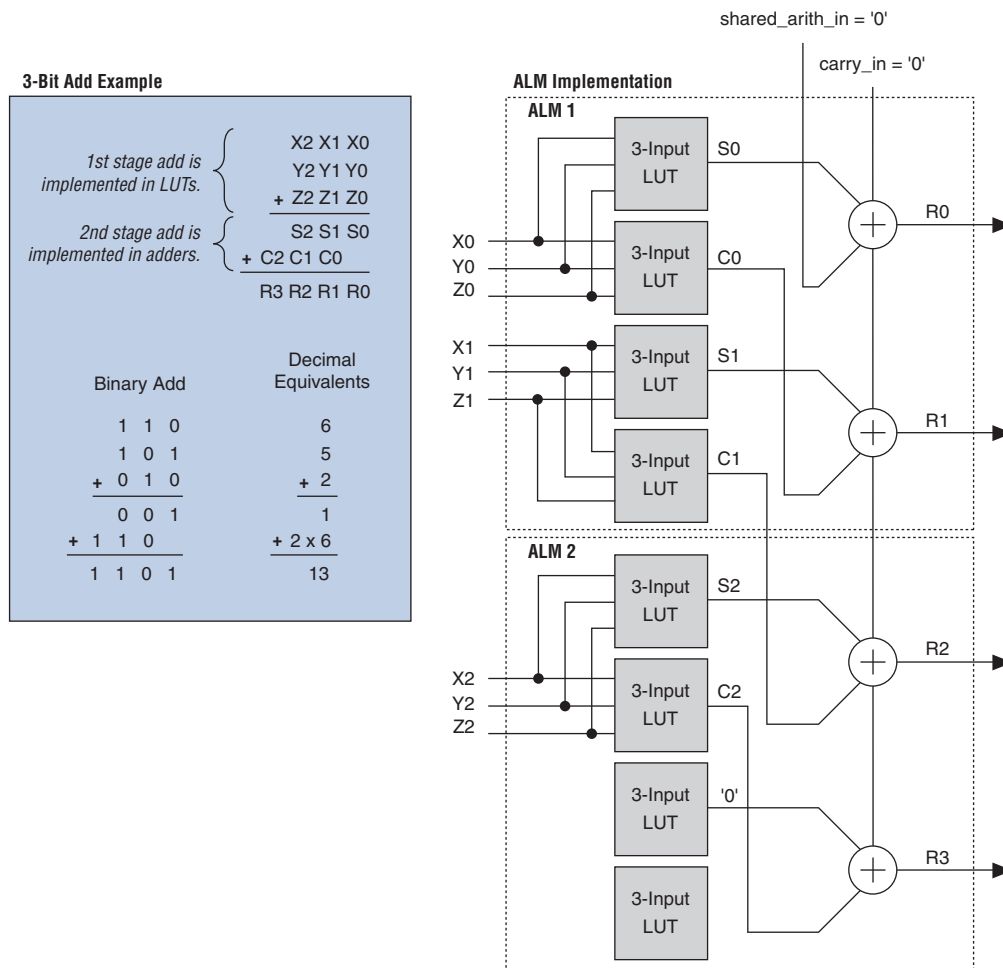


## Shared Arithmetic Chain

In addition to dedicated carry chain routing, the shared arithmetic chain available in shared arithmetic mode allows the ALM to implement a three-input add, which significantly reduces the resources necessary to implement large adder trees or correlator functions. Shared arithmetic chains can begin in either the first or fifth ALM in a LAB. The Quartus II Compiler automatically links LABs to create shared arithmetic chains longer than 16 (eight ALMs in arithmetic or shared arithmetic mode). For enhanced fitting, a long shared arithmetic chain runs vertically allowing fast horizontal connections to TriMatrix memory and DSP blocks. A shared arithmetic chain can continue as far as a full column. Similar to carry chains, shared arithmetic

chains are also top- or bottom-half bypassable. This capability allows the shared arithmetic chain to cascade through half of the ALMs in a LAB while leaving the other half available for narrower fan-in functionality. Every other LAB column is top-half bypassable, while the other LAB columns are bottom-half bypassable. For more information about shared arithmetic chain interconnect, refer to "MultiTrack Interconnect" on page 2–44.

## Register Chain

In addition to the general routing outputs, the ALMs in a LAB have register chain outputs. Register chain routing allows registers in the same LAB to be cascaded together. The register chain interconnect allows a LAB to use LUTs for a single combinational function and the registers to be used for an unrelated shift register implementation. These resources speed up connections between ALMs while saving local interconnect resources (refer to Figure 2–38). The Quartus II Compiler automatically takes advantage of these resources to improve utilization and performance. For more information about register chain interconnect, refer to "MultiTrack Interconnect" on page 2–44.

**Figure 2–38.** Register Chain within a LAB    *(Note 1)*
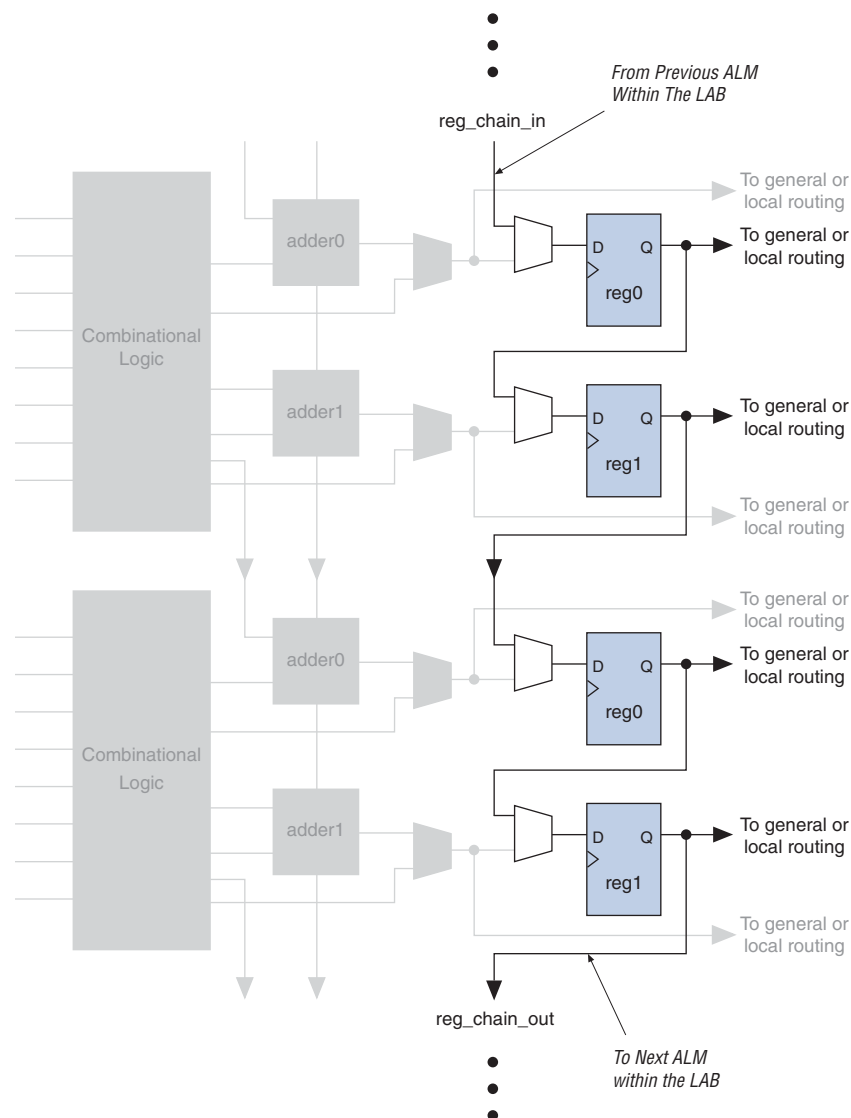


**Note to Figure 2–38:**

(1) The combinational or adder logic can be used to implement an unrelated, unregistered function.

## Clear and Preset Logic Control

LAB-wide signals control the logic for the register's clear and load/preset signals. The ALM directly supports an asynchronous clear and preset function. The register preset is achieved through the asynchronous load of a logic high. The direct asynchronous preset does not require a NOT gate push-back technique. Arria GX devices support simultaneous asynchronous load/preset and clear signals. An asynchronous clear signal takes precedence if both signals are asserted simultaneously. Each LAB supports up to two clears and one load/preset signal.

In addition to the clear and load/preset ports, Arria GX devices provide a device-wide reset pin (DEV_CLRn) that resets all registers in the device. An option set before compilation in the Quartus II software controls this pin. This device-wide reset overrides all other control signals.

# MultiTrack Interconnect

In Arria GX architecture, the MultiTrack interconnect structure with DirectDrive technology provides connections between ALMs, TriMatrix memory, DSP blocks, and device I/O pins. The MultiTrack interconnect consists of continuous, performance-optimized routing lines of different lengths and speeds used for inter- and intra-design block connectivity. The Quartus II Compiler automatically places critical design paths on faster interconnects to improve design performance.

DirectDrive technology is a deterministic routing technology that ensures identical routing resource usage for any function regardless of placement in the device. The MultiTrack interconnect and DirectDrive technology simplify the integration stage of block-based designing by eliminating the re-optimization cycles that typically follow design changes and additions.

The MultiTrack interconnect consists of row and column interconnects that span fixed distances. A routing structure with fixed length resources for all devices allows predictable and repeatable performance when migrating through different device densities. Dedicated row interconnects route signals to and from LABs, DSP blocks, and TriMatrix memory in the same row.

These row resources include:

■ Direct link interconnects between LABs and adjacent blocks

■ R4 interconnects traversing four blocks to the right or left

■ R24 row interconnects for high-speed access across the length of the device

The direct link interconnect allows a LAB, DSP block, or TriMatrix memory block to drive into the local interconnect of its left and right neighbors and then back into itself, providing fast communication between adjacent LABs and/or blocks without using row interconnect resources.

The R4 interconnects span four LABs, three LABs and one M512 RAM block, two LABs and one M4K RAM block, or two LABs and one DSP block to the right or left of a source LAB. These resources are used for fast row connections in a four-LAB region. Every LAB has its own set of R4 interconnects to drive either left or right. Figure 2–39 shows R4 interconnect connections from a LAB.

R4 interconnects can drive and be driven by DSP blocks and RAM blocks and row IOEs. For LAB interfacing, a primary LAB or LAB neighbor can drive a given R4 interconnect. For R4 interconnects that drive to the right, the primary LAB and right neighbor can drive onto the interconnect. For R4 interconnects that drive to the left, the primary LAB and its left neighbor can drive onto the interconnect. R4 interconnects can drive other R4 interconnects to extend the range of LABs they can drive. R4 interconnects can also drive C4 and C16 interconnects for connections from one row to another. Additionally, R4 interconnects can drive R24 interconnects.

**Figure 2–39.** R4 Interconnect Connections *(Note 1)*, *(2)*, *(3)*
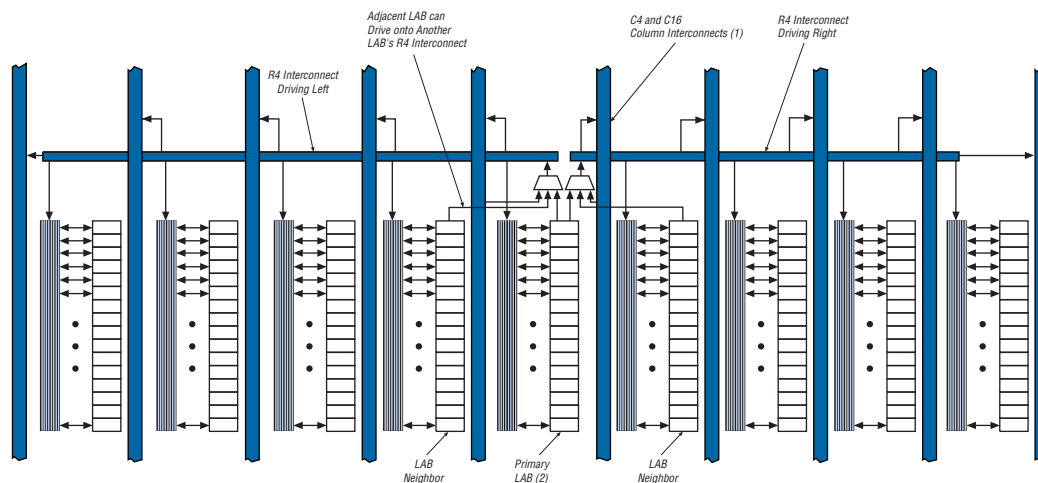


**Notes to Figure 2–39:**

(1) C4 and C16 interconnects can drive R4 interconnects.

(2) This pattern is repeated for every LAB in the LAB row.

(3) The LABs in Figure 2–39 show the 16 possible logical outputs per LAB.

R24 row interconnects span 24 LABs and provide the fastest resource for long row connections between LABs, TriMatrix memory, DSP blocks, and row IOEs. The R24 row interconnects can cross M-RAM blocks. R24 row interconnects drive to other row or column interconnects at every fourth LAB and do not drive directly to LAB local interconnects. R24 row interconnects drive LAB local interconnects via R4 and C4 interconnects. R24 interconnects can drive R24, R4, C16, and C4 interconnects. The column interconnect operates similarly to the row interconnect and vertically routes signals to and from LABs, TriMatrix memory, DSP blocks, and IOEs. Each column of LABs is served by a dedicated column interconnect.

These column resources include:

■ Shared arithmetic chain interconnects in a LAB

■ Carry chain interconnects in a LAB and from LAB to LAB

■ Register chain interconnects in a LAB

■ C4 interconnects traversing a distance of four blocks in up and down direction

■ C16 column interconnects for high-speed vertical routing through the device

Arria GX devices include an enhanced interconnect structure in LABs for routing shared arithmetic chains and carry chains for efficient arithmetic functions. The register chain connection allows the register output of one ALM to connect directly to the register input of the next ALM in the LAB for fast shift registers. These ALM-to-ALM connections bypass the local interconnect. The Quartus II Compiler automatically takes advantage of these resources to improve utilization and performance. Figure 2–40 shows shared arithmetic chain, carry chain, and register chain interconnects.

**Figure 2–40.** Shared Arithmetic Chain, Carry Chain and Register Chain Interconnects
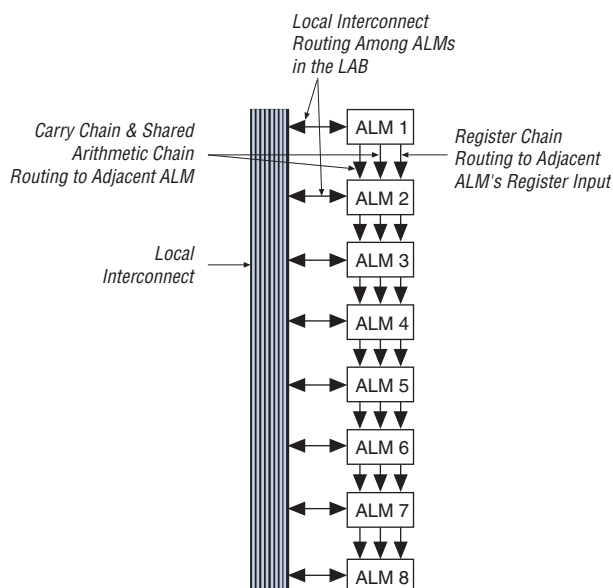


C4 interconnects span four LABs, M512, or M4K blocks up or down from a source LAB. Every LAB has its own set of C4 interconnects to drive either up or down. Figure 2–41 shows the C4 interconnect connections from a LAB in a column. C4 interconnects can drive and be driven by all types of architecture blocks, including DSP blocks, TriMatrix memory blocks, and column and row IOEs. For LAB interconnection, a primary LAB or its LAB neighbor can drive a given C4 interconnect. C4 interconnects can drive each other to extend their range as well as drive row interconnects for column-to-column connections.

**Figure 2–41.** C4 Interconnect Connections   *(Note 1)*
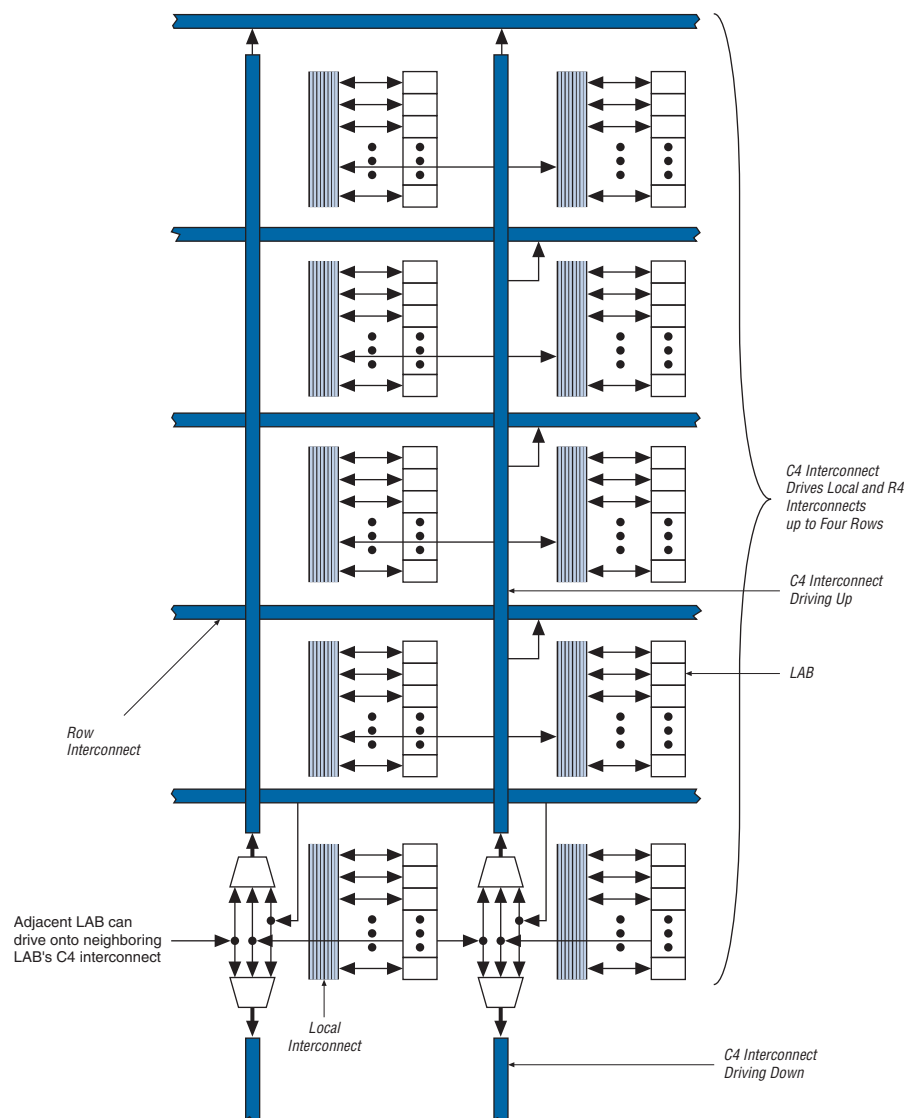


**Note to Figure 2–41:**

(1)   Each C4 interconnect can drive either up or down four rows.

C16 column interconnects span a length of 16 LABs and provide the fastest resource for long column connections between LABs, TriMatrix memory blocks, DSP blocks, and IOEs. C16 interconnects can cross M-RAM blocks and also drive to row and column interconnects at every fourth LAB. C16 interconnects drive LAB local interconnects via C4 and R4 interconnects and do not drive LAB local interconnects directly. All embedded blocks communicate with the logic array similar to LAB-to-LAB interfaces. Each block (that is, TriMatrix memory and DSP blocks) connects to row and column interconnects and has local interconnect regions driven by row and column interconnects. These blocks also have direct link interconnects for fast connections to and from a neighboring LAB. All blocks are fed by the row LAB clocks, `labclk[5..0]`.

Table 2–10 lists the routing scheme for Arria GX device.

**Table 2–10.** Arria GX Device Routing Scheme

| Source | Shared Arithmetic Chain | Carry Chain | Register Chain | Local Interconnect | Direct Link Interconnect | R4 Interconnect | R24 Interconnect | C4 Interconnect | C16 Interconnect | ALM | M512 RAM Block | M4K RAM Block | M-RAM Block | DSP Blocks | Column IOE | Row IOE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Shared arithmetic chain | — | — | — | — | — | — | — | — | — | ✓ | — | — | — | — | — | — |
| Carry chain | — | — | — | — | — | — | — | — | — | ✓ | — | — | — | — | — | — |
| Register chain | — | — | — | — | — | — | — | — | — | ✓ | — | — | — | — | — | — |
| Local interconnect | — | — | — | — | — | — | — | — | — | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Direct link interconnect | — | — | — | ✓ | — | — | — | — | — | — | — | — | — | — | — | — |
| R4 interconnect | — | — | — | ✓ | — | ✓ | ✓ | ✓ | ✓ | — | — | — | — | — | — | — |
| R24 interconnect | — | — | — | — | — | ✓ | ✓ | ✓ | ✓ | — | — | — | — | — | — | — |
| C4 interconnect | — | — | — | ✓ | — | ✓ | — | ✓ | ✓ | — | — | — | — | — | — | — |
| C16 interconnect | — | — | — | — | — | ✓ | ✓ | ✓ | ✓ | — | — | — | — | — | — | — |
| ALM | ✓ | ✓ | ✓ | ✓ | ✓ | — | — | ✓ | — | — | — | — | — | — | — | — |
| M512 RAM block | — | — | — | ✓ | ✓ | ✓ | — | ✓ | — | — | — | — | — | — | — | — |
| M4K RAM block | — | — | — | ✓ | ✓ | ✓ | — | ✓ | — | — | — | — | — | — | — | — |
| M-RAM block | — | — | — | — | — | ✓ | ✓ | ✓ | ✓ | — | — | — | — | — | — | — |
| DSP blocks | — | — | — | — | — | ✓ | ✓ | ✓ | — | — | — | — | — | — | — | — |
| Column IOE | — | — | — | — | — | ✓ | — | ✓ | ✓ | — | — | — | — | — | — | — |
| Row IOE | — | — | — | — | — | ✓ | ✓ | ✓ | ✓ | — | — | — | — | — | — | — |

# TriMatrix Memory

TriMatrix memory consists of three types of RAM blocks: M512, M4K, and M-RAM. Although these memory blocks are different, they can all implement various types of memory with or without parity, including true dual-port, simple dual-port, and single-port RAM, ROM, and FIFO buffers. Table 2–11 lists the size and features of the different RAM blocks.

**Table 2–11.** TriMatrix Memory Features  (Part 1 of 2)

| Memory Feature | M512 RAM Block (32 × 18 Bits) | M4K RAM Block (128 × 36 Bits) | M-RAM Block (4K × 144 Bits) |
|---|---|---|---|
| Maximum performance | 345 MHz | 380 MHz | 290 MHz |
| True dual-port memory | — | ✓ | ✓ |
| Simple dual-port memory | ✓ | ✓ | ✓ |
| Single-port memory | ✓ | ✓ | ✓ |
| Shift register | ✓ | ✓ | — |

**Table 2–11.** TriMatrix Memory Features  (Part 2 of 2)

| Memory Feature | M512 RAM Block (32 × 18 Bits) | M4K RAM Block (128 × 36 Bits) | M-RAM Block (4K × 144 Bits) |
|---|---|---|---|
| ROM | ✓ | ✓ | — |
| FIFO buffer | ✓ | ✓ | ✓ |
| Pack mode | — | ✓ | ✓ |
| Byte enable | ✓ | ✓ | ✓ |
| Address clock enable | — | ✓ | ✓ |
| Parity bits | ✓ | ✓ | ✓ |
| Mixed clock mode | ✓ | ✓ | ✓ |
| Memory initialization file (**.mif**) | ✓ | ✓ | — |
| Simple dual-port memory mixed width support | ✓ | ✓ | ✓ |
| True dual-port memory mixed width support | — | ✓ | ✓ |
| Power-up conditions | Outputs cleared | Outputs cleared | Outputs unknown |
| Register clears | Output registers | Output registers | Output registers |
| Mixed-port read-during-write | Unknown output/old data | Unknown output/old data | Unknown output |
| Configurations | 512 × 1<br>256 × 2<br>128 × 4<br>64 × 8<br>64 × 9<br>32 × 16<br>32 × 18 | 4K × 1<br>2K × 2<br>1K × 4<br>512 × 8<br>512 × 9<br>256 × 16<br>256 × 18<br>128 × 32<br>128 × 36 | 64K × 8<br>64K × 9<br>32K × 16<br>32K × 18<br>16K × 32<br>16K × 36<br>8K × 64<br>8K × 72<br>4K × 128<br>4K × 144 |

TriMatrix memory provides three different memory sizes for efficient application support. The Quartus II software automatically partitions the user-defined memory into the embedded memory blocks using the most efficient size combinations. You can also manually assign the memory to a specific block size or a mixture of block sizes.

## M512 RAM Block

The M512 RAM block is a simple dual-port memory block and is useful for implementing small FIFO buffers, DSP, and clock domain transfer applications. Each block contains 576 RAM bits (including parity bits). M512 RAM blocks can be configured in the following modes:

■ Simple dual-port RAM

■ Single-port RAM

■ FIFO

■ ROM

■ Shift register

When configured as RAM or ROM, you can use an initialization file to pre-load the memory contents.

M512 RAM blocks can have different clocks on its inputs and outputs. The `wren`, `datain`, and write address registers are all clocked together from one of the two clocks feeding the block. The read address, `rden`, and output registers can be clocked by either of the two clocks driving the block, allowing the RAM block to operate in read and write or input and output clock modes. Only the output register can be bypassed. The six `labclk` signals or local interconnect can drive the `inclock`, `outclock`, `wren`, `rden`, and `outclr` signals. Because of the advanced interconnect between the LAB and M512 RAM blocks, ALMs can also control the `wren` and `rden` signals and the RAM clock, clock enable, and asynchronous clear signals. Figure 2–42 shows the M512 RAM block control signal generation logic.

**Figure 2–42.** M512 RAM Block Control Signals
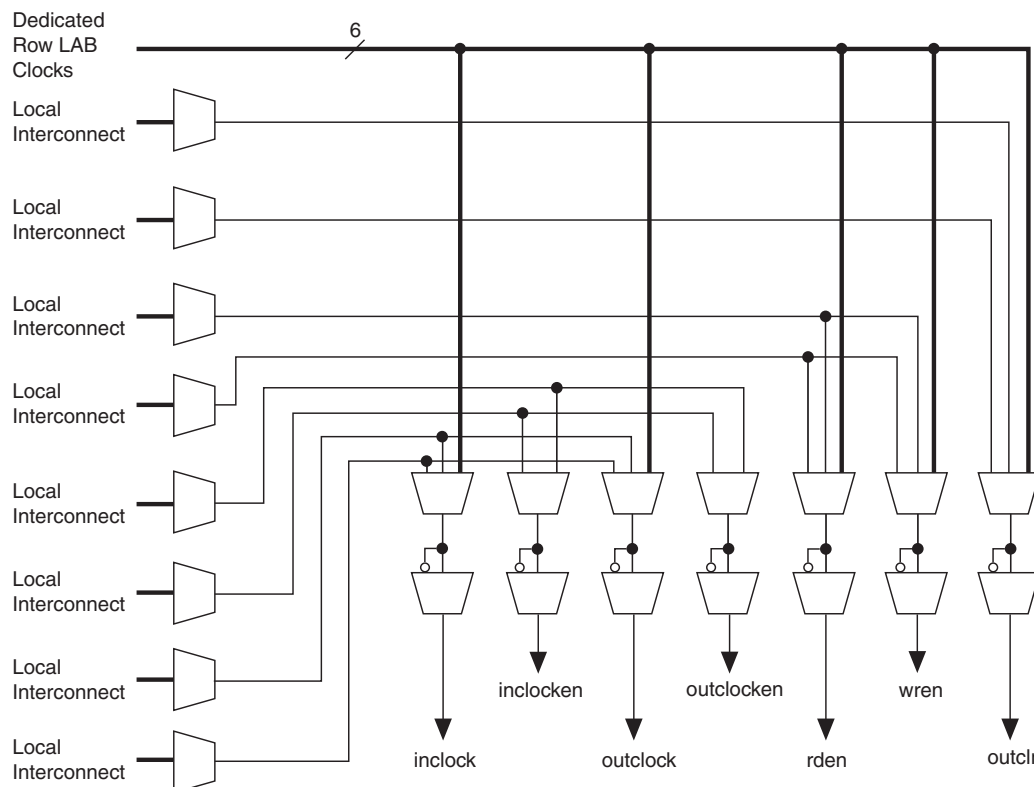


The RAM blocks in Arria GX devices have local interconnects to allow ALMs and interconnects to drive into RAM blocks. The M512 RAM block local interconnect is driven by the R4, C4, and direct link interconnects from adjacent LABs. The M512 RAM blocks can communicate with LABs on either the left or right side through these row interconnects or with LAB columns on the left or right side with the column interconnects. The M512 RAM block has up to 16 direct link input connections from the left adjacent LABs and another 16 from the right adjacent LAB. M512 RAM outputs can also connect to left and right LABs through direct link interconnect. The M512 RAM block has equal opportunity for access and performance to and from LABs on either its left or right side. Figure 2–43 shows the M512 RAM block to logic array interface.

**Figure 2–43.** M512 RAM Block LAB Row Interface
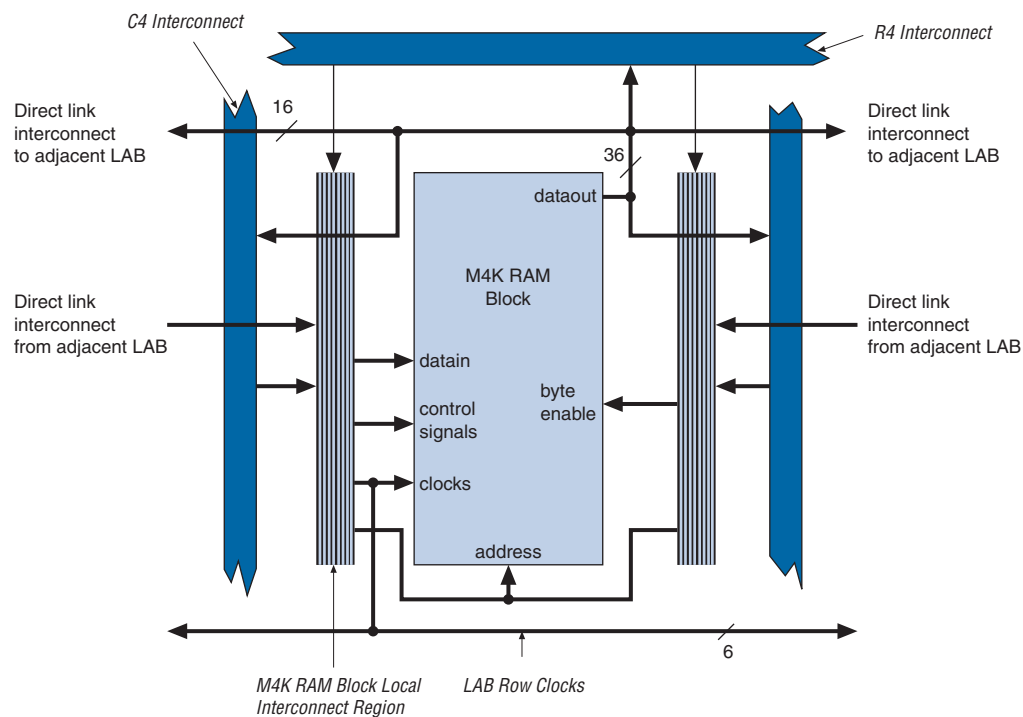


## M4K RAM Blocks

The M4K RAM block includes support for true dual-port RAM. The M4K RAM block is used to implement buffers for a wide variety of applications such as storing processor code, implementing lookup schemes, and implementing larger memory applications. Each block contains 4,608 RAM bits (including parity bits). M4K RAM blocks can be configured in the following modes:

■ True dual-port RAM

■ Simple dual-port RAM

■ Single-port RAM

■ FIFO

■ ROM

■ Shift register

When configured as RAM or ROM, you can use an initialization file to pre-load the memory contents.

M4K RAM blocks allow for different clocks on their inputs and outputs. Either of the two clocks feeding the block can clock M4K RAM block registers (renwe, address, byte enable, datain, and output registers). Only the output register can be bypassed. The six labclk signals or local interconnects can drive the control signals for the A and B ports of the M4K RAM block. ALMs can also control the clock_a, clock_b, renwe_a, renwe_b, clr_a, clr_b, clocken_a, and clocken_b signals, as shown in Figure 2–44.

**Figure 2–44.** M4K RAM Block Control Signals
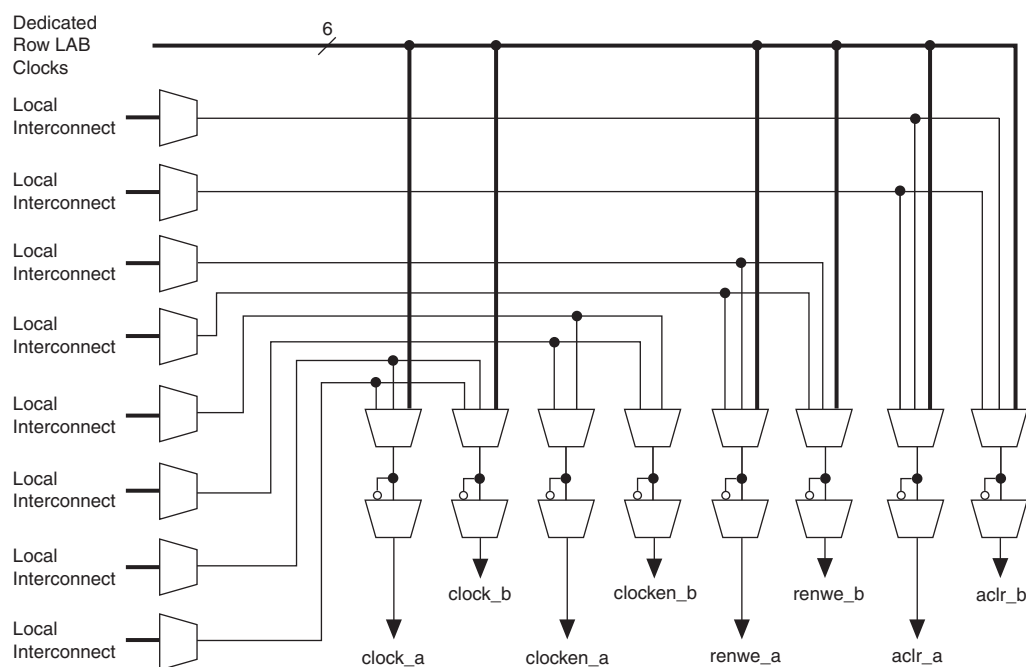


The R4, C4, and direct link interconnects from adjacent LABs drive the M4K RAM block local interconnect. The M4K RAM blocks can communicate with LABs on either the left or right side through these row resources or with LAB columns on either the right or left with the column resources. Up to 16 direct link input connections to the M4K RAM block are possible from the left adjacent LABs and another 16 are possible from the right adjacent LAB. M4K RAM block outputs can also connect to left and right LABs through direct link interconnect. Figure 2–45 shows the M4K RAM block to logic array interface.

**Figure 2–45.** M4K RAM Block LAB Row Interface
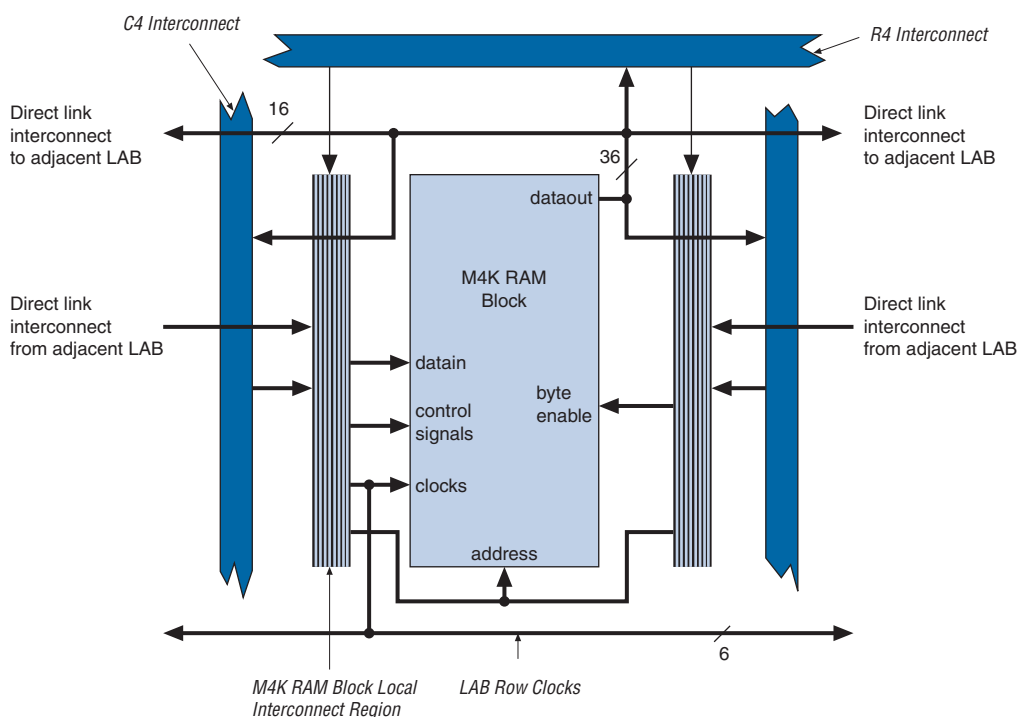


## M-RAM Block

The largest TriMatrix memory block, the M-RAM block, is useful for applications where a large volume of data must be stored on-chip. Each block contains 589,824 RAM bits (including parity bits). The M-RAM block can be configured in the following modes:

■ True dual-port RAM

■ Simple dual-port RAM

■ Single-port RAM

■ FIFO

You cannot use an initialization file to initialize the contents of a M-RAM block. All M-RAM block contents power up to an undefined value. Only synchronous operation is supported in the M-RAM block, so all inputs are registered. Output registers can be bypassed.

Similar to all RAM blocks, M-RAM blocks can have different clocks on their inputs and outputs. Either of the two clocks feeding the block can clock M-RAM block registers (renwe, address, byte enable, datain, and output registers). You can bypass the output register. The six labclk signals or local interconnect can drive the control signals for the A and B ports of the M-RAM block. ALMs can also control the clock_a, clock_b, renwe_a, renwe_b, clr_a, clr_b, clocken_a, and clocken_b signals, as shown in Figure 2–46.

**Figure 2–46.** M-RAM Block Control Signals
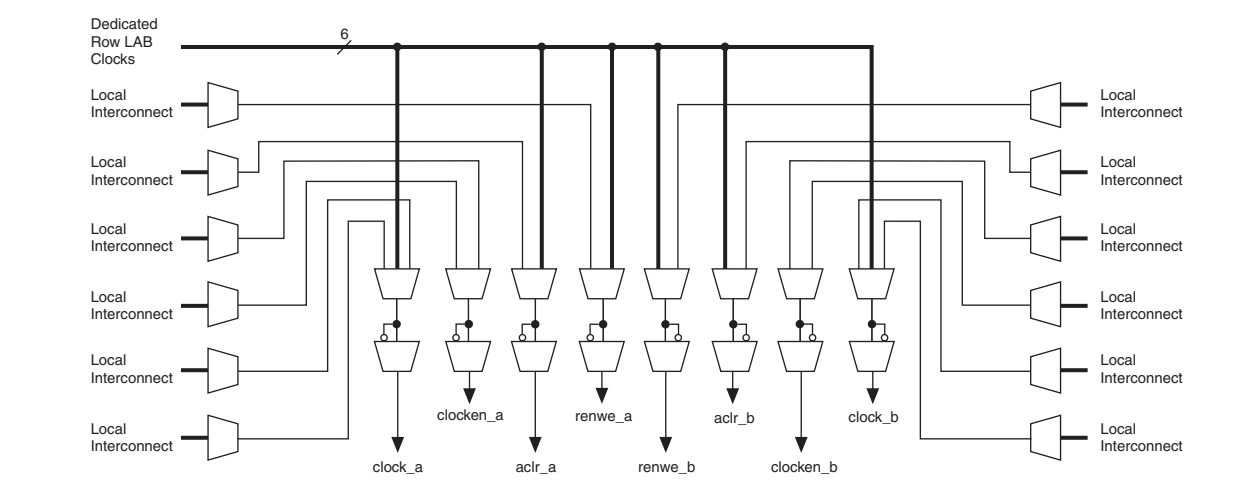


The R4, R24, C4, and direct link interconnects from adjacent LABs on either the right or left side drive the M-RAM block local interconnect. Up to 16 direct link input connections to the M-RAM block are possible from the left adjacent LABs and another 16 are possible from the right adjacent LAB. M-RAM block outputs can also connect to left and right LABs through direct link interconnect. Figure 2–47 shows an example floorplan for the EP1AGX90 device and the location of the M-RAM interfaces. Figure 2–48 and Figure 2–49 show the interface between the M-RAM block and the logic array.

**Figure 2–47.** EP1AGX90 Device with M-RAM Interface Locations    *(Note 1)*
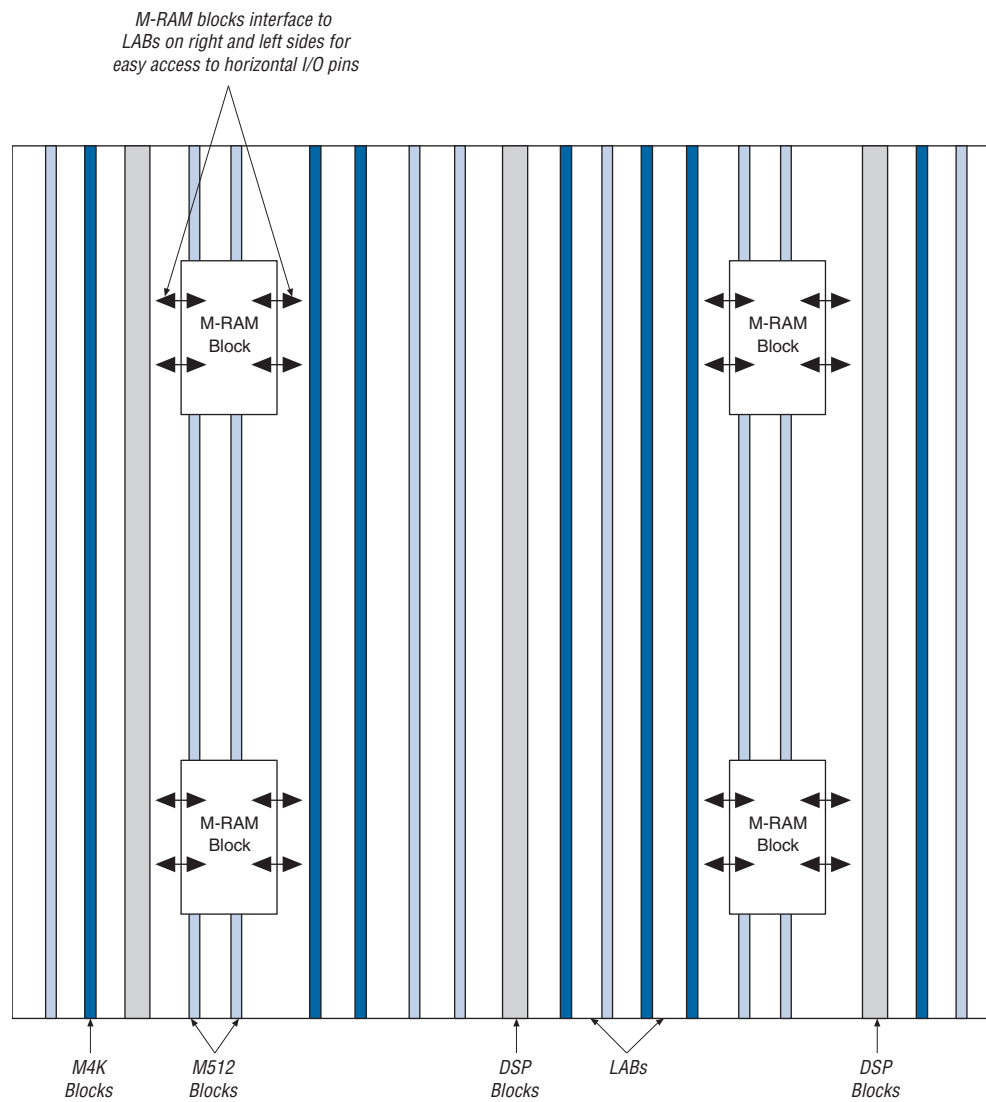


**Note to Figure 2–47:**

(1)    The device shown is an EP1AGX90 device. The number and position of M-RAM blocks vary in other devices.

**Figure 2–48.** M-RAM Block LAB Row Interface    *(Note 1)*
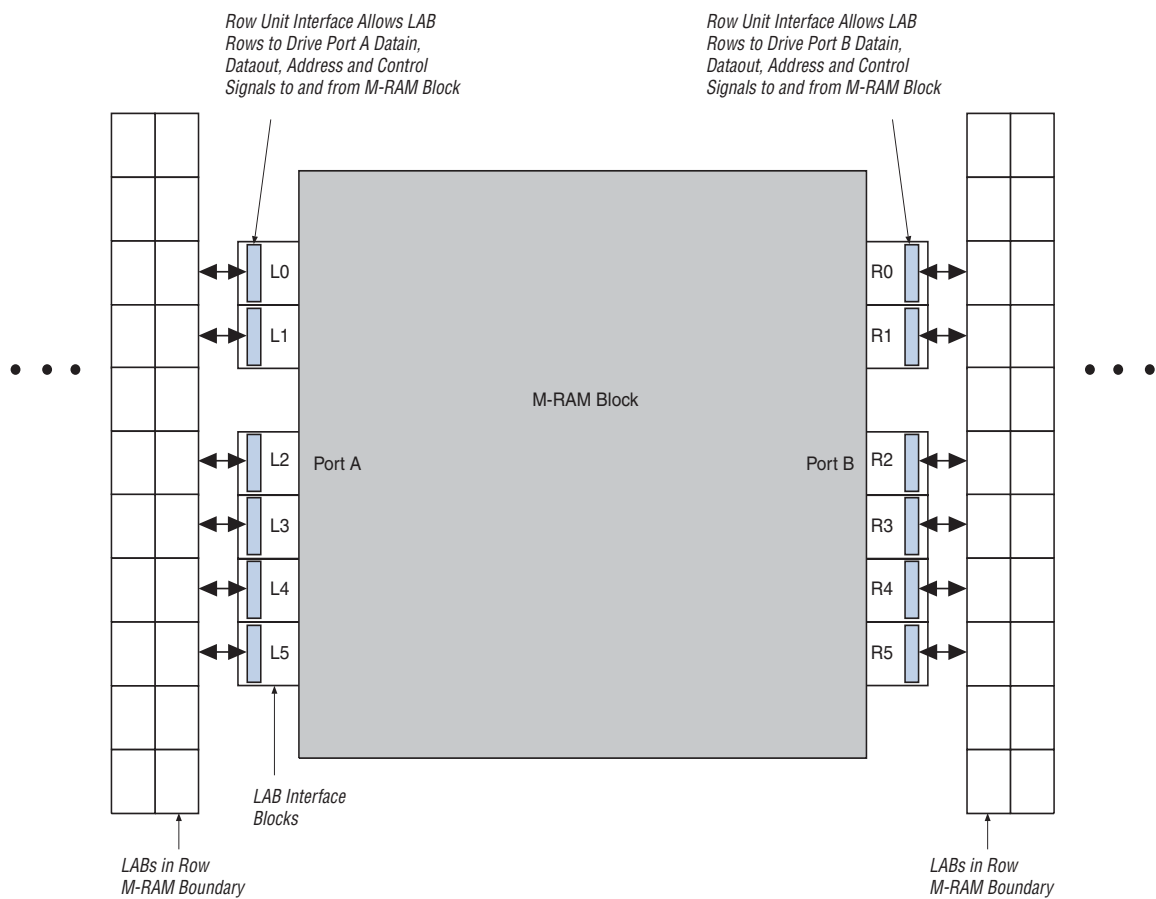


**Note to Figure 2–48:**

(1)    Only R24 and C16 interconnects cross the M-RAM block boundaries.

**Figure 2–49.** M-RAM Row Unit Interface to Interconnect
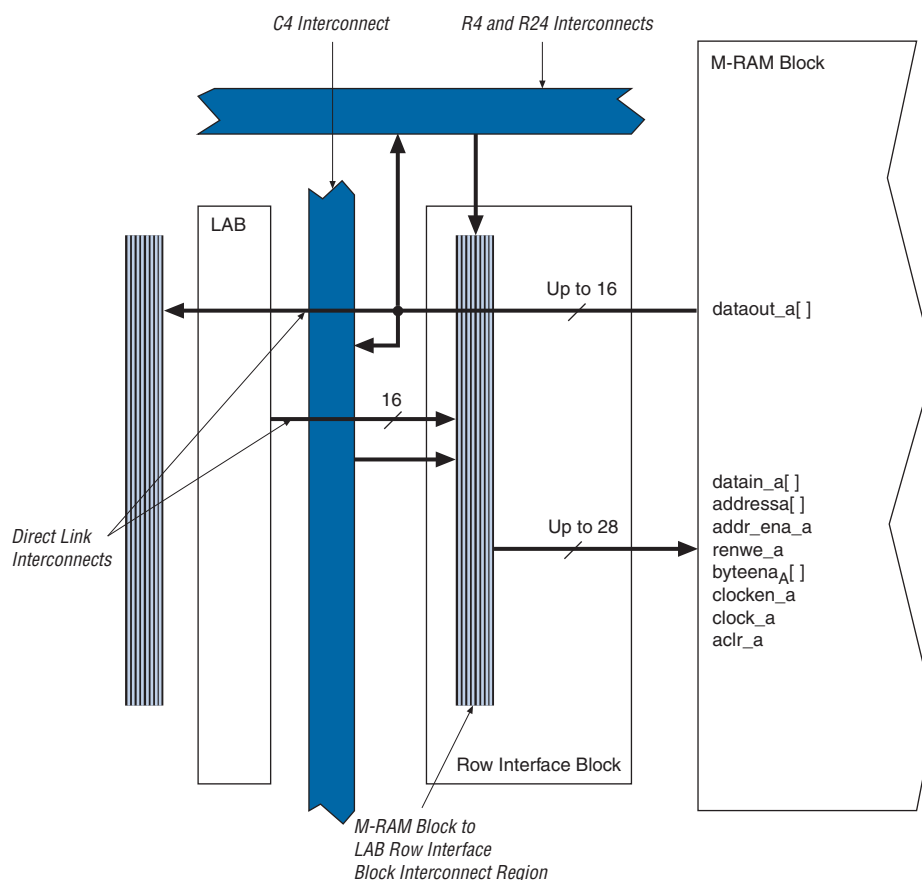


Table 2–12 lists the input and output data signal connections along with the address and control signal input connections to the row unit interfaces (L0 to L5 and R0 to R5).

**Table 2–12.** M-RAM Row Interface Unit Signals (Part 1 of 2)

| Unit Interface Block | Input Signals | Output Signals |
|---|---|---|
| L0 | `datain_a[14..0]`<br>`byteena_a[1..0]` | `dataout_a[11..0]` |
| L1 | `datain_a[29..15]`<br>`byteena_a[3..2]` | `dataout_a[23..12]` |
| L2 | `datain_a[35..30]`<br>`addressa[4..0]`<br>`addr_ena_a`<br>`clock_a`<br>`clocken_a`<br>`renwe_a`<br>`aclr_a` | `dataout_a[35..24]` |
| L3 | `addressa[15..5]`<br>`datain_a[41..36]` | `dataout_a[47..36]` |

**Table 2–12.** M-RAM Row Interface Unit Signals  (Part 2 of 2)

| Unit Interface Block | Input Signals | Output Signals |
|---|---|---|
| L4 | `datain_a[56..42]`<br>`byteena_a[5..4]` | `dataout_a[59..48]` |
| L5 | `datain_a[71..57]`<br>`byteena_a[7..6]` | `dataout_a[71..60]` |
| R0 | `datain_b[14..0]`<br>`byteena_b[1..0]` | `dataout_b[11..0]` |
| R1 | `datain_b[29..15]`<br>`byteena_b[3..2]` | `dataout_b[23..12]` |
| R2 | `datain_b[35..30]`<br>`addressb[4..0]`<br>`addr_ena_b`<br>`clock_b`<br>`clocken_b`<br>`renwe_b`<br>`aclr_b` | `dataout_b[35..24]` |
| R3 | `addressb[15..5]`<br>`datain_b[41..36]` | `dataout_b[47..36]` |
| R4 | `datain_b[56..42]`<br>`byteena_b[5..4]` | `dataout_b[59..48]` |
| R5 | `datain_b[71..57]`<br>`byteena_b[7..6]` | `dataout_b[71..60]` |

For more information about TriMatrix memory, refer to the *TriMatrix Embedded Memory Blocks in Arria GX Devices* chapter.

# Digital Signal Processing Block

The most commonly used DSP functions are finite impulse response (FIR) filters, complex FIR filters, infinite impulse response (IIR) filters, fast Fourier transform (FFT) functions, direct cosine transform (DCT) functions, and correlators. All of these use the multiplier as the fundamental building block. Additionally, some applications need specialized operations such as multiply-add and multiply-accumulate operations. Arria GX devices provide DSP blocks to meet the arithmetic requirements of these functions.

Each Arria GX device has two to four columns of DSP blocks to efficiently implement DSP functions faster than ALM-based implementations. Each DSP block can be configured to support up to:

■ Eight 9 × 9-bit multipliers

■ Four 18 × 18-bit multipliers

■ One 36 × 36-bit multiplier

As indicated, the Arria GX DSP block can support one 36 × 36-bit multiplier in a single DSP block and is true for any combination of signed, unsigned, or mixed sign multiplications.

Figure 2–50 shows one of the columns with surrounding LAB rows.

**Figure 2–50.** DSP Blocks Arranged in Columns
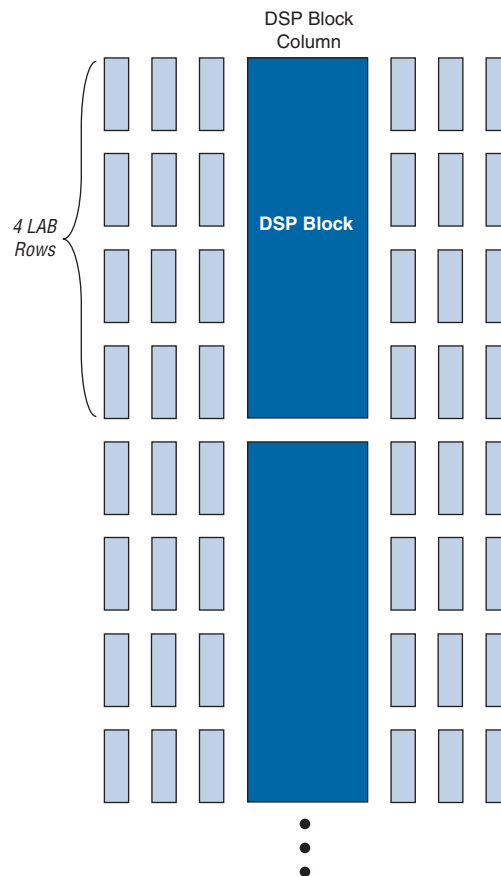


Table 2–13 lists the number of DSP blocks in each Arria GX device. DSP block multipliers can optionally feed an adder/subtractor or accumulator in the block depending on the configuration, which makes routing to ALMs easier, saves ALM routing resources, and increases performance because all connections and blocks are in the DSP block.

**Table 2–13.** DSP Blocks in Arria GX Devices    *(Note 1)*

| Device | DSP Blocks | Total 9 × 9 Multipliers | Total 18 × 18 Multipliers | Total 36 × 36 Multipliers |
|---|---|---|---|---|
| EP1AGX20 | 10 | 80 | 40 | 10 |
| EP1AGX35 | 14 | 112 | 56 | 14 |
| EP1AGX50 | 26 | 208 | 104 | 26 |
| EP1AGX60 | 32 | 256 | 128 | 32 |
| EP1AGX90 | 44 | 352 | 176 | 44 |

**Note to Table 2–13:**

(1)    This list only shows functions that can fit into a single DSP block. Multiple DSP blocks can support larger multiplication functions.

Additionally, DSP block input registers can efficiently implement shift registers for FIR filter applications. DSP blocks support Q1.15 format rounding and saturation. Figure 2–51 shows a top-level diagram of the DSP block configured for 18 × 18-bit multiplier mode.

**Figure 2–51.** DSP Block Diagram for 18 × 18-Bit Configuration
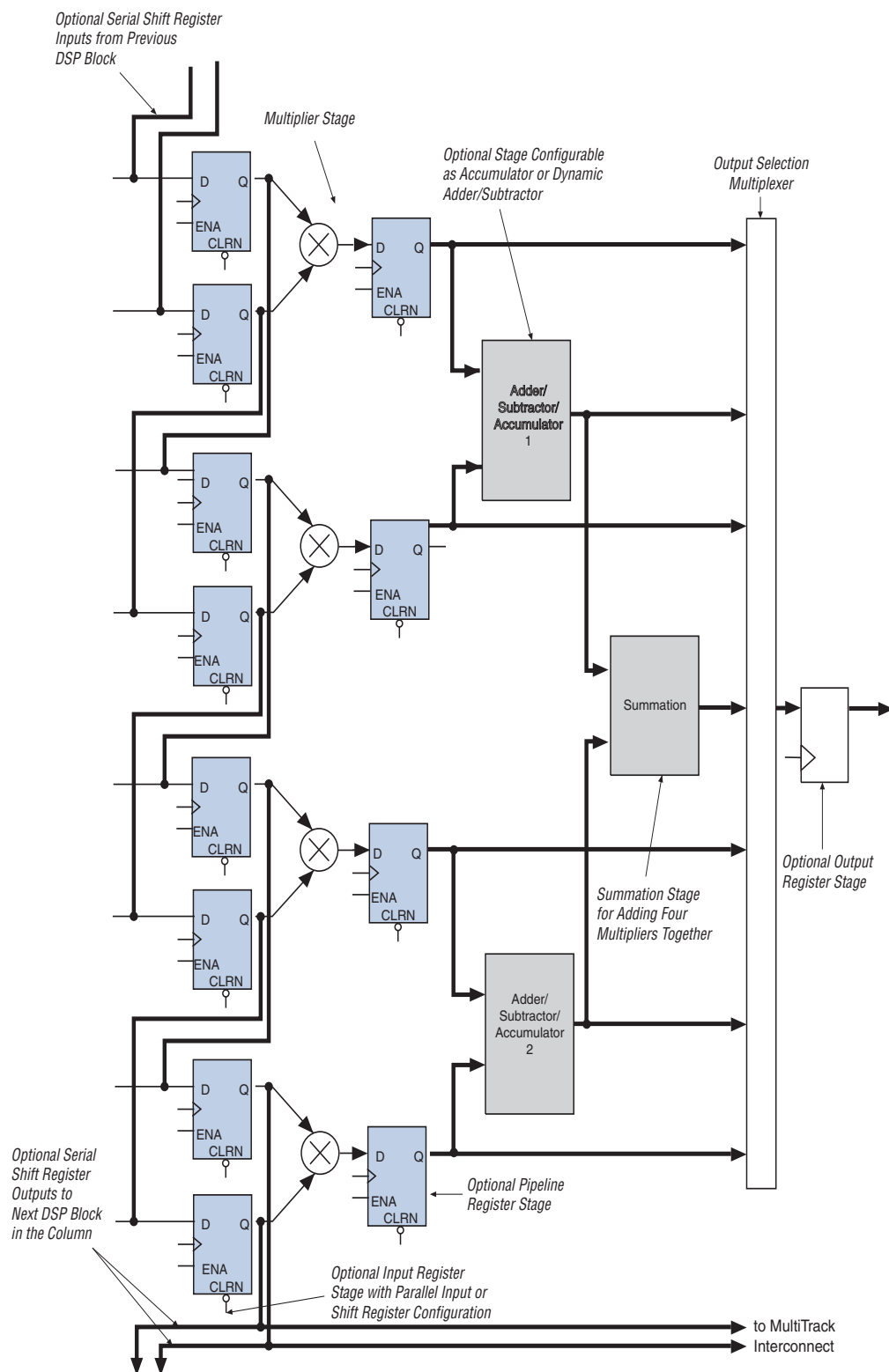
## Modes of Operation

The adder, subtractor, and accumulate functions of a DSP block have four modes of operation:

■ Simple multiplier

■ Multiply-accumulator

■ Two-multipliers adder

■ Four-multipliers adder

Table 2–14 shows the different number of multipliers possible in each DSP block mode according to size. These modes allow the DSP blocks to implement numerous applications for DSP including FFTs, complex FIR, FIR, 2D FIR filters, equalizers, IIR, correlators, matrix multiplication, and many other functions. DSP blocks also support mixed modes and mixed multiplier sizes in the same block. For example, half of one DSP block can implement one $18 \times 18$-bit multiplier in multiply-accumulator mode, while the other half of the DSP block implements four $9 \times 9$-bit multipliers in simple multiplier mode.

**Table 2–14.** Multiplier Size and Configurations per DSP Block

| DSP Block Mode | 9 × 9 | 18 × 18 | 36 × 36 |
|---|---|---|---|
| Multiplier | Eight multipliers with eight product outputs | Four multipliers with four product outputs | One multiplier with one product output |
| Multiply-accumulator | — | Two 52-bit multiply-accumulate blocks | — |
| Two-multipliers adder | Four two-multiplier adder (two 9 × 9 complex multiply) | Two two-multiplier adder (one 18 × 18 complex multiply) | — |
| Four-multipliers adder | Two four-multiplier adder | One four-multiplier adder | — |

## DSP Block Interface

The Arria GX device DSP block input registers can generate a shift register that can cascade down in the same DSP block column. Dedicated connections between DSP blocks provide fast connections between shift register inputs to cascade shift register chains. You can cascade registers within multiple DSP blocks for $9 \times 9$- or $18 \times 18$-bit FIR filters larger than four taps, with additional adder stages implemented in ALMs. If the DSP block is configured as $36 \times 36$ bits, the adder, subtractor, or accumulator stages are implemented in ALMs. Each DSP block can route the shift register chain out of the block to cascade multiple columns of DSP blocks.

The DSP block is divided into four block units that interface with four LAB rows on the left and right. Each block unit can be considered one complete $18 \times 18$-bit multiplier with 36 inputs and 36 outputs. A local interconnect region is associated with each DSP block. Like an LAB, this interconnect region can be fed with 16 direct link interconnects from the LAB to the left or right of the DSP block in the same row. R4 and C4 routing resources can access the DSP block's local interconnect region.

The outputs also work similarly to LAB outputs. Eighteen outputs from the DSP block can drive to the left LAB through direct link interconnects and 18 can drive to the right LAB though direct link interconnects. All 36 outputs can drive to R4 and C4 routing interconnects. Outputs can drive right- or left-column routing.

Figure 2–52 and Figure 2–53 show the DSP block interfaces to LAB rows.

**Figure 2–52.** DSP Block Interconnect Interface

**Figure 2–53.** DSP Block Interface to Interconnect



A bus of 44 control signals feeds the entire DSP block. These signals include clocks, asynchronous clears, clock enables, signed and unsigned control signals, addition and subtraction control signals, rounding and saturation control signals, and accumulator synchronous loads. The clock signals are routed from LAB row clocks and are generated from specific LAB rows at the DSP block interface. The LAB row source for control signals, data inputs, and outputs is shown in Table 2–15.

For more information about DSP blocks, refer to the *DSP Blocks in Arria GX Devices* chapter.

**Table 2–15.** DSP Block Signal Sources and Destinations

| LAB Row at Interface | Control Signals Generated | Data Inputs | Data Outputs |
|---|---|---|---|
| 0 | clock0<br>aclr0<br>ena0<br>mult01_saturate<br>addnsub1_round/<br>accum_round<br>addnsub1<br>signa<br>sourcea<br>sourceb | A1[17..0]<br>B1[17..0] | OA[17..0]<br>OB[17..0] |
| 1 | clock1<br>aclr1<br>ena1<br>accum_saturate<br>mult01_round<br>accum_sload<br>sourcea<br>sourceb<br>mode0 | A2[17..0]<br>B2[17..0] | OC[17..0]<br>OD[17..0] |
| 2 | clock2<br>aclr2<br>ena2<br>mult23_saturate<br>addnsub3_round/<br>accum_round<br>addnsub3<br>sign_b<br>sourcea<br>sourceb | A3[17..0]<br>B3[17..0] | OE[17..0]<br>OF[17..0] |
| 3 | clock3<br>aclr3<br>ena3<br>accum_saturate<br>mult23_round<br>accum_sload<br>sourcea<br>sourceb<br>mode1 | A4[17..0]<br>B4[17..0] | OG[17..0]<br>OH[17..0] |

# PLLs and Clock Networks

Arria GX devices provide a hierarchical clock structure and multiple PLLs with advanced features. The large number of clocking resources in combination with the clock synthesis precision provided by enhanced and fast PLLs provides a complete clock management solution.

## Global and Hierarchical Clocking

Arria GX devices provide 16 dedicated global clock networks and 32 regional clock networks (eight per device quadrant). These clocks are organized into a hierarchical clock structure that allows for up to 24 clocks per device region with low skew and delay. This hierarchical clocking scheme provides up to 48 unique clock domains in Arria GX devices.

There are 12 dedicated clock pins (CLK[15..12] and CLK[7..0]) to drive either the global or regional clock networks. Four clock pins drive each side of the device except the right side, as shown in Figure 2–54 and Figure 2–55. Internal logic and enhanced and fast PLL outputs can also drive the global and regional clock networks. Each global and regional clock has a clock control block, which controls the selection of the clock source and dynamically enables or disables the clock to reduce power consumption. Table 2–16 lists the global and regional clock features.

**Table 2–16.** Global and Regional Clock Features

| Feature | Global Clocks | Regional Clocks |
|---|---|---|
| Number per device | 16 | 32 |
| Number available per quadrant | 16 | 8 |
| Sources | Clock pins, PLL outputs, core routings, inter-transceiver clocks | Clock pins, PLL outputs, core routings, inter-transceiver clocks |
| Dynamic clock source selection | ✓ | — |
| Dynamic enable/disable | ✓ | ✓ |

### Global Clock Network

These clocks drive throughout the entire device, feeding all device quadrants. GCLK networks can be used as clock sources for all resources in the device IOEs, ALMs, DSP blocks, and all memory blocks. These resources can also be used for control signals, such as clock enables and synchronous or asynchronous clears fed from the external pin. The global clock networks can also be driven by internal logic for internally generated global clocks and asynchronous clears, clock enables, or other control signals with large fanout. Figure 2–54 shows the 12 dedicated CLK pins driving global clock networks.

**Figure 2–54.** Global Clocking
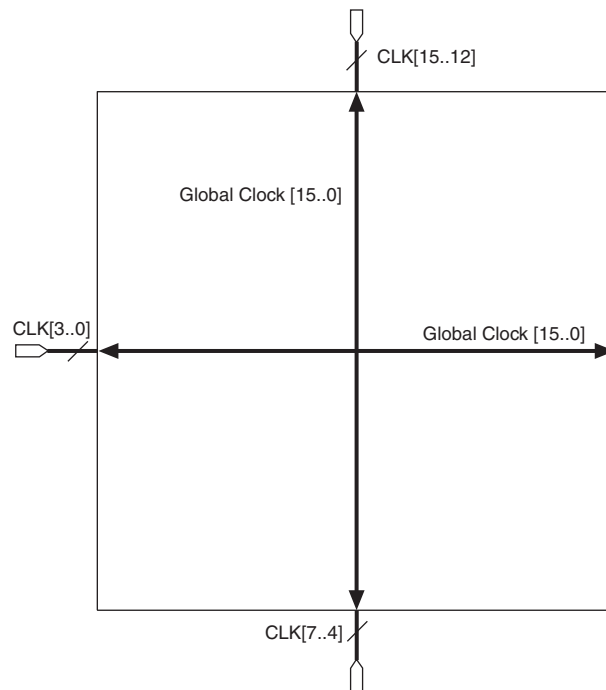


## Regional Clock Network

There are eight RCLK networks (RCLK[7..0]) in each quadrant of the Arria GX device that are driven by the dedicated CLK[15..12] and CLK[7..0] input pins, by PLL outputs, or by internal logic. The regional clock networks provide the lowest clock delay and skew for logic contained in a single quadrant. The CLK pins symmetrically drive the RCLK networks in a particular quadrant, as shown in Figure 2–55.

**Figure 2–55.** Regional Clocks
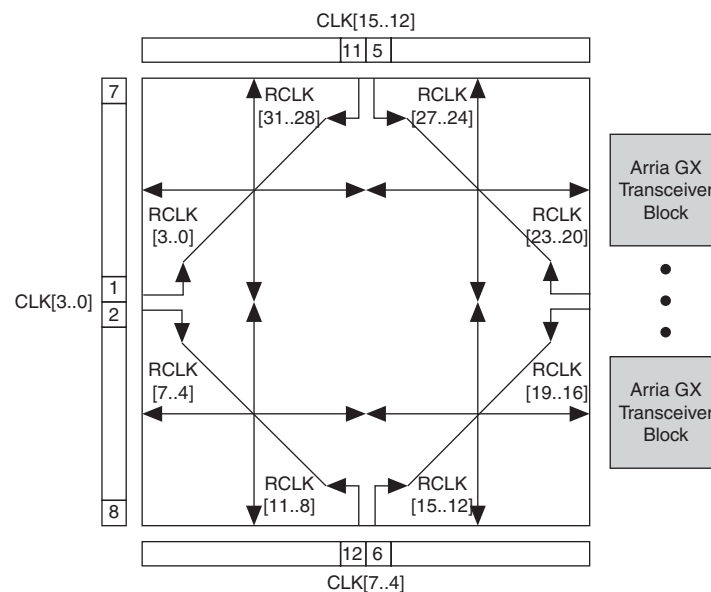


## Dual-Regional Clock Network

A single source (CLK pin or PLL output) can generate a dual-RCLK by driving two RCLK network lines in adjacent quadrants (one from each quadrant), which allows logic that spans multiple quadrants to use the same low skew clock. The routing of this clock signal on an entire side has approximately the same speed but slightly higher clock skew when compared with a clock signal that drives a single quadrant. Internal logic-array routing can also drive a dual-regional clock. Clock pins and enhanced PLL outputs on the top and bottom can drive horizontal dual-regional clocks. Clock pins and fast PLL outputs on the left and right can drive vertical dual-regional clocks, as shown in Figure 2–56. Corner PLLs cannot drive dual-regional clocks.

**Figure 2–56.** Dual-Regional Clocks
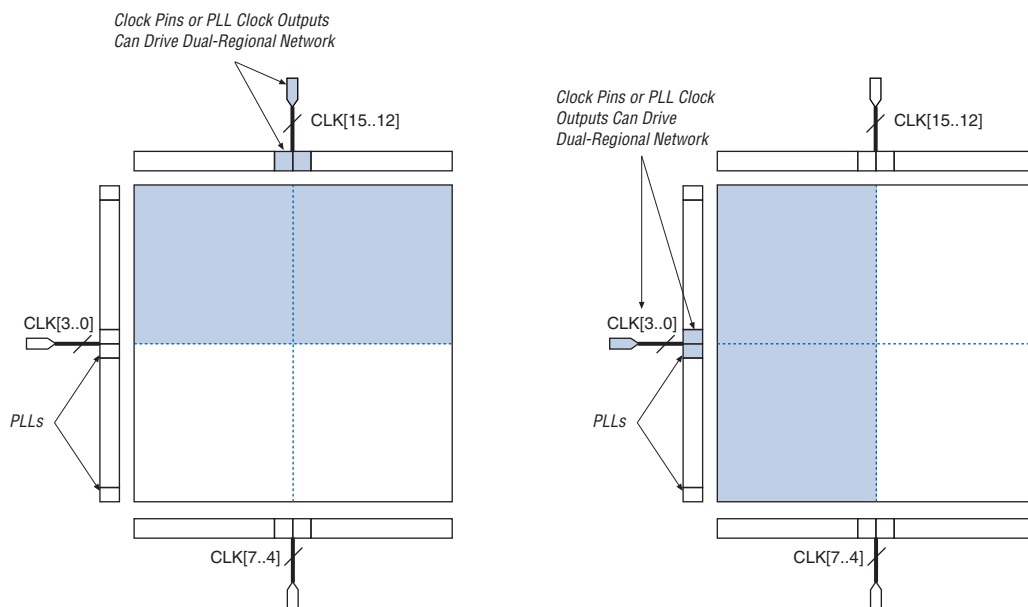


## Combined Resources

Within each quadrant, there are 24 distinct dedicated clocking resources consisting of 16 global clock lines and eight regional clock lines. Multiplexers are used with these clocks to form buses to drive LAB row clocks, column IOE clocks, or row IOE clocks. Another multiplexer is used at the LAB level to select three of the six row clocks to feed the ALM registers in the LAB (refer to Figure 2–57).

**Figure 2–57.** Hierarchical Clock Networks Per Quadrant
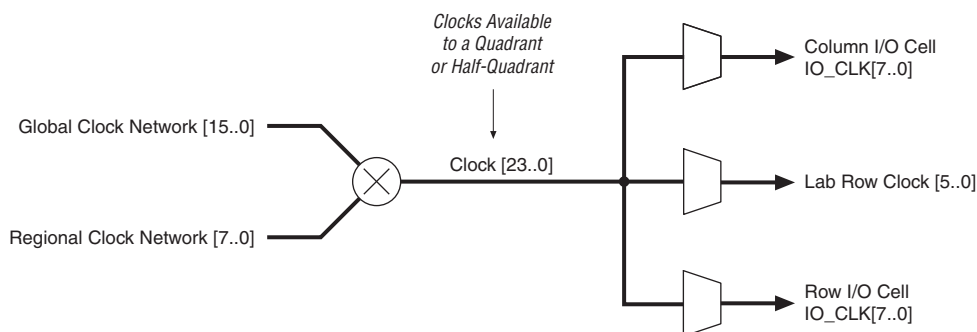


You can use the Quartus II software to control whether a clock input pin drives either a GCLK, RCLK, or dual-RCLK network. The Quartus II software automatically selects the clocking resources if not specified.

## Clock Control Block

Each GCLK, RCLK, and PLL external clock output has its own clock control block. The control block has two functions:

■ Clock source selection (dynamic selection for global clocks)

■ Clock power-down (dynamic clock enable or disable)

Figure 2–58 through Figure 2–60 show the clock control block for the global clock, regional clock, and PLL external clock output, respectively.

**Figure 2–58.** Global Clock Control Blocks
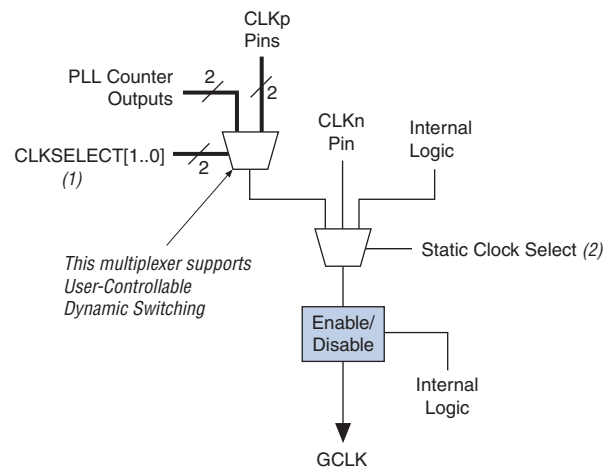


**Notes to Figure 2–58:**

(1) These clock select signals can be dynamically controlled through internal logic when the device is operating in user mode.

(2) These clock select signals can only be set through a configuration file (SRAM Object File [**.sof**] or Programmer Object File [**.pof**]) and cannot be dynamically controlled during user mode operation.

**Figure 2–59.** Regional Clock Control Blocks
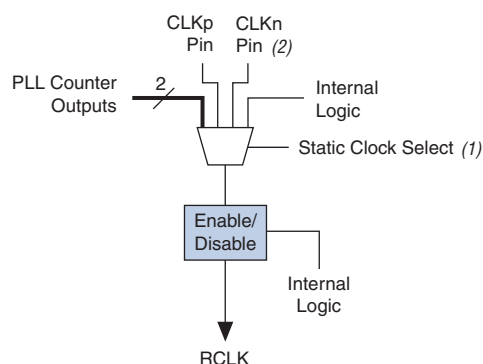


**Notes to Figure 2–59:**

(1) These clock select signals can only be set through a configuration file (**.sof** or **.pof**) and cannot be dynamically controlled during user mode operation.

(2) Only the CLKn pins on the top and bottom of the device feed to regional clock select.

**Figure 2–60.** External PLL Output Clock Control Blocks
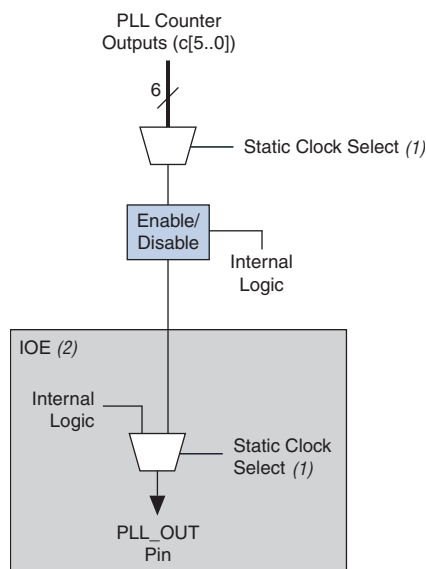


**Notes to Figure 2–60:**

(1) These clock select signals can only be set through a configuration file (**.sof** or **.pof**) and cannot be dynamically controlled during user mode operation.

(2) The clock control block feeds to a multiplexer within the PLL_OUT pin's IOE. The PLL_OUT pin is a dual-purpose pin. Therefore, this multiplexer selects either an internal signal or the output of the clock control block.

For the global clock control block, clock source selection can be controlled either statically or dynamically. You have the option of statically selecting the clock source by using the Quartus II software to set specific configuration bits in the configuration file (**.sof** or **.pof**) or controlling the selection dynamically by using internal logic to drive the multiplexer select inputs. When selecting statically, the clock source can be set to any of the inputs to the select multiplexer. When selecting the clock source dynamically, you can either select between two PLL outputs (such as the C0 or C1 outputs from one PLL), between two PLLs (such as the C0/C1 clock output of one PLL or the C0/C1 clock output of the other PLL), between two clock pins (such as CLK0 or CLK1), or between a combination of clock pins or PLL outputs.

For the regional and PLL_OUT clock control block, clock source selection can only be controlled statically using configuration bits. Any of the inputs to the clock select multiplexer can be set as the clock source.

Arria GX clock networks can be disabled (powered down) by both static and dynamic approaches. When a clock net is powered down, all logic fed by the clock net is in an off-state thereby reducing the overall power consumption of the device. GCLK and RCLK networks can be powered down statically through a setting in the configuration file (**.sof** or **.pof**). Clock networks that are not used are automatically powered down through configuration bit settings in the configuration file generated by the Quartus II software. The dynamic clock enable or disable feature allows the internal logic to control power up/down synchronously on GCLK and RCLK nets and PLL_OUT pins. This function is independent of the PLL and is applied directly on the clock network or PLL_OUT pin, as shown in Figure 2–58 through Figure 2–60.

## Enhanced and Fast PLLs

Arria GX devices provide robust clock management and synthesis using up to four enhanced PLLs and four fast PLLs. These PLLs increase performance and provide advanced clock interfacing and clock frequency synthesis. With features such as clock switchover, spread spectrum clocking, reconfigurable bandwidth, phase control, and reconfigurable phase shifting, the Arria GX device's enhanced PLLs provide you with complete control of your clocks and system timing. The fast PLLs provide general purpose clocking with multiplication and phase shifting as well as high-speed outputs for high-speed differential I/O support. Enhanced and fast PLLs work together with the Arria GX high-speed I/O and advanced clock architecture to provide significant improvements in system performance and bandwidth.

The Quartus II software enables the PLLs and their features without requiring any external devices. Table 2–17 lists the PLLs available for each Arria GX device and their type.

**Table 2–17.** Arria GX Device PLL Availability   *(Note 1)*, *(2)*

| Device | Fast PLLs | | | | | | | | Enhanced PLLs | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** *(3)* | **4** *(3)* | **7** | **8** | **9** *(3)* | **10** *(3)* | **5** | **6** | **11** | **12** |
| EP1AGX20 | ✓ | ✓ | — | — | — | — | — | — | ✓ | ✓ | — | — |
| EP1AGX35 | ✓ | ✓ | — | — | — | — | — | — | ✓ | ✓ | — | — |
| EP1AGX50 *(4)* | ✓ | ✓ | — | — | ✓ | ✓ | — | — | ✓ | ✓ | ✓ | ✓ |
| EP1AGX60 *(5)* | ✓ | ✓ | — | — | ✓ | ✓ | — | — | ✓ | ✓ | ✓ | ✓ |
| EP1AGX90 | ✓ | ✓ | — | — | ✓ | ✓ | — | — | ✓ | ✓ | ✓ | ✓ |

**Notes to Table 2–17:**

(1) The global or regional clocks in a fast PLL's transceiver block can drive the fast PLL input. A pin or other PLL must drive the global or regional source. The source cannot be driven by internally generated logic before driving the fast PLL.

(2) EP1AGX20C, EP1AGX35C/D, EP1AGX50C and EP1AGX60C/D devices only have two fast PLLs (PLLs 1 and 2), but the connectivity from these two PLLs to the global and regional clock networks remains the same as shown in this table.

(3) PLLs 3, 4, 9, and 10 are not available in Arria GX devices.

(4) 4 or 8 PLLs are available depending on C or D device and the package option.

(5) 4or 8 PLLs are available depending on C, D, or E device option.

Table 2–18 lists the enhanced PLL and fast PLL features in Arria GX devices.

**Table 2–18.** Arria GX PLL Features   (Part 1 of 2)

| Feature | Enhanced PLL | Fast PLL |
|---|---|---|
| Clock multiplication and division | $m/(n \times$ post-scale counter) *(1)* | $m/(n \times$ post-scale counter) *(2)* |
| Phase shift | Down to 125-ps increments *(3)*, *(4)* | Down to 125-ps increments *(3)*, *(4)* |
| Clock switchover | ✓ | ✓ *(5)* |
| PLL reconfiguration | ✓ | ✓ |
| Reconfigurable bandwidth | ✓ | ✓ |
| Spread spectrum clocking | ✓ | — |
| Programmable duty cycle | ✓ | ✓ |
| Number of internal clock outputs | 6 | 4 |
| Number of external clock outputs | Three differential/six single-ended | *(6)* |

**Table 2–18.** Arria GX PLL Features (Part 2 of 2)

| Feature | Enhanced PLL | Fast PLL |
|---|---|---|
| Number of feedback clock inputs | One single-ended or differential *(7)*, *(8)* | — |

**Notes to Table 2–18:**

(1) For enhanced PLLs, *m*, *n* range from 1 to 256 and post-scale counters range from 1 to 512 with 50% duty cycle.

(2) For fast PLLs, *m*, and post-scale counters range from 1 to 32. The *n* counter ranges from 1 to 4.

(3) The smallest phase shift is determined by the voltage controlled oscillator ($V_{CO}$) period divided by 8.

(4) For degree increments, Arria GX devices can shift all output frequencies in increments of at least 45. Smaller degree increments are possible depending on the frequency and divide parameters.

(5) Arria GX fast PLLs only support manual clock switchover.

(6) Fast PLLs can drive to any I/O pin as an external clock. For high-speed differential I/O pins, the device uses a data channel to generate `txclkout`.

(7) If the feedback input is used, you lose one (or two, if $f_{BIN}$ is differential) external clock output pin.

(8) Every Arria GX device has at least two enhanced PLLs with one single-ended or differential external feedback input per PLL.

Figure 2–61 shows a top-level diagram of the Arria GX device and PLL floorplan.

**Figure 2–61.** PLL Locations
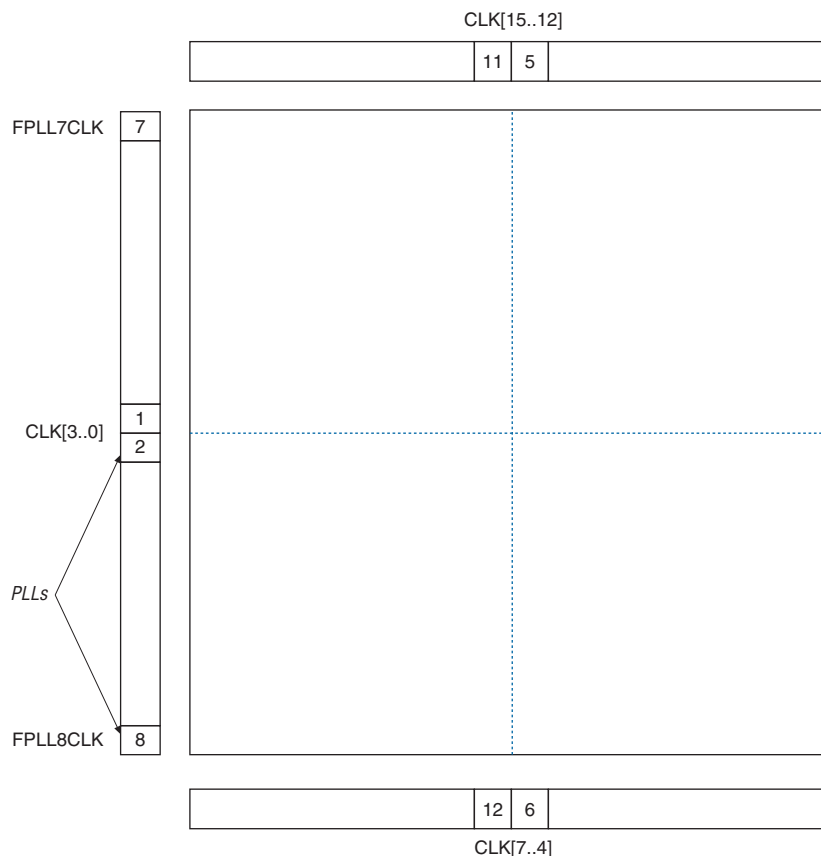


Figure 2–62 and Figure 2–63 shows global and regional clocking from the fast PLL outputs and side clock pins. The connections to the global and regional clocks from the fast PLL outputs, internal drivers, and `CLK` pins on the left side of the device are shown in Table 2–19.

**Figure 2–62.** Global and Regional Clock Connections from Center Clock Pins and Fast PLL Outputs  *(Note 1)*
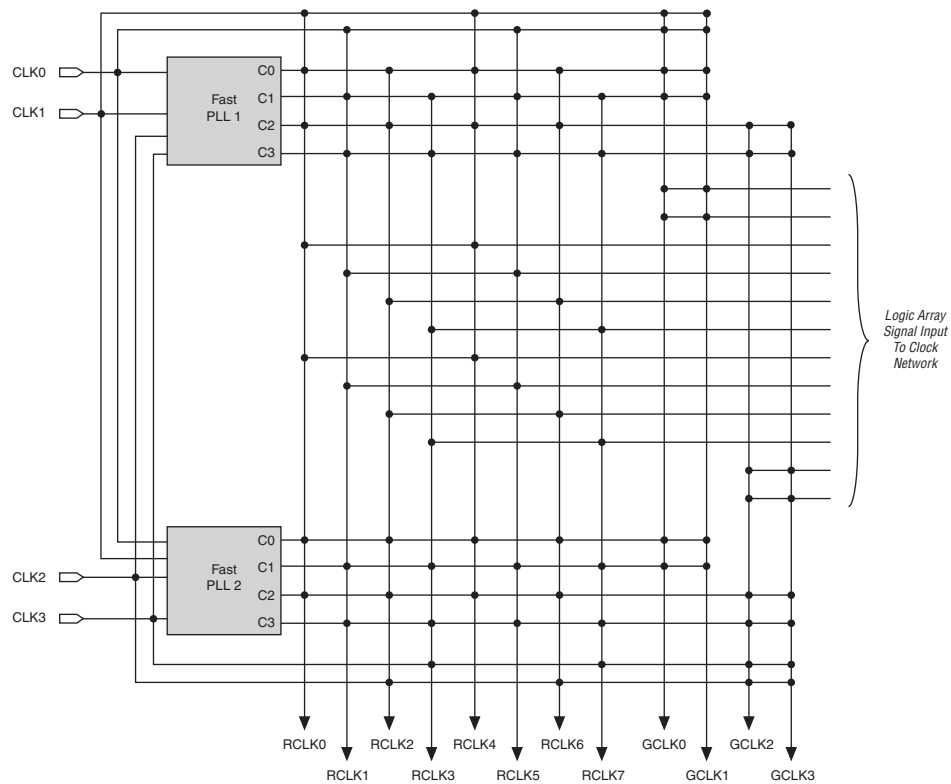


**Note to Figure 2–62:**

(1)  The global or regional clocks in a fast PLL's quadrant can drive the fast PLL input. A dedicated clock input pin or other PLL must drive the global or regional source. The source cannot be driven by internally generated logic before driving the fast PLL.

**Figure 2–63.** Global and Regional Clock Connections from Corner Clock Pins and Fast PLL Outputs *(Note 1)*
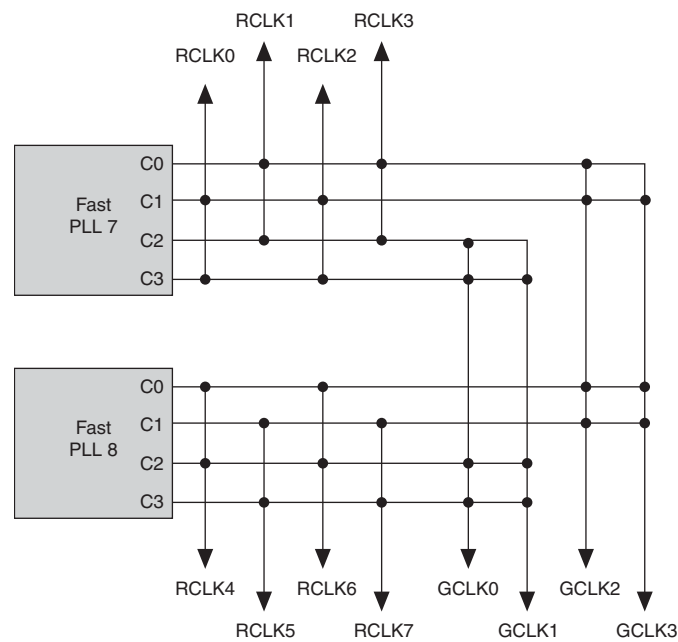


**Note to Figure 2–63:**

(1) The GCLK or RCLK in a fast PLL's quadrant can drive the fast PLL input. A dedicated clock input pin or other PLL must drive the global or regional source. The source cannot be driven by internally generated logic before driving the fast PLL.

**Table 2–19.** Global and Regional Clock Connections from Left Side Clock Pins and Fast PLL Outputs  (Part 1 of 2)

| Left Side Global & Regional Clock Network Connectivity | CLK0 | CLK1 | CLK2 | CLK3 | RCLK0 | RCLK1 | RCLK2 | RCLK3 | RCLK4 | RCLK5 | RCLK6 | RCLK7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Clock Pins** | | | | | | | | | | | | |
| CLK0p | ✔ | ✔ | — | — | ✔ | — | — | — | ✔ | — | — | — |
| CLK1p | ✔ | ✔ | — | — | — | ✔ | — | — | — | ✔ | — | — |
| CLK2p | — | — | ✔ | ✔ | — | — | ✔ | — | — | — | ✔ | — |
| CLK3p | — | — | ✔ | ✔ | — | — | — | ✔ | — | — | — | ✔ |
| **Drivers from Internal Logic** | | | | | | | | | | | | |
| GCLKDRV0 | ✔ | ✔ | — | — | — | — | — | — | — | — | — | — |
| GCLKDRV1 | ✔ | ✔ | — | — | — | — | — | — | — | — | — | — |
| GCLKDRV2 | — | — | ✔ | ✔ | — | — | — | — | — | — | — | — |
| GCLKDRV3 | — | — | ✔ | ✔ | — | — | — | — | — | — | — | — |
| RCLKDRV0 | — | — | — | — | ✔ | — | — | — | ✔ | — | — | — |
| RCLKDRV1 | — | — | — | — | — | ✔ | — | — | — | ✔ | — | — |
| RCLKDRV2 | — | — | — | — | — | — | ✔ | — | — | — | ✔ | — |
| RCLKDRV3 | — | — | — | — | — | — | — | ✔ | — | — | — | ✔ |
| RCLKDRV4 | — | — | — | — | ✔ | — | — | — | ✔ | — | — | — |
| RCLKDRV5 | — | — | — | — | — | ✔ | — | — | — | ✔ | — | — |
| RCLKDRV6 | — | — | — | — | — | — | ✔ | — | — | — | ✔ | — |

**Table 2–19.** Global and Regional Clock Connections from Left Side Clock Pins and Fast PLL Outputs (Part 2 of 2)

| Left Side Global & Regional Clock Network Connectivity | CLK0 | CLK1 | CLK2 | CLK3 | RCLK0 | RCLK1 | RCLK2 | RCLK3 | RCLK4 | RCLK5 | RCLK6 | RCLK7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RCLKDRV7 | — | — | — | — | — | — | — | ✓ | — | — | — | ✓ |
| **PLL 1 Outputs** | | | | | | | | | | | | |
| c0 | ✓ | ✓ | — | — | ✓ | — | ✓ | — | ✓ | — | ✓ | — |
| c1 | ✓ | ✓ | — | — | — | ✓ | — | ✓ | — | ✓ | — | ✓ |
| c2 | — | — | ✓ | ✓ | ✓ | — | ✓ | — | ✓ | — | ✓ | — |
| c3 | — | — | ✓ | ✓ | — | ✓ | — | ✓ | — | ✓ | — | ✓ |
| **PLL 2 Outputs** | | | | | | | | | | | | |
| c0 | ✓ | ✓ | — | — | — | ✓ | — | ✓ | — | ✓ | — | ✓ |
| c1 | ✓ | ✓ | — | — | ✓ | — | ✓ | — | ✓ | — | ✓ | — |
| c2 | — | — | ✓ | ✓ | — | ✓ | — | ✓ | — | ✓ | — | ✓ |
| c3 | — | — | ✓ | ✓ | ✓ | — | ✓ | — | ✓ | — | ✓ | — |
| **PLL 7 Outputs** | | | | | | | | | | | | |
| c0 | — | — | ✓ | ✓ | — | ✓ | — | ✓ | — | — | — | — |
| c1 | — | — | ✓ | ✓ | ✓ | — | ✓ | — | — | — | — | — |
| c2 | ✓ | ✓ | — | — | — | ✓ | — | ✓ | — | — | — | — |
| c3 | ✓ | ✓ | — | — | ✓ | — | ✓ | — | — | — | — | — |
| **PLL 8 Outputs** | | | | | | | | | | | | |
| c0 | — | — | ✓ | ✓ | — | — | — | ✓ | — | ✓ | — | — |
| c1 | — | — | ✓ | ✓ | — | — | — | — | — | ✓ | — | ✓ |
| c2 | ✓ | ✓ | — | — | — | — | — | ✓ | — | ✓ | — | — |
| c3 | ✓ | ✓ | — | — | — | — | — | — | — | ✓ | — | ✓ |

Figure 2–64 shows the global and regional clocking from enhanced PLL outputs and top and bottom CLK pins.

**Figure 2–64.** Global and Regional Clock Connections from Top and Bottom Clock Pins and Enhanced PLL Outputs *(Note 1)*
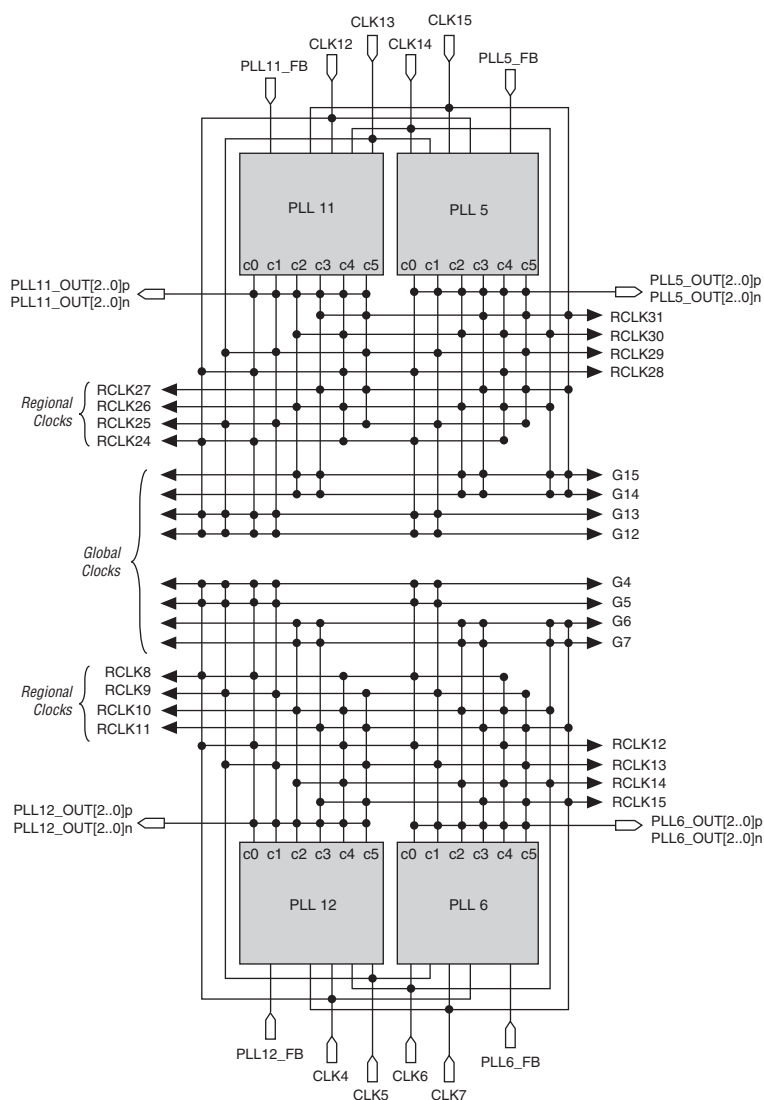


**Note to Figure 2–64:**

(1)  If the design uses the feedback input, you might lose one (or two if FBIN is differential) external clock output pin.

The connections to the global and regional clocks from the top clock pins and enhanced PLL outputs are shown in Table 2–20. The connections to the clocks from the bottom clock pins are shown in Table 2–21.

**Table 2–20.** Global and Regional Clock Connections from Top Clock Pins and Enhanced PLL Outputs

| Top Side Global and Regional Clock Network Connectivity | DLLCLK | CLK12 | CLK13 | CLK14 | CLK15 | RCLK24 | RCLK25 | RCLK26 | RCLK27 | RCLK28 | RCLK29 | RCLK30 | RCLK31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Clock pins** | | | | | | | | | | | | | |
| CLK12p | ✓ | ✓ | ✓ | — | — | ✓ | — | — | — | ✓ | — | — | — |
| CLK13p | ✓ | ✓ | ✓ | — | — | — | ✓ | — | — | — | ✓ | — | — |
| CLK14p | ✓ | — | — | ✓ | ✓ | — | — | ✓ | — | — | — | ✓ | — |
| CLK15p | ✓ | — | — | ✓ | ✓ | — | — | — | ✓ | — | — | — | ✓ |
| CLK12n | — | ✓ | — | — | — | ✓ | — | — | — | ✓ | — | — | — |
| CLK13n | — | — | ✓ | — | — | — | ✓ | — | — | — | ✓ | — | — |
| CLK14n | — | — | — | ✓ | — | — | — | ✓ | — | — | — | ✓ | — |
| CLK15n | — | — | — | — | ✓ | — | — | — | ✓ | — | — | — | ✓ |
| **Drivers from internal logic** | | | | | | | | | | | | | |
| GCLKDRV0 | — | ✓ | — | — | — | — | — | — | — | — | — | — | — |
| GCLKDRV1 | — | — | ✓ | — | — | — | — | — | — | — | — | — | — |
| GCLKDRV2 | — | — | — | ✓ | — | — | — | — | — | — | — | — | — |
| GCLKDRV3 | — | — | — | — | ✓ | — | — | — | — | — | — | — | — |
| RCLKDRV0 | — | — | — | — | — | ✓ | — | — | — | ✓ | — | — | — |
| RCLKDRV1 | — | — | — | — | — | — | ✓ | — | — | — | ✓ | — | — |
| RCLKDRV2 | — | — | — | — | — | — | — | ✓ | — | — | — | ✓ | — |
| RCLKDRV3 | — | — | — | — | — | — | — | — | ✓ | — | — | — | ✓ |
| RCLKDRV4 | — | — | — | — | — | ✓ | — | — | — | ✓ | — | — | — |
| RCLKDRV5 | — | — | — | — | — | — | ✓ | — | — | — | ✓ | — | — |
| RCLKDRV6 | — | — | — | — | — | — | — | ✓ | — | — | — | ✓ | — |
| RCLKDRV7 | — | — | — | — | — | — | — | — | ✓ | — | — | — | ✓ |
| **Enhanced PLL5 outputs** | | | | | | | | | | | | | |
| c0 | ✓ | ✓ | ✓ | — | — | ✓ | — | — | — | ✓ | — | — | — |
| c1 | ✓ | ✓ | ✓ | — | — | — | ✓ | — | — | — | ✓ | — | — |
| c2 | ✓ | — | — | ✓ | ✓ | — | — | ✓ | — | — | — | ✓ | — |
| c3 | ✓ | — | — | ✓ | ✓ | — | — | — | ✓ | — | — | — | ✓ |
| c4 | ✓ | — | — | — | — | ✓ | — | ✓ | — | ✓ | — | ✓ | — |
| c5 | ✓ | — | — | — | — | — | ✓ | — | ✓ | — | ✓ | — | ✓ |
| **Enhanced PLL 11 outputs** | | | | | | | | | | | | | |
| c0 | — | ✓ | ✓ | — | — | ✓ | — | — | — | ✓ | — | — | — |
| c1 | — | ✓ | ✓ | — | — | — | ✓ | — | — | — | ✓ | — | — |
| c2 | — | — | — | ✓ | ✓ | — | — | ✓ | — | — | — | ✓ | — |
| c3 | — | — | — | ✓ | ✓ | — | — | — | ✓ | — | — | — | ✓ |
| c4 | — | — | — | — | — | ✓ | — | ✓ | — | ✓ | — | ✓ | — |
| c5 | — | — | — | — | — | — | ✓ | — | ✓ | — | ✓ | — | ✓ |

**Table 2–21.** Global and Regional Clock Connections from Bottom Clock Pins and Enhanced PLL Outputs

| Bottom Side Global and Regional Clock Network Connectivity | DLLCLK | CLK4 | CLK5 | CLK6 | CLK7 | RCLK8 | RCLK9 | RCLK10 | RCLK11 | RCLK12 | RCLK13 | RCLK14 | RCLK15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Clock pins** | | | | | | | | | | | | | |
| CLK4p | ✓ | ✓ | ✓ | — | — | ✓ | — | — | — | ✓ | — | — | — |
| CLK5p | ✓ | ✓ | ✓ | — | — | — | ✓ | — | — | — | ✓ | — | — |
| CLK6p | ✓ | — | — | ✓ | ✓ | — | — | ✓ | — | — | — | ✓ | — |
| CLK7p | ✓ | — | — | ✓ | ✓ | — | — | — | ✓ | — | — | — | ✓ |
| CLK4n | — | ✓ | — | — | — | ✓ | — | — | — | ✓ | — | — | — |
| CLK5n | — | — | ✓ | — | — | — | ✓ | — | — | — | ✓ | — | — |
| CLK6n | — | — | — | ✓ | — | — | — | ✓ | — | — | — | ✓ | — |
| CLK7n | — | — | — | — | ✓ | — | — | — | ✓ | — | — | — | ✓ |
| **Drivers from internal logic** | | | | | | | | | | | | | |
| GCLKDRV0 | — | ✓ | — | — | — | — | — | — | — | — | — | — | — |
| GCLKDRV1 | — | — | ✓ | — | — | — | — | — | — | — | — | — | — |
| GCLKDRV2 | — | — | — | ✓ | — | — | — | — | — | — | — | — | — |
| GCLKDRV3 | — | — | — | — | ✓ | — | — | — | — | — | — | — | — |
| RCLKDRV0 | — | — | — | — | — | ✓ | — | — | — | ✓ | — | — | — |
| RCLKDRV1 | — | — | — | — | — | — | ✓ | — | — | — | ✓ | — | — |
| RCLKDRV2 | — | — | — | — | — | — | — | ✓ | — | — | — | ✓ | — |
| RCLKDRV3 | — | — | — | — | — | — | — | — | ✓ | — | — | — | ✓ |
| RCLKDRV4 | — | — | — | — | — | ✓ | — | — | — | ✓ | — | — | — |
| RCLKDRV5 | — | — | — | — | — | — | ✓ | — | — | — | ✓ | — | — |
| RCLKDRV6 | — | — | — | — | — | — | — | ✓ | — | — | — | ✓ | — |
| RCLKDRV7 | — | — | — | — | — | — | — | — | ✓ | — | — | — | ✓ |
| **Enhanced PLL 6 outputs** | | | | | | | | | | | | | |
| c0 | ✓ | ✓ | ✓ | — | — | ✓ | — | — | — | ✓ | — | — | — |
| c1 | ✓ | ✓ | ✓ | — | — | — | ✓ | — | — | — | ✓ | — | — |
| c2 | ✓ | — | — | ✓ | ✓ | — | — | ✓ | — | — | — | ✓ | |
| c3 | ✓ | — | — | ✓ | ✓ | — | — | — | ✓ | — | — | — | ✓ |
| c4 | ✓ | — | — | — | — | ✓ | — | ✓ | — | ✓ | — | ✓ | — |
| c5 | ✓ | — | — | — | — | — | ✓ | — | ✓ | — | ✓ | — | ✓ |
| **Enhanced PLL 12 outputs** | | | | | | | | | | | | | |
| c0 | — | ✓ | ✓ | — | — | ✓ | — | — | — | ✓ | — | — | — |
| c1 | — | ✓ | ✓ | — | — | — | ✓ | — | — | — | ✓ | — | — |
| c2 | — | — | — | ✓ | ✓ | — | — | ✓ | — | — | — | ✓ | — |
| c3 | — | — | — | ✓ | ✓ | — | — | — | ✓ | — | — | — | ✓ |
| c4 | — | — | — | — | — | ✓ | — | ✓ | — | ✓ | — | ✓ | — |
| c5 | — | — | — | — | — | — | ✓ | — | ✓ | — | ✓ | — | ✓ |

## Enhanced PLLs

Arria GX devices contain up to four enhanced PLLs with advanced clock management features. These features include support for external clock feedback mode, spread-spectrum clocking, and counter cascading. Figure 2–65 shows a diagram of the enhanced PLL.

**Figure 2–65.** Arria GX Enhanced PLL    *(Note 1)*



**Notes to Figure 2–65:**

(1) Each clock source can come from any of the four clock pins that are physically located on the same side of the device as the PLL.

(2) If the feedback input is used, you will lose one (or two, if FBIN is differential) external clock output pin.

(3) Each enhanced PLL has three differential external clock outputs or six single-ended external clock outputs.

(4) The global or regional clock input can be driven by an output from another PLL, a pin-driven dedicated global or regional clock, or through a clock control block provided the clock control block is fed by an output from another PLL or a pin-driven dedicated global or regional clock. An internally generated global signal cannot drive the PLL.

## Fast PLLs

Arria GX devices contain up to four fast PLLs with high-speed serial interfacing ability. Fast PLLs offer high-speed outputs to manage the high-speed differential I/O interfaces. Figure 2–66 shows a diagram of the fast PLL.

**Figure 2–66.** Arria GX Device Fast PLL
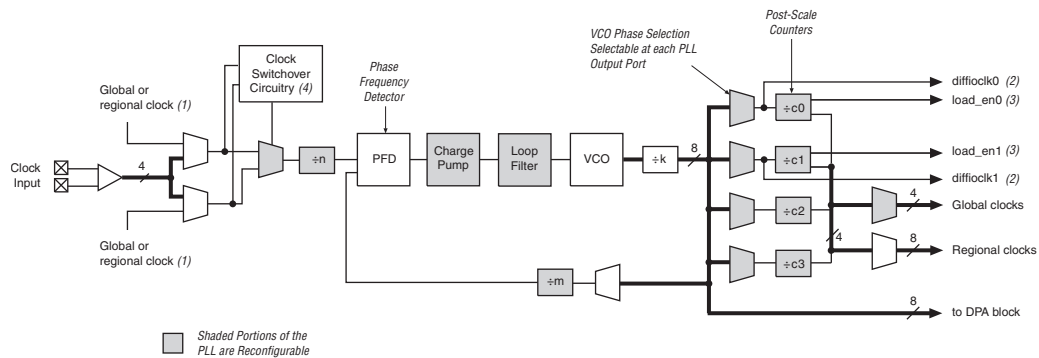


**Notes to Figure 2–66:**

(1) The global or regional clock input can be driven by an output from another PLL, a pin-driven dedicated global or regional clock, or through a clock control block provided the clock control block is fed by an output from another PLL or a pin-driven dedicated global or regional clock. An internally generated global signal cannot drive the PLL.

(2) In high-speed differential I/O support mode, this high-speed PLL clock feeds the serializer/deserializer (SERDES) circuitry. Arria GX devices only support one rate of data transfer per fast PLL in high-speed differential I/O support mode.

(3) This signal is a differential I/O SERDES control signal.

(4) Arria GX fast PLLs only support manual clock switchover.

For more information about enhanced and fast PLLs, refer to the *PLLs in Arria GX Devices* chapter. For more information about high-speed differential I/O support, refer to "High-Speed Differential I/O with DPA Support" on page 2–99.

# I/O Structure

Arria GX IOEs provide many features, including:

■ Dedicated differential and single-ended I/O buffers

■ 3.3-V, 64-bit, 66-MHz PCI compliance

■ 3.3-V, 64-bit, 133-MHz PCI-X 1.0 compliance

■ JTAG boundary-scan test (BST) support

■ On-chip driver series termination

■ OCT for differential standards

■ Programmable pull-up during configuration

■ Output drive strength control

■ Tri-state buffers

■ Bus-hold circuitry

■ Programmable pull-up resistors

■ Programmable input and output delays

■ Open-drain outputs

■ DQ and DQS I/O pins

■ DDR registers

The IOE in Arria GX devices contains a bidirectional I/O buffer, six registers, and a latch for a complete embedded bidirectional single data rate or DDR transfer. Figure 2–67 shows the Arria GX IOE structure. The IOE contains two input registers (plus a latch), two output registers, and two output enable registers. The design can use both input registers and the latch to capture DDR input and both output registers to drive DDR outputs. Additionally, the design can use the output enable (OE) register for fast clock-to-output enable timing. The negative edge-clocked OE register is used for DDR SDRAM interfacing. The Quartus II software automatically duplicates a single OE register that controls multiple output or bidirectional pins.

**Figure 2–67.** Arria GX IOE Structure
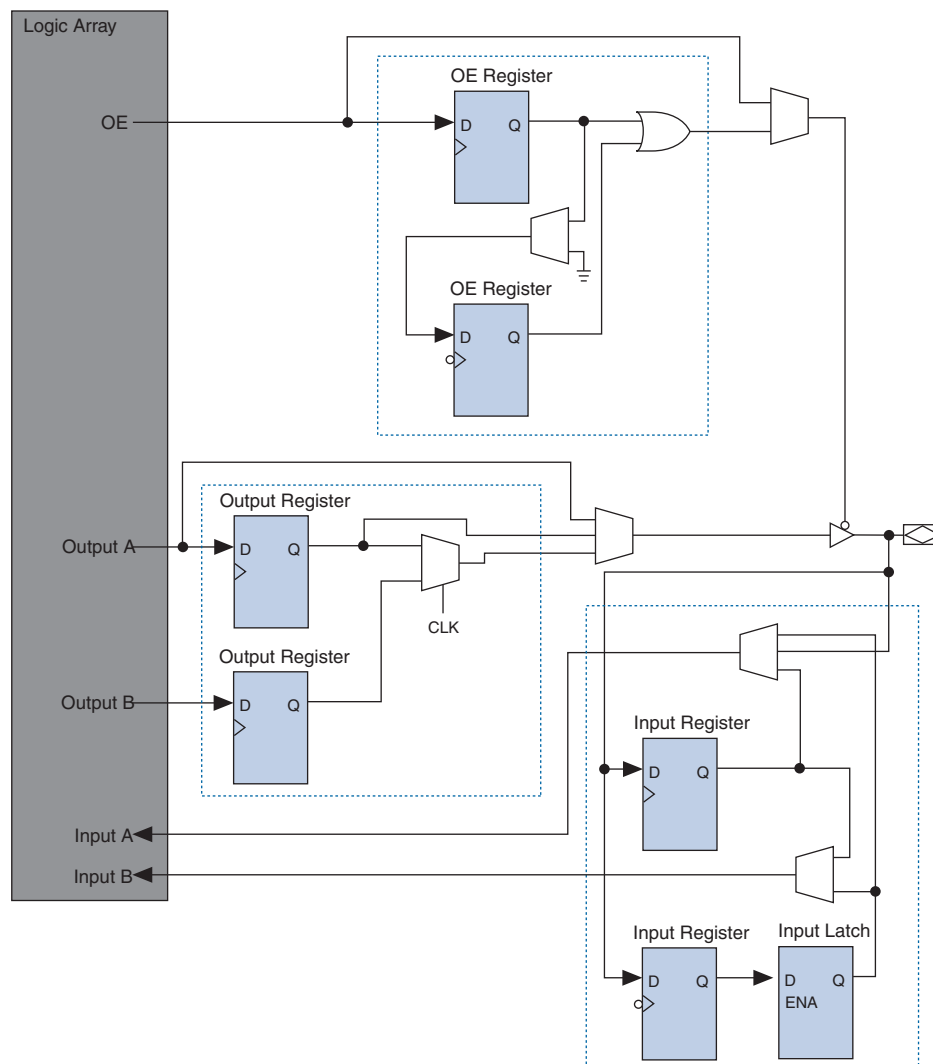


The IOEs are located in I/O blocks around the periphery of the Arria GX device. There are up to four IOEs per row I/O block and four IOEs per column I/O block. Row I/O blocks drive row, column, or direct link interconnects. Column I/O blocks drive column interconnects.

Figure 2–68 shows how a row I/O block connects to the logic array.

**Figure 2–68.** Row I/O Block Connection to the Interconnect
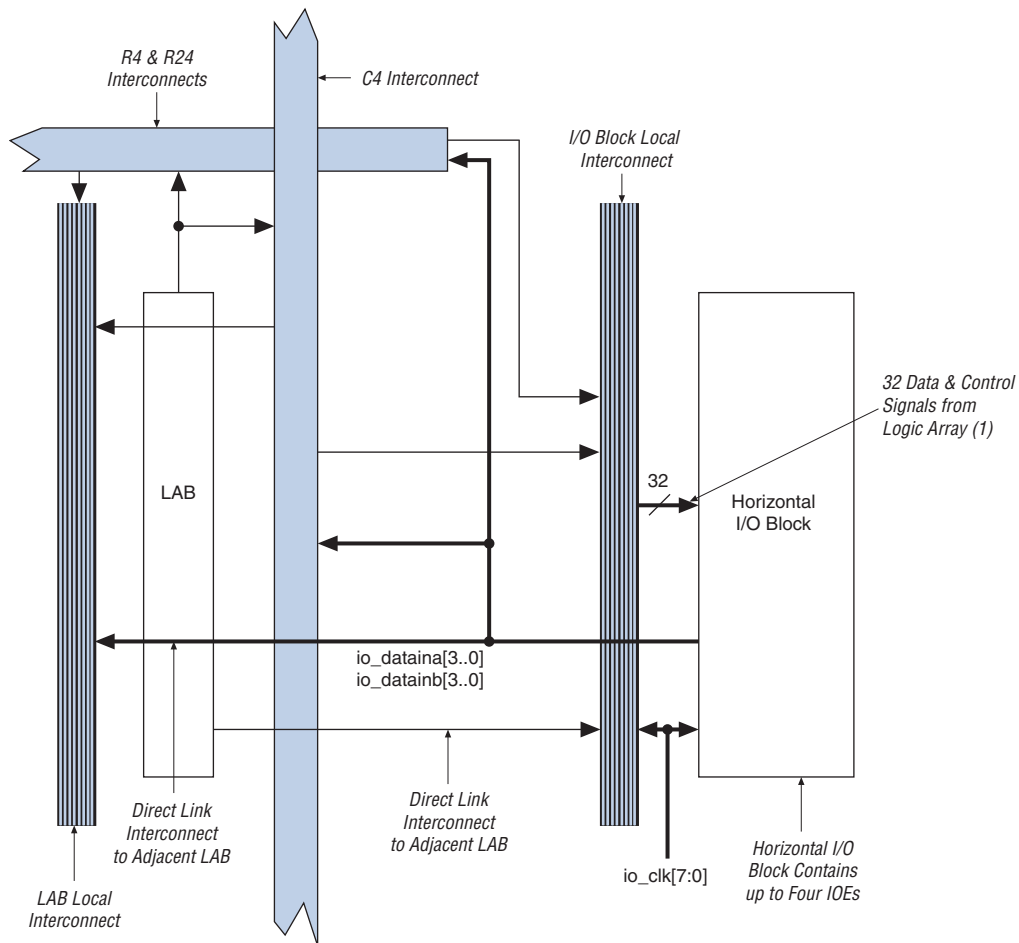


**Note to Figure 2–68:**

(1)  The 32 data and control signals consist of eight data out lines: four lines each for DDR applications `io_dataouta[3..0]` and `io_dataoutb[3..0]`, four output enables `io_oe[3..0]`, four input clock enables `io_ce_in[3..0]`, four output clock enables `io_ce_out[3..0]`, four clocks `io_clk[3..0]`, four asynchronous clear and preset signals `io_aclr/apreset[3..0]`, and four synchronous clear and preset signals `io_sclr/spreset[3..0]`.

Figure 2–69 shows how a column I/O block connects to the logic array.

**Figure 2–69.** Column I/O Block Connection to the Interconnect
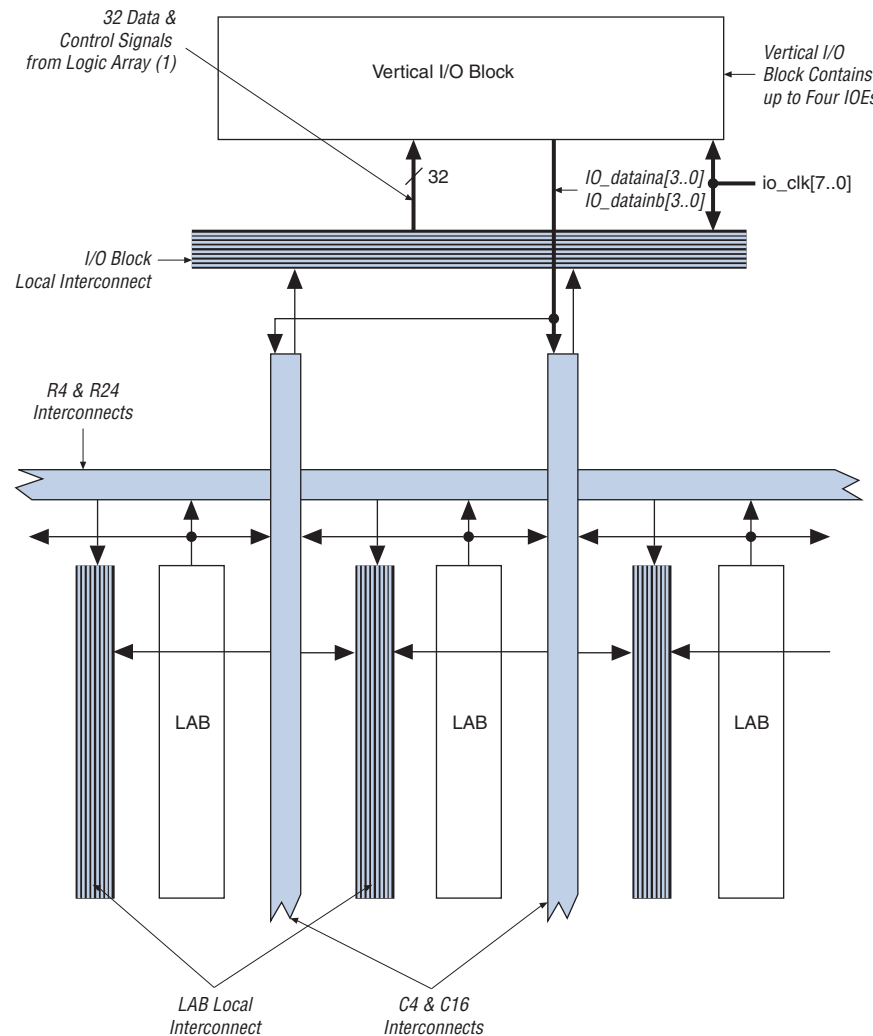


**Note to Figure 2–69:**

(1) The 32 data and control signals consist of eight data out lines: four lines each for DDR applications `io_dataouta[3..0]` and `io_dataoutb[3..0]`, four output enables `io_oe[3..0]`, four input clock enables `io_ce_in[3..0]`, four output clock enables `io_ce_out[3..0]`, four clocks `io_clk[3..0]`, four asynchronous clear and preset signals `io_aclr/apreset[3..0]`, and four synchronous clear and preset signals `io_sclr/spreset[3..0]`.

There are 32 control and data signals that feed each row or column I/O block. These control and data signals are driven from the logic array. The row or column IOE clocks, `io_clk[7..0]`, provide a dedicated routing resource for low-skew, high-speed clocks. I/O clocks are generated from global or regional clocks (refer to "PLLs and Clock Networks" on page 2–66).

Figure 2–70 shows the signal paths through the I/O block.

**Figure 2–70.** Signal Path Through the I/O Block
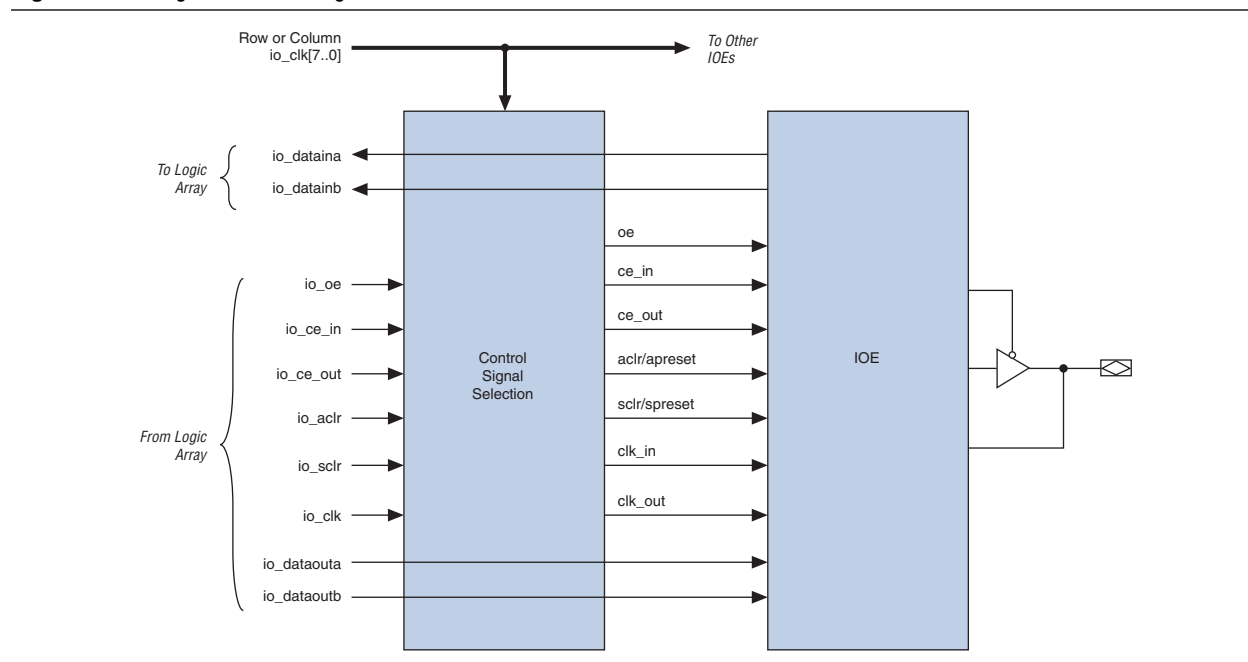


Each IOE contains its own control signal selection for the following control signals: `oe`, `ce_in`, `ce_out`, `aclr/apreset`, `sclr/spreset`, `clk_in`, and `clk_out`. Figure 2–71 shows the control signal selection.

**Figure 2–71.** Control Signal Selection per IOE *(Note 1)*
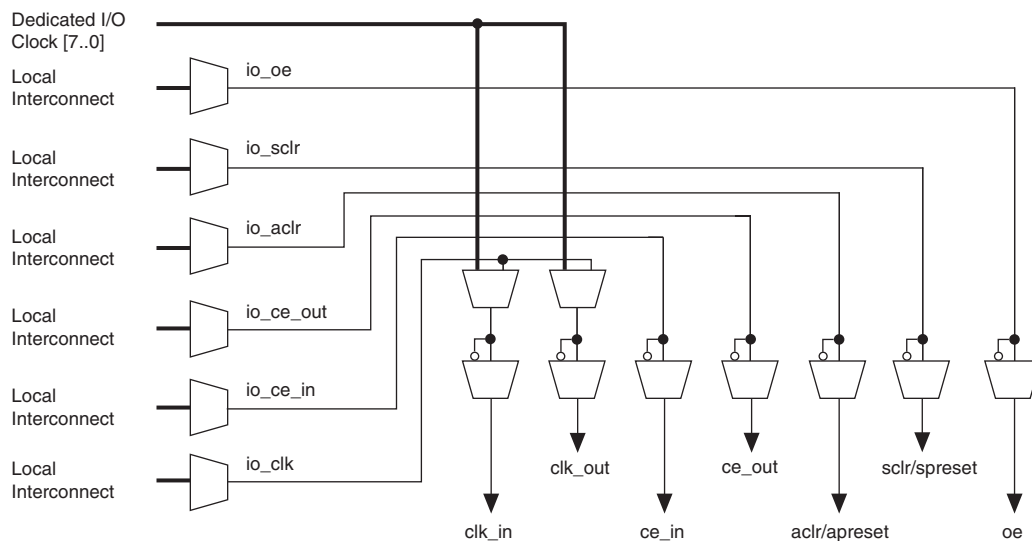


**Notes to Figure 2–71:**

(1) Control signals `ce_in`, `ce_out`, `aclr/apreset`, `sclr/spreset`, and `oe` can be global signals even though their control selection multiplexers are not directly fed by the `ioe_clk[7..0]` signals. The `ioe_clk` signals can drive the I/O local interconnect, which then drives the control selection multiplexers.

In normal bidirectional operation, you can use the input register for input data requiring fast setup times. The input register can have its own clock input and clock enable separate from the OE and output registers. The output register can be used for data requiring fast clock-to-output performance. You can use the OE register for fast clock-to-output enable timing. The OE and output register share the same clock source and the same clock enable source from the local interconnect in the associated LAB, dedicated I/O clocks, and the column and row interconnects. Figure 2–72 shows the IOE in bidirectional configuration.

**Figure 2–72.** Arria GX IOE in Bidirectional I/O Configuration   *(Note 1)*
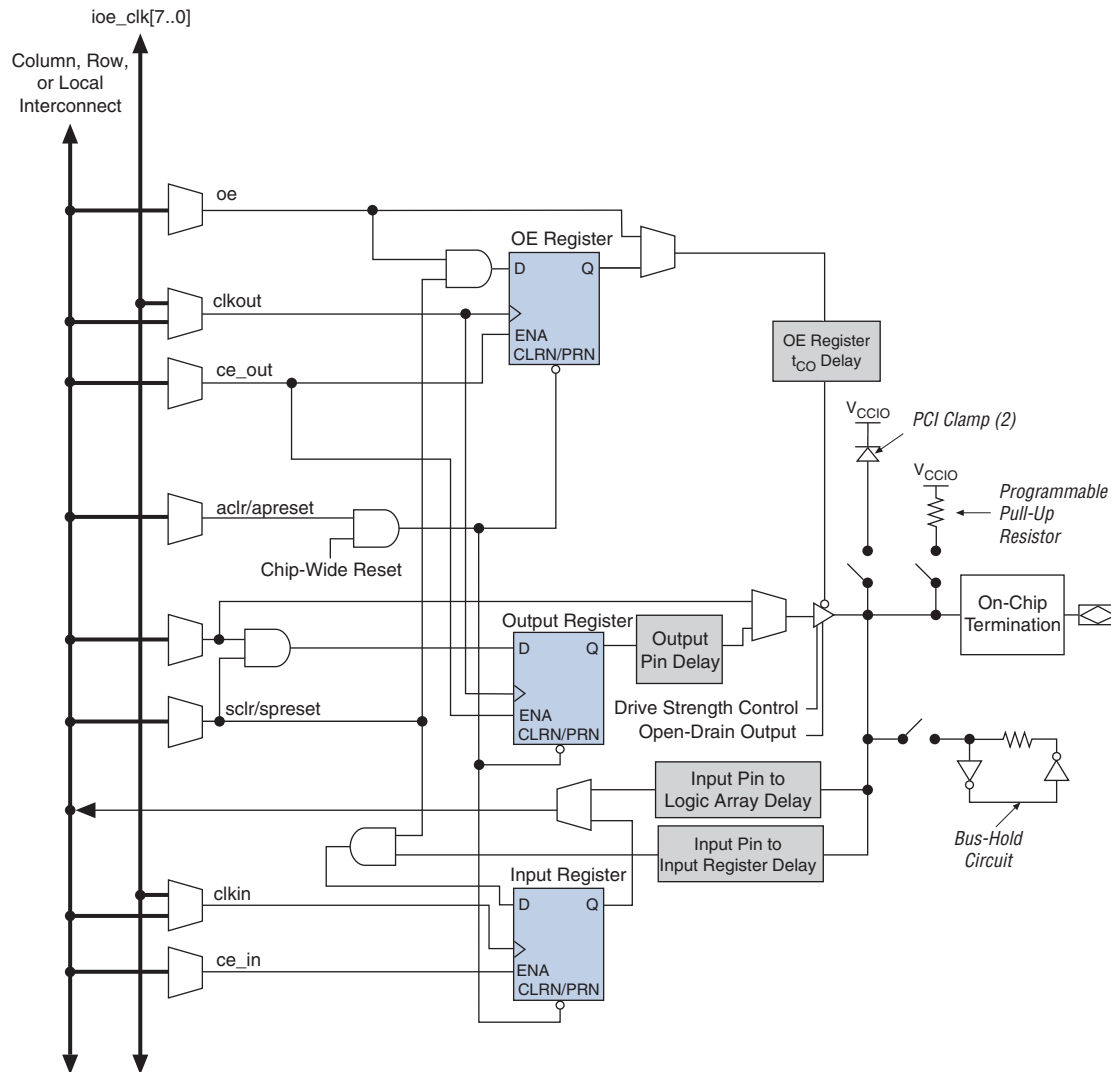


**Notes to Figure 2–72:**

(1)   All input signals to the IOE can be inverted at the IOE.
(2)   The optional PCI clamp is only available on column I/O pins.

The Arria GX device IOE includes programmable delays that can be activated to ensure input IOE register-to-logic array register transfers, input pin-to-logic array register transfers, or output IOE register-to-pin transfers.

A path in which a pin directly drives a register can require the delay to ensure zero hold time, whereas a path in which a pin drives a register through combinational logic may not require the delay. Programmable delays exist for decreasing input-pin-to-logic-array and IOE input register delays. The Quartus II Compiler can program these delays to automatically minimize setup time while providing a zero hold time. Programmable delays can increase the register-to-pin delays for output and/or output enable registers. Programmable delays are no longer required to ensure zero hold times for logic array register-to-IOE register transfers. The Quartus II Compiler can create zero hold time for these transfers. Table 2–22 shows the programmable delays for Arria GX devices.

**Table 2–22.** Arria GX Devices Programmable Delay Chain

| Programmable Delays | Quartus II Logic Option |
|---|---|
| Input pin to logic array delay | Input delay from pin to internal cells |
| Input pin to input register delay | Input delay from pin to input register |
| Output pin delay | Delay from output register to output pin |
| Output enable register $t_{CO}$ delay | Delay to output enable pin |

IOE registers in Arria GX devices share the same source for clear or preset. You can program preset or clear for each individual IOE. You can also program the registers to power up high or low after configuration is complete. If programmed to power up low, an asynchronous clear can control the registers. If programmed to power up high, an asynchronous preset can control the registers. This feature prevents the inadvertent activation of another device's active-low input upon power-up. If one register in an IOE uses a preset or clear signal, all registers in the IOE must use that same signal if they require preset or clear. Additionally, a synchronous reset signal is available for the IOE registers.

## Double Data Rate I/O Pins

Arria GX devices have six registers in the IOE, which support DDR interfacing by clocking data on both positive and negative clock edges. The IOEs in Arria GX devices support DDR inputs, DDR outputs, and bidirectional DDR modes. When using the IOE for DDR inputs, the two input registers clock double rate input data on alternating edges. An input latch is also used in the IOE for DDR input acquisition. The latch holds the data that is present during the clock high times, allowing both bits of data to be synchronous with the same clock edge (either rising or falling). Figure 2–73 shows an IOE configured for DDR input. Figure 2–74 shows the DDR input timing diagram.

**Figure 2–73.** Arria GX IOE in DDR Input I/O Configuration   *(Note 1)*
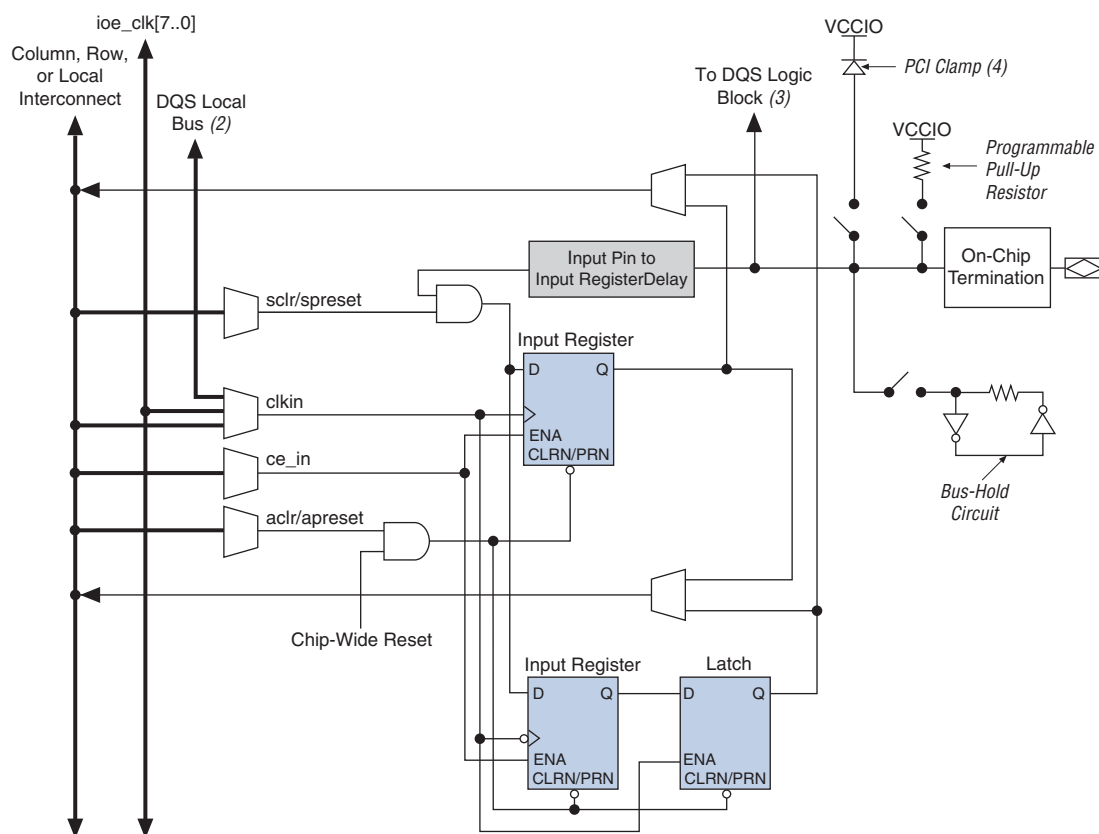


**Notes to Figure 2–73:**

(1)   All input signals to the IOE can be inverted at the IOE.
(2)   This signal connection is only allowed on dedicated DQ function pins.
(3)   This signal is for dedicated DQS function pins only.
(4)   The optional PCI clamp is only available on column I/O pins.

**Figure 2–74.** Input Timing Diagram in DDR Mode
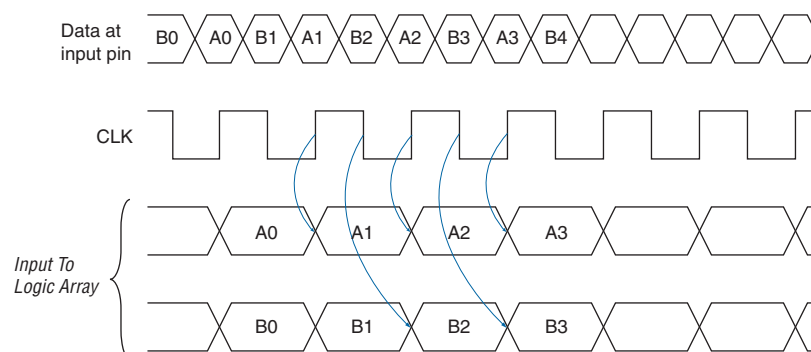
When using the IOE for DDR outputs, the two output registers are configured to clock two data paths from ALMs on rising clock edges. These output registers are multiplexed by the clock to drive the output pin at a ×2 rate. One output register clocks the first bit out on the clock high time, while the other output register clocks the second bit out on the clock low time. Figure 2–75 shows the IOE configured for DDR output. Figure 2–76 shows the DDR output timing diagram.

**Figure 2–75.** Arria GX IOE in DDR Output I/O Configuration *Notes (1), (2)*
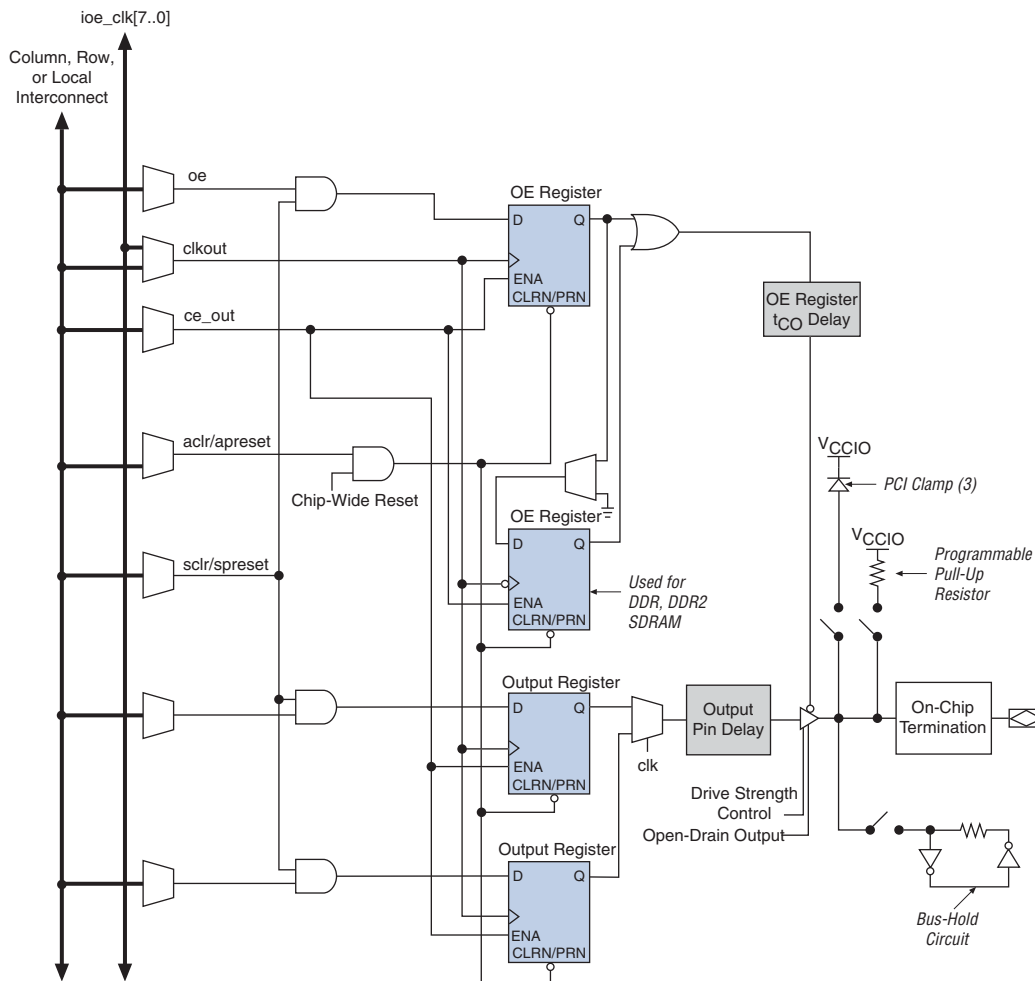


**Notes to Figure 2–75:**

(1)  All input signals to the IOE can be inverted at the IOE.

(2)  The tri-state buffer is active low. The DDIO megafunction represents the tri-state buffer as active-high with an inverter at the OE register data port.

(3)  The optional PCI clamp is only available on column I/O pins.

**Figure 2–76.** Output Timing Diagram in DDR Mode
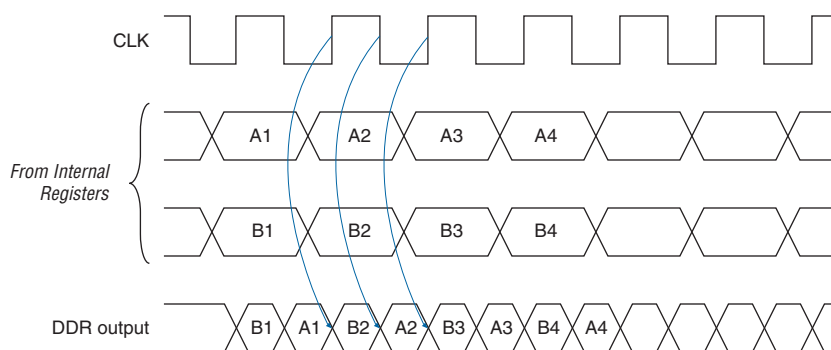


The Arria GX IOE operates in bidirectional DDR mode by combining the DDR input and DDR output configurations. The negative-edge-clocked OE register holds the `OE` signal inactive until the falling edge of the clock to meet DDR SDRAM timing requirements.

## External RAM Interfacing

In addition to the six I/O registers in each IOE, Arria GX devices also have dedicated phase-shift circuitry for interfacing with external memory interfaces, including DDR, DDR2 SDRAM, and SDR SDRAM. In every Arria GX device, the I/O banks at the top (Banks 3 and 4) and bottom (Banks 7 and 8) of the device support DQ and DQS signals with DQ bus modes of ×4, ×8/×9, ×16/×18, or ×32/×36. Table 2–23 shows the number of DQ and DQS buses that are supported per device.

**Table 2–23.** DQS and DQ Bus Mode Support  *(Note 1)*

| Device | Package | Number of ×4 Groups | Number of ×8/×9 Groups | Number of ×16/×18 Groups | Number of ×32/×36 Groups |
|---|---|---|---|---|---|
| EP1AGX20 | 484-pin FineLine BGA | 2 | 0 | 0 | 0 |
| EP1AGX35 | 484-pin FineLine BGA | 2 | 0 | 0 | 0 |
| | 780-pin FineLine BGA | 18 | 8 | 4 | 0 |
| EP1AGX50/60 | 484-pin FineLine BGA | 2 | 0 | 0 | 0 |
| | 780-pin FineLine BGA | 18 | 8 | 4 | 0 |
| | 1,152-pin FineLine BGA | 36 | 18 | 8 | 4 |
| EP1AGX90 | 1,152-pin FineLine BGA | 36 | 18 | 8 | 4 |

**Note to Table 2–23:**

(1)  Numbers are preliminary until devices are available.

A compensated delay element on each DQS pin automatically aligns input DQS synchronization signals with the data window of their corresponding DQ data signals. The DQS signals drive a local DQS bus in the top and bottom I/O banks. This DQS bus is an additional resource to the I/O clocks and is used to clock DQ input registers with the DQS signal.

The Arria GX device has two phase-shifting reference circuits, one on the top and one on the bottom of the device. The circuit on the top controls the compensated delay elements for all DQS pins on the top. The circuit on the bottom controls the compensated delay elements for all DQS pins on the bottom.

Each phase-shifting reference circuit is driven by a system reference clock, which must have the same frequency as the DQS signal. Clock pins CLK[15..12]p feed phase circuitry on the top of the device and clock pins CLK[7..4]p feed phase circuitry on the bottom of the device. In addition, PLL clock outputs can also feed the phase-shifting reference circuits. Figure 2–77 shows the phase-shift reference circuit control of each DQS delay shift on the top of the device. This same circuit is duplicated on the bottom of the device.

**Figure 2–77.** DQS Phase-Shift Circuitry    *(Note 1), (2)*
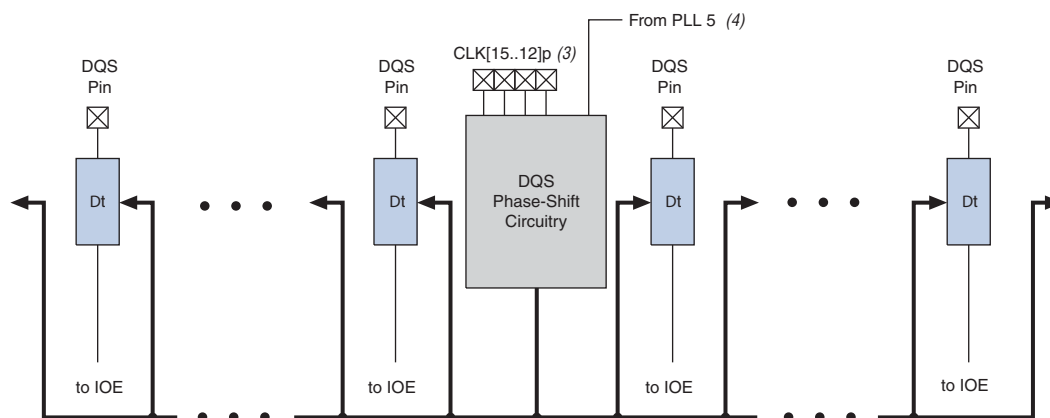


Notes to Figure 2–77:

(1) There are up to 18 pairs of DQS pins available on the top or bottom of the Arria GX device. There are up to 10 pairs on the right side and 8 pairs on the left side of the DQS phase-shift circuitry.

(2) The "t" module represents the DQS logic block.

(3) Clock pins CLK[15..12]p feed phase-shift circuitry on the top of the device and clock pins CLK[7..4]p feed the phase circuitry on the bottom of the device. You can also use a PLL clock output as a reference clock to phase shift circuitry.

(4) You can only use PLL 5 to feed the DQS phase-shift circuitry on the top of the device and PLL 6 to feed the DQS phase-shift circuitry on the bottom of the device.

These dedicated circuits combined with enhanced PLL clocking and phase-shift ability provide a complete hardware solution for interfacing to high-speed memory.

For more information about external memory interfaces, refer to the *External Memory Interfaces in Arria GX Devices* chapter.

## Programmable Drive Strength

The output buffer for each Arria GX device I/O pin has a programmable drive strength control for certain I/O standards. The LVTTL, LVCMOS, SSTL, and HSTL standards have several levels of drive strength that you can control. The default setting used in the Quartus II software is the maximum current strength setting that is used to achieve maximum I/O performance. For all I/O standards, the minimum setting is the lowest drive strength that guarantees the $I_{OH}/I_{OL}$ of the standard. Using minimum settings provides signal slew rate control to reduce system noise and signal overshoot.

Table 2–24 shows the possible settings for I/O standards with drive strength control.

**Table 2–24.** Programmable Drive Strength   *(Note 1)*

| I/O Standard | $I_{OH}$ / $I_{OL}$ Current Strength Setting (mA) for Column I/O Pins | $I_{OH}$ / $I_{OL}$ Current Strength Setting (mA) for Row I/O Pins |
|---|---|---|
| 3.3-V LVTTL | 24, 20, 16, 12, 8, 4 | 12, 8, 4 |
| 3.3-V LVCMOS | 24, 20, 16, 12, 8, 4 | 8, 4 |
| 2.5-V LVTTL/LVCMOS | 16, 12, 8, 4 | 12, 8, 4 |
| 1.8-V LVTTL/LVCMOS | 12, 10, 8, 6, 4, 2 | 8, 6, 4, 2 |
| 1.5-V LVCMOS | 8, 6, 4, 2 | 4, 2 |
| SSTL-2 Class I | 12, 8 | 12, 8 |
| SSTL-2 Class II | 24, 20, 16 | 16 |
| SSTL-18 Class I | 12, 10, 8, 6, 4 | 10, 8, 6, 4 |
| SSTL-18 Class II | 20, 18, 16, 8 | — |
| HSTL-18 Class I | 12, 10, 8, 6, 4 | 12, 10, 8, 6, 4 |
| HSTL-18 Class II | 20, 18, 16 | — |
| HSTL-15 Class I | 12, 10, 8, 6, 4 | 8, 6, 4 |
| HSTL-15 Class II | 20, 18, 16 | — |

**Note to Table 2–24:**

(1)   The Quartus II software default current setting is the maximum setting for each I/O standard.

## Open-Drain Output

Arria GX devices provide an optional open-drain (equivalent to an open collector) output for each I/O pin. This open-drain output enables the device to provide system-level control signals (for example, interrupt and write enable signals) that can be asserted by any of several devices.

## Bus Hold

Each Arria GX device I/O pin provides an optional bus-hold feature. Bus-hold circuitry can hold the signal on an I/O pin at its last-driven state. Because the bus-hold feature holds the last-driven state of the pin until the next input signal is present, an external pull-up or pull-down resistor is not needed to hold a signal level when the bus is tri-stated.

Bus-hold circuitry also pulls undriven pins away from the input threshold voltage where noise can cause unintended high-frequency switching. You can select this feature individually for each I/O pin. The bus-hold output drives no higher than $V_{CCIO}$ to prevent overdriving signals. If the bus-hold feature is enabled, the programmable pull-up option cannot be used. Disable the bus-hold feature when the I/O pin has been configured for differential signals.

Bus-hold circuitry uses a resistor with a nominal resistance (RBH) of approximately 7 kΩ to pull the signal level to the last-driven state. This information is provided for each $V_{CCIO}$ voltage level. Bus-hold circuitry is active only after configuration. When going into user mode, the bus-hold circuit captures the value on the pin present at the end of configuration.

For the specific sustaining current driven through this resistor and overdrive current used to identify the next-driven input level, refer to the *DC & Switching Characteristics* chapter.

## Programmable Pull-Up Resistor

Each Arria GX device I/O pin provides an optional programmable pull-up resistor during user mode. If you enable this feature for an I/O pin, the pull-up resistor (typically 25 kΩ) holds the output to the $V_{CCIO}$ level of the output pin's bank.

## Advanced I/O Standard Support

Arria GX device IOEs support the following I/O standards:

■ 3.3-V LVTTL/LVCMOS

■ 2.5-V LVTTL/LVCMOS

■ 1.8-V LVTTL/LVCMOS

■ 1.5-V LVCMOS

■ 3.3-V PCI

■ 3.3-V PCI-X mode 1

■ LVDS

■ LVPECL (on input and output clocks only)

■ Differential 1.5-V HSTL class I and II

■ Differential 1.8-V HSTL class I and II

■ Differential SSTL-18 class I and II

■ Differential SSTL-2 class I and II

■ 1.2-V HSTL class I and II

■ 1.5-V HSTL class I and II

■ 1.8-V HSTL class I and II

■ SSTL-2 class I and II

■ SSTL-18 class I and II

Table 2–25 describes the I/O standards supported by Arria GX devices.

**Table 2–25.** Arria GX Devices Supported I/O Standards

| I/O Standard | Type | Input Reference Voltage (V_REF) (V) | Output Supply Voltage (V_CCIO) (V) | Board Termination Voltage (V_TT) (V) |
|---|---|---|---|---|
| LVTTL | Single-ended | — | 3.3 | — |
| LVCMOS | Single-ended | — | 3.3 | — |
| 2.5 V | Single-ended | — | 2.5 | — |
| 1.8 V | Single-ended | — | 1.8 | — |
| 1.5-V LVCMOS | Single-ended | — | 1.5 | — |
| 3.3-V PCI | Single-ended | — | 3.3 | — |
| 3.3-V PCI-X mode 1 | Single-ended | — | 3.3 | — |
| LVDS | Differential | — | 2.5 *(3)* | — |
| LVPECL *(1)* | Differential | — | 3.3 | — |
| HyperTransport technology | Differential | — | 2.5 *(3)* | — |
| Differential 1.5-V HSTL class I and II *(2)* | Differential | 0.75 | 1.5 | 0.75 |
| Differential 1.8-V HSTL class I and II *(2)* | Differential | 0.90 | 1.8 | 0.90 |
| Differential SSTL-18 class I and II *(2)* | Differential | 0.90 | 1.8 | 0.90 |
| Differential SSTL-2 class I and II *(2)* | Differential | 1.25 | 2.5 | 1.25 |
| 1.2-V HSTL *(4)* | Voltage-referenced | 0.6 | 1.2 | 0.6 |
| 1.5-V HSTL class I and II | Voltage-referenced | 0.75 | 1.5 | 0.75 |
| 1.8-V HSTL class I and II | Voltage-referenced | 0.9 | 1.8 | 0.9 |
| SSTL-18 class I and II | Voltage-referenced | 0.90 | 1.8 | 0.90 |
| SSTL-2 class I and II | Voltage-referenced | 1.25 | 2.5 | 1.25 |

**Notes to Table 2–25:**

(1) This I/O standard is only available on input and output column clock pins.

(2) This I/O standard is only available on input clock pins and DQS pins in I/O banks 3, 4, 7, and 8, and output clock pins in I/O banks 9, 10, 11, and 12.

(3) $V_{CCIO}$ is 3.3 V when using this I/O standard in input and output column clock pins (in I/O banks 3, 4, 7, 8, 9, 10, 11, and 12).

(4) 1.2-V HSTL is only supported in I/O banks 4, 7, and 8.

> For more information about the I/O standards supported by Arria GX I/O banks, refer to the *Selectable I/O Standards in Arria GX Devices* chapter.

Arria GX devices contain six I/O banks and four enhanced PLL external clock output banks, as shown in Figure 2–78. The two I/O banks on the left of the device contain circuitry to support source-synchronous, high-speed differential I/O for LVDS inputs and outputs. These banks support all Arria GX I/O standards except PCI or PCI-X I/O pins, and SSTL-18 class II and HSTL outputs. The top and bottom I/O banks support all single-ended I/O standards. Additionally, enhanced PLL external clock output banks allow clock output capabilities such as differential support for SSTL and HSTL.

**Figure 2–78.** Arria GX I/O Banks    *(Note 1)*, *(2)*
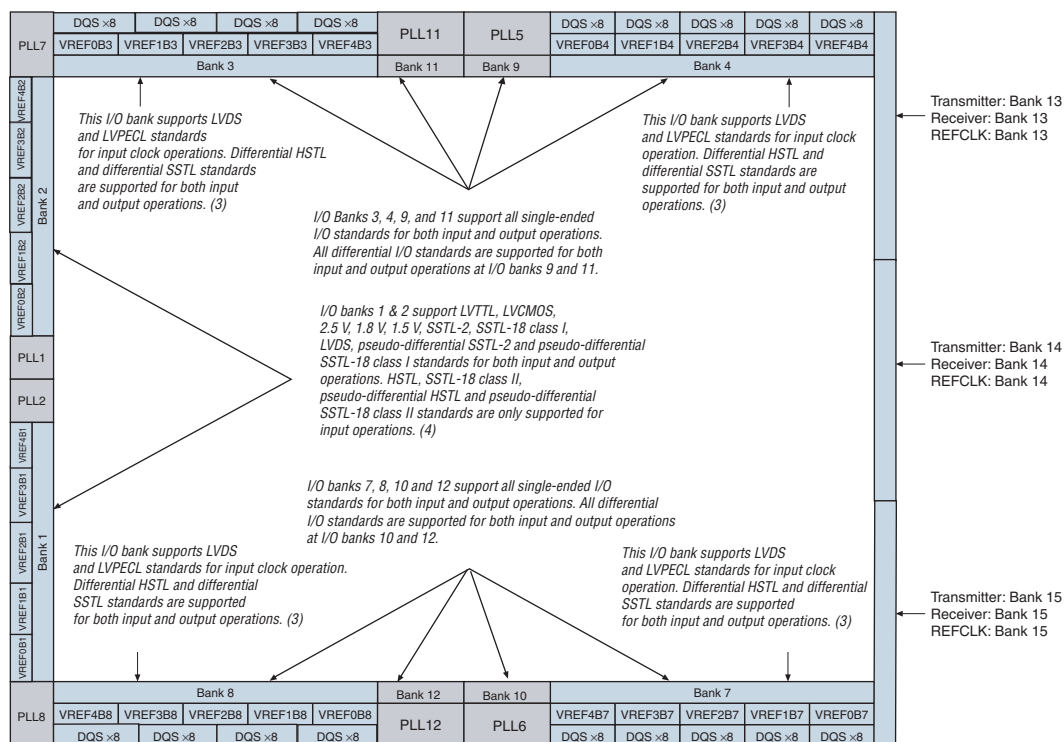


**Notes to Figure 2–78:**

(1) Figure 2–78 is a top view of the silicon die that corresponds to a reverse view for flip chip packages. It is a graphical representation only.

(2) Depending on the size of the device, different device members have different numbers of $V_{REF}$ groups. For the exact locations, refer to the pin list and the Quartus II software.

(3) Banks 9 through 12 are enhanced PLL external clock output banks.

(4) Horizontal I/O banks feature SERDES and DPA circuitry for high-speed differential I/O standards. For more information about differential I/O standards, refer to the *High-Speed Differential I/O Interfaces in Arria GX Devices* chapter.

Each I/O bank has its own VCCIO pins. A single device can support 1.5-, 1.8-, 2.5-, and 3.3-V interfaces; each bank can support a different $V_{CCIO}$ level independently. Each bank also has dedicated VREF pins to support the voltage-referenced standards (such as SSTL-2).

Each I/O bank can support multiple standards with the same $V_{CCIO}$ for input and output pins. Each bank can support one $V_{REF}$ voltage level. For example, when $V_{CCIO}$ is 3.3 V, a bank can support LVTTL, LVCMOS, and 3.3-V PCI for inputs and outputs.

## On-Chip Termination

Arria GX devices provide differential (for the LVDS technology I/O standard) and on-chip series termination to reduce reflections and maintain signal integrity. There is no calibration support for these on-chip termination resistors. On-chip termination simplifies board design by minimizing the number of external termination resistors required. Termination can be placed inside the package, eliminating small stubs that can still lead to reflections.

Arria GX devices provide two types of termination:

■ On-chip differential termination ($R_D$ OCT)

■ On-chip series termination ($R_S$ OCT)

Table 2–26 lists the Arria GX OCT support per I/O bank.

**Table 2–26.** On-Chip Termination Support by I/O Banks

| On-Chip Termination Support | I/O Standard Support | Top and Bottom Banks (3, 4, 7, 8) | Left Bank (1, 2) |
|---|---|:---:|:---:|
| Series termination | 3.3-V LVTTL | ✔ | ✔ |
| | 3.3-V LVCMOS | ✔ | ✔ |
| | 2.5-V LVTTL | ✔ | ✔ |
| | 2.5-V LVCMOS | ✔ | ✔ |
| | 1.8-V LVTTL | ✔ | ✔ |
| | 1.8-V LVCMOS | ✔ | ✔ |
| | 1.5-V LVTTL | ✔ | ✔ |
| | 1.5-V LVCMOS | ✔ | ✔ |
| | SSTL-2 class I and II | ✔ | ✔ |
| | SSTL-18 class I | ✔ | ✔ |
| | SSTL-18 class II | ✔ | — |
| | 1.8-V HSTL class I | ✔ | ✔ |
| | 1.8-V HSTL class II | ✔ | — |
| | 1.5-V HSTL class I | ✔ | ✔ |
| | 1.2-V HSTL | ✔ | — |
| Differential termination *(1)* | LVDS | — | ✔ |
| | HyperTransport technology | — | ✔ |

**Note to Table 2–26:**

(1) Clock pins `CLK1` and `CLK3`, and pins `FPLL[7..8]CLK` do not support differential on-chip termination. Clock pins `CLK0` and `CLK2`, do support differential on-chip termination. Clock pins in the top and bottom banks (`CLK[4..7, 12..15]`) do not support differential on-chip termination.

### On-Chip Differential Termination ($R_D$ OCT)

Arria GX devices support internal differential termination with a nominal resistance value of 100 $\Omega$ for LVDS input receiver buffers. LVPECL input signals (supported on clock pins only) require an external termination resistor. $R_D$ OCT is supported across the full range of supported differential data rates as shown in the *High-Speed I/O Specifications* section of the *DC & Switching Characteristics* chapter.

For more information about $R_D$ OCT, refer to the *High-Speed Differential I/O Interfaces with DPA in Arria GX Devices* chapter.

For more information about tolerance specifications for $R_D$ OCT, refer to the *DC & Switching Characteristics* chapter.

### On-Chip Series Termination (R$_S$ OCT)

Arria GX devices support driver impedance matching to provide the I/O driver with controlled output impedance that closely matches the impedance of the transmission line. As a result, reflections can be significantly reduced. Arria GX devices support R$_S$ OCT for single-ended I/O standards with typical R$_S$ values of 25 and 50 Ω. Once matching impedance is selected, current drive strength is no longer selectable. Table 2–26 shows the list of output standards that support R$_S$ OCT.

For more information about R$_S$ OCT supported by Arria GX devices, refer to the *Selectable I/O Standards in Arria GX Devices* chapter.

For more information about tolerance specifications for OCT without calibration, refer to the *DC & Switching Characteristics* chapter.

## MultiVolt I/O Interface

The Arria GX architecture supports the MultiVolt I/O interface feature that allows Arria GX devices in all packages to interface with systems of different supply voltages. Arria GX VCCINT pins must always be connected to a 1.2-V power supply. With a 1.2-V V$_{CCINT}$ level, input pins are 1.2-, 1.5-, 1.8-, 2.5-, and 3.3-V tolerant. The VCCIO pins can be connected to either a 1.2-, 1.5-, 1.8-, 2.5-, or 3.3-V power supply, depending on the output requirements. The output levels are compatible with systems of the same voltage as the power supply (for example, when VCCIO pins are connected to a 1.5-V power supply, the output levels are compatible with 1.5-V systems). Arria GX VCCPD power pins must be connected to a 3.3-V power supply. These power pins are used to supply the pre-driver power to the output buffers, which increases the performance of the output pins. The VCCPD pins also power configuration input pins and JTAG input pins.

Table 2–27 lists Arria GX MultiVolt I/O support.

**Table 2–27.** Arria GX MultiVolt I/O Support    *(Note 1)*

| V$_{CCIO}$ (V) | Input Signal (V) | | | | | Output Signal (V) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1.2 | 1.5 | 1.8 | 2.5 | 3.3 | 1.2 | 1.5 | 1.8 | 2.5 | 3.3 | 5.0 |
| 1.2 | *(4)* | ✓ *(2)* | ✓ *(2)* | ✓ *(2)* | ✓ *(2)* | ✓ *(4)* | — | — | — | — | — |
| 1.5 | *(4)* | ✓ | ✓ | ✓ *(2)* | ✓ *(2)* | ✓ *(3)* | ✓ | — | — | — | — |
| 1.8 | *(4)* | ✓ | ✓ | ✓ *(2)* | ✓ *(2)* | ✓ *(3)* | ✓ *(3)* | ✓ | — | — | — |
| 2.5 | *(4)* | — | — | ✓ | ✓ | ✓ *(3)* | ✓ *(3)* | ✓ *(3)* | ✓ | — | — |
| 3.3 | *(4)* | — | — | ✓ | ✓ | ✓ *(3)* | ✓ *(3)* | ✓ *(3)* | ✓ *(3)* | ✓ | ✓ |

**Notes to Table 2–27:**

(1)  To drive inputs higher than V$_{CCIO}$ but less than 4.0 V, disable the PCI clamping diode and select the **Allow LVTTL and LVCMOS input levels to overdrive input buffer** option in the Quartus II software.

(2)  The pin current may be slightly higher than the default value. You must verify that the driving device's V$_{OL}$ maximum and V$_{OH}$ minimum voltages do not violate the applicable Arria GX V$_{IL}$ maximum and V$_{IH}$ minimum voltage specifications.

(3)  Although V$_{CCIO}$ specifies the voltage necessary for the Arria GX device to drive out, a receiving device powered at a different level can still interface with the Arria GX device if it has inputs that tolerate the V$_{CCIO}$ value.

(4)  Arria GX devices support 1.2-V HSTL. They do not support 1.2-V LVTTL and 1.2-V LVCMOS.

The TDO and nCEO pins are powered by $V_{CCIO}$ of the bank that they reside. TDO is in I/O bank 4 and nCEO is in I/O Bank 7. Ideally, the $V_{CC}$ supplies for the I/O buffers of any two connected pins are at the same voltage level. This may not always be possible depending on the $V_{CCIO}$ level of TDO and nCEO pins on master devices and the configuration voltage level chosen by $V_{CCSEL}$ on slave devices. Master and slave devices can be in any position in the chain. The master device indicates that it is driving out TDO or nCEO to a slave device. For multi-device passive configuration schemes, the nCEO pin of the master device drives the nCE pin of the slave device. The VCCSEL pin on the slave device selects which input buffer is used for nCE. When $V_{CCSEL}$ is logic high, it selects the 1.8-V/1.5-V buffer powered by $V_{CCIO}$. When $V_{CCSEL}$ is logic low, it selects the 3.3-V/2.5-V input buffer powered by $V_{CCPD}$. The ideal case is to have the $V_{CCIO}$ of the nCEO bank in a master device match the $V_{CCSEL}$ settings for the nCE input buffer of the slave device it is connected to, but that may not be possible depending on the application.

Table 2–28 contains board design recommendations to ensure that nCEO can successfully drive nCE for all power supply combinations.

**Table 2–28.** Board Design Recommendations for nCEO and nCE Input Buffer Power

| nCE Input Buffer Power in I/O Bank 3 | Arria GX nCEO $V_{CCIO}$ Voltage Level in I/O Bank 7 | | | | |
|---|---|---|---|---|---|
| | $V_{CCIO}$ = 3.3 V | $V_{CCIO}$ = 2.5 V | $V_{CCIO}$ = 1.8 V | $V_{CCIO}$ = 1.5 V | $V_{CCIO}$ = 1.2 V |
| VCCSEL high ($V_{CCIO}$ Bank 3 = 1.5 V) | ✔ (1), (2) | ✔ (3), (4) | ✔ (5) | ✔ | ✔ |
| VCCSEL high ($V_{CCIO}$ Bank 3 = 1.8 V) | ✔ (1), (2) | ✔ (3), (4) | ✔ | ✔ | Level shifter required |
| VCCSEL low (nCE powered by $V_{CCPD}$ 3.3 V) | ✔ | ✔ (4) | ✔ (6) | Level shifter required | Level shifter required |

Notes to Table 2–28:

(1) Input buffer is 3.3-V tolerant.

(2) The nCEO output buffer meets $V_{OH}$ (MIN) = 2.4 V.

(3) Input buffer is 2.5-V tolerant.

(4) The nCEO output buffer meets $V_{OH}$ (MIN) = 2.0 V.

(5) Input buffer is 1.8-V tolerant.

(6) An external 250-Ω pull-up resistor is not required, but recommended if signal levels on the board are not optimal.

For JTAG chains, the TDO pin of the first device drives the TDI pin of the second device in the chain. The $V_{CCSEL}$ input on JTAG input I/O cells (TCK, TMS, TDI, and TRST) is internally hardwired to GND selecting the 3.3-V/2.5-V input buffer powered by $V_{CCPD}$. The ideal case is to have the $V_{CCIO}$ of the TDO bank from the first device to match the $V_{CCSEL}$ settings for TDI on the second device, but that may not be possible depending on the application. Table 2–29 contains board design recommendations to ensure proper JTAG chain operation.

**Table 2–29.** Supported TDO/TDI Voltage Combinations

| Device | TDI Input Buffer Power | Arria GX TDO $V_{CCIO}$ Voltage Level in I/O Bank 4 | | | | |
|---|---|---|---|---|---|---|
| | | $V_{CCIO}$ = 3.3 V | $V_{CCIO}$ = 2.5 V | $V_{CCIO}$ = 1.8 V | $V_{CCIO}$ = 1.5 V | $V_{CCIO}$ = 1.2 V |
| Arria GX | Always $V_{CCPD}$ (3.3 V) | ✓ (1) | ✓ (2) | ✓ (3) | Level shifter required | Level shifter required |
| Non-Arria GX | VCC = 3.3 V | ✓ (1) | ✓ (2) | ✓ (3) | Level shifter required | Level shifter required |
| | VCC = 2.5 V | ✓ (1), (4) | ✓ (2) | ✓ (3) | Level shifter required | Level shifter required |
| | VCC = 1.8 V | ✓ (1), (4) | ✓ (2), (5) | ✓ | Level shifter required | Level shifter required |
| | VCC = 1.5 V | ✓ (1), (4) | ✓ (2), (5) | ✓ (6) | ✓ | ✓ |

**Notes to Table 2–29:**

(1) The TDO output buffer meets $V_{OH}$ (MIN) = 2.4 V.

(2) The TDO output buffer meets $V_{OH}$ (MIN) = 2.0 V.

(3) An external 250-Ω pull-up resistor is not required, but recommended if signal levels on the board are not optimal.

(4) Input buffer must be 3.3-V tolerant.

(5) Input buffer must be 2.5-V tolerant.

(6) Input buffer must be 1.8-V tolerant.

# High-Speed Differential I/O with DPA Support

Arria GX devices contain dedicated circuitry for supporting differential standards at speeds up to 840 Mbps. LVDS differential I/O standards are supported in the Arria GX device. In addition, the LVPECL I/O standard is supported on input and output clock pins on the top and bottom I/O banks.

The high-speed differential I/O circuitry supports the following high-speed I/O interconnect standards and applications:

■ SPI-4 Phase 2 (POS-PHY Level 4)

■ SFI-4

■ Parallel RapidIO standard

There are two dedicated high-speed PLLs (PLL1 and PLL2) in the EP1AGX20 and EP1AGX35 devices and up to four dedicated high-speed PLLs (PLL1, PLL2, PLL7, and PLL8) in the EP1AGX50, EP1AGX60, and EP1AGX90 devices to multiply reference clocks and drive high-speed differential SERDES channels in I/O banks 1 and 2.

Table 2–30 through Table 2–34 list the number of channels that each fast PLL can clock in each of the Arria GX devices. In Table 2–30 through Table 2–34 the first row for each transmitter or receiver provides the maximum number of channels that each fast PLL can drive in its adjacent I/O bank (I/O Bank 1 **or** I/O Bank 2). The second row shows the maximum number of channels that each fast PLL can drive in both I/O banks (I/O Bank 1 **and** I/O Bank 2). For example, in the 780-pin FineLine BGA EP1AGX20

device, PLL 1 can drive a maximum of 16 transmitter channels in I/O Bank 2 or a maximum of 29 transmitter channels in I/O Banks 1 and 2. The Quartus II software can also merge receiver and transmitter PLLs when a receiver is driving a transmitter. In this case, one fast PLL can drive both the maximum numbers of receiver and transmitter channels.

☞ For more information, refer to the "Differential Pin Placement Guidelines" section in the *High-Speed Differential I/O Interfaces with DPA in Arria GX Devices* chapter.

**Table 2–30.** EP1AGX20 Device Differential Channels *(Note 1)*

| Package | Transmitter/Receiver | Total Channels | Center Fast PLLs | |
|---|---|---|---|---|
| | | | PLL1 | PLL2 |
| 484-pin FineLine BGA | Transmitter | 29 | 16 | 13 |
| | | | 13 | 16 |
| | Receiver | 31 | 17 | 14 |
| | | | 14 | 17 |
| 780-pin FineLine GBA | Transmitter | 29 | 16 | 13 |
| | | | 13 | 16 |
| | Receiver | 31 | 17 | 14 |
| | | | 14 | 17 |

**Note to Table 2–30:**

(1) The total number of receiver channels includes the four non-dedicated clock channels that can be optionally used as data channels.

**Table 2–31.** EP1AGX35 Device Differential Channels *(Note 1)*

| Package | Transmitter/Receiver | Total Channels | Center Fast PLLs | |
|---|---|---|---|---|
| | | | PLL1 | PLL2 |
| 484-pin FineLine BGA | Transmitter | 29 | 16 | 13 |
| | | | 13 | 16 |
| | Receiver | 31 | 17 | 14 |
| | | | 14 | 17 |
| 780-pin FineLine BGA | Transmitter | 29 | 16 | 13 |
| | | | 13 | 16 |
| | Receiver | 31 | 17 | 14 |
| | | | 14 | 17 |

**Note to Table 2–31:**

(1) The total number of receiver channels includes the four non-dedicated clock channels that can be optionally used as data channels.

**Table 2–32.** EP1AGX50 Device Differential Channels *(Note 1)*

| Package | Transmitter/ Receiver | Total Channels | Center Fast PLLs | | Corner Fast PLLs | |
|---|---|---|---|---|---|---|
| | | | PLL1 | PLL2 | PLL7 | PLL8 |
| 484-pin FineLine BGA | Transmitter | 29 | 16 | 13 | — | — |
| | | | 13 | 16 | — | — |
| | Receiver | 31 | 17 | 14 | — | — |
| | | | 14 | 17 | — | — |
| 780-pin FineLine BGA | Transmitter | 29 | 16 | 13 | — | — |
| | | | 13 | 16 | — | — |
| | Receiver | 31 | 17 | 14 | — | — |
| | | | 14 | 17 | — | — |
| 1,152-pin FineLine BGA | Transmitter | 42 | 21 | 21 | 21 | 21 |
| | | | 21 | 21 | — | — |
| | Receiver | 42 | 21 | 21 | 21 | 21 |
| | | | 21 | 21 | — | — |

**Note to Table 2–32:**

(1)   The total number of receiver channels includes the four non-dedicated clock channels that can be optionally used as data channels.

**Table 2–33.** EP1AGX60 Device Differential Channels *(Note 1)*

| Package | Transmitter/ Receiver | Total Channels | Center Fast PLLs | | Corner Fast PLLs | |
|---|---|---|---|---|---|---|
| | | | PLL1 | PLL2 | PLL7 | PLL8 |
| 484-pin FineLine BGA | Transmitter | 29 | 16 | 13 | — | — |
| | | | 13 | 16 | — | — |
| | Receiver | 31 | 17 | 14 | — | — |
| | | | 14 | 17 | — | — |
| 780-pin FineLine BGA | Transmitter | 29 | 16 | 13 | — | — |
| | | | 13 | 16 | — | — |
| | Receiver | 31 | 17 | 14 | — | — |
| | | | 14 | 17 | — | — |
| 1,152-pin FineLine BGA | Transmitter | 42 | 21 | 21 | 21 | 21 |
| | | | 21 | 21 | — | — |
| | Receiver | 42 | 21 | 21 | 21 | 21 |
| | | | 21 | 21 | — | — |

**Note to Table 2–33:**

(1)   The total number of receiver channels includes the four non-dedicated clock channels that can be optionally used as data channels.

**Table 2–34.** EP1AGX90 Device Differential Channels *(Note 1)*

| Package | Transmitter/Receiver | Total Channels | Center Fast PLLs | | Corner Fast PLLs |
|---------|----------------------|----------------|---------|---------|---------|
| | | | PLL1 | PLL2 | PLL7 |
| 1,152-pin FineLine BGA | Transmitter | 45 | 23 | 22 | 23 |
| | | | 22 | 23 | — |
| | Receiver | 47 | 23 | 24 | 23 |
| | | | 24 | 23 | — |

**Note to Table 2–34:**

(1)   The total number of receiver channels includes the four non-dedicated clock channels that can be optionally used as data channels.

## Dedicated Circuitry with DPA Support

Arria GX devices support source-synchronous interfacing with LVDS signaling at up to 840 Mbps. Arria GX devices can transmit or receive serial channels along with a low-speed or high-speed clock.

The receiving device PLL multiplies the clock by an integer factor W = 1 through 32. The SERDES factor J determines the parallel data width to deserialize from receivers or to serialize for transmitters. The SERDES factor J can be set to 4, 5, 6, 7, 8, 9, or 10 and does not have to equal the PLL clock-multiplication W value. A design using the dynamic phase aligner also supports all of these J factor values. For a J factor of 1, the Arria GX device bypasses the SERDES block. For a J factor of 2, the Arria GX device bypasses the SERDES block, and the DDR input and output registers are used in the IOE. Figure 2–79 shows the block diagram of the Arria GX transmitter channel.

**Figure 2–79.** Arria GX Transmitter Channel
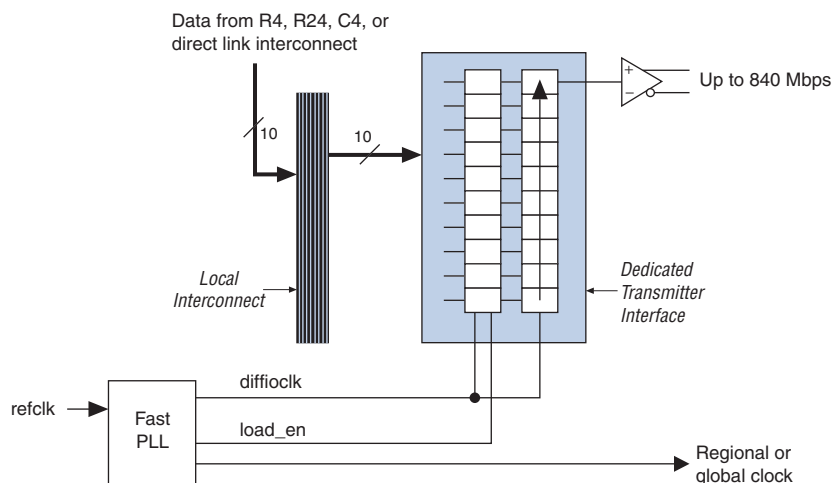


Each Arria GX receiver channel features a DPA block for phase detection and selection, a SERDES, a synchronizer, and a data realigner circuit. You can bypass the dynamic phase aligner without affecting the basic source-synchronous operation of the channel. In addition, you can dynamically switch between using the DPA block or bypassing the block via a control signal from the logic array.

Figure 2–80 shows the block diagram of the Arria GX receiver channel.

**Figure 2–80.** GX Receiver Channel



An external pin or global or regional clock can drive the fast PLLs, which can output up to three clocks: two multiplied high-speed clocks to drive the SERDES block and/or external pin, and a low-speed clock to drive the logic array. In addition, eight phase-shifted clocks from the $V_{CO}$ can feed to the DPA circuitry.

For more information about fast PLL, refer to the *PLLs in Arria GX Devices* chapter.

The eight phase-shifted clocks from the fast PLL feed to the DPA block. The DPA block selects the closest phase to the center of the serial data eye to sample the incoming data. This allows the source-synchronous circuitry to capture incoming data correctly regardless of channel-to-channel or clock-to-channel skew. The DPA block locks to a phase closest to the serial data phase. The phase-aligned DPA clock is used to write the data into the synchronizer.

The synchronizer sits between the DPA block and the data realignment and SERDES circuitry. Because every channel using the DPA block can have a different phase selected to sample the data, the synchronizer is needed to synchronize the data to the high-speed clock domain of the data realignment and the SERDES circuitry.

For high-speed source-synchronous interfaces such as POS-PHY 4 and the Parallel RapidIO standard, the source synchronous clock rate is not a byte- or SERDES-rate multiple of the data rate. Byte alignment is necessary for these protocols because the source synchronous clock does not provide a byte or word boundary as the clock is one half the data rate, not one eighth. The Arria GX device's high-speed differential I/O circuitry provides dedicated data realignment circuitry for user-controlled byte boundary shifting. This simplifies designs while saving ALM resources. You can use an ALM-based state machine to signal the shift of receiver byte boundaries until a specified pattern is detected to indicate byte alignment.

## Fast PLL and Channel Layout

The receiver and transmitter channels are interleaved as such that each I/O bank on the left side of the device has one receiver channel and one transmitter channel per LAB row. Figure 2–81 shows the fast PLL and channel layout in the EP1AGX20C, EP1AGX35C/D, EP1AGX50C/D and EP1AGX60C/D devices. Figure 2–82 shows the fast PLL and channel layout in EP1AGX60E and EP1AGX90E devices.

**Figure 2–81.** Fast PLL and Channel Layout in EP1AGX20C, EP1AGX35C/D, EP1AGX50C/D, EP1AGX60C/D Devices    *(Note 1)*
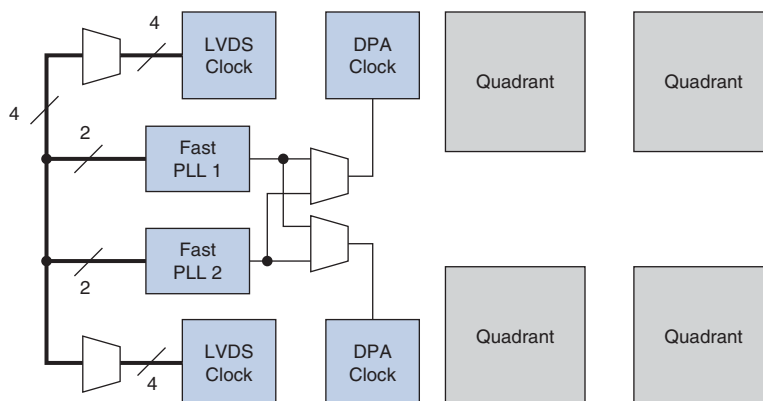


**Note to Figure 2–81:**

(1) For the number of channels each device supports, refer to Table 2–30.

**Figure 2–82.** Fast PLL and Channel Layout in EP1AGX60E and EP1AGX90E Devices    *(Note 1)*
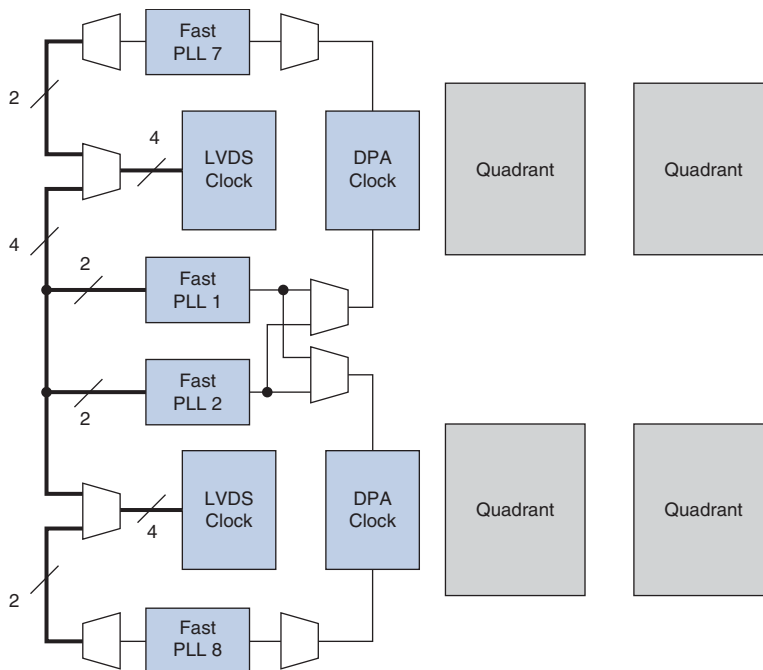


**Note to Figure 2–82:**

(1) For the number of channels each device supports, refer to Table 2–30 through Table 2–34.

# Document Revision History

Table 2–35 shows the revision history for this chapter.

**Table 2–35.** Document Revision History

| Date and Document Version | Changes Made | Summary of Changes |
|---|---|---|
| December 2009, v2.0 | ■ Document template update.<br>■ Minor text edits. | — |
| May 2008, v1.3 | Added "Reverse Serial Pre-CDR Loopback" and "Calibration Block" sub-sections to "Transmitter Path" section. | — |
| August 2007, v1.2 | Added "Referenced Documents" section. | — |
| June 2007, v1.1 | Added GIGE information. | — |
| May 2007 v1.0 | Initial release. | — |

# Introduction

All Arria® GX devices provide JTAG boundary-scan test (BST) circuitry that complies with the IEEE Std. 1149.1. You can perform JTAG boundary-scan testing either before or after, but not during configuration. Arria GX devices can also use the JTAG port for configuration with the Quartus® II software or hardware using either jam files (**.jam**) or jam byte-code files (**.jbc**).

This chapter contains the following sections:

■ "IEEE Std. 1149.1 JTAG Boundary-Scan Support"

■ "SignalTap II Embedded Logic Analyzer" on page 3–3

■ "Configuration" on page 3–3

■ "Automated Single Event Upset (SEU) Detection" on page 3–8

# IEEE Std. 1149.1 JTAG Boundary-Scan Support

Arria GX devices support I/O element (IOE) standard setting reconfiguration through the JTAG BST chain. The JTAG chain can update the I/O standard for all input and output pins any time before or during user-mode through the CONFIG_IO instruction. You can use this capability for JTAG testing before configuration when some of the Arria GX pins drive or receive from other devices on the board using voltage-referenced standards. Because the Arria GX device may not be configured before JTAG testing, the I/O pins may not be configured for appropriate electrical standards for chip-to-chip communication. Programming these I/O standards via JTAG allows you to fully test the I/O connections to other devices.

A device operating in JTAG mode uses four required pins, TDI, TDO, TMS, and TCK, and one optional pin, TRST. The TCK pin has an internal weak pull-down resistor, while the TDI, TMS, and TRST pins have weak internal pull-up resistors. The JTAG input pins are powered by the 3.3-V $V_{CCPD}$ pins. The TDO output pin is powered by the $V_{CCIO}$ power supply in I/O bank 4.

Arria GX devices also use the JTAG port to monitor the logic operation of the device with the SignalTap® II embedded logic analyzer. Arria GX devices support the JTAG instructions shown in Table 3–1.

☞ Arria GX, Cyclone® II, Cyclone, Stratix®, Stratix II, Stratix GX , and Stratix II GX devices must be within the first 17 devices in a JTAG chain. All of these devices have the same JTAG controller. If any of the Stratix, Arria GX, Cyclone, and Cyclone II devices are in the 18th or further position, they will fail configuration. This does not affect the functionality of the SignalTap® II embedded logic analyzer.

**Table 3–1.** Arria GX JTAG Instructions

| JTAG Instruction | Instruction Code | Description |
| --- | --- | --- |
| SAMPLE/PRELOAD | 00 0000 0101 | Allows a snapshot of signals at the device pins to be captured and examined during normal device operation and permits an initial data pattern to be output at the device pins. Also used by the SignalTap II embedded logic analyzer. |
| EXTEST *(1)* | 00 0000 1111 | Allows external circuitry and board-level interconnects to be tested by forcing a test pattern at the output pins and capturing test results at the input pins. |
| BYPASS | 11 1111 1111 | Places the 1-bit bypass register between the TDI and TDO pins, which allows the BST data to pass synchronously through selected devices to adjacent devices during normal device operation. |
| USERCODE | 00 0000 0111 | Selects the 32-bit USERCODE register and places it between the TDI and TDO pins, allowing the USERCODE to be serially shifted out of TDO. |
| IDCODE | 00 0000 0110 | Selects the IDCODE register and places it between TDI and TDO, allowing IDCODE to be serially shifted out of TDO. |
| HIGHZ *(1)* | 00 0000 1011 | Places the 1-bit bypass register between the TDI and TDO pins, which allows the BST data to pass synchronously through selected devices to adjacent devices during normal device operation, while tri-stating all of the I/O pins. |
| CLAMP *(1)* | 00 0000 1010 | Places the 1-bit bypass register between the TDI and TDO pins, which allows the BST data to pass synchronously through selected devices to adjacent devices during normal device operation while holding I/O pins to a state defined by the data in the boundary-scan register. |
| ICR instructions | — | Used when configuring an Arria GX device via the JTAG port with a USB-Blaster™, MasterBlaster™, ByteBlasterMV™, EthernetBlaster™, or ByteBlaster II download cable, or when using a **.jam** or **.jbc** via an embedded processor or JRunner™. |
| PULSE_NCONFIG | 00 0000 0001 | Emulates pulsing the nCONFIG pin low to trigger reconfiguration even though the physical pin is unaffected. |
| CONFIG_IO *(2)* | 00 0000 1101 | Allows configuration of I/O standards through the JTAG chain for JTAG testing. Can be executed before, during, or after configuration. Stops configuration if executed during configuration. Once issued, the CONFIG_IO instruction holds nSTATUS low to reset the configuration device. nSTATUS is held low until the IOE configuration register is loaded and the TAP controller state machine transitions to the UPDATE_DR state. |

**Notes to Table 3–1:**

(1) Bus hold and weak pull-up resistor features override the high-impedance state of HIGHZ, CLAMP, and EXTEST.

(2) For more information about using the CONFIG_IO instruction, refer to the *MorphIO: An I/O Reconfiguration Solution for Altera Devices* White Paper.

The Arria GX device instruction register length is 10 bits and the USERCODE register length is 32 bits. Table 3–2 and Table 3–3 show the boundary-scan register length and device IDCODE information for Arria GX devices.

**Table 3–2.** Arria GX Boundary-Scan Register Length

| Device | Boundary-Scan Register Length |
|--------|-------------------------------|
| EP1AGX20 | 1320 |
| EP1AGX35 | 1320 |
| EP1AGX50 | 1668 |
| EP1AGX60 | 1668 |
| EP1AGX90 | 2016 |

**Table 3–3.** 2-Bit Arria GX Device IDCODE

| Device | IDCODE (32 Bits) | | | |
|--------|------------------|---|---|---|
| | Version (4 Bits) | Part Number (16 Bits) | Manufacturer Identity (11 Bits) | LSB (1 Bit) |
| EP1AGX20 | 0000 | 0010 0001 0010 0001 | 000 0110 1110 | 1 |
| EP1AGX35 | 0000 | 0010 0001 0010 0001 | 000 0110 1110 | 1 |
| EP1AGX50 | 0000 | 0010 0001 0010 0010 | 000 0110 1110 | 1 |
| EP1AGX60 | 0000 | 0010 0001 0010 0010 | 000 0110 1110 | 1 |
| EP1AGX90 | 0000 | 0010 0001 0010 0011 | 000 0110 1110 | 1 |

# SignalTap II Embedded Logic Analyzer

Arria GX devices feature the SignalTap II embedded logic analyzer, which monitors design operation over a period of time through the IEEE Std. 1149.1 (JTAG) circuitry. You can analyze internal logic at speed without bringing internal signals to the I/O pins. This feature is particularly important for advanced packages, such as FineLine BGA (FBGA) packages, because it can be difficult to add a connection to a pin during the debugging process after a board is designed and manufactured.

# Configuration

The logic, circuitry, and interconnects in the Arria GX architecture are configured with CMOS SRAM elements. Altera® FPGAs are reconfigurable and every device is tested with a high coverage production test program so you do not have to perform fault testing and can instead focus on simulation and design verification.

Arria GX devices are configured at system power up with data stored in an Altera configuration device or provided by an external controller (for example, a MAX® II device or microprocessor). You can configure Arria GX devices using the fast passive parallel (FPP), active serial (AS), passive serial (PS), passive parallel asynchronous (PPA), and JTAG configuration schemes. Each Arria GX device has an optimized interface that allows microprocessors to configure it serially or in parallel, and synchronously or asynchronously. The interface also enables microprocessors to treat Arria GX devices as memory and configure them by writing to a virtual memory location, making reconfiguration easy.

In addition to the number of configuration methods supported, Arria GX devices also offer decompression and remote system upgrade features. The decompression feature allows Arria GX FPGAs to receive a compressed configuration bitstream and decompress this data in real-time, reducing storage requirements and configuration time. The remote system upgrade feature allows real-time system upgrades from remote locations of Arria GX designs. For more information, refer to "Configuration Schemes" on page 3–5.

## Operating Modes

The Arria GX architecture uses SRAM configuration elements that require configuration data to be loaded each time the circuit powers up. The process of physically loading the SRAM data into the device is called configuration. During initialization, which occurs immediately after configuration, the device resets registers, enables I/O pins, and begins to operate as a logic device. The I/O pins are tri-stated during power up, and before and during configuration. Together, the configuration and initialization processes are called command mode. Normal device operation is called user mode.

SRAM configuration elements allow you to reconfigure Arria GX devices in-circuit by loading new configuration data into the device. With real-time reconfiguration, the device is forced into command mode with a device pin. The configuration process loads different configuration data, re-initializes the device, and resumes user-mode operation. You can perform in-field upgrades by distributing new configuration files either within the system or remotely.

`PORSEL` is a dedicated input pin used to select power-on reset (POR) delay times of 12 ms or 100 ms during power up. When the `PORSEL` pin is connected to ground, the POR time is 100 ms. When the `PORSEL` pin is connected to $V_{CC}$, the POR time is 12 ms.

The `nIO_PULLUP` pin is a dedicated input that chooses whether the internal pull-up resistors on the user I/O pins and dual-purpose configuration I/O pins (`nCSO`, `ASDO`, `DATA[7..0]`, `nWS`, `nRS`, `RDYnBSY`, `nCS`, `CS`, `RUnLU`, `PGM[2..0]`, `CLKUSR`, `INIT_DONE`, `DEV_OE`, `DEV_CLR`) are on or off before and during configuration. A logic high (1.5, 1.8, 2.5, 3.3 V) turns off the weak internal pull-up resistors, while a logic low turns them on.

Arria GX devices also offer a new power supply, $V_{CCPD}$, which must be connected to 3.3 V in order to power the 3.3-V/2.5-V buffer available on the configuration input pins and JTAG pins. $V_{CCPD}$ applies to all the JTAG input pins (`TCK`, `TMS`, `TDI`, and `TRST`) and the following configuration pins: `nCONFIG`, `DCLK` (when used as an input), `nIO_PULLUP`, `DATA[7..0]`, `RUnLU`, `nCE`, `nWS`, `nRS`, `CS`, `nCS`, and `CLKUSR`. The $V_{CCSEL}$ pin allows the $V_{CCIO}$ setting (of the banks where the configuration inputs reside) to be independent of the voltage required by the configuration inputs. Therefore, when selecting the $V_{CCIO}$ voltage, you do not have to take the VIL and VIH levels driven to the configuration inputs into consideration. The configuration input pins, `nCONFIG`, `DCLK` (when used as an input), `nIO_PULLUP`, `RUnLU`, `nCE`, `nWS`, `nRS`, `CS`, `nCS`, and `CLKUSR`, have a dual buffer design: a 3.3-V/2.5-V input buffer and a 1.8-V/1.5-V input buffer. The $V_{CCSEL}$ input pin selects which input buffer is used. The 3.3-V/2.5-V input buffer is powered by $V_{CCPD}$, while the 1.8-V/1.5-V input buffer is powered by $V_{CCIO}$.

$V_{CCSEL}$ is sampled during power up. Therefore, the $V_{CCSEL}$ setting cannot change on-the-fly or during a reconfiguration. The $V_{CCSEL}$ input buffer is powered by $V_{CCINT}$ and must be hard-wired to $V_{CCPD}$ or ground. A logic high $V_{CCSEL}$ connection selects the 1.8-V/1.5-V input buffer, and a logic low selects the 3.3-V/2.5-V input buffer. $V_{CCSEL}$ should be set to comply with the logic levels driven out of the configuration device or MAX II microprocessor.

If the design must support configuration input voltages of 3.3 V/2.5 V, set $V_{CCSEL}$ to a logic low. You can set the $V_{CCIO}$ voltage of the I/O bank that contains the configuration inputs to any supported voltage. If the design must support configuration input voltages of 1.8 V/1.5 V, set $V_{CCSEL}$ to a logic high and the $V_{CCIO}$ of the bank that contains the configuration inputs to 1.8 V/1.5 V.

For more information about multi-volt support, including information about using `TDO` and `nCEO` in multi-volt systems, refer to the *Arria GX Architecture* chapter.

## Configuration Schemes

You can load the configuration data for an Arria GX device with one of five configuration schemes (refer to Table 3–4), chosen on the basis of the target application. You can use a configuration device, intelligent controller, or the JTAG port to configure an Arria GX device. A configuration device can automatically configure an Arria GX device at system power up.

You can configure multiple Arria GX devices in any of the five configuration schemes by connecting the configuration enable (`nCE`) and configuration enable output (`nCEO`) pins on each device. Arria GX FPGAs offer the following:

■ Configuration data decompression to reduce configuration file storage

■ Remote system upgrades for remotely updating Arria GX designs

Table 3–4 lists which configuration features can be used in each configuration scheme.

For more information about configuration schemes in Arria GX devices, refer to the *Configuring Arria GX Devices* chapter.

**Table 3–4.** Arria GX Configuration Features   (Part 1 of 2)

| Configuration Scheme | Configuration Method | Decompression | Remote System Upgrade |
|---|---|---|---|
| FPP | MAX II device or microprocessor and flash device | ✓ (1) | ✓ |
| | Enhanced configuration device | ✓ (2) | ✓ |
| AS | Serial configuration device | ✓ | ✓ (3) |
| PS | MAX II device or microprocessor and flash device | ✓ | ✓ |
| | Enhanced configuration device | ✓ | ✓ |
| | Download cable (4) | ✓ | — |
| PPA | MAX II device or microprocessor and flash device | — | ✓ |

**Table 3–4.** Arria GX Configuration Features   (Part 2 of 2)

| Configuration Scheme | Configuration Method | Decompression | Remote System Upgrade |
|---|---|---|---|
| JTAG | Download cable *(4)* | — | — |
|  | MAX II device or microprocessor and flash device | — | — |

Notes for Table 3–4:

(1)   In these modes, the host system must send a `DCLK` that is 4× the data rate.

(2)   The enhanced configuration device decompression feature is available, while the Arria GX decompression feature is not available.

(3)   Only remote update mode is supported when using the AS configuration scheme. Local update mode is not supported.

(4)   The supported download cables include the Altera USB-Blaster universal serial bus (USB) port download cable, MasterBlaster™ serial/USB communications cable, ByteBlaster II parallel port download cable, ByteBlasterMV parallel port download cable, and the EthernetBlaster download cable.

## Device Configuration Data Decompression

Arria GX FPGAs support decompression of configuration data, which saves configuration memory space and time. This feature allows you to store compressed configuration data in configuration devices or other memory and transmit this compressed bitstream to Arria GX FPGAs. During configuration, the Arria GX FPGA decompresses the bitstream in real time and programs its SRAM cells. Arria GX FPGAs support decompression in the FPP (when using a MAX II device or microprocessor and flash memory), AS, and PS configuration schemes. Decompression is not supported in the PPA configuration scheme nor in JTAG-based configuration.

## Remote System Upgrades

Shortened design cycles, evolving standards, and system deployments in remote locations are difficult challenges faced by system designers. Arria GX devices can help effectively deal with these challenges with their inherent re programmability and dedicated circuitry to perform remote system updates. Remote system updates help deliver feature enhancements and bug fixes without costly recalls, reduce time to market, and extend product life.

Arria GX FPGAs feature dedicated remote system upgrade circuitry to facilitate remote system updates. Soft logic (Nios® processor or user logic) implemented in the Arria GX device can download a new configuration image from a remote location, store it in configuration memory, and direct the dedicated remote system upgrade circuitry to initiate a reconfiguration cycle. The dedicated circuitry performs error detection during and after the configuration process, recovers from any error condition by reverting back to a safe configuration image, and provides error status information. This dedicated remote system upgrade circuitry avoids system downtime and is the critical component for successful remote system upgrades.

Remote system configuration is supported in the following Arria GX configuration schemes: FPP, AS, PS, and PPA. You can also implement remote system configuration in conjunction with Arria GX features such as real-time decompression of configuration data for efficient field upgrades.

For more information about remote configuration in Arria GX devices, refer to the *Remote System Upgrades with Arria GX Devices* chapter.

## Configuring Arria GX FPGAs with JRunner

The JRunner software driver configures Altera FPGAs, including Arria GX FPGAs, through the ByteBlaster™ II or ByteBlasterMV cables in JTAG mode. The programming input file supported is in Raw Binary File (**.rbf**) format. JRunner also requires a Chain Description File (**.cdf**) generated by the Quartus II software. JRunner is targeted for embedded JTAG configuration. The source code is developed for the Windows NT operating system (OS), but can be customized to run on other platforms.

For more information about the JRunner software driver, refer to the *AN414: JRunner Software Driver: An Embedded Solution for PLD JTAG Configuration* and the source files on the Altera website.

## Programming Serial Configuration Devices with SRunner

You can program a serial configuration device in-system by an external microprocessor using SRunner™. SRunner is a software driver developed for embedded serial configuration device programming that can be easily customized to fit into different embedded systems. SRunner software driver reads a raw programming data file (**.rpd**) and writes to serial configuration devices. The serial configuration device programming time using SRunner software driver is comparable to the programming time when using the Quartus II software.

For more information about SRunner, refer to the *AN418: SRunner: An Embedded Solution for Serial Configuration Device Programming* and the source code on the Altera website.

For more information about programming serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS64, and EPCS128) Data Sheet* in the *Configuration Handbook*.

## Configuring Arria GX FPGAs with the MicroBlaster Driver

The MicroBlaster™ software driver supports a raw binary file (RBF) programming input file and is ideal for embedded FPP or PS configuration. The source code is developed for the Windows NT operating system, although it can be customized to run on other operating systems.

For more information about the MicroBlaster software driver, refer to the *Configuring the MicroBlaster Fast Passive Parallel Software Driver* White Paper or the *AN423: Configuring the MicroBlaster Passive Serial Software Driver*.

## PLL Reconfiguration

The phase-locked loops (PLLs) in the Arria GX device family support reconfiguration of their multiply, divide, VCO-phase selection, and bandwidth selection settings without reconfiguring the entire device. You can use either serial data from the logic array or regular I/O pins to program the PLL's counter settings in a serial chain. This option provides considerable flexibility for frequency synthesis, allowing real-time variation of the PLL frequency and delay. The rest of the device is functional while reconfiguring the PLL.

For more information about Arria GX PLLs, refer to the *PLLs in Arria GX Devices* chapter.

# Automated Single Event Upset (SEU) Detection

Arria GX devices offer on-chip circuitry for automated checking of single event upset (SEU) detection. Some applications that require the device to operate error free at high elevations or in close proximity to Earth's North or South Pole requires periodic checks to ensure continued data integrity. The error detection cyclic redundancy check (CRC) feature controlled by the **Device and Pin Options** dialog box in the Quartus II software uses a 32-bit CRC circuit to ensure data reliability and is one of the best options for mitigating SEU.

You can implement the error detection CRC feature with existing circuitry in Arria GX devices, eliminating the need for external logic. Arria GX devices compute CRC during configuration. The Arria GX device checks the computed-CRC against an automatically computed CRC during normal operation. The CRC_ERROR pin reports a soft error when configuration SRAM data is corrupted, triggering device reconfiguration.

## Custom-Built Circuitry

Dedicated circuitry is built into Arria GX devices to automatically perform error detection. This circuitry constantly checks for errors in the configuration SRAM cells while the device is in user mode. You can monitor one external pin for the error and use it to trigger a reconfiguration cycle. You can select the desired time between checks by adjusting a built-in clock divider.

## Software Interface

Beginning with version 7.1 of the Quartus II software, you can turn on the automated error detection CRC feature in the **Device and Pin Options** dialog box. This dialog box allows you to enable the feature and set the internal frequency of the CRC between 400 kHz to 50 MHz. This controls the rate that the CRC circuitry verifies the internal configuration SRAM bits in the Arria GX FPGA.

For more information about CRC, refer to *AN 357: Error Detection Using CRC in Altera FPGAs*.

# Document Revision History

Table 3–5 lists the revision history for this chapter.

**Table 3–5.** Document Revision History

| Date and Document Version | Changes Made | Summary of Changes |
|---|---|---|
| December 2009, v2.0 | ■ Document template update.<br>■ Minor text edits. | — |
| May 2009<br>v1.4 | ■ Removed "Temperature Sensing Diode" section.<br>■ Updated Table 3–1 and Table 3–4. | — |
| May 2008<br>v1.3 | Updated note in "Introduction" section. | |
| | Minor text edits. | — |
| August 2007<br>v1.2 | Added the "Referenced Documents" section. | — |
| June 2007<br>v1.1 | Deleted Signal Tap II information from Table 3–1. | — |
| May 2007<br>v1.0 | Initial Release | — |

# Operating Conditions

Arria® GX devices are offered in both commercial and industrial grades. Both commercial and industrial devices are offered in –6 speed grade only.

This chapter contains the following sections:

- "Operating Conditions"
- "Power Consumption" on page 4–25
- "I/O Timing Model" on page 4–26
- "Typical Design Performance" on page 4–32
- "Block Performance" on page 4–84
- "IOE Programmable Delay" on page 4–86
- "Maximum Input and Output Clock Toggle Rate" on page 4–87
- "Duty Cycle Distortion" on page 4–95
- "High-Speed I/O Specifications" on page 4–100
- "PLL Timing Specifications" on page 4–103
- "External Memory Interface Specifications" on page 4–105
- "JTAG Timing Specifications" on page 4–106

Table 4–1 through Table 4–42 on page 4–25 provide information on absolute maximum ratings, recommended operating conditions, DC electrical characteristics, and other specifications for Arria GX devices.

## Absolute Maximum Ratings

Table 4–1 contains the absolute maximum ratings for the Arria GX device family.

**Table 4–1.** Arria GX Device Absolute Maximum Ratings    *(Note 1)*, *(2)*, *(3)* (Part 1 of 2)

| Symbol | Parameter | Conditions | Minimum | Maximum | Units |
|--------|-----------|------------|---------|---------|-------|
| $V_{CCINT}$ | Supply voltage | With respect to ground | –0.5 | 1.8 | V |
| $V_{CCIO}$ | Supply voltage | With respect to ground | –0.5 | 4.6 | V |
| $V_{CCPD}$ | Supply voltage | With respect to ground | –0.5 | 4.6 | V |
| $V_I$ | DC input voltage *(4)* | — | –0.5 | 4.6 | V |
| $I_{OUT}$ | DC output current, per pin | — | –25 | 40 | mA |
| $T_{STG}$ | Storage temperature | No bias | –65 | 150 | C |

**Table 4–1.** Arria GX Device Absolute Maximum Ratings    *(Note 1)*, *(2)*, *(3)* (Part 2 of 2)

| Symbol | Parameter | Conditions | Minimum | Maximum | Units |
|--------|-----------|------------|---------|---------|-------|
| $T_J$ | Junction temperature | BGA packages under bias | −55 | 125 | C |

**Notes to Table 4–1:**

(1) For more information about operating requirements for Altera® devices, refer to the *Arria GX Device Family Data Sheet* chapter.

(2) Conditions beyond those listed in Table 4–1 may cause permanent damage to a device. Additionally, device operation at the absolute maximum ratings for extended periods of time may have adverse affects on the device.

(3) Supply voltage specifications apply to voltage readings taken at the device pins, not at the power supply.

(4) During transitions, the inputs may overshoot to the voltage shown in Table 4–2 based upon the input duty cycle. The DC case is equivalent to 100% duty cycle. During transitions, the inputs may undershoot to −2.0 V for input currents less than 100 mA and periods shorter than 20 ns.

**Table 4–2.** Maximum Duty Cycles in Voltage Transitions    *(Note 1)*

| Symbol | Parameter | Condition | Maximum Duty Cycles (%) |
|--------|-----------|-----------|-------------------------|
| $V_I$ | Maximum duty cycles in voltage transitions | $V_I$ = 4.0 V | 100 |
|  |  | $V_I$ = 4.1 V | 90 |
|  |  | $V_I$ = 4.2 V | 50 |
|  |  | $V_I$ = 4.3 V | 30 |
|  |  | $V_I$ = 4.4 V | 17 |
|  |  | $V_I$ = 4.5 V | 10 |

**Note to Table 4–2:**

(1) During transition, the inputs may overshoot to the voltages shown based on the input duty cycle. The DC case is equivalent to 100% duty cycle.

## Recommended Operating Conditions

Table 4–3 lists the recommended operating conditions for the Arria GX device family.

**Table 4–3.** Arria GX Device Recommended Operating Conditions  (Part 1 of 2)    *(Note 1)* (Part 1 of 2)

| Symbol | Parameter | Conditions | Minimum | Maximum | Units |
|--------|-----------|------------|---------|---------|-------|
| $V_{CCINT}$ | Supply voltage for internal logic and input buffers | Rise time ≤ 100 ms *(3)* | 1.15 | 1.25 | V |
| $V_{CCIO}$ | Supply voltage for output buffers, 3.3-V operation | Rise time ≤ 100 ms *(3)*, *(6)* | 3.135 (3.00) | 3.465 (3.60) | V |
|  | Supply voltage for output buffers, 2.5-V operation | Rise time ≤ 100 ms *(3)* | 2.375 | 2.625 | V |
|  | Supply voltage for output buffers, 1.8-V operation | Rise time ≤ 100 ms *(3)* | 1.71 | 1.89 | V |
|  | Supply voltage for output buffers, 1.5-V operation | Rise time ≤ 100 ms *(3)* | 1.425 | 1.575 | V |
|  | Supply voltage for output buffers, 1.2-V operation | Rise time ≤ 100 ms *(3)* | 1.15 | 1.25 | V |
| $V_{CCPD}$ | Supply voltage for pre-drivers as well as configuration and JTAG I/O buffers. | 100 μs ≤ rise time ≤ 100 ms *(4)* | 3.135 | 3.465 | V |
| $V_I$ | Input voltage (refer to Table 4–2) | *(2)*, *(5)* | −0.5 | 4.0 | V |
| $V_O$ | Output voltage | — | 0 | $V_{CCIO}$ | V |

**Table 4–3.** Arria GX Device Recommended Operating Conditions  (Part 2 of 2)    *(Note 1)*  (Part 2 of 2)

| Symbol | Parameter | Conditions | Minimum | Maximum | Units |
|--------|-----------|------------|---------|---------|-------|
| $T_J$ | Operating junction temperature | For commercial use | 0 | 85 | C |
| | | For industrial use | −40 | 100 | C |

**Notes to Table 4–3:**

(1)  Supply voltage specifications apply to voltage readings taken at the device pins, not at the power supply.

(2)  During transitions, the inputs may overshoot to the voltage shown in Table 4–2 based upon the input duty cycle. The DC case is equivalent to 100% duty cycle. During transitions, the inputs may undershoot to −2.0 V for input currents less than 100 mA and periods shorter than 20 ns.

(3)  Maximum $V_{CC}$ rise time is 100 ms, and $V_{CC}$ must rise monotonically from ground to $V_{CC}$.

(4)  $V_{CCPD}$ must ramp-up from 0 V to 3.3 V within 100 μs to 100 ms. If $V_{CCPD}$ is not ramped up within this specified time, the Arria GX device will not configure successfully. If the system does not allow for a $V_{CCPD}$ ramp-up time of 100 ms or less, hold `nCONFIG` low until all power supplies are reliable.

(5)  All pins, including dedicated inputs, clock, I/O, and JTAG pins, can be driven before $V_{CCINT}$, $V_{CCPD}$, and $V_{CCIO}$ are powered.

(6)  $V_{CCIO}$ maximum and minimum conditions for PCI and PCI-X are shown in parentheses.

## Transceiver Block Characteristics

Table 4–4 through Table 4–6 on page 4–4 contain transceiver block specifications.

**Table 4–4.** Arria GX Transceiver Block Absolute Maximum Ratings     *(Note 1)*

| Symbol | Parameter | Conditions | Minimum | Maximum | Units |
|--------|-----------|------------|---------|---------|-------|
| $V_{CCA}$ | Transceiver block supply voltage | Commercial and industrial | −0.5 | 4.6 | V |
| $V_{CCP}$ | Transceiver block supply voltage | Commercial and industrial | −0.5 | 1.8 | V |
| $V_{CCR}$ | Transceiver block supply voltage | Commercial and industrial | −0.5 | 1.8 | V |
| $V_{CCT\_B}$ | Transceiver block supply voltage | Commercial and industrial | −0.5 | 1.8 | V |
| $V_{CCL\_B}$ | Transceiver block supply voltage | Commercial and industrial | −0.5 | 1.8 | V |
| $V_{CCH\_B}$ | Transceiver block supply voltage | Commercial and industrial | −0.5 | 2.4 | V |

**Note to Table 4–4:**

(1)  The device can tolerate prolonged operation at this absolute maximum, as long as the maximum specification is not violated.

**Table 4–5.** Arria GX Transceiver Block Operating Conditions

| Symbol | Parameter | Conditions | Minimum | Typical | Maximum | Units |
|--------|-----------|------------|---------|---------|---------|-------|
| $V_{CCA}$ | Transceiver block supply voltage | Commercial and industrial | 3.135 | 3.3 | 3.465 | V |
| $V_{CCP}$ | Transceiver block supply voltage | Commercial and industrial | 1.15 | 1.2 | 1.25 | V |
| $V_{CCR}$ | Transceiver block supply voltage | Commercial and industrial | 1.15 | 1.2 | 1.25 | V |
| $V_{CCT\_B}$ | Transceiver block supply voltage | Commercial and industrial | 1.15 | 1.2 | 1.25 | V |
| $V_{CCL\_B}$ | Transceiver block supply voltage | Commercial and industrial | 1.15 | 1.2 | 1.25 | V |
| $V_{CCH\_B}$ | Transceiver block supply voltage | Commercial and industrial | 1.15 | 1.2 | 1.25 | V |
| | | | 1.425 | 1.5 | 1.575 | V |
| $R_{REFB}$ *(1)* | Reference resistor | Commercial and industrial | 2K - 1% | 2K | 2K +1% | Ω |

**Note to Table 4–5:**

(1)  The DC signal on this pin must be as clean as possible. Ensure that no noise is coupled to this pin.

**Table 4–6.** Arria GX Transceiver Block AC Specification  (Part 1 of 3)

| Symbol / Description | Conditions | –6 Speed Grade Commercial and Industrial | | | Units |
|---|---|---|---|---|---|
| | | **Min** | **Typ** | **Max** | |
| **Reference clock** | | | | | |
| Input reference clock frequency | — | 50 | — | 622.08 | MHz |
| Absolute $V_{MAX}$ for a `REFCLK` Pin | — | — | — | 3.3 | V |
| Absolute $V_{MIN}$ for a `REFCLK` Pin | — | −0.3 | — | — | V |
| Rise/Fall time | — | — | 0.2 | — | UI |
| Duty cycle | — | 45 | — | 55 | % |
| Peak to peak differential input voltage $V_{ID}$ (diff p-p) | — | 200 | — | 2000 | mV |
| Spread spectrum clocking (1) | 0 to −0.5% | 30 | — | 33 | kHz |
| On-chip termination resistors | — | 115 ± 20% | | | Ω |
| $V_{ICM}$ (AC coupled) | — | 1200 ± 5% | | | mV |
| $V_{ICM}$ (DC coupled) (2) | PCI Express (PIPE) mode | 0.25 | — | 0.55 | V |
| **RREFB** | — | 2000 +/-1% | | | Ω |
| **Transceiver Clocks** | | | | | |
| Calibration block clock frequency | — | 10 | — | 125 | MHz |
| Calibration block minimum power-down pulse width | — | 30 | — | — | ns |
| `fixedclk` clock frequency (3) | — | 125 ±10% | | | MHz |
| `reconfig` clock frequency | SDI mode | 2.5 | — | 50 | MHz |
| Transceiver block minimum power-down pulse width | — | 100 | — | — | ns |
| **Receiver** | | | | | |
| Data rate | — | 600 | — | 3125 | Mbps |
| Absolute $V_{MAX}$ for a receiver pin (4) | — | — | — | 2.0 | V |
| Absolute $V_{MIN}$ for a receiver pin | — | −0.4 | — | — | V |
| Maximum peak-to-peak differential input voltage $V_{ID}$ (diff p-p) | Vicm = 0.85 V | — | — | 3.3 | V |
| Minimum peak-to-peak differential input voltage $V_{ID}$ (diff p-p) | DC Gain = 3 dB | 160 | — | — | mV |
| On-chip termination resistors | — | 100±15% | | | Ω |
| $V_{ICM}$ (15) | Vicm = 0.85 V setting | 850 ± 10% | 850 ± 10% | 850 ± 10% | mV |
| | Vicm = 1.2 V setting | 1200 ± 10% | 1200 ± 10% | 1200 ± 10% | mV |
| Bandwidth at 3.125 Gbps | BW = Low | — | 30 | — | MHz |
| | BW = Med | — | 40 | — | |
| | BW = High | — | 50 | — | |

**Table 4–6.** Arria GX Transceiver Block AC Specification  (Part 2 of 3)

| Symbol / Description | Conditions | –6 Speed Grade Commercial and Industrial | | | Units |
|---|---|---|---|---|---|
| | | Min | Typ | Max | |
| Bandwidth at 2.5 Gbps | BW = Low | — | 35 | — | MHz |
| | BW = Med | — | 50 | — | |
| | BW = High | — | 60 | — | |
| Return loss differential mode | 50 MHz to 1.25 GHz (PCI Express) | –10 | | | dB |
| | 100 MHz to 2.5 GHz (XAUI) | | | | |
| Return loss common mode | 50 MHz to 1.25 GHz (PCI Express) | –6 | | | dB |
| | 100 MHz to 2.5 GHz (XAUI) | | | | |
| Programmable PPM detector *(5)* | — | ± 62.5, 100, 125, 200, 250, 300, 500, 1000 | | | PPM |
| Run length *(6)* | — | 80 | | | UI |
| Programmable equalization | — | — | — | 5 | dB |
| Signal detect/loss threshold *(7)* | — | 65 | — | 175 | mV |
| CDR LTR TIme *(8)*, *(9)* | — | — | — | 75 | us |
| CDR Minimum T1b *(9)*, *(10)* | — | 15 | — | — | us |
| LTD lock time *(9)*, *(11)* | — | 0 | 100 | 4000 | ns |
| Data lock time from `rx_freqlocked` *(9)*, *(12)* | — | — | — | 4 | us |
| Programmable DC gain | — | 0, 3, 6 | | | dB |
| **Transmitter Buffer** | | | | | |
| Output Common Mode voltage (V$_{ocm}$) | — | 580 ± 10% | | | mV |
| On-chip termination resistors | — | 108±10% | | | Ω |
| Return loss differential mode | 50 MHz to 1.25 GHz (PCI Express) | –10 | | | dB |
| | 312 MHz to 625 MHz (XAUI) | | | | |
| | 625 MHz to 3.125GHz (XAUI) | –10 | | | _dB_ / _decade slope_ |
| Return loss common mode | 50 MHz to 1.25 GHz (PCI Express) | –6 | | | dB |
| Rise time | — | 35 | — | 65 | ps |
| Fall time | — | 35 | — | 65 | ps |
| Intra differential pair skew | V$_{OD}$ = 800 mV | — | — | 15 | ps |
| Intra-transceiver block skew (×4) *(13)* | — | — | — | 100 | ps |

**Table 4–6.** Arria GX Transceiver Block AC Specification  (Part 3 of 3)

| Symbol / Description | Conditions | –6 Speed Grade Commercial and Industrial | | | Units |
|---|---|---|---|---|---|
| | | Min | Typ | Max | |
| **Transmitter PLL** | | | | | |
| VCO frequency range | — | 500 | — | 1562.5 | MHz |
| Bandwidth at 3.125 Gbps | BW = Low | — | 3 | — | MHz |
| | BW = Med | — | 5 | — | |
| | BW = High | — | 9 | — | |
| Bandwidth at 2.5 Gbps | BW = Low | — | 1 | — | MHz |
| | BW = Med | — | 2 | — | |
| | BW = High | — | 4 | — | |
| TX PLL lock time from `gxb_powerdown` de-assertion *(9)*, *(14)* | — | — | — | 100 | us |
| **PCS** | | | | | |
| Interface speed per mode | — | 25 | — | 156.25 | MHz |
| Digital Reset Pulse Width | — | Minimum is 2 parallel clock cycles | | | — |

**Notes to Table 4–6:**

(1)  Spread spectrum clocking is allowed only in PCI Express (PIPE) mode if the upstream transmitter and the receiver share the same clock source.

(2)  The reference clock DC coupling option is only available in PCI Express (PIPE) mode for the HCSL I/O standard.

(3)  The `fixedclk` is used in PIPE mode receiver detect circuitry.

(4)  The device cannot tolerate prolonged operation at this absolute maximum.

(5)  The rate matcher supports only up to ± 300 PPM for PIPE mode and ± 100 PPM for GIGE mode.

(6)  This parameter is measured by embedding the run length data in a PRBS sequence.

(7)  Signal detect threshold detector circuitry is available only in PCI Express (PIPE mode).

(8)  Time taken for `rx_pll_locked` to go high from `rx_analogreset` deassertion. Refer to Figure 4–1.

(9)  For lock times specific to the protocols, refer to protocol characterization documents.

(10)  Time for which the CDR needs to stay in LTR mode after `rx_pll_locked` is asserted and before `rx_locktodata` is asserted in manual mode. Refer to Figure 4–1.

(11)  Time taken to recover valid data from GXB after the `rx_locktodata` signal is asserted in manual mode. Measurement results are based on PRBS31, for native data rates only. Refer to Figure 4–1.

(12)  Time taken to recover valid data from GXB after the `rx_freqlocked` signal goes high in automatic mode. Measurement results are based on PRBS31, for native data rates only. Refer to Figure 4–2.

(13)  This is applicable only to PCI Express (PIPE) ×4 and XAUI ×4 mode.

(14)  Time taken to lock `TX PLL` from `gxb_powerdown` deassertion.

(15)  The 1.2 V RX VICM settings is intended for DC-coupled LVDS links.

Figure 4–1 shows the lock time parameters in manual mode. Figure 4–2 shows the lock time parameters in automatic mode.

☞ LTD = Lock to data
LTR = Lock to reference clock

**Figure 4–1.** Lock Time Parameters for Manual Mode
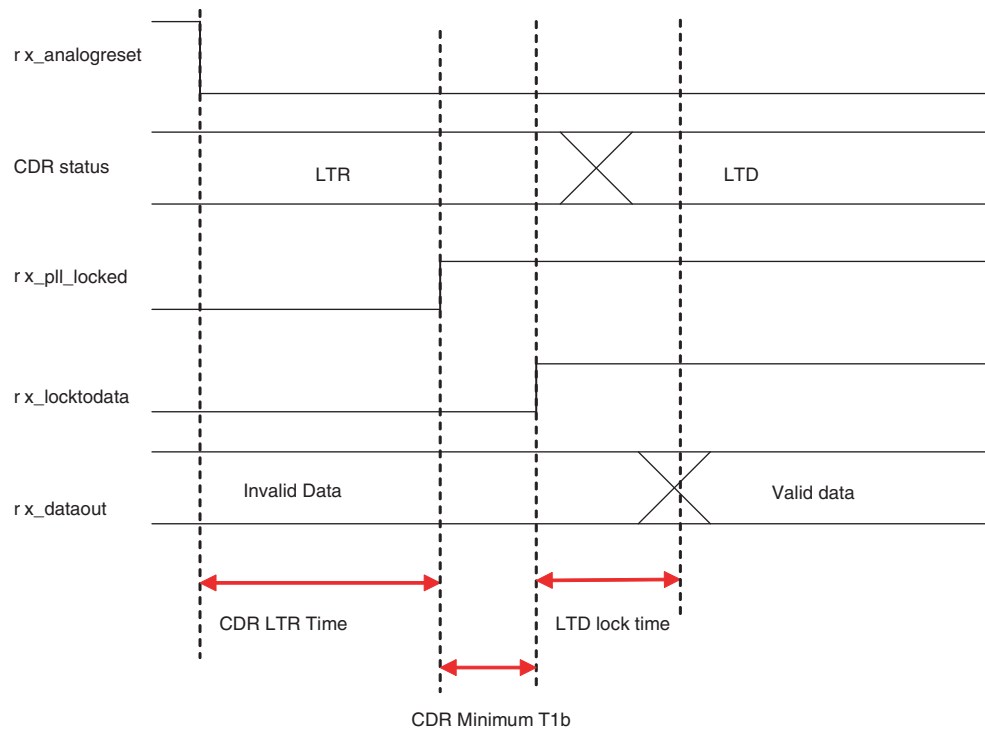


**Figure 4–2.** Lock Time Parameters for Automatic Mode
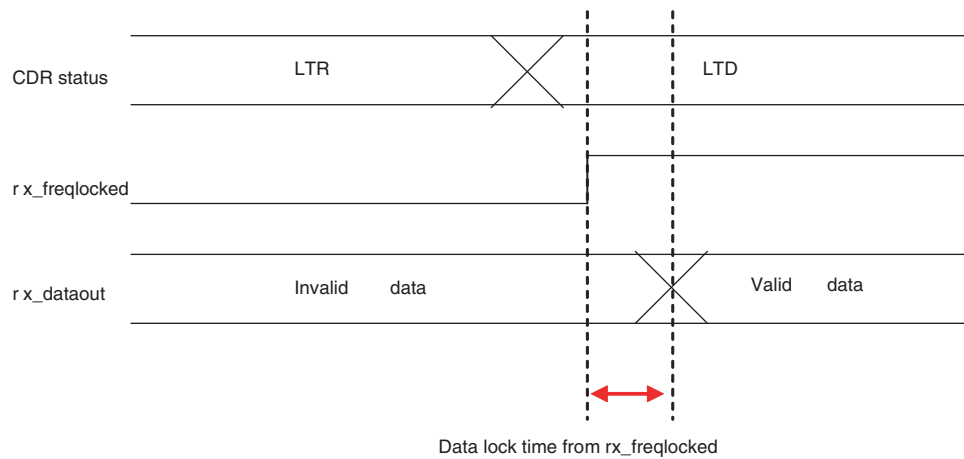
Figure 4–3 and Figure 4–4 show differential receiver input and transmitter output waveforms, respectively.

**Figure 4–3.** Receiver Input Waveform
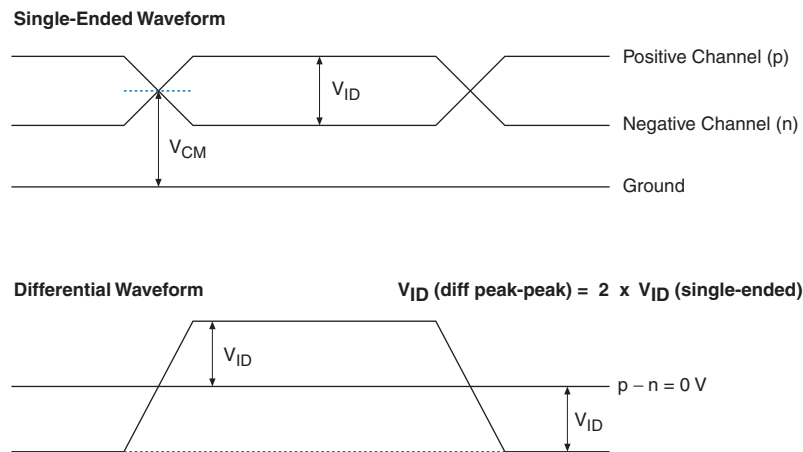


**Figure 4–4.** Transmitter Output Waveform
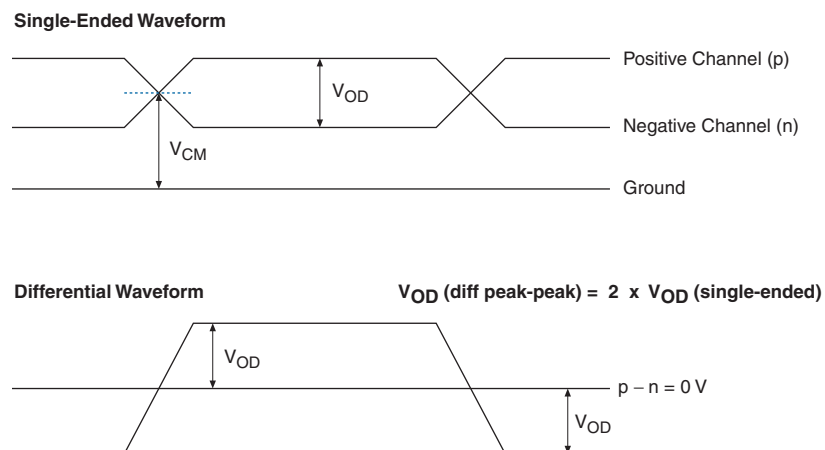


Table 4–7 lists the Arria GX transceiver block AC specification.

**Table 4–7.** Arria GX Transceiver Block AC Specification    *(Note 1)*, *(2)*, *(3)*  (Part 1 of 4)

| Description | Condition | −6 Speed Grade Commercial & Industrial | Units |
|---|---|---|---|
| **XAUI Transmit Jitter Generation** *(4)* | | | |
| Total jitter at 3.125 Gbps | REFCLK = 156.25 MHz<br>Pattern = CJPAT<br>$V_{OD}$ = 1200 mV<br>No Pre-emphasis | 0.3 | UI |

**Table 4–7.** Arria GX Transceiver Block AC Specification *(Note 1)*, *(2)*, *(3)* (Part 2 of 4)

| Description | Condition | –6 Speed Grade Commercial & Industrial | Units |
|---|---|---|---|
| Deterministic jitter at 3.125 Gbps | REFCLK = 156.25 MHz<br>Pattern = CJPAT<br>$V_{OD}$ = 1200 mV<br>No Pre-emphasis | 0.17 | UI |
| **XAUI Receiver Jitter Tolerance** *(4)* | | | |
| Total jitter | Pattern = CJPAT<br>No Equalization<br>DC Gain = 3 dB | > 0.65 | UI |
| Deterministic jitter | Pattern = CJPAT<br>No Equalization<br>DC Gain = 3 dB | > 0.37 | UI |
| Peak-to-peak jitter | Jitter frequency = 22.1 KHz | > 8.5 | UI |
| Peak-to-peak jitter | Jitter frequency = 1.875 MHz | > 0.1 | UI |
| Peak-to-peak jitter | Jitter frequency = 20 MHz | > 0.1 | UI |
| **PCI Express (PIPE) Transmitter Jitter Generation** *(5)* | | | |
| Total Transmitter Jitter Generation | Compliance Pattern; $V_{OD}$ = 800 mV;<br>Pre-emphasis = 49% | < 0.25 | UI p-p |
| **PCI Express (PIPE) Receiver Jitter Tolerance** *(5)* | | | |
| Total Receiver Jitter Tolerance | Compliance Pattern;<br>DC Gain = 3 db | > 0.6 | UI p-p |
| **Gigabit Ethernet (GIGE) Transmitter Jitter Generation** *(7)* | | | |
| Total Transmitter Jitter Generation (TJ) | CRPAT: $V_{OD}$ = 800 mV;<br>Pre-emphasis = 0% | < 0.279 | UI p-p |
| Deterministic Transmitter Jitter Generation (DJ) | CRPAT; $V_{OD}$ = 800 mV;<br>Pre-emphasis = 0% | < 0.14 | UI p-p |
| **Gigabit Ethernet (GIGE) Receiver Jitter Tolerance** | | | |
| Total Jitter Tolerance | CJPAT Compliance Pattern;<br>DC Gain = 0 dB | > 0.66 | UI p-p |
| Deterministic Jitter Tolerance | CJPAT Compliance Pattern;<br>DC Gain = 0 dB | > 0.4 | UI p-p |
| **Serial RapidIO (1.25 Gbps, 2.5 Gbps, and 3.125 Gbps) Transmitter Jitter Generation** *(6)* | | | |
| Total Transmitter Jitter Generation (TJ) | CJPAT Compliance Pattern;<br>$V_{OD}$ = 800 mV;<br>Pre-emphasis = 0% | < 0.35 | UI p-p |
| Deterministic Transmitter Jitter Generation (DJ) | CJPAT Compliance Pattern;<br>$V_{OD}$ = 800 mV;<br>Pre-emphasis = 0% | < 0.17 | UI p-p |

**Table 4–7.** Arria GX Transceiver Block AC Specification    *(Note 1)*, *(2)*, *(3)* (Part 3 of 4)

| Description | Condition | –6 Speed Grade Commercial & Industrial | Units |
|---|---|---|---|
| **Serial RapidIO (1.25 Gbps, 2.5 Gbps, and 3.125 Gbps) Receiver Jitter Tolerance** *(6)* | | | |
| Total Jitter Tolerance | CJPAT Compliance Pattern; DC Gain = 0 dB | > 0.65 | UI p-p |
| Combined Deterministic and Random Jitter Tolerance ($J_{DR}$) | CJPAT Compliance Pattern; DC Gain = 0 dB | > 0.55 | UI p-p |
| Deterministic Jitter Tolerance ($J_D$) | CJPAT Compliance Pattern; DC Gain = 0 dB | > 0.37 | UI p-p |
| Sinusoidal Jitter Tolerance | Jitter Frequency = 22.1 KHz | > 8.5 | UI p-p |
|  | Jitter Frequency = 200 KHz | > 1.0 | UI p-p |
|  | Jitter Frequency = 1.875 MHz | > 0.1 | UI p-p |
|  | Jitter Frequency = 20 MHz | > 0.1 | UI p-p |
| **SDI Transmitter Jitter Generation** *(8)* | | | |
| Alignment Jitter (peak-to-peak) | Data Rate = 1.485 Gbps (HD) REFCLK = 74.25 MHz Pattern = Color Bar Vod = 800 mV No Pre-emphasis Low-Frequency Roll-Off = 100 KHz | 0.2 | UIv |
|  | Data Rate = 2.97 Gbps (3G) REFCLK = 148.5 MHz Pattern = Color Bar Vod = 800 mV No Pre-emphasis Low-Frequency Roll-Off = 100 KHz | 0.3 | UI |

**Table 4–7.** Arria GX Transceiver Block AC Specification   *(Note 1)*, *(2)*, *(3)*  (Part 4 of 4)

| Description | Condition | –6 Speed Grade Commercial & Industrial | Units |
|---|---|---|---|
| **SDI Receiver Jitter Tolerance** *(8)* | | | |
| Sinusoidal Jitter Tolerance (peak-to-peak) | Jitter Frequency = 15 KHz<br>Data Rate = 2.97 Gbps (3G)<br>REFCLK = 148.5 MHz<br>Pattern = Single Line<br>Scramble Color Bar<br>No  Equalization<br>DC Gain = 0 dB | > 2 | UI |
| | Jitter Frequency = 100 KHz<br>Data Rate = 2.97 Gbps (3G)<br>REFCLK = 148.5 MHz<br>Pattern = Single Line Scramble Color Bar<br>No Equalization<br>DC Gain = 0 dB | > 0.3 | UI |
| | Jitter Frequency = 148.5 MHz<br>Data Rate = 2.97 Gbps (3G)<br>REFCLK  = 148.5 MHz<br>Pattern = Single Line<br>Scramble Color Bar<br>No Equalization<br>DC Gain = 0 dB | > 0.3 | UI |
| Sinusoidal Jitter Tolerance (peak-to-peak) | Jitter Frequency = 20 KHz<br>Data Rate = 1.485 Gbps (HD)<br>REFCLK = 74.25 MHz<br>Pattern = 75% Color Bar<br>No  Equalization<br>DC Gain = 0 dB | > 1 | UI |
| | Jitter Frequency = 100 KHz<br>Data Rate = 1.485 Gbps (HD)<br>REFCLK = 74.25 MHz<br>Pattern = 75% Color Bar<br>No Equalization<br>DC Gain = 0 dB | > 0.2 | UI |

**Notes to Table 4–7:**

(1)   Dedicated REFCLK pins were used to drive the input reference clocks.

(2)   Jitter numbers specified are valid for the stated conditions only.

(3)   Refer to the protocol characterization documents for detailed information.

(4)   The jitter numbers for XAUI are compliant to the IEEE802.3ae-2002 Specification.

(5)   The jitter numbers for PCI Express are compliant to the PCIe Base Specification 2.0.

(6)   The jitter numbers for Serial RapidIO are compliant to the RapidIO Specification 1.3.

(7)   The jitter numbers for GIGE are compliant to the IEEE802.3-2002 Specification.

(8)   The HD-SDI and 3G-SDI jitter numbers are compliant to the SMPTE292M and SMPTE424M specifications.

Table 4–8 and Table 4–9 list the transmitter and receiver PCS latency for each mode, respectively.

**Table 4–8.** PCS Latency   *(Note 1)*

| Functional Mode | Configuration | Transmitter PCS Latency | | | | | |
|---|---|---|---|---|---|---|---|
| | | TX PIPE | TX Phase Comp FIFO | Byte Serializer | TX State Machine | 8B/10B Encoder | Sum *(2)* |
| XAUI | — | — | 2–3 | 1 | 0.5 | 0.5 | 4–5 |
| PIPE | ×1, ×4, ×8 8-bit channel width | 1 | 3–4 | 1 | — | 1 | 6–7 |
| | ×1, ×4, ×8 16-bit channel width | 1 | 3–4 | 1 | — | 0.5 | 6–7 |
| GIGE | — | — | 2–3 | 1 | — | 1 | 4–5 |
| Serial RapidIO | 1.25 Gbps, 2.5 Gbps, 3.125 Gbps | — | 2–3 | 1 | — | 0.5 | 4–5 |
| SDI | HD10-bit channel width | — | 2–3 | 1 | — | 1 | 4–5 |
| | HD, 3G 20-bit channel width | — | 2–3 | 1 | — | 0.5 | 4–5 |
| BASIC Single Width | 8-bit/10-bit channel width | — | 2–3 | 1 | — | 1 | 4–5 |
| | 16-bit/20-bit channel width | — | 2–3 | 1 | — | 0.5 | 4–5 |

**Notes to Table 4–8:**

(1)  The latency numbers are with respect to the PLD-transceiver interface clock cycles.

(2)  The total latency number is rounded off in the Sum column.

**Table 4–9.** PCS Latency  (Part 1 of 2)   (Part 1 of 2)

| Functional Mode | Configuration | Receiver PCS Latency | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Word Aligner | Deskew FIFO | Rate Matcher *(3)* | 8B/10B Decoder | Receiver State Machine | Byte Deserializer | Byte Order | Receiver Phase Comp FIFO | Receiver PIPE | Sum *(2)* |
| XAUI | — | 2–2.5 | 2–2.5 | 5.5–6.5 | 0.5 | 1 | 1 | 1 | 1–2 | — | 14–17 |
| PIPE | ×1, ×4 8-bit channel width | 4–5 | — | 11–13 | 1 | — | 1 | 1 | 2–3 | 1 | 21–25 |
| | ×1, ×4 16-bit channel width | 2–2.5 | — | 5.5–6.5 | 0.5 | — | 1 | 1 | 2–3 | 1 | 13–16 |
| GIGE | — | 4–5 | — | 11–13 | 1 | — | 1 | 1 | 1–2 | — | 19–23 |
| Serial RapidIO | 1.25 Gbps, 2.5 Gbps, 3.125 Gbps | 2–2.5 | — | — | 0.5 | — | 1 | 1 | 1–2 | — | 6–7 |
| SDI | HD 10-bit channel width | 5 | — | — | 1 | — | 1 | 1 | 1–2 | — | 9–10 |
| | HD, 3G 20-bit channel width | 2.5 | — | — | 0.5 | — | 1 | 1 | 1–2 | — | 6–7 |

**Table 4–9.** PCS Latency  (Part 2 of 2)   (Part 2 of 2)

| Functional Mode | Configuration | Receiver PCS Latency | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Word Aligner | Deskew FIFO | Rate Matcher (3) | 8B/10B Decoder | Receiver State Machine | Byte Deserializer | Byte Order | Receiver Phase Comp FIFO | Receiver PIPE | Sum (2) |
| BASIC Single Width | 8/10-bit channel width; with Rate Matcher | 4–5 | — | 11–13 | 1 | — | 1 | 1 | 1–2 | 1 | 19–23 |
| | 8/10-bit channel width; without Rate Matcher | 4–5 | — | — | 1 | — | 1 | 1 | 1–2 | — | 8–10 |
| | 16/20-bit channel width; with Rate Matcher | 2–2.5 | — | 5.5–6.5 | 0.5 | — | 1 | 1 | 1–2 | — | 11–14 |
| | 16/20-bit channel width; without Rate Matcher | 2–2.5 | — | — | 0.5 | — | 1 | 1 | 1–2 | — | 6–7 |

**Notes to Table 4–9:**

(1)  The latency numbers are with respect to the PLD-transceiver interface clock cycles.

(2)  The total latency number is rounded off in the Sum column.

(3)  The rate matcher latency shown is the steady state latency. Actual latency may vary depending on the skip ordered set gap allowed by the protocol, actual PPM difference between the reference clocks, and so forth.

Table 4–10 through Table 4–13 show the typical $V_{OD}$ for data rates from 600 Mbps to 3.125 Gbps. The specification is for measurement at the package ball.

**Table 4–10.** Typical $V_{OD}$ Setting, TX Term = 100 $\Omega$

| $V_{cc}$ HTX = 1.5 V | $V_{OD}$ Setting (mV) | | | | |
|---|---|---|---|---|---|
| | 400 | 600 | 800 | 1000 | 1200 |
| $V_{OD}$ Typical (mV) | 430 | 625 | 830 | 1020 | 1200 |

**Table 4–11.** Typical $V_{OD}$ Setting, TX Term = 100 $\Omega$

| $V_{cc}$ HTX = 1.2 V | $V_{OD}$ Setting (mV) | | | | |
|---|---|---|---|---|---|
| | 320 | 480 | 640 | 800 | 960 |
| $V_{OD}$ Typical (mV) | 344 | 500 | 664 | 816 | 960 |

**Table 4–12.** Typical Pre-Emphasis (First Post-Tap),   *(Note 1)*

| $V_{cc}$ HTX = 1.5 V | First Post Tap Pre-Emphasis Level | | | | |
|---|---|---|---|---|---|
| $V_{OD}$ Setting (mV) | 1 | 2 | 3 | 4 | 5 |
| | TX Term = 100 $\Omega$ | | | | |
| 400 | 24% | 62% | 112% | 184% | — |
| 600 | — | 31% | 56% | 86% | 122% |
| 800 | — | 20% | 35% | 53% | 73% |

**Table 4–12.** Typical Pre-Emphasis (First Post-Tap), *(Note 1)*

| $V_{cc}$ HTX = 1.5 V | First Post Tap Pre-Emphasis Level | | | | |
|---|---|---|---|---|---|
| $V_{OD}$ Setting (mV) | 1 | 2 | 3 | 4 | 5 |
| 1000 | — | — | 23% | 36% | 49% |
| 1200 | — | — | 17% | 25% | 35% |

**Note to Table 4–12:**

(1) Applicable to data rates from 600 Mbps to 3.125 Gbps. Specification is for measurement at the package ball.

**Table 4–13.** Typical Pre-Emphasis (First Post-Tap), *(Note 1)*

| $V_{cc}$ HTX = 1.2 V | First Post Tap Pre-Emphasis Level | | | | |
|---|---|---|---|---|---|
| $V_{OD}$ Setting (mV) | 1 | 2 | 3 | 4 | 5 |
| | TX Term = 100 $\Omega$ | | | | |
| 320 | 24% | 61% | 114% | — | — |
| 480 | — | 31% | 55% | 86% | 121% |
| 640 | — | 20% | 35% | 54% | 72% |
| 800 | — | — | 23% | 36% | 49% |
| 960 | — | — | 18% | 25% | 35% |

**Note to Table 4–13:**

(1) Applicable to data rates from 600 Mbps to 3.125 Gbps. Specification is for measurement at the package ball.

# DC Electrical Characteristics

Table 4–14 lists the Arria GX device family DC electrical characteristics.

**Table 4–14.** Arria GX Device DC Operating Conditions  (Part 1 of 2)     *(Note 1)*

| Symbol | Parameter | Conditions | Device | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|---|
| $I_I$ | Input pin leakage current | $V_I = V_{CCIOmax}$ to 0 V *(2)* | All | −10 | — | 10 | μA |
| $I_{OZ}$ | Tri-stated I/O pin leakage current | $V_O = V_{CCIOmax}$ to 0 V *(2)* | All | −10 | — | 10 | μA |
| $I_{CCINT0}$ | $V_{CCINT}$ supply current (standby) | $V_I$ = ground, no load, no toggling inputs<br>$T_J = 25$ °C | EP1AGX20/35 | — | 0.30 | *(3)* | A |
| | | | EP1AGX50/60 | — | 0.50 | *(3)* | A |
| | | | EP1AGX90 | — | 0.62 | *(3)* | A |
| $I_{CCPD0}$ | $V_{CCPD}$ supply current (standby) | $V_I$ = ground, no load, no toggling inputs<br>$T_J = 25$ °C,<br>$V_{CCPD} = 3.3$V | EP1AGX20/35 | — | 2.7 | *(3)* | mA |
| | | | EP1AGX50/60 | — | 3.6 | *(3)* | mA |
| | | | EP1AGX90 | — | 4.3 | *(3)* | mA |
| $I_{CCIO0}$ | $V_{CCIO}$ supply current (standby) | $V_I$ = ground, no load, no toggling inputs<br>$T_J = 25$ °C | EP1AGX20/35 | — | 4.0 | *(3)* | mA |
| | | | EP1AGX50/60 | — | 4.0 | *(3)* | mA |
| | | | EP1AGX90 | — | 4.0 | *(3)* | mA |

**Table 4–14.** Arria GX Device DC Operating Conditions  (Part 2 of 2)    *(Note 1)*

| Symbol | Parameter | Conditions | Device | Min | Typ | Max | Units |
|--------|-----------|------------|--------|-----|-----|-----|-------|
| $R_{CONF}$ *(4)* | Value of I/O pin pull-up resistor before and during configuration | $V_i = 0$, $V_{CCIO} = 3.3$ V | — | 10 | 25 | 50 | kΩ |
| | | $V_i = 0$, $V_{CCIO} = 2.5$ V | — | 15 | 35 | 70 | kΩ |
| | | $V_i = 0$, $V_{CCIO} = 1.8$ V | — | 30 | 50 | 100 | kΩ |
| | | $V_i = 0$, $V_{CCIO} = 1.5$ V | — | 40 | 75 | 150 | kΩ |
| | | $V_i = 0$, $V_{CCIO} = 1.2$ V | — | 50 | 90 | 170 | kΩ |
| | Recommended value of I/O pin external pull-down resistor before and during configuration | — | — | — | 1 | 2 | kΩ |

**Notes to Table 4–14:**

(1) Typical values are for $T_A = 25$ °C, $V_{CCINT} = 1.2$ V, and $V_{CCIO} = 1.2$ V, 1.5 V, 1.8 V, 2.5 V, and 3.3 V.

(2) This value is specified for normal device operation. The value may vary during power-up. This applies for all $V_{CCIO}$ settings (3.3, 2.5, 1.8, 1.5, and 1.2 V).

(3) Maximum values depend on the actual TJ and design utilization. For maximum values, refer to the Excel-based PowerPlay Early Power Estimator (available at PowerPlay Early Power Estimators (EPE) and Power Analyzer) or the Quartus® II PowerPlay Power Analyzer feature for maximum values. For more information, refer to "Power Consumption" on page 4–25.

(4) Pin pull-up resistance values will be lower if an external source drives the pin higher than $V_{CCIO}$.

# I/O Standard Specifications

Table 4–15 through Table 4–38 show the Arria GX device family I/O standard specifications.

**Table 4–15.** LVTTL Specifications

| Symbol | Parameter | Conditions | Minimum | Maximum | Units |
|--------|-----------|------------|---------|---------|-------|
| $V_{CCIO}$ *(1)* | Output supply voltage | — | 3.135 | 3.465 | V |
| $V_{IH}$ | High-level input voltage | — | 1.7 | 4.0 | V |
| $V_{IL}$ | Low-level input voltage | — | −0.3 | 0.8 | V |
| $V_{OH}$ | High-level output voltage | $I_{OH} = -4$ mA *(2)* | 2.4 | — | V |
| $V_{OL}$ | Low-level output voltage | $I_{OL} = 4$ mA *(2)* | — | 0.45 | V |

**Notes to Table 4–15:**

(1) Arria GX devices comply to the narrow range for the supply voltage as specified in the EIA/JEDEC Standard, JESD8-B.

(2) This specification is supported across all the programmable drive strength settings available for this I/O standard.

**Table 4–16.** LVCMOS Specifications

| Symbol | Parameter | Conditions | Minimum | Maximum | Units |
|--------|-----------|------------|---------|---------|-------|
| $V_{CCIO}$ *(1)* | Output supply voltage | — | 3.135 | 3.465 | V |
| $V_{IH}$ | High-level input voltage | — | 1.7 | 4.0 | V |
| $V_{IL}$ | Low-level input voltage | — | −0.3 | 0.8 | V |
| $V_{OH}$ | High-level output voltage | $V_{CCIO} = 3.0$, $I_{OH} = -0.1$ mA *(2)* | $V_{CCIO} - 0.2$ | — | V |
| $V_{OL}$ | Low-level output voltage | $V_{CCIO} = 3.0$, $I_{OL} = 0.1$ mA *(2)* | — | 0.2 | V |

**Notes to Table 4–16:**

(1) Arria GX devices comply to the narrow range for the supply voltage as specified in the EIA/JEDEC Standard, JESD8-B.

(2) This specification is supported across all the programmable drive strength available for this I/O standard.

**Table 4–17.** 2.5-V I/O Specifications

| Symbol | Parameter | Conditions | Minimum | Maximum | Units |
|--------|-----------|------------|---------|---------|-------|
| $V_{CCIO}$ (1) | Output supply voltage | — | 2.375 | 2.625 | V |
| $V_{IH}$ | High-level input voltage | — | 1.7 | 4.0 | V |
| $V_{IL}$ | Low-level input voltage | — | −0.3 | 0.7 | V |
| $V_{OH}$ | High-level output voltage | $I_{OH}$ = −1 mA (2) | 2.0 | — | V |
| $V_{OL}$ | Low-level output voltage | $I_{OL}$ = 1 mA (2) | — | 0.4 | V |

**Notes to Table 4–17:**

(1) The Arria GX device $V_{CCIO}$ voltage level support of 2.5 to 5% is narrower than defined in the normal range of the EIA/JEDEC standard.

(2) This specification is supported across all the programmable drive settings available for this I/O standard.

**Table 4–18.** 1.8-V I/O Specifications

| Symbol | Parameter | Conditions | Minimum | Maximum | Units |
|--------|-----------|------------|---------|---------|-------|
| $V_{CCIO}$ (1) | Output supply voltage | — | 1.71 | 1.89 | V |
| $V_{IH}$ | High-level input voltage | — | $0.65 \times V_{CCIO}$ | 2.25 | V |
| $V_{IL}$ | Low-level input voltage | — | −0.3 | $0.35 \times V_{CCIO}$ | V |
| $V_{OH}$ | High-level output voltage | $I_{OH}$ = −2 mA (2) | $V_{CCIO}$ − 0.45 | — | V |
| $V_{OL}$ | Low-level output voltage | $I_{OL}$ = 2 mA (2) | — | 0.45 | V |

**Notes to Table 4–18:**

(1) The Arria GX device $V_{CCIO}$ voltage level support of 1.8 to 5% is narrower than defined in the normal range of the EIA/JEDEC standard.

(2) This specification is supported across all the programmable drive settings available for this I/O standard, as shown in *Arria GX Architecture* chapter.

**Table 4–19.** 1.5-V I/O Specifications

| Symbol | Parameter | Conditions | Minimum | Maximum | Units |
|--------|-----------|------------|---------|---------|-------|
| $V_{CCIO}$ (1) | Output supply voltage | — | 1.425 | 1.575 | V |
| $V_{IH}$ | High-level input voltage | — | $0.65\ V_{CCIO}$ | $V_{CCIO}$ + 0.3 | V |
| $V_{IL}$ | Low-level input voltage | — | −0.3 | $0.35\ V_{CCIO}$ | V |
| $V_{OH}$ | High-level output voltage | $I_{OH}$ = −2 mA (2) | $0.75\ V_{CCIO}$ | — | V |
| $V_{OL}$ | Low-level output voltage | $I_{OL}$ = 2 mA (2) | — | $0.25\ V_{CCIO}$ | V |

**Notes to Table 4–19:**

(1) The Arria GX device $V_{CCIO}$ voltage level support of 1.5 to 5% is narrower than defined in the normal range of the EIA/JEDEC standard.

(2) This specification is supported across all the programmable drive settings available for this I/O standard, as shown in the *Arria GX Architecture* chapter.

Figure 4–5 and Figure 4–6 show receiver input and transmitter output waveforms, respectively, for all differential I/O standards (LVDS and LVPECL).

**Figure 4–5.** Receiver Input Waveforms for Differential I/O Standards

**Single-Ended Waveform**

$V_{ID}$

$V_{CM}$

Positive Channel (p) = $V_{IH}$

Negative Channel (n) = $V_{IL}$

Ground

**Differential Waveform**

$V_{ID}$

$V_{ID \text{ (Peak-to-Peak)}}$

$V_{ID}$

p − n = 0 V
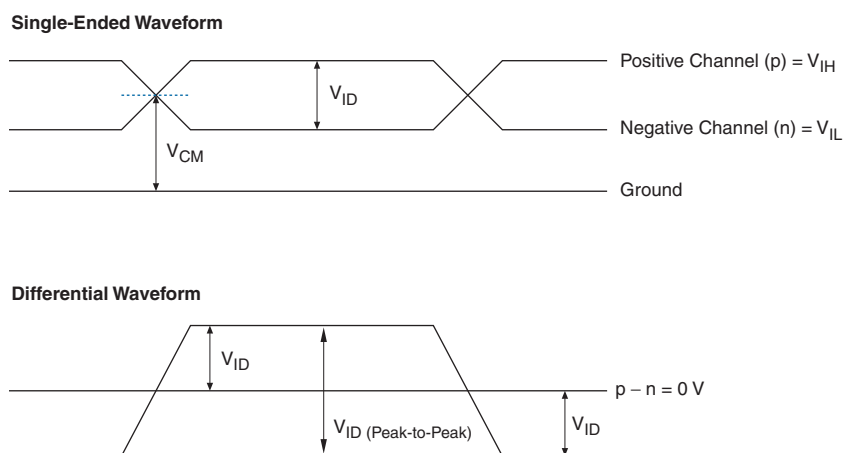
**Figure 4–6.** Transmitter Output Waveforms for Differential I/O Standards

**Single-Ended Waveform**

$V_{OD}$

$V_{CM}$

Positive Channel (p) = $V_{OH}$

Negative Channel (n) = $V_{OL}$

Ground

**Differential Waveform**

$V_{OD}$

$V_{OD}$

p − n = 0 V
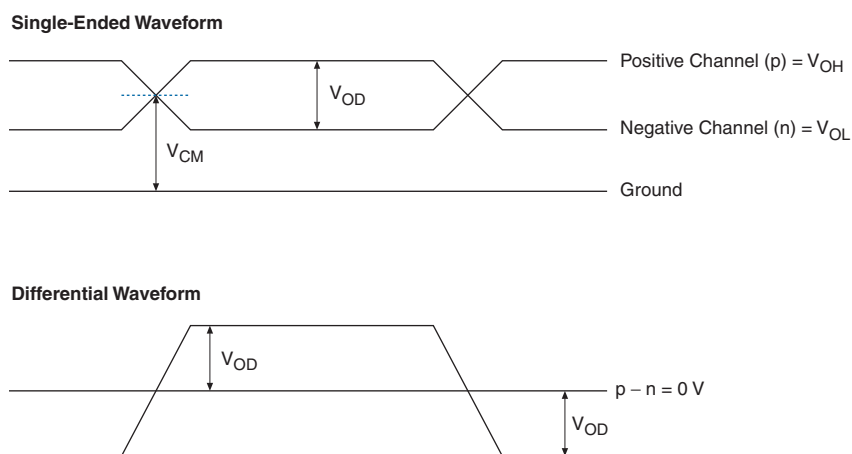
**Table 4–20.** 2.5-V LVDS I/O Specifications

| Symbol | Parameter | Conditions | Minimum | Typical | Maximum | Units |
|--------|-----------|------------|---------|---------|---------|-------|
| $V_{CCIO}$ | I/O supply voltage for left and right I/O banks (1, 2, 5, and 6) | — | 2.375 | 2.5 | 2.625 | V |
| $V_{ID}$ | Input differential voltage swing (single-ended) | — | 100 | 350 | 900 | mV |
| $V_{ICM}$ | Input common mode voltage | — | 200 | 1,250 | 1,800 | mV |
| $V_{OD}$ | Output differential voltage (single-ended) | $R_L = 100\ \Omega$ | 250 | — | 450 | mV |
| $V_{OCM}$ | Output common mode voltage | $R_L = 100\ \Omega$ | 1.125 | — | 1.375 | V |
| $R_L$ | Receiver differential input discrete resistor (external to Arria GX devices) | — | 90 | 100 | 110 | $\Omega$ |

**Table 4–21.** 3.3-V LVDS I/O Specifications

| Symbol | Parameter | Conditions | Minimum | Typical | Maximum | Units |
|--------|-----------|------------|---------|---------|---------|-------|
| $V_{CCIO}$ (1) | I/O supply voltage for top and bottom PLL banks (9, 10, 11, and 12) | — | 3.135 | 3.3 | 3.465 | V |
| $V_{ID}$ | Input differential voltage swing (single-ended) | — | 100 | 350 | 900 | mV |
| $V_{ICM}$ | Input common mode voltage | — | 200 | 1,250 | 1,800 | mV |
| $V_{OD}$ | Output differential voltage (single-ended) | $R_L = 100\ \Omega$ | 250 | — | 710 | mV |
| $V_{OCM}$ | Output common mode voltage | $R_L = 100\ \Omega$ | 840 | — | 1,570 | mV |
| $R_L$ | Receiver differential input discrete resistor (external to Arria GX devices) | — | 90 | 100 | 110 | $\Omega$ |

**Note to Table 4–21:**

(1) The top and bottom clock input differential buffers in I/O banks 3, 4, 7, and 8 are powered by $V_{CCINT}$, not $V_{CCIO}$. The PLL clock output/feedback differential buffers are powered by VCC_PLL_OUT. For differential clock output/feedback operation, connect VCC_PLL_OUT to 3.3 V.

**Table 4–22.** 3.3-V PCML Specifications

| Symbol | Parameter | Minimum | Typical | Maximum | Units |
|--------|-----------|---------|---------|---------|-------|
| $V_{CCIO}$ | I/O supply voltage | 3.135 | 3.3 | 3.465 | V |
| $V_{ID}$ | Input differential voltage swing (single-ended) | 300 | — | 600 | mV |
| $V_{ICM}$ | Input common mode voltage | 1.5 | — | 3.465 | V |
| $V_{OD}$ | Output differential voltage (single-ended) | 300 | 370 | 500 | mV |
| $\Delta V_{OD}$ | Change in $V_{OD}$ between high and low | — | — | 50 | mV |
| $V_{OCM}$ | Output common mode voltage | 2.5 | 2.85 | 3.3 | V |
| $\Delta V_{OCM}$ | Change in $V_{OCM}$ between high and low | — | — | 50 | mV |
| $V_T$ | Output termination voltage | — | $V_{CCIO}$ | — | V |
| $R_1$ | Output external pull-up resistors | 45 | 50 | 55 | $\Omega$ |
| $R_2$ | Output external pull-up resistors | 45 | 50 | 55 | $\Omega$ |

**Table 4–23.** LVPECL Specifications

| Parameter | Conditions | Minimum | Typical | Maximum | Units | Parameter |
|-----------|------------|---------|---------|---------|-------|-----------|
| $V_{CCIO}$ (1) | I/O supply voltage | — | 3.135 | 3.3 | 3.465 | V |
| $V_{ID}$ | Input differential voltage swing (single-ended) | — | 300 | 600 | 1,000 | mV |
| $V_{ICM}$ | Input common mode voltage | — | 1.0 | — | 2.5 | V |
| $V_{OD}$ | Output differential voltage (single-ended) | $R_L = 100\ \Omega$ | 525 | — | 970 | mV |
| $V_{OCM}$ | Output common mode voltage | $R_L = 100\ \Omega$ | 1,650 | — | 2,250 | mV |
| $R_L$ | Receiver differential input resistor | — | 90 | 100 | 110 | $\Omega$ |

**Note to Table 4–23:**

(1) The top and bottom clock input differential buffers in I/O banks 3, 4, 7, and 8 are powered by $V_{CCINT}$, not $V_{CCIO}$. The PLL clock output/feedback differential buffers are powered by VCC_PLL_OUT. For differential clock output/feedback operation, connect VCC_PLL_OUT to 3.3 V.

**Table 4–24.** 3.3-V PCI Specifications

| Symbol | Parameter | Conditions | Minimum | Typical | Maximum | Units |
|---|---|---|---|---|---|---|
| $V_{CCIO}$ | Output supply voltage | — | 3.0 | 3.3 | 3.6 | V |
| $V_{IH}$ | High-level input voltage | — | 0.5 $V_{CCIO}$ | — | $V_{CCIO}$ + 0.5 | V |
| $V_{IL}$ | Low-level input voltage | — | −0.3 | — | 0.3 $V_{CCIO}$ | V |
| $V_{OH}$ | High-level output voltage | $I_{OUT}$ = −500 μA | 0.9 $V_{CCIO}$ | — | — | V |
| $V_{OL}$ | Low-level output voltage | $I_{OUT}$ = 1,500 μA | — | — | 0.1 $V_{CCIO}$ | V |

**Table 4–25.** PCI-X Mode 1 Specifications

| Symbol | Parameter | Conditions | Minimum | Maximum | Units |
|---|---|---|---|---|---|
| $V_{CCIO}$ | Output supply voltage | — | 3.0 | 3.6 | V |
| $V_{IH}$ | High-level input voltage | — | 0.5 $V_{CCIO}$ | $V_{CCIO}$ + 0.5 | V |
| $V_{IL}$ | Low-level input voltage | — | −0.3 | 0.35 $V_{CCIO}$ | V |
| $V_{IPU}$ | Input pull-up voltage | — | 0.7 $V_{CCIO}$ | — | V |
| $V_{OH}$ | High-level output voltage | $I_{OUT}$ = −500 μA | 0.9 $V_{CCIO}$ | — | V |
| $V_{OL}$ | Low-level output voltage | $I_{OUT}$ = 1,500 μA | — | 0.1 $V_{CCIO}$ | V |

**Table 4–26.** SSTL-18 Class I Specifications

| Symbol | Parameter | Conditions | Minimum | Typical | Maximum | Units |
|---|---|---|---|---|---|---|
| $V_{CCIO}$ | Output supply voltage | — | 1.71 | 1.8 | 1.89 | V |
| $V_{REF}$ | Reference voltage | — | 0.855 | 0.9 | 0.945 | V |
| $V_{TT}$ | Termination voltage | — | $V_{REF}$ − 0.04 | $V_{REF}$ | $V_{REF}$ + 0.04 | V |
| $V_{IH}$ (DC) | High-level DC input voltage | — | $V_{REF}$ + 0.125 | — | — | V |
| $V_{IL}$ (DC) | Low-level DC input voltage | — | — | — | $V_{REF}$ − 0.125 | V |
| $V_{IH}$ (AC) | High-level AC input voltage | — | $V_{REF}$ + 0.25 | — | — | V |
| $V_{IL}$ (AC) | Low-level AC input voltage | — | — | — | $V_{REF}$ − 0.25 | V |
| $V_{OH}$ | High-level output voltage | $I_{OH}$ = −6.7 mA *(1)* | $V_{TT}$ + 0.475 | — | — | V |
| $V_{OL}$ | Low-level output voltage | $I_{OL}$ = 6.7 mA *(1)* | — | — | $V_{TT}$ − 0.475 | V |

**Note to Table 4–26:**

(1) This specification is supported across all the programmable drive settings available for this I/O standard as shown in the *Arria GX Architecture* chapter.

**Table 4–27.** SSTL-18 Class II Specifications

| Symbol | Parameter | Conditions | Minimum | Typical | Maximum | Units |
|---|---|---|---|---|---|---|
| $V_{CCIO}$ | Output supply voltage | — | 1.71 | 1.8 | 1.89 | V |
| $V_{REF}$ | Reference voltage | — | 0.855 | 0.9 | 0.945 | V |
| $V_{TT}$ | Termination voltage | — | $V_{REF}$ − 0.04 | $V_{REF}$ | $V_{REF}$ + 0.04 | V |
| $V_{IH}$ (DC) | High-level DC input voltage | — | $V_{REF}$ + 0.125 | — | — | V |
| $V_{IL}$ (DC) | Low-level DC input voltage | — | — | — | $V_{REF}$ − 0.125 | V |
| $V_{IH}$ (AC) | High-level AC input voltage | — | $V_{REF}$ + 0.25 | — | — | V |
| $V_{IL}$ (AC) | Low-level AC input voltage | — | — | — | $V_{REF}$ − 0.25 | V |

**Table 4–27.** SSTL-18 Class II Specifications

| Symbol | Parameter | Conditions | Minimum | Typical | Maximum | Units |
|--------|-----------|------------|---------|---------|---------|-------|
| $V_{OH}$ | High-level output voltage | $I_{OH} = -13.4$ mA *(1)* | $V_{CCIO} - 0.28$ | — | — | V |
| $V_{OL}$ | Low-level output voltage | $I_{OL} = 13.4$ mA *(1)* | — | — | 0.28 | V |

**Note to Table 4–27:**

(1) This specification is supported across all the programmable drive settings available for this I/O standard as shown in the *Arria GX Architecture* chapter.

**Table 4–28.** SSTL-18 Class I & II Differential Specifications

| Symbol | Parameter | Minimum | Typical | Maximum | Units |
|--------|-----------|---------|---------|---------|-------|
| $V_{CCIO}$ | Output supply voltage | 1.71 | 1.8 | 1.89 | V |
| $V_{SWING}$ (DC) | DC differential input voltage | 0.25 | — | — | V |
| $V_X$ (AC) | AC differential input cross point voltage | $(V_{CCIO}/2) - 0.175$ | — | $(V_{CCIO}/2) + 0.175$ | V |
| $V_{SWING}$ (AC) | AC differential input voltage | 0.5 | — | — | V |
| $V_{ISO}$ | Input clock signal offset voltage | — | $0.5\ V_{CCIO}$ | — | V |
| $\Delta V_{ISO}$ | Input clock signal offset voltage variation | — | 200 | — | mV |
| $V_{OX}$ (AC) | AC differential cross point voltage | $(V_{CCIO}/2) - 0.125$ | — | $(V_{CCIO}/2) + 0.125$ | V |

**Table 4–29.** SSTL-2 Class I Specifications

| Symbol | Parameter | Conditions | Minimum | Typical | Maximum | Units |
|--------|-----------|------------|---------|---------|---------|-------|
| $V_{CCIO}$ | Output supply voltage | — | 2.375 | 2.5 | 2.625 | V |
| $V_{TT}$ | Termination voltage | — | $V_{REF} - 0.04$ | $V_{REF}$ | $V_{REF} + 0.04$ | V |
| $V_{REF}$ | Reference voltage | — | 1.188 | 1.25 | 1.313 | V |
| $V_{IH}$ (DC) | High-level DC input voltage | — | $V_{REF} + 0.18$ | — | 3.0 | V |
| $V_{IL}$ (DC) | Low-level DC input voltage | — | −0.3 | — | $V_{REF} - 0.18$ | V |
| $V_{IH}$ (AC) | High-level AC input voltage | — | $V_{REF} + 0.35$ | — | — | V |
| $V_{IL}$ (AC) | Low-level AC input voltage | — | — | — | $V_{REF} - 0.35$ | V |
| $V_{OH}$ | High-level output voltage | $I_{OH} = -8.1$ mA *(1)* | $V_{TT} + 0.57$ | — | | V |
| $V_{OL}$ | Low-level output voltage | $I_{OL} = 8.1$ mA *(1)* | — | — | $V_{TT} - 0.57$ | V |

**Note to Table 4–29:**

(1) This specification is supported across all the programmable drive settings available for this I/O standard as shown in the *Arria GX Architecture* chapter.

**Table 4–30.** SSTL-2 Class II Specifications   (Part 1 of 2)

| Symbol | Parameter | Conditions | Minimum | Typical | Maximum | Units |
|--------|-----------|------------|---------|---------|---------|-------|
| $V_{CCIO}$ | Output supply voltage | — | 2.375 | 2.5 | 2.625 | V |
| $V_{TT}$ | Termination voltage | — | $V_{REF} - 0.04$ | $V_{REF}$ | $V_{REF} + 0.04$ | V |
| $V_{REF}$ | Reference voltage | — | 1.188 | 1.25 | 1.313 | V |
| $V_{IH}$ (DC) | High-level DC input voltage | — | $V_{REF} + 0.18$ | — | $V_{CCIO} + 0.3$ | V |

**Table 4–30.** SSTL-2 Class II Specifications   (Part 2 of 2)

| Symbol | Parameter | Conditions | Minimum | Typical | Maximum | Units |
|--------|-----------|------------|---------|---------|---------|-------|
| $V_{IL}$ (DC) | Low-level DC input voltage | — | −0.3 | — | $V_{REF} - 0.18$ | V |
| $V_{IH}$ (AC) | High-level AC input voltage | — | $V_{REF} + 0.35$ | — | — | V |
| $V_{IL}$ (AC) | Low-level AC input voltage | — | — | — | $V_{REF} - 0.35$ | V |
| $V_{OH}$ | High-level output voltage | $I_{OH} = -16.4$ mA *(1)* | $V_{TT} + 0.76$ | — | — | V |
| $V_{OL}$ | Low-level output voltage | $I_{OL} = 16.4$ mA *(1)* | — | — | $V_{TT} - 0.76$ | V |

**Note to Table 4–30:**

(1) This specification is supported across all the programmable drive settings available for this I/O standard as shown in the *Arria GX Architecture* chapter.

**Table 4–31.** SSTL-2 Class I & II Differential Specifications   *(Note 1)*

| Symbol | Parameter | Minimum | Typical | Maximum | Units |
|--------|-----------|---------|---------|---------|-------|
| $V_{CCIO}$ | Output supply voltage | 2.375 | 2.5 | 2.625 | V |
| $V_{SWING}$ (DC) | DC differential input voltage | 0.36 | — | — | V |
| $V_X$ (AC) | AC differential input cross point voltage | $(V_{CCIO}/2) - 0.2$ | — | $(V_{CCIO}/2) + 0.2$ | V |
| $V_{SWING}$ (AC) | AC differential input voltage | 0.7 | — | — | V |
| $V_{ISO}$ | Input clock signal offset voltage | — | 0.5 $V_{CCIO}$ | — | V |
| $\Delta V_{ISO}$ | Input clock signal offset voltage variation | — | 200 | — | mV |
| $V_{OX}$ (AC) | AC differential output cross point voltage | $(V_{CCIO}/2) - 0.2$ | — | $(V_{CCIO}/2) + 0.2$ | V |

**Note to Table 4–31:**

(1) This specification is supported across all the programmable drive settings available for this I/O standard as shown in the *Arria GX Architecture* chapter.

**Table 4–32.** 1.2-V HSTL Specifications

| Symbol | Parameter | Minimum | Typical | Maximum | Units |
|--------|-----------|---------|---------|---------|-------|
| $V_{CCIO}$ | Output supply voltage | 1.14 | 1.2 | 1.26 | V |
| $V_{REF}$ | Reference voltage | 0.48 $V_{CCIO}$ | 0.5 $V_{CCIO}$ | 0.52 $V_{CCIO}$ | V |
| $V_{IH}$ (DC) | High-level DC input voltage | $V_{REF} + 0.08$ | — | $V_{CCIO} + 0.15$ | V |
| $V_{IL}$ (DC) | Low-level DC input voltage | −0.15 | — | $V_{REF} - 0.08$ | V |
| $V_{IH}$ (AC) | High-level AC input voltage | $V_{REF} + 0.15$ | — | $V_{CCIO} + 0.24$ | V |
| $V_{IL}$ (AC) | Low-level AC input voltage | −0.24 | — | $V_{REF} - 0.15$ | V |
| $V_{OH}$ | High-level output voltage | $V_{REF} + 0.15$ | — | $V_{CCIO} + 0.15$ | V |
| $V_{OL}$ | Low-level output voltage | −0.15 | — | $V_{REF} - 0.15$ | V |

**Table 4–33.** 1.5-V HSTL Class I Specifications

| Symbol | Parameter | Conditions | Minimum | Typical | Maximum | Units |
|---|---|---|---|---|---|---|
| $V_{CCIO}$ | Output supply voltage | — | 1.425 | 1.5 | 1.575 | V |
| $V_{REF}$ | Input reference voltage | — | 0.713 | 0.75 | 0.788 | V |
| $V_{TT}$ | Termination voltage | — | 0.713 | 0.75 | 0.788 | V |
| $V_{IH}$ (DC) | DC high-level input voltage | — | $V_{REF}$ + 0.1 | — | — | V |
| $V_{IL}$ (DC) | DC low-level input voltage | — | −0.3 | — | $V_{REF}$ − 0.1 | V |
| $V_{IH}$ (AC) | AC high-level input voltage | — | $V_{REF}$ + 0.2 | — | — | V |
| $V_{IL}$ (AC) | AC low-level input voltage | — | — | — | $V_{REF}$ − 0.2 | V |
| $V_{OH}$ | High-level output voltage | $I_{OH}$ = 8 mA *(1)* | $V_{CCIO}$ − 0.4 | — | — | V |
| $V_{OL}$ | Low-level output voltage | $I_{OH}$ = −8 mA *(1)* | — | — | 0.4 | V |

**Note to Table 4–33:**

(1) This specification is supported across all the programmable drive settings available for this I/O standard as shown in the *Arria GX Architecture* chapter.

**Table 4–34.** 1.5-V HSTL Class II Specifications

| Symbol | Parameter | Conditions | Minimum | Typical | Maximum | Units |
|---|---|---|---|---|---|---|
| $V_{CCIO}$ | Output supply voltage | — | 1.425 | 1.50 | 1.575 | V |
| $V_{REF}$ | Input reference voltage | — | 0.713 | 0.75 | 0.788 | V |
| $V_{TT}$ | Termination voltage | — | 0.713 | 0.75 | 0.788 | V |
| $V_{IH}$ (DC) | DC high-level input voltage | — | $V_{REF}$ + 0.1 | — | — | V |
| $V_{IL}$ (DC) | DC low-level input voltage | — | −0.3 | — | $V_{REF}$ − 0.1 | V |
| $V_{IH}$ (AC) | AC high-level input voltage | — | $V_{REF}$ + 0.2 | — | — | V |
| $V_{IL}$ (AC) | AC low-level input voltage | — | — | — | $V_{REF}$ − 0.2 | V |
| $V_{OH}$ | High-level output voltage | $I_{OH}$ = 16 mA *(1)* | $V_{CCIO}$ − 0.4 | — | — | V |
| $V_{OL}$ | Low-level output voltage | $I_{OH}$ = −16 mA *(1)* | — | — | 0.4 | V |

**Note to Table 4–34:**

(1) This specification is supported across all the programmable drive settings available for this I/O standard, as shown in the *Arria GX Architecture* chapter.

**Table 4–35.** 1.5-V HSTL Class I & II Differential Specifications

| Symbol | Parameter | Minimum | Typical | Maximum | Units |
|---|---|---|---|---|---|
| $V_{CCIO}$ | I/O supply voltage | 1.425 | 1.5 | 1.575 | V |
| $V_{DIF}$ (DC) | DC input differential voltage | 0.2 | — | — | V |
| $V_{CM}$ (DC) | DC common mode input voltage | 0.68 | — | 0.9 | V |
| $V_{DIF}$ (AC) | AC differential input voltage | 0.4 | — | — | V |
| $V_{OX}$ (AC) | AC differential cross point voltage | 0.68 | — | 0.9 | V |

**Table 4–36.** 1.8-V HSTL Class I Specifications

| Symbol | Parameter | Conditions | Minimum | Typical | Maximum | Units |
|---|---|---|---|---|---|---|
| $V_{CCIO}$ | Output supply voltage | — | 1.71 | 1.80 | 1.89 | V |
| $V_{REF}$ | Input reference voltage | — | 0.85 | 0.90 | 0.95 | V |
| $V_{TT}$ | Termination voltage | — | 0.85 | 0.90 | 0.95 | V |
| $V_{IH}$ (DC) | DC high-level input voltage | — | $V_{REF}$ + 0.1 | — | — | V |
| $V_{IL}$ (DC) | DC low-level input voltage | — | −0.3 | — | $V_{REF}$ − 0.1 | V |
| $V_{IH}$ (AC) | AC high-level input voltage | — | $V_{REF}$ + 0.2 | — | — | V |
| $V_{IL}$ (AC) | AC low-level input voltage | — | — | — | $V_{REF}$ − 0.2 | V |
| $V_{OH}$ | High-level output voltage | $I_{OH}$ = 8 mA *(1)* | $V_{CCIO}$ − 0.4 | — | — | V |
| $V_{OL}$ | Low-level output voltage | $I_{OH}$ = −8 mA *(1)* | — | — | 0.4 | V |

**Note to Table 4–36:**

(1) This specification is supported across all the programmable drive settings available for this I/O standard, as shown in the *Arria GX Architecture* chapter.

**Table 4–37.** 1.8-V HSTL Class II Specifications

| Symbol | Parameter | Conditions | Minimum | Typical | Maximum | Units |
|---|---|---|---|---|---|---|
| $V_{CCIO}$ | Output supply voltage | — | 1.71 | 1.80 | 1.89 | V |
| $V_{REF}$ | Input reference voltage | — | 0.85 | 0.90 | 0.95 | V |
| $V_{TT}$ | Termination voltage | — | 0.85 | 0.90 | 0.95 | V |
| $V_{IH}$ (DC) | DC high-level input voltage | — | $V_{REF}$ + 0.1 | — | — | V |
| $V_{IL}$ (DC) | DC low-level input voltage | — | −0.3 | — | $V_{REF}$ − 0.1 | V |
| $V_{IH}$ (AC) | AC high-level input voltage | — | $V_{REF}$ + 0.2 | — | — | V |
| $V_{IL}$ (AC) | AC low-level input voltage | — | — | — | $V_{REF}$ − 0.2 | V |
| $V_{OH}$ | High-level output voltage | $I_{OH}$ = 16 mA *(1)* | $V_{CCIO}$ − 0.4 | — | — | V |
| $V_{OL}$ | Low-level output voltage | $I_{OH}$ = −16 mA *(1)* | — | — | 0.4 | V |

**Note to Table 4–37:**

(1) This specification is supported across all the programmable drive settings available for this I/O standard, as shown in the *Arria GX Architecture* chapter in volume 1 of the *Arria GX Device Handbook*.

**Table 4–38.** 1.8-V HSTL Class I & II Differential Specifications

| Symbol | Parameter | Minimum | Typical | Maximum | Units |
|---|---|---|---|---|---|
| $V_{CCIO}$ | I/O supply voltage | 1.71 | 1.80 | 1.89 | V |
| $V_{DIF}$ (DC) | DC input differential voltage | 0.2 | — | — | V |
| $V_{CM}$ (DC) | DC common mode input voltage | 0.78 | — | 1.12 | V |
| $V_{DIF}$ (AC) | AC differential input voltage | 0.4 | — | — | V |
| $V_{OX}$ (AC) | AC differential cross point voltage | 0.68 | — | 0.9 | V |

## Bus Hold Specifications

Table 4–39 shows the Arria GX device family bus hold specifications.

**Table 4–39.** Bus Hold Parameters

| Parameter | Conditions | $V_{CCIO}$ Level | | | | | | | | | | Units |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1.2 V | | 1.5 V | | 1.8 V | | 2.5 V | | 3.3 V | | |
| | | Min | Max | Min | Max | Min | Max | Min | Max | Min | Max | |
| Low sustaining current | $V_{IN} > V_{IL}$ (maximum) | 22.5 | — | 25 | — | 30 | — | 50 | — | 70 | — | μA |
| High sustaining current | $V_{IN} < V_{IH}$ (minimum) | −22.5 | — | −25 | — | −30 | — | −50 | — | −70 | — | μA |
| Low overdrive current | 0 V $<V_{IN} < V_{CCIO}$ | — | 120 | — | 160 | — | 200 | — | 300 | — | 500 | μA |
| High overdrive current | 0 V $< V_{IN} < V_{CCIO}$ | — | −120 | — | −160 | — | −200 | — | −300 | — | −500 | μA |
| Bus-hold trip point | — | 0.45 | 0.95 | 0.5 | 1.0 | 0.68 | 1.07 | 0.7 | 1.7 | 0.8 | 2.0 | V |

## On-Chip Termination Specifications

Table 4–40 and Table 4–41 define the specification for internal termination resistance tolerance when using series or differential on-chip termination.

**Table 4–40.** Series On-Chip Termination Specification for Top and Bottom I/O Banks

| Symbol | Description | Conditions | Resistance Tolerance | | |
|---|---|---|---|---|---|
| | | | Commercial Max | Industrial Max | Units |
| 25-Ω $R_S$ 3.3/2.5 | Internal series termination without calibration (25-Ω setting) | $V_{CCIO}$ = 3.3/2.5V | ±30 | ±30 | % |
| 50-Ω $R_S$ 3.3/2.5 | Internal series termination without calibration (50-Ω setting) | $V_{CCIO}$ = 3.3/2.5V | ±30 | ± 30 | % |
| 25-Ω $R_S$ 1.8 | Internal series termination without calibration (25-Ω setting) | $V_{CCIO}$ = 1.8V | ±30 | ±30 | % |
| 50-Ω $R_S$ 1.8 | Internal series termination without calibration (50-Ω setting) | $V_{CCIO}$ = 1.8V | ±30 | ±30 | % |
| 50-Ω $R_S$ 1.5 | Internal series termination without calibration (50-Ω setting) | $V_{CCIO}$ = 1.5V | ±36 | ±36 | % |
| 50-Ω $R_S$ 1.2 | Internal series termination without calibration (50-Ω setting) | $V_{CCIO}$ = 1.2V | ±50 | ±50 | % |

**Table 4–41.** Series On-Chip Termination Specification for Left I/O Banks

| Symbol | Description | Conditions | Resistance Tolerance | | |
|---|---|---|---|---|---|
| | | | Commercial Max | Industrial Max | Units |
| 25-$\Omega$ $R_S$ 3.3/2.5 | Internal series termination without calibration (25-$\Omega$ setting) | $V_{CCIO}$ = 3.3/2.5V | ±30 | ±30 | % |
| 50-$\Omega$ $R_S$ 3.3/2.5/1.8 | Internal series termination without calibration (50-$\Omega$ setting) | $V_{CCIO}$ = 3.3/2.5/1.8V | ±30 | ±30 | % |
| 50-$\Omega$ $R_S$ 1.5 | Internal series termination without calibration (50-$\Omega$ setting) | $V_{CCIO}$ = 1.5V | ±36 | ±36 | % |
| $R_D$ | Internal differential termination for LVDS (100-$\Omega$ setting) | $V_{CCIO}$ = 2.5V | ±20 | ±25 | % |

## Pin Capacitance

Table 4–42 shows the Arria GX device family pin capacitance.

**Table 4–42.** Arria GX Device Capacitance    *(Note 1)*

| Symbol | Parameter | Typical | Units |
|---|---|---|---|
| $C_{IOTB}$ | Input capacitance on I/O pins in I/O banks 3, 4, 7, and 8. | 5.0 | pF |
| $C_{IOL}$ | Input capacitance on I/O pins in I/O banks 1 and 2, including high-speed differential receiver and transmitter pins. | 6.1 | pF |
| $C_{CLKTB}$ | Input capacitance on top/bottom clock input pins: `CLK[4..7]` and `CLK[12..15]`. | 6.0 | pF |
| $C_{CLKL}$ | Input capacitance on left clock inputs: `CLK0` and `CLK2`. | 6.1 | pF |
| $C_{CLKL+}$ | Input capacitance on left clock inputs: `CLK1` and `CLK3`. | 3.3 | pF |
| $C_{OUTFB}$ | Input capacitance on dual-purpose clock output/feedback pins in PLL banks 11 and 12. | 6.7 | pF |

**Note to Table 4–42:**

(1)  Capacitance is sample-tested only. Capacitance is measured using time-domain reflections (TDR). Measurement accuracy is within ±0.5 pF.

## Power Consumption

Altera offers two ways to calculate power for a design: the Excel-based PowerPlay early power estimator power calculator and the Quartus II PowerPlay power analyzer feature.

The interactive Excel-based PowerPlay Early Power Estimator is typically used prior to designing the FPGA in order to get an estimate of device power. The Quartus II PowerPlay Power Analyzer provides better quality estimates based on the specifics of the design after place-and-route is complete. The power analyzer can apply a combination of user-entered, simulation-derived and estimated signal activities which, combined with detailed circuit models, can yield very accurate power estimates.

In both cases, these calculations should only be used as an estimation of power, not as a specification.

For more information about PowerPlay tools, refer to the *PowerPlay Early Power Estimator and PowerPlay Power Analyzer* page and the *PowerPlay Power Analysis* chapter in volume 3 of the *Quartus II Handbook*.

For typical $I_{CC}$ standby specifications, refer to Table 4–14 on page 4–14 .

# I/O Timing Model

The DirectDrive technology and MultiTrack interconnect ensures predictable performance, accurate simulation, and accurate timing analysis across all Arria GX device densities and speed grades. This section describes and specifies the performance of I/Os.

All specifications are representative of worst-case supply voltage and junction temperature conditions.

☞ The timing numbers listed in the tables of this section are extracted from the Quartus II software version 7.1.

## Preliminary, Correlated, and Final Timing

Timing models can have either preliminary, correlated, or final status. The Quartus II software issues an informational message during design compilation if the timing models are preliminary. Table 4–43 lists the status of the Arria GX device timing models.

■ **Preliminary** status means the timing model is subject to change. Initially, timing numbers are created using simulation results, process data, and other known parameters. These tests are used to make the preliminary numbers as close to the actual timing parameters as possible.

■ **Correlated** numbers are based on actual device operation and testing. These numbers reflect the actual performance of the device under worst-case voltage and junction temperature conditions.

■ **Final timing** numbers are based on complete correlation to actual devices and addressing any minor deviations from the correlated timing model. When the timing models are final, all or most of the Arria GX family devices have been completely characterized and no further changes to the timing model are expected.

**Table 4–43.** Arria GX Device Timing Model Status

| Device | Preliminary | Correlated | Final |
|--------|-------------|------------|-------|
| EP1AGX20 | — | — | ✔ |
| EP1AGX35 | — | — | ✔ |
| EP1AGX50 | — | — | ✔ |
| EP1AGX60 | — | — | ✔ |
| EP1AGX90 | — | — | ✔ |

## I/O Timing Measurement Methodology

Different I/O standards require different baseline loading techniques for reporting timing delays. Altera characterizes timing delays with the required termination for each I/O standard and with 0 pF (except for PCI and PCI-X which use 10 pF) loading and the timing is specified up to the output pin of the FPGA device. The Quartus II software calculates the I/O timing for each I/O standard with a default baseline loading as specified by the I/O standards.

The following measurements are made during device characterization. Altera measures clock-to-output delays ($t_{CO}$) at worst-case process, minimum voltage, and maximum temperature (PVT) for default loading conditions shown in Table 4–44.

Use the following equations to calculate clock pin to output pin timing for Arria GX devices:

**Equation 4–1.**

$t_{CO}$ from clock pin to I/O pin = delay from clock pad to I/O output register + IOE output register clock-to-output delay + delay from output register to output pin + I/O output delay

$t_{xz}/t_{zx}$ from clock pin to I/O pin = delay from clock pad to I/O output register + IOE output register clock-to-output delay + delay from output register to output pin + I/O output delay + output enable pin delay

Simulation using IBIS models is required to determine the delays on the PCB traces in addition to the output pin delay timing reported by the Quartus II software and the timing model in the device handbook.

1. Simulate the output driver of choice into the generalized test setup, using values from Table 4–44.

2. Record the time to $V_{MEAS}$.

3. Simulate the output driver of choice into the actual PCB trace and load, using the appropriate IBIS model or capacitance value to represent the load.

4. Record the time to $V_{MEAS}$.

5. Compare the results of steps 2 and 4. The increase or decrease in delay should be added to or subtracted from the I/O Standard Output Adder delays to yield the actual worst-case propagation delay (clock-to-output) of the PCB trace.

The Quartus II software reports the timing with the conditions shown in Table 4–44 using the above equation. Figure 4–7 shows the model of the circuit that is represented by the output timing of the Quartus II software.

**Figure 4–7.** Output Delay Timing Reporting Setup Modeled by Quartus II
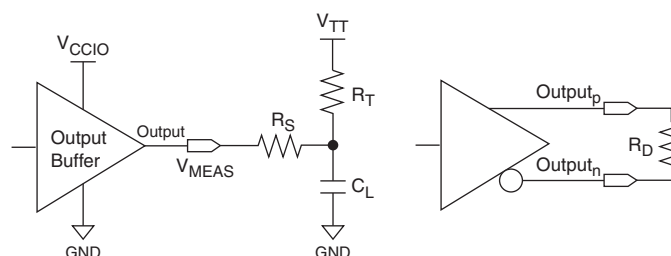


Notes to **Figure 4–7**:

(1) Output pin timing is reported at the output pin of the FPGA device. Additional delays for loading and board trace delay need to be accounted for with IBIS model simulations.

(2) $V_{CCPD}$ is 3.085 V unless otherwise specified.

(3) $V_{CCINT}$ is 1.12 V unless otherwise specified.

**Table 4–44.** Output Timing Measurement Methodology for Output Pins   *(Note 1), (2), (3)*

| I/O Standard | Loading and Termination | | | | | | Measurement Point |
|---|---|---|---|---|---|---|---|
| | $R_S$ ($\Omega$) | $R_D$ ($\Omega$) | $R_T$ ($\Omega$) | $V_{CCIO}$ (V) | $V_{TT}$ (V) | $C_L$ (pF) | $V_{MEAS}$ (V) |
| LVTTL *(4)* | — | — | — | 3.135 | — | 0 | 1.5675 |
| LVCMOS *(4)* | — | — | — | 3.135 | — | 0 | 1.5675 |
| 2.5 V *(4)* | — | — | — | 2.375 | — | 0 | 1.1875 |
| 1.8 V *(4)* | — | — | — | 1.710 | — | 0 | 0.855 |
| 1.5 V *(4)* | — | — | — | 1.425 | — | 0 | 0.7125 |
| PCI *(5)* | — | — | — | 2.970 | — | 10 | 1.485 |
| PCI-X *(5)* | — | — | — | 2.970 | — | 10 | 1.485 |
| SSTL-2 Class I | 25 | — | 50 | 2.325 | 1.123 | 0 | 1.1625 |
| SSTL-2 Class II | 25 | — | 25 | 2.325 | 1.123 | 0 | 1.1625 |
| SSTL-18 Class I | 25 | — | 50 | 1.660 | 0.790 | 0 | 0.83 |
| SSTL-18 Class II | 25 | — | 25 | 1.660 | 0.790 | 0 | 0.83 |
| 1.8-V HSTL Class I | — | — | 50 | 1.660 | 0.790 | 0 | 0.83 |
| 1.8-V HSTL Class II | — | — | 25 | 1.660 | 0.790 | 0 | 0.83 |
| 1.5-V HSTL Class I | — | — | 50 | 1.375 | 0.648 | 0 | 0.6875 |
| 1.5-V HSTL Class II | — | — | 25 | 1.375 | 0.648 | 0 | 0.6875 |
| 1.2-V HSTL with OCT | — | — | — | 1.140 | — | 0 | 0.570 |
| Differential SSTL-2 Class I | 25 | — | 50 | 2.325 | 1.123 | 0 | 1.1625 |
| Differential SSTL-2 Class II | 25 | — | 25 | 2.325 | 1.123 | 0 | 1.1625 |
| Differential SSTL-18 Class I | 50 | — | 50 | 1.660 | 0.790 | 0 | 0.83 |
| Differential SSTL-18 Class II | 25 | — | 25 | 1.660 | 0.790 | 0 | 0.83 |
| 1.5-V differential HSTL Class I | — | — | 50 | 1.375 | 0.648 | 0 | 0.6875 |
| 1.5-V differential HSTL Class II | — | — | 25 | 1.375 | 0.648 | 0 | 0.6875 |
| 1.8-V differential HSTL Class I | — | — | 50 | 1.660 | 0.790 | 0 | 0.83 |

**Table 4–44.** Output Timing Measurement Methodology for Output Pins *(Note 1), (2), (3)*

| I/O Standard | Loading and Termination | | | | | | Measurement Point |
|---|---|---|---|---|---|---|---|
| | $R_S$ ($\Omega$) | $R_D$ ($\Omega$) | $R_T$ ($\Omega$) | $V_{CCIO}$ (V) | $V_{TT}$ (V) | $C_L$ (pF) | $V_{MEAS}$ (V) |
| 1.8-V differential HSTL Class II | — | — | 25 | 1.660 | 0.790 | 0 | 0.83 |
| LVDS | — | 100 | — | 2.325 | — | 0 | 1.1625 |
| LVPECL | — | 100 | — | 3.135 | — | 0 | 1.5675 |

**Notes to Table 4–44:**

(1) Input measurement point at internal node is 0.5 $V_{CCINT}$.

(2) Output measuring point for $V_{MEAS}$ at buffer output is 0.5 $V_{CCIO}$.

(3) Input stimulus edge rate is 0 to $V_{CC}$ in 0.2 ns (internal signal) from the driver preceding the I/O buffer.

(4) Less than 50-mV ripple on $V_{CCIO}$ and $V_{CCPD}$, $V_{CCINT}$ = 1.15 V with less than 30-mV ripple.

(5) $V_{CCPD}$ = 2.97 V, less than 50-mV ripple on $V_{CCIO}$ and $V_{CCPD}$, $V_{CCINT}$ = 1.15 V.

Figure 4–8 and Figure 4–9 show the measurement setup for output disable and output enable timing.

**Figure 4–8.** Measurement Setup for $t_{xz}$ *(Note 1)*
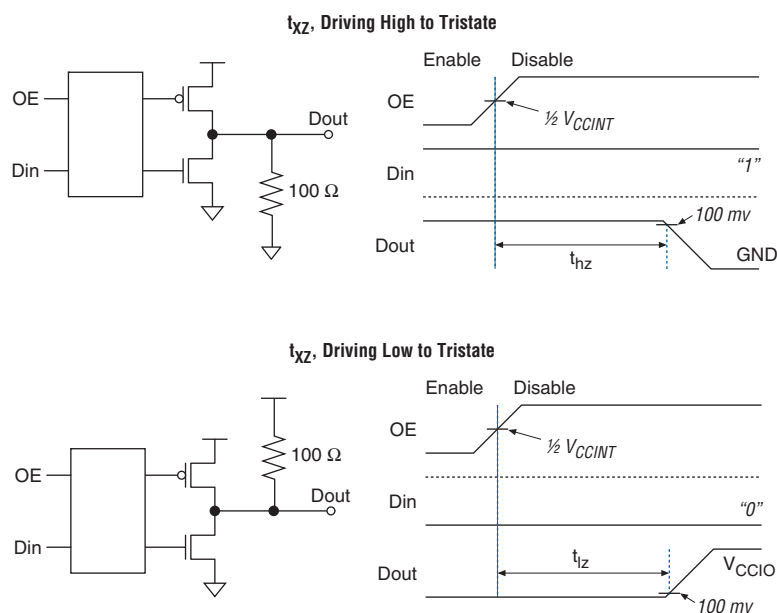


**Note to Figure 4–8:**

(1) $V_{CCINT}$ is 1.12 V for this measurement.

**Figure 4–9.** Measurement Setup for $t_{zx}$
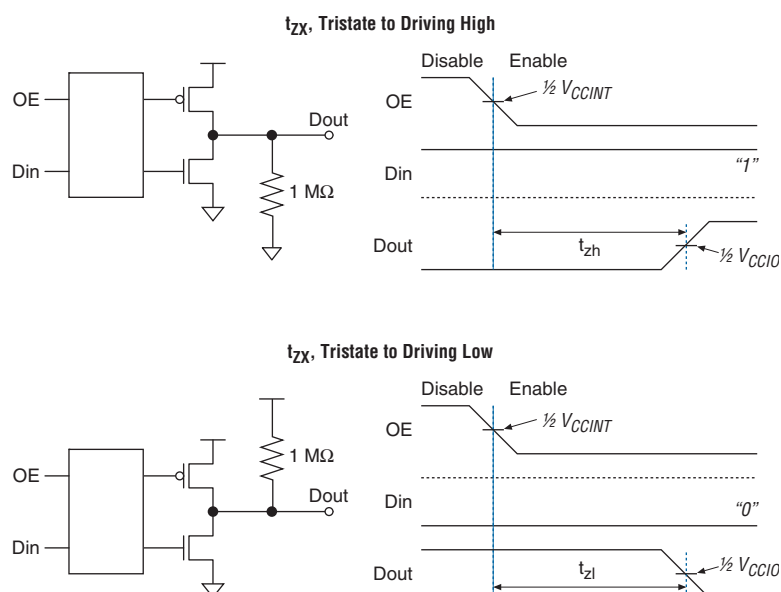


Table 4–45 specifies the input timing measurement setup.

**Table 4–45.** Timing Measurement Methodology for Input Pins    *(Note 1)*, *(2)*, *(3)*, *(4)*  (Part 1 of 2)

| I/O Standard | Measurement Conditions | | | Measurement Point |
|---|---|---|---|---|
| | $V_{CCIO}$ (V) | $V_{REF}$ (V) | Edge Rate (ns) | VMEAS (V) |
| LVTTL *(5)* | 3.135 | — | 3.135 | 1.5675 |
| LVCMOS *(5)* | 3.135 | — | 3.135 | 1.5675 |
| 2.5 V *(5)* | 2.375 | — | 2.375 | 1.1875 |
| 1.8 V *(5)* | 1.710 | — | 1.710 | 0.855 |
| 1.5 V *(5)* | 1.425 | — | 1.425 | 0.7125 |
| PCI *(6)* | 2.970 | — | 2.970 | 1.485 |
| PCI-X *(6)* | 2.970 | — | 2.970 | 1.485 |
| SSTL-2 Class I | 2.325 | 1.163 | 2.325 | 1.1625 |
| SSTL-2 Class II | 2.325 | 1.163 | 2.325 | 1.1625 |
| SSTL-18 Class I | 1.660 | 0.830 | 1.660 | 0.83 |
| SSTL-18 Class II | 1.660 | 0.830 | 1.660 | 0.83 |
| 1.8-V HSTL Class I | 1.660 | 0.830 | 1.660 | 0.83 |
| 1.8-V HSTL Class II | 1.660 | 0.830 | 1.660 | 0.83 |
| 1.5-V HSTL Class I | 1.375 | 0.688 | 1.375 | 0.6875 |
| 1.5-V HSTL Class II | 1.375 | 0.688 | 1.375 | 0.6875 |
| 1.2-V HSTL with OCT | 1.140 | 0.570 | 1.140 | 0.570 |
| Differential SSTL-2 Class I | 2.325 | 1.163 | 2.325 | 1.1625 |
| Differential SSTL-2 Class II | 2.325 | 1.163 | 2.325 | 1.1625 |
| Differential SSTL-18 Class I | 1.660 | 0.830 | 1.660 | 0.83 |

**Table 4–45.** Timing Measurement Methodology for Input Pins    *(Note 1)*, *(2)*, *(3)*, *(4)*  (Part 2 of 2)

| I/O Standard | Measurement Conditions | | | Measurement Point |
|---|---|---|---|---|
| | $V_{CCIO}$ (V) | $V_{REF}$ (V) | Edge Rate (ns) | VMEAS (V) |
| Differential SSTL-18 Class II | 1.660 | 0.830 | 1.660 | 0.83 |
| 1.5-V differential HSTL Class I | 1.375 | 0.688 | 1.375 | 0.6875 |
| 1.5-V differential HSTL Class II | 1.375 | 0.688 | 1.375 | 0.6875 |
| 1.8-V differential HSTL Class I | 1.660 | 0.830 | 1.660 | 0.83 |
| 1.8-V differential HSTL Class II | 1.660 | 0.830 | 1.660 | 0.83 |
| LVDS | 2.325 | — | 0.100 | 1.1625 |
| LVPECL | 3.135 | — | 0.100 | 1.5675 |

**Notes to Table 4–45:**

(1)  Input buffer sees no load at buffer input.

(2)  Input measuring point at buffer input is 0.5 $V_{CCIO}$.

(3)  Output measuring point is 0.5 $V_{CC}$ at internal node.

(4)  Input edge rate is 1 V/ns.

(5)  Less than 50-mV ripple on $V_{CCIO}$ and $V_{CCPD}$, $V_{CCINT}$ = 1.15 V with less than 30-mV ripple.

(6)  $V_{CCPD}$ = 2.97 V, less than 50-mV ripple on $V_{CCIO}$ and $V_{CCPD}$, $V_{CCINT}$ = 1.15 V.

## Clock Network Skew Adders

The Quartus II software models skew within dedicated clock networks such as global and regional clocks. Therefore, the intra-clock network skew adder is not specified. Table 4–46 specifies the intra clock skew between any two clock networks driving any registers in the Arria GX device.

**Table 4–46.** Clock Network Specifications

| Name | Description | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| Clock skew adder EP1AGX20/35 *(1)* | Inter-clock network, same side | — | — | ± 50 | ps |
| | Inter-clock network, entire chip | — | — | ± 100 | ps |
| Clock skew adder EP1AGX50/60 *(1)* | Inter-clock network, same side | — | — | ± 50 | ps |
| | Inter-clock network, entire chip | — | — | ± 100 | ps |
| Clock skew adder EP1AGX90 *(1)* | Inter-clock network, same side | — | — | ± 55 | ps |
| | Inter-clock network, entire chip | — | — | ± 110 | ps |

**Note to Table 4–46:**

(1)  This is in addition to intra-clock network skew, which is modeled in the Quartus II software.

## Default Capacitive Loading of Different I/O Standards

See Table 4–47 for default capacitive loading of different I/O standards.

**Table 4–47.** Default Loading of Different I/O Standards for Arria GX Devices  (Part 1 of 2)

| I/O Standard | Capacitive Load | Units |
|---|---|---|
| LVTTL | 0 | pF |
| LVCMOS | 0 | pF |
| 2.5 V | 0 | pF |

**Table 4–47.** Default Loading of Different I/O Standards for Arria GX Devices  (Part 2 of 2)

| I/O Standard | Capacitive Load | Units |
|---|---|---|
| 1.8 V | 0 | pF |
| 1.5 V | 0 | pF |
| PCI | 10 | pF |
| PCI-X | 10 | pF |
| SSTL-2 Class I | 0 | pF |
| SSTL-2 Class II | 0 | pF |
| SSTL-18 Class I | 0 | pF |
| SSTL-18 Class II | 0 | pF |
| 1.5-V HSTL Class I | 0 | pF |
| 1.5-V HSTL Class II | 0 | pF |
| 1.8-V HSTL Class I | 0 | pF |
| 1.8-V HSTL Class II | 0 | pF |
| Differential SSTL-2 Class I | 0 | pF |
| Differential SSTL-2 Class II | 0 | pF |
| Differential SSTL-18 Class I | 0 | pF |
| Differential SSTL-18 Class II | 0 | pF |
| 1.5-V differential HSTL Class I | 0 | pF |
| 1.5-V differential HSTL Class II | 0 | pF |
| 1.8-V differential HSTL Class I | 0 | pF |
| 1.8-V differential HSTL Class II | 0 | pF |
| LVDS | 0 | pF |

# Typical Design Performance

The following section describes the typical design performance for the Arria GX device family.

## User I/O Pin Timing

Table 4–48 through Table 4–77 show user I/O pin timing for Arria GX devices. I/O buffer $t_{SU}$, $t_H$, and $t_{CO}$ are reported for the cases when I/O clock is driven by a non-PLL global clock (GCLK) and a PLL driven global clock (GCLK-PLL). For $t_{SU}$, $t_H$, and $t_{CO}$ using regional clock, add the value from the adder tables listed for each device to the GCLK/GCLK-PLL values for the device.

### EP1AGX20 I/O Timing Parameters

Table 4–48 through Table 4–51 show the maximum I/O timing parameters for EP1AGX20 devices for I/O standards which support general purpose I/O pins.

Table 4–48 describes the row pin delay adders when using the regional clock in Arria GX devices.

**Table 4–48.** EP1AGX20 Row Pin Delay Adders for Regional Clock

| Parameter | Fast Corner | | –6 Speed Grade | Units |
|---|---|---|---|---|
| | Industrial | Commercial | | |
| RCLK input adder | 0.117 | 0.117 | 0.273 | ns |
| RCLK PLL input adder | 0.011 | 0.011 | 0.019 | ns |
| RCLK output adder | −0.117 | −0.117 | −0.273 | ns |
| RCLK PLL output adder | −0.011 | −0.011 | −0.019 | ns |

Table 4–49 describes I/O timing specifications.

**Table 4–49.** EP1AGX20 Column Pins Input Timing Parameters   (Part 1 of 3)

| I/O Standard | Clock | Parameter | Fast Corner | | –6 Speed Grade | Units |
|---|---|---|---|---|---|---|
| | | | Industrial | Commercial | | |
| 3.3-V LVTTL | GCLK | $t_{SU}$ | 1.251 | 1.251 | 2.915 | ns |
| | | $t_H$ | −1.146 | −1.146 | −2.638 | ns |
| | GCLK PLL | $t_{SU}$ | 2.693 | 2.693 | 6.021 | ns |
| | | $t_H$ | −2.588 | −2.588 | −5.744 | ns |
| 3.3-V LVCMOS | GCLK | $t_{SU}$ | 1.251 | 1.251 | 2.915 | ns |
| | | $t_H$ | −1.146 | −1.146 | −2.638 | ns |
| | GCLK PLL | $t_{SU}$ | 2.693 | 2.693 | 6.021 | ns |
| | | $t_H$ | −2.588 | −2.588 | −5.744 | ns |
| 2.5 V | GCLK | $t_{SU}$ | 1.261 | 1.261 | 2.897 | ns |
| | | $t_H$ | −1.156 | −1.156 | −2.620 | ns |
| | GCLK PLL | $t_{SU}$ | 2.703 | 2.703 | 6.003 | ns |
| | | $t_H$ | −2.598 | −2.598 | −5.726 | ns |
| 1.8 V | GCLK | $t_{SU}$ | 1.327 | 1.327 | 3.107 | ns |
| | | $t_H$ | −1.222 | −1.222 | −2.830 | ns |
| | GCLK PLL | $t_{SU}$ | 2.769 | 2.769 | 6.213 | ns |
| | | $t_H$ | −2.664 | −2.664 | −5.936 | ns |
| 1.5 V | GCLK | $t_{SU}$ | 1.330 | 1.330 | 3.200 | ns |
| | | $t_H$ | −1.225 | −1.225 | −2.923 | ns |
| | GCLK PLL | $t_{SU}$ | 2.772 | 2.772 | 6.306 | ns |
| | | $t_H$ | −2.667 | −2.667 | −6.029 | ns |
| SSTL-2 CLASS I | GCLK | $t_{SU}$ | 1.075 | 1.075 | 2.372 | ns |
| | | $t_H$ | −0.970 | −0.970 | −2.095 | ns |
| | GCLK PLL | $t_{SU}$ | 2.517 | 2.517 | 5.480 | ns |
| | | $t_H$ | −2.412 | −2.412 | −5.203 | ns |

**Table 4–49.** EP1AGX20 Column Pins Input Timing Parameters (Part 2 of 3)

| I/O Standard | Clock | Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|
| | | | Industrial | Commercial | | |
| SSTL-2 CLASS II | GCLK | $t_{SU}$ | 1.075 | 1.075 | 2.372 | ns |
| | | $t_H$ | −0.970 | −0.970 | −2.095 | ns |
| | GCLK PLL | $t_{SU}$ | 2.517 | 2.517 | 5.480 | ns |
| | | $t_H$ | −2.412 | −2.412 | −5.203 | ns |
| SSTL-18 CLASS I | GCLK | $t_{SU}$ | 1.113 | 1.113 | 2.479 | ns |
| | | $t_H$ | −1.008 | −1.008 | −2.202 | ns |
| | GCLK PLL | $t_{SU}$ | 2.555 | 2.555 | 5.585 | ns |
| | | $t_H$ | −2.450 | −2.450 | −5.308 | ns |
| SSTL-18 CLASS II | GCLK | $t_{SU}$ | 1.114 | 1.114 | 2.479 | ns |
| | | $t_H$ | −1.009 | −1.009 | −2.202 | ns |
| | GCLK PLL | $t_{SU}$ | 2.556 | 2.556 | 5.587 | ns |
| | | $t_H$ | −2.451 | −2.451 | −5.310 | ns |
| 1.8-V HSTL CLASS I | GCLK | $t_{SU}$ | 1.113 | 1.113 | 2.479 | ns |
| | | $t_H$ | −1.008 | −1.008 | −2.202 | ns |
| | GCLK PLL | $t_{SU}$ | 2.555 | 2.555 | 5.585 | ns |
| | | $t_H$ | −2.450 | −2.450 | −5.308 | ns |
| 1.8-V HSTL CLASS II | GCLK | $t_{SU}$ | 1.114 | 1.114 | 2.479 | ns |
| | | $t_H$ | −1.009 | −1.009 | −2.202 | ns |
| | GCLK PLL | $t_{SU}$ | 2.556 | 2.556 | 5.587 | ns |
| | | $t_H$ | −2.451 | −2.451 | −5.310 | ns |
| 1.5-V HSTL CLASS I | GCLK | $t_{SU}$ | 1.131 | 1.131 | 2.607 | ns |
| | | $t_H$ | −1.026 | −1.026 | −2.330 | ns |
| | GCLK PLL | $t_{SU}$ | 2.573 | 2.573 | 5.713 | ns |
| | | $t_H$ | −2.468 | −2.468 | −5.436 | ns |
| 1.5-V HSTL CLASS II | GCLK | $t_{SU}$ | 1.132 | 1.132 | 2.607 | ns |
| | | $t_H$ | −1.027 | −1.027 | −2.330 | ns |
| | GCLK PLL | $t_{SU}$ | 2.574 | 2.574 | 5.715 | ns |
| | | $t_H$ | −2.469 | −2.469 | −5.438 | ns |
| 3.3-V PCI | GCLK | $t_{SU}$ | 1.256 | 1.256 | 2.903 | ns |
| | | $t_H$ | −1.151 | −1.151 | −2.626 | ns |
| | GCLK PLL | $t_{SU}$ | 2.698 | 2.698 | 6.009 | ns |
| | | $t_H$ | −2.593 | −2.593 | −5.732 | ns |
| 3.3-V PCI-X | GCLK | $t_{SU}$ | 1.256 | 1.256 | 2.903 | ns |
| | | $t_H$ | −1.151 | −1.151 | −2.626 | ns |
| | GCLK PLL | $t_{SU}$ | 2.698 | 2.698 | 6.009 | ns |
| | | $t_H$ | −2.593 | −2.593 | −5.732 | ns |

**Table 4–49.** EP1AGX20 Column Pins Input Timing Parameters (Part 3 of 3)

| I/O Standard | Clock | Parameter | Fast Corner | | –6 Speed Grade | Units |
|---|---|---|---|---|---|---|
| | | | Industrial | Commercial | | |
| LVDS | GCLK | $t_{SU}$ | 1.106 | 1.106 | 2.489 | ns |
| | | $t_H$ | −1.001 | −1.001 | −2.212 | ns |
| | GCLK PLL | $t_{SU}$ | 2.530 | 2.530 | 5.564 | ns |
| | | $t_H$ | −2.425 | −2.425 | −5.287 | ns |

Table 4–50 describes I/O timing specifications.

**Table 4–50.** EP1AGX20 Row Pins output Timing Parameters (Part 1 of 2)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Model | | –6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| 3.3-V LVTTL | 4 mA | GCLK | $t_{CO}$ | 2.904 | 2.904 | 6.699 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.485 | 1.485 | 3.627 | ns |
| 3.3-V LVTTL | 8 mA | GCLK | $t_{CO}$ | 2.776 | 2.776 | 6.059 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.357 | 1.357 | 2.987 | ns |
| 3.3-V LVTTL | 12 mA | GCLK | $t_{CO}$ | 2.720 | 2.720 | 6.022 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.301 | 1.301 | 2.950 | ns |
| 3.3-V LVCMOS | 4 mA | GCLK | $t_{CO}$ | 2.776 | 2.776 | 6.059 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.357 | 1.357 | 2.987 | ns |
| 3.3-V LVCMOS | 8 mA | GCLK | $t_{CO}$ | 2.670 | 2.670 | 5.753 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.251 | 1.251 | 2.681 | ns |
| 2.5 V | 4 mA | GCLK | $t_{CO}$ | 2.759 | 2.759 | 6.033 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.340 | 1.340 | 2.961 | ns |
| 2.5 V | 8 mA | GCLK | $t_{CO}$ | 2.656 | 2.656 | 5.775 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.237 | 1.237 | 2.703 | ns |
| 2.5 V | 12 mA | GCLK | $t_{CO}$ | 2.637 | 2.637 | 5.661 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.218 | 1.218 | 2.589 | ns |
| 1.8 V | 2 mA | GCLK | $t_{CO}$ | 2.829 | 2.829 | 7.052 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.410 | 1.410 | 3.980 | ns |
| 1.8 V | 4 mA | GCLK | $t_{CO}$ | 2.818 | 2.818 | 6.273 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.399 | 1.399 | 3.201 | ns |
| 1.8 V | 6 mA | GCLK | $t_{CO}$ | 2.707 | 2.707 | 5.972 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.288 | 1.288 | 2.900 | ns |
| 1.8 V | 8 mA | GCLK | $t_{CO}$ | 2.676 | 2.676 | 5.858 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.257 | 1.257 | 2.786 | ns |
| 1.5 V | 2 mA | GCLK | $t_{CO}$ | 2.789 | 2.789 | 6.551 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.370 | 1.370 | 3.479 | ns |
| 1.5 V | 4 mA | GCLK | $t_{CO}$ | 2.682 | 2.682 | 5.950 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.263 | 1.263 | 2.878 | ns |

**Table 4–50.** EP1AGX20 Row Pins output Timing Parameters  (Part 2 of 2)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Model | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| SSTL-2 CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.626 | 2.626 | 5.614 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.207 | 1.207 | 2.542 | ns |
| SSTL-2 CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.602 | 2.602 | 5.538 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.183 | 1.183 | 2.466 | ns |
| SSTL-2 CLASS II | 16 mA | GCLK | $t_{CO}$ | 2.568 | 2.568 | 5.407 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.149 | 1.149 | 2.335 | ns |
| SSTL-18 CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.614 | 2.614 | 5.556 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.195 | 1.195 | 2.484 | ns |
| SSTL-18 CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.618 | 2.618 | 5.485 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.199 | 1.199 | 2.413 | ns |
| SSTL-18 CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.594 | 2.594 | 5.468 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.175 | 1.175 | 2.396 | ns |
| SSTL-18 CLASS I | 10 mA | GCLK | $t_{CO}$ | 2.597 | 2.597 | 5.447 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.178 | 1.178 | 2.375 | ns |
| 1.8-V HSTL CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.595 | 2.595 | 5.466 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.176 | 1.176 | 2.394 | ns |
| 1.8-V HSTL CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.598 | 2.598 | 5.430 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.179 | 1.179 | 2.358 | ns |
| 1.8-V HSTL CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.580 | 2.580 | 5.426 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.161 | 1.161 | 2.354 | ns |
| 1.8-V HSTL CLASS I | 10 mA | GCLK | $t_{CO}$ | 2.584 | 2.584 | 5.415 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.165 | 1.165 | 2.343 | ns |
| 1.8-V HSTL CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.575 | 2.575 | 5.414 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.156 | 1.156 | 2.342 | ns |
| 1.5-V HSTL CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.594 | 2.594 | 5.443 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.175 | 1.175 | 2.371 | ns |
| 1.5-V HSTL CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.597 | 2.597 | 5.429 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.178 | 1.178 | 2.357 | ns |
| 1.5-V HSTL CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.582 | 2.582 | 5.421 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.163 | 1.163 | 2.349 | ns |
| LVDS | — | GCLK | $t_{CO}$ | 2.654 | 2.654 | 5.613 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.226 | 1.226 | 2.530 | ns |

Table 4–51 describes I/O timing specifications.

**Table 4–51.**  EP1AGX20 Column Pins Output Timing Parameters  (Part 1 of 4)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Corner | | –6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| 3.3-V LVTTL | 4 mA | GCLK | $t_{CO}$ | 2.909 | 2.909 | 6.541 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.467 | 1.467 | 3.435 | ns |
| 3.3-V LVTTL | 8 mA | GCLK | $t_{CO}$ | 2.764 | 2.764 | 6.169 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.322 | 1.322 | 3.063 | ns |
| 3.3-V LVTTL | 12 mA | GCLK | $t_{CO}$ | 2.697 | 2.697 | 6.169 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.255 | 1.255 | 3.063 | ns |
| 3.3-V LVTTL | 16 mA | GCLK | $t_{CO}$ | 2.671 | 2.671 | 6.000 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.229 | 1.229 | 2.894 | ns |
| 3.3-V LVTTL | 20 mA | GCLK | $t_{CO}$ | 2.649 | 2.649 | 5.875 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.207 | 1.207 | 2.769 | ns |
| 3.3-V LVTTL | 24 mA | GCLK | $t_{CO}$ | 2.642 | 2.642 | 5.877 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.200 | 1.200 | 2.771 | ns |
| 3.3-V LVCMOS | 4 mA | GCLK | $t_{CO}$ | 2.764 | 2.764 | 6.169 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.322 | 1.322 | 3.063 | ns |
| 3.3-V LVCMOS | 8 mA | GCLK | $t_{CO}$ | 2.672 | 2.672 | 5.874 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.230 | 1.230 | 2.768 | ns |
| 3.3-V LVCMOS | 12 mA | GCLK | $t_{CO}$ | 2.644 | 2.644 | 5.796 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.202 | 1.202 | 2.690 | ns |
| 3.3-V LVCMOS | 16 mA | GCLK | $t_{CO}$ | 2.651 | 2.651 | 5.764 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.209 | 1.209 | 2.658 | ns |
| 3.3-V LVCMOS | 20 mA | GCLK | $t_{CO}$ | 2.638 | 2.638 | 5.746 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.196 | 1.196 | 2.640 | ns |
| 3.3-V LVCMOS | 24 mA | GCLK | $t_{CO}$ | 2.627 | 2.627 | 5.724 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.185 | 1.185 | 2.618 | ns |
| 2.5 V | 4 mA | GCLK | $t_{CO}$ | 2.726 | 2.726 | 6.201 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.284 | 1.284 | 3.095 | ns |
| 2.5 V | 8 mA | GCLK | $t_{CO}$ | 2.674 | 2.674 | 5.939 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.232 | 1.232 | 2.833 | ns |
| 2.5 V | 12 mA | GCLK | $t_{CO}$ | 2.653 | 2.653 | 5.822 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.211 | 1.211 | 2.716 | ns |
| 2.5 V | 16 mA | GCLK | $t_{CO}$ | 2.635 | 2.635 | 5.748 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.193 | 1.193 | 2.642 | ns |
| 1.8 V | 2 mA | GCLK | $t_{CO}$ | 2.766 | 2.766 | 7.193 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.324 | 1.324 | 4.087 | ns |
| 1.8 V | 4 mA | GCLK | $t_{CO}$ | 2.771 | 2.771 | 6.419 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.329 | 1.329 | 3.313 | ns |

**Table 4–51.** EP1AGX20 Column Pins Output Timing Parameters  (Part 2 of 4)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| 1.8 V | 6 mA | GCLK | $t_{CO}$ | 2.695 | 2.695 | 6.155 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.253 | 1.253 | 3.049 | ns |
| 1.8 V | 8 mA | GCLK | $t_{CO}$ | 2.697 | 2.697 | 6.064 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.255 | 1.255 | 2.958 | ns |
| 1.8 V | 10 mA | GCLK | $t_{CO}$ | 2.651 | 2.651 | 5.987 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.209 | 1.209 | 2.881 | ns |
| 1.8 V | 12 mA | GCLK | $t_{CO}$ | 2.652 | 2.652 | 5.930 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.210 | 1.210 | 2.824 | ns |
| 1.5 V | 2 mA | GCLK | $t_{CO}$ | 2.746 | 2.746 | 6.723 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.304 | 1.304 | 3.617 | ns |
| 1.5 V | 4 mA | GCLK | $t_{CO}$ | 2.682 | 2.682 | 6.154 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.240 | 1.240 | 3.048 | ns |
| 1.5 V | 6 mA | GCLK | $t_{CO}$ | 2.685 | 2.685 | 6.036 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.243 | 1.243 | 2.930 | ns |
| 1.5 V | 8 mA | GCLK | $t_{CO}$ | 2.644 | 2.644 | 5.983 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.202 | 1.202 | 2.877 | ns |
| SSTL-2 CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.629 | 2.629 | 5.762 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.184 | 1.184 | 2.650 | ns |
| SSTL-2 CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.612 | 2.612 | 5.712 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.167 | 1.167 | 2.600 | ns |
| SSTL-2 CLASS II | 16 mA | GCLK | $t_{CO}$ | 2.590 | 2.590 | 5.639 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.145 | 1.145 | 2.527 | ns |
| SSTL-2 CLASS II | 20 mA | GCLK | $t_{CO}$ | 2.591 | 2.591 | 5.626 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.146 | 1.146 | 2.514 | ns |
| SSTL-2 CLASS II | 24 mA | GCLK | $t_{CO}$ | 2.587 | 2.587 | 5.624 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.142 | 1.142 | 2.512 | ns |
| SSTL-18 CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.626 | 2.626 | 5.733 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.184 | 1.184 | 2.627 | ns |
| SSTL-18 CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.630 | 2.630 | 5.694 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.185 | 1.185 | 2.582 | ns |
| SSTL-18 CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.609 | 2.609 | 5.675 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.164 | 1.164 | 2.563 | ns |
| SSTL-18 CLASS I | 10 mA | GCLK | $t_{CO}$ | 2.614 | 2.614 | 5.673 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.169 | 1.169 | 2.561 | ns |
| SSTL-18 CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.608 | 2.608 | 5.659 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.163 | 1.163 | 2.547 | ns |
| SSTL-18 CLASS II | 8 mA | GCLK | $t_{CO}$ | 2.597 | 2.597 | 5.625 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.152 | 1.152 | 2.513 | ns |

**Table 4–51.** EP1AGX20 Column Pins Output Timing Parameters  (Part 3 of 4)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| SSTL-18 CLASS II | 16 mA | GCLK | $t_{CO}$ | 2.609 | 2.609 | 5.603 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.164 | 1.164 | 2.491 | ns |
| SSTL-18 CLASS II | 18 mA | GCLK | $t_{CO}$ | 2.605 | 2.605 | 5.611 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.160 | 1.160 | 2.499 | ns |
| SSTL-18 CLASS II | 20 mA | GCLK | $t_{CO}$ | 2.605 | 2.605 | 5.609 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.160 | 1.160 | 2.497 | ns |
| 1.8-V HSTL CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.629 | 2.629 | 5.664 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.187 | 1.187 | 2.558 | ns |
| 1.8-V HSTL CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.634 | 2.634 | 5.649 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.189 | 1.189 | 2.537 | ns |
| 1.8-V HSTL CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.612 | 2.612 | 5.638 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.167 | 1.167 | 2.526 | ns |
| 1.8-V HSTL CLASS I | 10 mA | GCLK | $t_{CO}$ | 2.616 | 2.616 | 5.644 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.171 | 1.171 | 2.532 | ns |
| 1.8-V HSTL CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.608 | 2.608 | 5.637 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.163 | 1.163 | 2.525 | ns |
| 1.8-V HSTL CLASS II | 16 mA | GCLK | $t_{CO}$ | 2.591 | 2.591 | 5.401 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.146 | 1.146 | 2.289 | ns |
| 1.8-V HSTL CLASS II | 18 mA | GCLK | $t_{CO}$ | 2.593 | 2.593 | 5.412 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.148 | 1.148 | 2.300 | ns |
| 1.8-V HSTL CLASS II | 20 mA | GCLK | $t_{CO}$ | 2.593 | 2.593 | 5.421 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.148 | 1.148 | 2.309 | ns |
| 1.5-V HSTL CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.629 | 2.629 | 5.663 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.187 | 1.187 | 2.557 | ns |
| 1.5-V HSTL CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.633 | 2.633 | 5.641 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.188 | 1.188 | 2.529 | ns |
| 1.5-V HSTL CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.615 | 2.615 | 5.643 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.170 | 1.170 | 2.531 | ns |
| 1.5-V HSTL CLASS I | 10 mA | GCLK | $t_{CO}$ | 2.615 | 2.615 | 5.645 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.170 | 1.170 | 2.533 | ns |
| 1.5-V HSTL CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.609 | 2.609 | 5.643 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.164 | 1.164 | 2.531 | ns |
| 1.5-V HSTL CLASS II | 16 mA | GCLK | $t_{CO}$ | 2.596 | 2.596 | 5.455 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.151 | 1.151 | 2.343 | ns |
| 1.5-V HSTL CLASS II | 18 mA | GCLK | $t_{CO}$ | 2.599 | 2.599 | 5.465 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.154 | 1.154 | 2.353 | ns |
| 1.5-V HSTL CLASS II | 20 mA | GCLK | $t_{CO}$ | 2.601 | 2.601 | 5.478 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.156 | 1.156 | 2.366 | ns |

**Table 4–51.** EP1AGX20 Column Pins Output Timing Parameters  (Part 4 of 4)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| 3.3-V PCI | — | GCLK | $t_{CO}$ | 2.755 | 2.755 | 5.791 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.313 | 1.313 | 2.685 | ns |
| 3.3-V PCI-X | — | GCLK | $t_{CO}$ | 2.755 | 2.755 | 5.791 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.313 | 1.313 | 2.685 | ns |
| LVDS | — | GCLK | $t_{CO}$ | 3.621 | 3.621 | 6.969 | ns |
| | | GCLK PLL | $t_{CO}$ | 2.190 | 2.190 | 3.880 | ns |

Table 4–52 through Table 4–53 list EP1AGX20 regional clock (RCLK) adder values that should be added to GCLK values. These adder values are used to determine I/O timing when the I/O pin is driven using the regional clock. This applies for all I/O standards supported by Arria GX with general purpose I/O pins.

Table 4–52 describes row pin delay adders when using the regional clock in Arria GX devices.

**Table 4–52.** EP1AGX20 Row Pin Delay Adders for Regional Clock

| Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|
| | Industrial | Commercial | | |
| RCLK input adder | 0.117 | 0.117 | 0.273 | ns |
| RCLK PLL input adder | 0.011 | 0.011 | 0.019 | ns |
| RCLK output adder | −0.117 | −0.117 | −0.273 | ns |
| RCLK PLL output adder | −0.011 | −0.011 | −0.019 | ns |

Table 4–53 lists column pin delay adders when using the regional clock in Arria GX devices.

**Table 4–53.** EP1AGX20 Column Pin Delay Adders for Regional Clock

| Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|
| | Industrial | Commercial | | |
| RCLK input adder | 0.081 | 0.081 | 0.223 | ns |
| RCLK PLL input adder | −0.012 | −0.012 | −0.008 | ns |
| RCLK output adder | −0.081 | −0.081 | −0.224 | ns |
| RCLK PLL output adder | 1.11 | 1.11 | 2.658 | ns |

## EP1AGX35 I/O Timing Parameters

Table 4–54 through Table 4–57 list the maximum I/O timing parameters for EP1AGX35 devices for I/O standards which support general purpose I/O pins.

Table 4–54 lists I/O timing specifications.

**Table 4–54.** EP1AGX35 Row Pins Input Timing Parameters  (Part 1 of 2)

| I/O Standard | Clock | Parameter | Fast Model | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|
| | | | Industrial | Commercial | | |
| 3.3-V LVTTL | GCLK | $t_{SU}$ | 1.561 | 1.561 | 3.556 | ns |
| | | $t_H$ | −1.456 | −1.456 | −3.279 | ns |
| | GCLK PLL | $t_{SU}$ | 2.980 | 2.980 | 6.628 | ns |
| | | $t_H$ | −2.875 | −2.875 | −6.351 | ns |
| 3.3-V LVCMOS | GCLK | $t_{SU}$ | 1.561 | 1.561 | 3.556 | ns |
| | | $t_H$ | −1.456 | −1.456 | −3.279 | ns |
| | GCLK PLL | $t_{SU}$ | 2.980 | 2.980 | 6.628 | ns |
| | | $t_H$ | −2.875 | −2.875 | −6.351 | ns |
| 2.5 V | GCLK | $t_{SU}$ | 1.573 | 1.573 | 3.537 | ns |
| | | $t_H$ | −1.468 | −1.468 | −3.260 | ns |
| | GCLK PLL | $t_{SU}$ | 2.992 | 2.992 | 6.609 | ns |
| | | $t_H$ | −2.887 | −2.887 | −6.332 | ns |
| 1.8 V | GCLK | $t_{SU}$ | 1.639 | 1.639 | 3.744 | ns |
| | | $t_H$ | −1.534 | −1.534 | −3.467 | ns |
| | GCLK PLL | $t_{SU}$ | 3.058 | 3.058 | 6.816 | ns |
| | | $t_H$ | −2.953 | −2.953 | −6.539 | ns |
| 1.5 V | GCLK | $t_{SU}$ | 1.642 | 1.642 | 3.839 | ns |
| | | $t_H$ | −1.537 | −1.537 | −3.562 | ns |
| | GCLK PLL | $t_{SU}$ | 3.061 | 3.061 | 6.911 | ns |
| | | $t_H$ | −2.956 | −2.956 | −6.634 | ns |
| SSTL-2 CLASS I | GCLK | $t_{SU}$ | 1.385 | 1.385 | 3.009 | ns |
| | | $t_H$ | −1.280 | −1.280 | −2.732 | ns |
| | GCLK PLL | $t_{SU}$ | 2.804 | 2.804 | 6.081 | ns |
| | | $t_H$ | −2.699 | −2.699 | −5.804 | ns |
| SSTL-2 CLASS II | GCLK | $t_{SU}$ | 1.385 | 1.385 | 3.009 | ns |
| | | $t_H$ | −1.280 | −1.280 | −2.732 | ns |
| | GCLK PLL | $t_{SU}$ | 2.804 | 2.804 | 6.081 | ns |
| | | $t_H$ | −2.699 | −2.699 | −5.804 | ns |
| SSTL-18 CLASS I | GCLK | $t_{SU}$ | 1.417 | 1.417 | 3.118 | ns |
| | | $t_H$ | −1.312 | −1.312 | −2.841 | ns |
| | GCLK PLL | $t_{SU}$ | 2.836 | 2.836 | 6.190 | ns |
| | | $t_H$ | −2.731 | −2.731 | −5.913 | ns |

**Table 4–54.** EP1AGX35 Row Pins Input Timing Parameters  (Part 2 of 2)

| I/O Standard | Clock | Parameter | Fast Model | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|
| | | | Industrial | Commercial | | |
| SSTL-18 CLASS II | GCLK | $t_{SU}$ | 1.417 | 1.417 | 3.118 | ns |
| | | $t_H$ | −1.312 | −1.312 | −2.841 | ns |
| | GCLK PLL | $t_{SU}$ | 2.836 | 2.836 | 6.190 | ns |
| | | $t_H$ | −2.731 | −2.731 | −5.913 | ns |
| 1.8-V HSTL CLASS I | GCLK | $t_{SU}$ | 1.417 | 1.417 | 3.118 | ns |
| | | $t_H$ | −1.312 | −1.312 | −2.841 | ns |
| | GCLK PLL | $t_{SU}$ | 2.836 | 2.836 | 6.190 | ns |
| | | $t_H$ | −2.731 | −2.731 | −5.913 | ns |
| 1.8-V HSTL CLASS II | GCLK | $t_{SU}$ | 1.417 | 1.417 | 3.118 | ns |
| | | $t_H$ | −1.312 | −1.312 | −2.841 | ns |
| | GCLK PLL | $t_{SU}$ | 2.836 | 2.836 | 6.190 | ns |
| | | $t_H$ | −2.731 | −2.731 | −5.913 | ns |
| 1.5-V HSTL CLASS I | GCLK | $t_{SU}$ | 1.443 | 1.443 | 3.246 | ns |
| | | $t_H$ | −1.338 | −1.338 | −2.969 | ns |
| | GCLK PLL | $t_{SU}$ | 2.862 | 2.862 | 6.318 | ns |
| | | $t_H$ | −2.757 | −2.757 | −6.041 | ns |
| 1.5-V HSTL CLASS II | GCLK | $t_{SU}$ | 1.443 | 1.443 | 3.246 | ns |
| | | $t_H$ | −1.338 | −1.338 | −2.969 | ns |
| | GCLK PLL | $t_{SU}$ | 2.862 | 2.862 | 6.318 | ns |
| | | $t_H$ | −2.757 | −2.757 | −6.041 | ns |
| LVDS | GCLK | $t_{SU}$ | 1.341 | 1.341 | 3.088 | ns |
| | | $t_H$ | −1.236 | −1.236 | −2.811 | ns |
| | GCLK PLL | $t_{SU}$ | 2.769 | 2.769 | 6.171 | ns |
| | | $t_H$ | −2.664 | −2.664 | −5.894 | ns |

Table 4–55 lists I/O timing specifications.

**Table 4–55.** EP1AGX35 Column Pins Input Timing Parameters  (Part 1 of 3)

| I/O Standard | Clock | Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|
| | | | Industrial | Commercial | | |
| 3.3-V LVTTL | GCLK | $t_{SU}$ | 1.251 | 1.251 | 2.915 | ns |
| | | $t_H$ | −1.146 | −1.146 | −2.638 | ns |
| | GCLK PLL | $t_{SU}$ | 2.693 | 2.693 | 6.021 | ns |
| | | $t_H$ | −2.588 | −2.588 | −5.744 | ns |
| 3.3-V LVCMOS | GCLK | $t_{SU}$ | 1.251 | 1.251 | 2.915 | ns |
| | | $t_H$ | −1.146 | −1.146 | −2.638 | ns |
| | GCLK PLL | $t_{SU}$ | 2.693 | 2.693 | 6.021 | ns |
| | | $t_H$ | −2.588 | −2.588 | −5.744 | ns |

**Table 4–55.** EP1AGX35 Column Pins Input Timing Parameters  (Part 2 of 3)

| I/O Standard | Clock | Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|
| | | | Industrial | Commercial | | |
| 2.5 V | GCLK | $t_{SU}$ | 1.261 | 1.261 | 2.897 | ns |
| | | $t_H$ | −1.156 | −1.156 | −2.620 | ns |
| | GCLK PLL | $t_{SU}$ | 2.703 | 2.703 | 6.003 | ns |
| | | $t_H$ | −2.598 | −2.598 | −5.726 | ns |
| 1.8 V | GCLK | $t_{SU}$ | 1.327 | 1.327 | 3.107 | ns |
| | | $t_H$ | −1.222 | −1.222 | −2.830 | ns |
| | GCLK PLL | $t_{SU}$ | 2.769 | 2.769 | 6.213 | ns |
| | | $t_H$ | −2.664 | −2.664 | −5.936 | ns |
| 1.5 V | GCLK | $t_{SU}$ | 1.330 | 1.330 | 3.200 | ns |
| | | $t_H$ | −1.225 | −1.225 | −2.923 | ns |
| | GCLK PLL | $t_{SU}$ | 2.772 | 2.772 | 6.306 | ns |
| | | $t_H$ | −2.667 | −2.667 | −6.029 | ns |
| SSTL-2 CLASS I | GCLK | $t_{SU}$ | 1.075 | 1.075 | 2.372 | ns |
| | | $t_H$ | −0.970 | −0.970 | −2.095 | ns |
| | GCLK PLL | $t_{SU}$ | 2.517 | 2.517 | 5.480 | ns |
| | | $t_H$ | −2.412 | −2.412 | −5.203 | ns |
| SSTL-2 CLASS II | GCLK | $t_{SU}$ | 1.075 | 1.075 | 2.372 | ns |
| | | $t_H$ | −0.970 | −0.970 | −2.095 | ns |
| | GCLK PLL | $t_{SU}$ | 2.517 | 2.517 | 5.480 | ns |
| | | $t_H$ | −2.412 | −2.412 | −5.203 | ns |
| SSTL-18 CLASS I | GCLK | $t_{SU}$ | 1.113 | 1.113 | 2.479 | ns |
| | | $t_H$ | −1.008 | −1.008 | −2.202 | ns |
| | GCLK PLL | $t_{SU}$ | 2.555 | 2.555 | 5.585 | ns |
| | | $t_H$ | −2.450 | −2.450 | −5.308 | ns |
| SSTL-18 CLASS II | GCLK | $t_{SU}$ | 1.114 | 1.114 | 2.479 | ns |
| | | $t_H$ | −1.009 | −1.009 | −2.202 | ns |
| | GCLK PLL | $t_{SU}$ | 2.556 | 2.556 | 5.587 | ns |
| | | $t_H$ | −2.451 | −2.451 | −5.310 | ns |
| 1.8-V HSTL CLASS I | GCLK | $t_{SU}$ | 1.113 | 1.113 | 2.479 | ns |
| | | $t_H$ | −1.008 | −1.008 | −2.202 | ns |
| | GCLK PLL | $t_{SU}$ | 2.555 | 2.555 | 5.585 | ns |
| | | $t_H$ | −2.450 | −2.450 | −5.308 | ns |
| 1.8-V HSTL CLASS II | GCLK | $t_{SU}$ | 1.114 | 1.114 | 2.479 | ns |
| | | $t_H$ | −1.009 | −1.009 | −2.202 | ns |
| | GCLK PLL | $t_{SU}$ | 2.556 | 2.556 | 5.587 | ns |
| | | $t_H$ | −2.451 | −2.451 | −5.310 | ns |

**Table 4–55.** EP1AGX35 Column Pins Input Timing Parameters  (Part 3 of 3)

| I/O Standard | Clock | Parameter | Fast Corner | | –6 Speed Grade | Units |
|---|---|---|---|---|---|---|
| | | | Industrial | Commercial | | |
| 1.5-V HSTL CLASS I | GCLK | $t_{SU}$ | 1.131 | 1.131 | 2.607 | ns |
| | | $t_H$ | −1.026 | −1.026 | −2.330 | ns |
| | GCLK PLL | $t_{SU}$ | 2.573 | 2.573 | 5.713 | ns |
| | | $t_H$ | −2.468 | −2.468 | −5.436 | ns |
| 1.5-V HSTL CLASS II | GCLK | $t_{SU}$ | 1.132 | 1.132 | 2.607 | ns |
| | | $t_H$ | −1.027 | −1.027 | −2.330 | ns |
| | GCLK PLL | $t_{SU}$ | 2.574 | 2.574 | 5.715 | ns |
| | | $t_H$ | −2.469 | −2.469 | −5.438 | ns |
| 3.3-V PCI | GCLK | $t_{SU}$ | 1.256 | 1.256 | 2.903 | ns |
| | | $t_H$ | −1.151 | −1.151 | −2.626 | ns |
| | GCLK PLL | $t_{SU}$ | 2.698 | 2.698 | 6.009 | ns |
| | | $t_H$ | −2.593 | −2.593 | −5.732 | ns |
| 3.3-V PCI-X | GCLK | $t_{SU}$ | 1.256 | 1.256 | 2.903 | ns |
| | | $t_H$ | −1.151 | −1.151 | −2.626 | ns |
| | GCLK PLL | $t_{SU}$ | 2.698 | 2.698 | 6.009 | ns |
| | | $t_H$ | −2.593 | −2.593 | −5.732 | ns |
| LVDS | GCLK | $t_{SU}$ | 1.106 | 1.106 | 2.489 | ns |
| | | $t_H$ | −1.001 | −1.001 | −2.212 | ns |
| | GCLK PLL | $t_{SU}$ | 2.530 | 2.530 | 5.564 | ns |
| | | $t_H$ | −2.425 | −2.425 | −5.287 | ns |

Table 4–56 lists I/O timing specifications.

**Table 4–56.** EP1AGX35 Row Pins Output Timing Parameters  (Part 1 of 3)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Model | | –6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| 3.3-V LVTTL | 4 mA | GCLK | $t_{CO}$ | 2.904 | 2.904 | 6.699 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.485 | 1.485 | 3.627 | ns |
| 3.3-V LVTTL | 8 mA | GCLK | $t_{CO}$ | 2.776 | 2.776 | 6.059 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.357 | 1.357 | 2.987 | ns |
| 3.3-V LVTTL | 12 mA | GCLK | $t_{CO}$ | 2.720 | 2.720 | 6.022 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.301 | 1.301 | 2.950 | ns |
| 3.3-V LVCMOS | 4 mA | GCLK | $t_{CO}$ | 2.776 | 2.776 | 6.059 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.357 | 1.357 | 2.987 | ns |
| 3.3-V LVCMOS | 8 mA | GCLK | $t_{CO}$ | 2.670 | 2.670 | 5.753 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.251 | 1.251 | 2.681 | ns |
| 2.5 V | 4 mA | GCLK | $t_{CO}$ | 2.759 | 2.759 | 6.033 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.340 | 1.340 | 2.961 | ns |

**Table 4–56.** EP1AGX35 Row Pins Output Timing Parameters (Part 2 of 3)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Model | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| 2.5 V | 8 mA | GCLK | $t_{CO}$ | 2.656 | 2.656 | 5.775 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.237 | 1.237 | 2.703 | ns |
| 2.5 V | 12 mA | GCLK | $t_{CO}$ | 2.637 | 2.637 | 5.661 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.218 | 1.218 | 2.589 | ns |
| 1.8 V | 2 mA | GCLK | $t_{CO}$ | 2.829 | 2.829 | 7.052 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.410 | 1.410 | 3.980 | ns |
| 1.8 V | 4 mA | GCLK | $t_{CO}$ | 2.818 | 2.818 | 6.273 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.399 | 1.399 | 3.201 | ns |
| 1.8 V | 6 mA | GCLK | $t_{CO}$ | 2.707 | 2.707 | 5.972 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.288 | 1.288 | 2.900 | ns |
| 1.8 V | 8 mA | GCLK | $t_{CO}$ | 2.676 | 2.676 | 5.858 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.257 | 1.257 | 2.786 | ns |
| 1.5 V | 2 mA | GCLK | $t_{CO}$ | 2.789 | 2.789 | 6.551 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.370 | 1.370 | 3.479 | ns |
| 1.5 V | 4 mA | GCLK | $t_{CO}$ | 2.682 | 2.682 | 5.950 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.263 | 1.263 | 2.878 | ns |
| SSTL-2 CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.626 | 2.626 | 5.614 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.207 | 1.207 | 2.542 | ns |
| SSTL-2 CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.602 | 2.602 | 5.538 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.183 | 1.183 | 2.466 | ns |
| SSTL-2 CLASS II | 16 mA | GCLK | $t_{CO}$ | 2.568 | 2.568 | 5.407 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.149 | 1.149 | 2.335 | ns |
| SSTL-18 CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.614 | 2.614 | 5.556 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.195 | 1.195 | 2.484 | ns |
| SSTL-18 CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.618 | 2.618 | 5.485 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.199 | 1.199 | 2.413 | ns |
| SSTL-18 CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.594 | 2.594 | 5.468 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.175 | 1.175 | 2.396 | ns |
| SSTL-18 CLASS I | 10 mA | GCLK | $t_{CO}$ | 2.597 | 2.597 | 5.447 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.178 | 1.178 | 2.375 | ns |
| 1.8-V HSTL CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.595 | 2.595 | 5.466 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.176 | 1.176 | 2.394 | ns |
| 1.8-V HSTL CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.598 | 2.598 | 5.430 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.179 | 1.179 | 2.358 | ns |
| 1.8-V HSTL CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.580 | 2.580 | 5.426 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.161 | 1.161 | 2.354 | ns |
| 1.8-V HSTL CLASS I | 10 mA | GCLK | $t_{CO}$ | 2.584 | 2.584 | 5.415 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.165 | 1.165 | 2.343 | ns |

**Table 4–56.** EP1AGX35 Row Pins Output Timing Parameters  (Part 3 of 3)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Model | | –6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| 1.8-V HSTL CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.575 | 2.575 | 5.414 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.156 | 1.156 | 2.342 | ns |
| 1.5-V HSTL CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.594 | 2.594 | 5.443 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.175 | 1.175 | 2.371 | ns |
| 1.5-V HSTL CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.597 | 2.597 | 5.429 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.178 | 1.178 | 2.357 | ns |
| 1.5-V HSTL CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.582 | 2.582 | 5.421 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.163 | 1.163 | 2.349 | ns |
| LVDS | — | GCLK | $t_{CO}$ | 2.654 | 2.654 | 5.613 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.226 | 1.226 | 2.530 | ns |

Table 4–57 lists I/O timing specifications.

**Table 4–57.** EP1AGX35 Column Pins Output Timing Parameters   (Part 1 of 4)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Corner | | –6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| 3.3-V LVTTL | 4 mA | GCLK | $t_{CO}$ | 2.909 | 2.909 | 6.541 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.467 | 1.467 | 3.435 | ns |
| 3.3-V LVTTL | 8 mA | GCLK | $t_{CO}$ | 2.764 | 2.764 | 6.169 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.322 | 1.322 | 3.063 | ns |
| 3.3-V LVTTL | 12 mA | GCLK | $t_{CO}$ | 2.697 | 2.697 | 6.169 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.255 | 1.255 | 3.063 | ns |
| 3.3-V LVTTL | 16 mA | GCLK | $t_{CO}$ | 2.671 | 2.671 | 6.000 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.229 | 1.229 | 2.894 | ns |
| 3.3-V LVTTL | 20 mA | GCLK | $t_{CO}$ | 2.649 | 2.649 | 5.875 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.207 | 1.207 | 2.769 | ns |
| 3.3-V LVTTL | 24 mA | GCLK | $t_{CO}$ | 2.642 | 2.642 | 5.877 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.200 | 1.200 | 2.771 | ns |
| 3.3-V LVCMOS | 4 mA | GCLK | $t_{CO}$ | 2.764 | 2.764 | 6.169 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.322 | 1.322 | 3.063 | ns |
| 3.3-V LVCMOS | 8 mA | GCLK | $t_{CO}$ | 2.672 | 2.672 | 5.874 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.230 | 1.230 | 2.768 | ns |
| 3.3-V LVCMOS | 12 mA | GCLK | $t_{CO}$ | 2.644 | 2.644 | 5.796 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.202 | 1.202 | 2.690 | ns |
| 3.3-V LVCMOS | 16 mA | GCLK | $t_{CO}$ | 2.651 | 2.651 | 5.764 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.209 | 1.209 | 2.658 | ns |
| 3.3-V LVCMOS | 20 mA | GCLK | $t_{CO}$ | 2.638 | 2.638 | 5.746 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.196 | 1.196 | 2.640 | ns |

**Table 4–57.** EP1AGX35 Column Pins Output Timing Parameters   (Part 2 of 4)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| 3.3-V LVCMOS | 24 mA | GCLK | $t_{CO}$ | 2.627 | 2.627 | 5.724 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.185 | 1.185 | 2.618 | ns |
| 2.5 V | 4 mA | GCLK | $t_{CO}$ | 2.726 | 2.726 | 6.201 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.284 | 1.284 | 3.095 | ns |
| 2.5 V | 8 mA | GCLK | $t_{CO}$ | 2.674 | 2.674 | 5.939 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.232 | 1.232 | 2.833 | ns |
| 2.5 V | 12 mA | GCLK | $t_{CO}$ | 2.653 | 2.653 | 5.822 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.211 | 1.211 | 2.716 | ns |
| 2.5 V | 16 mA | GCLK | $t_{CO}$ | 2.635 | 2.635 | 5.748 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.193 | 1.193 | 2.642 | ns |
| 1.8 V | 2 mA | GCLK | $t_{CO}$ | 2.766 | 2.766 | 7.193 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.324 | 1.324 | 4.087 | ns |
| 1.8 V | 4 mA | GCLK | $t_{CO}$ | 2.771 | 2.771 | 6.419 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.329 | 1.329 | 3.313 | ns |
| 1.8 V | 6 mA | GCLK | $t_{CO}$ | 2.695 | 2.695 | 6.155 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.253 | 1.253 | 3.049 | ns |
| 1.8 V | 8 mA | GCLK | $t_{CO}$ | 2.697 | 2.697 | 6.064 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.255 | 1.255 | 2.958 | ns |
| 1.8 V | 10 mA | GCLK | $t_{CO}$ | 2.651 | 2.651 | 5.987 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.209 | 1.209 | 2.881 | ns |
| 1.8 V | 12 mA | GCLK | $t_{CO}$ | 2.652 | 2.652 | 5.930 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.210 | 1.210 | 2.824 | ns |
| 1.5 V | 2 mA | GCLK | $t_{CO}$ | 2.746 | 2.746 | 6.723 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.304 | 1.304 | 3.617 | ns |
| 1.5 V | 4 mA | GCLK | $t_{CO}$ | 2.682 | 2.682 | 6.154 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.240 | 1.240 | 3.048 | ns |
| 1.5 V | 6 mA | GCLK | $t_{CO}$ | 2.685 | 2.685 | 6.036 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.243 | 1.243 | 2.930 | ns |
| 1.5 V | 8 mA | GCLK | $t_{CO}$ | 2.644 | 2.644 | 5.983 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.202 | 1.202 | 2.877 | ns |
| SSTL-2 CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.629 | 2.629 | 5.762 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.184 | 1.184 | 2.650 | ns |
| SSTL-2 CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.612 | 2.612 | 5.712 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.167 | 1.167 | 2.600 | ns |
| SSTL-2 CLASS II | 16 mA | GCLK | $t_{CO}$ | 2.590 | 2.590 | 5.639 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.145 | 1.145 | 2.527 | ns |
| SSTL-2 CLASS II | 20 mA | GCLK | $t_{CO}$ | 2.591 | 2.591 | 5.626 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.146 | 1.146 | 2.514 | ns |

**Table 4–57.** EP1AGX35 Column Pins Output Timing Parameters  (Part 3 of 4)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| SSTL-2 CLASS II | 24 mA | GCLK | $t_{CO}$ | 2.587 | 2.587 | 5.624 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.142 | 1.142 | 2.512 | ns |
| SSTL-18 CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.626 | 2.626 | 5.733 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.184 | 1.184 | 2.627 | ns |
| SSTL-18 CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.630 | 2.630 | 5.694 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.185 | 1.185 | 2.582 | ns |
| SSTL-18 CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.609 | 2.609 | 5.675 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.164 | 1.164 | 2.563 | ns |
| SSTL-18 CLASS I | 10 mA | GCLK | $t_{CO}$ | 2.614 | 2.614 | 5.673 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.169 | 1.169 | 2.561 | ns |
| SSTL-18 CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.608 | 2.608 | 5.659 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.163 | 1.163 | 2.547 | ns |
| SSTL-18 CLASS II | 8 mA | GCLK | $t_{CO}$ | 2.597 | 2.597 | 5.625 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.152 | 1.152 | 2.513 | ns |
| SSTL-18 CLASS II | 16 mA | GCLK | $t_{CO}$ | 2.609 | 2.609 | 5.603 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.164 | 1.164 | 2.491 | ns |
| SSTL-18 CLASS II | 18 mA | GCLK | $t_{CO}$ | 2.605 | 2.605 | 5.611 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.160 | 1.160 | 2.499 | ns |
| SSTL-18 CLASS II | 20 mA | GCLK | $t_{CO}$ | 2.605 | 2.605 | 5.609 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.160 | 1.160 | 2.497 | ns |
| 1.8-V HSTL CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.629 | 2.629 | 5.664 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.187 | 1.187 | 2.558 | ns |
| 1.8-V HSTL CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.634 | 2.634 | 5.649 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.189 | 1.189 | 2.537 | ns |
| 1.8-V HSTL CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.612 | 2.612 | 5.638 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.167 | 1.167 | 2.526 | ns |
| 1.8-V HSTL CLASS I | 10 mA | GCLK | $t_{CO}$ | 2.616 | 2.616 | 5.644 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.171 | 1.171 | 2.532 | ns |
| 1.8-V HSTL CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.608 | 2.608 | 5.637 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.163 | 1.163 | 2.525 | ns |
| 1.8-V HSTL CLASS II | 16 mA | GCLK | $t_{CO}$ | 2.591 | 2.591 | 5.401 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.146 | 1.146 | 2.289 | ns |
| 1.8-V HSTL CLASS II | 18 mA | GCLK | $t_{CO}$ | 2.593 | 2.593 | 5.412 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.148 | 1.148 | 2.300 | ns |
| 1.8-V HSTL CLASS II | 20 mA | GCLK | $t_{CO}$ | 2.593 | 2.593 | 5.421 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.148 | 1.148 | 2.309 | ns |
| 1.5-V HSTL CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.629 | 2.629 | 5.663 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.187 | 1.187 | 2.557 | ns |

**Table 4–57.** EP1AGX35 Column Pins Output Timing Parameters   (Part 4 of 4)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| 1.5-V HSTL CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.633 | 2.633 | 5.641 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.188 | 1.188 | 2.529 | ns |
| 1.5-V HSTL CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.615 | 2.615 | 5.643 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.170 | 1.170 | 2.531 | ns |
| 1.5-V HSTL CLASS I | 10 mA | GCLK | $t_{CO}$ | 2.615 | 2.615 | 5.645 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.170 | 1.170 | 2.533 | ns |
| 1.5-V HSTL CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.609 | 2.609 | 5.643 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.164 | 1.164 | 2.531 | ns |
| 1.5-V HSTL CLASS II | 16 mA | GCLK | $t_{CO}$ | 2.596 | 2.596 | 5.455 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.151 | 1.151 | 2.343 | ns |
| 1.5-V HSTL CLASS II | 18 mA | GCLK | $t_{CO}$ | 2.599 | 2.599 | 5.465 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.154 | 1.154 | 2.353 | ns |
| 1.5-V HSTL CLASS II | 20 mA | GCLK | $t_{CO}$ | 2.601 | 2.601 | 5.478 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.156 | 1.156 | 2.366 | ns |
| 3.3-V PCI | — | GCLK | $t_{CO}$ | 2.755 | 2.755 | 5.791 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.313 | 1.313 | 2.685 | ns |
| 3.3-V PCI-X | — | GCLK | $t_{CO}$ | 2.755 | 2.755 | 5.791 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.313 | 1.313 | 2.685 | ns |
| LVDS | — | GCLK | $t_{CO}$ | 3.621 | 3.621 | 6.969 | ns |
| | | GCLK PLL | $t_{CO}$ | 2.190 | 2.190 | 3.880 | ns |

Table 4–58 through Table 4–59 list EP1AGX35 regional clock (RCLK) adder values that should be added to GCLK values. These adder values are used to determine I/O timing when the I/O pin is driven using the regional clock. This applies for all I/O standards supported by Arria GX with general purpose I/O pins.

Table 4–58 describes row pin delay adders when using the regional clock in Arria GX devices.

**Table 4–58.** EP1AGX35 Row Pin Delay Adders for Regional Clock

| Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|
| | Industrial | Commercial | | |
| RCLK input adder | 0.126 | 0.126 | 0.281 | ns |
| RCLK PLL input adder | 0.011 | 0.011 | 0.018 | ns |
| RCLK output adder | −0.126 | −0.126 | −0.281 | ns |
| RCLK PLL output adder | −0.011 | −0.011 | −0.018 | ns |

Table 4–59 lists column pin delay adders when using the regional clock in Arria GX devices.

**Table 4–59.** EP1AGX35 Column Pin Delay Adders for Regional Clock

| Parameter | Fast Corner | | –6 Speed Grade | Units |
|---|---|---|---|---|
| | Industrial | Commercial | | |
| RCLK input adder | 0.099 | 0.099 | 0.254 | ns |
| RCLK PLL input adder | −0.012 | −0.012 | −0.01 | ns |
| RCLK output adder | −0.086 | −0.086 | −0.244 | ns |
| RCLK PLL output adder | 1.253 | 1.253 | 3.133 | ns |

## EP1AGX50 I/O Timing Parameters

Table 4–60 through Table 4–63 list the maximum I/O timing parameters for EP1AGX50 devices for I/O standards which support general purpose I/O pins.

Table 4–60 lists I/O timing specifications.

**Table 4–60.** EP1AGX50 Row Pins Input Timing Parameters (Part 1 of 2)

| I/O Standard | Clock | Parameter | Fast Model | | –6 Speed Grade | Units |
|---|---|---|---|---|---|---|
| | | | Industrial | Commercial | | |
| 3.3-V LVTTL | GCLK | $t_{SU}$ | 1.550 | 1.550 | 3.542 | ns |
| | | $t_H$ | −1.445 | −1.445 | −3.265 | ns |
| | GCLK PLL | $t_{SU}$ | 2.978 | 2.978 | 6.626 | ns |
| | | $t_H$ | −2.873 | −2.873 | −6.349 | ns |
| 3.3-V LVCMOS | GCLK | $t_{SU}$ | 1.550 | 1.550 | 3.542 | ns |
| | | $t_H$ | −1.445 | −1.445 | −3.265 | ns |
| | GCLK PLL | $t_{SU}$ | 2.978 | 2.978 | 6.626 | ns |
| | | $t_H$ | −2.873 | −2.873 | −6.349 | ns |
| 2.5 V | GCLK | $t_{SU}$ | 1.562 | 1.562 | 3.523 | ns |
| | | $t_H$ | −1.457 | −1.457 | −3.246 | ns |
| | GCLK PLL | $t_{SU}$ | 2.990 | 2.990 | 6.607 | ns |
| | | $t_H$ | −2.885 | −2.885 | −6.330 | ns |
| 1.8 V | GCLK | $t_{SU}$ | 1.628 | 1.628 | 3.730 | ns |
| | | $t_H$ | −1.523 | −1.523 | −3.453 | ns |
| | GCLK PLL | $t_{SU}$ | 3.056 | 3.056 | 6.814 | ns |
| | | $t_H$ | −2.951 | −2.951 | −6.537 | ns |
| 1.5 V | GCLK | $t_{SU}$ | 1.631 | 1.631 | 3.825 | ns |
| | | $t_H$ | −1.526 | −1.526 | −3.548 | ns |
| | GCLK PLL | $t_{SU}$ | 3.059 | 3.059 | 6.909 | ns |
| | | $t_H$ | −2.954 | −2.954 | −6.632 | ns |

**Table 4–60.** EP1AGX50 Row Pins Input Timing Parameters  (Part 2 of 2)

| I/O Standard | Clock | Parameter | Fast Model | | −6 Speed Grade | Units |
| --- | --- | --- | --- | --- | --- | --- |
| | | | Industrial | Commercial | | |
| SSTL-2 CLASS I | GCLK | $t_{SU}$ | 1.375 | 1.375 | 2.997 | ns |
| | | $t_H$ | −1.270 | −1.270 | −2.720 | ns |
| | GCLK PLL | $t_{SU}$ | 2.802 | 2.802 | 6.079 | ns |
| | | $t_H$ | −2.697 | −2.697 | −5.802 | ns |
| SSTL-2 CLASS II | GCLK | $t_{SU}$ | 1.375 | 1.375 | 2.997 | ns |
| | | $t_H$ | −1.270 | −1.270 | −2.720 | ns |
| | GCLK PLL | $t_{SU}$ | 2.802 | 2.802 | 6.079 | ns |
| | | $t_H$ | −2.697 | −2.697 | −5.802 | ns |
| SSTL-18 CLASS I | GCLK | $t_{SU}$ | 1.406 | 1.406 | 3.104 | ns |
| | | $t_H$ | −1.301 | −1.301 | −2.827 | ns |
| | GCLK PLL | $t_{SU}$ | 2.834 | 2.834 | 6.188 | ns |
| | | $t_H$ | −2.729 | −2.729 | −5.911 | ns |
| SSTL-18 CLASS II | GCLK | $t_{SU}$ | 1.407 | 1.407 | 3.106 | ns |
| | | $t_H$ | −1.302 | −1.302 | −2.829 | ns |
| | GCLK PLL | $t_{SU}$ | 2.834 | 2.834 | 6.188 | ns |
| | | $t_H$ | −2.729 | −2.729 | −5.911 | ns |
| 1.8-V HSTL CLASS I | GCLK | $t_{SU}$ | 1.406 | 1.406 | 3.104 | ns |
| | | $t_H$ | −1.301 | −1.301 | −2.827 | ns |
| | GCLK PLL | $t_{SU}$ | 2.834 | 2.834 | 6.188 | ns |
| | | $t_H$ | −2.729 | −2.729 | −5.911 | ns |
| 1.8-V HSTL CLASS II | GCLK | $t_{SU}$ | 1.407 | 1.407 | 3.106 | ns |
| | | $t_H$ | −1.302 | −1.302 | −2.829 | ns |
| | GCLK PLL | $t_{SU}$ | 2.834 | 2.834 | 6.188 | ns |
| | | $t_H$ | −2.729 | −2.729 | −5.911 | ns |
| 1.5-V HSTL CLASS I | GCLK | $t_{SU}$ | 1.432 | 1.432 | 3.232 | ns |
| | | $t_H$ | −1.327 | −1.327 | −2.955 | ns |
| | GCLK PLL | $t_{SU}$ | 2.860 | 2.860 | 6.316 | ns |
| | | $t_H$ | −2.755 | −2.755 | −6.039 | ns |
| 1.5-V HSTL CLASS II | GCLK | $t_{SU}$ | 1.433 | 1.433 | 3.234 | ns |
| | | $t_H$ | −1.328 | −1.328 | −2.957 | ns |
| | GCLK PLL | $t_{SU}$ | 2.860 | 2.860 | 6.316 | ns |
| | | $t_H$ | −2.755 | −2.755 | −6.039 | ns |
| LVDS | GCLK | $t_{SU}$ | 1.341 | 1.341 | 3.088 | ns |
| | | $t_H$ | −1.236 | −1.236 | −2.811 | ns |
| | GCLK PLL | $t_{SU}$ | 2.769 | 2.769 | 6.171 | ns |
| | | $t_H$ | −2.664 | −2.664 | −5.894 | ns |

Table 4–61 lists I/O timing specifications.

**Table 4–61.** EP1AGX50 Column Pins Input Timing Parameters  (Part 1 of 2)

| I/O Standard | Clock | Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|
| | | | Industrial | Commercial | | |
| 3.3-V LVTTL | GCLK | $t_{SU}$ | 1.242 | 1.242 | 2.902 | ns |
| | | $t_H$ | −1.137 | −1.137 | −2.625 | ns |
| | GCLK PLL | $t_{SU}$ | 2.684 | 2.684 | 6.009 | ns |
| | | $t_H$ | −2.579 | −2.579 | −5.732 | ns |
| 3.3-V LVCMOS | GCLK | $t_{SU}$ | 1.242 | 1.242 | 2.902 | ns |
| | | $t_H$ | −1.137 | −1.137 | −2.625 | ns |
| | GCLK PLL | $t_{SU}$ | 2.684 | 2.684 | 6.009 | ns |
| | | $t_H$ | −2.579 | −2.579 | −5.732 | ns |
| 2.5 V | GCLK | $t_{SU}$ | 1.252 | 1.252 | 2.884 | ns |
| | | $t_H$ | −1.147 | −1.147 | −2.607 | ns |
| | GCLK PLL | $t_{SU}$ | 2.694 | 2.694 | 5.991 | ns |
| | | $t_H$ | −2.589 | −2.589 | −5.714 | ns |
| 1.8 V | GCLK | $t_{SU}$ | 1.318 | 1.318 | 3.094 | ns |
| | | $t_H$ | −1.213 | −1.213 | −2.817 | ns |
| | GCLK PLL | $t_{SU}$ | 2.760 | 2.760 | 6.201 | ns |
| | | $t_H$ | −2.655 | −2.655 | −5.924 | ns |
| 1.5 V | GCLK | $t_{SU}$ | 1.321 | 1.321 | 3.187 | ns |
| | | $t_H$ | −1.216 | −1.216 | −2.910 | ns |
| | GCLK PLL | $t_{SU}$ | 2.763 | 2.763 | 6.294 | ns |
| | | $t_H$ | −2.658 | −2.658 | −6.017 | ns |
| SSTL-2 CLASS I | GCLK | $t_{SU}$ | 1.034 | 1.034 | 2.314 | ns |
| | | $t_H$ | −0.929 | −0.929 | −2.037 | ns |
| | GCLK PLL | $t_{SU}$ | 2.500 | 2.500 | 5.457 | ns |
| | | $t_H$ | −2.395 | −2.395 | −5.180 | ns |
| SSTL-2 CLASS II | GCLK | $t_{SU}$ | 1.034 | 1.034 | 2.314 | ns |
| | | $t_H$ | −0.929 | −0.929 | −2.037 | ns |
| | GCLK PLL | $t_{SU}$ | 2.500 | 2.500 | 5.457 | ns |
| | | $t_H$ | −2.395 | −2.395 | −5.180 | ns |
| SSTL-18 CLASS I | GCLK | $t_{SU}$ | 1.104 | 1.104 | 2.466 | ns |
| | | $t_H$ | −0.999 | −0.999 | −2.189 | ns |
| | GCLK PLL | $t_{SU}$ | 2.546 | 2.546 | 5.573 | ns |
| | | $t_H$ | −2.441 | −2.441 | −5.296 | ns |
| SSTL-18 CLASS II | GCLK | $t_{SU}$ | 1.074 | 1.074 | 2.424 | ns |
| | | $t_H$ | −0.969 | −0.969 | −2.147 | ns |
| | GCLK PLL | $t_{SU}$ | 2.539 | 2.539 | 5.564 | ns |
| | | $t_H$ | −2.434 | −2.434 | −5.287 | ns |

**Table 4–61.** EP1AGX50 Column Pins Input Timing Parameters   (Part 2 of 2)

| I/O Standard | Clock | Parameter | Fast Corner | | −6 Speed Grade | Units |
| | | | Industrial | Commercial | | |
|---|---|---|---|---|---|---|
| 1.8-V HSTL CLASS I | GCLK | $t_{SU}$ | 1.104 | 1.104 | 2.466 | ns |
| | | $t_H$ | −0.999 | −0.999 | −2.189 | ns |
| | GCLK PLL | $t_{SU}$ | 2.546 | 2.546 | 5.573 | ns |
| | | $t_H$ | −2.441 | −2.441 | −5.296 | ns |
| 1.8-V HSTL CLASS II | GCLK | $t_{SU}$ | 1.074 | 1.074 | 2.424 | ns |
| | | $t_H$ | −0.969 | −0.969 | −2.147 | ns |
| | GCLK PLL | $t_{SU}$ | 2.539 | 2.539 | 5.564 | ns |
| | | $t_H$ | −2.434 | −2.434 | −5.287 | ns |
| 1.5-V HSTL CLASS I | GCLK | $t_{SU}$ | 1.122 | 1.122 | 2.594 | ns |
| | | $t_H$ | −1.017 | −1.017 | −2.317 | ns |
| | GCLK PLL | $t_{SU}$ | 2.564 | 2.564 | 5.701 | ns |
| | | $t_H$ | −2.459 | −2.459 | −5.424 | ns |
| 1.5-V HSTL CLASS II | GCLK | $t_{SU}$ | 1.094 | 1.094 | 2.557 | ns |
| | | $t_H$ | −0.989 | −0.989 | −2.280 | ns |
| | GCLK PLL | $t_{SU}$ | 2.557 | 2.557 | 5.692 | ns |
| | | $t_H$ | −2.452 | −2.452 | −5.415 | ns |
| 3.3-V PCI | GCLK | $t_{SU}$ | 1.247 | 1.247 | 2.890 | ns |
| | | $t_H$ | −1.142 | −1.142 | −2.613 | ns |
| | GCLK PLL | $t_{SU}$ | 2.689 | 2.689 | 5.997 | ns |
| | | $t_H$ | −2.584 | −2.584 | −5.720 | ns |
| 3.3-V PCI-X | GCLK | $t_{SU}$ | 1.247 | 1.247 | 2.890 | ns |
| | | $t_H$ | −1.142 | −1.142 | −2.613 | ns |
| | GCLK PLL | $t_{SU}$ | 2.689 | 2.689 | 5.997 | ns |
| | | $t_H$ | −2.584 | −2.584 | −5.720 | ns |
| LVDS | GCLK | $t_{SU}$ | 1.106 | 1.106 | 2.489 | ns |
| | | $t_H$ | −1.001 | −1.001 | −2.212 | ns |
| | GCLK PLL | $t_{SU}$ | 2.530 | 2.530 | 5.564 | ns |
| | | $t_H$ | −2.425 | −2.425 | −5.287 | ns |

Table 4–62 lists I/O timing specifications.

**Table 4–62.** EP1AGX50 Row Pins Output Timing Parameters   (Part 1 of 3)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Model | | −6 Speed Grade | Units |
| | | | | Industrial | Commercial | | |
|---|---|---|---|---|---|---|---|
| 3.3-V LVTTL | 4 mA | GCLK | $t_{CO}$ | 2.915 | 2.915 | 6.713 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.487 | 1.487 | 3.629 | ns |
| 3.3-V LVTTL | 8 mA | GCLK | $t_{CO}$ | 2.787 | 2.787 | 6.073 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.359 | 1.359 | 2.989 | ns |

**Table 4–62.** EP1AGX50 Row Pins Output Timing Parameters   (Part 2 of 3)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Model | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| 3.3-V LVTTL | 12 mA | GCLK | $t_{CO}$ | 2.731 | 2.731 | 6.036 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.303 | 1.303 | 2.952 | ns |
| 3.3-V LVCMOS | 4 mA | GCLK | $t_{CO}$ | 2.787 | 2.787 | 6.073 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.359 | 1.359 | 2.989 | ns |
| 3.3-V LVCMOS | 8 mA | GCLK | $t_{CO}$ | 2.681 | 2.681 | 5.767 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.253 | 1.253 | 2.683 | ns |
| 2.5 V | 4 mA | GCLK | $t_{CO}$ | 2.770 | 2.770 | 6.047 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.342 | 1.342 | 2.963 | ns |
| 2.5 V | 8 mA | GCLK | $t_{CO}$ | 2.667 | 2.667 | 5.789 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.239 | 1.239 | 2.705 | ns |
| 2.5 V | 12 mA | GCLK | $t_{CO}$ | 2.648 | 2.648 | 5.675 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.220 | 1.220 | 2.591 | ns |
| 1.8 V | 2 mA | GCLK | $t_{CO}$ | 2.840 | 2.840 | 7.066 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.412 | 1.412 | 3.982 | ns |
| 1.8 V | 4 mA | GCLK | $t_{CO}$ | 2.829 | 2.829 | 6.287 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.401 | 1.401 | 3.203 | ns |
| 1.8 V | 6 mA | GCLK | $t_{CO}$ | 2.718 | 2.718 | 5.986 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.290 | 1.290 | 2.902 | ns |
| 1.8 V | 8 mA | GCLK | $t_{CO}$ | 2.687 | 2.687 | 5.872 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.259 | 1.259 | 2.788 | ns |
| 1.5 V | 2 mA | GCLK | $t_{CO}$ | 2.800 | 2.800 | 6.565 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.372 | 1.372 | 3.481 | ns |
| 1.5 V | 4 mA | GCLK | $t_{CO}$ | 2.693 | 2.693 | 5.964 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.265 | 1.265 | 2.880 | ns |
| SSTL-2 CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.636 | 2.636 | 5.626 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.209 | 1.209 | 2.544 | ns |
| SSTL-2 CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.612 | 2.612 | 5.550 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.185 | 1.185 | 2.468 | ns |
| SSTL-2 CLASS II | 16 mA | GCLK | $t_{CO}$ | 2.578 | 2.578 | 5.419 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.151 | 1.151 | 2.337 | ns |
| SSTL-18 CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.625 | 2.625 | 5.570 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.197 | 1.197 | 2.486 | ns |
| SSTL-18 CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.628 | 2.628 | 5.497 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.201 | 1.201 | 2.415 | ns |
| SSTL-18 CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.604 | 2.604 | 5.480 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.177 | 1.177 | 2.398 | ns |
| SSTL-18 CLASS I | 10 mA | GCLK | $t_{CO}$ | 2.607 | 2.607 | 5.459 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.180 | 1.180 | 2.377 | ns |

**Table 4–62.** EP1AGX50 Row Pins Output Timing Parameters   (Part 3 of 3)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Model | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| 1.8-V HSTL CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.606 | 2.606 | 5.480 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.178 | 1.178 | 2.396 | ns |
| 1.8-V HSTL CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.608 | 2.608 | 5.442 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.181 | 1.181 | 2.360 | ns |
| 1.8-V HSTL CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.590 | 2.590 | 5.438 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.163 | 1.163 | 2.356 | ns |
| 1.8-V HSTL CLASS I | 10 mA | GCLK | $t_{CO}$ | 2.594 | 2.594 | 5.427 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.167 | 1.167 | 2.345 | ns |
| 1.8-V HSTL CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.585 | 2.585 | 5.426 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.158 | 1.158 | 2.344 | ns |
| 1.5-V HSTL CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.605 | 2.605 | 5.457 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.177 | 1.177 | 2.373 | ns |
| 1.5-V HSTL CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.607 | 2.607 | 5.441 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.180 | 1.180 | 2.359 | ns |
| 1.5-V HSTL CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.592 | 2.592 | 5.433 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.165 | 1.165 | 2.351 | ns |
| LVDS | — | GCLK | $t_{CO}$ | 2.654 | 2.654 | 5.613 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.226 | 1.226 | 2.530 | ns |

Table 4–63 lists I/O timing specifications.

**Table 4–63.** EP1AGX50 Column Pins Output Timing Parameters   (Part 1 of 4)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| 3.3-V LVTTL | 4 mA | GCLK | $t_{CO}$ | 2.948 | 2.948 | 6.608 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.476 | 1.476 | 3.447 | ns |
| 3.3-V LVTTL | 8 mA | GCLK | $t_{CO}$ | 2.797 | 2.797 | 6.203 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.331 | 1.331 | 3.075 | ns |
| 3.3-V LVTTL | 12 mA | GCLK | $t_{CO}$ | 2.722 | 2.722 | 6.204 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.264 | 1.264 | 3.075 | ns |
| 3.3-V LVTTL | 16 mA | GCLK | $t_{CO}$ | 2.694 | 2.694 | 6.024 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.238 | 1.238 | 2.906 | ns |
| 3.3-V LVTTL | 20 mA | GCLK | $t_{CO}$ | 2.670 | 2.670 | 5.896 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.216 | 1.216 | 2.781 | ns |
| 3.3-V LVTTL | 24 mA | GCLK | $t_{CO}$ | 2.660 | 2.660 | 5.895 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.209 | 1.209 | 2.783 | ns |
| 3.3-V LVCMOS | 4 mA | GCLK | $t_{CO}$ | 2.797 | 2.797 | 6.203 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.331 | 1.331 | 3.075 | ns |

**Table 4–63.** EP1AGX50 Column Pins Output Timing Parameters   (Part 2 of 4)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| 3.3-V LVCMOS | 8 mA | GCLK | $t_{CO}$ | 2.695 | 2.695 | 5.893 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.239 | 1.239 | 2.780 | ns |
| 3.3-V LVCMOS | 12 mA | GCLK | $t_{CO}$ | 2.663 | 2.663 | 5.809 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.211 | 1.211 | 2.702 | ns |
| 3.3-V LVCMOS | 16 mA | GCLK | $t_{CO}$ | 2.666 | 2.666 | 5.776 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.218 | 1.218 | 2.670 | ns |
| 3.3-V LVCMOS | 20 mA | GCLK | $t_{CO}$ | 2.651 | 2.651 | 5.758 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.205 | 1.205 | 2.652 | ns |
| 3.3-V LVCMOS | 24 mA | GCLK | $t_{CO}$ | 2.638 | 2.638 | 5.736 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.194 | 1.194 | 2.630 | ns |
| 2.5 V | 4 mA | GCLK | $t_{CO}$ | 2.754 | 2.754 | 6.240 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.293 | 1.293 | 3.107 | ns |
| 2.5 V | 8 mA | GCLK | $t_{CO}$ | 2.697 | 2.697 | 5.963 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.241 | 1.241 | 2.845 | ns |
| 2.5 V | 12 mA | GCLK | $t_{CO}$ | 2.672 | 2.672 | 5.837 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.220 | 1.220 | 2.728 | ns |
| 2.5 V | 16 mA | GCLK | $t_{CO}$ | 2.654 | 2.654 | 5.760 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.202 | 1.202 | 2.654 | ns |
| 1.8 V | 2 mA | GCLK | $t_{CO}$ | 2.804 | 2.804 | 7.295 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.333 | 1.333 | 4.099 | ns |
| 1.8 V | 4 mA | GCLK | $t_{CO}$ | 2.808 | 2.808 | 6.479 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.338 | 1.338 | 3.325 | ns |
| 1.8 V | 6 mA | GCLK | $t_{CO}$ | 2.717 | 2.717 | 6.195 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.262 | 1.262 | 3.061 | ns |
| 1.8 V | 8 mA | GCLK | $t_{CO}$ | 2.719 | 2.719 | 6.098 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.264 | 1.264 | 2.970 | ns |
| 1.8 V | 10 mA | GCLK | $t_{CO}$ | 2.671 | 2.671 | 6.012 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.218 | 1.218 | 2.893 | ns |
| 1.8 V | 12 mA | GCLK | $t_{CO}$ | 2.671 | 2.671 | 5.953 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.219 | 1.219 | 2.836 | ns |
| 1.5 V | 2 mA | GCLK | $t_{CO}$ | 2.779 | 2.779 | 6.815 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.313 | 1.313 | 3.629 | ns |
| 1.5 V | 4 mA | GCLK | $t_{CO}$ | 2.703 | 2.703 | 6.210 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.249 | 1.249 | 3.060 | ns |
| 1.5 V | 6 mA | GCLK | $t_{CO}$ | 2.705 | 2.705 | 6.118 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.252 | 1.252 | 2.942 | ns |
| 1.5 V | 8 mA | GCLK | $t_{CO}$ | 2.660 | 2.660 | 6.014 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.211 | 1.211 | 2.889 | ns |

**Table 4–63.** EP1AGX50 Column Pins Output Timing Parameters  (Part 3 of 4)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| SSTL-2 CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.648 | 2.648 | 5.777 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.202 | 1.202 | 2.675 | ns |
| SSTL-2 CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.628 | 2.628 | 5.722 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.185 | 1.185 | 2.625 | ns |
| SSTL-2 CLASS II | 16 mA | GCLK | $t_{CO}$ | 2.606 | 2.606 | 5.649 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.163 | 1.163 | 2.552 | ns |
| SSTL-2 CLASS II | 20 mA | GCLK | $t_{CO}$ | 2.606 | 2.606 | 5.636 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.164 | 1.164 | 2.539 | ns |
| SSTL-2 CLASS II | 24 mA | GCLK | $t_{CO}$ | 2.601 | 2.601 | 5.634 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.160 | 1.160 | 2.537 | ns |
| SSTL-18 CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.643 | 2.643 | 5.749 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.193 | 1.193 | 2.639 | ns |
| SSTL-18 CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.649 | 2.649 | 5.708 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.203 | 1.203 | 2.607 | ns |
| SSTL-18 CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.626 | 2.626 | 5.686 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.182 | 1.182 | 2.588 | ns |
| SSTL-18 CLASS I | 10 mA | GCLK | $t_{CO}$ | 2.630 | 2.630 | 5.685 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.187 | 1.187 | 2.586 | ns |
| SSTL-18 CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.625 | 2.625 | 5.669 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.181 | 1.181 | 2.572 | ns |
| SSTL-18 CLASS II | 8 mA | GCLK | $t_{CO}$ | 2.614 | 2.614 | 5.635 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.170 | 1.170 | 2.538 | ns |
| SSTL-18 CLASS II | 16 mA | GCLK | $t_{CO}$ | 2.623 | 2.623 | 5.613 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.182 | 1.182 | 2.516 | ns |
| SSTL-18 CLASS II | 18 mA | GCLK | $t_{CO}$ | 2.616 | 2.616 | 5.621 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.178 | 1.178 | 2.524 | ns |
| SSTL-18 CLASS II | 20 mA | GCLK | $t_{CO}$ | 2.616 | 2.616 | 5.619 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.178 | 1.178 | 2.522 | ns |
| 1.8-V HSTL CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.637 | 2.637 | 5.676 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.196 | 1.196 | 2.570 | ns |
| 1.8-V HSTL CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.645 | 2.645 | 5.659 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.207 | 1.207 | 2.562 | ns |
| 1.8-V HSTL CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.623 | 2.623 | 5.648 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.185 | 1.185 | 2.551 | ns |
| 1.8-V HSTL CLASS I | 10 mA | GCLK | $t_{CO}$ | 2.627 | 2.627 | 5.654 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.189 | 1.189 | 2.557 | ns |
| 1.8-V HSTL CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.619 | 2.619 | 5.647 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.181 | 1.181 | 2.550 | ns |

**Table 4–63.** EP1AGX50 Column Pins Output Timing Parameters   (Part 4 of 4)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| 1.8-V HSTL CLASS II | 16 mA | GCLK | $t_{CO}$ | 2.602 | 2.602 | 5.574 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.164 | 1.164 | 2.314 | ns |
| 1.8-V HSTL CLASS II | 18 mA | GCLK | $t_{CO}$ | 2.604 | 2.604 | 5.578 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.166 | 1.166 | 2.325 | ns |
| 1.8-V HSTL CLASS II | 20 mA | GCLK | $t_{CO}$ | 2.604 | 2.604 | 5.577 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.166 | 1.166 | 2.334 | ns |
| 1.5-V HSTL CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.637 | 2.637 | 5.675 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.196 | 1.196 | 2.569 | ns |
| 1.5-V HSTL CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.644 | 2.644 | 5.651 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.206 | 1.206 | 2.554 | ns |
| 1.5-V HSTL CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.626 | 2.626 | 5.653 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.188 | 1.188 | 2.556 | ns |
| 1.5-V HSTL CLASS I | 10 mA | GCLK | $t_{CO}$ | 2.626 | 2.626 | 5.655 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.188 | 1.188 | 2.558 | ns |
| 1.5-V HSTL CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.620 | 2.620 | 5.653 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.182 | 1.182 | 2.556 | ns |
| 1.5-V HSTL CLASS II | 16 mA | GCLK | $t_{CO}$ | 2.607 | 2.607 | 5.573 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.169 | 1.169 | 2.368 | ns |
| 1.5-V HSTL CLASS II | 18 mA | GCLK | $t_{CO}$ | 2.610 | 2.610 | 5.571 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.172 | 1.172 | 2.378 | ns |
| 1.5-V HSTL CLASS II | 20 mA | GCLK | $t_{CO}$ | 2.612 | 2.612 | 5.581 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.174 | 1.174 | 2.391 | ns |
| 3.3-V PCI | — | GCLK | $t_{CO}$ | 2.786 | 2.786 | 5.803 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.322 | 1.322 | 2.697 | ns |
| 3.3-V PCI-X | — | GCLK | $t_{CO}$ | 2.786 | 2.786 | 5.803 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.322 | 1.322 | 2.697 | ns |
| LVDS | — | GCLK | $t_{CO}$ | 3.621 | 3.621 | 6.969 | ns |
| | | GCLK PLL | $t_{CO}$ | 2.190 | 2.190 | 3.880 | ns |

Table 4–64 through Table 4–65 list EP1AGX50 regional clock (RCLK) adder values that should be added to the GCLK values. These adder values are used to determine I/O timing when the I/O pin is driven using the regional clock. This applies for all I/O standards supported by Arria GX with general purpose I/O pins.

Table 4–64 lists row pin delay adders when using the regional clock in Arria GX devices.

**Table 4–64.** EP1AGX50 Row Pin Delay Adders for Regional Clock

| Parameter | Fast Corner | | –6 Speed Grade | Units |
|---|---|---|---|---|
| | Industrial | Commercial | | |
| RCLK input adder | 0.151 | 0.151 | 0.329 | ns |
| RCLK PLL input adder | 0.011 | 0.011 | 0.016 | ns |
| RCLK output adder | −0.151 | −0.151 | −0.329 | ns |
| RCLK PLL output adder | −0.011 | −0.011 | −0.016 | ns |

Table 4–65 lists column pin delay adders when using the regional clock in Arria GX devices.

**Table 4–65.** EP1AGX50 Column Pin Delay Adders for Regional Clock

| Parameter | Fast Corner | | –6 Speed Grade | Units |
|---|---|---|---|---|
| | Industrial | Commercial | | |
| RCLK input adder | 0.146 | 0.146 | 0.334 | ns |
| RCLK PLL input adder | −1.713 | −1.713 | −3.645 | ns |
| RCLK output adder | −0.146 | −0.146 | −0.336 | ns |
| RCLK PLL output adder | 1.716 | 1.716 | 4.488 | ns |

### EP1AGX60 I/O Timing Parameters

Table 4–66 through Table 4–69 list the maximum I/O timing parameters for EP1AGX60 devices for I/O standards which support general purpose I/O pins.

Table 4–66 lists I/O timing specifications.

**Table 4–66.** EP1AGX60 Row Pins Input Timing Parameters  (Part 1 of 3)

| I/O Standard | Clock | Parameter | Fast Model | | –6 Speed Grade | Units |
|---|---|---|---|---|---|---|
| | | | Industrial | Commercial | | |
| 3.3-V LVTTL | GCLK | $t_{SU}$ | 1.413 | 1.413 | 3.113 | ns |
| | | $t_H$ | −1.308 | −1.308 | −2.836 | ns |
| | GCLK PLL | $t_{SU}$ | 2.975 | 2.975 | 6.536 | ns |
| | | $t_H$ | −2.870 | −2.870 | −6.259 | ns |
| 3.3-V LVCMOS | GCLK | $t_{SU}$ | 1.413 | 1.413 | 3.113 | ns |
| | | $t_H$ | −1.308 | −1.308 | −2.836 | ns |
| | GCLK PLL | $t_{SU}$ | 2.975 | 2.975 | 6.536 | ns |
| | | $t_H$ | −2.870 | −2.870 | −6.259 | ns |
| 2.5 V | GCLK | $t_{SU}$ | 1.425 | 1.425 | 3.094 | ns |
| | | $t_H$ | −1.320 | −1.320 | −2.817 | ns |
| | GCLK PLL | $t_{SU}$ | 2.987 | 2.987 | 6.517 | ns |
| | | $t_H$ | −2.882 | −2.882 | −6.240 | ns |

**Table 4–66.** EP1AGX60 Row Pins Input Timing Parameters  (Part 2 of 3)

| I/O Standard | Clock | Parameter | Fast Model | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|
| | | | Industrial | Commercial | | |
| 1.8 V | GCLK | $t_{SU}$ | 1.477 | 1.477 | 3.275 | ns |
| | | $t_H$ | −1.372 | −1.372 | −2.998 | ns |
| | GCLK PLL | $t_{SU}$ | 3.049 | 3.049 | 6.718 | ns |
| | | $t_H$ | −2.944 | −2.944 | −6.441 | ns |
| 1.5 V | GCLK | $t_{SU}$ | 1.480 | 1.480 | 3.370 | ns |
| | | $t_H$ | −1.375 | −1.375 | −3.093 | ns |
| | GCLK PLL | $t_{SU}$ | 3.052 | 3.052 | 6.813 | ns |
| | | $t_H$ | −2.947 | −2.947 | −6.536 | ns |
| SSTL-2 CLASS I | GCLK | $t_{SU}$ | 1.237 | 1.237 | 2.566 | ns |
| | | $t_H$ | −1.132 | −1.132 | −2.289 | ns |
| | GCLK PLL | $t_{SU}$ | 2.800 | 2.800 | 5.990 | ns |
| | | $t_H$ | −2.695 | −2.695 | −5.713 | ns |
| SSTL-2 CLASS II | GCLK | $t_{SU}$ | 1.237 | 1.237 | 2.566 | ns |
| | | $t_H$ | −1.132 | −1.132 | −2.289 | ns |
| | GCLK PLL | $t_{SU}$ | 2.800 | 2.800 | 5.990 | ns |
| | | $t_H$ | −2.695 | −2.695 | −5.713 | ns |
| SSTL-18 CLASS I | GCLK | $t_{SU}$ | 1.255 | 1.255 | 2.649 | ns |
| | | $t_H$ | −1.150 | −1.150 | −2.372 | ns |
| | GCLK PLL | $t_{SU}$ | 2.827 | 2.827 | 6.092 | ns |
| | | $t_H$ | −2.722 | −2.722 | −5.815 | ns |
| SSTL-18 CLASS II | GCLK | $t_{SU}$ | 1.255 | 1.255 | 2.649 | ns |
| | | $t_H$ | −1.150 | −1.150 | −2.372 | ns |
| | GCLK PLL | $t_{SU}$ | 2.827 | 2.827 | 6.092 | ns |
| | | $t_H$ | −2.722 | −2.722 | −5.815 | ns |
| 1.8-V HSTL CLASS I | GCLK | $t_{SU}$ | 1.255 | 1.255 | 2.649 | ns |
| | | $t_H$ | −1.150 | −1.150 | −2.372 | ns |
| | GCLK PLL | $t_{SU}$ | 2.827 | 2.827 | 6.092 | ns |
| | | $t_H$ | −2.722 | −2.722 | −5.815 | ns |
| 1.8-V HSTL CLASS II | GCLK | $t_{SU}$ | 1.255 | 1.255 | 2.649 | ns |
| | | $t_H$ | −1.150 | −1.150 | −2.372 | ns |
| | GCLK PLL | $t_{SU}$ | 2.827 | 2.827 | 6.092 | ns |
| | | $t_H$ | −2.722 | −2.722 | −5.815 | ns |
| 1.5-V HSTL CLASS I | GCLK | $t_{SU}$ | 1.281 | 1.281 | 2.777 | ns |
| | | $t_H$ | −1.176 | −1.176 | −2.500 | ns |
| | GCLK PLL | $t_{SU}$ | 2.853 | 2.853 | 6.220 | ns |
| | | $t_H$ | −2.748 | −2.748 | −5.943 | ns |

**Table 4–66.** EP1AGX60 Row Pins Input Timing Parameters  (Part 3 of 3)

| I/O Standard | Clock | Parameter | Fast Model | | −6 Speed Grade | Units |
| --- | --- | --- | --- | --- | --- | --- |
| | | | Industrial | Commercial | | |
| 1.5-V HSTL CLASS II | GCLK | $t_{SU}$ | 1.281 | 1.281 | 2.777 | ns |
| | | $t_H$ | −1.176 | −1.176 | −2.500 | ns |
| | GCLK PLL | $t_{SU}$ | 2.853 | 2.853 | 6.220 | ns |
| | | $t_H$ | −2.748 | −2.748 | −5.943 | ns |
| LVDS | GCLK | $t_{SU}$ | 1.208 | 1.208 | 2.664 | ns |
| | | $t_H$ | −1.103 | −1.103 | −2.387 | ns |
| | GCLK PLL | $t_{SU}$ | 2.767 | 2.767 | 6.083 | ns |
| | | $t_H$ | −2.662 | −2.662 | −5.806 | ns |

Table 4–67 lists I/O timing specifications.

**Table 4–67.** EP1AGX60 Column Pins Input Timing Parameters   (Part 1 of 3)

| I/O Standard | Clock | Parameter | Fast Corner | | −6 Speed Grade | Units |
| --- | --- | --- | --- | --- | --- | --- |
| | | | Industrial | Commercial | | |
| 3.3-V LVTTL | GCLK | $t_{SU}$ | 1.124 | 1.124 | 2.493 | ns |
| | | $t_H$ | −1.019 | −1.019 | -2.216 | ns |
| | GCLK PLL | $t_{SU}$ | 2.694 | 2.694 | 5.928 | ns |
| | | $t_H$ | −2.589 | −2.589 | -5.651 | ns |
| 3.3-V LVCMOS | GCLK | $t_{SU}$ | 1.124 | 1.124 | 2.493 | ns |
| | | $t_H$ | −1.019 | −1.019 | -2.216 | ns |
| | GCLK PLL | $t_{SU}$ | 2.694 | 2.694 | 5.928 | ns |
| | | $t_H$ | −2.589 | −2.589 | -5.651 | ns |
| 2.5 V | GCLK | $t_{SU}$ | 1.134 | 1.134 | 2.475 | ns |
| | | $t_H$ | −1.029 | −1.029 | -2.198 | ns |
| | GCLK PLL | $t_{SU}$ | 2.704 | 2.704 | 5.910 | ns |
| | | $t_H$ | −2.599 | −2.599 | -5.633 | ns |
| 1.8 V | GCLK | $t_{SU}$ | 1.200 | 1.200 | 2.685 | ns |
| | | $t_H$ | −1.095 | −1.095 | -2.408 | ns |
| | GCLK PLL | $t_{SU}$ | 2.770 | 2.770 | 6.120 | ns |
| | | $t_H$ | −2.665 | −2.665 | -5.843 | ns |
| 1.5 V | GCLK | $t_{SU}$ | 1.203 | 1.203 | 2.778 | ns |
| | | $t_H$ | −1.098 | −1.098 | -2.501 | ns |
| | GCLK PLL | $t_{SU}$ | 2.773 | 2.773 | 6.213 | ns |
| | | $t_H$ | −2.668 | −2.668 | -5.936 | ns |
| SSTL-2 CLASS I | GCLK | $t_{SU}$ | 0.948 | 0.948 | 1.951 | ns |
| | | $t_H$ | −0.843 | −0.843 | -1.674 | ns |
| | GCLK PLL | $t_{SU}$ | 2.519 | 2.519 | 5.388 | ns |
| | | $t_H$ | −2.414 | −2.414 | -5.111 | ns |

**Table 4–67.** EP1AGX60 Column Pins Input Timing Parameters   (Part 2 of 3)

| I/O Standard | Clock | Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|
| | | | Industrial | Commercial | | |
| SSTL-2 CLASS II | GCLK | $t_{SU}$ | 0.948 | 0.948 | 1.951 | ns |
| | | $t_H$ | −0.843 | −0.843 | −1.674 | ns |
| | GCLK PLL | $t_{SU}$ | 2.519 | 2.519 | 5.388 | ns |
| | | $t_H$ | −2.414 | −2.414 | −5.111 | ns |
| SSTL-18 CLASS I | GCLK | $t_{SU}$ | 0.986 | 0.986 | 2.057 | ns |
| | | $t_H$ | −0.881 | −0.881 | −1.780 | ns |
| | GCLK PLL | $t_{SU}$ | 2.556 | 2.556 | 5.492 | ns |
| | | $t_H$ | −2.451 | −2.451 | −5.215 | ns |
| SSTL-18 CLASS II | GCLK | $t_{SU}$ | 0.987 | 0.987 | 2.058 | ns |
| | | $t_H$ | −0.882 | −0.882 | −1.781 | ns |
| | GCLK PLL | $t_{SU}$ | 2.558 | 2.558 | 5.495 | ns |
| | | $t_H$ | −2.453 | −2.453 | −5.218 | ns |
| 1.8-V HSTL CLASS I | GCLK | $t_{SU}$ | 0.986 | 0.986 | 2.057 | ns |
| | | $t_H$ | −0.881 | −0.881 | −1.780 | ns |
| | GCLK PLL | $t_{SU}$ | 2.556 | 2.556 | 5.492 | ns |
| | | $t_H$ | −2.451 | −2.451 | −5.215 | ns |
| 1.8-V HSTL CLASS II | GCLK | $t_{SU}$ | 0.987 | 0.987 | 2.058 | ns |
| | | $t_H$ | −0.882 | −0.882 | −1.781 | ns |
| | GCLK PLL | $t_{SU}$ | 2.558 | 2.558 | 5.495 | ns |
| | | $t_H$ | −2.453 | −2.453 | −5.218 | ns |
| 1.5-V HSTL CLASS I | GCLK | $t_{SU}$ | 1.004 | 1.004 | 2.185 | ns |
| | | $t_H$ | −0.899 | −0.899 | −1.908 | ns |
| | GCLK PLL | $t_{SU}$ | 2.574 | 2.574 | 5.620 | ns |
| | | $t_H$ | −2.469 | −2.469 | −5.343 | ns |
| 1.5-V HSTL CLASS II | GCLK | $t_{SU}$ | 1.005 | 1.005 | 2.186 | ns |
| | | $t_H$ | −0.900 | −0.900 | −1.909 | ns |
| | GCLK PLL | $t_{SU}$ | 2.576 | 2.576 | 5.623 | ns |
| | | $t_H$ | −2.471 | −2.471 | −5.346 | ns |
| 3.3-V PCI | GCLK | $t_{SU}$ | 1.129 | 1.129 | 2.481 | ns |
| | | $t_H$ | −1.024 | −1.024 | −2.204 | ns |
| | GCLK PLL | $t_{SU}$ | 2.699 | 2.699 | 5.916 | ns |
| | | $t_H$ | −2.594 | −2.594 | −5.639 | ns |
| 3.3-V PCI-X | GCLK | $t_{SU}$ | 1.129 | 1.129 | 2.481 | ns |
| | | $t_H$ | −1.024 | −1.024 | −2.204 | ns |
| | GCLK PLL | $t_{SU}$ | 2.699 | 2.699 | 5.916 | ns |
| | | $t_H$ | −2.594 | −2.594 | −5.639 | ns |

**Table 4–67.** EP1AGX60 Column Pins Input Timing Parameters   (Part 3 of 3)

| I/O Standard | Clock | Parameter | Fast Corner | | –6 Speed Grade | Units |
|---|---|---|---|---|---|---|
| | | | Industrial | Commercial | | |
| LVDS | GCLK | $t_{SU}$ | 0.980 | 0.980 | 2.062 | ns |
| | | $t_H$ | −0.875 | −0.875 | −1.785 | ns |
| | GCLK PLL | $t_{SU}$ | 2.557 | 2.557 | 5.512 | ns |
| | | $t_H$ | −2.452 | −2.452 | −5.235 | ns |

Table 4–68 lists I/O timing specifications.

**Table 4–68.** EP1AGX60 Row Pins Output Timing Parameters   (Part 1 of 2)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Model | | –6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| 3.3-V LVTTL | 4 mA | GCLK | $t_{CO}$ | 3.052 | 3.052 | 7.142 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.490 | 1.490 | 3.719 | ns |
| 3.3-V LVTTL | 8 mA | GCLK | $t_{CO}$ | 2.924 | 2.924 | 6.502 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.362 | 1.362 | 3.079 | ns |
| 3.3-V LVTTL | 12 mA | GCLK | $t_{CO}$ | 2.868 | 2.868 | 6.465 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.306 | 1.306 | 3.042 | ns |
| 3.3-V LVCMOS | 4 mA | GCLK | $t_{CO}$ | 2.924 | 2.924 | 6.502 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.362 | 1.362 | 3.079 | ns |
| 3.3-V LVCMOS | 8 mA | GCLK | $t_{CO}$ | 2.818 | 2.818 | 6.196 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.256 | 1.256 | 2.773 | ns |
| 2.5 V | 4 mA | GCLK | $t_{CO}$ | 2.907 | 2.907 | 6.476 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.345 | 1.345 | 3.053 | ns |
| 2.5 V | 8 mA | GCLK | $t_{CO}$ | 2.804 | 2.804 | 6.218 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.242 | 1.242 | 2.795 | ns |
| 2.5 V | 12 mA | GCLK | $t_{CO}$ | 2.785 | 2.785 | 6.104 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.223 | 1.223 | 2.681 | ns |
| 1.8 V | 2 mA | GCLK | $t_{CO}$ | 2.991 | 2.991 | 7.521 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.419 | 1.419 | 4.078 | ns |
| 1.8 V | 4 mA | GCLK | $t_{CO}$ | 2.980 | 2.980 | 6.742 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.408 | 1.408 | 3.299 | ns |
| 1.8 V | 6 mA | GCLK | $t_{CO}$ | 2.869 | 2.869 | 6.441 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.297 | 1.297 | 2.998 | ns |
| 1.8 V | 8 mA | GCLK | $t_{CO}$ | 2.838 | 2.838 | 6.327 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.266 | 1.266 | 2.884 | ns |
| 1.5 V | 2 mA | GCLK | $t_{CO}$ | 2.951 | 2.951 | 7.020 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.379 | 1.379 | 3.577 | ns |
| 1.5 V | 4 mA | GCLK | $t_{CO}$ | 2.844 | 2.844 | 6.419 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.272 | 1.272 | 2.976 | ns |

**Table 4–68.** EP1AGX60 Row Pins Output Timing Parameters   (Part 2 of 2)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Model | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| SSTL-2 CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.774 | 2.774 | 6.057 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.211 | 1.211 | 2.633 | ns |
| SSTL-2 CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.750 | 2.750 | 5.981 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.187 | 1.187 | 2.557 | ns |
| SSTL-2 CLASS II | 16 mA | GCLK | $t_{CO}$ | 2.716 | 2.716 | 5.850 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.153 | 1.153 | 2.426 | ns |
| SSTL-18 CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.776 | 2.776 | 6.025 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.204 | 1.204 | 2.582 | ns |
| SSTL-18 CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.780 | 2.780 | 5.954 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.208 | 1.208 | 2.511 | ns |
| SSTL-18 CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.756 | 2.756 | 5.937 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.184 | 1.184 | 2.494 | ns |
| SSTL-18 CLASS I | 10 mA | GCLK | $t_{CO}$ | 2.759 | 2.759 | 5.916 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.187 | 1.187 | 2.473 | ns |
| 1.8-V HSTL CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.757 | 2.757 | 5.935 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.185 | 1.185 | 2.492 | ns |
| 1.8-V HSTL CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.760 | 2.760 | 5.899 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.188 | 1.188 | 2.456 | ns |
| 1.8-V HSTL CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.742 | 2.742 | 5.895 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.170 | 1.170 | 2.452 | ns |
| 1.8-V HSTL CLASS I | 10 mA | GCLK | $t_{CO}$ | 2.746 | 2.746 | 5.884 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.174 | 1.174 | 2.441 | ns |
| 1.8-V HSTL CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.737 | 2.737 | 5.883 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.165 | 1.165 | 2.440 | ns |
| 1.5-V HSTL CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.756 | 2.756 | 5.912 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.184 | 1.184 | 2.469 | ns |
| 1.5-V HSTL CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.759 | 2.759 | 5.898 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.187 | 1.187 | 2.455 | ns |
| 1.5-V HSTL CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.744 | 2.744 | 5.890 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.172 | 1.172 | 2.447 | ns |
| LVDS | — | GCLK | $t_{CO}$ | 2.787 | 2.787 | 6.037 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.228 | 1.228 | 2.618 | ns |

Table 4–69 lists I/O timing specifications.

**Table 4–69.** EP1AGX60 Column Pins Output Timing Parameters  (Part 1 of 4)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| 3.3-V LVTTL | 4 mA | GCLK | $t_{CO}$ | 3.036 | 3.036 | 6.963 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.466 | 1.466 | 3.528 | ns |
| 3.3-V LVTTL | 8 mA | GCLK | $t_{CO}$ | 2.891 | 2.891 | 6.591 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.321 | 1.321 | 3.156 | ns |
| 3.3-V LVTTL | 12 mA | GCLK | $t_{CO}$ | 2.824 | 2.824 | 6.591 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.254 | 1.254 | 3.156 | ns |
| 3.3-V LVTTL | 16 mA | GCLK | $t_{CO}$ | 2.798 | 2.798 | 6.422 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.228 | 1.228 | 2.987 | ns |
| 3.3-V LVTTL | 20 mA | GCLK | $t_{CO}$ | 2.776 | 2.776 | 6.297 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.206 | 1.206 | 2.862 | ns |
| 3.3-V LVTTL | 24 mA | GCLK | $t_{CO}$ | 2.769 | 2.769 | 6.299 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.199 | 1.199 | 2.864 | ns |
| 3.3-V LVCMOS | 4 mA | GCLK | $t_{CO}$ | 2.891 | 2.891 | 6.591 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.321 | 1.321 | 3.156 | ns |
| 3.3-V LVCMOS | 8 mA | GCLK | $t_{CO}$ | 2.799 | 2.799 | 6.296 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.229 | 1.229 | 2.861 | ns |
| 3.3-V LVCMOS | 12 mA | GCLK | $t_{CO}$ | 2.771 | 2.771 | 6.218 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.201 | 1.201 | 2.783 | ns |
| 3.3-V LVCMOS | 16 mA | GCLK | $t_{CO}$ | 2.778 | 2.778 | 6.186 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.208 | 1.208 | 2.751 | ns |
| 3.3-V LVCMOS | 20 mA | GCLK | $t_{CO}$ | 2.765 | 2.765 | 6.168 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.195 | 1.195 | 2.733 | ns |
| 3.3-V LVCMOS | 24 mA | GCLK | $t_{CO}$ | 2.754 | 2.754 | 6.146 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.184 | 1.184 | 2.711 | ns |
| 2.5 V | 4 mA | GCLK | $t_{CO}$ | 2.853 | 2.853 | 6.623 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.283 | 1.283 | 3.188 | ns |
| 2.5 V | 8 mA | GCLK | $t_{CO}$ | 2.801 | 2.801 | 6.361 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.231 | 1.231 | 2.926 | ns |
| 2.5 V | 12 mA | GCLK | $t_{CO}$ | 2.780 | 2.780 | 6.244 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.210 | 1.210 | 2.809 | ns |
| 2.5 V | 16 mA | GCLK | $t_{CO}$ | 2.762 | 2.762 | 6.170 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.192 | 1.192 | 2.735 | ns |
| 1.8 V | 2 mA | GCLK | $t_{CO}$ | 2.893 | 2.893 | 7.615 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.323 | 1.323 | 4.180 | ns |
| 1.8 V | 4 mA | GCLK | $t_{CO}$ | 2.898 | 2.898 | 6.841 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.328 | 1.328 | 3.406 | ns |

**Table 4–69.** EP1AGX60 Column Pins Output Timing Parameters  (Part 2 of 4)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| 1.8 V | 6 mA | GCLK | $t_{CO}$ | 2.822 | 2.822 | 6.577 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.252 | 1.252 | 3.142 | ns |
| 1.8 V | 8 mA | GCLK | $t_{CO}$ | 2.824 | 2.824 | 6.486 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.254 | 1.254 | 3.051 | ns |
| 1.8 V | 10 mA | GCLK | $t_{CO}$ | 2.778 | 2.778 | 6.409 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.208 | 1.208 | 2.974 | ns |
| 1.8 V | 12 mA | GCLK | $t_{CO}$ | 2.779 | 2.779 | 6.352 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.209 | 1.209 | 2.917 | ns |
| 1.5 V | 2 mA | GCLK | $t_{CO}$ | 2.873 | 2.873 | 7.145 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.303 | 1.303 | 3.710 | ns |
| 1.5 V | 4 mA | GCLK | $t_{CO}$ | 2.809 | 2.809 | 6.576 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.239 | 1.239 | 3.141 | ns |
| 1.5 V | 6 mA | GCLK | $t_{CO}$ | 2.812 | 2.812 | 6.458 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.242 | 1.242 | 3.023 | ns |
| 1.5 V | 8 mA | GCLK | $t_{CO}$ | 2.771 | 2.771 | 6.405 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.201 | 1.201 | 2.970 | ns |
| SSTL-2 CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.757 | 2.757 | 6.184 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.184 | 1.184 | 2.744 | ns |
| SSTL-2 CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.740 | 2.740 | 6.134 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.167 | 1.167 | 2.694 | ns |
| SSTL-2 CLASS II | 16 mA | GCLK | $t_{CO}$ | 2.718 | 2.718 | 6.061 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.145 | 1.145 | 2.621 | ns |
| SSTL-2 CLASS II | 20 mA | GCLK | $t_{CO}$ | 2.719 | 2.719 | 6.048 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.146 | 1.146 | 2.608 | ns |
| SSTL-2 CLASS II | 24 mA | GCLK | $t_{CO}$ | 2.715 | 2.715 | 6.046 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.142 | 1.142 | 2.606 | ns |
| SSTL-18 CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.753 | 2.753 | 6.155 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.183 | 1.183 | 2.720 | ns |
| SSTL-18 CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.758 | 2.758 | 6.116 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.185 | 1.185 | 2.676 | ns |
| SSTL-18 CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.737 | 2.737 | 6.097 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.164 | 1.164 | 2.657 | ns |
| SSTL-18 CLASS I | 10 mA | GCLK | $t_{CO}$ | 2.742 | 2.742 | 6.095 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.169 | 1.169 | 2.655 | ns |
| SSTL-18 CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.736 | 2.736 | 6.081 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.163 | 1.163 | 2.641 | ns |
| SSTL-18 CLASS II | 8 mA | GCLK | $t_{CO}$ | 2.725 | 2.725 | 6.047 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.152 | 1.152 | 2.607 | ns |

**Table 4–69.** EP1AGX60 Column Pins Output Timing Parameters  (Part 3 of 4)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| SSTL-18 CLASS II | 16 mA | GCLK | $t_{CO}$ | 2.737 | 2.737 | 6.025 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.164 | 1.164 | 2.585 | ns |
| SSTL-18 CLASS II | 18 mA | GCLK | $t_{CO}$ | 2.733 | 2.733 | 6.033 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.160 | 1.160 | 2.593 | ns |
| SSTL-18 CLASS II | 20 mA | GCLK | $t_{CO}$ | 2.733 | 2.733 | 6.031 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.160 | 1.160 | 2.591 | ns |
| 1.8-V HSTL CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.756 | 2.756 | 6.086 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.186 | 1.186 | 2.651 | ns |
| 1.8-V HSTL CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.762 | 2.762 | 6.071 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.189 | 1.189 | 2.631 | ns |
| 1.8-V HSTL CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.740 | 2.740 | 6.060 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.167 | 1.167 | 2.620 | ns |
| 1.8-V HSTL CLASS I | 10 mA | GCLK | $t_{CO}$ | 2.744 | 2.744 | 6.066 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.171 | 1.171 | 2.626 | ns |
| 1.8-V HSTL CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.736 | 2.736 | 6.059 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.163 | 1.163 | 2.619 | ns |
| 1.8-V HSTL CLASS II | 16 mA | GCLK | $t_{CO}$ | 2.719 | 2.719 | 5.823 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.146 | 1.146 | 2.383 | ns |
| 1.8-V HSTL CLASS II | 18 mA | GCLK | $t_{CO}$ | 2.721 | 2.721 | 5.834 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.148 | 1.148 | 2.394 | ns |
| 1.8-V HSTL CLASS II | 20 mA | GCLK | $t_{CO}$ | 2.721 | 2.721 | 5.843 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.148 | 1.148 | 2.403 | ns |
| 1.5-V HSTL CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.756 | 2.756 | 6.085 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.186 | 1.186 | 2.650 | ns |
| 1.5-V HSTL CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.761 | 2.761 | 6.063 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.188 | 1.188 | 2.623 | ns |
| 1.5-V HSTL CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.743 | 2.743 | 6.065 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.170 | 1.170 | 2.625 | ns |
| 1.5-V HSTL CLASS I | 10 mA | GCLK | $t_{CO}$ | 2.743 | 2.743 | 6.067 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.170 | 1.170 | 2.627 | ns |
| 1.5-V HSTL CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.737 | 2.737 | 6.065 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.164 | 1.164 | 2.625 | ns |
| 1.5-V HSTL CLASS II | 16 mA | GCLK | $t_{CO}$ | 2.724 | 2.724 | 5.877 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.151 | 1.151 | 2.437 | ns |
| 1.5-V HSTL CLASS II | 18 mA | GCLK | $t_{CO}$ | 2.727 | 2.727 | 5.887 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.154 | 1.154 | 2.447 | ns |
| 1.5-V HSTL CLASS II | 20 mA | GCLK | $t_{CO}$ | 2.729 | 2.729 | 5.900 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.156 | 1.156 | 2.460 | ns |

**Table 4–69.** EP1AGX60 Column Pins Output Timing Parameters  (Part 4 of 4)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| 3.3-V PCI | — | GCLK | $t_{CO}$ | 2.882 | 2.882 | 6.213 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.312 | 1.312 | 2.778 | ns |
| 3.3-V PCI-X | — | GCLK | $t_{CO}$ | 2.882 | 2.882 | 6.213 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.312 | 1.312 | 2.778 | ns |
| LVDS | — | GCLK | $t_{CO}$ | 3.746 | 3.746 | 7.396 | ns |
| | | GCLK PLL | $t_{CO}$ | 2.185 | 2.185 | 3.973 | ns |

Table 4–70 through Table 4–71 list EP1AGX60 regional clock (RCLK) adder values that should be added to the GCLK values. These adder values are used to determine I/O timing when the I/O pin is driven using the regional clock. This applies for all I/O standards supported by Arria GX with general purpose I/O pins.

Table 4–70 describes row pin delay adders when using the regional clock in Arria GX devices.

**Table 4–70.** EP1AGX60 Row Pin Delay Adders for Regional Clock

| Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|
| | Industrial | Commercial | | |
| RCLK input adder | 0.138 | 0.138 | 0.311 | ns |
| RCLK PLL input adder | −0.003 | −0.003 | −0.006 | ns |
| RCLK output adder | −0.138 | −0.138 | −0.311 | ns |
| RCLK PLL output adder | 0.003 | 0.003 | 0.006 | ns |

Table 4–71 lists column pin delay adders when using the regional clock in Arria GX devices.

**Table 4–71.** EP1AGX60 Column Pin Delay Adders for Regional Clock

| Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|
| | Industrial | Commercial | | |
| RCLK input adder | 0.153 | 0.153 | 0.344 | ns |
| RCLK PLL input adder | −1.066 | −1.066 | −2.338 | ns |
| RCLK output adder | −0.153 | −0.153 | −0.343 | ns |
| RCLK PLL output adder | 1.721 | 1.721 | 4.486 | ns |

### EP1AGX90 I/O Timing Parameters

Table 4–72 through Table 4–75 list the maximum I/O timing parameters for EP1AGX90 devices for I/O standards which support general purpose I/O pins.

Table 4–72 lists I/O timing specifications.

**Table 4–72.** EP1AGX90 Row Pins Input Timing Parameters  (Part 1 of 2)

| I/O Standard | Clock | Parameter | Fast Model | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|
| | | | Industrial | Commercial | | |
| 3.3-V LVTTL | GCLK | $t_{SU}$ | 1.295 | 1.295 | 2.873 | ns |
| | | $t_H$ | −1.190 | −1.190 | −2.596 | ns |
| | GCLK PLL | $t_{SU}$ | 3.366 | 3.366 | 7.017 | ns |
| | | $t_H$ | −3.261 | −3.261 | −6.740 | ns |
| 3.3-V LVCMOS | GCLK | $t_{SU}$ | 1.295 | 1.295 | 2.873 | ns |
| | | $t_H$ | −1.190 | −1.190 | −2.596 | ns |
| | GCLK PLL | $t_{SU}$ | 3.366 | 3.366 | 7.017 | ns |
| | | $t_H$ | −3.261 | −3.261 | −6.740 | ns |
| 2.5 V | GCLK | $t_{SU}$ | 1.307 | 1.307 | 2.854 | ns |
| | | $t_H$ | −1.202 | −1.202 | −2.577 | ns |
| | GCLK PLL | $t_{SU}$ | 3.378 | 3.378 | 6.998 | ns |
| | | $t_H$ | −3.273 | −3.273 | −6.721 | ns |
| 1.8 V | GCLK | $t_{SU}$ | 1.381 | 1.381 | 3.073 | ns |
| | | $t_H$ | −1.276 | −1.276 | −2.796 | ns |
| | GCLK PLL | $t_{SU}$ | 3.434 | 3.434 | 7.191 | ns |
| | | $t_H$ | −3.329 | −3.329 | −6.914 | ns |
| 1.5 V | GCLK | $t_{SU}$ | 1.384 | 1.384 | 3.168 | ns |
| | | $t_H$ | −1.279 | −1.279 | −2.891 | ns |
| | GCLK PLL | $t_{SU}$ | 3.437 | 3.437 | 7.286 | ns |
| | | $t_H$ | −3.332 | −3.332 | −7.009 | ns |
| SSTL-2 CLASS I | GCLK | $t_{SU}$ | 1.121 | 1.121 | 2.329 | ns |
| | | $t_H$ | −1.016 | −1.016 | −2.052 | ns |
| | GCLK PLL | $t_{SU}$ | 3.187 | 3.187 | 6.466 | ns |
| | | $t_H$ | −3.082 | −3.082 | −6.189 | ns |
| SSTL-2 CLASS II | GCLK | $t_{SU}$ | 1.121 | 1.121 | 2.329 | ns |
| | | $t_H$ | −1.016 | −1.016 | −2.052 | ns |
| | GCLK PLL | $t_{SU}$ | 3.187 | 3.187 | 6.466 | ns |
| | | $t_H$ | −3.082 | −3.082 | −6.189 | ns |
| SSTL-18 CLASS I | GCLK | $t_{SU}$ | 1.159 | 1.159 | 2.447 | ns |
| | | $t_H$ | −1.054 | −1.054 | −2.170 | ns |
| | GCLK PLL | $t_{SU}$ | 3.212 | 3.212 | 6.565 | ns |
| | | $t_H$ | −3.107 | −3.107 | −6.288 | ns |
| SSTL-18 CLASS II | GCLK | $t_{SU}$ | 1.157 | 1.157 | 2.441 | ns |
| | | $t_H$ | −1.052 | −1.052 | −2.164 | ns |
| | GCLK PLL | $t_{SU}$ | 3.235 | 3.235 | 6.597 | ns |
| | | $t_H$ | −3.130 | −3.130 | −6.320 | ns |

**Table 4–72.** EP1AGX90 Row Pins Input Timing Parameters  (Part 2 of 2)

| I/O Standard | Clock | Parameter | Fast Model | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|
| | | | Industrial | Commercial | | |
| 1.8-V HSTL CLASS I | GCLK | $t_{SU}$ | 1.159 | 1.159 | 2.447 | ns |
| | | $t_H$ | −1.054 | −1.054 | −2.170 | ns |
| | GCLK PLL | $t_{SU}$ | 3.212 | 3.212 | 6.565 | ns |
| | | $t_H$ | −3.107 | −3.107 | −6.288 | ns |
| 1.8-V HSTL CLASS II | GCLK | $t_{SU}$ | 1.157 | 1.157 | 2.441 | ns |
| | | $t_H$ | −1.052 | −1.052 | −2.164 | ns |
| | GCLK PLL | $t_{SU}$ | 3.235 | 3.235 | 6.597 | ns |
| | | $t_H$ | −3.130 | −3.130 | −6.320 | ns |
| 1.5-V HSTL CLASS I | GCLK | $t_{SU}$ | 1.185 | 1.185 | 2.575 | ns |
| | | $t_H$ | −1.080 | −1.080 | −2.298 | ns |
| | GCLK PLL | $t_{SU}$ | 3.238 | 3.238 | 6.693 | ns |
| | | $t_H$ | −3.133 | −3.133 | −6.416 | ns |
| 1.5-V HSTL CLASS II | GCLK | $t_{SU}$ | 1.183 | 1.183 | 2.569 | ns |
| | | $t_H$ | −1.078 | −1.078 | −2.292 | ns |
| | GCLK PLL | $t_{SU}$ | 3.261 | 3.261 | 6.725 | ns |
| | | $t_H$ | −3.156 | −3.156 | −6.448 | ns |
| LVDS | GCLK | $t_{SU}$ | 1.098 | 1.098 | 2.439 | ns |
| | | $t_H$ | −0.993 | −0.993 | −2.162 | ns |
| | GCLK PLL | $t_{SU}$ | 3.160 | 3.160 | 6.566 | ns |
| | | $t_H$ | −3.055 | −3.055 | −6.289 | ns |

Table 4–73 lists I/O timing specifications.

\

**Table 4–73.** EP1AGX90 Column Pins Input Timing Parameters  (Part 1 of 3)

| I/O Standard | Clock | Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|
| | | | Industrial | Commercial | | |
| 3.3-V LVTTL | GCLK | $t_{SU}$ | 1.018 | 1.018 | 2.290 | ns |
| | | $t_H$ | −0.913 | −0.913 | −2.013 | ns |
| | GCLK PLL | $t_{SU}$ | 3.082 | 3.082 | 6.425 | ns |
| | | $t_H$ | −2.977 | −2.977 | −6.148 | ns |
| 3.3-V LVCMOS | GCLK | $t_{SU}$ | 1.018 | 1.018 | 2.290 | ns |
| | | $t_H$ | −0.913 | −0.913 | −2.013 | ns |
| | GCLK PLL | $t_{SU}$ | 3.082 | 3.082 | 6.425 | ns |
| | | $t_H$ | −2.977 | −2.977 | −6.148 | ns |
| 2.5 V | GCLK | $t_{SU}$ | 1.028 | 1.028 | 2.272 | ns |
| | | $t_H$ | −0.923 | −0.923 | −1.995 | ns |
| | GCLK PLL | $t_{SU}$ | 3.092 | 3.092 | 6.407 | ns |
| | | $t_H$ | −2.987 | −2.987 | −6.130 | ns |

**Table 4–73.** EP1AGX90 Column Pins Input Timing Parameters  (Part 2 of 3)

| I/O Standard | Clock | Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|
| | | | Industrial | Commercial | | |
| 1.8 V | GCLK | $t_{SU}$ | 1.094 | 1.094 | 2.482 | ns |
| | | $t_H$ | −0.989 | −0.989 | −2.205 | ns |
| | GCLK PLL | $t_{SU}$ | 3.158 | 3.158 | 6.617 | ns |
| | | $t_H$ | −3.053 | −3.053 | −6.340 | ns |
| 1.5 V | GCLK | $t_{SU}$ | 1.097 | 1.097 | 2.575 | ns |
| | | $t_H$ | −0.992 | −0.992 | −2.298 | ns |
| | GCLK PLL | $t_{SU}$ | 3.161 | 3.161 | 6.710 | ns |
| | | $t_H$ | −3.056 | −3.056 | −6.433 | ns |
| SSTL-2 CLASS I | GCLK | $t_{SU}$ | 0.844 | 0.844 | 1.751 | ns |
| | | $t_H$ | −0.739 | −0.739 | −1.474 | ns |
| | GCLK PLL | $t_{SU}$ | 2.908 | 2.908 | 5.886 | ns |
| | | $t_H$ | −2.803 | −2.803 | −5.609 | ns |
| SSTL-2 CLASS II | GCLK | $t_{SU}$ | 0.844 | 0.844 | 1.751 | ns |
| | | $t_H$ | −0.739 | −0.739 | −1.474 | ns |
| | GCLK PLL | $t_{SU}$ | 2.908 | 2.908 | 5.886 | ns |
| | | $t_H$ | −2.803 | −2.803 | −5.609 | ns |
| SSTL-18 CLASS I | GCLK | $t_{SU}$ | 0.880 | 0.880 | 1.854 | ns |
| | | $t_H$ | −0.775 | −0.775 | −1.577 | ns |
| | GCLK PLL | $t_{SU}$ | 2.944 | 2.944 | 5.989 | ns |
| | | $t_H$ | −2.839 | −2.839 | −5.712 | ns |
| SSTL-18 CLASS II | GCLK | $t_{SU}$ | 0.883 | 0.883 | 1.858 | ns |
| | | $t_H$ | −0.778 | −0.778 | −1.581 | ns |
| | GCLK PLL | $t_{SU}$ | 2.947 | 2.947 | 5.993 | ns |
| | | $t_H$ | −2.842 | −2.842 | −5.716 | ns |
| 1.8-V HSTL CLASS I | GCLK | $t_{SU}$ | 0.880 | 0.880 | 1.854 | ns |
| | | $t_H$ | −0.775 | −0.775 | −1.577 | ns |
| | GCLK PLL | $t_{SU}$ | 2.944 | 2.944 | 5.989 | ns |
| | | $t_H$ | −2.839 | −2.839 | −5.712 | ns |
| 1.8-V HSTL CLASS II | GCLK | $t_{SU}$ | 0.883 | 0.883 | 1.858 | ns |
| | | $t_H$ | −0.778 | −0.778 | −1.581 | ns |
| | GCLK PLL | $t_{SU}$ | 2.947 | 2.947 | 5.993 | ns |
| | | $t_H$ | −2.842 | −2.842 | −5.716 | ns |
| 1.5-V HSTL CLASS I | GCLK | $t_{SU}$ | 0.898 | 0.898 | 1.982 | ns |
| | | $t_H$ | −0.793 | −0.793 | −1.705 | ns |
| | GCLK PLL | $t_{SU}$ | 2.962 | 2.962 | 6.117 | ns |
| | | $t_H$ | −2.857 | −2.857 | −5.840 | ns |

**Table 4–73.** EP1AGX90 Column Pins Input Timing Parameters  (Part 3 of 3)

| I/O Standard | Clock | Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|
| | | | Industrial | Commercial | | |
| 1.5-V HSTL CLASS II | GCLK | $t_{SU}$ | 0.901 | 0.901 | 1.986 | ns |
| | | $t_H$ | −0.796 | −0.796 | −1.709 | ns |
| | GCLK PLL | $t_{SU}$ | 2.965 | 2.965 | 6.121 | ns |
| | | $t_H$ | −2.860 | −2.860 | −5.844 | ns |
| 3.3-V PCI | GCLK | $t_{SU}$ | 1.023 | 1.023 | 2.278 | ns |
| | | $t_H$ | −0.918 | −0.918 | −2.001 | ns |
| | GCLK PLL | $t_{SU}$ | 3.087 | 3.087 | 6.413 | ns |
| | | $t_H$ | −2.982 | −2.982 | −6.136 | ns |
| 3.3-V PCI-X | GCLK | $t_{SU}$ | 1.023 | 1.023 | 2.278 | ns |
| | | $t_H$ | −0.918 | −0.918 | −2.001 | ns |
| | GCLK PLL | $t_{SU}$ | 3.087 | 3.087 | 6.413 | ns |
| | | $t_H$ | −2.982 | −2.982 | −6.136 | ns |
| LVDS | GCLK | $t_{SU}$ | 0.891 | 0.891 | 1.920 | ns |
| | | $t_H$ | −0.786 | −0.786 | −1.643 | ns |
| | GCLK PLL | $t_{SU}$ | 2.963 | 2.963 | 6.066 | ns |
| | | $t_H$ | −2.858 | −2.858 | −5.789 | ns |

Table 4–74 lists I/O timing specifications.

**Table 4–74.** EP1AGX90 Row Pins Output Timing Parameters   (Part 1 of 3)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Model | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| 3.3-V LVTTL | 4 mA | GCLK | $t_{CO}$ | 3.170 | 3.170 | 7.382 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.099 | 1.099 | 3.238 | ns |
| 3.3-V LVTTL | 8 mA | GCLK | $t_{CO}$ | 3.042 | 3.042 | 6.742 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.971 | 0.971 | 2.598 | ns |
| 3.3-V LVTTL | 12 mA | GCLK | $t_{CO}$ | 2.986 | 2.986 | 6.705 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.915 | 0.915 | 2.561 | ns |
| 3.3-V LVCMOS | 4 mA | GCLK | $t_{CO}$ | 3.042 | 3.042 | 6.742 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.971 | 0.971 | 2.598 | ns |
| 3.3-V LVCMOS | 8 mA | GCLK | $t_{CO}$ | 2.936 | 2.936 | 6.436 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.865 | 0.865 | 2.292 | ns |
| 2.5 V | 4 mA | GCLK | $t_{CO}$ | 3.025 | 3.025 | 6.716 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.954 | 0.954 | 2.572 | ns |
| 2.5 V | 8 mA | GCLK | $t_{CO}$ | 2.922 | 2.922 | 6.458 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.851 | 0.851 | 2.314 | ns |
| 2.5 V | 12 mA | GCLK | $t_{CO}$ | 2.903 | 2.903 | 6.344 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.832 | 0.832 | 2.200 | ns |

**Table 4–74.** EP1AGX90 Row Pins Output Timing Parameters   (Part 2 of 3)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Model | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| 1.8 V | 2 mA | GCLK | $t_{CO}$ | 3.087 | 3.087 | 7.723 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.034 | 1.034 | 3.605 | ns |
| 1.8 V | 4 mA | GCLK | $t_{CO}$ | 3.076 | 3.076 | 6.944 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.023 | 1.023 | 2.826 | ns |
| 1.8 V | 6 mA | GCLK | $t_{CO}$ | 2.965 | 2.965 | 6.643 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.912 | 0.912 | 2.525 | ns |
| 1.8 V | 8 mA | GCLK | $t_{CO}$ | 2.934 | 2.934 | 6.529 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.881 | 0.881 | 2.411 | ns |
| 1.5 V | 2 mA | GCLK | $t_{CO}$ | 3.047 | 3.047 | 7.222 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.994 | 0.994 | 3.104 | ns |
| 1.5 V | 4 mA | GCLK | $t_{CO}$ | 2.940 | 2.940 | 6.621 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.887 | 0.887 | 2.503 | ns |
| SSTL-2 CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.890 | 2.890 | 6.294 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.824 | 0.824 | 2.157 | ns |
| SSTL-2 CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.866 | 2.866 | 6.218 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.800 | 0.800 | 2.081 | ns |
| SSTL-2 CLASS II | 16 mA | GCLK | $t_{CO}$ | 2.832 | 2.832 | 6.087 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.766 | 0.766 | 1.950 | ns |
| SSTL-18 CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.872 | 2.872 | 6.227 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.819 | 0.819 | 2.109 | ns |
| SSTL-18 CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.878 | 2.878 | 6.162 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.800 | 0.800 | 2.006 | ns |
| SSTL-18 CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.854 | 2.854 | 6.145 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.776 | 0.776 | 1.989 | ns |
| SSTL-18 CLASS I | 10 mA | GCLK | $t_{CO}$ | 2.857 | 2.857 | 6.124 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.779 | 0.779 | 1.968 | ns |
| 1.8-V HSTL CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.853 | 2.853 | 6.137 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.800 | 0.800 | 2.019 | ns |
| 1.8-V HSTL CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.858 | 2.858 | 6.107 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.780 | 0.780 | 1.951 | ns |
| 1.8-V HSTL CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.840 | 2.840 | 6.103 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.762 | 0.762 | 1.947 | ns |
| 1.8-V HSTL CLASS I | 10 mA | GCLK | $t_{CO}$ | 2.844 | 2.844 | 6.092 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.766 | 0.766 | 1.936 | ns |
| 1.8-V HSTL CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.835 | 2.835 | 6.091 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.757 | 0.757 | 1.935 | ns |
| 1.5-V HSTL CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.852 | 2.852 | 6.114 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.799 | 0.799 | 1.996 | ns |

**Table 4–74.** EP1AGX90 Row Pins Output Timing Parameters   (Part 3 of 3)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Model | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| 1.5-V HSTL CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.857 | 2.857 | 6.106 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.779 | 0.779 | 1.950 | ns |
| 1.5-V HSTL CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.842 | 2.842 | 6.098 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.764 | 0.764 | 1.942 | ns |
| LVDS | — | GCLK | $t_{CO}$ | 2.898 | 2.898 | 6.265 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.831 | 0.831 | 2.129 | ns |

Table 4–75 lists I/O timing specifications.

**Table 4–75.** EP1AGX90 Column Pins Output Timing Parameters  (Part 1 of 4)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| 3.3-V LVTTL | 4 mA | GCLK | $t_{CO}$ | 3.141 | 3.141 | 7.164 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.077 | 1.077 | 3.029 | ns |
| 3.3-V LVTTL | 8 mA | GCLK | $t_{CO}$ | 2.996 | 2.996 | 6.792 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.932 | 0.932 | 2.657 | ns |
| 3.3-V LVTTL | 12 mA | GCLK | $t_{CO}$ | 2.929 | 2.929 | 6.792 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.865 | 0.865 | 2.657 | ns |
| 3.3-V LVTTL | 16 mA | GCLK | $t_{CO}$ | 2.903 | 2.903 | 6.623 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.839 | 0.839 | 2.488 | ns |
| 3.3-V LVTTL | 20 mA | GCLK | $t_{CO}$ | 2.881 | 2.881 | 6.498 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.817 | 0.817 | 2.363 | ns |
| 3.3-V LVTTL | 24 mA | GCLK | $t_{CO}$ | 2.874 | 2.874 | 6.500 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.810 | 0.810 | 2.365 | ns |
| 3.3-V LVCMOS | 4 mA | GCLK | $t_{CO}$ | 2.996 | 2.996 | 6.792 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.932 | 0.932 | 2.657 | ns |
| 3.3-V LVCMOS | 8 mA | GCLK | $t_{CO}$ | 2.904 | 2.904 | 6.497 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.840 | 0.840 | 2.362 | ns |
| 3.3-V LVCMOS | 12 mA | GCLK | $t_{CO}$ | 2.876 | 2.876 | 6.419 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.812 | 0.812 | 2.284 | ns |
| 3.3-V LVCMOS | 16 mA | GCLK | $t_{CO}$ | 2.883 | 2.883 | 6.387 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.819 | 0.819 | 2.252 | ns |
| 3.3-V LVCMOS | 20 mA | GCLK | $t_{CO}$ | 2.870 | 2.870 | 6.369 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.806 | 0.806 | 2.234 | ns |
| 3.3-V LVCMOS | 24 mA | GCLK | $t_{CO}$ | 2.859 | 2.859 | 6.347 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.795 | 0.795 | 2.212 | ns |
| 2.5 V | 4 mA | GCLK | $t_{CO}$ | 2.958 | 2.958 | 6.824 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.894 | 0.894 | 2.689 | ns |

**Table 4–75.** EP1AGX90 Column Pins Output Timing Parameters  (Part 2 of 4)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| 2.5 V | 8 mA | GCLK | $t_{CO}$ | 2.906 | 2.906 | 6.562 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.842 | 0.842 | 2.427 | ns |
| 2.5 V | 12 mA | GCLK | $t_{CO}$ | 2.885 | 2.885 | 6.445 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.821 | 0.821 | 2.310 | ns |
| 2.5 V | 16 mA | GCLK | $t_{CO}$ | 2.867 | 2.867 | 6.371 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.803 | 0.803 | 2.236 | ns |
| 1.8 V | 2 mA | GCLK | $t_{CO}$ | 2.998 | 2.998 | 7.816 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.934 | 0.934 | 3.681 | ns |
| 1.8 V | 4 mA | GCLK | $t_{CO}$ | 3.003 | 3.003 | 7.042 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.939 | 0.939 | 2.907 | ns |
| 1.8 V | 6 mA | GCLK | $t_{CO}$ | 2.927 | 2.927 | 6.778 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.863 | 0.863 | 2.643 | ns |
| 1.8 V | 8 mA | GCLK | $t_{CO}$ | 2.929 | 2.929 | 6.687 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.865 | 0.865 | 2.552 | ns |
| 1.8 V | 10 mA | GCLK | $t_{CO}$ | 2.883 | 2.883 | 6.610 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.819 | 0.819 | 2.475 | ns |
| 1.8 V | 12 mA | GCLK | $t_{CO}$ | 2.884 | 2.884 | 6.553 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.820 | 0.820 | 2.418 | ns |
| 1.5 V | 2 mA | GCLK | $t_{CO}$ | 2.978 | 2.978 | 7.346 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.914 | 0.914 | 3.211 | ns |
| 1.5 V | 4 mA | GCLK | $t_{CO}$ | 2.914 | 2.914 | 6.777 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.850 | 0.850 | 2.642 | ns |
| 1.5 V | 6 mA | GCLK | $t_{CO}$ | 2.917 | 2.917 | 6.659 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.853 | 0.853 | 2.524 | ns |
| 1.5 V | 8 mA | GCLK | $t_{CO}$ | 2.876 | 2.876 | 6.606 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.812 | 0.812 | 2.471 | ns |
| SSTL-2 CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.859 | 2.859 | 6.381 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.797 | 0.797 | 2.250 | ns |
| SSTL-2 CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.842 | 2.842 | 6.331 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.780 | 0.780 | 2.200 | ns |
| SSTL-2 CLASS II | 16 mA | GCLK | $t_{CO}$ | 2.820 | 2.820 | 6.258 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.758 | 0.758 | 2.127 | ns |
| SSTL-2 CLASS II | 20 mA | GCLK | $t_{CO}$ | 2.821 | 2.821 | 6.245 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.759 | 0.759 | 2.114 | ns |
| SSTL-2 CLASS II | 24 mA | GCLK | $t_{CO}$ | 2.817 | 2.817 | 6.243 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.755 | 0.755 | 2.112 | ns |
| SSTL-18 CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.858 | 2.858 | 6.356 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.794 | 0.794 | 2.221 | ns |

**Table 4–75.** EP1AGX90 Column Pins Output Timing Parameters  (Part 3 of 4)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Corner | | −6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| SSTL-18 CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.860 | 2.860 | 6.313 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.798 | 0.798 | 2.182 | ns |
| SSTL-18 CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.839 | 2.839 | 6.294 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.777 | 0.777 | 2.163 | ns |
| SSTL-18 CLASS I | 10 mA | GCLK | $t_{CO}$ | 2.844 | 2.844 | 6.292 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.782 | 0.782 | 2.161 | ns |
| SSTL-18 CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.838 | 2.838 | 6.278 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.776 | 0.776 | 2.147 | ns |
| SSTL-18 CLASS II | 8 mA | GCLK | $t_{CO}$ | 2.827 | 2.827 | 6.244 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.765 | 0.765 | 2.113 | ns |
| SSTL-18 CLASS II | 16 mA | GCLK | $t_{CO}$ | 2.839 | 2.839 | 6.222 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.777 | 0.777 | 2.091 | ns |
| SSTL-18 CLASS II | 18 mA | GCLK | $t_{CO}$ | 2.835 | 2.835 | 6.230 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.773 | 0.773 | 2.099 | ns |
| SSTL-18 CLASS II | 20 mA | GCLK | $t_{CO}$ | 2.835 | 2.835 | 6.228 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.773 | 0.773 | 2.097 | ns |
| 1.8-V HSTL CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.861 | 2.861 | 6.287 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.797 | 0.797 | 2.152 | ns |
| 1.8-V HSTL CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.864 | 2.864 | 6.268 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.802 | 0.802 | 2.137 | ns |
| 1.8-V HSTL CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.842 | 2.842 | 6.257 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.780 | 0.780 | 2.126 | ns |
| 1.8-V HSTL CLASS I | 10 mA | GCLK | $t_{CO}$ | 2.846 | 2.846 | 6.263 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.784 | 0.784 | 2.132 | ns |
| 1.8-V HSTL CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.838 | 2.838 | 6.256 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.776 | 0.776 | 2.125 | ns |
| 1.8-V HSTL CLASS II | 16 mA | GCLK | $t_{CO}$ | 2.821 | 2.821 | 6.020 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.759 | 0.759 | 1.889 | ns |
| 1.8-V HSTL CLASS II | 18 mA | GCLK | $t_{CO}$ | 2.823 | 2.823 | 6.031 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.761 | 0.761 | 1.900 | ns |
| 1.8-V HSTL CLASS II | 20 mA | GCLK | $t_{CO}$ | 2.823 | 2.823 | 6.040 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.761 | 0.761 | 1.909 | ns |
| 1.5-V HSTL CLASS I | 4 mA | GCLK | $t_{CO}$ | 2.861 | 2.861 | 6.286 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.797 | 0.797 | 2.151 | ns |
| 1.5-V HSTL CLASS I | 6 mA | GCLK | $t_{CO}$ | 2.863 | 2.863 | 6.260 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.801 | 0.801 | 2.129 | ns |
| 1.5-V HSTL CLASS I | 8 mA | GCLK | $t_{CO}$ | 2.845 | 2.845 | 6.262 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.783 | 0.783 | 2.131 | ns |

**Table 4–75.** EP1AGX90 Column Pins Output Timing Parameters (Part 4 of 4)

| I/O Standard | Drive Strength | Clock | Parameter | Fast Corner | | –6 Speed Grade | Units |
|---|---|---|---|---|---|---|---|
| | | | | Industrial | Commercial | | |
| 1.5-V HSTL CLASS I | 10 mA | GCLK | $t_{CO}$ | 2.845 | 2.845 | 6.264 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.783 | 0.783 | 2.133 | ns |
| 1.5-V HSTL CLASS I | 12 mA | GCLK | $t_{CO}$ | 2.839 | 2.839 | 6.262 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.777 | 0.777 | 2.131 | ns |
| 1.5-V HSTL CLASS II | 16 mA | GCLK | $t_{CO}$ | 2.826 | 2.826 | 6.074 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.764 | 0.764 | 1.943 | ns |
| 1.5-V HSTL CLASS II | 18 mA | GCLK | $t_{CO}$ | 2.829 | 2.829 | 6.084 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.767 | 0.767 | 1.953 | ns |
| 1.5-V HSTL CLASS II | 20 mA | GCLK | $t_{CO}$ | 2.831 | 2.831 | 6.097 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.769 | 0.769 | 1.966 | ns |
| 3.3-V PCI | — | GCLK | $t_{CO}$ | 2.987 | 2.987 | 6.414 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.923 | 0.923 | 2.279 | ns |
| 3.3-V PCI-X | — | GCLK | $t_{CO}$ | 2.987 | 2.987 | 6.414 | ns |
| | | GCLK PLL | $t_{CO}$ | 0.923 | 0.923 | 2.279 | ns |
| LVDS | — | GCLK | $t_{CO}$ | 3.835 | 3.835 | 7.541 | ns |
| | | GCLK PLL | $t_{CO}$ | 1.769 | 1.769 | 3.404 | ns |

Table 4–76 through Table 4–77 list the EP1AGX90 regional clock (RCLK) adder values that should be added to the GCLK values. These adder values are used to determine I/O timing when the I/O pin is driven using the regional clock. This applies for all I/O standards supported by Arria GX with general purpose I/O pins.

Table 4–76 lists row pin delay adders when using the regional clock in Arria GX devices.

**Table 4–76.** EP1AGX90 Row Pin Delay Adders for Regional Clock

| Parameter | Fast Corner | | –6 Speed Grade | Units |
|---|---|---|---|---|
| | Industrial | Commercial | | |
| RCLK input adder | 0.175 | 0.175 | 0.418 | ns |
| RCLK PLL input adder | 0.007 | 0.007 | 0.015 | ns |
| RCLK output adder | −0.175 | −0.175 | −0.418 | ns |
| RCLK PLL output adder | −0.007 | −0.007 | −0.015 | ns |

Table 4–77 lists column pin delay adders when using the regional clock in Arria GX devices.

**Table 4–77.** EP1AGX90 Column Pin Delay Adders for Regional Clock

| Parameter | Fast Corner | | –6 Speed Grade | Units |
|-----------|------------|-----------|----------------|-------|
|           | Industrial | Commercial |                |       |
| `RCLK` input adder | 0.138 | 0.138 | 0.354 | ns |
| `RCLK` PLL input adder | −1.697 | −1.697 | −3.607 | ns |
| `RCLK` output adder | −0.138 | −0.138 | −0.353 | ns |
| `RCLK` PLL output adder | 1.966 | 1.966 | 5.188 | ns |

## Dedicated Clock Pin Timing

Table 4–79 through Table 4–98 list clock pin timing for Arria GX devices when the clock is driven by the global clock, regional clock, periphery clock, and a PLL.

Table 4–78 lists Arria GX clock timing parameters.

**Table 4–78.** Arria GX Clock Timing Parameters

| Symbol | Parameter |
|--------|-----------|
| $t_{CIN}$ | Delay from clock pad to I/O input register |
| $t_{COUT}$ | Delay from clock pad to I/O output register |
| $t_{PLLCIN}$ | Delay from PLL `inclk` pad to I/O input register |
| $t_{PLLCOUT}$ | Delay from PLL `inclk` pad to I/O output register |

### EP1AGX20 Clock Timing Parameters

Table 4–79 through Table 4–80 list the `GCLK` clock timing parameters for EP1AGX20 devices.

Table 4–79 lists clock timing specifications.

**Table 4–79.** EP1AGX20 Row Pins Global Clock Timing Parameters

| Parameter | Fast Model | | –6 Speed Grade | Units |
|-----------|-----------|-----------|----------------|-------|
|           | Industrial | Commercial |                |       |
| `tcin` | 1.394 | 1.394 | 3.161 | ns |
| `tcout` | 1.399 | 1.399 | 3.155 | ns |
| `tpllcin` | −0.027 | −0.027 | 0.091 | ns |
| `tpllcout` | −0.022 | −0.022 | 0.085 | ns |

Table 4–80 lists clock timing specifications.

**Table 4–80.** EP1AGX20 Row Pins Global Clock Timing Parameters

| Parameter | Fast Model | | –6 Speed Grade | Units |
|-----------|------------|------------|----------------|-------|
| | **Industrial** | **Commercial** | | |
| $t_{CIN}$ | 1.655 | 1.655 | 3.726 | ns |
| $t_{COUT}$ | 1.655 | 1.655 | 3.726 | ns |
| $t_{PLLCIN}$ | 0.236 | 0.236 | 0.655 | ns |
| $t_{PLLCOUT}$ | 0.236 | 0.236 | 0.655 | ns |

Table 4–81 through Table 4–82 list the RCLK clock timing parameters for EP1AGX20 devices.

Table 4–81 lists clock timing specifications.

**Table 4–81.** EP1AGX20 Row Pins Regional Clock Timing Parameters

| Parameter | Fast Model | | –6 Speed Grade | Units |
|-----------|------------|------------|----------------|-------|
| | **Industrial** | **Commercial** | | |
| $t_{CIN}$ | 1.283 | 1.283 | 2.901 | ns |
| $t_{COUT}$ | 1.288 | 1.288 | 2.895 | ns |
| $t_{PLLCIN}$ | −0.034 | −0.034 | 0.077 | ns |
| $t_{PLLCOUT}$ | −0.029 | −0.029 | 0.071 | ns |

Table 4–82 lists clock timing specifications.

**Table 4–82.** EP1AGX20 Row Pins Regional Clock Timing Parameters

| Parameter | Fast Model | | –6 Speed Grade | Units |
|-----------|------------|------------|----------------|-------|
| | **Industrial** | **Commercial** | | |
| $t_{CIN}$ | 1.569 | 1.569 | 3.487 | ns |
| $t_{COUT}$ | 1.569 | 1.569 | 3.487 | ns |
| $t_{PLLCIN}$ | 0.278 | 0.278 | 0.706 | ns |
| $t_{PLLCOUT}$ | 0.278 | 0.278 | 0.706 | ns |

## EP1AGX35 Clock Timing Parameters

Table 4–83 through Table 4–84 list the GCLK clock timing parameters for EP1AGX35 devices.

Table 4–83 lists clock timing specifications.

**Table 4–83.** EP1AGX35 Row Pins Global Clock Timing Parameters (Part 1 of 2)

| Parameter | Fast Model | | –6 Speed Grade | Units |
|-----------|------------|------------|----------------|-------|
| | **Industrial** | **Commercial** | | |
| $t_{CIN}$ | 1.394 | 1.394 | 3.161 | ns |
| $t_{COUT}$ | 1.399 | 1.399 | 3.155 | ns |

**Table 4–83.** EP1AGX35 Row Pins Global Clock Timing Parameters  (Part 2 of 2)

| Parameter | Fast Model | | −6 Speed Grade | Units |
|---|---|---|---|---|
| | Industrial | Commercial | | |
| $t_{PLLCIN}$ | −0.027 | −0.027 | 0.091 | ns |
| $t_{PLLCOUT}$ | −0.022 | −0.022 | 0.085 | ns |

Table 4–84 lists clock timing specifications.

**Table 4–84.** EP1AGX35 Row Pins Global Clock Timing Parameters

| Parameter | Fast Model | | −6 Speed Grade | Units |
|---|---|---|---|---|
| | Industrial | Commercial | | |
| $t_{CIN}$ | 1.655 | 1.655 | 3.726 | ns |
| $t_{COUT}$ | 1.655 | 1.655 | 3.726 | ns |
| $t_{PLLCIN}$ | 0.236 | 0.236 | 0.655 | ns |
| $t_{PLLCOUT}$ | 0.236 | 0.236 | 0.655 | ns |

Table 4–85 through Table 4–86 list the RCLK clock timing parameters for EP1AGX35 devices.

Table 4–85 lists clock timing specifications.

**Table 4–85.** EP1AGX35 Row Pins Regional Clock Timing Parameters

| Parameter | Fast Model | | −6 Speed Grade | Units |
|---|---|---|---|---|
| | Industrial | Commercial | | |
| $t_{CIN}$ | 1.283 | 1.283 | 2.901 | ns |
| $t_{COUT}$ | 1.288 | 1.288 | 2.895 | ns |
| $t_{PLLCIN}$ | −0.034 | −0.034 | 0.077 | ns |
| $t_{PLLCOUT}$ | −0.029 | −0.029 | 0.071 | ns |

Table 4–86 lists clock timing specifications.

**Table 4–86.** EP1AGX35 Row Pins Regional Clock Timing Parameters

| Parameter | Fast Model | | −6 Speed Grade | Units |
|---|---|---|---|---|
| | Industrial | Commercial | | |
| $t_{CIN}$ | 1.569 | 1.569 | 3.487 | ns |
| $t_{COUT}$ | 1.569 | 1.569 | 3.487 | ns |
| $t_{PLLCIN}$ | 0.278 | 0.278 | 0.706 | ns |
| $t_{PLLCOUT}$ | 0.278 | 0.278 | 0.706 | ns |

### EP1AGX50 Clock Timing Parameters

Table 4–87 through Table 4–88 list the GCLK clock timing parameters for EP1AGX50 devices.

Table 4–87 lists clock timing specifications.

**Table 4–87.** EP1AGX50 Row Pins Global Clock Timing Parameters

| Parameter | Fast Model | | −6 Speed Grade | Units |
|-----------|------------|------------|----------------|-------|
|           | Industrial | Commercial |                |       |
| $t_{CIN}$ | 1.529 | 1.529 | 3.587 | ns |
| $t_{COUT}$ | 1.534 | 1.534 | 3.581 | ns |
| $t_{PLLCIN}$ | −0.024 | −0.024 | 0.181 | ns |
| $t_{PLLCOUT}$ | −0.019 | −0.019 | 0.175 | ns |

Table 4–88 lists clock timing specifications.

**Table 4–88.** EP1AGX50 Row Pins Global Clock Timing Parameters

| Parameter | Fast Model | | −6 Speed Grade | Units |
|-----------|------------|------------|----------------|-------|
|           | Industrial | Commercial |                |       |
| $t_{CIN}$ | 1.793 | 1.793 | 4.165 | ns |
| $t_{COUT}$ | 1.793 | 1.793 | 4.165 | ns |
| $t_{PLLCIN}$ | 0.238 | 0.238 | 0.758 | ns |
| $t_{PLLCOUT}$ | 0.238 | 0.238 | 0.758 | ns |

Table 4–89 through Table 4–90 list the RCLK clock timing parameters for EP1AGX50 devices.

Table 4–89 lists clock timing specifications.

**Table 4–89.** EP1AGX50 Row Pins Regional Clock Timing Parameters

| Parameter | Fast Model | | −6 Speed Grade | Units |
|-----------|------------|------------|----------------|-------|
|           | Industrial | Commercial |                |       |
| $t_{CIN}$ | 1.396 | 1.396 | 3.287 | ns |
| $t_{COUT}$ | 1.401 | 1.401 | 3.281 | ns |
| $t_{PLLCIN}$ | −0.017 | −0.017 | 0.195 | ns |
| $t_{PLLCOUT}$ | −0.012 | −0.012 | 0.189 | ns |

Table 4–90 lists clock timing specifications.

**Table 4–90.** EP1AGX50 Row Pins Regional Clock Timing Parameters

| Parameter | Fast Model | | −6 Speed Grade | Units |
|---|---|---|---|---|
| | Industrial | Commercial | | |
| $t_{CIN}$ | 1.653 | 1.653 | 3.841 | ns |
| $t_{COUT}$ | 1.651 | 1.651 | 3.839 | ns |
| $t_{PLLCIN}$ | 0.245 | 0.245 | 0.755 | ns |
| $t_{PLLCOUT}$ | 0.245 | 0.245 | 0.755 | ns |

## EP1AGX60 Clock Timing Parameters

Table 4–91 to Table 4–92 on page 4–82 list the GCLK clock timing parameters for EP1AGX60 devices.

Table 4–91 lists clock timing specifications.

**Table 4–91.** EP1AGX60 Row Pins Global Clock Timing Parameters

| Parameter | Fast Model | | −6 Speed Grade | Units |
|---|---|---|---|---|
| | Industrial | Commercial | | |
| $t_{CIN}$ | 1.531 | 1.531 | 3.593 | ns |
| $t_{COUT}$ | 1.536 | 1.536 | 3.587 | ns |
| $t_{PLLCIN}$ | −0.023 | −0.023 | 0.188 | ns |
| $t_{PLLCOUT}$ | −0.018 | −0.018 | 0.182 | ns |

Table 4–92 lists clock timing specifications.

**Table 4–92.** EP1AGX60 Row Pins Global Clock Timing Parameters

| Parameter | Fast Model | | −6 Speed Grade | Units |
|---|---|---|---|---|
| | Industrial | Commercial | | |
| $t_{CIN}$ | 1.792 | 1.792 | 4.165 | ns |
| $t_{COUT}$ | 1.792 | 1.792 | 4.165 | ns |
| $t_{PLLCIN}$ | 0.238 | 0.238 | 0.758 | ns |
| $t_{PLLCOUT}$ | 0.238 | 0.238 | 0.758 | ns |

Table 4–93 through Table 4–94 list the RCLK clock timing parameters for EP1AGX60 devices.

Table 4–93 lists clock timing specifications.

**Table 4–93.** EP1AGX60 Row Pins Regional Clock Timing Parameters

| Parameter | Fast Model | | −6 Speed Grade | Units |
|---|---|---|---|---|
| | Industrial | Commercial | | |
| $t_{CIN}$ | 1.382 | 1.382 | 3.268 | ns |
| $t_{COUT}$ | 1.387 | 1.387 | 3.262 | ns |
| $t_{PLLCIN}$ | −0.031 | −0.031 | 0.174 | ns |
| $t_{PLLCOUT}$ | −0.026 | −0.026 | 0.168 | ns |

Table 4–94 lists clock timing specifications.

**Table 4–94.** EP1AGX60 Row Pins Regional Clock Timing Parameters

| Parameter | Fast Model | | −6 Speed Grade | Units |
|---|---|---|---|---|
| | Industrial | Commercial | | |
| $t_{CIN}$ | 1.649 | 1.649 | 3.835 | ns |
| $t_{COUT}$ | 1.651 | 1.651 | 3.839 | ns |
| $t_{PLLCIN}$ | 0.245 | 0.245 | 0.755 | ns |
| $t_{PLLCOUT}$ | 0.245 | 0.245 | 0.755 | ns |

## EP1AGX90 Clock Timing Parameters

Table 4–95 through Table 4–96 list the GCLK clock timing parameters for EP1AGX90 devices.

Table 4–95 lists clock timing specifications.

**Table 4–95.** EP1AGX90 Row Pins Global Clock Timing Parameters

| Parameter | Fast Model | | −6 Speed Grade | Units |
|---|---|---|---|---|
| | Industrial | Commercial | | |
| $t_{CIN}$ | 1.630 | 1.630 | 3.799 | ns |
| $t_{COUT}$ | 1.635 | 1.635 | 3.793 | ns |
| $t_{PLLCIN}$ | −0.422 | −0.422 | −0.310 | ns |
| $t_{PLLCOUT}$ | −0.417 | −0.417 | −0.316 | ns |

Table 4–96 lists clock timing specifications.

**Table 4–96.** EP1AGX90 Row Pins Global Clock Timing Parameters

| Parameter | Fast Model | | –6 Speed Grade | Units |
|---|---|---|---|---|
| | Industrial | Commercial | | |
| $t_{CIN}$ | 1.904 | 1.904 | 4.376 | ns |
| $t_{COUT}$ | 1.904 | 1.904 | 4.376 | ns |
| $t_{PLLCIN}$ | −0.153 | −0.153 | 0.254 | ns |
| $t_{PLLCOUT}$ | −0.153 | −0.153 | 0.254 | ns |

Table 4–97 through Table 4–98 list the RCLK clock timing parameters for EP1AGX90 devices.

Table 4–97 lists clock timing specifications.

**Table 4–97.** EP1AGX90 Row Pins Regional Clock Timing Parameters

| Parameter | Fast Model | | –6 Speed Grade | Units |
|---|---|---|---|---|
| | Industrial | Commercial | | |
| $t_{CIN}$ | 1.462 | 1.462 | 3.407 | ns |
| $t_{COUT}$ | 1.467 | 1.467 | 3.401 | ns |
| $t_{PLLCIN}$ | −0.430 | −0.430 | −0.322 | ns |
| $t_{PLLCOUT}$ | −0.425 | −0.425 | −0.328 | ns |

Table 4–98 lists clock timing specifications.

**Table 4–98.** EP1AGX90 Row Pins Regional Clock Timing Parameters

| Parameter | Fast Model | | –6 Speed Grade | Units |
|---|---|---|---|---|
| | Industrial | Commercial | | |
| $t_{CIN}$ | 1.760 | 1.760 | 4.011 | ns |
| $t_{COUT}$ | 1.760 | 1.760 | 4.011 | ns |
| $t_{PLLCIN}$ | −0.118 | −0.118 | 0.303 | ns |
| $t_{PLLCOUT}$ | −0.118 | −0.118 | 0.303 | ns |

# Block Performance

Table 4–99 shows the Arria GX performance for some common designs. All performance values were obtained with the Quartus II software compilation of library of parameterized modules (LPM) or MegaCore functions for finite impulse response (FIR) and fast Fourier transform (FFT) designs.

Table 4–99 lists performance notes.

**Table 4–99.** Arria GX Performance Notes

| Applications | | Resources Used | | | Performance |
|---|---|---|---|---|---|
| | | ALUTs | TriMatrix Memory Blocks | DSP Blocks | –6 Speed Grade |
| LE | 16-to-1 multiplexer | 5 | 0 | 0 | 168.41 |
| | 32-to-1 multiplexer | 11 | 0 | 0 | 334.11 |
| | 16-bit counter | 16 | 0 | 0 | 374.0 |
| | 64-bit counter | 64 | 0 | 0 | 168.41 |
| TriMatrix Memory M512 block | Simple dual-port RAM 32 x 18 bit | 0 | 1 | 0 | 348.0 |
| | FIFO 32 x 18 bit | 0 | 1 | 0 | 333.22 |
| TriMatrix Memory M4K block | Simple dual-port RAM 128 x 36 bit | 0 | 1 | 0 | 344.71 |
| | True dual-port RAM 128 x 18 bit | 0 | 1 | 0 | 348.0 |
| TriMatrix Memory MegaRAM block | Single port RAM 4K x 144 bit | 0 | 2 | 0 | 244.0 |
| | Simple dual-port RAM 4K x 144 bit | 0 | 1 | 0 | 292.0 |
| | True dual-port RAM 4K x 144 bit | 0 | 2 | 0 | 244.0 |
| | Single port RAM 8K x 72 bit | 0 | 1 | 0 | 247.0 |
| | Simple dual-port RAM 8K x 72 bit | 0 | 1 | 0 | 292.0 |
| | Single port RAM 16K x 36 bit | 0 | 1 | 0 | 254.0 |
| | Simple dual-port RAM 16K x 36 bit | 0 | 1 | 0 | 292.0 |
| | True dual-port RAM 16K x 36 bit | 0 | 1 | 0 | 251.0 |
| | Single port RAM 32K x 18 bit | 0 | 1 | 0 | 317.36 |
| | Simple dual-port RAM 32K x 18 bit | 0 | 1 | 0 | 292.0 |
| | True dual-port RAM 32K x 18 bit | 0 | 1 | 0 | 251.0 |
| | Single port RAM 64K x 9 bit | 0 | 1 | 0 | 254.0 |
| | Simple dual-port RAM 64K x 9 bit | 0 | 1 | 0 | 292.0 |
| | True dual-port RAM 64K x 9 bit | 0 | 1 | 0 | 251.0 |

**Table 4–99.** Arria GX Performance Notes

| Applications | | Resources Used | | | Performance |
|---|---|---|---|---|---|
| | | ALUTs | TriMatrix Memory Blocks | DSP Blocks | –6 Speed Grade |
| DSP block | 9 x 9-bit multiplier | 0 | 0 | 1 | 335.35 |
| | 18 x 18-bit multiplier | 0 | 0 | 2 | 285.0 |
| | 18 x 18-bit multiplier | 0 | 0 | 4 | 335.35 |
| | 36 x 36-bit multiplier | 0 | 0 | 8 | 174.4 |
| | 36 x 36-bit multiplier | 0 | 0 | 8 | 285.0 |
| | 18-bit 4-tap FIR filter | 0 | 0 | 8 | 163.0 |
| Larger Designs | 8-bit 16-tap parallel FIR filter | 0 | 0 | 4 | 163.0 |

# IOE Programmable Delay

For IOE programmable delay, refer to Table 4–100 through Table 4–101.

Table 4–100 lists IOE programmable delays.

**Table 4–100.** Arria GX IOE Programmable Delay on Row Pins

| Parameter | Paths Affected | Available Settings | Fast Model | | | | –6 Speed Grade | | Units |
|---|---|---|---|---|---|---|---|---|---|
| | | | Industrial | | Commercial | | | | |
| | | | Min Offset | Max Offset | Min Offset | Max Offset | Min Offset | Max Offset | |
| Input delay from pin to internal cells | Pad to I/O dataout to core | 8 | 0 | 1.782 | 0 | 1.782 | 0 | 4.124 | ns |
| Input delay from pin to input register | Pad to I/O input register | 64 | 0 | 2.054 | 0 | 2.054 | 0 | 4.689 | ns |
| Delay from output register to output pin | I/O output register to pad | 2 | 0 | 0.332 | 0 | 0.332 | 0 | 0.717 | ns |
| Output enable pin delay | txz/tzx | 2 | 0 | 0.32 | 0 | 0.32 | 0 | 0.693 | ns |

Table 4–101 lists IOE programmable delays.

**Table 4–101.** Arria GX IOE Programmable Delay on Column Pins

| Parameter | Paths Affected | Available Settings | Fast Model | | | | –6 Speed Grade | | Units |
| | | | Industrial | | Commercial | | | | |
| | | | Min Offset | Max Offset | Min Offset | Max Offset | Min Offset | Max Offset | |
|---|---|---|---|---|---|---|---|---|---|
| Input delay from pin to internal cells | Pad to I/O dataout to core | 8 | 0 | 1.781 | 0 | 1.781 | 0 | 4.132 | ns |
| Input delay from pin to input register | Pad to I/O input register | 64 | 0 | 2.053 | 0 | 2.053 | 0 | 4.697 | ns |
| Delay from output register to output pin | I/O output register to pad | 2 | 0 | 0.332 | 0 | 0.332 | 0 | 0.717 | ns |
| Output enable pin delay | txz/tzx | 2 | 0 | 0.32 | 0 | 0.32 | 0 | 0.693 | ns |

# Maximum Input and Output Clock Toggle Rate

Maximum clock toggle rate is defined as the maximum frequency achievable for a clock type signal at an I/O pin. The I/O pin can be a regular I/O pin or a dedicated clock I/O pin.

The maximum clock toggle rate is different from the maximum data bit rate. If the maximum clock toggle rate on a regular I/O pin is 300 MHz, the maximum data bit rate for dual data rate (DDR) could be potentially as high as 600 Mbps on the same I/O pin.

Table 4–105, Table 4–106, and Table 4–107 provide output toggle rates at the default capacitive loading. Use the Quartus II software to obtain output toggle rates at loads different from the default capacitive loading.

Table 4–102 shows the maximum input clock toggle rates for Arria GX device column I/O pins.

**Table 4–102.** Arria GX Maximum Input Toggle Rate for Column I/O Pins

| I/O Standards | –6 Speed Grade | Units |
|---|---|---|
| 3.3-V LVTTL | 420 | MHz |
| 3.3-V LVCMOS | 420 | MHz |
| 2.5 V | 420 | MHz |
| 1.8 V | 420 | MHz |
| 1.5 V | 420 | MHz |
| SSTL-2 CLASS I | 467 | MHz |
| SSTL-2 CLASS II | 467 | MHz |
| SSTL-18 CLASS I | 467 | MHz |

**Table 4–102.**  Arria GX Maximum Input Toggle Rate for Column I/O Pins

| I/O Standards | –6 Speed Grade | Units |
|---|---|---|
| SSTL-18 CLASS II | 467 | MHz |
| 1.8-V HSTL CLASS I | 467 | MHz |
| 1.8-V HSTL CLASS II | 467 | MHz |
| 1.5-V HSTL CLASS I | 467 | MHz |
| 1.5-V HSTL CLASS II | 467 | MHz |
| 3.3-V PCI | 420 | MHz |
| 3.3-V PCI-X | 420 | MHz |

Table 4–103 shows the maximum input clock toggle rates for Arria GX device row I/O pins.

**Table 4–103.**  Arria GX Maximum Input Toggle Rate for Row I/O Pins

| I/O Standards | –6 Speed Grade | Units |
|---|---|---|
| 3.3-V LVTTL | 420 | MHz |
| 3.3-V LVCMOS | 420 | MHz |
| 2.5 V | 420 | MHz |
| 1.8 V | 420 | MHz |
| 1.5 V | 420 | MHz |
| SSTL-2 CLASS I | 467 | MHz |
| SSTL-2 CLASS II | 467 | MHz |
| SSTL-18 CLASS I | 467 | MHz |
| SSTL-18 CLASS II | 467 | MHz |
| 1.8-V HSTL CLASS I | 467 | MHz |
| 1.8-V HSTL CLASS II | 467 | MHz |
| 1.5-V HSTL CLASS I | 467 | MHz |
| 1.5-V HSTL CLASS II | 467 | MHz |
| LVDS | 392 | MHz |

Table 4–104 shows the maximum input clock toggle rates for Arria GX device dedicated clock pins.

**Table 4–104.**  Arria GX Maximum Input Clock Rate for Dedicated Clock Pins   (Part 1 of 2)

| I/O Standards | –6 Speed Grade | Units |
|---|---|---|
| 3.3-V LVTTL | 373 | MHz |
| 3.3-V LVCMOS | 373 | MHz |
| 2.5 V | 373 | MHz |
| 1.8 V | 373 | MHz |
| 1.5 V | 373 | MHz |
| SSTL-2 CLASS I | 467 | MHz |
| SSTL-2 CLASS II | 467 | MHz |
| 3.3-V PCI | 373 | MHz |

**Table 4–104.** Arria GX Maximum Input Clock Rate for Dedicated Clock Pins   (Part 2 of 2)

| I/O Standards | –6 Speed Grade | Units |
|---|---|---|
| 3.3-V PCI-X | 373 | MHz |
| SSTL-18 CLASS I | 467 | MHz |
| SSTL-18 CLASS II | 467 | MHz |
| 1.8-V HSTL CLASS I | 467 | MHz |
| 1.8-V HSTL CLASS II | 467 | MHz |
| 1.5-V HSTL CLASS I | 467 | MHz |
| 1.5-V HSTL CLASS II | 467 | MHz |
| 1.2-V HSTL | 233 | MHz |
| DIFFERENTAL SSTL-2 | 467 | MHz |
| DIFFERENTIAL 2.5-V SSTL CLASS II | 467 | MHz |
| DIFFERENTIAL 1.8-V SSTL CLASS I | 467 | MHz |
| DIFFERENTIAL 1.8-V SSTL CLASS II | 467 | MHz |
| DIFFERENTIAL 1.8-V HSTL CLASS I | 467 | MHz |
| DIFFERENTIAL 1.8-V HSTL CLASS II | 467 | MHz |
| DIFFERENTIAL 1.5-V HSTL CLASS I | 467 | MHz |
| DIFFERENTIAL 1.5-V HSTL CLASS II | 467 | MHz |
| DIFFERENTIAL 1.2-V HSTL | 233 | MHz |
| LVDS | 640 | MHz |
| LVDS *(1)* | 373 | MHz |

**Note to Table 4–104:**

(1) This set of numbers refers to the VIO dedicated input clock pins.

Table 4–105 shows the maximum output clock toggle rates for Arria GX device column I/O pins.

**Table 4–105.** Arria GX Maximum Output Toggle Rate for Column I/O Pins  (Part 1 of 3)

| I/O Standards | Drive Strength | –6 Speed Grade | Units |
|---|---|---|---|
| 3.3-V LVTTL | 4 mA | 196 | MHz |
| | 8 mA | 303 | MHz |
| | 12 mA | 393 | MHz |
| | 16 mA | 486 | MHz |
| | 20 mA | 570 | MHz |
| | 24 mA | 626 | MHz |

**Table 4–105.** Arria GX Maximum Output Toggle Rate for Column I/O Pins  (Part 2 of 3)

| I/O Standards | Drive Strength | –6 Speed Grade | Units |
|---|---|---|---|
| 3.3-V LVCMOS | 4 mA | 215 | MHz |
| | 8 mA | 411 | MHz |
| | 12 mA | 626 | MHz |
| | 16 mA | 819 | MHz |
| | 20 mA | 874 | MHz |
| | 24 mA | 934 | MHz |
| 2.5 V | 4 mA | 168 | MHz |
| | 8 mA | 355 | MHz |
| | 12 mA | 514 | MHz |
| | 16 mA | 766 | MHz |
| 1.8 V | 2 mA | 97 | MHz |
| | 4 mA | 215 | MHz |
| | 6 mA | 336 | MHz |
| | 8 mA | 486 | MHz |
| | 10 mA | 706 | MHz |
| | 12 mA | 925 | MHz |
| 1.5 V | 2 mA | 168 | MHz |
| | 4 mA | 303 | MHz |
| | 6 mA | 350 | MHz |
| | 8 mA | 392 | MHz |
| SSTL-2 CLASS I | 8 mA | 280 | MHz |
| | 12 mA | 327 | MHz |
| SSTL-2 CLASS II | 16 mA | 280 | MHz |
| | 20 mA | 327 | MHz |
| | 24 mA | 327 | MHz |
| SSTL-18 CLASS I | 4 mA | 140 | MHz |
| | 6 mA | 186 | MHz |
| | 8 mA | 280 | MHz |
| | 10 mA | 373 | MHz |
| | 12 mA | 373 | MHz |
| SSTL-18 CLASS II | 8 mA | 140 | MHz |
| | 16 mA | 327 | MHz |
| | 18 mA | 373 | MHz |
| | 20 mA | 420 | MHz |
| 1.8-V HSTL CLASS I | 4 mA | 280 | MHz |
| | 6 mA | 420 | MHz |
| | 8 mA | 561 | MHz |
| | 10 mA | 561 | MHz |
| | 12 mA | 607 | MHz |

**Table 4–105.** Arria GX Maximum Output Toggle Rate for Column I/O Pins  (Part 3 of 3)

| I/O Standards | Drive Strength | –6 Speed Grade | Units |
|---|---|---|---|
| 1.8-V HSTL CLASS II | 16 mA | 420 | MHz |
| | 18 mA | 467 | MHz |
| | 20 mA | 514 | MHz |
| 1.5-V HSTL CLASS I | 4 mA | 280 | MHz |
| | 6 mA | 420 | MHz |
| | 8 mA | 561 | MHz |
| | 10 mA | 607 | MHz |
| | 12 mA | 654 | MHz |
| 1.5-V HSTL CLASS II | 16 mA | 514 | MHz |
| | 18 mA | 561 | MHz |
| | 20 mA | 561 | MHz |
| 3.3-V PCI | — | 626 | MHz |
| 3.3-V PCI-X | — | 626 | MHz |

Table 4–106 shows the maximum output clock toggle rates for Arria GX device row I/O pins.

**Table 4–106.** Arria GX Maximum Output Toggle Rate for Row I/O Pins

| I/O Standards | Drive Strength | –6 Speed Grade | Units |
|---|---|---|---|
| 3.3-V LVTTL | 4 mA | 196 | MHz |
| | 8 mA | 303 | MHz |
| | 12 mA | 393 | MHz |
| 3.3-V LVCMOS | 4 mA | 215 | MHz |
| | 8 mA | 411 | MHz |
| 2.5 V | 4 mA | 168 | MHz |
| | 8 mA | 355 | MHz |
| | 12 mA | 514 | MHz |
| 1.8 V | 2 mA | 97 | MHz |
| | 4 mA | 215 | MHz |
| | 6 mA | 336 | MHz |
| | 8 mA | 486 | MHz |
| 1.5 V | 2 mA | 168 | MHz |
| | 4 mA | 303 | MHz |
| SSTL-2 CLASS I | 8 mA | 280 | MHz |
| | 12 mA | 327 | MHz |
| SSTL-2 CLASS II | 16 mA | 280 | MHz |
| SSTL-18 CLASS I | 4 mA | 140 | MHz |
| | 6 mA | 186 | MHz |
| | 8 mA | 280 | MHz |
| | 10 mA | 373 | MHz |

**Table 4–106.** Arria GX Maximum Output Toggle Rate for Row I/O Pins

| I/O Standards | Drive Strength | –6 Speed Grade | Units |
|---|---|---|---|
| 1.8-V HSTL CLASS I | 4 mA | 280 | MHz |
| | 6 mA | 420 | MHz |
| | 8 mA | 561 | MHz |
| | 10 mA | 561 | MHz |
| | 12 mA | 607 | MHz |
| 1.5-V HSTL CLASS I | 4 mA | 280 | MHz |
| | 6 mA | 420 | MHz |
| | 8 mA | 561 | MHz |
| LVDS | — | 598 | MHz |

Table 4–107 lists maximum output clock rate for dedicated clock pins.

**Table 4–107.** Arria GX Maximum Output Clock Rate for Dedicated Clock Pins  (Part 1 of 4)

| I/O Standards | Drive Strength | –6 Speed Grade | Units |
|---|---|---|---|
| 3.3-V LVTTL | 4 mA | 196 | MHz |
| | 8 mA | 303 | MHz |
| | 12 mA | 393 | MHz |
| | 16 mA | 486 | MHz |
| | 20 mA | 570 | MHz |
| | 24 mA | 626 | MHz |
| 3.3-V LVCMOS | 4 mA | 215 | MHz |
| | 8 mA | 411 | MHz |
| | 12 mA | 626 | MHz |
| | 16 mA | 819 | MHz |
| | 20 mA | 874 | MHz |
| | 24 mA | 934 | MHz |
| 2.5 V | 4 mA | 168 | MHz |
| | 8 mA | 355 | MHz |
| | 12 mA | 514 | MHz |
| | 16 mA | 766 | MHz |
| 1.8 V | 2 mA | 97 | MHz |
| | 4 mA | 215 | MHz |
| | 6 mA | 336 | MHz |
| | 8 mA | 486 | MHz |
| | 10 mA | 706 | MHz |
| | 12 mA | 925 | MHz |
| 1.5 V | 2 mA | 168 | MHz |
| | 4 mA | 303 | MHz |
| | 6 mA | 350 | MHz |
| | 8 mA | 392 | MHz |

**Table 4–107.** Arria GX Maximum Output Clock Rate for Dedicated Clock Pins  (Part 2 of 4)

| I/O Standards | Drive Strength | –6 Speed Grade | Units |
|---|---|---|---|
| SSTL-2 CLASS I | 8 mA | 280 | MHz |
| | 12 mA | 327 | MHz |
| SSTL-2 CLASS II | 16 mA | 280 | MHz |
| | 20 mA | 327 | MHz |
| | 24 mA | 327 | MHz |
| SSTL-18 CLASS I | 4 mA | 140 | MHz |
| | 6 mA | 186 | MHz |
| | 8 mA | 280 | MHz |
| | 10 mA | 373 | MHz |
| | 12 mA | 373 | MHz |
| SSTL-18 CLASS II | 8 mA | 140 | MHz |
| | 16 mA | 327 | MHz |
| | 18 mA | 373 | MHz |
| | 20 mA | 420 | MHz |
| 1.8-V HSTL CLASS I | 4 mA | 280 | MHz |
| | 6 mA | 420 | MHz |
| | 8 mA | 561 | MHz |
| | 10 mA | 561 | MHz |
| | 12 mA | 607 | MHz |
| 1.8-V HSTL CLASS II | 16 mA | 420 | MHz |
| | 18 mA | 467 | MHz |
| | 20 mA | 514 | MHz |
| 1.5-V HSTL CLASS I | 4 mA | 280 | MHz |
| | 6 mA | 420 | MHz |
| | 8  mA | 561 | MHz |
| | 10mA | 607 | MHz |
| | 12 mA | 654 | MHz |
| 1.5-V HSTL CLASS II | 16 mA | 514 | MHz |
| | 18 mA | 561 | MHz |
| | 20 mA | 561 | MHz |
| | 24 mA | 278 | MHz |
| DIFFERENTIAL SSTL-2 | 8 mA | 280 | MHz |
| | 12 mA | 327 | MHz |
| DIFFERENTIAL 2.5-V SSTL CLASS II | 16 mA | 280 | MHz |
| | 20 mA | 327 | MHz |
| | 24 mA | 327 | MHz |

**Table 4–107.** Arria GX Maximum Output Clock Rate for Dedicated Clock Pins  (Part 3 of 4)

| I/O Standards | Drive Strength | –6 Speed Grade | Units |
|---|---|---|---|
| DIFFERENTIAL 1.8-V SSTL CLASS I | 4 mA | 140 | MHz |
| | 6 mA | 186 | MHz |
| | 8 mA | 280 | MHz |
| | 10 mA | 373 | MHz |
| | 12 mA | 373 | MHz |
| DIFFERENTIAL 1.8-V SSTL CLASS II | 8 mA | 140 | MHz |
| | 16 mA | 327 | MHz |
| | 18 mA | 373 | MHz |
| | 20 mA | 420 | MHz |
| DIFFERENTIAL 1.8-V HSTL CLASS I | 4 mA | 280 | MHz |
| | 6 mA | 420 | MHz |
| | 8 mA | 561 | MHz |
| | 10 mA | 561 | MHz |
| | 12 mA | 607 | MHz |
| DIFFERENTIAL 1.8-V HSTL CLASS II | 16 mA | 420 | MHz |
| | 18 mA | 467 | MHz |
| | 20 mA | 514 | MHz |
| DIFFERENTIAL 1.5-V HSTL CLASS I | 4 mA | 280 | MHz |
| | 6 mA | 420 | MHz |
| | 8 mA | 561 | MHz |
| | 10 mA | 607 | MHz |
| | 12 mA | 654 | MHz |
| DIFFERENTIAL 1.5-V HSTL CLASS II | 16 mA | 514 | MHz |
| | 18 mA | 561 | MHz |
| | 20 mA | 561 | MHz |
| | 24 mA | 278 | MHz |
| 3.3-V PCI | — | 626 | MHz |
| 3.3-V PCI-X | — | 626 | MHz |
| LVDS | — | 280 | MHz |
| HYPERTRANSPORT | — | 116 | MHz |
| LVPECL | — | 280 | MHz |
| 3.3-V LVTTL | SERIES_25_OHMS | 327 | MHz |
| | SERIES_50_OHMS | 327 | MHz |
| 3.3-V LVCMOS | SERIES_25_OHMS | 280 | MHz |
| | SERIES_50_OHMS | 280 | MHz |
| 2.5 V | SERIES_25_OHMS | 280 | MHz |
| | SERIES_50_OHMS | 280 | MHz |
| 1.8 V | SERIES_25_OHMS | 420 | MHz |
| | SERIES_50_OHMS | 420 | MHz |

**Table 4–107.** Arria GX Maximum Output Clock Rate for Dedicated Clock Pins (Part 4 of 4)

| I/O Standards | Drive Strength | –6 Speed Grade | Units |
|---|---|---|---|
| 1.5 V | SERIES_50_OHMS | 373 | MHz |
| SSTL-2 CLASS I | SERIES_50_OHMS | 467 | MHz |
| SSTL-2 CLASS II | SERIES_25_OHMS | 467 | MHz |
| SSTL-18 CLASS I | SERIES_50_OHMS | 327 | MHz |
| SSTL-18 CLASS II | SERIES_25_OHMS | 420 | MHz |
| 1.8-V HSTL CLASS I | SERIES_50_OHMS | 561 | MHz |
| 1.8-V HSTL CLASS II | SERIES_25_OHMS | 420 | MHz |
| 1.5-V HSTL CLASS I | SERIES_50_OHMS | 467 | MHz |
| 1.2-V HSTL | SERIES_50_OHMS | 233 | MHz |
| DIFFERENTIAL SSTL-2 | SERIES_50_OHMS | 467 | MHz |
| DIFFERENTIAL 2.5-V SSTL CLASS II | SERIES_25_OHMS | 467 | MHz |
| DIFFERENTIAL 1.8-V SSTL CLASS I | SERIES_50_OHMS | 327 | MHz |
| DIFFERENTIAL 1.8-V SSTL CLASS II | SERIES_25_OHMS | 420 | MHz |
| DIFFERENTIAL 1.8-V HSTL CLASS I | SERIES_50_OHMS | 561 | MHz |
| DIFFERENTIAL 1.8-V HSTL CLASS II | SERIES_25_OHMS | 420 | MHz |
| DIFFERENTIAL 1.5-V HSTL CLASS I | SERIES_50_OHMS | 467 | MHz |
| DIFFERENTIAL 1.2-V HSTL | SERIES_50_OHMS | 233 | MHz |

# Duty Cycle Distortion

Duty cycle distortion (DCD) describes how much the falling edge of a clock is off from its ideal position. The ideal position is when both the clock high time (CLKH) and the clock low time (CLKL) equal half of the clock period (T), as shown in Figure 4–10. DCD is the deviation of the non-ideal falling edge from the ideal falling edge, such as D1 for the falling edge A and D2 for the falling edge B (refer to Figure 4–10). The maximum DCD for a clock is the larger value of D1 and D2.

**Figure 4–10.** Duty Cycle Distortion
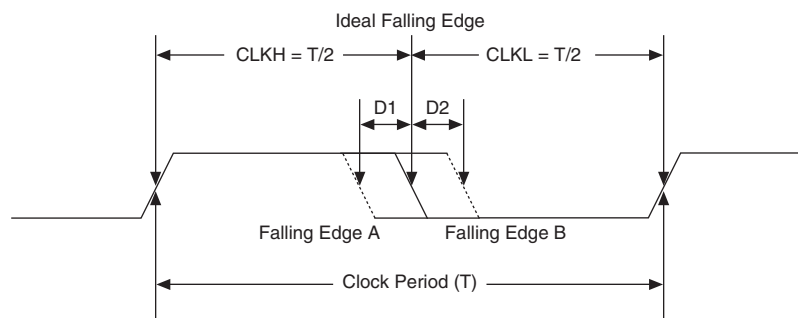


DCD expressed in absolution derivation, for example, D1 or D2 in Figure 4–10, is clock-period independent. DCD can also be expressed as a percentage, and the percentage number is clock-period dependent. DCD as a percentage is defined as:

(T/2 – D1) / T (the low percentage boundary)

(T/2 + D2) / T (the high percentage boundary)

## DCD Measurement Techniques

DCD is measured at an FPGA output pin driven by registers inside the corresponding I/O element (IOE) block. When the output is a single data rate signal (non-DDIO), only one edge of the register input clock (positive or negative) triggers output transitions (Figure 4–11). Therefore, any DCD present on the input clock signal or caused by the clock input buffer or different input I/O standard does not transfer to the output signal.

**Figure 4–11.** DCD Measurement Technique for Non-DDIO (Single-Data Rate) Outputs
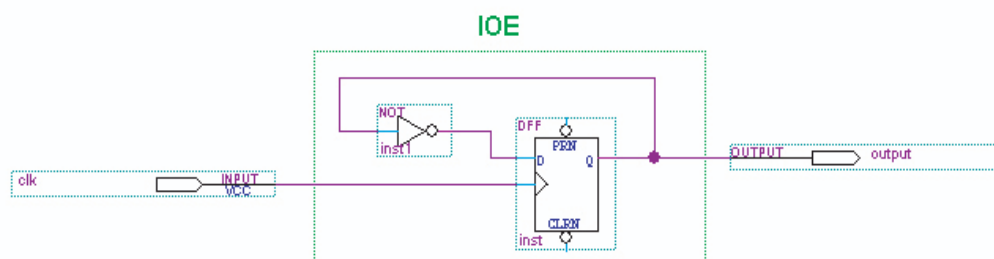


However, when the output is a double data rate input/output (DDIO) signal, both edges of the input clock signal (positive and negative) trigger output transitions (Figure 4–12). Therefore, any distortion on the input clock and the input clock buffer affect the output DCD.

**Figure 4–12.** DCD Measurement Technique for DDIO (Double-Data Rate) Outputs
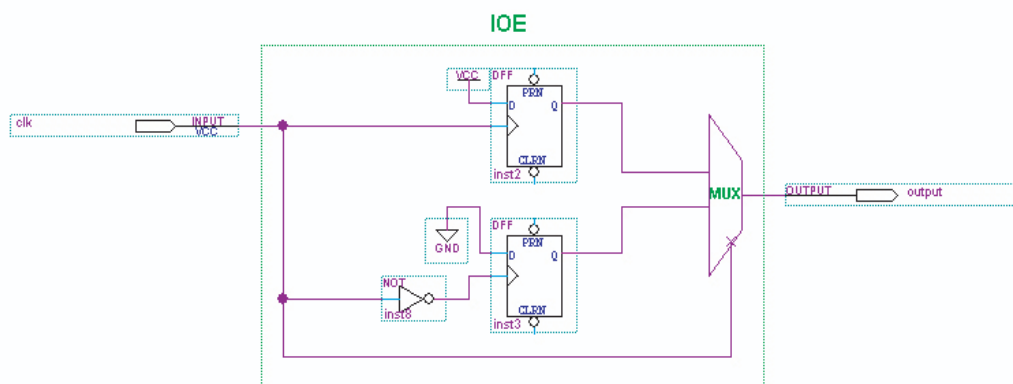


When an FPGA PLL generates the internal clock, the PLL output clocks the IOE block. As the PLL only monitors the positive edge of the reference clock input and internally re-creates the output clock signal, any DCD present on the reference clock is filtered out. Therefore, the DCD for a DDIO output with PLL in the clock path is better than the DCD for a DDIO output without PLL in the clock path.

Table 4–108 through Table 4–113 show the maximum DCD in absolution derivation for different I/O standards on Arria GX devices. Examples are also provided that show how to calculate DCD as a percentage.

**Table 4–108.** Maximum DCD for Non-DDIO Output on Row I/O Pins

| Row I/O Output Standard | Maximum DCD (ps) for Non-DDIO Output | |
| --- | --- | --- |
| | –6 Speed Grade | Units |
| 3.3-V LVTTTL | 275 | ps |
| 3.3-V LVCMOS | 155 | ps |
| 2.5 V | 135 | ps |
| 1.8 V | 180 | ps |
| 1.5-V LVCMOS | 195 | ps |
| SSTL-2 Class I | 145 | ps |
| SSTL-2 Class II | 125 | ps |
| SSTL-18 Class I | 85 | ps |
| 1.8-V HSTL Class I | 100 | ps |
| 1.5-V HSTL Class I | 115 | ps |
| LVDS | 80 | ps |

Here is an example for calculating the DCD as a percentage for a non-DDIO output on a row I/O:

If the non-DDIO output I/O standard is SSTL-2 Class II, the maximum DCD is 125 ps (see Table 4–109). If the clock frequency is 267 MHz, the clock period T is:

$T = 1/ f = 1 / 267 \text{ MHz} = 3.745 \text{ ns} = 3,745 \text{ ps}$

To calculate the DCD as a percentage:

$(T/2 - DCD) / T = (3,745\ ps/2 - 125\ ps) / 3,745\ ps = 46.66\%$ (for low boundary)

$(T/2 + DCD) / T = (3,745\ ps/2 + 125\ ps) / 3,745\ ps = 53.33\%$ (for high boundary)

Therefore, the DCD percentage for the output clock at 267 MHz is from 46.66% to 53.33%.

**Table 4–109.** Maximum DCD for Non-DDIO Output on Column I/O Pins

| Column I/O Output Standard I/O Standard | Maximum DCD (ps) for Non-DDIO Output −6 Speed Grade | Units |
|---|---|---|
| 3.3-V LVTTL | 220 | ps |
| 3.3-V LVCMOS | 175 | ps |
| 2.5 V | 155 | ps |
| 1.8 V | 110 | ps |
| 1.5-V LVCMOS | 215 | ps |
| SSTL-2 Class I | 135 | ps |
| SSTL-2 Class II | 130 | ps |
| SSTL-18 Class I | 115 | ps |
| SSTL-18 Class II | 100 | ps |
| 1.8-V HSTL Class I | 110 | ps |
| 1.8-V HSTL Class II | 110 | ps |
| 1.5-V HSTL Class I | 115 | ps |
| 1.5-V HSTL Class II | 80 | ps |
| 1.2-V HSTL-12 | 200 | ps |
| LVPECL | 80 | ps |

**Table 4–110.** Maximum DCD for DDIO Output on Row I/O Pins Without PLL in the Clock Path *Note (1)*

| Maximum DCD (ps) for Row DDIO Output I/O Standard | Input I/O Standard (No PLL in the Clock Path) | | | | | Units |
|---|---|---|---|---|---|---|
| | TTL/CMOS | | SSTL-2 | SSTL/HSTL | LVDS | |
| | 3.3/2.5V | 1.8/1.5V | 2.5V | 1.8/1.5V | 3.3V | |
| 3.3-V LVTTL | 440 | 495 | 170 | 160 | 105 | ps |
| 3.3-V LVCMOS | 390 | 450 | 120 | 110 | 75 | ps |
| 2.5 V | 375 | 430 | 105 | 95 | 90 | ps |
| 1.8 V | 325 | 385 | 90 | 100 | 135 | ps |
| 1.5-V LVCMOS | 430 | 490 | 160 | 155 | 100 | ps |
| SSTL-2 Class I | 355 | 410 | 85 | 75 | 85 | ps |
| SSTL-2 Class II | 350 | 405 | 80 | 70 | 90 | ps |
| SSTL-18 Class I | 335 | 390 | 65 | 65 | 105 | ps |
| 1.8-V HSTL Class I | 330 | 385 | 60 | 70 | 110 | ps |
| 1.5-V HSTL Class I | 330 | 390 | 60 | 70 | 105 | ps |

**Table 4–110.** Maximum DCD for DDIO Output on Row I/O Pins Without PLL in the Clock Path *Note (1)*

| Maximum DCD (ps) for Row DDIO Output I/O Standard | Input I/O Standard (No PLL in the Clock Path) | | | | | Units |
|---|---|---|---|---|---|---|
| | TTL/CMOS | | SSTL-2 | SSTL/HSTL | LVDS | |
| | 3.3/2.5V | 1.8/1.5V | 2.5V | 1.8/1.5V | 3.3V | |
| LVDS | 180 | 180 | 180 | 180 | 180 | ps |

**Note to Table 4–110:**

(1) Table 4–110 assumes the input clock has zero DCD.

**Table 4–111.** Maximum DCD for DDIO Output on Column I/O Pins Without PLL in the Clock Path *(Note 1)*

| Maximum DCD (ps) for DDIO Column Output I/O Standard | Input IO Standard (No PLL in the Clock Path) | | | | Units |
|---|---|---|---|---|---|
| | TTL/CMOS | | SSTL-2 | SSTL/HSTL | |
| | 3.3/2.5V | 1.8/1.5V | 2.5V | 1.8/1.5V | |
| 3.3-V LVTTL | 440 | 495 | 170 | 160 | ps |
| 3.3-V LVCMOS | 390 | 450 | 120 | 110 | ps |
| 2.5 V | 375 | 430 | 105 | 95 | ps |
| 1.8 V | 325 | 385 | 90 | 100 | ps |
| 1.5-V LVCMOS | 430 | 490 | 160 | 155 | ps |
| SSTL-2 Class I | 355 | 410 | 85 | 75 | ps |
| SSTL-2 Class II | 350 | 405 | 80 | 70 | ps |
| SSTL-18 Class I | 335 | 390 | 65 | 65 | ps |
| SSTL-18 Class II | 320 | 375 | 70 | 80 | ps |
| 1.8-V HSTL Class I | 330 | 385 | 60 | 70 | ps |
| 1.8-V HSTL Class II | 330 | 385 | 60 | 70 | ps |
| 1.5-V HSTL Class I | 330 | 390 | 60 | 70 | ps |
| 1.5-V HSTL Class II | 330 | 360 | 90 | 100 | ps |
| LVPECL | 180 | 180 | 180 | 180 | ps |

**Note to Table 4–111:**

(1) Table 4–111 assumes the input clock has zero DCD.

**Table 4–112.** Maximum DCD for DDIO Output on Row I/O Pins With PLL in the Clock Path

| Maximum DCD (ps) for Row DDIO Output I/O Standard | Arria GX Devices (PLL Output Feeding DDIO) | Units |
|---|---|---|
| | –6 Speed Grade | |
| 3.3-V LVTTL | 105 | ps |
| 3.3-V LVCMOS | 75 | ps |
| 2.5V | 90 | ps |
| 1.8V | 100 | ps |
| 1.5-V LVCMOS | 100 | ps |
| SSTL-2 Class I | 75 | ps |
| SSTL-2 Class II | 70 | ps |

**Table 4–112.** Maximum DCD for DDIO Output on Row I/O Pins With PLL in the Clock Path

| Maximum DCD (ps) for Row DDIO Output I/O Standard | Arria GX Devices (PLL Output Feeding DDIO) | Units |
|---|---|---|
| | −6 Speed Grade | |
| SSTL-18 Class I | 65 | ps |
| 1.8-V HSTL Class I | 70 | ps |
| 1.5-V HSTL Class I | 70 | ps |
| LVDS | 180 | ps |

**Table 4–113.** Maximum DCD for DDIO Output on Column I/O Pins With PLL in the Clock Path

| Maximum DCD (ps) for Column DDIO Output I/O Standard | Arria GX Devices (PLL Output Feeding DDIO) | Units |
|---|---|---|
| | −6 Speed Grade | |
| 3.3-V LVTTL | 160 | ps |
| 3.3-V LVCMOS | 110 | ps |
| 2.5V | 95 | ps |
| 1.8V | 100 | ps |
| 1.5-V LVCMOS | 155 | ps |
| SSTL-2 Class I | 75 | ps |
| SSTL-2 Class II | 70 | ps |
| SSTL-18 Class I | 65 | ps |
| SSTL-18 Class II | 80 | ps |
| 1.8-V HSTL Class I | 70 | ps |
| 1.8-V HSTL Class II | 70 | ps |
| 1.5-V HSTL Class I | 70 | ps |
| 1.5-V HSTL Class II | 100 | ps |
| 1.2-V HSTL | 155 | ps |
| LVPECL | 180 | ps |

# High-Speed I/O Specifications

Table 4–114 lists high-speed timing specifications definitions.

**Table 4–114.** High-Speed Timing Specifications and Definitions  (Part 1 of 2)

| High-Speed Timing Specifications | Definitions |
|---|---|
| $t_C$ | High-speed receiver/transmitter input and output clock period. |
| $f_{HSCLK}$ | High-speed receiver/transmitter input and output clock frequency. |
| J | Deserialization factor (width of parallel data bus). |
| W | PLL multiplication factor. |
| $t_{RISE}$ | Low-to-high transmission time. |
| $t_{FALL}$ | High-to-low transmission time. |

**Table 4–114.** High-Speed Timing Specifications and Definitions  (Part 2 of 2)

| High-Speed Timing Specifications | Definitions |
|---|---|
| Timing unit interval (TUI) | The timing budget allowed for skew, propagation delays, and data sampling window. (TUI = 1/(Receiver Input Clock Frequency × Multiplication Factor) = $t_C/w$). |
| $f_{HSDR}$ | Maximum/minimum LVDS data transfer rate ($f_{HSDR}$ = 1/TUI), non-DPA. |
| $f_{HSDRDPA}$ | Maximum/minimum LVDS data transfer rate ($f_{HSDRDPA}$ = 1/TUI), DPA. |
| Channel-to-channel skew (TCCS) | The timing difference between the fastest and slowest output edges, including $t_{CO}$ variation and clock skew. The clock is included in the TCCS measurement. |
| Sampling window (SW) | The period of time during which the data must be valid in order to capture it correctly. The setup and hold times determine the ideal strobe position within the sampling window. |
| Input jitter | Peak-to-peak input jitter on high-speed PLLs. |
| Output jitter | Peak-to-peak output jitter on high-speed PLLs. |
| $t_{DUTY}$ | Duty cycle on high-speed transmitter output clock. |
| $t_{LOCK}$ | Lock time for high-speed transmitter and receiver PLLs. |

Table 4–115 shows the high-speed I/O timing specifications.

**Table 4–115.** High-Speed I/O Specifications   (Part 1 of 2)*Note (1), (2)*

| Symbol | Conditions | −6 Speed Grade | | | Units |
|---|---|---|---|---|---|
| | | Min | Typ | Max | |
| $f_{HSCLK}$ (clock frequency) $f_{HSCLK} = f_{HSDR}$ / W | W = 2 to 32 (LVDS, HyperTransport technology) *(3)* | 16 | — | 420 | MHz |
| | W = 1 (SERDES bypass, LVDS only) | 16 | — | 500 | MHz |
| | W = 1 (SERDES used, LVDS only) | 150 | — | 640 | MHz |
| $f_{HSDR}$ (data rate) | J = 4 to 10 (LVDS, HyperTransport technology) | 150 | — | 840 | Mbps |
| | J = 2 (LVDS, HyperTransport technology) | *(4)* | — | 700 | Mbps |
| | J = 1 (LVDS only) | *(4)* | — | 500 | Mbps |
| $f_{HSDRDPA}$ (DPA data rate) | J = 4 to 10 (LVDS, HyperTransport technology) | 150 | — | 840 | Mbps |
| TCCS | All differential I/O standards | — | — | 200 | ps |
| SW | All differential I/O standards | 440 | — | — | ps |
| Output jitter | — | — | — | 190 | ps |
| Output $t_{RISE}$ | All differential I/O standards | — | — | 290 | ps |
| Output $t_{FALL}$ | All differential I/O standards | — | — | 290 | ps |
| $t_{DUTY}$ | — | 45 | 50 | 55 | % |
| DPA run length | — | — | — | 6,400 | UI |
| DPA jitter tolerance | Data channel peak-to-peak jitter | 0.44 | — | — | UI |

**Table 4–115.** High-Speed I/O Specifications   (Part 2 of 2)*Note (1), (2)*

| Symbol | Conditions | | | −6 Speed Grade | | | Units |
|---|---|---|---|---|---|---|---|
| | | | | Min | Typ | Max | |
| DPA lock time | Standard | Training Pattern | Transition Density | | — | — | Number of repetitions |
| | SPI-4 | 000000000011 11111111 | 10% | 256 | — | — | |
| | Parallel Rapid I/O | 00001111 | 25% | 256 | — | — | |
| | | 10010000 | 50% | 256 | — | — | |
| | Miscellaneous | 10101010 | 100% | 256 | — | — | |
| | | 01010101 | — | 256 | — | — | |

**Notes to Table 4–115:**

(1)   When J = 4 to 10, the SERDES block is used.

(2)   When J = 1 or 2, the SERDES block is bypassed.

(3)   The input clock frequency and the W factor must satisfy the following fast PLL VCO specification: $150 \leq$ input clock frequency $\times W \leq 1{,}040$.

(4)   The minimum specification is dependent on the clock source (fast PLL, enhanced PLL, clock pin, and so on) and the clock routing resource (global, regional, or local) used. The I/O differential buffer and input register do not have a minimum toggle rate.

# PLL Timing Specifications

Table 4–116 and Table 4–117 describe the Arria GX PLL specifications when operating in both the commercial junction temperature range (0 to 85 C) and the industrial junction temperature range (–40 to 100 C), except for the clock switchover and phase-shift stepping features. These two features are only supported from the 0 to 100 C junction temperature range.

**Table 4–116.** Enhanced PLL Specifications  (Part 1 of 2)

| Name | Description | Min | Typ | Max | Units |
|------|-------------|-----|-----|-----|-------|
| $f_{IN}$ | Input clock frequency | 2 | — | 500 | MHz |
| $f_{INPFD}$ | Input frequency to the PFD | 2 | — | 420 | MHz |
| $f_{INDUTY}$ | Input clock duty cycle | 40 | — | 60 | % |
| $f_{ENDUTY}$ | External feedback input clock duty cycle | 40 | — | 60 | % |
| $t_{INJITTER}$ | Input or external feedback clock input jitter tolerance in terms of period jitter. Bandwidth ≤ 0.85 MHz | — | 0.5 | — | ns (peak-to-peak) |
| | Input or external feedback clock input jitter tolerance in terms of period jitter. Bandwidth > 0.85 MHz | — | 1.0 | — | ns (peak-to-peak) |
| $t_{OUTJITTER}$ | Dedicated clock output period jitter | 50 | 100 | 250 | ps (p-p) |
| $t_{FCOMP}$ | External feedback compensation time | — | — | 10 | ns |
| $f_{OUT}$ | Output frequency for internal global or regional clock | 1.5 *(2)* | — | 550 | MHz |
| $f_{SCANCLK}$ | Scanclk frequency | — | — | 100 | MHz |
| $t_{CONFIGEPLL}$ | Time required to reconfigure scan chains for EPLLs | — | $174/f_{SCANCLK}$ | — | ns |
| $f_{OUT\_EXT}$ | PLL external clock output frequency | 1.5 *(2)* | | *(1)* | MHz |
| $f_{OUTDUTY}$ | Duty cycle for external clock output | 45 | 50 | 55 | % |
| $t_{LOCK}$ | Time required for the PLL to lock from the time it is enabled or the end of device configuration | — | 0.03 | 1 | ms |
| $t_{DLOCK}$ | Time required for the PLL to lock dynamically after automatic clock switchover between two identical clock frequencies | — | — | 1 | ms |
| $f_{SWITCHOVER}$ | Frequency range where the clock switchover performs properly | 1.5 | 1 | 500 | MHz |
| $f_{CLBW}$ | PLL closed-loop bandwidth | 0.13 | 1.2 | 16.9 | MHz |
| $f_{VCO}$ | PLL VCO operating range | 300 | — | 840 | MHz |
| $f_{SS}$ | Spread-spectrum modulation frequency | 100 | — | 500 | kHz |
| % spread | Percent down spread for a given clock frequency | 0.4 | 0.5 | 0.6 | % |
| $t_{PLL\_PSERR}$ | Accuracy of PLL phase shift | — | — | ±30 | ps |
| $t_{ARESET}$ | Minimum pulse width on `areset` signal. | 10 | — | — | ns |
| $t_{ARESET\_RECONFIG}$ | Minimum pulse width on the `areset` signal when using PLL reconfiguration. Reset the PLL after `scandone` goes high. | 500 | — | — | ns |

**Table 4–116.** Enhanced PLL Specifications  (Part 2 of 2)

| Name | Description | Min | Typ | Max | Units |
|------|-------------|-----|-----|-----|-------|
| $t_{RECONFIGWAIT}$ | The time required for the wait after the reconfiguration is done and the areset is applied. | — | — | 2 | us |

**Notes to Table 4–116:**

(1) This is limited by the I/O $f_{MAX}$.

(2) If the counter cascading feature of the PLL is used, there is no minimum output clock frequency.

**Table 4–117.** Fast PLL Specifications  (Part 1 of 2)

| Name | Description | Min | Typ | Max | Units |
|------|-------------|-----|-----|-----|-------|
| $f_{IN}$ | Input clock frequency | 16.08 | — | 640 | MHz |
| $f_{INPFD}$ | Input frequency to the PFD | 16.08 | — | 500 | MHz |
| $f_{INDUTY}$ | Input clock duty cycle | 40 | — | 60 | % |
| $t_{INJITTER}$ | Input clock jitter tolerance in terms of period jitter. Bandwidth ≤ 2 MHz | — | 0.5 | — | ns (p-p) |
| | Input clock jitter tolerance in terms of period jitter. Bandwidth > 0.2 MHz | — | 1.0 | — | ns (p-p) |
| $f_{VCO}$ | Upper VCO frequency range | 300 | — | 840 | MHz |
| | Lower VCO frequency range | 150 | — | 420 | MHz |
| $f_{OUT}$ | PLL output frequency to `GCLK` or `RCLK` | 4.6875 | — | 550 | MHz |
| | PLL output frequency to LVDS or DPA clock | 150 | — | 840 | MHz |
| $f_{OUT\_EXT}$ | PLL clock output frequency to regular I/O | 4.6875 | — | *(1)* | MHz |
| $t_{CONFIGPLL}$ | Time required to reconfigure scan chains for fast PLLs | — | $75/f_{SCANCLK}$ | — | ns |
| $f_{CLBW}$ | PLL closed-loop bandwidth | 1.16 | 5 | 28 | MHz |
| $t_{LOCK}$ | Time required for the PLL to lock from the time it is enabled or the end of the device configuration | — | 0.03 | 1 | ms |
| $t_{PLL\_PSERR}$ | Accuracy of PLL phase shift | — | — | ±30 | ps |
| $t_{ARESET}$ | Minimum pulse width on `areset` signal. | 10 | — | — | ns |

**Table 4–117.** Fast PLL Specifications  (Part 2 of 2)

| Name | Description | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| $t_{ARESET\_RECONFIG}$ | Minimum pulse width on the `areset` signal when using PLL reconfiguration. Reset the PLL after `scandone` goes high. | 500 | — | — | ns |

**Note to Table 4–117:**

(1)  This is limited by the I/O $f_{MAX}$.

# External Memory Interface Specifications

Table 4–118 through Table 4–122 list Arria GX device specifications for the dedicated circuitry used for interfacing with external memory devices.

**Table 4–118.** DLL Frequency Range Specifications

| Frequency Mode | Frequency Range (MHz) |
|---|---|
| 0 | 100 to 175 |
| 1 | 150 to 230 |
| 2 | 200 to 310 |

**Table 4–119.** DQS Jitter Specifications for DLL-Delayed Clock (tDQS_JITTER) ,  *(Note 1)*

| Number of DQS Delay Buffer Stages *(2)* | Commercial (ps) | Industrial (ps) |
|---|---|---|
| 1 | 80 | 110 |
| 2 | 110 | 130 |
| 3 | 130 | 180 |
| 4 | 160 | 210 |

**Notes to Table 4–119:**

(1)  Peak-to-peak period jitter on the phase-shifted DQS clock. For example, jitter on two delay stages under commercial conditions is 200 ps peak-to-peak or 100 ps.

(2)  Delay stages used for requested DQS phase shift are reported in a project's Compilation Report in the Quartus II software.

**Table 4–120.** DQS Phase-Shift Error Specifications for DLL-Delayed Clock ($t_{DQS\_PSERR}$)

| Number of DQS Delay Buffer Stages | –6 Speed Grade (ps) |
|---|---|
| 1 | 35 |
| 2 | 70 |
| 3 | 105 |
| 4 | 140 |

**Table 4–121.** DQS Bus Clock Skew Adder Specifications ($t_{DQS\_CLOCK\_SKEW\_ADDER}$)

| Mode | DQS Clock Skew Adder (ps) |
|---|---|
| 4 DQ per DQS | 40 |
| 9 DQ per DQS | 70 |
| 18 DQ per DQS | 75 |
| 36 DQ per DQS | 95 |

**Table 4–122.** DQS Phase Offset Delay Per Stage (ps)     *Note (1), (2), (3)*

| Speed Grade | Positive Offset | | Negative Offset | |
|---|---|---|---|---|
| | Min | Max | Min | Max |
| –6 | 10 | 16 | 8 | 12 |

**Notes to Table 4–122:**

(1) The delay settings are linear.

(2) The valid settings for phase offset are –32 to +31.

(3) The typical value equals the average of the minimum and maximum values.

# JTAG Timing Specifications

Figure 4–13 shows the timing requirements for the JTAG signals

**Figure 4–13.** Arria GX JTAG Waveforms.
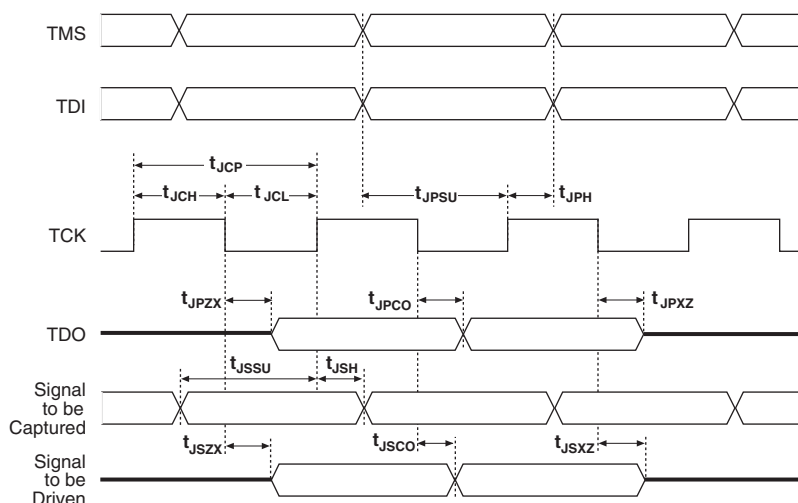
Table 4–123 lists the JTAG timing parameters and values for Arria GX devices.

**Table 4–123.** Arria GX JTAG Timing Parameters and Values

| Symbol | Parameter | Min | Max | Units |
|--------|-----------|-----|-----|-------|
| $t_{JCP}$ | TCK clock period | 30 | — | ns |
| $t_{JCH}$ | TCK clock high time | 12 | — | ns |
| $t_{JCL}$ | TCK clock low time | 12 | — | ns |
| $t_{JPSU}$ | JTAG port setup time | 4 | — | ns |
| $t_{JPH}$ | JTAG port hold time | 5 | — | ns |
| $t_{JPCO}$ | JTAG port clock to output | — | 9 | ns |
| $t_{JPZX}$ | JTAG port high impedance to valid output | — | 9 | ns |
| $t_{JPXZ}$ | JTAG port valid output to high impedance | — | 9 | ns |
| $t_{JSSU}$ | Capture register setup time | 4 | — | ns |
| $t_{JSH}$ | Capture register hold time | 5 | — | ns |
| $t_{JSCO}$ | Update register clock to output | — | 12 | ns |
| $t_{JSZX}$ | Update register high impedance to valid output | — | 12 | ns |
| $t_{JSXZ}$ | Update register valid output to high impedance | — | 12 | ns |

# Document Revision History

Table 4–124 lists the revision history for this chapter.

**Table 4–124.** Document Revision History

| Date and Document Version | Changes Made | Summary of Changes |
|---|---|---|
| December 2009, v2.0 | ■ Updated Table 4–104, Table 4–105, and Table 4–106.<br>■ Document template update.<br>■ Minor text edits. | — |
| April 2009 v1.4 | ■ Updated Table 4–6 and Table 4–7.<br>■ Updated "Maximum Input and Output Clock Toggle Rate" section. | — |
| May 2008 v1.3 | Updated:<br>■ Table 4–5<br>■ Table 4–7<br>■ Table 4–8<br>■ Table 4–9<br>■ Table 4–10<br>■ Table 4–11<br>■ Table 4–12<br>■ Table 4–13<br>■ Table 4–14<br>■ Table 4–15<br>■ Table 4–16<br>■ Table 4–17<br>■ Table 4–43<br>■ Table 4–116<br>■ Table 4–117 | — |
| | Updated:<br>■ Figure 4–4 | — |
| | Minor text edits. | — |
| August 2007 v1.2 | Removed "Preliminary" from each page. | — |
| | Removed "Preliminary" note from Tables 4–44, 4–45, and 4–47. | — |
| | Added "Referenced Documents" section. | — |
| June 2007 v1.1 | Updated Table 4–99. | — |
| | Added GIGE information. | — |
| May 2007 v1.0 | Initial release. | — |

# Software

Arria® GX devices are supported by the Altera® Quartus® II design software, which provides a comprehensive environment for system-on-a-programmable-chip (SOPC) design. The Quartus II software includes HDL and schematic design entry, compilation and logic synthesis, full simulation and advanced timing analysis, SignalTap® II logic analyzer, and device configuration.

For more information about the Quartus II software features, refer to the *Quartus II Development Software Handbook* .

The Quartus II software supports the Windows XP/2000/NT, Sun Solaris 8/9, Linux Red Hat v7.3, Linux Red Hat Enterprise 3, and HP-UX operating systems. It also supports seamless integration with industry-leading EDA tools through the NativeLink interface.

# Device Pin-Outs

Arria GX device pin-outs are available on the Altera web site at www.altera.com.

# Ordering Information

Figure 5–1 describes the ordering codes for Arria GX devices.

For more information on a specific package, refer to the *Package Information for Arria GX Devices* chapter.

**Figure 5–1.** Arria GX Device Packaging Ordering Information
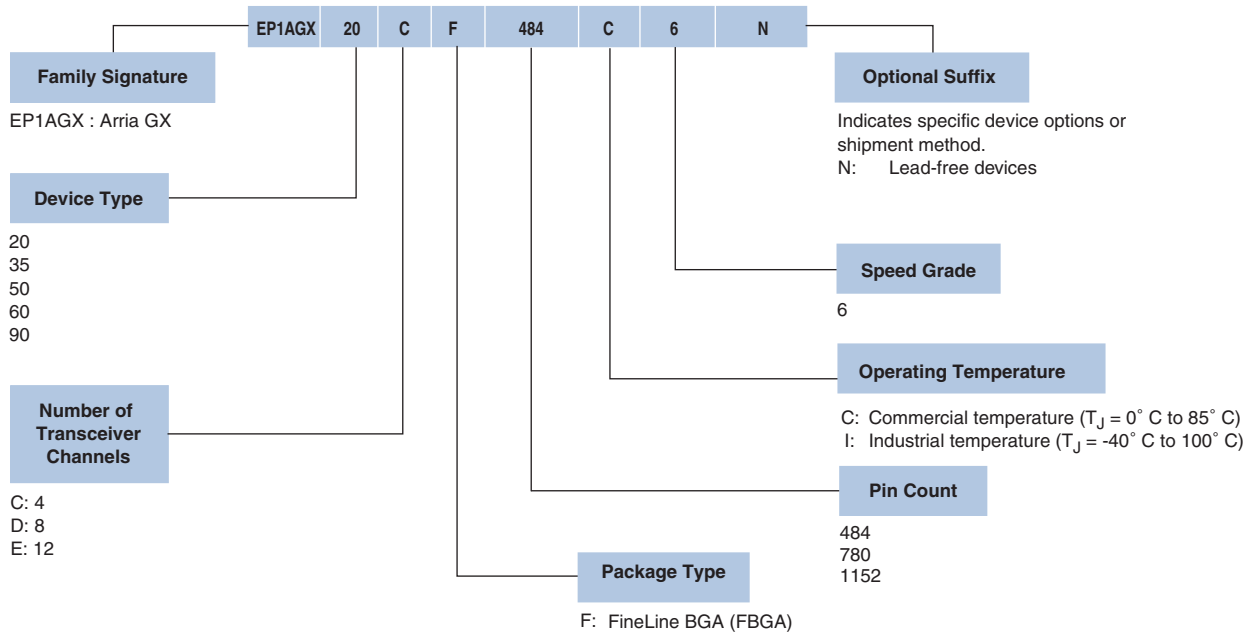


## Document Revision History

Table 5–1 shows the revision history for this chapter.

**Table 5–1.** Document Revision History

| Date and Document Version | Changes Made | Summary of Changes |
|---|---|---|
| December 2009, v2.0 | ■ Document template update.<br>■ Minor text edits. | — |
| August 2007, v1.1 | Added the "Referenced Documents" section. | — |
| May 2007, v1.0 | Initial Release. | — |

## About this Handbook

This handbook provides comprehensive information about the Altera® Arria® GX family of devices.

## How to Contact Altera

For the most up-to-date information about Altera products, see the following table.

| Contact   *(Note 1)* | Contact Method | Address |
|---|---|---|
| Technical support | Website | www.altera.com/support |
| Technical training | Website | www.altera.com/training |
| | Email | custrain@altera.com |
| Product literature | Email | www.altera.com/literature |
| Non-technical support (General) | Email | nacomp@altera.com |
| (Software Licensing) | Email | authorization@altera.com |

**Note:**

(1)   You can also contact your local Altera sales office or sales representative.

## Typographic Conventions

The following table shows the typographic conventions that this document uses.

| Visual Cue | Meaning |
|---|---|
| **Bold Type with Initial Capital Letters** | Indicates command names and dialog box titles. For example, **Save As** dialog box. |
| **bold type** | Indicates directory names, project names, disk drive names, file names, file name extensions, dialog box options, software utility names, and other GUI labels. For example, **\qdesigns** directory, **d:** drive, and **chiptrip.gdf** file. |
| *Italic Type with Initial Capital Letters* | Indicates document titles. For example, *AN 519: Stratix IV Design Guidelines.* |
| *Italic type* | Indicates variables. For example, *n* + 1.<br><br>Variable names are enclosed in angle brackets (< >). For example, *<file name>* and *<project name>*.**pof** file. |
| Initial Capital Letters | Indicates keyboard keys and menu names. For example, Delete key and the Options menu. |
| "Subheading Title" | Quotation marks indicate references to sections within a document and titles of Quartus II Help topics. For example, "Typographic Conventions." |

| Visual Cue | Meaning |
|---|---|
| Courier type | Indicates signal, port, register, bit, block, and primitive names. For example, `data1`, `tdi`, and `input`. Active-low signals are denoted by suffix `n`. For example, `resetn`. |
| | Indicates command line commands and anything that must be typed exactly as it appears. For example, `c:\qdesigns\tutorial\chiptrip.gdf`. |
| | Also indicates sections of an actual file, such as a Report File, references to parts of files (for example, the AHDL keyword `SUBDESIGN`), and logic function names (for example, `TRI`). |
| 1., 2., 3., and a., b., c., and so on. | Numbered steps indicate a list of items when the sequence of the items is important, such as the steps listed in a procedure. |
| ■  ■ | Bullets indicate a list of items when the sequence of the items is not important. |
| ☞ | The hand points to information that requires special attention. |
| ⚠ CAUTION | A caution calls attention to a condition or possible situation that can damage or destroy the product or your work. |
| ⚠ WARNING | A warning calls attention to a condition or possible situation that can cause you injury. |
| ↵ | The angled arrow instructs you to press **Enter**. |
| 👣 | The feet direct you to more information about a particular topic. |

# Arria GX Device Handbook, Volume 2

# Chapter Revision Dates

The chapters in this book, *Arria GX Device Handbook, Volume 2*, were revised on the following dates. Where chapters or groups of chapters are available separately, part numbers are listed.

Chapter 1.  Arria GX Transceiver Architecture
Revised: *May 2008*
Part number: *AGX52001-2.0*

Chapter 2.  Arria GX Transceiver Protocol Support and Additional Features
Revised: *May 2008*
Part number: *AGX52002-2.0*

Chapter 3.  Arria GX ALT2GXB Megafunction User Guide
Revised: *May 2008*
Part number: *AGX52003-2.0*

Chapter 4.  Specifications and Additional Information
Revised: *May 2007*
Part number: *AGX52004-1.0*

Chapter 5.  PLLs in Arria GX Devices
Revised: *May 2008*
Part number: *AGX52005-1.2*

Chapter 6.  TriMatrix Embedded Memory Blocks in Arria GX Devices
Revised: *May 2008*
Part number: *AGX52006-1.2*

Chapter 7.  External Memory Interfaces in Arria GX Devices
Revised: *May 2008*
Part number: *AGX52007-1.2*

Chapter 8.  Selectable I/O Standards in Arria GX Devices
Revised: *May 2008*
Part number: *AGX52008-1.2*

Chapter 9.  High-Speed Differential I/O Interfaces with DPA in Arria GX Devices
Revised: *May 2008*
Part number: *AGX52009-1.2*

# Contents

# Section I. Arria GX Transceiver User Guide

## Chapter 1. Arria GX Transceiver Architecture

## Chapter 2. Arria GX Transceiver Protocol Support and Additional Features

## Chapter 3. Arria GX ALT2GXB Megafunction User Guide

## Chapter 4. Specifications and Additional Information

# Section II. Clock Management

## Chapter 5. PLLs in Arria GX Devices

# Section III. Memory

## Chapter 6. TriMatrix Embedded Memory Blocks in Arria GX Devices

# Section IV. I/O Standards

## Chapter 8. Selectable I/O Standards in Arria GX Devices

## Chapter 9. High-Speed Differential I/O Interfaces with DPA in Arria GX Devices

# Section V. Digital Signal Processing (DSP)

## Chapter 10. DSP Blocks in Arria GX Devices

# Section VI. Configuration & Remote System Upgrades

## Chapter 11. Configuring Arria GX Devices

## Chapter 12. Remote System Upgrades with Arria GX Devices

## Chapter 13. IEEE 1149.1 (JTAG) Boundary-Scan Testing for Arria GX Devices

# Section VII. PCB Layout Guidelines

## Chapter 14. Package Information for Arria GX Devices

# About this Handbook

This handbook provides comprehensive information about the Altera®
Arria™ GX family of devices.

## How to Contact Altera

For the most up-to-date information about Altera products, refer to the
following table.

| Contact *(1)* | Contact Method | Address |
|---|---|---|
| Technical support | Website | www.altera.com/support |
| Technical training | Website | www.altera.com/training |
| | Email | custrain@altera.com |
| Product literature | Website | www.altera.com/literature |
| Non-technical support (General) | Email | nacomp@altera.com |
| (Software Licensing) | Email | authorization@altera.com |

*Note to table:*
(1)    You can also contact your local Altera sales office or sales representative.

## Typographic Conventions

This document uses the typographic conventions shown below.

| Visual Cue | Meaning |
|---|---|
| **Bold Type with Initial Capital Letters** | Command names, dialog box titles, checkbox options, and dialog box options are shown in bold, initial capital letters. Example: **Save As** dialog box. |
| **bold type** | External timing parameters, directory names, project names, disk drive names, filenames, filename extensions, and software utility names are shown in bold type. Examples: **f$_{MAX}$, \qdesigns** directory, **d:** drive, **chiptrip.gdf** file. |
| *Italic Type with Initial Capital Letters* | Document titles are shown in italic type with initial capital letters. Example: *AN 75: High-Speed Board Design.* |
| *Italic type* | Internal timing parameters and variables are shown in italic type. Examples: $t_{PIA}$, *n* + 1.<br><br>Variable names are enclosed in angle brackets (< >) and shown in italic type. Example: *<file name>*, *<project name>***.pof** file. |

| Visual Cue | Meaning |
|---|---|
| Initial Capital Letters | Keyboard keys and menu names are shown with initial capital letters. Examples: Delete key, the Options menu. |
| "Subheading Title" | References to sections within a document and titles of on-line help topics are shown in quotation marks. Example: "Typographic Conventions." |
| Courier type | Signal and port names are shown in lowercase Courier type. Examples: `data1`, `tdi`, `input`. Active-low signals are denoted by suffix `n`, e.g., `resetn`.<br><br>Anything that must be typed exactly as it appears is shown in Courier type. For example: `c:\qdesigns\tutorial\chiptrip.gdf`. Also, sections of an actual file, such as a Report File, references to parts of files (e.g., the AHDL keyword `SUBDESIGN`), as well as logic function names (e.g., `TRI`) are shown in Courier. |
| 1., 2., 3., and a., b., c., etc. | Numbered steps are used in a list of items when the sequence of the items is important, such as the steps listed in a procedure. |
| ■ ● • | Bullets are used in a list of items when the sequence of the items is not important. |
| ✓ | The checkmark indicates a procedure that consists of one step only. |
| ☞ | The hand points to information that requires special attention. |
| ⚠ CAUTION | A caution calls attention to a condition or possible situation that can damage or destory the product or the user's work. |
| ⚠ WARNING | The warning calls attention to a condition or possible situation that could cause injury to the user. |
| ↵ | The angled arrow indicates you should press the Enter key. |
| 👣 | The feet direct you to more information on a particular topic. |

# Section I. Arria GX Transceiver User Guide

This section provides information on the configuration modes for Arria™ GX devices. It also includes information on testing, Arria GX port and parameter information, and pin constraint information.

This section includes the following chapters:

- Chapter 1, Arria GX Transceiver Architecture

- Chapter 2, Arria GX Transceiver Protocol Support and Additional Features

- Chapter 3, Arria GX ALT2GXB Megafunction User Guide

- Chapter 4, Specifications and Additional Information

## Revision History

Refer to each chapter for its own specific revision history. For information on when each chapter was updated, refer to the Chapter Revision Dates section, which appears in the full handbook.

# 1. Arria GX Transceiver Architecture

## Introduction

Arria™ GX is a protocol-optimized FPGA family that leverages Altera's advanced multi-gigabit transceivers. The Arria GX transceiver blocks build on the success of the Stratix® II GX family and are optimally designed to support the following serial connectivity protocols (functional modes):

- XAUI
- PCI Express (PIPE)
- Gigabit Ethernet (GIGE)
- SDI
- Serial RapidIO®
- Basic Mode

## Building Blocks

Arria GX transceivers are structured into full duplex (transmitter and receiver) four-channel groups called transceiver blocks. The Arria GX device family offers up to 12 transceiver channels (three transceiver blocks) per device. You can configure each transceiver block to one of the supported functional modes; for example, four GIGE ports or one four-lane (×4) PCI Express (PIPE) port. In Arria GX devices that offer more than one transceiver block, you can configure each transceiver block to a different functional mode; for example, one transceiver block configured as a four-lane (×4) PCI Express (PIPE) port and the other transceiver block can be configured as four GIGE ports.

Figure 1–1 shows the Arria GX transceiver block diagram divided into transmitter and receiver circuits.

*Figure 1–1. Arria GX Gigabit Transceiver Block Diagram*
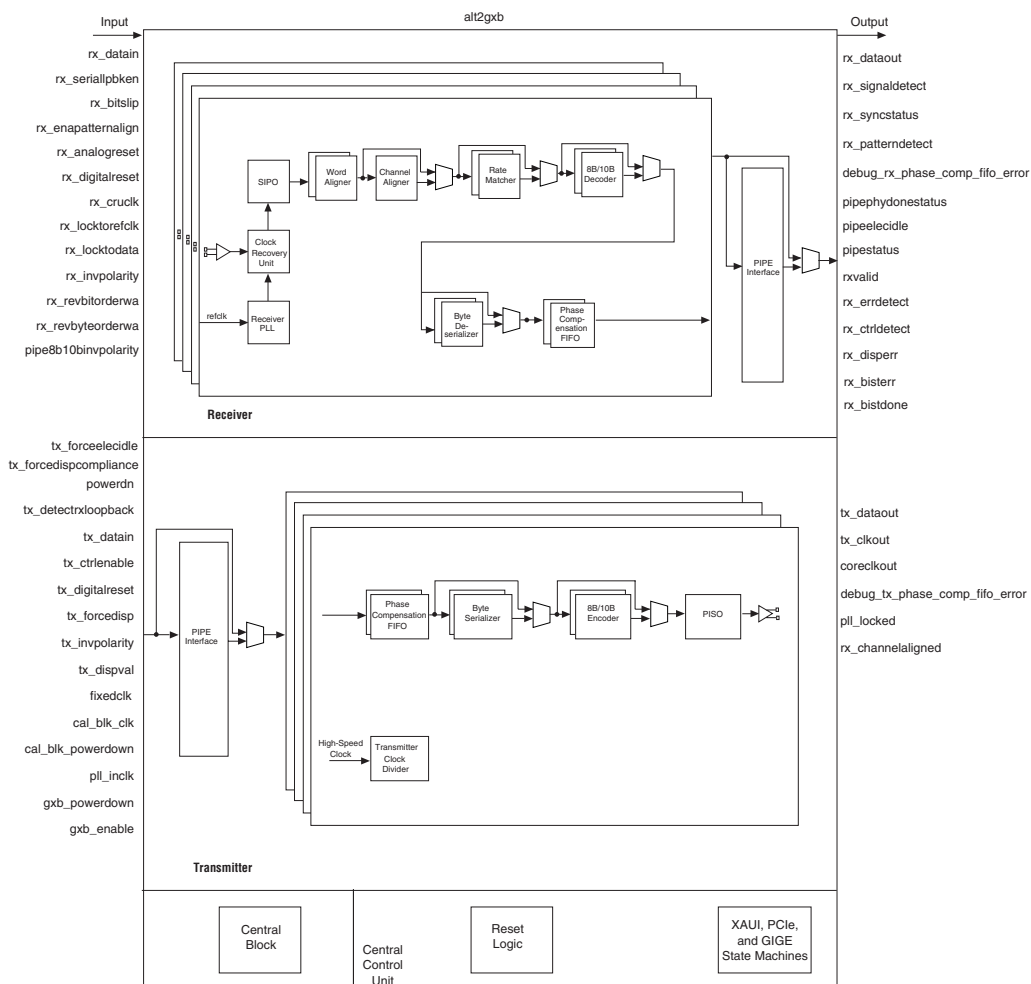
# Port List

You instantiate the Arria GX transceivers using the ALT2GXB MegaCore® instance provided in the Quartus® II MegaWizard® Plug-In Manager. The ALT2GXB instance allows you to configure the transceivers for your intended protocol and select optional control and status ports to and from the instantiated transceiver channels.

| Port Name | Input/Output | Description | Scope |
|---|---|---|---|
| *Table 1–1. Arria GX ALT2GXB Ports (Part 1 of 6)* | | | |
| **Receiver Physical Coding Sublayer (PCS) Ports** | | | |
| `rx_dataout` | Output | Receiver parallel data output. The bus width depends on the channel width multiplied by the number of channels per instance. | — |
| `rx_clkout` | Output | Recovered clock from the receiver channel. | Channel |
| `rx_coreclk` | Input | Optional read clock port for the receiver phase compensation first-in first-out (FIFO). If not selected, the Quartus II software automatically selects `rx_clkout/tx_clkout` as the read clock for receiver phase compensation FIFO. If selected, you must drive this port with a clock that is frequency locked to `rx_clkout/tx_clkout`. | Channel |
| `rx_enapatternalign` | Input | Enables word aligner to align to the comma. This port can be either edge or level sensitive based on the word aligner mode. | Channel |
| `rx_bitslip` | Input | Word aligner bit slip control. The word aligner slips a bit of the current word boundary every rising edge of this signal. | Channel |
| `rx_rlv` | Output | Run-length violation indicator. A high signal is driven when the run length (consecutive '1's or '0's) of the received data exceeds the configured limit. | Channel |
| `pipe8b10binvpolarity` | Input | Physical Interface for PCI Express (PIPE) polarity inversion at the 8B/10B decoder input. This port inverts the data at the input to the 8B/10B decoder. | Channel |

| Table 1–1. Arria GX ALT2GXB Ports (Part 2 of 6) | | | |
|---|---|---|---|
| **Port Name** | **Input/ Output** | **Description** | **Scope** |
| pipestatus | Output | PIPE receiver status port. In case of multiple status signals, the lower number signal takes precedence.<br>000 - Received data OK<br>001 - 1 skip added (not supported)<br>010 - 1 skip removed (not supported)<br>011 - Receiver detected<br>100 - 8B/10B decoder error<br>101 - Elastic buffer overflow<br>110 - Elastic buffer underflow<br>111 - Received disparity error | Channel |
| pipephydonestatus | Output | PIPE indicates a mode transition completion-power transition and rx_detect. A pulse is given. | Channel |
| rx_pipedatavalid | Output | PIPE valid data indicator on the rx_dataout port. | Channel |
| pipeelecidle | Output | PIPE signal detect for PCI Express. | Channel |
| rx_digitalreset | Input | Reset port for the receiver PCS block. This port resets all the digital logic in the receiver channel. The minimum pulse width is two parallel clock cycles. | Channel |
| rx_bisterr | Output | Built-in self test (BIST) block error flag. This port latches high if an error is detected. Assertion of rx_digitalreset resets the BIST verifier, which clears the error flag. | Channel |
| rx_bistdone | Output | Built-in self test verifier done flag. This port goes high if the receiver finishes reception of the test sequence. | Channel |
| rx_ctrldetect | Output | Receiver control code indicator port. Indicates whether the data at the output of rx_dataout is a control or data word. Used with the 8B/10B decoder. | Channel |
| rx_errdetect | Output | 8B/10B code group violation signal. Indicates that the data at the output of rx_dataout has a code violation or a disparity error. Used with disparity error signal to differentiate between a code group error and/or a disparity error. In addition, in XAUI mode, rx_errdetect is asserted in the corresponding byte position when ALT2GXB substitutes the received data with 9'b1FE because of XAUI protocol violations. | Channel |

## Table 1–1. Arria GX ALT2GXB Ports   (Part 3 of 6)

| Port Name | Input/ Output | Description | Scope |
|---|---|---|---|
| rx_syncstatus | Output | Indicates when the word aligner either aligns to a new word boundary (in single width mode the rx_patterndetect port is level sensitive), indicates that a resynchronization is needed (the rx_patterndetect is edge sensitive), or indicates if synchronization is achieved or not (the dedicated synchronization state machine is used). | Channel |
| rx_disperr | Output | 8B/10B disparity error indicator port. Indicates that the data at the output of rx_dataout has a disparity error. | Channel |
| rx_patterndetect | Output | Indicates when the word aligner detects the alignment pattern in the current word boundary. | Channel |
| rx_invpolarity | Input | Inverts the polarity of the received data at the input of the word aligner | Channel |
| rx_revbitorderwa | Input | Available in Basic mode with bit-slip word alignment enabled. Reverses the bit-order of the received data at a byte level at the output of the word aligner. | Channel |
| debug_rx_phase_comp_ fifo_error | Output | Indicates receiver phase compensation FIFO overrun or underrun situation | Channel |
| **Receiver Physical Media Attachment (PMA)** | | | |
| rx_pll_locked | Output | Receiver PLL locked signal. Indicates if the receiver PLL is phase locked to the CRU reference clock. | Channel |
| rx_analogreset | Input | Receiver analog reset. Resets all analog circuits in the receiver PMA. | Channel |
| rx_freqlocked | Output | CRU mode indicator port. Indicates if the CRU is locked to data mode or locked to the reference clock mode. 0 – Receiver CRU is in lock-to-reference clock mode 1 – Receiver CRU is in lock-to-data mode | Channel |
| rx_signaldetect | Output | Signal detect port. In PIPE mode, indicates if a signal that meets the specified range is present at the input of the receiver buffer. In all other modes, rx_signaldetect is forced high and must not be used as an indication of a valid signal at receiver input. | Channel |

*Table 1–1. Arria GX ALT2GXB Ports  (Part 4 of 6)*

| Port Name | Input/Output | Description | Scope |
|---|---|---|---|
| rx_seriallpbken | Input | Serial loopback control port.<br>0 – normal data path, no serial loopback<br>1 – serial loopback | Channel |
| rx_locktodata | Input | Lock-to-data control for the CRU. Use with rx_locktorefclk. | Channel |
| rx_locktorefclk | Input | Lock-to-reference lock mode for the CRU. Use with rx_locktodata.<br>rx_locktodata/rx_locktorefclk<br>0/0 – CRU is in automatic mode<br>0/1 – CRU is in lock-to-reference clock<br>1/0 – CRU is in lock-to-data mode<br>1/1 – CRU is in lock-to-data mode | Channel |
| rx_cruclk | Input | Receiver PLL/CRU reference clock. | Channel |
| **Transmitter PCS** | | | |
| tx_datain | Input | Transmitter parallel data input. The bus width depends on the channel width for the selected functional mode multiplied by the number of channels in the instance. | Channel |
| tx_clkout | Output | PLD logic array clock from the transceiver to the PLD. In an individual-channel mode, there is one tx_clkout per channel. | Channel |
| tx_coreclk | Input | Optional write clock port for the transmitter phase compensation FIFO. If not selected, the Quartus II software automatically selects tx_clkout as the write clock for transmitter phase compensation FIFO. If selected, you must drive this port with a clock that is frequency locked to tx_clkout. | Channel |
| tx_detectrxloopback | Input | PIPE receiver detect / loopback pin. Depending on the power-down state (P0 or P1), the signal either activates receiver detect or loopback. | Channel |
| tx_forceelecidle | Input | PIPE Electrical Idle mode. | Channel |
| tx_forcedispcompliance | Input | PIPE forced negative disparity port for transmission of the compliance pattern. The pattern requires starting at a negative disparity. Assertion of this port at the first byte ensures that the first byte has a negative disparity. This port must be deasserted after the first byte. | Channel |

### Table 1–1. Arria GX ALT2GXB Ports   (Part 5 of 6)

| Port Name | Input/Output | Description | Scope |
|---|---|---|---|
| `powerdn` | Input | PIPE power mode port. This port sets the power mode of the associated PCI Express channel. The power modes are as follows:<br>2'b00: P0 – Normal operation<br>2'b01: P0s – Low recovery time latency, power saving state<br>2'b10: P1 – Longer recovery time (64 μs max) latency, lower power state<br>2'b11: P2 – Lowest power state | Channel |
| `tx_digitalreset` | Input | Reset port for the transmitter PCS block. This port resets all the digital logic in the transmit channel. The minimum pulse width is two parallel clock cycles. | Channel |
| `tx_ctrlenable` | Input | Transmitter control code indicator port. Indicates whether the data at the `tx_datain` port is a control or data word. This port is used with the 8B/10B encoder. | Channel |
| `tx_invpolarity` | Input | Available in all modes. Inverts the polarity of the data to be transmitted at the transmitter PCS-PMA interface (input to the serializer). | Channel |
| `debug_tx_phase_comp_fifo_error` | Output | Indicates transmitter phase compensation FIFO overrun or underrun situation. | Channel |
| **Transmitter PMA** | | | |
| `fixedclk` | Input | 125-MHz clock for receiver detect circuitry in PCI Express (PIPE) mode. | Channel |
| **CMU PMA** | | | |
| `gxb_powerdown` | Input | Transceiver block reset and power down. This resets and powers down all circuits in the transceiver block. This does not affect the `REFCLK` buffers and reference clock lines. | Transceiver block |
| `pll_locked` | Output | PLL locked indicator for the transmitter PLLs. | Transceiver block |
| `pll_inclk` | Input | Reference clocks for the transmitter PLLs. | Transceiver block |
| **Calibration Block** | | | |
| `cal_blk_clk` | Input | Calibration clock for the transceiver termination blocks. This clock supports frequencies from 10 MHz to 125 MHz. | Device |

| Table 1–1. Arria GX ALT2GXB Ports (Part 6 of 6) | | | |
|---|---|---|---|
| **Port Name** | **Input/ Output** | **Description** | **Scope** |
| `cal_blk_powerdown` (active low) | Input | Power-down signal for the calibration block. Assertion of this signal may interrupt data transmission and reception. Use this signal to re-calibrate the termination resistors if temperature and/or voltage changes warrant it. | Device |
| **External Signals** | | | |
| `tx_dataout` | Output | Transmitter serial output port. | Channel |
| `rx_datain` | Input | Receiver serial input port. | Channel |
| `rrefb` *(1)* | Output | Reference resistor port. This port is always used and must be tied to a 2K-Ω resistor to ground. This port is highly sensitive to noise. There must be no noise coupled to this port. | Device |
| `refclk` *(1)* | Input | Dedicated reference clock inputs (two per transceiver block) for the transceiver. The buffer structure is similar to the receiver buffer, but the termination is not calibrated. | Transceiver block |
| `gxb_enable` | Input | Dedicated transceiver block enable pin. If instantiated, this port must be tied to the `pll_ena` input pin. A high level on this signal enables the transceiver block; a low level disables it. | Transceiver block |

*Note to Table 1–1:*
(1)    These are dedicated pins for the transceiver and do not appear in the MegaWizard Plug-In Manager.

## Transmitter Channel Architecture

This section provides a brief description about sub-blocks within the transmitter channel (shown in Figure 1–2). The sub-blocks are described in order from the PLD-transmitter parallel interface to the serial transmitter buffer.

*Figure 1–2. Arria GX Transmitter Channel Block Diagram*
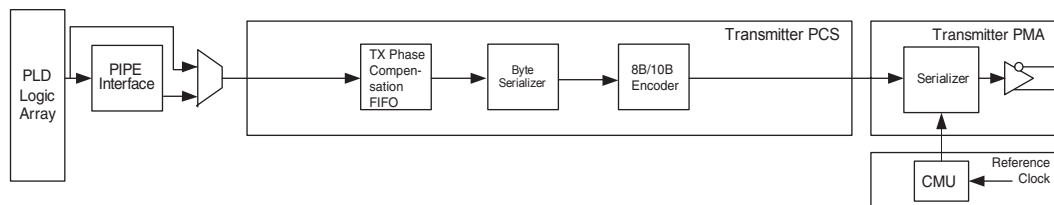
## Clock Multiplier Unit

Each transceiver block has a clock multiplier unit (CMU) that takes in a reference clock and synthesizes two clocks: a high-speed serial clock to serialize the data and a low-speed parallel clock used to clock the transmitter digital logic (PCS) and the PLD-transceiver interface.

The CMU is further divided into three sub-blocks

■ Transmitter PLL
■ Central clock divider block
■ Local clock divider block

Each transceiver block has one transmitter PLL, one central clock divider and four local clock dividers. One local clock divider is located in each transmitter channel of the transceiver block.

Figure 1–3 shows a block diagram of the CMU block within each transceiver block.

*Figure 1–3. Clock Multiplier Unit Block Diagram*
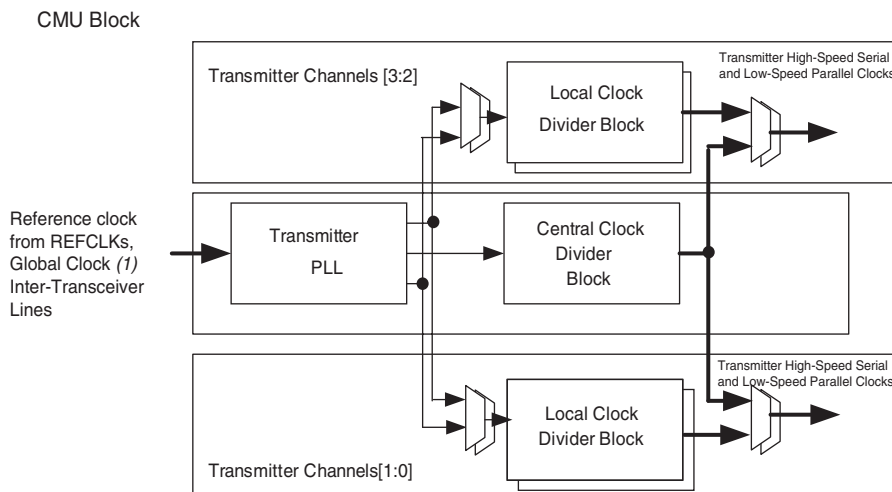


*Note to Figure 1–3:*
(1)     The global clock line must be driven from an input pin only.

*Transmitter PLL*

The transmitter PLL multiplies the input reference clock to generate the high-speed serial clock required to support the intended protocol. It synthesizes a half-rate high-speed serial clock that runs at half the frequency of the serial data rate for which it is configured; for example, the transmitter PLL runs at 625 MHz when configured in 1.25-Gbps GIGE functional mode.

The transmitter PLL output feeds the central clock divider block and the local clock divider blocks. These clock divider blocks divide the high-speed serial clock to generate the low-speed parallel clock for the transceiver PCS logic and the PLD-transceiver interface clock. Depending on the functional mode for which the transceiver block is configured, either the central clock divider block or the local clock divider block is used to generate the low-speed parallel clock.

Figure 1–4 shows a block diagram of the transmitter PLL.

*Figure 1–4. Transmitter PLL*
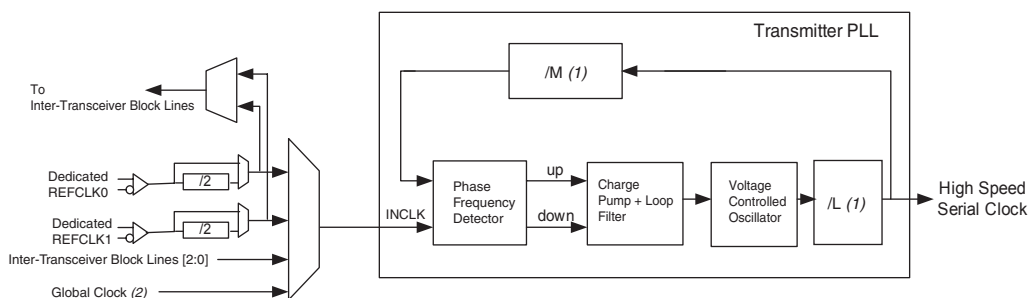


*Notes to Figure 1–4:*
(1) You only need to select the protocol and the available input reference clock frequency in the Quartus II MegaWizard Plug-In Manager. Based on your selections, the MegaWizard Plug-In Manager automatically selects the necessary /M and /L dividers (clock multiplication factors).
(2) The global clock line must be driven from an input pin only.

The reference clock input to the transmitter PLL can be derived from:

■ One of the two available dedicated reference clock input pins (REFCLK0 or REFCLK1) of the associated transceiver block
■ PLD global clock network (must be driven directly from an input clock pin and cannot be driven by user logic or enhanced PLL)
■ Inter-transceiver block lines driven by reference clock input pins of other transceiver blocks

☞ Altera recommends using the dedicated reference clock input pins (REFCLK0 or REFCLK1) to provide the reference clock for the transmitter PLL.

**Transmitter PLL Bandwidth Setting**

The Arria GX transmitter PLLs in the transceiver offer a programmable bandwidth setting. The bandwidth of a PLL is the measure of its ability to track the input clock and jitter. It is determined by the -3dB frequency of the closed-loop gain of the PLL.

There are three bandwidth settings: high, medium, and low. The high bandwidth setting filters out internal noise from the $V_{CO}$ because it tracks the input clock above the frequency of the internal $V_{CO}$ noise. With the low bandwidth setting, if the noise on the input reference clock is greater than the internal noise of the $V_{CO}$, the PLL filters out the noise above the -3dB frequency of the closed-loop gain of the PLL. The medium bandwidth setting is a compromise between the high and low settings.

The -3dB frequencies for these settings can vary because of the non-linear nature and frequency dependencies of the circuit.

**Dedicated Reference Clock Input Pins**

Each transceiver block has two dedicated reference clock input pins (REFCLK0 and REFCLK1). The clock route from REFCLK0 and REFCLK1 pins in each transceiver block has an optional pre-divider that divides the reference clock by two before feeding it to the transmitter PLL (shown in Figure 1–4). The refclk pre-divider is required if one of the following conditions is satisfied:

■ If the input clock frequency is greater than 325 MHz.
■ For functional modes with a data rate less than 3.125 Gbps (the data rate is specified in the **what is the data rate?** option in the **General** tab of the ALT2GXB MegaWizard):
  ● If the input clock frequency is greater than or equal to 100 MHz AND
  ● If the ratio of data rate to input clock frequency is 4, 5, or 25

**Reference Clock From PLD Global Clock Network**

You can drive the reference clock to the transmitter PLL from a PLD global clock network. If you choose this option, you must drive the global PLD reference clock line from a non-REFCLK FPGA input pin. You cannot use a clock generated by PLD logic or an enhanced PLL to drive the reference clock input to the transmitter PLL.

☞ The Quartus II software requires the following setting for the non-`REFCLK` FPGA input pin used to drive the reference clock input:

Assignment name: *Stratix II GX/Arria GX REFCLK coupling and termination setting*

Value: *Use as regular IO*.

**Inter-Transceiver Block Line Routing**
The inter-transceiver block lines allow the dedicated reference clock input pins of one transceiver block to drive the transmitter and receiver PLL of other transceiver blocks. There are a maximum of three inter-transceiver block routing lines available in the Arria GX device family. Each transceiver block can drive one inter-transceiver block line from either one of its associated reference clock pins. The inter-transceiver block lines can drive any or all of the transmitter and receiver PLLs in the device. The inter-transceiver block lines offer flexibility when multiple channels in separate transceiver blocks share a common reference clock frequency.

The inter-transceiver block lines also drive the reference clock from the `REFCLK` pins into the PLD fabric, which reduces the need to drive multiple clocks of the same frequency into the device. If a divide-by-two reference clock pre-divider is used, the inter-transceiver block line driven by the corresponding `REFCLK` pin cannot be used to clock PLD logic.

The Quartus II software automatically uses the appropriate inter-transceiver line if the transceiver block is being clocked by the dedicated reference clock (`REFCLK`) pin of another transceiver block.

Figure 1–5 shows the inter-transceiver block line interface to the transceivers in the gigabit transceiver blocks and to the PLD.

*Figure 1–5. Inter-Transceiver Block Line Routing*
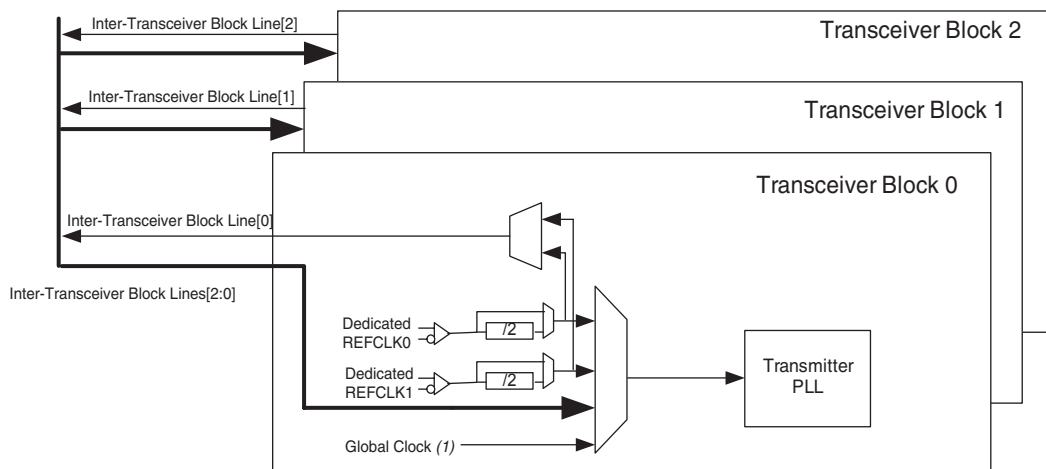


*Note to Figure 1–5:*
(1)    The global clock line must be driven from an input pin only.

☞    Depending on the functional mode, the Quartus II software automatically selects the appropriate transmitter PLL bandwidth.

### Central Clock Divider Block

The central clock divider block is located in the central block of the transceiver block (refer to Figure 1–6). This block provides the high-speed clock for the serializer and the low-speed clock for the transceiver's PCS logic within the transceiver block in a four-lane mode.

Figure 1–6 shows the central clock divider block. The /4 and /5 block generates the slow-speed clock based on the serialization factor. The high-speed clock goes directly into each channel's serializer.

**Figure 1–6. Central Clock Divider Block**
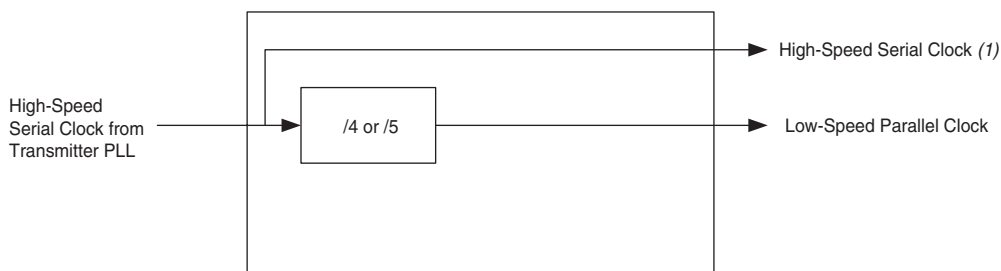


*Notes to Figure 1–6:*
(1)   This feeds the PCS logic.

The central clock divider block feeds all the channels in the transceiver block when in PIPE ×4 mode. This ensures that the serializer in each channel outputs the same bit number at the same time and minimizes the channel-to-channel skew.

### Transmitter Local Clock Divider Block

The Tx local clock divider blocks are located in each transmitter channel of the transceiver block. The purpose of this block is to provide the high-speed clock for the serializer and the low-speed clock for the transmitter data path and the PLD for all the transmitters within the transceiver block. This allows for each of the transmitter channels to run at different rates. The /n divider offers /1, /2, and /4 factors to provide capability to reduce base frequency of the driving PLL to half or a quarter rate. This allows each transmitter channel to run at /1, /2, or /4 of the original data rate.

Figure 1–7 shows the transmitter local clock divider block.

**Figure 1–7. Transmitter Local Clock Divider Block**
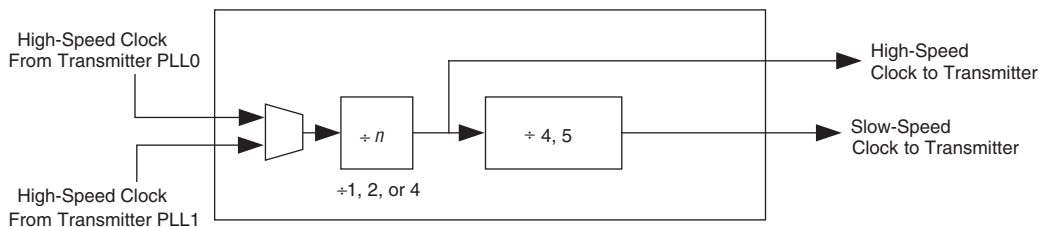
Each transmitter local clock divider block is operated independently so there is no guarantee that each channel sends out the same bit at the same time.

### Clock Synthesis

Each PLL in a transceiver block receives a reference clock and generates a high-speed clock that is forwarded to the clock generator blocks. There are two types of clock generators:

■ Transmitter local clock divider block
■ Central clock divider block

The transmitter local clock divider block resides in the transmit channel and synthesizes the high-speed serial clock (used by the serializer) and slow-speed clock (used by the transmitter's PCS logic). The central clock divider block resides in the transceiver block outside the transmit or receive channels. This block synthesizes the high-speed serial clock (used by the serializer) and slow-speed clock (used by the transceiver block PCS logic—transmitter and receiver (if the rate matcher is used)). The PLD clock is also supplied by the central clock divider block and goes through the divide-by-two block (located in the central block of the transceiver block) if the byte serializer/deserializer is used.

The PLLs in the transceiver have half rate voltage-controlled oscillators (VCOs) that run at half the rate of the data stream. When in the individual channel mode, the slow-speed clocks for the transmitter logic and the serializer need only be a /4, or a /5 divider to support a ×8 and ×10 serialization factor. Table 1–2 shows the divider settings for achieving the available serialization factor.

| Table 1–2. Serialization Factor and Divider Settings | |
|---|---|
| **Serialization Factor** | **Divider Setting** |
| ×8 | /4 |
| ×10 | /5 |

In the four-lane mode, the central clock divider block supplies all the necessary clocks for the entire transceiver block.

The reference clock ranges from 50 MHz to 622.08 MHz. The phase frequency detector (PFD) has a minimum frequency limit of 50 MHz and a maximum frequency limit of 325 MHz.

The refclk pre-divider (/2 ) is available if you use the dedicated refclk pins for the input reference clock. The refclk pre-divider is required if one of the following conditions is satisfied:

■ If the input clock frequency is greater than 325 MHz.
■ For functional modes with a data rate less than 3.125 Gbps (the data rate is specified in the **what is the data rate?** option in the **General** tab of the ALT2GXB MegaWizard):
   ● If the input clock frequency is greater than or equal to 100 MHz AND
   ● If the ratio of data rate to input clock frequency is 4, 5, or 25

### Transceiver Clock Distribution

This section describes single lane and four-lane configurations for the high speed and low speed transceiver clocks. All protocol support falls in the single lane configuration except for the four-lane PIPE mode and XAUI. The four-lane PIPE mode uses the four-lane configuration.

**Single Lane**
In a single lane configuration, the PLLs in the central block supply the high speed clock. Then the clock generation blocks in each transmitter channel divides down the high speed clock to the frequency needed to support its particular data rate. In this configuration, two separate clocks can be supplied through the central block to provide support for two separate base frequencies. The transmitter clock generation blocks can divide those down to create additional frequencies for specific data rate requirements. Each of the four transmitter channels can operate at a different data rate with the use of the individual transmitter local clock dividers and both Transmitter PLL0 and Transmitter PLL1.

☞ If you instantiate four channels and are not in PIPE ×4, XAUI, or Basic single-width mode with ×4 clocking, the Quartus II software automatically chooses the single lane configuration.

Figure 1–8 shows clock distribution for individual channel configuration.

*Figure 1–8. Clock Distribution for Individual Channel Configuration*
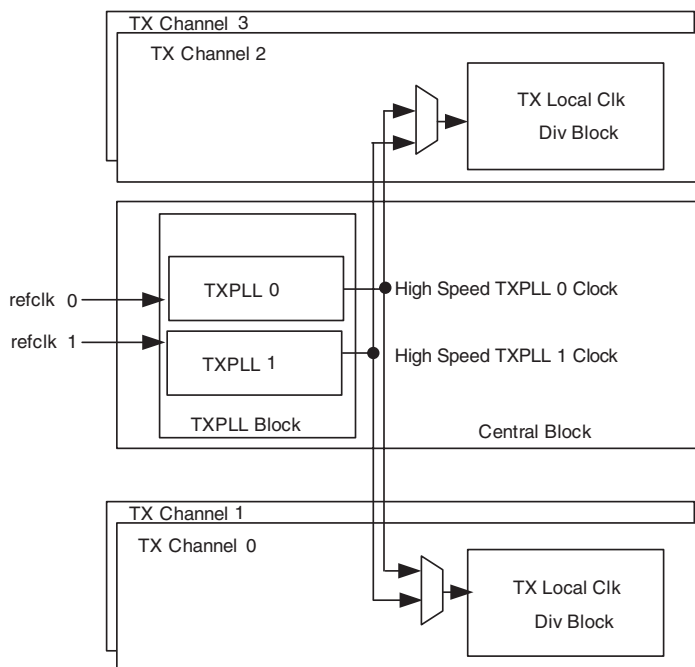


**Four-Lane Mode**

In a four-lane configuration (shown in Figure 1–9), the central block generates the parallel and serial clocks that feed the transmitter channels within the transceiver. All channels in a transceiver must operate at the same data rate. This configuration is only supported in PIPE ×4, XAUI and Basic mode with ×4 clocking.

*Figure 1–9. Clock Distribution for a Four-Lane Configuration Note (1)*
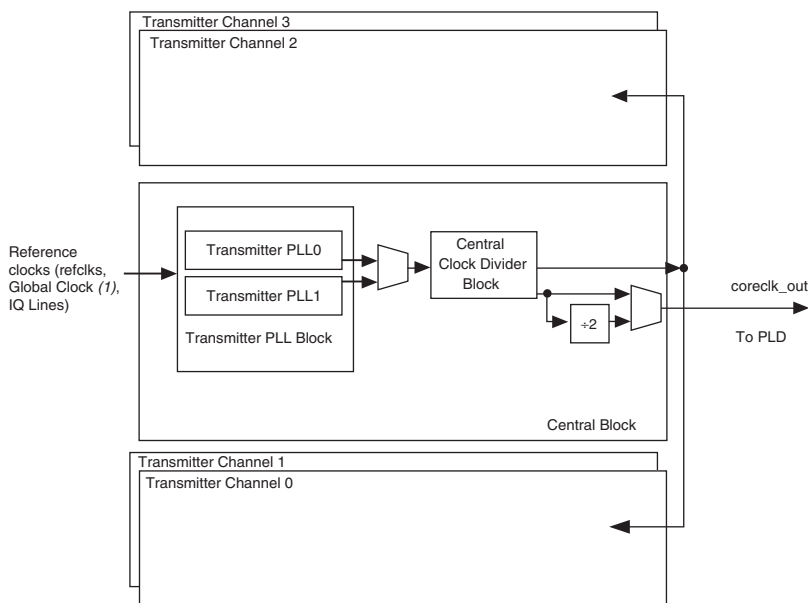


*Note to Figure 1–9:*
(1)    The global clock line must be driven by an input pin.

Figure 1–10 shows how single transceiver block devices EP1AGX20CF, EP1AGX35CF, EP1AGX50CF and EP1AGX60CF devices are configured for PCI-E ×4 mode. When ArriaGX devices are used in ×4 bonded mode for PCI-E, physical Lane 0 of the transmitter should be connected to physical Lane 0 of the receiver and vice versa.

*Figure 1–10. Two Transceiver Block Device with One ×4 PCI-E Link*
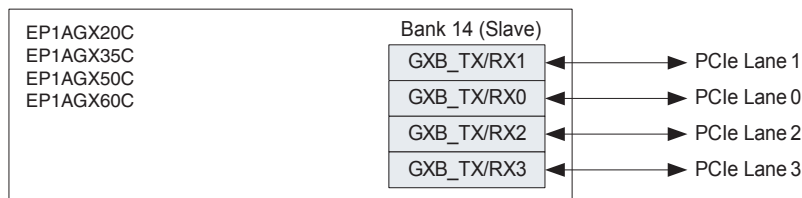


The two transceiver block devices EP1AGX35DF, EP1AGX50DF, and EP1AGX60DF support only two PCI-E ×4 links. Fig Figure 1–11shows the PCI-E ×4 configuration.

*Figure 1–11. Two Transceiver Block Device with Two ×4 PCI-E Links*
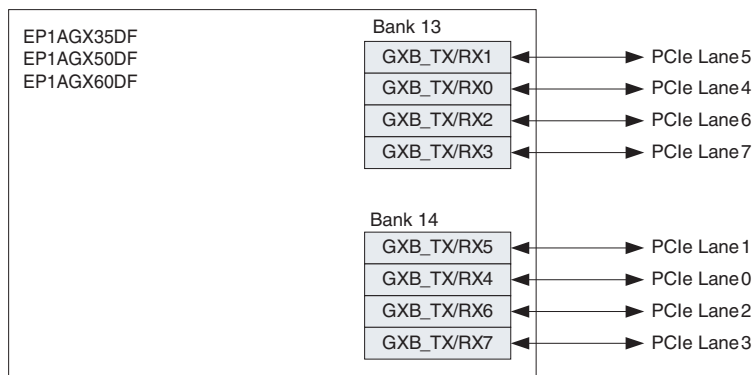


The three transceiver block devices EP1AGX60EF and EP1AGX90EF support up to three PCI-E ×4 links. Figure 1–12 shows the PCI-E ×4 configuration.

*Figure 1–12. Three Transceiver Block Device with Three ×4 PCI-E Links*
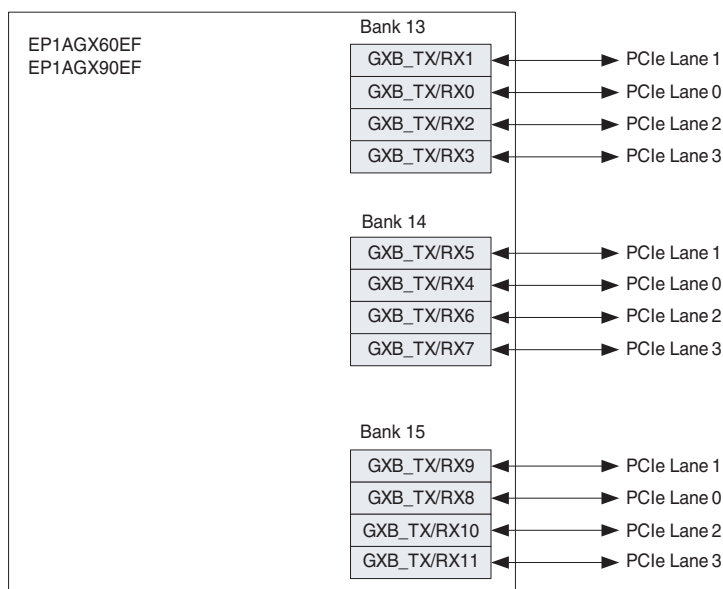
### Channel Clock Distribution

This section describes clocking within each channel for:

■ Individual channels in Basic (without ×4 clocking enabled), PIPE ×1, GIGE, Serial RapidIO, and SDI modes

■ Bonded channels in XAUI, PIPE ×4, and Basic (with ×4 clocking enabled) modes

**Individual Channels Clocking**
In individual channel modes, the transmitter logic is clocked by the slow speed clock from the clock divider block. The transmitter phase compensation FIFO buffer and the PIPE interface (in PIPE mode) are clocked by the tx_clkout clock of the channel that is fed back to the transmitter channel from the PLD logic. Figure 1–13 shows the clock routing for the transmitter channel.

*Figure 1–13. Individual Channel Transmitter Logic Clocking*
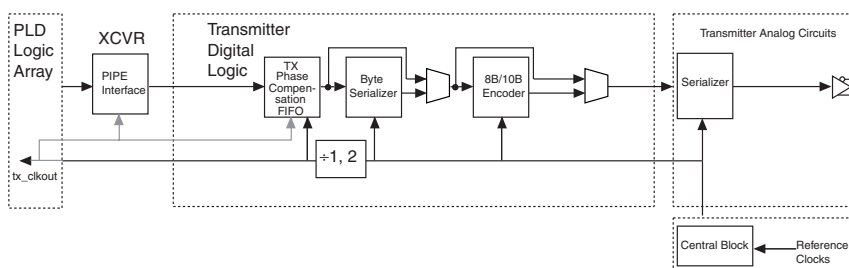


The receiver logic clocking has two clocking methods: one when rate matching is used and the other when rate matching is not used.

If rate matching is used (PIPE, GIGE, and Basic modes), the receiver logic from the serializer to the rate matcher is clocked by the recovered clock from its associated channel. The rest of the logic is clocked by the slow clock from the clock divider block of its associated channel. The read side of the phase compensation FIFO buffer and the PIPE interface (for PIPE mode) is clocked by the tx_clkout fed back through the PLD logic. Figure 1–14 shows the clocking of the receiver logic with the rate matcher.

*Figure 1–14. Individual Channel Receiver Logic Clocking with Rate Matching*
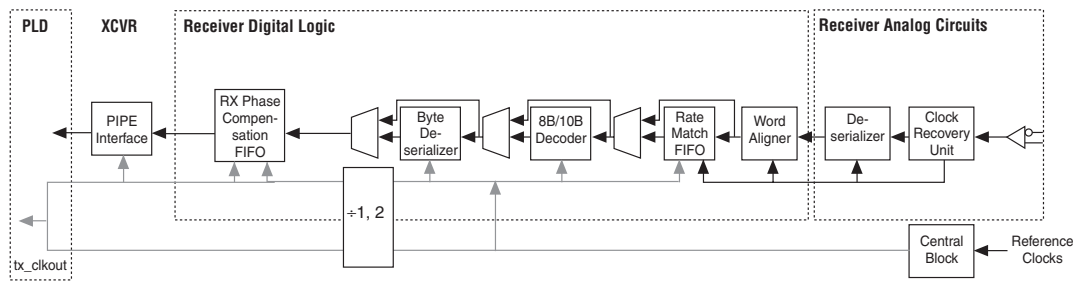


If rate matching is not used (Basic, SDI, and Serial RapidIO modes), then the receiver logic is clocked by the recovered clock of its associated channel (Figure 1–15). The receiver phase compensation FIFO buffer's read port is clocked by the recovered clock that is fed back from the PLD logic array as rx_clkout.

*Figure 1–15. Individual Channel Receiver Logic Clocking Without Rate Matching*
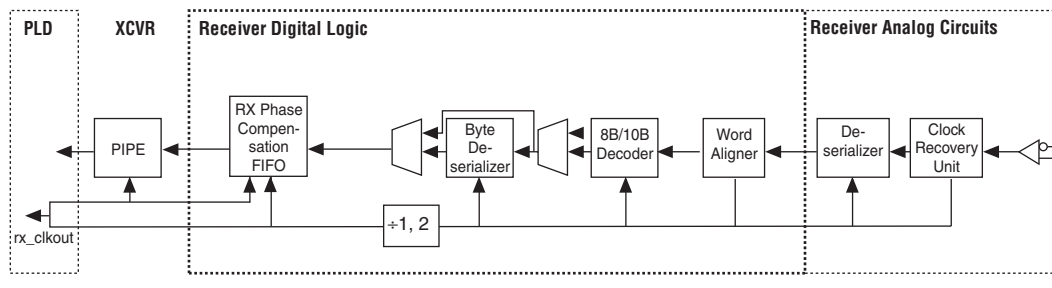


**Transmitter Clocking (Bonded Channels)**

The clocking in bonded channel modes (Figure 1–16) is different from that of the individual channel. All the transmitters are synchronized to the same transmitter PLL and clock divider from the central block. In ×4 bonded channel modes, the central clock divider of the transceiver block clocks all four channels.

The transmitter logic up to the read port of the transmitter phase compensation FIFO buffer is clocked by the slow speed clock from the central block. The PIPE interface and the write port of the transmitter phase compensation FIFO buffer is clocked by the coreclkout signal routed from the PLD.

*Figure 1–16. Transmitter Channel Clocking in Transceiver Mode*
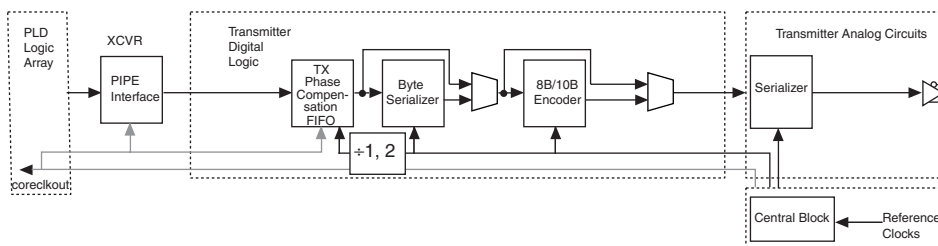


For the receiver logic, in XAUI mode (Figure 1–17), the local recovered clock feeds the logic up to the write clock of the deskew FIFO buffer. The recovered clock from Channel 0 feeds the read clock of the deskew FIFO buffer and the write port of the rate matcher. The slow clock from the central block feeds the rest of the logic up to the write port of the phase compensation FIFO buffer. The coreclkout signal routed through the PLD from the central block feeds the read side of the phase compensation FIFO buffer.

*Figure 1–17. Receiver Channel Clocking in XAUI Mode*
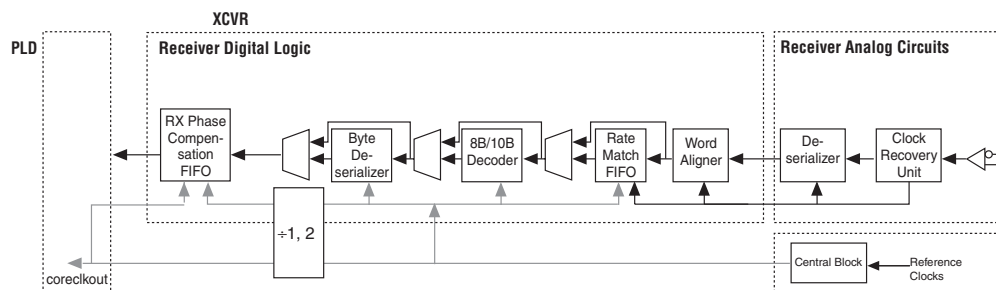


In the PIPE ×4 mode (Figure 1–18), the local recovered clock feeds the logic up to the write port of the rate matcher FIFO buffer. The slow clock from the central block feeds the rest of the logic up to the write port of the phase compensation FIFO buffer. The coreclkout signal routed through the PLD from the central block feeds the read side of the phase compensation FIFO buffer.

*Figure 1–18. Receiver Channel PIPE 4 Mode*



*Figure 1–18. Receiver Channel PIPE 4 Mode*
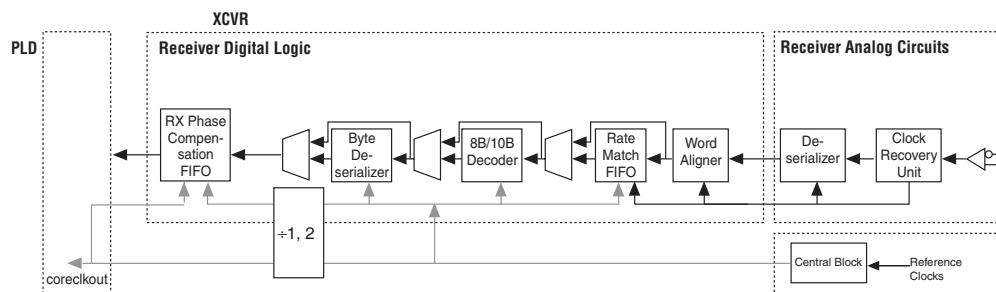
## Transmitter Phase Compensation FIFO

A transmitter phase compensation FIFO (Figure 1–19) is located at each transmitter channel's logic array interface. It compensates for the phase difference between the transmitter PCS clock and the local PLD clock.

In individual channel mode (for example, GIGE and Serial RapidIO), the low-speed parallel clock (or its divide-by-two version if the byte serializer is used) from the local clock divider block of each channel clocks the read port of its transmitter phase compensation FIFO buffer. This clock is also forwarded to the logic array on `tx_clkout` port of its associated channel. If the `tx_coreclk` port is not instantiated, the clock signal on the `tx_clkout` port of Channel 0 is automatically fed back to clock the write port of the transmitter phase compensation FIFOs in all channels within the transceiver block. If the `tx_coreclk` port is instantiated, the clock signal driven on the `tx_coreclk` port clocks the write port of the transmitter phase compensation FIFO of its associated channel. You must ensure that the clock on the `tx_coreclk` port is frequency locked to the read clock of the transmitter phase compensation FIFO. For more information about using the PLD core clock (`tx_coreclk`), refer to "PLD-Transceiver Interface Clocking" on page 1–68.

In bonded channel mode (for example, ×4 PCI Express (PIPE)), the low speed parallel clock from the central clock divider block is divided by two. This divide-by-two clock clocks the read port of the transmitter phase compensation FIFO. This clock is also forwarded to the logic array on the `coreclkout` port. If the `tx_coreclk` port is not instantiated, the clock signal on the `coreclkout` port is automatically fed back to clock the write port of transmitter phase compensation FIFO buffers in all channels within the transceiver block. If the `tx_coreclk` port is instantiated, the clock signal driven on the `tx_coreclk` port clocks the write port of the transmitter phase compensation FIFO of its associated channel. You must ensure that the clock on the `tx_coreclk` port is

frequency locked to the read clock of the transmitter phase compensation FIFO. For more information about using the PLD core clock (`tx_coreclk`), refer to "PLD-Transceiver Interface Clocking" on page 1–68.

*Figure 1–19. Transmitter Phase Compensation FIFO*
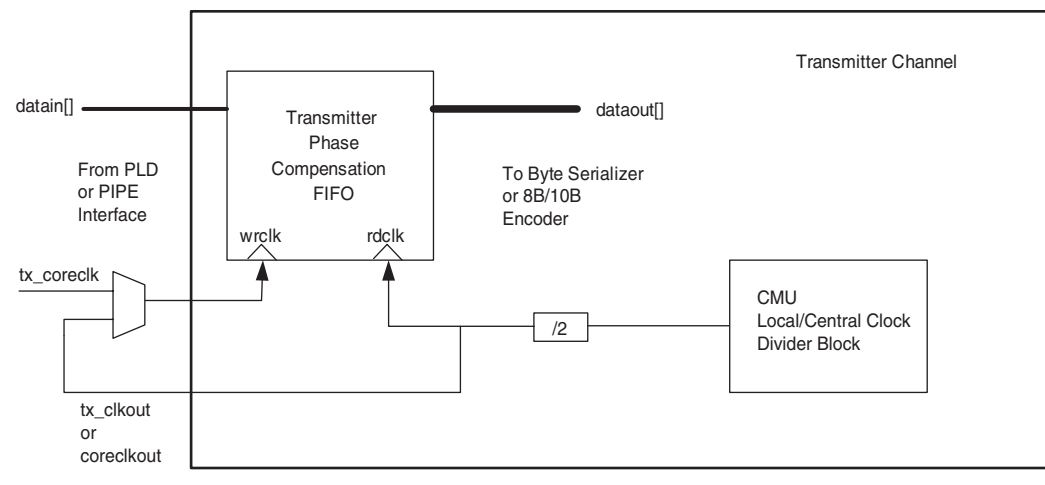


### Transmitter Phase Compensation FIFO Error Flag

The write port of the transmitter phase compensation FIFO can be clocked by either the CMU output clock or its divide-by-two version (`tx_clkout` or `coreclkout`) or a PLD clock. The read port is always clocked by the CMU output clock or its divide-by-two version. In all configurations, the write clock and the read clock must have 0 parts per million (PPM) difference to avoid overrun/underflow of the phase compensation FIFO.

An optional `debug_tx_phase_comp_fifo_error` port is available in all modes to indicate transmitter phase compensation FIFO overrun/underflow condition. This feature should be used for debug purposes only if link errors are observed.

## Byte Serializer

The byte serializer (Figure 1–20) takes in 16- or 20-bit wide data from the transmitter phase compensation FIFO buffer and serializes it into 8- or 10-bit wide data at twice the speed. This allows clocking the PLD-transceiver interface at half the speed as compared to the transmitter PCS logic. The byte serializer is bypassed in GIGE mode.

*Figure 1–20. Byte Serializer Note (1)*
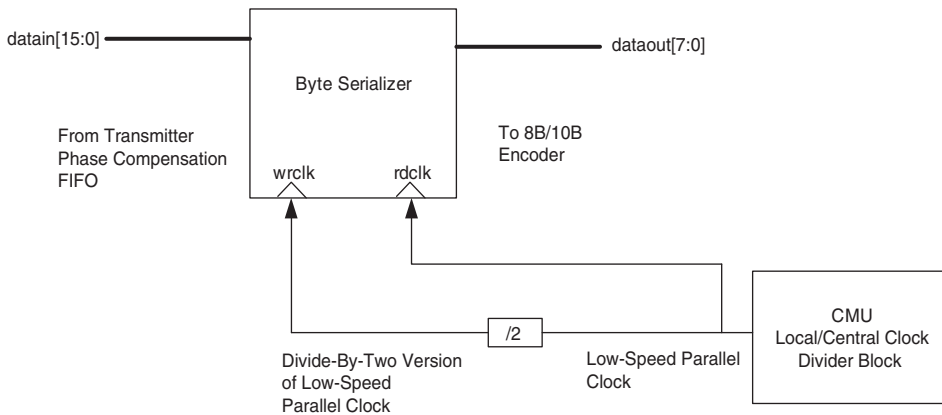


*Note to Figure 1–20:*
(1)  `datain` and `dataout` may also be 20 bits and 10 bits wide, respectively.

After serialization, the byte serializer transmits the least significant byte (LSByte) first and the most significant byte (MSByte) last.

Figure 1–21 shows byte serializer input and output. `datain[15:0]` is the input to the byte serializer from the transmitter phase compensation FIFO and `dataout[7:0]` is the output of the byte serializer. `datain` may also be 20 bits wide and `dataout` may be 10 bits wide depending on implementation.

*Figure 1–21. Byte Serializer Operation*
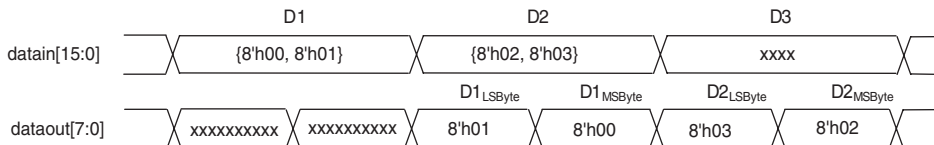


In Figure 1–21, the LSByte is transmitted before the MSByte from the transmitter byte serializer. For input data D1, the output data is $D1_{LSByte}$ and then $D1_{MSByte}$.

## 8B/10B Encoder

The 8B/10B encoder block takes in 8-bit data from the byte serializer or transmitter phase compensation FIFO buffer (if the byte serializer is not used). It generates a 10-bit code group with proper running disparity from the 8-bit character and a 1-bit control identifier (tx_ctrlenable). The 10-bit code group is fed to the serializer. The 8B/10B encoder conforms to the IEEE 802.3 1998 edition standard.

Figure 1–22 shows the 8B/10B conversion format.

For additional information about 8B/10B encoding rules, refer to the *Specifications and Additional Information* chapter in volume 2 of the *Arria GX Device Handbook*.

*Figure 1–22. 8B/10B Encoder*
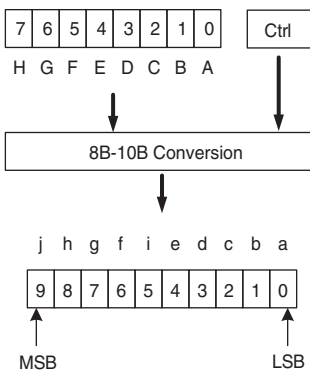


The 10-bit encoded data output from the 8B/10B encoder is fed to the serializer that transmits the data from LSB to MSB.

### Reset Behavior

The transmitter digital reset (tx_digitalreset) signal resets the 8B/10B encoder. During reset, the running disparity and data registers are cleared and the 8B/10B encoder outputs a K28.5 pattern from the RD- column continuously. Once out of reset, the 8B/10B encoder starts with a negative disparity (RD-) and transmits three K28.5 code groups for synchronizing before it starts encoding the input data or control character.

Figure 1–23 shows the 8B/10B encoder's reset behavior. When in reset (`tx_digitalreset` is high), a K28.5- (K28.5 10-bit code group from the RD- column) is sent continuously until `tx_digitalreset` is low. The transmitter channel pipelining causes some "don't cares (10'hxxx)" until the first of three K28.5 is sent. User data follows the third K28.5.

*Figure 1–23. 8B/10B Encoder Output During Reset*
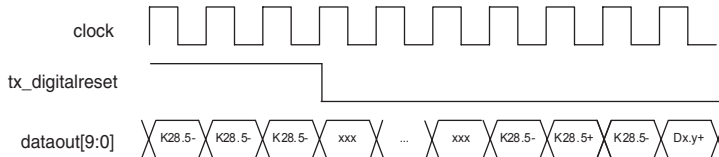


### Control Code Group Encoding

A control identifier (`tx_ctrlenable`) input signal specifies whether the 8-bit input character is to be encoded as a control word (Kx.y) or data word (Dx.y). When `tx_ctrlenable` is low, the input character is encoded as data (Dx.y). When `tx_ctrlenable` is high, the input character is encoded as a control word (Kx.y). The waveform in Figure 1–24 shows that the second 0xBC character is encoded as a control word (K28.5). The rest of the characters are encoded as data (Dx.y).

*Figure 1–24. Control Code Group Identification*
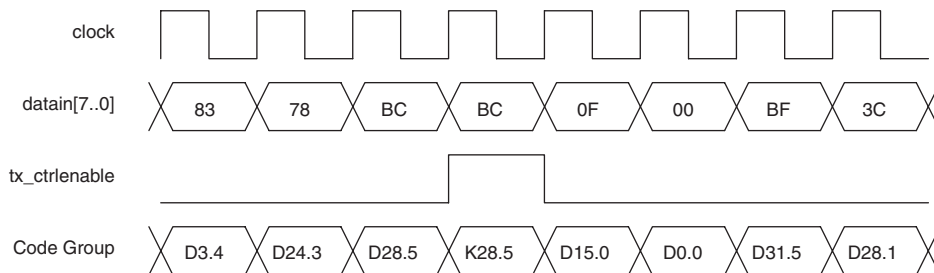


☞ The 8B/10B encoder does not check whether the code group word entered is one of the 12 valid codes. If you enter an invalid control code, the resultant 10-bit code group may be encoded as an invalid code (does not map to a valid Dx.y or Kx.y code group), or unintended valid Dx.y code group, depending on the value entered.

*Transmitter Force Disparity*

Upon power on or reset, the 8B/10B encoder has a negative disparity and chooses the 10-bit code from the RD- column. The Transmitter Force Disparity feature allows altering the running disparity via the `tx_forcedisp` and `tx_dispval` ports.

Two optional ports, `tx_forcedisp` and `tx_dispval`, are available in 8B/10B enabled Basic mode. A high value on the `tx_forcedisp` bit will change the disparity value of the data to the value indicated by the associated `tx_dispval` bit. If the `tx_forcedisp` bit is low, then `tx_dispval` is ignored and the current running disparity is not altered. Forcing disparity can either maintain the current running disparity calculations if the forced disparity value (on the `tx_dispval` bit) happens to match the current running disparity, or flip the current running disparity calculations if it does not. If the forced disparity flips the current running disparity, the downstream 8B/10B decoder may detect a disparity error that should be tolerated by the downstream device.

Figure 1–25 shows the current running disparity being altered in Basic mode by forcing a positive disparity on a negative disparity K28.5. In this example, a series of K28.5 code groups are continuously being sent. The stream alternates between a positive ending running disparity (RD+) K28.5 and a negative ending running disparity (RD-) K28.5 as governed by the 8B/10B encoder to maintain a neutral overall disparity. The current running disparity at time n+3 indicates that the K28.5 in time n+4 should be encoded with a negative disparity. Since the `tx_forcedisp` is high at time n+4, and `tx_dispval` is also high, the K28.5 at time n+4 is encoded as a positive disparity code group. As the `tx_forcedisp` is low at n+5, the K28.5 will take the current running disparity of n+4 and encode the K28.5 in time n+5 with a negative disparity. If the `tx_forcedisp` were driven high at time n+5, that K28.5 would also be encoded with positive disparity.

*Figure 1–25. Transmitter Force Disparity Feature in Basic Mode*
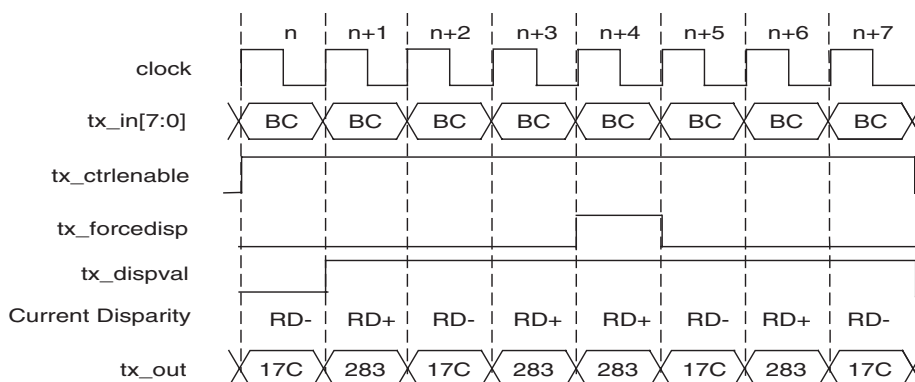


*Transmitter Polarity Inversion*

The positive and negative signals of a serial differential link might accidentally be swapped during board layout. Solutions such as a board re-spin or major updates to the PLD logic can prove expensive. The transmitter polarity inversion feature is provided to correct this situation.

An optional `tx_invpolarity` port is available in all modes to dynamically enable the transmitter polarity inversion feature. A high on the `tx_invpolarity` port inverts the polarity of every bit of the 8- or 10-bit input data word to the serializer in the transmitter data path. Since inverting the polarity of each bit has the same effect as swapping the positive and negative signals of the differential link, correct data is seen by the receiver. The `tx_invpolarity` is a dynamic signal and may cause initial disparity errors at the receiver of an 8B/10B encoded link. The downstream system must be able to tolerate these disparity errors.

Figure 1–26 illustrates the transmitter polarity inversion feature in a 10-bit wide data path configuration.

*Figure 1–26. Transmitter Polarity Inversion*
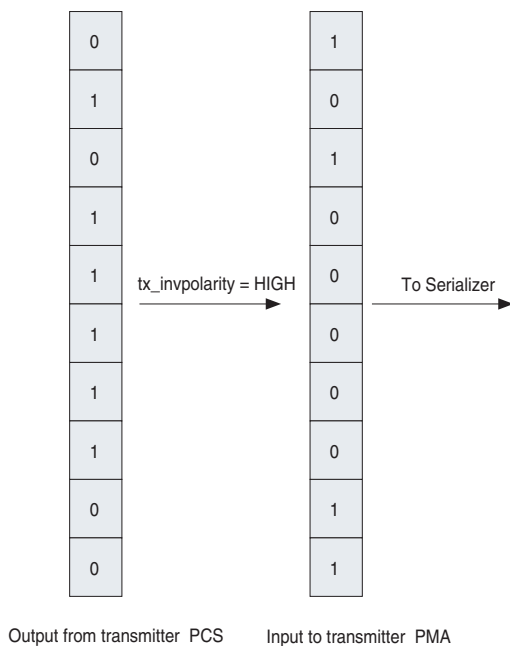


Transmitter Bit Reversal

By default, the Arria GX transmitted bit order is LSBit to MSBit. In Basic mode, the least significant bit of the 8/10-bit data word is transmitted first and the most significant bit is transmitted last. The Transmitter Bit Reversal feature allows reversing the transmitted bit order as MSBit to LSBit.

If the Transmitter Bit Reversal feature is enabled in Basic mode, the 8-bit D[7:0] or 10-bit D[9:0] data at the input of the serializer gets rewired to D[0:7] or D[0:9], respectively. Flipping the parallel data using this feature and transmitting LSBit to MSBit effectively provides MSBit to LSBit transmission.

Figure 1–27 illustrates the transmitter bit reversal feature in a Basic mode 10-bit wide data path configuration.

*Figure 1–27. Transmitter Bit Reversal in Basic Mode*



| D[9] | | D[0] |
|------|---|------|
| D[8] | | D[1] |
| D[7] | | D[2] |
| D[6] | | D[3] |
| D[5] | TX Bit Reversal = Enabled | D[4] | To Serializer |
| D[4] | | D[5] |
| D[3] | | D[6] |
| D[2] | | D[7] |
| D[1] | | D[8] |
| D[0] | | D[9] |

Output from transmitter  PCS          Input to transmitter PMA
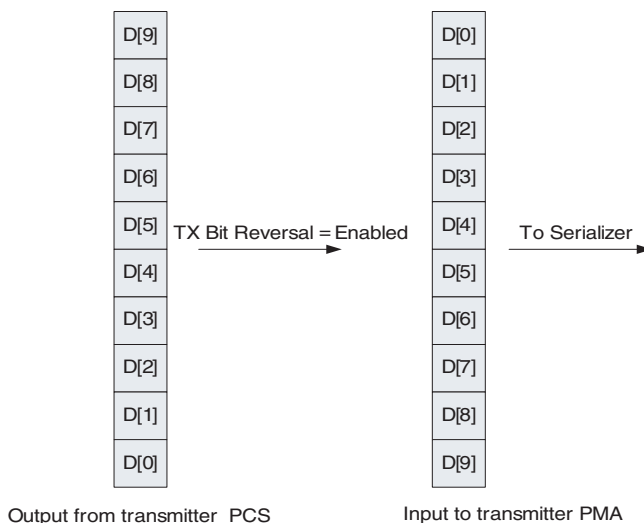
## Serializer

The serializer block clocks in 8- or 10-bit data using the low-speed parallel clock and clocks out serial data using the high-speed serial clock from the central or local clock divider blocks. The serializer natively feeds the data LSB to MSB to the transmitter output buffer.

Figure 1–28 shows the serializer block diagram.

*Figure 1–28. Serializer*
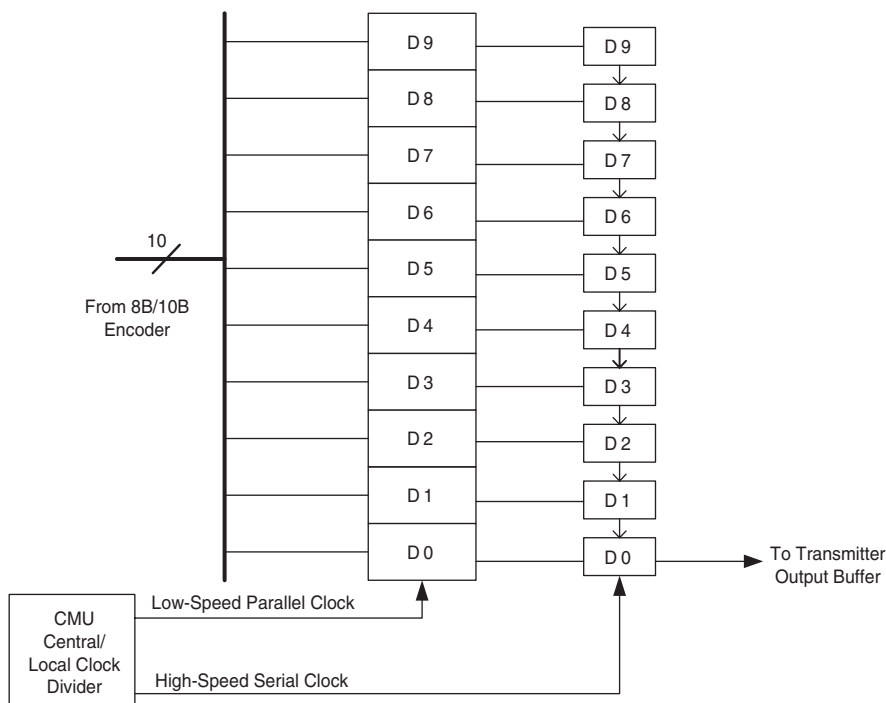


Figure 1–29 shows the serial bit order at the serializer output. In this example, 10'b17C data is serialized and transmitted from LSB to MSB.

*Figure 1–29. Serializer Bit Order*
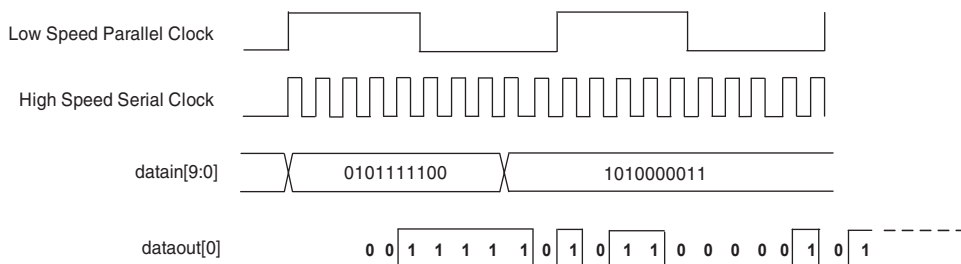
## Transmitter Buffer

The Arria GX transmitter buffers support 1.2-V and 1.5-V pseudo current mode logic (PCML) up to 3.125 Gbps and can drive 40 inches of FR4 trace across two connectors. The transmitter buffer (refer to Figure 1–30) has additional circuitry to improve signal integrity-programmable output voltage, programmable pre-emphasis circuit, and internal termination circuitry-and the capability to detect the presence of a downstream receiver. The Arria GX transmitter buffer supports a common mode of 600 or 700 mV.

*Figure 1–30. Transmitter Buffer*
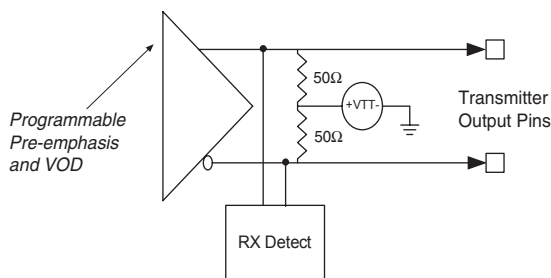


### Programmable Voltage Output Differential

Arria GX devices allow you to customize the differential output voltage (VOD) to handle different trace lengths, various backplanes, and receiver requirements (refer to Figure 1–31). You select the VOD from a range between 400 and 1200 mV, as shown in Table 1–3.

*Figure 1–31. VOD (Differential) Signal Level*

**Single-Ended Waveform**

$V_A$

$\pm V_{OD}$

$V_B$

**Differential Waveform**

+600

$+V_{OD}$

0-V Differential

$V_{OD}$

$-V_{OD}$

-600

$V_{OD}$ (Differential)

$= V_A - V_B$
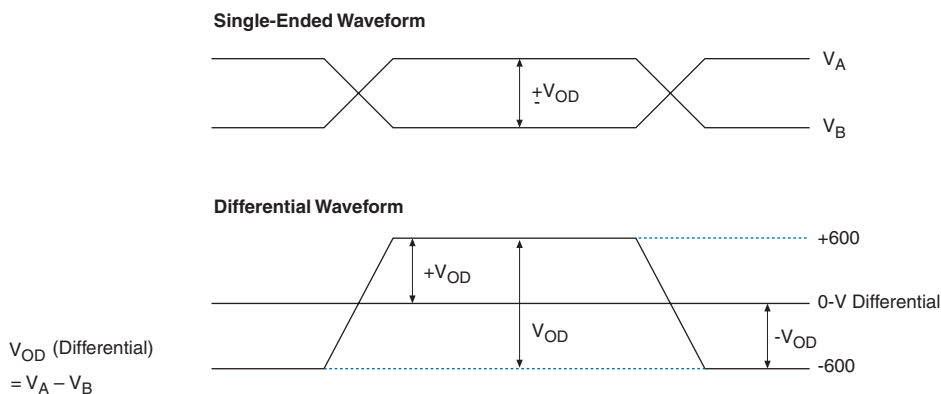
Table 1–3 shows the VOD setting per supply voltage for an on-chip termination value of 100 Ω .

| Table 1–3. VOD Differential Peak to Peak | |
|---|---|
| **1.2-V $V_{CC}$** | **1.5-V $V_{CC}$** |
| **100-Ω (mV)** | **100-Ω (mV)** |
| — | 400 |
| 480 | 600 |
| 640 | 800 |
| 800 | 1000 |
| 960 | 1200 |

You set the VOD values in the MegaWizard Plug-In Manager.

The transmitter buffer is powered by either a 1.2-V or a 1.5-V power supply. You choose the transmitter buffer power ($V_{CCH}$) of 1.2 V or 1.5 V through the ALT2GXB MegaWizard Plug-In Manager (the **What is the transmit buffer power ($V_{CCH}$)?** option). The transmitter buffer power supply in Arria GX devices is transceiver-based. The 1.2 V power supply supports the 1.2-V PCML standard.

You specify the static VOD settings through the ALT2GXB MegaWizard Plug-In Manager.

### Programmable Pre-Emphasis

The programmable pre-emphasis module in each transmit buffer boosts the high frequencies in the transmit data signal, which may be attenuated in the transmission media. Using pre-emphasis can maximize the data eye opening at the far-end receiver.

The transmission line's transfer function can be represented in the frequency domain as a low pass filter. Any frequency components below the -3dB frequency pass through with minimal losses. Frequency components greater than the -3dB frequency are attenuated. This variation in frequency response yields data dependent jitter and other ISI effects. By applying pre-emphasis, the high frequency components are boosted, that is, pre-emphasized. Pre-emphasis equalizes the frequency response at the receiver so the difference between the low frequency and high frequency components are reduced, which minimizes the ISI effects from the transmission medium.

The pre-emphasis requirements increase as data rates through legacy backplanes increase. The Arria GX transmitter buffer employs a pre-emphasis circuit with up to 184% of pre-emphasis to correct for losses in the transmission medium.

You set pre-emphasis settings through a slider menu in the ALT2GXB MegaWizard Plug-In Manager. Arria GX devices support the first five settings for first post-tap pre-emphasis. Specify the first post-tap pre-emphasis settings through the MegaWizard Plug-In Manager.

### Transmitter Termination

The Arria GX transmitter buffer includes on-chip differential termination of 100 $\Omega$. The resistance is adjusted by the on-chip calibration circuit in the calibration block (refer to "Calibration Blocks" on page 1–82 for more information), which compensates for temperature, voltage, and process changes. You can disable the on-chip termination to use external termination. If you select external termination, the transmitter common mode is also tri-stated.

You set the transmitter termination setting through a pull-down menu in the ALT2GXB MegaWizard Plug-In Manager.

### PCI Express Receiver Detect

The Arria GX transmitter buffer has a built-in receiver detection circuit for use in the PIPE mode. This circuit detects if there is a receiver downstream by sending out a pulse on the common mode of the

transmitter and monitoring the reflection. This mode requires the transmitter buffer to be tri-stated (in Electrical Idle mode) and the use of on-chip termination and a 125 MHz `fixedclk` signal.

This feature is only available in the PIPE mode. You enable it by setting the `tx_forceelecidle` and `tx_detectrxloopback` ports to 1'b1. You must set the `powerdn` port to 2'b10 to place the transmitter in the PCI-Express P1 power down state. The results of the receiver detect are encoded on the `pipestatus` port.

### PCI Express Electrical Idle

The Arria GX transmitter buffer supports PCI Express Electrical Idle (or individual transmitter tri-state). This feature is only active in the PIPE mode. The `tx_forceelecidle` port puts the transmitter buffer in Electrical Idle mode. This port is available in all PCI Express power-down modes and has a specific use in each mode. Table 1–4 shows the usage in each power mode.

*Table 1–4. Power Mode Usage*

| Power Mode | Usage |
|---|---|
| P0 | `tx_forceelecidle` must be asserted. If this signal is deasserted, it indicates that there is valid data. |
| P1 | `tx_forceelecidle` must be asserted. |
| P2 | When deasserted, the beacon signal must be transmitted. |

## Receiver Channel Architecture

This section provides a brief description about sub-blocks within the receiver channel (Figure 1–32). The sub-blocks are described in order from the serial receiver input buffer to the receiver phase compensation FIFO buffer at the transceiver-PLD interface.

*Figure 1–32. Receiver Channel Block Diagram*
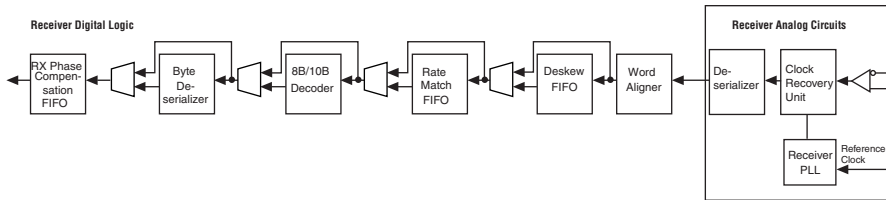
## Receiver Buffer

The Arria GX receiver buffers support 1.2-V, 1.5-V, 3.3-V PCML (pseudo-current mode logic), differential LVPECL and LVDS I/O standards. The receiver buffers support data rates from 600 Mbps to 3.125 Gbps and are capable of compensating up to 40 inches of FR4 trace across two connectors. The receiver buffer (Figure 1–33) has additional circuitry to improve signal integrity, including a programmable equalization circuit and internal termination circuitry. Through a signal detect circuit, the receiver buffers can also detect if a signal of predefined amplitude exists at the receiver.

*Figure 1–33. Receiver Buffer*
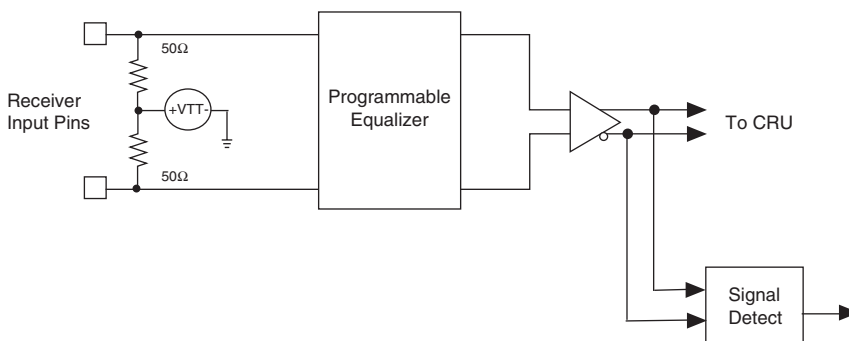


### Receiver Termination

The Arria GX receiver buffer has an optional on-chip differential termination of 100 Ω. You can set the receiver termination resistance setting using one of these options:

- Set receiver termination resistance by:
    a. Set the receiver termination resistance option in the MegaWizard Plug-In Manager if on-chip termination is used. Arria GX supports 100 Ω termination. If the design requires external receive termination, turn on the **Use External Receiver Termination** option.

    b. You make the differential termination assignment per pin in the Quartus II software. (On the Assignments menu, point to **Assignment Organizer**, and click **Options for Individual Nodes Only**. Then click **Stratix II GX GXB Termination Value**.)

- Verify and set the receiver termination settings before compilation.

### Signal Threshold Detection Circuit

The signal detect feature is supported only in PIPE mode. The signal detect/loss threshold detector senses if the specified voltage level exists at the receiver buffer. This detector has a hysteresis response, that filters out any high frequency ringing caused by inter symbol interference or high frequency losses in the transmission medium. The `rx_signaldetect` signal indicates if a signal conforms to the signal detection settings. A high level indicates that the signal conforms to the settings, a low level indicates that the signal does not conform to the settings.

The signal detect levels are to be determined by characterization. The signal detect levels may vary because of changing data patterns.

The signal/detect loss threshold detector also switches the receiver PLL/CRU from lock-to-reference mode to lock-to-data mode. The lock-to-reference and lock-to-data modes dictate whether the VCO of the clock recovery unit (CRU) is trained by the reference clock or by the data stream.

You can bypass the signal/detect loss threshold detection circuit by choosing the **Forced Signal Detect** option in the MegaWizard Plug-In Manager. This is useful in lossy environments where the voltage thresholds might not meet the lowest voltage threshold setting. Forcing this signal high enables the receiver PLL to switch from VCO training based on the reference clock to the incoming data without detecting a valid voltage threshold.

### Receiver Common Mode

Arria GX transceivers support the receiver buffer common mode voltages of 0.85 V and 1.2 V. Altera recommends selecting 0.85 V as the receiver buffer common mode voltage.

### Programmable Equalization

The Arria GX device offers an equalization circuit in each gigabit transceiver block receiver channel to increase noise margins and help reduce the effects of high frequency losses. The programmable equalizer compensates for the high frequency losses that distort the signal and reduces the noise margin of the transmission medium by equalizing the frequency response. There are five equalizer control settings allowed for an Arria GX device (including a setting with no equalization). In addition to equalization, Arria GX devices offer an equalizer DC gain option. There are three legal settings for DC gain. You specify the equalizer settings (Equalization Settings and DC Gain) through the MegaWizard Plug-In Manager.

The transmission line's transfer function can be represented in the frequency domain as a low pass filter. Any frequency components below the -3dB frequency pass through with minimal losses. Frequency components that are greater than the -3dB frequency are attenuated. This variation in frequency response yields data-dependent jitter and other ISI effects. By applying equalization, the low frequency components are attenuated. This equalizes the frequency response such that the delta between the low frequency and high frequency components is reduced, which in return minimizes the ISI effects from the transmission medium.

## Receiver PLL

Each transceiver channel has its own receiver PLL that is fed by an input reference clock. The reference clock frequency depends on the functional mode for which the transceiver channel is configured for. The clock recovery unit (CRU) controls whether the receiver PLL locks to the input reference clock (lock-to-reference mode) or the incoming serial data (lock-to-data mode). Refer to "Clock Recovery Unit (CRU)" on page 1–41 for more details on lock-to-reference and lock-to-data modes. The receiver PLL, in conjunction with the clock recovery unit, generates two clocks: a high speed serial clock that clocks the deserializer and a low-speed parallel clock that clocks the receiver's digital logic.

☞ This section only discusses the receiver PLL operation in lock-to-reference mode. For lock-to-data mode, refer to "Clock Recovery Unit (CRU)" on page 1–41.

Figure 1–34 shows the block diagram of the receiver PLL in lock-to-reference mode.

*Figure 1–34. Receiver PLL Block Diagram*
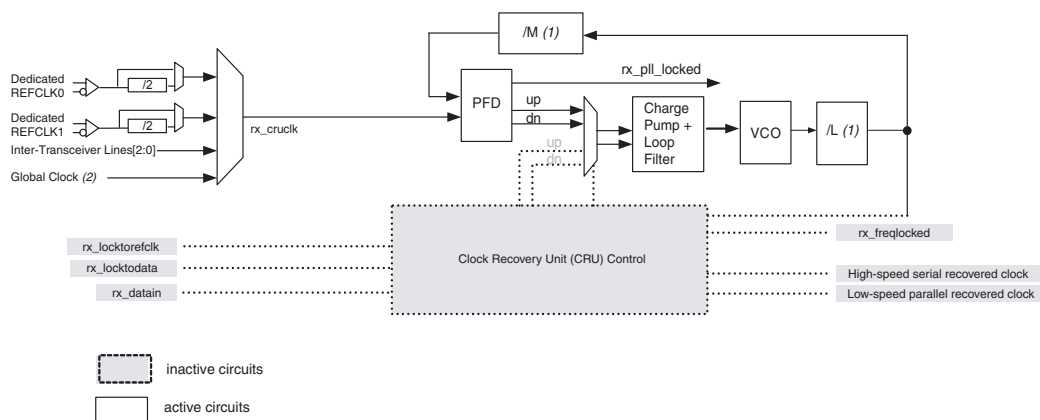


Notes to *Figure 1–34*:
(1)    You only need to select the protocol and the available input reference clock frequency in the Quartus II MegaWizard
       Plug-In Manager. Based on your selections, the MegaWizard Plug-In Manager automatically selects the necessary
       /M and /L dividers.
(2)    The global clock line must be driven from an input pin only.

The reference clock input to the receiver PLL can be derived from:

■    One of the two available dedicated reference clock input pins
     (REFCLK0 or REFCLK1) of the associated transceiver block
■    PLD clock network (must be driven directly from an input clock pin
     and cannot be driven by user logic or enhanced PLL)
■    Inter-transceiver block lines driven by reference clock input pins of
     other transceiver blocks

Depending on the functional mode, the Quartus II software automatically
selects the appropriate receiver PLL bandwidth.

### Clock Synthesis

The maximum input frequency of the receiver PLL's phase frequency
detector (PFD) is 325 MHz. To achieve a reference clock frequency above
this limitation, the divide by 2 pre-divider on the dedicated local REFCLK
path is automatically enabled by the Quartus II software. This divides the
reference clock frequency by a factor of 2, and the /M PLL multiplier
multiplies this pre-divided clock to yield the configured data rate. For
example, in a situation with a data rate of 2500 Mbps and a reference clock
of 500 MHz, the reference clock must be assigned to the REFCLK port
where the 500 MHz reference clock can be divided by 2, yielding a

250 MHz clock at the PFD. The VCO runs at half the data rate, so the selected multiplication factor should yield a 1250 MHz high speed clock. The Quartus II software automatically selects a multiplication factor of ×5 in this case to generate a 1250 MHz clock from the pre-divided 250 MHz clock.

If the /2 pre-divider is used, the reference clock must be fed by a dedicated reference clock input (REFCLK) pin. Otherwise, the Quartus II compiler gives a Fitter error.

The pre-divider and the multiplication factors are automatically set by the Quartus II software. The MegaWizard Plug-In Manager takes the data rate input and provides a list of the available reference clock frequencies that fall within the supported multiplication factors that you can select.

### PPM Frequency Threshold Detector

The PPM frequency threshold detector senses whether the incoming reference clock to the clock recovery unit (CRU) and the PLL VCO of the CRU are within a prescribed PPM tolerance range. Valid parameters are 62.5, 100, 125, 200, 250, 300, 500, or 1000 PPM. The default parameter, if no assignments are made, is 1000 PPM. The output of the PPM frequency threshold detector is one of the variables that assert the rx_freqlocked signal. Refer to for more details regarding the rx_freqlocked signal.

### Receiver Bandwidth Type

The Arria GX receiver PLL in the CRU offers a programmable bandwidth setting. The PLL bandwidth is the measure of the PLL's ability to track the input data and jitter. The bandwidth is determined by the -3dB frequency of the closed-loop gain of the PLL.

A higher bandwidth setting helps reject noise from the VCO and power supplies. A low bandwidth setting filters out more high frequency data input jitter.

Valid receiver bandwidth settings are low, medium, or high. The -3dB frequencies for these settings vary because of the non-linear nature and data dependencies of the circuit. You can vary the bandwidth to adjust and customize the performance on specific systems.

## Clock Recovery Unit (CRU)

The CRU (Figure 1–35) in each transceiver channel recovers the clock from the received serial data stream. You can set the CRU to lock to the received serial data phase and frequency (lock-to-data mode) to eliminate

any clock-to-data skew or to keep the receiver PLL locked to the reference clock (lock-to-reference mode). The switch between lock-to-data and lock-to-reference modes can be done automatically or manually. The CRU, in conjunction with the receiver PLL, generates two clocks: a high-speed serial recovered clock that feeds the deserializer and a low-speed parallel recovered clock that feeds the receiver's digital logic.

*Figure 1–35. Clock Recovery Unit*
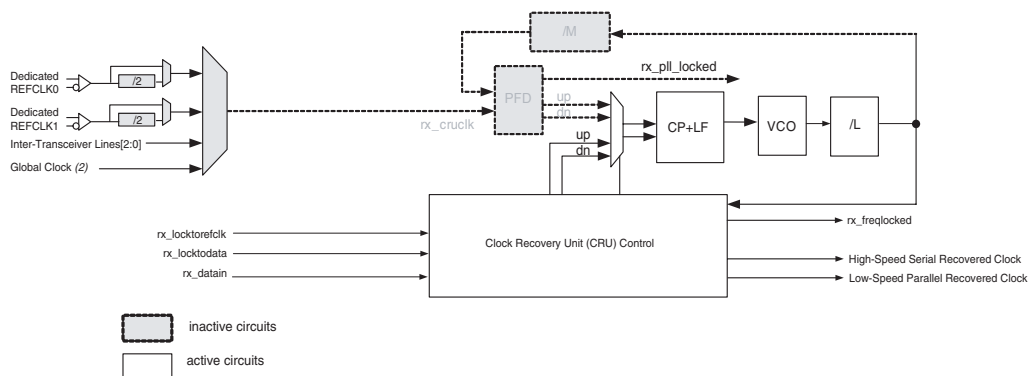


Notes to *Figure 1–35*:
(1) You only need to select the protocol and the available input reference clock frequency in the Quartus II MegaWizard Plug-In Manager. Based on your selections, the MegaWizard Plug-In Manager automatically selects the necessary /M and /L dividers.
(2) The global clock line must be driven from an input pin only.

### Automatic Lock Mode

After coming out of reset in automatic lock mode, the CRU initially sets the receiver PLL to lock to the input reference clock (lock-to-reference mode). After the receiver PLL locks to the input reference clock, the CRU automatically sets it to lock to the incoming serial data (lock-to-data mode) when the following two conditions are met:

■ The receiver PLL output clock is within the configured PPM frequency threshold setting with respect to its reference clock (frequency locked)
■ The reference clock and receiver PLL output clock are phase matched within approximately 0.08 UI (phase locked)

When the receiver PLL and CRU are in lock-to-reference mode, the PPM detector and the phase detector circuits monitor the relationship of the reference clock to the receiver PLL VCO output. If the frequency difference is within the configured PPM setting (as set in the MegaWizard Plug-In Manager) and the phase difference is within 0.08 UI, the CRU

switches to lock-to-data mode. The switch from lock-to-reference to lock-to-data mode is indicated by the assertion of the `rx_freqlocked` signal.

In lock-to-data mode, the receiver PLL uses a phase detector to keep the recovered clock phase-matched to the data. If the PLL does not stay locked to data due to frequency drift or severe amplitude attenuation, the CRU switches back to lock-to-reference mode to lock the PLL to the reference clock. In automatic lock mode, the following condition forces the CRU to fall out of lock-to-data mode:

> The CRU PLL is not within the configured PPM frequency threshold setting with respect to its reference clock.

The switch from lock-to-data to lock-to-reference mode is indicated by the de-assertion of `rx_freqlocked` signal.

When the CRU is in lock-to-data mode (`rx_freqlocked` is asserted), it tries to phase-match the PLL with the incoming data. As a result, the phase of the PLL output clock may differ from the reference clock due to which `rx_pll_locked` signal might get de-asserted. You should ignore the `rx_pll_locked` signal when the `rx_freqlocked` signal is asserted high.

### Manual Lock Mode

Two optional input pins (`rx_locktorefclk` and `rx_locktodata`) allow you to control whether the CRU PLL automatically or manually switches between lock-to-reference mode and lock-to-data mode. This enables you to bypass the default automatic switchover circuitry if either `rx_locktorefclk` or `rx_locktodata` is instantiated.

When the `rx_locktorefclk` signal is asserted, the CRU forces the receiver PLL to lock to the reference clock. When the `rx_locktodata` signal is asserted, the CRU forces the receiver PLL to lock-to-data. When both signals are asserted, the `rx_locktodata` signal takes precedence over the `rx_locktorefclk` signal, forcing the receiver PLL to lock-to-data.

The PPM threshold frequency detector and phase relationship detector reaction times may be too long for some applications. You can manually control the CRU to reduce PLL lock times using the `rx_locktorefclk` and `rx_locktodata` ports. Using the manual mode may reduce the time it takes for the CRU to switch from lock-to-reference mode to lock-to-data mode. You can assert the `rx_locktorefclk` to initially

train the PLL to the reference clock. Once the receiver PLL locks to the reference clock, you can assert the rx_locktodata signal to force the PLL to lock to the incoming data.

When the rx_locktorefclk signal is asserted high, the rx_freqlocked signal does not have any significance and is always driven low, indicating that the CRU is in lock-to-reference mode. When the rx_locktodata signal is asserted high, the rx_freqlocked signal is always driven high, indicating that the CRU is in lock-to-data mode. If both signals are de-asserted, the CRU is in automatic lock mode.

Table 1–5 shows a summary of the control signals.

*Table 1–5. CRU User Control Lock Signals*

| rx_locktorefclk | rx_locktodata | CRU Mode |
|---|---|---|
| 1 | 0 | Lock-to-reference clock |
| x | 1 | Lock to data |
| 0 | 0 | Automatic |

## Deserializer

The deserializer block clocks in serial input data from the receiver buffer using the high-speed serial recovered clock and deserializes it into 8- or 10-bit parallel data using the low-speed parallel recovered clock. It feeds the deserialized data to the word aligner as shown in Figure 1–36.

Figure 1–36. Deserializer
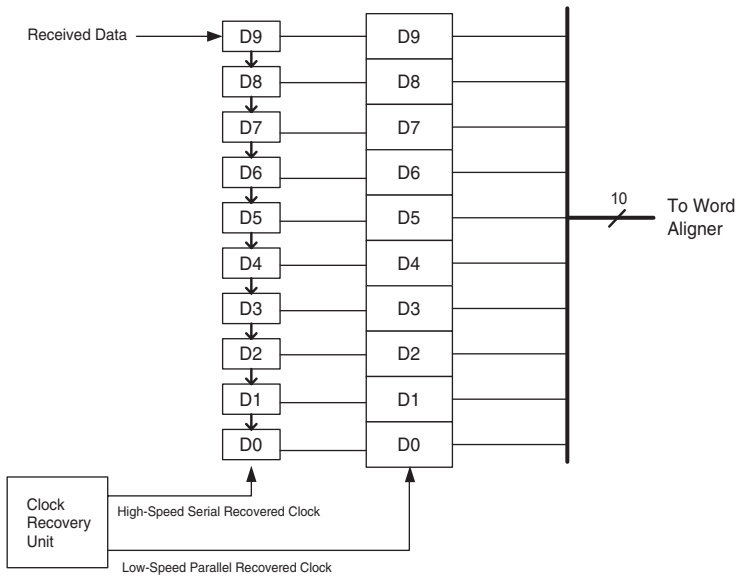


Figure 1–37 shows the serial bit order of the deserializer block input and the parallel data output of the deserializer block. A serial stream (0101111100) is deserialized to a value 10'h17C. The serial data is assumed to be received LSB to MSB.

Figure 1–37. Deserializer Bit Order
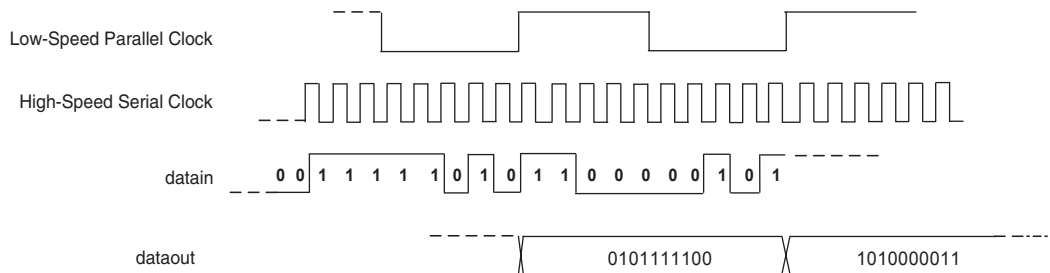
*Receiver Polarity Inversion*

The positive and negative signals of a serial differential link might be accidentally swapped during board layout. Solutions such as a board re-spin or major updates to the PLD logic can prove expensive. The receiver polarity inversion feature is provided to correct this situation.

An optional rx_invpolarity port is available in all modes to dynamically enable the receiver polarity inversion feature. A high on the rx_invpolarity port inverts the polarity of every bit of the 8- or 10-bit input data word to the word aligner in the receiver data path. Since inverting the polarity of each bit has the same effect as swapping the positive and negative signals of the differential link, correct data is seen by the receiver. The rx_invpolarity is a dynamic signal and may cause initial disparity errors in an 8B/10B encoded link. The downstream system must be able to tolerate these disparity errors.

The receiver polarity inversion feature is different from the PCI Express (PIPE) 8B/10B polarity inversion feature. The receiver polarity inversion feature inverts the polarity of the data bits at the input of the word aligner. The PCI Express (PIPE) 8B/10B polarity inversion feature inverts the polarity of the data bits at the input of the 8B/10B decoder and is available only in PCI Express (PIPE) mode. Enabling the generic receiver polarity inversion and the PCI Express (PIPE) 8B/10B polarity inversion simultaneously is not allowed in PCI Express (PIPE) mode.

Figure 1–38 illustrates the receiver polarity inversion feature.

*Figure 1–38. Receiver Polarity Inversion*



Output from Deserializer          Input to Word Aligner
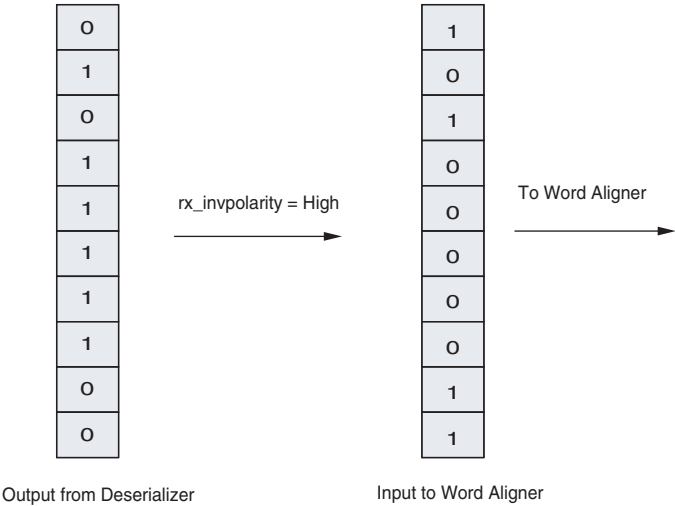
## Word Aligner

The word aligner (refer to Figure 1–39) clocks in received data from the deserializer using the low-speed recovered clock. It restores the word boundary of the upstream transmitter based on the pre-defined word alignment character for the selected protocol. In addition to restoring the word boundary, the word aligner also implements a synchronization state machine in all functional modes to achieve lane synchronization.

Figure 1–39 shows the block diagram for the word aligner block.

*Figure 1–39. Word Aligner*
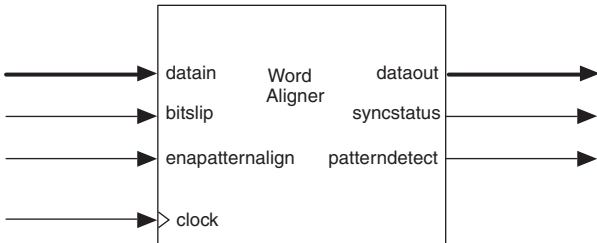
The word aligner consists of four sub-modules:

- Aligner block
- Pattern detect block
- Manual bit-slip block
- Run-length checker

There are two modes in which the word aligner works: basic mode and automatic synchronization state machine mode. The following sections explain each of the blocks in each mode of operation. The word aligner cannot be bypassed and must be used. However, you can use the `rx_enapatternalign` port to set the word alignment to not align to the pattern.

### Basic Mode

In basic mode, there are three blocks active in the word aligner:

- Pattern detector
- Manual word aligner
- Automatic synchronization state machine

The pattern detector detects if the pattern exists in the current word boundary. The manual alignment identifies the alignment pattern across the byte boundaries and aligns to the correct byte boundary. The synchronization state machine detects the number of alignment patterns and good code groups for synchronization and goes out of synchronization if code group errors (bad code groups) are detected. Figure 1–40 and Table 1–6 show the supported alignment modes when basic mode is selected.

*Figure 1–40. Word Aligner Components in Basic Mode*
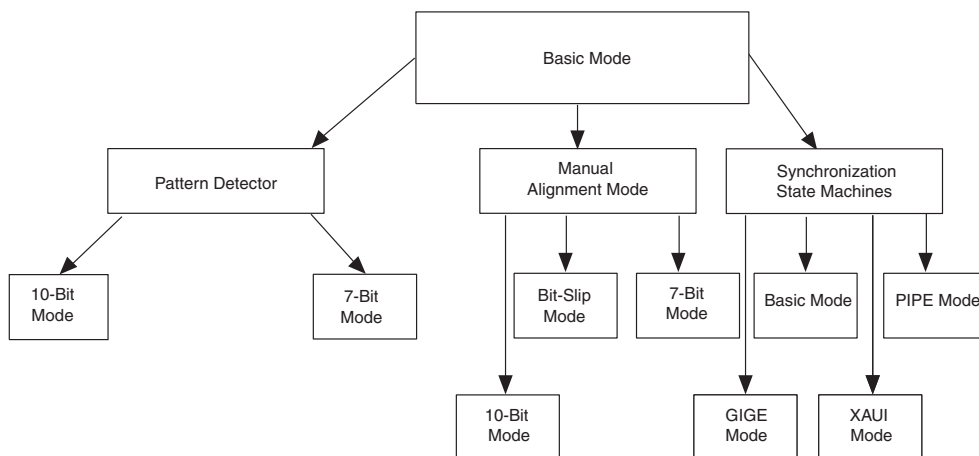


*Table 1–6. Word Alignment Modes*

| Word Alignment Mode | Effective Mode | Control Signals | Status Signals |
|---|---|---|---|
| Synchronization state machine | PCI Express, XAUI, GIGE, Serial RapidIO, or Basic | Automatically controlled to adhere to the specified standard or by user entered parameter | `rx_syncstatus` `rx_patterndetect` |
| Manual 7- and 10-bit alignment mode | Alignment to detected pattern when allowed by the `rx_enapatternalign` signal | `rx_enapatternalign` | `rx_syncstatus` `rx_patterndetect` |
| Manual bit-slipping alignment mode | Manual bit slip controlled by the PLD logic array | `rx_bitslip` | `rx_patterndetect` |

**Pattern Detector Module**

The pattern detector matches a pre-defined alignment pattern to the current byte boundary. When the pattern detector locates the alignment pattern, the optional `rx_patterndetect` signal is asserted for the duration of one clock cycle to signify that the alignment pattern exists in the current word boundary. The pattern detector module only indicates that the signal exists and does not modify the word boundary.

Modification of the word boundary is discussed in the sections "Manual Alignment Modes" on page 1–51 and "Synchronization State Machine Mode" on page 1–55.

In the MegaWizard, you can program a 7-bit or a 10-bit pattern for the pattern detector to recognize. The pattern used for pattern matching is automatically derived from the word alignment pattern in the MegaWizard. For the 7-bit and 10-bit patterns, the actual alignment pattern specified in the MegaWizard and its complement are checked. Table 1–7 shows the supported alignment patterns.

| Table 1–7. Supported Alignment Patterns | | |
|---|---|---|
| **Pattern Detect Mode** | **Supported Protocols** | **Pattern Checked** |
| 7 bit | Basic, GIGE (enhanced only) | Actual and complement |
| 10 bit | Basic, XAUI, GIGE, Serial RapidIO, and PIPE | Actual and complement |

In 8B/10B encoded data, actual and complement pattern indicates positive and negative disparities.

**7-Bit Pattern Mode**
In the 7-bit pattern detection mode (use this mode with 8B/10B code), the pattern detector matches the seven LSBs of the 10-bit alignment pattern, which you specified in your ALT2GXB custom megafunction variation, in the current word boundary. Both positive and negative disparities are also checked in this mode.

The 7-bit pattern mode can mask out the three MSBs of the data, which allows the pattern detector to recognize multiple alignment patterns. For example, in the 8B/10B encoded data, a /K28.5/ (b'0011111010), /K28.1/ (b'0011111001), and /K28.7/ (b'0011111000) share seven common LSBs. Masking the three MSBs allows the pattern detector to resolve all three alignment patterns and indicate them on the rx_patterndetect port.

In 7-bit pattern mode, the word aligner still aligns to a 10 bit word boundary. The specified 7-bit pattern forms the least significant seven bits of the 10-bit word.

**10-Bit Pattern Mode**
In the 10-bit pattern detection mode (use this mode with 8B/10B code), the module matches the 10-bit alignment pattern you specified in your ALT2GXB custom megafunction variation with the data and its complement in the current word boundary. Both positive and negative

disparities are checked by the pattern checker in this mode. For example, if you specify a /K28.5/ (b'0011111010) pattern as the comma, `rx_patterndetect` is asserted if b'0011111010 or b'1100000101 is detected in the incoming data.

### Manual Alignment Modes

The word aligner has two manual alignment modes (7- and 10-bits) when the transceiver data path is in Basic mode.

**7-bit Alignment Mode**
In the 7-bit alignment mode (use the 8B/10B encoded data with this mode), the module looks for the 7-bit alignment pattern you specified in the MegaWizard Plug-In Manager in the incoming data stream. The 7-bit alignment mode is useful because it can mask out the three most significant bits of the data, which allows the word aligner to align to multiple alignment patterns. For example, in the 8B/10B encoded data, a /K28.5/ (b'0011111010), /K28.1/ (b'0011111001), and /K28.7/ (b'0011111000) share seven common LSBs. Masking the three MSBs allows the word aligner to resolve all three alignment patterns synchronized to it. The word aligner places the boundary of the 7-bit pattern in the LSByte position with bit positions [0..7]. The true and complement of the patterns is checked.

Use the `rx_enapatternalign` port to enable the 7-bit manual word alignment mode. When the `rx_enapatternalign` signal is high, the word aligner detects the specified alignment patterns and realigns the byte boundary if needed. The `rx_syncstatus` port is asserted for one parallel clock cycle to signify that the word boundary was detected across the current word boundary and has synchronized to the new boundary, if a rising edge was detected previously on the `rx_enapatternalign` port. You must differentiate if the acquired byte boundary is correct, because the 7-bit pattern can appear between word boundaries. For example, in the standard 7-bit alignment pattern 7'b1111100, if a K28.7 is followed by a K28.5, the 7-bit alignment pattern appears on K28.7, between K28.7 and K28.5, and also again in K28.5 (refer to Figure 1–41).

*Figure 1–41. Cross Boundary 7-Bit Comma When /K28.7 is Followed by /K28.5*
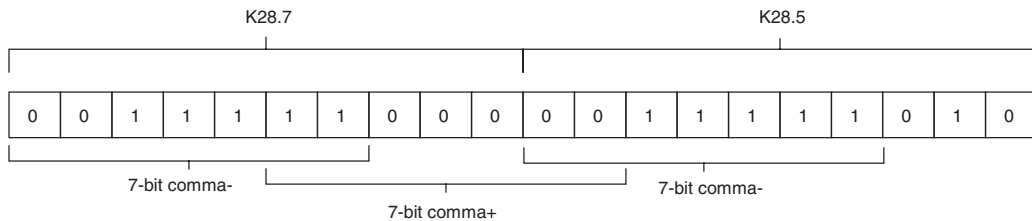


**Manual 10-Bit Alignment Mode**
You can configure the word aligner to align to a 10-bit word boundary. The internal word alignment circuitry shifts to the correct word boundary if the alignment pattern specified in the pattern detector is detected in the data stream.

The rx_enapatternalign port enables the word alignment in the manual 10-bit alignment mode. When the rx_enapatternalign signal is high, the word aligner detects the specified alignment pattern and realigns the byte boundary if necessary. The rx_syncstatus port is asserted for one parallel clock cycle to signify that the word boundary has been detected across the word boundary and has synchronized to the new boundary.

The rx_enapatternalign signal is held high if the alignment pattern is known to be unique and does not appear across the byte boundaries of other data. For example, if an 8B/10B encoding scheme guarantees that the /K28.5/ code group is a unique pattern in the data stream, the rx_enapatternalign port is held at a constant high.

If the alignment pattern can exist between word boundaries, the rx_enapatternalign port must be controlled by the user logic in the PLD to avoid false word alignment. For example, assume that 8B/10B is used and a /+D19.1/ (b'110010 1001) character is specified as the alignment pattern. In this case, a false word boundary is detected if a /-D15.1/ (b'010111 1001) is followed by a /+D18.1/ (b'010011 1001). Refer to Figure 1–42.

*Figure 1–42. False Word Boundary Alignment if Alignment Pattern Exists Across Word Boundaries, Basic Mode*
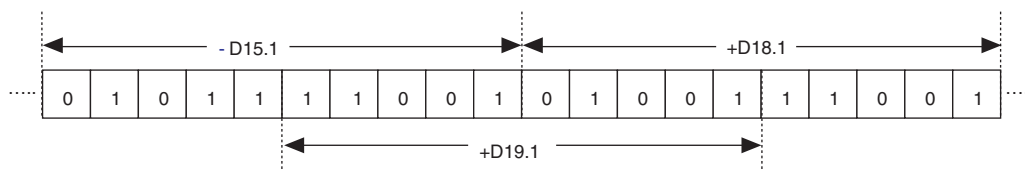


In this example, the rx_enapatternalign signal is deasserted after the word aligner locates the initial word alignment to prevent false word boundary alignment. When the rx_enapatternalign signal is deasserted, the current word boundary is locked even if the alignment pattern is detected across different boundaries. In this case, the rx_syncstatus acts as a re-synchronization signal to signify that the alignment pattern was detected, but the boundary is different than the current boundary. You must monitor this signal and reassert the rx_enapatternalign signal if realignment is desired.

Figure 1–43 shows an example of how the word aligner signals interact in 10-bit alignment mode. In this example, a /K28.5/ (10'b0011111010) is specified as the alignment pattern. The rx_enapatternalign signal is held high at time n, so alignment occurs whenever an alignment pattern exists in the pattern. The rx_patterndetect signal is asserted for one clock cycle to signify that the pattern exists on the re-aligned boundary. The rx_syncstatus signal is also asserted for one clock cycle to signify that the boundary has been synchronized. At time n + 1, the rx_enapatternalign signal is deasserted to instruct the word aligner to lock the current word boundary.

The alignment pattern is detected at time n + 2, but it exists on a different boundary than the current locked boundary. The bit orientation of the Arria GX device is LSB to MSB, so the alignment pattern exists across time n + 2 and n + 3 (refer to Figure 1–43). In this condition the rx_patterndetect remains low because the alignment pattern does not exist on the current word boundary, but the rx_syncstatus signal is asserted for one clock cycle to signify a resynchronization condition. This means that the alignment pattern has been detected across another word boundary.

The user logic design in the PLD must decide whether or not to assert the rx_enapatternalign to reinitiate the word alignment process. At time n + 5 the rx_patterndetect signal is asserted for one clock cycle to signify that the alignment pattern has been detected on the current word boundary.

*Figure 1–43. Word Aligner Symbol Interaction in 10-Bit Manual Alignment Mode*
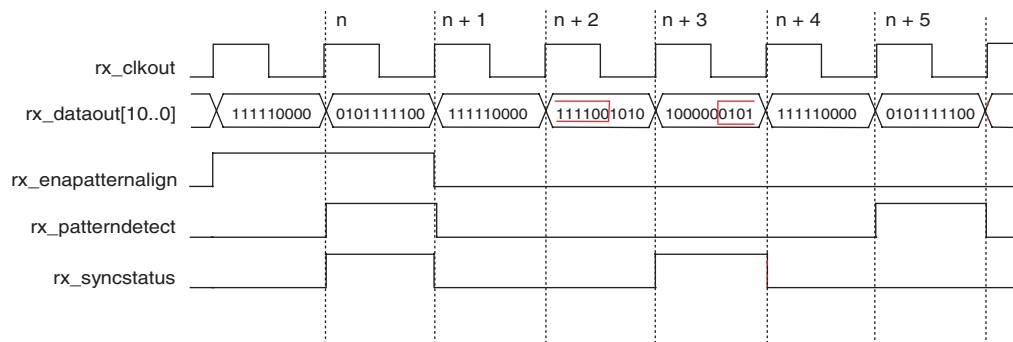


**Manual Bit-Slip Alignment Mode**

You can also achieve word alignment by enabling the manual bit-slip option in the MegaWizard Plug-In Manager. With this option enabled, the transceiver shifts the word boundary MSB to LSB one bit every parallel clock cycle. The transceiver shifts the word boundary every time the bit-slipping circuitry detects a rising edge of the `rx_bitslip` signal. At each rising edge of the `rx_bitslip` signal, the word boundary slips one bit. The bit that arrives at the receiver first is skipped. When the word boundary matches the alignment pattern you specified in the MegaWizard Plug-In Manager, the `rx_patterndetect` signal is asserted for one clock cycle. You must implement the logic in the PLD logic array to control the bit-slip circuitry.

The bit slipper is useful if the alignment pattern changes dynamically when the Arria GX device is in user mode. You can implement the controller in the logic array, so you can build a custom controller to dynamically change the alignment pattern without needing to reprogram the Arria GX device.

Figure 1–44 shows an example of how the word aligner signals interact in the manual bit slip alignment mode. For this example, 8'b00111100 is specified as the alignment pattern and an 8'b11110000 value is held at the `rx_datain` port.

Every rising edge on the `rx_bitslip` port causes the `rx_dataout` data to shift one bit from the MSB to the LSB by default. This is shown at time n + 2 where the 8'b11110000 data is shifted to a value of 8'b01111000. At this state the `rx_patterndetect` signal is held low because the specified alignment pattern does not exist in the current word boundary.

The `rx_bitslip` is disabled at time n + 3 and re-enabled at time n + 4. The output of the `rx_dataout` now matches the specified alignment pattern, thus the `rx_patterndetect` signal is asserted for one clock cycle. At time n + 5, the `rx_patterndetect` signal is still asserted because the alignment pattern still exists in the current word boundary. Finally, at time n + 6 the `rx_dataout` boundary is shifted again and the `rx_patterndetect` signal is deasserted to signify that the word boundary does not contain the alignment pattern.

*Figure 1–44. Word Aligner Symbol Interaction in Manual Bit-Slip Mode*
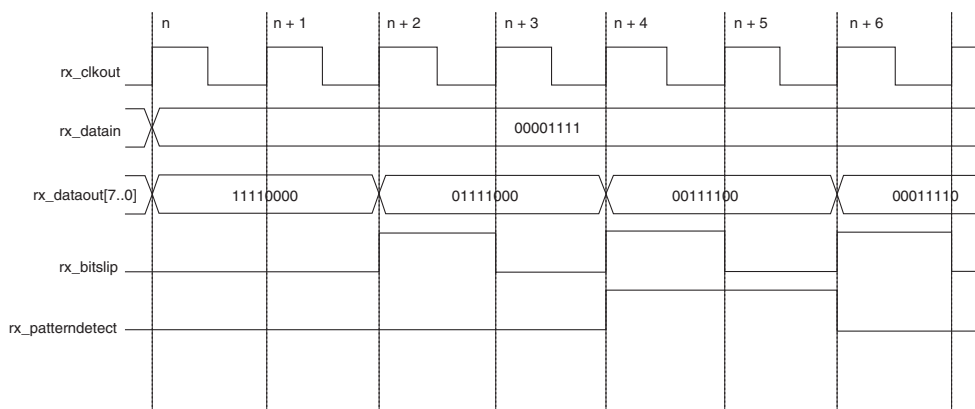


### Synchronization State Machine Mode

You can choose to have the link synchronization handled by a state machine. Unlike the manual alignment mode where there is no built-in hysteresis to go into or fall out of synchronization, the synchronization state machine offers automatic detection of a valid number of alignment patterns and synchronization and detection of code group errors for automatically falling out of synchronization. The synchronization state machine is available in the Basic, XAUI, GIGE, and PIPE modes. For the XAUI, GIGE, and PIPE modes, the number of alignment patterns, consecutive code groups, and bad code groups are fixed. You must use the 8B/10B code for the synchronization state machine. In XAUI, GIGE, and PIPE modes, the 8B/10B encoder/decoder is embedded in the transceiver data path. In Basic mode, you can configure the MegaWizard Plug-In Manager to either use or bypass the 8B/10B encoder/decoder in the transceiver. If the synchronization state machine is enabled and the 8B/10B encoder/decoder is bypassed, the 8B/10B encoder/decoder logic must be implemented outside the transceiver as a requirement for using the synchronization state machine.

In Basic mode, you can configure the state machine to suit a variety of standard and custom protocols. In the MegaWizard Plug-In Manager, you can program the number of alignment patterns to acquire link synchronization. You can program the number of bad code groups to fall out of synchronization. You can program the number of good code groups to negate a bad code group. You enter these values in the MegaWizard Plug-In Manager. The `rx_syncstatus` port indicates the link status. A high level indicates link synchronization is achieved, a low level indicates that synchronization has not yet been achieved or that there were enough code group errors to fall out of synchronization. Figure 1–45 shows a flowchart of the synchronization state machine.

*Figure 1–45. Word Aligner Synchronization State Machine Flow Chart*
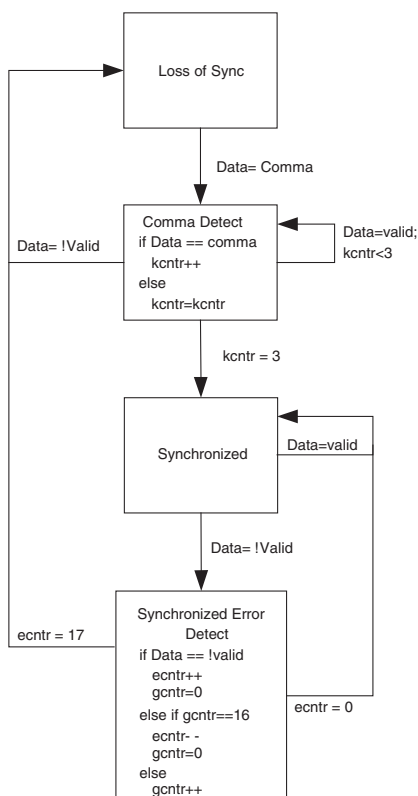


The maximum value for the number of valid alignment patterns and good code groups is 256. The maximum value of invalid or bad code groups to fall out of synchronization is 8. For example, if 3 is set for the number of good code groups, then when 3 consecutive good code groups

are detected after a bad code group, the effect of the bad code group on synchronization is negated. This does not negate the bad code group that actually triggers the loss of synchronization. To negate a loss of synchronization, the protocol defined number of alignment patterns must be received.

When either XAUI or GIGE mode is used, the synchronization and word alignment is handled automatically by a built-in state machine that adheres to either the IEEE 802.3ae or IEEE 802.3 synchronization specifications, respectively. If you specify either standard, the alignment pattern is automatically defaulted to /K28.5/ (b'0011111010).

When you specify the XAUI protocol, code-group synchronization is achieved upon the reception of four /K28.5/ commas. Each comma can be followed by any number of valid code groups. Invalid code groups are not allowed during the synchronization stage. When code-group synchronization is achieved the optional rx_syncstatus signal is asserted.

For more information about the operation of the synchronization phase, refer to clause 47-48 of the IEEE P802.3ae standard or XAUI mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*.

If you specify the GIGE protocol, code-group synchronization is achieved upon the reception of three consecutive ordered sets. An ordered set starts with the /K28.5/ comma and can be followed by an odd number of valid data code groups. Invalid code groups are not allowed during the reception of three ordered-sets. When code-group synchronization is achieved, the optional rx_syncstatus signal is asserted.

In PIPE mode, lane synchronization is achieved when the word aligner sees four good /K28.5/ commas and 16 good code groups. This is accomplished through the reception of four good PCI Express training sequences (TS1 or TS2). The PCI-Express fast training sequence (FTS) can also be used to achieve lane or link synchronization, but requires at least five of these training sequences. The rx_syncstatus signal is asserted when synchronization is achieved and is deasserted when the word aligner receives 23 code group errors.

### Run Length Checker

The programmable run-length violation circuit resides in the word aligner block and detects consecutive 1s or 0s in the data. If the data stream exceeds the preset maximum number of consecutive 1s or 0s, the violation is signified by the assertion of the rx_rlv signal.

This signal is not synchronized to the parallel data and appears in the logic array earlier than the run-length violation data. To ensure that the PLD can latch this signal in systems where there are frequency variations between the recovered clock and the PLD logic array clock, the `rx_rlv` signal is asserted for a minimum of two clock cycles. The `rx_rlv` signal may be asserted longer, depending on the run-length of the received data.

The run-length violation circuit detects up to a run length of 128 (for an 8-bit deserialization factor) or 160 (for a 10-bit deserialization factor). The settings are in increments of 4 or 5 for the 8-bit or 10-bit deserialization factors, respectively.

### Receiver Bit Reversal

By default, the Arria GX receiver assumes an LSB to MSB transmission. If the transmission order is MSB to LSB, then the receiver will put out the bit-flipped version of the data on the PLD interface. The Receiver Bit Reversal feature is available to correct this situation.

The Receiver Bit Reversal feature is available only in Basic mode. If the Receiver Bit Reversal feature is enabled, the 10-bit data D[9:0] at the output of the word aligner gets rewired to D[0:9]. Flipping the parallel data using this feature allows the receiver to put out the correctly bit-ordered data on the PLD interface in case of MSBit to LSBit transmission.

Because the receiver bit reversal is done at the output of the word aligner, a dynamic bit reversal would also require a reversal of word alignment pattern. As a result, the Receiver Bit Reversal feature is dynamic only if the receiver uses manual bit-slip alignment mode (no word alignment pattern). The Receiver Bit Reversal feature is static in all other Basic mode configurations and can be enabled through the MegaWizard Plug-In Manager. In configurations where this feature is dynamic, an `rx_revbitordwa` port is available to control the bit reversal dynamically. A high on the `rx_revbitordwa` port reverses the bit order at the input of the word aligner.

Figure 1–46 illustrates the receiver bit reversal feature in Basic 10-bit wide data path configuration.

*Figure 1–46. Receiver Bit Reversal in Basic Mode*

| | | | |
|---|---|---|---|
| D[9] | | | D[0] |
| D[8] | | | D[1] |
| D[7] | | | D[2] |
| D[6] | | | D[3] |
| D[5] | RX Bit Reversal | = Enabled | D[4] |
| D[4] | →  | | D[5] |
| D[3] | | | D[6] |
| D[2] | | | D[7] |
| D[1] | | | D[8] |
| D[0] | | | D[9] |

Output of Word Aligner before RX bit reversal — Output of Word Aligner after RX bit reversal
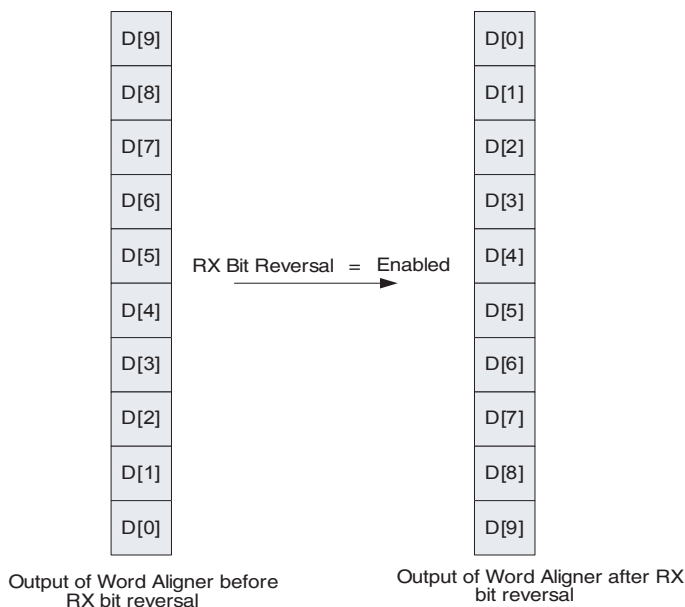
## Channel Aligner (Deskew)

The channel aligner is automatically used when implementing the XAUI protocol to ensure that the channels are aligned with respect to each other. The channel aligner uses a 16-word deep FIFO buffer and is available only in the XAUI mode.

For additional information about the Channel Aligner block, refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*.

## Rate Matcher

In asynchronous systems, the upstream transmitter and the local receiver may be clocked with independent reference clock sources. Frequency differences in the order of a few hundred PPM can potentially corrupt the data at the receiver. The rate matcher compensates for small clock frequency differences between the upstream transmitter and the local receiver clocks by inserting or removing skip characters or ordered-sets from the inter-packet gap (IPG) or idle streams. It inserts a skip character or ordered-set if the local receiver is running a faster clock than the upstream transmitter. It deletes a skip character or ordered-set if the local

receiver is running a slower clock than the upstream transmitter. The rate matcher is available in PCI Express (PIPE), GIGE, XAUI, and Basic functional modes.

The rate matcher consists of a 20-word-deep FIFO buffer and necessary logic to detect and perform the insertion and deletion functions. The write port of the rate matcher FIFO is clocked by the low-speed parallel recovered clock. The read port is clocked by the low-speed parallel clock from the CMU central or local clock divider block (Figure 1–47).

*Figure 1–47. Rate Matcher*
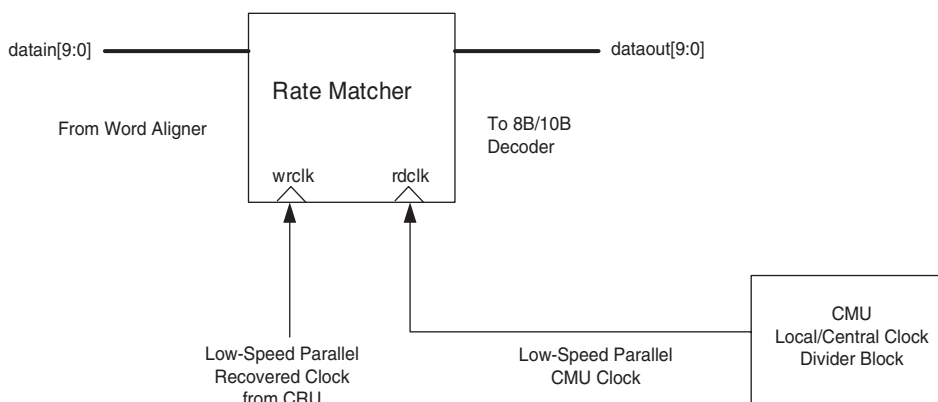


For information about the rate matcher in PIPE, GIGE, and XAUI modes, refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*.

### Basic Mode General Rate Matching

In Basic mode, the rate matcher supports up to 300 PPM differences between the upstream transmitter and the receiver. The rate matcher looks for the skip ordered set (SOS), which is a /K28.5/ comma followed by three programmable neutral disparity skip characters (for example, /K28.0/). For general rate matching, you can customize the SOS to support a variety of protocols, including custom protocols. The SOS must contain a valid control code group (Kx.y), followed by any neutral disparity skip code group (any Kx.y or Dx.y of neutral disparity, for example, K28.0). The rate matcher deletes or inserts skip characters when necessary to prevent the rate matching FIFO buffer from overflowing or underflowing.

The rate matcher in Basic mode can delete any number of skip characters as necessary in a cluster as long as there are skip characters to delete. There are no restrictions regarding deleting more than one skip character in a cluster of skip characters. Figure 1–48 shows an example of a Basic mode rate matcher deletion of two skip characters. Although the skip characters are programmable, the /K28.0/ control group is used for illustration purposes.

*Figure 1–48. Basic Mode Deletion of Two Skip Characters*
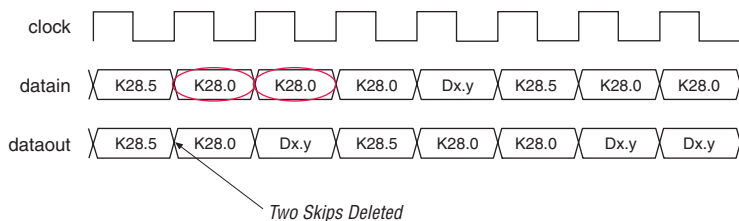


The rate matcher inserts skip characters as required for rate matching. For a given skip ordered set, the rate matcher inserts skip characters so that the total number of consecutive skip characters does not exceed five at the output of the rate matching FIFO buffer. Figure 1–49 shows an example where a skip character insertion is made on the second set of skip ordered sets because the first set has the maximum number of skip characters.

*Figure 1–49. Basic Mode Insertion of a Skip Character*
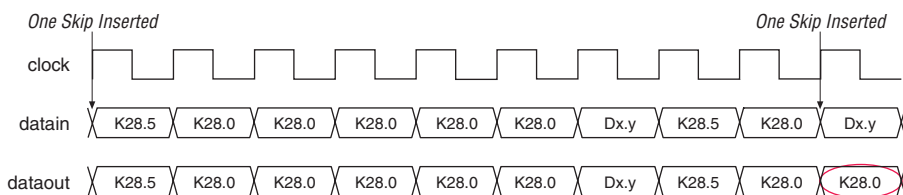


The Arria GX rate matcher in Basic mode has FIFO buffer overflow and underflow protection. In the event of a FIFO buffer overflow the rate matcher deletes any data after the overflow condition to prevent FIFO buffer pointer corruption until the rate matcher is not full. In an underflow condition, the rate matcher inserts 9'h1FE (/K30.7/) until the FIFO buffer is not empty. These measures ensure that the FIFO buffer gracefully exits the overflow and underflow condition without requiring a FIFO buffer reset.

### 8B/10B Decoder

The 8B/10B decoder takes in 10-bit data from the rate matcher and decodes it into 8-bit data + 1-bit control identifier, thereby restoring the original transmitted data at the receiver. The decoded data is fed to either the byte deserializer or the receiver phase compensation FIFO buffer (depending on protocol). The 8B/10B decoder conforms to IEEE 802.3 1998 edition standards.

Figure 1–50 shows a 10-bit code group decoded to an 8-bit data and a 1-bit control indicator.

*Figure 1–50. 10-Bit to 8-Bit Conversion*
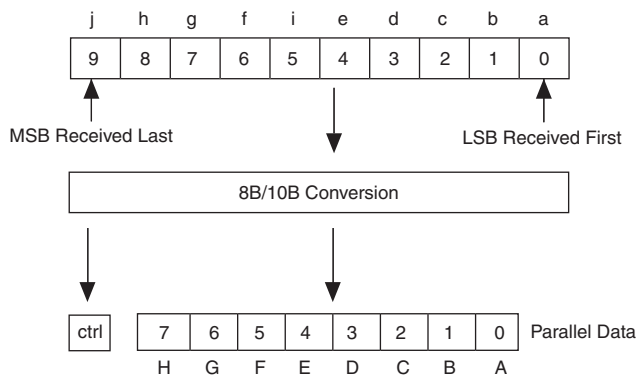


### Control Code Group Detection

The 8B/10B decoder differentiates between data and control codes through the rx_ctrldetect port. If the received 10-bit code group is a control code group (Kx.y), the rx_ctrldetect signal is driven high. If it is a data code group (Dx.y), the rx_ctrldetect signal is driven low.

Figure 1–51 shows an example waveform demonstrating the receipt of a K28.5 code group (BC + ctrl). The rx_ctrldetect=1'b1 is aligned with 8'hbc, indicating that it is a control code group. The rest of the codes received are Dx.y code groups.

*Figure 1–51. Control Code Group Detection*
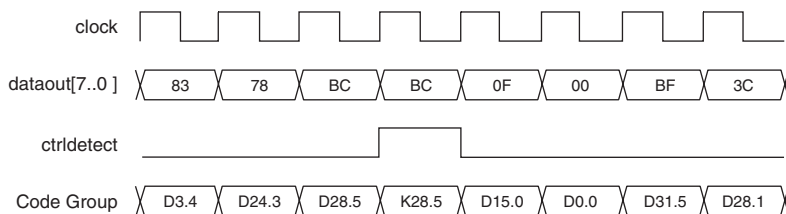


### Code Group Error Detection

If the received 10-bit code group is not a part of valid `Dx.y` or `Kx.y` code groups, the 8B/10B decoder block asserts an error flag on the `rx_errdetect` port. The error flag signal (`rx_errdetect`) has the same data path delay from the 8B/10B decoder to the PLD-transceiver interface as the invalid code group.

In GIGE, XAUI, and PIPE modes, the invalid code is replaced by a /K30.7/ code (8'hFE on `rx_dataout` + 1'b1 on `rx_ctrldetect`). In all other modes, the value of the invalid code value can vary and should be ignored

### Disparity Error Detection

If the received 10-bit code group is detected with incorrect running disparity, the 8B/10B decoder block asserts an error flag on the `rx_disperr` and `rx_errdetect` ports.

Refer to the *Specifications and Additional Information* chapter in volume 2 of the *Arria GX Device Handbook* for information about the disparity calculation.

If negative disparity is calculated for the last 10-bit code group, a neutral or positive disparity 10-bit code group is expected. If the 8B/10B decoder does not receive a neutral or positive disparity 10-bit code group, the `rx_disperr` signal goes high, indicating that the code group received has a disparity error. Similarly, if a neutral or negative disparity is expected and a 10-bit code group with positive disparity is received, the `rx_disperr` signal goes high.

The detection of the disparity error might be delayed, depending on the data that follows the actual disparity error. The 8B/10B control codes terminate propagation of the disparity error. Any disparity errors propagated stop at the control code group, terminating that disparity error.

In GIGE and XAUI modes, the code that contains a disparity error is replaced by a /K30.7/ code (8'hFE on rx_dataout + rx_ctrldetect). In all other modes, the code with incorrect disparity should be treated as an invalid code and ignored.

Figure 1–52 shows a case where the disparity is violated. A K28.5 code group has an 8-bit value of 8'hbc and a 10-bit value that depends on the disparity calculation at the point of the generation of the K28.5 code group. The 10-bit value is 10'b0011111010 (10'h17c) for RD– or 10'b1100000101 (10'h283) for RD+. If the running disparity at time n - 1 is negative, the expected code group at time must be from the RD– column. A K28.5 does not have a balanced 10-bit code group (equal number of 1s and 0s), so the expected RD code group must toggle back and forth between RD– and RD+. At time n + 3, the 8B/10B decoder received a RD+ K28.5 code group (10'h283), which makes the current running disparity negative. At time n + 4, because the current disparity is negative, a K28.5 from the RD– column is expected, but a K28.5 code group from the RD+ is received instead. This prompts rx_disperr to go high during time n + 4 to indicate that this particular K28.5 code group had a disparity error. The current running disparity at the end of time n + 4 is negative because a K28.5 from the RD+ column was received. Based on the current running disparity at the end of time n + 5, a positive disparity K28.5 code group (from the RD–) column is expected at time n + 5.

*Figure 1–52. Disparity Error Detection*
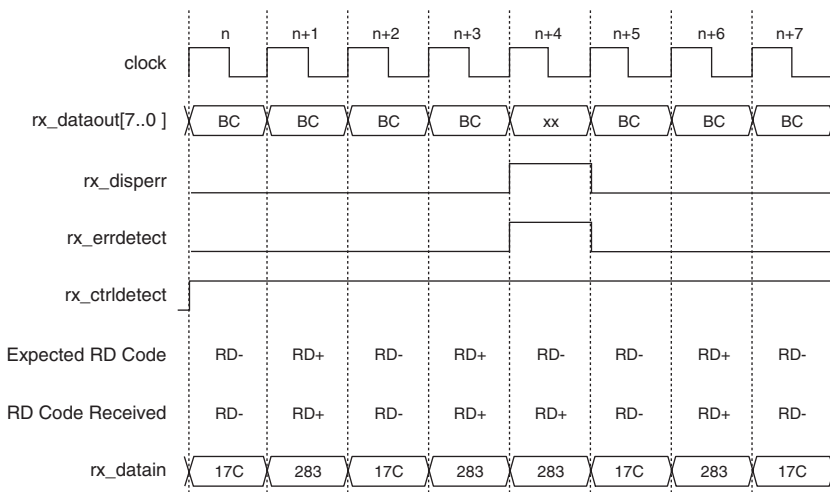
*Reset Condition*

The reset for the 8B/10B decoder block is derived from the receiver digital reset (`rx_digitalreset`). When `rx_digitalreset` is asserted, the 8B/10B decoder block resets. In reset, the disparity registers are cleared and the outputs of the 8B/10B decoder block are driven low. After reset, the 8B/10B decoder starts with unknown disparity, depending on the disparity of the data it receives. The decoder calculates the initial running disparity based on the first valid code group received.

☞ The receiver block must be word aligned after reset before the 8B/10B decoder can decode valid data or control codes. If word alignment has not been achieved, the data from the 8B/10B decoder should be considered invalid and discarded.

*Polarity Inversion*

The 8B/10B decoder has a PCI Express compatible polarity inversion on the data bus prior to 8B/10B decoding. This polarity inversion inverts the bits of the incoming data stream prior to the 8B/10B decoding block to fix potential P-N polarity inversion on the differential input buffer. You use the optional `pipe8b10binvpolarity` port to invert the inputs to the 8B/10B decoder dynamically from the PLD.

## Byte Deserializer

The byte deserializer (Figure 1–53) takes in 8- or 10-bit wide data from the 8B/10B decoder and deserializes it into 16- or 20-bit wide data at half the speed. This allows clocking the PLD-transceiver interface at half the speed as compared to the receiver PCS logic. The byte deserializer is bypassed in GIGE mode.

*Figure 1–53. Byte Deserializer*
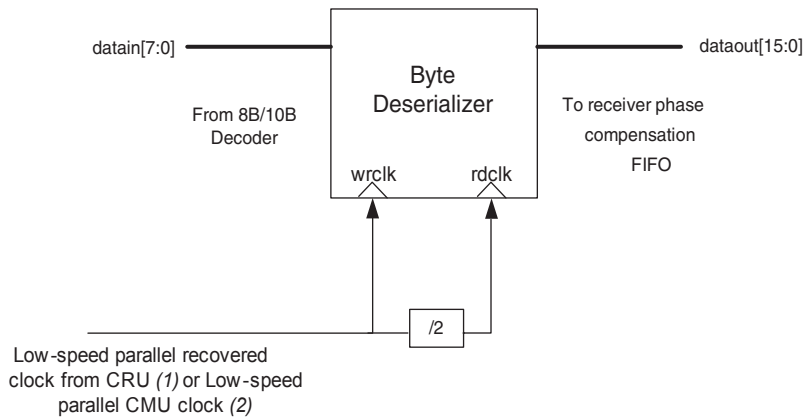


*Notes to Figure 1–53:*
(1) Write port is clocked by low-speed parallel recovered clock if rate matcher is not used.
(2) Write port is clocked by low-speed parallel CMU clock if rate matcher is used.

If the byte deserializer is used, the byte ordering at the receiver output might be different than what was transmitted. Figure 1–54 shows the 16-bit transmitted data pattern with A at the lower byte, followed by B at the upper byte. C and D follow in the next lower and upper bytes, respectively. At the byte deserializer, byte A arrives when it is stuffing the upper byte instead of stuffing the lower byte. This is a non-deterministic swap because it depends on PLL lock times and link delay. Implement byte-ordering logic in the PLD to correct this situation.

*Figure 1–54. Intended Transmitted Pattern and Incorrect Byte Position at Receiver After Byte Serializer*



## Receiver Phase Compensation FIFO Buffer

The receiver phase compensation FIFO buffer (Figure 1–55) is located at the FPGA logic array interface in the receiver block and is used to compensate for phase difference between the receiver clock and the clock from the PLD. The receiver phase compensation FIFO buffer operates in

two modes: low latency and high latency. In low latency mode, the FIFO buffer is four words deep. The Quartus II software chooses the low latency mode automatically for every mode except the PCI-Express PIPE mode (which automatically uses high latency mode). In high latency mode, the FIFO buffer is eight words deep.

*Figure 1–55. Receiver Phase Compensation FIFO Buffer*
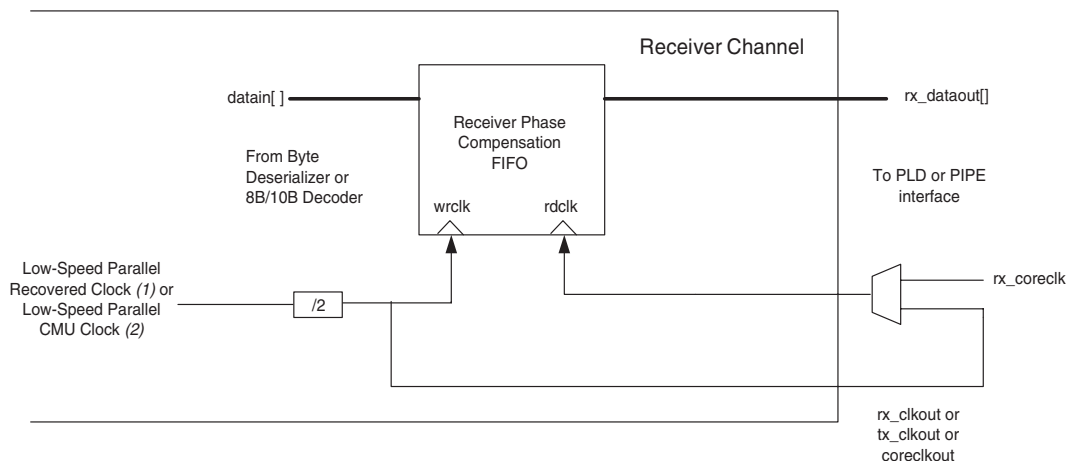


*Notes to Figure 1–55:*
(1)    Write port is clocked by low-speed parallel recovered clock when rate matcher is not used.
(2)    Write port is clocked by low-speed parallel CMU clock when rate matcher is used.

In Basic mode, the write port is clocked by the recovered clock from the CRU. This clock is half the rate if the byte deserializer is used. The read clock is clocked by the associated channel's recovered clock.

☞     The receiver phase compensation FIFO is always used and cannot be bypassed.

In four-channel (×4) bonding mode, all the read pointers are derived from a common source so that there is no need to synchronize the data of each channel in the PLD logic.

*Receiver Phase Compensation FIFO Error Flag*

Depending on the transceiver configuration, the write port of the receiver phase compensation FIFO can be clocked by either the recovered clock (rx_clk) or transmitter PLL output clock (tx_clkout or coreclkout). The read port can be clocked by the recovered clock (rx_clkout), transmitter PLL output clock (tx_clkout or

coreclkout) or a PLD clock. In all configurations, the write clock and the read clock must have 0 PPM difference to avoid overrun/underflow of the phase compensation FIFO.

An optional debug_rx_phase_comp_fifo_error port is available in all modes to indicate receiver phase compensation FIFO overrun/underflow condition. debug_rx_phase_comp_fifo_error is asserted high when the phase compensation FIFO gets either full or empty. This feature is useful to verify the phase compensation FIFO overrun/underflow condition as a probable cause of link errors.

# PLD-Transceiver Interface Clocking

The transmitter phase compensation FIFO present at each channel's PLD-transmitter interface compensates for the phase difference between the PLD clock that produces the data to be transmitted and the transmitter PCS clock. The receiver phase compensation FIFO present at each channel's PLD-receiver interface compensates for the phase difference between the PLD clock that processes the received data and the receiver PCS clock.

Depending on the functional mode, the Quartus II software automatically selects appropriate clocks to clock the read port of the transmitter phase compensation FIFO and the write port of the receiver phase compensation FIFO.

The write clock of the transmitter phase compensation FIFO and the read clock of the receiver phase compensation FIFO are part of the PLD-transceiver interface clocks. Arria GX transceivers provide the following two options for selecting these PLD-transceiver interface clocks:

■ Automatic Phase Compensation FIFO clock selection
■ User Controlled Phase Compensation FIFO clock selection

The automatic phase compensation FIFO clock selection is a simpler option, but could lead to higher clock resource utilization as compared to user controlled phase compensation FIFO clock selection. This could be critical in designs with high clock resource requirements.

### Automatic Phase Compensation FIFO Clock Selection

If you do not instantiate the tx_coreclk and rx_coreclk ports for the Arria GX transceiver instance in the MegaWizard Plug-In Manager, the Quartus II software automatically selects appropriate clocks to clock the write port of the transmitter phase compensation FIFO and the read clock of the receiver phase compensation FIFO.

Table 1–8 lists the clock sources that the Quartus II software automatically selects for the transmitter and receiver phase compensation FIFOs, depending on the functional mode.

| *Table 1–8. Clock Sources for the Transmitter and Receiver Phase Compensation FIFOs* | | |
|---|---|---|
| **Functional Mode** | **Write port clock selection for Transmitter Phase Compensation FIFO** | **Read port clock selection for Receiver Phase Compensation FIFO** |
| Individual-channel mode with rate matcher | `tx_clkout[0]` from channel 0 clocks the FIFO write port in all channels in the same transceiver block. | `tx_clkout[0]` from channel 0 clocks the FIFO read port in all channels in the same transceiver block. |
| Individual-channel mode without rate matcher | `tx_clkout[0]` from channel 0 clocks the FIFO write port in all channels in the same transceiver block. | `rx_clkout` from each channel clocks the FIFO read port of its associated channel. |
| Bonded-channel mode with/without rate matcher | `coreclkout` clocks the FIFO write port in all channels in the same transceiver block. | `coreclkout` clocks the FIFO read port in all channels in the same transceiver block. |

In an individual-channel mode without rate matcher (Serial RapidIO), a total of five global/regional clock resources per transceiver block are used by the PLD-transceiver interface clocks. Four clock resources are used by the `rx_clkout` signal of each channel being routed back to clock the read port of its receiver phase compensation FIFO. One clock resource is used by the `tx_clkout[0]` signal of Channel 0 being routed back to clock the write port of all transmitter phase compensation FIFOs in the transceiver block.

Figure 1–56 shows the minimum PLD-Interface clock utilization per transceiver block when configured in individual-channel mode without the rate matcher.

*Figure 1–56. Minimum PLD-Interface Clock Utilization Per Transceiver Block Without the Rate Matcher*
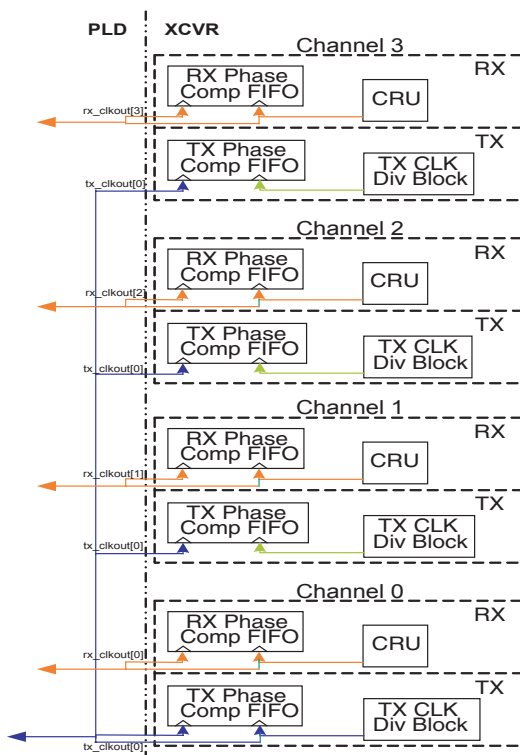


The PLD-transceiver clock utilization can be reduced by driving the transmitter and receiver phase compensation FIFOs with a single clock. This is possible only if the driving clock is frequency-locked to the transceiver output clocks (tx_clkout, coreclkout, or rx_clkout). To control the write and read clock selection for the transmitter and receiver phase compensation FIFO, you must instantiate the tx_coreclk and rx_coreclk ports for the transceiver channels.

## User Controlled Phase Compensation FIFO Clock Selection

Instead of the Quartus II software automatically selecting the write and read clocks of the transmitter and receiver phase compensation FIFOs, respectively, you can manually connect appropriate clocks by instantiating the `tx_coreclk` and `rx_coreclk` ports in the MegaWizard Plug-In Manager. For all like channels configured in the same functional mode and running off the same clock source, you can connect the `tx_coreclk` and `rx_coreclk` ports of all channels together and drive them using the same clock source. You can use a PLD clock input pin or a transceiver clock (`tx_clkout[0]/coreclkout/rx_clkout`) to clock the `tx_coreclk/rx_coreclk` ports (Figure 1–57).

*Figure 1–57. User Controlled Phase Compensation FIFO Clock*
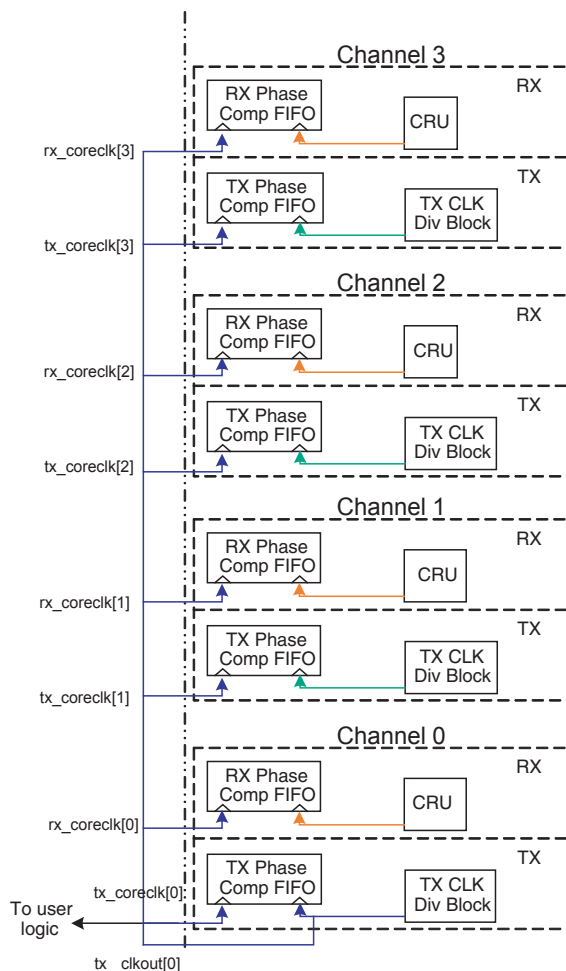


⟹ If the rx_clkout signal is used as a driver, it can only drive the rx_coreclk ports. It cannot drive the tx_coreclk ports. If tx_coreclk and rx_coreclk need to be driven with the same clock, you must use the tx_clkout signal as the clock driver.

If the clock signal on tx_coreclk is used to clock the write side of the transmitter phase compensation FIFO, you must make sure that it is frequency locked to the transmitter PCS clock reading from the FIFO. If the clock signal on rx_coreclk is used to clock the read side of the

receiver phase compensation FIFO, you must make sure that it is frequency locked to the receiver PCS clock writing into the FIFO. Any frequency differences may cause data corruption.

To help guard against incorrect usage, the use of the `tx_coreclk` and `rx_coreclk` options requires clock assignments in the assignment organizer. If no assignments are used, the Quartus II software will issue a compilation error.

There are four settings to enable the PLD interface clocking options:

■ **Stratix II GX/Arria GX GXB Shared Clock Group Setting**
■ **Stratix II GX/Arria GX GXB Shared Clock Group Driver Setting**
■ **Stratix II GX/Arria GX 0PPM Clock Group Setting**
■ **Stratix II GX/Arria GX 0PPM Clock Group Driver Setting**

There are two main settings, Shared Clock and 0 PPM Clock, each with a driver and clock group setting. When specifying clock groups, an integer identifier is used as the group name to differentiate the different clock group settings from each another.

The **Stratix II GX/Arria GX GXB Shared Clock Group Setting** is the safest assignment. The Quartus II compiler analyzes the netlist during compilation to ensure transmitter channel members are derived from the same source. The Quartus II software gives a fitting error for incompatible assignments. The software cannot check for the output of the receiver frequency locked to the driving clock as the exact frequency is dictated by the upstream transmitter's source clock. You must ensure that the `rx_coreclk` is derived from the same source clock as the upstream transmitter.

The **Stratix II GX/Arria GX GXB Shared Clock Group Driver Setting** assignment must be made to the source channel of the `tx_clkout` or `coreclkout`. Specifying anything but the transmitter channels (the source for the `tx_clkout` or `coreclkout`) results in a Fitter error. If the source clock is not from `tx_clkout` or `coreclkout` (for example, the source is from `rx_clkout` or from a PLD clock input), the 0 PPM setting must be used instead.

For example, in a synchronous system, the transmitter and receiver are running off the same clock. To make `tx_clkout[0]` the clock driver, the **Stratix II GX/Arria GX GXB Shared Clock Group Driver Setting** is made in the assignment editor on the `tx_dataout[0]` name. You can use a group identifier value of "1" to identify the group that this driver feeds. The **Stratix II GX/Arria GX GXB Shared Clock Group Setting** is made to all the `rx_datain` channels that the `tx_dataout[0]` output clock drives.

☞ The other `tx_dataout` channels do not need an assignment because the Quartus II software automatically groups the like transmitters in a transceiver block. A group identifier value of "1" is also made to the `rx_datain` assignments.

The assignments in the Assignment Editor are shown in Table 1–9.

| Table 1–9. Assignment Editor | |
|---|---|
| To: | `tx_dataout[0]` |
| Assignment name: | Stratix II GX/Arria GX GXB Shared Clock Group Driver Setting |
| Value: | 1 |
| To: | `rx_datain[]` (note that the [] signifies the entire `rx_datain` group) |
| Assignment name: | Stratix II GX/Arria GX GXB Shared Clock Group Setting |
| Value: | 1 |

The **Stratix II GX/Arria GX 0PPM Clock Group Setting** is for more advanced users that know the clocking configuration of the entire system and want to reduce the PLD global clock resource and PLD interface clock resource utilization. The Quartus II compiler does not perform any checking on the clock source. It is up to you to ensure that there is no frequency difference from the associated transceiver clock of the group and the driving clock to the `tx_coreclk` and `rx_coreclk` ports.

The **Stratix II GX/Arria GX 0PPM Clock Group Driver Setting** can be used with any of the transceiver output clocks (`tx_clkout`, `rx_clkout`, and `coreclkout`) as well as any PLD clock input pins, transceiver dedicated REFCLK pin, or PLD PLL output. User logic cannot be used as a driver. As with the shared clock group setting, the driver setting for the transceiver output clocks is made to the associated channel. For example, for `tx_clkout` or `coreclkout`, the transmitter channel name is specified. When the `rx_clkout` is the driver, the receiver channel name of the associated `rx_clkout` is specified. For the PLD input clock pins and the transceiver REFCLK pins, the name of the clock pin can be specified. For the PLL output, the PLL clock output port of the PLL can be found in the Node Finder and entered as the driver name. An integer value is specified for the group identification.

The **Stratix II GX/Arria GX 0PPM Clock Group Setting** is made to the transmitter or receiver channel names.

The assignments in the Assignment Editor are shown in Table 1–10.

| Table 1–10. Assignment Editor | |
|---|---|
| To: | `tx_dataout[0]`, `pld_clk_pin_name`, `refclk_pin`, and `pll_outclk` |
| Assignment name: | Stratix II GX/Arria GX GXB 0PPM Clock Group Driver Setting |
| Value: | 1 |
| To: | `rx_datain[]` and `tx_dataout[]` |
| Assignment name: | Stratix II GX/Arria GX GXB 0PPM Clock Group Setting |
| Value: | 1 |

For a complete set of features supported in each protocol, refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*.

## Loopback Modes

There are several loopback modes available on the Arria GX transceiver block that allow you to isolate portions of the circuit. All paths are designed to run up to full speed. The available loopback paths are:

■ Serial loopback available in all functional modes except PCI Express (PIPE)
■ Reverse serial loopback available in Basic mode with 8B/10B
■ PCI Express PIPE reverse parallel loopback available in PCI Express protocol
■ Reverse serial pre-CDR loopback available in Basic mode with 8B/10B Reverse serial loopback available in Basic mode with 8B/10B

### Serial Loopback

Figure 1–58 shows the data path for serial loopback. A data stream is fed to the transmitter from the FPGA logic array and has the option of utilizing all the blocks in the transmitter. The data, in serial form, then traverses from the transmitter to the receiver. The serial data is the data that is transmitted from the Arria GX device. Once the data enters the receiver in serial form, it can use any of the receiver blocks and is then fed into the FPGA logic array.

Use the `rx_seriallpbken` port to dynamically enable serial loopback on a channel by channel basis. When `rx_seriallpbken` is high, all blocks that are active when the signal is low are still active. When the serial loopback is enabled, the `tx_dataout` port is still active and drives out the output pins.

Serial loopback is often used to check the entire path of the transceiver. The data is retimed through different clock domains and an alignment pattern is still necessary for the word aligner.

*Figure 1–58. Arria GX Block in Serial Loopback Mode*
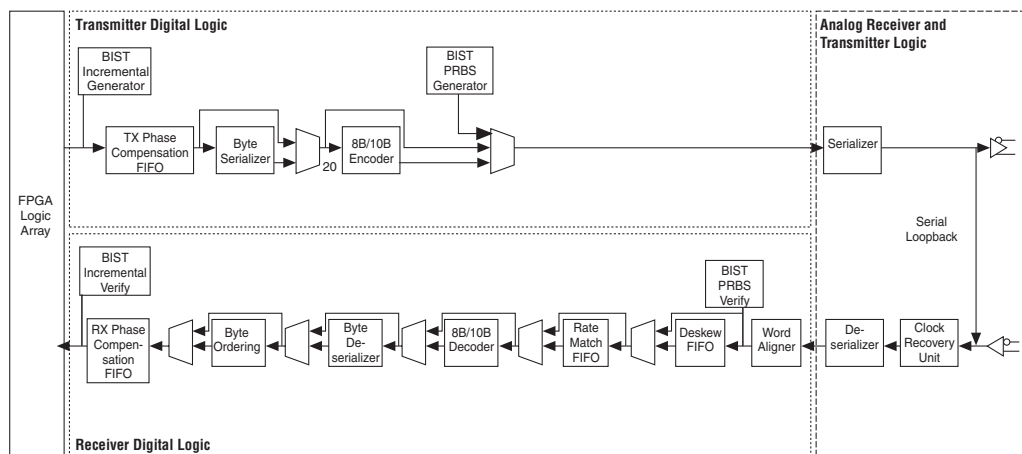


## PCI Express PIPE Reverse Parallel Loopback

Figure 1–59 shows the data path for the PCI Express PIPE reverse parallel loopback. This data path is not flexible because it must be compliant with the PCI Express PIPE specification. The data comes in from the `rx_datain` ports. The receiver uses the CRU, deserializer, word aligner, and rate matching FIFO buffer, loops back to the transmitter serializer, and then goes out the transmitter `tx_dataout` ports. The data also goes to the PLD fabric on the receiver side to the `tx_dataout` port. The deskew FIFO buffer is not enabled in this loopback mode. This loopback mode is optionally controlled dynamically through the `tx_detectrxloopback` port.

☞   This is the only loopback allowed in the PIPE mode.

*Figure 1–59. Arria GX Block in PCI Express PIPE Reverse Parallel Loopback Mode*
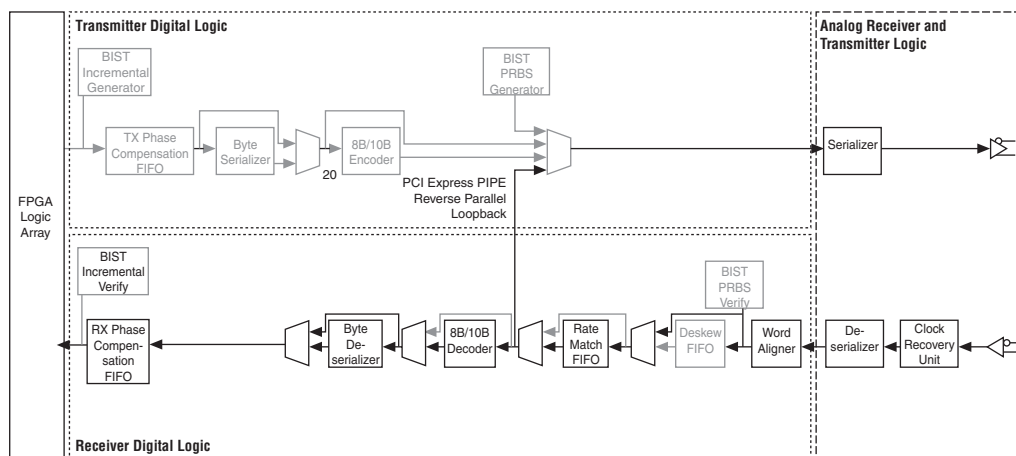


## Reverse Serial Loopback

Reverse serial loopback is a subprotocol in Basic mode. It requires 8B/10B, and the word aligner pattern of K28.5. No dynamic pin control is available to select or deselect reverse serial loopback. The active block of the transmitter is only the buffer. The data sent to the receiver is retimed with the recovered clock and sent out to the transmitter.

The data path for reverse serial loopback is shown in Figure 1–60. Data comes in from the `rx_datain` ports in the receiver. The data is then fed through the CDR block in serial form directly to the `tx_dataout` ports in the transmitter block.

You can enable reverse serial loopback for all channels through the MegaWizard Plug-In Manager. Any pre-emphasis setting on the transmitter buffer is ignored in reverse serial loopback. The data flows through the active blocks of the receiver and into the logic array.

Reverse serial loopback is often implemented when using a bit error rate tester (BERT).

*Figure 1–60. Arria GX Block in Reverse Serial Loopback Mode*
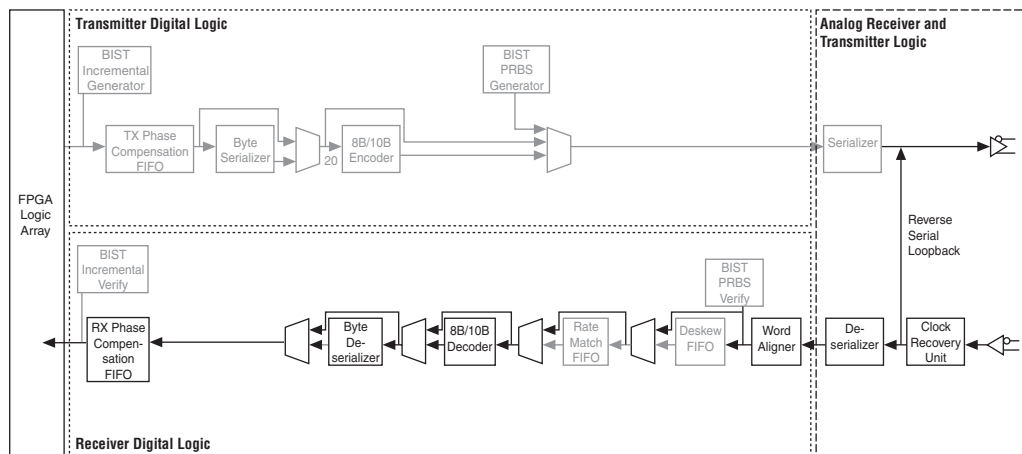
## Reverse Serial Pre-CDR Loopback

The reverse serial pre-CDR loopback uses the analog portion of the transceiver. An external source (pattern generator or transceiver) generates the source data. The high-speed serial source data arrives at the high-speed differential receiver input buffer, loops back before the CRU unit, and is transmitted though the high-speed differential transmitter output buffer. This loopback mode is for test or verification use only to verify the signal being received after the gain and equalization improvements of the input buffer. The signal at the output is not exactly what is received, because the signal goes through the output buffer and the VOD is changed to the VOD setting level. The pre-emphasis settings have no effect.

*Figure 1–61. Arria GX Block in Reverse Serial Pre-CDR Loopback Mode*
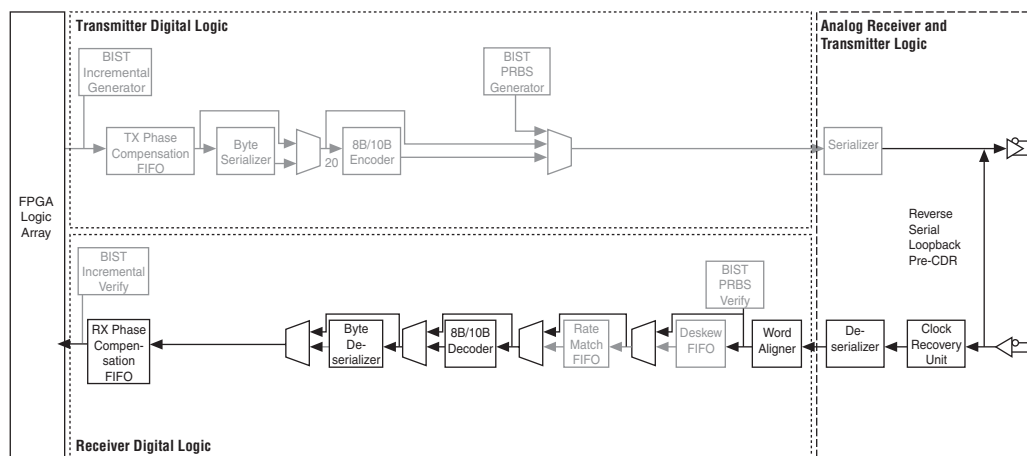


### Incremental Pattern Generator

The incremental data generator sweeps through all the valid 8B/10B data and control characters. This mode is only available in Basic mode with the BIST/parallel loopback subprotocol in the Quartus II software. You can also enable the incremental BIST verifier to perform a quick verification of the 8B/10B encoder/decoder paths.

In incremental mode, the BIST generator sends out the data pattern in the following sequence: K28.5 (comma), K27.7 (start of frame, SOF), Data (00 FF incremental), K28.0, K28.1, K28.2, K28.3, K28.4, K28.6, K28.7, K23.7, K30.7, K29.7 (end of frame, EOF), and then repeats. You must enable the 8B/10B encoder for proper operation. No dynamic control pin is available to enable or disable the loopback. Test result pins are rx_bistdone and rx_bisterr. The rx_bistdone signal goes high at the end of the sequence. If the verifier detects an error before it is finished, rx_bisterr pulses high as long as the data is in error.

## Built-In Self Test Modes

In addition to the regular data flow blocks, each transceiver channel contains an embedded built-in self test (BIST) generator and corresponding verifier block that you can use for quick device and setup verification ( Figure 1–62). The generators reside in the transmitter block and the verifier in the receiver block. The generators can generate PRBS patterns. The verifiers are only available for the PRBS patterns. The BIST modes are only available as subprotocols under Basic mode.

*Figure 1–62. Built-In Self Test Mode*
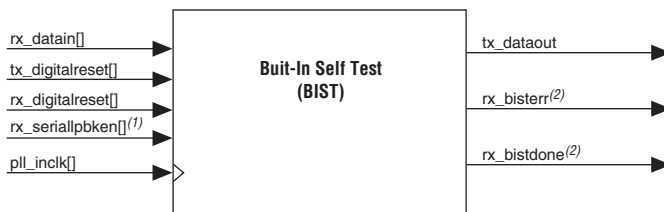


*Notes to Figure 1–62:*
(1)  `rx_seriallpbken[]` is required in PRBS.
(2)  `rx_bisterr[]` and `rx_bistdone[]` are only available in PRBS and BIST modes.

Figure 1–63 shows the PRBS blocks with loopback used in the transceiver channel.

*Figure 1–63. PRBS Blocks With Loopback in Transceiver Channel*
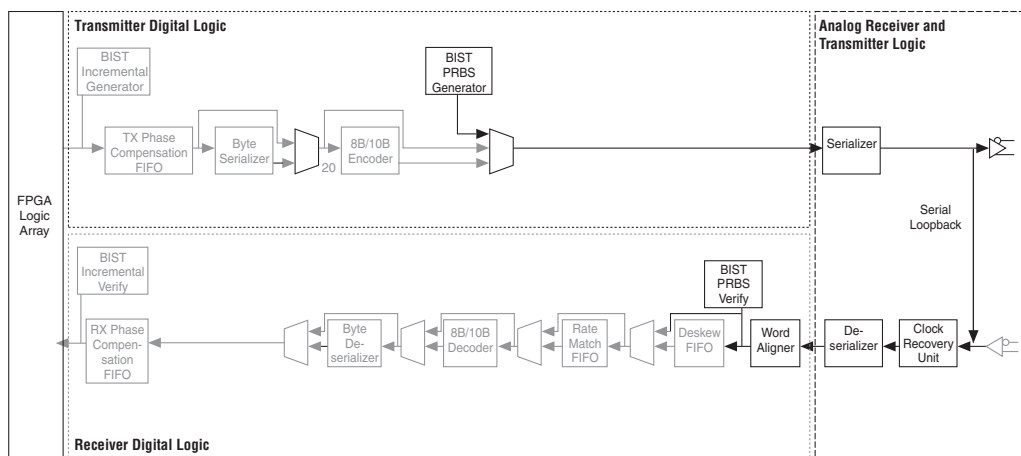


## BIST in Basic Mode

Basic mode supports PRBS10 pattern generation and verification. PRBS10 is supported with or without serial loopback.

☞  The PRBS10 pattern is only available when the SERDES factor is 10 bits.

Table 1–11 shows the BIST patterns for Basic mode.

| Table 1–11. Available BIST Patterns in Basic Mode | | | | | |
|---|---|---|---|---|---|
| **Pattern** | **Word Aligner Alignment Pattern** | **Byte Order Align Pattern** | **Description** | **Basic Mode** | |
| | | | | **8 Bit** | **10 Bit** |
| PRBS10 | 10'h3FF | N/A | $X^{10} + X^7 + 1$ | — | ✓ |

### PRBS10

Pseudo-Random Bit Sequences (PRBS) are commonly used in systems to verify the integrity and robustness of the data transmission paths. When the SERDES factor is 10, use the PRBS10 pattern. The PRBS generator yields 2^10-1 unique patterns. You can use PRBS with or without serial loopback. In PRBS/ serial loopback mode, the `rx_seriallpbken` signal is available. In the PRBS/no loopback mode, this control signal is not available.

You enable PRBS mode in the Quartus II ALT2GXB MegaWizard Plug-In Manager. PRBS10 does not use the 8B/10B encoder and decoder. The 8B/10B encoder and decoder are bypassed automatically in the PRBS mode.

The advantage of using a PRBS data stream is that the randomness yields an environment that stresses the transmission medium. In the data stream, you can observe both random jitter and deterministic jitter using a time interval analyzer, bit error rate tester, or oscilloscope.

The PRBS verifier can provide a quick check through the non-8B/10B path of the transceiver block. The PRBS verifier is active once the receiver channel is synchronized. Set the alignment pattern to 10'h3FF for the 10-bit SERDES modes.

The verifier stops checking the patterns after receiving all the PRBS patterns (1023 patterns for 10-bit mode). The `rx_bistdone` signal goes high, indicating that the verifier has completed. If the verifier detects an error before it is finished, `rx_bisterr` pulses high for the time the data is incorrect. Use the `rx_digitalreset` signal to re-start the PRBS verification.

The 8B/10B encoder is enabled, so the data stream is DC balanced. 8B/10B encoding guarantees a run length of less than 5 UI, which yields a less stressful pattern versus the PRBS data. However, since the PRBS generator bypasses the 8B/10B paths, the incremental BIST can test this path.

# Calibration Blocks

The Arria GX gigabit transceiver block contains calibration circuits to calibrate the on-chip termination, the PLLs, and the output buffers. The calibration circuits are divided into two main blocks: the PLL and output buffer calibration block and the termination resistor calibration block (refer to Figure 1–64). Each transceiver block contains a PLL and output buffer calibration block that calibrates the PLLs and output buffers within that particular transceiver block. Each device contains one termination resistor calibration block that calibrates all the termination resistors in the transceiver channels of the entire device.

*Figure 1–64. Calibration Block*
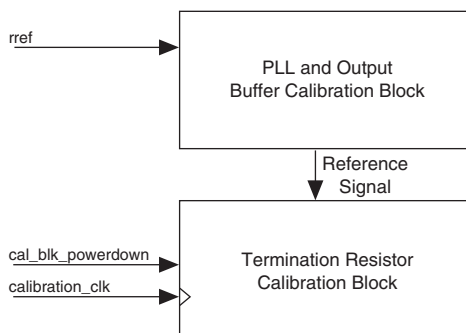


## PLL and Output Buffer Calibration Block

Each Arria GX transceiver block contains a PLL and output buffer calibration circuit to counter the effects of PVT (process, voltage, and temperature) on the PLL and output buffer. Each transceiver block's calibration circuit uses a voltage reference derived from an external reference resistor. There is one reference resistor required for each active transceiver block in Arria GX devices. Unused transceiver blocks (except the transceiver blocks feeding the termination resistor calibration block) can be left unconnected or be tied to the 3.3 V transceiver analog $V_{CC}$ (if the transceiver block's 3.3 V analog supply is connected to 3.3 V).

## Termination Resistor Calibration Block

The Arria GX transceiver's on-chip termination resistors in the transceiver channels of the entire device are calibrated by a single calibration block. This block ensures that process, voltage, and temperature variations do not have an impact on the termination resistor value. There is only one termination resistor calibration block per device.

The calibration block uses the reference resistor of transceiver block 0 or transceiver block 1, depending on the device and package. The calibration block uses the reference resistor in transceiver block 0 for EP1AGX20/35 and EP1AGX50/60 devices (except in the F484 package). The reference resistor in transceiver block 1 is used for EP1AGX20/35 and EP1AGX50/60 devices in the F484 package, and for the EP1AGX90 device. A reference resistor must be connected to either transceiver block 0 or transceiver block 1 to ensure proper operation of the calibration block, whether or not the transceiver block is in use. Failing to connect the reference resistor of the transceiver block feeding the calibration block results in incorrect termination values for all the termination resistors in the transceivers of the entire device.

The termination resistor calibration circuit requires a calibration clock. You can use a global clock line if the REFCLK pins are used for the reference clock. You can instantiate a calibration clock port in the MegaWizard Plug-In Manager to supply your own clock through the cal_blk_clk port.

The frequency range of the cal_blk_clk is 10 MHz to 125 MHz. If there are no slow speed clocks available, use a divide down circuit (for example, a ripple counter) to divide the available clock to a frequency in that range. The quality of the calibration clock is not an issue, so PLD local routing is sufficient to route the calibration clock.

For multiple ALT2GXB instances in the same device, if all the instances are the same, the calibration block must be active and the cal_blk_clk port of all instances must be tied to a common clock. Physically, there is one cal_blk_clk port per device. The Quartus II software provides an error message if the cal_blk_clk port is tied to different clock sources, because this would be impossible to fit into a device. If there are different configurations of the ALT2GXB instance, only one must have the calibration block instantiated. If multiple instances of the ALT2GXB custom megafunction variation have the calibration block instantiated, then all the cal_blk_clk ports must be tied to the same clock source.

The calibration block can be powered down through the optional cal_blk_powerdown port (this is an active low input). Powering down the calibration block during operations may yield transmit and receive data errors. Only use this port to reset the calibration block to initiate a recalibration of the termination resistors to account for variations in temperature or voltage. The minimum pulse duration for this port is determined by characterization. If external termination is used on all signals, the calibration block in ALT2GXB need not be used.

**Referenced Documents**

This chapter references the following documents:

- *Arria GX Transceiver Protocol Support and Additional Features*
- *Specifications and Additional Information*

## Document Revision History

Table 1–12 shows the revision history for this chapter.

| *Table 1–12. Document Revision History* | | |
|---|---|---|
| **Date and Document Version** | **Changes Made** | **Summary of Changes** |
| May 2008, v2.0 | ● Added sections "Transmitter PLL Bandwidth Setting", "Central Clock Divider Block", "Transmitter Local Clock Divider Block", "Clock Synthesis", "Transceiver Clock Distribution", "Single Lane", "Four-Lane Mode", "Channel Clock Distribution", "Individual Channels Clocking", "Transmitter Clocking (Bonded Channels)", "Transmitter Force Disparity", "Transmitter Bit Reversal", "Transmitter Termination", "PCI Express Receiver Detect", "PCI Express Electrical Idle", "Receiver Buffer", "Receiver Termination", "Signal Threshold Detection Circuit", "Receiver Common Mode", "Programmable Equalization", "Clock Synthesis", "PPM Frequency Threshold Detector", "Receiver Bandwidth Type", "Basic Mode", "Pattern Detector Module", "7-Bit Pattern Mode", "10-Bit Pattern Mode", "7-bit Alignment Mode", "Manual 10-Bit Alignment Mode", "Manual Bit-Slip Alignment Mode", "Synchronization State Machine Mode", "Run Length Checker", "Receiver Bit Reversal", "Channel Aligner (Deskew)", "Basic Mode General Rate Matching", "Polarity Inversion", "Receiver Phase Compensation FIFO Error Flag", "Serial Loopback", "PCI Express PIPE Reverse Parallel Loopback", "Reverse Serial Loopback", "Reverse Serial Pre-CDR Loopback", "Built-In Self Test Modes", "BIST in Basic Mode", "PRBS10", "Calibration Blocks", "PLL and Output Buffer Calibration Block", and "Termination Resistor Calibration Block"<br>● Updated sections "Building Blocks", "Port List", "Dedicated Reference Clock Input Pins", "Byte Serializer", "8B/10B Encoder", "Transmitter Polarity Inversion", "Serializer", "Transmitter Buffer", "Receiver Channel Architecture", "Code Group Error Detection","Disparity Error Detection", "Byte Deserializer", "Receiver Phase Compensation FIFO Buffer", and "Loopback Modes" | Major update. Addition of new material. |
| August 2007, v1.2 | Added the "Referenced Documents" section. | — |
| | Minor text edits. | — |
| June 2007 v1.1 | Added GIGE information. | — |
| May 2007 v1.0 | Initial release. | — |

## Introduction

Arria™ GX transceivers have a dedicated physical coding sublayer (PCS) and physical media attachment (PMA) circuitry to support PCI Express (PIPE), Gigabit Ethernet (GIGE), and Serial RapidIO® protocols.

Table 2–1 lists the Arria GX transceiver datapath modules employed in each mode.

| Table 2–1. Arria GX Transceiver Datapath Modules | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Functional Mode | Transmitter /Receiver Phase Compensation FIFO | Byte Serializer/ Deserializer | 8B/10B Encoder/ Decoder | Word Aligner | Rate Matcher | PLD-Transceiver Interface Width (bits) | PLD-Transceiver Interface Frequency (MHz) | PCS Frequency (MHz) |
| PCI Express (PIPE) | ✔ | ✔ | ✔ | ✔ | ✔ *(1)* | 16 | 125 | 250 |
| GIGE | ✔ | — | ✔ | ✔ | ✔ | 8 | 125 | 125 |
| Serial RapidIO (1.25Gbps) | ✔ | ✔ | ✔ | ✔ | — | 16 | 62.5 | 125 |
| Serial RapidIO (2.5Gbps) | ✔ | ✔ | ✔ | ✔ | — | 16 | 125 | 250 |
| Serial RapidIO (3.125Gbps) | ✔ | ✔ | ✔ | ✔ | — | 16 | 156.25 | 312.5 |
| SDI - HD (1.483Gbps) | ✔ | — | — | Bit-Slip | — | 10/20 | 148.3 | 148.3/296.6 |
| SDI - HD (1.485Gbps) | ✔ | — | — | Bit-Slip | — | 10/20 | 148.5 | 148.5/297 |
| SDI - 3G (2.967Gbps) | ✔ | — | — | Bit-Slip | — | 20 | 148.35 | 296.7 |
| SDI - 3G (2.97Gbps) | ✔ | — | — | Bit-Slip | — | 20 | 148.5 | 297 |
| XAUI (3.125Gbps) | ✔ | ✔ | ✔ | ✔ | ✔ | 16 | 156.25 | 312.5 |

*Note to Table 2–1:*

(1) The rate matcher can be bypassed in low-latency (synchronous) PCI Express (PIPE) mode.

# PCI Express (PIPE) Mode

PCI Express is an evolution of peripheral component interconnect (PCI). PCI is bandwidth-limited for today's applications because it relies on synchronous single-ended type signaling with a wide multi-drop data bus. Clock and data-trace matching is required with PCI. PCI Express uses differential serial signaling with an embedded clock to enable an effective data rate of 2 Gbps per lane to overcome the limitations of PCI.

Arria GX transceivers support ×1 (single-lane) and ×4 (four-lane) link widths when configured in PCI Express (PIPE) mode. The Arria GX family supports up to twelve duplex (transmitter and receiver) ×1 links and up to three ×4 links per device. Transceiver channels configured in ×4 PCI Express (PIPE) mode must be physically located in the same transceiver block with logical Lane 0 assigned to physical Channel 0, logical Lane 1 assigned to physical Channel 1 and so on.

In addition to providing the transceiver PCS and PMA circuitry, Arria GX transceivers support the following protocol-specific features:

■ PCI Express synchronization state machine
■ Receiver detection
■ Electrical idle generation/detection
■ Beacon transmission
■ Polarity inversion
■ Power state management

☞ This section is organized into transmitter and receiver data path modules when configured for PCI Express (PIPE) mode. The description for each module only covers details specific to PCI Express (PIPE) functional mode support. Familiarity of PCI Express protocol and PCI Express (PIPE) specifications is assumed.

For a general description of each module, refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

## PCI Express (PIPE) Mode Transmitter Architecture

This section lists sub-blocks within the transmitter channel configured in PCI Express (PIPE) mode (Figure 2–1). The sub-blocks are described in order from the PLD transceiver parallel interface to the serial transmitter buffer.

*Figure 2–1. PCI Express (PIPE) Transmitter Architecture*
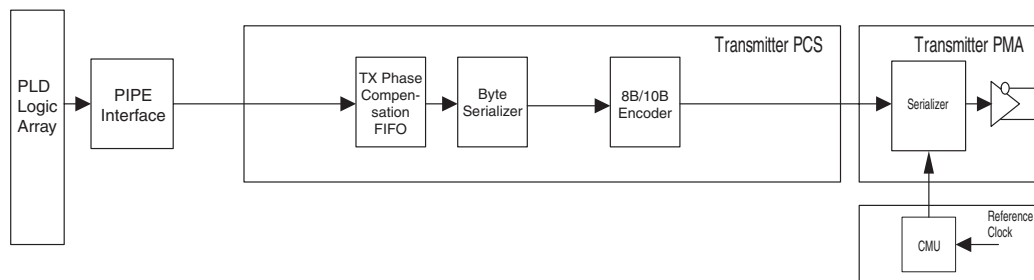


### Clock Multiplier Unit

The clock multiplier unit (CMU) takes in a reference clock and synthesizes the clocks that are used to clock the transmitter digital logic (PCS), the serializer, and the PLD-transceiver interface.

For more details about CMU architecture, refer to the Clock Multiplier Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

In ×1 PCI Express (PIPE) mode, the CMU block consists of the following components:

- Transmitter PLL that generates high-speed serial clock for the serializer
- Local clock divider block that generates low-speed parallel clock for transmitter digital logic and PLD-transceiver interface

In ×4 PCI Express (PIPE) mode, the CMU block consists of the following components:

- Transmitter PLL that generates high-speed serial clock for the serializer
- Central clock divider block that generates low-speed parallel clock for transmitter digital logic and PLD-transceiver interface of each channel in the transceiver block

**Input Reference Clock**

In PCI Express (PIPE) mode, the only supported input reference clock frequency is 100 MHz.

The reference clock input to the transmitter PLL can be derived from the following pins:

■ One of the two available dedicated reference clock input pins (REFCLK0 or REFCLK1) of the associated transceiver block
■ Inter-transceiver block lines driven by reference clock input pins of other transceiver blocks

☞ Altera recommends using the dedicated reference clock input pins (REFCLK0 or REFCLK1) to provide a reference clock for the transmitter PLL.

Table 2–2 specifies the input reference clock options available in PCI Express (PIPE) mode.

*Table 2–2. PCI Express (PIPE) Mode Input Reference Clock Specifications*

| Frequency | I/O Standard | Coupling | Termination |
|---|---|---|---|
| 100 MHz | 1.2V PCML, 1.5V PCML, 3.3V PCML, Differential LVPECL, LVDS | AC | On-chip |
| | HCSL *(1)* | DC *(2)* | Off-chip |

*Notes to Table 2–2:*
(1) In PCI Express (PIPE) mode, you have the option of selecting the HCSL standard for the reference clock if compliance to PCI Express is required. The Quartus® II software automatically selects DC coupling with external termination for the signal if configured as HCSL.
(2) Refer to Figure 2–2 for an example termination scheme.

Figure 2–2 shows an example termination scheme for the reference clock signal when configured as HCSL.

*Figure 2–2. DC Coupling and External Termination Scheme for PCI Express Reference Clock*
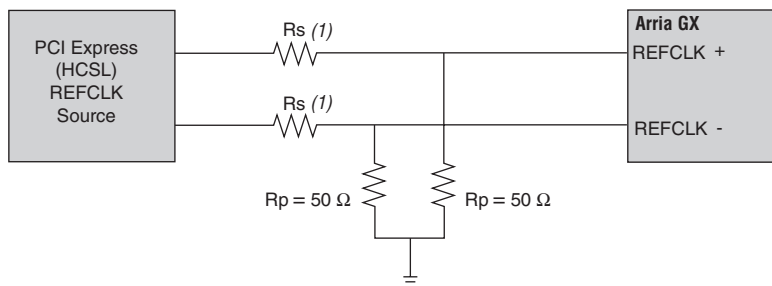


*Note to Figure 2–2:*
(1) Select resistor values as recommended by the PCI Express clock source vendor.

**Clock Synthesis**

In PCI Express (PIPE) mode, the reference clock pre-divider divides the 100-MHz input reference clock by two. The resulting 50-MHz clock is fed to the transmitter PLL. Because the transmitter PLL implements a half-rate VCO, it multiplies the 50 MHz input clock by 25 to generate a 1250-MHz high-speed serial clock. This high-speed serial clock feeds the central clock divider and four local clock dividers of the transceiver block.

In ×4 PCI Express (PIPE) mode, the central clock divider in the transceiver block divides the 1250-MHz clock from the transmitter PLL by five to generate a 250-MHz parallel clock. This low-speed parallel clock output from the central clock divider block is used to clock the transmitter digital logic (PCS) in all channels of the transceiver block. The central clock divider block also forwards the high-speed serial clock from the transmitter PLL to the serializer within each channel. Because all four channels in the transceiver block are clocked with the same clock, the channel-to-channel skew is minimized.

In ×1 PCI Express (PIPE) mode, the local clock divider in each channel of the transceiver block divides the 1250-MHz clock from the transmitter PLL by five to generate a 250-MHz parallel clock. This low-speed parallel clock output from the local clock divider block is used to clock the transmitter digital logic (PCS) of the associated channel. The local clock divider block also forwards the high-speed serial clock from the transmitter PLL to the serializer within its associated channel.

☞ The Quartus II software automatically selects the appropriate transmitter PLL bandwidth suited for the PCI Express (PIPE) data rate.

Figure 2–3 shows the CMU implemented in PCI Express (PIPE) mode.

*Figure 2–3. PCI Express (PIPE) Mode CMU*
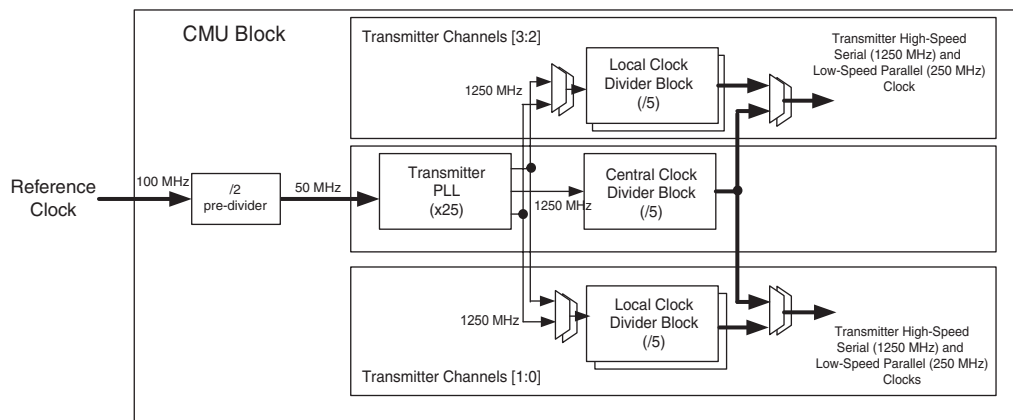


### Transmitter Phase Compensation FIFO Buffer

The transmitter phase compensation FIFO buffer compensates for the phase difference between the PLD clock that clocks in parallel data into the transmitter and the PCS clock that clocks the rest of the transmitter digital logic.

Refer to the Transmitter Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about transmitter phase compensation FIFO buffer architecture.

In PCI Express (PIPE) mode, the 250-MHz clock generated by the CMU clock divider block is divided by two. The resulting 125-MHz clock is used to clock the read port of the FIFO buffer. This 125-MHz clock is also forwarded to the PLD logic array (on the `tx_clkout` port in ×1 PCI Express (PIPE) mode or the `coreclkout` port in ×4 PCI Express (PIPE) mode). If the `tx_coreclk` port is not instantiated, the clock signal on the `tx_clkout` port of channel 0 is routed back to clock the write side of the transmitter phase compensation FIFO buffer in all channels with the transceiver block. The 16-bit PLD-transceiver interface clocked at 125-MHz results in an effective PCI Express (PIPE) data rate of 2 Gbps.

In PCI Express (PIPE) mode, the transmitter phase compensation FIFO is eight words deep. The latency through the FIFO is three to four PLD-transceiver interface clock cycles.

Figure 2–4 shows the block diagram of transmitter phase compensation FIFO in PCI Express (PIPE) mode.

*Figure 2–4. TX Phase Compensation FIFO in PCI Express (PIPE) Mode*
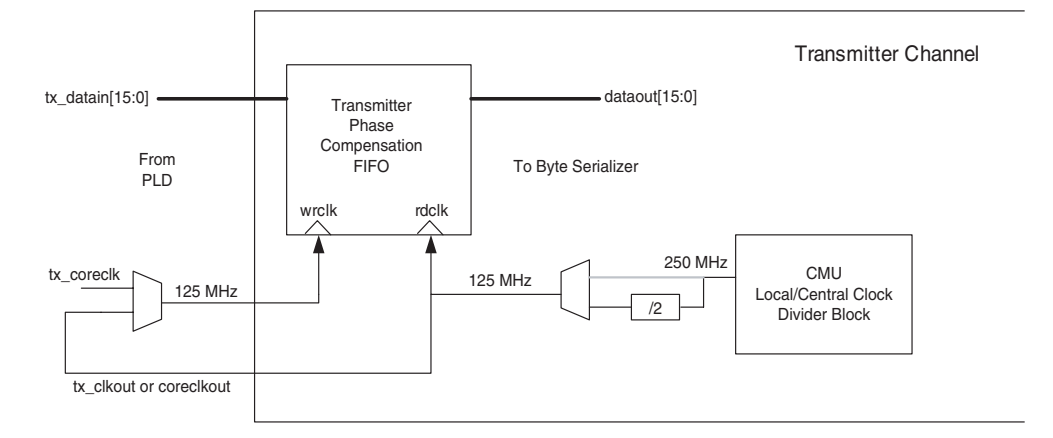


### Byte Serializer

In PCI Express (PIPE) mode, the PLD-transceiver interface data is 16-bits wide and is clocked into the transmitter phase compensation FIFO at 125 MHz. The byte serializer clocks in the 16-bit wide data from the transmitter phase compensation FIFO at 125 MHz and clocks out 8-bit data to the 8B/10B encoder at 250 MHz. This allows clocking the PLD-transceiver interface at half the speed.

For more details about byte serializer architecture, refer to the Byte Serializer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

The write port of the byte serializer is clocked by the divide-by-two version of the low-speed parallel clock from the CMU. The read port is clocked by the low-speed parallel clock from the CMU. The byte serializer clocks out the least significant byte (LSByte) of the 16-bit data first and the most significant byte (MSByte) last.

Figure 2–5 shows the block diagram of the byte serializer in PCI Express (PIPE) mode.

*Figure 2–5. Byte Serializer in PCI Express (PIPE) Mode*
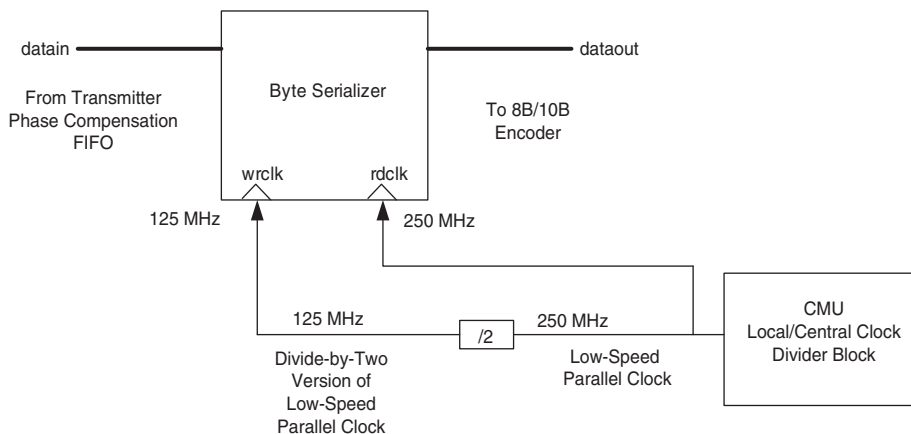


### 8B/10B Encoder

In PCI Express (PIPE) mode, the 8B/10B encoder clocks in 8-bit data and 1-bit control identifier from the byte serializer and generates 10-bit encoded data. The 10-bit encoded data is fed to the serializer.

For more details about the 8B/10B encoder functionality, refer to the 8B/10B Encoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

**Compliance Pattern Transmission Support**
PCI Express has an option to transmit a compliance pattern for testing purposes. The compliance pattern must be transmitted beginning with a negative disparity. In PCI Express (PIPE) mode, you set the negative disparity with the tx_forcedispcompliance port.

Asserting the tx_forcedispcompliance port sets the LSByte of the 16-bit PLD-transmitter interface data to be encoded with a negative disparity. The tx_forcedispcompliance port must be de-asserted after the first word of the compliance pattern is clocked into the transceiver.

☞ The compliance pattern generator is not part of the Arria GX transceiver and must be designed using the PLD logic. This feature allows you to begin the compliance pattern only with a negative disparity.

### Serializer

In PCI Express (PIPE) mode, the 10-bit encoded data from the 8B/10B encoder is clocked into the 10:1 serializer with the low-speed parallel clock at 250 MHz. The 10-bit data is clocked out of the serializer LSByte to MSByte at both edges of the high-speed serial clock at 1250 MHz. The resulting 2.5 Gbps serial data output of the serializer is fed into the transmitter output buffer.

Refer to the Serializer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about the serializer architecture.

### Transmitter Buffer

Table 2–3 shows the transmitter buffer settings when configured in PCI Express (PIPE) mode.

| Table 2–3. Transmitter Buffer Settings in PCI Express (PIPE) Mode | |
|---|---|
| **Settings** | **Value** |
| I/O Standard | 1.2-V PCML *(2)* |
| Programmable Differential Output Voltage ($V_{OD}$) | 320-960 mV |
| Common Mode Voltage ($V_{CM}$) | 600 mV *(1)* |
| Differential Termination | 100 Ω *(2)* |
| Programmable Transmitter Pre-Emphasis | Enabled *(3)* |
| $V_{CCH}$ (Transmitter Buffer Power) | 1.2 V |

Notes to *Table 2–3*:
(1) The common mode voltage ($V_{CM}$) is fixed in the MegaWizard® Plug-In Manager and cannot be changed.
(2) The I/O standard and differential termination settings are defaulted to 1.2-V PCML and 100 Ω, respectively. If you select any other setting for the I/O standard or differential termination in the Assignment Editor, the Quartus II compiler will issue an error message.
(3) The transmitter buffer has five programmable first post-tap pre-emphasis settings.

**Transmitter Electrical Idle**

In PCI Express (PIPE) mode, you can force the transmitter into electrical idle condition during P0 and P2 power state by asserting the `tx_forceelecidle` signal high. In electrical idle state, the transmitter buffer is tri-stated. The `tx_forceelecidle` signal must always be asserted high in P0 and P1 power states. Refer to "Power State Management" on page 2–22 for more details about PCI Express (PIPE) mode power states.

**Receiver Detect**

PCI Express Base Specification requires the transmitter to be capable of detecting a far-end receiver before beginning link training. Arria GX transceivers have dedicated receiver detect circuitry that is activated in PCI Express (PIPE) mode.

The receiver detect circuitry is available only in the P1 power state, and is set through the `tx_detectrxloopback` port, and requires a 125 MHz `fixedclk` signal. Refer to "Power State Management" on page 2–22 for more details about PCI Express (PIPE) mode power states.

In P1 power state, the transmitter output buffer is tri-stated, because the transmitter is in electrical idle. A high on the `tx_detectrxloopback` port triggers the receiver detect circuitry to alter the transmitter buffer common mode voltage. The sudden change in common mode voltage appears as a step voltage at the tri-stated transmitter buffer output. If a receiver (that complies with PCI Express input impedance requirements) is present at the far end, the time constant of the step voltage is higher. If a receiver is not present or is powered down, the time constant of the step voltage is lower. The receiver detect circuitry snoops the transmitter buffer output for the time constant of the step voltage to detect the presence of the receiver at the far end.

A high pulse is driven on the `pipephydonestatus` port and 3'b011 is driven on the `pipestatus` port (refer to "Receiver Status" on page 2–21) to indicate that a receiver has been detected. There is some latency after asserting the `tx_detectrxloopback` signal, before the receiver detection is indicated on the `pipephydonestatus` port.

☞ The `tx_forceelecidle` port must be asserted at least 10 parallel clock cycles prior to the `tx_detectrxloopback` port to ensure that the transmitter buffer is tri-stated.

**Beacon Transmission**

The beacon is an optional 30-kHz to 500-MHz in-band signal that wakes the receiver from a P2 power state. This signal is optional; the Arria GX device does not have dedicated beacon transmission circuitry. The Arria GX device supports the transmission of the beacon signal through

a 10-bit encoded code group that has a five 1's pulse (for example, K28.5) (10'b0101111100). Because the beacon signal is a pulse that ranges from 2 ns to 500 ns, sending out a K28.5 at 2.5 Gbps meets the lower requirement with its five 1's pulse. (Though other 8B/10B code groups might meet the beacon requirement, this document uses the K28.5 control code group as the beacon signal.) The beacon transmission takes place only in the P2 power state. The `tx_forceelecidle` port controls when the transmitter is in Electrical Idle or not. This port must be de-asserted in order to transmit the K28.5 code group for beacon transmission.

## PCI Express (PIPE) Mode Receiver Architecture

This section lists sub-blocks within the receiver channel configured in PCI Express (PIPE) mode (Figure 2–6). The sub-blocks are described in order from the serial receiver input buffer to the receiver phase compensation FIFO buffer at the transceiver-PLD interface.

*Figure 2–6. PCI Express (PIPE) Mode Receiver Architecture*
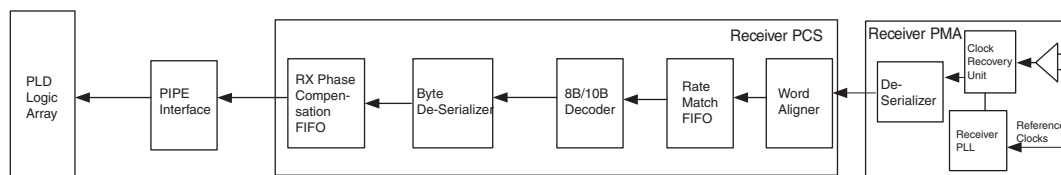


### Receiver Buffer

Table 2–4 shows the receiver buffer settings when configured in PCI Express (PIPE) mode.

*Table 2–4. Receiver Buffer Settings in PCI Express (PIPE) Mode (Part 1 of 2)*

| Settings | Value |
|---|---|
| I/O Standard | 1.2-V PCML, 1.5-V PCML, 3.3-V PCML, Differential LVPECL, LVDS |
| Input Common Mode Voltage (Rx $V_{CM}$) | 850 mV, 1200 mV *(1)* |
| Differential Termination | 100 Ω *(2)* |
| Programmable equalization | Enabled *(3)* |

**Table 2–4. Receiver Buffer Settings in PCI Express (PIPE) Mode (Part 2 of 2)**

| Settings | Value |
|----------|-------|
| Coupling | AC |

*Notes to Table 2–4:*

(1) The common mode voltage (Rx $V_{CM}$) is selectable in the MegaWizard® Plug-In Manager.
(2) The differential termination setting is defaulted to 100 Ω. If you select any other setting for differential termination in the Assignment Editor, the Quartus II compiler issues an error message.
(3) The receiver buffer has five programmable equalization settings.

**Signal Detect Threshold Circuitry**

In PCI Express (PIPE) mode, the receiver buffer incorporates a signal detect threshold circuitry. The signal detect threshold circuitry senses whether the specified threshold voltage level exists at the receiver buffer. This detector has a hysteresis response that filters out any high frequency ringing caused by inter symbol interference or high frequency losses in the transmission medium.

The `rx_signaldetect` signal indicates whether the signal at the receiver buffer conforms to the signal detection settings. A high level on the `rx_signaldetect` port indicates that the signal conforms to the settings and a low level indicates that the signal does not conform to the settings. The Quartus II software automatically defaults to the appropriate signal detect threshold based on the PCI Express electrical idle specifications.

*Receiver PLL and Clock Recovery Unit (CRU)*

In PCI Express (PIPE) mode, the receiver PLL in each transceiver channel is fed by a 100 MHz input reference clock. The receiver PLL in conjunction with the clock recovery unit generates two clocks: a high-speed serial recovered clock at 1250 MHz (half-rate VCO) that feeds the deserializer, and a low-speed parallel recovered clock at 250 MHz that feeds the receiver's digital logic.

You can set the clock recovery unit in either automatic lock mode or manual lock mode. In automatic lock mode, the PPM detector and the phase detector within the receiver channel automatically switches the receiver PLL between lock-to-reference and lock-to-data modes. In manual lock mode, you can control the receiver PLL switch between lock-to-reference and lock-to-data modes via the `rx_locktorefclk` and `rx_locktodata` signals.

Refer to the Receiver PLL section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details on the CRU lock modes.

The reference clock input to the receiver PLL can be derived from the following pins:

■ One of the two available dedicated reference clock input pins (REFCLK0 or REFCLK1) of the associated transceiver block
■ Inter-transceiver block lines driven by reference clock input pins of other transceiver blocks

### Deserializer

The 1:10 deserializer clocks in serial data from the receiver buffer using the high-speed recovered clock. The 10-bit deserialized data is clocked out to the word aligner using the low-speed recovered clock at 250 MHz. The deserializer assumes that the transmission bit order is LSB to MSB; for example, the LSB of a data word is received earlier in time than its MSB.

Refer to the Deserializer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about the deserializer architecture.

### Word Aligner

The word aligner clocks in the 10-bit data from the deserializer and restores the word boundary of the upstream transmitter. Besides restoring the word boundary, it also implements a synchronization state machine as specified in the PCI Express Base Specification to achieve lane synchronization.

Refer to the section "Word Aligner" on page 2–13 in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about the word aligner architecture.

In PCI Express (PIPE) mode, the word aligner consists of the following three modules:

■ Pattern detector module
■ Pattern aligner module
■ Run-length violation detector module

**Pattern Detector**

In PCI Express (PIPE) mode, the Quartus II software automatically configures 10-bit K28.5 (10'b0101111100) as the word alignment pattern. After coming out of reset (`rx_digitalreset`), when the pattern detector detects either disparities of the K28.5 control word, it asserts the `rx_patterndetect` signal for one parallel clock cycle. When the pattern aligner has aligned the incoming data to the desired word boundary, the pattern detector asserts the `rx_patterndetect` signal only if the word alignment pattern is found in the current word boundary.

**Pattern Aligner**

In PCI Express (PIPE) mode, the pattern aligner incorporates an automatic synchronization state machine. The Quartus II software automatically configures the synchronization state machine to indicate lane synchronization when the receiver receives four good /K28.5/ control code groups. Synchronization can be accomplished through the reception of four good PCI Express training sequences (TS1 or TS2) or four fast training sequences (FTS). Lane synchronization is indicated on the `rx_syncstatus` port of each channel. A high on the `rx_syncstatus` port indicates that the lane is synchronized and a low indicates that it has fallen out of synchronization.

Table 2–5 lists the synchronization state machine parameters when configured in PCI Express (PIPE) mode.

*Table 2–5. Synchronization State Machine Parameters in PCI Express (PIPE) Mode*

| | |
|---|---|
| Number of valid /K28.5/ code groups received to achieve synchronization (kcntr) | 4 |
| Number of errors received to lose synchronization (ecntr) | 17 |
| Number of continuous good code groups received to reduce the error count by 1 (gcntr) | 16 |

Figure 2–7 shows a state diagram of the PCI Express (PIPE) synchronization.

*Figure 2–7. PCI-Express (PIPE) Synchronization State Machine*
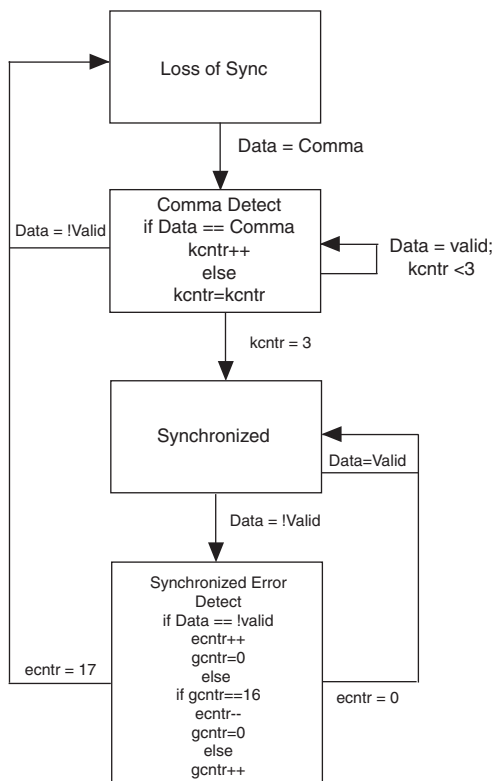


Tables 2–6 and 2–7 list the TS1 and TS2 training sequences, respectively. A PCI Express fast training sequence consists of a /K28.5/, followed by three /K28.1/ code groups.

*Table 2–6. PCI Express TS1 Ordered Set   (Part 1 of 2)*

| Symbol Number | Allowed Values | Encoded Values | Description |
|---|---|---|---|
| 0 | — | K28.5 | Comma code group for symbol alignment |
| 1 | 0–255 | D0.0–D31.7, and K23.7 | Link number with component |
| 2 | 0–31 | D0.0–D31.0, and K23.7 | Lane number within port |

*Table 2–6. PCI Express TS1 Ordered Set  (Part 2 of 2)*

| Symbol Number | Allowed Values | Encoded Values | Description |
|---|---|---|---|
| 3 | 0–255 | D0.0–D31.7 | `N_FTS`. The number of fast training ordered sets required by the receiver to obtain reliable bit and symbol lock. |
| 4 | 2 | D2.0 | Data rate identifier<br>Bit 0–Reserved, set to 0<br>Bit 1 = 1, generation 1 (2.5Gbps) data rate supported<br>Bit 2..7–Reserved, set to 0 |
| 5 | Bit 0 = 0, 1<br>Bit 1 = 0, 1<br>Bit 2 = 0, 1<br>Bit 3 = 0, 1<br>Bit 4..7 = 0 | D0.0, D1.0, D2.0, D4.0, and D8.0 | Training control<br><br>**Bit 0 – Hot reset**<br>Bit 0 = 0, de-assert<br>Bit 0 = 1, assert<br><br>**Bit 1 – Disable link**<br>Bit 1 = 0, de-assert<br>Bit 1 = 1, assert<br><br>**Bit 1 – Loopback**<br>Bit 2 = 0, de-assert<br>Bit 2 = 1, assert<br><br>**Bit 3 – Disable scrambling**<br>Bit 3 = 0, de-assert<br>Bit 3 = 1, assert<br><br>**Bit 4..7 – Reserved**<br>Bit 0 = 0, de-assert<br><br>Set to 0 |
| 6–15 | — | D10.2 | TS1 identifier |

*Table 2–7. PCI Express TS2 Ordered Set  (Part 1 of 2)*

| Symbol Number | Allowed Values | Encoded Values | Description |
|---|---|---|---|
| 0 | — | K28.5 | Comma code group for symbol alignment. |
| 1 | 0–255 | D0.0–D31.7, and K23.7 | Link number with component. |
| 2 | 0–31 | D0.0–D31.0, and K23.7 | Lane number within port. |

| Table 2–7. PCI Express TS2 Ordered Set   (Part 2 of 2) | | | |
|---|---|---|---|
| Symbol Number | Allowed Values | Encoded Values | Description |
| 3 | 0–255 | D0.0–D31.7 | N_FTS. The number of fast training ordered sets required by the receiver to obtain reliable bit and symbol lock. |
| 4 | 2 | D2.0 | Data rate identifier<br>Bit 0–Reserved, set to 0<br>Bit 1 = 1, generation 1 (2.5Gbps) data rate supported<br>Bit 2..7–Reserved, set to 0 |
| 5 | Bit 0 = 0, 1<br>Bit 1 = 0, 1<br>Bit 2 = 0, 1<br>Bit 3 = 0, 1<br>Bit 4..7 = 0 | D0.0, D1.0, D2.0, D4.0, and D8.0 | Training control<br><br>**Bit 0 – Hot reset**<br>Bit 0 = 0, de-assert<br>Bit 0 = 1, assert<br><br>**Bit 1 – Disable link**<br>Bit 1 = 0, de-assert<br>Bit 1 = 1, assert<br><br>**Bit 1 – Loopback**<br>Bit 2 = 0, de-assert<br>Bit 2 = 1, assert<br><br>**Bit 3 – Disable scrambling**<br>Bit 3 = 0, de-assert<br>Bit 3 = 1, assert<br><br>**Bit 4..7 – Reserved**<br>Bit 0 = 0, de-assert<br><br>Set to 0 |
| 6–15 | — | D5.2 | TS2 identifier |

*Rate Matcher*

In PCI Express (PIPE) mode, the rate matcher can compensate up to ± 300 parts per million (PPM) (600 PPM total) frequency difference between the upstream transmitter and the receiver. In ×1 and ×4 PCI Express (PIPE) mode, the write port of the rate matcher FIFO in each receiver channel is clocked by its low-speed parallel recovered clock. In ×1 PCI Express (PIPE) mode, the read port is clocked by the low-speed parallel clock output of the CMU local clock divider block. In ×4 PCI Express (PIPE) mode, the read port is clocked by the low-speed parallel clock output of the CMU central clock divider block.

The rate matcher logic looks for skip ordered sets (SKP), which contains a /K28.5/ comma followed by three /K28.0/ skip characters. It deletes or inserts /K28.0/ skip characters as necessary from or to the rate matcher FIFO. The rate matcher can delete only one skip character in a consecutive cluster of skip characters and can insert only one skip character per skip cluster.

Figure 2–8 shows an example of a PCI Express (PIPE) mode rate matcher deletion of two skip characters.

*Figure 2–8. PCI Express (PIPE) Mode Rate Matcher Deletion*
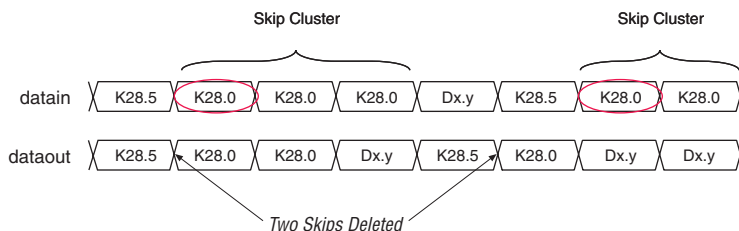


The rate matcher in PCI Express (PIPE) mode has FIFO buffer overflow and underflow protection. In the event of a FIFO buffer overflow, the rate matcher deletes any data after detecting the overflow condition to prevent FIFO pointer corruption until the rate matcher is not full. In an underflow condition, the rate matcher inserts 9'h1FE (/K30.7/) until the FIFO buffer is not empty. These measures ensure that the FIFO buffer can gracefully exit the overflow/underflow condition without requiring a FIFO reset. The rate matcher FIFO overflow and underflow condition is indicated on the pipestatus port.

### 8B/10B Decoder

In PCI Express (PIPE) mode, the 8B/10B decoder clocks in 10-bit data from the rate matcher and decodes it into 8-bit data + 1-bit control identifier. The 8-bit decoded data is fed to the byte deserializer.

For more details about the 8B/10B decoder functionality, refer to the 8B/10B Encoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

If the received 10-bit code is not a part of valid Dx.y or Kx.y code groups, the 8B/10B decoder block asserts an error flag on rx_errdetect port. The 8B/10B decoder replaces the invalid code group with /K30.7/ code

(8'hFE + 1'b1 after decoding). The error flag signal (rx_errdetect) has the same data path delay from the 8B/10B decoder to the PLD-transceiver interface as the invalid code group.

If the received 10-bit code is detected with incorrect running disparity, the 8B/10B decoder block asserts an error flag on the rx_disperr and rx_errdetect ports. The error flag signal (rx_disperr) has the same delay from the 8B/10B decoder to the PLD-transceiver interface as the received data.

**Polarity Inversion**

The 8B/10B decoder supports the PCI Express (PIPE) compatible polarity inversion feature. This polarity inversion feature inverts the bits of the incoming data stream prior to the 8B/10B decoding block to fix accidental P-N polarity inversion on the differential input buffer. You use the pipe8b10binvpolarity port to invert the inputs to the 8B/10B decoder dynamically from the PLD.

☞    You must not enable the receiver polarity inversion feature if you enable the PCI Express polarity inversion.

*Byte Deserializer*

In PCI Express (PIPE) mode, the PLD-receiver interface data is 16-bits wide and is clocked out of the receiver phase compensation FIFO at 125 MHz. The byte deserializer clocks in the 8-bit wide data from the 8B/10B decoder at 250 MHz and clocks out 16-bit wide data to the receiver phase compensation FIFO at 125 MHz. This allows clocking the PLD-transceiver interface at half the speed.

✎    For more details about byte deserializer architecture, refer to the Byte Deserializer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.*

In ×1 PCI Express (PIPE) mode, the write port of the byte deserializer is clocked by the low-speed parallel clock output from the CMU local clock divider block (tx_clkout) and the read port is clocked by divide-by-two version of this clock. In ×4 PCI Express (PIPE) mode, the write port of the byte deserializer is clocked by the low-speed parallel clock output from the CMU central clock divider block (coreclkout) and the read port is clocked by divide-by-two version of this clock.

Due to 8-bit to 16-bit byte deserialization, the byte ordering at the PLD-receiver interface might be incorrect. You implement the byte ordering logic in the PLD core to correct for this situation.

Figure 2–9 shows the block diagram of the byte serializer in PCI Express (PIPE) mode.

*Figure 2–9. Byte Deserializer in PCI Express (PIPE) Mode*
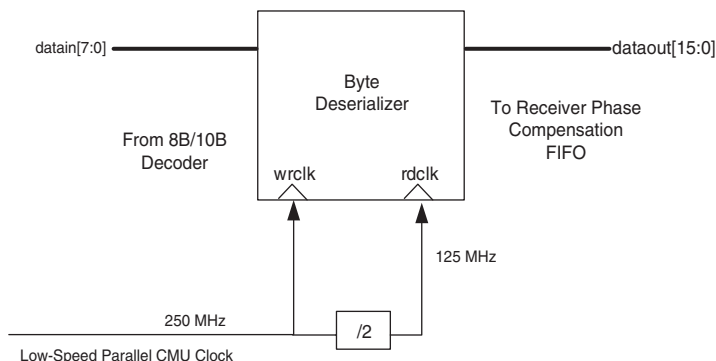


### Receiver Phase Compensation FIFO Buffer

The receiver phase compensation FIFO buffer compensates for the phase difference between the local receiver PLD clock and the receiver PCS clock.
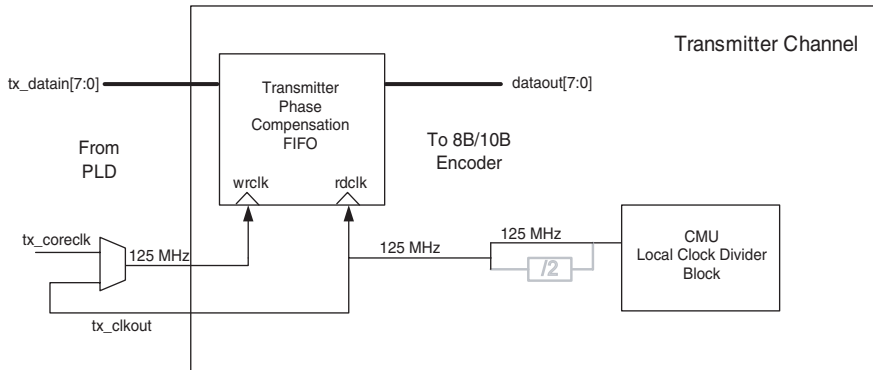
For more details about receiver phase compensation FIFO buffer architecture, refer to the Receiver Phase Compensation FIFO Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

In PCI Express (PIPE) mode, the 250-MHz clock generated by the CMU clock divider block is divided by two. The resulting 125-MHz clock is used to clock the write port of the FIFO buffer. This 125-MHz clock is also forwarded to the PLD logic array (on the `tx_clkout` port in ×1 PCI Express (PIPE) mode or the `coreclkout` port in ×4 PCI Express (PIPE) mode). If the `rx_coreclk` port is not instantiated, the clock signal on the `tx_clkout/coreclkout` port is routed back to clock the read side of the receiver phase compensation FIFO buffer. The 16-bit PLD-receiver interface, clocked at 125 MHz, results in an effective PCI Express (PIPE) data rate of 2 Gbps.

In PCI Express (PIPE) mode, the receiver phase compensation FIFO is eight words deep. The latency through the FIFO is two to three PLD-transceiver interface clock cycles.

Figure 2–10 shows the block diagram of transmitter phase compensation FIFO in PCI Express (PIPE) mode.

*Figure 2–10. Receiver Phase Compensation FIFO in PCI Express (PIPE) Mode*
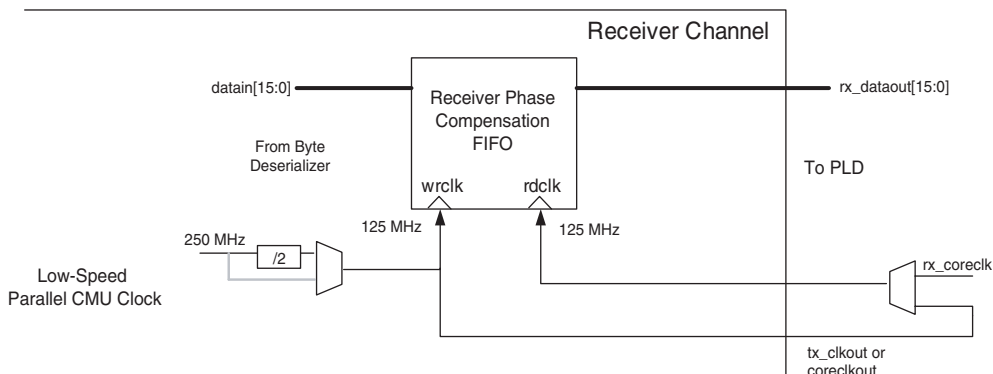


## Receiver Status

PCI Express (PIPE) specifies a receiver status indicator that reports the status of the PHY (PCS and PMA). In PCI Express (PIPE) mode, the receiver status is communicated to the PLD logic by the three-bit `pipestatus` port. This port reports the status, as shown in Table 2–8. If more than one event occurs at the same time, the signal is resolved with the higher priority status. The skip character added and removed flags (3'b001 and 3'b010) are not supported. The `pipestatus` port may be encoded to 3b'001 and 3'b010, which should be ignored. It does not indicate that a skip has been added or removed and should be considered the same as 3'b000—received data. If the upper MAC layer must know when a skip character was added or removed, Altera recommends monitoring the number of skip characters received. The transmitter should send three skip characters in a standard skip-ordered set.

*Table 2–8. pipestatus Description and Priority   (Part 1 of 2)*

| pipestatus | Description | Priority |
|---|---|---|
| 3'b000 | Received data | 6 |
| 3'b001 | One skip character added (not supported) | N/A |
| 3'b010 | One skip character removed (not supported) | N/A |
| 3'b011 | Receiver detected | 1 |
| 3'b100 | 8B/10B decoder error | 2 |
| 3'b101 | Elastic buffer overflow | 3 |

| Table 2–8. pipestatus Description and Priority   (Part 2 of 2) | | |
|---|---|---|
| **pipestatus** | **Description** | **Priority** |
| 3'b110 | Elastic buffer underflow | 4 |
| 3'b111 | Received disparity error | 5 |

## Power State Management

The four supported power states in Arria GX when configured in PIPE mode are:

- PO — normal power state
- POs — low recovery time
- P1 — lower than PO
- P2 — lowest power state

There are four supported power states in Arria GX transceivers when configured in PIPE mode: P0, P0s, P1, and P2. P0 is the normal power state. P0s is a low recovery time power state that is lower than P0. P1 is a lower power state than P0s and has higher latency to come out of this state. P2 is the lowest power state.

The powerdn port transitions the transceiver into different power states. The encoded value is shown in Table 2–9. The pipephydonestatus signal reacts to the powerdn request and pulses high for one parallel clock cycle.

There are specific functions that are performed at each of the power states. The power-down states are for PCI Express (PIPE) emulation. The transceiver does not go into actual power saving mode, with the exception of the transmitter buffer for Electrical Idle.

Table 2–9 shows each power state and its function.

| Table 2–9. Power State Functions and Descriptions | | | |
|---|---|---|---|
| **Power State** | **powerdn** | **Function** | **Description** |
| P0 | 2'b00 | Transmits normal data, transmits Electrical Idle, or enters into loopback mode. | Normal operation mode |
| P0s | 2'b01 | Only transmits Electrical Idle. | Low recovery time power saving state |
| P1 | 2'b10 | Transmitter buffer is powered down and can do a receiver detect while in this state. | High recovery time power saving state |
| P2 | 2'b11 | Transmits Electrical Idle or a beacon to wake up the downstream receiver. | Lowest power saving state |

The two signals associated with the power states are:
`tx_detectrxloopback` and `tx_forceelecidle`. The
`tx_detectrxloopback` signal controls whether the channel goes into
loopback when the power state is in P0 or receiver detect when in P1 state.
This signal does not have any affect in any other power states. The
`tx_forceelecidle` signal governs when the transmitter goes into an
electrical idle state. The `tx_forceelecidle` signal is asserted in P0s
and P1 states and de-asserted in P0 state. In P2 state, under normal
conditions, the `tx_forceelecidle` signal is asserted and then
de-asserted when the beacon signal must be sent out, signifying the intent
to exit the P2 power-down state.

Table 2–10 shows the behavior of the `tx_detectrxloopback` and
`tx_forceelecidle` signals in the power states.

| *Table 2–10. Power States and Functions Allowed in Each Power State* | | |
|---|---|---|
| **Power State** | **tx_detectrxloopback** | **tx_forceelecidle** |
| P0 | 0: normal mode<br>1: data path in loopback mode | 0: Must be de-asserted.<br>1: Illegal mode |
| P0s | Don't care | 0: Illegal mode<br>1: Must be asserted in this state |
| P1 | 0: Electrical Idle<br>1: receiver detect | 0: Illegal mode<br>1: Must be asserted in this state |
| P2 | Don't care | De-asserted in this state for sending beacon.<br>Otherwise asserted. |

## NFTS Fast Recovery IP (NFRI)

The PCI Express fast training sequences (FTS) are used for bit and byte
synchronization to transition from P0s state to P0 state. The PCI Express
standard specifies the required time period for this transition to be
between 16 ns and 4 μs. The default PCI Express (PIPE) settings do not
meet this requirement. You must enable the NFTS fast recovery IP (NFRI)
for the receiver to transition from P0s to P0 within 4 μs by selecting the
**Enable fast recovery mode** option in the MegaWizard Plug-In Manager.

### PCI Express (PIPE) Mode Default Settings

In the PCI Express (PIPE) mode default settings (without NFRI enabled),
the receiver PLL is in automatic lock mode. The PLL moves from
lock-to-reference mode to lock-to-data mode based on the
`rx_freqlocked` being asserted. For the `rx_freqlocked` signal to be
asserted, the CRU clock should be within the PPM threshold settings of
the receiver PLL reference clock. The PPM detector checks the PPM

threshold settings by comparing the CRU PLL clock output with the reference clock for approximately 32768 clock cycles. For a 250 MHz PLD interface clock frequency, this comparison time period exceeds 4 μs, which violates the PCI Express specification.

The NFRI, if enabled, controls the `rx_locktorefclk` and `rx_locktodata` signals to meet the 4 μs transition time from P0s to P0 power state.

☞ If you select the `rx_locktorefclk` and `rx_locktodata` signals in the MegaWizard Plug-In Manager (CRU Manual Lock mode), the **Enable fast recovery mode** option cannot be selected.

When you select the **Enable fast recovery mode** option, you must consider the following:

■ NFRI is created in the PLD side for each PCI Express (PIPE) channel
■ NFRI is a soft IP, so it consumes logic resources
■ This block is self-contained, so no input/output ports are available to access the soft IP

### Low-Latency (Synchronous) PCI Express (PIPE) Mode

The Arria GX receiver data path employs a rate match FIFO in PCI Express (PIPE) mode to compensate up to ±300 PPM difference between the upstream transmitter and the local receiver reference clock. The low-latency (synchronous) PCI Express (PIPE) mode allows bypassing the rate match FIFO in synchronous systems that derive the transmitter and receiver reference clocks from the same source. You can bypass the rate match FIFO by not selecting the **Enable Rate Match FIFO** option in the ALT2GXB MegaWizard Plug-In Manager.

The rate match FIFO can be bypassed in both ×1 and ×4 PCI Express (PIPE) modes. In normal PCI Express (PIPE) mode, the receiver blocks following the rate match FIFO are clocked by `tx_clkout` (×1 mode) or `coreclkout` (×4 mode) of the local port. In low-latency (synchronous) PCI Express (PIPE) mode, because the rate match FIFO is bypassed, these receiver blocks are clocked by the recovered clocks of the respective channels.

Except for the rate match FIFO being bypassed and the resulting changes in transceiver internal clocking, the low-latency (synchronous) PCI Express (PIPE) mode shares the same data path and state machines as the normal PCI Express (PIPE) mode. However, some features supported in normal PCI Express (PIPE) mode are not supported in low-latency (synchronous) PCI Express (PIPE) mode.

*PCI Express (PIPE) Reverse Parallel Loopback*

In normal PCI Express (PIPE) mode, if the transceiver is in P0 power state, a high value on the `tx_rxdetectloop` signal forces a reverse parallel loopback, as discussed in PCI Express (PIPE) Reverse Parallel Loopback section. Parallel data at the output of the receiver rate match FIFO gets looped back to the input of the transmitter serializer.

In low-latency (synchronous) PCI Express (PIPE) mode, since the rate match FIFO is bypassed, this feature is not supported. A high value on the `tx_rxdetectloop` signal when the transceiver is in P1 power state will not force it to perform reverse parallel loopback.

*Link Width Negotiation*

In normal ×4 PCI Express (PIPE) configuration, the receiver phase compensation FIFO control signals (write/read enable, and so forth) are shared among all lanes within the link. As a result, all lanes are truly bonded and the lane-lane skew meets the PCI Express specification.

In low-latency (synchronous) PCI Express (PIPE) configuration, the receiver phase compensation FIFO of individual lanes do not share control signals. The write port of the receiver phase compensation FIFO of each lane is clocked by its recovered clock. As a result, the lanes within a link are not bonded. You should perform external lane de-skewing to ensure proper link width negotiation.

*Receiver Status*

Because the rate match FIFO is bypassed in low-latency (synchronous) PCI Express (PIPE) mode, status signal combinations related to the rate match FIFO on the `pipestatus[2:0]` port become irrelevant and must not be interpreted (Table 2–11).

| Table 2–11. pipestatus Signal   (Part 1 of 2) | | |
|---|---|---|
| **pipestatus[2:0]** | **Normal PIPE** | **Synchronous PIPE** |
| 000 | Received Data OK | Received Data OK |
| 001 | Not supported | Not supported |
| 010 | Not supported | Not supported |
| 011 | Receiver Detected | Receiver Detected |
| 100 | 8B/10B Decoder Error | 8B/10B Decoder Error |
| 101 | Elastic Buffer Overflow | Not supported |

*Table 2–11. pipestatus Signal   (Part 2 of 2)*

| pipestatus[2:0] | Normal PIPE | Synchronous PIPE |
|:---:|:---:|:---:|
| 110 | Elastic Buffer Underflow | Not supported |
| 111 | Received Disparity Error | Received Disparity Error |

## Gigabit Ethernet (GIGE) mode

IEEE 802.3 defines the 1000 Base-X PHY as an intermediate, or transition, layer that interfaces various physical media with the media access control (MAC) in a gigabit ethernet system. It shields the MAC layer from the specific nature of the underlying medium. The 1000 Base-X PHY is divided into three sub-layers:

■ Physical coding sublayer (PCS)
■ Physical media attachment (PMA)
■ Physical medium dependent (PMD)

The PCS sublayer interfaces to the MAC through the gigabit medium independent interface (GMII). The 1000 Base-X PHY defines a physical interface data rate of 1 Gbps.

Figure 2–11 shows the 1000 Base-X PHY position in a Gigabit Ethernet OSI reference model.

*Figure 2–11. GIGE OSI Reference Model*
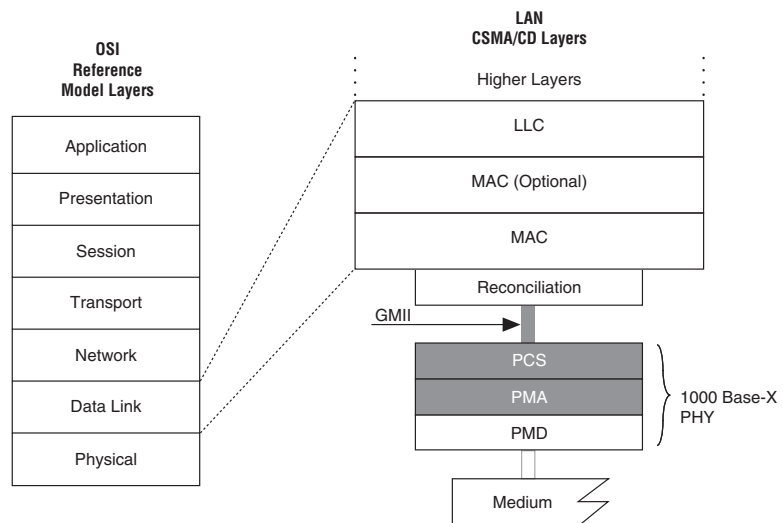
When Arria GX transceivers are configured in GIGE functional mode, they provide many of the PCS and PMA functions defined in the IEEE 802.3 specification; for example:

- 8B/10B encoding/decoding
- Synchronization
- Upstream transmitter and local receiver clock frequency compensation (rate matching)
- Clock recovery from the encoded data forwarded by the receiver PMD
- Serialization/deserialization

☞ Arria GX transceivers do not have built-in support for other PCS functions, such as auto-negotiation, collision-detect, and carrier-sense. If required, you must implement these functions in PLD logic array or external circuits.

For more information about additional features available in the Arria GX transceiver, refer to the GIGE-Enhanced sub-protocol in the *Arria GX Megafunction User Guide*.

This section is organized into transmitter and receiver data path modules when configured for GIGE mode. The description for each module only covers details specific to GIGE functional mode support. This docuent assumes that you are familiar with the IEEE 802.3 Ethernet specification.

For a general description of each module, refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

## GIGE Mode Transmitter Architecture

This section lists sub-blocks within the transmitter channel configured in GIGE mode (Figure 2–12). The sub-blocks are described in order from the PLD-Transceiver parallel interface to the serial transmitter buffer.

*Figure 2–12. GIGE Transmitter Architecture*
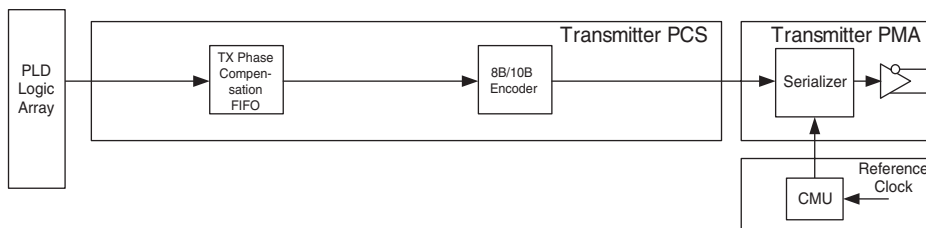
*Clock Multiplier Unit (CMU)*

The clock multiplier unit takes in a reference clock and synthesizes the clocks that are used to clock the transmitter digital logic (PCS), the serializer, and the PLD-transceiver interface.

For more details about CMU architecture, refer to the Clock Multiplier Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

In GIGE mode, the CMU block consists of:

■ Transmitter PLL that generates high-speed serial clock for the serializer
■ Local clock divider block that generates low-speed parallel clock for transmitter digital logic and PLD-transceiver interface

**Input Reference Clock**

You can select either a 62.5 MHz or 125 MHz input reference clock frequency while configuring the transceiver in GIGE mode using the Quartus II MegaWizard Plug-In Manager.

The reference clock input to the transmitter PLL can be derived from one of three components:

■ One of the two available dedicated reference clock input pins (REFCLK0 or REFCLK1) of the associated transceiver block
■ PLD global clock network (must be driven directly from an input clock pin and cannot be driven by user logic or enhanced PLL)
■ Inter-transceiver block lines driven by reference clock input pins of other transceiver blocks

☞ Altera recommends using the dedicated reference clock input pins (REFCLK0 or REFCLK1) to provide reference clock for the transmitter PLL.

The reference clock divide-by-two pre-divider is bypassed in GIGE mode.

Table 2–12 specifies the input reference clock options available in GIGE mode.

| Table 2–12. GIGE Mode Input Reference Clock Specification | | | |
|---|---|---|---|
| Frequency | I/O Standard | Coupling | Termination |
| 62.5 MHz | 1.2 V PCML, 1.5 V PCML, 3.3 V PCML, Differential LVPECL, LVDS | AC | On-chip |
| 125 MHz | | | |

**Clock Synthesis**

In GIGE mode, the input reference clock of 125 MHz (or 62.5 MHz) is fed to the transmitter PLL. Because the transmitter PLL implements a half-rate VCO, it multiplies the 125 MHz (or 62.5 MHz) input clock by 5 (or 10) to generate a 625 MHz high-speed serial clock. This high-speed serial clock feeds the local clock divider block in each GIGE channel instantiated within the transceiver block.

The local clock divider in each channel of the transceiver block divides the 625 MHz clock from the transmitter PLL by 5 to generate a 125 MHz parallel clock. This low-speed parallel clock output from the local clock divider block is used to clock the transmitter digital logic (PCS) of the associated channel. The local clock divider block also forwards the high-speed serial clock from the transmitter PLL to the serializer within its associated channel.

☞ The Quartus II software automatically selects the appropriate transmitter PLL bandwidth suited for GIGE data rate.

Figure 2–13 shows the CMU implemented in GIGE mode.

*Figure 2–13. GIGE Mode CMU*
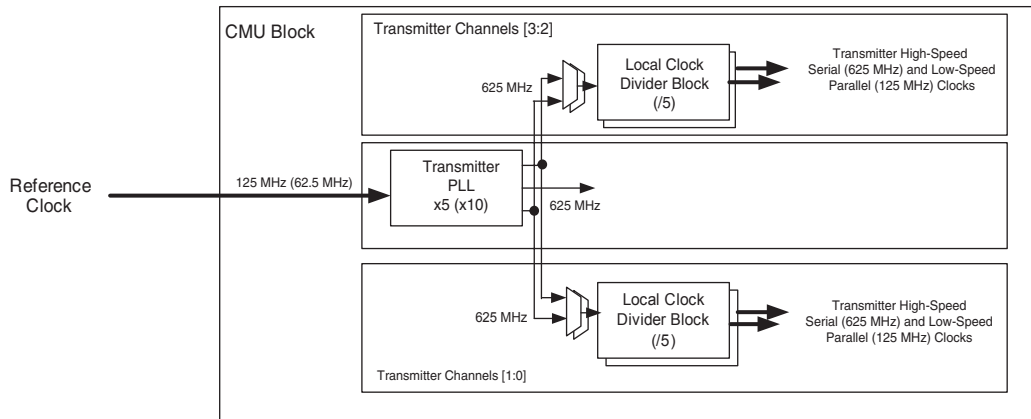


### Transmitter Phase Compensation FIFO Buffer

The transmitter phase compensation FIFO buffer compensates for the phase difference between the PLD clock that clocks in parallel data into the transmitter and the PCS clock that clocks the rest of the transmitter digital logic.

For more details about the transmitter phase compensation FIFO buffer architecture, refer to the Transmitter Phase Compensation FIFO Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

In GIGE mode, the 125 MHz clock generated by the CMU local clock divider is used to clock the read port of the FIFO buffer. This 125 MHz clock is also forwarded to the PLD logic array (on the `tx_clkout` port). If the `tx_coreclk` port is not instantiated, the clock signal on the `tx_clkout` port is automatically routed back to clock the write side of the transmitter phase compensation FIFO buffer. The 8-bit PLD-transceiver interface clocked at 125 MHz results into an effective GIGE data rate of 1 Gbps.

In GIGE mode, the transmitter phase compensation FIFO is four words deep. The latency through the FIFO is two to three PLD-transceiver interface clock cycles.

Figure 2–14 shows the block diagram of transmitter phase compensation FIFO in GIGE mode.

*Figure 2–14. Transmitter Phase Compensation FIFO in GIGE Mode*



### 8B/10B Encoder

In GIGE mode, the 8B/10B encoder clocks in 8-bit data and 1-bit control identifier from the transmitter phase compensation FIFO and generates a 10-bit encoded data. The 10-bit encoded data is fed to the serializer.

For more details about the 8B/10B encoder functionality, refer to the 8B/10B Encoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

**GIGE Protocol — Ordered Sets and Special Code Groups**
Table 2–13 lists ordered sets and special code groups used in the GIGE functional mode.

*Table 2–13. GIGE Ordered Sets   (Part 1 of 2)*

| Code Group | Ordered Set | Number of Code Groups | Encoding |
|---|---|---|---|
| /C/ | Configuration | — | Alternating /C1/ and /C2/ |
| /C1/ | Configuration 1 | 4 | /K28.5/D21.5/Config_Reg *(1)* |
| /C2/ | Configuration 2 | 4 | /K28.5/D2.2/Config_Reg *(1)* |

*Table 2–13. GIGE Ordered Sets  (Part 2 of 2)*

| Code Group | Ordered Set | Number of Code Groups | Encoding |
|---|---|---|---|
| /I/ | IDLE | — | Correcting /I1/, Preserving /I2/ |
| /I1/ | IDLE 1 | 2 | /K28.5/D5.6 |
| /I2/ | IDLE 2 | 2 | /K28.5/D16.2 |
| | Encapsulation | — | — |
| /R/ | Carrier_Extend | 1 | /K23.7/ |
| /S/ | Start_of_Packet | 1 | /K27.7/ |
| /T/ | End_of_Packet | 1 | /K29.7/ |
| /V/ | Error_Propagation | 1 | /K30.7/ |

*Note to Table 2–13:*

(1)    Two data code groups representing the Config_Reg value.

**Idle Ordered-Set Generation**

IEEE 802.3 requires the GIGE PHY to transmit idle ordered sets (/I/) continuously and repetitively whenever the GMII is idle. This ensures that the receiver maintains bit and word synchronization whenever there is no active data to be transmitted.

In GIGE functional mode, any /Dx.y/ following a /K28.5/ comma is replaced by the transmitter with either a /D5.6/ (/I1/ ordered set) or a /D16.2/ (/I2/ ordered set), depending on the current running disparity. The exception is when the data following the /K28.5/ is /D21.5/ (/C1/ ordered set) or /D2.2/ (/C2/) ordered set. If the running disparity before the /K28.5/ is positive, a /I1/ ordered set is generated. If the running disparity is negative, a /I2/ ordered set is generated. The disparity at the end of a /I1/ is the opposite of that at the beginning of the /I1/. The disparity at the end of a /I2/ is the same as the beginning running disparity (right before the idle code group). This ensures a negative running disparity at the end of an idle ordered set. A /Kx.y/ following a /K28.5/ is not replaced.

Figure 2–15 shows the automatic idle ordered set generation. Note that /D14.3/, /D24.0/, and /D15.8/ are replaced by /D5.6/ or /D16.2/ (for /I1/, /I2/ ordered sets). /D21.5/ (part of the /C1/ order set) is not replaced.

*Figure 2–15. Idle Ordered Set Generation in GIGE Mode*
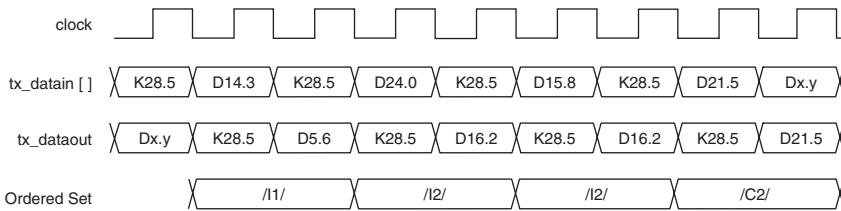


### Reset Condition

After power-up or reset, the GIGE transmitter outputs three /K28.5/ commas before user data can be sent. This affects the synchronization ordered set transmission.

After reset (`tx_digitalreset`), the 8B/10B encoder automatically sends three /K28.5/ commas. Depending on when you start outputting the synchronization sequence, there could be an even or odd number of /Dx.y/ sent as the transmitter before the synchronization sequence. The last of the three automatically sent /K28.5/and the first user-sent /Dx.y/ are treated as one idle ordered set. This can be a problem if there are an even number of /Dx.y/ transmitted before the start of the synchronization sequence.

Figure 2–16 shows an example of even numbers of /Dx.y/ between the last automatically sent /K28.5/ and the first user-sent /K28.5/. The first user-sent ordered set is ignored, so three additional ordered sets are required for proper synchronization. Figure 2–16 shows one don't care data between the `tx_digitalreset` signal going low and the first of three automatic K28.5, but there could be more.

*Figure 2–16. GIGE Synchronization Ordered Set Considerations After Reset*
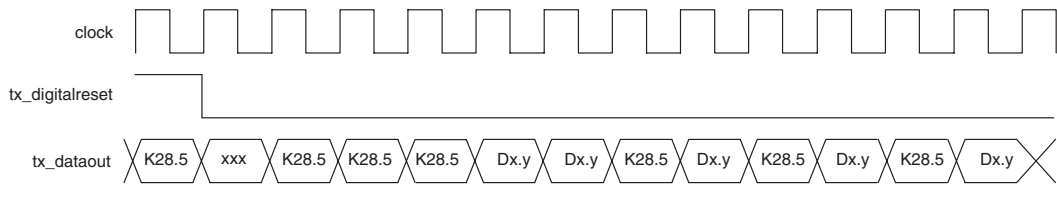
*Serializer*

In GIGE mode, the 10-bit encoded data from the 8B/10B encoder is clocked into the 10:1 serializer with the low-speed parallel clock at 125 MHz. The 10-bit data is clocked out of the serializer LSB to MSB at the high-speed effective serial clock rate at 1250 MHz. The serial data output of the serializer is fed into the transmitter output buffer.

For more details about the serializer architecture, refer to the Serializer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

*Transmitter Buffer*

Table 2–14 shows the transmitter buffer settings when configured in GIGE mode.

*Table 2–14. Transmitter Buffer Settings in GIGE Mode*

| Settings | Value |
|---|---|
| I/O Standard | 1.5-V PCML *(1)* |
| Programmable Differential Output Voltage (V$_{OD}$) | 400 — 1200 mV |
| Common Mode Voltage (V$_{CM}$) | 600 mV, 700 mV *(1)* |
| Differential Termination | 100 Ω *(2)* |
| Programmable Transmitter Pre-Emphasis | Enabled *(3)* |
| V$_{CCH}$ (Transmitter Buffer Power) | 1.5 V |

*Notes to Table 2–14:*
(1) The common mode voltage (V$_{CM}$) setting is selectable in the MegaWizard Plug-In Manager.
(2) The I/O standard and differential termination settings are defaulted to 1.5-V PCML and 100 Ω, respectively. If you select any other setting for I/O standard or differential termination in the Assignment Editor, the Quartus II compiler will issue an error message.
(3) The transmitter buffer has five programmable first post-tap pre-emphasis settings.

## GIGE Mode Receiver Architecture

This section lists sub-blocks within the receiver channel configured in GIGE mode (Figure 2–17). The sub-blocks are described in order from the serial receiver input buffer to the receiver phase compensation FIFO buffer at the transceiver-PLD interface.

*Figure 2–17. GIGE Mode Receiver Architecture*
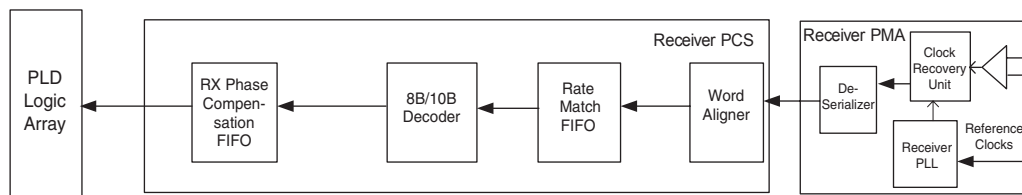


## Receiver Buffer

Table 2–15 shows the receiver buffer settings when configured in GIGE mode.

| Table 2–15. Receiver Buffer Settings in GIGE Mode | |
|---|---|
| **Settings** | **Value** |
| I/O Standard | 1.2-V PCML, 1.5-V PCML, 3.3-V PCML, Differential LVPECL, LVDS |
| Input Common Mode Voltage (Rx $V_{CM}$) | 850 mV, 1200 mV *(1)* |
| Differential Termination | 100 Ω *(2)* |
| Programmable Equalization | Enabled *(3)* |
| Coupling | AC |

Notes to Table 2–15:
(1) The common mode voltage (Rx $V_{CM}$) is selectable in the MegaWizard Plug-In Manager.
(2) The differential termination setting is defaulted to 100 Ω. If you select any other setting for differential termination in the Assignment Editor, the Quartus II compiler will issue an error message.
(3) The receiver buffer has five programmable equalization settings.

**Receiver PLL and Clock Recovery Unit**
In GIGE mode, the receiver PLL in each transceiver channel is fed by a 125 MHz or a 62.5 MHz input reference clock. The receiver PLL in conjunction with the CRU generates two clocks: a high-speed serial recovered clock at 625 MHz (half-rate PLL) that feeds the deserializer and a low-speed parallel recovered clock at 125 MHz that feeds the receiver's digital logic.

You can set the clock recovery unit in either automatic lock mode or manual lock mode. In automatic lock mode, the PPM detector and the phase detector within the receiver channel automatically switches the

receiver PLL between lock-to-reference and lock-to-data modes. In manual lock mode, you can control the receiver PLL switch between lock-to-reference and lock-to-data modes via the `rx_locktorefclk` and `rx_locktodata` signals.

For more details about the CRU lock modes, refer to the Receiver PLL and Clock Recovery Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

The reference clock input to the receiver PLL can be derived from:

■ One of the two available dedicated reference clock input pins (REFCLK0 or REFCLK1) of the associated transceiver block
■ PLD global clock network (must be driven directly from an input clock pin and cannot be driven by user logic or enhanced PLL)
■ Inter-transceiver block lines driven by reference clock input pins of other transceiver blocks

Table 2–16 specifies the input reference clock options available in GIGE mode.

*Table 2–16. GIGE Mode Input Reference Clock Specification*

| Frequency | I/O Standard | Coupling | Termination |
|-----------|--------------|----------|-------------|
| 125 MHz | 1.2 V PCML, 1.5 V PCML, 3.3 V PCML, Differential LVPECL, LVDS | AC | On-chip |
| 62.5 MHz | | | |

*Deserializer*

The 1:10 deserializer clocks in serial data from the receiver buffer using the high-speed recovered clock. The 10-bit de-serialized data is clocked out to the word aligner using the low-speed recovered clock at 125 MHz. The deserializer assumes that the transmission bit order is LSB to MSB; for example, the LSB of a data word is received earlier in time than its MSB.

For more details about the deserializer architecture, refer to the Deserializer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

*Word Aligner*

The word aligner clocks in the 10-bit data from the deserializer and restores the word boundary of the upstream transmitter. Besides restoring the word boundary, it also implements a synchronization state machine as specified in the IEEE 802.3 specification to achieve receiver synchronization.

In GIGE mode, the word aligner is comprised of the following three modules:

■ Pattern detector module
■ Pattern aligner module
■ Run-length violation detector module

**Pattern Detector**
In GIGE mode, the Quartus II software automatically configures 10-bit K28.5 (10'b0101111100) as the word alignment pattern. After coming out of reset (`rx_digitalreset`), when the pattern detector detects either disparities of the K28.5 control word, it asserts the `rx_patterndetect` signal for one parallel clock cycle. When the pattern aligner has aligned the incoming data to the desired word boundary, the pattern detector asserts the `rx_patterndetect` signal only if the word alignment pattern is found in the current word boundary.

**Pattern Aligner**
In GIGE mode, the pattern aligner incorporates an automatic synchronization state machine. The Quartus II software automatically configures the synchronization state machine to indicate synchronization when the receiver receives three consecutive synchronization ordered sets. An ordered set defined for synchronization is a /K28.5/ code group followed by an odd number of valid /Dx.y/ code groups. The fastest way for the receiver to achieve synchronization is to receive three continuous {/K28.5/, /Dx.y/} ordered sets.

Receiver synchronization is indicated on the `rx_syncstatus` port of each channel. A high on the `rx_syncstatus` port indicates that the lane is synchronized and a low indicates that it has fallen out of synchronization. The receiver loses synchronization when it detects four invalid code groups separated by less than three valid code groups or when it is reset.

Table 2–17 lists the synchronization state machine parameters when configured in GIGE mode.

| *Table 2–17. Synchronization State Machine Parameters in GIGE Mode* | |
|---|---|
| Number of valid {/K28.5/, /Dx,y/} ordered-sets received to achieve synchronization | 3 |
| Number of errors received to lose synchronization | 4 |
| Number of continuous good code groups received to reduce the error count by 1 | 4 |

Figure 2–18 shows the synchronization state machine implemented in GIGE mode.

*Figure 2–18. GIGE Synchronization State Machine*
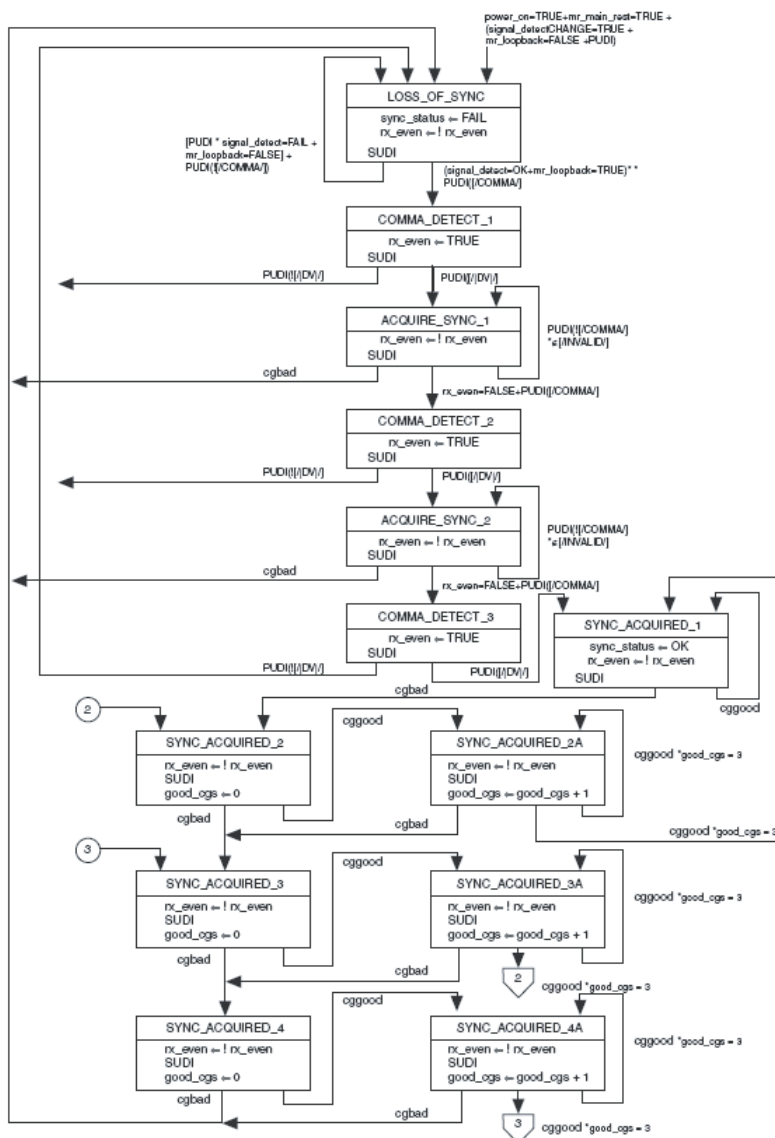
The word aligner block asserts an error flag on the `rx_disperr` and `rx_errdetect` ports if the received 10-bit code is detected with incorrect running disparity. The error flag signal (`rx_disperr`) has the same delay from the word aligner to the PLD-transceiver interface as the received data.

### Rate Matcher

In GIGE mode, the rate matcher can compensate up to ±100 PPM (200 PPM total) frequency difference between the upstream transmitter and the receiver. The write port of the rate matcher FIFO in each receiver channel is clocked by its low-speed parallel recovered clock. The read port is clocked by the low-speed parallel clock output of the CMU local clock divider block.

The rate matcher logic inserts or deletes /I2/ idle ordered-sets to/from the rate matcher FIFO during the inter-frame or inter-packet gap (IFG or IPG). /I2/ is selected as the rate matching ordered-set since it maintains the running disparity unlike /I1/ that alters the running disparity. Since the /I2/ ordered-set contains two 10-bit code groups (/K28.5/, /D16.2/), twenty bits are inserted or deleted at a time for rate matching.

☞ The rate matcher logic has the capability to insert or delete /C1/ or /C2/ configuration ordered sets when GIGE-Enhanced mode is chosen as the sub-protocol in the MegaWizard Plug-In Manager.

🔖 Refer to the *Arria GX ALT2GXB Megafunction User Guide* for details on GIGE-Enhanced mode.

Figure 2–19 shows an example of /I2/ deletion and Figure 2–20 shows an example of /I2/ insertion in a GIGE mode rate matcher.

*Figure 2–19. GIGE Rate Matcher /I2/ Deletion*
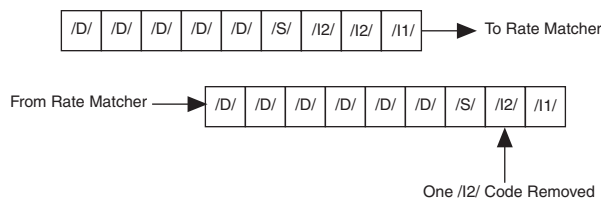
*Figure 2–20. GIGE Rate Matcher /I2/ Insertion*
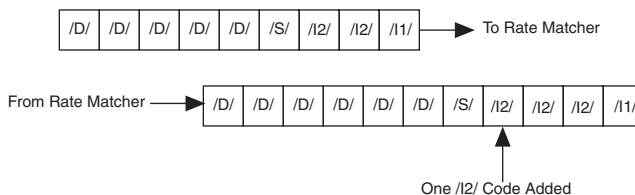


If the frequency PPM difference between the upstream transmitter and the local receiver is high or if the packet size is too large, the rate matcher FIFO buffer can face an overflow or underflow situation.

### 8B/10B Decoder

In GIGE mode, the 8B/10B decoder clocks in 10-bit data from the rate matcher and decodes it into 8-bit data + 1-bit control identifier. The 10-bit decoded data is fed to the receiver phase compensation FIFO buffer.

For more details about the 8B/10B decoder functionality, refer to the 8B/10B Encoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

If the received 10-bit code group is not a part of valid Dx.y or Kx.y code groups, the 8B/10B decoder block asserts an error flag on the `rx_errdetect` port. The error flag signal (`rx_errdetect`) has the same data path delay from the 8B/10B decoder to the PLD-transceiver interface as the invalid code group.

### Receiver Phase Compensation FIFO

The receiver phase compensation FIFO buffer compensates for the phase difference between the local receiver PLD clock and the receiver PCS clock.

For more details about the receiver phase compensation FIFO buffer architecture, refer to the Receiver Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

In GIGE mode, the 125 MHz clock generated by the CMU local clock divider block clocks the write port of the FIFO buffer. This 125 MHz clock is also forwarded to the PLD logic array (on the corresponding `tx_clkout` port). If the `rx_coreclk` port is not instantiated, the clock signal on the `tx_clkout` port is automatically routed back to clock the

read side of the receiver phase compensation FIFO buffer. The 8-bit PLD-receiver interface clocked at 125 MHz results in an effective GIGE data rate of 1 Gbps.

In GIGE mode, the receiver phase compensation FIFO is four words deep. The latency through the FIFO is one to two PLD-transceiver interface clock cycles.

Figure 2–21 shows the block diagram of receiver phase compensation FIFO in GIGE mode.

*Figure 2–21. Receiver Phase Compensation FIFO in GIGE Mode*
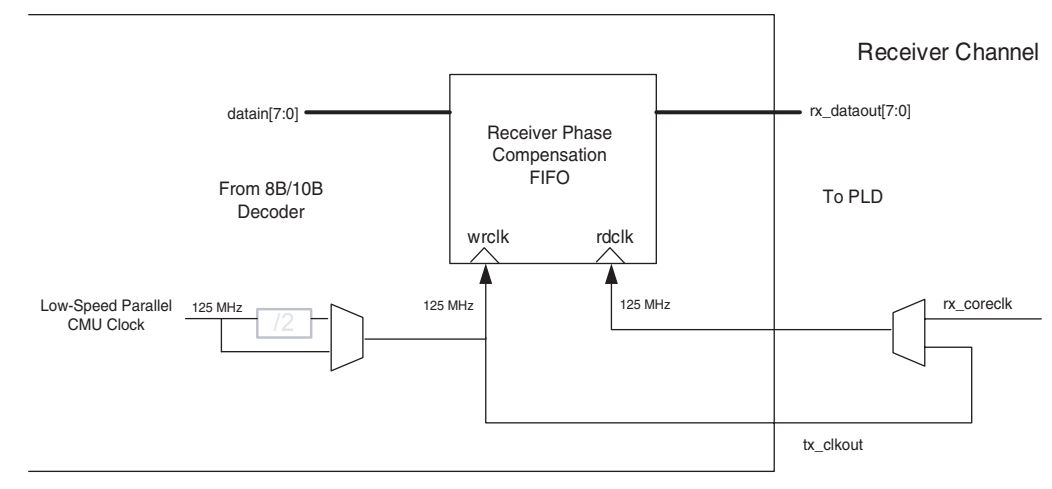


## UNH-IOL Gigabit Ethernet Compliance

For UNH-IOL compliance in GIGE mode, the following architectural features are available when GIGE-Enhanced sub-protocol is chosen in the Megawizard Plug-In Manager.

■ 7-bit word alignment using the synchronization state machine.
■ Insertion and deletion of /C1/ and /C2/ configuration ordered sets by the rate matcher during the Auto-negotiation phase.

Refer to the *Arria GX ALT2GXB Megafunction User Guide* for details regarding additional ports generated for GIGE-Enhanced mode.

# Serial RapidIO Mode

The RapidIO standard is a high-performance, packet-switched interconnect technology designed to pass data and control information between microprocessors, digital signal, communications, and network processors, system memories, and peripheral devices. Serial RapidIO physical layer specification defines three line rates at 1.25 Gbps, 2.5 Gbps, and 3.125 Gbps. It also supports two link widths — single-lane (×1) and bonded four-lane (×4) at each line rate.

Arria GX transceivers support both single-lane (×1) and four-lane (×4) Serial RapidIO link widths at 1.25 Gbps and 2.5 Gbps and single-lane link widths at 3.125 Gbps. In ×4 Serial RapidIO mode, the four transceiver channels are not bonded and are clocked independently, as four individual channels.

When configured in Serial RapidIO functional mode, Arria GX transceivers provide the following PCS and PMA functions:

- 8B/10B encoding/decoding
- Word alignment
- Lane Synchronization State Machine
- Clock recovery from the encoded data
- Serialization/deserialization

☞ Arria GX transceivers do not have built-in support for other PCS functions, such as clock frequency compensation between upstream transmitter clock and local receiver clock (rate matcher), idle sequence generation, and lane alignment in ×4 mode. Depending on your system requirements, you must implement these functions in the logic array or external circuits.

This section is organized into transmitter and receiver data path modules when configured for Serial RapidIO mode. The description for each module only covers details specific to Serial RapidIO functional mode support. This document assumes that you are familiar with the RapidIO Interconnect Specification v1.3.

For a general description of each module, refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

## Serial RapidIO Mode Transmitter Architecture

This section lists sub-blocks within the transmitter channel configured in Serial RapidIO mode (Figure 2–22). The sub-blocks are described from the PLD-Transceiver parallel interface to the serial transmitter buffer.

*Figure 2–22. Serial RapidIO Transmitter Architecture*
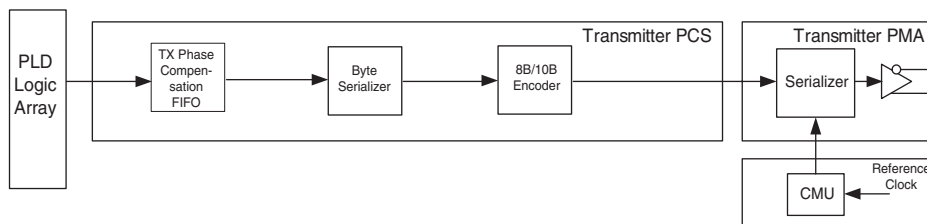


*Clock Multiplier Unit (CMU)*

The clock multiplier unit takes in a reference clock and synthesizes the clocks that are used to clock the transmitter digital logic (PCS), the serializer, and the PLD-transceiver interface.

For more details about CMU architecture, refer to the Clock Multiplier Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

In Serial RapidIO mode, the CMU block consists of:

■ Transmitter PLL that generates high-speed serial clock for the serializer
■ Local clock divider block that generates low-speed parallel clock for transmitter digital logic and PLD-transceiver interface

**Input Reference Clock**
Table 2–18 lists the input reference clock frequencies allowed in Serial RapidIO mode.

The reference clock input to the transmitter PLL can be derived from:

■ One of the two available dedicated reference clock input pins (REFCLK0 or REFCLK1) of the associated transceiver block
■ PLD global clock network (must be driven directly from an input clock pin and cannot be driven by user logic or enhanced PLL)
■ Inter-transceiver block lines driven by reference clock input pins of other transceiver blocks

☞ Altera recommends using the dedicated reference clock input
pins (REFCLK0 or REFCLK1) to provide reference clock for the
transmitter PLL.

*Table 2–18. Serial RapidIO Mode Input Reference Clock Specifications*

| Data Rate (Gbps) | Reference Clock Frequency (MHz) | I/O Standard | Coupling | Termination |
|---|---|---|---|---|
| 1.25 | 62.5, 78.125, 125, 156.25, 250, 312.5 | 1.2V PCML, 1.5V PCML, 3.3V PCML, Differential LVPECL, LVDS | AC | On-chip |
| 2.5 | 50, 62.5, 78.125, 100, 125, 156.25, 250, 312.5, 500 | | | |
| 3.125 | 62.5, 78.125, 97.6563, 125, 156.25, 195.3125, 312.5, 390.625 | | | |

**Clock Synthesis**

In Serial RapidIO mode, the input reference clock is fed to the transmitter
PLL. Because the transmitter PLL implements a half-rate VCO, it
multiplies the input reference clock to generate a 625-MHz (1.25-Gbps
Serial RapidIO) or 1250-MHz (2.5 Gbps Serial RapidIO) or 1562.5-MHz
(3.125-Gbps Serial RapidIO) high-speed serial clock. This high-speed
serial clock feeds the local clock divider block in each Serial RapidIO
channel instantiated within the transceiver block. Table 2–19 lists the
transmitter PLL multiplication factors that the Quartus II software
automatically selects, depending on the Serial RapidIO data rate and
input reference clock frequency selection.

*Table 2–19. Serial RapidIO Mode Transmitter PLL Multiplication Factors (Part 1 of 2)*

| Data Rate (Gbps) | Reference Clock Frequency (MHz) | Transmitter PLL Multiplication Factor |
|---|---|---|
| 1.25 | 62.5 | 10 |
| | 78.125 | 8 |
| | 125 | 5 |
| | 156.25 | 4 |
| | 250 (pre-divide by 2) | 5 |
| | 312.5 | 2 |

| Table 2–19. Serial RapidIO Mode Transmitter PLL Multiplication Factors (Part 2 of 2) | | |
|---|---|---|
| **Data Rate (Gbps)** | **Reference Clock Frequency (MHz)** | **Transmitter PLL Multiplication Factor** |
| 2.5 | 50 | 25 |
| | 62.5 | 20 |
| | 78.125 | 16 |
| | 100 (pre-divide by 2) | 25 |
| | 125 | 10 |
| | 156.25 | 8 |
| | 250 | 5 |
| | 312.5 | 4 |
| | 500 (pre-divide by 2) | 5 |
| 3.125 | 62.5 | 25 |
| | 78.125 | 20 |
| | 97.6563 | 16 |
| | 125 (pre-divide by 2) | 25 |
| | 156.25 | 10 |
| | 195.3125 | 8 |
| | 312.5 | 5 |
| | 390.625 (pre-divide by 2) | 8 |

In Serial RapidIO 1.25-Gbps (2.5-Gbps, 3.125-Gbps) mode, the local clock divider in each channel of the transceiver block divides the 625-MHz (1250-MHz, 1562.5-MHz) clock from the transmitter PLL by five to generate a 125-MHz (250-MHz, 312.5-MHz) parallel clock. This low-speed parallel clock output from the local clock divider block is used to clock the transmitter digital logic (PCS) of the associated channel. The local clock divider block also forwards the high-speed serial clock from the transmitter PLL to the serializer within its associated channel.

☞ The Quartus II software automatically selects the appropriate transmitter PLL bandwidth suited for Serial RapidIO data rate.

### Transmitter Phase Compensation FIFO Buffer

The transmitter phase compensation FIFO buffer compensates for the phase difference between the PLD clock that clocks in parallel data into the transmitter and the PCS clock that clocks the rest of the transmitter digital logic.

For more details about the transmitter phase compensation FIFO buffer architecture, refer to the transmitter Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

In Serial RapidIO 1.25-Gbps (2.5-Gbps, 3.125-Gbps) mode, the 125-MHz (250-MHz, 312.5-MHz) clock generated by the CMU clock divider block is divided by 2. The resulting 62.5-MHz (125-MHz, 156.25-MHz) clock is used to clock the read port of the FIFO buffer. This divide-by-two clock is also forwarded to the PLD logic array (on the `tx_clkout` port of its associated channel). If the `tx_coreclk` port is not instantiated, the clock signal on the `tx_clkout` port is automatically routed back to clock the write side of the transmitter phase compensation FIFO buffer. The 16-bit PLD-transceiver interface clocked at 62.5 MHz (125 MHz, 156.25 MHz) results into an effective Serial RapidIO data rate of 1.25 Gbps (2.5 Gbps, 3.125 Gbps).

In Serial RapidIO mode, the transmitter phase compensation FIFO is four words deep. The latency through the FIFO is two to three PLD-transceiver interface clock cycles.

Figure 2–23 shows the block diagram of transmitter phase compensation FIFO in Serial RapidIO mode.

*Figure 2–23. Transmitter Phase Compensation FIFO in Serial RapidIO Mode Note (1)*
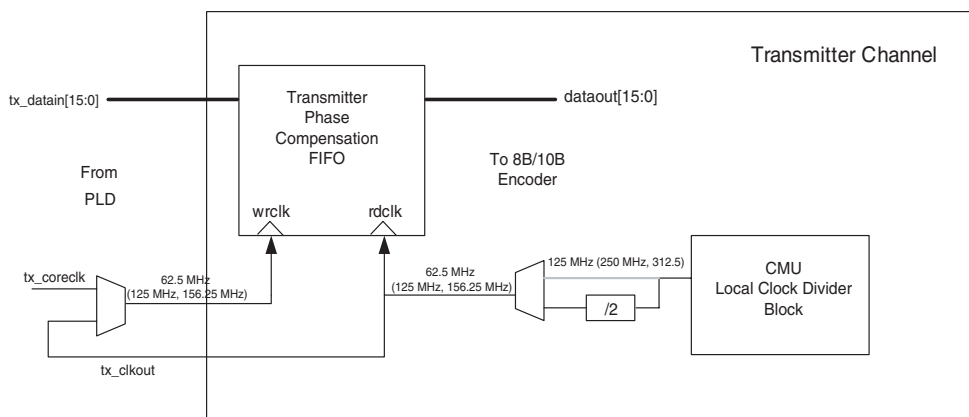


*Note to Figure 2–23:*
(1) The clock frequencies inside the parenthesis apply to 2.5 Gbps and 3.125 Gbps Serial RapidIO mode and the ones outside apply to 1.25 Gbps Serial RapidIO mode.

*Byte Serializer*

In Serial RapidIO 1.25 Gbps (2.5 Gbps, 3.125 Gbps) mode, the PLD-transceiver interface data is 16 bits wide and is clocked into the transmitter phase compensation FIFO at 62.5 MHz (125 MHz, 156.25 MHz). The byte serializer clocks in the 16-bit wide data from the transmitter phase compensation FIFO at 62.5 MHz (125 MHz, 156.25 MHz) and clocks out 8-bit data to the 8B/10B encoder at 125 MHz (250 MHz, 312.5 MHz). This allows clocking the PLD-transceiver interface at half the speed.

For more details about the byte serializer architecture, refer to the Byte Serializer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

The write port of the byte serializer is clocked by the divide-by-two version of the low-speed parallel clock from CMU. The read port is clocked by the low-speed parallel clock from CMU. The byte serializer clocks out the least significant byte of the 16-bit data first and the most significant byte last.

Figure 2–24 shows the block diagram of the byte serializer in Serial RapidIO mode.

*Figure 2–24. Byte Serializer in Serial RapidIO Mode Note (1)*
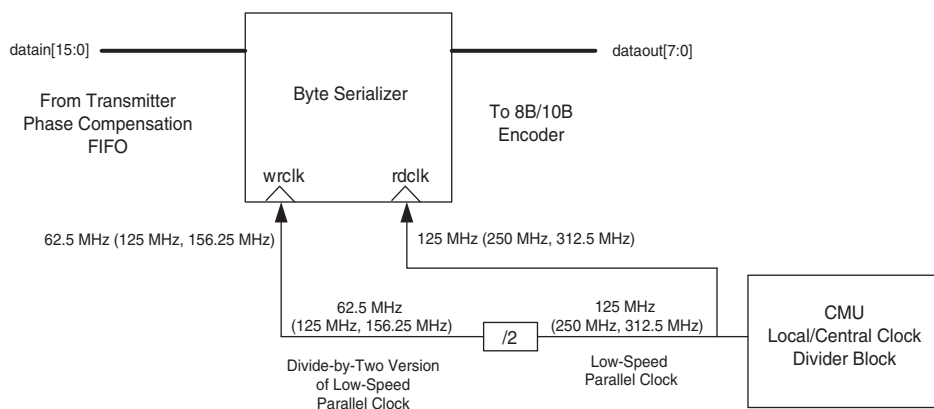


*Note to Figure 2–24:*
(1)  The clock frequencies inside the parenthesis apply to 2.5 Gbps and 3.125 Gbps Serial RapidIO mode and the ones outside apply to 1.25 Gbps Serial RapidIO mode.

### 8B/10B Encoder

In Serial RapidIO mode, the 8B/10B encoder clocks in 8-bit data and 1-bit control identifier from the transmitter phase compensation FIFO and generates a 10-bit encoded data. The 10-bit encoded data is fed to the serializer.

For more details about the 8B/10B encoder functionality, refer to the 8B/10B Encoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

### Serializer

In Serial RapidIO 1.25 Gbps (2.5 Gbps, 3.125 Gbps) mode, the 10-bit encoded data from the 8B/10B encoder is clocked into the 10:1 serializer with the low-speed parallel clock at 125 MHz (250 MHz, 312.5 MHz). The 10-bit data is clocked out of the serializer LSB to MSB at the high-speed effective serial clock rate at 1250 MHz (2500 MHz, 3125 MHz). The serial data output of the serializer is fed into the transmitter output buffer.

For more details about the serializer architecture, refer to the Serializer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

### Transmitter Buffer

Table 2–20 shows the transmitter buffer settings when configured in Serial RapidIO mode.

*Table 2–20. Transmitter Buffer Settings in Serial RapidIO Mode (Part 1 of 2)*

| Settings | Value |
|---|---|
| I/O Standard | 1.5-V PCML *(1)* |
| Programmable Differential Output Voltage ($V_{OD}$) | 400 - 1200 mV |
| Common Mode Voltage ($V_{CM}$) | 600 mV, 700 mV *(1)* |
| Differential Termination | 100 $\Omega$ *(2)* |
| Programmable pre-emphasis | Enabled *(3)* |

*Table 2–20. Transmitter Buffer Settings in Serial RapidIO Mode (Part 2 of 2)*

| Settings | Value |
|----------|-------|
| $V_{CCH}$ (Transmitter Buffer Power) | 1.5 V |

*Notes to Table 2–20:*

(1)   The common mode voltage ($V_{CM}$) setting is selectable in the MegaWizard Plug-In Manager.

(2)   The I/O standard and differential termination settings are defaulted to 1.5-V PCML and 100 Ω, respectively. If you select any other setting for the I/O standard or differential termination in the Assignment Editor, the Quartus II compiler issues an error message.

(3)   The transmitter buffer has five programmable first post-tap pre-emphasis settings.

## Serial RapidIO Mode Receiver Architecture

This section lists sub-blocks within the receiver channel configured in Serial RapidIO mode (Figure 2–25). The sub-blocks are described in order from the serial receiver input buffer to the receiver phase compensation FIFO buffer at the transceiver-PLD interface.

*Figure 2–25. Serial RapidIO Mode Receiver Architecture*
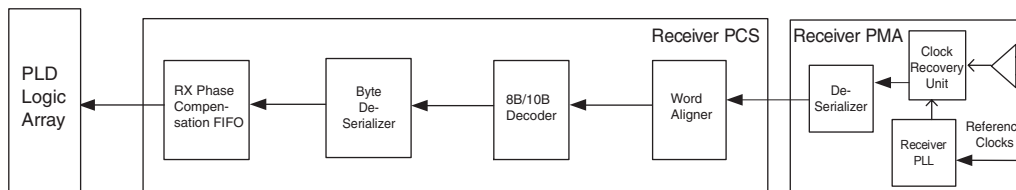


### Receiver Buffer

Table 2–21 shows the receiver buffer settings when configured in Serial RapidIO mode.

*Table 2–21. Receiver Buffer Settings in Serial RapidIO Mode   (Part 1 of 2)*

| Settings | Value |
|----------|-------|
| I/O Standard | 1.2-V PCML, 1.5-V PCML, 3.3-V PCML, Differential LVPECL, LVDS |
| Input Common Mode Voltage (Rx $V_{CM}$) | 850 mV, 1200 mV *(1)* |

| Table 2–21. Receiver Buffer Settings in Serial RapidIO Mode  (Part 2 of 2) | |
| --- | --- |
| **Settings** | **Value** |
| Differential Termination | 100 Ω *(2)* |
| Programmable Equalization | Enabled *(3)* |
| Coupling | AC |

*Notes to Table 2–21:*
(1)  The common mode voltage (Rx $V_{CM}$) is selectable in the MegaWizard Plug-In Manager.
(2)  The differential termination setting is defaulted to 100 Ω. If you select any other setting for differential termination in the Assignment Editor, the Quartus II compiler issues an error message.
(3)  The receiver buffer has five programmable equalization settings.

### Receiver PLL and Clock Recovery Unit

In Serial RapidIO 1.25 Gbps (2.5 Gbps, 3.125 Gbps) mode, the receiver PLL in each transceiver channel is fed by an input reference clock. The receiver PLL in conjunction with the clock recovery unit generates two clocks: a half-rate high-speed serial recovered clock at 625 MHz (1250 MHz, 1562.5 MHz) that feeds the deserializer and a low-speed parallel recovered clock at 125 MHz (250 MHz, 312.5 MHz) that feeds the receiver's digital logic.

You can set the clock recovery unit in either automatic lock mode or manual lock mode. In automatic lock mode, the PPM detector and the phase detector within the receiver channel automatically switch the receiver PLL between lock-to-reference and lock-to-data modes. In manual lock mode, you can control the receiver PLL switch between lock-to-reference and lock-to-data modes via the rx_locktorefclk and rx_locktodata signals.

For more details about the CRU lock modes, refer to the Receiver PLL section and Clock Recovery Unit (CRU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

The reference clock input to the receiver PLL can be derived from one of the following components:

■ One of the two available dedicated reference clock input pins (REFCLK0 or REFCLK1) of the associated transceiver block
■ PLD global clock network (must be driven directly from an input clock pin and cannot be driven by user logic or enhanced PLL)
■ Inter-transceiver block lines driven by reference clock input pins of other transceiver blocks

Table 2–22 specifies the receiver input reference clock options available in Serial RapidIO mode.

*Table 2–22. Serial RapidIO Mode Input Reference Clock Specifications*

| Data Rate (Gbps) | Reference Clock Frequency (MHz) | I/O Standard | Coupling | Termination |
|---|---|---|---|---|
| 1.25 | 62.5, 78.125,125, 156.25, 250, 312.5 | 1.2 V PCML, 1.5 V PCML, 3.3 V PCML, Differential LVPECL, LVDS | AC | On-chip |
| 2.5 | 50, 62.5, 78.125, 100, 125, 156.25, 250, 312.5, 500 | | | |
| 3.125 | 62.5, 78.125, 97.6563, 125, 156.25, 195.3125, 312.5, 390.625 | | | |

*Deserializer*

In Serial RapidIO 1.25 Gbps (2.5 Gbps, 3.125 Gbps) mode, the 1:10 deserializer clocks in serial data from the receiver buffer using the high-speed serial recovered clock. The 10-bit de-serialized data is clocked out to the word aligner using the low-speed parallel recovered clock at 125 MHz (250 MHz, 312.5 MHz). The deserializer assumes that the transmission bit order is LSB to MSB; that is, the LSB of a data word is received earlier in time than its MSB.

For more details on the deserializer architecture, refer to the Deserializer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

*Word Aligner*

The word aligner clocks in the 10-bit data from the deserializer and restores the word boundary of the upstream transmitter.

For more details about the word aligner architecture, refer to the section *"Word Aligner" on page 2–13* in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

In Serial RapidIO mode, the word aligner comprises of the following three modules:

■ Pattern detector module
■ Pattern aligner module
■ Run-length violation detection module

**Pattern Detector**

In Serial RapidIO mode, the Quartus II software automatically configures 10-bit K28.5 (10'b0101111100) as the word alignment pattern. After coming out of reset (`rx_digitalreset`), when the pattern detector detects either disparities of the K28.5 control word, it asserts the `rx_patterndetect` signal for one parallel clock cycle. When the pattern aligner has aligned the incoming data to the desired word boundary, the pattern detector asserts `rx_patterndetect` signal only if the word alignment pattern is found in the current word boundary.

**Pattern Aligner**

In Serial RapidIO mode, the pattern aligner employs an automatic synchronization state machine. The Quartus II software automatically configures the synchronization state machine to indicate synchronization when the receiver receives 127 K28.5 (10'b0101111100 or 10'b1010000011) synchronization code groups without receiving an intermediate invalid code group. Once synchronized, the state machine indicates loss of synchronization when it detects three invalid code groups separated by fewer than 255 valid code groups or when it is reset.

Receiver synchronization is indicated on the `rx_syncstatus` port of each channel. A high on the `rx_syncstatus` port indicates that the lane is synchronized and a low indicates that it has fallen out of synchronization.

Table 2–23 lists the synchronization state machine parameters when configured in Serial RapidIO mode.

*Table 2–23. Synchronization State Machine Parameters in Serial RapidIO Mode*

| | |
|---|---|
| Number of valid K28.5 code groups received to achieve synchronization | 127 |
| Number of errors received to lose synchronization | 3 |
| Number of continuous good code groups received to reduce the error count by 1 | 255 |

☞ In an 8B/10B encoded data stream, a /K28.7/ special code group followed by any of the data code groups /D3.y/, /D11.y/, /D12.y/, /D19.y/, /D20.y/, /D28.y/ or /K28.y/ (where y ranges from 0 to 7), may cause the /K28.5/ alignment pattern to appear across the word boundary. Serial RapidIO protocol allows /K28.7/ transmission only during test and debug.

Figure 2–26 shows the synchronization state machine implemented in Serial RapidIO functional mode.

*Figure 2–26. Synchronization State Machine in Serial RapidIO Mode*
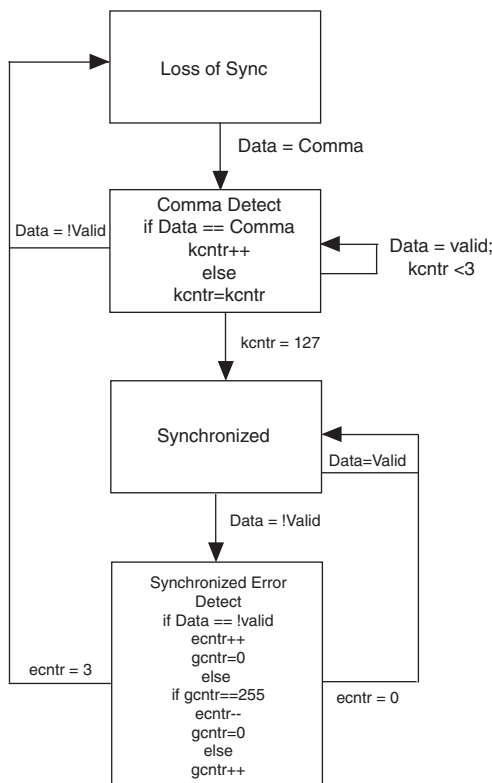


The word aligner block asserts an error flag on the `rx_disperr` and `rx_errdetect` ports if the received 10-bit code is detected with incorrect running disparity. The error flag signal (`rx_disperr`) has the same delay from the word aligner to the PLD-transceiver interface as the received data.

### 8B/10B Decoder

In Serial RapidIO mode, the 8B/10B decoder clocks in 10-bit data from the word aligner and decodes it into 8-bit data + 1-bit control identifier. The 8-bit decoded data is fed to the byte deserializer.

For more details about the 8B/10B decoder functionality, refer to the 8B/10B Decoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

If the received 10-bit code group is not a part of valid Dx.y or Kx.y code groups, the 8B/10B decoder block asserts an error flag on the `rx_errdetect` port. The error flag signal (`rx_errdetect`) has the same data path delay from the 8B/10B decoder to the PLD-transceiver interface as the invalid code group.

### Byte Deserializer

In Serial RapidIO 1.25 Gbps (2.5 Gbps, 3.125 Gbps) mode, the PLD-receiver interface data is 16 bits wide and is clocked out of the receiver phase compensation FIFO at 62.5 MHz (125 MHz, 156.25 MHz). The byte deserializer clocks in the 8-bit wide data from the 8B/10B decoder at 125 MHz (250 MHz, 312.5 MHz) and clocks out 16-bit wide data to the receiver phase compensation FIFO at 62.5 MHz (125 MHz, 156.25 MHz). This allows clocking the PLD-transceiver interface at half the speed.
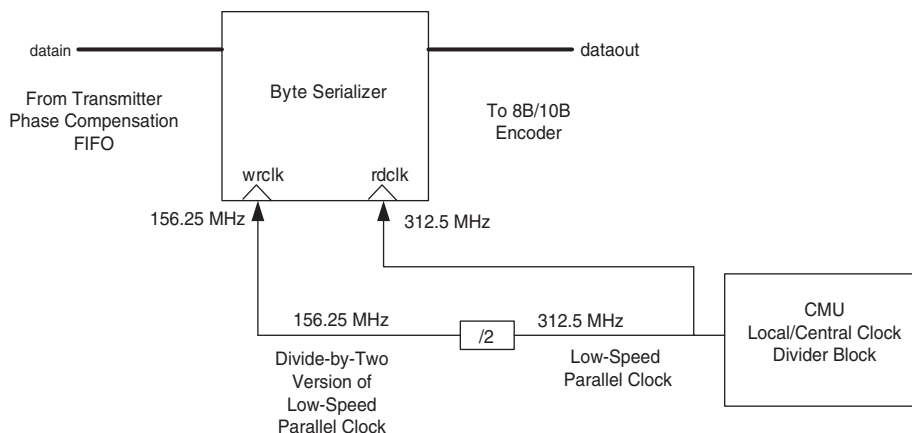
For more details about byte deserializer architecture, refer to the Byte Deserializer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

In Serial RapidIO mode, the write port of the byte deserializer is clocked by the low-speed parallel recovered clock and the read port is clocked by divide-by-two version of this clock.

Due to 8-bit to 16-bit byte deserialization, the byte ordering at the PLD-receiver interface might be incorrect. If required, you must implement the byte ordering logic in the PLD core to correct for this situation.

Figure 2–27 shows the block diagram of the byte deserializer in Serial RapidIO mode.

*Figure 2–27. Byte Deserializer in Serial RapidIO Mode Note (1)*
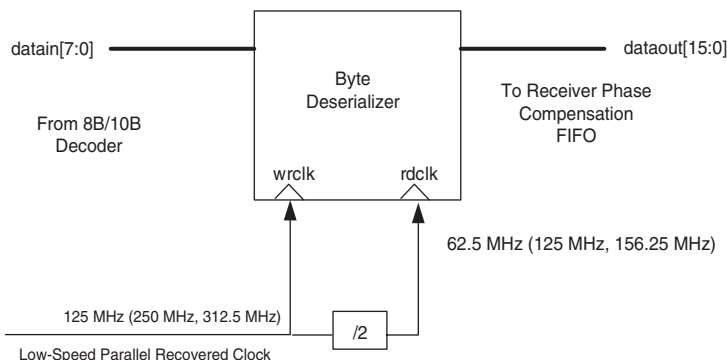


*Note to Figure 2–27:*
(1) The clock frequencies inside the parenthesis apply to 2.5 Gbps and 3.125 Gbps Serial RapidIO mode and the ones outside apply to 1.25 Gbps Serial RapidIO mode.

### Receiver Phase Compensation FIFO Buffer

The receiver phase compensation FIFO buffer compensates for the phase difference between the local receiver PLD clock and the receiver PCS clock.

For more details about the receiver phase compensation FIFO buffer architecture, refer to the Receiver Phase Compensation FIFO Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

In Serial RapidIO 1.25 Gbps (2.5 Gbps, 3.125 Gbps) mode, the 125 MHz (250 MHz, 312.5 MHz) low-speed parallel recovered clock is divided by 2. The resulting 62.5 MHz (125 MHz, 156.25 MHz) clock is used to clock the write port of the FIFO buffer. This divide-by-two clock is also forwarded to the PLD logic array (on the rx_clkout port). If the rx_coreclk port is not instantiated, the recovered clock signal on the rx_clkout port is automatically routed back to clock the read side of the receiver phase compensation FIFO buffer. The 16-bit PLD-receiver interface clocked at 62.5 MHz (125 MHz, 156.25 MHz) results into an effective Serial RapidIO data rate of 1 Gbps (2 Gbps, 3.125 Gbps).

In Serial RapidIO mode, the receiver phase compensation FIFO is four words deep. The latency through the FIFO is one to two PLD-transceiver interface clock cycles.

Figure 2–28 shows the block diagram of receiver phase compensation FIFO in Serial RapidIO mode.

*Figure 2–28. Receiver Phase Compensation FIFO in RapidIO Mode Note (1)*
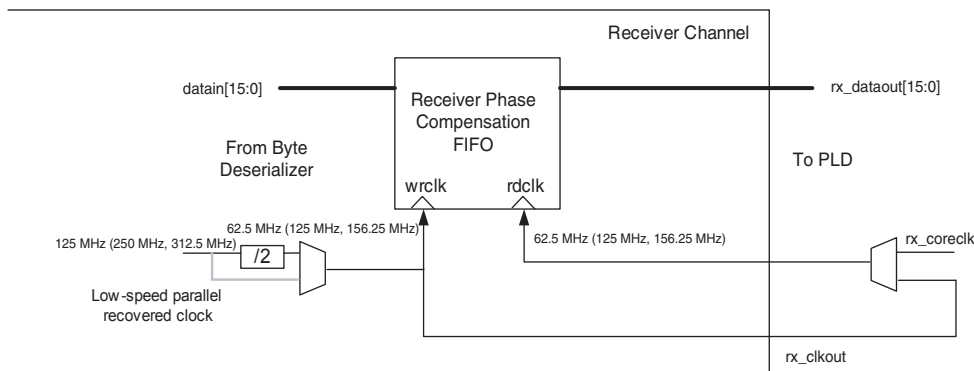


*Note to Figure 2–28:*
(1)    The clock frequencies inside the parenthesis apply to 2.5 Gbps and 3.125 Gbps Serial RapidIO mode and the ones outside apply to 1.25 Gbps Serial RapidIO mode.

# Basic Single-Width Mode

Use the Basic single-width mode for custom protocols that are not part of the pre-defined supported protocols; for example, PIPE. With some restrictions, the following PCS blocks are available:

■    Transmitter phase compensation FIFO buffer
■    Transmitter byte serializer
■    8B/10B encoder
■    Word aligner
■    Rate matcher
■    8B/10B decoder
■    Byte deserializer
■    Byte ordering block
■    Receiver phase compensation FIFO buffer

The byte ordering block is available only in reverse serial loopback configuration in Basic mode. The rate matcher is coupled with the 8B/10B code groups, which requires the use of the 8B/10B encoder or decoder either in the PCS or PLD logic array.

## Basic Single-Width Mode with x4 Clocking

In Basic single-width mode, the ALT2GXB MegaWizard Plug-In Manager provides a ×4 option under the **Which subprotocol will you be using?** option. If you select this option, all four transmitter channels within the

transceiver block are clocked by clocks generated from the central clock divider block. The low-speed clock from the central clock divider block clocks the bonded transmitter PCS logic in all four channels. This reduces the transmitter channel-to-channel skew within the transceiver block. Each receiver channel within the transceiver block is clocked individually by the recovered clock from its own CRU.

☞ Configuring transceivers in this mode yields low transmitter channel-to-channel skew within a transceiver block. It does not provide skew reduction for channels placed across transceiver blocks.

Figure 2–29 shows the data path in this mode.

*Figure 2–29. Basic Single-Width Mode with ×4 Clocking*
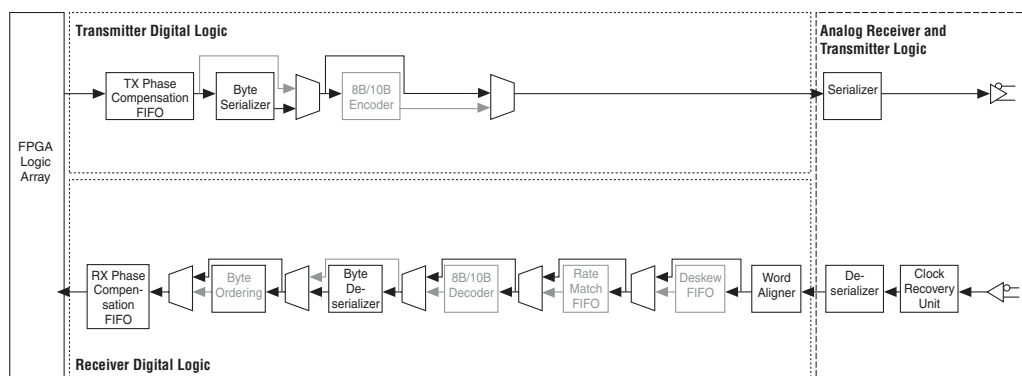


The transmitter data path consists of a 16-bit PLD-transceiver interface, transmitter phase compensation FIFO, 16:8-bit byte serializer, and 8:1 serializer.

The receiver data path consists of the CRU, 1:8 deserializer, bit-slip word aligner, 8:16 byte deserializer, receiver phase compensation FIFO, and 16-bit Transceiver-PLD interface.

**Transceiver Placement Limitations**
If one or more channels in a transceiver block are configured to Basic single-width mode with ×4 clocking option enabled, the remaining channels in that transceiver block must either have the same configuration or must be unused. All used channels within a transceiver block configured to this mode must also run at the same data rate. All

channels within the transceiver block configured to this mode must be instantiated using the same ALT2GXB MegaWizard Plug-In Manager instance.

Figures 2–30 and 2–31 show examples of legal and illegal transceiver placements with respect to the Basic single-width mode with ×4 clocking enabled.

*Figure 2–30. Examples of Legal Transceiver Placement*



*Figure 2–31. Examples of Illegal Transceiver Placement*



**Clocking and Reset Recommendations**

To minimize the transmitter channel to channel skew across transceiver blocks, Altera recommends that you follow the protocols listed below:

■ Using the dedicated REFCLK pins of the centrally located transceiver block in your design to provide the input reference clock for all transceiver blocks. This reduces the skew on the input reference clock driving the CMU PLL in each transceiver block. For example, in a design with 12 channels placed across Banks 13, 14, and 15, use the REFCLK pins of Bank 14 to provide the input reference clock.

■ De-asserting the tx_digitalreset signal of all used transceiver blocks simultaneously after pll_locked signal from all active transceiver blocks goes high.

Figure 2–32 shows the recommended clocking for 12 transceiver channels across transceiver banks 13, 14, and 15 in the EP1AGX90EF1152 device.

*Figure 2–32. Clocking Recommendations to Minimize Transmitter Channel-To-Channel Skew*
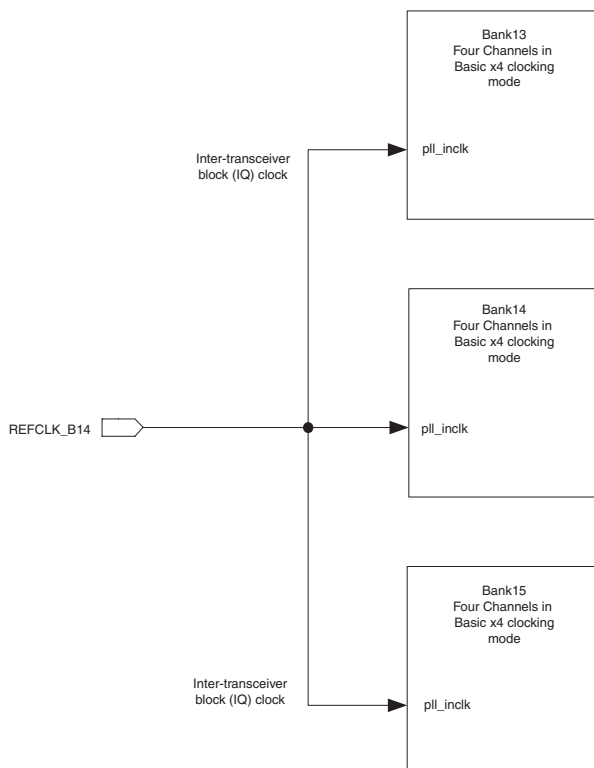


**XAUI Mode**

This section briefly introduces the XAUI standard and the code groups and ordered sets associated with this self-managed interface. For full details about the XAUI standard, refer to clause 47 and 48 in the 10 Gigabit Ethernet standard (IEEE 802.3ae).

Arria GX devices contain embedded macros dedicated to the XAUI protocol, including synchronization, channel deskew, rate matching, XGMII Extender Sublayer (XGXS) to 10 Gigabit Media Independent Interface (XGMII) and XGMII to XGXS code-group conversion macros.

The XAUI standard is an optional self-managed interface that is inserted between the reconciliation sublayer and the PHY layer to transparently extend the physical reach of XGMII.

This section is organized into transmitter and receiver data path modules when configured for XAUI mode. The description for each module only covers details specific to XAUI functional mode support.

For a general description of each module, refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

XAUI addresses several physical limitations of XGMII. XGMII signaling is based on the HSTL Class I single-ended I/O standard, which has an electrical distance limitation of approximately 7 cm. XAUI uses a low-voltage differential signaling method, so the electrical limitation is increased to approximately 50 cm. Another advantage of XAUI is the simplification of backplane and board trace routing. XGMII is composed of 32 transmit channels, 32 receive channels, one transmit clock, one receive clock, four transmitter control characters, and four receive control characters for a total of a 74-pin wide interface. XAUI consists of four differential transmitter channels and four differential receiver channels for a total of a 16-pin wide interface. This reduction in pin count significantly simplifies the routing process in the layout design. Figure 2–33 shows the relationships between the XGMII and XAUI layers.

*Figure 2–33. XGMII and XAUI Relationship*
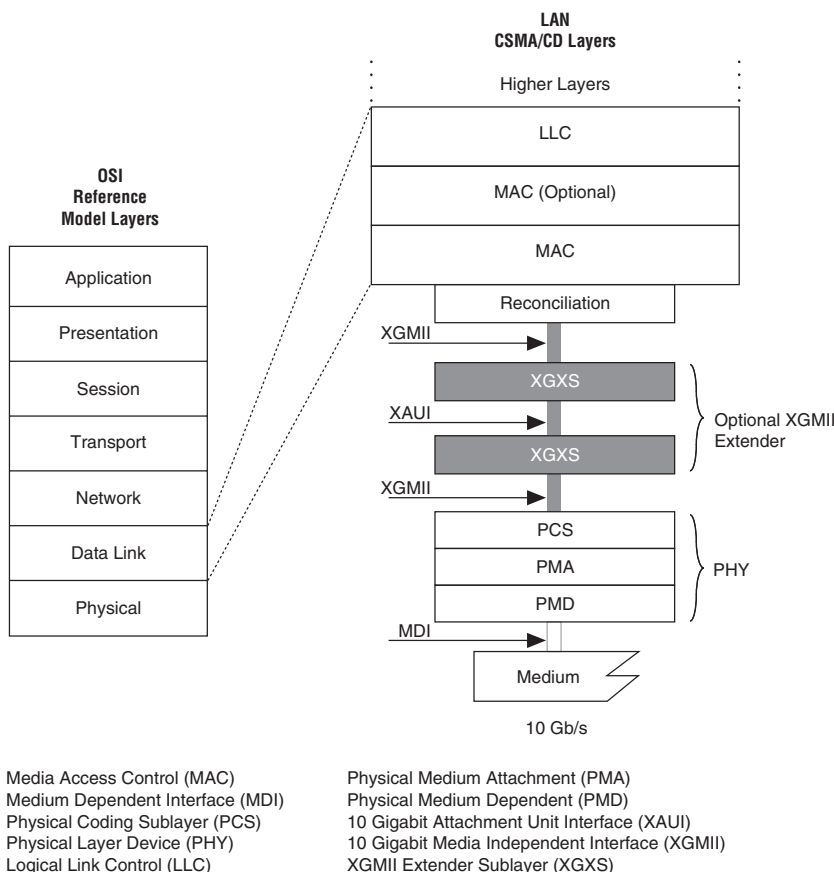


The XGMII interface consists of four lanes of eight bits. At the transmit side of the XAUI interface, the data and control characters are converted within the XGXS into an 8B/10B encoded data stream. Each data stream is then transmitted across a single differential pair running at 3.125 Gbps. At the XAUI receiver, the incoming data is decoded and mapped back to the 32-bit XGMII format. This provides a transparent extension of the physical reach of the XGMII and also reduces the interface pin count.

XAUI functions as a self-managed interface because code group synchronization, channel deskew, and clock domain decoupling is handled with no upper layer support requirements. This functionality is

based on the PCS code groups that are used during the IPG time and idle periods. PCS code groups are mapped by the XGXS to XGMII characters specified in Table 2–24.

*Table 2–24. XGMII Character to PCS Code-Group Mapping*

| XGMII TXC | XGMII TXD *(1)* | PCS Code Group | Description |
|:---:|:---:|:---:|:---|
| 0 | 00 through FF | Dxx.y | Normal data transmission |
| 1 | 07 | K28.0, K28.3, or K28.5 | Idle in \|\|I\|\| |
| 1 | 07 | K28.5 | Idle in \|\|T\|\| |
| 1 | 9C | K28.4 | Sequence |
| 1 | FB | K27.7 | Start |
| 1 | FD | K29.7 | Terminate |
| 1 | FE | K30.7 | Error |
| 1 | Other value | — | Reserved XGMII character |
| 1 | Any other value | K30.7 | Deleted XGMII character |

*Note to Table 2–24:*
(1) Values in TXD column are in hexadecimal.

Figure 2–34 shows an example of the mapping between XGMII characters and the PCS code groups that are used in XAUI. The idle characters are mapped to a pseudo random sequence of /A/, /R/, and /K/ code groups.

*Figure 2–34. XGMII Character to PCS Code-Group Mapping*

The PCS code-groups are sent via PCS ordered sets. PCS ordered sets consist of combinations of special and data code groups defined as a column of code groups. These ordered sets are composed of four code groups beginning in Lane 0. Table 2–25 lists the defined idle ordered sets (||I||) that are used for the self managed properties of XAUI.

*Table 2–25. Defined Idle Ordered Set*

| Code | Ordered Set | Number of Code Groups | Encoding |
|---|---|---|---|
| ||I|| | Idle | | Substitute for XGMII Idle |
| ||K|| | Synchronization column | 4 | /K28.5/K28.5/K28.5/K28.5 |
| ||R|| | Skip column | 4 | /K28.0/K28.0/K28.0/K28.0 |
| ||A|| | Align column | 4 | /K28.3/K28.3/K28.3/K28.3 |

## XAUI Mode Transmitter Architecture

This section lists sub-blocks within the transmitter channel configured in XAUI mode (Figure 2–35). The sub-blocks are described in order from the PLD-Transceiver parallel interface to the serial transmitter buffer.

*Figure 2–35. XAUI Transmitter Architecture*
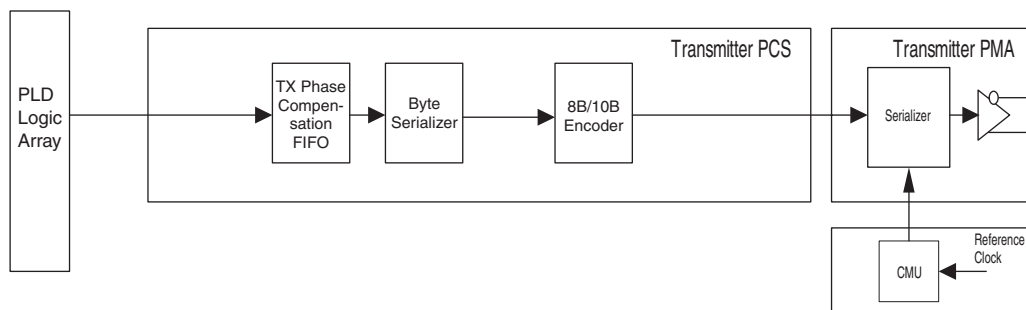


### Clock Multiplier Unit (CMU)

The clock multiplier unit takes in a reference clock and synthesizes the clocks that are used to clock the transmitter digital logic (PCS), the serializer, and the PLD-transceiver interface.

For more details about CMU architecture, refer to the Clock Multiplier Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

In XAUI mode, the CMU block consists of the following components:

■ Transmitter PLL that generates high-speed serial clock for the serializer
■ Local clock divider block that generates low-speed parallel clock for transmitter digital logic and PLD-transceiver interface

**Input Reference Clock**
In XAUI mode for Arria GX devices, the only supported input reference clock frequency is 156.25 MHz.

The reference clock input to the transmitter PLL can be derived from the following components:

■ One of the two available dedicated reference clock input pins (REFCLK0 or REFCLK1) of the associated transceiver block
■ Inter-transceiver block lines driven by reference clock input pins of other transceiver blocks

Altera recommends using the dedicated reference clock input pins (REFCLK0 or REFCLK1) to provide reference clock for the transmitter PLL.

*Dedicated Reference Clock Pin Specifications*

Table 2–26 shows the I/O standards allowed for the reference clock pins.

| *Table 2–26. Xaui Mode Reference Clock Specifications* | | | |
|---|---|---|---|
| **Frequency** | **I/O Standard** | **Coupling** | **Termination** |
| 156.25 MHz | 1.2-V PCML, 1.5-V PCML, 3.3-V PCML, Differential LVPECL, LVDS | AC | On-chip |

In ×4 mode for XAUI, the central clock divider in the transceiver block divides the 1562.5 MHz clock from the transmitter PLL by 5 to generate a 312.5 MHz parallel clock. This low-speed parallel clock output from the central clock divider block is used to clock the transmitter digital logic (PCS) in all channels of the transceiver block. The central clock divider block also forwards the high-speed serial clock from the transmitter PLL to the serializer within each channel. Because all four channels in the transceiver block are clocked with the same clock, the channel-to-channel skew is minimized.

🖝 The Quartus II software automatically selects the appropriate transmitter PLL bandwidth suited for the XAUI data rate.

Figure 2–36 shows the CMU implemented in XAUI mode.

*Figure 2–36. XAUI Mode CMU*
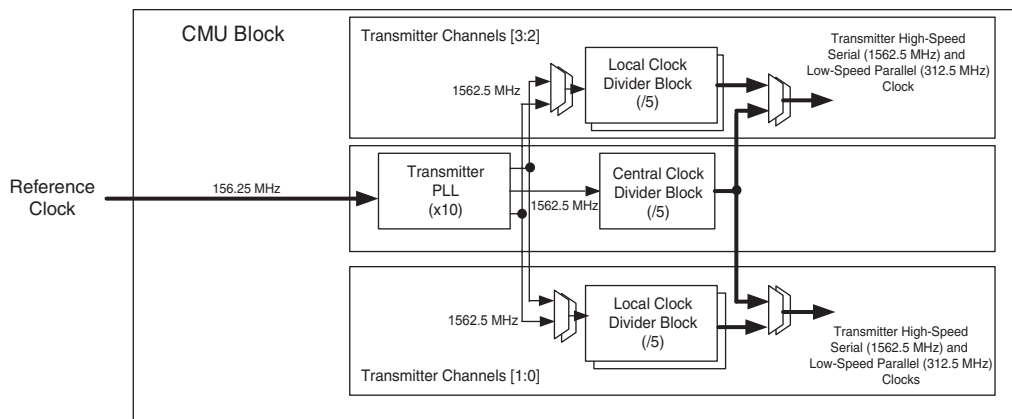


**Clock Synthesis**

In XAUI mode, the 156.25-input reference clock is fed to the transmitter PLL. Since the transmitter PLL implements a half-rate VCO, it multiplies the 156.25-MHz input clock by 10 to generate a 1562.5-MHz (3.125-Gbps) high speed serial clock. This high-speed serial clock feeds the central clock divider and four local clock dividers of the transceiver block.

*Transmitter Phase Compensation FIFO Buffer*

The transmitter phase compensation FIFO buffer compensates for the phase difference between the PLD clock that clocks in parallel data into the transmitter and the PCS clock that clocks the rest of the transmitter digital logic.

👣 For more details about the transmitter phase compensation FIFO buffer architecture, refer to the Transmitter Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

In XAUI 3.125 Gbps mode, the 312.5 MHz clock generated by the CMU clock divider block is divided by two. The resulting 156.25 MHz clock is used to clock the read port of the FIFO buffer. This divide-by-two clock is also forwarded to the PLD logic array (on the `tx_clkout` port of its

associated channel). If the `tx_coreclk` port is not instantiated, the clock signal on the `tx_clkout` port is automatically routed back to clock the write side of the transmitter phase compensation FIFO buffer. The 16-bit PLD-transceiver interface clocked at 156.25 MHz results in an effective XAUI data rate of 3.125 Gbps.

In XAUI mode, the transmitter phase compensation FIFO is four words deep. The latency through the FIFO is two to three PLD transceiver interface clock cycles.

Figure 2–37 shows the block diagram of transmitter phase compensation FIFO in XAUI mode.

*Figure 2–37. Transmitter Phase Compensation FIFO in XAUI Mode*
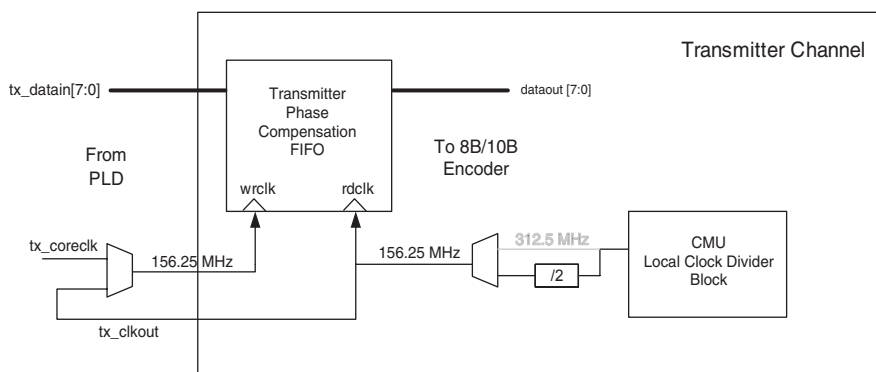


*Byte Serializer*

In XAUI 3.125 Gbps mode the PLD-transceiver interface data is 16 bits wide and is clocked into the transmitter phase compensation FIFO at 156.25 MHz. The byte serializer clocks in the 16-bit wide data from the transmitter phase compensation FIFO at 156.25 MHz and clocks out 8-bit data to the 8B/10B encoder at 312.5 MHz. This allows clocking the PLD-transceiver interface at half the speed.

For more details about the byte serializer architecture, refer to the Byte Serializer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

The write port of the byte serializer is clocked by the divide-by-two version of the low-speed parallel clock from CMU. The read port is clocked by the low-speed parallel clock from CMU. The byte serializer clocks out the least significant byte of the 16-bit data first and the most significant byte last.

Figure 2–38 shows the block diagram of the byte serializer in XAUI mode.

*Figure 2–38. Byte Serializer in XAUI Mode*



### 8B/10B Encoder

In XAUI mode, the 8B/10B encoder clocks in 8-bit data and 1-bit control identifier from the transmitter phase compensation FIFO and generates a 10-bit encoded data. The 10-bit encoded data is fed to the serializer.

For more details about the 8B/10B encoder functionality, refer to the 8B/10B Encoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

**XGMII Character to PCS Code-Group Mapping**
In XAUI mode, the 8B/10B encoder in Arria GX devices is controlled by a global transmitter state machine that maps various 8-bit XGMII codes to 10-bit PCS code groups. This state machine complies with the IEEE 802.3ae PCS transmit specification. Figure 2–39 shows the PCS transmit source state diagram specified in clause 48 of the IEEE P802.3ae.

*Figure 2–39. IEEE 802.3ae PCS Transmit Source State Diagram*

!reset
!(TX=||IDLE|| + TX=||Q||

**SEND_Q**
IF TX=||T|| THEN cvtx_terminate
tx_code_group<39:0> ⇐
ENCODE(TX)
PUDR

next_ifg = A_CNT≠0

(next_ifg + A_CNT≠0)

reset

**SEND_A**
tx_code_group<39:0> ⇐ ||A||
next_ifg ⇐ K
PUDR

**SEND_K**
tx_code_group<39:0> ⇐ ||K||
next_ifg ⇐ A
PUDR
UCT
B

Q_det

!Q_det

**SEND_Q**
tx_code_group<39:0> ⇐ TQMSG
Q_det ⇐ K
PUDR
UCT

A_CNT≠0 *
cod_sel=1
B

A

**SEND_RANDOM_K**
tx_code_group<39:0> ⇐ ||K||
PUDR

**SEND_RANDOM_R**
tx_code_group<39:0> ⇐ ||R||

A_CNT≠0 *
cod_sel=1
A

A_CNT≠0 *
cod_sel=1
B

A_CNT=0

A_CNT=0

A
A_CNT≠0 *
cod_sel=1

**SEND_RANDOM_A**
tx_code_group<39:0> ⇐ ||A||
PUDR

Q_det

B
!Q_det *
cod_sel=1

**SEND_RANDOM_Q**
tx_code_group<39:0> ⇐ TQMSG
Q_det ⇐ FALSE
PUDR

A
!Q_det *
cod_set=1

B
cod_set=1

A
cod_set=1

Table 2–27 lists the XGMII character to PCS code-group mapping.

| Table 2–27. XGMII Character to PCS Code-Group Mapping | | | |
|---|---|---|---|
| **XGMII TXC** | **XGMII TXD** *(1)* | **PCS Code Group** | **Description** |
| 0 | 00 through FF | Dxx.y | Normal data transmission |
| 1 | 07 | K28.0, K28.3, or K28.5 | Idle in \|\|I\|\| |
| 1 | 07 | K28.5 | Idle in \|\|T\|\| |
| 1 | 9C | K28.4 | Sequence |
| 1 | FB | K27.7 | Start |
| 1 | FD | K29.7 | Terminate |
| 1 | FE | K30.7 | Error |
| 1 | Other value | | Reserved XGMII character |
| 1 | Any other value | K30.7 | Invalid XGMII character |

*Note to Table 2–27:*

(1)    Values in TXD column are in hexadecimal.

### Serializer

In XAUI 3.125 Gbps mode, the 10-bit encoded data from the 8B/10B encoder is clocked into the 10:1 serializer with the low speed parallel clock at 312.5 MHz. The 10-bit data is clocked out of the serializer LSB to MSB at the high-speed effective serial clock rate at 3125 MHz. The serial data output of the serializer is fed into the transmitter output buffer.
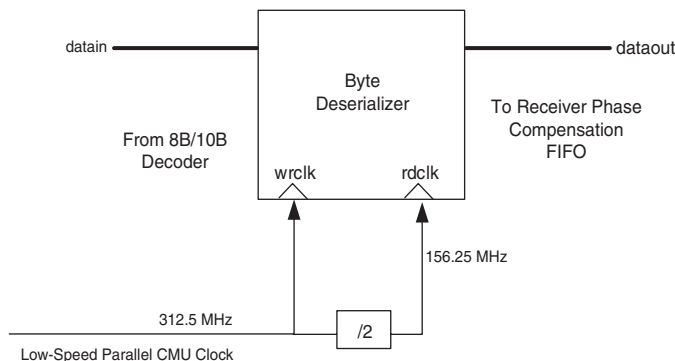
For more details about the serializer architecture, refer to the serializer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

*Transmitter Buffer*

Table 2–28 shows the transmitter buffer settings when configured in XAUI mode.

*Table 2–28. Transmitter Buffer Settings in XAUI Mode*

| Settings | Value |
|---|---|
| I/O Standard | 1.5-V PCML *(1)* |
| Programmable Differential Output Voltage (V$_{OD}$) | 400 - 1200 mV |
| Common Mode Voltage (V$_{CM}$) | 600 mV, 700 mV *(1)* |
| Differential Termination | 100 Ω *(2)* |
| Programmable pre-emphasis | Enabled *(3)* |
| V$_{CCH}$ (Transmitter Buffer Power) | 1.5 V |

*Notes to Table 2–28:*
(1) The common mode voltage (V$_{CM}$) settings are selectable in the MegaWizard Plug-In Manager.
(2) The I/O standard and differential termination settings are defaulted to 1.5-V PCML and 100 Ω respectively. If you select any other setting for the I/O standard or differential termination in the Assignment Editor, the Quartus II compiler issues an error message.
(3) The transmitter buffer has five programmable first post-tap pre-emphasis settings.

## XAUI Mode Receiver Architecture

This section lists sub-blocks within the receiver channel configured in XAUI mode (Figure 2–40). The sub-blocks are described in order from the serial receiver input buffer to the receiver phase compensation FIFO buffer at the transceiver-PLD interface.

*Figure 2–40. XAUI Mode Receiver Architecture*
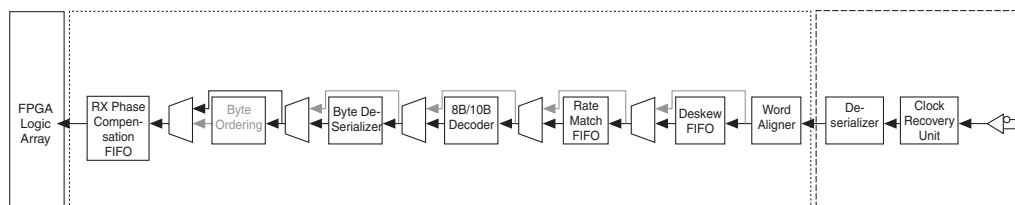
*Receiver Buffer*

Table 2–29 shows the receiver buffer settings when configured in XAUI mode.

**Table 2–29. Receiver Buffer Settings in XAUI Mode**

| Settings | Value |
|---|---|
| I/O Standard | 1.2-V PCML, 1.5-V PCML, 3.3-V PCML, Differential LVPECL, LVDS |
| Input Common Mode Voltage (Rx V$_{CM}$) | 850 mV, 1200 mV *(1)* |
| Differential Termination | 100 Ω *(2)* |
| Programmable equalization | Enabled *(3)* |
| Coupling | AC |

*Notes to Table 2–29:*
(1) The common mode voltage (Rx VCM) is selectable in the MegaWizard Plug-In Manager.
(2) The differential termination setting is defaulted to **100 Ω** If you select any other setting for differential termination in the Assignment Editor, the Quartus II compiler issues an error message.
(3) The receiver buffer has five programmable equalization settings.

*Receiver PLL and Clock Recovery Unit*

In XAUI 3.125 Gpbs mode, the receiver PLL in each transceiver channel is fed by an input reference clock. The receiver PLL in conjunction with the clock recovery unit generates two clocks: a half-rate high-speed serial recovered clock at 1562.5 MHz that feeds the deserializer and a low-speed parallel recovered clock at 312.5 MHz that feeds the receiver's digital logic.

You can set the clock recovery unit in either automatic lock mode or manual lock mode. In automatic lock mode, the PPM detector and the phase detector within the receiver channel automatically switch the receiver PLL between lock-to-reference and lock-to-data modes. In manual lock mode, you can control the receiver PLL switch between lock to-reference and lock-to-data modes via the rx_locktorefclk and rx_locktodata signals.

For more details about the CRU lock modes, refer to the Receiver PLL section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

The reference clock input to the receiver PLL can be derived from one of the following pins:

■ One of the two available dedicated reference clock input pins (REFCLK0 or REFCLK1) of the associated transceiver block
■ PLD global clock network (must be driven directly from an input clock pin and cannot be driven by user logic or enhanced PLL)
■ Inter-transceiver block lines driven by reference clock input pins of other transceiver blocks

### Deserializer

In XAUI 3.125 Gbps mode, the 1:10 deserializer clocks in serial data from the receiver buffer using the high-speed serial recovered clock. The 10-bit deserialized data is clocked out to the word aligner using the low-speed parallel recovered clock at 312.5 MHz. The deserializer assumes that the transmission bit order is LSB to MSB; that is, the LSB of a data word is received earlier in time than its MSB.

For more details about the deserializer architecture, refer to the Deserializer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

### Word Aligner

The word aligner clocks in 10-bit data from the deserializer and restores the word boundary of the upstream transmitter.

For more details about the word aligner architecture, refer to the Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

In XAUI mode, the word aligner comprises of the following three modules:

■ Pattern detector module
■ Pattern aligner module
■ Run-length violation detection module

**Pattern Detector**
In XAUI mode, the Quartus II software automatically configures 10-bit K28.5 (10'b0101111100) as the word alignment pattern. After coming out of reset (rx_digitalreset), when the pattern detector detects either disparities of the K28.5 control word, it asserts the rx_patterndetect signal for one parallel clock cycle. When the pattern aligner has aligned

the incoming data to the desired word boundary, the pattern detector asserts `rx_patterndetect` signal only if the word alignment pattern is found in the current word boundary.

**Pattern Aligner**

In XAUI mode, the pattern aligner employs an automatic synchronization state machine. The Quartus II software automatically configures the synchronization state machine to indicate synchronization when the receiver receives 4 K28.5 (10'b0101111100 or 10'b1010000011) synchronization code groups without receiving an intermediate invalid code group. Once synchronized, the state machine indicates loss of synchronization when it detects 4 invalid code groups separated by less than 4 valid code groups or when it is reset.

Receiver synchronization is indicated on the `rx_syncstatus` port of each channel. A high on the `rx_syncstatus` port indicates that the lane is synchronized and a low indicates that it has fallen out of synchronization.

Table 2–30 lists the synchronization state machine parameters when configured in XAUI mode.

| Table 2–30. Synchronization State Machine Parameters in XAUI Mode | |
| --- | --- |
| Number of valid K28.5 code groups received to achieve synchronization | 4 |
| Number of errors received to lose synchronization | 4 |
| Number of continuous good code groups received to reduce the error count by 1 | 4 |

*Synchronization State Machine in XAUI Mode*

When XAUI mode is used, the synchronization and word alignment is handled automatically by a built-in state machine that adheres to either the IEEE 802.3ae or IEEE 802.3 synchronization specifications, respectively. If you specify either standard, the alignment pattern is automatically defaulted to /K28.5/ (b'0011111010).

XAUI uses an embedded clocking scheme that re-times the data that potentially can alter the code-group boundary. The boundaries of the code groups are re-aligned through a synchronization process specified in clause 48 of the IEEE P802.3ae standard, which states that synchronization is achieved upon the reception of four /K28.5/ commas.

When you specify the XAUI protocol, code-group synchronization is achieved upon the reception of four /K28.5/ commas. Each comma can be followed by any number of valid code groups. Invalid code groups are

not allowed during the synchronization stage. When code-group synchronization is achieved the optional rx_syncstatus signal is asserted.

Refer to clause 47-48 of the IEEE P802.3ae standard or "XAUI Mode" on page 2–60 for more information about the operation of the synchronization phase.

When you configure Arria GX devices to the XAUI protocol, the built in pattern detector, word aligner, and XAUI state machines adhere to the PCS synchronization specification. After all the conditions for synchronization have been met, the rx_syncstatus signal is asserted and only de-asserts if synchronization is lost.

Figure 2–41 shows the PCS synchronization state diagram specified in clause 48 of the IEEE P802.3ae.

*Figure 2–41. IEEE 802.3ae PCS Synchronization State Diagram*
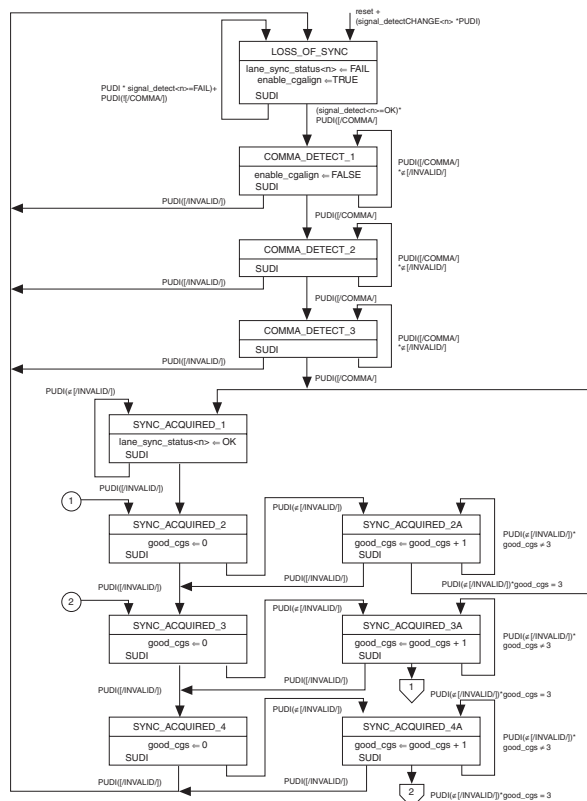
The word aligner block asserts an error flag on the `rx_disperr` and `rx_errdetect` ports if the received 10-bit code is detected with incorrect running disparity. The error flag signal (`rx_disperr`) has the same delay from the word aligner to the PLD-transceiver interface as the received data.

### Channel Aligner (Deskew)

It is possible for ordered sets to be misaligned with respect to one another because of board skew or differences between the independent clock recoveries per serial lane. Channel alignment, also referred to as deskew or channel bonding, realigns the ordered sets by using the alignment code group, referred to as /A/. The /A/ code group is transmitted simultaneously on all four lanes, constituting an ||A|| ordered set, during idles or IPG. XAUI receivers use these code groups to resolve any lane to lane skew. Skew between the lanes can be up to 40 UI (12.8 ns) as specified in the standard, which relaxes the board design constraints.

Figure 2–42 shows lane skew at the receiver input and how the deskew circuitry uses the /A/ code group to deskew the channels.

*Figure 2–42. Lane Deskew with the /A/ Code Group*
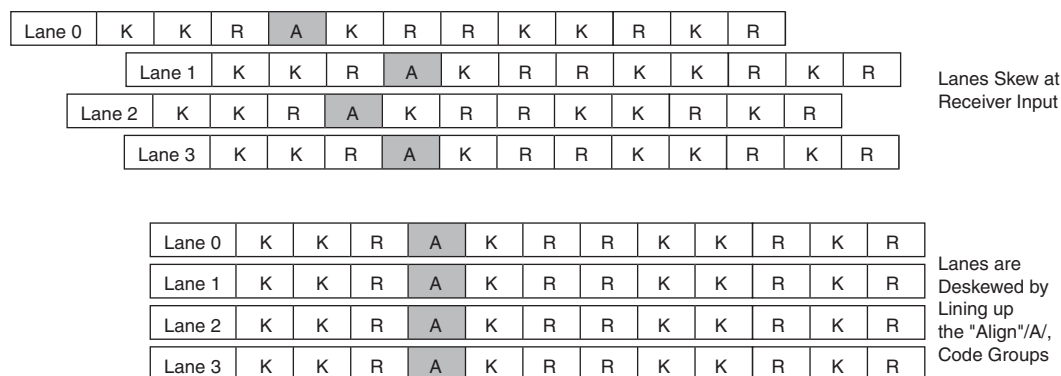


Arria GX devices manage XAUI channel alignment with a dedicated deskew macro that consists of a 16-word-deep FIFO buffer controlled by a XAUI deskew state machine. The XAUI deskew state machine first looks for the /A/ code group within each channel. When the XAUI deskew state machine detects /A/ in each channel, the deskew FIFO buffer is enabled. The deskew state machine now monitors the reception of /A/ code groups. When four aligned /A/ code groups have been received the `rx_channelaligned` is asserted. The deskew state machine continues to monitor the reception of /A/ code groups and

de-asserts the rx_channelaligned signal if alignment conditions are lost. This built-in deskew macro is only enabled for the XAUI protocol. Figure 2–43 shows the PCS deskew state diagram specified in clause 48 of the IEEE P802.3ae.

*Figure 2–43. IEEE 802.3ae PCS Deskew State Diagram*
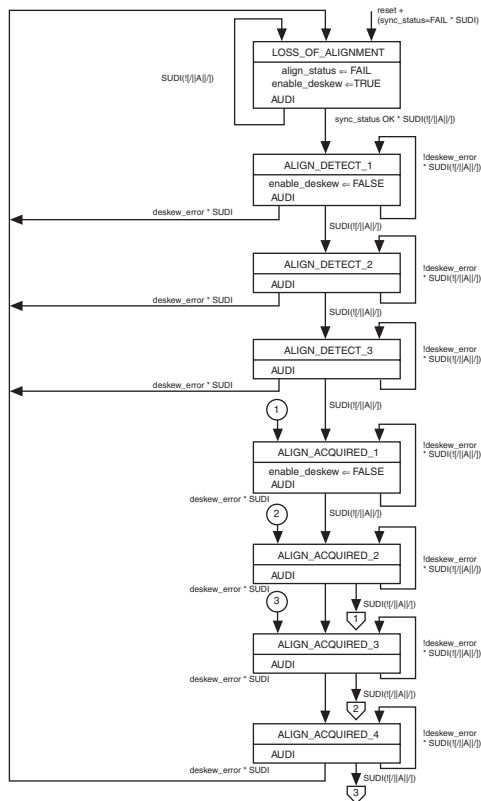


### Rate Matcher

XAUI can operate in multi-crystal environments, which can tolerate frequency variations of 100 PPM between crystals. Arria GX devices contain embedded circuitry to perform clock rate compensation, which is achieved by inserting or removing the PCS SKIP code group (/R/) from the IPG or idle stream. This process is called rate matching and is sometimes referred to as clock rate compensation.

The rate matcher in Arria GX devices consists of a 12-word-deep FIFO buffer, with control logic that you can configure to support XAUI, GIGE, or custom modes. In XAUI mode the controller begins to write data into the FIFO buffer whenever the `rx_channelaligned` signal is asserted. Within the control logic there is a FIFO counter that keeps track of the read and write executions. When the FIFO counter reaches a value of greater than nine, the receivers delete the /R/ code-group simultaneously across all channels during IPG or idle conditions. If the FIFO counter is fewer than five, the receivers insert the /R/ code-group simultaneously across all channels during IPG or idle conditions.

The rate matcher in XAUI mode operates in a synchronized four mode and supports up to a 100 PPM clock difference between the upstream transmitter and receiver. In this mode, the rate matcher can insert or delete a column of /R/ characters as denoted by the ||R|| designation, depending on whether the FIFO buffer is approaching an empty or full condition. The rate matcher does not operate until the XAUI synchronization state machine achieves word alignment and channel alignment. Until that point, the rate matcher is not active (read and write pointers do not move).

If the ||R|| code words are not received on all channels, rate matching does not occur and may lead to over/underflow conditions in the rate-matching FIFO buffer. If this situation occurs, the data output of the receiver outputs a constant 9'h19C (8'h9C on the `rx_dataout` output and 1'b1 on the `rx_ctrldetect` output) in Lane 0 (rest of the lane are data 8'h00). The receiver digital reset must be asserted and the lanes resynchronized before data can be received.

☞   This circuitry compensates for 100 PPM frequency variations.

### 8B/10B Decoder

In XAUI mode, the 8B/10B decoder clocks in 10-bit data from the word aligner and decodes it into 8-bit data + 1-bit control identifier. The 8-bit decoded data is fed to the byte deserializer.

👣   For more details about the 8B/10B decoder functionality, refer to the 8B/10B Encoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

If the received 10-bit code group is not a part of valid Dx.y or Kx.y code groups, the 8B/10B decoder block asserts an error flag on the `rx_errdetect` port. The error flag signal (`rx_errdetect`) has the same data path delay from the 8B/10B decoder to the PLD-transceiver interface as the invalid code group.

If the received 10-bit code group is detected with incorrect running disparity, the 8B/10B decoder block asserts an error flag on the `rx_disperr` and `rx_errdetect` ports. The error flag signal (`rx_disperr`) has the same delay from the 8B/10B decoder to the PLD-transceiver interface as the received data.

### PCS Code Group to XGMII Character Mapping

In XAUI mode, the 8B/10B decoder in Arria GX devices is controlled by a global receiver state machine that maps various PCS code groups into specific 8-bit XGMII codes. Table 2–31 lists the PCS code group to XGMII character mapping.

*Table 2–31. PCS Code Group to XGMII Character Mapping*

| XGMII RXC | XGMII RXD | PCS Code Group | Description |
|-----------|-----------|----------------|-------------|
| 0 | 00 through FF | Dxx.y | Normal data transmission |
| 1 | 07 | K28.0, K28.3, or K28.5 | Idle in [[I]] |
| 1 | 07 | K28.5 | Idle in [[T]] |
| 1 | 9C | K28.4 | Sequence |
| 1 | FB | K27.7 | Start |
| 1 | FD | K29.7 | Terminate |
| 1 | FE | K30.7 | Error |
| 1 | FE | Invalid code group | Received code group |

*Note to Table 2–31:*
(1)    Values in RXD column are in hexadecimal.

### Byte Deserializer

In XAUI 3.125 Gbps mode, the PLD-receiver interface data is 16 bits wide and is clocked out of the receiver phase compensation FIFO at 156.25 MHz. The byte deserializer clocks in the 8-bit wide data from the 8B/10B decoder at 312.5 MHz and clocks out 16-bit wide data to the receiver phase compensation FIFO at 156.25 MHz. This allows clocking the PLD-transceiver interface at half the speed.

For more details about byte deserializer architecture, refer to the Byte Deserializer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

In XAUI mode, the write port of the byte deserializer is clocked by the low-speed parallel recovered clock and the read port is clocked by divide-by-two version of this clock.

Due to 8- to 16-bit byte deserialization, the byte ordering at the PLD receiver interface might be incorrect. If required, you must implement the byte ordering logic in the PLD core to correct for this situation.

Figure 2–44 shows the block diagram of the byte deserializer in XAUI mode.

*Figure 2–44. Byte Deserializer in XAUI Mode*



### Receiver Phase Compensation FIFO Buffer

The receiver phase compensation FIFO buffer compensates for the phase difference between the local receiver PLD clock and the receiver PCS clock.

For more details about the receiver phase compensation FIFO buffer architecture, refer to the Receiver Phase Compensation FIFO Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

In XAUI 3.125 Gbps mode, the 312.5 MHz low-speed parallel recovered clock is divided by 2. The resulting 156.25 MHz clock is used to clock the `write` port of the FIFO buffer. This divide-by-two clock is also forwarded to the PLD logic array (on the `rx_clkout` port). If the `rx_coreclk` port is not instantiated, the recovered clock signal on the `rx_clkout` port is automatically routed back to clock the read side of the receiver phase compensation FIFO buffer. The 16-bit PLD-receiver interface clocked at 156.25 MHz results in an effective XAUI data rate of 3.125 Gbps.

In XAUI mode, the receiver phase compensation FIFO is four words deep. The latency through the FIFO is one to two PLD-transceiver interface clock cycles.

Figure 2–45 shows the block diagram of receiver phase compensation FIFO in XAUI mode.

*Figure 2–45. Receiver Phase Compensation FIFO in XAUI Mode*
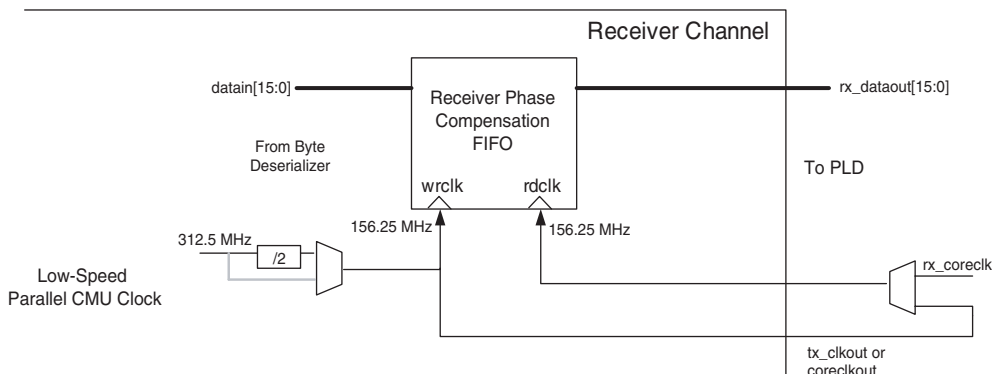


# Serial Digital Interface (SDI) Mode

The Society of Motion Picture and Television Engineers (SMPTE) defines various Serial Digital Interface (SDI) standards for transmission of uncompressed video.

The following three SMPTE standards are popular in video broadcasting applications:

- SMPTE 259M standard— more popularly known as the standard definition (SD) SDI, is defined to carry video data at 270 Mbps.
- SMPTE 292M standard— more popularly known as the high definition (HD) SDI, is defined to carry video data at either 1485 Mbps or 1483.5 Mbps.
- SMPTE 424M standard— more popularly known as the third generation (3G) SDI, is defined to carry video data at either 2970 Mbps or 2967 Mbps.

You can configure Arria GX transceivers in HD SDI or 3G SDI configuration using the ALT2GXB MegaWizard Plug-In Manager.

Figure 2–46 shows the ALT2GXB transceiver data path in SDI mode.

*Figure 2–46. SDI Mode Data Path*
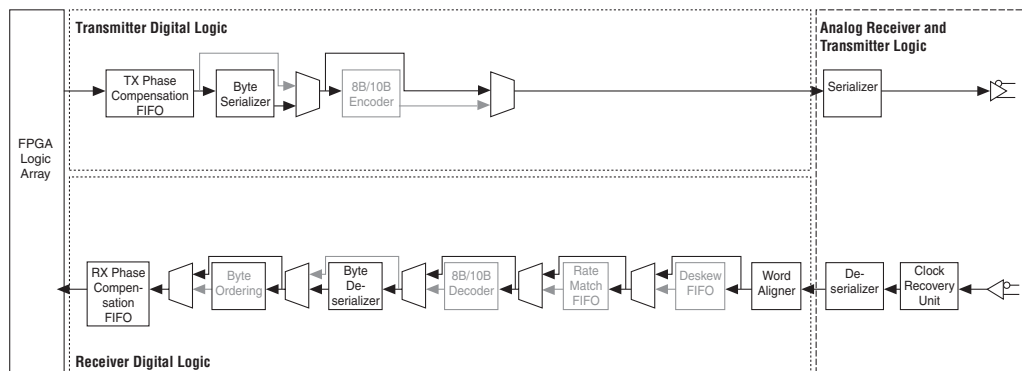


Table 2–32 shows ALT2GXB configurations supported by the Arria GX transceivers in SDI mode.

*Table 2–32. ALT2GXB Configuration in SDI Mode*

| Configuration | Data Rate (Mbps) | REFCLK Frequencies (MHz) | Channel Width |
|---|---|---|---|
| HD | 1485 | 74.25, 148.5 | 10 bit, 20 bit |
| | 1483.5 | 74.175, 148.35 | 10 bit, 20 bit |
| 3G | 2970 | 148.5, 297 | Only 20-bit interface allowed in 3G |
| | 2967 | 148.35, 296.7 | Only 20-bit interface allowed in 3G |

*Transmitter Data Path*

In the 10-bit channel width SDI configuration, the transmitter data path consists of the transmitter phase compensation FIFO and the 10:1 serializer. In the 20-bit channel width SDI configuration, the transmitter data path also includes the byte serializer.

☞ In SDI mode, the transmitter is purely a parallel-to-serial converter. SDI transmitter functions, such as scrambling and cyclic redundancy check (CRC) code generation, must be implemented in the FPGA logic array.

### Receiver Data Path

In the 10-bit channel width SDI configuration, the receiver data path is comprised of the CRU, the 1:10 deserializer, the word aligner in bit-slip mode, and the receiver phase compensation FIFO. In the 20-bit channel width SDI configuration, the receiver data path also includes the byte deserializer.

☞ SDI receiver functions, such as descrambling, framing, and CRC checker, must be implemented in the FPGA logic array.

### Receiver Word Alignment/Framing

In SDI systems, because the word alignment and framing happens after descrambling, the word aligner in the receiver data path is not useful. Altera recommends driving the ALT2GXB `rx_bitslip` signal low to avoid the word aligner from inserting bits in the received data stream.

☞ Altera offers SDI MegaCore® function that can be configured at SD-SDI, HD-SDI, and 3G-SDI data rates. The SDI MegaCore function implements system level functions such as scrambling and de-scrambling and CRC generation and checking. It also offers the capability of configuring the three SDI data rates (SD, HD, and 3G) dynamically on the same transceiver channel.

For more information about the SDI MegaCore function, refer to the *SDI MegaCore Function User Guide*.

## Reset Control and Power-Down

Arria GX transceivers provide multiple reset signals to reset the analog and digital circuits in the transceiver channels. Besides individual channel resets, Arria GX transceivers also provide power-down signals that you can assert to power-down the entire transceiver block to reduce power consumption (Figure 2–47).

*Figure 2–47. Reset Signals*
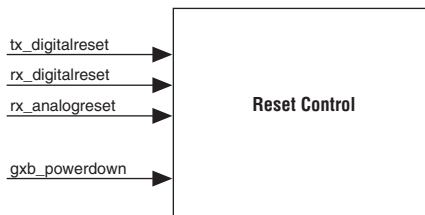
## User Reset and Power-Down Signals

Each transceiver block and each channel in the transceiver block of the Arria GX device has individual reset signals to reset the digital and analog circuits in the channel. The `tx_digitalreset`, `rx_digitalreset`, and `rx_analogreset` signals affect the channels individually. The `gxb_powerdown` signal affects the entire transceiver block.

☞      All reset and power-down signals are optional. Altera strongly recommends using the reset and power-down signals and following the reset sequence detailed in this section.

■ `tx_digitalreset`—This signal resets all digital logic in the transmitter. This signal operates independently from the other reset signals. The minimum pulse width is two parallel cycles.
  - In Basic mode, Altera recommends de-asserting the `tx_digitalreset` signal of all used transceiver blocks simultaneously after the `pll_locked` signal from all active transceiver blocks goes high.
■ `rx_digitalreset`—This signal resets all digital logic in the receiver. This signal operates independently from the other reset signals. The minimum pulse width is two parallel cycles.
■ `rx_analogreset`—This signal resets part of the analog portion of the receiver CRU. This signal operates independently from the other reset signals. The minimum pulse width is two parallel cycles.
■ `gxb_powerdown`—This signal powers down the entire transceiver block, including the transmitter PLL. All digital and analog circuits are also reset. This signal operates independently from the other reset signals. The minimum pulse width is 100 ns.

Table 2–33 lists the transceiver modules that get affected by each reset and power-down signal.

*Table 2–33. Blocks Affected by Reset and Power-Down Signals   (Part 1 of 2)*

| Transceiver Blocks | rx_digitalreset | rx_analogreset | tx_digitalreset | gxb_powerdown |
|---|:---:|:---:|:---:|:---:|
| Transmitter phase compensation FIFO buffer and byte serializer | — | — | ✓ | ✓ |
| Transmitter 8B/10B encoder | — | — | ✓ | ✓ |
| Transmitter serializer | — | — | — | ✓ |
| Transmitter analog circuits | — | — | — | ✓ |
| Transmitter PLLs | — | — | — | ✓ |
| Transmitter analog circuits | — | — | — | ✓ |
| Receiver deserializer | — | — | — | ✓ |

*Table 2–33. Blocks Affected by Reset and Power-Down Signals   (Part 2 of 2)*

| Transceiver Blocks | rx_digitalreset | rx_analogreset | tx_digitalreset | gxb_powerdown |
|---|:---:|:---:|:---:|:---:|
| Receiver word aligner | ✓ | — | — | ✓ |
| Receiver rate matcher | ✓ | — | — | ✓ |
| Receiver 8B/10B decoder | ✓ | — | — | ✓ |
| Receiver phase compensation FIFO buffer and byte deserializer | ✓ | — | — | ✓ |
| Receiver PLL and CRU | — | ✓ | — | ✓ |
| Receiver analog circuits | — | — | — | ✓ |

The recommended reset sequence varies depending on whether the CRU is configured in automatic lock mode or manual lock mode.

## Recommended Reset Sequence for GIGE and Serial RapidIO in CRU Automatic Lock Mode

Figure 2–48 shows a sample reset sequence for GIGE, Serial RapidIO, XAUI, SDI, and Basic modes when the CRU is configured in automatic lock mode.

*Figure 2–48. Reset Sequence for GIGE, Serial RapidIO, XAUI, SDI and Basic in Automatic Mode*
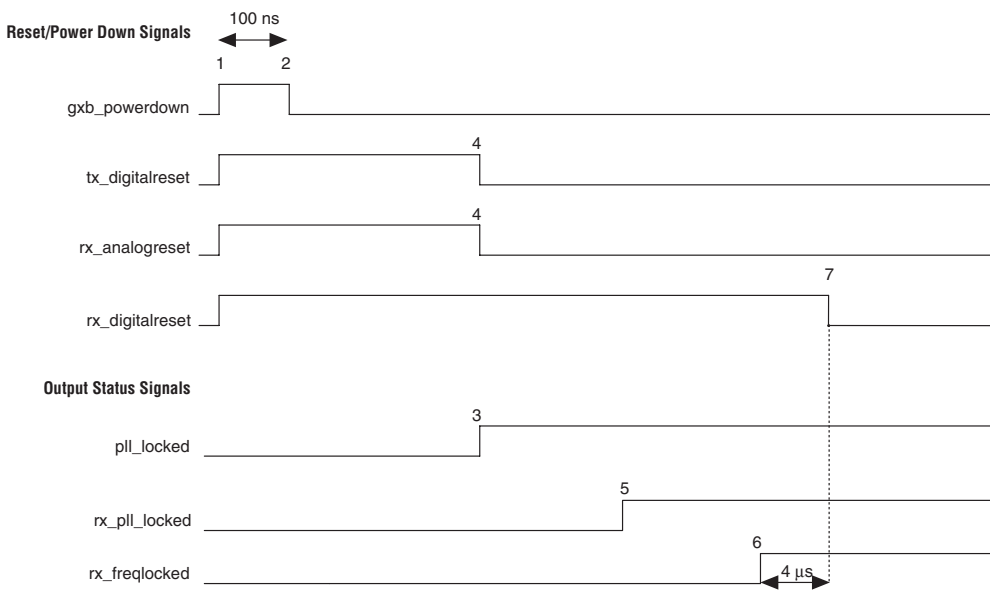
After power on, follow these steps:

1. Assert the `gxb_powerdown` port for a minimum period of 100 ns (time between markers 1 and 2).

2. Keep the `tx_digitalreset`, `rx_digitalreset`, and `rx_analogreset` asserted during this time period.

3. After you de-assert the `gxb_powerdown` signal, the transmitter PLL starts locking to the transmitter input reference clock. Once the transmitter PLL locks (as indicated by the `pll_locked` signal going high), you de-assert the `tx_digitalreset` signal.

4. After you de-assert the `rx_analogreset` signal, the receiver PLL starts locking to the receiver input reference clock (in automatic lock mode).

5. Once the receiver PLL locks to the input reference clock, the `rx_pll_locked` signal goes high. The internal PPM detector takes some time to calculate the PPM difference between the receiver PLL output clock and the input reference clock.

6. Once it calculates the PPM difference to be within the pre-defined limits, the `rx_freqlocked` signal goes high. At this point the CRU enters lock-to-data mode and the receiver PLL starts locking to the received data.

7. You de-assert the `rx_digitalreset` 4 µs after the `rx_freqlocked` signal goes high.

## Recommended Reset Sequence for GIGE, Serial RapidIO, XAUI, SDI, and Basic Modes in CRU Manual Lock Mode

Figure 2–49 shows a sample reset sequence for GIGE, Serial RapidIO, XAUI, SDI, and Basic modes when the CRU is configured in manual lock mode.

*Figure 2–49. Reset Sequence for GIGE and Serial RapidIO in Manual Mode*
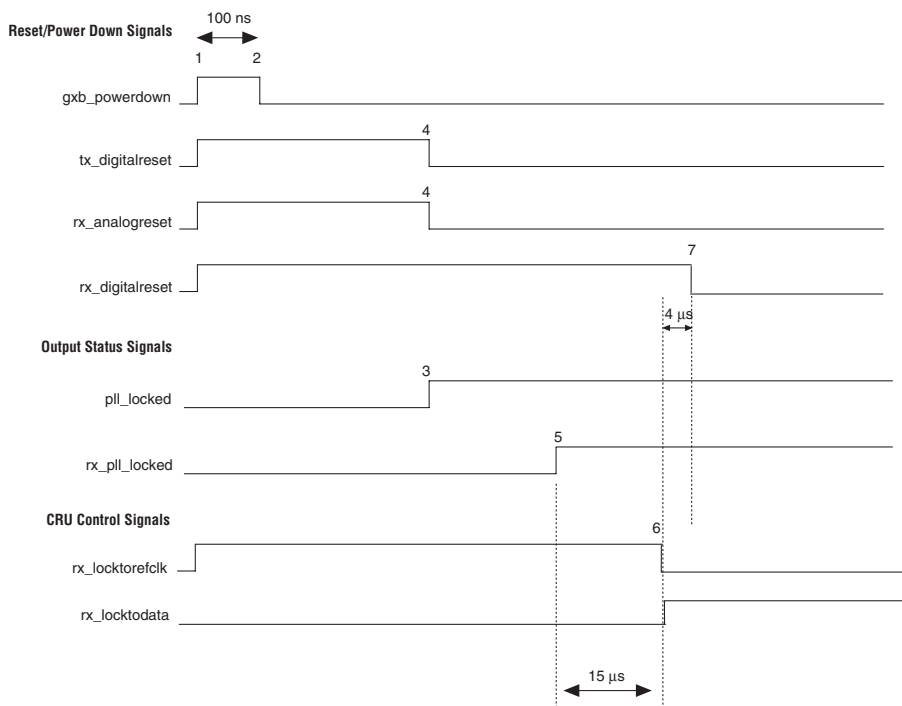


After power-on, follow these steps:

1. Assert the `gxb_powerdown` port for a minimum period of 100 ns (time between markers 1 and 2). Keep the `tx_digitalreset`, `rx_digitalreset`, `rx_analogreset`, and `rx_locktorefclk` signals asserted during this time period.

2. After you de-assert the `gxb_powerdown` signal, the transmitter PLL starts locking to the transmitter input reference clock.

3. Once the transmitter PLL locks (as indicated by the `pll_locked` signal going high), you de-assert the `tx_digitalreset` signal.

4. After you de-assert the `rx_analogreset` signal, the receiver PLL starts locking to the receiver input reference clock since `rx_locktorefclk` is asserted.

5. Wait for at least 15 µs (time between markers 5 and 6) after the `rx_pll_locked` signal goes high and then de-assert the `rx_locktorefclk` signal.

6. At the same time assert the `rx_locktodata` signal. At this point the CRU enters lock-to-data mode and the receiver PLL starts locking to the received data.

7. You de-assert the `rx_digitalreset` at least 4 µs (time between markers 6 and 7) after asserting the `rx_locktodata` signal.

### Recommended Reset Sequence for PCI Express (PIPE) Mode

In PCI Express (PIPE) mode, the `rx_freqlocked` signal does not go high during the PCI Express (PIPE) compliance testing phase because of receiving Electrical Idle. For all other modes, the reset sequence looks for the `rx_freqlocked` signal to de-assert `rx_digitalreset`.

Figure 2–50 shows the reset sequence for PCI Express (PIPE) mode.

*Figure 2–50. Reset Sequence for PCI Express (PIPE) Mode*
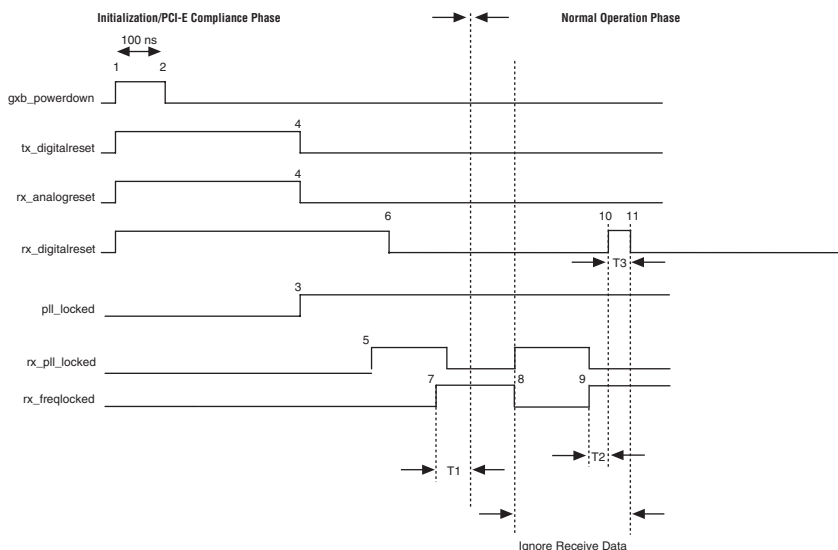
*Initialization and PCI Express Compliance Phase*

After the device is powered up, a PCI Express-compliant device may perform compliance testing. Because rx_digitalreset must be de-asserted during compliance testing, waiting for the rx_freqlocked signal to de-assert rx_digitalreset is not recommended.

De-assert the tx_digitalreset signal after the pll_locked signal goes high. De-assert the rx_digitalreset when the rx_pll_locked signal goes high (unlike GIGE and Serial RapidIO modes, where you wait until rx_freqlocked goes high).

The parallel data sent to the PLD logic array in the receive side may not be valid until 4 μs after rx_freqlocked goes high.

*Normal Operation Phase*

During normal operation, the receive data is valid and the rx_freqlocked signal is high. In this situation, when rx_freqlocked is de-asserted, (marker 8 in Figure 2–50), wait for the rx_freqlocked signal to go high again and assert rx_digitalreset (marker 10 in Figure 2–50) for two parallel receive clock cycles.

The data from the transceiver block is not valid between the time when rx_freqlocked goes low until rx_digitalreset is de-asserted. The PLD logic should ignore the data during this time period (the time period between markers 8 and 11 in Figure 2–50).

☞    Minimum T1 and T2 period is 4 μs. Minimum T3 period is two parallel receive clock cycles.

**Rate Matcher FIFO Buffer Overflow and Underflow Condition**
During the normal operation phase, monitor the overflow and underflow status of the rate matcher FIFO buffer. If there is overflow and underflow on the rate matcher FIFO buffer, assert the rx_digitalreset signal for two receive parallel clock cycles. You can monitor the rate matcher FIFO buffer status through the pipestatus[2:0] signal from the PCI Express (PIPE) interface. This condition is shown in Figure 2–51.

*Figure 2–51. PCI Express (PIPE) Mode Reset During Rate Matcher FIFO Buffer Overflow & Underflow Condition*
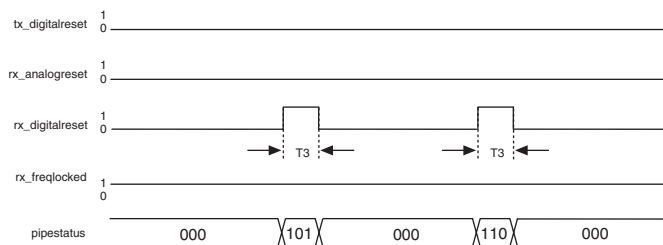


*Notes to Figure 2–51:*
(1) Pipestatus = 101 represents elastic overflow (not available in Low-Latency [Synchronous] PCI Express [PIPE] mode).
(2) Pipestatus = 110 represents elastic overflow (not available in Low-Latency [Synchronous] PCI Express [PIPE] mode).

## Power-Down

The Quartus II software automatically selects the power-down channel feature, which takes effect when you configure the Arria GX device. All unused transceiver channels and blocks in a design are powered down to reduce the overall power consumption.

☞ The gxb_powerdown port is optional. In simulation, if the gxb_powerdown port is not instantiated, you must assert the tx_digitalreset, rx_digitalreset and rx_analogreset signals appropriately for correct simulation behavior. If the gxb_powerdown port is instantiated and other reset signals are not used, you must assert the gxb_powerdown signal for at least one parallel clock cycle for correct simulation behavior. In simulation, you can de-assert the rx_digitalreset immediately after rx_freqlocked signal goes high to reduce the simulation run time. It is not necessary to wait for 4 µs (as suggested in the actual reset sequence).

☞ In PCI Express (PIPE) mode simulation, you must assert the tx_forceelecidle signal for at least one parallel clock cycle before transmitting normal data for correct simulation behavior.

## TimeQuest Timing Analyzer

Quartus II software designs targeted towards the Arria GX device family use the TimeQuest Timing Analyzer for static timing analysis. Starting with Quartus II software versions 7.1 and 7.1 sp1, the TimeQuest Timing

Analyzer does not automatically constrain the transceiver reset ports and asynchronous input/output ports. As a result, the TimeQuest Timing Analyzer does not perform timing analysis on these paths.

The TimeQuest Timing Analyzer reports these unconstrained paths in RED in the Timing Analyzer report. You must manually add the constraints in the Synopsys Design Constraints (**.sdc**) file for the TimeQuest Timing Analyzer to analyze these paths.

### Unconstrained Reset Ports

In the Quartus II software versions 7.1 and 7.1 sp1, the TimeQuest Timing Analyzer does not constrain the following transceiver reset ports:

■ `gxb_powerdown`
■ `tx_digitalreset`
■ `rx_digitalreset`
■ `rx_analogreset`

**Identifying Unconstrained Reset Ports**
To identify the unconstrained reset/powerdown ports, follow these steps:

1. After compiling your design, in the **Tools** drop-down menu, select the **TimeQuest Timing Analyzer**. This opens up the Quartus II TimeQuest Timing Analyzer window.

2. In the **Tasks** pane, execute **Report Unconstrained Paths**. This reports all unconstrained paths in RED in the **Report** pane.

3. In the **Report** pane, expand the **Unconstrained Paths** option and further expand the **Setup Analysis** or **Hold Analysis** option.

4. Under **Setup Analysis** or **Hold Analysis**, appears **Unconstrained Input Port Paths**, **Unconstrained Output Port Paths**, or both, depending on how the reset/powerdown ports are driven.

    a. If a reset/powerdown port is driven by an input pin, it is listed in the **Unconstrained Input Port Paths** report.

    b. If a reset/powerdown port is driven by synchronous logic, it is listed in the **Unconstrained Output Port Paths** report.

5. In the **Unconstrained Input Port Paths** and **Unconstrained Output Port Paths** reports, the unconstrained reset/powerdown ports of your ALT2GXB instances are listed under the **To** column.

Consider the design example in Figure 2–52.

*Figure 2–52. Example Design for TimeQuest Timing Analyzer Constraints*
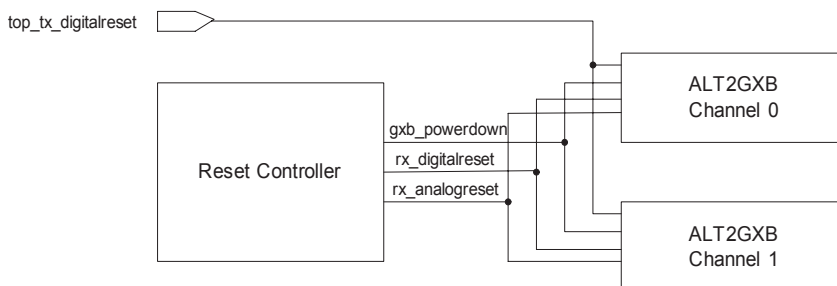


In the design example in Figure 2–52, all reset/powerdown ports for the two channels are driven by the reset controller (except the `tx_digitalreset` port). The `tx_digitalreset` port is driven from an input pin.

Figures 2–53 and 2–54 show the TimeQuest Timing Analyzer Report for **Unconstrained Input Port Paths** and **Unconstrained Output Port Paths**, respectively.

*Figure 2–53. Unconstrained Input Port Paths*

*Figure 2–54. Unconstrained Output Port Paths*
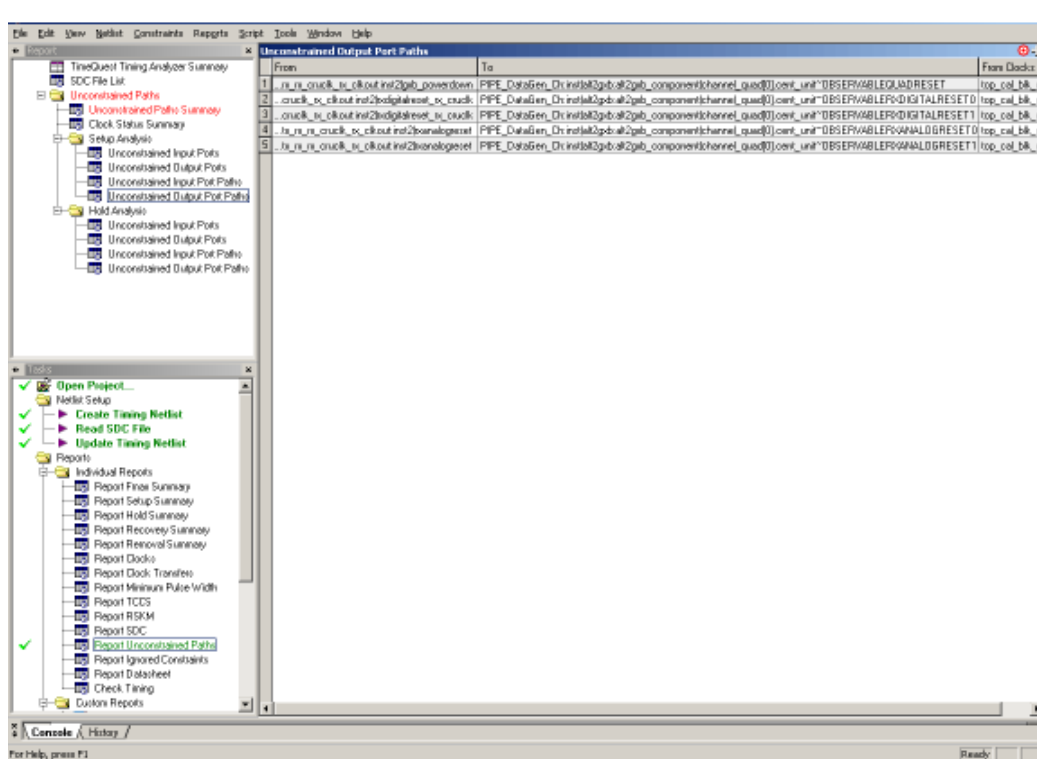


Having identified the unconstrained reset/powerdown ports in the design, the next step is to constrain these ports.

### Setting Reset/Powerdown Port Timing Constraints

You must add the reset/powerdown port timing constraints either directly in the **.sdc** file or through the TimeQuest Timing Analyzer GUI.

To add the timing constraints using the TimeQuest GUI, follow these steps:

1. In either the **Unconstrained Input Port Paths** or **Unconstrained Output Port Paths** report, locate the reset/powerdown ports.

2. In the **To** column, right-click the reset/powerdown port and select **Set Max Delay**.

3. On the resulting window, enter an initial **Delay Value** of **4** ns.

4. In the **To** column, right click on the reset/powerdown port again and select **Set Min Delay**.

5. On the resulting window, enter an initial **Delay Value** of **1.2** ns.

☞ The difference between the maximum delay and minimum delay is set to **2.8** ns, which is the maximum skew allowed on reset/powerdown ports.

6. Set the maximum and minimum delay for all transceiver reset/powerdown ports in your design, according to steps 1-5.

7. In the **Tasks** pane of the TimeQuest Timing Analyzer, double-click **Update Timing Netlist** and **Write SDC File**. Double-clicking on each of these causes them to execute.

8. Confirm that the above timing constraints were added to the **.sdc** file linked with your design.

9. Run the Quartus II Fitter.

10. After the Quartus II Fitter operation completes, in the **Tasks** pane of the TimeQuest Timing Analyzer window, double-click on **Update Timing Netlist**. The Update Timing Netlist task then executes.

11. Execute **Report Top Failing Paths** by double-clicking this option in the **Tasks** pane of the TimeQuest Timing Analyzer window.

12. Assuming all other paths in your design meet timing, one or more of the paths involving reset/powerdown ports might report timing violations. This is because the design is not able to meet the preliminary timing constraints of 4 ns (maximum delay) and 1.2 ns (minimum delay).

13. Note the slack in the timing report for all failing paths and adjust the maximum delay and the minimum delay values in the file. Maintain a difference of 2.8 ns between the maximum delay and the minimum delay for each reset/powerdown port.

14. After adjusting the delay values, execute **Update Timing Netlist** and run the Quartus II Fitter again.

15. After the Quartus II Fitter operation completes, execute **Update Timing Netlist**.

16. Execute **Report Top Failing Paths** once again. If there are any failing paths involving the reset/powerdown ports, adjust the delay values in the **.sdc** file and repeat the procedure until no failing paths are reported.

Consider the previous design example in which all unconstrained ports were identified. The following example shows how to set the constraints for the gxb_powerdown port. The same procedure must be followed for all other reset ports.

After setting the maximum and minimum delay for the gxb_powerdown port, the **.sdc** file should have the constraints detailed in Example 2–1 and Example 2–2:

*Example 2–1. Settings for Maximum Delay in the gxb_powerdown Port*

```
#****************************************************
# Set Maximum Delay
#****************************************************

set_max_delay -from  [get_keepers
{reset_seq_tx_rx_rx_cruclk_rx_clkout:inst2|gxb_powerd
own}]  -to  [get_ports
{PIPE_DataGen_Ch:inst|alt2gxb:alt2gxb_component|chann
el_quad[0].cent_unit~OBSERVABLEQUADRESET}] 4.000
```

*Example 2–2. Settings for Minimum Delay in the gxb_powerdown Port*

```
#****************************************************
# Set Minimum Delay
#****************************************************

set_min_delay -from  [get_keepers
{reset_seq_tx_rx_rx_cruclk_rx_clkout:inst2|gxb_powerd
own}]  -to  [get_ports
{PIPE_DataGen_Ch:inst|alt2gxb:alt2gxb_component|chann
el_quad[0].cent_unit~OBSERVABLEQUADRESET}] 1.200
```

After running the Quartus II fitter with the above timing constraints for the gxb_powerdown port, the following slack is reported on this path after executing **Report Top Failing Paths** (Figure 2–55).

*Figure 2–55. Slack Reported for the gxb_powerdown Port*

**SLACK: -0.798 ns (VIOLATED)**

| Path Summary | |
|---|---|
| Property ▽ | Value |
| 1 | To Node | PIPE_DataGen_Ch:inst|alt2gxb:alt2gxb_component|channel_quad[0].cent_unit~OBSERVABLEQUADRESET |
| 2 | Slack | -0.798 (VIOLATED) |
| 3 | Launch Clock | cal_blk_clk |
| 4 | Latch Clock | n/a |
| 5 | From Node | reset_seq_tx_rx_rx_cruclk_rx_clkout:inst2|gxb_powerdown |
| 6 | Data Required Time | 4.000 |
| 7 | Data Arrival Time | 4.798 |

Because the data arrival time is later than the data required time by 0.798 ns, the maximum delay and minimum delay should both be incremented by 0.8 ns in the **.sdc** file. The new **.sdc** file should have the modified constraints for the gxb_powerdown port indicated in Example 2–3 and Example 2–4.

*Example 2–3. Modified Settings for Maximum Delay for the gxb_powerdown Port*

```
#****************************************************
# Set Maximum Delay
#****************************************************

set_max_delay -from  [get_keepers
{reset_seq_tx_rx_rx_cruclk_rx_clkout:inst2|gxb_powerd
own}]  -to  [get_ports
{PIPE_DataGen_Ch:inst|alt2gxb:alt2gxb_component|chann
el_quad[0].cent_unit~OBSERVABLEQUADRESET}] 4.8
```

*Example 2–4. Modified Settings for Minimum Delay for the gxb_powerdown Port*

```
#****************************************************
# Set Minimum Delay
#****************************************************

set_min_delay -from  [get_keepers
{reset_seq_tx_rx_rx_cruclk_rx_clkout:inst2|gxb_powerd
own}]  -to  [get_ports
{PIPE_DataGen_Ch:inst|alt2gxb:alt2gxb_component|chann
el_quad[0].cent_unit~OBSERVABLEQUADRESET}] 2.000
```

After modifying the **.sdc** file and running the Quartus II Fitter, the **Update Timing Netlist** option should be executed, followed by **Report Top Failing Paths**. If the gxb_powerdown port still shows in the failing paths, modify the slack appropriately in the **.sdc** file and repeat the procedure until timing is met on this path.

Follow the same procedure to set timing constraints on all transceiver reset/powerdown ports in your design.

☞ You should set constraints and meet timing for both fast and slow timing models. The same maximum and minimum delay constraints might not be able to meet timing for both timing models. This is acceptable as long as the skew is within the specified period (2.8 ns) for each path in the **.sdc** file for each timing model.

## Unconstrained Asynchronous ALT2GXB Ports

In the Quartus II software versions 7.1 and 7.1 sp1, the TimeQuest Timing Analyzer does not automatically constrain transceiver asynchronous input/output ports. These ports are listed in Table 2–34.

*Table 2–34. TImeQuest Timing Analyzer Port Names Versus ALT2GXB Port Names*

| TimeQuest Timing Analyzer Port Name | ALT2GXB Port Name |
|---|---|
| ala2size | rx_ala2size |
| enapatternalign | rx_enapatternalign |
| bitslip | rx_bitslip |
| rlv | rx_rlv |
| invpol | rx_invpolarity |
| enabyteord | rx_enabyteord |
| pipe8b10binvpolarity | pipe8b10binvpolarity |
| revbitorderwa | rx_revbitorderwa |
| bisterr | rx_bisterr |
| bistdone | rx_bitstdone |
| phaselockloss | rx_pll_locked |
| freqlock | rx_freqlocked |
| seriallpbkben | rx_seriallpbken |

You must add the timing constraints manually in the **.sdc** file or for the TimeQuest Timing Analyzer to analyze these paths. For these asynchronous ports, you only need to set a maximum delay constraint of **10** ns in the **.sdc** file.

To identify all unconstrained ALT2GXB asynchronous ports, execute **Report Unconstrained Paths** in TimeQuest Timing Analyzer after running the Quartus II Fitter. Set a maximum delay of **10** ns for all such ports in the **.sdc** file.

For example, if the rx_invpolarity signal is driven by the signal top_rx_invpolarity on an input pin, the **.sdc** file constraint for this port should be set as shown in Example 2–5.

*Example 2–5. Constraints for the rx_invpolarity Port*

```
set_max_delay -from  [get_ports {top_rx_invpolarity}]
-to  [get_keepers
{xcvr_inst.receive~OBSERVABLEINVPOL}] 10.000
```

Follow the same procedure to constrain all asynchronous ALT2GXB ports in your design before closing timing analysis for your design.

## Referenced Document

This chapter references the following documents:

- *Arria GX ALT2GXB Megafunction User Guide*
- *Arria GX Transceiver Architecture*
- *SDI MegaCore Function User Guide*

## Document Revision History

Table 2–35 shows the revision history for this chapter.

| Table 2–35. Document Revision History | | |
|---|---|---|
| **Date and Document Version** | **Changes Made** | **Summary of Changes** |
| May 2008, v2.0 | Added "Basic Single-Width Mode", "Serial Digital Interface (SDI) Mode", "XAUI Mode" and "UNH-IOL Gigabit Ethernet Compliance" sections.<br>Updated "Serial RapidIO Mode Transmitter Architecture" section. | — |
| August 2007, v1.2 | Added the "Referenced Document" section. | — |
| | Minor text edits. | — |
| June 2007 v1.1 | Added "TimeQuest Timing Analyzer" section. | — |
| | Added GIGE information. | — |
| May 2007 v1.0 | Initial release. | — |

## Introduction

The MegaWizard® Plug-In Manager in the Quartus® II software creates or modifies design files that contain custom megafunction variations that can then be instantiated in a design file. The MegaWizard Plug-In Manager provides a wizard that allows you to specify options for the ALT2GXB megafunction. You can use the wizard to set the ALT2GXB megafunction features in the design.

Start the MegaWizard Plug-In Manager using one of the following methods:

- Choose the **MegaWizard Plug-In Manager** command (Tools menu).
- In the Block Editor, click **MegaWizard Plug-In Manager** in the **Symbol** dialog box (Edit menu).
- Start the stand-alone version of the MegaWizard Plug-In Manager by typing the following command at the command prompt: `qmegawiz`.

The ALT2GXB MegaWizard Plug-In Manager allows you to configure one or more transceiver channels.

This chapter contains the following sections:

Figure 3–1 shows the first page of the MegaWizard Plug-In Manager. To generate an ALT2GXB custom megafunction variation, select **Create a new custom megafunction variation** and click **Next**.

*Figure 3–1. MegaWizard Plug-In Manager (Page 1)*
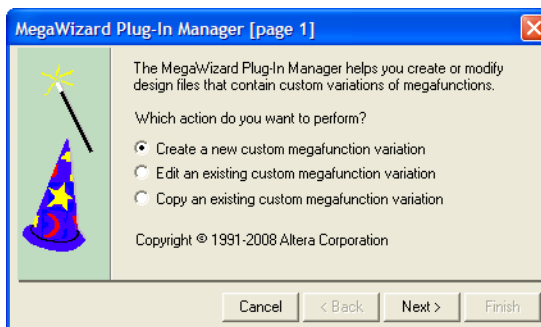


Figure 3–2 shows the second page of the MegaWizard Plug-In Manager. Select **Arria GX** as the device family.

*Figure 3–2. MegaWizard Plug-In Manager (Page 2)*
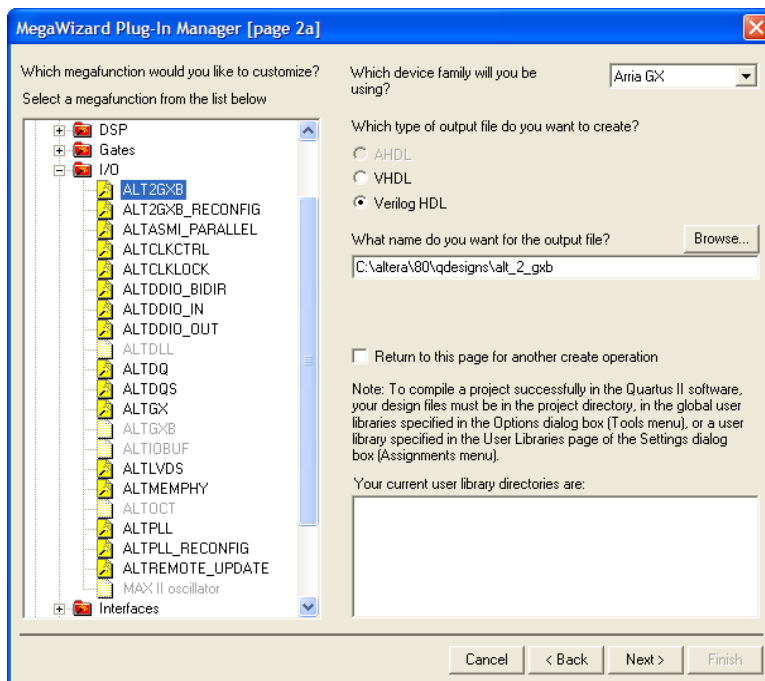
# Basic Mode

This section provides descriptions of the options available on the individual pages of the ALT2GXB MegaWizard Plug-In Manager for Basic mode. The MegaWizard Plug-In Manager provides a warning if any of the settings you choose are illegal.

Figure 3–3 shows page 3 of the ALT2GXB MegaWizard Plug-In Manager in Basic mode.

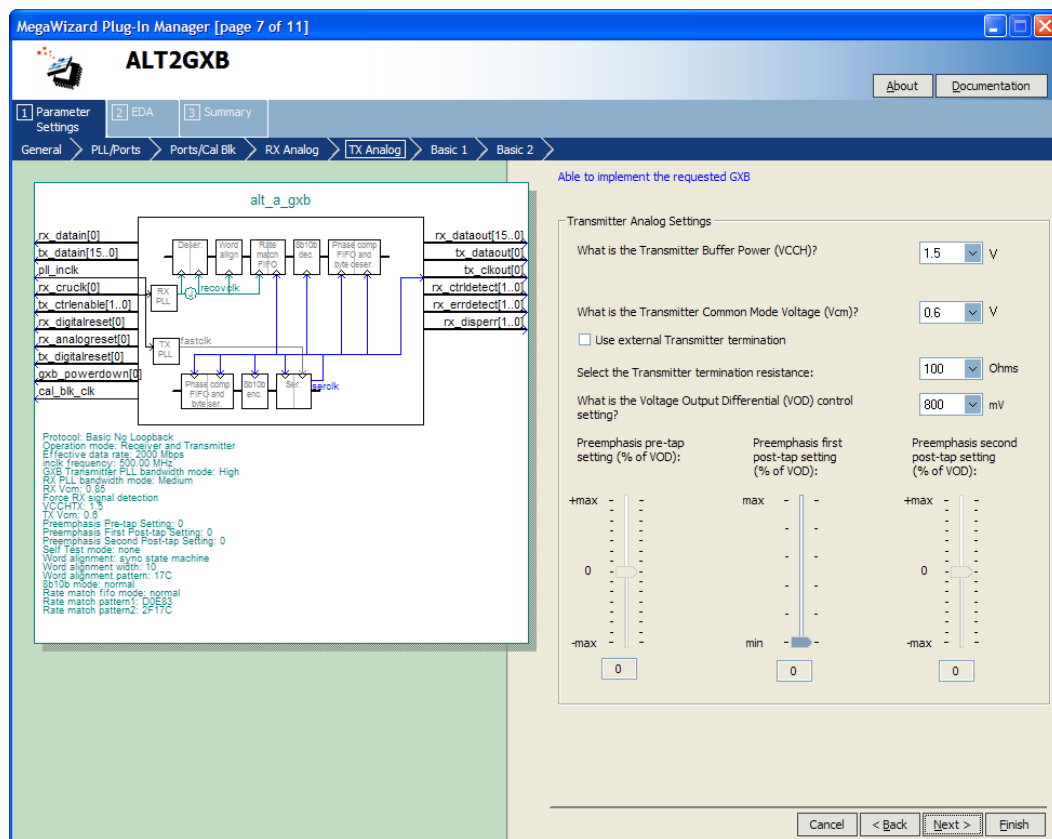*Figure 3–3. MegaWizard Plug-In Manager - ALT2GXB (General)*
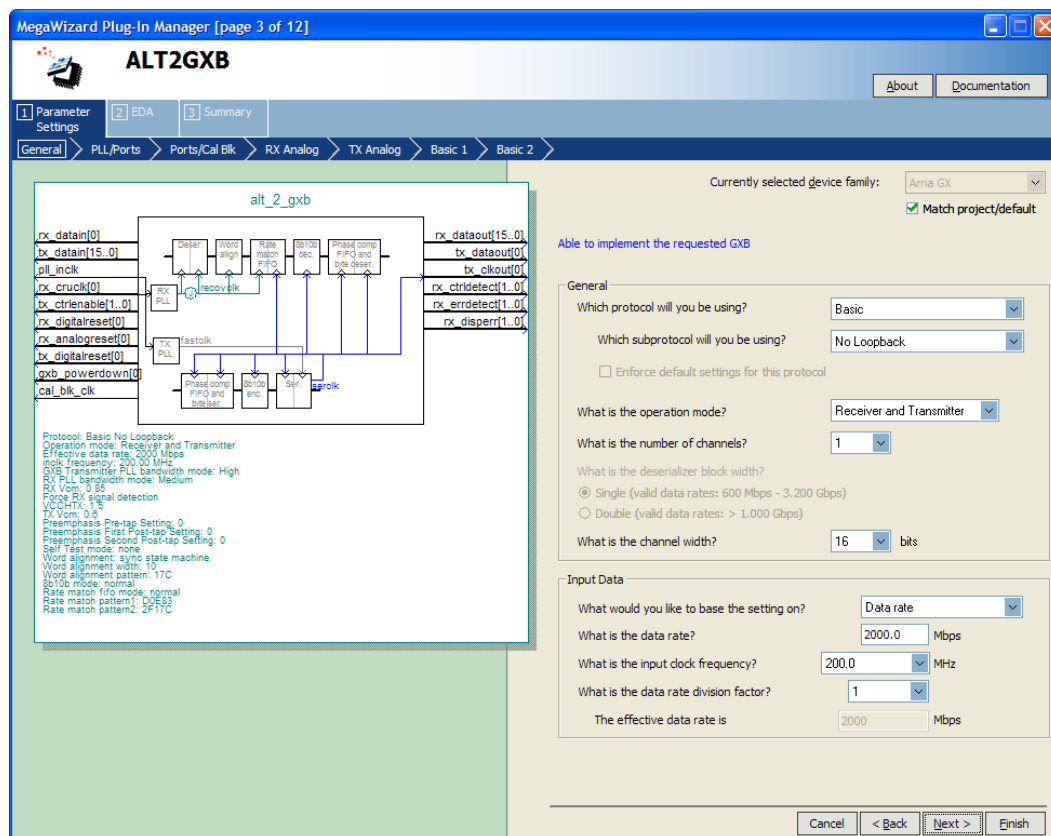
Table 3–1 describes the available options on page 3 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

| Table 3–1. MegaWizard Plug-In Manager Options (Page 3 for Basic Mode)   (Part 1 of 2) | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Which protocol will you be using? | Determines the specific protocol or modes under which the transceiver operates. For Basic mode, you must select the **Basic** protocol. | |
| Which subprotocol will you be using? | In Basic mode, the subprotocols are the diagnostic modes. The available options are as follows:<br>● No loopback – This is the normal operation of the transceiver.<br>● Serial loopback – This mode loops the user data from the transmitter path back to the receiver path right before the buffers. Serial loopback can be controlled dynamically.<br>● Reverse serial loopback – This is a loopback after the receiver's CDR block to the transmitter buffer. The RX path in the PCS is active but the TX side is not.<br>● Reverse serial loopback (pre-CDR) – This is the loopback before the receiver's CDR block to the transmitter buffer. The RX path in the PCS is active but the TX side is not.<br>● PRBS/Serial loopback – This is another serial loopback mode, but with the PRBS BIST block active. The PRBS pattern depends on the serializer/deserializer (SERDES) factor.<br>● ×4 – This mode can be used to implement the SFI-5 interface. In this mode, all four channels within the transceiver block are clocked from its central clock divider block to minimize transmitter channel-to-channel skew. | Loopback Modes and Built-In Self-Test Modes sections in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| Enforce default settings for this protocol | This selection is not active in Basic mode because there is no pre-defined protocol. | |
| What is the operation mode? | The available operation modes are receiver only, transmitter only, and receiver and transmitter. | |
| What is the number of channels? | This option determines how many duplicate channels this ALT2GXB instance contains. | |
| What is the deserializer block width? | This option sets the transceiver data path width and defaults to single width mode.<br><br>Single width—In this mode, the transceiver operates between 600 Mbps to 3.125 Gbps. | |

| Table 3–1. MegaWizard Plug-In Manager Options (Page 3 for Basic Mode) (Part 2 of 2) | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| What is the channel width? | This option determines the transceiver-to-PLD interface width.<br><br>In single-width mode, selecting 8 or 10 bits bypasses the byte serializer/deserializer. If you select 16 or 20 bits, the byte serializer/deserializer is used. | Byte Serializer and Deserializer sections in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| What would you like to base the setting on? | This option allows you to do one of following:<br>● Enter a data rate and select an input clock frequency through a pull-down menu (with the data rate selection).<br>● Enter your input clock frequency through a pull-down menu (with the data rate selection) or enter your input clock frequency and select from the available data rates for a clock frequency. | |
| What is the data rate? | Determines the TX and RX PLL VCO frequency. | |
| What is the input clock frequency? | Determines the input clock frequency you want as a reference clock for the transceiver. | |
| What is the data rate division factor? | This setting, in conjunction with the selected data rate, determines the effective data rate for the transceiver channel. Division factors of 1, 2, and 4 are available. For example, a data rate setting of 3000 Mbps and at data rate division factor of 2 yields an effective data rate of 1500 Mbps. | |

Figure 3–4 shows page 4 of the ALT2GXB MegaWizard Plug-In Manager for Basic mode.
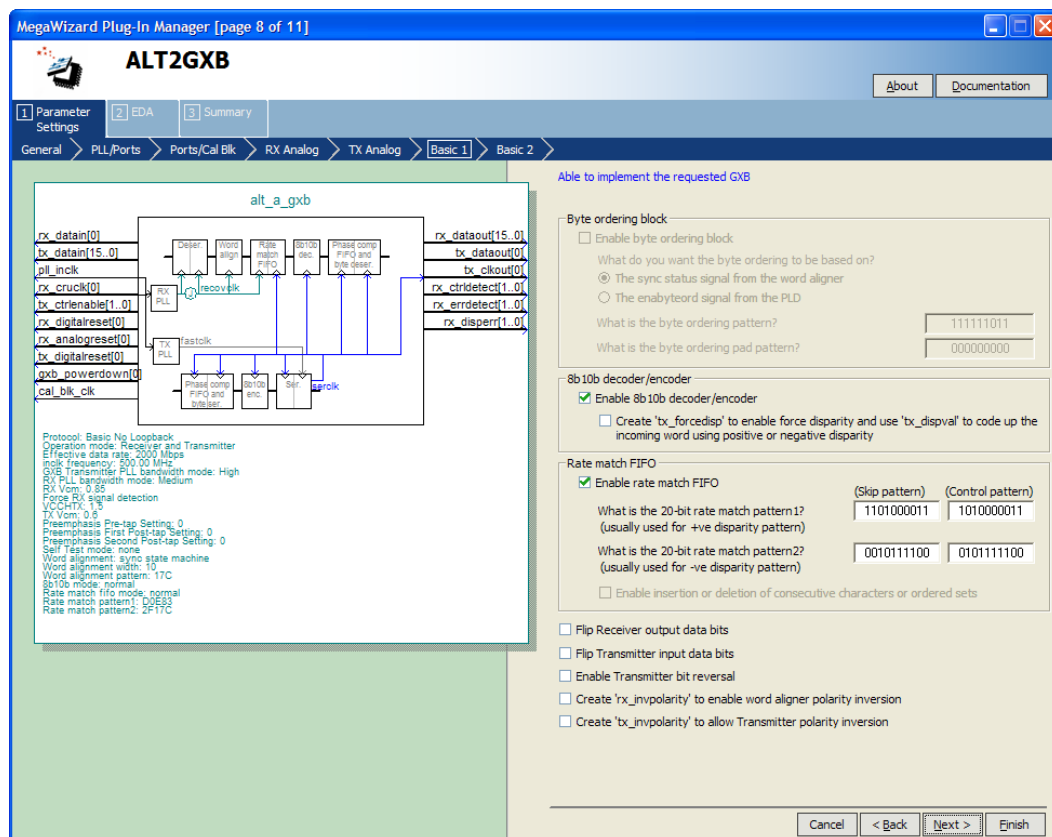
*Figure 3–4. MegaWizard Plug-In Manager - ALT2GXB (PLL/Ports)*

Table 3–2 describes the available options on page 4 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| Train Receiver PLL clock from PLL inclk | If you turn this option on, your design uses the input reference clock to the transmitter PLL to train the receiver PLL. This reduces the need to supply a separate receiver PLL reference clock. | Clock Recovery Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What is the GXB Transmitter PLL bandwidth mode? | This option selects the transmitter PLL bandwidth and the allowed options are low, medium and high. | Clock Multiplier Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What is the Receiver PLL bandwidth mode? | This option selects the receiver PLL bandwidth and the allowed options are low, medium and high. | Clock Recovery Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What is the acceptable PPM threshold between the Receiver PLL VCO and the CRU clock? | This option determines the parts per million (PPM) difference that affects the automatic receiver clock recovery unit (CRU) switchover between lock-to-data and lock-to-reference. There are additional factors that affect the CRU's transition. | Clock Recovery Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create gxb_powerdown port to power down the Quad | This signal can be used to reset and power down all circuits in the transceiver block. It does not power down the REFCLK buffers and reference clock lines. | Reset Control and Power Down section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create gxb_enable port to enable the Quad | This signal can be used to enable Arria GX transceiver blocks. If instantiated, this port must be tied to the dedicated gigabit transceiver block enable input pin. | Reset Control and Power Down section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create rx_analogreset port | Receiver analog reset port. | Reset Control and Power Down" section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |

*Table 3–2. MegaWizard Plug-In Manager Options (Page 4 for Basic Mode)   (Part 1 of 2)*

*Table 3–2. MegaWizard Plug-In Manager Options (Page 4 for Basic Mode)   (Part 2 of 2)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| Create `rx_digitalreset` port | Receiver digital reset port. Resets the PCS portion of the receiver. Altera® recommends using this port along with logic to implement the recommended reset sequence. | Reset Control and Power Down section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| Create `tx_digitalreset` port | Transmitter digital reset port. Resets the PCS portion of the transmitter. Altera recommends using this port along with logic to implement the recommended reset sequence. | Reset Control and Power Down section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| Create `pll_locked` port to indicate PLL is in lock with the reference input clock | PLL locked indicator for the transmitter PLLs. | Clock Multiplier Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| Create `rx_locktorefclk` port to lock the RX PLL to the reference clock | Lock-to-reference lock mode for the CRU. Use with `rx_locktodata`. `rx_locktodata/rx_locktorefclk` 0/0—CRU is in automatic mode 0/1—CRU is in lock-to-reference clock 1/0—CRU is in lock-to-data mode 1/1—CRU is in lock-to-data mode | Clock Recovery Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| Create `rx_locktodata` port to lock the RX PLL to the received data | Lock-to-data control for the CRU. Use with `rx_locktorefclk`. | Clock Recovery Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| Create `rx_pll_locked` port to indicate RX PLL is in lock with the reference clock | Receiver PLL locked signal. Indicates if the receiver PLL is phase locked to the CRU reference clock. | Clock Recovery Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| Create `rx_freqlocked` port to indicate RX PLL is in lock with the received data | CRU mode indicator port. Indicates if the CRU is locked to data mode or locked to the reference clock mode. 0—Receiver CRU is in lock-to-reference clock mode 1—Receiver CRU is in lock-to-data mode | Clock Recovery Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |

Figure 3–5 shows page 5 of the ALT2GXB MegaWizard Plug-In Manager for Basic mode.

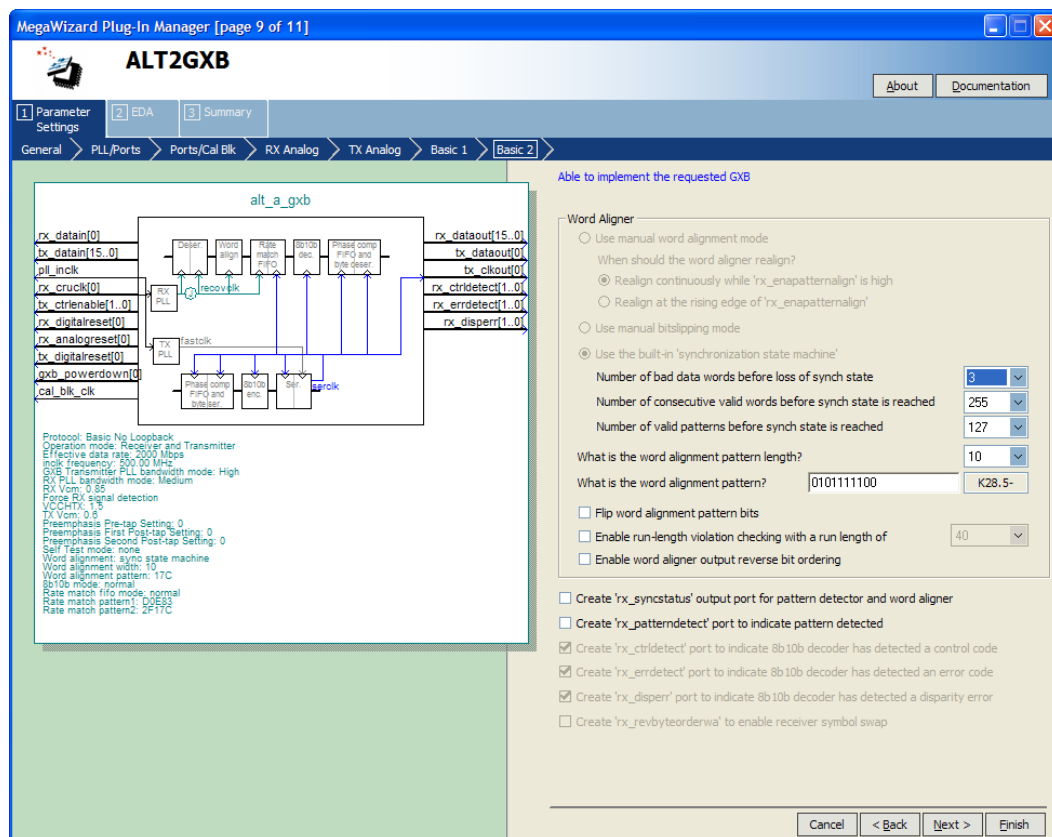*Figure 3–5. MegaWizard Plug-In Manager - ALT2GXB (Ports/Cal Blk)*

Table 3–3 describes the available options on page 5 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

| *Table 3–3. MegaWizard Plug-In Manager Options (Page 5 for Basic Mode)   (Part 1 of 2)* | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Create `rx_signaldetect` port to indicated data input signal detection | Signal detect port. In PCI Express (PIPE) mode, indicates if a signal that meets the specified range is present at the input of the receiver buffer. In all other modes, `rx_signaldetect` is forced high and must not be used as an indication of a valid signal at receiver input. | Receiver Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| Create `debug_rx_phase_comp _fifo_error` output port | This optional output port indicates Receiver Phase Compensation FIFO overflow/under run condition. Note that no PPM difference is allowed between FIFO read and write clocks. Use this port for debug purposes only. | Receiver Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| Create `debug_tx_phase_comp _fifo_error` output port | This optional output port indicates Transmitter Phase Compensation FIFO overflow/under run condition. Note that no PPM difference is allowed between FIFO read and write clocks. Use this port for debug purposes only. | Transmitter Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| Create `rx_coreclk` port to connect to the read clock of the RX phase compensation FIFO | This optional input port allows you to clock the read side of the Receiver Phase Compensation FIFO with a non-transceiver PLD clock. | Transceiver Clocking section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| Create `tx_coreclk` port to connect to the write clock of the TX phase compensation FIFO | This optional input port allows you to clock the write side of the Transmitter Phase Compensation FIFO with a non-transceiver PLD clock. | Transceiver Clocking section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |

| Table 3–3. MegaWizard Plug-In Manager Options (Page 5 for Basic Mode) (Part 2 of 2) | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Use calibration block | This option allows you to select which instance of the ALT2GXB megafunction instantiates the calibration block. Only one instance of the ALT2GXB megafunction is required to instantiate the calibration block. | Calibration Blocks section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| Create active low `cal_blk_powerdown` to power down the calibration block | Power-down signal for the calibration block. Assertion of this signal may interrupt data transmission and reception. Use this signal to re-calibrate the termination resistors if temperature and/or voltage changes warrant it. | Calibration Blocks section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |

Figure 3–6 shows page 6 of the ALT2GXB MegaWizard Plug-In Manager for Basic mode.

*Figure 3–6. MegaWizard Plug-In Manager - ALT2GXB (RX Analog)*
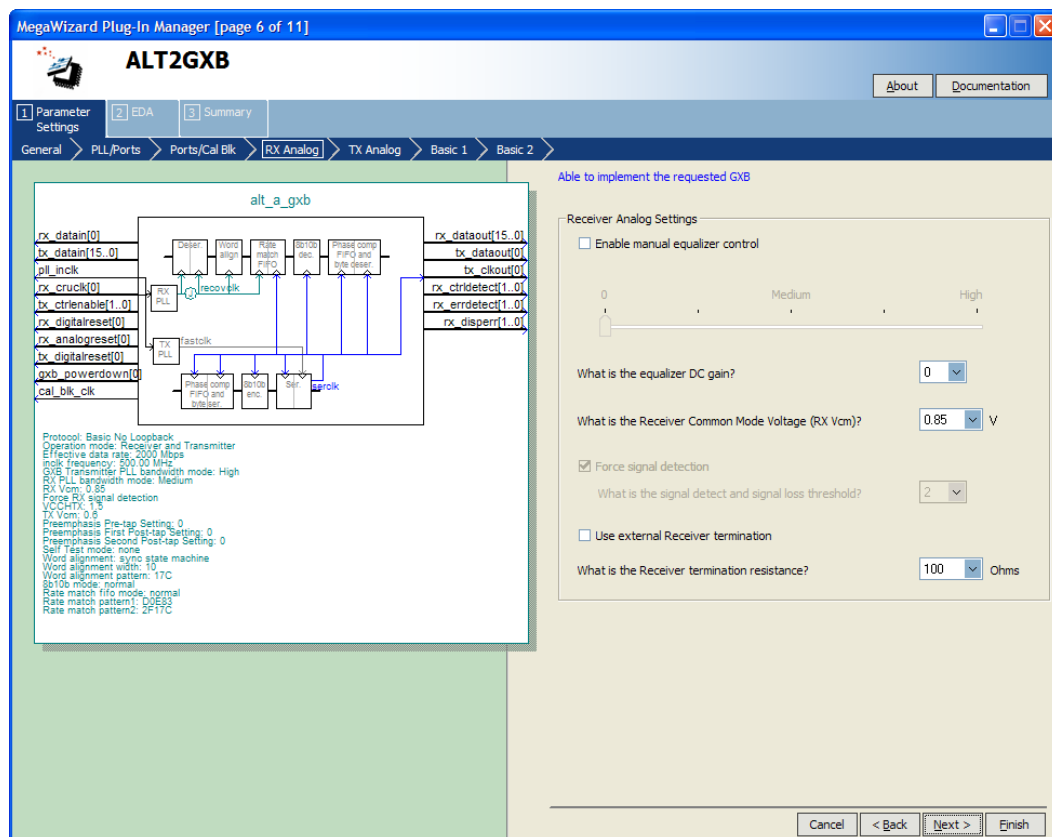
Table 3–4 describes the available options on page 6 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

| Table 3–4. MegaWizard Plug-In Manager Options (Page 6 for Basic Mode)   (Part 1 of 2) | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Enable manual equalizer control | This option enables the 0–4 setting options for manual equalizer control. | Receiver Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| What is the equalizer DC gain? | This enables the DC gain option and the legal settings are 0, 1, 2, and 3. | Receiver Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| What is the Receiver Common Mode Voltage (RX $V_{CM}$)? | The receiver common mode voltage is programmable. The selections available are 0.85 V and 1.2 V. | Receiver Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| Force signal detection | This option is not available in Basic mode. | Receiver Buffer Section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| What is the signal detect and signal loss threshold? | This option is not available in Basic mode. | Receiver Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |

*Table 3–4. MegaWizard Plug-In Manager Options (Page 6 for Basic Mode)   (Part 2 of 2)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| Use external receiver termination | This option is available if you use an external termination resistor instead of the on-chip termination (OCT). If checked, this option turns off the receiver OCT. | Receiver Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| What is the receiver termination resistance? | This option selects the receiver termination value. The only supported receiver termination resistance value is 100 Ω. | Receiver Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |

Figure 3–7 shows page 7 of the ALT2GXB MegaWizard Plug-In Manager for Basic mode.

*Figure 3–7. MegaWizard Plug-In Manager - ALT2GXB (TX Analog)*

Table 3–5 describes the available options on page 7 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

*Table 3–5. MegaWizard Plug-In Manager Options (Page 7 for Basic Mode)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| What is the Transmitter Buffer Power (VCCH)? | This setting is for information only and is used to calculate the $V_{OD}$ from the buffer power supply ($V_{CCH}$) and the transmitter termination to derive the proper $V_{OD}$ range. In Basic mode, this option is fixed at 1.5 V | Transmitter Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What is the Transmitter Common Mode Voltage ($V_{CM}$)? | The transmitter common mode voltage setting is selectable between 0.6 V and 0.7 V. | Transmitter Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Use external Transmitter termination | This option is available if you use an external termination resistor instead of the OCT. Checking this option turns off the transmitter OCT. | Transmitter Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Select the Transmitter termination resistance | This option selects the transmitter termination value. This option defaults to 100 Ω for Arria GX devices. | Transmitter Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What is the Voltage Output Differential (VOD) control setting? | This option selects the $V_{OD}$ of the transmitter buffer. The differential output voltage is programmable between 400 mV and 1200 mV in steps of 200 mV. The available $V_{OD}$ settings change based on $V_{CCH}$. | Transmitter Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Pre-emphasis pre-tap setting (% of VOD) | This option is not available in Arria GX devices and is fixed at 0. | |
| Pre-emphasis first post-tap setting (% of VOD) | This option sets the amount of pre-emphasis on the transmitter buffer using first post-tap. The options available are 0, 1, 2, 3, 4, and 5. | Transmitter Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Pre-emphasis second post-tap setting (% of VOD) | This option is not available in Arria GX devices and is fixed at 0. | |

Figure 3–8 shows page 8 of the MegaWizard Plug-In Manager for Basic protocol mode set up.

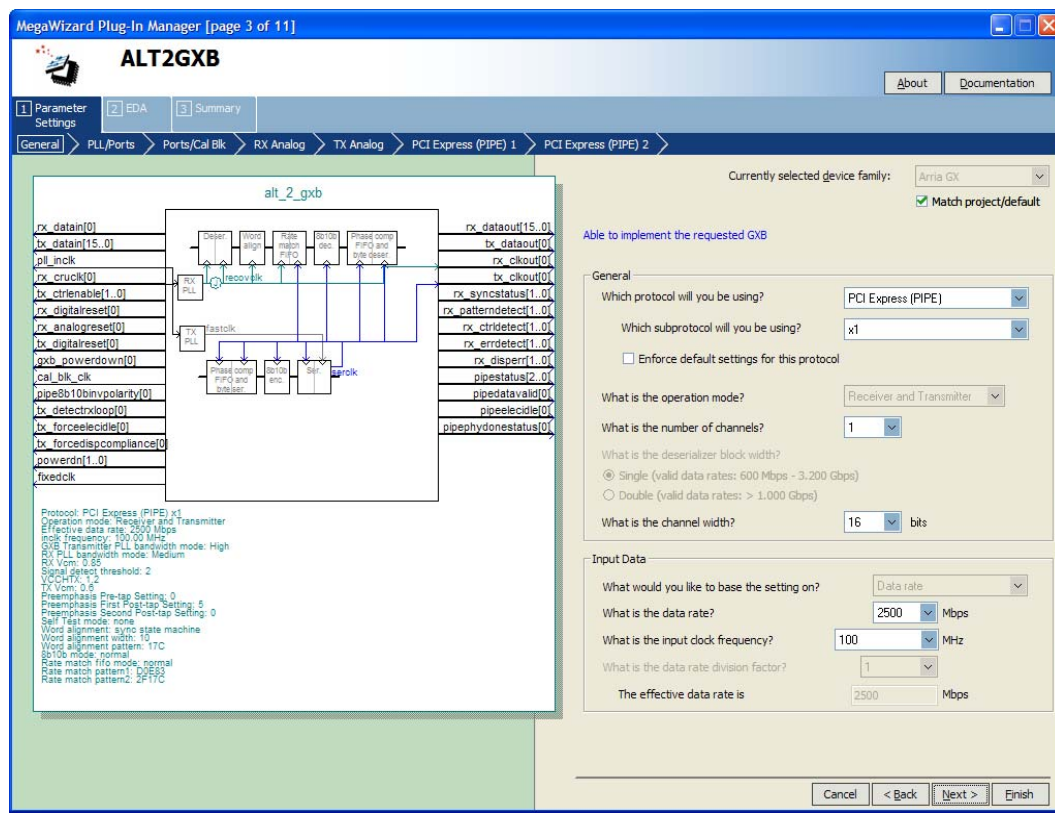*Figure 3–8. MegaWizard Plug-In Manager - ALT2GXB (Basic 1)*



Table 3–6 describes the available options on page 8 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

*Table 3–6. MegaWizard Plug-In Manager Options (Page 8 for Basic Mode)   (Part 1 of 3)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| Enable byte ordering block | This option is not available in Arria GX devices. | — |
| What do you want the byte ordering to be based on? | This option is not available in Arria GX devices. | — |

*Table 3–6. MegaWizard Plug-In Manager Options (Page 8 for Basic Mode)   (Part 2 of 3)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| What is the byte ordering pattern? | This option is not available in Arria GX devices. | — |
| What is the byte ordering pad pattern? | This option is not available in Arria GX devices. | — |
| Enable 8B/10B decoder/encoder | This option enables the 8B/10B encoder and decoder. This option is only available if the channel width is 8 or 16 bits. | — |
| Create `tx_forcedisp` to enable Force disparity and use `tx_dispval` to code up the incoming word using positive or negative disparity | This option allows you to force positive or negative disparity on transmitted data in 8B/10B configurations. | 8B/10B Encoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Enable rate match FIFO | This option enables the rate matcher and is only available with the 8B/10B decoder. | Rate Matcher section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What is the 20-bit rate match pattern1? (usually used for +ve disparity pattern) | Enter the positive disparity rate matcher pattern and control code pattern. The skip pattern is used for insertion or deletion. The control pattern identifies which group of skip patterns to use for rate matching. If only one disparity is needed for rate matching, enter the same pattern for both rate matching patterns (pattern1 and pattern2). | Rate Matcher section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What is the 20-bit rate match pattern2? (usually used for -ve disparity pattern) | Enter the negative disparity rate matcher pattern and control code pattern. The skip pattern is used for insertion or deletion. The control pattern identifies which group of skip patterns to use for rate matching. If only one disparity is needed for rate matching, enter the same pattern for both rate matching patterns (pattern1 and pattern2). | Rate Matcher section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Flip Receiver output data bits | This option reverses the bit order of the data at the receiver-PLD interface at a byte level to support MSBit-to-LSBit transmission protocols. The default transmission order is LSBit-to-MSBit. | |
| Flip Transmitter input data bits | This option reverses the bit order of the data bits at the input of the transmitter at a byte level to support MSBit-to-LSBit transmission protocols. The default transmission order is LSBit-to-MSBit. | |

| Table 3–6. MegaWizard Plug-In Manager Options (Page 8 for Basic Mode)   (Part 3 of 3) | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Enable Transmitter bit reversal | This option inverts (flips) the bit order of the data bits at the transmitter PCS-PMA interface at a byte level to support MSBit-to-LSBit transmission protocols. The default transmission is LSBit-to-MSBit. | 8B/10B encoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `rx_invpolarity` to enable word aligner polarity inversion | This optional port allows you to dynamically reverse the polarity of the received data at the input of the word aligner. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `tx_invpolarity` to allow Transmitter polarity inversion | This optional port allows you to dynamically reverse the polarity of the data to be transmitted at the transmitter PCS-PMA interface. | 8B/10B Encoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |

Figure 3–9 shows page 9 of the MegaWizard Plug-In Manager for Basic protocol mode set up.

*Figure 3–9. MegaWizard Plug-In Manager - ALT2GXB (Basic 2)*

Table 3–7 describes the available options on page 9 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

| *Table 3–7. MegaWizard Plug-In Manager Options (Page 9 for Basic Mode)   (Part 1 of 3)* | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Use manual word alignment mode | This option sets the word aligner in manual alignment mode. (Manual alignment, bit-slipping, and built-in state machine are mutually exclusive options.) | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| When should the word aligner realign? | This option sets the behavior of the `rx_enapatternalign` signal to either edge or level sensitive. Altera recommends using edge sensitive for scrambled data (non-8B/10B) traffic and level sensitive for 8B/10B traffic. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Use manual bit slipping mode | This option sets the word aligner to use the bit-slip port to alter the byte boundary one bit at a time. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Use the built-in 'synchronization state machine' | This option sets the word aligner to use the built-in synchronization state machine. The behavior is similar to the PIPE synchronization state machine with adjustable synchronization thresholds. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Number of bad data words before loss of synch state | Use this option with the built-in state machine to transition from a synchronized state to an unsynchronized state. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Number of consecutive valid words before synch state is reached | This option sets the word aligner to check for a given number of good code groups. Use this option with the built-in state machine in conjunction with the **Number of valid patterns before synchronization state is reached** option to achieve synchronization. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Number of valid patterns before synch state is reached | This option checks for the number of valid alignment patterns seen. Use this option with the built-in state machine in conjunction with the **Number of consecutive valid words before synch state is reached** option to achieve synchronization. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |

**Table 3–7. MegaWizard Plug-In Manager Options (Page 9 for Basic Mode)   (Part 2 of 3)**

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| What is the word alignment pattern length? | This option sets the word alignment length. The available choices depend on whether 8B/10B is used and which word alignment mode is used. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What is the word alignment pattern? | Enter the word alignment pattern here. The length of the alignment pattern is based on the word alignment pattern length. In bit-slip mode, this option triggers `rx_patterndetect`. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Flip word alignment pattern bits | This option reverses the bit order of the alignment pattern at a byte level to support MSB-to-LSB transmission protocols. The default transmission order is LSB-to-MSB. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Enable run-length violation checking with a run length of | This option activates the run-length violation circuit. You can program the run length at which the circuit triggers the `rx_rlv` signal. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Enable word aligner output reverse bit ordering | In manual bit-slip mode, this option creates an input port `rx_revbitorderwa` to dynamically reverse the bit order at the output of the receiver word aligner. In Basic mode, this option statically configures the receiver to always reverse the bit order of the data at the output of the word aligner. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `rx_syncstatus` output port for pattern detector and word aligner | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `rx_patterndetect` port to indicate pattern detected | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `rx_ctrldetect` port to indicate 8B/10B decoder has detected a control code | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | 8B/10B Decoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |

| *Table 3–7. MegaWizard Plug-In Manager Options (Page 9 for Basic Mode)   (Part 3 of 3)* | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Create `rx_errdetect` port to indicate 8B/10B decoder has detected an error code | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | 8B/10B Decoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `rx_disperr` port to indicate 8B/10B decoder has detected a disparity code | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | 8B/10B Decoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `rx_revbyteorderwa` to enable receiver symbol swap | This option is not available for Arria GX devices. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |

Figure 3–10 shows page 10 of the MegaWizard Plug-In Manager for Basic protocol mode set up. The Generate simulation model creates a behavioral model (**.vo** or **.vho**) of the transceiver instance for third-party simulators. The **Generate Netlist** option generates a netlist for the third party EDA synthesis tool to estimate timing and resource utilization for the ALT2GXB instance.

*Figure 3–10. MegaWizard Plug-In Manager - ALT2GXB (EDA)*
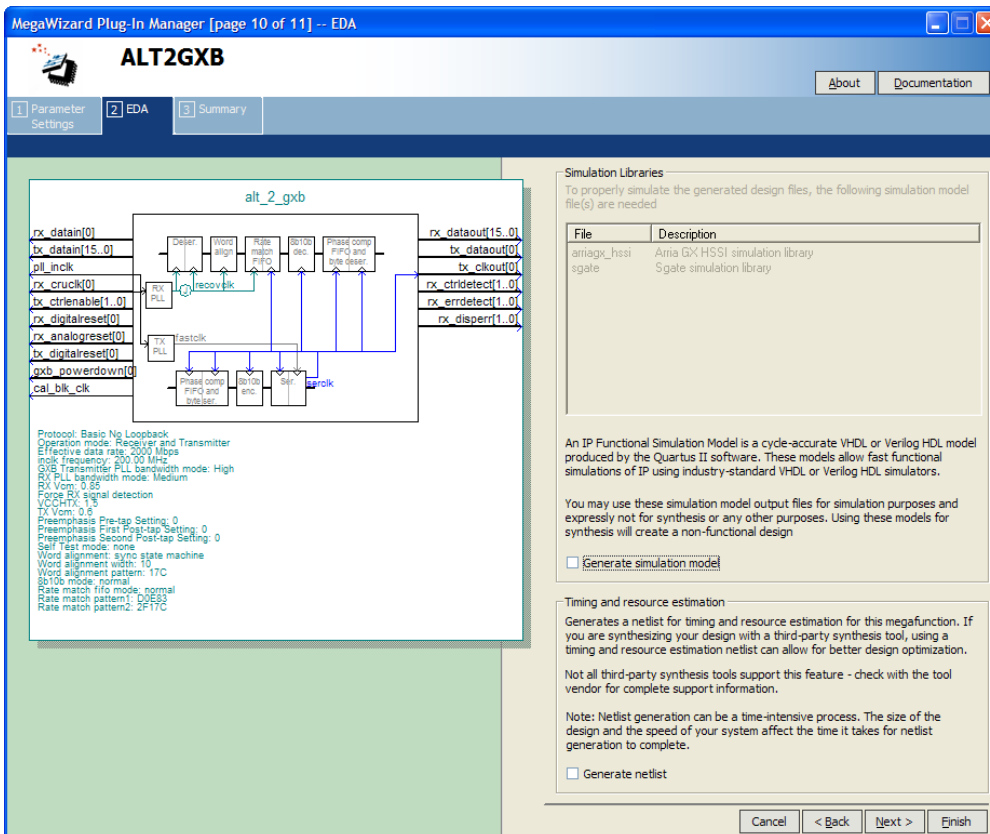
Figure 3–11 shows page 11 (last page) of the MegaWizard Plug-In Manager for Basic protocol mode set up. You can select optional files on this page. After you make your selections, click **Finish** to generate the files.

*Figure 3–11. MegaWizard Plug-In Manager - ALT2GXB (Summary)*
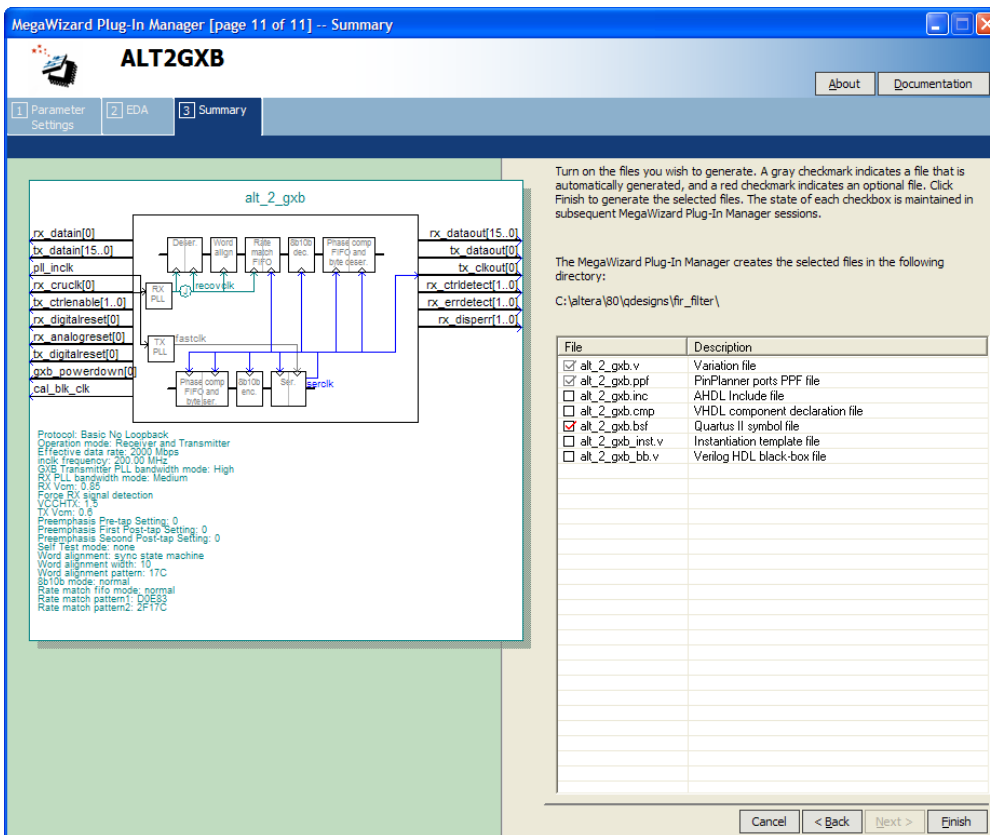


## PCI Express (PIPE) Mode

This section provides descriptions of the options available on the individual pages of the ALT2GXB MegaWizard Plug-In Manager for the PCI Express (PIPE) mode. The MegaWizard Plug-In Manager provides a warning if any of the settings you choose are illegal.

Figure 3–12 shows page 3 of the ALT2GXB MegaWizard Plug-In Manager for PCI Express (PIPE) mode.

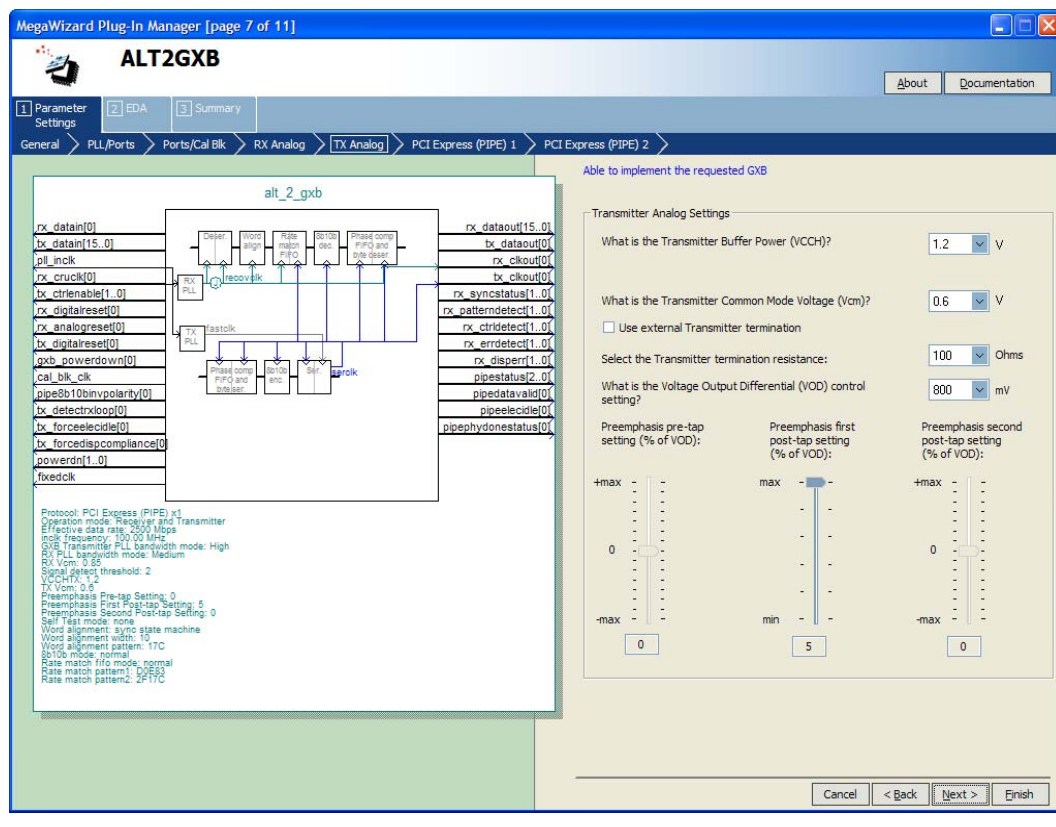*Figure 3–12. MegaWizard Plug-In Manager - ALT2GXB (General)*

Table 3–8 describes the available options on page 3 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

| Table 3–8. MegaWizard Plug-In Manager Options (Page 3 for PCI Express [PIPE] Mode)   (Part 1 of 2) | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Which protocol will you be using? | Determines the specific protocol or modes under which the transceiver operates. For PCI Express (PIPE) mode, you must select the PCI Express (PIPE) protocol. | — |
| Which subprotocol will you be using? | In PCI Express (PIPE) mode, the subprotocols are the supported link widths: 1 or 4. | PCI Express (PIPE) Mode section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Enforce default settings for this protocol | Selecting this option skips the **PCI** screen in the PCI Express (PIPE) MegaWizard Plug-In Manager. The **PCI** screen allows you to select the PCI Express (PIPE) specific ports for your design. If you select this option, all PCI Express (PIPE) specific ports are used. | — |
| What is the operation mode? | Only the receiver and transmitter (full duplex) mode is allowed in the PCI Express (PIPE) mode. Receiver-only and transmitter only modes are not allowed. | — |
| What is the number of channels? | This determines how many duplicate channels this ALT2GXB instance contains. In a x4 subprotocol, the number of channels increments by 4. | — |
| What is the deserializer block width? | This option is unavailable in PCI Express (PIPE) mode. | — |
| What is the channel width? | This option determines the PLD-transceiver interface width. Only 16-bit interface width is supported. | Byte Serializer and Byte Deserializer sections in the *Arria GX Architecture* chapter in volume 1 of the *Arria GX Device Handbook* |
| What would you like to base the setting on? | This option is unavailable because the data rate is fixed at 2500 Mbps for PCI Express (PIPE) mode. | — |
| What is the data rate? | This option is unavailable because the data rate is fixed at 2500 Mbps for PCI Express (PIPE) mode. | — |

*Table 3–8. MegaWizard Plug-In Manager Options (Page 3 for PCI Express [PIPE] Mode)   (Part 2 of 2)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| What is the input clock frequency? | Determines the input reference clock frequency for the transceiver. In PCI Express (PIPE) mode, only 100 MHz is allowed. | PCI Express (PIPE) Mode section in the *Arria GX Architecture* chapter in volume 1 of the *Arria GX Device Handbook* |
| What is the data rate division factor? | This option is unavailable in PCI Express (PIPE) mode. | — |

Figure 3–13 shows page 4 of the ALT2GXB MegaWizard Plug-In Manager for PCI Express (PIPE) mode.

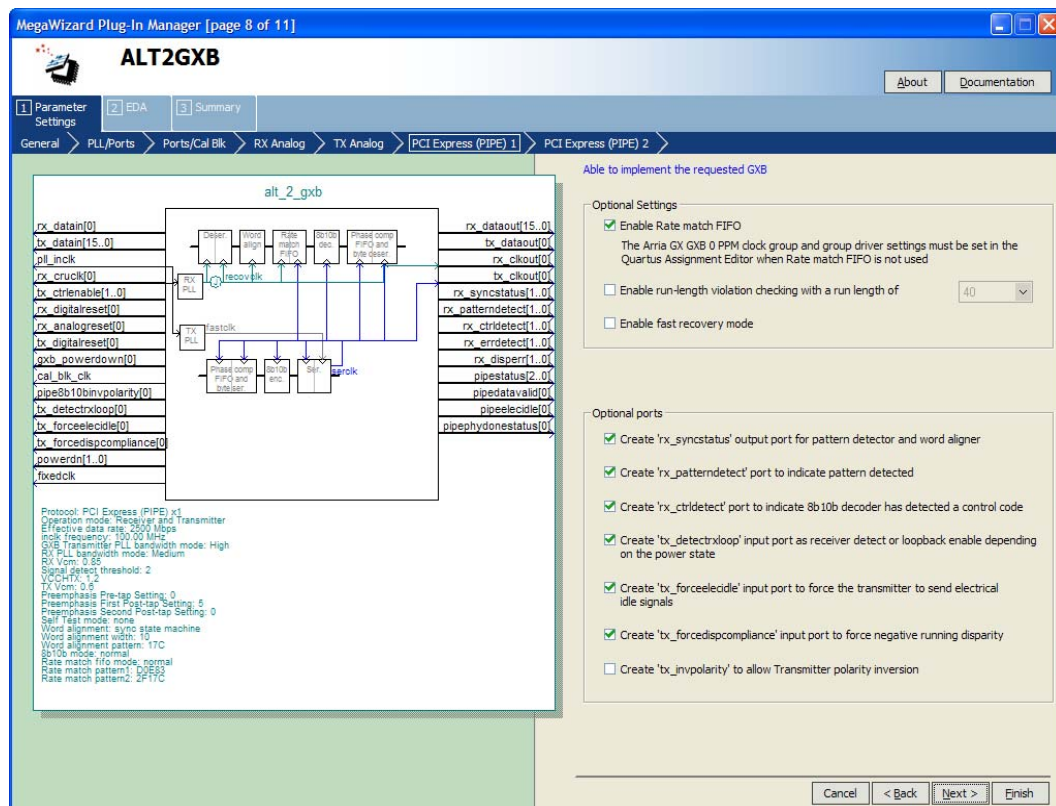*Figure 3–13. MegaWizard Plug-In Manager - ALT2GXB (PLL/Ports)*
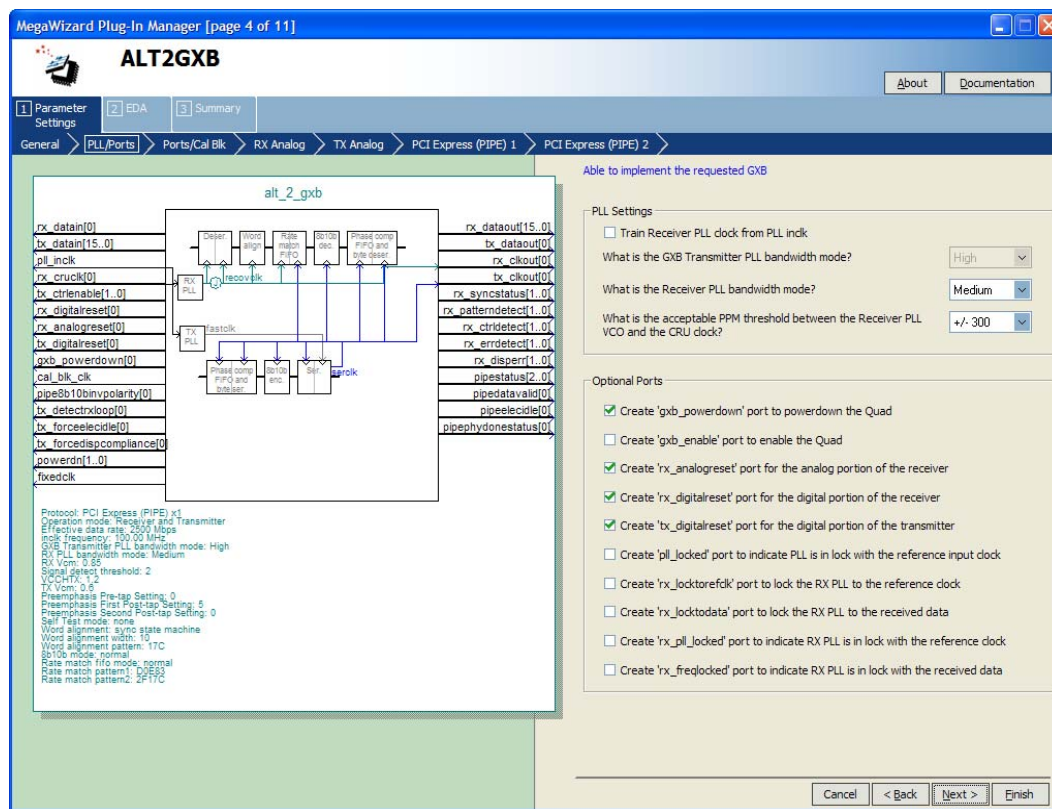
Table 3–9 describes the available options on page 4 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

| Table 3–9. MegaWizard Plug-In Manager Options (Page 4 for PCI Express [PIPE] Mode)   (Part 1 of 2) | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Train Receiver PLL clock from PLL_inclk | If you select this option, the transmitter input reference clock (pll_inclk) drives the receiver PLL input reference clock also. If you do not select this option, the signal on the rx_cruclk port drives the receiver PLL input reference clock. | — |
| What is the GXB Transmitter PLL bandwidth mode? | This option is not available in PCI Express (PIPE) mode because the transmitter PLL bandwidth is fixed at high. | — |
| What is the Receiver PLL bandwidth mode? | This option is not available in PCI Express (PIPE) mode because the receiver PLL bandwidth is fixed at medium. | — |
| What is the acceptable PPM threshold between the Receiver PLL VCO and the CRU clock? | This option determines the PPM difference that affects the automatic receiver clock recovery unit (CRU) switchover between lock-to-data and lock-to-reference. (There are additional factors that affect the CRU's transition.) | Clock Recovery Unit (CRU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create gxb_powerdown port to power down the Quad | This signal can be used to reset and power down all circuits in the transceiver block. It does not power down the REFCLK buffers and reference clock lines. | Reset Control and Power Down section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| Create gxb_enable port to enable the Quad | This signal can be used to enable Arria GX transceiver blocks. If instantiated, this port must be tied to the dedicated gigabit transceiver block enable input pin. | Reset Control and Power Down section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| Create rx_analogreset port for the analog portion of the receiver | Receiver analog reset port. | Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create rx_digitalreset port for the digital portion of the receiver | Receiver digital reset port. Resets the PCS logic of the receiver. Altera recommends using this port to implement the recommended reset sequence. | Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |

| *Table 3–9. MegaWizard Plug-In Manager Options (Page 4 for PCI Express [PIPE] Mode)   (Part 2 of 2)* | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Create `tx_digitalreset` port for the digital portion of the transmitter | Transmitter digital reset port. Resets the PCS logic of the transmitter. Altera recommends using this port to implement the recommended reset sequence. | Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `pll_locked` port to indicate PLL is in lock with the reference input clock | PLL locked indicator for the transmitter PLLs. | Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `rx_locktorefclk` port to lock the RX PLL to the reference clock | Lock-to-reference lock mode for the CRU. Use with `rx_locktodata`.<br>`rx_locktodata/rx_locktorefclk`<br>0/0—CRU is in automatic mode<br>0/1—CRU is in lock-to-reference clock<br>1/0—CRU is in lock-to-data mode<br>1/1—CRU is in lock-to-data mode | Clock Recovery Unit (CRU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `rx_locktodata` port to lock the RX PLL to the received data | Lock-to-data control for the CRU. Use with `rx_locktorefclk`. | Clock Recovery Unit (CRU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `rx_pll_locked` port to indicate RX PLL is in lock with the reference clock | Receiver PLL locked signal. Indicates if the receiver PLL is phase locked to the CRU reference clock. | Reset Control and Power Down section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `rx_freqlocked` port to indicate RX PLL is in lock with the received data | CRU mode indicator port. Indicates if the CRU is locked to data mode or locked to the reference clock mode.<br>0—Receiver CRU is in lock-to-reference clock mode<br>1—Receiver CRU is in lock-to-data mode | Clock Recovery Unit (CRU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |

Figure 3–14 shows page 5 of the ALT2GXB MegaWizard Plug-In Manager for PCI Express (PIPE) mode.

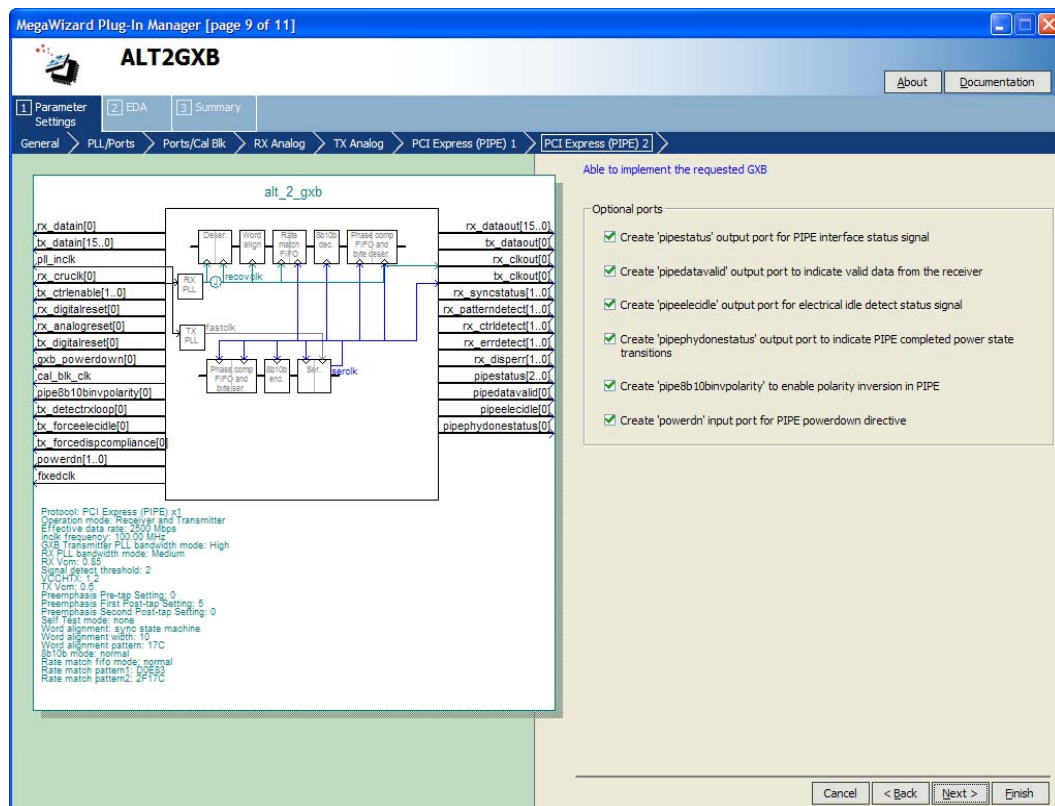*Figure 3–14. MegaWizard Plug-In Manager - ALT2GXB (Ports/Cal Blk)*
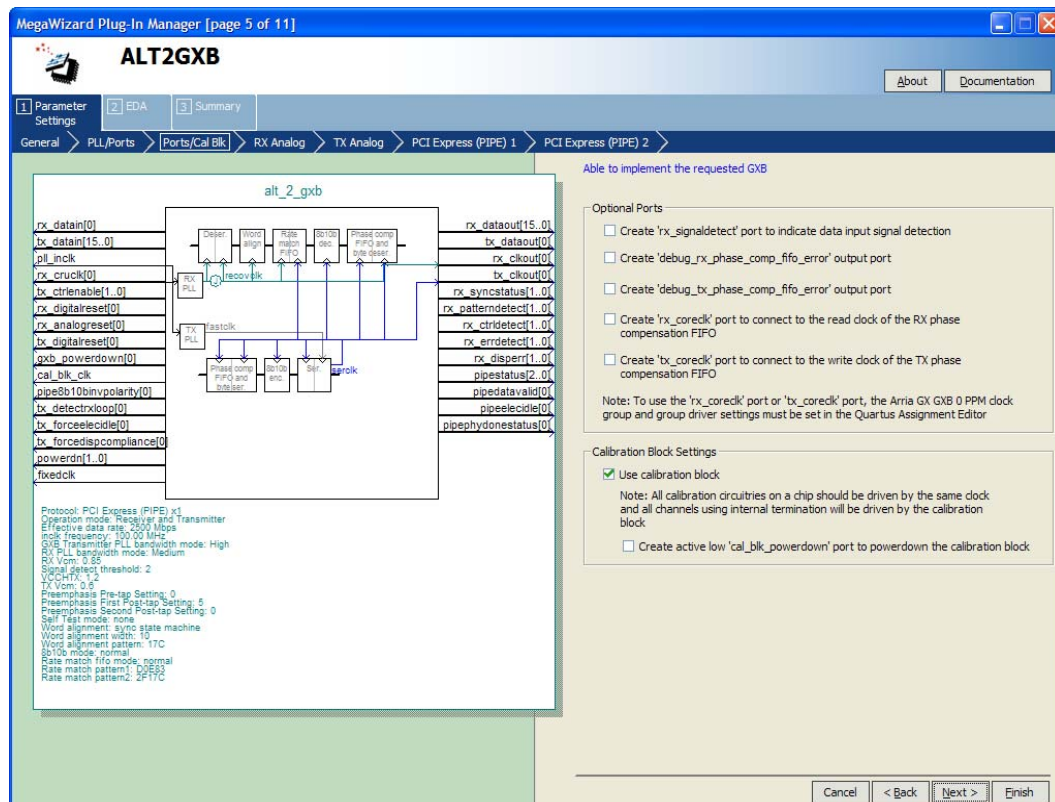
Table 3–10 describes the available options on page 5 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

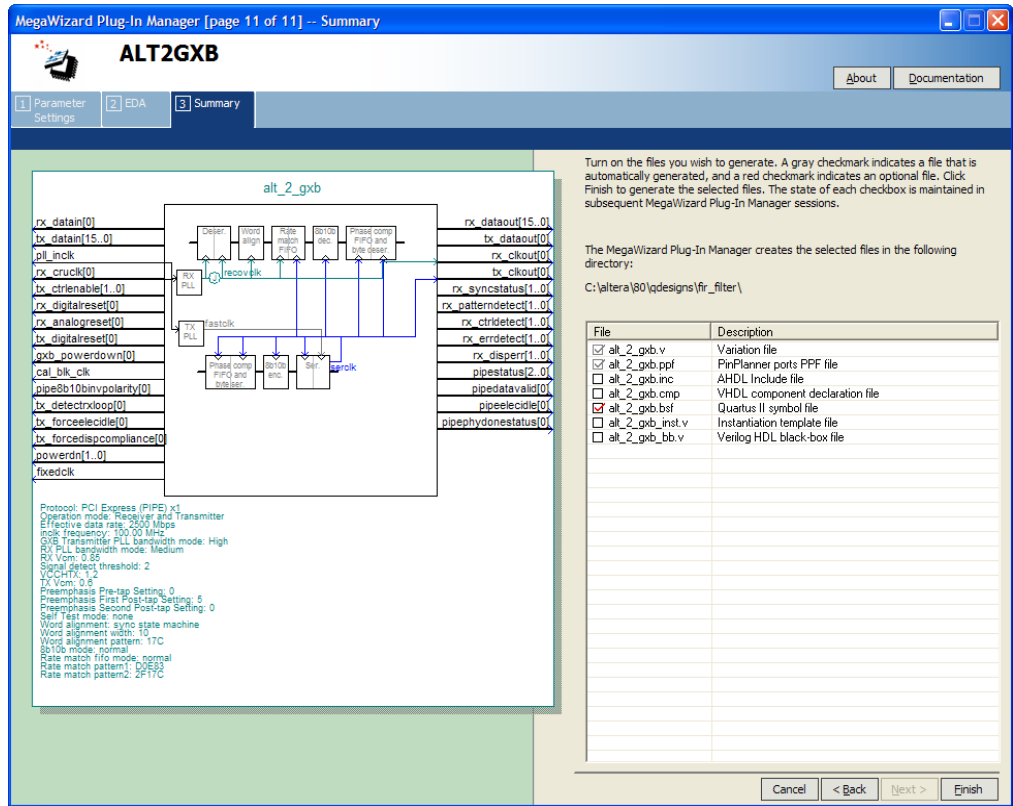*Table 3–10. MegaWizard Plug-In Manager Options (Page 5 for PCI Express [PIPE] Mode)   (Part 1 of 2)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| Create rx_signaldetect port to indicate data input signal detection | Signal detect port. In PCI Express (PIPE) mode, indicates if a signal that meets the specified range is present at the input of the receiver buffer. In all other modes, rx_signaldetect is forced high and must not be used as an indication of a valid signal at receiver input. | Receiver Buffer section under PCI Express (PIPE) mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create debug_rx_phase_comp_ fifo_error output port | This optional output port indicates Receiver Phase Compensation FIFO overflow/under run condition. Note that no PPM difference is allowed between FIFO read and write clocks. Use this port for debugging purposes only. | Receiver Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create debug_tx_phase_comp_ fifo_error output port | This optional output port indicates Transmitter Phase Compensation FIFO overflow/under run condition. Note that no PPM difference is allowed between FIFO read and write clocks. Use this port for debug purposes only. | Transmitter Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create rx_coreclk port to connect to the read clock of the RX phase compensation FIFO | This optional input port allows you to clock the read side of the Receiver Phase Compensation FIFO with a non-transceiver PLD clock. | PLD-Transceiver Interface Clocking section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create tx_coreclk port to connect to the write clock of the TX phase compensation FIFO | This optional input port allows you to clock the write side of the Transmitter Phase Compensation FIFO with a non-transceiver PLD clock. | PLD-Transceiver Interface Clocking section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |

| *Table 3–10. MegaWizard Plug-In Manager Options (Page 5 for PCI Express [PIPE] Mode) (Part 2 of 2)* | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Use calibration block | This option allows you to select which instance of the ALT2GXB megafunction instantiates the calibration block. Only one instance of the ALT2GXB megafunction is required to instantiate the calibration block. | Calibration Block section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create active low `cal_blk_powerdown` port to powerdown the calibration block | Power-down signal for the calibration block. Assertion of this signal may interrupt data transmission and reception. Use this signal to re-calibrate the termination resistors if temperature and/or voltage changes warrant it. | Calibration Block section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |

Figure 3–12 shows page 6 of the ALT2GXB MegaWizard Plug-In Manager for PCI Express (PIPE) mode.

*Figure 3–15. MegaWizard Plug-In Manager - ALT2GXB (RX Analog)*
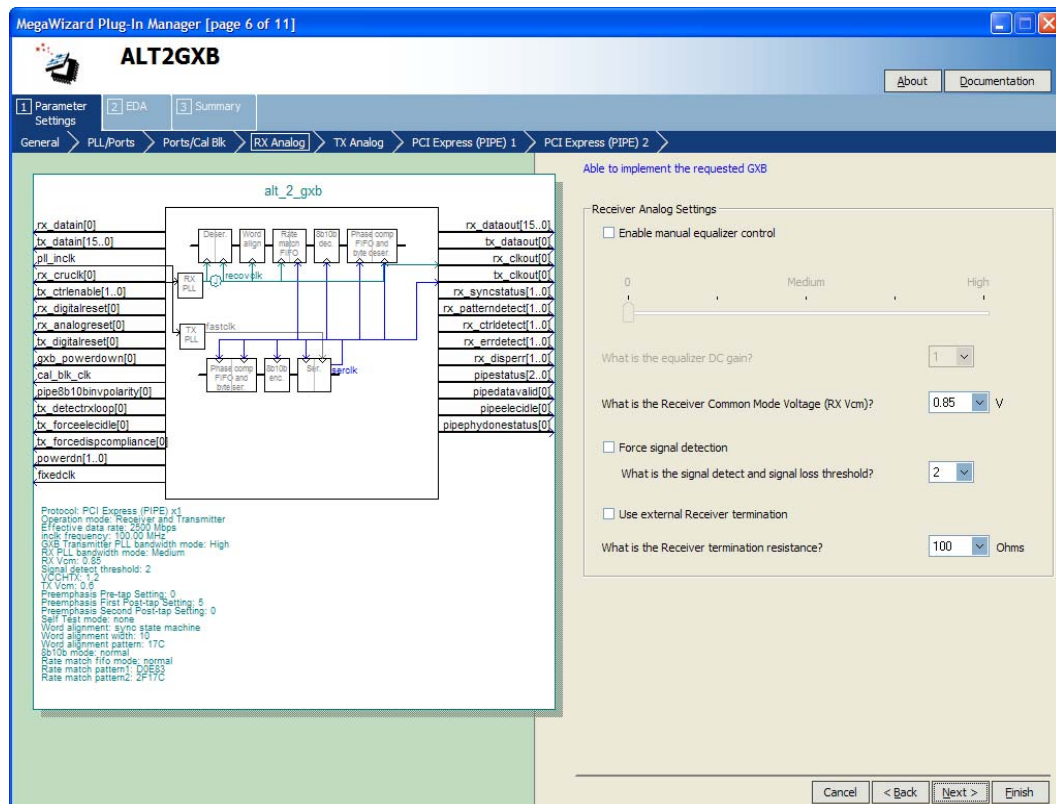
Table 3–11 describes the available options on page 6 of the MegaWizard Plug-In Manager for your `ALT2GXB` custom megafunction variation.

| Table 3–11. MegaWizard Plug-In Manager Options (Page 6 for PCI Express [PIPE] Mode) | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Enable manual equalizer control | This option enables the 0–4 setting options for manual equalizer control. | Receiver Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What is the equalizer DC gain? | In PIPE mode, a DC gain setting of 1 is forced. | Receiver Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What is the Receiver Common Mode Voltage (RX $V_{CM}$)? | The receiver common mode voltage is programmable. The options available are 0.85 V and 1.2 V. | Receiver Buffer section under PCI Express (PIPE) mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Force signal detection | This option disables the signal detect circuit. You must not select this option as signal detect circuitry is required for electrical idle detection at the receiver. | Receiver Buffer section under PCI Express (PIPE) mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| What is the signal detect and signal loss threshold? | This option sets the trip point of the signal detect circuit. You must select a threshold level of **2** in PCI Express (PIPE) mode. | Receiver Buffer section under PCI Express (PIPE) mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |

| _Table 3–11. MegaWizard Plug-In Manager Options (Page 6 for PCI Express [PIPE] Mode)_ | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Use external receiver termination | This option is available if you use an external termination resistor instead of the OCT. If checked, this option turns off the receiver OCT. | — |
| What is the receiver termination resistance? | In PCI Express (PIPE) mode, the only supported receiver termination resistance is 100 Ω | Receiver Buffer section under PCI Express (PIPE) mode in the _Arria GX Transceiver Protocol Support and Additional Features_ chapter in volume 2 of the _Arria GX Device Handbook_ |

Figure 3–16 shows page 7 of the ALT2GXB MegaWizard Plug-In Manager for PCI Express (PIPE) mode.

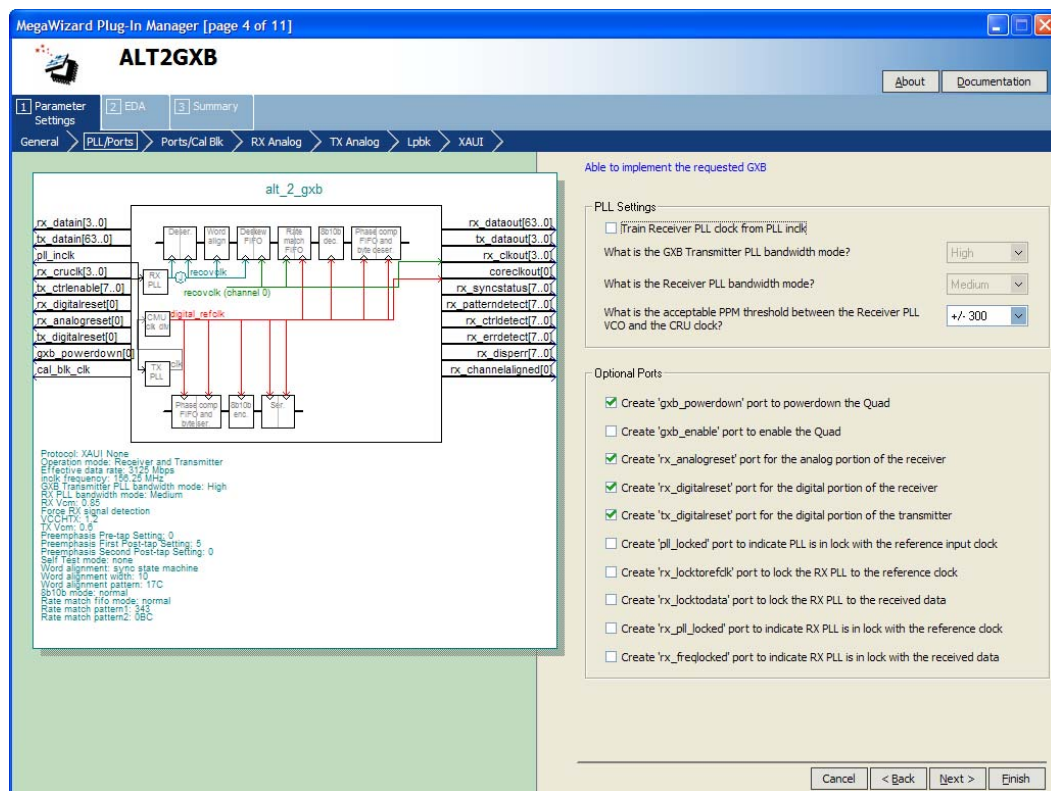*Figure 3–16. MegaWizard Plug-In Manager - ALT2GXB (TX Analog)*

Table 3–12 describes the available options on page 7 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

*Table 3–12. MegaWizard Plug-In Manager Options (Page 7 for PCI Express [PIPE] Mode)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| What is the Transmitter Buffer Power (VCCH)? | In PCI Express (PIPE) mode, the transmitter buffer power is fixed at 1.2 V. You must connect the $V_{CCH}$ power pins of a PCI Express (PIPE) transceiver bank to a 1.2 V power supply. You must select 1.2 V PCML I/O standard for the transmitter data output pins. | Transmitter Buffer section under PCI Express (PIPE) Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| What is the Transmitter Common Mode Voltage ($V_{CM}$)? | In PCI Express (PIPE) mode, the transmitter common mode voltage is fixed at 0.6 V. | Transmitter Buffer section under PCI Express (PIPE) Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Use external Transmitter termination | This option is available if you want to use an external termination resistor instead of the OCT. Checking this option turns off the transmitter OCT. | — |
| Select the Transmitter termination resistance | In PCI Express (PIPE) mode, the only supported receiver termination resistance is 100 Ω | Transmitter Buffer section under PCI Express (PIPE) Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| What is the Voltage Output Differential (VOD) control setting? | This option selects the $V_{OD}$ of the transmitter buffer. The differential output voltage is programmable between 400 mV and 1200 mV in steps of 200 mV. The available $V_{OD}$ settings change based on $V_{CCH}$. | Transmitter Buffer section under PCI Express (PIPE) Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Pre-emphasis pre-tap setting (% of VOD) | This option is not available in Arria GX devices and is fixed at 0. | — |

| *Table 3–12. MegaWizard Plug-In Manager Options (Page 7 for PCI Express [PIPE] Mode)* | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Pre-emphasis first post-tap setting (% of VOD) | This option sets the amount of pre-emphasis on the transmitter buffer using first post-tap. The options available are 0, 1, 2, 3, 4, and 5. | Transmitter Buffer section under PCI Express (PIPE) Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Pre-emphasis second post-tap setting (% of VOD) | This option is not available in Arria GX devices and is fixed at 0. | — |

Figure 3–17 shows page 8 of the ALT2GXB MegaWizard Plug-In Manager for PCI Express (PIPE) mode. If the **Enforce default settings for this protocol** option is selected, this page does not appear in the MegaWizard Plug-In Manager.

*Figure 3–17. MegaWizard Plug-In Manager - ALT2GXB (PCI Express [PIPE] 1)*

Table 3–13 describes the available options on page 8 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

| Table 3–13. MegaWizard Plug-In Manager Options (Page 8 for PCI Express [PIPE] Mode)  (Part 1 of 2) | | |
|---|---|---|
| ALT2GXB Setting | Description | Reference |
| Enable Rate match FIFO | This option enables bypassing of the rate match FIFO in the receiver data path (Low-latency [Synchronous] PCI Express [PIPE] mode). | Low-latency (Synchronous) PCI Express (PIPE) Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Enable run-length violation checking with a run length of | This option activates the run length violation circuit. You can program the run length at which the circuit triggers the rx_rlv signal. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Enable fast recovery mode | This option creates the NFTS fast recovery IP required to meet the PCI Express (PIPE) specification in the PLD logic array. | NFTS Fast Recovery IP (NFRI) section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create rx_syncstatus output port for pattern detector and word aligner | Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Word Aligner section under PCI Express (PIPE) Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create rx_patterndetect output port to indicate pattern detected | Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Word Aligner section under PCI Express (PIPE) Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create rx_ctrldetect output port to indicate 8B/10B decoder has detected a control code | Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | 8B/10B Decoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |

*Table 3–13. MegaWizard Plug-In Manager Options (Page 8 for PCI Express [PIPE] Mode)   (Part 2 of 2)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| Create `tx_detectrxloop` input port as receiver detect or loopback enable, depending on the power state | Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | PCI Express (PIPE) Mode section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `tx_forceelecidle` input port to force the transmitter to send Electrical Idle signals | Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | PCI Express (PIPE) Mode section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `tx_forcedispcompliance` input port to force negative running disparity | Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook 2* for information about this port. | PCI Express (PIPE) Mode section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `tx_invpolarity` to allow Transmitter polarity inversion | This optional port allows you to dynamically reverse the polarity of the data to be transmitted at the transmitter PCS-PMA interface. | 8B/10B Encoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook\* |

Figure 3–18 shows page 9 of the ALT2GXB MegaWizard Plug-In Manager for PCI Express (PIPE) mode. If the **Enforce default settings for this protocol** option is selected, this page does not appear in the MegaWizard Plug-In Manager.

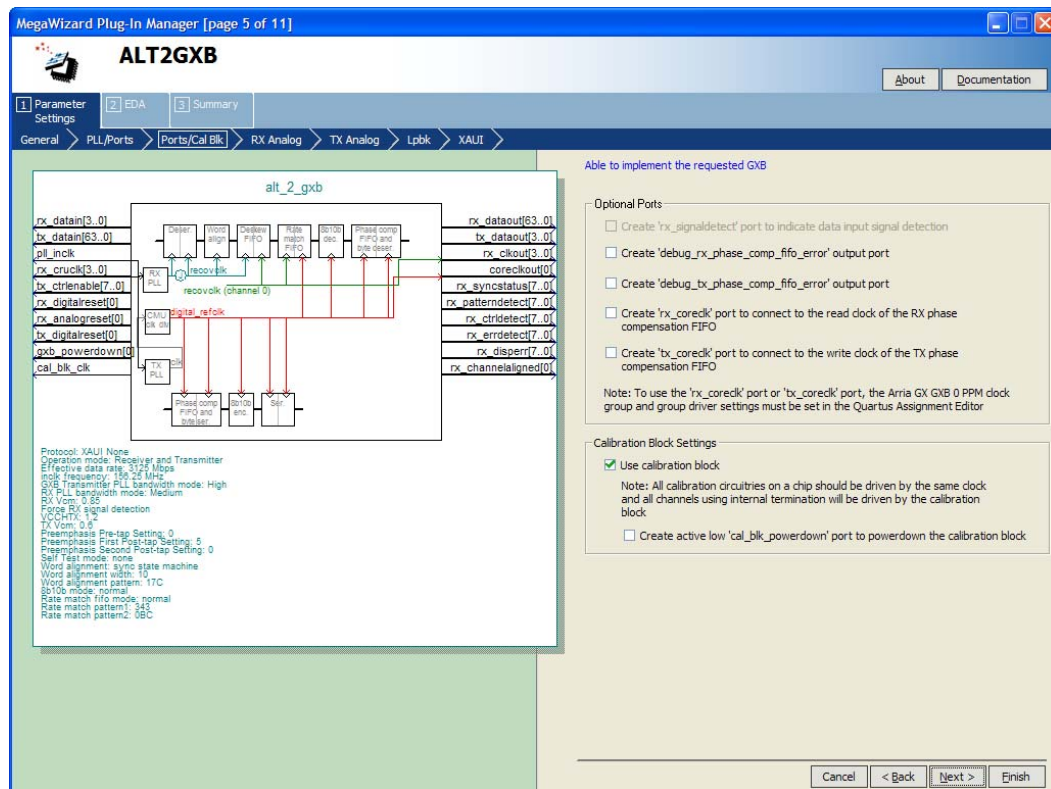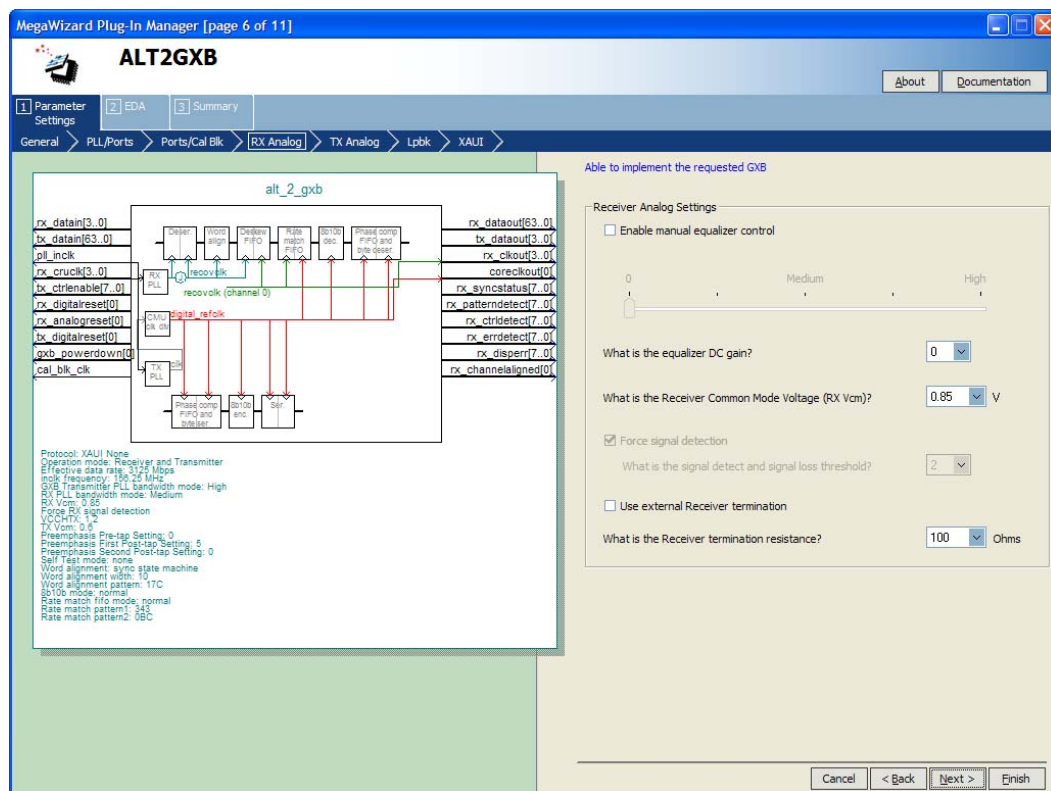*Figure 3–18. MegaWizard Plug-In Manager - ALT2GXB (PCI Express [PIPE] 2)*

Table 3–13 describes the available options on page 9 of the MegaWizard Plug-In Manager for your `ALT2GXB` custom megafunction variation.

*Table 3–14. MegaWizard Plug-In Manager Options (Page 9 for PCI Express [PIPE] Mode)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| Create `pipestatus` output port for PIPE interface status signal | Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Receiver Status section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `pipedatavalid` output port to indicate valid data from the receiver | Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | PCI Express (PIPE) Mode section *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `pipeelecidle` output port for Electrical Idle detect status signal | Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | PCI Express (PIPE) Mode section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `pipephydonestatus` output port to indicate PIPE completed power state transitions | Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | PCI Express (PIPE) Mode section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `pipe8b/10binvpolarity` to enable polarity inversion in PIPE | Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | PCI Express (PIPE) Mode section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create powerdn input port for PIPE powerdown directive | Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | PCI Express (PIPE) Mode section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |

Figure 3–19 shows page 10 of the MegaWizard Plug-In Manager for the PCI Express (PIPE) protocol selection. The **Generate simulation model** option creates a behavioral model (**.vo** or **.vho**) of the transceiver instance for third-party simulators. The **Generate a netlist for synthesis area and timing estimation** option creates a netlist file (**.syn**) for third-party synthesis tools.

*Figure 3–19. MegaWizard Plug-In Manager - ALT2GXB (EDA)*

Figure 3–20 shows page 11 (the last page) of the MegaWizard Plug-In Manager for the PCI Express (PIPE) protocol set up. You can select optional files on this page. After you make your selections, click **Finish** to generate the files.

*Figure 3–20. MegaWizard Plug-In Manager - ALT2GXB (Summary)*



# XAUI Mode

This section provides descriptions of the options available on the individual pages of the ALT2GXB MegaWizard Plug-In Manager for XAUI mode. The MegaWizard Plug-In Manager provides a warning if any of the settings you choose are illegal.

☞ The word aligner and rate matcher operations and patterns are pre-configured for XAUI mode and cannot be altered.

Figure 3–21 shows page 3 of the ALT2GXB MegaWizard Plug-In Manager for XAUI mode.

*Figure 3–21. MegaWizard Plug-In Manager - ALT2GXB (General)*



Table 3–15 describes the available options on page 3 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

| *Table 3–15. MegaWizard Plug-In Manager Options (Page 3 for XAUI Mode)   (Part 1 of 2)* | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Which protocol will you be using? | Determines the specific protocol or modes under which the transceiver operates. For XAUI mode you must select the **XAUI**. | — |
| Which subprotocol will you be using? | Not applicable to XAUI mode. | — |

| *Table 3–15. MegaWizard Plug-In Manager Options (Page 3 for XAUI Mode)   (Part 2 of 2)* | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Enforce default settings for this protocol | Selecting this option skips the **XAUI** screen of the XAUI MegaWizard Plug-In Manager. The **XAUI** screen allows you to select the XAUI-specific ports for your design. If you select this option, all XAUI-specific ports are used. | — |
| What is the operation mode? | Only receiver and transmitter (full duplex) is allowed in the XAUI protocol. Receiver only and transmitter only modes are not allowed. | — |
| What is the number of channels? | This selects how many duplicate channels this ALT2GXB instance contains. In XAUI mode, the number of channels increments by 4. | — |
| What is the deserializer block width? | XAUI mode only operates in a single-width mode. | — |
| What is the channel width? | This option determines the transceiver-to-PLD interface width. Only 16-bit channel width is allowed in XAUI mode. | Byte Serializer and Byte Deserializer sections in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| What would you like to base the setting on? | This option is not available in XAUI mode. | — |
| What is the data rate? | The data rate is fixed at 3.125 Gbps for the XAUI protocol. | — |
| What is the input clock frequency? | Determines the input reference clock frequency for the transceiver. The Quartus II software automatically selects the input reference clock frequency based on the entered data rate. | — |
| What is the data rate division factor? | This option is not available in XAUI mode. | — |

Figure 3–22 shows page 4 of the ALT2GXB MegaWizard Plug-In Manager for XAUI mode.

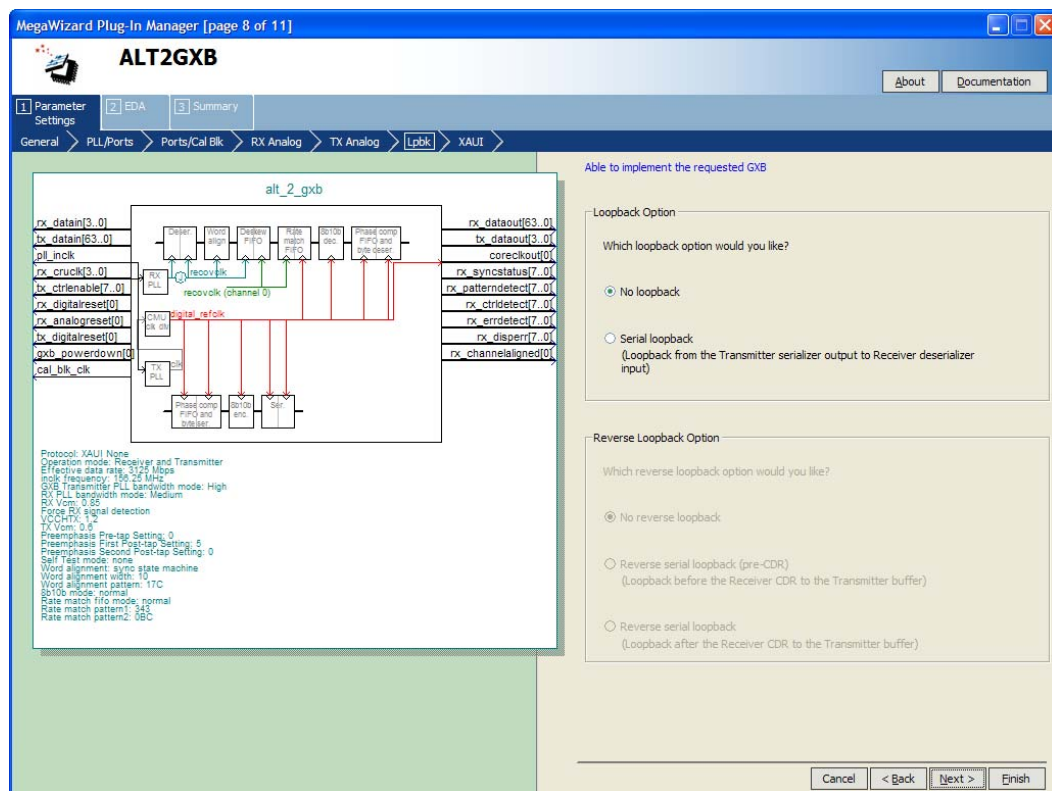*Figure 3–22. MegaWizard Plug-In Manager - ALT2GXB (PLL/Ports)*

Table 3–16 describes the available options on page 4 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

**Table 3–16. MegaWizard Plug-In Manager Options (Page 4 for XAUI Mode)   (Part 1 of 2)**

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| Train Receiver PLL clock from PLL inclk | If you turn this option on, your design uses the input reference clock to the transmitter PLL to train the receiver PLL. This reduces the need to supply a separate receiver PLL reference clock. | Clock Recovery Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What is the GXB Transmitter PLL bandwidth mode? | In XAUI mode, only high bandwidth is supported for the transmitter PLL. | Clock Recovery Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What is the Receiver PLL bandwidth mode? | In XAUI mode, only medium bandwidth is supported for the receiver PLL and VCO. | Clock Recovery Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What is the acceptable PPM threshold between the Receiver PLL VCO and the CRU clock? | This option determines the PPM difference that affects the automatic receiver CRU switchover between lock-to-data and lock-to-reference. (There are additional factors that affect CRU's transition.) | Clock Recovery Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `gxb_powerdown` port to power down the Quad | This signal can be used to reset and power down all circuits in the transceiver block. It does not power down the REFCLK buffers and reference clock lines. | Reset Control and Power Down section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| Create `gxb_enable` port to enable the Quad | This signal can be used to enable Arria GX transceiver blocks. If instantiated, this port must be tied to the dedicated gigabit transceiver block enable input pin. | Reset Control and Power Down section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| Create `rx_analogreset` port for the analog portion of the receiver | Receiver analog reset port. | Reset Control and Power Down section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |

*Table 3–16. MegaWizard Plug-In Manager Options (Page 4 for XAUI Mode)   (Part 2 of 2)*

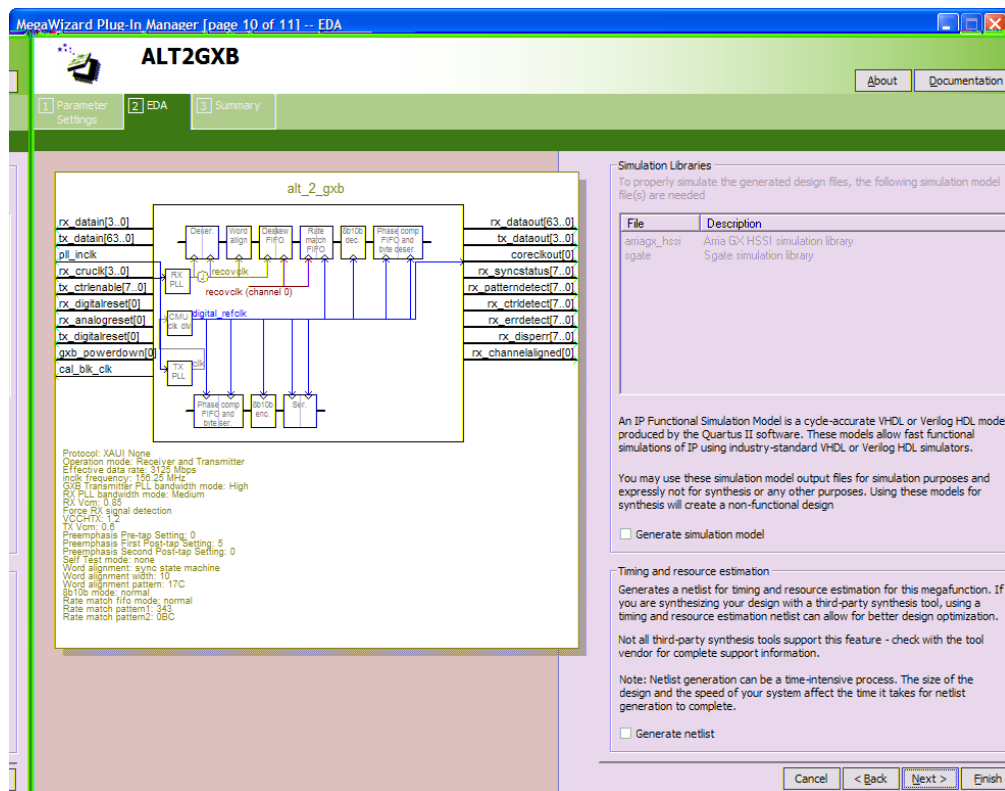| ALT2GXB Setting | Description | Reference |
|---|---|---|
| Create `rx_digitalreset` port for the digital portion of the receiver | Receiver digital reset port. Resets the PCS portion of the receiver. Altera recommends using this port along with logic to implement the recommended reset sequence. | Reset Control and Power Down section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `tx_digitalreset` port for the digital portion of the transmitter | Transmitter digital reset port. Resets the PCS portion of the transmitter. Altera recommends using this port along with logic to implement the recommended reset sequence. | Reset Control and Power Down section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `pll_locked` port to indicate PLL is in lock with the reference input clock | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Clock Multiplier Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `rx_locktorefclk` port to lock the RX PLL to the reference clock | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Clock Recovery Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `rx_locktodata` port to lock the RX PLL to the received data | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Clock Recovery Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `rx_pll_locked` port to indicate RX PLL is in lock with the reference clock | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Clock Recovery Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `rx_freqlocked` port to indicate RX PLL is in lock with the received data | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Clock Recovery Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |

Figure 3–22 shows page 5 of the ALT2GXB MegaWizard Plug-In Manager for XAUI mode.

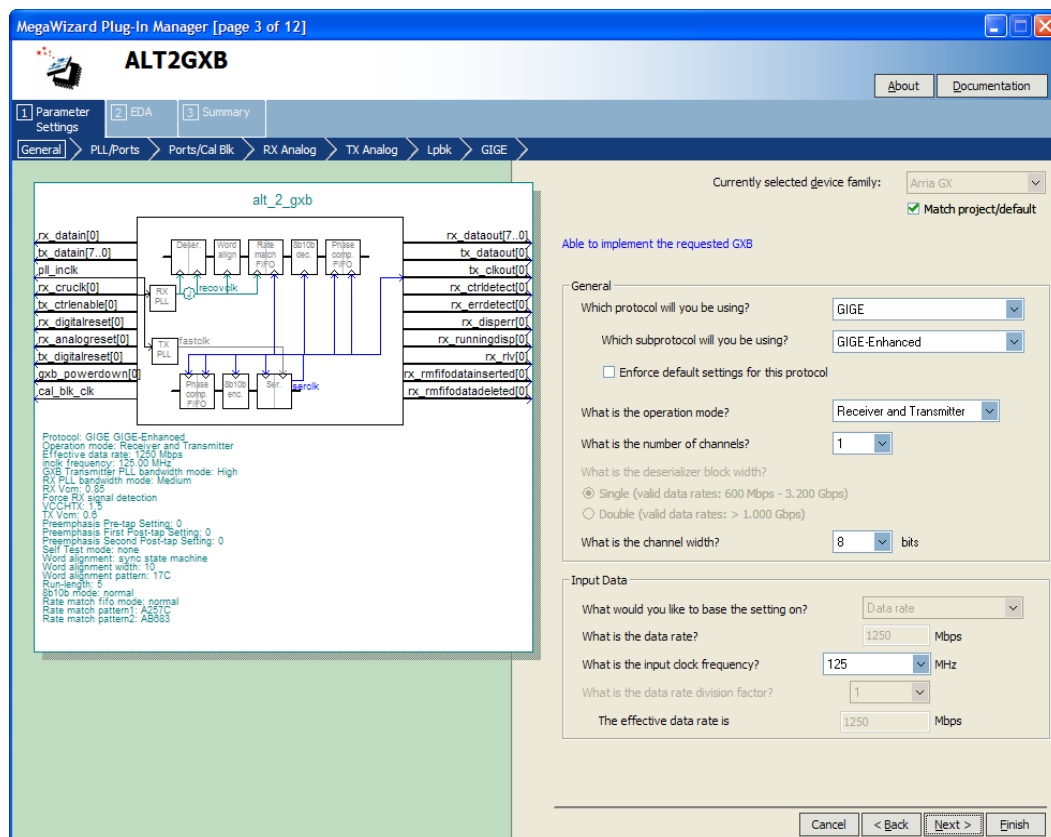*Figure 3–23. MegaWizard Plug-In Manager - ALT2GXB (Ports/Cal Blk)*

Table 3–17 describes the available options on page 5 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

*Table 3–17. MegaWizard Plug-In Manager Options (Page 5 for XAUI Mode)   (Part 1 of 2)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| Create `rx_signaldetect` port to indicate data input signal detection | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Receiver Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `debug_rx_phase_comp_fifo_error` output port | This optional output port indicates Receiver Phase Compensation FIFO overflow/under run condition. Note that no PPM difference is allowed between FIFO read and write clocks. Use this port for debug purpose only. | Receiver Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `debug_tx_phase_comp_fifo_error` output port | This optional output port indicates Transmitter Phase Compensation FIFO overflow/under run condition. Note that no PPM difference is allowed between FIFO read and write clocks. Use this port for debug purposes only. | Transmitter Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `rx_coreclk` port to connect to the read clock of the RX phase compensation FIFO | This optional input port allows you to clock the read side of the Receiver Phase Compensation FIFO with a non-transceiver PLD clock. | Transceiver Clocking section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `tx_coreclk` port to connect to the write clock of the TX phase compensation FIFO | This optional input port allows you to clock the write side of the Transmitter Phase Compensation FIFO with a non-transceiver PLD clock. | Transmitter Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |

| *Table 3–17. MegaWizard Plug-In Manager Options (Page 5 for XAUI Mode)   (Part 2 of 2)* | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Use calibration block | This option allows you to select which instance of the ALT2GXB megafunction instantiates the calibration block. Only one instance of the ALT2GXB megafunction is required to instantiate the calibration block. | Calibration Blocks section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `cal_blk_powerdown` to power down the calibration block | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. for information about this port. | Calibration Blocks section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |

Figure 3–24 shows page 6 of the ALT2GXB MegaWizard Plug-In Manager for XAUI mode.

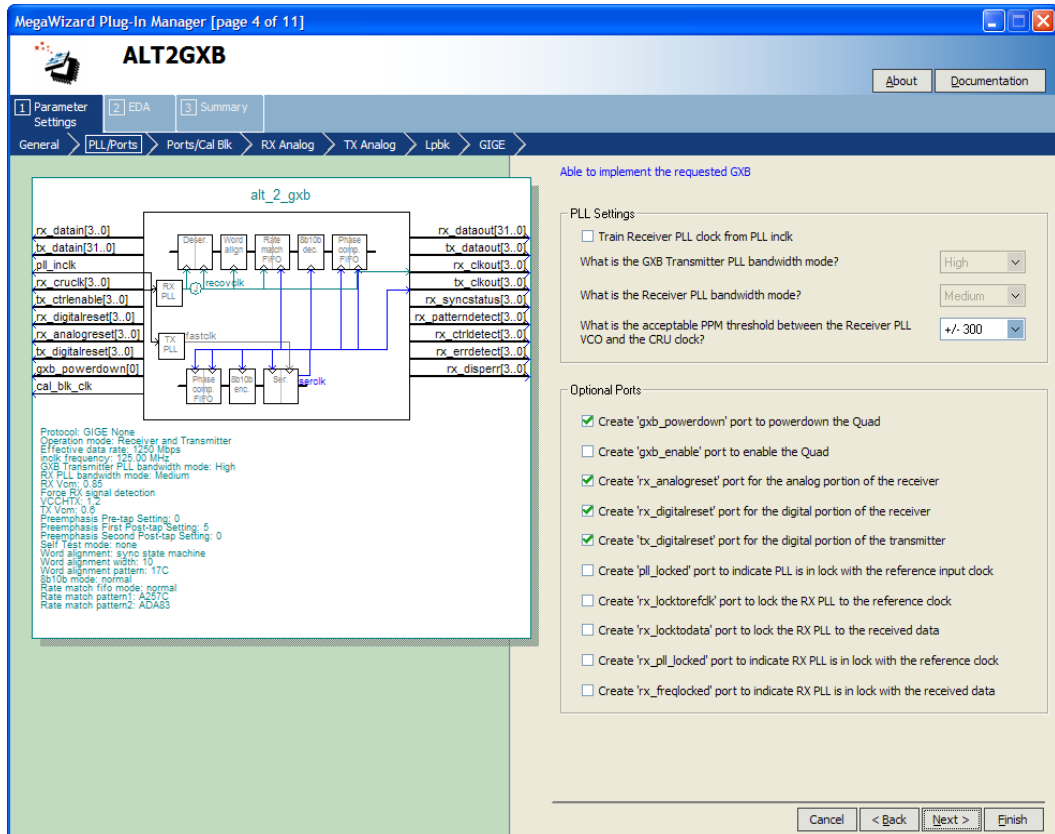*Figure 3–24. MegaWizard Plug-In Manager - ALT2GXB (RX Analog)*

Table 3–18 describes the available options on page 6 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

| *Table 3–18. MegaWizard Plug-In Manager Options (Page 6 for XAUI Mode)* | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Enable static equalizer control | This option enables the 0–4 setting options for manual equalizer control. | Receiver Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What is the Receiver Common Mode Voltage (RX V$_{CM}$)? | The receiver common mode voltage is programmable. The selections available are 0.85 V or 1.2 V. | Receiver Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Force signal detection | This option is available only in PCI Express (PIPE) mode. | Receiver Buffer Section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What is the signal detect and signal loss threshold? | This option is available only in PCI Express (PIPE) mode. | Receiver Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Use external receiver termination | This option is available if you use an external termination resistor instead of the on-chip termination OCT. If checked, this option turns off the receiver OCT. | Receiver Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What is the Receiver termination resistance? | This option selects the receiver termination value. In Arria GX devices, the receiver termination value is fixed at 100 Ω | Receiver Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |

Figure 3–25 shows page 7 of the ALT2GXB MegaWizard Plug-In Manager for XAUI mode.

*Figure 3–25. MegaWizard Plug-In Manager - ALT2GXB (TX Analog)*

Table 3–19 describes the available options on page 7 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

*Table 3–19. MegaWizard Plug-In Manager Options (Page 7 for XAUI Mode)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| What is the Transmitter Buffer Power (VCCH)? | This setting is for information only and is used to calculate the $V_{OD}$ from the buffer power supply ($V_{CCH}$) and the transmitter termination to derive the proper $V_{OD}$ range. In XAUI mode, this option is fixed at 1.5 V | Transmitter Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What is the Transmitter Common Mode Voltage ($V_{CM}$)? | The transmitter common mode voltage setting is selectable between 0.6 V and 0.7 V. | Transmitter Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Use external Transmitter termination | This option is available if you use an external termination resistor instead of the on-chip termination OCT. Checking this option turns off the transmitter OCT. | Transmitter Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Select the Transmitter termination resistance | This option selects the transmitter termination value. This option is also used in the calculation of the available $V_{OD}$. | Transmitter Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What is the Voltage Output Differential (VOD) control setting? | This option selects the $V_{OD}$ of the transmitter buffer. The differential output voltage is programmable between 400 mV and 1200 mV in steps of 200 mV. The available $V_{OD}$ settings change based on $V_{CCH}$. | Transmitter Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Pre-emphasis pre-tap setting (% of VOD) | This option is not available in Arria GX devices and is fixed at 0. | Transmitter Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Pre-emphasis first post-tap setting (% of VOD) | This option sets the amount of pre-emphasis on the transmitter buffer using first post-tap. The options available are 0, 1, 2, 3, 4, and 5. | Transmitter Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Pre-emphasis second post-tap setting (% of VOD) | This option is not available in Arria GX devices and is fixed at 0. | — |

Figure 3–26 shows page 8 of the ALT2GXB MegaWizard Plug-In Manager for XAUI mode.

*Figure 3–26. MegaWizard Plug-In Manager - ALT2GXB (Loopback)*

Table 3–20 describes the available options on page 8 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

| *Table 3–20. MegaWizard Plug-In Manager Options (Page 8 for XAUI Mode)* | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Which loopback option would you like? | There are two option available in XAUI mode: no loopback and serial loopback.<br>● No loopback - this is the default mode.<br>● Serial loopback - if you select serial loopback, the `rx_seriallpbken` port is available to control the serial loopback feature dynamically. A 1'b1 enables serial loopback and a 1'b0 disables loopback on a channel-by-channel basis. Altera recommends controlling all four channels simultaneously. A digital reset must be asserted for the transceiver. | Loopback Modes section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Which reverse loopback option would you like? | This option is not available in XAUI mode. | Loopback Modes section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |

Figure 3–27 shows page 9 of the ALT2GXB MegaWizard Plug-In Manager for XAUI mode. If the **Enforce default settings for this protocol** option is selected, this page does not appear in the MegaWizard Plug-In Manager.

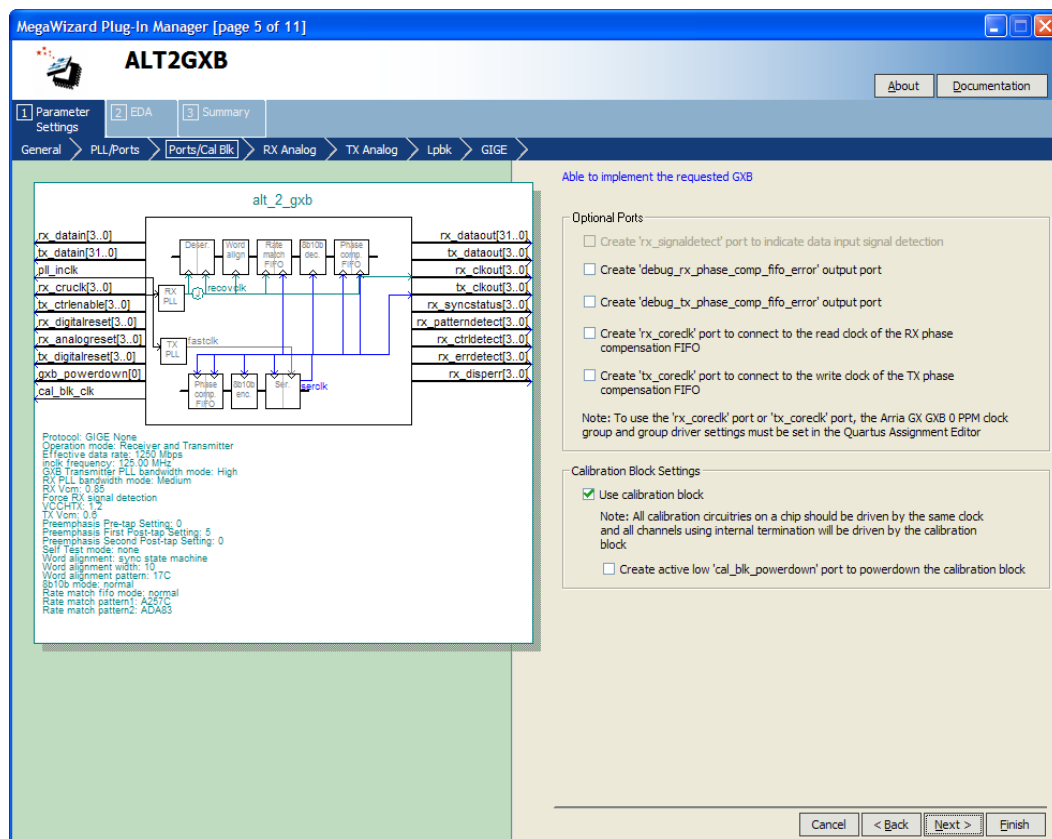*Figure 3–27. MegaWizard Plug-In Manager - ALT2GXB (XAUI)*

Table 3–21 describes the available options on page 9 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

*Table 3–21. MegaWizard Plug-In Manager Options (Page 9 for XAUI Mode)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| Enable run-length violation checking with a run length of | This option activates the run-length violation circuit. You can program the run length at which the circuit triggers the `rx_rlv` signal. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `rx_syncstatus` output port for pattern detector and word aligner | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `rx_patterndetect` port to indicate pattern detected | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `rx_invpolarity` to enable word aligner polarity inversion | This optional port allows you to dynamically reverse the polarity of the received data at the input of the word aligner. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `rx_ctrldetect` port to indicate 8B/10B decoder has detected a control code | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | 8B/10B Decoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `rx_errdetect` port to indicate 8B/10B decoder has detected an error code | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | 8B/10B Decoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `rx_disperr` port to indicate 8B/10B decoder has detected a disparity error | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | 8B/10B Decoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `tx_invpolarity` to allow Transmitter polarity inversion | This optional port allows you to dynamically reverse the polarity of the data to be transmitted at the transmitter PCS-PMA interface. | 8B/10B Encoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |

Figure 3–28 shows page 10 of the MegaWizard Plug-In Manager for the XAUI protocol selection. The **Generate simulation model** option creates a behavioral model (**.vo** or **.vho**) of the transceiver instance for third-party simulators. The **Generate Netlist** option generates a netlist for the third party EDA synthesis tool to be able to estimate timing and resource utilization for the ALT2GXB instance.

*Figure 3–28. MegaWizard Plug-In Manager - ALT2GXB (EDA)*

Figure 3–29 shows page 11 (the last page) of the MegaWizard Plug-In Manager for the XAUI protocol set up. You can select optional files on this page. After you make your selections, click **Finish** to generate the files.

*Figure 3–29. MegaWizard Plug-In Manager - ALT2GXB (Summary)*



## GIGE Mode

This section provides descriptions of the options available on the individual pages of the ALT2GXB MegaWizard Plug-In Manager for GIGE mode. The MegaWizard Plug-In Manager provides a warning if any of the settings you choose are illegal.

Figure 3–30 shows page 3 of the ALT2GXB MegaWizard Plug-In Manager for GIGE mode.

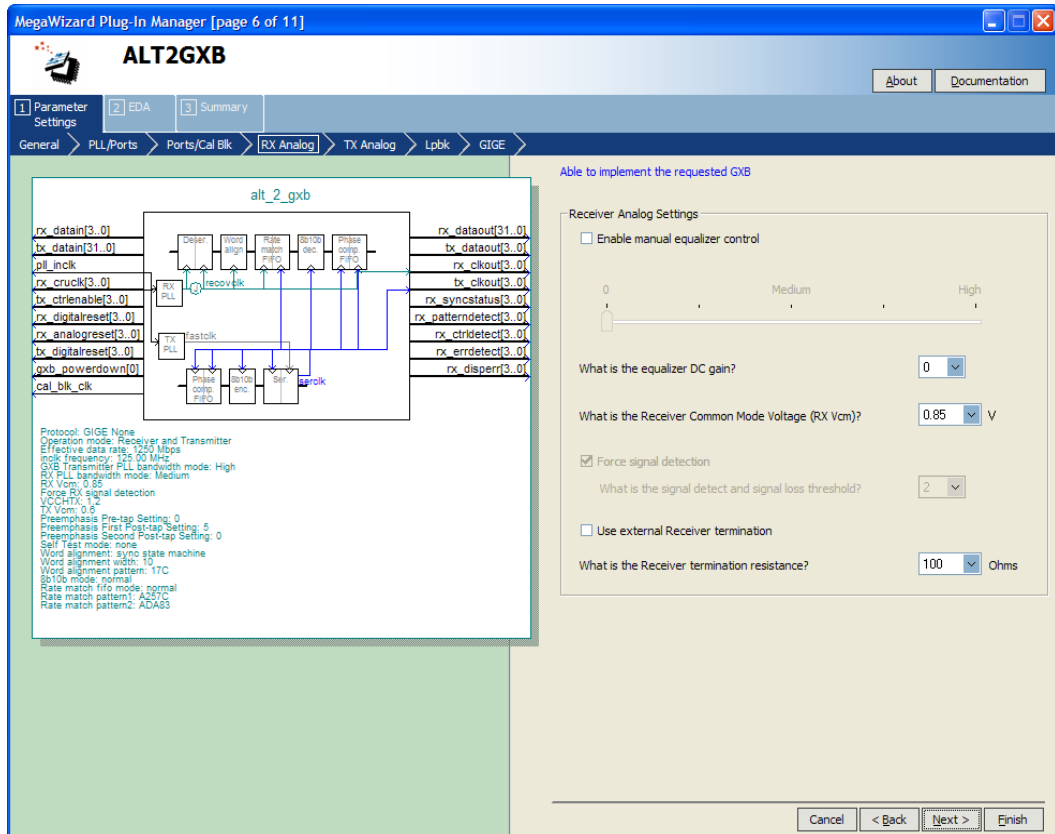*Figure 3–30. MegaWizard Plug-In Manager - ALT2GXB (General)*

Table 3–22 describes the available options on page 3 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

*Table 3–22. MegaWizard Plug-In Manager Options (Page 3 for GIGE Mode)   (Part 1 of 2)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| Which protocol will you be using? | Determines the specific protocol or modes under which the transceiver operates. For GIGE mode, you must select the **GIGE**. | — |
| Which subprotocol will you be using? | The options available here are:<br>● **None**: Select this option for GIGE mode, when UNH-IOL Compliance is not required.<br>● **GIGE-Enhanced**: Select this option when your system implementation has Auto-Negotiation phase or if either /K28.1/, /K28.7/ code group is used in the synchronization ordered set /K/D/.<br><br>Selecting the GIGE-Enhanced mode enables 7-bit word alignment mode and Rate matcher insertion/deletion of C1/C2 configuration ordered sets as required by the Auto negotiation test suite used for UNH-IOL compliance. Three additional output ports: rx_runningdisp, rx_rmfifodatainserted and rx_rmfifodatadeleted are also enabled automatically when this option is selected. | — |
| Enforce default settings for this protocol | Selecting this option skips the **GIGE** screen of the GIGE MegaWizard Plug-In Manager. The **GIGE** screen allows you to select the GIGE-specific ports for your design. If you select this option, all GIGE-specific ports are used. | — |
| What is the operation mode? | The transmitter only and receiver and transmitter (full duplex) modes are allowed in GIGE protocol. The receiver only mode is not available. | — |
| What is the number of channels? | This selects how many duplicate channels this ALT2GXB instance contains. In GIGE mode, the number of channels increments by 1. | — |
| What is the deserializer block width? | This option is unavailable in GIGE mode. | — |
| What is the channel width? | This option determines the PLD-transceiver interface width. Only 8-bit interface width is supported. | Byte Serializer and Byte Deserializer sections in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| What would you like to base the setting on? | This option is unavailable because the data rate is fixed at 1250 Mbps for GIGE mode. | — |

| _Table 3–22. MegaWizard Plug-In Manager Options (Page 3 for GIGE Mode)   (Part 2 of 2)_ | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| What is the data rate? | This option is unavailable because the data rate is fixed at 1250 Mbps for GIGE mode. | — |
| What is the input clock frequency? | Determines the input reference clock frequency for the transceiver. In GIGE mode, input reference clock frequencies of 62.5 MHz and 125 MHz are supported. | GIGE Mode section in the _Arria GX Transceiver Protocol Support and Additional Features_ chapter in volume 2 of the _Arria GX Device Handbook_ |
| What is the data rate division factor? | This option is unavailable in GIGE mode. | — |

Figure 3–31 shows page 4 of the ALT2GXB MegaWizard Plug-In Manager for GIGE mode.

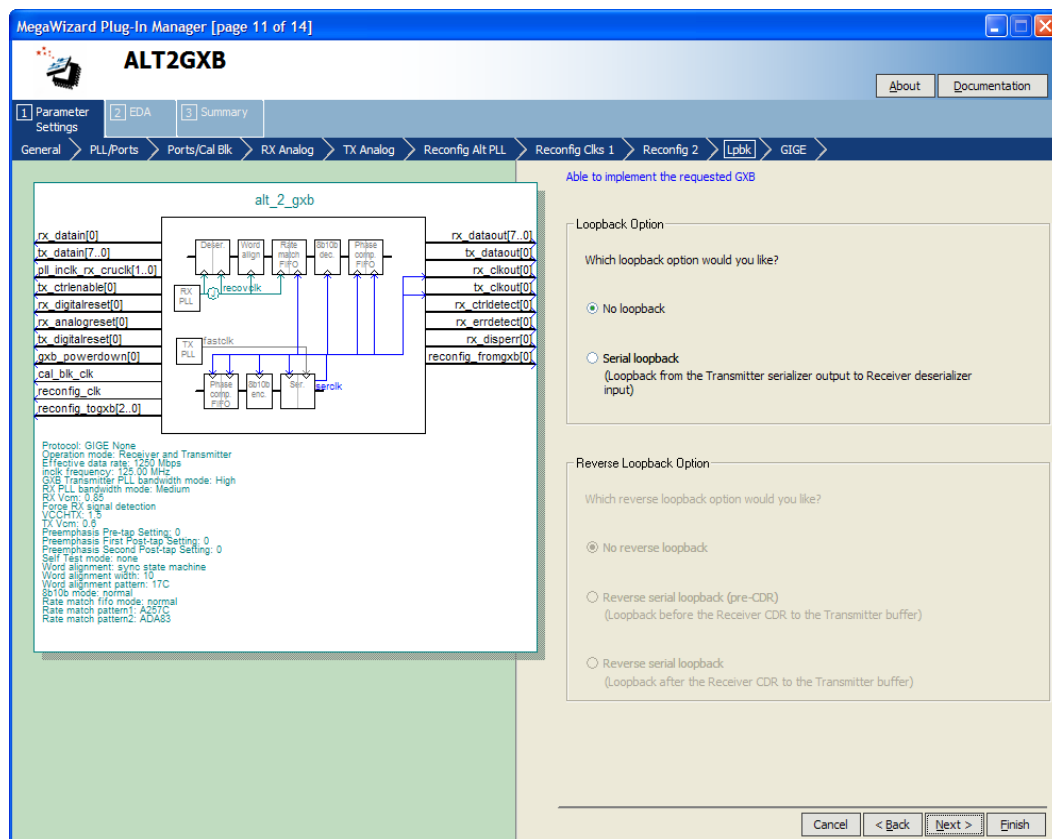*Figure 3–31. MegaWizard Plug-In Manager - ALT2GXB (PLL/Ports)*

Table 3–23 describes the available options on page 4 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

| Table 3–23. MegaWizard Plug-In Manager Options (Page 4 for GIGE Mode)   (Part 1 of 3) | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Train Receiver PLL clock from PLL_inclk | If you select this option, the transmitter input reference clock (pll_inclk) drives the receiver PLL input reference clock also.<br>If you do not select this option, the signal on the rx_cruclk port drives the receiver PLL input reference clock. | — |
| What is the GXB Transmitter PLL bandwidth mode? | This option is not available in GIGE mode because the transmitter PLL bandwidth is fixed at high. | — |
| What is the Receiver PLL bandwidth mode? | This option is not available in GIGE mode because the receiver PLL bandwidth is fixed at medium. | — |
| What is the acceptable PPM threshold between the Receiver PLL VCO and the CRU clock? | This option determines the PPM difference that affects the automatic receiver CRU switchover between lock-to-data and lock-to-reference. (There are additional factors that affect the CRU's transition.) | Clock Recovery Unit (CRU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create gxb_powerdown port to power down the Quad | This signal can be used to reset and power down all circuits in the transceiver block. It does not power down the REFCLK buffers and reference clock lines. | Reset Control and Power Down section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| Create gxb_enable port to enable the Quad | This signal can be used to enable Arria GX transceiver blocks. If instantiated, this port must be tied to the dedicated gigabit transceiver block enable input pin. | Reset Control and Power Down section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| Create rx_analogreset port for the analog portion of the receiver | Receiver analog reset port. | Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |

*Table 3–23. MegaWizard Plug-In Manager Options (Page 4 for GIGE Mode)   (Part 2 of 3)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| Create `rx_digitalreset` port for the digital portion of the receiver | Receiver digital reset port. Resets the PCS logic of the receiver. Altera recommends using this port to implement the recommended reset sequence. | Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `tx_digitalreset` port for the digital portion of the transmitter | Transmitter digital reset port. Resets the PCS logic of the transmitter. Altera recommends using this port to implement the recommended reset sequence. | Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `pll_locked` port to indicate PLL is in lock with the reference input clock | Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `rx_locktorefclk` port to lock the RX PLL to the reference clock | Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Clock Recovery Unit (CRU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `rx_locktodata` port to lock the RX PLL to the received data | Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Clock Recovery Unit (CRU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `rx_pll_locked` port to indicate RX PLL is in lock with the reference clock | Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |

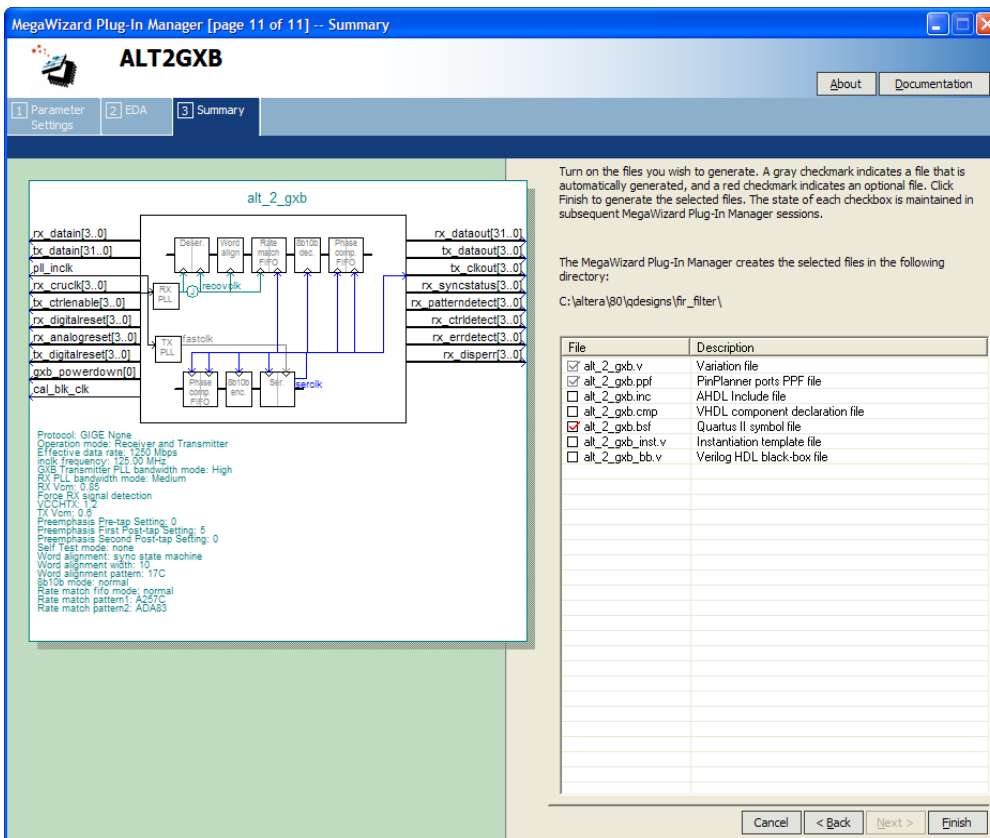| Table 3–23. MegaWizard Plug-In Manager Options (Page 4 for GIGE Mode) (Part 3 of 3) | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Create `rx_freqlocked` port to indicate RX PLL is in lock with the received data | Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Clock Recovery Unit (CRU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `rx_signaldetect` port to indicate data input signal detection | This option is not available in GIGE mode. | — |
| Create `debug_rx_phase_comp_fifo_error` output port | This optional output port indicates Receiver Phase Compensation FIFO overflow/under run condition. Note that no PPM difference is allowed between FIFO read and write clocks. Use this port for debug purposes only. | Receiver Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `debug_tx_phase_comp_fifo_error` output port | This optional output port indicates Transmitter Phase Compensation FIFO overflow/under run condition. Note that no PPM difference is allowed between FIFO read and write clocks. Use this port for debug purposes only. | Transmitter Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `rx_coreclk` port to connect to the read clock of the RX phase compensation FIFO | This optional input port allows you to clock the read side of the Receiver Phase Compensation FIFO with a non-transceiver PLD clock. | PLD-Transceiver Interface Clocking section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `tx_coreclk` port to connect to the write clock of the TX phase compensation FIFO | This optional input port allows you to clock the write side of the Transmitter Phase Compensation FIFO with a non-transceiver PLD clock. | PLD-Transceiver Interface Clocking section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |

Figure 3–31 shows page 5 of the ALT2GXB MegaWizard Plug-In Manager for GIGE mode.

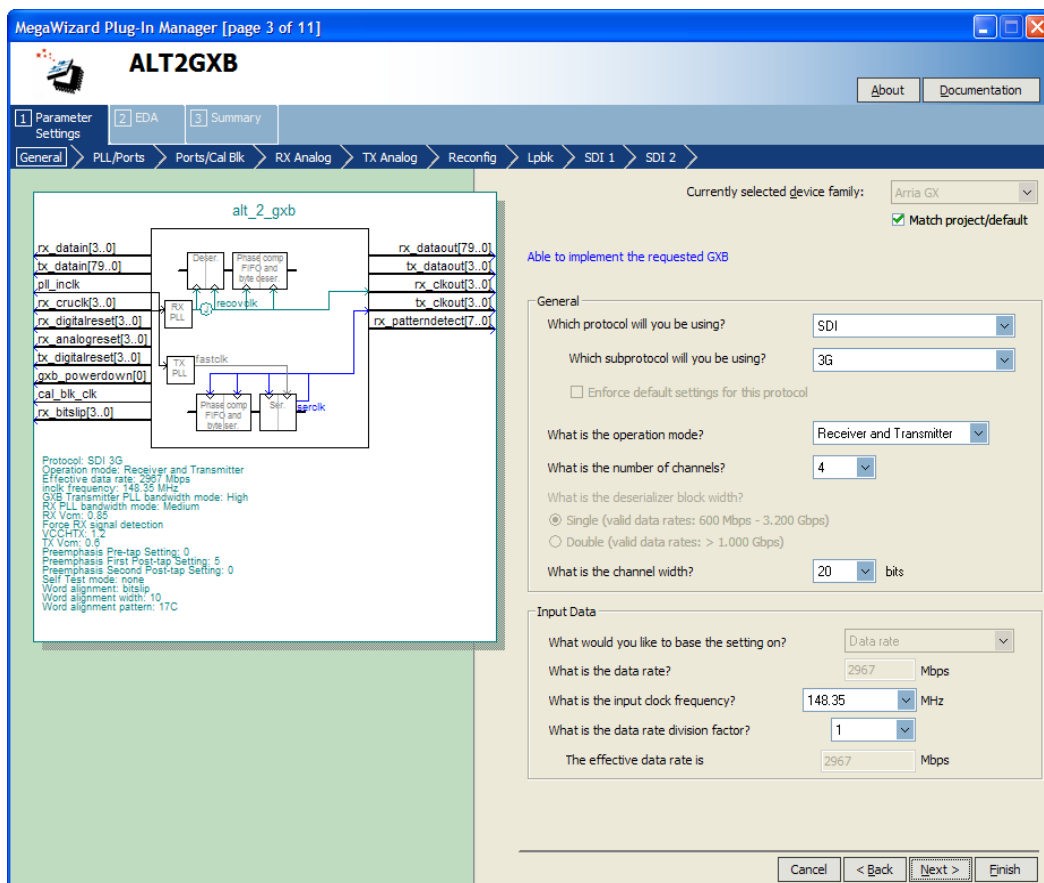*Figure 3–32. MegaWizard Plug-In Manager - ALT2GXB (Ports/Cal Blk)*

Table 3–23 describes the available options on page 5 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

| *Table 3–24. MegaWizard Plug-In Manager Options (Page 5 for GIGE Mode)   (Part 1 of 2)* | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Create rx_signaldetect port to indicate data input signal detection | This option is not available in GIGE mode. | — |
| Create debug_rx_phase_comp_ fifo_error output port | This optional output port indicates Receiver Phase Compensation FIFO overflow/under run condition. Note that no PPM difference is allowed between FIFO read and write clocks. Use this port for debug purposes only. | Receiver Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create debug_tx_phase_comp_ fifo_error output port | This optional output port indicates Transmitter Phase Compensation FIFO overflow/under run condition. Note that no PPM difference is allowed between FIFO read and write clocks. Use this port for debug purposes only. | Transmitter Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create rx_coreclk port to connect to the read clock of the RX phase compensation FIFO | This optional input port allows you to clock the read side of the Receiver Phase Compensation FIFO with a non-transceiver PLD clock. | PLD-Transceiver Interface Clocking section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create tx_coreclk port to connect to the write clock of the TX phase compensation FIFO | This optional input port allows you to clock the write side of the Transmitter Phase Compensation FIFO with a non-transceiver PLD clock. | PLD-Transceiver Interface Clocking section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |

| *Table 3–24. MegaWizard Plug-In Manager Options (Page 5 for GIGE Mode)  (Part 2 of 2)* | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Use calibration block | This option allows you to select which instance of the ALT2GXB megafunction instantiates the calibration block. Only one instance of the ALT2GXB megafunction is required to instantiate the calibration block. | Calibration Block section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create active low `cal_blk_powerdown` to power down the calibration block | Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Calibration Block section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |

Figure 3–33 shows page 6 of the `ALT2GXB` MegaWizard Plug-In Manager for GIGE mode.

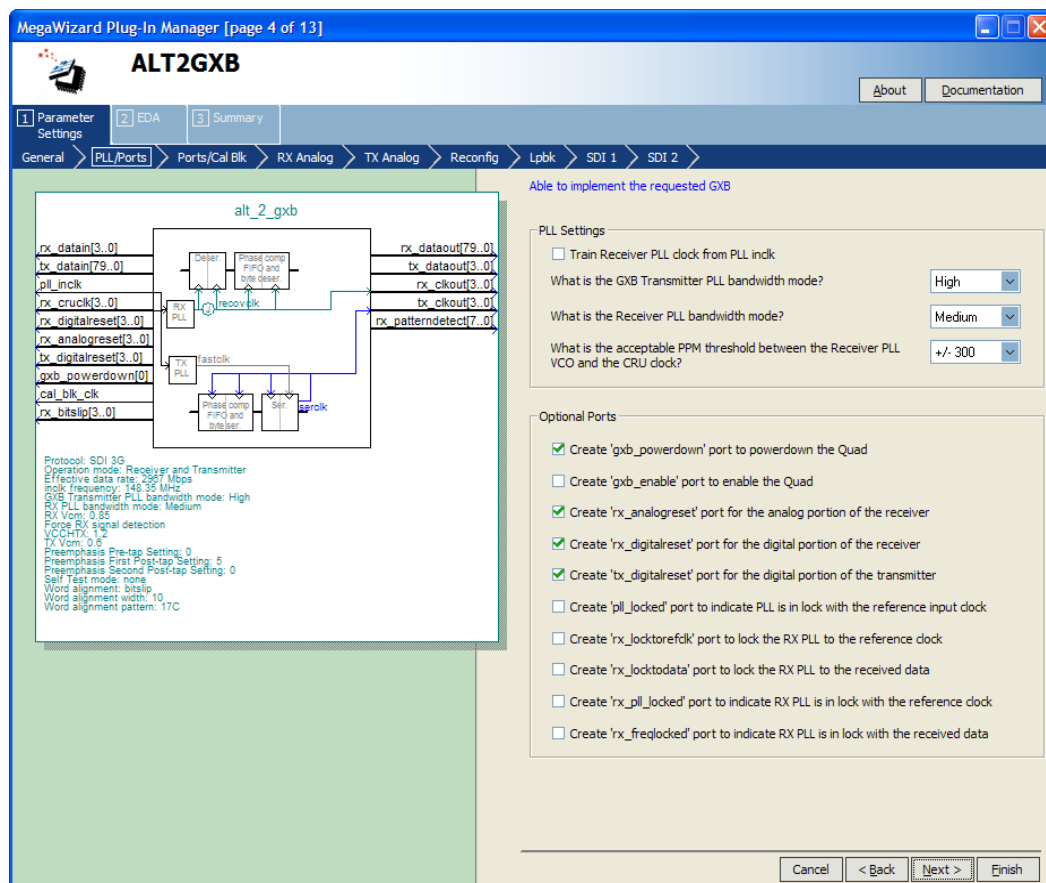*Figure 3–33. MegaWizard Plug-In Manager - ALT2GXB (RX Analog)*

Table 3–25 describes the available options on page 6 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

| *Table 3–25. MegaWizard Plug-In Manager Options (Page 6 for GIGE Mode)* | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Enable manual equalizer control | This option enables the 0–4 setting options for manual equalizer control. | Receiver Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| What is the equalizer DC gain? | This enables the DC gain option and the legal settings are 0, 1, 2, and 3. | Receiver Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| What is the Receiver Common Mode Voltage (RX $V_{CM}$)? | The receiver common mode voltage is programmable. The selections available are 0.85 V and 1.2 V. | Receiver Buffer section under GIGE Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Force signal detection | This option is unavailable in GIGE mode and is always forced selected. | — |
| What is the signal detect and signal loss threshold? | This option is unavailable in GIGE mode as signal detection is forced. | — |
| Use external receiver termination | This option is available if you use an external termination resistor instead of the on-chip termination OCT. If checked, this option turns off the receiver OCT. | — |
| What is the Receiver termination resistance? | In GIGE mode, the only supported receiver termination resistance is 100 Ω | Receiver Buffer section under GIGE Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |

Figure 3–34 shows page 7 of the ALT2GXB MegaWizard Plug-In Manager for GIGE mode.

*Figure 3–34. MegaWizard Plug-In Manager - ALT2GXB (TX Analog)*

Table 3–26 describes the available options on page 7 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

**Table 3–26. MegaWizard Plug-In Manager Options (Page 7 for GIGE Mode)**

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| What is the Transmitter Buffer Power (VCCH)? | In GIGE mode, the transmitter buffer power can be either 1.2 V or 1.5 V. You must connect the $V_{CCH}$ power pins of a GIGE transceiver bank to a 1.2 V or 1.5 V power supply. You must select 1.2 V PCML or 1.5 V PCML I/O standard for the transmitter data output pins. | Transmitter Buffer section under GIGE Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| What is the Transmitter Common Mode Voltage ($V_{CM}$)? | In GIGE mode, the transmitter common mode voltage is selectable between 0.6 V and 0.7 V. | Transmitter Buffer section under GIGE Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Use external Transmitter termination | This option is available if you use an external termination resistor instead of the on-chip termination OCT. Checking this option turns off the transmitter OCT. | — |
| Select the Transmitter termination resistance | In GIGE mode, the only supported receiver termination resistance is 100 Ω | Transmitter Buffer section under GIGE Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| What is the Voltage Output Differential (VOD) control setting? | This option selects the $V_{OD}$ of the transmitter buffer. The differential output voltage is programmable between 400 mV to 1200 mV in steps of 200 mV. The available $V_{OD}$ settings change based on $V_{CCH}$. | Transmitter Buffer section under GIGE Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Pre-emphasis pre-tap setting (% of VOD) | This option is not available in Arria GX devices and is fixed at 0. | — |

| Table 3–26. MegaWizard Plug-In Manager Options (Page 7 for GIGE Mode) | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Pre-emphasis first post-tap setting (% of VOD) | This option sets the amount of pre-emphasis on the transmitter buffer using first post-tap. The options available are 0, 1, 2, 3, 4, and 5. | Transmitter Buffer section under GIGE Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Pre-emphasis second post-tap setting (% of VOD) | This option is not available in Arria GX devices and is fixed at 0. | — |

Figure 3–35 shows page 8 of the `ALT2GXB` MegaWizard Plug-In Manager for GIGE mode.

*Figure 3–35. MegaWizard Plug-In Manager - ALT2GXB (Loopback)*

Table 3–27 describes the available options on page 8 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

| Table 3–27. MegaWizard Plug-In Manager Options (Page 8 for GIGE Mode) | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Which loopback option would you like? | No loopback and serial loopback options are available in GIGE mode.<br>No loopback is the default mode. If you select serial loopback, the rx_seriallpbken port is available to control the serial loopback feature dynamically. A 1'b1 enables serial loopback and a 1'b0 disables loopback on a channel-by-channel basis. | Loopback Modes section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |

Figure 3–36 shows page 9 of the ALT2GXB MegaWizard Plug-In Manager for GIGE mode. If the **Enforce default settings for this protocol** option is selected, this page does not appear in the MegaWizard Plug-In Manager.

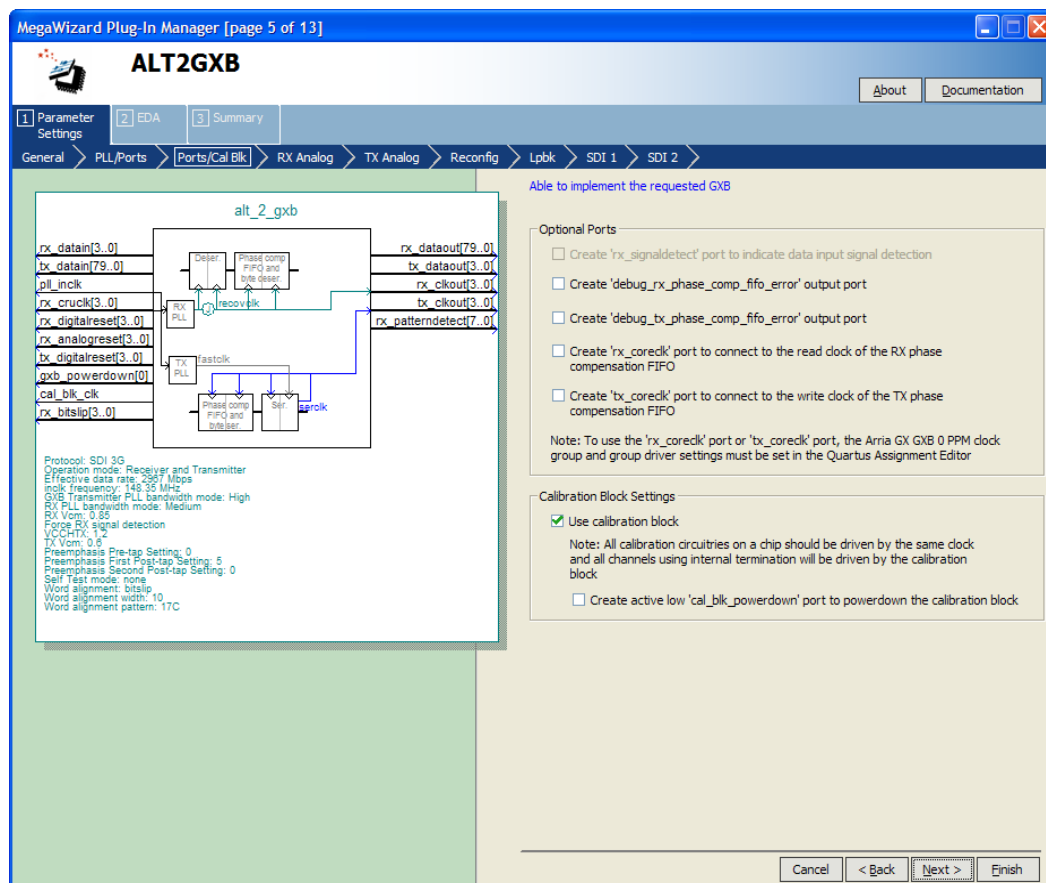*Figure 3–36. MegaWizard Plug-In Manager - ALT2GXB (GIGE)*
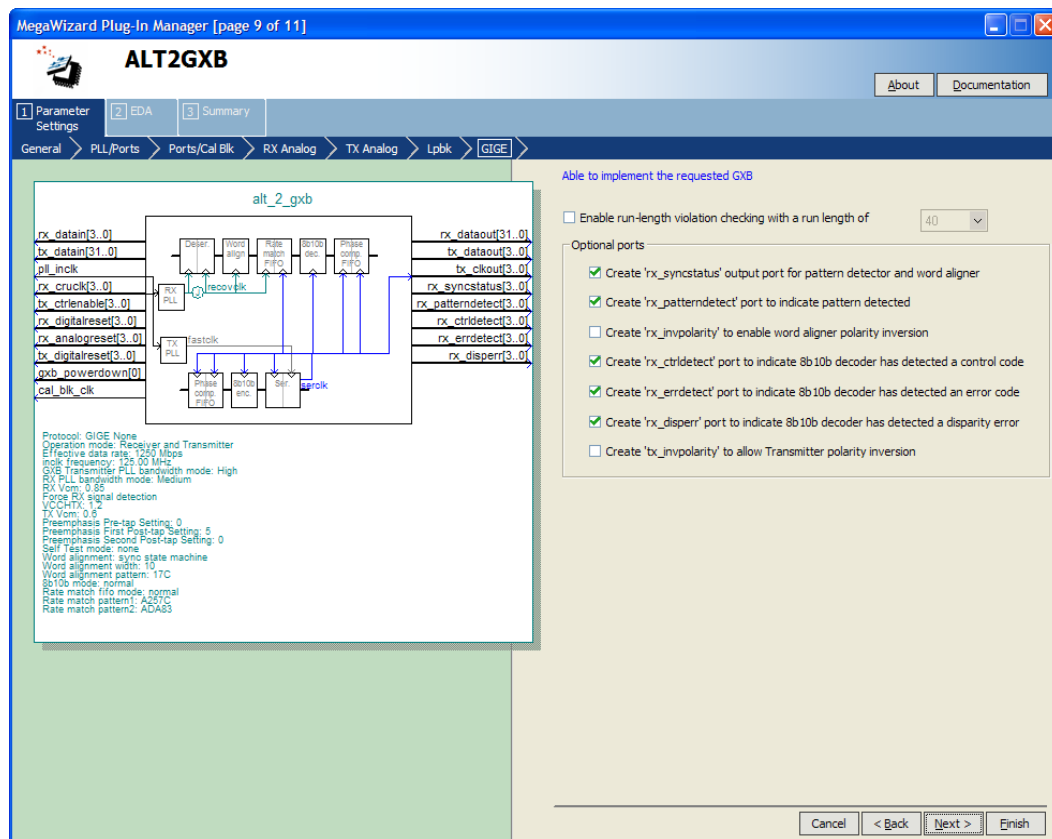
Table 3–28 describes the available options on page 9 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

| Table 3–28. MegaWizard Plug-In Manager Options (Page 9 for GIGE Mode) | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Enable run-length violation checking with a run length of | This option activates the run-length violation circuit. You can program the run length at which the circuit triggers the rx_rlv signal. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create rx_syncstatus output port for pattern detector and word aligner | Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Word Aligner section under GIGE mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create rx_patterndetect output port to indicate pattern detected | Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Word Aligner section under GIGE Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create rx_invpolarity to enable word aligner polarity inversion | This optional port allows you to dynamically reverse the polarity of the received data at the input of the word aligner. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create rx_ctrldetect output port to indicate 8B/10B decoder has detected a control code | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | 8B/10B Decoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create rx_errdetect port to indicate 8B/10B decoder has detected an error code | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | 8B/10B Decoder section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |

| Table 3–28. MegaWizard Plug-In Manager Options (Page 9 for GIGE Mode) | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Create `rx_disperr` port to indicate 8B/10B decoder has detected a disparity error | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | 8B/10B Decoder section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `tx_invpolarity` to allow Transmitter polarity inversion | This optional port allows you to dynamically reverse the polarity of the data to be transmitted at the transmitter PCS-PMA interface. | 8B/10B Encoder section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |

Figure 3–37 shows page 10 of the MegaWizard Plug-In Manager for the GIGE protocol selection. The **Generate simulation model** option creates a behavioral model (**.vo** or **.vho**) of the transceiver instance for third-party simulators. The **Generate a netlist for synthesis area and timing estimation** option creates a netlist file (**.syn**) for third-party synthesis tools.

*Figure 3–37. MegaWizard Plug-In Manager - ALT2GXB (EDA)*

Figure 3–38 shows page 11 (the last page) of the MegaWizard Plug-In Manager for the GIGE protocol set up. You can select optional files on this page. After you make your selections, click **Finish** to generate the files.

*Figure 3–38. MegaWizard Plug-In Manager - ALT2GXB (Summary)*



## SDI Mode

This section provides descriptions of the options available on the individual pages of the ALT2GXB MegaWizard Plug-In Manager for SDI mode. The MegaWizard Plug-In Manager provides a warning if any of the settings you choose are illegal.

Figure 3–39 shows page 3 of the ALT2GXB MegaWizard Plug-In Manager for SDI mode.

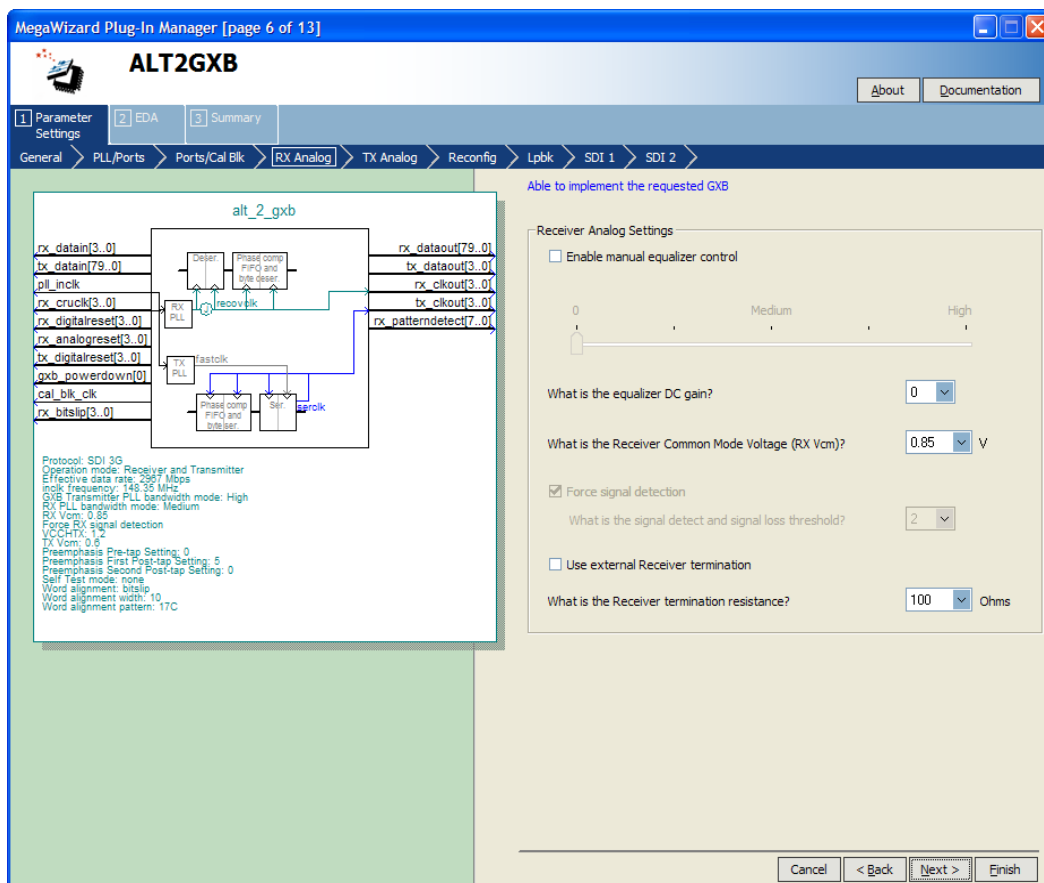*Figure 3–39. MegaWizard Plug-In Manager - ALT2GXB (General)*

Table 3–29 describes the available options on page 3 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

*Table 3–29. MegaWizard Plug-In Manager Options (Page 3 for SDI Mode)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| Which protocol will you be using? | Determines the specific protocol or modes under which the transceiver operates. For SDI mode, you must select the SDI protocol. | — |
| Which subprotocol will you be using? | In SDI mode, the two available subprotocols are:<br>● 3G: third-generation (3 Gbps) SDI at 2970 Mbps or 2967 Mbps<br>● HD: high-definition SDI at 1485 Mbps or 1483.5 Mbps | — |
| Enforce default settings for this protocol | This option is not available in SDI mode. | — |
| What is the operation mode? | The transmitter only, receiver only, and receiver and transmitter (full duplex) modes are allowed in SDI protocol. | — |
| What is the number of channels? | This selects how many duplicate channels this ALT2GXB instance contains. | — |
| What is the deserializer block width? | SDI mode only operates in single-width mode. Double-width mode is not supported. | — |
| What is the channel width? | This option determines the transceiver-to-PLD interface width. In SDI mode, 10-bit and 20-bit channel widths are allowed. In 10-bit configuration, the byte serializer is not used. In 20-bit configuration, the byte serializer is used. | *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What would you like to base the setting on? | This option not available in SDI mode. | — |
| What is the data rate? | This field is automatically set based on the subprotocol (3G or HD) and the input clock frequency selection. | — |
| What is the input clock frequency? | Four input reference clock options are available, depending on the subprotocol (3G or HD).<br>● For 3G subprotocol, the available options are: 148.5 MHz and 297 MHz for 2970 Mbps data rate and 148.35 MHz and 296.7 MHz for 2967 Mbps data rate<br>● For HD subprotocol, the available option are: 74.25 MHz and 148.5 MHz for 1485 Mbps data rate and 74.175 MHz and 148.35 MHz for 1483.5 Mbps data rate | — |
| What is the data rate division factor? | This option is not available in SDI Mode. | — |

Figure 3–40 shows page 4 of the ALT2GXB MegaWizard Plug-In Manager for SDI mode.

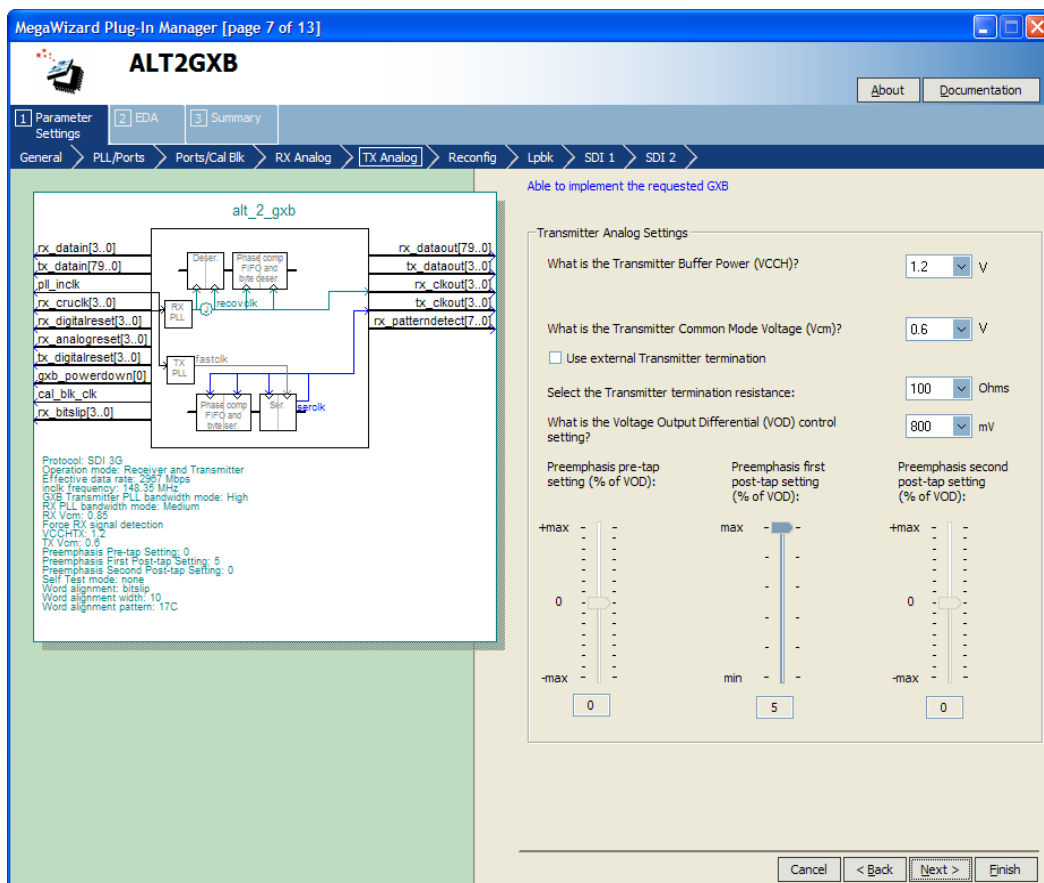*Figure 3–40. MegaWizard Plug-In Manager - ALT2GXB (PLL/Ports)*

Table 3–30 describes the available options on page 4 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

*Table 3–30. MegaWizard Plug-In Manager Options (Page 4 for SDI Mode)  (Part 1 of 3)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| Train Receiver PLL clock from PLL inclk | If you turn this option on, your design uses the input reference clock to the transmitter PLL to train the receiver PLL. This reduces the need to supply a separate receiver PLL reference clock. | Clock Recovery Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What is the GXB Transmitter PLL bandwidth mode? | Three available bandwidth options are high, medium, and low. The default transmitter PLL bandwidth is high. | Clock Multiplier Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What is the Receiver PLL bandwidth mode? | Three available bandwidth options are high, medium, and low. The default receiver PLL bandwidth is medium. | Clock Recovery Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What is the acceptable PPM threshold between the Receiver PLL VCO and the CRU clock? | This option determines the PPM difference that affects the automatic receiver CRU switchover between lock-to-data and lock-to-reference. (There are additional factors that affect CRU's transition.) | Clock Recovery Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `gxb_powerdown` port to power down the Quad | This signal can be used to reset and power down all circuits in the transceiver block. It does not power down the REFCLK buffers and reference clock lines. | Reset Control and Power Down section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| Create `gxb_enable` port to enable the Quad | This signal can be used to enable Arria GX transceiver blocks. If instantiated, this port must be tied to the dedicated gigabit transceiver block enable input pin. | Reset Control and Power Down section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| Create `rx_analogreset` port for the analog portion of the receiver | Receiver analog reset port. | Reset Control and Power Down section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |

| Table 3–30. MegaWizard Plug-In Manager Options (Page 4 for SDI Mode)   (Part 2 of 3) | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Create `rx_digitalreset` port for the digital portion of the receiver | Receiver digital reset port. Resets the PCS portion of the receiver. Altera recommends using this port along with logic to implement the recommended reset sequence. | Reset Control and Power Down section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `tx_digitalreset` port for the digital portion of the receiver | Transmitter digital reset port. Resets the PCS portion of the transmitter. Altera recommends using this port along with logic to implement the recommended reset sequence. | Reset Control and Power Down section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `pll_locked` port to indicate PLL is in lock with the reference input clock | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Clock Multiplier Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `rx_locktorefclk` port to lock the RX PLL to the reference clock | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Clock Recovery Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `rx_locktodata` port to lock the RX PLL to the received data | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Clock Recovery Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `rx_pll_locked` port to indicate RX PLL is in lock with the reference clock | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Clock Recovery Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `rx_freqlocked` port to indicate RX PLL is in lock with the received data | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Clock Recovery Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |

*Table 3–30. MegaWizard Plug-In Manager Options (Page 4 for SDI Mode)   (Part 3 of 3)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| Create `rx_signaldetect` port to indicate data input signal detection | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Receiver Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `debug_rx_phase_comp_fifo_error` output port | This optional output port indicates Receiver Phase Compensation FIFO overflow/under run condition. Note that no PPM difference is allowed between FIFO read and write clocks. Use this port for debug purposes only. | Receiver Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `debug_tx_phase_comp_fifo_error` output port | This optional output port indicates Transmitter Phase Compensation FIFO overflow/under run condition. Note that no PPM difference is allowed between FIFO read and write clocks. Use this port for debug purposes only. | Transmitter Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `rx_coreclk` port to connect to the read clock of the RX phase compensation FIFO | This optional input port allows you to clock the read side of the Receiver Phase Compensation FIFO with a non-transceiver PLD clock. | Transceiver Clocking section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `tx_coreclk` port to connect to the write clock of the TX phase compensation FIFO | This optional input port allows you to clock the write side of the Transmitter Phase Compensation FIFO with a non-transceiver PLD clock. | Transceiver Clocking section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |

Figure 3–40 shows page 5 of the ALT2GXB MegaWizard Plug-In Manager for SDI mode.

*Figure 3–41. MegaWizard Plug-In Manager - ALT2GXB (Ports/Cal Blk)*

Table 3–30 describes the available options on page 5 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

*Table 3–31. MegaWizard Plug-In Manager Options (Page 5 for SDI Mode)   (Part 1 of 2)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| Create `rx_signaldetect` port to indicate data input signal detection | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Receiver Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `debug_rx_phase_comp_fifo_error` output port | This optional output port indicates Receiver Phase Compensation FIFO overflow/under run condition. Note that no PPM difference is allowed between FIFO read and write clocks. Use this port for debug purposes only. | Receiver Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `debug_tx_phase_comp_fifo_error` output port | This optional output port indicates Transmitter Phase Compensation FIFO overflow/under run condition. Note that no PPM difference is allowed between FIFO read and write clocks. Use this port for debug purposes only. | Transmitter Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `rx_coreclk` port to connect to the read clock of the RX phase compensation FIFO | This optional input port allows you to clock the read side of the Receiver Phase Compensation FIFO with a non-transceiver PLD clock. | Transceiver Clocking section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `tx_coreclk` port to connect to the write clock of the TX phase compensation FIFO | This optional input port allows you to clock the write side of the Transmitter Phase Compensation FIFO with a non-transceiver PLD clock. | Transceiver Clocking section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |

| Table 3–31. MegaWizard Plug-In Manager Options (Page 5 for SDI Mode) (Part 2 of 2) | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Use calibration block | This option allows you to select which instance of the ALT2GXB megafunction instantiates the calibration block. Only one instance of the ALT2GXB megafunction is required to instantiate the calibration block. | Calibration Blocks section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Create `cal_blk_powerdown` to power down the calibration block | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Calibration Blocks section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |

Figure 3–42 shows page 6 of the ALT2GXB MegaWizard Plug-In Manager for SDI mode.

*Figure 3–42. MegaWizard Plug-In Manager - ALT2GXB (RX Analog)*

Table 3–32 describes the available options on page 6 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

| Table 3–32. MegaWizard Plug-In Manager Options (Page 6 for SDI Mode) | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Enable static equalizer control | This option enables the 0–4 setting options for manual equalizer control. | Receiver Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What is the equalizer DC gain? | This enables the DC gain option. The legal settings are 0, 1, 2, 3. | Receiver Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What is the Receiver Common Mode Voltage (RX $V_{CM}$)? | The receiver common mode voltage is programmable. The selections available are 0.85 V and 1.2 V. | Receiver Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Force signal detection | This option is not available in SDI mode. | Receiver Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What is the signal detect and signal loss threshold? | This option is not available in SDI mode. | Receiver Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Use external receiver termination | This option is available if you want to use an external termination resistor instead of the on-chip termination OCT. If checked, this option turns off the receiver OCT. | Receiver Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What is the receiver termination resistance? | This option selects the receiver termination value. The receiver termination value is fixed at 100 $\Omega$ in Arria GX devices. | Receiver Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |

Figure 3–43 shows page 7 of the ALT2GXB MegaWizard Plug-In Manager for SDI mode.

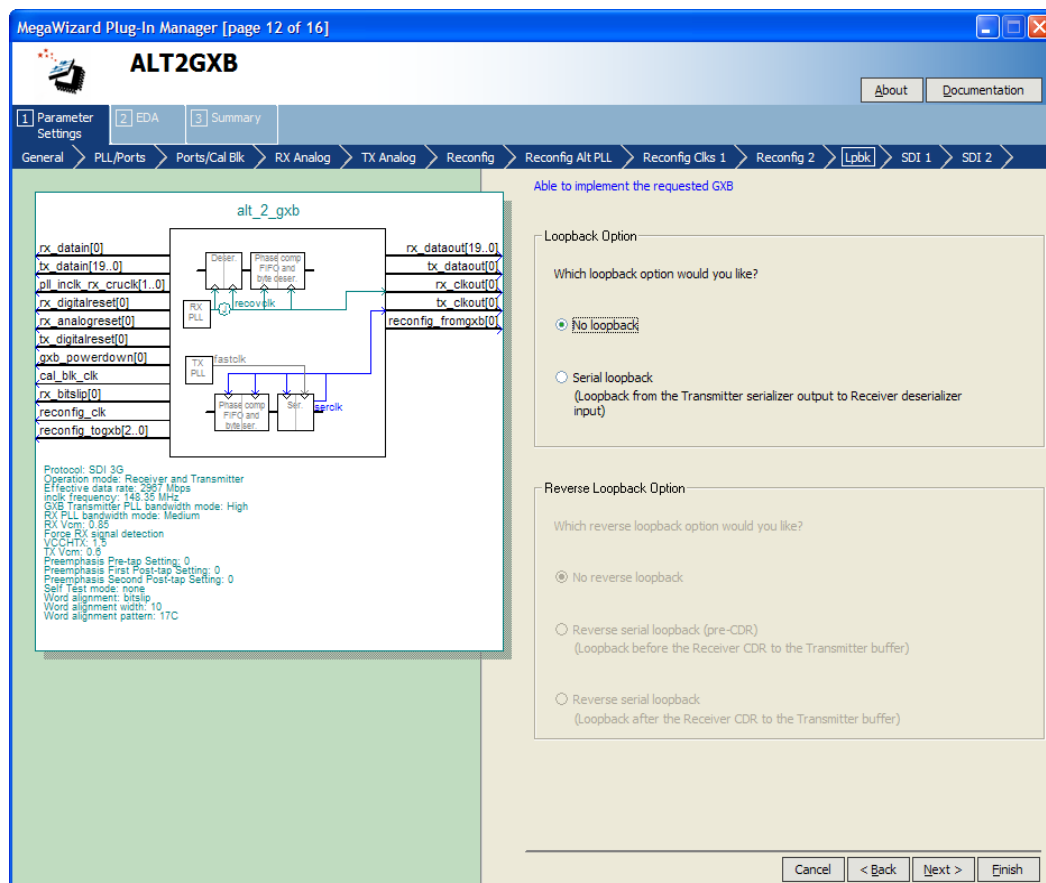*Figure 3–43. MegaWizard Plug-In Manager - ALT2GXB (TX Analog)*

Table 3–33 describes the available options on page 7 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

**Table 3–33. MegaWizard Plug-In Manager Options (Page 7 for SDI Mode)**

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| What is the Transmitter Buffer Power (VCCH)? | This setting is for information only and is used to calculate the $V_{OD}$ from the buffer power supply ($V_{CCH}$) and the transmitter termination to derive the proper $V_{OD}$ range. In SDI mode, this option is fixed at 1.5 V. | Transmitter Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What is the Transmitter Common Mode Voltage ($V_{CM}$)? | The transmitter common mode voltage setting is selectable between 0.6 V and 0.7 V. | Transmitter Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Use external Transmitter termination | This option is available if you use an external termination resistor instead of the on-chip termination OCT. Checking this option turns off the transmitter OCT. | Transmitter Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Select the Transmitter termination resistance | This option selects the transmitter termination value. This option is also used in the calculation of the available $V_{OD}$. | Transmitter Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What is the Voltage Output Differential (VOD) control setting? | This option selects the $V_{OD}$ of the transmitter buffer. The differential output voltage is programmable between 400 mV and 1200 mV in steps of 200 mV. The available $V_{OD}$ settings change based on $V_{CCH}$. | Transmitter Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Pre-emphasis pre-tap setting (% of VOD) | This option is not available in Arria GX devices and is fixed at 0. | |
| Pre-emphasis first post-tap setting (% of VOD) | This option sets the amount of pre-emphasis on the transmitter buffer using first post-tap. The options available are 0, 1, 2, 3, 4, and 5. | Transmitter Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| Pre-emphasis second post-tap setting (% of VOD) | This option is not available in Arria GX devices and is fixed at 0. | |

Figure 3–44 shows page 8 of the ALT2GXB MegaWizard Plug-In Manager for SDI mode.

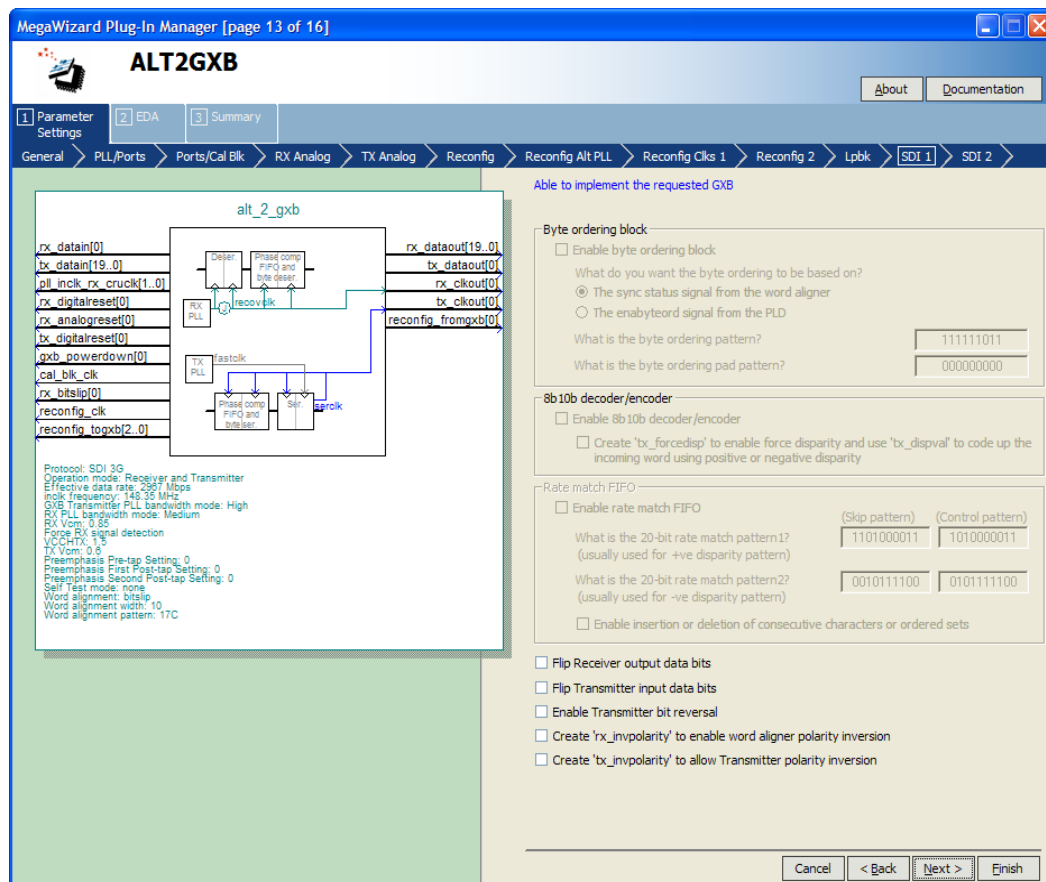*Figure 3–44. MegaWizard Plug-In Manager - ALT2GXB (Reconfig)*
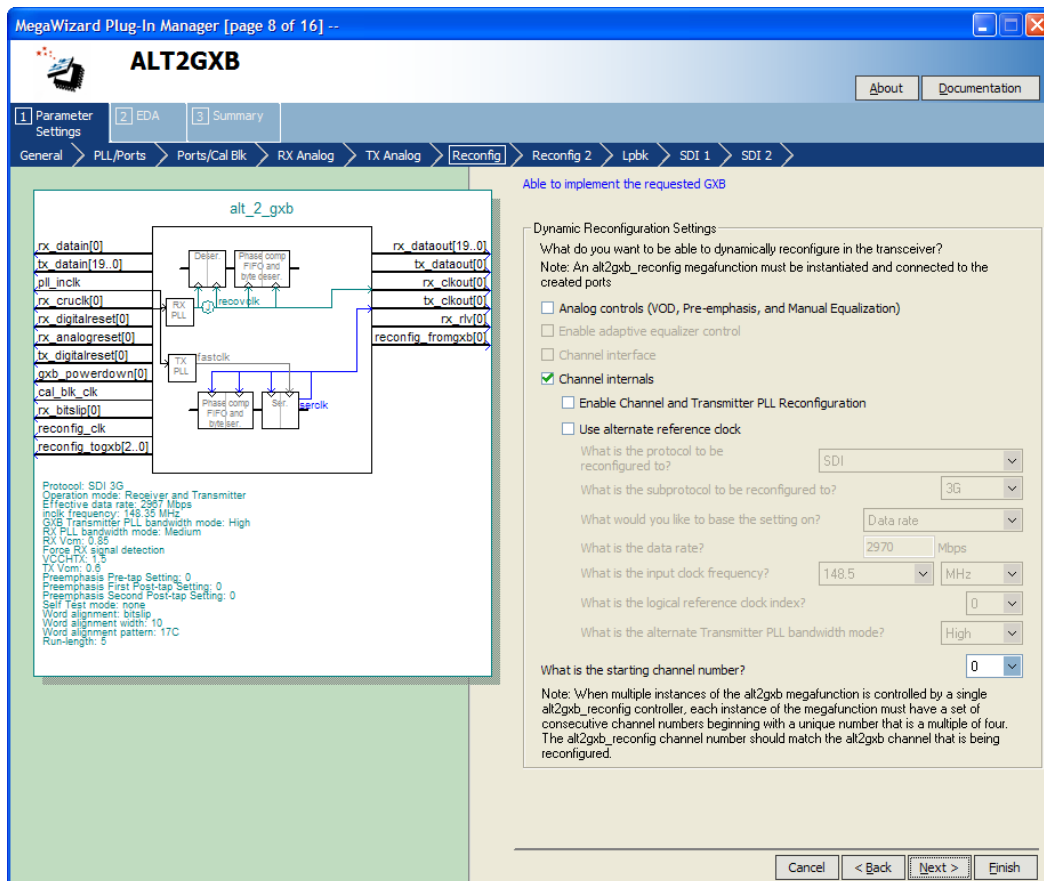
Table 3–34 describes the available options on page 8 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

| Table 3–34. MegaWizard Plug-In Manager Options (Page 8 for SDI Mode) | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| What do you want to be able to dynamically reconfigure in the transceiver? | Available options are:<br>● Analog controls: Dynamically reconfigures the PMA control settings like Vod, Pre-emphasis, Equalization, etc.<br>● Channel Internals: Enables MIF-based reconfiguration among modes that have different data paths within the channel but same PLD interface signals. When this option is enabled, two mutually exclusive options, **Enable Channel and Transmitter PLL Reconfiguration** and **Use alternate reference clock**, are available. | *Stratix II GX Dynamic Reconfiguration* chapter in volume 2 of the Stratix II GX Device Handbook. |
| What is the starting channel number? | The range for the dynamic reconfiguration starting channel number setting is 0—156, in multiples of 4. It is in multiples of 4 because the dynamic reconfiguration interface is per transceiver block. The range of 0—156 is the logical channel address, based purely on the number of possible ALT2GXB instances. | *Stratix II GX Dynamic Reconfiguration* chapter in volume 2 of the Stratix II GX Device Handbook. |

Figure 3–45 shows page 9 of the ALT2GXB MegaWizard Plug-In Manager for SDI mode. This page appears only when the E**nable Channel and Transmitter PLL Reconfiguration** option is selected in the Reconfig page (Page 8).

*Figure 3–45. MegaWizard Plug-In Manager - ALT2GXB (Reconfig Alt PLL)*
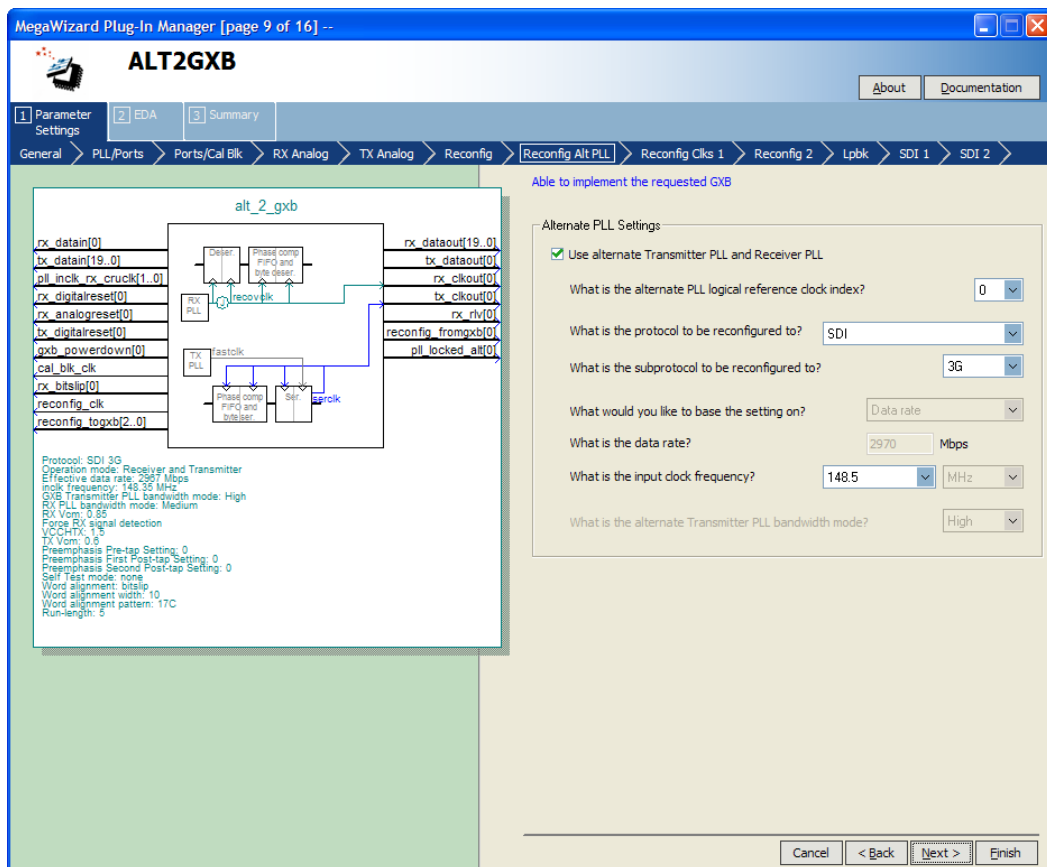


Table 3–35 describes the available options on page 9 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

*Table 3–35. MegaWizard Plug-In Manager Options (Page 9 for SDI Mode)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| Use alternate Transmitter PLL and Receiver PLL | Selecting this option sets up the transmitter channel to listen to one of the two PLLs in its transceiver block. The information regarding which PLL it listens to is stored in the MIF. | *Stratix II GX Dynamic Reconfiguration* chapter in volume 2 of the Stratix II GX Device Handbook. |

Figure 3–46 shows page 10 of the ALT2GXB MegaWizard Plug-In Manager for SDI mode. This page appears only when the E**nable Channel and Transmitter PLL Reconfiguration** option is selected in the Reconfig page (Page 8).

*Figure 3–46. MegaWizard Plug-In Manager - ALT2GXB (Reconfig Clks 1)*
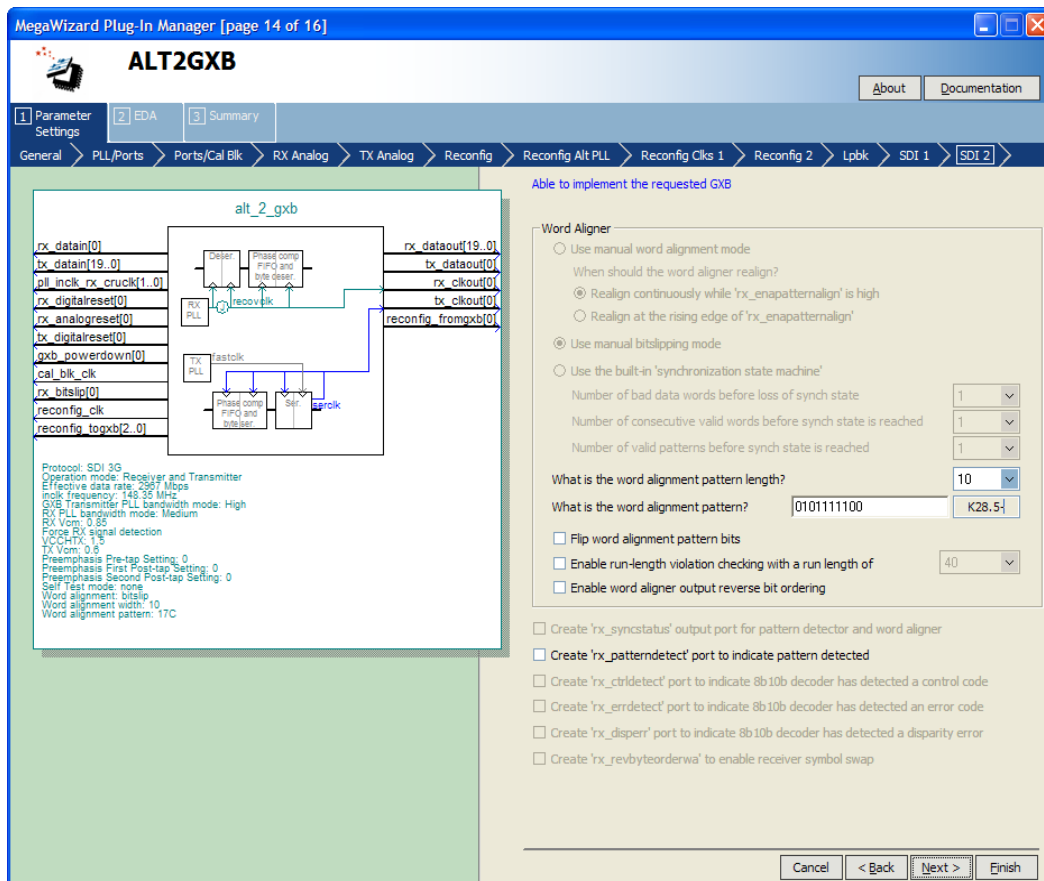
**Arria GX ALT2GXB Megafunction User Guide**


Table 3–36 describes the available options on page 10 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

**Table 3–36. MegaWizard Plug-In Manager Options (Page 10 for SDI Mode)**

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| What is the main PLL logical reference clock index? | This option allows you to select the logical index for the PLL that you intend to use with the current configuration. This option is meaningful only if you select the **Use alternate Transmitter PLL and Receiver PLL** option on the **Reconfig Alt PLL** page. | *Stratix II GX Dynamic Reconfiguration* chapter in volume 2 of the Stratix II GX Device Handbook. |
| How many input clocks? | This field allows you to select the number of reference clock inputs needed to meet your **CMU PLL reconfiguration** design goals. A maximum of five input reference clocks are allowed. | *Stratix II GX Dynamic Reconfiguration* chapter in volume 2 of the Stratix II GX Device Handbook. |
| What is the selected input clock source for the Transmitter PLL and Receiver PLL? | If you select more than one input reference clock sources for the transmitter and/or receiver PLL, this option allows you to select the clock source for the current configuration. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the Stratix II GX Device Handbook. |
| What is the selected input clock source for the alternate Transmitter PLL and Receiver PLL? | If you select the **Use alternate Transmitter PLL and Receiver PLL** option, you can select the clock source for the alternate Transmitter PLL and the Receiver PLL. | *Stratix II GX Dynamic Reconfiguration* chapter in volume 2 of the Stratix II GX Device Handbook. |
| What is the reconfig protocol driven by clock 0? | If you select more than one input reference clock sources for the transmitter and/or receiver PLL, these options allow you to select the functional mode for the respective reference clock source. | *Stratix II GX Dynamic Reconfiguration* chapter in volume 2 of the Stratix II GX Device Handbook. |
| What is clock 0 input frequency? | If you select more than one input reference clock sources for the transmitter and/or receiver PLL, these options allow you to select the reference clock frequencies for each clock source. | *Stratix II GX Dynamic Reconfiguration* chapter in volume 2 of the Stratix II GX Device Handbook. |
| Use clock 0 reference clock divider | If you select more than one input reference clock source for the transmitter and/or receiver PLL, these options allow you to instruct the MegaWizard about the REFCLK pre-divider on input reference clocks. | *Stratix II GX Dynamic Reconfiguration* chapter in volume 2 of the Stratix II GX Device Handbook. |

Figure 3–47 shows page 11 of the ALT2GXB MegaWizard Plug-In Manager for SDI mode. This page appears only when the E**nable Channel and Transmitter PLL Reconfiguration** option is selected in the Reconfig page (Page 8).

*Figure 3–47. MegaWizard Plug-In Manager - ALT2GXB (Reconfig 2)*

Table 3–37 describes the available options on page 11 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

*Table 3–37. MegaWizard Plug-In Manager Options (Page 11 for SDI Mode)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| How should the receivers be clocked? | Three options are available:<br>● Share a single transmitter core clock between receivers<br>● Use the respective channel transmitter core clock<br>● Use the respective channel receiver core clocks | *Stratix II GX Dynamic Reconfiguration* chapter in volume 2 of the Stratix II GX Device Handbook. |
| How should the transmitters be clocked? | Two options are available:<br>● Share a single transmitter core clock between transmitters<br>● Use the respective channel transmitter core clocks | *Stratix II GX Dynamic Reconfiguration* chapter in volume 2 of the Stratix II GX Device Handbook. |
| Create `rx_revbitorderwa` input port to use receiver enable bit reversal | This optional input port allows you to dynamically reverse the bit order at the output of the receiver word aligner. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the Stratix II GX Device Handbook. |
| Check a control box to use the corresponding control port | You can select various control and status signals depending on what protocol(s) you intend to dynamically reconfigure the transceiver to. | *Stratix II GX Dynamic Reconfiguration* chapter in volume 2 of the Stratix II GX Device Handbook. |

Figure 3–48 shows page 12 of the ALT2GXB MegaWizard Plug-In Manager for SDI mode.

*Figure 3–48. MegaWizard Plug-In Manager - ALT2GXB (Loopback)*

Table 3–38 describes the available options on page 12 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

| *Table 3–38. MegaWizard Plug-In Manager Options (Page 12 for SDI Mode)* | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Which loopback option would you like? | There are two options available in SDI mode: no loopback and serial loopback.<br>● No loopback - this is the default mode.<br>● Serial loopback - if you select serial loopback, the `rx_seriallpbken` port is available to control the serial loopback feature dynamically. A 1'b1 enables serial loopback and a 1'b0 disables loopback on a channel-by-channel basis. Altera recommends controlling all four channels simultaneously. A digital reset must be asserted for the transceiver. | Loopback Modes section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Stratix II GX Device Handbook.* |
| Reverse Loopback option | This option is not available in SDI mode. | Loopback Modes section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Stratix II GX Device Handbook.* |

Figure 3–49 shows page 13 of the ALT2GXB MegaWizard Plug-In Manager for SDI mode. If the **Enforce default settings for this protocol** option is selected, this page does not appear in the MegaWizard.

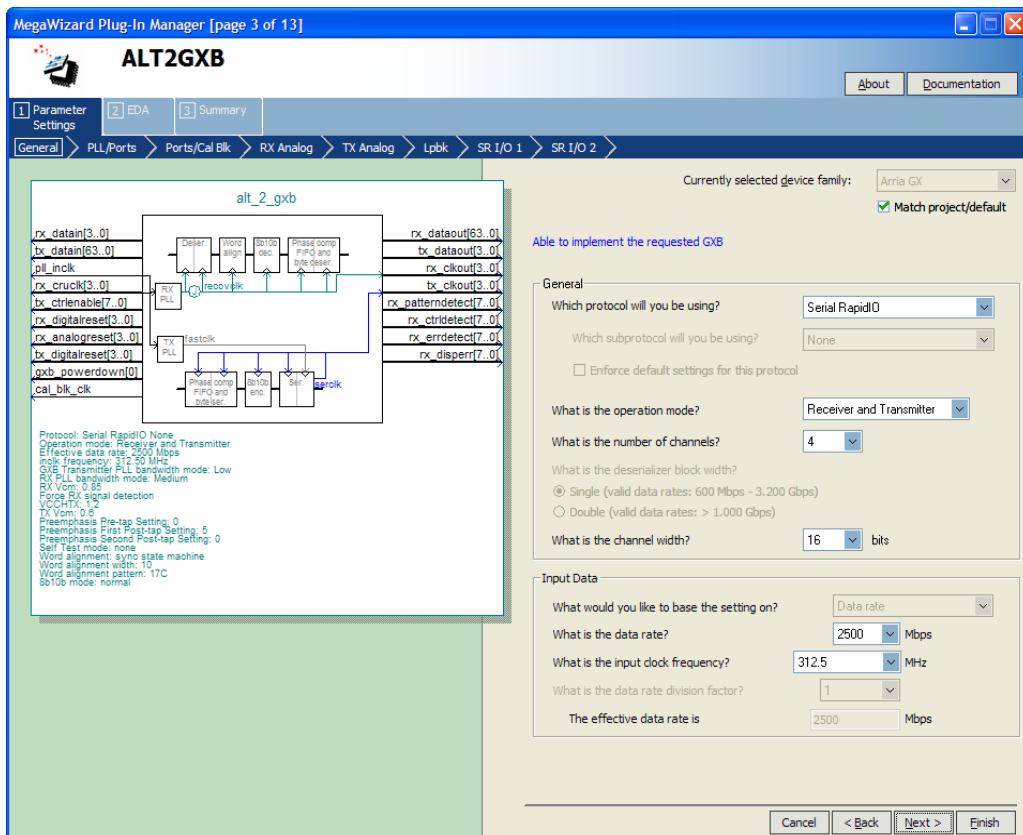*Figure 3–49. MegaWizard Plug-In Manager - ALT2GXB (SDI 1)*

Table 3–39 describes the available options on page 13 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

*Table 3–39. MegaWizard Plug-In Manager Options (Page 13 for SDI Mode)   (Part 1 of 2)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| Enable byte ordering block | This option is not available in Arria GX devices. | — |
| Enable 8B/10B decoder/encoder | This option is force-selected in SDI mode since 8B/10B decoder/encoder is always used. | 8B/10 Encoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Stratix II GX Device Handbook.* |
| Create `tx_forcedisp` to enable Force disparity and use `tx_dispval` to code up the incoming word using positive or negative disparity | This option allows you to force positive or negative disparity on transmitted data in 8B/10B configurations. | 8B/10 Encoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Stratix II GX Device Handbook.* |
| Enable rate match FIFO | This option is not available in SDI mode since the rate match FIFO is always bypassed. | Rate Matcher section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Stratix II GX Device Handbook.* |
| Flip Receiver output data bits | This option reverses the bit order of the data at the receiver-PLD interface at a byte level to support MSBit-to-LSBit transmission protocols. The default transmission order is LSBit-to-MSBit. | |
| Flip Transmitter input data bits | This option reverses the bit order of the data bits at the input of the transmitter at a byte level to support MSBit-to-LSBit transmission protocols. The default transmission order is LSBit-to-MSBit. | |
| Enable Transmitter bit reversal | This option inverts (flips) the bit order of the data bits at the transmitter PCS-PMA interface at a byte level to support MSBit-to-LSBit transmission protocols. The default transmission is LSBit-to-MSBit. | 8B/10B Encoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Stratix II GX Device Handbook.* |

| *Table 3–39. MegaWizard Plug-In Manager Options (Page 13 for SDI Mode) (Part 2 of 2)* | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Create `rx_invpolarity` to enable word aligner polarity inversion | This optional port allows you to dynamically reverse the polarity of the received data at the input of the word aligner. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Stratix II GX Device Handbook.* |
| Create `tx_invpolarity` to allow Transmitter polarity inversion | This optional port allows you to dynamically reverse the polarity of the data to be transmitted at the transmitter PCS-PMA interface. | 8B/10B Encoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Stratix II GX Device Handbook.* |

Figure 3–50 shows page 14 of the ALT2GXB MegaWizard Plug-In Manager for SDI mode.

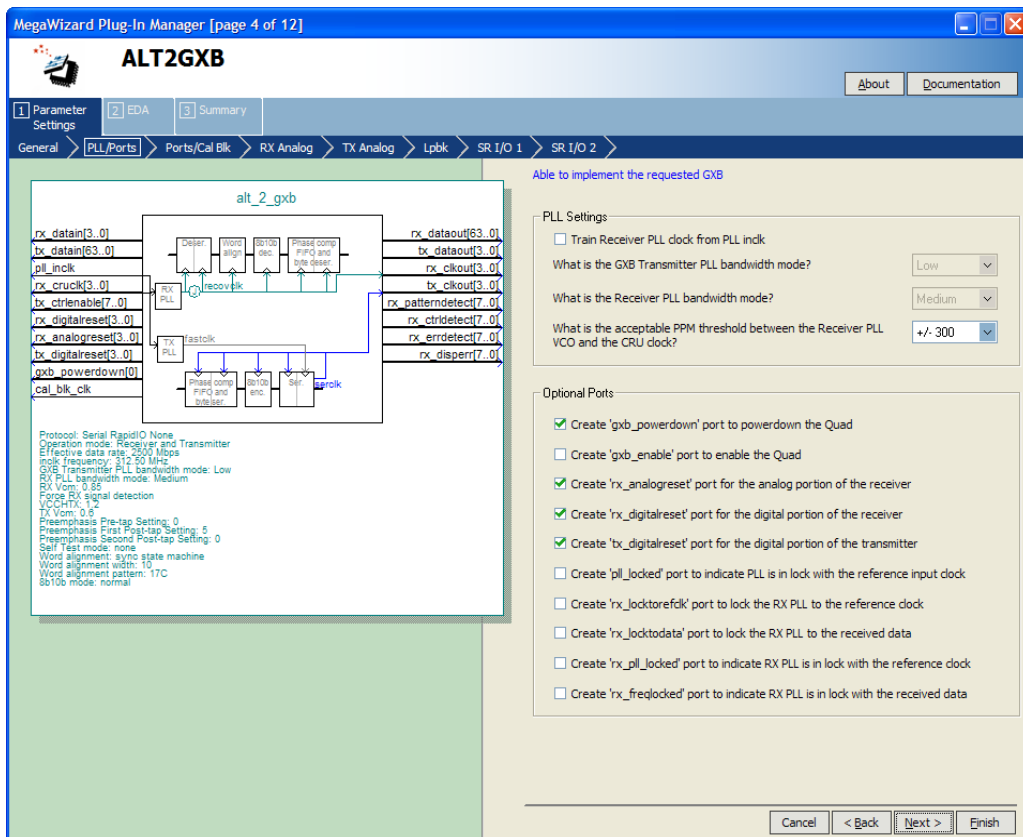*Figure 3–50. MegaWizard Plug-In Manager - ALT2GXB (SDI 2)*

Table 3–40 describes the available options on page 14 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

| Table 3–40. MegaWizard Plug-In Manager Options (Page 14 for SDI Mode)   (Part 1 of 3) | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Use manual word alignment mode | This option is not available in SDI mode as the word aligner uses the bit-slip port to alter the byte boundary one bit at a time. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Stratix II GX Device Handbook.* |
| Use manual bitslipping mode | This option sets the word aligner to use the bit-slip port to alter the byte boundary one bit at a time. This option is force selected in SDI mode. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Stratix II GX Device Handbook.* |
| Use the built-in 'synchronization state machine' | This option is not available in SDI mode as the word aligner uses the bit-slip port to alter the byte boundary one bit at a time. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Stratix II GX Device Handbook.* |
| Number of bad data words before loss of synch state | This option is not available in SDI mode. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Stratix II GX Device Handbook.* |
| Number of consecutive valid words before synch state is reached | This option is not available in SDI mode. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Stratix II GX Device Handbook.* |
| Number of valid patterns before synch state is reached | This option is not available in SDI mode. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Stratix II GX Device Handbook.* |
| What is the word alignment pattern length? | This option sets the word alignment length. The available choices are 7 bit and 10 bit. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Stratix II GX Device Handbook.* |

*Table 3–40. MegaWizard Plug-In Manager Options (Page 14 for SDI Mode)   (Part 2 of 3)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| What is the word alignment pattern? | Enter the word alignment pattern here. The length of the alignment pattern is based on the word alignment pattern length. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Stratix II GX Device Handbook.* |
| Flip word alignment pattern bits | This option reverses the bit order of the alignment pattern at a byte level to support MSB-to-LSB transmission protocols. The default transmission order is LSB-to-MSB. | |
| Enable run-length violation checking with a run length of | This option activates the run-length violation circuit. You can program the run length at which the circuit triggers the `rx_rlv` signal. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Stratix II GX Device Handbook.* |
| Enable word aligner output reverse bit ordering | This option statically configures the receiver to reverse the bit order of the data at the output of the word aligner. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Stratix II GX Device Handbook.* |
| Create `rx_syncstatus` output port for pattern detector and word aligner | Refer to the *Stratix II GX Transceiver Architecture Overview* chapter in volume 2 of the Stratix II GX Device Handbook for information about this port. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Stratix II GX Device Handbook.* |
| Create `rx_patterndetect` port to indicate pattern detected | Refer to the *Stratix II GX Transceiver Architecture Overview* chapter in volume 2 of the Stratix II GX Device Handbook for information about this port. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Stratix II GX Device Handbook.* |
| Create `rx_ctrldetect` port to indicate 8B/10B decoder has detected a control code | This option is not available in SDI mode. | |
| Create `rx_errdetect` port to indicate 8B/10B decoder has detected an error code | This option is not available in SDI mode. | |

| *Table 3–40. MegaWizard Plug-In Manager Options (Page 14 for SDI Mode)   (Part 3 of 3)* | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Create `rx_disperr` port to indicate 8B/10B decoder has detected a disparity code | This option is not available in SDI mode. | |
| Create `rx_revbyteorderwa` to enable receiver symbol swap | This option is not available in SDI mode. | |

Figure 3–51 shows page 15 of the MegaWizard Plug-In Manager for the SDI protocol selection. The **Generate simulation model** option creates a behavioral model (**.vo** or **.vho**) of the transceiver instance for third-party simulators. The **Generate Netlist** option generates a netlist for third party EDA synthesis tool to be able to estimate timing and resource utilization for the ALT2GXB instance.

*Figure 3–51. MegaWizard Plug-In Manager - ALT2GXB (EDA)*
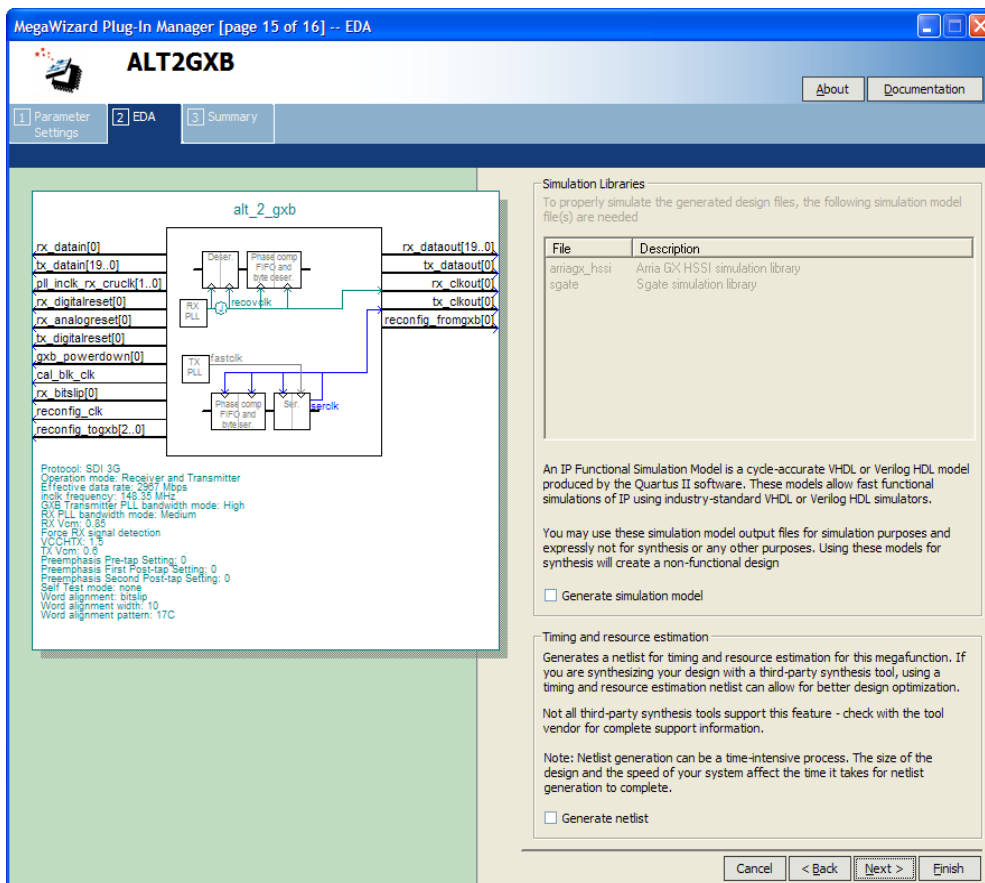
Figure 3–52 shows page 16 (the last page) of the MegaWizard Plug-In Manager for the SDI protocol set up. You can select optional files on this page. After you make your selections, click **Finish** to generate the files.

*Figure 3–52. MegaWizard Plug-In Manager - ALT2GXB (Summary)*
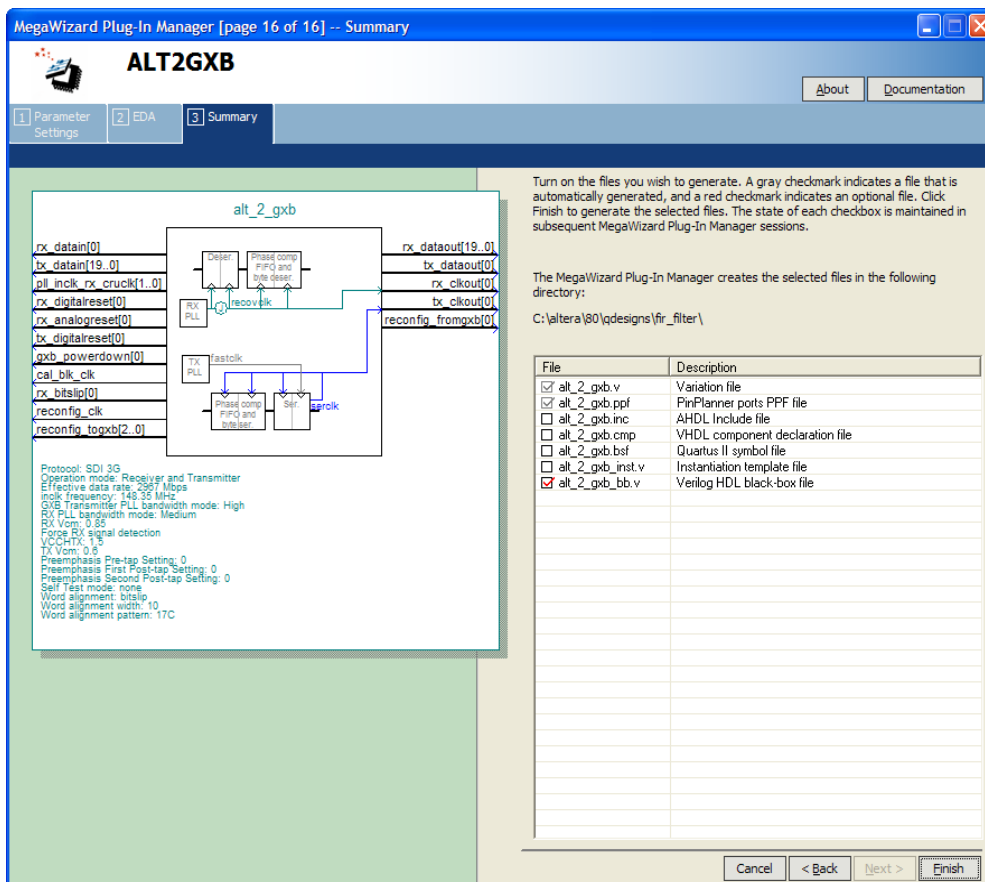


## Serial RapidIO Mode

This section provides descriptions of the options available on the individual pages of the ALT2GXB MegaWizard Plug-In Manager for Serial RapidIO mode. The MegaWizard Plug-In Manager provides a warning if any of the settings you choose are illegal.

Figure 3–53 shows page 3 of the `ALT2GXB` MegaWizard Plug-In Manager in Serial RapidIO mode.

*Figure 3–53. MegaWizard Plug-In Manager - ALT2GXB (General)*

Table 3–41 describes the available options on page 3 of the MegaWizard Plug-In Manager for your `ALT2GXB` custom megafunction variation.

| Table 3–41. MegaWizard Plug-In Manager Options (Page 3 for Serial RapidIO Mode) | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Which protocol will you be using? | Determines the specific protocol or modes under which the transceiver operates. For Serial RapidIO mode, you must select the **Serial RapidIO**. | — |
| Which subprotocol will you be using? | Not applicable to Serial RapidIO mode. | — |
| Enforce default settings for this protocol | Not applicable to Serial RapidIO mode. | — |
| What is the operation mode? | The available operation modes are receiver only, transmitter only, and receiver and transmitter. | — |
| What is the number of channels? | This option determines how many duplicate channels this ALT2GXB instance contains. | — |
| What is the deserializer block width? | This option is unavailable in Serial RapidIO mode. | — |
| What is the channel width? | This option determines the PLD-transceiver interface width. Only 16-bit interface width is supported. | Byte Serializer and Byte Deserializer sections in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| What would you like to base the setting on? | This option is unavailable in Serial RapidIO mode. | — |
| What is the data rate? | In Serial RapidIO mode, data rates of 1250 Mbps, 2500 Mbps, and 3125 Mbps are supported. | — |
| What is the input clock frequency? | Determines the input reference clock frequency for the transceiver. The following input reference clock frequencies are supported for each data rate option:<br>● 1250 Mbps: 62.5 MHz, 78.125 MHz, 125 MHz, 156.25MHz, 250 MHz, 312.5 MHz<br>● 2500 Mbps: 50 MHz, 62.5 MHz, 78.125 MHz, 100 MHz, 125 MHz, 156.25MHz, 250 MHz, 312.5 MHz, 500 Mhz.<br>● 3125 Mbps: 62.5 MHz, 78.125 MHz, 97.6563 MHz, 125 MHz, 156.25MHz, 195.3125 MHz, 312.5 MHz, 390.625 MHz. | Serial RapidIO Mode section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| What is the data rate division factor? | This option is unavailable in Serial RapidIO mode. | — |

Figure 3–54 shows page 4 of the `ALT2GXB` MegaWizard Plug-In Manager for Serial RapidIO mode.

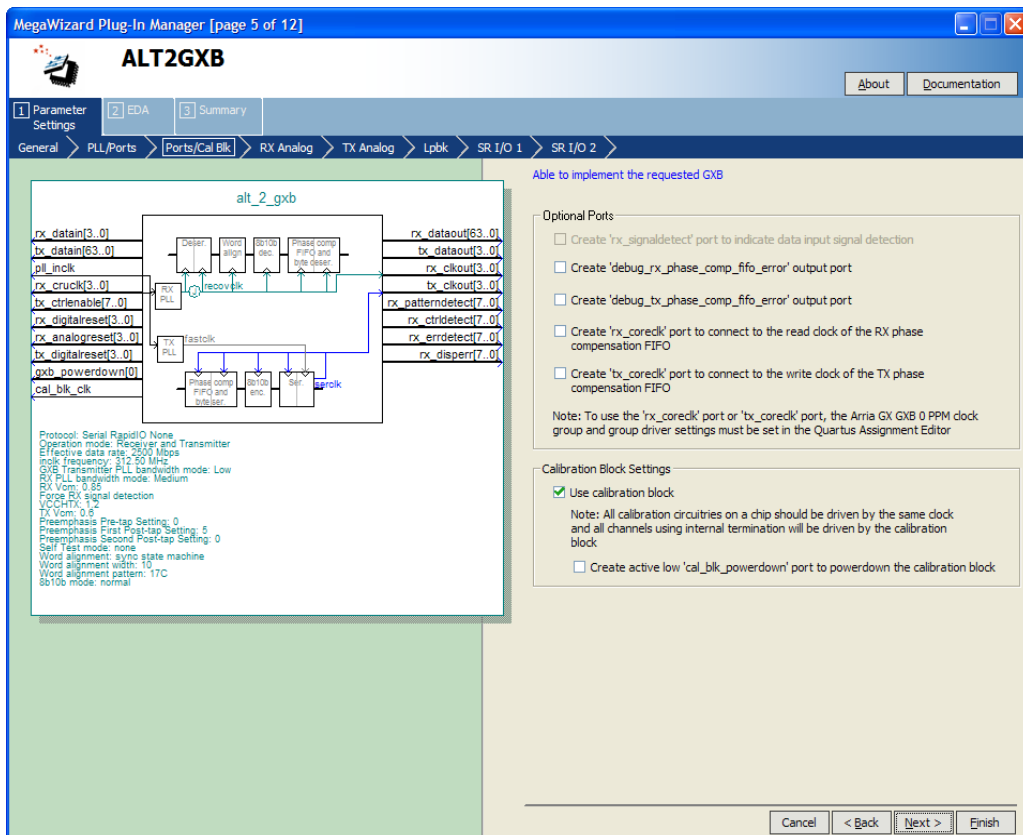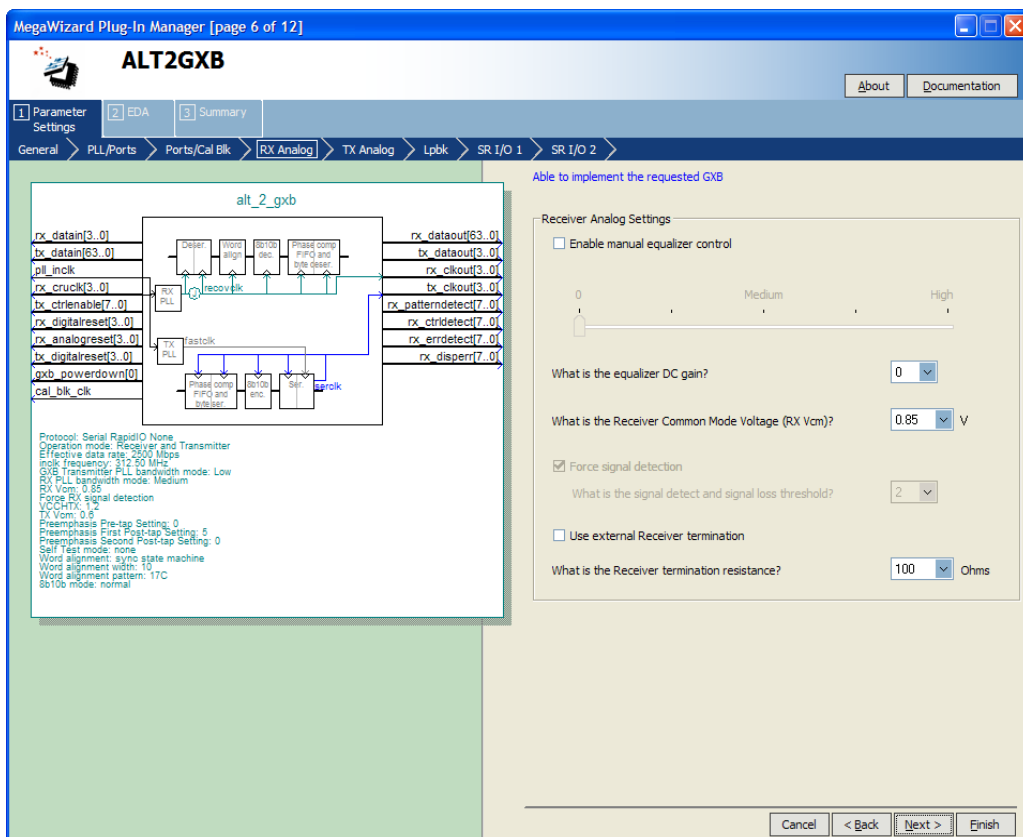*Figure 3–54. MegaWizard Plug-In Manager - ALT2GXB (PLL/Ports)*

Table 3–42 describes the available options on page 4 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

**Table 3–42. MegaWizard Plug-In Manager Options (Page 4 for Serial RapidIO Mode)**

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| Train Receiver PLL clock from PLL inclk | If you select this option, the transmitter input reference clock (pll_inclk) drives the receiver PLL input reference clock also.<br>If you do not select this option, the signal on the rx_cruclk port drives the receiver PLL input reference clock. | — |
| What is the GXB Transmitter PLL bandwidth mode? | This option is not available in Serial RapidIO mode because the transmitter PLL bandwidth is fixed at high. | — |
| What is the Receiver PLL bandwidth mode? | This option is not available in Serial RapidIO mode because the receiver PLL bandwidth is fixed at medium. | — |
| What is the acceptable PPM threshold between the Receiver PLL VCO and the CRU clock? | This option determines the PPM difference that affects the automatic receiver CRU switchover between lock-to-data and lock-to-reference. (There are additional factors that affect the CRU's transition.) | Clock Recovery Unit (CRU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create gxb_powerdown port to power down the Quad | This signal can be used to reset and power down all circuits in the transceiver block. It does not power down the REFCLK buffers and reference clock lines. | Reset Control and Power Down section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| Create gxb_enable port to enable the Quad | This signal can be used to enable Arria GX transceiver blocks. If instantiated, this port must be tied to the dedicated gigabit transceiver block enable input pin. | Reset Control and Power Down section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook.* |
| Create rx_analogreset port for the analog portion of the receiver | Receiver analog reset port. | Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |

*Table 3–42. MegaWizard Plug-In Manager Options (Page 4 for Serial RapidIO Mode)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| Create `rx_digitalreset` port for the digital portion of the receiver | Receiver digital reset port. Resets the PCS logic of the receiver. Altera recommends using this port to implement the recommended reset sequence. | Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `tx_digitalreset` port for the digital portion of the transmitter | Transmitter digital reset port. Resets the PCS logic of the transmitter. Altera recommends using this port to implement the recommended reset sequence. | Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `pll_locked` port to indicate PLL is in lock with the reference input clock | Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `rx_locktorefclk` port to lock the RX PLL to the reference clock | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Clock Recovery Unit (CRU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `rx_locktodata` port to lock the RX PLL to the received data | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Clock Recovery Unit (CRU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `rx_pll_locked` port to indicate RX PLL is in lock with the reference clock | Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |

*Table 3–42. MegaWizard Plug-In Manager Options (Page 4 for Serial RapidIO Mode)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| Create `rx_freqlocked` port to indicate RX PLL is in lock with the received data | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Clock Recovery Unit (CRU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `rx_signaldetect` port to indicate data input signal detection | This option is not available in Serial RapidIO mode. | — |
| Create `debug_rx_phase_comp_ fifo_error` output port | This optional output port indicates Receiver Phase Compensation FIFO overflow/under run condition. Note that no PPM difference is allowed between FIFO read and write clocks. Use this port for debug purposes only. | Receiver Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `debug_tx_phase_comp_ fifo_error` output port | This optional output port indicates Transmitter Phase Compensation FIFO overflow/under run condition. Note that no PPM difference is allowed between FIFO read and write clocks. Use this port for debug purposes only. | Transmitter Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `rx_coreclk` port to connect to the read clock of the RX phase compensation FIFO | This optional input port allows you to clock the read side of the Receiver Phase Compensation FIFO with a non-transceiver PLD clock. | PLD-Transceiver Interface Clocking section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `tx_coreclk` port to connect to the write clock of the TX phase compensation FIFO | This optional input port allows you to clock the write side of the Transmitter Phase Compensation FIFO with a non-transceiver PLD clock. | PLD-Transceiver Interface Clocking section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |

Figure 3–54 shows page 5 of the `ALT2GXB` MegaWizard Plug-In Manager for Serial RapidIO mode.

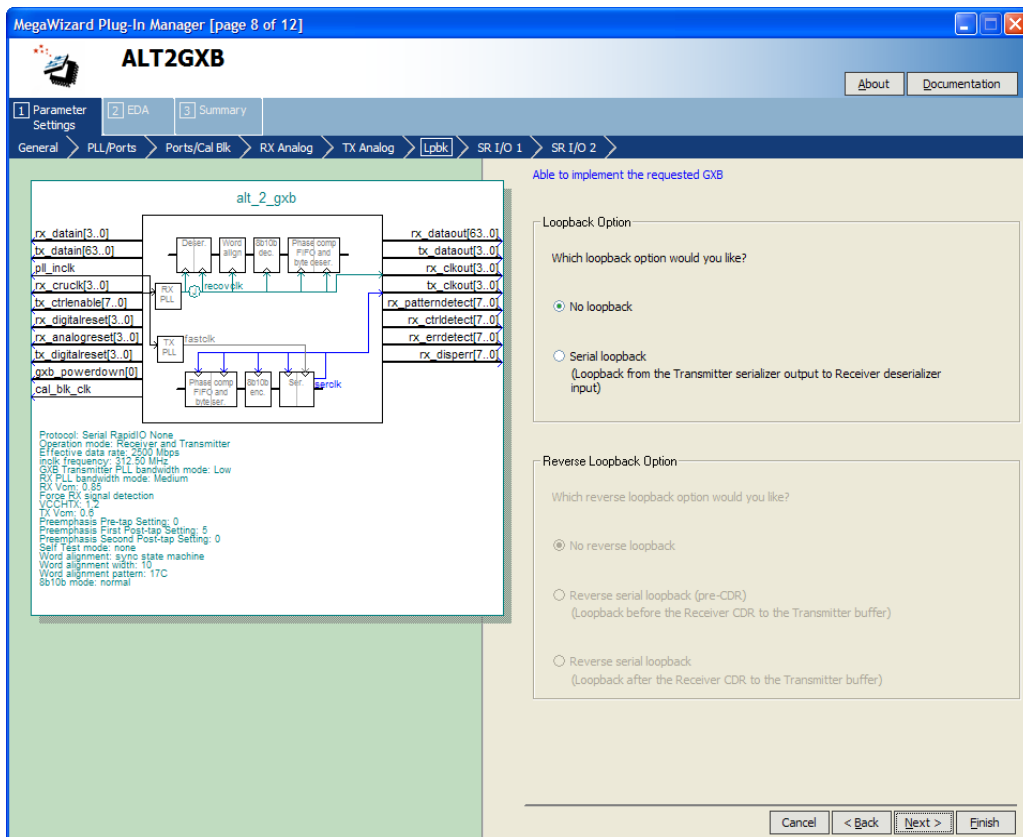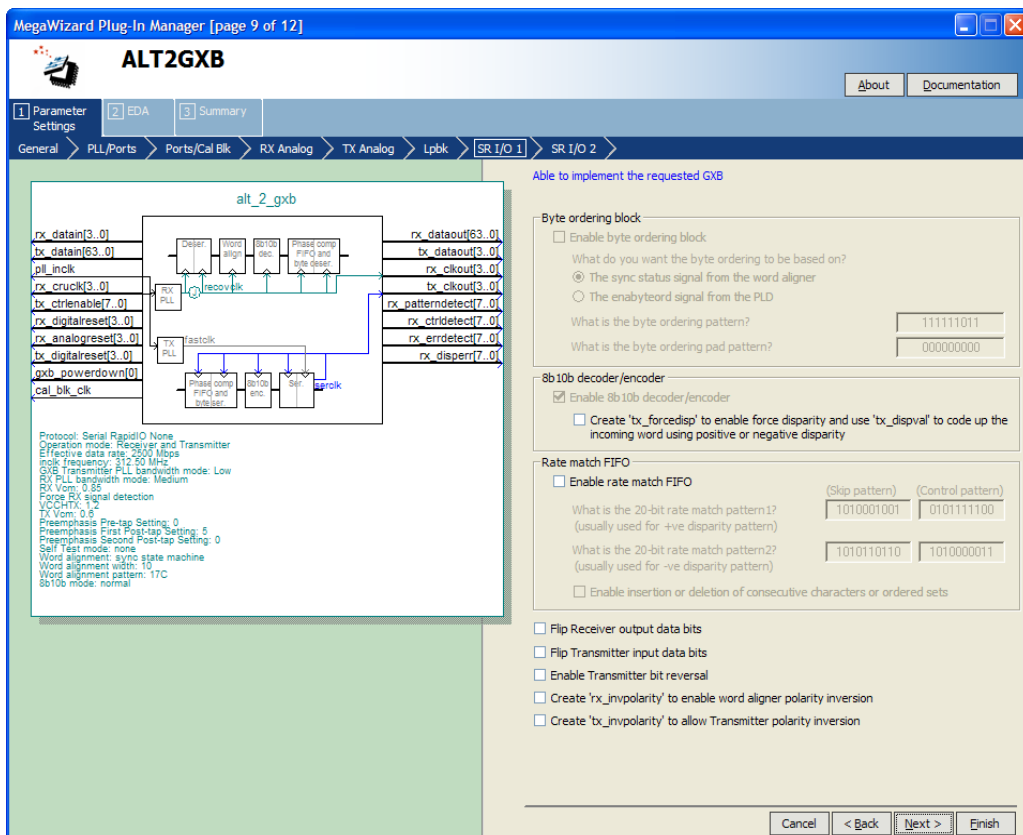*Figure 3–55. MegaWizard Plug-In Manager - ALT2GXB (Ports/Cal Blk)*

Table 3–42 describes the available options on page 5 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

**Table 3–43. MegaWizard Plug-In Manager Options (Page 5 for Serial RapidIO Mode)**

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| Create rx_signaldetect port to indicate data input signal detection | This option is not available in Serial RapidIO mode. | — |
| Create debug_rx_phase_comp_fifo_error output port | This optional output port indicates Receiver Phase Compensation FIFO overflow/under run condition. Note that no PPM difference is allowed between FIFO read and write clocks. Use this port for debug purposes only. | Receiver Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create debug_tx_phase_comp_fifo_error output port | This optional output port indicates Transmitter Phase Compensation FIFO overflow/under run condition. Note that no PPM difference is allowed between FIFO read and write clocks. Use this port for debug purposes only. | Transmitter Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create rx_coreclk port to connect to the read clock of the RX phase compensation FIFO | This optional input port allows you to clock the read side of the Receiver Phase Compensation FIFO with a non-transceiver PLD clock. | PLD-Transceiver Interface Clocking section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create tx_coreclk port to connect to the write clock of the TX phase compensation FIFO | This optional input port allows you to clock the write side of the Transmitter Phase Compensation FIFO with a non-transceiver PLD clock. | PLD-Transceiver Interface Clocking section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |

| *Table 3–43. MegaWizard Plug-In Manager Options (Page 5 for Serial RapidIO Mode)* | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Use calibration block | This option allows you to select which instance of the ALT2GXB megafunction instantiates the calibration block. Only one instance of the ALT2GXB megafunction is required to instantiate the calibration block. | Calibration Block section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create active low `cal_blk_powerdown` to power down the calibration block | Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook,* for information about this port. | Calibration Block section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |

Figure 3–56 shows page 6 of the `ALT2GXB` MegaWizard Plug-In Manager for Serial RapidIO mode.

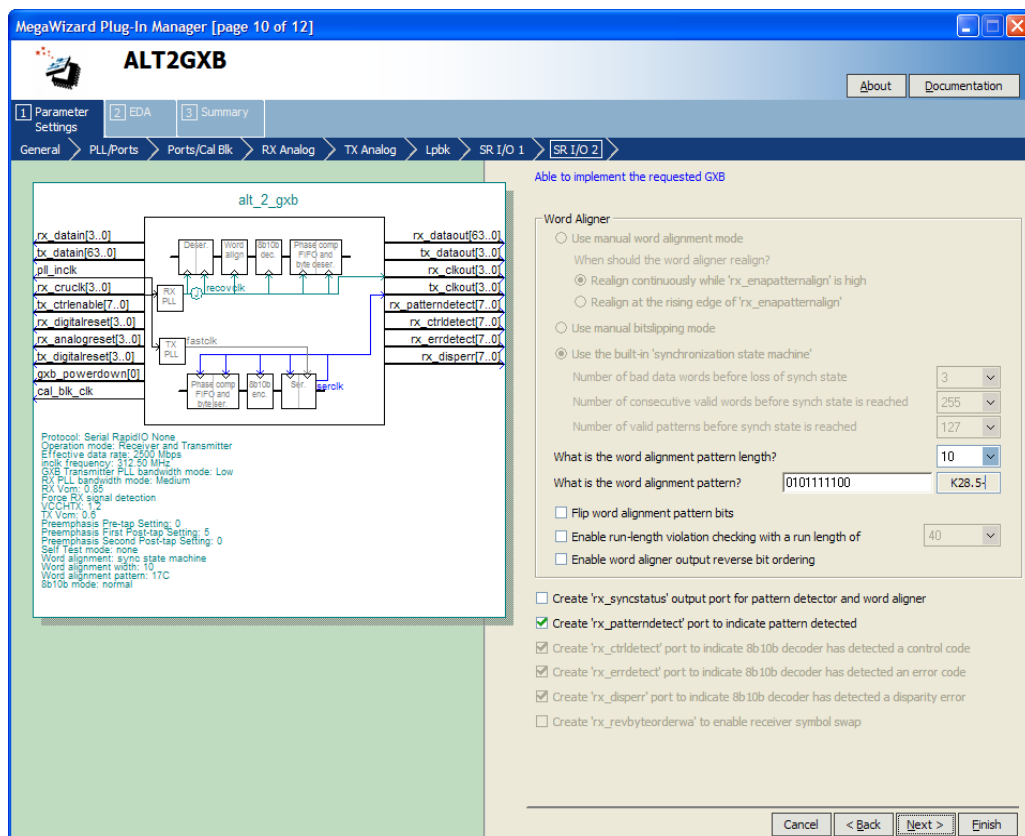*Figure 3–56. MegaWizard Plug-In Manager - ALT2GXB (RX Analog)*

Table 3–44 describes the available options on page 6 of the MegaWizard Plug-In Manager for your `ALT2GXB` custom megafunction variation.

| *Table 3–44. MegaWizard Plug-In Manager Options (Page 6 for Serial RapidIO Mode)* | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Enable manual equalizer control | This option enables the 0–4 setting options for manual equalizer control. | Receiver Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What is the equalizer DC gain? | This enables the DC gain option and the legal settings are 0, 1, 2, and 3. | Receiver Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*. |
| What is the Receiver Common Mode Voltage (RX V$_{CM}$)? | The receiver common mode voltage is programmable, and the selections available are 0.85 V and 1.2 V. | Receiver Buffer section under Serial RapidIO Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Force signal detection | This option is not available in Serial RapidIO mode and is always forced selected. | — |
| What is the signal detect and signal loss threshold? | This option is not available in Serial RapidIO mode as signal detection is forced. | — |
| Use external receiver termination | This option is available if you use an external termination resistor instead of the on-chip termination OCT. If checked, this option turns off the receiver OCT. | — |
| What is the receiver termination resistance? | In Serial RapidIO mode, the only supported receiver termination resistance is 100 Ω | Receiver Buffer section under Serial RapidIO Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |

Figure 3–57 shows page 7 of the ALT2GXB MegaWizard Plug-In Manager for Serial RapidIO mode.

*Figure 3–57. MegaWizard Plug-In Manager - ALT2GXB (TX Analog)*
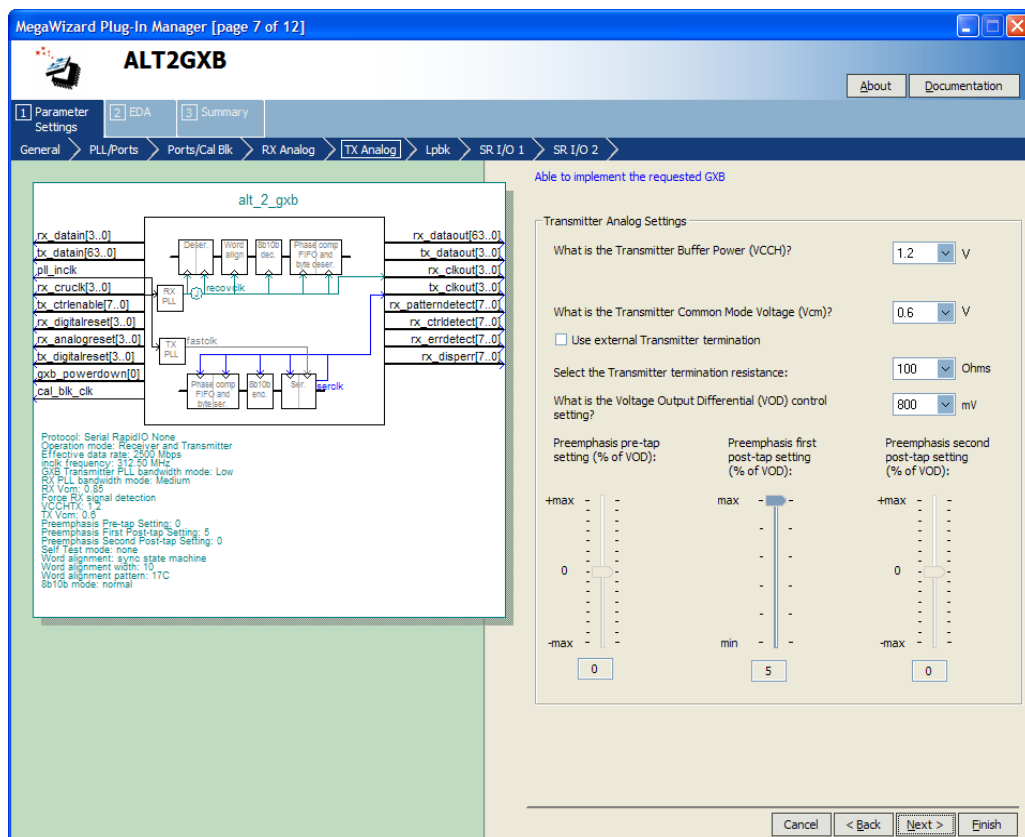
Table 3–45 describes the available options on page 7 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

***Table 3–45. MegaWizard Plug-In Manager Options (Page 7 for Serial RapidIO Mode)***

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| What is the Transmitter Buffer Power (VCCH)? | This setting is for information only and is used to calculate the $V_{OD}$ from the buffer power supply ($V_{CCH}$) and the transmitter termination to derive the proper $V_{OD}$ range. In serial RapidIO mode, this option is fixed at 1.5 V | Transmitter Buffer section under Serial RapidIO Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| What is the Transmitter Common Mode Voltage ($V_{CM}$)? | In Serial RapidIO mode, the transmitter common mode voltage is selectable between 0.6 V and 0.7 V. | Transmitter Buffer section under Serial RapidIO mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Use external Transmitter termination | This option is available if you want to use an external termination resistor instead of the on-chip termination OCT. Checking this option turns off the transmitter OCT. | — |
| Select the Transmitter termination resistance | In Serial RapidIO mode, the only supported receiver termination resistance is 100 Ω | Transmitter Buffer section under Serial RapidIO mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| What is the Voltage Output Differential (VOD) control setting? | This option selects the $V_{OD}$ of the transmitter buffer. The differential output voltage is programmable between 400 mV and 1200 mV in steps of 200 mV. The available $V_{OD}$ settings change based on $V_{CCH}$. | Transmitter Buffer section under Serial RapidIO Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Pre-emphasis pre-tap setting (% of VOD) | This option is not available in Arria GX devices and is fixed at 0. | — |

| *Table 3–45. MegaWizard Plug-In Manager Options (Page 7 for Serial RapidIO Mode)* | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Pre-emphasis first post-tap setting (% of VOD) | This option sets the amount of pre-emphasis on the transmitter buffer using first post-tap. The options available are 0, 1, 2, 3, 4, and 5. | Transmitter Buffer section under Serial RapidIO Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Pre-emphasis second post-tap setting (% of VOD) | This option is not available in Arria GX devices and is fixed at 0. | — |

Figure 3–58 shows page 8 of the ALT2GXB MegaWizard Plug-In Manager for Serial RapidIO mode.

*Figure 3–58. MegaWizard Plug-In Manager - ALT2GXB (Loopback)*

Table 3–46 describes the available options on page 8 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

| Table 3–46. MegaWizard Plug-In Manager Options (Page 8 for Serial RapidIO Mode) | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Which loopback option would you like? | No loopback and serial loopback options are available in Serial RapidIO mode.<br>● No loopback is the default mode.<br>● If you select serial loopback, the rx_seriallpbken port is available to control the serial loopback feature dynamically. A 1'b1 enables serial loopback and a 1'b0 disables loopback on a channel-by-channel basis. | Loopback Modes section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |

Figure 3–59 shows page 9 of the ALT2GXB MegaWizard Plug-In Manager for Serial RapidIO mode.

*Figure 3–59. MegaWizard Plug-In Manager - ALT2GXB (SR I/O 1)*



Table 3–47 describes the available options on page 9 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

*Table 3–47. MegaWizard Plug-In Manager Options (Page 9 for Serial RapidIO Mode)   (Part 1 of 2)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| Enable byte ordering block | This option is not available in Arria GX devices. | — |
| Enable 8B/10B decoder/encoder | This option is unavailable in Serial RapidIO mode and is always forced selected to enable 8B/10B decoder/encoder. | — |

*Table 3–47. MegaWizard Plug-In Manager Options (Page 9 for Serial RapidIO Mode)   (Part 2 of 2)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| Create `tx_forcedisp` to enable Force disparity and use `tx_dispval` to code up the incoming word using positive or negative disparity | This option is unavailable in Serial RapidIO mode. | — |
| Enable rate match FIFO | This option is unavailable in Serial RapidIO mode as the rate matcher is not supported. | — |
| Flip Receiver output data bits | This option reverses the bit order of the data at the receiver-PLD interface at a byte level. | — |
| Flip Transmitter input data bits | This option reverses the bit order of the data bits at the input of the transmitter at a byte level. | — |
| Enable Transmitter bit reversal | This option is unavailable in Serial RapidIO mode. | — |
| Create `rx_invpolarity` to enable word aligner polarity inversion | This optional port allows you to dynamically reverse the polarity of the received data at the input of the word aligner. | Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `tx_invpolarity` to allow Transmitter polarity inversion | This optional port allows you to dynamically reverse the polarity of the data to be transmitted at the transmitter PCS-PMA interface. | 8B/10B Encoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |

Figure 3–60 shows page 10 of the MegaWizard Plug-In Manager for Serial RapidIO protocol set up.

*Figure 3–60. MegaWizard Plug-In Manager - ALT2GXB (SR I/O 2)*



Table 3–48 describes the available options on page 10 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

*Table 3–48. MegaWizard Plug-In Manager Options (Page 10 for Serial RapidIO Mode)   (Part 1 of 4)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| Use manual word alignment mode | This option is unavailable in Serial RapidIO mode. | — |
| Use manual bit slipping mode. | This option is unavailable in Serial RapidIO mode. | — |

*Table 3–48. MegaWizard Plug-In Manager Options (Page 10 for Serial RapidIO Mode)   (Part 2 of 4)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| Use the built-in 'synchronization state machine' | This option is forced selected in Serial RapidIO mode. | Word Aligner section under Serial RapidIO Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Number of bad data words before loss of synch state | Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Word Aligner section under Serial RapidIO Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Number of consecutive valid words before synch state is reached | Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Word Aligner section under Serial RapidIO mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Number of valid patterns before synch state is reached | Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Word Aligner section under Serial RapidIO Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| What is the word alignment pattern length? | The word alignment pattern length is fixed to 10 in Serial RapidIO mode. | Word Aligner section under Serial RapidIO mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| What is the word alignment pattern? | Enter the 10-bit word alignment pattern here. | Word Aligner section under Serial RapidIO Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |

*Table 3–48. MegaWizard Plug-In Manager Options (Page 10 for Serial RapidIO Mode)   (Part 3 of 4)*

| ALT2GXB Setting | Description | Reference |
|---|---|---|
| Flip word alignment pattern bits | This option reverses the bit order of the alignment pattern at a byte level to support MSB-to-LSB transmission protocols. The default transmission order is LSB-to-MSB. | — |
| Enable run-length violation checking with a run length of | This option activates the run-length violation circuit. You can program the run length at which the circuit triggers the rx_rlv signal. | Word Aligner section n the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Enable word aligner output reverse bit ordering | This option is unavailable in Serial RapidIO mode. | — |
| Create rx_syncstatus output port for pattern detector and word aligner | Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Word Aligner section under Serial RapidIO Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create rx_patterndetect output port to indicate pattern detected | Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | Word Aligner section under Serial RapidIO Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create rx_ctrldetect output port to indicate 8B/10B decoder has detected a control code | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | 8B/10B Decoder section n the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create rx_errdetect port to indicate 8B/10B decoder has detected an error code | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | 8B/10B Decoder section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |

| Table 3–48. MegaWizard Plug-In Manager Options (Page 10 for Serial RapidIO Mode)  (Part 4 of 4) | | |
|---|---|---|
| **ALT2GXB Setting** | **Description** | **Reference** |
| Create `rx_disperr` port to indicate 8B/10B decoder has detected a disparity error | Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port. | 8B/10B Decoder section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* |
| Create `rx_revbyteorderwa` to enable receiver symbol swap | This option is unavailable in Serial RapidIO mode. | — |

Figure 3–61 shows page 11 of the MegaWizard Plug-In Manager for the Serial RapidIO protocol selection. The **Generate simulation model** option creates a behavioral model (**.vo** or **.vho**) of the transceiver instance for third-party simulators. The **Generate a netlist for synthesis area and timing estimation** option creates a netlist file (**.syn**) for third-party synthesis tools.

*Figure 3–61. MegaWizard Plug-In Manager - ALT2GXB (EDA)*
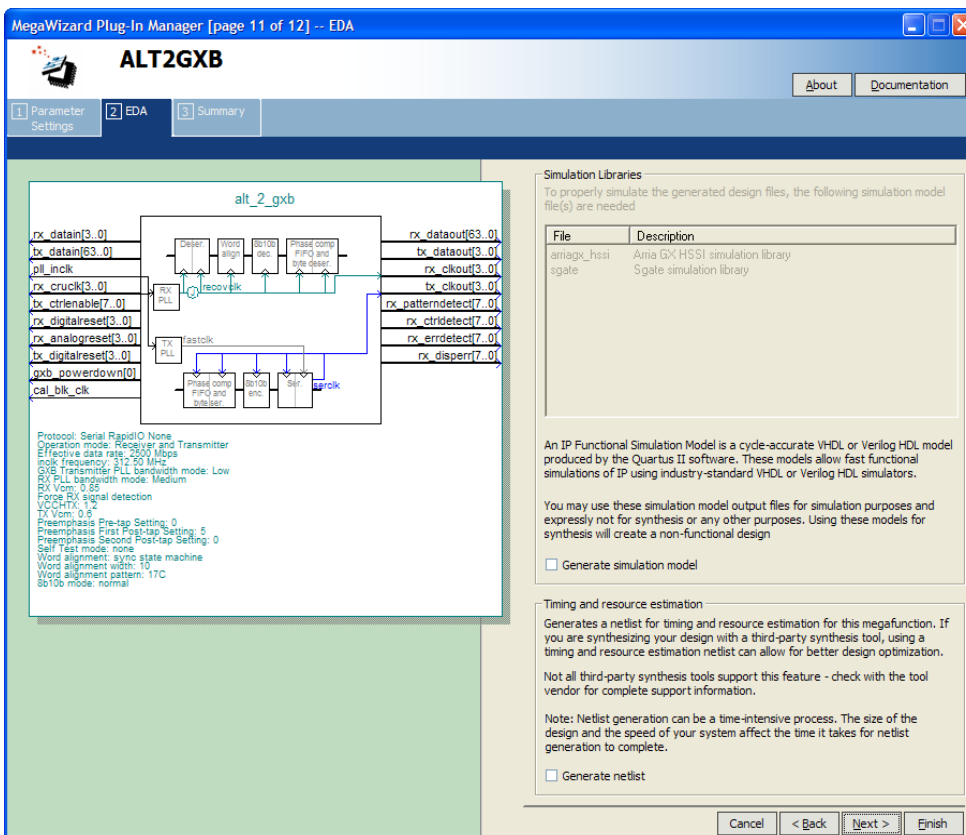
Figure 3–62 shows page 12 (last page) of the MegaWizard Plug-In Manager for Serial RapidIO protocol set up. You can select optional files on this page. After you make your selections, click **Finish** to generate the files.

*Figure 3–62. MegaWizard Plug-In Manager - ALT2GXB (Summary)*
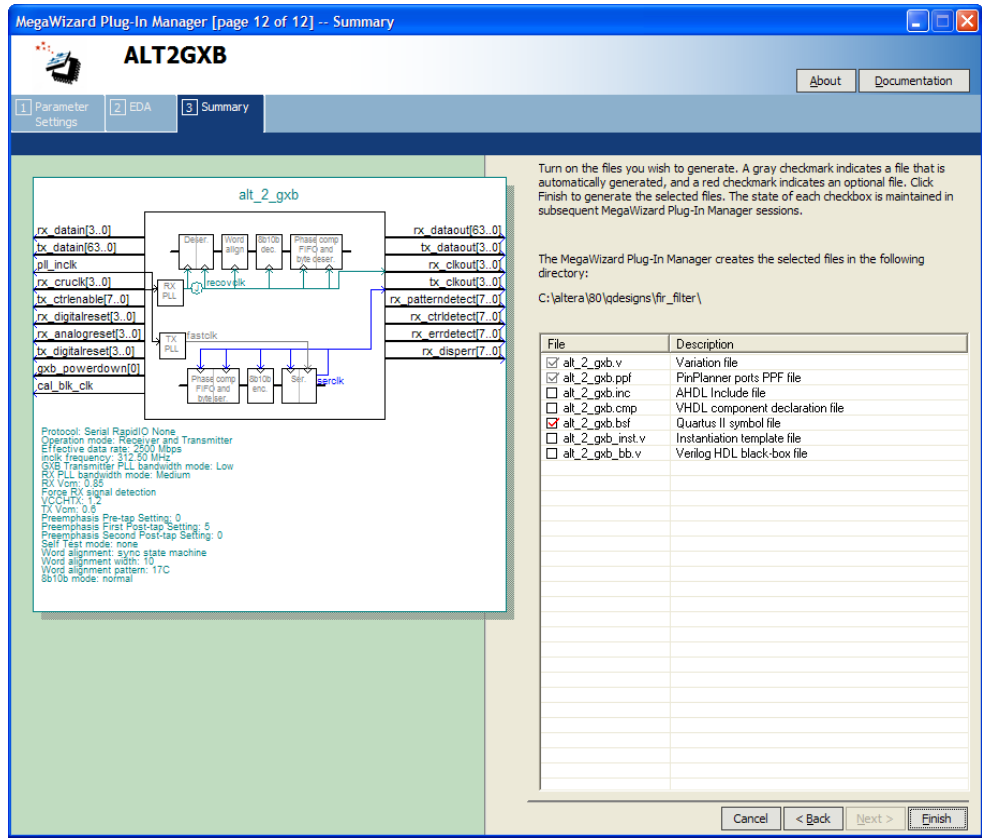


## Referenced Documents

This chapter references the following documents:

- *Arria GX Architecture* chapter in volume 1 of the *Arria GX Device Handbook*
- *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
- *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*

■ *Stratix II GX Transceiver Architecture Overview* chapter in volume 2 of the *Stratix II GX Device Handbook*

## Document Revision History

Table 3–49 shows the revision history for this chapter.

| Table 3–49. Document Revision History | | |
|---|---|---|
| **Date and Document Version** | **Changes Made** | **Summary of Changes** |
| August 2007, v1.2 | Added the "Referenced Documents" section. | — |
| | Minor text edits. | — |
| June 2007, v1.1 | Added GIGE information. | — |
| May 2007, v1.0 | Initial Release. | — |

## 8B/10B Code

This section provides information about the data and control codes for Arria™ GX devices.

### Code Notation

The 8B/10B data and control codes are referred to as Dx.y and Kx.y, respectively. The 8-bit byte – H G F E D C B A, where H is the most significant bit (MSB) and A is the significant bit (LSB) – is broken up into two groups, x and y, where x is the five lower bits (E D C B A) and y is the three upper bits (H G F). Figure 4–1 shows the designation for 3C hex.

*Figure 4–1. Sample Notation for 3C hex*
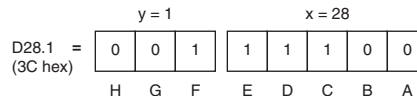


There are 256 Dx.y and 12 Kx.y valid 8-bit codes. These codes have two 10-bit equivalent codes associated with each 8-bit code. The 10-bit codes have either a neutral disparity or a non-neutral disparity. With neutral disparity, two neutral disparity 10-bit codes are associated with an 8-bit code. With non-neutral disparity 10-bit code, a positive and a negative disparity code are associated with the 8-bit code.

The positive disparity 10-bit code is associated in the RD– column. The negative disparity 10-bit code is associated in the RD+ column.

### Disparity Calculation

Running disparity is calculated based on the sub-blocks of the 10-bit code. The 10-bit code is divided into two sub-blocks, a 6-bit sub-block (abcdei) and a 4-bit sub-block (fghj), as shown in Figure 4–2.

*Figure 4–2. 10-Bit Grouping of 6-bit & 4-Bit Sub-Blocks*
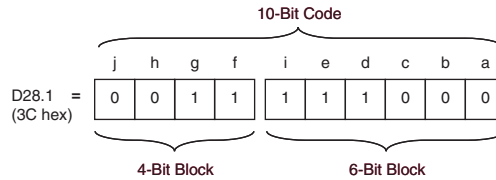


The running disparity at the beginning of the 6-bit sub-block is the running disparity at the end of the previous 10-bit code. The running disparity of the 4-bit sub-block is the running disparity at the end of the 6-bit sub-block. The running disparity at the end of the 4-bit sub-block is the running disparity of the 10-bit code (refer to Figure 4–3).

*Figure 4–3. Running Disparity Between Sub-Blocks*
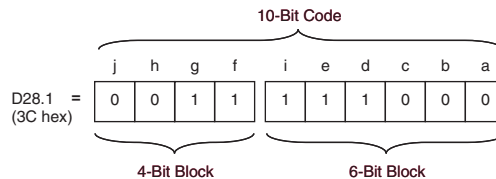


The running disparity calculation rules are as follows:

■ The current running disparity at the end of a sub-block is positive if any of the following is true:
   ● The sub-block contains more ones than zeros
   ● The 6-bit sub-block is 6'b000111
   ● The 4-bit sub-block is 4'b0011

■ The current running disparity at the end of a sub-block is negative if any of the following is true:
   ● The sub-block contains more zeros than ones
   ● The 6-bit sub-block is 6'b111000
   ● The 4-bit sub-block is 4'b1100

If those conditions are not met, the running disparity at the end of the sub-block is the same as at the beginning of the sub-block.

## Supported Codes

The 8B/10B scheme defines the 12 control codes listed in Table 4–1 for synchronization, alignment, and general application purposes.

| K Code | Octal Value | 8-Bit Code | 10-Bit Code RD– | 10-Bit Code RD+ |
|---|---|---|---|---|
| | | HGF_EDCBA | abcdei_fghj | |
| K28.0 | 1C | 8'b000_11100 | 10'b001111_0100 | 10'b110000_1011 |
| K28.1 | 3C | 8'b001_11100 | 10'b001111_1001 | 10'b110000_0110 |
| K28.2 | 5C | 8'b010_11100 | 10'b001111_0101 | 10'b110000_1010 |
| K28.3 | 7C | 8'b011_11100 | 10'b001111_0011 | 10'b110000_1100 |
| K28.4 | 9C | 8'b100_11100 | 10'b001111_0010 | 10'b110000_1101 |
| K28.5 *(1)* | BC | 8'b101_11100 | 10'b001111_1010 | 10'b110000_0101 |
| K28.6 | DC | 8'b110_11100 | 10'b001111_0110 | 10'b110000_1001 |
| K28.7 | FC | 8'b111_11100 | 10'b001111_1000 | 10'b110000_0111 |
| K23.7 | F7 | 8'b111_10111 | 10'b111010_1000 | 10'b000101_0111 |
| K27.7 | FB | 8'b111_11011 | 10'b110110_1000 | 10'b001001_0111 |
| K29.7 | FD | 8'b111_11101 | 10'b101110_1000 | 10'b010001_0111 |
| K30.7 | FE | 8'b111_11110 | 10'b011110_1000 | 10'b100001_0111 |

*Table 4–1. Supported K Codes*

*Note to Table 4–1:*

(1)   K28.5 is a comma code used for word alignment and indicates an IDLE state.

Table 4–2 shows the valid data code-groups.

| Code-Group Name | Octet Value | Octet Bits | Current RD– | Current RD+ |
|---|---|---|---|---|
| | | HGF EDCBA | abcdei fghj | |
| D0.0 | 00 | 000 00000 | 100111 0100 | 011000 1011 |
| D1.0 | 01 | 000 00001 | 011101 0100 | 100010 1011 |
| D2.0 | 02 | 000 00010 | 101101 0100 | 010010 1011 |
| D3.0 | 03 | 000 00011 | 110001 1011 | 110001 0100 |
| D4.0 | 04 | 000 00100 | 110101 0100 | 001010 1011 |
| D5.0 | 05 | 000 00101 | 101001 1011 | 101001 0100 |
| D6.0 | 06 | 000 00110 | 011001 1011 | 011001 0100 |
| D7.0 | 07 | 000 00111 | 111000 1011 | 000111 0100 |

*Table 4–2. Valid Data Code-Groups   (Part 1 of 9)*

| Code-Group Name | Octet Value | Octet Bits | Current RD– | Current RD+ |
|---|---|---|---|---|
| | | **HGF EDCBA** | **abcdei fghj** | |
| D8.0 | 08 | 000 01000 | 111001 0100 | 000110 1011 |
| D9.0 | 09 | 000 01001 | 100101 1011 | 100101 0100 |
| D10.0 | 0A | 000 01010 | 010101 1011 | 010101 0100 |
| D11.0 | 0B | 000 01011 | 110100 1011 | 110100 0100 |
| D12.0 | 0C | 000 01100 | 001101 1011 | 001101 0100 |
| D13.0 | 0D | 000 01101 | 101100 1011 | 101100 0100 |
| D14.0 | 0E | 000 01110 | 011100 1011 | 011100 0100 |
| D15.0 | 0F | 000 01111 | 010111 0100 | 101000 1011 |
| D16.0 | 10 | 000 10000 | 011011 0100 | 100100 1011 |
| D17.0 | 11 | 000 10001 | 100011 1011 | 100011 0100 |
| D18.0 | 12 | 000 10010 | 010011 1011 | 010011 0100 |
| D19.0 | 13 | 000 10011 | 110010 1011 | 110010 0100 |
| D20.0 | 14 | 000 10100 | 001011 1011 | 001011 0100 |
| D21.0 | 15 | 000 10101 | 101010 1011 | 101010 0100 |
| D22.0 | 16 | 000 10110 | 011010 1011 | 011010 0100 |
| D23.0 | 17 | 000 10111 | 111010 0100 | 000101 1011 |
| D24.0 | 18 | 000 11000 | 110011 0100 | 001100 1011 |
| D25.0 | 19 | 000 11001 | 100110 1011 | 100110 0100 |
| D26.0 | 1A | 000 11010 | 010110 1011 | 010110 0100 |
| D27.0 | 1B | 000 11011 | 110110 0100 | 001001 1011 |
| D28.0 | 1C | 000 11100 | 001110 1011 | 001110 0100 |
| D29.0 | 1D | 000 11101 | 101110 0100 | 010001 1011 |
| D30.0 | 1E | 000 11110 | 011110 0100 | 100001 1011 |
| D31.0 | 1F | 000 11111 | 101011 0100 | 010100 1011 |
| D0.1 | 20 | 001 00000 | 100111 1001 | 011000 1001 |
| D1.1 | 21 | 001 00001 | 011101 1001 | 100010 1001 |
| D2.1 | 22 | 001 00010 | 101101 1001 | 010010 1001 |
| D3.1 | 23 | 001 00011 | 110001 1001 | 110001 1001 |
| D4.1 | 24 | 001 00100 | 110101 1001 | 001010 1001 |
| D5.1 | 25 | 001 00101 | 101001 1001 | 101001 1001 |
| D6.1 | 26 | 001 00110 | 011001 1001 | 011001 1001 |
| D7.1 | 27 | 001 00111 | 111000 1001 | 000111 1001 |
| D8.1 | 28 | 001 01000 | 111001 1001 | 000110 1001 |

*Table 4–2. Valid Data Code-Groups   (Part 2 of 9)*

| Table 4–2. Valid Data Code-Groups   (Part 3 of 9) | | | | |
|---|---|---|---|---|
| Code-Group Name | Octet Value | Octet Bits | Current RD– | Current RD+ |
| | | HGF EDCBA | abcdei fghj | |
| D9.1 | 29 | 001 01001 | 100101 1001 | 100101 1001 |
| D10.1 | 2A | 001 01010 | 010101 1001 | 010101 1001 |
| D11.1 | 2B | 001 01011 | 110100 1001 | 110100 1001 |
| D12.1 | 2C | 001 01100 | 001101 1001 | 001101 1001 |
| D13.1 | 2D | 001 01101 | 101100 1001 | 101100 1001 |
| D14.1 | 2E | 001 01110 | 011100 1001 | 011100 1001 |
| D15.1 | 2F | 001 01111 | 010111 1001 | 101000 1001 |
| D16.1 | 30 | 001 10000 | 011011 1001 | 100100 1001 |
| D17.1 | 31 | 001 10001 | 100011 1001 | 100011 1001 |
| D18.1 | 32 | 001 10010 | 010011 1001 | 010011 1001 |
| D19.1 | 33 | 001 10011 | 110010 1001 | 110010 1001 |
| D20.1 | 34 | 001 10100 | 001011 1001 | 001011 1001 |
| D21.1 | 35 | 001 10101 | 101010 1001 | 101010 1001 |
| D22.1 | 36 | 001 10110 | 011010 1001 | 011010 1001 |
| D23.1 | 37 | 001 10111 | 111010 1001 | 000101 1001 |
| D24.1 | 38 | 001 11000 | 110011 1001 | 001100 1001 |
| D25.1 | 39 | 001 11001 | 100110 1001 | 100110 1001 |
| D26.1 | 3A | 001 11010 | 010110 1001 | 010110 1001 |
| D27.1 | 3B | 001 11011 | 110110 1001 | 001001 1001 |
| D28.1 | 3C | 001 11100 | 001110 1001 | 001110 1001 |
| D29.1 | 3D | 001 11101 | 101110 1001 | 010001 1001 |
| D30.1 | 3E | 001 11110 | 011110 1001 | 100001 1001 |
| D31.1 | 3F | 001 11111 | 101011 1001 | 010100 1001 |
| D0.2 | 40 | 010 00000 | 100111 0101 | 011000 0101 |
| D1.2 | 41 | 010 00001 | 011101 0101 | 100010 0101 |
| D2.2 | 42 | 010 00010 | 101101 0101 | 010010 0101 |
| D3.2 | 43 | 010 00011 | 110001 0101 | 110001 0101 |
| D4.2 | 44 | 010 00100 | 110101 0101 | 001010 0101 |
| D5.2 | 45 | 010 00101 | 101001 0101 | 101001 0101 |
| D6.2 | 46 | 010 00110 | 011001 0101 | 011001 0101 |
| D7.2 | 47 | 010 00111 | 111000 0101 | 000111 0101 |
| D8.2 | 48 | 010 01000 | 111001 0101 | 000110 0101 |
| D9.2 | 49 | 010 01001 | 100101 0101 | 100101 0101 |

| Code-Group Name | Octet Value | Octet Bits | Current RD– | Current RD+ |
|---|---|---|---|---|
| | | HGF EDCBA | abcdei fghj | |
| D10.2 | 4A | 010 01010 | 010101 0101 | 010101 0101 |
| D11.2 | 4B | 010 01011 | 110100 0101 | 110100 0101 |
| D12.2 | 4C | 010 01100 | 001101 0101 | 001101 0101 |
| D13.2 | 4D | 010 01101 | 101100 0101 | 101100 0101 |
| D14.2 | 4E | 010 01110 | 011100 0101 | 011100 0101 |
| D15.2 | 4F | 010 01111 | 010111 0101 | 101000 0101 |
| D16.2 | 50 | 010 10000 | 011011 0101 | 100100 0101 |
| D17.2 | 51 | 010 10001 | 100011 0101 | 100011 0101 |
| D18.2 | 52 | 010 10010 | 010011 0101 | 010011 0101 |
| D19.2 | 53 | 010 10011 | 110010 0101 | 110010 0101 |
| D20.2 | 54 | 010 10100 | 001011 0101 | 001011 0101 |
| D21.2 | 55 | 010 10101 | 101010 0101 | 101010 0101 |
| D22.2 | 56 | 010 10110 | 011010 0101 | 011010 0101 |
| D23.2 | 57 | 010 10111 | 111010 0101 | 000101 0101 |
| D24.2 | 58 | 010 11000 | 110011 0101 | 001100 0101 |
| D25.2 | 59 | 010 11001 | 100110 0101 | 100110 0101 |
| D26.2 | 5A | 010 11010 | 010110 0101 | 010110 0101 |
| D27.2 | 5B | 010 11011 | 110110 0101 | 001001 0101 |
| D28.2 | 5C | 010 11100 | 001110 0101 | 001110 0101 |
| D29.2 | 5D | 010 11101 | 101110 0101 | 010001 0101 |
| D30.2 | 5E | 010 11110 | 011110 0101 | 100001 0101 |
| D31.2 | 5F | 010 11111 | 101011 0101 | 010100 0101 |
| D0.3 | 60 | 011 00000 | 100111 0011 | 011000 1100 |
| D1.3 | 61 | 011 00001 | 011101 0011 | 100010 1100 |
| D2.3 | 62 | 011 00010 | 101101 0011 | 010010 1100 |
| D3.3 | 63 | 011 00011 | 110001 1100 | 110001 0011 |
| D4.3 | 64 | 011 00100 | 110101 0011 | 001010 1100 |
| D5.3 | 65 | 011 00101 | 101001 1100 | 101001 0011 |
| D6.3 | 66 | 011 00110 | 011001 1100 | 011001 0011 |
| D7.3 | 67 | 011 00111 | 111000 1100 | 000111 0011 |
| D8.3 | 68 | 011 01000 | 111001 0011 | 000110 1100 |
| D9.3 | 69 | 011 01001 | 100101 1100 | 100101 0011 |
| D10.3 | 6A | 011 01010 | 010101 1100 | 010101 0011 |

*Table 4–2. Valid Data Code-Groups (Part 4 of 9)*

*Table 4–2. Valid Data Code-Groups   (Part 5 of 9)*

| Code-Group Name | Octet Value | Octet Bits | Current RD– | Current RD+ |
|---|---|---|---|---|
| | | HGF EDCBA | abcdei fghj | |
| D11.3 | 6B | 011 01011 | 110100 1100 | 110100 0011 |
| D12.3 | 6C | 011 01100 | 001101 1100 | 001101 0011 |
| D13.3 | 6D | 011 01101 | 101100 1100 | 101100 0011 |
| D14.3 | 6E | 011 01110 | 011100 1100 | 011100 0011 |
| D15.3 | 6F | 011 01111 | 010111 0011 | 101000 1100 |
| D16.3 | 70 | 011 10000 | 011011 0011 | 100100 1100 |
| D17.3 | 71 | 011 10001 | 100011 1100 | 100011 0011 |
| D18.3 | 72 | 011 10010 | 010011 1100 | 010011 0011 |
| D19.3 | 73 | 011 10011 | 110010 1100 | 110010 0011 |
| D20.3 | 74 | 011 10100 | 001011 1100 | 001011 0011 |
| D21.3 | 75 | 011 10101 | 101010 1100 | 101010 0011 |
| D22.3 | 76 | 011 10110 | 011010 1100 | 011010 0011 |
| D23.3 | 77 | 011 10111 | 111010 0011 | 000101 1100 |
| D24.3 | 78 | 011 11000 | 110011 0011 | 001100 1100 |
| D25.3 | 79 | 011 11001 | 100110 1100 | 100110 0011 |
| D26.3 | 7A | 011 11010 | 010110 1100 | 010110 0011 |
| D27.3 | 7B | 011 11011 | 110110 0011 | 001001 1100 |
| D28.3 | 7C | 011 11100 | 001110 1100 | 001110 0011 |
| D29.3 | 7D | 011 11101 | 101110 0011 | 010001 1100 |
| D30.3 | 7E | 011 11110 | 011110 0011 | 100001 1100 |
| D31.3 | 7F | 011 11111 | 101011 0011 | 010100 1100 |
| D0.4 | 80 | 100 00000 | 100111 0010 | 011000 1101 |
| D1.4 | 81 | 100 00001 | 011101 0010 | 100010 1101 |
| D2.4 | 82 | 100 00010 | 101101 0010 | 010010 1101 |
| D3.4 | 83 | 100 00011 | 110001 1101 | 110001 0010 |
| D4.4 | 84 | 100 00100 | 110101 0010 | 001010 1101 |
| D5.4 | 85 | 100 00101 | 101001 1101 | 101001 0010 |
| D6.4 | 86 | 100 00110 | 011001 1101 | 011001 0010 |
| D7.4 | 87 | 100 00111 | 111000 1101 | 000111 0010 |
| D8.4 | 88 | 100 01000 | 111001 0010 | 000110 1101 |
| D9.4 | 89 | 100 01001 | 100101 1101 | 100101 0010 |
| D10.4 | 8A | 100 01010 | 010101 1101 | 010101 0010 |
| D11.4 | 8B | 100 01011 | 110100 1101 | 110100 0010 |

| Table 4–2. Valid Data Code-Groups (Part 6 of 9) | | | | |
|---|---|---|---|---|
| **Code-Group Name** | **Octet Value** | **Octet Bits** | **Current RD–** | **Current RD+** |
| | | **HGF EDCBA** | **abcdei fghj** | |
| D12.4 | 8C | 100 01100 | 001101 1101 | 001101 0010 |
| D13.4 | 8D | 100 01101 | 101100 1101 | 101100 0010 |
| D14.4 | 8E | 100 01110 | 011100 1101 | 011100 0010 |
| D15.4 | 8F | 100 01111 | 010111 0010 | 101000 1101 |
| D16.4 | 90 | 100 10000 | 011011 0010 | 100100 1101 |
| D17.4 | 91 | 100 10001 | 100011 1101 | 100011 0010 |
| D18.4 | 92 | 100 10010 | 010011 1101 | 010011 0010 |
| D19.4 | 93 | 100 10011 | 110010 1101 | 110010 0010 |
| D20.4 | 94 | 100 10100 | 001011 1101 | 001011 0010 |
| D21.4 | 95 | 100 10101 | 101010 1101 | 101010 0010 |
| D22.4 | 96 | 100 10110 | 011010 1101 | 011010 0010 |
| D23.4 | 97 | 100 10111 | 111010 0010 | 000101 1101 |
| D24.4 | 98 | 100 11000 | 110011 0010 | 001100 1101 |
| D25.4 | 99 | 100 11001 | 100110 1101 | 100110 0010 |
| D26.4 | 9A | 100 11010 | 010110 1101 | 010110 0010 |
| D27.4 | 9B | 100 11011 | 110110 0010 | 001001 1101 |
| D28.4 | 9C | 100 11100 | 001110 1101 | 001110 0010 |
| D29.4 | 9D | 100 11101 | 101110 0010 | 010001 1101 |
| D30.4 | 9E | 100 11110 | 011110 0010 | 100001 1101 |
| D31.4 | 9F | 100 11111 | 101011 0010 | 010100 1101 |
| D0.5 | A0 | 101 00000 | 100111 1010 | 011000 1010 |
| D1.5 | A1 | 101 00001 | 011101 1010 | 100010 1010 |
| D2.5 | A2 | 101 00010 | 101101 1010 | 010010 1010 |
| D3.5 | A3 | 101 00011 | 110001 1010 | 110001 1010 |
| D4.5 | A4 | 101 00100 | 110101 1010 | 001010 1010 |
| D5.5 | A5 | 101 00101 | 101001 1010 | 101001 1010 |
| D6.5 | A6 | 101 00110 | 011001 1010 | 011001 1010 |
| D7.5 | A7 | 101 00111 | 111000 1010 | 000111 1010 |
| D8.5 | A8 | 101 01000 | 111001 1010 | 000110 1010 |
| D9.5 | A9 | 101 01001 | 100101 1010 | 100101 1010 |
| D10.5 | AA | 101 01010 | 010101 1010 | 010101 1010 |
| D11.5 | AB | 101 01011 | 110100 1010 | 110100 1010 |
| D12.5 | AC | 101 01100 | 001101 1010 | 001101 1010 |

*Table 4–2. Valid Data Code-Groups  (Part 7 of 9)*

| Code-Group Name | Octet Value | Octet Bits | Current RD− | Current RD+ |
|---|---|---|---|---|
| | | HGF EDCBA | abcdei fghj | |
| D13.5 | AD | 101 01101 | 101100 1010 | 101100 1010 |
| D14.5 | AE | 101 01110 | 011100 1010 | 011100 1010 |
| D15.5 | AF | 101 01111 | 010111 1010 | 101000 1010 |
| D16.5 | B0 | 101 10000 | 011011 1010 | 100100 1010 |
| D17.5 | B1 | 101 10001 | 100011 1010 | 100011 1010 |
| D18.5 | B2 | 101 10010 | 010011 1010 | 010011 1010 |
| D19.5 | B3 | 101 10011 | 110010 1010 | 110010 1010 |
| D20.5 | B4 | 101 10100 | 001011 1010 | 001011 1010 |
| D21.5 | B5 | 101 10101 | 101010 1010 | 101010 1010 |
| D22.5 | B6 | 101 10110 | 011010 1010 | 011010 1010 |
| D23.5 | B7 | 101 10111 | 111010 1010 | 000101 1010 |
| D24.5 | B8 | 101 11000 | 110011 1010 | 001100 1010 |
| D25.5 | B9 | 101 11001 | 100110 1010 | 100110 1010 |
| D26.5 | BA | 101 11010 | 010110 1010 | 010110 1010 |
| D27.5 | BB | 101 11011 | 110110 1010 | 001001 1010 |
| D28.5 | BC | 101 11100 | 001110 1010 | 001110 1010 |
| D29.5 | BD | 101 11101 | 101110 1010 | 010001 1010 |
| D30.5 | BE | 101 11110 | 011110 1010 | 100001 1010 |
| D31.5 | BF | 101 11111 | 101011 1010 | 010100 1010 |
| D0.6 | C0 | 110 00000 | 100111 0110 | 011000 0110 |
| D1.6 | C1 | 110 00001 | 011101 0110 | 100010 0110 |
| D2.6 | C2 | 110 00010 | 101101 0110 | 010010 0110 |
| D3.6 | C3 | 110 00011 | 110001 0110 | 110001 0110 |
| D4.6 | C4 | 110 00100 | 110101 0110 | 001010 0110 |
| D5.6 | C5 | 110 00101 | 101001 0110 | 101001 0110 |
| D6.6 | C6 | 110 00110 | 011001 0110 | 011001 0110 |
| D7.6 | C7 | 110 00111 | 111000 0110 | 000111 0110 |
| D8.6 | C8 | 110 01000 | 111001 0110 | 000110 0110 |
| D9.6 | C9 | 110 01001 | 100101 0110 | 100101 0110 |
| D10.6 | CA | 110 01010 | 010101 0110 | 010101 0110 |
| D11.6 | CB | 110 01011 | 110100 0110 | 110100 0110 |
| D12.6 | CC | 110 01100 | 001101 0110 | 001101 0110 |
| D13.6 | CD | 110 01101 | 101100 0110 | 101100 0110 |

*Table 4–2. Valid Data Code-Groups (Part 8 of 9)*

| Code-Group Name | Octet Value | Octet Bits | Current RD– | Current RD+ |
|---|---|---|---|---|
| | | HGF EDCBA | abcdei fghj | |
| D14.6 | CE | 110 01110 | 011100 0110 | 011100 0110 |
| D15.6 | CF | 110 01111 | 010111 0110 | 101000 0110 |
| D16.6 | D0 | 110 10000 | 011011 0110 | 100100 0110 |
| D17.6 | D1 | 110 10001 | 100011 0110 | 100011 0110 |
| D18.6 | D2 | 110 10010 | 010011 0110 | 010011 0110 |
| D19.6 | D3 | 110 10011 | 110010 0110 | 110010 0110 |
| D20.6 | D4 | 110 10100 | 001011 0110 | 001011 0110 |
| D21.6 | D5 | 110 10101 | 101010 0110 | 101010 0110 |
| D22.6 | D6 | 110 10110 | 011010 0110 | 011010 0110 |
| D23.6 | D7 | 110 10111 | 111010 0110 | 000101 0110 |
| D24.6 | D8 | 110 11000 | 110011 0110 | 001100 0110 |
| D25.6 | D9 | 110 11001 | 100110 0110 | 100110 0110 |
| D26.6 | DA | 110 11010 | 010110 0110 | 010110 0110 |
| D27.6 | DB | 110 11011 | 110110 0110 | 001001 0110 |
| D28.6 | DC | 110 11100 | 001110 0110 | 001110 0110 |
| D29.6 | DD | 110 11101 | 101110 0110 | 010001 0110 |
| D30.6 | DE | 110 11110 | 011110 0110 | 100001 0110 |
| D31.6 | DF | 110 11111 | 101011 0110 | 010100 0110 |
| D0.7 | E0 | 111 00000 | 100111 0001 | 011000 1110 |
| D1.7 | E1 | 111 00001 | 011101 0001 | 100010 1110 |
| D2.7 | E2 | 111 00010 | 101101 0001 | 010010 1110 |
| D3.7 | E3 | 111 00011 | 110001 1110 | 110001 0001 |
| D4.7 | E4 | 111 00100 | 110101 0001 | 001010 1110 |
| D5.7 | E5 | 111 00101 | 101001 1110 | 101001 0001 |
| D6.7 | E6 | 111 00110 | 011001 1110 | 011001 0001 |
| D7.7 | E7 | 111 00111 | 111000 1110 | 000111 0001 |
| D8.7 | E8 | 111 01000 | 111001 0001 | 000110 1110 |
| D9.7 | E9 | 111 01001 | 100101 1110 | 100101 0001 |
| D10.7 | EA | 111 01010 | 010101 1110 | 010101 0001 |
| D11.7 | EB | 111 01011 | 110100 1110 | 110100 1000 |
| D12.7 | EC | 111 01100 | 001101 1110 | 001101 0001 |
| D13.7 | ED | 111 01101 | 101100 1110 | 101100 1000 |
| D14.7 | EE | 111 01110 | 011100 1110 | 011100 1000 |

| Table 4–2. Valid Data Code-Groups (Part 9 of 9) | | | | |
|---|---|---|---|---|
| Code-Group Name | Octet Value | Octet Bits | Current RD– | Current RD+ |
| | | HGF EDCBA | abcdei fghj | |
| D15.7 | EF | 111 01111 | 010111 0001 | 101000 1110 |
| D16.7 | F0 | 111 10000 | 011011 0001 | 100100 1110 |
| D17.7 | F1 | 111 10001 | 100011 0111 | 100011 0001 |
| D18.7 | F2 | 111 10010 | 010011 0111 | 010011 0001 |
| D19.7 | F3 | 111 10011 | 110010 1110 | 110010 0001 |
| D20.7 | F4 | 111 10100 | 001011 0111 | 001011 0001 |
| D21.7 | F5 | 111 10101 | 101010 1110 | 101010 0001 |
| D22.7 | F6 | 111 10110 | 011010 1110 | 011010 0001 |
| D23.7 | F7 | 111 10111 | 111010 0001 | 000101 1110 |
| D24.7 | F8 | 111 11000 | 110011 0001 | 001100 1110 |
| D25.7 | F9 | 111 11001 | 100110 1110 | 100110 0001 |
| D26.7 | FA | 111 11010 | 010110 1110 | 010110 0001 |
| D27.7 | FB | 111 11011 | 110110 0001 | 001001 1110 |
| D28.7 | FC | 111 11100 | 001110 1110 | 001110 0001 |
| D29.7 | FD | 111 11101 | 101110 0001 | 010001 1110 |
| D30.7 | FE | 111 11110 | 011110 0001 | 100001 1110 |
| D31.7 | FF | 111 11111 | 101011 0001 | 010100 1110 |

# Document Revision History

Table 4–3 shows the revision history for this document.

| Table 4–3. Document Revision History | | |
|---|---|---|
| Date and Document Version | Changes Made | Summary of Changes |
| May 2007 v1.0 | Initial Release | N/A |

# Section II. Clock Management

This section provides information on clock management in Arria™ GX devices. It describes the enhanced and fast phase-locked loops (PLLs) that support clock management and synthesis for on-chip clock management, external system clock management, and high-speed I/O interfaces.

This section includes the following chapter:

■ Chapter 5, PLLs in Arria GX Devices

**Revision History**
Refer to each chapter for its own specific revision history. For information on when each chapter was updated, refer to the Chapter Revision Dates section, which appears in the full handbook.

# 5. PLLs in Arria GX Devices

## Introduction

Arria™ GX device phase-locked loops (PLLs) provide robust clock management and synthesis for device clock management, external system clock management, and high-speed I/O interfaces. These PLLs are highly versatile and can be used as a zero delay buffer, a jitter attenuator, low skew fan out buffer, or a frequency synthesizer.

Arria GX devices feature up to four enhanced PLLs and up to four fast PLLs. Both enhanced and fast PLLs are feature rich, supporting advanced capabilities such as clock switchover, reconfigurable phase shift, PLL reconfiguration, and reconfigurable bandwidth. You can use PLLs for general-purpose clock management, supporting multiplication, phase shifting, and programmable duty cycle. In addition, enhanced PLLs support external clock feedback mode, spread-spectrum clocking, and counter cascading. Fast PLLs offer high-speed outputs to manage high-speed differential I/O interfaces.

Arria GX devices also support power-down mode where clock networks that are not being used can easily be turned off, reducing overall power consumption of the device. In addition, Arria GX PLLs support dynamic selection of the PLL input clock from up to five possible sources, giving you the flexibility to choose from multiple (up to four) clock sources to feed the primary and secondary clock input ports.

The Altera® Quartus® II software enables the PLLs and their features without requiring any external devices.

This chapter contains the following sections:

Table 5–1 shows the PLLs available for each Arria GX device.

| Table 5–1. Arria GX Device PLL Availability Note (1) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Device** | **Fast PLLs** | | | | **Enhanced PLLs** | | | |
| | **1** | **2** | **7** | **8** | **5** | **6** | **11** | **12** |
| EP1AGX20 (2) | ✓ | ✓ | — | — | ✓ | ✓ | — | — |
| EP1AGX35 (2) | ✓ | ✓ | — | — | ✓ | ✓ | — | — |
| EP1AGX50 (2) | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| EP1AGX60 (3) | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| EP1AGX90 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

*Notes for Table 5–1:*
(1) The global or regional clocks in a fast PLL's transceiver block can drive the fast PLL input. A pin or other PLL must drive the global or regional source. The source cannot be driven by internally generated logic before driving the fast PLL.
(2) EP1AGX20, EP1AGX35, EP1AGX50 and EP1AGX60 devices only have two fast PLLs (PLLs 1 and 2).
(3) EP1AGX60 devices in F484 and F780 devices have two fast PLLs (PLL 1 and 2) and two enhanced PLLs. Arria GX devices in the F1152 package support all eight PLLs.

Table 5–2 shows the enhanced PLL and fast PLL features in Arria GX devices.

| Table 5–2. Arria GX PLL Features   (Part 1 of 2) | | |
|---|---|---|
| **Feature** | **Enhanced PLL** | **Fast PLL** |
| Clock multiplication and division | *m/(n* post-scale counter) *(1)* | *m/(n* post-scale counter) *(2)* |
| Phase shift | Down to 125-ps increments *(3)* | Down to 125-ps increments *(3)* |
| Clock switchover | ✓ | ✓*(4)* |
| PLL reconfiguration | ✓ | ✓ |
| Reconfigurable bandwidth | ✓ | ✓ |
| Spread-spectrum clocking | ✓ | — |
| Programmable duty cycle | ✓ | ✓ |
| Number of clock outputs per PLL *(5)* | 6 | 4 |
| Number of dedicated external clock outputs per PLL | Three differential or six single-ended | *(6)* |

### Table 5–2. Arria GX PLL Features   (Part 2 of 2)

| Feature | Enhanced PLL | Fast PLL |
|---|---|---|
| Number of feedback clock inputs per PLL | 1 *(7)* | — |

*Notes to Table 5–2:*

(1)  For enhanced PLLs, *m* and *n* range from 1 to 512 with a 50% duty cycle. Post-scale counters range from 1 to 512 with a 50% duty cycle. For non-50% duty-cycle clock outputs, post-scale counters range from 1 to 256.

(2)  Fast PLLs can range from 1 to 4. The post-scale and *m* counters range from 1 to 32. For non-50% duty-cycle clock outputs, post-scale counters range from 1 to 16.

(3)  The smallest phase shift is determined by the voltage controlled oscillator (VCO) period divided by eight. The supported phase-shift range is from 125 to 250 ps. Arria GX devices can shift all output frequencies in increments of at least 45. Smaller degree increments are possible depending on the frequency and divide parameters. For non-50% duty cycle clock outputs post-scale counters range from 1 to 256.

(4)  Arria GX fast PLLs only support manual clock switchover.

(5)  Clock outputs can be driven to internal clock networks or to a pin.

(6)  PLL clock outputs of the fast PLLs can drive to any I/O pin to be used as an external clock output. For high-speed differential I/O pins, the device uses a data channel to generate the transmitter output clock (`txclkout`).

(7)  If the design uses external feedback input pins, you will lose one (or two, if `fbin` is differential) dedicated output clock pins.

Figure 5–1 shows a top-level diagram of the Arria GX device and PLL locations. See "Clock Control Block" on page 5–77 for more information about PLL connections to global and regional clocks networks.

*Figure 5–1. Arria GX PLL Locations*
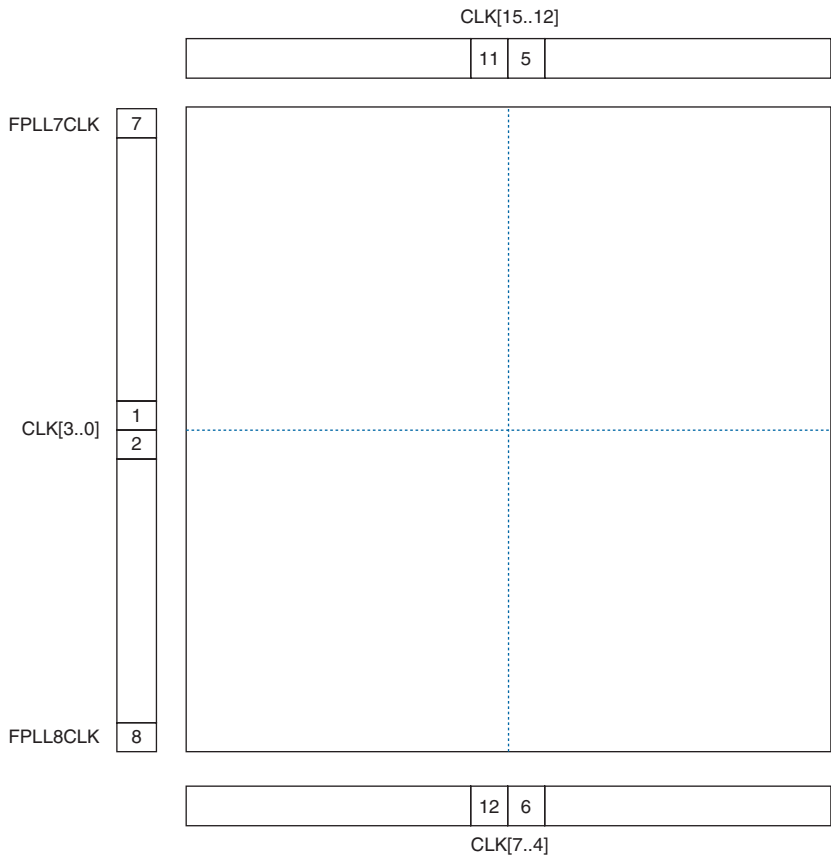


**Notes to Figure 5–1:**
(1) EP1AGX20 and EP1AGX35 devices have two enhanced and two fast PLLs.
(2) EP1AGX50 devices in the F484 package have two enhanced PLLs (5 and 6), two fast PLLs (1 and 2), two enhanced and two fast PLLs (1 and 2) in the F780 package, and four enhanced, four fast PLLs in the F1152 package.
(3) EP1AGX60 devices in the F484 and F780 packages have two enhanced and two fast PLLs, and four enhanced and four fast PLLs in the F1152 package.
(4) EP1AGX60 devices have four enhanced and four fast PLLs in the F1152 package.
(5) The corner fast PLLs (7 and 8) are enabled only in the F1152 package offering.

# Enhanced PLLs

Arria GX devices contain up to four enhanced PLLs with advanced clock management features. The main goal of a PLL is to synchronize the phase and frequency of an internal and external clock to an input reference clock. There are a number of components that comprise a PLL to achieve this phase alignment.

## Enhanced PLL Hardware Overview

Arria GX PLLs align the rising edge of the reference input clock to a feedback clock using the phase-frequency detector (PFD). The falling edges are determined by duty-cycle specifications. The PFD produces an up or down signal that determines whether the VCO needs to operate at a higher or lower frequency.

PFD output is applied to the charge pump and loop filter, which produces a control voltage for setting the VCO frequency. If the PFD produces an up signal, the VCO frequency increases; a down signal decreases the VCO frequency. The PFD outputs these up and down signals to a charge pump. If the charge pump receives an up signal, current is driven into the loop filter. Conversely, if the charge pump receives a down signal, current is drawn from the loop filter.

The loop filter converts these up and down signals to a voltage that is used to bias the VCO. The loop filter also removes glitches from the charge pump and prevents voltage over-shoot, which filters the jitter on the VCO.

The voltage from the loop filter determines how fast the VCO operates. The VCO is implemented as a four-stage differential ring oscillator. A divide counter ($m$) is inserted in the feedback loop to increase the VCO frequency above the input reference frequency. VCO frequency ($f_{VCO}$) is equal to ($m$) times the input reference clock ($f_{REF}$). The input reference clock ($f_{REF}$) to the PFD is equal to the input clock ($f_{IN}$) divided by the pre-scale counter ($n$). Therefore, the feedback clock ($f_{FB}$) applied to one input of the PFD is locked to the $f_{REF}$ that is applied to the other input of the PFD.

The VCO output can feed up to six post-scale counters (C0, C1, C2, C3, C4, and C5). These post-scale counters allow a number of harmonically related frequencies to be produced within the PLL.

Figure 5–2 shows a simplified block diagram of the major components of the Arria GX enhanced PLL. Figure 5–3 shows the enhanced PLL's outputs and dedicated clock outputs.

*Figure 5–2. Arria GX Enhanced PLL Note (3), (4)*
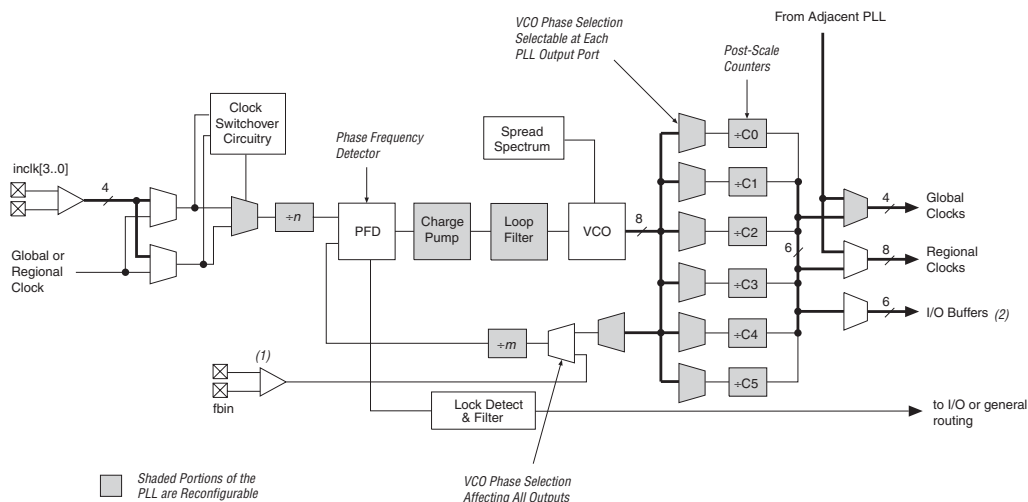


*Notes to Figure 5–2:*
(1) Each clock source can come from any of the four clock pins located on the same side of the device as the PLL.
(2) PLLs 5, 6, 11, and 12 each have six single-ended dedicated clock outputs or three differential dedicated clock outputs.
(3) If the design uses external feedback input pins, you will lose one (or two, if fbin is differential) dedicated output clock pin. Every Arria GX device has at least two enhanced PLLs with one single-ended or differential external feedback input per PLL.
(4) The global or regional clock input can be driven by an output from another PLL, a pin-driven dedicated global or regional clock, or through a clock control block provided the clock control block is fed by an output from another PLL or a pin-driven dedicated global or regional clock. An internally generated global signal cannot drive the PLL.

### External Clock Outputs

Enhanced PLLs 5, 6, 11, and 12 each support up to six single-ended clock outputs (or three differential pairs), as shown in Figure 5–3.

*Figure 5–3. External Clock Outputs for Enhanced PLLs 5, 6, 11, and 12*
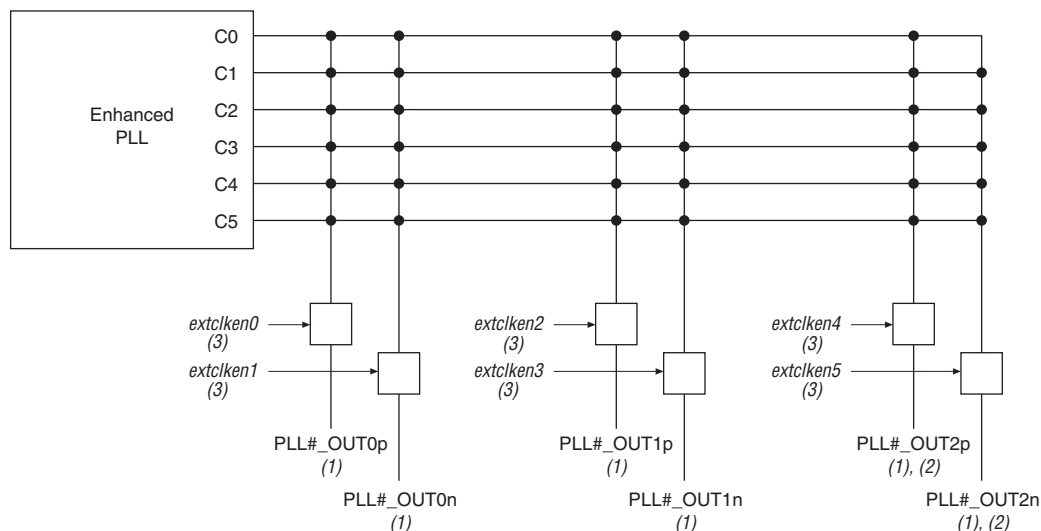


*Notes to Figure 5–3:*
(1) These clock output pins can be fed by any one of the C[5..0] counters.
(2) These clock output pins are used as either external clock outputs or for external feedback. If the design uses external feedback input pins, you will lose one (or two, if fbin is differential) dedicated output clock pin.
(3) These external clock enable signals are available only when using the ALTCLKCTRL megafunction.

Any of the six output counters C[5..0] can feed the dedicated external clock outputs, as shown in Figure 5–5. Therefore, one counter or frequency can drive all output pins available from a given PLL. The dedicated output clock pins (PLL#_OUT) from each enhanced PLL are powered by a separate power pin (for example, VCC_PLL5_OUT, VCC_PLL6_OUT, etc.), reducing the overall output jitter by providing improved isolation from switching I/O pins.

*Figure 5–4. External Clock Output Connectivity to PLL Output Counters for Enhanced PLLs 5, 6, 11, and 12*
*Note (1)*
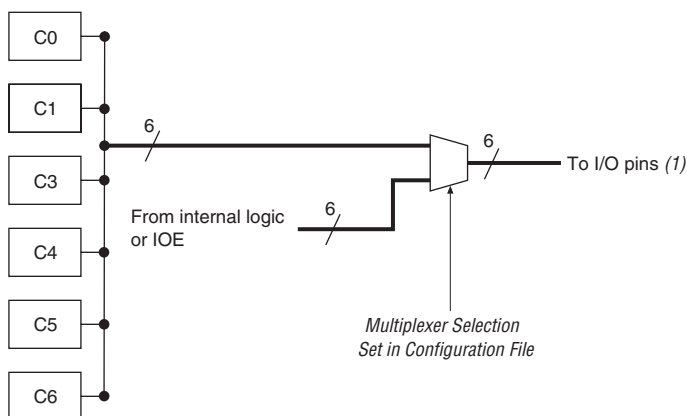


*Note to Figure 5–4:*
(1)    The design can use each external clock output pin as a general-purpose output pin from the logic array. These pins are multiplexed with I/O element (IOE) outputs.

Each pin of a single-ended output pair can either be in phase or 180º out of phase. The Quartus II software places the NOT gate in the design into the IOE to implement 180º phase with respect to the other pin in the pair. The clock output pin pairs support the same I/O standards as standard output pins (in the top and bottom banks) as well as LVDS, LVPECL, differential HSTL, and differential SSTL. See Table 5–5, under "Enhanced PLL Pins" on page 5–11 to determine which I/O standards the enhanced PLL clock pins support.

When in single-ended or differential mode, one power pin supports six single-ended or three differential outputs. Both outputs use the same I/O standard in single-ended mode to maintain performance. You can also use the external clock output pins as user output pins if external enhanced PLL clocking is not needed.

The enhanced PLL can also drive out to any regular I/O pin through the global or regional clock network. For this case, jitter on the output clock is pending characterization

## Enhanced PLL Software Overview

Arria GX enhanced PLLs are enabled in the Quartus II software by using the ALTPLL megafunction. Figure 5–5 shows the available ports (as they are named in the Quartus II ALTPLL megafunction) of the Arria GX enhanced PLL.

*Figure 5–5. Enhanced PLL Ports*
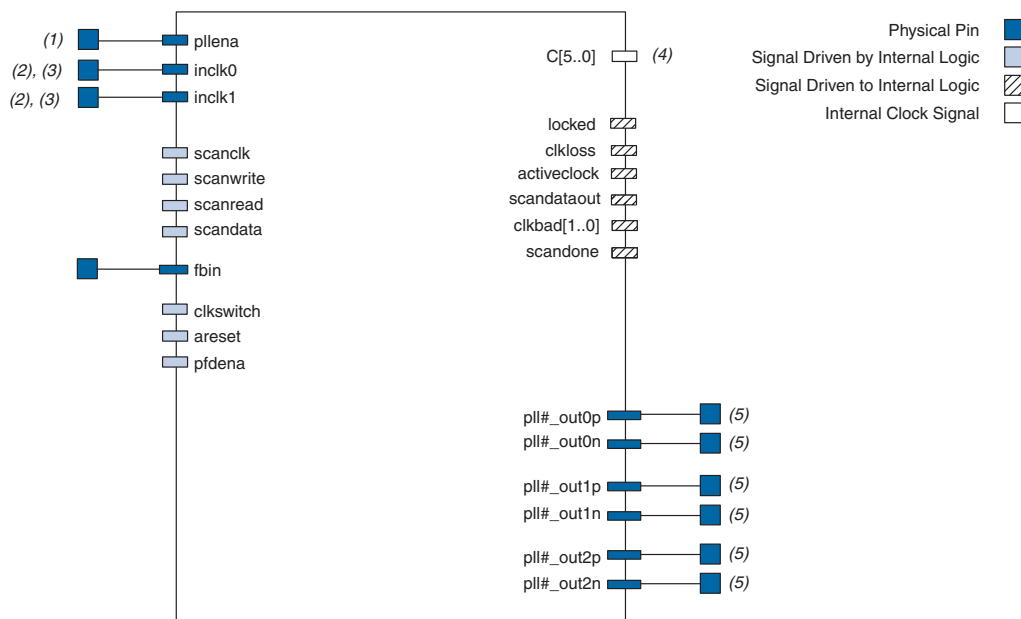


*Notes to Figure 5–5:*
(1)  Enhanced and fast PLLs share this input pin.
(2)  These are either single-ended or differential pins.
(3)  The primary and secondary clock input can be fed from any one of four clock pins located on the same side of the device as the PLL.
(4)  `C[5..0]` can drive to the global or regional clock networks or the dedicated external clock output pins.
(5)  These dedicated output clocks are fed by the `C[5..0]` counters.

Tables 5–3 and 5–4 describe all the enhanced PLL ports.

*Table 5–3. Enhanced PLL Input Signals   (Part 1 of 2)*

| Port | Description | Source | Destination |
|------|-------------|--------|-------------|
| `inclk0` | Primary clock input to the PLL. | Pin or another PLL | counter |
| `inclk1` | Secondary clock input to the PLL. | Pin or another PLL | counter |
| `fbin` | External feedback input to the PLL. | Pin | PFD |
| `pllena` | Enable pin for enabling or disabling all or a set of PLLs. Active high. | Pin | General PLL control signal |
| `clkswitch` | Switch-over signal used to initiate external clock switch-over control. Active high. | Logic array | PLL switch-over circuit |

**Table 5–3. Enhanced PLL Input Signals (Part 2 of 2)**

| Port | Description | Source | Destination |
|------|-------------|--------|-------------|
| areset | Signal used to reset the PLL which resynchronizes all the counter outputs. Active high. | Logic array | General PLL control signal |
| pfdena | Enables the outputs from the phase frequency detector. Active high. | Logic array | PFD |
| scanclk | Serial clock signal for the real-time PLL reconfiguration feature. | Logic array | Reconfiguration circuit |
| scandata | Serial input data stream for the real-time PLL reconfiguration feature. | Logic array | Reconfiguration circuit |
| scanwrite | Enables writing the data in the scan chain into the PLL. Active high. | Logic array | Reconfiguration circuit |
| scanread | Enables scan data to be written into the scan chain. Active high. | Logic array | Reconfiguration circuit |

**Table 5–4. Enhanced PLL Output Signals (Part 1 of 2)**

| Port | Description | Source | Destination |
|------|-------------|--------|-------------|
| C[5..0] | PLL output counters driving regional, global or external clocks. | PLL counter | Internal or external clock |
| pll#_out[2..0]p pll#_out[2..0]n | These are three differential or six single-ended external clock output pins fed from the C[5..0] PLL counters, and every output can be driven by any counter. *p* and *n* are the positive (*p*) and negative (*n*) pins for differential pins. | PLL counter | Pin(s) |
| clkloss | Signal indicating the switch-over circuit detected a switch-over condition. | PLL switch-over circuit | Logic array |
| clkbad[1..0] | Signals indicating which reference clock is no longer toggling. clkbad1 indicates inclk1 status, clkbad0 indicates inclk0 status. 1= good; 0 = bad | PLL switch-over circuit | Logic array |
| locked | Lock or gated lock output from lock detect circuit. Active high. | PLL lock detect | Logic array |

| Table 5–4. Enhanced PLL Output Signals (Part 2 of 2) | | | |
|---|---|---|---|
| **Port** | **Description** | **Source** | **Destination** |
| `activeclock` | Signal to indicate which clock (0 = `inclk0` or 1 = `inclk1`) is driving the PLL. If this signal is low, `inclk0` drives the PLL, If this signal is high, `inclk1` drives the PLL | PLL clock multiplexer | Logic array |
| `scandataout` | Output of the last shift register in the scan chain. | PLL scan chain | Logic array |
| `scandone` | Signal indicating when the PLL has completed reconfiguration. 1 to 0 transition indicates that the PLL has been reconfigured. | PLL scan chain | Logic array |

### Enhanced PLL Pins

Table 5–5 lists the I/O standards support by enhanced PLL clock outputs.

| Table 5–5. I/O Standards Supported for Enhanced PLL Pins *Note (1)* (Part 1 of 2) | | | |
|---|---|---|---|
| **I/O Standard** | **Input inclk** | **fbin** | **Output extclk** |
| LVTTL | ✓ | ✓ | ✓ |
| LVCMOS | ✓ | ✓ | ✓ |
| 2.5 V | ✓ | ✓ | ✓ |
| 1.8 V | ✓ | ✓ | ✓ |
| 1.5 V | ✓ | ✓ | ✓ |
| 3.3-V PCI | ✓ | ✓ | ✓ |
| 3.3-V PCI-X | ✓ | ✓ | ✓ |
| SSTL-2 Class I | ✓ | ✓ | ✓ |
| SSTL-2 Class II | ✓ | ✓ | ✓ |
| SSTL-18 Class I | ✓ | ✓ | ✓ |
| SSTL-18 Class II | ✓ | ✓ | ✓ |
| 1.8-V HSTL Class I | ✓ | ✓ | ✓ |
| 1.8-V HSTL Class II | ✓ | ✓ | ✓ |
| 1.5-V HSTL Class I | ✓ | ✓ | ✓ |
| 1.5-V HSTL Class II | ✓ | ✓ | ✓ |
| Differential SSTL-2 Class I | ✓ | ✓ | ✓ |
| Differential SSTL-2 Class II | ✓ | ✓ | ✓ |

*Table 5–5. I/O Standards Supported for Enhanced PLL Pins Note (1)*
*(Part 2 of 2)*

| I/O Standard | Input inclk | fbin | Output extclk |
|---|:---:|:---:|:---:|
| Differential SSTL-18 Class I | ✓ | ✓ | ✓ |
| Differential SSTL-18 Class II | ✓ | ✓ | ✓ |
| 1.8-V differential HSTL Class I | ✓ | ✓ | ✓ |
| 1.8-V differential HSTL Class II | ✓ | ✓ | ✓ |
| 1.5-V differential HSTL Class I | ✓ | ✓ | ✓ |
| 1.5-V differential HSTL Class II | ✓ | ✓ | ✓ |
| LVDS | ✓ | ✓ | ✓ |
| HyperTransport technology | — | — | — |
| Differential LVPECL | ✓ | ✓ | ✓ |

*Note to Table 5–5:*
(1) The enhanced PLL external clock output bank does not allow a mixture of both single-ended and differential I/O standards.

Table 5–6 shows the physical pins and their purpose for Arria GX enhanced PLLs. For inclk port connections to pins, see "Clock Control Block" on page 5–77.

*Table 5–6. Arria GX Enhanced PLL Pins Note (1)  (Part 1 of 2)*

| Pin | Description |
|---|---|
| CLK4p/n | Single-ended or differential pins that can drive the inclk port for PLLs 6 or 12. |
| CLK5p/n | Single-ended or differential pins that can drive the inclk port for PLLs 6 or 12. |
| CLK6p/n | Single-ended or differential pins that can drive the inclk port for PLLs 6 or 12. |
| CLK7p/n | Single-ended or differential pins that can drive the inclk port for PLLs 6 or 12. |
| CLK12p/ | Single-ended or differential pins that can drive the inclk port for PLLs 5 or 11. |
| CLK13p/ | Single-ended or differential pins that can drive the inclk port for PLLs 5 or 11. |
| CLK14p/n | Single-ended or differential pins that can drive the inclk port for PLLs 5 or 11. |
| CLK15p/n | Single-ended or differential pins that can drive the inclk port for PLLs 5 or 11. |
| PLL5_FBp/n | Single-ended or differential pins that can drive the fbin port for PLL 5. |
| PLL6_FBp/n | Single-ended or differential pins that can drive the fbin port for PLL 6. |
| PLL11_FBp/n | Single-ended or differential pins that can drive the fbin port for PLL 11. |
| PLL12_FBp/n | Single-ended or differential pins that can drive the fbin port for PLL 12. |
| pllena | Dedicated input pin that drives the pllena port of all or a set of PLLs. If you do not use this pin, connect it to ground. |

| Pin | Description |
|-----|-------------|
| `PLL5_OUT[2..0]p/n` | Single-ended or differential pins driven by `C[5..0]` ports from PLL 5. |
| `PLL6_OUT[2..0]p/n` | Single-ended or differential pins driven by `C[5..0]` ports from PLL 6. |
| `PLL11_OUT[2..0]p/n` | Single-ended or differential pins driven by `C[5..0]` ports from PLL 11. |
| `PLL12_OUT[2..0]p/n` | Single-ended or differential pins driven by `C[5..0]` ports from PLL 12. |
| `VCCA_PLL5` | Analog power for PLL 5. You must connect this pin to 1.2 V, even if the PLL is not used. |
| `GNDA_PLL5` | Analog ground for PLL 5. You can connect this pin to the GND plane on the board. |
| `VCCA_PLL6` | Analog power for PLL 6. You must connect this pin to 1.2 V, even if the PLL is not used. |
| `GNDA_PLL6` | Analog ground for PLL 6. You can connect this pin to the GND plane on the board. |
| `VCCA_PLL11` | Analog power for PLL 11. You must connect this pin to 1.2 V, even if the PLL is not used. |
| `GNDA_PLL11` | Analog ground for PLL 11. You can connect this pin to the GND plane on the board. |
| `VCCA_PLL12` | Analog power for PLL 12. You must connect this pin to 1.2 V, even if the PLL is not used. |
| `GNDA_PLL12` | Analog ground for PLL 12. You can connect this pin to the GND plane on the board. |
| `VCCD_PLL` | Digital power for PLLs. You must connect this pin to 1.2 V, even if the PLL is not used. |
| `VCC_PLL5_OUT` | External clock output $V_{CCIO}$ power for PLL5_OUT0p, PLL5_OUT0n, PLL5_OUT1p, PLL5_OUT1n, PLL5_OUT2p, and PLL5_OUT2n outputs from PLL 5. |
| `VCC_PLL6_OUT` | External clock output $V_{CCIO}$ power for PLL6_OUT0p, PLL6_OUT0n, PLL6_OUT1p, PLL6_OUT1n and PLL6_OUT2p, PLL6_OUT2n outputs from PLL 6. |
| `VCC_PLL11_OUT` | External clock output $V_{CCIO}$ power for PLL11_OUT0p, PLL11_OUT0n, PLL11_OUT1p, PLL11_OUT1n and PLL11_OUT2p, PLL11_OUT2n outputs from PLL 11. |
| `VCC_PLL12_OUT` | External clock output $V_{CCIO}$ power for PLL12_OUT0p, PLL12_OUT0n, PLL12_OUT1p, PLL12_OUT1n and PLL12_OUT2p, PLL12_OUT2n outputs from PLL 12. |

*Table 5–6. Arria GX Enhanced PLL Pins Note (1)  (Part 2 of 2)*

*Note to Table 5–6:*
(1)   The negative leg pins (`CLKn`, `PLL_FBn`, and `PLL_OUTn`) are only required with differential signaling.

# Fast PLLs

Arria GX devices contain up to four fast PLLs. Fast PLLs have high-speed differential I/O interface capability along with general purpose features.

## Fast PLL Hardware Overview

Figure 5–6 shows a diagram of the fast PLL for Arria GX devices.

*Figure 5–6. Arria GX Fast PLL Block Diagram*
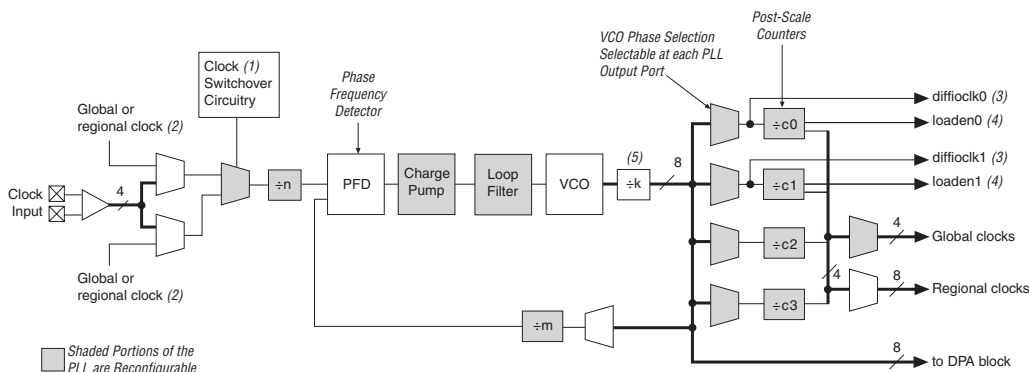


*Notes to Figure 5–6:*
(1) Arria GX fast PLLs only support manual clock switchover.
(2) The global or regional clock input can be driven by an output from another PLL, a pin-driven dedicated global or regional clock, or through a clock control block provided the clock control block is fed by an output from another PLL or a pin-driven dedicated global or regional clock.
(3) In high-speed differential I/O support mode, this high-speed PLL clock feeds SERDES. Arria GX devices only support one rate of data transfer per fast PLL in high-speed differential I/O support mode.
(4) This signal is a high-speed differential I/O support SERDES control signal.
(5) If the design enables this ÷2 counter, the device can use a VCO frequency range of 150 to 520 MHz.

### External Clock Outputs

Each fast PLL supports differential or single-ended outputs for source-synchronous transmitters or general-purpose external clocks. There are no dedicated external clock output pins. The fast PLL global or regional outputs can drive any I/O pin as an external clock output pin. The I/O standards supported by any particular bank determines what standards are possible for an external clock output driven by the fast PLL in that bank.

For more information, see the *Selectable I/O Standards in Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook.*

## Fast PLL Software Overview

Arria GX fast PLLs are enabled in the Quartus II software by using the ALTPLL megafunction. Figure 5–7 shows the available ports (as they are named in the Quartus II ALTPLL megafunction) of the Arria GX fast PLL.

*Figure 5–7. Arria GX Fast PLL Ports and Physical Destinations*
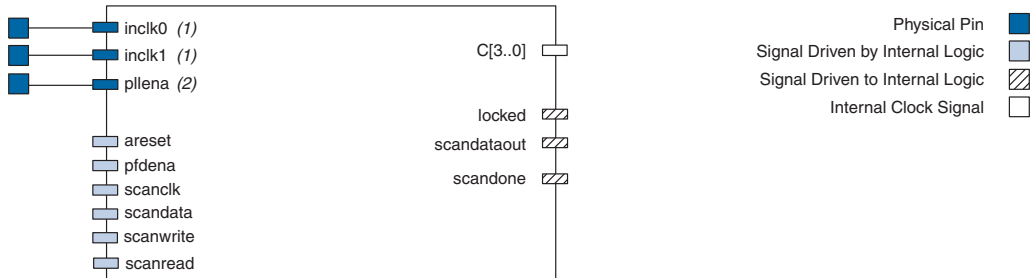


*Notes to Figure 5–7:*
(1)   This input pin is either single-ended or differential.
(2)   This input pin is shared by all enhanced and fast PLLs.

Tables 5–7 and 5–8 show the description of all fast PLL ports.

| Table 5–7. Fast PLL Input Signals | | | |
|---|---|---|---|
| **Name** | **Description** | **Source** | **Destination** |
| inclk0 | Primary clock input to the fast PLL. | Pin or another PLL | counter |
| inclk1 | Secondary clock input to the fast PLL. | Pin or another PLL | counter |
| pllena | Enable pin for enabling or disabling all or a set of PLLs. Active high. | Pin | PLL control signal |
| clkswitch | Switch-over signal used to initiate external clock switch-over control. Active high. | Logic array | Reconfiguration circuit |
| areset | Enables the up/down outputs from the phase-frequency detector. Active high. | Logic array | PLL control signal |
| pfdena | Enables the up/down outputs from the phase-frequency detector. Active high. | Logic array | PFD |
| scanclk | Serial clock signal for the real-time PLL control feature. | Logic array | Reconfiguration circuit |
| scandata | Serial input data stream for the real-time PLL control feature. | Logic array | Reconfiguration circuit |

**Table 5–7. Fast PLL Input Signals**

| Name | Description | Source | Destination |
|------|-------------|--------|-------------|
| scanwrite | Enables writing the data in the scan chain into the PLL Active high. | Logic array | Reconfiguration circuit |
| scanread | Enables scan data to be written into the scan chain Active high. | Logic array | Reconfiguration circuit |

**Table 5–8. Fast PLL Output Signals**

| Name | Description | Source | Destination |
|------|-------------|--------|-------------|
| C[3..0] | PLL outputs driving regional or global clock. | PLL counter | Internal clock |
| locked | Lock or gated lock output from lock detect circuit. Active high. | PLL lock detect | Logic array |
| scandataout | Output of the last shift register in the scan chain. | PLL scan chain | Logic array |
| scandone | Signal indicating when the PLL has completed reconfiguration. 1 to 0 transition indicates the PLL has been reconfigured. | PLL scan chain | Logic array |

## Fast PLL Pins

Table 5–9 shows the I/O standards supported by the fast PLL input pins.

**Table 5–9. I/O Standards Supported for Arria GX Fast PLL Pins  (Part 1 of 2)**

| I/O Standard | inclk |
|--------------|:-----:|
| LVTTL | ✓ |
| LVCMOS | ✓ |
| 2.5 V | ✓ |
| 1.8 V | ✓ |
| 1.5 V | ✓ |
| 3.3-V PCI | — |
| 3.3-V PCI-X | — |
| SSTL-2 Class I | ✓ |
| SSTL-2 Class II | ✓ |
| SSTL-18 Class I | ✓ |
| SSTL-18 Class II | ✓ |
| 1.8-V HSTL Class I | ✓ |

| Table 5–9. I/O Standards Supported for Arria GX Fast PLL Pins  (Part 2 of 2) | |
|---|:---:|
| **I/O Standard** | **inclk** |
| 1.8-V HSTL Class II | ✓ |
| 1.5-V HSTL Class I | ✓ |
| 1.5-V HSTL Class II | ✓ |
| Differential SSTL-2 Class I | — |
| Differential SSTL-2 Class II | — |
| Differential SSTL-18 Class I | — |
| Differential SSTL-18 Class II | — |
| 1.8-V differential HSTL Class I | — |
| 1.8-V differential HSTL Class II | — |
| 1.5-V differential HSTL Class I | — |
| 1.5-V differential HSTL Class II | — |
| LVDS | ✓ |
| HyperTransport technology | ✓ |
| Differential LVPECL | — |

Table 5–10 shows the physical pins and their purpose for the fast PLLs. For `inclk` port connections to pins, see "Clocking" on page 5–59.

| Table 5–10. Fast PLL Pins *Note (1)* | |
|---|---|
| **Pin** | **Description** |
| CLK0p/n | Single-ended or differential pins that can drive the `inclk` port for PLLs 1, 2, 7 or 8. |
| CLK1p/n | Single-ended or differential pins that can drive the `inclk` port for PLLs 1, 2, 7 or 8. |
| CLK2p/n | Single-ended or differential pins that can drive the `inclk` port for PLLs 1, 2, 7 or 8. |
| CLK3p/n | Single-ended or differential pins that can drive the `inclk` port for PLLs 1, 2, 7 or 8. |
| FPLL7CLKp/n | Single-ended or differential pins that can drive the `inclk` port for PLL 7. |
| FPLL8CLKp/n | Single-ended or differential pins that can drive the `inclk` port for PLL 8. |
| pllena | Dedicated input pin that drives the `pllena` port of all or a set of PLLs. If you do not use this pin, connect it to GND. |
| VCCD_PLL | Digital power for PLLs. You must connect this pin to 1.2 V, even if the PLL is not used. |
| VCCA_PLL1 | Analog power for PLL 1. You must connect this pin to 1.2 V, even if the PLL is not used. |
| GNDA_PLL1 | Analog ground for PLL 1. Your can connect this pin to the GND plane on the board. |
| VCCA_PLL2 | Analog power for PLL 2. You must connect this pin to 1.2 V, even if the PLL is not used. |
| GNDA_PLL2 | Analog ground for PLL 2. You can connect this pin to the GND plane on the board. |
| GNDA_PLL7 | Analog ground for PLL 7. You can connect this pin to the GND plane on the board. |

**Table 5–10. Fast PLL Pins** *Note (1)*

| Pin | Description |
|-----|-------------|
| VCCA_PLL8 | Analog power for PLL 8. You must connect this pin to 1.2 V, even if the PLL is not used. |
| GNDA_PLL8 | Analog ground for PLL 8. You can connect this pin to the GND plane on the board. |

*Note to Table 5–10:*

(1)   The negative leg pins (CLKn and FPLL_CLKn) are only required with differential signaling.

# Clock Feedback Modes

Arria GX PLLs support up to five different clock feedback modes. Each mode allows clock multiplication and division, phase shifting, and programmable duty cycle. Each PLL must be driven by one of its own dedicated clock input pins for proper clock compensation. The clock input pin connections for each PLL are listed in Table 5–20 on page 5–68. Table 5–11 shows which modes are supported by which PLL type.

**Table 5–11. Clock Feedback Mode Availability**

| Clock Feedback Mode | Mode Available in Enhanced PLLs | Fast PLLs |
|---------------------|:---:|:---:|
| Source synchronous mode | Yes | Yes |
| No compensation mode | Yes | Yes |
| Normal mode | Yes | Yes |
| Zero delay buffer mode | Yes | No |
| External feedback mode | Yes | No |

## Source-Synchronous Mode

If data and clock arrive at the same time at the input pins, they are guaranteed to keep the same phase relationship at the clock and data ports of any IOE input register. Figure 5–8 shows an example waveform of clock and data in this mode. Source-synchronous mode is recommended for source-synchronous data transfers. Data and clock signals at the IOE experience similar buffer delays as long as you use the same I/O standard.

*Figure 5–8. Phase Relationship Between Clock and Data in Source-Synchronous Mode*
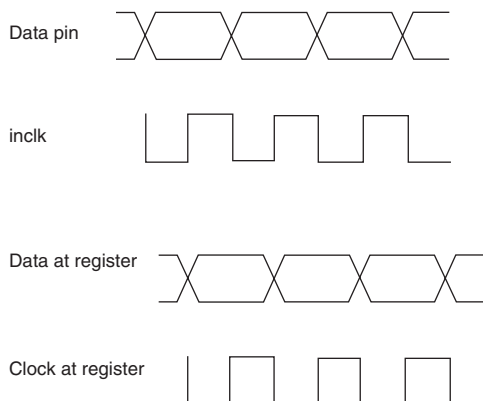


In source-synchronous mode, enhanced PLLs compensate for clock delay to the top and bottom I/O registers and fast PLLs compensate for clock delay to the side I/O registers. While implementing source-synchronous receivers in these I/O banks, use the corresponding PLL type for best matching between clock and data delays (from input pins to register ports).

☞ Set the input pin to the register delay chain within the IOE to zero in the Quartus II software for all data pins clocked by a source-synchronous mode PLL.

## No Compensation Mode

In this mode, the PLL does not compensate for any clock networks. This provides better jitter performance because the clock feedback into the PFD does not pass through as much circuitry. Both the PLL internal and external clock outputs are phase shifted with respect to the PLL clock input. Figure 5–9 shows an example waveform of the PLL clocks' phase relationship in this mode.

*Figure 5–9. Phase Relationship between PLL Clocks in No Compensation Mode*
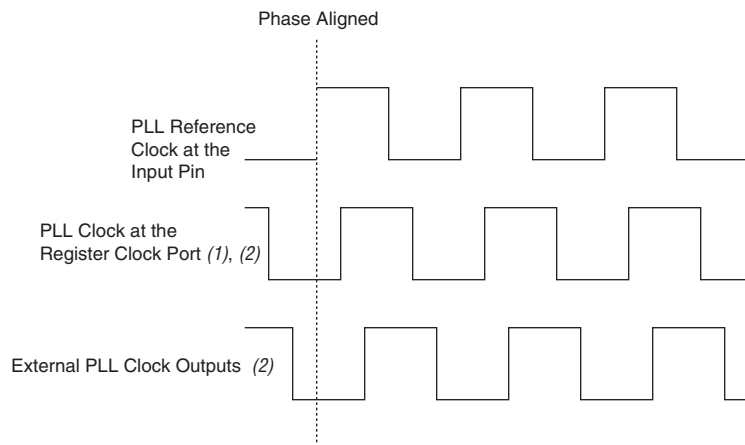


*Notes to Figure 5–9:*
(1)    Internal clocks fed by the PLL are phase-aligned to each other.
(2)    The PLL clock outputs can lead or lag the PLL input clocks.

## Normal Mode

An internal clock in normal mode is phase-aligned to the input clock pin. The external clock output pin will have a phase delay relative to the clock input pin if connected in this mode. In normal mode, the delay introduced by the Global Clock or Regional Clock network is fully compensated. Figure 5–10 shows an example waveform of the PLL clocks' phase relationship in normal mode.

*Figure 5–10. Phase Relationship Between PLL Clocks in Normal Mode*
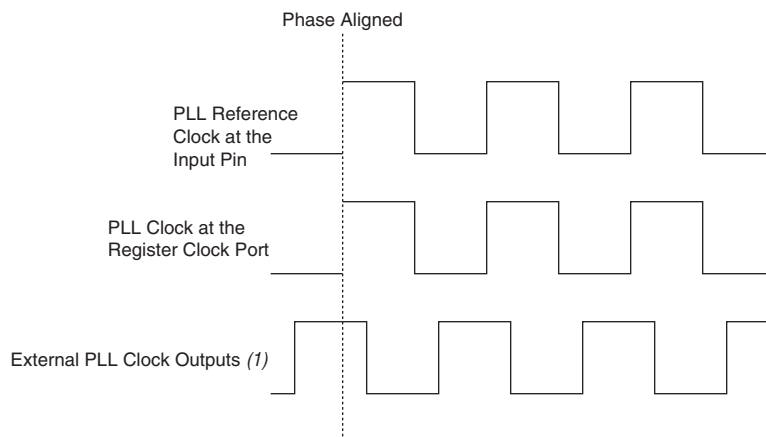


*Note to Figure 5–10:*
(1)    The external clock output can lead or lag the PLL internal clock signals.

## Zero Delay Buffer Mode

In zero delay buffer mode, the external clock output pin is phase-aligned with the clock input pin for zero delay through the device. Figure 5–11 shows an example waveform of the PLL clocks' phase relationship in zero delay buffer mode. When using this mode, Altera requires that you use the same I/O standard on the input clock and output clocks. When using single-ended I/O standards, the `inclk` port of the PLL must be fed by the dedicated `CLKp` input pin.

*Figure 5–11. Phase Relationship Between PLL Clocks in Zero Delay Buffer Mode*
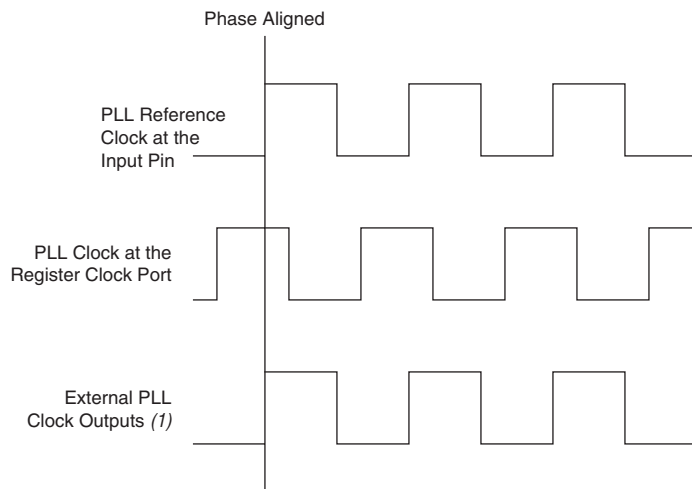


*Note to Figure 5–11:*
(1)    The internal PLL clock output can lead or lag the external PLL clock outputs.

## External Feedback Mode

In external feedback mode, the external feedback input pin, `fbin`, is phase-aligned with the clock input pin, (see Figure 5–12). Aligning these clocks allows you to remove clock delay and skew between devices. External feedback mode is possible on all enhanced PLLs. PLLs 5, 6, 11, and 12 support feedback for one of the dedicated external outputs, either one single-ended or one differential pair. In this mode, one *C* counter feeds back to the PLL `fbin` input, becoming part of the feedback loop. In external feedback mode, you will use one of the dedicated external clock outputs (two if a differential I/O standard is used) as the PLL `fbin` input pin. When using external feedback mode, Altera requires that you use the same I/O standard on the input clock, feedback input, and output clocks. When using single-ended I/O standards, the `inclk` port of the PLL must be fed by the dedicated `CLKp` input pin.

*Figure 5–12. Phase Relationship Between PLL Clocks in External Feedback Mode*
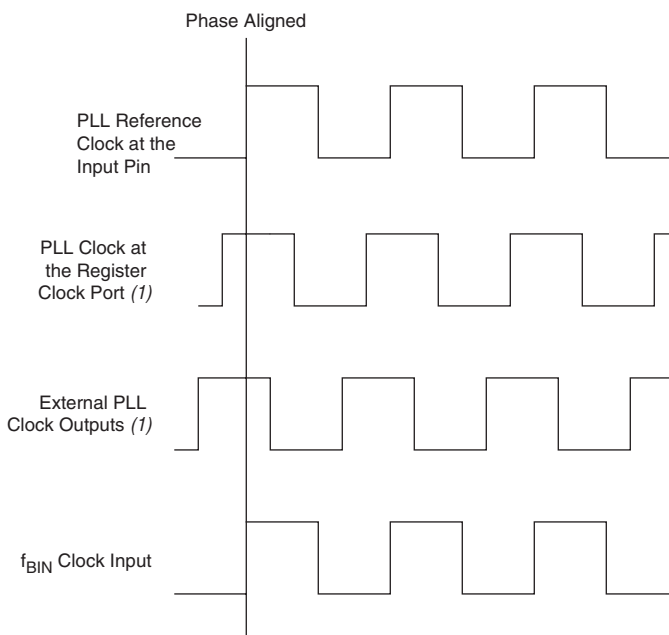


*Note to Figure 5–12:*
(1)    The PLL clock outputs can lead or lag the `fbin` clock input.

## Hardware Features

Arria GX PLLs support a number of features for general purpose clock management. This section discusses clock multiplication and division implementation, phase-shifting implementations, and programmable duty cycles. Table 5–12 shows which feature is available in which type of Arria GX PLL.

*Table 5–12. Arria GX PLL Hardware Features   (Part 1 of 2)*

| Hardware Features | Availability Enhanced PLL | Fast PLL |
|---|---|---|
| Clock multiplication and division | $m$ ($n$ × post-scale counter) | $m$ ($n$ × post-scale counter) |
| $m$ counter value | Ranges from 1 through 512 | Ranges from 1 through 32 |
| counter value | Ranges from 1 through 512 | Ranges from 1 through 4 |
| Post-scale counter values | Ranges from 1 through 512 *(1)* | Ranges from 1 through 32 *(2)* |
| Phase shift | Down to 125-ps increments *(3)* | Down to 125-ps increments *(3)* |

*Table 5–12. Arria GX PLL Hardware Features  (Part 2 of 2)*

| Hardware Features | Availability Enhanced PLL | Fast PLL |
|---|---|---|
| Programmable duty cycle | Yes | Yes |

*Notes to Table 5–12:*

(1) Post-scale counters range from 1 through 512 if the output clock uses a 50% duty cycle. For any output clocks using a non-50% duty cycle, the post-scale counters range from 1 through 256.

(2) Post-scale counters range from 1 through 32 if the output clock uses a 50% duty cycle. For any output clocks using a non-50% duty cycle, the post-scale counters range from 1 through 16.

(3) The smallest phase shift is determined by the VCO period divided by eight. For degree increments, the Arria GX device can shift all output frequencies in increments of at least 45. Smaller degree increments are possible depending on the frequency and divide parameters.

## Clock Multiplication and Division

Each Arria GX PLL provides clock synthesis for PLL output ports using $m/(n \times \text{post-scale counter})$ scaling factors. The input clock is divided by a pre-scale factor, $n$, and is then multiplied by the $m$ feedback factor. The control loop drives VCO to match $f_{in}$ ($m/n$). Each output port has a unique post-scale counter that divides down the high-frequency VCO. For multiple PLL outputs with different frequencies, the VCO is set to the least common multiple of the output frequencies that meets its frequency specifications. For example, if output frequencies required from one PLL are 33 and 66 MHz, the Quartus II software sets VCO to 660 MHz (the least common multiple of 33 and 66 MHz within VCO range). Then, the post-scale counters scale down the VCO frequency for each output port.

There is one pre-scale counter, $n$, and one multiply counter, $m$, per PLL, with a range of 1 to 512 for both $m$ and $n$ in enhanced PLLs. For fast PLLs, $m$ ranges from 1 to 32 while $n$ ranges from 1 to 4. There are six generic post-scale counters in enhanced PLLs that can feed regional clocks, global clocks, or external clock outputs, all ranging from 1 to 512 with a 50% duty cycle setting for each PLL. The post-scale counters range from 1 to 256 with any non-50% duty cycle setting. In fast PLLs, there are four post-scale counters (C0, C1, C2, and C3) for the regional and global clock output ports. All post-scale counters range from 1 to 32 with a 50% duty cycle setting. For non-50% duty cycle clock outputs, the post-scale counters range from 1 to 16. If the design uses a high-speed I/O interface, you can connect the dedicated dffioclk clock output port to allow the high-speed VCO frequency to drive the serializer/deserializer (SERDES).

The Quartus II software automatically chooses the appropriate scaling factors according to the input frequency, multiplication, and division values entered into the ALTPLL megafunction.

## Phase-Shift Implementation

Phase shift is used to implement a robust solution for clock delays in Arria GX devices. Phase shift is implemented by using a combination of the VCO phase output and counter starting time. VCO phase output and counter starting time is the most accurate method of inserting delays Because it is purely based on counter settings, which are independent of process, voltage, and temperature.

You can phase shift the output clocks from Arria GX enhanced PLLs in either:

■ Fine resolution using VCO phase taps
■ Coarse resolution using counter starting time

The VCO phase tap and counter starting time is implemented by allowing any of the output counters (C[5..0] or $m$) to use any of the eight phases of the VCO as the reference clock. This allows you to adjust the delay time with a fine resolution. The minimum delay time that you can insert using this method is defined by:

$$\Phi_{fine} = \frac{1}{8f_{VCO}} = \frac{N}{8Mf_{REF}}$$

where $f_{REF}$ is input reference clock frequency.

■ For example, if $f_{REF}$ is 100 MHz, $n$ is 1, and $m$ is 8, then $f_{VCO}$ is 800 MHz and $\Phi_{fine}$ equals 156.25 ps. This phase shift is defined by the PLL operating frequency, which is governed by the reference clock frequency and the counter settings.

You can also delay the start of the counters for a predetermined number of counter clocks. You can express phase shift as:

$$\Phi_{coarse} = \frac{C-1}{f_{VCO}} = \frac{(C-1)N}{Mf_{REF}}$$

where $C$ is the count value set for the counter delay time, (this is the initial setting in the PLL usage section of the compilation report in the Quartus II software). If the initial value is 1, $C-1 = 0^{\circ}$ phase shift.

$\mathbb{G}$   Figure 5–13 shows an example of phase shift insertion using the fine resolution and VCO phase taps method. The eight phases from the VCO are shown and labeled for reference. For this example, CLK0 is based off the 0phase from the VCO and has the $C$ value for the counter set to one. The CLK1 signal is divided by four, two VCO clocks for high time and two VCO clocks for low time. CLK1 is based off the 135 phase tap from the VCO and also has the $C$ value for the counter set to one. The CLK1 signal is also divided by four. In this case, the two clocks are offset by $3\Phi_{fine}$. CLK2 is based off the $0^\circ$ phase from the VCO but has the $C$ value for the counter set to three. This creates a delay of $2\Phi_{coarse}$, (two complete VCO periods).

*Figure 5–13. Delay Insertion Using VCO Phase Output and Counter Delay Time*
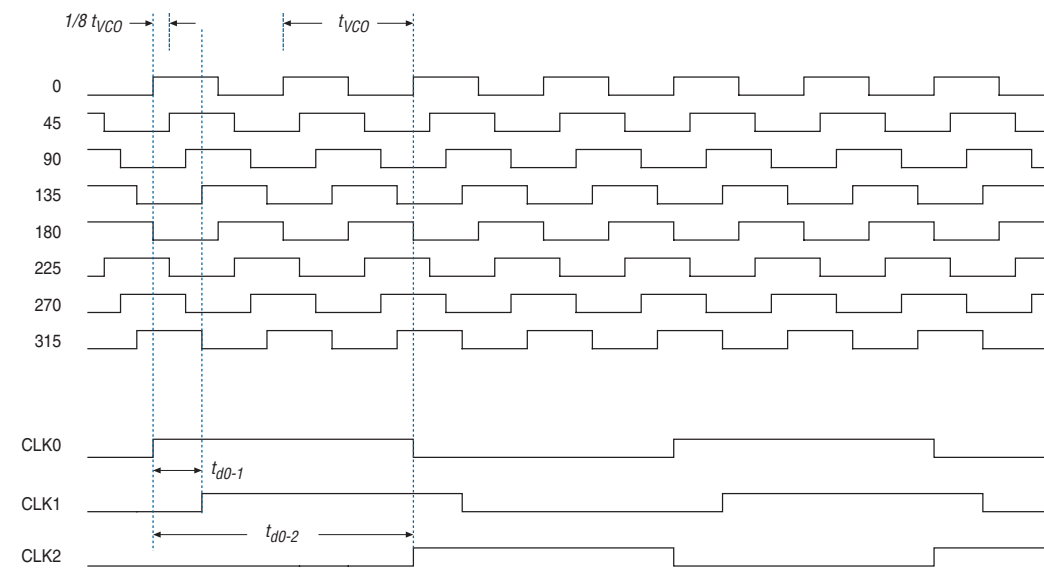


You can use the coarse and fine phase shifts as described above to implement clock delays in Arria GX devices. The phase-shift parameters are set in the Quartus II software.

## Programmable Duty Cycle

The programmable duty cycle allows enhanced and fast PLLs to generate clock outputs with a variable duty cycle. This feature is supported on each enhanced and fast PLL post-scale counter C[]. The duty cycle setting is achieved by a low- and high-time count setting for post-scale

counters. The Quartus II software uses the frequency input and required multiply or divide rate to determine the duty cycle choices. The post-scale counter value determines the precision of the duty cycle. The precision is defined by 50% divided by the post-scale counter value. The closest value to 100% is not achievable for a given counter value. For example, if the C0 counter is ten, steps of 5% are possible for duty cycle choices between 5 to 90%.

If the device uses external feedback, you must set the duty cycle for the counter driving the fbin pin to 50%. Combining the programmable duty cycle with programmable phase shift allows the generation of precise non-overlapping clocks.

## Advanced Clear and Enable Control

There are several control signals for clearing and enabling PLLs and their outputs. You can use these signals to control PLL resynchronization and gate PLL output clocks for low-power applications.

### Enhanced Lock Detect Circuit

The lock output indicates that the PLL has locked onto the reference clock. Without any additional circuitry, the lock signal may toggle as the PLL begins tracking the reference clock. You may need to gate the lock signal for use as a system control. Either a gated lock signal or an ungated lock signal from the locked port can drive the logic array or an output pin. Arria GX enhanced and fast PLLs include a programmable counter that holds the lock signal low for a user-selected number of input clock transitions. This allows the PLL to lock before enabling the lock signal. You can use the Quartus II software to set the 20-bit counter value.

Figure 5–14 shows the timing waveform for lock and gated lock signals.

*Figure 5–14. Timing Waveform for Lock and Gated Lock Signals*
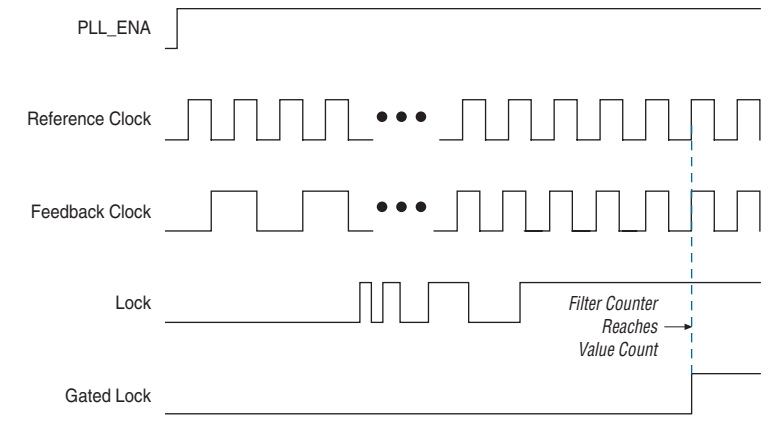


The device resets and enables both the counter and the PLL simultaneously when the pllena signal is asserted or the areset signal is deasserted. Enhanced PLLs and fast PLLs support this feature. To ensure correct circuit operation, and to ensure that the output clocks have the correct phase relationship with respect to the input clock, Altera recommends that the input clock be running before the Arria GX device is finished configuring.

### pllena

The pllena pin is a dedicated pin that enables or disables all PLLs on the Arria GX device. When the pllena pin is low, the clock output ports are driven low and all the PLLs go out of lock. When the pllena pin goes high again, the PLLs relock and resynchronize to the input clocks. You can choose which PLLs are controlled by the pllena signal by connecting the pllena input port of the ALTPLL megafunction to the common pllena input pin.

Also, whenever the PLL loses lock for any reason (for example, excessive inclk jitter, clock switchover, PLL reconfiguration, power supply noise, etc.), the PLL must be reset with the areset signal to guarantee correct phase relationship between the PLL output clocks. If the phase relationship between the input clock versus output clock, and between different output clocks from the PLL is not important in your design, the PLL need not be reset.

The level of the VCCSEL pin selects the pllena input buffer power. Therefore, if VCCSEL is high, the pllena pin's 1.8/1.5-V input buffer is powered by $V_{CCIO}$ of the bank that pllena resides in. If VCCSEL is low (GND), the pllena pin's 3.3/2.5-V input buffer is powered by $V_{CCPD}$.

For more information about the VCCSEL pin, refer to the *Configuring Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*.

### pfdena

The pfdena signals control the phase frequency detector (PFD) output with a programmable gate. If you disable the PFD, the VCO operates at its last set value of control voltage and frequency with some long-term drift to a lower frequency. The system continues running when the PLL goes out of lock or the input clock is disabled. By maintaining the last locked frequency, the system has time to store its current settings before shutting down. You can either use your own control signal or clkloss or gated locked status signals, to trigger pfdena.

### areset

The areset signal is the reset or resynchronization input for each PLL. The device input pins or internal logic can drive these input signals. When driven high, the PLL counters reset, clearing the PLL output and placing the PLL out of lock. The VCO is set back to its nominal setting (~700 MHz). When driven low again, the PLL will resynchronize to its input as it relocks. If the target VCO frequency is below this nominal frequency, the output frequency starts at a higher value than desired as the PLL locks.

The areset signal should be asserted every time the PLL loses lock to guarantee the correct phase relationship between the PLL input clock and output clocks. You should include the areset signal in designs if any of the following conditions are true:

■ PLL reconfiguration or clock switchover is enabled in the design.
■ Phase relationships between the PLL input clock and output clocks need to be maintained after a loss of lock condition.
■ If the input clock to the PLL is not toggling or is unstable upon power up, assert the areset signal after the input clock is toggling, making sure to stay within the input jitter specification.

☞ Altera recommends that you use the areset and locked signals in your designs to control and observe the status of your PLL.

*clkena*

If the system cannot tolerate the higher output frequencies when using pfdena higher value, the clkena signals can disable the output clocks until the PLL locks. The clkena signals control the regional, global, and external clock outputs. The clkena signals are registered on the falling edge of the counter output clock to enable or disable the clock without glitches. See Figure 5–53 on page 5–81 for more information about the clkena signals.

## Advanced Features

Arria GX PLLs offer a variety of advanced features, such as counter cascading, clock switchover, PLL reconfiguration, reconfigurable bandwidth, and spread-spectrum clocking. Table 5–13 shows which advanced features are available in which type of Arria GX PLL.

*Table 5–13. Arria GX PLL Advanced Features*

| Advanced Feature | Availability | |
|---|:---:|:---:|
| | **Enhanced PLLs** | **Fast PLLs**(1) |
| Counter cascading | ✓ | — |
| Clock switchover | ✓ | ✓ |
| PLL reconfiguration | ✓ | ✓ |
| Reconfigurable bandwidth | ✓ | ✓ |
| Spread-spectrum clocking | ✓ | — |

*Note to Table 5–13:*
(1)   Arria GX fast PLLs only support manual clock switchover, not automatic clock switchover.

### Counter Cascading

The Arria GX enhanced PLL supports counter cascading to create post-scale counters larger than 512. This is implemented by feeding the output of one counter into the input of the next counter in a cascade chain, as shown in Figure 5–15.

*Figure 5–15. Counter Cascading*
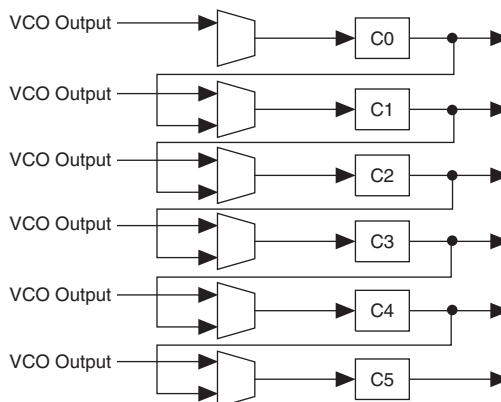


When cascading counters to implement a larger division of the high-frequency VCO clock, the cascaded counters behave as one counter with the product of the individual counter settings. For example, if `C0` = 4 and `C1` = 2, the cascaded value is `C0` × `C1` = 8.

☞ Arria GX fast PLLs do not support counter cascading.

Counter cascading is set in the configuration file, meaning they can not be cascaded using PLL reconfiguration.

## Clock Switchover

The clock switchover feature allows the PLL to switch between two reference input clocks. Use the clock switchover feature for clock redundancy or for a dual clock domain application such as in a system that turns on the redundant clock if the primary clock stops running. The design can perform clock switchover automatically, when the clock is no longer toggling, or based on a user control signal, `clkswitch`.

☞ Enhanced PLLs support both automatic and manual switchover, while fast PLLs only support manual switchover.

### Automatic Clock Switchover

Arria GX device PLLs support a fully configurable clock switchover capability. Figure 5–16 shows the block diagram of the switch-over circuit built into the enhanced PLL. When the primary clock signal is not present, the clock sense block automatically switches from the primary to the

secondary clock for PLL reference. The design sends out the `clkbad0`, `clkbad1`, and `clkloss` signals from the PLL to implement a custom switchover circuit.

*Figure 5–16. Automatic Clock Switchover Circuit Block Diagram*
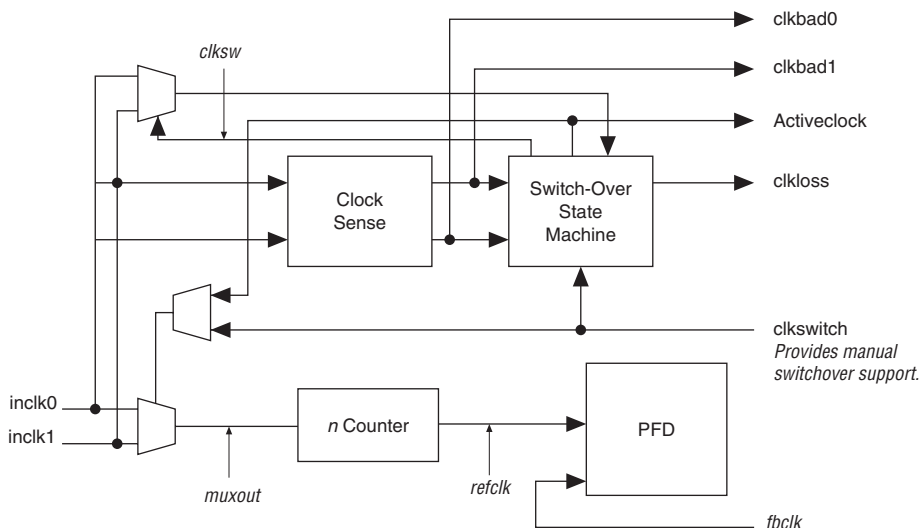


There are two possible ways to use the clock switchover feature.

■ Use the switchover circuitry for switching from a primary to secondary input of the same frequency. For example, in applications that require a redundant clock with the same frequency as the primary clock, the switchover state machine generates a signal that controls the multiplexer select input shown on the bottom of Figure 5–16. In this case, the secondary clock becomes the reference clock for the PLL. This automatic switchover feature only works for switching from the primary to secondary clock.

■ Use the CLKSWITCH input for user- or system-controlled switch conditions. This is possible for same-frequency switchover or to switch between inputs of different frequencies. For example, if `inclk0` is 66 MHz and `inclk1` is 100 MHz, you must control the switchover because the automatic clock-sense circuitry cannot monitor primary and secondary clock frequencies with a frequency difference of more than 20%. This feature is useful when clock sources can originate from multiple cards on the backplane, requiring a system-controlled switchover between frequencies of operation. You should choose the secondary clock frequency so the

VCO operates within the recommended range of 500 to 1,000 MHz. You should also set the *m* and *n* counters accordingly to keep the VCO operating frequency in the recommended range.

Figure 5–17 shows an example waveform of the switchover feature when using automatic `clkloss` detection. Here, the `inclk0` signal gets stuck low. After the `inclk0` signal is stuck at low for approximately two clock cycles, the clock sense circuitry drives the `clkbad0` signal high. Also, because the reference clock signal is not toggling, the `clkloss` signal goes low, indicating a switch condition. Then, the switchover state machine controls the multiplexer through the `clksw` signal to switch to the secondary clock.

*Figure 5–17. Automatic Switchover Upon Clock Loss Detection*
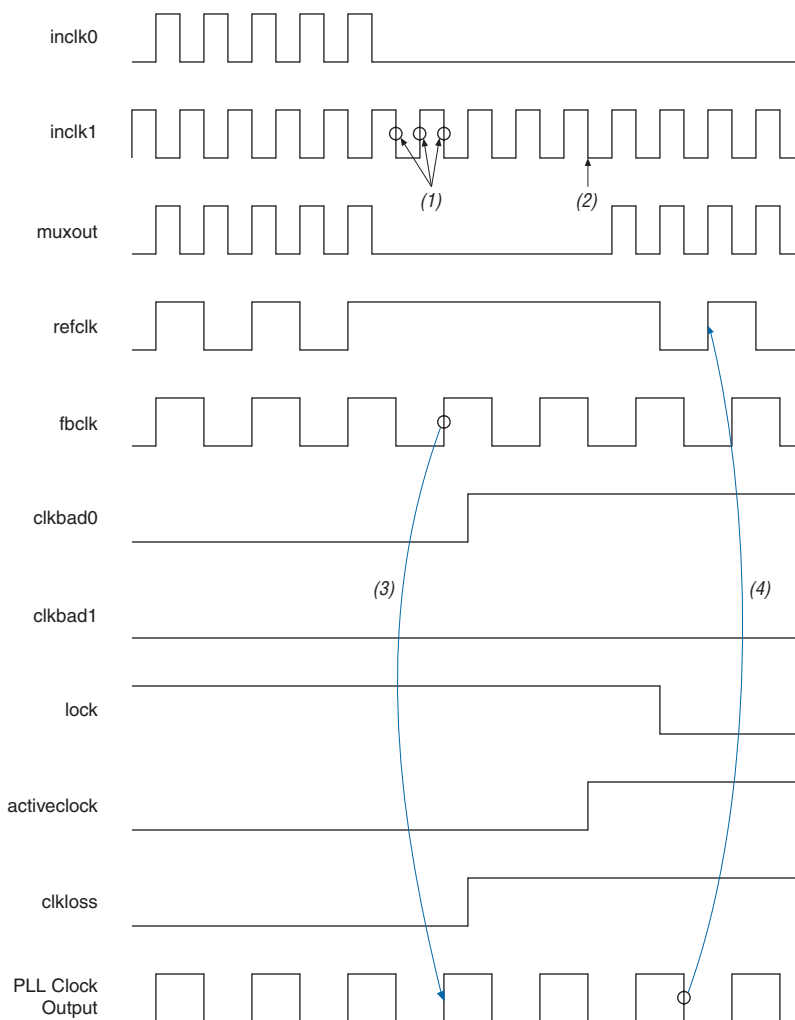


*Notes to Figure 5–17:*
(1) The number of clock edges before allowing switchover is determined by the counter setting.
(2) Switchover is enabled on the falling edge of `inclk1`.
(3) The rising edge of `fbclk` causes the VCO frequency to decrease.
(4) The rising edge of `refclk` starts the PLL lock process again, and the VCO frequency increases.

The switch-over state machine has two counters that count the edges of the primary and secondary clocks; counter0 counts the number of inclk0 edges and counter1 counts the number of inclk1 edges. The counters get reset to zero when the count values reach 1, 1; 1, 2; 2, 1; or 2, 2 for inclk0 and inclk1, respectively. For example, if counter0 counts two edges, its count is set to two and if counter1 counts two edges before counter0 sees another edge, they are both reset to 0. If for some reason one of the counters counts to three, it means the other clock missed an edge. The clkbad0 or clkbad1 signal goes high and the switchover circuitry signals a switch condition. See Figure 5–18.

*Figure 5–18. Clock-Edge Detection for Switchover*
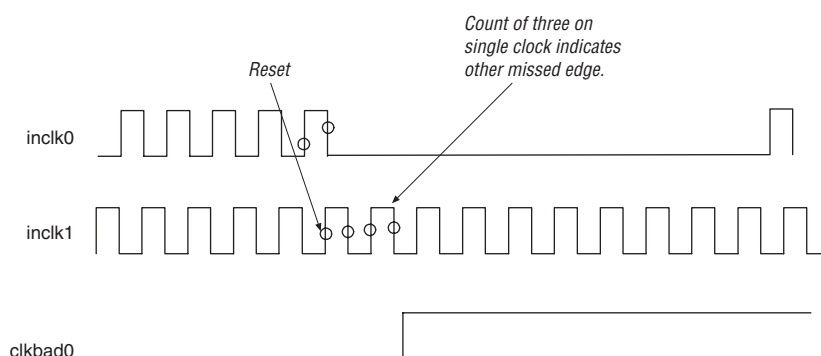


**Manual Override**
When using automatic switchover, you can switch input clocks by using the manual override feature with clkswitch input.

☞     The manual override feature available in automatic clock switchover is different from manual clock switchover.

Figure 5–19 shows an example of a waveform illustrating the switchover feature when controlled by clkswitch. In this case, both clock sources are functional and inclk0 is selected as the primary clock. clkswitch goes high, which starts the switchover sequence. On the falling edge of inclk0, the counter's reference clock, muxout, is gated off to prevent any clock glitching. On the falling edge of inclk1, the reference clock multiplexer switches from inclk0 to inclk1 as the PLL reference. This is also when the clkswitch signal changes to indicate which clock is selected as primary and which is secondary.

The clkloss signal mirrors the clkswitch signal and activeclock mirrors clkswitch in manual override mode. Because both clocks are still functional during the manual switch, neither clkbad signal goes

high. Because the switchover circuit is edge-sensitive, the falling edge of the clkswitch signal does not cause the circuit to switch back from inclk1 to inclk0. When the clkswitch signal goes high again, the process repeats. clkswitch and automatic switch only work if the clock being switched to is available. If the clock is not available, the state machine waits until the clock is available.

*Figure 5–19. Clock Switchover Using the CLKSWITCH Control*
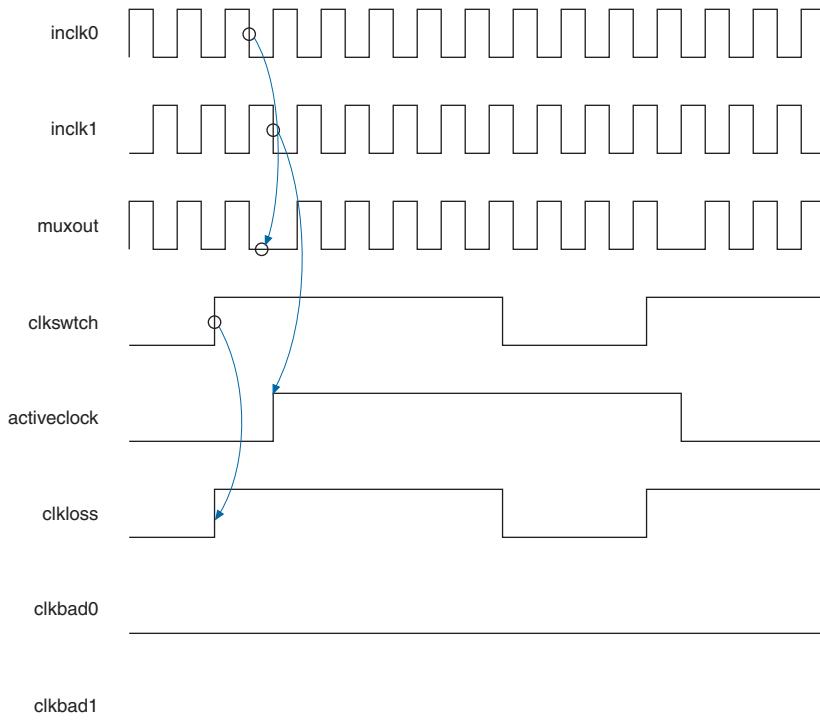


Figure 5–20 shows a simulation of using switchover for two different reference frequencies. In this example simulation, the reference clock is either 100 or 66 MHz. The PLL begins with $f_{in}$=100 MHz and is allowed to lock. At 20 ms, the clock is switched to the secondary clock, which is at 66 MHz.

**Figure 5–20. Switchover Simulation** *Note (1)*
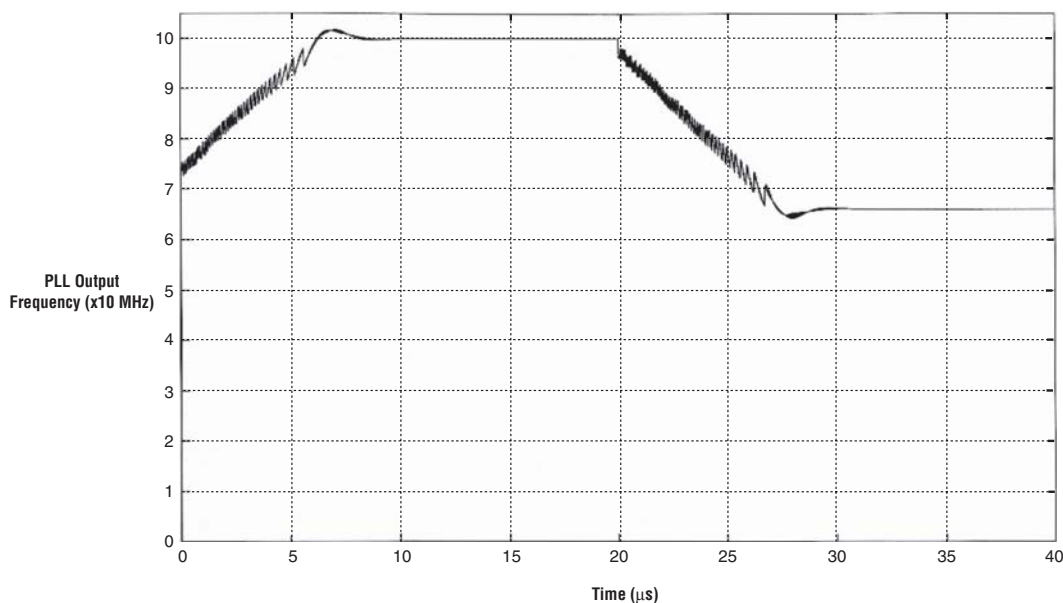


*Note to Figure 5–20:*

(1)  This simulation was performed under the following conditions: the counter is set to 2, the *m* counter is set to 16, and the output counter is set to 8. Therefore, the VCO operates at 800 MHz for the 100-MHz input references and at 528 MHz for the 66-MHz reference input.

## Lock Signal-Based Switchover

The lock circuitry can initiate automatic switchover. This is useful for cases where the input clock is still clocking, but its characteristics have changed so that the PLL is not locked to it. The switchover enable is based on both the gated and ungated lock signals. If the ungated lock is low, switchover is not enabled until the gated lock has reached its terminal count. You must activate switchover enable if the gated lock is high, but the ungated lock goes low. The switchover timing for this mode is similar to the waveform shown in Figure 5–19 for `clkswitch` control, except switchover enable replaces `clkswitch`. Figure 5–21 shows the switchover enable circuit when controlled by lock and gated lock.

**Figure 5–21. Switchover Enable Circuit**
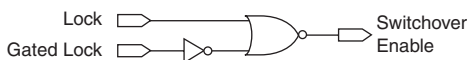
### Manual Clock Switchover

Arria GX enhanced and fast PLLs support manual switchover, where the clkswitch signal controls whether inclk0 or inclk1 is the input clock to the PLL. If clkswitch is low, inclk0 is selected; if clkswitch is high, inclk1 is selected. Figure 5–22 shows the block diagram of the manual switchover circuit in fast PLLs. The block diagram of the manual switchover circuit in enhanced PLLs is shown in Figure 5–22.

*Figure 5–22. Manual Clock Switchover Circuitry in Fast PLLs*
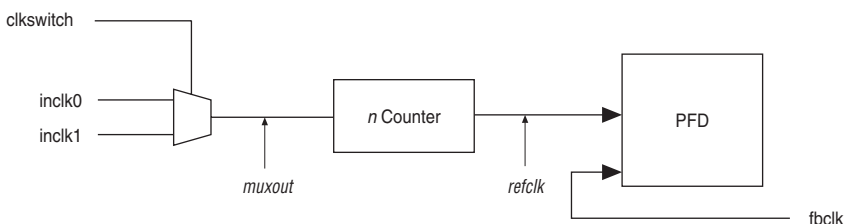


Figure 5–23 shows an example of a waveform illustrating the switchover feature when controlled by clkswitch. In this case, both clock sources are functional and inclk0 is selected as the primary clock. clkswitch goes high, which starts the switchover sequence. On the falling edge of inclk0, the counter's reference clock, muxout, is gated off to prevent any clock glitching. On the rising edge of inclk1, the reference clock multiplex switches from inclk0 to inclk1 as the PLL reference. When the clkswitch signal goes low, the process repeats, causing the circuit to switch back from inclk1 to inclk0.

*Figure 5–23. Manual Switchover*
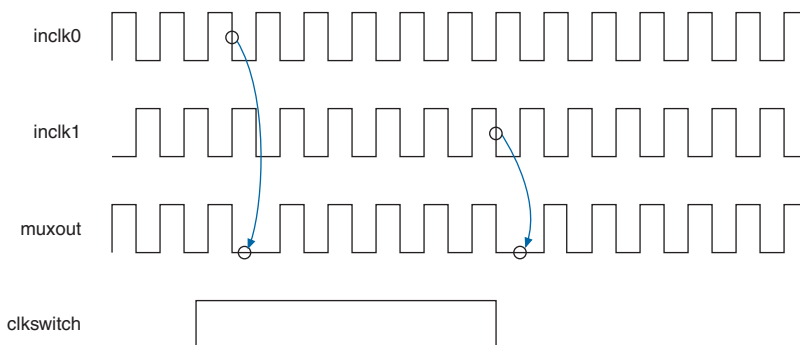
*Software Support*

Table 5–14 summarizes the signals used for clock switchover.

| Table 5–14. ALTPLL Megafunction Clock Switchover Signals | | | |
|---|---|---|---|
| **Port** | **Description** | **Source** | **Destination** |
| inclk0 | Reference clk0 to the PLL. | I/O pin | Clock switchover circuit |
| inclk1 | Reference clk1 to the PLL. | I/O pin | Clock switchover circuit |
| clkbad0 *(1)* | Signal indicating that inclk0 is no longer toggling. | Clock switchover circuit | Logic array |
| clkbad1 *(1)* | Signal indicating that inclk1 is no longer toggling. | Clock switchover circuit | Logic array |
| clkswitch | Switchover signal used to initiate clock switchover asynchronously. When used in manual switchover, clkswitch is used as a select signal between inclk0 and inclk1 clkswitch = 0 inclk0 is selected and vice versa. | Logic array or I/O pin | Clock switchover circuit |
| clkloss *(1)* | Signal indicating that the switchover circuit detected a switch condition. | Clock switchover circuit | Logic array |
| locked | Signal indicating that the PLL has lost lock. | PLL | Clock switchover circuit |
| activeclock *(1)* | Signal to indicate which clock (0 = inclk0, 1= inclk1) is driving the PLL. | PLL | Logic array |

*Note for Table 5–14:*
(1)    These ports are only available for enhanced PLLs, in automatic mode, and when using automatic switchover.

All the switchover ports shown in Table 5–14 are supported in the ALTPLL megafunction in the Quartus II software. The ALTPLL megafunction supports two methods for clock switchover:

■    When selecting an enhanced PLL, you can enable both automatic and manual switchover, making all the clock switchover ports available.
■    When selecting a fast PLL, you can only enable the manual clock switchover option to select between inclk0 or inclk1. The clkloss, activeclock, clkbad0, and clkbad1 signals are not available when you select manual switchover.

If the primary and secondary clock frequencies are different, the Quartus II software selects the proper parameters to keep the VCO within the recommended frequency range.

For more information about PLL support in the Quartus II software, see the *altpll Megafunction User Guide*.

### Guidelines

Use the following guidelines to design with clock switchover in PLLs:

■ When using automatic switchover, the `clkswitch` signal has a minimum pulse width based on the two reference clock periods. The `clkswitch` pulse width must be greater than or equal to the period of the current reference clock ($t_{from\_clk}$) multiplied by two plus the rounded-up version of the ratio of the two reference clock periods. For example, if $t_{to\_clk}$ is equal to $t_{from\_clk}$, the `clkswitch` pulse width should be at least three times the period of the clock pulse.

$$t_{clkswitchmin} \geq t_{from\_clk} \times [2 + int_{round\_up} (t_{to\_clk} \div t_{from\_clk})]$$

■ Applications that require a clock switchover feature and a small frequency drift should use a low-bandwidth PLL. The low-bandwidth PLL reacts slower than a high-bandwidth PLL to reference input clock changes. When switchover happens, a low-bandwidth PLL propagates the stopping of the clock to the output slower than a high-bandwidth PLL. A low-bandwidth PLL filters out jitter on the reference clock. However, be aware that the low-bandwidth PLL also increases lock time.

■ Arria GX device PLLs can use both automatic clock switchover and `clkswitch` input simultaneously. Therefore, the switchover circuitry can automatically switch from the primary to the secondary clock. Once the primary clock stabilizes again, the `clkswitch` signal can switch back to the primary clock. During switchover, `PLL_VCO` continues to run and slows down, generating frequency drift on the PLL outputs. The `clkswitch` signal controls switchover with its rising edge only.

■ If the clock switchover event is glitch-free, after the switch occurs, there is still a finite resynchronization period to lock onto a new clock as the VCO ramps up. The exact amount of time it takes for the PLL to relock depends on the PLL configuration. Use the PLL programmable bandwidth feature to adjust the relock time.

■ If the phase relationship between the input clock to the PLL and output clock from the PLL is important in your design, assert `areset` for 10 ns after performing a clock switchover. Wait for the locked signal (or gated lock) to go high before re-enabling the output clocks from the PLL.

■ Figure 5–24 shows how the VCO frequency gradually decreases when the primary clock is lost and then increases as the VCO locks on to the secondary clock. After the VCO locks on to the secondary clock, some overshoot can occur (an over-frequency condition) in the VCO frequency.

*Figure 5–24. VCO Switchover Operating Frequency*
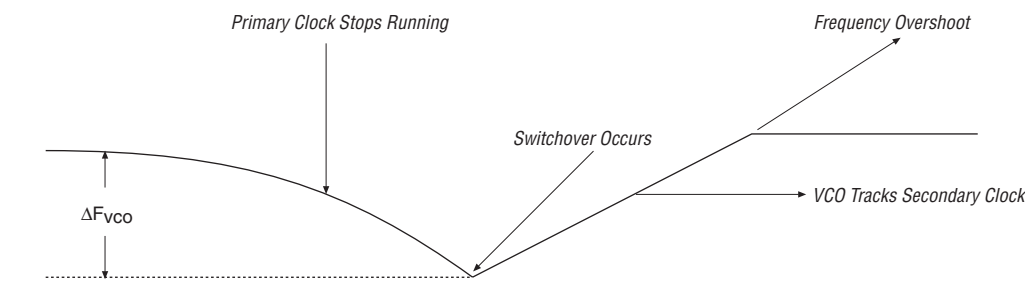


■ Disable the system during switchover if it is not tolerant to frequency variations during the PLL resynchronization period. There are two ways to disable the system. First, the system may require some time to stop before switchover occurs. The switchover circuitry includes an optional five-bit counter to delay when the reference clock is switched. You have the option to control the time-out setting on this counter (up to 32 cycles of latency) before the clock source switches. You can use these cycles for disaster recovery. The clock output frequency varies slightly during those 32 cycles because the VCO can still drift without an input clock. Programmable bandwidth can control the PLL response to limit drift during this 32 cycle period.

■ A second option available is the ability to use the PFD enable signal (pfdena) along with user-defined control logic. In this case you can use the clk0_bad and clk1_bad status signals to turn off PFD so the VCO maintains its last frequency. You can also use the state machine to switch over to the secondary clock. Upon re-enabling the PFD, the output clock enable signals (clkena) can disable clock outputs during the switchover and resynchronization period. Once the lock indication is stable, the system can re-enable the output clock(s).

# Reconfigurable Bandwidth

Arria GX enhanced and fast PLLs provide advanced control of the PLL bandwidth using the PLL loop's programmable characteristics, including loop filter and charge pump.

### Background

PLL bandwidth is the measure of the PLL's ability to track the input clock and jitter. The closed-loop gain 3-dB frequency in the PLL determines the PLL bandwidth. The bandwidth is approximately the unity gain point for open loop PLL response. As Figure 5–25 shows, these points correspond to approximately the same frequency.

*Figure 5–25. Open- and Closed-Loop Response Bode Plots*

**Open-Loop Reponse Bode Plot**

*Increasing the PLL's bandwidth in effect pushes the open loop response out.*

0 dB

**Gain**

**Frequency**

**Closed-Loop Reponse Bode Plot**

**Gain**

**Frequency**
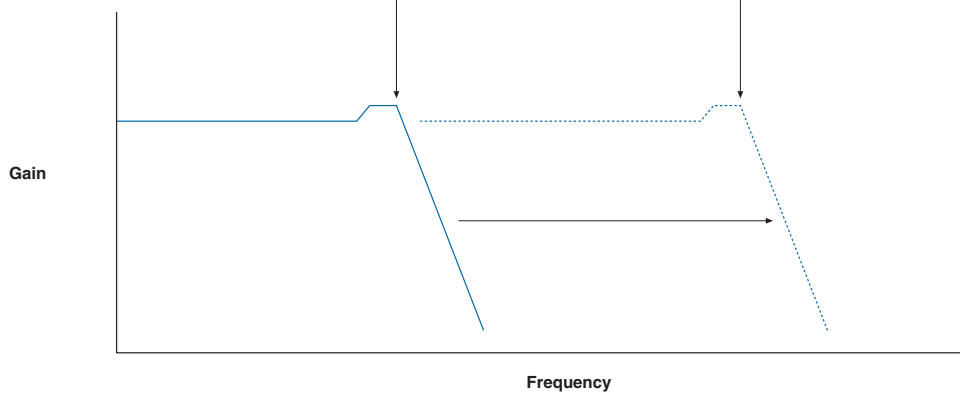
A high-bandwidth PLL provides a fast lock time and tracks jitter on the reference clock source, passing it through to the PLL output. A low-bandwidth PLL filters out reference clock, but increases lock time. Arria GX enhanced and fast PLLs allow you to control the bandwidth over a finite range to customize the PLL characteristics for a particular application. The programmable bandwidth feature in Arria GX PLLs benefits applications requiring clock switchover (for example, TDMA frequency hopping wireless and redundant clocking).

The bandwidth and stability of such a system is determined by the charge pump current, loop filter resistor value, high-frequency capacitor value (in the loop filter), and *m*-counter value. You can use the Quartus II software to control these factors and to set the bandwidth to the desired value within a given range.

You can set the bandwidth to the appropriate value to balance the need for jitter filtering and lock time. Figures 5–26 and 5–27 show the output of a low- and high-bandwidth PLL, respectively, as it locks onto the input clock.

*Figure 5–26. Low-Bandwidth PLL Lock Time*
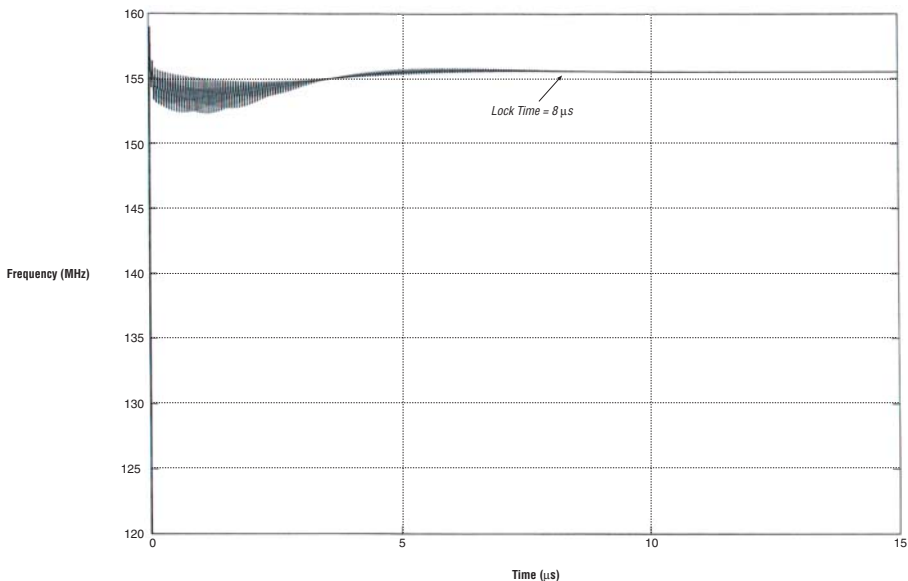
*Figure 5–27. High-Bandwidth PLL Lock Time*
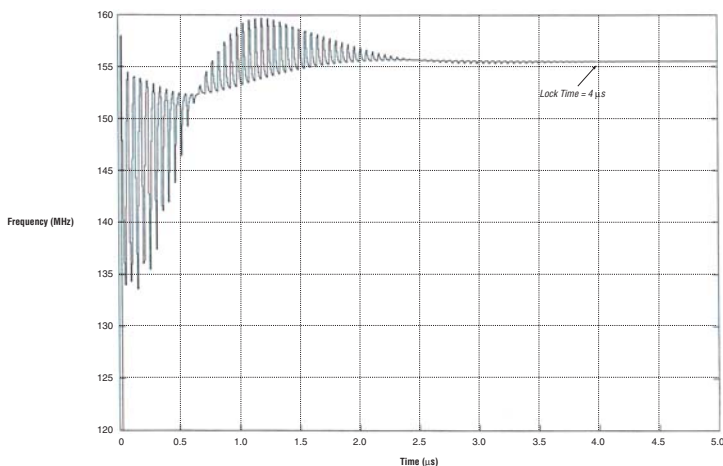


A high-bandwidth PLL can benefit a system that has two cascaded PLLs. If the first PLL uses spread spectrum (as user-induced jitter), the second PLL can track the jitter that is feeding it by using a high-bandwidth setting. A low-bandwidth PLL can, in this case, lose lock due to the spread-spectrum-induced jitter on the input clock.

A low-bandwidth PLL benefits a system using clock switchover. When clock switchover happens, the PLL input temporarily stops. A low-bandwidth PLL would react more slowly to changes to its input clock and take longer to drift to a lower frequency (caused by the input stopping) than a high-bandwidth PLL. Figures 5–28 and 5–29 demonstrate this property. The two plots show the effects of clock switchover with a low- or high-bandwidth PLL. When clock switchover happens, the output of the low-bandwidth PLL (see Figure 5–28) drifts to a lower frequency more slowly than the high-bandwidth PLL output (see Figure 5–29).

*Figure 5–28. Effect of Low Bandwidth on Clock Switchover*
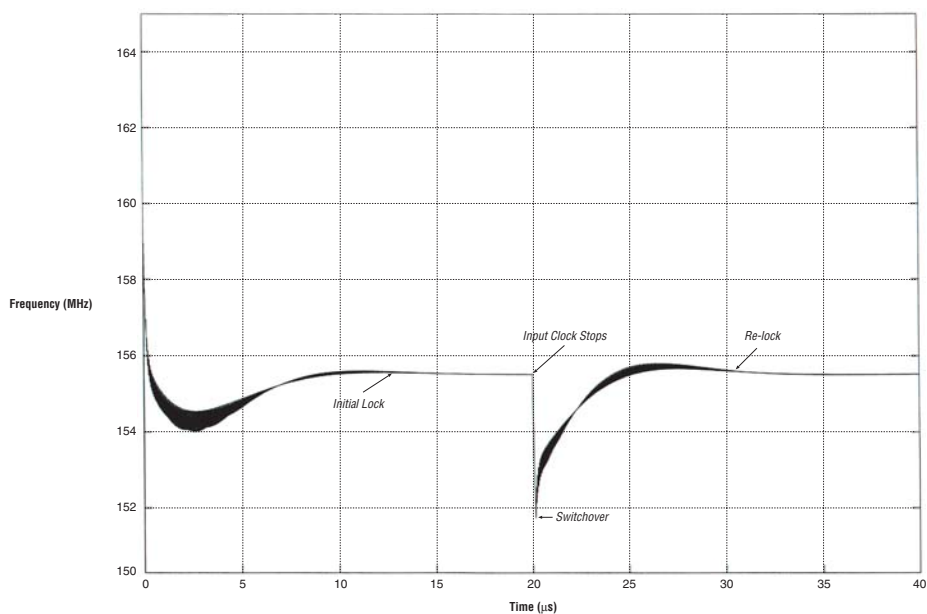
*Figure 5–29. Effect of High Bandwidth on Clock Switchover*
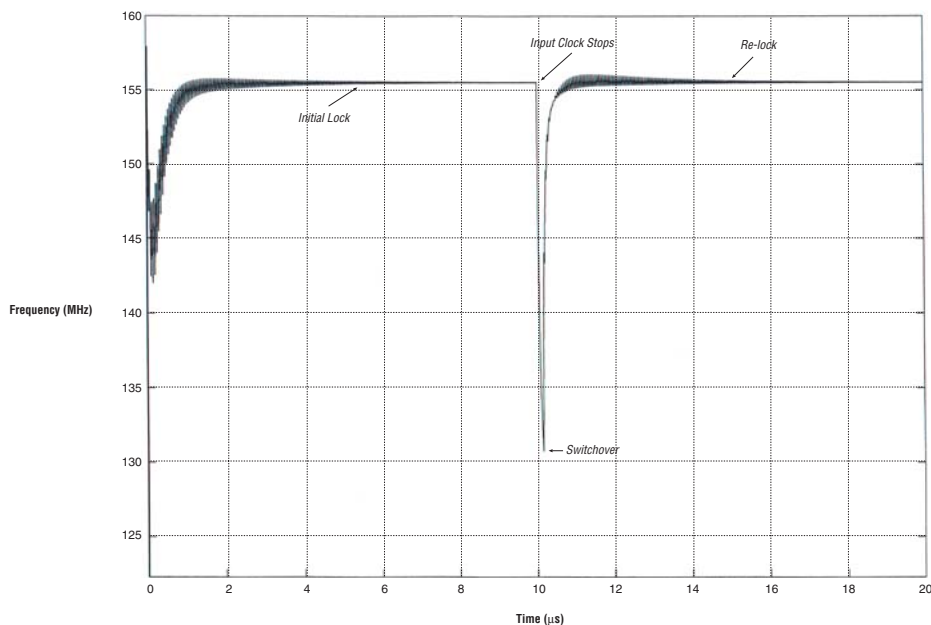


## Implementation

Traditionally, external components such as the VCO or loop filter control a PLL's bandwidth. Most loop filters are made up of passive components such as resistors and capacitors that take up unnecessary board space and increase cost. With Arria GX PLLs, all the components are contained within the device to increase performance and decrease cost.

Arria GX PLLs implement reconfigurable bandwidth by giving you control of the charge pump current, loop filter resistor (R), and high-frequency capacitor $C_H$ values (see Table 5–15). The Arria GX enhanced PLL bandwidth ranges from 130 kHz to 16.9 MHz. The Arria GX fast PLL bandwidth ranges from 1.16 to 28 MHz.

The charge pump current directly affects PLL bandwidth. The higher the charge pump current, the higher the PLL bandwidth. You can choose from a fixed set of values for the charge pump current. Figure 5–30 shows the loop filter and the components that can be set through the Quartus II software. The components are the loop filter resistor, R, and high frequency capacitor, $C_H$, and the charge pump current, $I_{UP}$ or $I_{DN}$.

*Figure 5–30. Loop Filter Programmable Components*
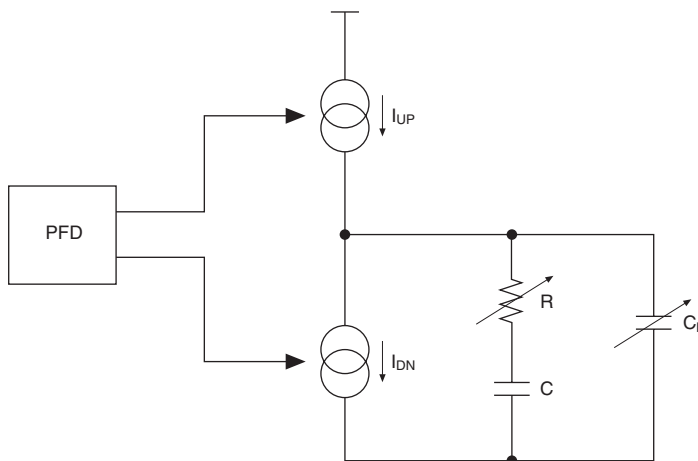


### Software Support

The Quartus II software provides two levels of bandwidth control:

**Megafunction-Based Bandwidth Setting**
The first level of programmable bandwidth allows you to enter a value for the desired bandwidth directly into the Quartus II software using the ALTPLL megafunction. You can also set the bandwidth parameter in the ALTPLL megafunction to the desired bandwidth. The Quartus II software selects the best bandwidth parameters available to match your bandwidth request. If the individual bandwidth setting request is not available, the Quartus II software selects the closest achievable value.

**Advanced Bandwidth Setting**
An advanced level of control is also possible using advanced loop filter parameters. You can dynamically change the charge pump current, loop filter resistor value, and loop filter (high frequency) capacitor value. The

parameters for these changes are: `charge_pump_current`, `loop_filter_r`, and `loop_filter_c`. Each parameter supports the specific range of values listed in Table 5–15.

| Table 5–15. Advanced Loop Filter Parameters | |
|---|---|
| **Parameter** | **Values** |
| Resistor values (kΩ) | *(1)* |
| High-frequency capacitance values (pF) | *(1)* |
| Charge pump current settings (μA) | *(1)* |

*Note to Table 5–15:*
(1)   For more information, see *AN 367: Implementing PLL Reconfiguration in Stratix II Devices*. The information presented in AN 367 applies to Arria GX enhanced and fast PLLs as well.

For more information about Quartus II software support of reconfigurable bandwidth, see the PLL Reconfiguration section in the *Embedded Peripherals* section of the *Quartus II Handbook*.

# PLL Reconfiguration

PLLs use several divide counters and different VCO phase taps to perform frequency synthesis and phase shifts. In Arria GX enhanced and fast PLLs, the counter value and phase are configurable in real time. In addition, you can change the loop filter and charge pump components, which affect the PLL bandwidth, on-the-fly. You can control these PLL components to update the output clock frequency, PLL bandwidth, and phase-shift variation in real time, without the need to reconfigure the entire FPGA.

For more information about PLL reconfiguration in Arria GX devices, see *AN 367: Implementing PLL Reconfiguration in Stratix II Devices*. The information presented in AN 367 applies to Arria GX enhanced and fast PLLs as well.

# Spread-Spectrum Clocking

Digital clocks are square waves with short rise times and a 50% duty cycle. These high-speed clocks concentrate a significant amount of energy in a narrow bandwidth at the target frequency and at higher frequency harmonics. This results in high energy peaks and increased electromagnetic interference (EMI). The radiated noise from the energy peaks travels in free air and, if not minimized, can lead to corrupted data and intermittent system errors, which can jeopardize system reliability.

Traditional methods for limiting EMI include shielding, filtering, and multi-layer printed circuit boards (PCBs). However, these methods significantly increase overall system cost and sometimes are not enough

to meet EMI compliance. Spread-spectrum technology provides you with a simple and effective technique for reducing EMI without additional cost and the trouble of re-designing a board.

Spread-spectrum technology modulates the target frequency over a small range. For example, if a 100-MHz signal has a 0.5% down-spread modulation, the frequency is swept from 99.5 to 100 MHz. Figure 5–31 gives a graphical representation of the energy present in a spread-spectrum signal versus a non-spread spectrum-signal. It is apparent that instead of concentrating the energy at the target frequency, the energy is re-distributed across a wider band of frequencies, which reduces peak energy. Not only is there a reduction in fundamental peak EMI components, but there is also a reduction in EMI of the higher order harmonics. Because some regulations focus on peak EMI emissions rather than average EMI emissions, spread-spectrum technology is a valuable method of EMI reduction.

*Figure 5–31. Spread-Spectrum Signal Energy Versus Non-Spread-Spectrum Signal Energy*
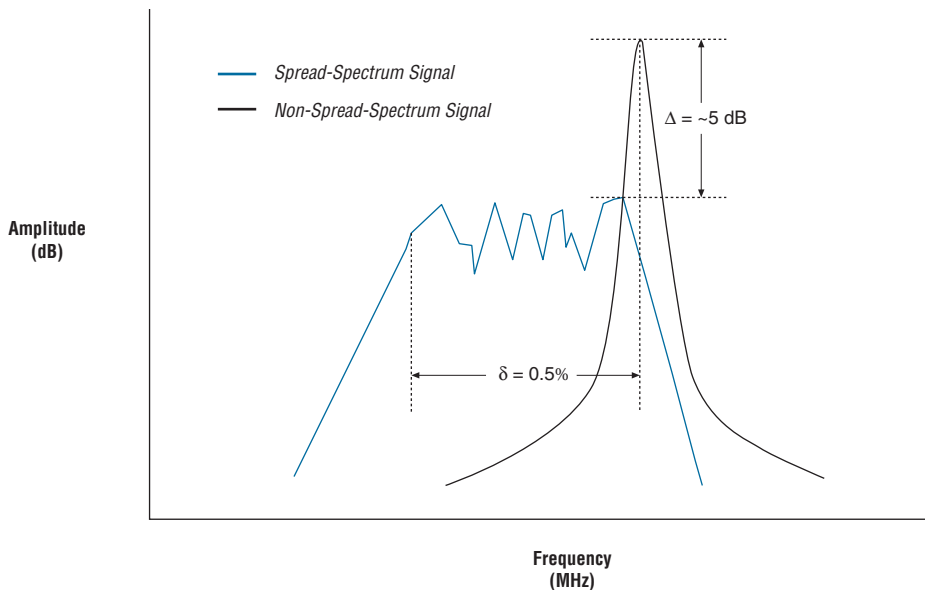
Spread-spectrum technology would benefit a design with high EMI emissions and/or strict EMI requirements. Device-generated EMI is dependent on frequency and output voltage swing amplitude and edge rate. For example, a design using LVDS already has low EMI emissions because of low-voltage swing. The differential LVDS signal also allows for EMI rejection within the signal. Therefore, this situation may not require spread-spectrum technology.

☞    Spread-spectrum clocking is only supported in Arria GX enhanced PLLs, not fast PLLs.

### Implementation

Arria GX device enhanced PLLs feature spread-spectrum technology to reduce the EMIs emitted from the device. The enhanced PLL provides approximately 0.5% down spread using a triangular, also known as linear, modulation profile. The modulation frequency is programmable and ranges from approximately 100 to 500 kHz. The spread percentage is based on the clock input to the PLL and the $m$ and $n$ settings. Spread-spectrum technology reduces peak energy by four to six dB at the target frequency. However, this number is dependent on bandwidth and $m$ and $n$ counter values and can vary from design to design.

Spread percentage, also known as modulation width, is defined as the percentage that the design modulates the target frequency. A negative (–) percentage indicates a down spread, a positive (+) percentage indicates an up spread, and a (±) indicates a center spread. Modulation frequency is the frequency of the spreading signal, or how fast the signal sweeps from the minimum to the maximum frequency. Down-spread modulation shifts target frequency down by half the spread percentage, centering the modulated waveforms on a new target frequency.

The $m$ and $n$ counter values are toggled at the same time between two fixed values. The loop filter then slowly changes the VCO frequency to provide the spreading effect, which results in a triangular modulation. An additional spread-spectrum counter (shown in Figure 5–32) sets the modulation frequency. Figure 5–32 shows how spread-spectrum technology is implemented in the Arria GX device enhanced PLL.

*Figure 5–32. Arria GX Spread-Spectrum Circuit Block*
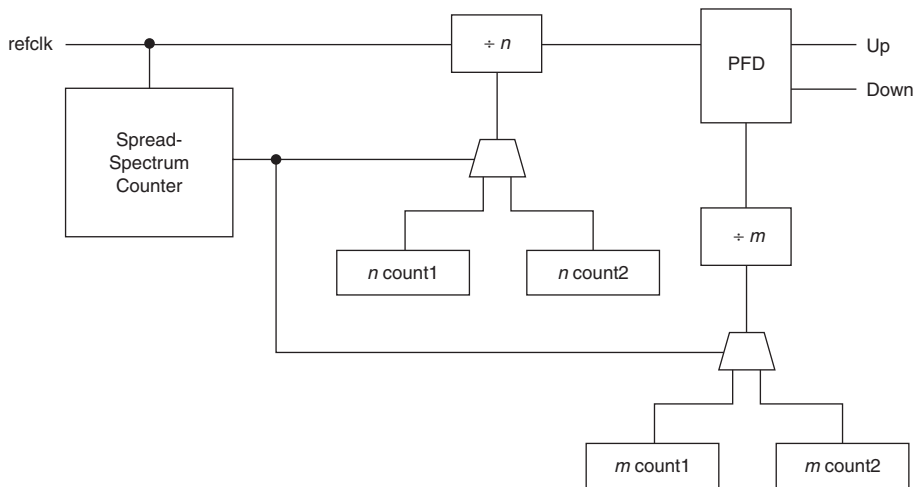


Figure 5–33 shows a VCO frequency waveform when toggling between different counter values. Because the enhanced PLL switches between two different $m$ and $n$ values, the result is a straight line between two frequencies, which gives a linear modulation. The magnitude of modulation is determined by the ratio of two $m/n$ sets. The percent spread is determined by:

percent spread $= (f_{VCOmax} - f_{VCOmin})/f_{VCOmax} = 1 - [(m_2 \times n_1)/(m_1 \times n_2)]$.

The maximum and minimum VCO frequency is defined as:

■  $f_{VCOmax} = (m_1/n_1) \times f_{REF}$
■  $f_{VCOmin} = (m_2/n_2) \times f_{REF}$

*Figure 5–33. VCO Frequency Modulation Waveform*
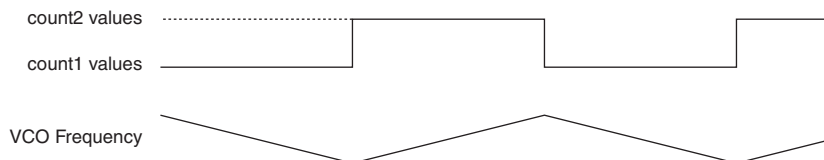
*Software Support*

You can enter the desired down-spread percentage and modulation frequency in the ALTPLL megafunction through the Quartus II software. Alternatively, you can set the **downspread** parameter in the ALTPLL megafunction to the desired down-spread percentage. Timing analysis ensures the design operates at the maximum spread frequency and meets all timing requirements.

For more information about PLL support in the Quartus II software, see the *altpll Megafunction User Guide*.

*Guidelines*

If the design cascades PLLs, the source (upstream) PLL must have a low-bandwidth setting while the destination (downstream) PLL must have a high-bandwidth setting. The upstream PLL must have a low-bandwidth setting because a PLL does not generate jitter higher than its bandwidth. The downstream PLL must have a high bandwidth setting to track the jitter. The design must use the spread-spectrum feature in a low-bandwidth PLL so the Quartus II software will automatically set the spread-spectrum PLL bandwidth to low.

☞ If you use the programmable or reconfigurable bandwidth features, you cannot use spread spectrum.

Arria GX devices can accept a spread-spectrum input with typical modulation frequencies. However, the device cannot automatically detect that the input is a spread-spectrum signal. Instead, the input signal looks like deterministic jitter at the input of the downstream PLL.

Spread spectrum can have a minor effect on the output clock by increasing period jitter. Period jitter is the deviation of a clock's cycle time from its previous cycle position. Period jitter measures the variation of the clock output transition from its ideal position over consecutive edges.

With down-spread modulation, the peak of the modulated waveform is the actual target frequency. Therefore, the system never exceeds the maximum clock speed. To maintain reliable communication, the entire system and subsystem should use the Arria GX device as the clock source. Communication could fail if the Arria GX logic array is clocked by the spread-spectrum clock, but the data it receives from another device is not clocked by the spread spectrum.

Because spread-spectrum affects the $m$ counter values, all spread-spectrum PLL outputs are effected. Therefore, if only one spread-spectrum signal is needed, the clock signal should use a separate PLL without other outputs from that PLL.

No special considerations are needed when using spread-spectrum with the clock switchover feature. This is because the clock switchover feature does not affect the *m* and *n* counter values, which are the counter values switching when using the spread-spectrum feature.

## Board Layout

The enhanced and fast PLL circuits in Arria GX devices contain analog components embedded in a digital device. These analog components have separate power and ground pins to minimize noise generated by the digital components. Arria GX enhanced and fast PLLs use separate $V_{CC}$ and ground pins to isolate circuitry and improve noise resistance.

### VCCA and GNDA

Each enhanced and fast PLL uses separate $V_{CC}$ and ground pin pairs for their analog circuitry. The analog circuit power and ground pin for each PLL is called VCCA_PLL<*PLL number*> and GNDA_PLL<*PLL number*>. Connect the VCCA power pin to a 1.2-V power supply, even if you do not use the PLL. Isolate the power connected to VCCA from the power to the rest of the Arria GX device or any other digital device on the board. You can use one of three different methods of isolating the VCCA pin: separate VCCA power planes, a partitioned VCCA island within the VCCINT plane, and thick VCCA traces.

#### Separate VCCA Power Plane

A mixed signal system is already partitioned into analog and digital sections, each with its own power planes on the board. To isolate the VCCA pin using a separate VCCA power plane, connect the VCCA pin to the analog 1.2-V power plane.

#### Partitioned VCCA Island Within the VCCINT Plane

Fully digital systems do not have a separate analog power plane on the board. Because it is expensive to add new planes to the board, you can create islands for VCCA_PLL. Figure 5–34 shows an example board layout with an analog power island. The dielectric boundary that creates the island should be 25 mils thick. Figure 5–35 shows a partitioned plane within VCCINT for VCCA.

*Figure 5–34. VCCINT Plane Partitioned for VCCA Island*
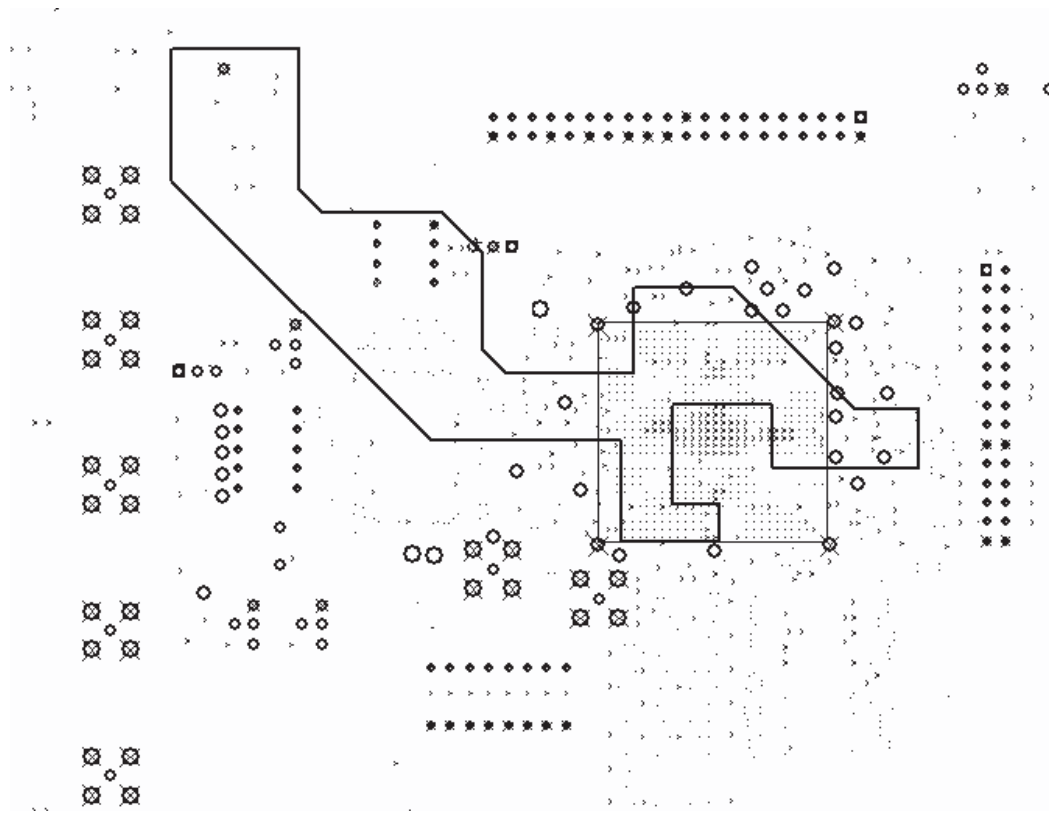


*Thick VCCA Trace*

Because of board constraints, you may not be able to partition a VCCA island. Instead, run a thick trace from the power supply to each VCCA pin. The traces should be at least 20 mils thick.

In each of these three cases, you should filter each VCCA_PLL pin with a decoupling circuit, as shown in Figure 5–35. Place a ferrite bead that exhibits high impedance at frequencies of 50 MHz or higher and a 10-μF tantalum parallel capacitor where the power enters the board. Decouple each VCCA_PLL pin with a 0.1-μF and 0.001-μF parallel combination of ceramic capacitors located as close as possible to the Arria GX device. You can connect the GNDA_PLL pins directly to the same ground plane as the device's digital ground.

*Figure 5–35. PLL Power Schematic for Arria GX PLLs*
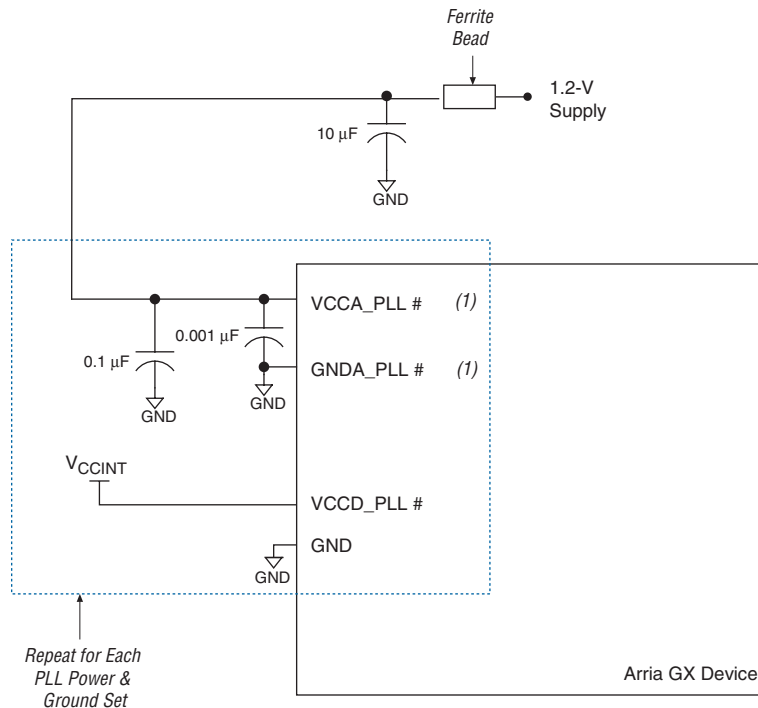


*Note to Figure 5–35*:
(1)   This applies to all Arria GX PLLs.

## VCCD

The digital power and ground pins are labeled VCCD_PLL<*PLL number*> and GND_PLL<*PLL number*>. The VCCD pin supplies the power for the digital circuitry in the PLL. Connect these VCCD pins to the quietest digital supply on the board. In most systems, this is the digital 1.2-V supply supplied to the device's VCCINT pins. Connect the VCCD pins to a power supply even if you do not use the PLL. When connecting the VCCD pins to VCCINT, you do not need any filtering or isolation. You can connect the GND pins directly to the same ground plane as the device's digital ground (see Figure 5–35).

## External Clock Output Power

Enhanced PLLs 5, 6, 11, and 12 also have isolated power pins for their dedicated external clock outputs (VCC_PLL5_OUT, VCC_PLL6_OUT, VCC_PLL11_OUT, and VCC_PLL12_OUT, respectively). Because the dedicated external clock outputs from a particular enhanced PLL are powered by separate power pins, they are less susceptible to noise. They also reduce the overall jitter of the output clock by providing improved isolation from switching I/O pins.

☞  I/O pins that reside in PLL banks 9 through 12 are powered by the VCC_PLL<*5, 6, 11, or 12*>_OUT pins, respectively. If a particular device does not support PLLs 11 or 12, any I/O pins that reside in bank 11 are powered by the VCCIO3 pin, and any I/O pins that reside in bank 12 are powered by the VCCIO8 pin.

The VCC_PLL_OUT pins can by powered by 3.3, 2.5, 1.8, or 1.5 V, depending on the I/O standard for the clock output from a particular enhanced PLL, as shown in Figure 5–36.

*Figure 5–36. External Clock Output Pin Association with Output Power*
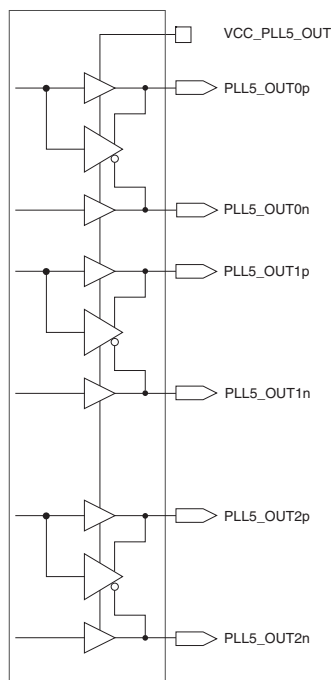
🖙 Filter each isolated power pin with a decoupling circuit shown in Figure 5–37. Decouple the isolated power pins with parallel combination of 0.1- and 0.001-µF ceramic capacitors located as close as possible to the Arria GX device.

*Figure 5–37. Arria GX PLL External Clock Output Power Ball Connection*
*Note (1)*
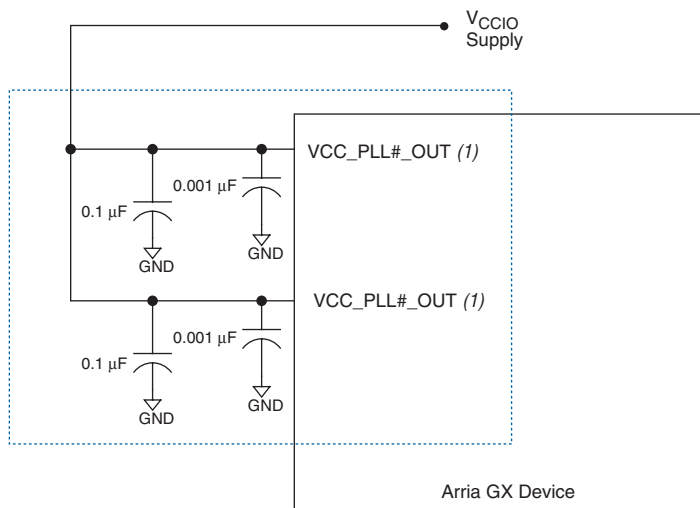


*Note to Figure 5–37:*
(1)    This applies only to enhanced PLLs 5, 6, 11, and 12.

## Guidelines

Use the following guidelines for optimal jitter performance on the external clock outputs from enhanced PLLs 5, 6, 11, and 12. If all outputs are running at the same frequency, these guidelines are not necessary to improve performance.

■    Use phase shift to ensure edges are not coincident on all the clock outputs.
■    Use phase shift to skew clock edges with respect to each other for best jitter performance.

If you cannot drive multiple clocks of different frequencies and phase shifts or isolate banks, you should control the drive capability on the lower-frequency clock. Reducing how much current the output buffer has to supply can reduce noise. Minimize capacitive load on the slower frequency output and configure the output buffer to lower current

strength. The higher-frequency output should have an improved performance, but this may degrade the performance of your lower-frequency clock output.

# PLL Specifications

For information about PLL timing specifications, refer to the *DC & Switching Characteristics* chapter in volume 1 of the *Arria GX Device Handbook*.

# Clocking

Arria GX devices provide a hierarchical clock structure and multiple PLLs with advanced features. The large number of clocking resources in combination with clock synthesis precision provided by enhanced and fast PLLs provides a complete clock-management solution.

## Global and Hierarchical Clocking

Arria GX devices provide 16 dedicated global clock networks and 32 regional clock networks. These clocks are organized into a hierarchical clock structure that allows for 24 unique clock sources per device quadrant with low skew and delay. This hierarchical clocking scheme provides up to 48 unique clock domains within the entire Arria GX device. Table 5–16 lists the clock resources available on Arria GX devices.

There are 12 dedicated clock pins on Arria GX devices to drive either the global or regional clock networks. Four clock pins drive three sides of the Arria GX device, as shown in Figures 5–38 and 5–39. Enhanced and fast PLL outputs can also drive the global and regional clock networks.

| Table 5–16. Clock Resource Availability in Arria GX Devices (Part 1 of 2) | |
|---|---|
| **Description** | **Arria GX Device Availability** |
| Number of clock input pins | 12 |
| Number of global clock networks | 16 |
| Number of regional clock networks | 32 |
| Global clock input sources | Clock input pins, PLL outputs, logic array, inter-transceiver clocks |
| Regional clock input sources | Clock input pins, PLL outputs, logic array, inter-transceiver clocks |

**Table 5–16. Clock Resource Availability in Arria GX Devices (Part 2 of 2)**

| Description | Arria GX Device Availability |
|---|---|
| Number of unique clock sources in a quadrant | 24 (16 GCLK and 8 RCLK clocks) |
| Number of unique clock sources in the entire device | 48 (16 GCLK and 32 RCLK clocks) |
| Power-down mode | GCLK, RCLK networks, dual-regional clock region |
| Clocking regions for high fan-out applications | Quadrant region, dual-regional, entire device via GCLK or RCLK networks |

*Global Clock Network*

Global clocks drive throughout the entire device, feeding all device quadrants. All resources within the device IOEs, adaptive logic modules (ALMs), digital signal processing (DSP) blocks, and all memory blocks can use the global clock networks as clock sources. These resources can also be used for control signals, such as clock enables and synchronous or asynchronous clears fed by an external pin. Internal logic can also drive the global clock networks for internally generated global clocks and asynchronous clears, clock enables, or other control signals with large fanout. Figure 5–38 shows the 12 dedicated CLK pins driving global clock networks.

*Figure 5–38. Global Clocking Note (1)*
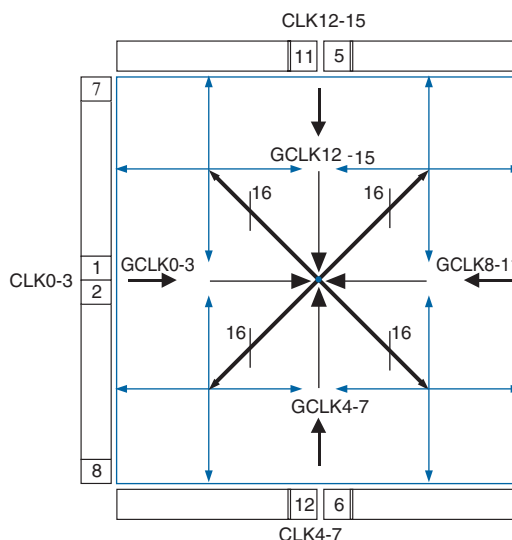


*Note to Figure 5–38:*
(1)    Arria GX devices do not have PLLs 3, 4, 9, and 10 or clock pins 8, 9, 10, and 11.

*Regional Clock Network*

Eight regional clock networks within each quadrant of the Arria GX device are driven by dedicated `CLK` input pins or from PLL outputs. The regional clock networks only pertain to the quadrant they drive into. The regional clock networks provide the lowest clock delay and skew for logic contained within a single quadrant. Internal logic can also drive the regional clock networks for internally generated regional clocks and asynchronous clears, clock enables, or other control signals with large fanout. The `CLK` pins symmetrically drive the RCLK networks within a particular quadrant, as shown in Figure 5–39. Refer to Table 5–17 on page 5–65 and Table 5–18 on page 5–65 for RCLK connections from `CLK` pins and PLLs.

*Figure 5–39. Regional Clocking Note (1)*
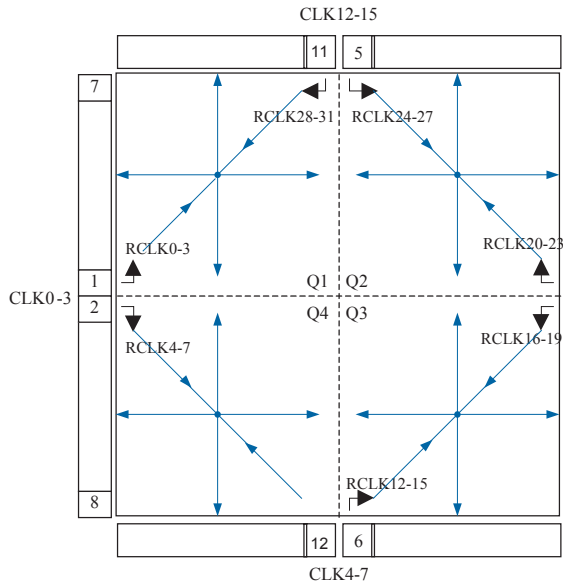


*Note to Figure 5–39:*
(1)    Arria GX devices do not have PLLs 3, 4, 9, and 10 or clock pins 8, 9, 10, and 11.

## Clock Sources Per Region

Each Arria GX device has 16 global clock networks and 32 regional clock networks that provide 48 unique clock domains for the entire device. There are 24 unique clocks available in each quadrant (16 global clocks and 8 regional clocks) as the input resources for registers (see Figure 5–40).

*Figure 5–40. Hierarchical Clock Networks Per Quadrant*
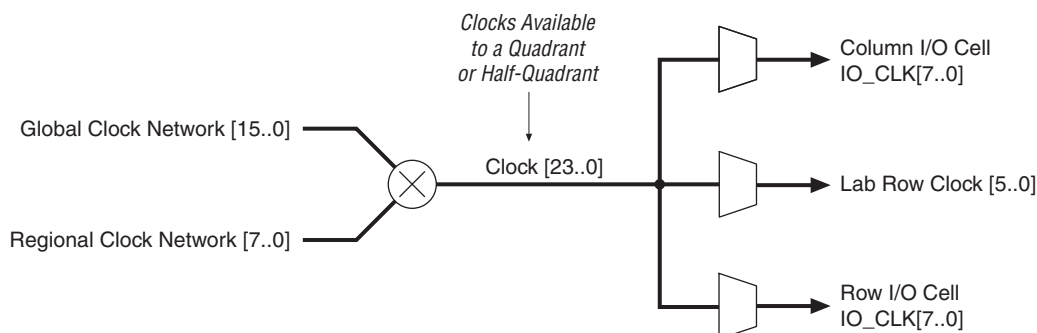


Arria GX clock networks provide three different clocking regions:

■ Entire device clock region
■ Quadrant clock region
■ Dual-regional clock region

These clock network options provide more flexibility for routing signals that have high fan-out to improve interface timing. By having various sized clock regions, it is possible to prioritize the number of registers the network can reach versus the total delay of the network.

In the first clock scheme, a source (not necessarily a clock signal) drives a global clock network that can be routed through the entire device. This has the maximum delay for a low-skew high-fan-out signal but allows the signal to reach every block within the device. This is a good option for routing global resets or clear signals.

In the second clock scheme, a source drives a single-quadrant region. This represents the fastest, low-skew, high-fan-out signal routing resource within a quadrant. The limitation to this resource is that it only covers a single quadrant.

In the third clock scheme, a single source (clock pin or PLL output) can generate a dual-regional clock by driving two regional clock network lines (one from each quadrant). This allows logic that spans multiple quadrants to utilize the same low-skew clock. The routing of this signal on an entire side has approximately the same speed as in a quadrant clock region. The internal logic-array routing that can drive a regional clock also supports this feature. This means internal logic can drive a dual-regional clock network. Corner fast PLL outputs only span one quadrant and hence cannot form a dual-regional clock network. Figure 5–41 shows this feature pictorially.

*Figure 5–41. Arria GX Dual-Regional Clock Region*



Clock pins or PLL outputs can drive half of the device to create dual-reginal clocking regions for improved I/O interface timing.
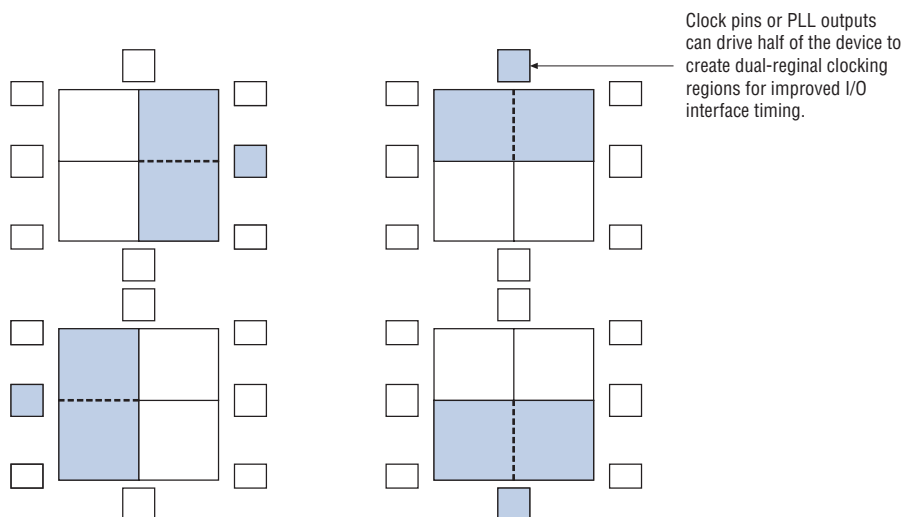
The 12 clock input pins, enhanced or fast PLL outputs, and internal logic array can be the clock input sources to drive onto either global or regional clock networks. The CLKn pins also drive the global clock network, as shown in Table 5–20 on page 5–68. Tables 5–17 and 5–18 show the connectivity between the CLK pins as well as the global and regional clock networks.

**Clock Inputs**
The 12 clock input pins (CLK) are also used for high-fan-out control signals, such as asynchronous clears, presets, clock enables, or protocol signals such as TRDY and IRDY for PCI through global or regional clock networks.

**Internal Logic Array**
Each global and regional clock network can also be driven by logic-array routing to enable internal logic to drive a high-fan-out, low-skew signal.

**PLL Outputs**
All clock networks can be driven by the PLL counter outputs.

Table 5–17 shows the connection of clock pins to global clock resources. The reason for the higher level of connectivity is to support user-controllable global clock multiplexing.

| Table 5–17. Clock Input Pin Connectivity to Global Clock Networks | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Clock Resource** | **CLK(p) (Pin)** | | | | | | | | | | | |
| | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **12** | **13** | **14** | **15** |
| GCLK0 | ✓ | ✓ | — | — | — | — | — | — | — | — | — | — |
| GCLK1 | ✓ | ✓ | — | — | — | — | — | — | — | — | — | — |
| GCLK2 | — | — | ✓ | ✓ | — | — | — | — | — | — | — | — |
| GCLK3 | — | — | ✓ | ✓ | — | — | — | — | — | — | — | — |
| GCLK4 | — | — | — | — | ✓ | ✓ | — | — | — | — | — | — |
| GCLK5 | — | — | — | — | ✓ | ✓ | — | — | — | — | — | — |
| GCLK6 | — | — | — | — | — | — | ✓ | ✓ | — | — | — | — |
| GCLK7 | — | — | — | — | — | — | ✓ | ✓ | — | — | — | — |
| GCLK8 | — | — | — | — | — | — | — | — | — | — | — | — |
| GCLK9 | — | — | — | — | — | — | — | — | — | — | — | — |
| GCLK10 | — | — | — | — | — | — | — | — | — | — | — | — |
| GCLK11 | — | — | — | — | — | — | — | — | — | — | — | — |
| GCLK12 | — | — | — | — | — | — | — | — | — | — | ✓ | ✓ |
| GCLK13 | — | — | — | — | — | — | — | — | — | — | ✓ | ✓ |
| GCLK14 | — | — | — | — | — | — | — | — | ✓ | ✓ | — | — |
| GCLK15 | — | — | — | — | — | — | — | — | ✓ | ✓ | — | — |

*Note to Table 5–17:*

(1)    Clock pins 8, 9, 10, and 11 are not available in Arria GX devices.

Table 5–18 summarizes the connectivity between the clock pins and the regional clock networks. Here, each clock pin can drive two regional clock networks, facilitating stitching of the clock networks to support the ability to drive two quadrants with the same clock or signal.

| Table 5–18. Clock Input Pin Connectivity to Regional Clock Networks *Note (1)*   *(Part 1 of 2)* | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Clock Resource** | **CLK(p) (Pin)** | | | | | | | | | | | |
| | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **12** | **13** | **14** | **15** |
| RCLK0 | ✓ | — | — | — | — | — | — | — | — | — | — | — |
| RCLK1 | — | ✓ | — | — | — | — | — | — | — | — | — | — |

*Table 5–18. Clock Input Pin Connectivity to Regional Clock Networks Note (1)  (Part 2 of 2)*

| Clock Resource | CLK(p) (Pin) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 12 | 13 | 14 | 15 |
| RCLK2 | — | — | ✓ | | — | — | — | — | — | — | — | — |
| RCLK3 | — | — | — | ✓ | — | — | — | — | — | — | — | — |
| RCLK4 | ✓ | — | — | — | — | — | — | — | — | — | — | — |
| RCLK5 | — | ✓ | — | — | — | — | — | — | — | — | — | — |
| RCLK6 | — | — | ✓ | — | — | — | — | — | — | — | — | — |
| RCLK7 | — | — | — | ✓ | — | — | — | — | — | — | — | — |
| RCLK8 | — | — | — | — | ✓ | — | — | — | — | — | — | — |
| RCLK9 | — | — | — | — | — | ✓ | — | — | — | — | — | — |
| RCLK10 | — | — | — | — | — | — | ✓ | — | — | — | — | — |
| RCLK11 | — | — | — | — | — | — | — | ✓ | — | — | — | — |
| RCLK12 | — | — | — | — | ✓ | — | — | — | — | — | — | — |
| RCLK13 | — | — | — | — | — | ✓ | — | — | — | — | — | — |
| RCLK14 | — | — | — | — | — | — | ✓ | | — | — | — | — |
| RCLK15 | — | — | — | — | — | — | — | ✓ | — | — | — | — |
| RCLK16 | — | — | — | — | — | — | — | — | — | — | — | — |
| RCLK17 | — | — | — | — | — | — | — | — | — | — | — | — |
| RCLK18 | — | — | — | — | — | — | — | — | — | — | — | — |
| RCLK19 | — | — | — | — | — | — | — | — | — | — | — | — |
| RCLK20 | — | — | — | — | — | — | — | — | — | — | — | — |
| RCLK21 | — | — | — | — | — | — | — | — | — | — | — | — |
| RCLK22 | — | — | — | — | — | — | — | — | — | — | — | — |
| RCLK23 | — | — | — | — | — | — | — | — | — | — | — | — |
| RCLK24 | — | — | — | — | — | — | — | — | — | — | ✓ | — |
| RCLK25 | — | — | — | — | — | — | — | — | — | — | — | ✓ |
| RCLK26 | — | — | — | — | — | — | — | — | ✓ | — | — | — |
| RCLK27 | — | — | — | — | — | — | — | — | — | ✓ | — | — |
| RCLK28 | — | — | — | — | — | — | — | — | — | — | ✓ | — |
| RCLK29 | — | — | — | — | — | — | — | — | — | — | — | ✓ |
| RCLK30 | — | — | — | — | — | — | — | — | ✓ | — | — | — |
| RCLK31 | — | — | — | — | — | — | — | — | — | ✓ | — | — |

*Note to Table 5–18:*

(1)    Clock pins 8, 9, 10, and 11 are not available in Arria GX devices.

## Clock Input Connections

Four `CLK` pins drive each enhanced PLL. You can use any of the pins for clock switchover inputs into the PLL. The `CLK` pins are the primary clock source for clock switchover, which is controlled in the Quartus II software. Enhanced PLLs 5, 6, 11, and 12 also have feedback input pins.

Input clocks for fast PLLs 1, 2, 3, and 4 come from the `CLK` pins. A multiplexer chooses one of two possible `CLK` pins to drive each PLL. This multiplexer is not a clock switchover multiplexer and is only used for clock input connectivity.

Either an `FPLLCLK` input pin or a `CLK` pin can drive the fast PLLs in the corners (7 and 8) when used for general-purpose applications. `CLK` pins cannot drive these fast PLLs in high-speed differential I/O mode.

Table 5–19 shows which PLLs are available in each Arria GX device and which input clock pin drives which PLLs.

| *Table 5–19. Arria GX Device PLLs and PLL Clock Pin Drivers   (Part 1 of 2)* | | | | | | | |
|---|---|---|---|---|---|---|---|
| | **All Devices** | | | | **EP1AGX50 to EP1AGX90 Devices** | | |
| **Input Pin** | **Fast PLLs** | | **Enhanced PLLs** | | **Fast PLLs** | | **Enhanced PLLs** | |
| | 1 | 2 | 5 | 6 | 7 | 8 | 11 | 12 |
| `CLK0` | ✓ | ✓ | — | — | ✓ | ✓ | — | — |
| `CLK1` | ✓ | ✓ | — | — | ✓ | ✓ | — | — |
| `CLK2` | ✓ | ✓ | — | — | ✓ | ✓ | — | — |
| `CLK3` | ✓ | ✓ | — | — | ✓ | ✓ | — | — |
| `CLK4` | — | — | — | ✓ | — | — | — | ✓ |
| `CLK5` | — | — | — | ✓ | — | — | — | ✓ |
| `CLK6` | — | — | — | ✓ | — | — | — | ✓ |
| `CLK7` | — | — | — | ✓ | — | — | — | ✓ |
| `CLK12` | — | — | ✓ | — | — | — | ✓ | — |
| `CLK13` | — | — | ✓ | — | — | — | ✓ | — |
| `CLK14` | — | — | ✓ | — | — | — | ✓ | — |
| `CLK15` | — | — | ✓ | — | — | — | ✓ | — |
| `PLL5_FB` | — | — | ✓ | — | — | — | — | — |
| `PLL6_FB` | — | — | — | ✓ | — | — | — | — |
| `PLL11_FB` | — | — | — | — | — | — | ✓ | — |
| `PLL12_FB` | — | — | — | — | — | — | — | ✓ |

**Table 5–19. Arria GX Device PLLs and PLL Clock Pin Drivers   (Part 2 of 2)**

| Input Pin | All Devices | | | | EP1AGX50 to EP1AGX90 Devices | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | Fast PLLs | | Enhanced PLLs | | Fast PLLs | | Enhanced PLLs | |
| | 1 | 2 | 5 | 6 | 7 | 8 | 11 | 12 |
| PLL_ENA | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| FPLL7CLK | — | — | — | — | ✓ | — | — | — |
| FPLL8CLK | — | — | — | — | — | ✓ | — | — |
| FPLL9CLK | — | — | — | — | — | — | — | — |
| FPLL10CLK | — | — | — | — | — | — | — | — |

*Notes to Table 5–19:*
(1)   PLLs 3, 4, 9, and 10 are not available in Arria GX devices.
(2)   Clock connection is available. For more information about the maximum frequency, contact Altera Applications Group.
(3)   This is a dedicated high-speed clock input. For more information about the maximum frequency, contact Altera Applications.
(4)   Input pins CLK[11..8] are not available in Arria GX devices.

### CLK(n) Pin Connectivity to Global Clock Networks

In Arria GX devices, the clk(n) pins can also feed the global clock network. Table 5–20 shows the clk(n) pin connectivity to global clock networks.

**Table 5–20. CLK(n) Pin Connectivity to Global Clock Network**

| Clock Resource | CLK(n) pin | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | 4 | 5 | 6 | 7 | 12 | 13 | 14 | 15 |
| GCLK4 | ✓ | — | — | — | — | — | — | — |
| GCLK5 | — | ✓ | — | — | — | — | — | — |
| GCLK6 | — | — | ✓ | — | — | — | — | — |
| GCLK7 | — | — | — | ✓ | — | — | — | — |
| GCLK12 | — | — | — | — | — | — | ✓ | — |
| GCLK13 | — | — | — | — | — | — | — | ✓ |
| GCLK14 | — | — | — | — | ✓ | — | — | — |
| GCLK15 | — | — | — | — | — | ✓ | — | — |

## Clock Source Control For Enhanced PLLs

The clock input multiplexer for enhanced PLLs is shown in Figure 5–42. This block allows selection of the PLL clock reference from several different sources. The clock source to an enhanced PLL can come from any one of four clock input pins CLK[3..0], or from a logic-array clock. Clock input pin connections to respective enhanced PLLs are shown in Table 5–20. The multiplexer-select lines are set in the configuration file only. Once programmed, this block cannot be changed without loading a new configuration file. The Quartus II software automatically sets the multiplexer-select signals depending on the clock sources that you select in your design.

*Figure 5–42. Enhanced PLL Clock Input Multiplex Logic*
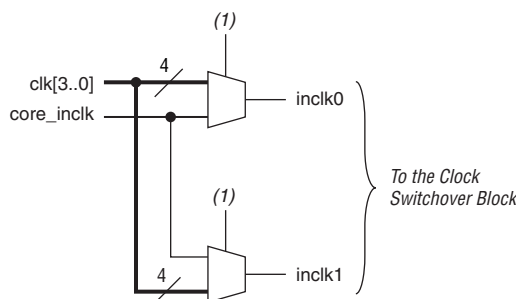


*Note to Figure 5–42:*
(1)    Input clock multiplexing is controlled through a configuration file only and cannot be dynamically controlled in user mode.

## Clock Source Control for Fast PLLs

Each center fast PLL has five clock input sources, four from clock input pins and one from a logic array signal. When using clock input pins as the clock source, you can perform manual clock switchover among the input clock sources. The clock input multiplexer control signals for performing clock switchover are from core signals. Figure 5–43 shows the clock input multiplexer control circuit for a center fast PLL.

*Figure 5–43. Center Fast PLL Clock Input Multiplexer Control*
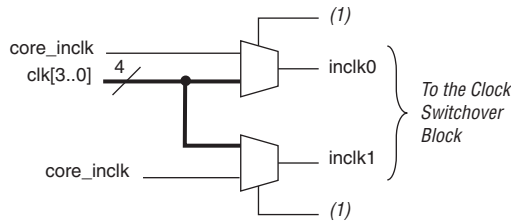


*Note to Figure 5–43:*
(1)  Input clock multiplexing is controlled through a configuration file only and cannot
     be dynamically controlled in user mode.

Each corner fast PLL has three clock input sources, one from a dedicated corner clock input pin, one from a center clock input pin, and one from a logic array clock. Figure 5–44 shows a block diagram of the clock input multiplexer control circuit for a corner fast PLL. Only the corner FPLLCLK pin is fully compensated.

*Figure 5–44. Corner Fast PLL Clock Input Multiplexer Control*
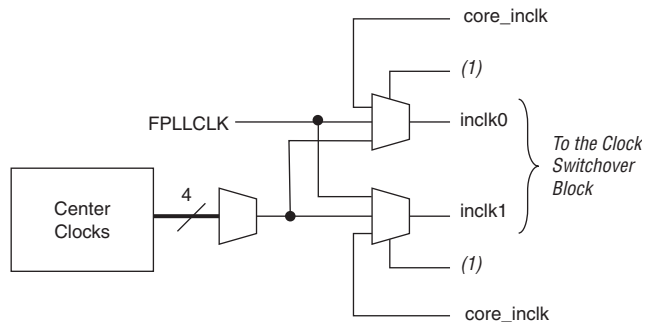


*Note to Figure 5–44:*
(1)  Input clock multiplexing is controlled through a configuration file only and cannot
     be dynamically controlled in user mode.

## Delay Compensation for Fast PLLs

Each center fast PLL can be fed by any one of four possible input clock pins. Among the four clock input signals, only two are fully compensated; for example, the clock delay to the fast PLL matches the delay in the data input path when used in the LVDS receiver mode. The two clock inputs that match the data input path are located next to the fast PLL. The two clock inputs that do not match the data input path are

located next to the neighboring fast PLL. Figure 5–45 shows the above description for the left-side center fast PLL pair. If the PLL is used in non-LVDS modes, you can use any of the four dedicated clock inputs and are compensated.

Fast PLL 1 and PLL 2 can choose among CLK[3..0] as the clock input source. However, for fast PLL 1, only CLK0 and CLK1 have their delay matched to the data input path delay when used in LVDS receiver mode operation. The delay from CLK2 or CLK3 to fast PLL 1 does not match the data input delay. For fast PLL 2, only CLK2 and CLK3 have their delay matched to the data input path delay in LVDS receiver mode operation. The delay from CLK0 or CLK1 to fast PLL 2 does not match the data input delay. The same arrangement applies to the right-side center fast PLL pair. For corner fast PLLs, only the corner FPLLCLK pins are fully compensated. For LVDS receiver operation, Altera recommends using the delay compensated clock pins only.

*Figure 5–45. Delay Compensated Clock Input Pins for Center Fast PLL Pair*
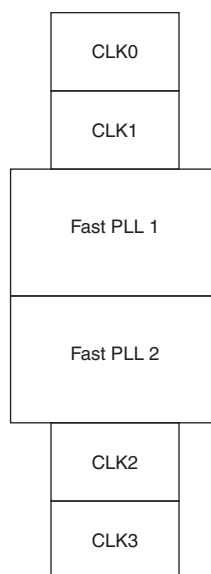


## Clock Output Connections

Enhanced PLLs have outputs for eight regional clock outputs and four global clock outputs. There is line sharing between clock pins, global and regional clock networks, and all PLL outputs. See Table 5–17 through Table 5–21 and Figure 5–46 through Figure 5–50 to validate your clocking

scheme. The Quartus II software automatically maps to regional and global clocks to avoid any restrictions. Enhanced PLLs 5, 6, 11, and 12 drive out to single-ended pins, as shown in Table 5–21.

You can connect each fast PLL 1, 2 output (C0, C1, C2, and C3) to either a global or regional clock. There is line sharing between clock pins, FPLLCLK pins, global and regional clock networks, and all PLL outputs. The Quartus II software automatically maps to regional and global clocks to avoid any restrictions.

Figure 5–46 shows clock input and output connections from the enhanced PLLs.

☞      EP1AGX20, EP1AGX35, and EX1AGX50 devices in the F484 package have only two enhanced PLLs (5 and 6), but the connectivity from these two PLLs to the global or regional clock networks remains the same.

EP1AGX50, EP1AGX60, and EP1AGX90 devices in the 1,152-pin package contain eight PLLs.

*Figure 5–46. Arria GX Top and Bottom Enhanced PLLs, Clock Pin, and Logic Array Signal Connectivity to Global and Regional Clock Networks Note (1)*
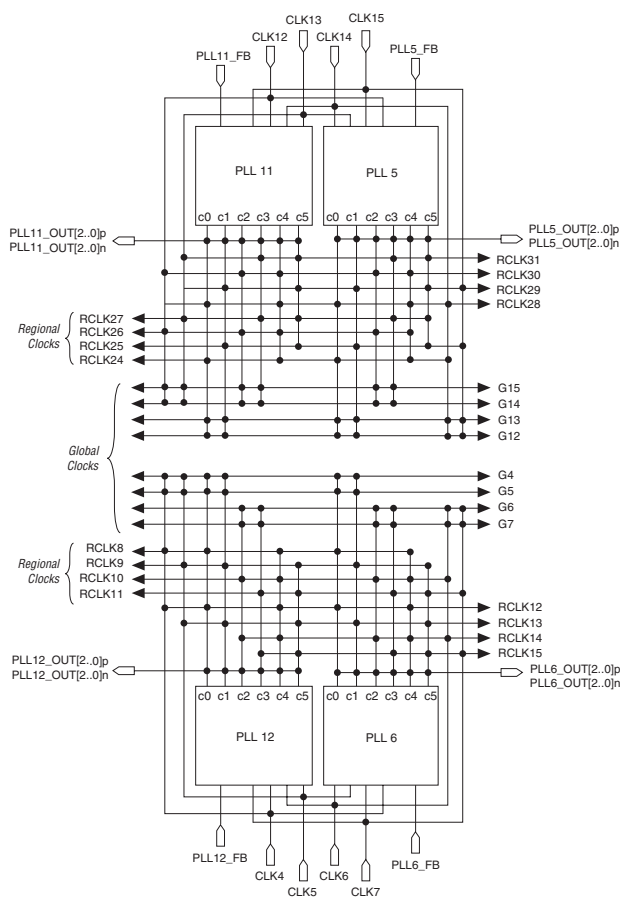


**Note to Figure 5–46:**

(1) Redundant connection dots facilitate stitching of the clock networks to support the ability to drive two quadrants with the same clock.

Table 5–21 shows the global and regional clocks that the PLL outputs drive.

| Table 5–21. Arria GX Global and Regional Clock Outputs From PLLs   (Part 1 of 2) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | All Devices (1) | | | | EP1AGX50 and Higher Devices (2) | | | |
| | Fast PLLs | | Enhanced PLLs | | Fast PLLs | | Enhanced PLLs | |
| Clock Network | PLL Number and Type | | | | | | | |
| | 1 | 2 | 5 | 6 | 7 | 8 | 11 | 12 |
| GCLK0 | ✓ | ✓ | — | — | ✓ | ✓ | — | — |
| GCLK1 | ✓ | ✓ | — | — | ✓ | ✓ | — | — |
| GCLK2 | ✓ | ✓ | — | — | ✓ | ✓ | — | — |
| GCLK3 | ✓ | ✓ | — | — | ✓ | ✓ | — | — |
| GCLK4 | — | — | — | ✓ | — | — | — | ✓ |
| GCLK5 | — | — | — | ✓ | — | — | — | ✓ |
| GCLK6 | — | — | — | ✓ | — | — | — | ✓ |
| GCLK7 | — | — | — | ✓ | — | — | — | ✓ |
| GCLK8 | — | — | — | — | — | — | — | — |
| GCLK9 | — | — | — | — | — | — | — | — |
| GCLK10 | — | — | — | — | — | — | — | — |
| GCLK11 | — | — | — | — | — | — | — | — |
| GCLK12 | — | — | ✓ | — | — | — | ✓ | — |
| GCLK13 | — | — | ✓ | — | — | — | ✓ | — |
| GCLK14 | — | — | ✓ | — | — | — | ✓ | — |
| GCLK15 | — | — | ✓ | — | — | — | ✓ | — |
| RCLK0 | ✓ | ✓ | — | — | ✓ | — | — | — |
| RCLK1 | ✓ | ✓ | — | — | ✓ | — | — | — |
| RCLK2 | ✓ | ✓ | — | — | ✓ | — | — | — |
| RCLK3 | ✓ | ✓ | — | — | ✓ | — | — | — |
| RCLK4 | ✓ | ✓ | — | — | — | ✓ | — | — |
| RCLK5 | ✓ | ✓ | — | — | — | ✓ | — | — |
| RCLK6 | ✓ | ✓ | — | — | — | ✓ | — | — |
| RCLK7 | ✓ | ✓ | — | — | — | ✓ | — | — |
| RCLK8 | — | — | — | ✓ | — | — | — | ✓ |
| RCLK9 | — | — | — | ✓ | — | — | — | ✓ |
| RCLK10 | — | — | — | ✓ | — | — | — | ✓ |

| Table 5–21. Arria GX Global and Regional Clock Outputs From PLLs   (Part 2 of 2) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | All Devices (1) | | | | EP1AGX50 and Higher Devices (2) | | | |
| | Fast PLLs | | Enhanced PLLs | | Fast PLLs | | Enhanced PLLs | |
| Clock Network | PLL Number and Type | | | | | | | |
| | 1 | 2 | 5 | 6 | 7 | 8 | 11 | 12 |
| RCLK11 | — | — | — | ✔ | — | — | — | ✔ |
| RCLK12 | — | — | — | ✔ | — | — | — | ✔ |
| RCLK13 | — | — | — | ✔ | — | — | — | ✔ |
| RCLK14 | — | — | — | ✔ | — | — | — | ✔ |
| RCLK15 | — | — | — | ✔ | — | — | — | ✔ |
| RCLK16 | — | — | — | — | — | — | — | — |
| RCLK17 | — | — | — | — | — | — | — | — |
| RCLK18 | — | — | — | — | — | — | — | — |
| RCLK19 | — | — | — | — | — | — | — | — |
| RCLK20 | — | — | — | — | — | — | — | — |
| RCLK21 | — | — | — | — | — | — | — | — |
| RCLK22 | — | — | — | — | — | — | — | — |
| RCLK23 | — | — | — | — | — | — | — | — |
| RCLK24 | — | — | ✔ | — | — | — | ✔ | — |
| RCLK25 | — | — | ✔ | — | — | — | ✔ | — |
| RCLK26 | — | — | ✔ | — | — | — | ✔ | — |
| RCLK27 | — | — | ✔ | — | — | — | ✔ | — |
| RCLK28 | — | — | ✔ | — | — | — | ✔ | — |
| RCLK29 | — | — | ✔ | — | — | — | ✔ | — |
| RCLK30 | — | — | ✔ | — | — | — | ✔ | — |
| RCLK31 | — | — | ✔ | — | — | — | ✔ | — |
| External Clock Output | | | | | | | | |
| PLL5_OUT[3..0]p/n | — | — | ✔ | — | — | — | — | — |
| PLL6_OUT[3..0]p/n | — | — | — | ✔ | — | — | — | — |
| PLL11_OUT[3..0]p/n | — | — | — | — | — | — | ✔ | — |
| PLL12_OUT[3..0]p/n | — | — | — | — | — | — | — | ✔ |

Notes to Table 5–21:
(1)   PLLs 3, 4, 9, and 10 are not available in Arria GX devices.
(2)   EP1AGX60 devices in 1,152-pin packages contain eight PLLs. EP1AGX60 devices in 484-pin and 780-pin packages contain fast PLLs 1 and 2, and enhanced PLLs 5, 6, 11, and 12.

Fast PLLs also drive high-speed SERDES clocks for differential I/O interfacing. For information about these FPLLCLK pins, contact Altera Applications Group.

Figure 5–48 shows the global and regional clock input and output connections from Arria GX fast PLLs.

*Figure 5–47. Arria GX Center Fast PLLs, Clock Pin, and Logic Array Signal Connectivity to Global and Regional Clock Networks*
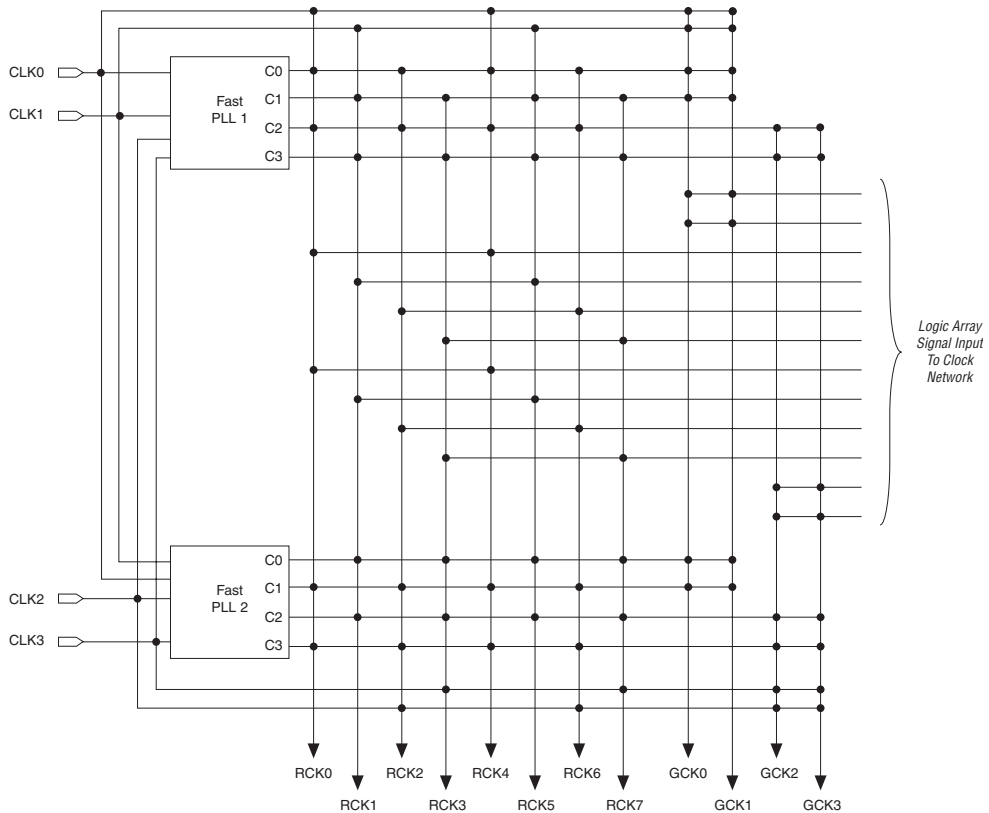


*Notes to Figure 5–47:*
(1)   Redundant connection dots facilitate stitching of the clock networks to support the ability to drive two quadrants with the same clock.

*Figure 5–48. Arria GX Corner Fast PLLs, Clock Pin, and Logic Array Signal Connectivity to Global and Regional Clock Networks Note (1)*
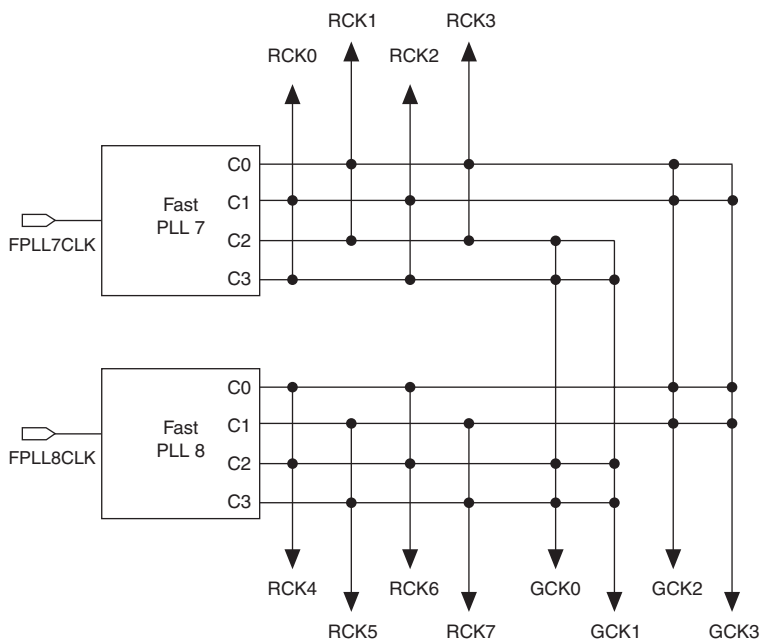


Note to *Figure 5–48*:
(1)  Corner fast PLLs can also be driven through the global or regional clock networks. Global or regional clock input to the fast PLL can be driven from another PLL or a pin-driven global or regional clock.

# Clock Control Block

Each global and regional clock has its own clock control block. The control block has two functions:

■  Clock source selection (dynamic selection for global clocks)
■  Clock power down (dynamic clock enable or disable)

Figures 5–49 and 5–50 show global clock- and regional clock-select blocks, respectively.

*Figure 5–49. Arria GX Global Clock Control Block*
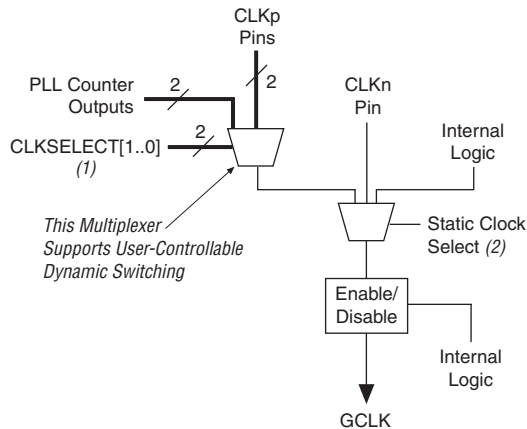


*Notes to Figure 5–49:*
(1)  These clock-select signals can only be dynamically controlled through internal logic when the device is operating in user mode.
(2)  These clock select signals can only be set through a configuration file and cannot be dynamically controlled during user-mode operation.

*Figure 5–50. Arria GX Regional Clock Control Block*
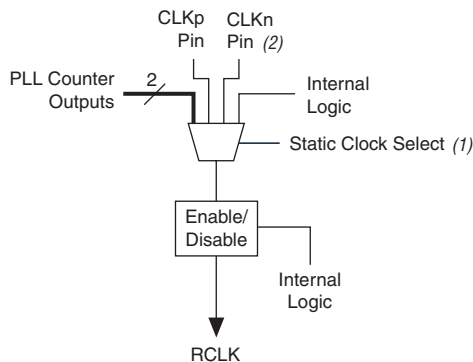


*Notes to Figure 5–50:*
(1)  These clock-select signals can only be dynamically controlled through a configuration file and cannot be dynamically controlled during user-mode operation.
(2)  Only the CLK pins on the top and bottom for the device feed to regional clock select blocks.

For the global clock select block, you can control the clock source selection either statically or dynamically. You have the option to statically select the clock source in the configuration file generated by the Quartus II software, or you can control the selection dynamically by using internal logic to drive the multiplexer select inputs. When selecting statically, you can set the clock source to any of the inputs to the select multiplexer. When selecting the clock source dynamically, you can either select two PLL outputs (such as CLK0 or CLK1), or a combination of clock pins or PLL outputs.

When using the ALTCLKCTRL megafunction to implement the clock source dynamic selection, the inputs from the clock pins feed the inclk[0..1] ports of the multiplexer, while the PLL outputs feed the inclk[2..3] ports. You can choose from among these inputs using the CLKSELECT[1..0] signal.

For the regional clock select block, you can only control the clock source selection statically using configuration bits. You can set any of the inputs to the clock select multiplexer the clock source.

You can disable (power down) Arria GX clock networks by both static and dynamic approaches. When a clock net is powered down, all the logic fed by the clock net is in an off-state, thereby reducing the overall power consumption of the device.

Global and regional clock networks that are not used are automatically powered down through configuration bit settings in the configuration file (SRAM Object File (**.sof**) or Programmer Object File (**.pof**)) generated by the Quartus II software.

The dynamic clock enable or disable feature allows the internal logic to control power up or down synchronously on GCLK and RCLK nets, including dual-regional clock regions. This function is independent of the PLL and is applied directly on the clock network, as shown in Figure 5–49 on page 78 and Figure 5–50 on page 5–78.

You can set the input clock sources and the clkena signals for the global and regional clock network multiplexers through the Quartus II software using the ALTCLKCTRL megafunction. You can also enable or disable the dedicated external clock output pins using the ALTCLKCTRL megafunction. Figure 5–51 shows the external PLL output clock control block.

*Figure 5–51. Arria GX External PLL Output Clock Control Block*
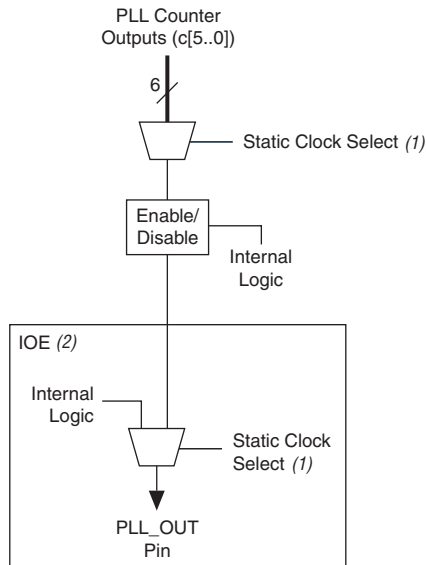


*Notes to Figure 5–51:*
(1)  Clock select signals can only be set through a configuration file and cannot be dynamically controlled during user mode operation.
(2)  The clock control block feeds to a multiplexer within the PLL_OUT pin's IOE. The PLL_OUT pin is a dual-purpose pin. Therefore, this multiplexer selects either an internal signal or the output of the clock control block.

## clkena Signals

Figure 5–52 shows how clkena is implemented.

*Figure 5–52. clkena Implementation*
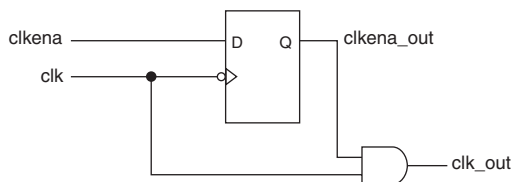


In Arria GX devices, the clkena signals are supported at the clock network level. This allows you to gate off the clock even when a PLL is not being used.

The clkena signals can also be used to control the dedicated external clocks from enhanced PLLs. Upon re-enabling, the PLL does not need a resynchronization or relock period unless the PLL is using external feedback mode. Figure 5–53 shows the waveform example for a clock output enable. clkena is synchronous to the falling edge of the counter output.

*Figure 5–53. clkena Signals*
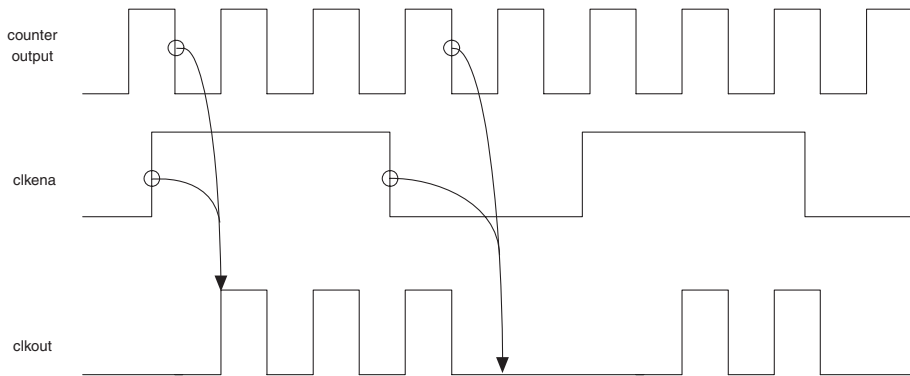


*Note to Figure 5–53*
(1)    You can use the clkena signals to enable or disable the global and regional networks or the PLL_OUT pins.

The PLL can remain locked independent of the clkena signals because the loop-related counters are not affected. This feature is useful for applications that require a low power or sleep mode. Upon re-enabling, the PLL does not need a resynchronization or relock period. The clkena signal can also disable clock outputs if the system is not tolerant to frequency overshoot during resynchronization.

## Conclusion

The Arria GX device's enhanced and fast PLLs provide you with complete control of device clocks and system timing. These PLLs are capable of offering flexible system-level clock management that was previously only available in discrete PLL devices. The embedded PLLs meet and exceed the features offered by these high-end discrete devices, reducing the need for other timing devices in the system.

## Referenced Documents

This chapter references the following documents:

■    *altpll Megafunction User Guide*
■    *AN 367: Implementing PLL Reconfiguration in Stratix II Devices*
■    *Configuring Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*

■ *DC & Switching Characteristics* chapter in volume 1 of the *Arria GX Device Handbook*
■ *Embedded Peripherals* section of the *Quartus II Handbook*
■ *Selectable I/O Standards in Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*

## Document Revision History

Table 5–22 shows the revision history for this chapter.

| Table 5–22. Document Revision History | | |
|---|---|---|
| **Date and Document Version** | **Changes Made** | **Summary of Changes** |
| May 2008 v1.2 | ● Updated note 3 from Table 5–1.<br>● Updated notes 2 and 3 from Figure 5–1. | — |
| | Minor text edits. | — |
| August 2007 v1.1 | Added the "Referenced Documents" section. | — |
| | Minor text edits. | — |
| May 2007 v1.0 | Initial Release | — |

# Section III. Memory

This section provides information on the TriMatrix™ embedded memory blocks internal to Arria™ GX devices and the supported external memory interfaces.

This section contains the following chapters:

- Chapter 6, TriMatrix Embedded Memory Blocks in Arria GX Devices

- Chapter 7, External Memory Interfaces in Arria GX Devices

## Revision History

Refer to each chapter for its own specific revision history. For information on when each chapter was updated, refer to the Chapter Revision Dates section, which appears in the full handbook.

# 6. TriMatrix Embedded Memory Blocks in Arria GX Devices

## Introduction

Arria™ GX devices feature the TriMatrix memory structure, consisting of three sizes of embedded RAM blocks that efficiently address the memory needs of FPGA designs.

TriMatrix memory includes 512-bit M512 blocks, 4-Kbit M4K blocks, and 512-Kbit M-RAM blocks, which are each configurable to support many features. TriMatrix memory provides up to 4,477,824 bits of RAM at up to 380 MHz operation.

This chapter contains the following sections:

## TriMatrix Memory Overview

TriMatrix architecture provides complex memory functions for different applications in FPGA designs. For example, M512 blocks are used for first-in first-out (FIFO) functions and clock domain buffering where memory bandwidth is critical; M4K blocks are ideal for applications requiring medium-sized memory, such as asynchronous transfer mode (ATM) cell processing; and M-RAM blocks are suitable for large buffering applications, such as internet protocol (IP) packet buffering and system cache.

TriMatrix memory blocks support various memory configurations, including single-port, simple dual-port, true dual-port (also known as bidirectional dual-port), shift register, and ROM modes. TriMatrix memory architecture also includes advanced features and capabilities, such as parity-bit support, byte enable support, pack mode support, address clock enable support, mixed port width support, and mixed clock mode support.

When applied to output registers, the asynchronous clear signal clears the output registers and the effects are seen immediately. Register clears are only supported on output registers.

Table 6–1 summarizes the features supported by the three sizes of TriMatrix memory.

*Table 6–1. Summary of TriMatrix Memory Features*

| Feature | M512 Blocks | M4K Blocks | M-RAM Blocks |
|---|---|---|---|
| Maximum performance | 345 MHz | 380 MHz | 290 MHz |
| Total RAM bits (including parity bits) | 576 | 4,608 | 589,824 |
| Configurations | 512 × 1<br>256 × 2<br>128 × 4<br>64 × 8<br>64 × 9<br>32 × 16<br>32 × 18 | 4K × 1<br>2K × 2<br>1K × 4<br>512 × 8<br>512 × 9<br>256 × 16<br>256 × 18<br>128 × 32<br>128 × 36 | 64K × 8<br>64K × 9<br>32K × 16<br>32K × 18<br>16K × 32<br>8K × 64<br>8K × 72<br>4K × 128<br>4K × 144 |
| Parity bits | ✓ | ✓ | ✓ |
| Byte enable | ✓ | ✓ | ✓ |
| Pack mode | | ✓ | ✓ |
| Address clock enable | | ✓ | ✓ |
| Single-port memory | ✓ | ✓ | ✓ |
| Simple dual-port memory | ✓ | ✓ | ✓ |
| True dual-port memory | | ✓ | ✓ |
| Embedded shift register | ✓ | ✓ | |
| ROM | ✓ | ✓ | |
| FIFO buffer | ✓ | ✓ | ✓ |
| Simple dual-port mixed width support | ✓ | ✓ | ✓ |
| True dual-port mixed width support | | ✓ | ✓ |
| Memory initialization file (.**mif**) | ✓ | ✓ | |
| Mixed-clock mode | ✓ | ✓ | ✓ |
| Power-up condition | Outputs cleared | Outputs cleared | Outputs unknown |
| Register clears | Output registers only | Output registers only | Output registers only |
| Same-port read-during-write | New data available at positive clock edge | New data available at positive clock edge | New data available at positive clock edge |
| Mixed-port read-during-write | Outputs set to unknown or old data | Outputs set to unknown or old data | Unknown output |

Table 6–2 shows the capacity of the TriMatrix memory blocks for each device in the Arria GX family.

| Device | M512 Blocks | M4K Blocks | M-RAM Blocks | Total RAM Bits |
|--------|-------------|------------|--------------|----------------|
| EP1AGX20 | 166 | 118 | 1 | 1,229,184 |
| EP1AGX35 | 197 | 140 | 1 | 1,348,416 |
| EP1AGX50 | 313 | 242 | 2 | 2,475,072 |
| EP1AGX60 | 326 | 252 | 2 | 2,528,640 |
| EP1AGX90 | 478 | 400 | 4 | 4,477,824 |

*Table 6–2. TriMatrix Memory Capacity in Arria GX Devices*

## Parity Bit Support

All TriMatrix memory blocks (M512, M4K, and M-RAM) support one parity bit for each byte.

Parity bits add to the amount of memory in each RAM block. For example, the M512 block has 576 bits, 64 of which are optionally used for parity bit storage. The parity bit, along with logic implemented in adaptive logic modules (ALMs), implements parity checking for error detection to ensure data integrity. Parity-size data words can also be used for other purposes such as storing user-specified control bits.

## Byte Enable Support

All TriMatrix memory blocks support byte enables that mask the input data so that only specific bytes, nibbles, or bits of data are written. The unwritten bytes or bits retain the previous written value. The write enable (`wren`) signals, along with the byte enable (`byteena`) signals, control the RAM blocks' write operations. The default value for the byte enable signals is high (enabled), in which case writing is controlled only by the write enable signals. There is no clear port to the byte enable registers.

### M512 Blocks

M512 blocks support byte enables for data widths of 16 and 18 bits only. The byte-enable feature for memory block configurations with widths of less than two bytes (×16/×18) is not supported. For memory configurations less than two bytes wide, the write enable or clock enable signals can optionally be used to control the write operation.

Table 6–3 summarizes the byte selection.

| Table 6–3. Byte Enable for Arria GX M512 Blocks *Note (1)* | | |
|---|---|---|
| **byteena[1..0]** | **data ×16** | **data ×18** |
| [0] = 1 | `[7..0]` | `[8..0]` |
| [1] = 1 | `[15..8]` | `[17..9]` |

*Note to Table 6–3:*
(1)    Any combination of byte enables is possible.

### M4K Blocks

M4K blocks support byte enables for any combination of data widths of 16, 18, 32, and 36 bits only. The byte-enable feature for memory block configurations with widths of less than two bytes (×16/×18) is not supported. For memory configurations less than two bytes wide, the write enable or clock enable signals can optionally be used to control the write operation. Table 6–4 summarizes the byte selection.

| Table 6–4. Byte Enable for Arria GX M4K Blocks *Note (1)* | | | | |
|---|---|---|---|---|
| **byteena[3..0]** | **data ×16** | **data ×18** | **data ×32** | **data ×36** |
| [0] = 1 | `[7..0]` | `[8..0]` | `[7..0]` | `[8..0]` |
| [1] = 1 | `[15..8]` | `[17..9]` | `[15..8]` | `[17..9]` |
| [2] = 1 | - | - | `[23..16]` | `[26..18]` |
| [3] = 1 | - | - | `[31..24]` | `[35..27]` |

*Note to Table 6–4:*
(1)    Any combination of byte enables is possible.

### M-RAM Blocks

M-RAM blocks support byte enables for any combination of data widths of 16, 18, 32, 36, 64, and 72 bits. The byte-enable feature for memory block configurations with widths of less than two bytes (×16/×18) is not supported. In the ×128 and ×144 simple dual-port modes, the two sets of byte enable signals (`byteena_a` and `byteena_b`) combine to form the necessary 16 byte enables. In ×128 and ×144 modes, byte enables are only supported when using single clock mode. However, the Quartus® II software can implement byte enable in other clocking modes for ×128 or ×144 widths but will use twice as many M-RAM resources.

If clock enables are used in ×128 or ×144 mode, you must use the same clock enable setting for both A and B ports. Table 6–5 summarizes the byte selection for M-RAM blocks.

| byteena | data ×16 | data ×18 | data ×32 | data ×36 | data ×64 | data ×72 |
|---------|----------|----------|----------|----------|----------|----------|
| [0] = 1 | [7..0] | [8..0] | [7..0] | [8..0] | [7..0] | [8..0] |
| [1] = 1 | [15..8] | [17..9] | [15..8] | [17..9] | [15..8] | [17..9] |
| [2] = 1 | - | - | [23..16] | [26..18] | [23..16] | [26..18] |
| [3] = 1 | - | - | [31..24] | [35..27] | [31..24] | [35..27] |
| [4] = 1 | - | - | - | - | [39..32] | [44..36] |
| [5] = 1 | - | - | - | - | [47..40] | [53..45] |
| [6] = 1 | - | - | - | - | [55..48] | [62..54] |
| [7] = 1 | - | - | - | - | [63..56] | [71..63] |

*Table 6–5. Byte Enable for Arria GX M-RAM Blocks  Note (1)*

*Note to Table 6–5:*
(1)  Any combination of byte enables is possible.

Table 6–6 summarizes the byte selection for ×144 mode.

| byteena | data ×128 | data ×144 |
|---------|-----------|-----------|
| [0] = 1 | [7..0] | [8..0] |
| [1] = 1 | [15..8] | [17..9] |
| [2] = 1 | [23..16] | [26..18] |
| [3] = 1 | [31..24] | [35..27] |
| [4] = 1 | [39..32] | [44..36] |
| [5] = 1 | [47..40] | [53..45] |
| [6] = 1 | [55..48] | [62..54] |
| [7] = 1 | [63..56] | [71..63] |
| [8] = 1 | [71..64] | [80..72] |
| [9] = 1 | [79..72] | [89..73] |
| [10] = 1 | [87..80] | [98..90] |
| [11] = 1 | [95..88] | [107..99] |
| [12] = 1 | [103..96] | [116..108] |
| [13] = 1 | [111..104] | [125..117] |

*Table 6–6. Arria GX M-RAM Combined Byte for ×144 Mode   (Part 1 of 2) Note (1)*

*Table 6–6. Arria GX M-RAM Combined Byte for ×144 Mode   (Part 2 of 2)*
*Note (1)*

| byteena | data ×128 | data ×144 |
|---------|-----------|-----------|
| [14] = 1 | [119..112] | [134..126] |
| [15] = 1 | [127..120] | [143..135] |

*Note to Table 6–6:*
(1)   Any combination of byte enables is possible.

### Byte Enable Functional Waveform

Figure 6–1 shows how the write enable (wren) and byte enable (byteena) signals control the operations of the RAM.

When a byte enable bit is de-asserted during a write cycle, the corresponding data byte output appears as a "don't care" or unknown value. When a byte enable bit is asserted during a write cycle, the corresponding data byte output will be the newly written data.

*Figure 6–1. Arria GX Byte Enable Functional Waveform*
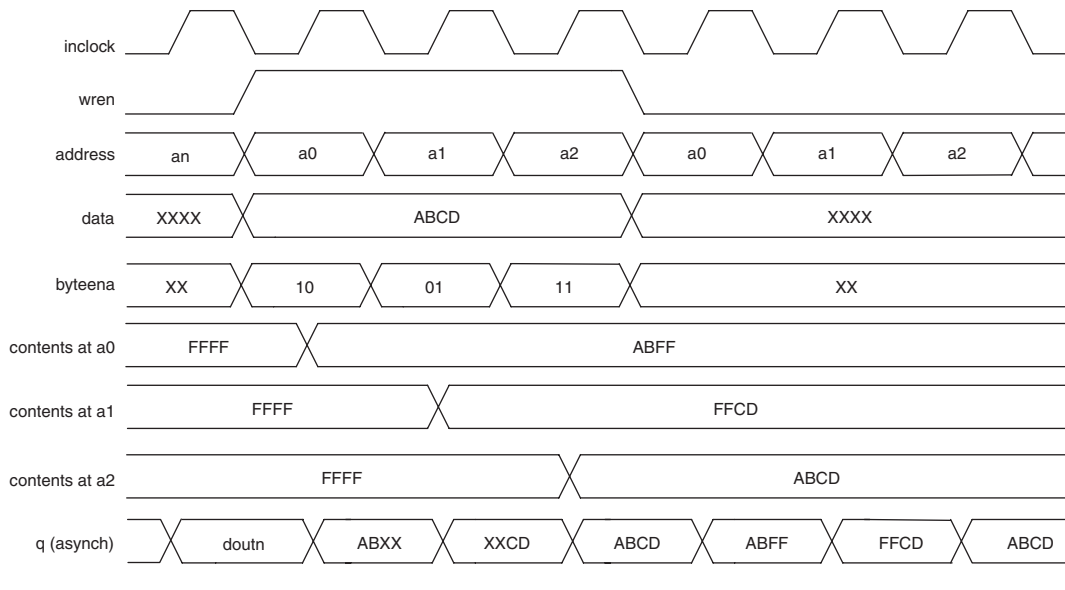
## Pack Mode Support

Arria GX M4K and M-RAM memory blocks support pack mode. In M4K and M-RAM memory blocks, you can implement two single-port memory blocks in a single block under the following conditions:

■ Each of the two independent block sizes is equal to or less than half of the M4K or M-RAM block size.
■ Each of the single-port memory blocks is configured in single-clock mode.

Thus, each of the single-port memory blocks access up to half of the M4K or M-RAM memory resources such as clock, clock enables, and asynchronous clear signals.

☞ For more information, refer to "Single-Port Mode" on page 6–10 and "Single-Clock Mode" on page 6–27.

## Address Clock Enable Support

Arria GX M4K and M-RAM memory blocks support address clock enable, which is used to hold the previous address value for as long as the signal is enabled. When the memory blocks are configured in dual-port mode, each port has its own independent address clock enable.

Figure 6–2 shows an address clock enable block diagram. Placed in the address register, the address signal output by the address register is fed back to the input of the register via a multiplexer. The multiplexer output is selected by the address clock enable (addressstall) signal. Address latching is enabled when the addressstall signal turns high. The output of the address register is then continuously fed into the input of the register; therefore, the address value can be held until the addressstall signal turns low.

*Figure 6–2. Arria GX Address Clock Enable Block Diagram*



Address clock enable is typically used for cache memory applications, which require one port for read and another port for write. The default value for the address clock enable signals is low (disabled). Figures 6–3 and 6–4 show the address clock enable waveform during read and write cycles, respectively.

*Figure 6–3. Arria GX Address Clock Enable During Read Cycle Waveform*

*Figure 6–4. Arria GX Address Clock Enable During Write Cycle Waveform*
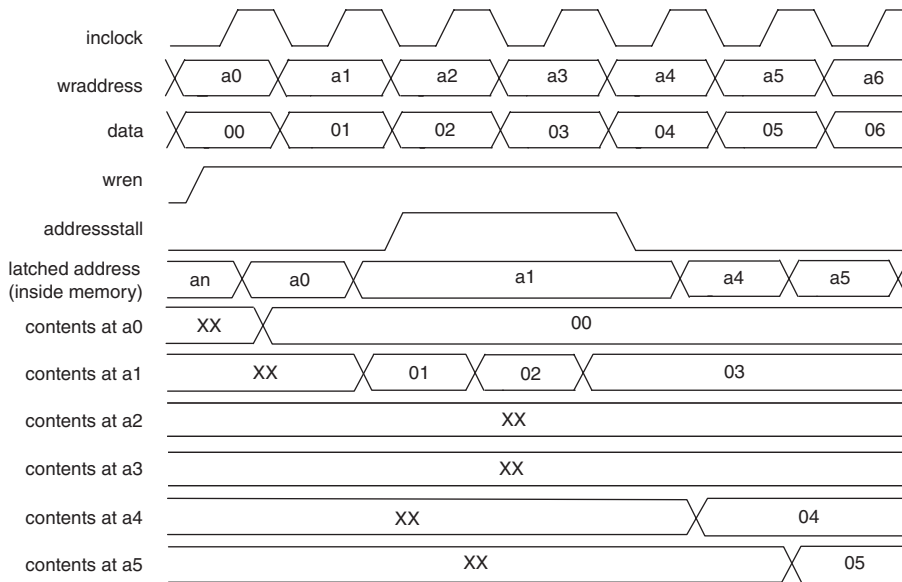


## Memory Modes

Arria GX TriMatrix memory blocks include input registers that synchronize writes and output registers to pipeline data to improve system performance. All TriMatrix memory blocks are fully synchronous, meaning that all inputs are registered, but outputs can be either registered or unregistered.

☞ TriMatrix memory does not support asynchronous memory (unregistered inputs).

Depending on which TriMatrix memory block you use, the memory has various modes, including:

■ Single-port
■ Simple dual-port
■ True dual-port (bidirectional dual-port)
■ Shift-register
■ ROM
■ FIFO

☞ Violating the setup or hold time on the memory block address registers could corrupt memory contents. This applies to both read and write operations.

## Single-Port Mode

All TriMatrix memory blocks support single-port mode that supports non-simultaneous read and write operations. Figure 6–5 shows the single-port memory configuration for TriMatrix memory.

*Figure 6–5. Single-Port Memory Note (1)*
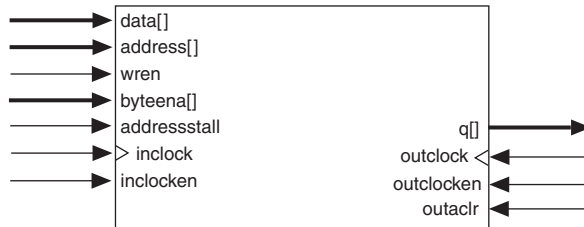


*Note to Figure 6–5:*
(1)  Two single-port memory blocks can be implemented in a single M4K or M-RAM block.

M4K and M-RAM memory blocks can also be halved and used for two independent single-port RAM blocks. The Altera® Quartus II software automatically uses this single-port memory packing when running low on memory resources. To force two single-port memories into one M4K or M-RAM block, first ensure that each of the two independent RAM blocks is equal to or less than half the size of the M4K or M-RAM block. Second, assign both single-port RAMs to the same M4K or M-RAM block.

In single-port RAM configuration, the outputs can only be in read-during-write mode, which means that during the write operation, data written to the RAM flows through to the RAM outputs. When the output registers are bypassed, the new data is available on the rising edge of the same clock cycle on which it was written. For more information about read-during-write mode, refer to "Read-During-Write Operation at the Same Address" on page 6–32.

Table 6–7 shows the port width configurations for TriMatrix blocks in single-port mode.

*Table 6–7. Arria GX Port Width Configurations for M512, M4K, and M-RAM Blocks (Single-Port Mode)*

|  | M512 Blocks | M4K Blocks | M-RAM Blocks |
|---|---|---|---|
| Port Width Configurations | 512 × 1<br>256 × 2<br>128 × 4<br>64 × 8<br>64 × 9<br>32 × 16<br>32 × 18 | 4K × 1<br>2K × 2<br>1K × 4<br>512 × 8<br>512 × 9<br>256 × 16<br>256 × 18<br>128 × 32<br>128 × 36 | 64K × 8<br>64K × 9<br>32K × 16<br>32K × 18<br>16K × 32<br>16K × 36<br>8K × 64<br>8K × 72<br>4K × 128<br>4K × 144 |

Figure 6–6 shows timing waveforms for read and write operations in single-port mode.

*Figure 6–6. Arria GX Single-Port Timing Waveforms Note (1)*
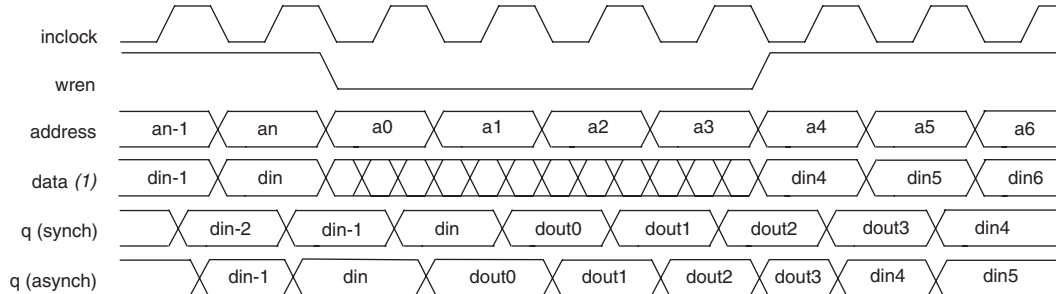


*Note to Figure 6–6:*
(1)    The crosses in the data waveform during read mean "don't care."

## Simple Dual-Port Mode

All TriMatrix memory blocks support simple dual-port mode which supports a simultaneous read and write operation. Figure 6–7 shows the simple dual-port memory configuration for TriMatrix memory.

*Figure 6–7. Arria GX Simple Dual-Port Memory Note (1)*
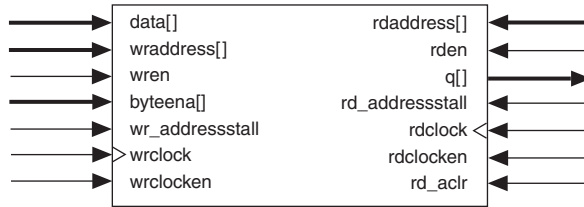


*Note to Figure 6–7:*
(1) Simple dual-port RAM supports input and output clock mode in addition to the read and write clock mode shown.

TriMatrix memory supports mixed-width configurations, allowing different read and write port widths. Tables 6–8 through 6–10 show mixed-width configurations for M512, M4K, and M-RAM blocks, respectively.

*Table 6–8. Arria GX M512 Block Mixed-Width Configurations (Simple Dual-Port Mode)*

| Read Port | Write Port | | | | | | |
|---|---|---|---|---|---|---|---|
| | 512 × 1 | 256 × 2 | 128 × 4 | 64 × 8 | 32 × 16 | 64 × 9 | 32 × 18 |
| 512 × 1 | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| 256 × 2 | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| 128 × 4 | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| 64 × 8 | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| 32 × 16 | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| 64 × 9 | | | | | | ✓ | ✓ |
| 32 × 18 | | | | | | ✓ | ✓ |

*Table 6–9. Arria GX M4K Block Mixed-Width Configurations (Simple Dual-Port Mode)   (Part 1 of 2)*

| Read Port | Write Port | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 4K × 1 | 2K × 2 | 1K × 4 | 512 × 8 | 256 × 16 | 128 × 32 | 512 × 9 | 256 × 18 | 128 × 36 |
| 4K × 1 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | |
| 2K × 2 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | |

*Table 6–9. Arria GX M4K Block Mixed-Width Configurations (Simple Dual-Port Mode)  (Part 2 of 2)*

| Read Port | Write Port | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 4K × 1 | 2K × 2 | 1K × 4 | 512 × 8 | 256 × 16 | 128 × 32 | 512 × 9 | 256 × 18 | 128 × 36 |
| 1K × 4 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | |
| 512 × 8 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | |
| 256 × 16 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | |
| 128 × 32 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | |
| 512 × 9 | | | | | | | ✓ | ✓ | ✓ |
| 256 × 18 | | | | | | | ✓ | ✓ | ✓ |
| 128 × 36 | | | | | | | ✓ | ✓ | ✓ |

*Table 6–10. Arria GX M-RAM Block Mixed-Width Configurations (Simple Dual-Port Mode)*

| Read Port | Write Port | | | | |
|---|---|---|---|---|---|
| | 64K × 9 | 32K × 18 | 18K × 36 | 8K × 72 | 4K × 144 |
| 64K × 9 | ✓ | ✓ | ✓ | ✓ | |
| 32K × 18 | ✓ | ✓ | ✓ | ✓ | |
| 18K × 36 | ✓ | ✓ | ✓ | ✓ | |
| 8K × 72 | ✓ | ✓ | ✓ | ✓ | |
| 4K × 144 | | | | | ✓ |

In simple dual-port mode, M512 and M4K blocks have one write enable and one read enable signal. However, M-RAM blocks contain only a write enable signal, which is held high to perform a write operation. M-RAM blocks are always enabled for read operations. The Quartus II software can emulate a read-enable signal for M-RAM blocks by using the clock-enable signal if it is not already used. If the read address and write address select the same address location during a write operation, M-RAM block output is unknown.

TriMatrix memory blocks do not support a clear port on the write enable and read enable registers. When the read enable is deactivated, the current data is retained at the output ports. If the read enable is activated during a write operation with the same address location selected, the

simple dual-port RAM output is either unknown or can be set to output the old data stored at the memory address. For more information, refer to "Read-During-Write Operation at the Same Address" on page 6–32.

Figure 6–8 shows timing waveforms for read and write operations in simple dual-port mode.

*Figure 6–8. Arria GX Simple Dual-Port Timing Waveforms Notes (1), (2)*
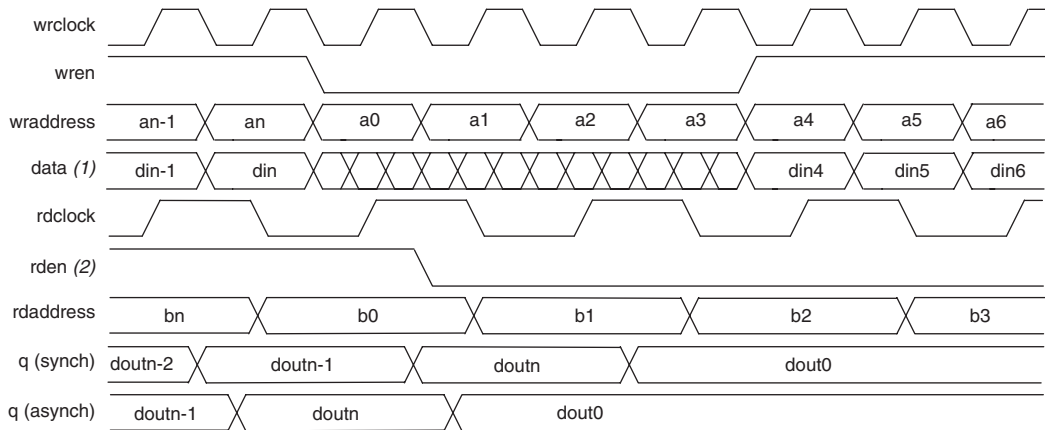
*Notes to Figure 6–8:*
(1)   The crosses in the data waveform during read mean "don't care."
(2)   The read enable rden signal is not available in M-RAM blocks. The M-RAM block in simple dual-port mode always reads out the data stored at the current read address location.

## True Dual-Port Mode

Arria GX M4K and M-RAM memory blocks support true dual-port mode. True dual-port mode supports any combination of two-port operations: two reads, two writes, or one read and one write at two different clock frequencies. Figure 6–9 shows Arria GX true dual-port memory configuration.

*Figure 6–9. Arria GX True Dual-Port Memory Note (1)*
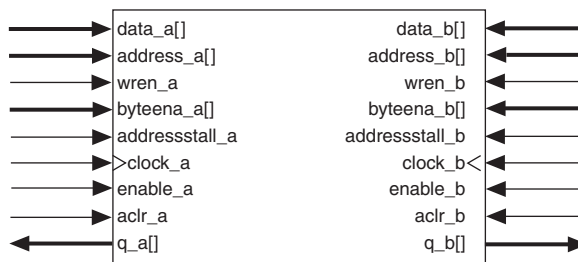


*Note to Figure 6–9:*
(1)    True dual-port memory supports input/output clock mode in addition to the
       independent clock mode shown.

The widest bit configuration of the M4K and M-RAM blocks in true
dual-port mode is as follows:

■    256 × 16-bit (×18-bit with parity) (M4K)
■    8K × 64-bit (×72-bit with parity) (M-RAM)

The 128 × 32-bit (×36-bit with parity) configuration of the M4K block and
the 4K × 128-bit (×144-bit with parity) configuration of the M-RAM block
are unavailable because the number of output drivers is equivalent to the
maximum bit width of the respective memory block. Because true
dual-port RAM has outputs on two ports, the maximum width of the true
dual-port RAM equals half of the total number of output drivers.
Table 6–11 lists the possible M4K block mixed-port width configurations.

*Table 6–11. Arria GX M4K Block Mixed-Port Width Configurations (True Dual-Port)*

| Read Port | Write Port | | | | | | |
|---|---|---|---|---|---|---|---|
| | 4K × 1 | 2K × 2 | 1K × 4 | 512 × 8 | 256 × 16 | 512 × 9 | 256 × 18 |
| 4K × 1 | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| 2K × 2 | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| 1K × 4 | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| 512 × 8 | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| 256 × 16 | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| 512 × 9 | | | | | | ✓ | ✓ |
| 256 × 18 | | | | | | ✓ | ✓ |

Table 6–12 lists the possible M-RAM block mixed-port width configurations.

**Table 6–12. Arria GX M-RAM Block Mixed-Port Width Configurations (True Dual-Port)**

| Read Port | Write Port | | | |
|---|---|---|---|---|
| | 64K × 9 | 32K × 18 | 18K × 36 | 8K × 72 |
| 64K × 9 | ✓ | ✓ | ✓ | ✓ |
| 32K × 18 | ✓ | ✓ | ✓ | ✓ |
| 18K × 36 | ✓ | ✓ | ✓ | ✓ |
| 8K × 72 | ✓ | ✓ | ✓ | ✓ |

In true dual-port configuration, the RAM outputs can only be configured for read-during-write mode. This means that during write operation, data being written to the A or B port of the RAM flows through to the A or B outputs, respectively. When the output registers are bypassed, the new data is available on the rising edge of the same clock cycle on which it was written. For waveforms and information on mixed-port read-during-write mode, refer to "Read-During-Write Operation at the Same Address" on page 6–32.

Potential write contentions must be resolved external to the RAM because writing to the same address location at both ports results in unknown data storage at that location. For a valid write operation to the same address of the M-RAM block, the rising edge of the write clock for port A must occur following the maximum write cycle time interval after the rising edge of the write clock for port B. Data is written on the rising edge of the write clock for the M-RAM block.

Because data is written into the M512 and M4K blocks at the falling edge of the write clock, the rising edge of the write clock for port A should occur following half of the maximum write cycle time interval after the falling edge of the write clock for port B. If this timing is not met, the data stored in that particular address will be invalid.

☞ For the maximum synchronous write cycle time, refer to the *Arria GX Device Family Data Sheet* in volume 1 of the *Arria GX Device Handbook*.

Figure 6–10 shows true dual-port timing waveforms for the write operation at port A and the read operation at port B.

*Figure 6–10. Arria GX True Dual-Port Timing Waveforms Note (1)*
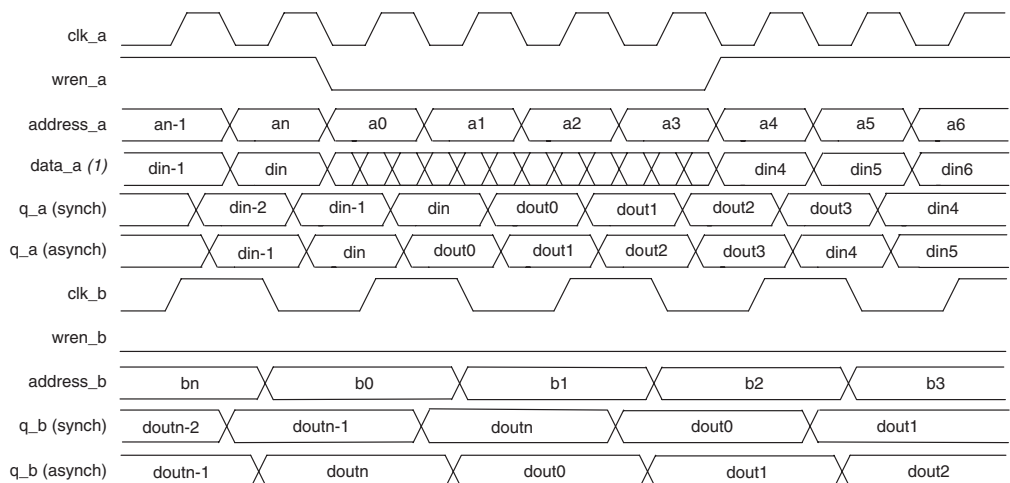


*Note to Figure 6–10:*
(1) The crosses in the `data_a` waveform during write mean "don't care."

## Shift-Register Mode

All Arria GX memory blocks support shift-register mode.

Embedded memory block configurations can implement shift registers for digital signal processing (DSP) applications, such as finite impulse response (FIR) filters, pseudo-random number generators, multi-channel filtering, and auto-correlation and cross-correlation functions. These and other DSP applications require local data storage, traditionally implemented with standard flip flops that quickly exhaust many logic cells for large shift registers. A more efficient alternative is to use embedded memory as a shift-register block, which saves logic cell and routing resources.

The size of a (w m n) shift register is determined by the input data width (w), the length of the taps (m), and the number of taps (n), and must be less than or equal to the maximum number of memory bits in the respective block: 576 bits for the M512 block, 4,608 bits for the M4K block, and 589,824 bits for the MRAM block. In addition, the size of w n must be less than or equal to the maximum width of the respective block: 18 bits

for the M512 block, 36 bits for the M4K block, and 144 bits for the MRAM block. If a larger shift register is required, the memory blocks can be cascaded.

In M512 and M4K blocks, data is written into each address location at the falling edge of the clock and read from the address at the rising edge of the clock. Shift-register mode logic automatically controls the positive and negative edge clocking to shift the data in one clock cycle. The MRAM block performs reads and writes on the rising edge. Figure 6–11 shows the TriMatrix memory block in the shift-register mode.

*Figure 6–11. Arria GX Shift-Register Memory Configuration*
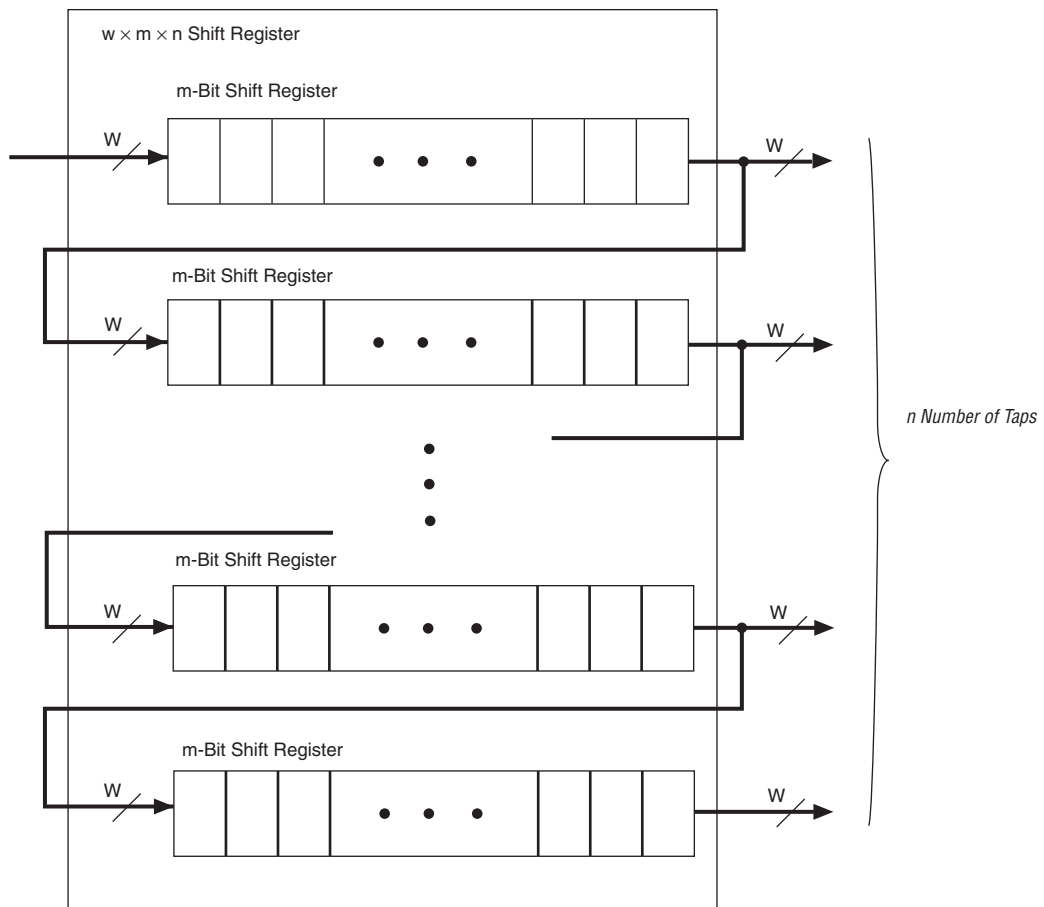
### ROM Mode

M512 and M4K memory blocks support ROM mode. A memory initialization file (**.mif**) initializes the ROM contents of these blocks. The address lines of the ROM are registered. The outputs can be registered or unregistered. The ROM read operation is identical to the read operation in the single-port RAM configuration.

### FIFO Buffers Mode

TriMatrix memory blocks support FIFO mode. M512 memory blocks are ideal for designs with many shallow FIFO buffers. All memory configurations have synchronous inputs; however, the FIFO buffer outputs are always combinational. Simultaneous read and write from an empty FIFO buffer is not supported.

For more information about FIFO buffers, refer to the *Single- and Dual-Clock FIFO Megafunctions User Guide* and *FIFO Partitioner Megafunction User Guide*.

## Clock Modes

Depending on which TriMatrix memory mode you select, the following clock modes are available:

■ Independent
■ Input and output
■ Read and write
■ Single-clock

Table 6–13 shows these clock modes supported by all TriMatrix blocks when configured as respective memory modes.

| *Table 6–13. Arria GX TriMatrix Memory Clock Modes* | | | |
|---|---|---|---|
| **Clocking Modes** | **True Dual-Port Mode** | **Simple Dual-Port Mode** | **Single-Port Mode** |
| Independent | ✓ | | |
| Input/output | ✓ | ✓ | ✓ |
| Read/write | | ✓ | |
| Single clock | ✓ | ✓ | ✓ |

## Independent Clock Mode

TriMatrix memory blocks can implement independent clock mode for true dual-port memory. In this mode, a separate clock is available for each port (A and B). Clock A controls all registers on the port A side, while clock B controls all registers on the port B side. Each port also supports independent clock enables for port A and B registers. Asynchronous clear signals for the registers; however, are not supported.

Figure 6–12 shows a TriMatrix memory block in independent clock mode.

*Figure 6–12. Arria GX TriMatrix Memory Block in Independent Clock Mode*
*Note (1)*
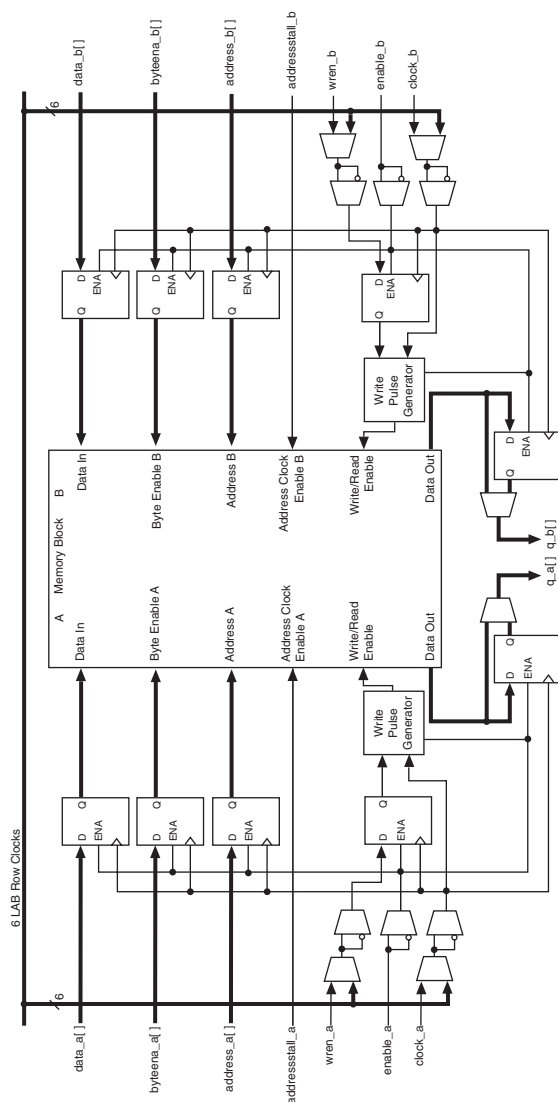


*Note to Figure 6–12:*
(1)    Violating the setup or hold time on the memory block address registers could corrupt the memory contents. This applies to both read and write operations.

## Input and Output Clock Mode

Arria GX TriMatrix memory blocks can implement input and output clock mode for true and simple dual-port memory. On each of the two ports, A and B, one clock controls all registers for the following inputs into the memory block: data input, write enable, and address. The other clock controls the blocks' data output registers. Each memory block port also supports independent clock enables for input and output registers. Asynchronous clear signals for the registers; however, are not supported.

Figures 6–13 through 6–15 show the memory block in input and output clock mode for true dual-port, simple dual-port, and single-port modes, respectively.

*Figure 6–13. Arria GX Input/Output Clock Mode in True Dual-Port Mode*
*Note (1)*
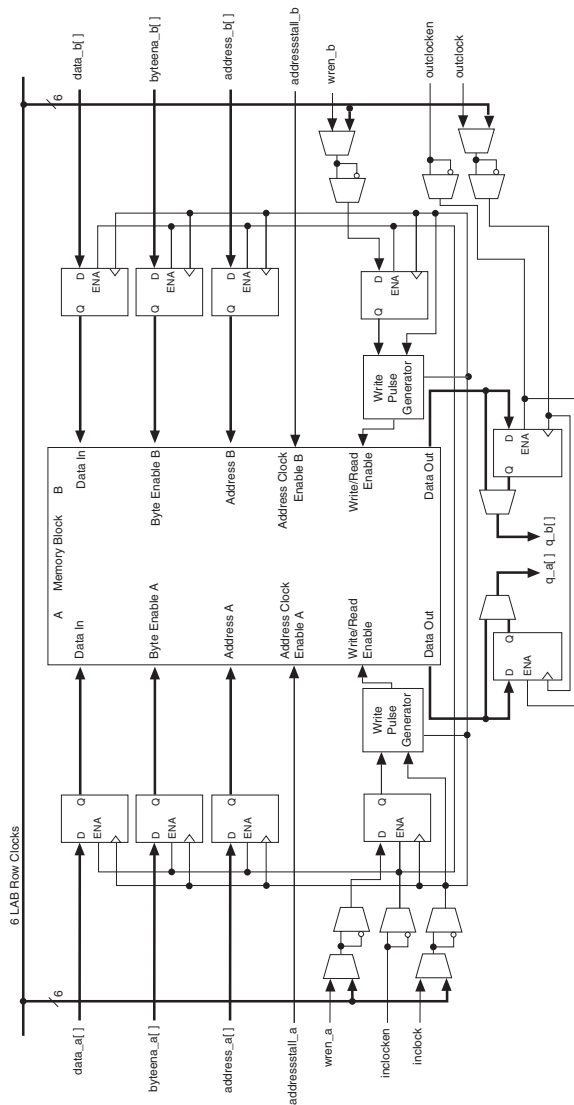


*Note to Figure 6–13:*
(1)     Violating the setup or hold time on the memory block address registers could corrupt the memory contents. This applies to both read and write operations.

*Figure 6–14. Arria GX Input/Output Clock Mode in Simple Dual-Port Mode Notes (1), (2), (3)*
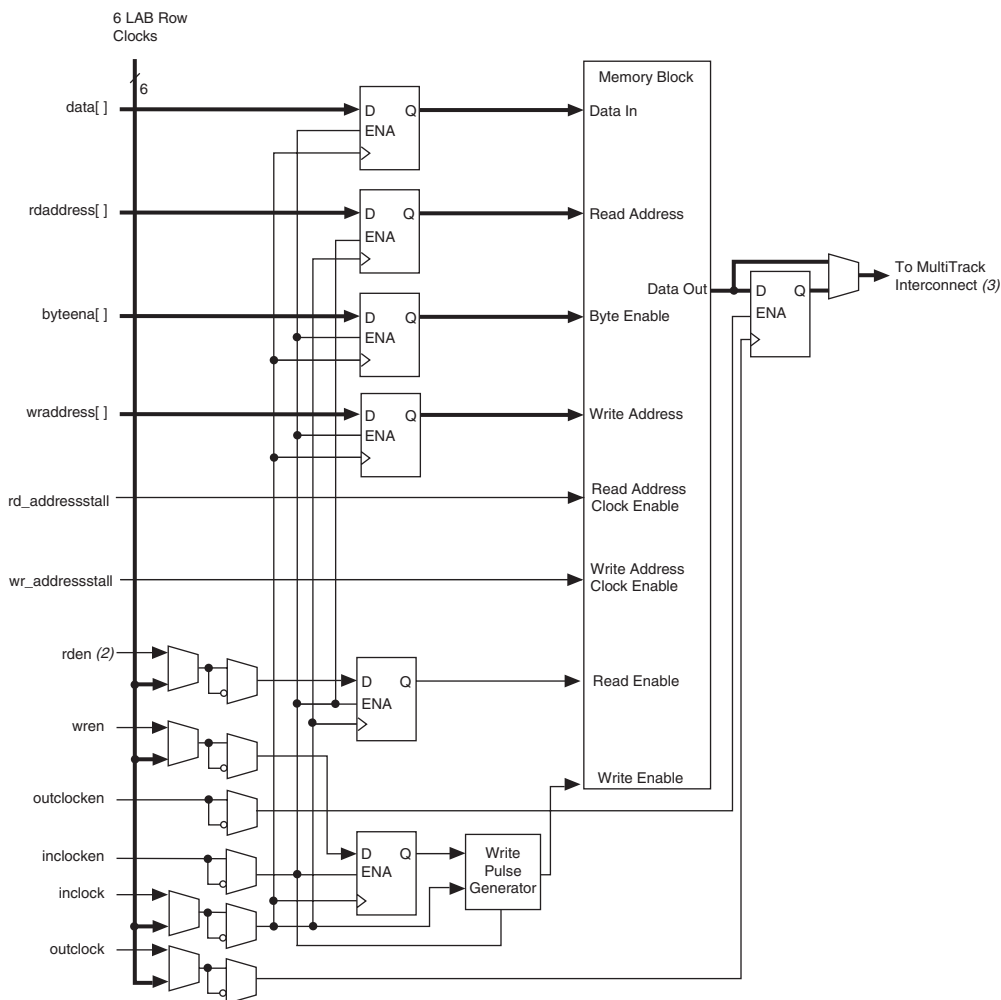


Notes to Figure 6–14:
(1)  Violating the setup or hold time on the memory block address registers could corrupt the memory contents. This applies to both read and write operations.
(2)  The read enable rden signal is not available in the M-RAM block. An M-RAM block in simple dual-port mode is always reading out the data stored at the current read address location.
(3)  For more information about the MultiTrack interconnect, refer to the *Arria GX Device Family Data Sheet* in volume 1 of the *Arria GX Device Handbook*.

*Figure 6–15. Arria GX Input/Output Clock Mode in Single-Port Mode Notes (1), (2)*
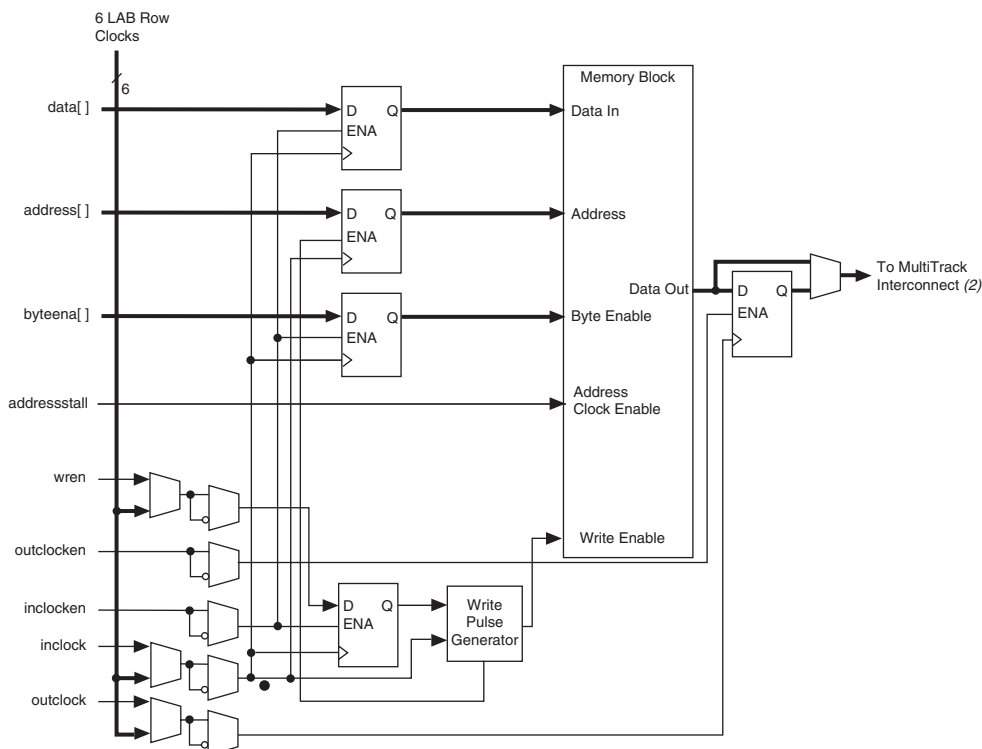


**Notes to Figure 6–15:**
(1) Violating the setup or hold time on the memory block address registers could corrupt the memory contents. This applies to both read and write operations.
(2) For more information about the MultiTrack interconnect, refer to the *Arria GX Device Family Data Sheet* in volume 1 of the *Arria GX Device Handbook*.

## Read and Write Clock Mode

Arria GX TriMatrix memory blocks can implement read and write clock mode for simple dual-port memory. This mode uses up to two clocks. The write clock controls the blocks' data inputs, write address, and write enable signals. The read clock controls the data output, read address, and read-enable signals. The memory blocks support independent clock enables for each clock for the read- and write-side registers. However, asynchronous clear signals for the registers are not supported.
Figure 6–16 shows a memory block in read and write clock mode.

*Figure 6–16. ArriaGX Read and Write Clock Mode Notes (1), (2), (3)*
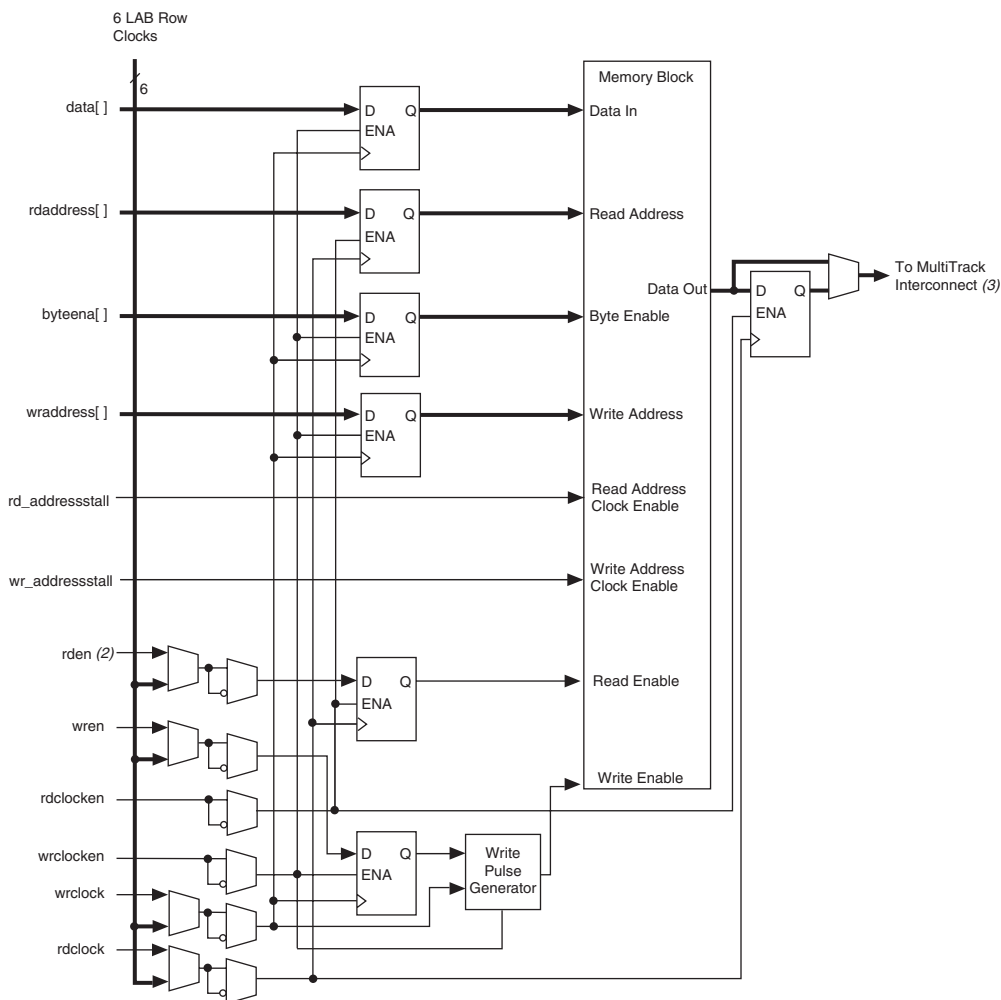


***Notes to Figure 6–16:***
(1)   Violating the setup or hold time on the memory block address registers could corrupt the memory contents. This applies to both read and write operations.
(2)   The read enable `rden` signal is not available in the M-RAM block. An M-RAM block in simple dual-port mode is always reading the data stored at the current read address location.
(3)   For more information about the MultiTrack interconnect, refer to the *Arria GX Device Family Data Sheet* in volume 1 of the *Arria GX Device Handbook*.

## Single-Clock Mode

Arria GX TriMatrix memory blocks implement single-clock mode for true dual-port, simple dual-port, and single-port memory. In this mode, a single clock, together with clock enable, is used to control all registers of the memory block. However, asynchronous clear signals for the registers are not supported. Figures 6–17 through 6–19 show the memory block in single-clock mode for true dual-port, simple dual-port, and single-port modes, respectively.

*Figure 6–17. ArriaGX Single-Clock Mode in True Dual-Port Mode Note (1)*
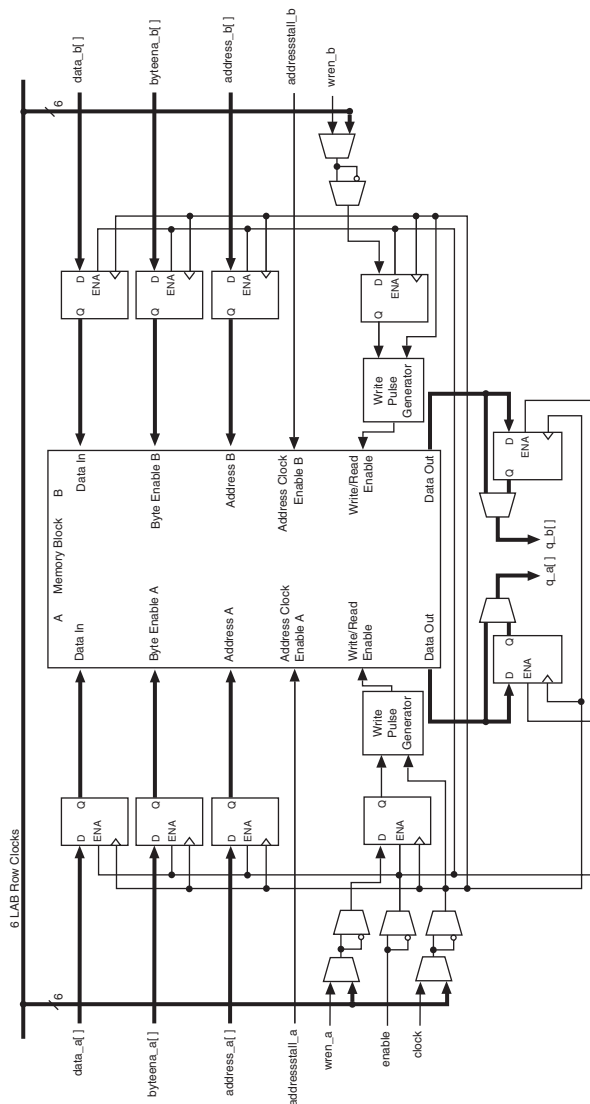


*Note to Figure 6–17:*
(1)  Violating the setup or hold time on the memory block address registers could corrupt the memory contents. This applies to both read and write operations.

*Figure 6–18. ArriaGX Single-Clock Mode in Simple Dual-Port Mode Note (1), (2), (3)*
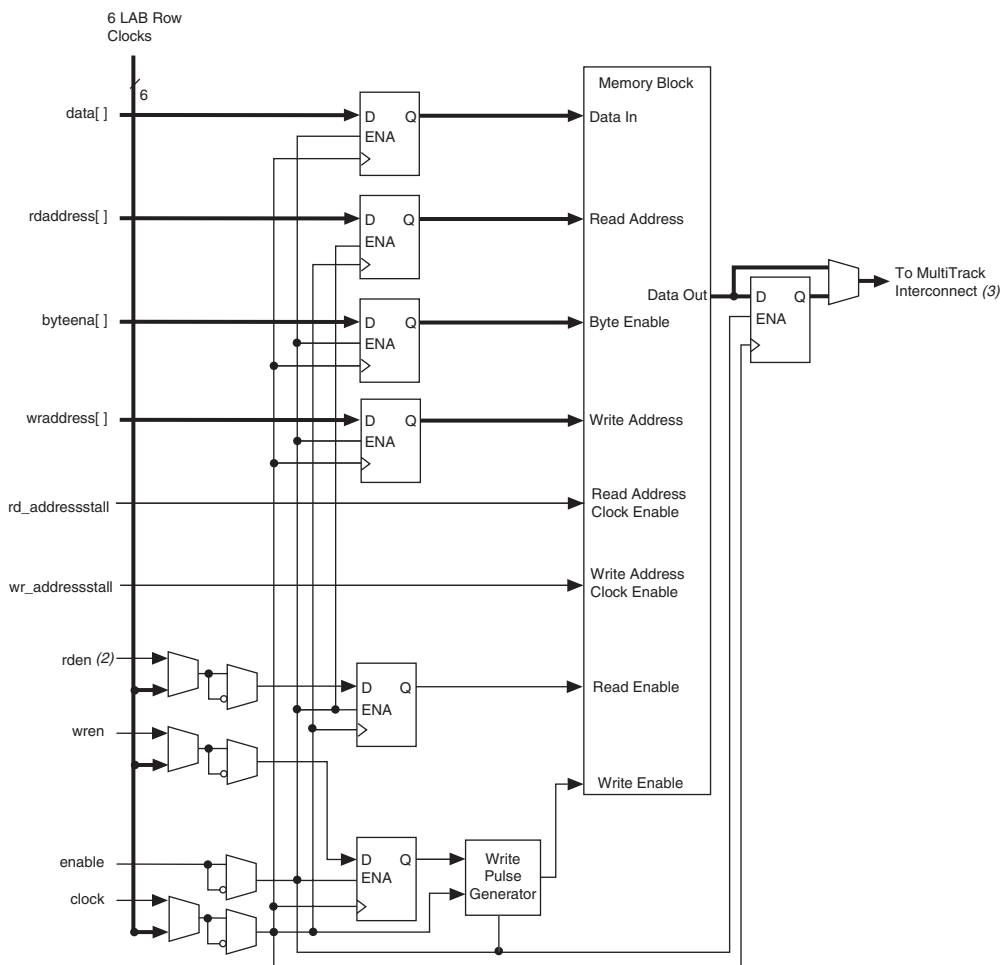


*Notes to Figure 6–18:*
(1) Violating the setup or hold time on the memory block address registers could corrupt the memory contents. This applies to both read and write operations.
(2) The read enable rden signal is not available in the M-RAM block. A M-RAM block in simple dual-port mode is always reading the data stored at the current read address location.
(3) For more information about the MultiTrack interconnect, refer to the *Arria GX Device Family Data Sheet* in volume 1 of the *Arria GX Device Handbook*.

*Figure 6–19. Figure7–19.ArriaGX Single-Clock Mode in Single-Port Mode Note (1), (2)*
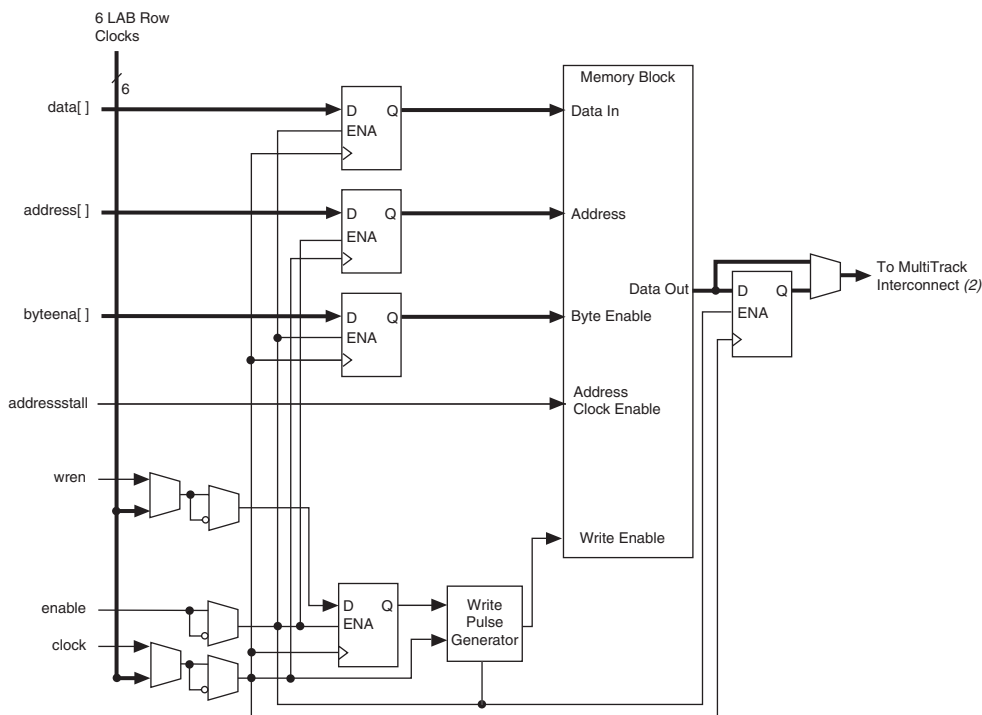


*Notes to Figure 6–19:*
(1)  Violating the setup or hold time on the memory block address registers could corrupt the memory contents. This applies to both read and write operations.
(2)  For more information about the MultiTrack interconnect, refer to the *Arria GX Device Family Data Sheet* in volume 1 of the *Arria GX Device Handbook*.

# Designing With TriMatrix Memory

When instantiating TriMatrix memory, it is important to understand the features that set it apart from other memory architectures. The following sections describe the unique attributes and functionality of TriMatrix memory.

## Selecting TriMatrix Memory Blocks

The Quartus II software automatically partitions user-defined memory into embedded memory blocks using the most efficient size combinations. The memory can also be manually assigned to a specific block size or a mixture of block sizes. Table 6–1 on page 6–2 is a guide for selecting a TriMatrix memory block size based on supported features.

For more information about selecting the appropriate memory block, refer to *AN 207: TriMatrix Memory Selection Using the Quartus II Software*.

## Synchronous and Pseudo-Asynchronous Modes

TriMatrix memory architecture implements synchronous RAM by registering the input and output signals to the RAM block. The inputs to all TriMatrix memory blocks are registered providing synchronous write cycles, while the output registers can be bypassed. In a synchronous operation, RAM generates its own self-timed strobe write-enable signal derived from the global or regional clock. In contrast, a circuit using an asynchronous RAM must generate the RAM write enable signal while ensuring that its data and address signals meet setup and hold time specifications relative to the write enable signal. During a synchronous operation, the RAM is used in pipelined mode (inputs and outputs registered) or flow-through mode (only inputs registered). However, in an asynchronous memory, neither the input nor the output is registered.

While Arria GX devices do not support asynchronous memory, they do support a pseudo-asynchronous read where the output data is available during the clock cycle when the read address is driven into it. Pseudo-asynchronous reading is possible in the simple and true dual-port modes of the M512 and M4K blocks by clocking the read-enable and read-address registers on the negative clock edge and bypassing the output registers.

For more information, refer to *AN 210: Converting Memory from Asynchronous to Synchronous for Stratix & Stratix GX Designs*.

## Power-Up Conditions & Memory Initialization

Upon power up, TriMatrix memory is in an idle state. The M512 and M4K block outputs always power up to zero, regardless of whether the output registers are used or bypassed. Even if a MIF (**.mif**) is used to pre-load the contents of the RAM block, the outputs will still power up as cleared. For example, if address 0 is pre-initialized to FF, the M512 and M4K blocks power up with the output at 00.

M-RAM blocks do not support **.mif** files; therefore, they cannot be pre-loaded with data upon power up. M-RAM blocks asynchronous outputs and memory controls always power up to an unknown state. If M-RAM block outputs are registered, the registers power up as cleared. When a read is performed immediately after power up, the output from the read operation is undefined since the M-RAM contents are not initialized. The read operation continues to be undefined for a given address until a write operation is performed for that address.

# Read-During-Write Operation at the Same Address

The "Same-Port Read-During-Write Mode" and "Mixed-Port Read-During-Write Mode" sections describe the functionality of the various RAM configurations when reading from an address during a write operation at that same address. There are two read-during-write data flows: same-port and mixed-port. Figure 6–20 shows the difference between these flows.

*Figure 6–20. ArriaGX Read-During-Write Data Flow*
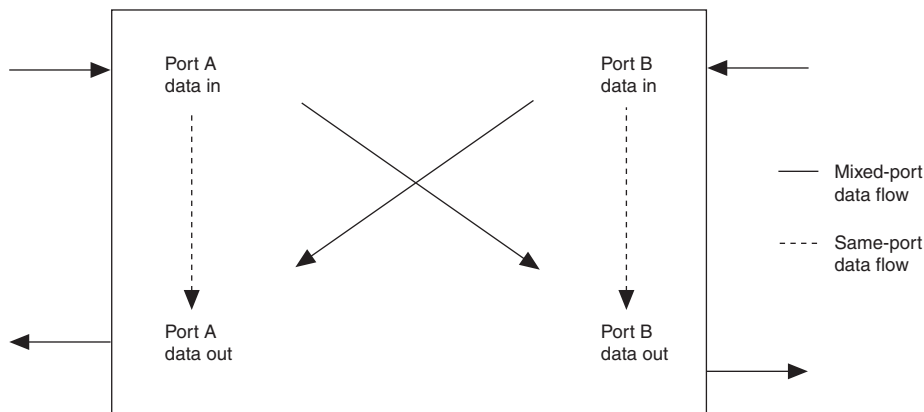


## Same-Port Read-During-Write Mode

For a read-during-write operation of a single-port RAM or the same port of a true dual-port RAM, the new data is available on the rising edge of the same clock cycle on which it was written. This behavior is valid on all memory block sizes. Figure 6–21 shows a sample functional waveform. When using byte enables in true dual-port RAM mode, the outputs for the masked bytes on the same port are unknown (refer to Figure 6–1 on page 6–6). The non-masked bytes are read out, as shown in Figure 6–21.

*Figure 6–21. Arria GX Same-Port Read-During-Write Functionality Note (1)*
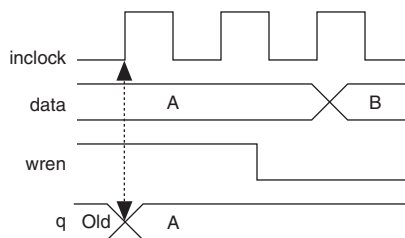


*Note to Figure 6–21:*
(1)    Outputs are not registered.

## Mixed-Port Read-During-Write Mode

This mode is used when a RAM in simple or true dual-port mode has one port reading and the other port writing to the same address location with the same clock.

The READ_DURING_WRITE_MODE_MIXED_PORTS parameter for M512 and M4K memory blocks determines whether to output the old data at the address or a "don't care" value. Setting this parameter to OLD_DATA outputs the old data at that address. Setting this parameter to DONT_CARE outputs a "don't care" or unknown value. Figures 6–22 and 6–23 show sample functional waveforms where both ports have the same address. These figures assume that the outputs are not registered.

The DONT_CARE setting allows memory implementation in any TriMatrix memory block, whereas the OLD_DATA setting restricts memory implementation to only M512 or M4K memory blocks. Selecting DONT_CARE gives the compiler more flexibility when placing memory functions into TriMatrix memory.

The RAM outputs are unknown for a mixed-port read-during-write operation of the same address location of an M-RAM block, as shown in Figure 6–23.

*Figure 6–22. Arria GX Mixed-Port Read-During-Write: OLD_DATA*
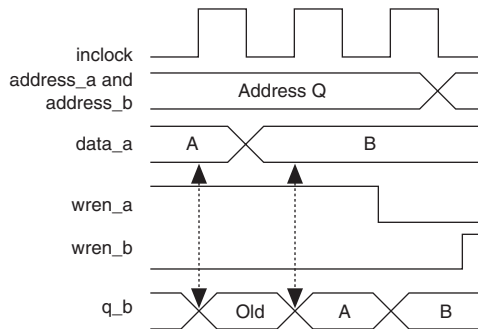


*Figure 6–23. Arria GX Mixed-Port Read-During-Write: DONT_CARE*
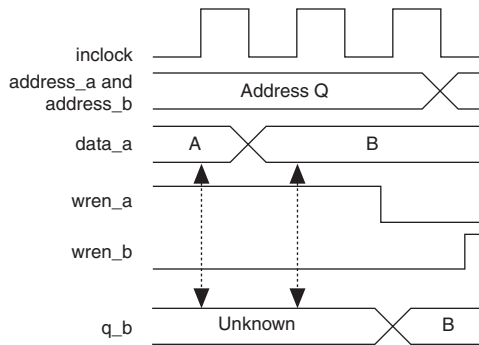


Mixed-port read-during-write is not supported when two different clocks are used in a dual-port RAM. The output value is unknown during a mixed-port read-during-write operation.

**Conclusion**
The TriMatrix memory structure of Arria GX devices provides an enhanced RAM architecture with high memory bandwidth. It addresses the needs of different memory applications in FPGA designs with features such as different memory block sizes and modes, byte enables, parity bit storage, address clock enables, mixed clock mode, shift register mode, mixed-port width support, and true dual-port mode.

## Referenced Documents

This chapter references the following documents:

- *AN 207: TriMatrix Memory Selection Using the Quartus II Software*
- *AN 210: Converting Memory from Asynchronous to Synchronous for Stratix & Stratix GX Designs*
- *Arria GX Device Family Data Sheet* in volume 1 of the *Arria GX Device Handbook*
- *FIFO Partitioner Megafunction User Guide*
- *Single- and Dual-Clock FIFO Megafunctions User Guide*

## Document Revision History

Table 6–14 shows the revision history for this chapter.

| Table 6–14. Document Revision History | | |
|---|---|---|
| **Date and Document Version** | **Changes Made** | **Summary of Changes** |
| May 2008, v1.2 | Updated the "Introduction" section. | — |
| | Minor text edits. | — |
| August 2007, v1.1 | Added the "Referenced Documents" section. | — |
| May 2007, v1.0 | Initial release. | — |

# 7. External Memory Interfaces in Arria GX Devices

**Introduction**

Arria™ GX devices support external memory interfaces, including DDR SDRAM, DDR2 SDRAM, and SDR SDRAM. Its dedicated phase-shift circuitry allows the Arria GX device to interface with an external memory at twice the system clock speed (up to 233 MHz/466 megabits per second (Mbps) with DDR2 SDRAM). In addition to external memory interfaces, you can also use the dedicated phase-shift circuitry for other applications that require a shifted input clock signal.

Most new memory architectures use a DDR I/O interface. Although Arria GX devices also support the mature and well established SDR external memory, this chapter focuses on DDR memory standards. These DDR memory standards cover a broad range of applications for embedded processor systems, image processing, storage, communications, and networking.

Arria GX devices offer external memory support in top and bottom I/O banks. Figure 7–1 shows Arria GX device memory support.

☞ If your system requires memory interface support, you *must* use the ALTMEMPHY megafunction.

This chapter contains the following sections:

*Figure 7–1. External Memory Support*
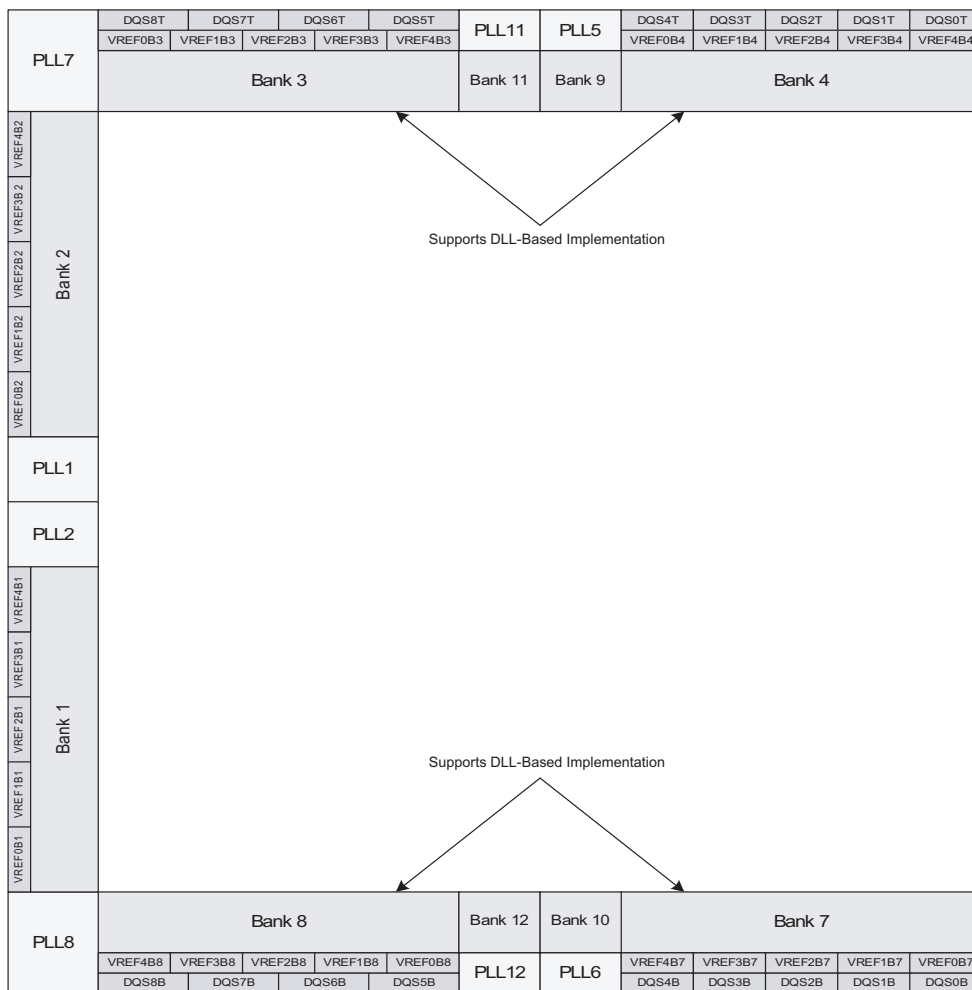


Notes to *Figure 7–1:*
(1) For more information about the ALTMEMPHY megafunction data path, refer to the *ALTMEMPHY Megafunction User Guide.*
(2) EP1AGX20/35 and EP1AGX50/60 devices in the F484 package support external memory interfaces in the top I/O banks only.

Table 7–1 summarizes the maximum clock rate Arria GX devices support with external memory devices.

**Table 7–1. Arria GX Maximum Clock Rate Support for External Memory Interfaces** *Notes (1), (2)*

| Memory Standards | −6 Speed Grade (MHz) |
|---|---|
| DDR2 SDRAM *(3), (4)* | 233 |
| DDR SDRAM *(3), (4)* | 200 |

*Notes to Table 7–1:*
(1) Memory interface timing specifications are dependent on the memory, board, physical interface, and core logic. Refer to each memory interface application note for more details about how each specification is generated.
(2) Numbers are preliminary until characterization is final. The timing information featured in the Quartus® II software version 7.1 was used to define these clock rates.
(3) This applies to interfaces with both modules and components.
(4) These memory interfaces are supported using the ALTMEMPHY megafunction.

This chapter describes the hardware features in Arria GX devices that facilitate high-speed memory interfacing for each DDR memory standard.

# External Memory Standards

The following sections briefly describe external memory standards supported by Arria GX devices. Altera® offers a complete solution for these memories, including clear-text data path, memory controller, and timing analysis.

## DDR and DDR2 SDRAM

DDR SDRAM is a memory architecture that transmits and receives data at twice the clock speed. These devices transfer data on both the rising and falling edges of the clock signal. DDR2 SDRAM is a second-generation memory based on the DDR SDRAM architecture. It transfers data to Arria GX devices at up to 233 MHz/466 Mbps. Arria GX devices can support DDR SDRAM at up to 200 MHz/400 Mbps.

### Interface Pins

DDR and DDR2 SDRAM devices use interface pins such as data (DQ), data strobe (DQS), clock, command, and address pins. Data is sent and captured at twice the system clock rate by transferring data on the clock's positive and negative edges. The commands and addresses still use only one active (positive) edge of a clock. DDR and DDR2 SDRAM use single-ended data strobes (DQS). DDR2 SDRAM can also use optional

differential data strobes (DQS and DQS#). However, Arria GX devices do not support the optional differential data strobes for DDR2 SDRAM interfaces. You can leave the DDR SDRAM memory DQS# pin unconnected. Only the shifted DQS signal from the DQS logic block is used to capture data.

DDR and DDR2 SDRAM ×16 devices use two DQS pins. Each DQS pin is associated with eight DQ pins. However, this is not the same as the ×16/×18 mode in Arria GX devices (see "Data and Data Strobe Pins" on page 7–8). To support a ×16 DDR2 SDRAM device, you need to configure Arria GX devices to use two sets of DQ pins in ×8/×9 mode. Similarly, if your ×32 memory device uses four DQS pins, where each DQS pin is associated with eight DQ pins, you need to configure the Arria GX devices to use four sets of DQS/DQ groups in ×8/×9 mode.

Connect the memory device's DQ and DQS pins to Arria GX DQ and DQS pins, respectively, as listed in the Arria GX pin tables. DDR and DDR2 SDRAM also use active-high data mask, DM, and pins for writes. You can connect the memory's DM pins to any of Arria GX I/O pins in the same bank as the DQ pins of the FPGA. There is one DM pin per DQS/DQ group in a DDR or DDR2 SDRAM device.

For more information about interfacing with DDR SDRAM, refer to *AN 327: Interfacing DDR SDRAM with Stratix II Devices* and *AN 328: Interfacing DDR2 SDRAM with Stratix II Devices*.

You can use any of the user I/O pins for commands and addresses to the DDR and DDR2 SDRAM. You may need to generate these signals from the system clock's negative edge.

The clocks to the SDRAM device are called CK and CK# pins. Use any of the user I/O pins via the DDR registers to generate the CK and CK# signals to meet the DDR SDRAM or DDR2 SDRAM device's $t_{DQSS}$ requirement. The memory device's $t_{DQSS}$ specification requires that the write DQS signal's positive edge must be within 25% of the positive edge of the DDR SDRAM or DDR2 SDRAM clock input. Using regular I/O pins for CK and CK# also ensures that any PVT variations on the DQS signals are tracked the same way by these CK and CK# pins. Figure 7–2 shows a diagram that illustrates how to generate these clocks.

*Figure 7–2. Clock Generation for External Memory Interfaces in Arria GX Devices*



*Note to Figure 7–2:*
(1)   CK and CK# are the clocks to the memory devices.

## Read and Write Operations
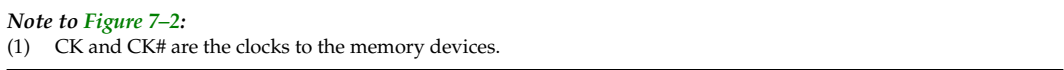
When reading from the memory, DDR and DDR2 SDRAM devices send the data edge-aligned with respect to the data strobe. To properly read the data in, the data strobe needs to be center-aligned with respect to the data inside the FPGA. Arria GX devices feature dedicated circuitry to shift this data strobe to the middle of the data window. Figure 7–3 shows an example of how the memory sends out the data and data strobe for a burst-of-two operation.

*Figure 7–3. Example of a 90º Shift on the DQS Signal Notes (1), (2)*
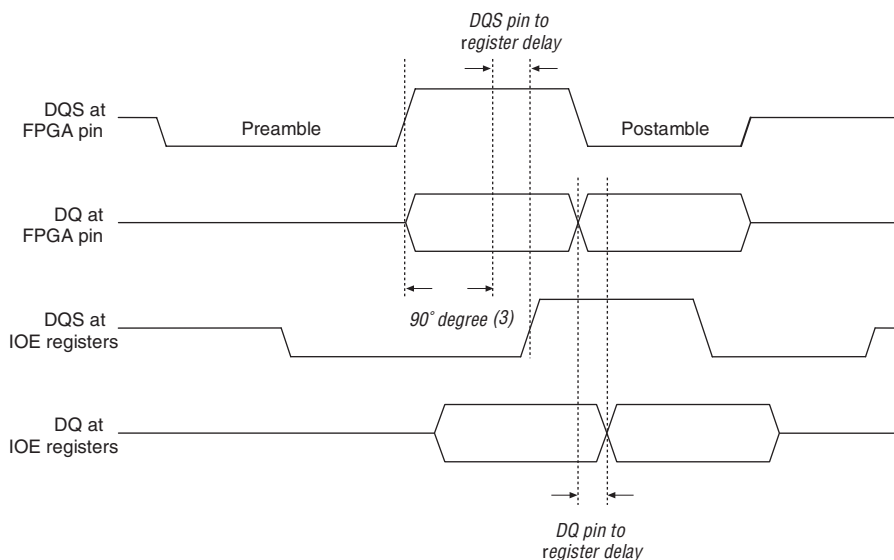


*Notes to Figure 7–3:*
(1)    DDR2 SDRAM does not support a burst length of two.
(2)    The phase shift required for your system should be based on your timing analysis and may not be 90º.

During write operations to a DDR or DDR2 SDRAM device, the FPGA needs to send the data to the memory center-aligned with respect to the data strobe. Arria GX devices use a PLL to center-align the data by generating a 0º phase-shifted system clock for the write data strobes and a –90º phase-shifted write clock for the write data pins for DDR and DDR2 SDRAM. Figure 7–4 shows an example of the relationship between the data and data strobe during a burst-of-four write.

*Figure 7–4. DQ and DQS Relationship During a DDR and DDR2 SDRAM Write Note (1)*
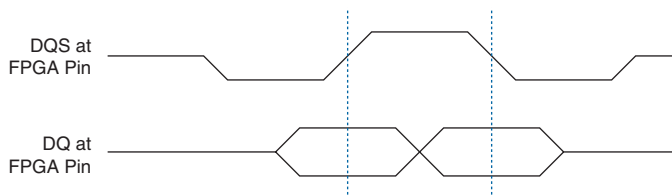


*Note to Figure 7–4:*
(1)    This example shows a write for a burst length of two. DDR SDRAM devices do not support burst lengths of two.

For more information about DDR SDRAM and DDR2 SDRAM specifications, refer to the JEDEC standard publications JESD79C and JESD79-2, respectively, at www.jedec.org.

# Arria GX DDR Memory Support Overview

This section describes Arria GX features that enable high-speed memory interfacing. It first describes Arria GX memory pins and then DQS phase-shift circuitry and DDR I/O registers. Table 7–2 shows the I/O standard associated with the external memory interfaces.

| Table 7–2. External Memory Support in Arria GX Devices | |
|---|---|
| **Memory Standard** | **I/O Standard** |
| DDR2 SDRAM | SSTL-18 Class II *(1)* |
| DDR SDRAM | SSTL-2 Class II |

*Note to Table 7–2:*

(1)   Arria GX devices support 1.8-V HSTL/SSTL-18 Class I and II I/O standards in I/O banks 3, 4, 7, and 8.

Arria GX devices support data strobe or read clock signal (DQS) used in DDR SDRAM and DDR2 SDRAM devices with dedicated circuitry.

For more information about memory interfaces, see the appropriate Stratix II or Stratix II GX memory interfaces application note available at www.altera.com.

Arria GX devices contain dedicated circuitry to shift incoming DQS signals by 0º, 22.5º, 30º, 36º, 45º, 60º, 67.5º, 72º, 90º, 108º, 120º, or 144º, depending on the delay-locked loop (DLL) mode. There are four DLL modes. The DQS phase-shift circuitry uses a frequency reference to dynamically generate control signals for the delay chains in each of the DQS pins, allowing it to compensate for process, voltage, and temperature (PVT) variations. This phase-shift circuitry has been enhanced in Arria GX devices to support more phase-shift options with less jitter.

Besides DQS dedicated phase-shift circuitry, each DQS pin has its own DQS logic block that sets the delay for the signal input to the pin. Using DQS dedicated phase-shift circuitry with the DQS logic block allows for phase-shift fine-tuning. Additionally, every IOE in an Arria GX device contains six registers and one latch to achieve DDR operation.

## DDR Memory Interface Pins

Arria GX devices use data (DQ), data strobe (DQS), and clock pins to interface with external memory.

Figure 7–5 shows DQ and DQS pins in the Arria GX I/O banks on the top of the device. A similar arrangement is repeated at the bottom of the device.

*Figure 7–5. DQ and DQS Pins Per I/O Bank*
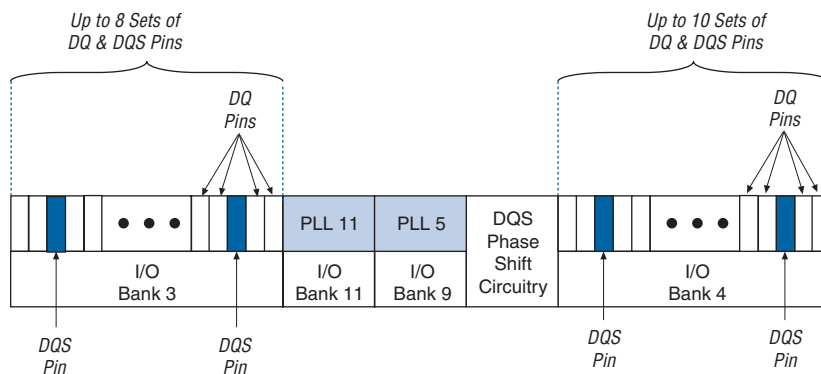


### Data and Data Strobe Pins

Arria GX data pins for DDR memory interfaces are called DQ pins. Arria GX devices can use either bidirectional data strobes or unidirectional read clocks. Depending on the external memory interface, either the memory device's read data strobes or read clocks feed the Arria GX DQS pins.

Arria GX DQS pins connect to the DQS pins in DDR and DDR2 SDRAM interfaces. In every Arria GX device, the I/O banks at the top (I/O banks 3 and 4) and bottom (I/O banks 7 and 8) of the device support DDR memory up to 233 MHz/466 Mbps (with DDR2). These I/O banks support DQS signals with DQ bus modes of ×4, ×8/×9, ×16/×18, or ×32/×36.

In ×4 mode, each DQS pin drives up to four DQ pins within that group. In ×8/×9 mode, each DQS pin drives up to nine DQ pins within that group to support one parity bit and eight data bits. If the parity bit or any data bit is not used, you can use the extra DQ pins as regular user I/O pins. Similarly, with ×16/×18 and ×32/×36 modes, each DQS pin drives up to 18 and 36 DQ pins, respectively. There are two parity bits in the

×16/×18 mode and four parity bits in the ×32/×36 mode. Table 7–3 shows the number of DQS/DQ groups supported in each Arria GX package for DLL-based implementations.

**Table 7–3. Arria GX DQS and DQ Bus Mode Support** *Note (1)*

| Package | Number of ×4 Groups | Number of ×8/×9 Groups | Number of ×16/×18 Groups | Number of ×32/×36 Groups |
|---|---|---|---|---|
| 484-pin FineLine BGA | 2 | 0 | 0 | 0 |
| 780-pin FineLine BGA | 18 | 8 | 4 | 0 |
| 1,152-pin FineLine BGA | 36 | 18 | 8 | 4 |

*Note to Table 7–3:*
(1)    Check the pin table for each DQS/DQ group in the different modes.

The DQS pins are listed in the Arria GX pin tables as DQS[17..0]T or DQS[17..0]B. The T denotes pins on the top of the device; the B denotes pins on the bottom of the device. Corresponding DQ pins are marked as DQ[17..0]. The numbering scheme starts from right to left on the package bottom view. When not used as DQ or DQS pins, these pins are available as regular I/O pins. Figure 7–6 shows the DQS pins in Arria GX I/O banks.

**Figure 7–6. DQS Pins in Arria GX I/O Banks** *Note (1)*, *(2)*

**Top I/O Banks**

| DQS17T | DQS16T | DQS15T | ● ● ● | DQS10T | **PLL 11** | **PLL 5** | DQS Phase Shift Circuitry | DQS9T | DQS8T | ● ● ● | DQS0T |
|---|---|---|---|---|---|---|---|---|---|---|---|

| I/O Bank 3 | | | | | I/O Bank 11 | I/O Bank 9 | | I/O Bank 4 | | | |

**Bottom I/O Banks**

| I/O Bank 8 | | | | | I/O Bank 12 | I/O Bank 10 | | I/O Bank 7 | | | |

| DQS17B | DQS16B | DQS15B | ● ● ● | DQS10B | **PLL 12** | **PLL 6** | DQS Phase Shift Circuitry | DQS9B | DQS8B | ● ● ● | DQS0B |
|---|---|---|---|---|---|---|---|---|---|---|---|

*Notes to Figure 7–6:*
(1)    There are up to 18 pairs of DQS pins on both the top and bottom of the device.
(2)    See Table 7–3 for DQS bus mode support based on the package.

The DQ pin numbering is based on ×4 mode. There are up to eight DQS/DQ groups in ×4 mode in I/O banks 3 and 8 and up to 10 DQS/DQ groups in ×4 mode in I/O banks 4 and 7. In ×8/×9 mode, two adjacent ×4 DQS/DQ groups plus one parity pin are combined; one DQS pin from the combined groups can drive all the DQ and parity pins. Since there is an even number of DQS/DQ groups in an I/O bank, combining groups is efficient. Similarly, in ×16/×18 mode, four adjacent ×4 DQS/DQ groups plus two parity pins are combined and one DQS pin from the combined groups can drive all the DQ and parity pins. In ×32/×36 mode, eight adjacent DQS/DQ groups are combined and one DQS pin can drive all the DQ and parity pins in the combined groups.

☞     On the top and bottom side of the device, the DQ and DQS pins must be configured as bidirectional DDR pins to enable the DQS phase-shift circuitry. You must use the ALTMEMPHY megafunction to configure the DQ and DQS paths, respectively.

### Clock Pins

You can use any of the DDR I/O registers to generate clocks to the memory device. For better performance, use the same I/O bank as the data and address and command pins.

### Address and Command Pins

You can use any of the user I/O pins in the top or bottom bank of the device for addresses and commands. For better performance, use the same I/O bank as the data pins.

### Other Pins (Parity, DM Pins)

You can use any of the DQ pins for parity pins in Arria GX devices. The Arria GX device family has support for parity in ×8/×9, ×16/×18, and ×32/×36 mode. There is one parity bit available per eight bits of data pins.

The data mask and DM pins are only required when writing to DDR SDRAM and DDR2 SDRAM devices. A low signal on the DM pins indicates that the write is valid. If the DM signal is high, the memory masks the DQ signals. You can use any I/O pins in the same bank as the DQ pins for DM signals. Each group of DQS and DQ signals in DDR and DDR2 SDRAM devices requires a DM pin. The DDR I/O output registers, clocked by the −90° shifted clock, creates the DM signals, similar to DQ output signals.

☞     Perform timing analysis to calculate write clock phase shift.

## DQS Phase-Shift Circuitry

Arria GX phase-shift circuitry and DQS logic block controls the DQS pins. Each Arria GX device contains two phase-shifting circuits. There is one circuit for I/O banks 3 and 4 and another circuit for I/O banks 7 and 8. The phase-shifting circuit on the top of the device can control all the DQS pins in the top I/O banks; the phase-shifting circuit on the bottom of the device can control all the DQS pins in the bottom I/O banks. Figure 7–7 shows DQS pin connections to the DQS logic block and DQS phase-shift circuitry.

*Figure 7–7. DQS Pins and DQS Phase-Shift Circuitry Note (1)*
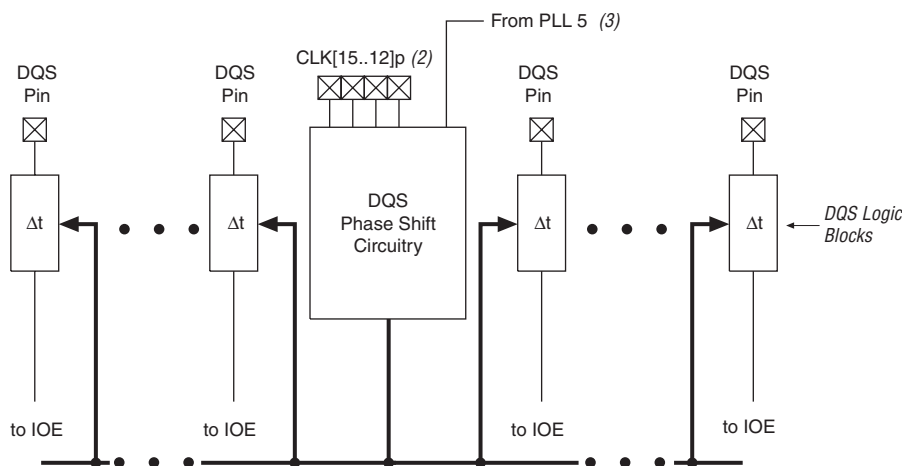


*Notes to Figure 7–7:*
(1) There are up to 18 pairs of DQS pins available on the top or bottom of the Arria GX device, up to eight on the left side of the DQS phase-shift circuitry (I/O banks 3 and 8) and up to ten on the right side (I/O banks 4 and 7).
(2) Clock pins CLK[15..12]p feed the phase-shift circuitry on the top of the device; clock pins CLK[7..4]p feed the phase-shift circuitry on the bottom of the device. You can also use a phase-locked loop (PLL) clock output as a reference clock to the phase-shift circuitry. You can also use the reference clock in the logic array.
(3) You can only use PLL 5 to feed DQS phase-shift circuitry on the top of the device and PLL 6 to feed DQS phase-shift circuitry on the bottom of the device.

Figure 7–8 shows the connections between the DQS phase-shift circuitry and the DQS logic block.

*Figure 7–8. DQS Phase-Shift Circuitry and DQS Logic Block Connections Note (1)*
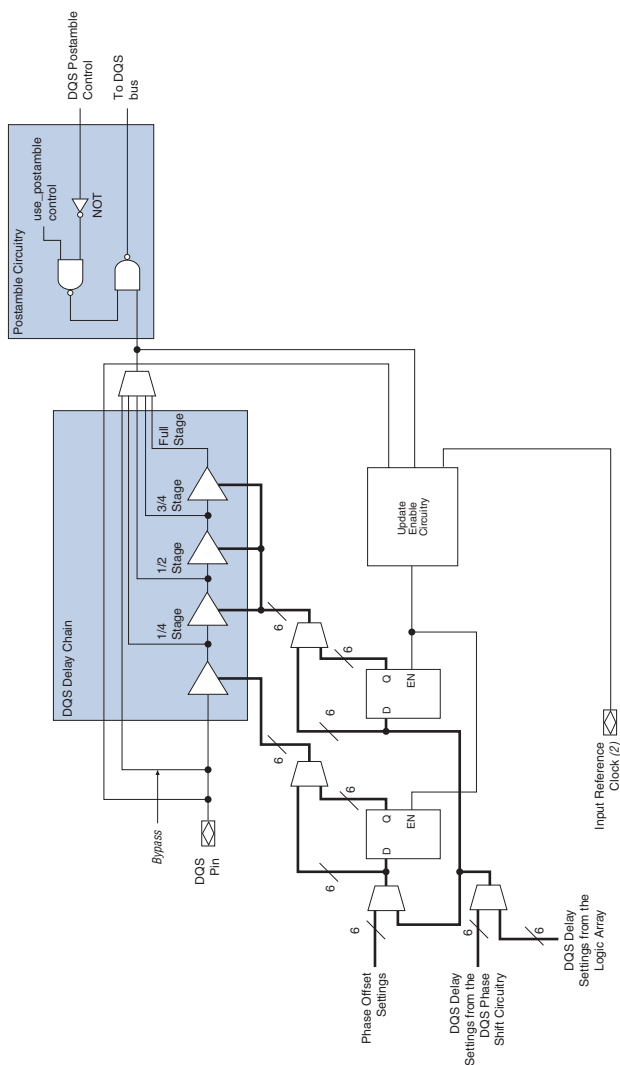


**Notes to Figure 7–8:**
(1)    All features of the DQS phase-shift circuitry and DQS logic block are controlled from the ALTMEMPHY megafunction in the Quartus II software.
(2)    DQS logic block is available on every DQS pin.
(3)    There is one DQS phase-shift circuit on the top and bottom side of the device.
(4)    The input reference clock can come from CLK[15..12]p or PLL 5 for the DQS phase-shift circuitry on the top side of the device or from CLK[7..4]p or PLL 6 for the DQS phase-shift circuitry on the bottom side of the device.
(5)    Each individual DQS pin can have individual DQS delay settings to and from the logic array.
(6)    This register is one of the DQS IOE input registers.

Phase-shift circuitry is only used during read transactions where the DQS pins are acting as input clocks or strobes. Phase-shift circuitry can shift the incoming DQS signal by 0º, 22.5º, 30º, 36º, 45º, 60º, 67.5º, 72º, 90º, 108º, 120º, or 144º. The shifted DQS signal is then used as clocks at the DQ IOE input registers.

Figure 7–3 on page 7–6 shows an example where the DQS signal is shifted by 90º. The DQS signal goes through the 90º shift delay set by the DQS phase-shift circuitry and the DQS logic block and some routing delay from the DQS pin to the DQ IOE registers. DQ signals only goes through routing delay from the DQ pin to the DQ IOE registers and maintains the 90º relationship between the DQS and DQ signals at the DQ IOE registers since the software automatically sets delay chains to match the routing delay between the pins and the IOE registers for the DQ and DQS input paths.

All 18 DQS pins on either the top or bottom of the device can have their input signal phase shifted by a different degree amount but all must be referenced at one particular frequency. For example, you can have a 90º phase shift on DQS0T and have a 60º phase shift on DQS1T, both referenced from a 200-MHz clock. Not all phase-shift combinations are supported, however. The phase shifts on the same side of the device must all be a multiple of 22.5º (up to 90º), a multiple of 30º (up to 120º), or a multiple of 36º (up to 144º).

In order to generate the correct phase shift with the DLL used, you must provide a clock signal of the same frequency as the DQS signal to the DQS phase-shift circuitry. Any of the CLK[15..12]p clock pins can feed the phase circuitry on the top of the device (I/O banks 3 and 4) or any of the CLK[7..4]p clock pins can feed the phase circuitry on the bottom of the device (I/O banks 7 and 8). Arria GX devices can also use PLLs 5 or 6 as the reference clock to the DQS phase-shift circuitry on the top or bottom of the device, respectively. PLL 5 is connected to the DQS phase-shift circuitry on the top side of the device; PLL 6 is connected to the DQS phase-shift circuitry on the bottom side of the device. Both the top and bottom phase-shift circuits need unique clock pins or PLL clock outputs for the reference clock.

☞       When you have a PLL dedicated only to generate the DLL input reference clock, you must set the PLL mode to **No Compensation** or the Quartus II software will change the setting automatically. Because there are no other PLL outputs used, the PLL does not need to compensate for any clock paths.

*DLL*

DQS phase-shift circuitry uses a delay-locked loop (DLL) to dynamically measure the clock period needed by the DQS pin (see Figure 7–8). DQS phase-shift circuitry then uses the clock period to generate the correct phase shift. The DLL in the Arria GX DQS phase-shift circuitry can operate between 100 and 233 MHz. Phase-shift circuitry needs a maximum of 256 clock cycles to calculate the correct input clock period. Data sent during these clock cycles may not be properly captured.

☞ Although the DLL can run up to 233 MHz, other factors may prevent you from interfacing with a 233-MHz external memory device.

☞ You can still use DQS phase-shift circuitry for any memory interfaces that are less than 100 MHz. The DQS signal is shifted by 2.5 ns. You can add more shift by using the phase offset module. Even if the DQS signal is not shifted exactly to the middle of the DQ valid window, the IOE is still be able to capture the data in this low frequency application.

There are three different frequency modes for the Arria GX DLL. Each frequency mode provides different phase shift, as shown in Table 7–4.

*Table 7–4. Arria GX DLL Frequency Modes*

| Frequency Mode | Frequency Range (MHz) | Available Phase Shift | Number of Delay Chains |
|---|---|---|---|
| 0 | 100–175 | 30, 60, 90, 120 | 12 |
| 1 | 150–230 | 22.5, 45, 67.5, 90 | 16 |
| 2 | 200–310 | 30, 60, 90, 120 | 12 |

In frequency mode 0, Arria GX devices use a 6-bit setting to implement phase-shift delay. In frequency modes 1 and 2, Arria GX devices only use a 5-bit setting to implement phase-shift delay.

You can reset the DLL from either the logic array or a user I/O pin. This signal is not shown in Figure 7–9. Each time the DLL is reset, you must wait for 256 clock cycles before you can capture the data properly. Additionally, if the DLL reference clock is stopped and restarted thereafter, such as during SDRAM refresh cycles, a minimum of 16 clock cycles is needed before capturing data properly.

*Figure 7–9. Simplified Diagram of the DQS Phase-Shift Circuitry Note (1)*
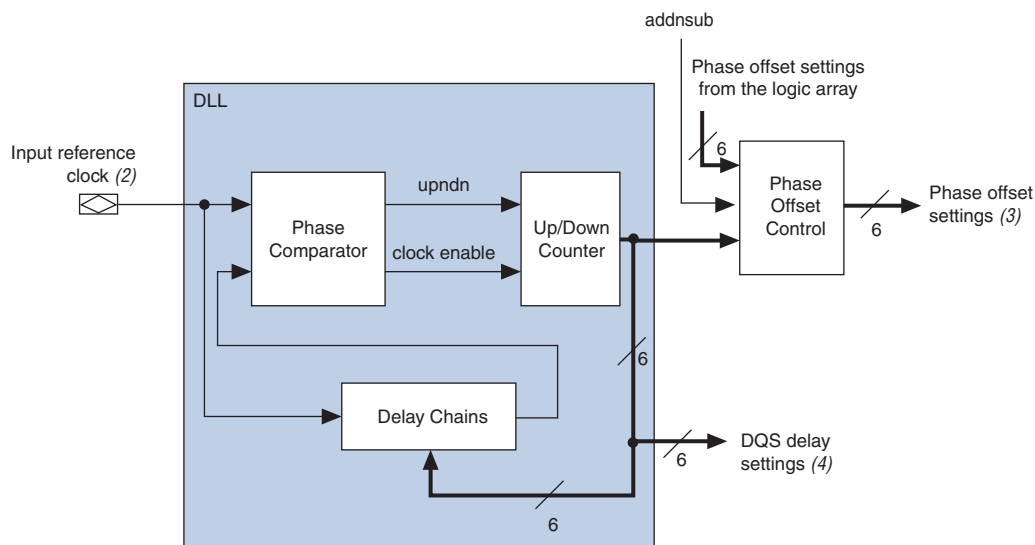


*Notes to Figure 7–9:*
(1)   All features of the DQS phase-shift circuitry are accessible from the ALTMEMPHY megafunction in the Quartus II software.
(2)   The input reference clock for DQS phase-shift circuitry on the top side of the device can come from `CLK[15..12]p` or PLL 5. The input reference clock for DQS phase-shift circuitry on the bottom side of the device can come from `CLK[7..4]p` or PLL 6.
(3)   Phase offset settings can only go to DQS logic blocks.
(4)   DQS delay settings can go to the logic array and/or the DQS logic block.

The input reference clock goes into the DLL to a chain of up to 16 delay elements. The phase comparator compares the signal coming out of the end of the delay element chain to the input reference clock. The phase comparator then issues the upndn signal to the up/down counter. This signal increments or decrements a 6-bit delay setting (DQS delay settings) that increases or decreases the delay through the delay element chain to bring the input reference clock and the signals coming out of the delay element chain in phase.

DQS delay settings contain control bits to shift the signal on the input DQS pin by the amount set in the ALTMEMPHY megafunction. For 0 shift, both the DLL and DQS logic blocks are bypassed. Since Arria GX DQS and DQ pins are designed such that the pin-to-IOE delays are matched, the skew between the DQ and DQS pins at the DQ IOE registers is negligible when you implement 0 shift. You can feed the DQS delay settings to the DQS logic block and logic array.

*Phase Offset Control*

DQS phase-shift circuitry also contains a phase offset control module that can add or subtract a phase offset amount from the DQS delay setting (phase offset settings from the logic array in Figure 7–10). You should use the phase offset control module for making small shifts to the input signal and use the DQS phase-shift circuitry for larger signal shifts.

You can either use a static phase offset or a dynamic phase offset to implement the additional phase shift. The available additional phase shift is implemented in 2s-complement between settings –64 to +63 for frequency mode 0 and between settings –32 to +31 for frequency modes 1, 2, and 3.

For more information about the value for each step, refer to the *DC & Switching Characteristics* chapter in volume 1 of the *Arria GX Device Handbook*. If you need one additional degree phase shift, you must convert the delay amount to degrees in the operating frequency.

When using the static phase offset, you can specify the phase offset amount in the ALTMEMPHY megafunction as a positive number for addition or a negative number for subtraction. You can also have a dynamic phase offset that is always added to, subtracted from, or both added to and subtracted from the DLL phase shift. When you always add or subtract, you can dynamically input the phase offset amount into the `dll_offset[5..0]` port. When you want to both add and subtract dynamically, you control the `addnsub` signal in addition to the `dll_offset[5..0]` signals.

## DQS Logic Block

Each DQS pin is connected to a separate DQS logic block (see Figure 7–10). The logic block contains DQS delay chains and postamble circuitry.

*Figure 7–10. Simplified Diagram of the DQS Logic Block Note (1)*



*Notes to Figure 7–10:*
(1) All features of the DQS logic block are controllable from the ALTMEMPHY megafunction in the Quartus II software.
(2) The input reference clock for DQS phase-shift circuitry on the top side of the device can come from `CLK[15..12]p` or PLL 5. The input reference clock for DQS phase-shift circuitry on the top side of the device can come from `CLK[7..4]p` or PLL 6.
(3) This register is one of the DQS IOE input registers.

### DQS Delay Chains

DQS delay chains consist of a set of variable delay elements to allow the input DQS signals to be shifted by the amount given by the DQS phase-shift circuitry or the logic array. There are four delay elements in the DQS delay chain; the first delay chain closest to the DQS pin can either be shifted by the DQS delay settings or by the sum of the DQS delay setting and the phase-offset setting. The number of delay chains used is transparent to the users because the ALTMEMPHY megafunction automatically sets it. DQS delay settings can come from DQS phase-shift circuitry on the same side of the device as the target DQS logic block or from the logic array. When you apply a 0º shift in the ALTMEMPHY megafunction, DQS delay chains are bypassed.

The delay elements in the DQS logic block mimic the delay elements in the DLL. The amount of delay is equal to the sum of the delay element's intrinsic delay and the product of the number of delay steps and the value of the delay steps.

Both the DQS delay settings and the phase-offset settings pass through a latch before going into the DQS delay chains. The latches are controlled by the update enable circuitry to allow enough time for any changes in the DQS delay setting bits to arrive to all the delay elements. This allows them to be adjusted at the same time. The update enable circuitry enables the latch to allow enough time for the DQS delay settings to travel from the DQS phase-shift circuitry to all the DQS logic blocks before the next change. It uses the input reference clock to generate the update enable output. The ALTMEMPHY megafunction uses this circuit by default. See Figure 7–11 for an example waveform of the update enable circuitry output.

The shifted DQS signal then goes to the DQS bus to clock the IOE input registers of the DQ pins. It can also go into the logic array for resynchronization purposes.

*Figure 7–11. DQS Update Enable Waveform*
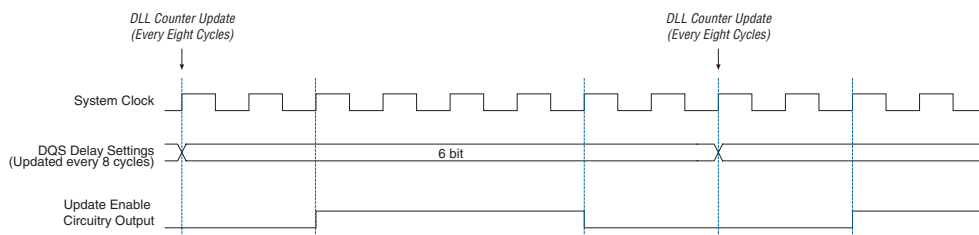
### DQS Postamble Circuitry

For external memory interfaces that use a bidirectional read strobe like DDR and DDR2 SDRAM, the DQS signal is low before going to or coming from a high-impedance state. See Figure 7–3 on page 7–6. The state where DQS is low, just after a high-impedance state, is called the preamble; the state where DQS is low, just before it returns to a high-impedance state, is called the postamble. There are preamble and postamble specifications for both read and write operations in DDR and DDR2 SDRAM. DQS postamble circuitry ensures data is not lost when there is noise on the DQS line at the end of a read postamble time. It is to be used with one of the DQS IOE input registers such that the DQS postamble control signal can ground the shifted DQS signal used to clock the DQ input registers at the end of a read operation. This ensures that any glitches on the DQS input signals at the end of the read postamble time do not affect the DQ IOE registers.

For more information about DDR SDRAM and DDR2 SDRAM, refer to *AN 327: Interfacing DDR SDRAM with Stratix II Devices* and *AN 328: Interfacing DDR2 SDRAM with Stratix II Devices*.

## DDR Registers

Each IOE in an Arria GX device contains six registers and one latch. Two registers and a latch are used for input, two registers are used for output, and two registers are used for output enable control. The second output enable register provides the write preamble for the DQS strobe in DDR external memory interfaces. This active-low output enable register extends the high-impedance state of the pin by a half clock cycle to provide the external memory's DQS write preamble time specification. Figure 7–12 shows the six registers and the latch in the Arria GX IOE. Figure 7–13 shows how the second OE register extends the DQS high-impedance state by half a clock cycle during a write operation.

*Figure 7–12. Bidirectional DDR I/O Path in Arria GX Devices Note (1)*
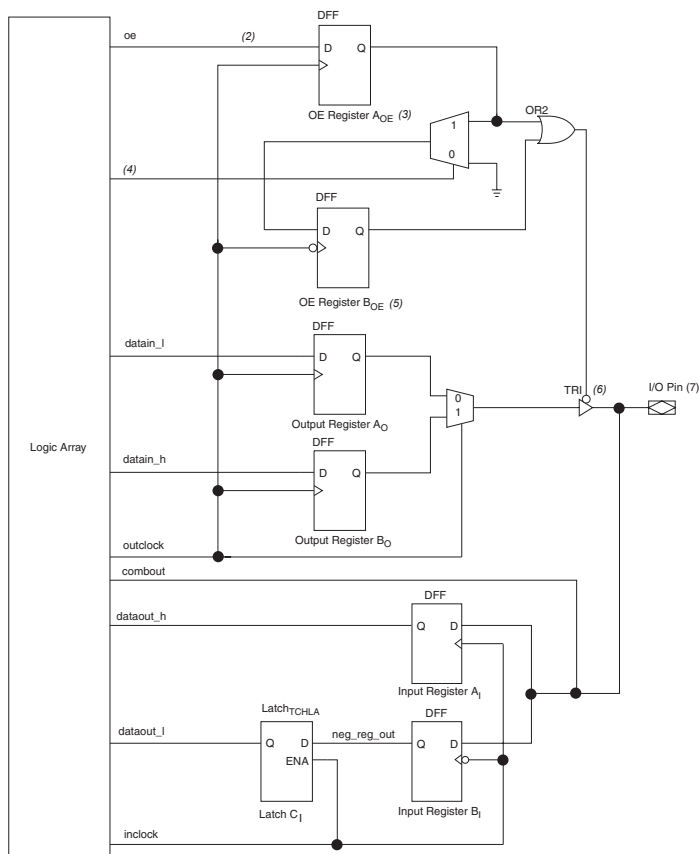


*Notes to Figure 7–12:*
(1) All control signals can be inverted at the IOE. The signal names used here match the Quartus II software naming convention.
(2) The OE signal is active low, but the Quartus II software implements this as active high and automatically adds an inverter before input to the $A_{OE}$ register during compilation.
(3) The $A_{OE}$ register generates the enable signal for general-purpose DDR I/O applications.
(4) This select line is to choose whether the OE signal should be delayed by half-a-clock cycle.
(5) The $B_{OE}$ register generates the delayed enable signal for the write strobes or write clocks for memory interfaces.
(6) The tri-state enable is active low by default. However, you can design it to be active high. The combinational control path for the tri-state is not shown in this diagram.
(7) You can also have combinational output to the I/O pin; this path is not shown in the diagram.
(8) On the top and bottom I/O banks, the clock to this register can be an inverted register A's clock or a separate clock (inverted or non-inverted).

*Figure 7–13. Extending the OE Disable by Half-a-Clock Cycle for a Write Transaction Note (1)*
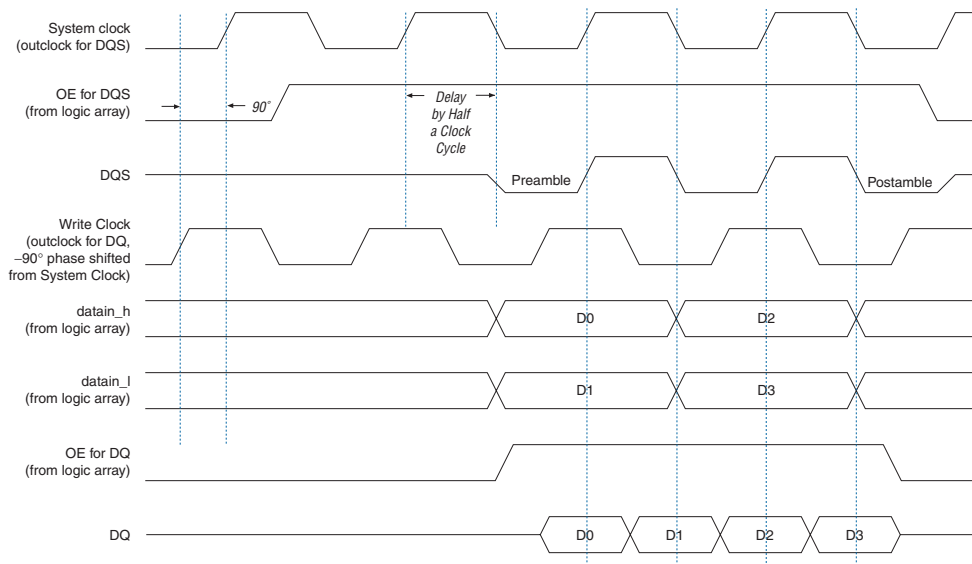


*Note to Figure 7–13:*

(1)    This waveform reflects the software simulation result. The OE signal is active low on the device. However, the
       Quartus II software implements this signal as active high and automatically adds an inverter before the $A_{OE}$ register
       D input.

Figures 7–14 and 7–15 summarize the IOE registers used for the DQ and DQS signals.

*Figure 7–14. DQ Configuration in Arria GX IOE Note (1)*
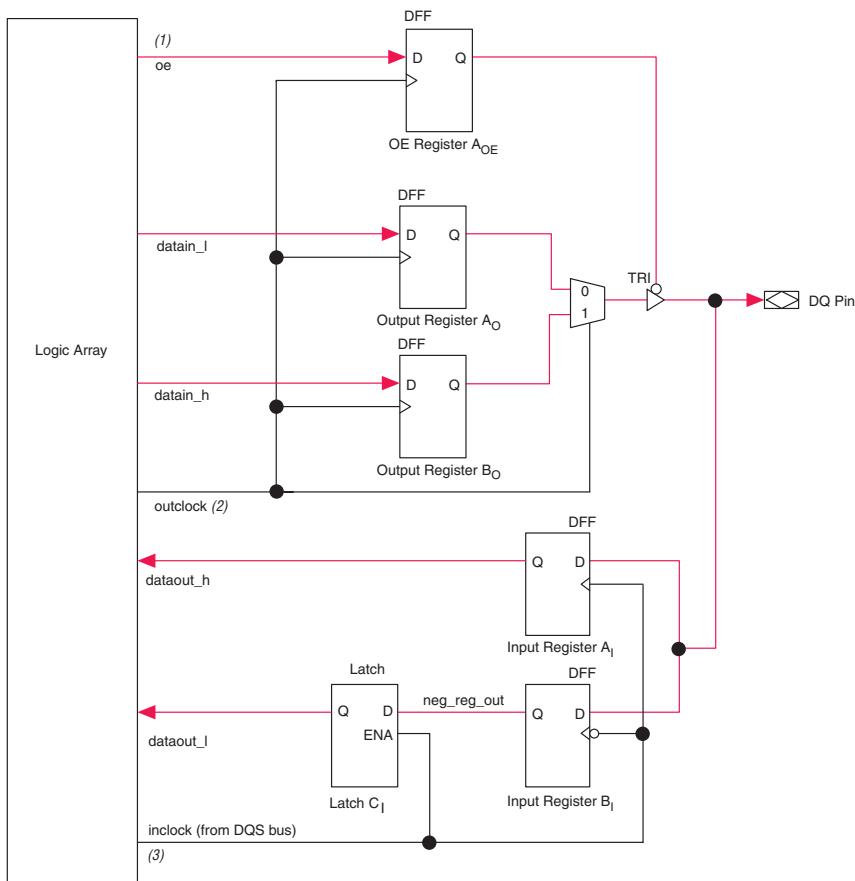


*Notes to Figure 7–14:*
(1)    You should use the ALTMEMPHY megafunction to generate the data path for your memory interface.
(2)    The `OE` signal is active low, but the Quartus II software implements this as active high and automatically adds an inverter before the `OE` register $A_{OE}$ during compilation.
(3)    The `outclock` signal for DDR and DDR2 SDRAM interfaces has a 90° phase-shift relationship with the system clock. The shifted DQS signal can clock this register.
(4)    The shifted DQS signal must be inverted before going to the DQ IOE. The inversion is automatic if you use the ALTMEMPHY megafunction to generate the DQ signals.
(5)    On the top and bottom I/O banks, the clock to this register can be an inverted register A's clock or a separate clock (inverted or non-inverted).

*Figure 7–15. DQS Configuration in Arria GX IOE Note (1)*
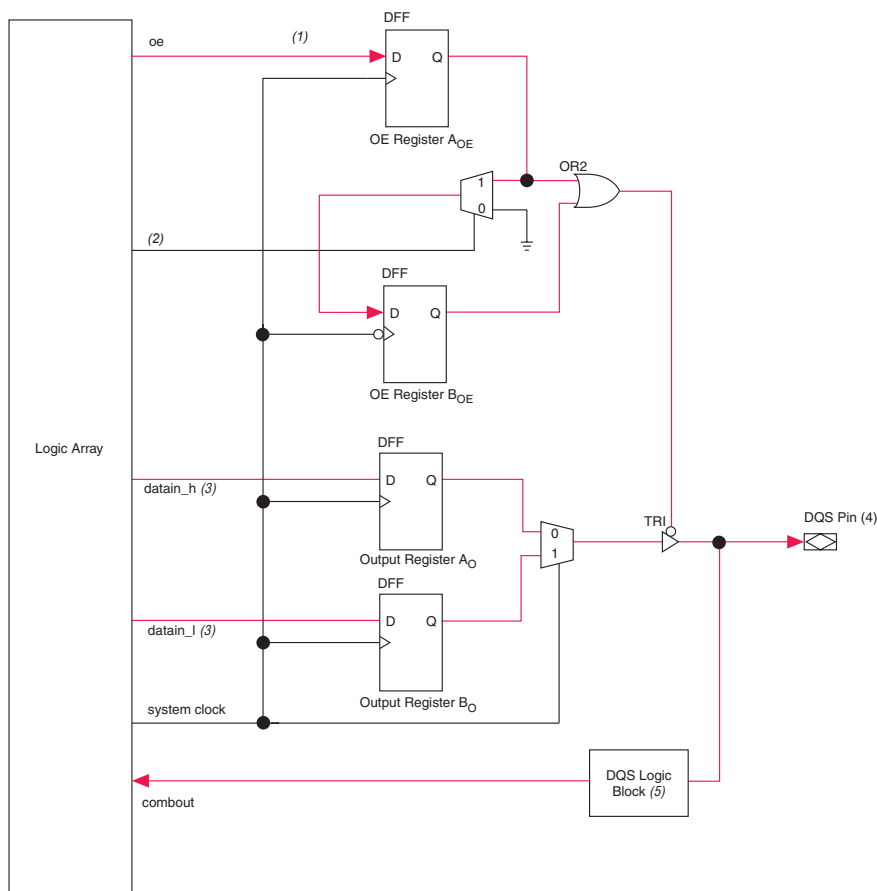


**Notes to Figure 7–15:**
(1) Use the ALTMEMPHY megafunction to generate the data path for your memory interface.
(2) The OE signal is active low, but the Quartus II software implements this as active high and automatically adds an inverter before OE register $A_{OE}$ during compilation.
(3) The select line can be chosen in the ALTMEMPHY megafunction.
(4) The datain_l and datain_h pins are usually connected to ground and $V_{CC}$, respectively.
(5) DQS postamble circuitry and handling is not shown in this diagram. For more information, refer to *AN 327: Interfacing DDR SDRAM with Stratix II Devices* and *AN 328: Interfacing DDR2 SDRAM with Stratix II Devices*.
(6) DQS logic blocks are only available with DQS pins.
(7) You must invert this signal before it reaches the DQ IOE. This signal is automatically inverted if you use the ALTMEMPHY megafunction to generate the DQ signals.

For interfaces to DDR SDRAM and DDR2 SDRAM, the Arria GX DDR IOE structure requires you to invert the incoming DQS signal to ensure proper data transfer. By default, the ALTMEMPHY megafunction adds the inverter to the `inclock` port when it generates DQ blocks. As shown in Figure 7–12 on page 7–20, the `inclock` signal's rising edge clocks the $A_I$ register, `inclock` signal's falling edge clocks the $B_I$ register, and latch $C_I$ is opened when `inclock` is 1. In a DDR memory read operation, the last data coincides with DQS being low. If you do not invert the DQS pin, you will not get this last data as the latch does not open until the next rising edge of the DQS signal.

Figure 7–16 shows waveforms of the circuit shown in Figure 7–14 on page 7–22.

The first set of waveforms in Figure 7–16 shows the edge-aligned relationship between the DQ and DQS signals at the Arria GX device pins. The second set of waveforms in Figure 7–16 shows what happens if the shifted DQS signal is not inverted; the last data, $D_n$, does not get latched into the logic array as DQS goes to tri-state after the read postamble time. The third set of waveforms in Figure 7–16 shows a proper read operation with the DQS signal inverted after the 90° shift; the last data, $D_n$, does get latched. In this case the outputs of register $A_I$ and latch $C_I$, which correspond to `dataout_h` and `dataout_l` ports, are now switched because of the DQS inversion. Register $A_I$, register $B_I$, and latch $C_I$ refer to the nomenclature in Figure 7–14 on page 7–22.

*Figure 7–16. DQ Captures with Non-Inverted and Inverted Shifted DQS*

**DQ & DQS Signals**

DQ at the pin $D_{n-1}$ $D_n$

DQS at the pin

**Shifted DQS Signal is Not Inverted**

DQS shifted by 90°

Output of register $A_1$ (dataout_h) $D_{n-1}$

Output of register $B_1$ $D_{n-2}$ $D_n$

Output of latch $C_1$ (dataout_l) $D_{n-2}$
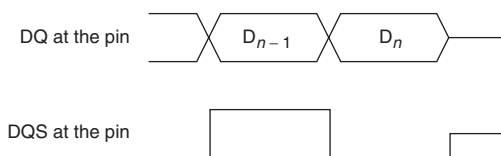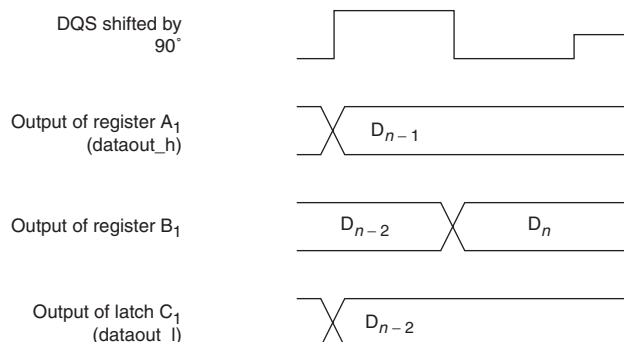
**Shifted DQS Signal is Inverted**

DQS inverted and shifted by 90°

Output of register $A_1$ (dataout_h) $D_{n-2}$ $D_n$

Output of register $B_1$ $D_{n-1}$

Output of latch $C_1$ (dataout_l) $D_{n-3}$ $D_{n-1}$
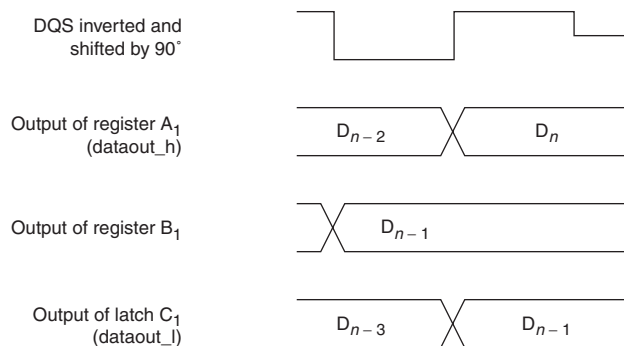
### PLL

When using the Arria GX top and bottom I/O banks (I/O banks 3, 4, 7, or 8) to interface with a DDR memory, at least one PLL with two outputs is needed to generate the system clock and write clock. The system clock generates the DQS write signals, commands, and addresses. The write clock is either shifted by $-90^\circ$ or $90^\circ$ from the system clock and is used to generate the DQ signals during writes.

For DDR and DDR2 SDRAM interfaces above 200 MHz, Altera also recommends a second read PLL to help ease resynchronization.

## Conclusion

Arria GX devices support SDR SDRAM, DDR SDRAM, and DDR2 SDRAM external memories. Arria GX devices feature high-speed interfaces that transfer data between external memory devices at up to 233 MHz/466 Mbps. DQS phase-shift circuitry and DQS logic blocks within Arria GX devices allow you to fine-tune the phase shifts for the input clocks or strobes to properly align clock edges as needed to capture data.

## Referenced Documents

This chapter references the following documents:

- *ALTMEMPHY Megafunction User Guide*
- *AN 327: Interfacing DDR SDRAM with Stratix II Devices*
- *AN 328: Interfacing DDR2 SDRAM with Stratix II Devices*
- *DC & Switching Characteristics* chapter in volume 1 of the *Arria GX Device Handbook*

## Document Revision History

Table 7–5 shows the revision history for this chapter.

*Table 7–5. Document Revision History*

| Date and Document Version | Changes Made | Summary of Changes |
|---|---|---|
| May 2008 v1.2 | Updated the "DLL" section. | — |
| | Minor text edits. | — |
| August 2007 v1.1 | Added the "Referenced Documents" section. | — |
| May 2007 v1.0 | Initial Release | — |

# Section IV. I/O Standards

This section provides information on Arria™ GX single-ended, voltage-referenced, and differential I/O standards.

This section contains the following chapters:

■ Chapter 8, Selectable I/O Standards in Arria GX Devices

■ Chapter 9, High-Speed Differential I/O Interfaces with DPA in Arria GX Devices

## Revision History

Refer to each chapter for its own specific revision history. For information on when each chapter was updated, refer to the Chapter Revision Dates section, which appears in the full handbook.

# 8. Selectable I/O Standards in Arria GX Devices

## Introduction

This chapter provides guidelines for using industry I/O standards in Arria™ GX devices, including:

- I/O features
- I/O standards
- External memory interfaces
- I/O banks
- Design considerations

This chapter contains the following sections:

## Arria GX I/O Features

Arria GX devices contain an abundance of adaptive logic modules (ALMs), embedded memory, high-bandwidth digital signal processing (DSP) blocks, and extensive routing resources, all of which can operate at very high core speed.

The Arria GX device's I/O structure is designed to ensure that these internal capabilities are fully utilized. There are numerous I/O features to assist in high-speed data transfer into and out of the device including:

- Single-ended, non-voltage-referenced and voltage-referenced I/O standards
- High-speed differential I/O standards featuring serializer/deserializer (SERDES), dynamic phase alignment (DPA), capable of 840 megabit per second (Mbps) performance for low-voltage differential signaling (LVDS), Hypertransport technology, high-speed transceiver logic (HSTL), stub-series terminated logic (SSTL), and LVPECL

☞ HSTL, SSTL, and LVPECL I/O standards are used only for PLL clock inputs and outputs in differential mode.

■ Double data rate (DDR) I/O pins
■ Programmable output drive strength for voltage-referenced and non-voltage-referenced single-ended I/O standards
■ Programmable bus-hold
■ Programmable pull-up resistor
■ Open-drain output
■ On-chip series termination
■ On-chip differential termination
■ Peripheral component interconnect (PCI) clamping diode
■ Hot socketing

For a detailed description of each I/O feature, refer to the *Arria GX Architecture* chapter in volume 1 of the *Arria GX Device Handbook.*

## Arria GX I/O Standards Support

Arria GX devices support a wide range of industry I/O standards. Table 8–1 shows which I/O standards Arria GX devices support as well as typical applications.

*Table 8–1. Arria GX I/O Standard Applications  (Part 1 of 2)*

| I/O Standard | Application |
|---|---|
| LVTTL | General purpose |
| LVCMOS | General purpose |
| 2.5 V | General purpose |
| 1.8 V | General purpose |
| 1.5 V | General purpose |
| 3.3-V PCI | PC and embedded system |
| 3.3-V PCI-X | PC and embedded system |
| SSTL-2 Class I | DDR SDRAM |
| SSTL-2 Class II | DDR SDRAM |
| SSTL-18 Class I | DDR2 SDRAM |
| SSTL-18 Class II | DDR2 SDRAM |
| 1.8-V HSTL Class I | SRAM interfaces |
| 1.8-V HSTL Class II | SRAM interfaces |
| 1.5-V HSTL Class I | SRAM interfaces |
| 1.5-V HSTL Class II | SRAM interfaces |
| 1.2-V HSTL | General purpose |
| Differential SSTL-2 Class I | DDR SDRAM |
| Differential SSTL-2 Class II | DDR SDRAM |
| Differential SSTL-18 Class I | DDR2 SDRAM |

| *Table 8–1. Arria GX I/O Standard Applications  (Part 2 of 2)* | |
|---|---|
| **I/O Standard** | **Application** |
| Differential SSTL-18 Class II | DDR2 SDRAM |
| 1.8-V differential HSTL Class I | Clock interfaces |
| 1.8-V differential HSTL Class II | Clock interfaces |
| 1.5-V differential HSTL Class I | Clock interfaces |
| 1.5-V differential HSTL Class II | Clock interfaces |
| LVDS | High-speed communications |
| HyperTransport technology | PCB interfaces |
| Differential LVPECL | Video graphics and clock distribution |

## Single-Ended I/O Standards

In non-voltage-referenced single-ended I/O standards, the voltage at the input must be above a set voltage to be considered "on" (high, or logic value 1) or below another voltage to be considered "off" (low, or logic value 0). Voltages between the limits are undefined logically, and may fall into either a logic value 0 or 1. The non-voltage-referenced single-ended I/O standards supported by Arria GX devices are:

- Low-voltage transistor-transistor logic (LVTTL)
- Low-voltage complementary metal-oxide semiconductor (LVCMOS)
- 1.5 V
- 1.8 V
- 2.5 V
- 3.3-V PCI
- 3.3-V PCI-X

Voltage-referenced, single-ended I/O standards provide faster data rates. These standards use a constant reference voltage at the input levels. The incoming signals are compared with this constant voltage and the difference between the two defines "on" and "off" states.

☞      Arria GX devices support SSTL and HSTL voltage-referenced I/O standards.

### LVTTL

The LVTTL standard is formulated under the EIA/JEDEC Standard, JESD8-B (Revision of JESD8-A): Interface Standard for Nominal 3-V/3.3-V Supply Digital Integrated Circuits.

The standard defines DC interface parameters for digital circuits operating from a 3.0- or 3.3-V power supply and driving or being driven by LVTTL-compatible devices. The 3.3-V LVTTL standard is a general-purpose, single-ended standard used for 3.3-V applications. This I/O standard does not require input reference voltages ($V_{REF}$) or termination voltages ($V_{TT}$).

☞     Arria GX devices support both input and output levels for 3.3-V LVTTL operation.

Arria GX devices support a $V_{CCIO}$ voltage level of 3.3 V ±5% as specified as the narrow range for the voltage supply by the EIA/JEDEC standard.

### LVCMOS

The LVCMOS standard is formulated under the EIA/JEDEC Standard, JESD8-B (Revision of JESD8-A): Interface Standard for Nominal 3-V/3.3-V Supply Digital Integrated Circuits.

The standard defines DC interface parameters for digital circuits operating from a 3.0- or 3.3-V power supply and driving or being driven by LVCMOS-compatible devices. The 3.3-V LVCMOS I/O standard is a general-purpose, single-ended standard used for 3.3-V applications. While LVCMOS has its own output specification, it specifies the same input voltage requirements as LVTTL. These I/O standards do not require $V_{REF}$ or $V_{TT}$.

☞     Arria GX devices support both input and output levels for 3.3-V LVCMOS operation.

Arria GX devices support a $V_{CCIO}$ voltage level of 3.3 V ±5% as specified as the narrow range for the voltage supply by the EIA/JEDEC standard.

### 1.5 V

The 1.5-V I/O standard is formulated under the EIA/JEDEC Standard, JESD8-11: 1.5-V ±0.1-V (Normal Range) and 0.9-V – 1.6-V (Wide Range) Power Supply Voltage and Interface Standard for Non-Terminated Digital Integrated Circuit.

The standard defines the DC interface parameters for high-speed, low-voltage, non-terminated digital circuits driving or being driven by other 1.5-V devices. This standard is a general-purpose, single-ended standard used for 1.5-V applications. It does not require the use of a $V_{REF}$ or a $V_{TT}$.

🖙 Arria GX devices support both input and output levels for 1.5-V operation $V_{CCIO}$ voltage level support of 1.8 V ±5%, which is narrower than defined in the Normal Range of the EIA/JEDEC standard.

### 1.8 V

The 1.8-V I/O standard is formulated under the EIA/JEDEC Standard, EIA/JESD8-7: 1.8-V ±0.15-V (Normal Range), and 1.2-V – 1.95-V (Wide Range) Power Supply Voltage and Interface Standard for Non-Terminated Digital Integrated Circuit.

The standard defines the DC interface parameters for high-speed, low-voltage, non-terminated digital circuits driving or being driven by other 1.8-V devices. This standard is a general-purpose, single-ended standard used for 1.8-V applications. It does not require the use of a $V_{REF}$ or a $V_{TT}$.

🖙 Arria GX devices support both input and output levels for 1.8-V operation with $V_{CCIO}$ voltage level support of 1.8 V ±5%, which is narrower than defined in the Normal Range of the EIA/JEDEC standard.

### 2.5 V

The 2.5-V I/O standard is formulated under the EIA/JEDEC Standard, EIA/JESD8-5: 2.5-V ± 0.2-V (Normal Range), and 1.8-V – 2.7-V (Wide Range) Power Supply Voltage and Interface Standard for Non-Terminated Digital Integrated Circuit.

The standard defines the DC interface parameters for high-speed, low-voltage, non-terminated digital circuits driving or being driven by other 2.5-V devices. This standard is a general-purpose, single-ended standard used for 2.5-V applications. It does not require the use of a $V_{REF}$ or a $V_{TT}$.

🖙 Arria GX devices support both input and output levels for 2.5-V operation with $V_{CCIO}$ voltage level support of 2.5 V ±5%, which is narrower than defined in the Normal Range of the EIA/JEDEC standard.

### 3.3-V PCI

The 3.3-V PCI I/O standard is formulated under the PCI Local Bus Specification Revision 2.2 developed by the PCI Special Interest Group (SIG).

The PCI local bus specification is used for applications that interface to the PCI local bus, which provides a processor-independent data path between highly integrated peripheral controller components, peripheral add-in boards, and processor/memory systems. The conventional PCI specification revision 2.2 defines the PCI hardware environment including the protocol, electrical, mechanical, and configuration specifications for the PCI devices and expansion boards. This standard requires 3.3-V $V_{CCIO}$. Arria GX devices are fully compliant with the 3.3-V PCI Local Bus Specification Revision 2.2 and meet 64-bit/33-MHz operating frequency and timing requirements.

☞ The 3.3-V PCI standard does not require input reference voltages or board terminations. Arria GX devices support both input and output levels.

### 3.3-V PCI-X

The 3.3-V PCI-X I/O standard is formulated under the PCI-X Local Bus Specification Revision 1.0a developed by the PCI SIG.

The PCI-X 1.0 standard is used for applications that interface to the PCI local bus. The standard enables the design of systems and devices that operate at clock speeds up to 133 MHz, or 1 Gbps for a 64-bit bus. The PCI-X 1.0 protocol enhancements enable devices to operate much more efficiently, providing more usable bandwidth at any clock frequency. By using the PCI-X 1.0 standard, you can design devices to meet PCI-X 1.0 requirements and operate as conventional 33- and 66-MHz PCI devices when installed in those systems. This standard requires 3.3-V $V_{CCIO}$. Arria GX devices are fully compliant with the 3.3-V PCI-X Specification Revision 1.0a and meet the 133-MHz operating frequency and timing requirements. The 3.3-V PCI-X standard does not require input reference voltages or board terminations.

☞ Arria GX devices support both input and output levels operation.

### SSTL-2 Class I & SSTL-2 Class II

The 2.5-V SSTL-2 standard is formulated under the JEDEC Standard, JESD8-A: Stub Series Terminated Logic for 2.5-V (SSTL_2).

The SSTL-2 I/O standard is a 2.5-V memory bus standard used for applications such as high-speed DDR SDRAM interfaces. This standard defines the input and output specifications for devices that operate in the SSTL-2 logic switching range of 0.0 to 2.5 V. This standard improves operation in conditions where a bus must be isolated from large stubs. SSTL-2 requires a 1.25-V $V_{REF}$ and a 1.25-V $V_{TT}$ to which the series and termination resistors are connected (Figures 8–1 and 8–2).

☞ Arria GX devices support both input and output levels operation.

*Figure 8–1. 2.5-V SSTL Class I Termination*
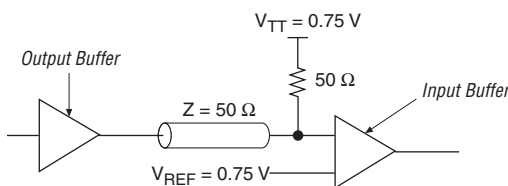


*Figure 8–2. 2.5-V SSTL Class II Termination*
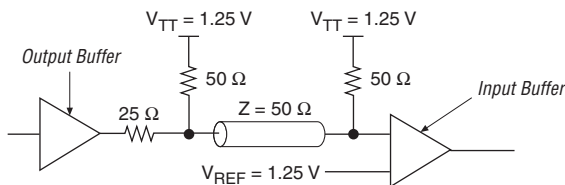


### SSTL-18 Class I & SSTL-18 Class II

The 1.8-V SSTL-18 standard is formulated under the JEDEC Standard, JESD8-15: Stub Series Terminated Logic for 1.8-V (SSTL_18).

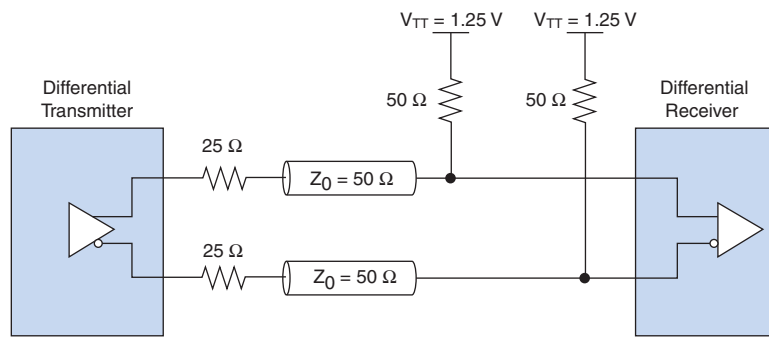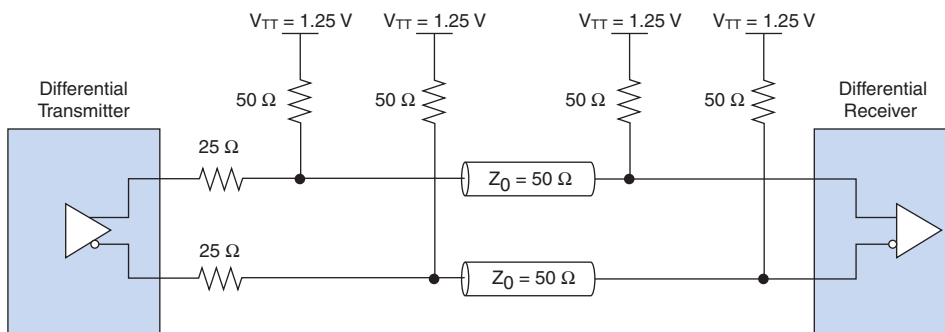The SSTL-18 I/O standard is a 1.8-V memory bus standard used for applications such as high-speed DDR2 SDRAM interfaces. This standard is similar to SSTL-2 and defines input and output specifications for

devices that are designed to operate in the SSTL-18 logic switching range 0.0 to 1.8 V. SSTL-18 requires a 0.9-V $V_{REF}$ and a 0.9-V $V_{TT}$ to which the series and termination resistors are connected.

There are no class definitions for the SSTL-18 standard in the JEDEC specification. The specification of this I/O standard is based on an environment that consists of both series and parallel terminating resistors. Altera® provides solutions to two derived applications in the JEDEC specification, and names them Class I and Class II to be consistent with other SSTL standards. Figures 8–3 and 8–4 show SSTL-18 Class I and II termination, respectively.

☞ Arria GX devices support both input and output levels operation.

*Figure 8–3. Figure9–3.1.8-V SSTL Class I Termination*
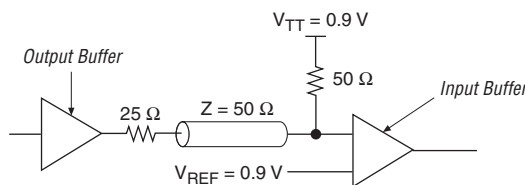


*Figure 8–4. Figure9–4.1.8-V SSTL Class II Termination*
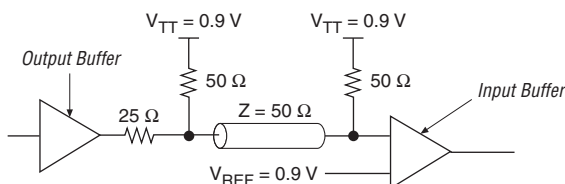


*1.8-V HSTL Class I & 1.8-V HSTL Class II*

The HSTL standard is a technology-independent I/O standard developed by JEDEC to provide voltage scalability. It is used for applications designed to operate in the 0.0- to 1.8-V HSTL logic switching range.

Although JEDEC specifies a maximum $V_{CCIO}$ value of 1.6 V, there are various memory chip vendors with HSTL standards that require a $V_{CCIO}$ of 1.8 V. Arria GX devices support interfaces to chips with $V_{CCIO}$ of 1.8 V

for HSTL. Figures 8–5 and 8–6 show the nominal $V_{REF}$ and $V_{TT}$ required to track the higher value of $V_{CCIO}$. The value of $V_{REF}$ is selected to provide optimum noise margin in the system.

☞    Arria GX devices support both input and output levels operation.

*Figure 8–5. 1.8-V HSTL Class I Termination*
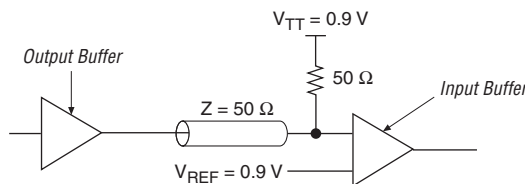


*Figure 8–6. 1.8-V HSTL Class II Termination*
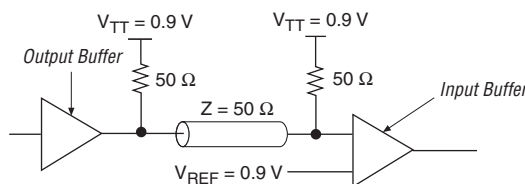


*1.5-V HSTL Class I & 1.5-V HSTL Class II*

The 1.5-V HSTL standard is formulated under the EIA/JEDEC Standard, EIA/JESD8-6: A 1.5-V Output Buffer Supply Voltage Based Interface Standard for Digital Integrated Circuits.

The 1.5-V HSTL I/O standard is used for applications designed to operate in the 0.0- to 1.5-V HSTL logic nominal switching range. This standard defines single-ended input and output specifications for all HSTL-compliant digital integrated circuits. The 1.5-V HSTL I/O standard in Arria GX devices are compatible with the 1.8-V HSTL I/O standard in APEX 20KE, APEX20KC, and in Arria GX devices themselves because the input and output voltage thresholds are compatible (Figures 8–7 and 8–8).

☞    Arria GX devices support both input and output levels with $V_{REF}$ and $V_{TT}$.

*Figure 8–7. 1.5-V HSTL Class I Termination*
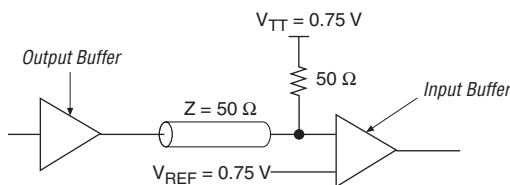


*Figure 8–8. 1.5-V HSTL Class II Termination*
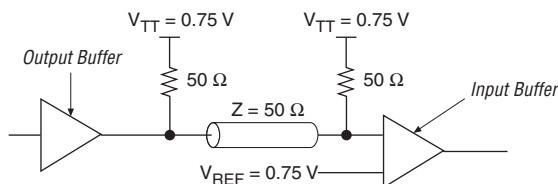


*1.2-V HSTL*

Although there is no EIA/JEDEC standard available for the 1.2-V HSTL standard, Altera supports it for applications that operate in the 0.0 to 1.2-V HSTL logic nominal switching range. 1.2-V HSTL can be terminated through series on-chip termination (OCT). Figure 8–9 shows the termination scheme.

*Figure 8–9. 1.2-V HSTL Termination*
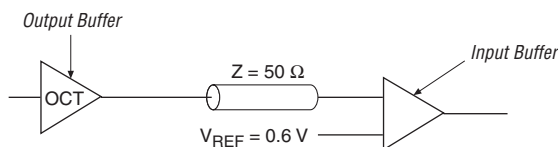


## Differential I/O Standards

Differential I/O standards are used to achieve even faster data rates with higher noise immunity. Apart from LVDS, LVPECL, and HyperTransport technology, Arria GX devices also support differential versions of SSTL and HSTL standards.

For detailed information about differential I/O standards, refer to the *High-Speed Differential I/O Interfaces with DPA in Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook.*

### Differential SSTL-2 Class I & Differential SSTL-2 Class II

The 2.5-V differential SSTL-2 standard is formulated under the JEDEC Standard, JESD8-9A: Stub Series Terminated Logic for 2.5-V (SSTL_2).

This I/O standard is a 2.5-V standard used for applications such as high-speed DDR SDRAM clock interfaces. This standard supports differential signals in systems using the SSTL-2 standard and supplements the SSTL-2 standard for differential clocks. Arria GX devices support both input and output levels. Figures 8–10 and 8–11 show details about differential SSTL-2 termination.

☞ Arria GX devices support differential SSTL-2 I/O standards in pseudo-differential mode, which is implemented by using two SSTL-2 single-ended buffers.

The Quartus® II software only supports pseudo-differential standards on the INCLK, FBIN, and EXTCLK ports of enhanced PLL, as well as on DQS pins when DQS megafunction (ALTDQS, Bidirectional Data Strobe) is used. Two single-ended output buffers are automatically programmed to have opposite polarity so as to implement a pseudo-differential output. A proper $V_{REF}$ voltage is required for the two single-ended input buffers to implement a pseudo-differential input. In this case, only the positive polarity input is used in the speed path while the negative input is not connected internally. In other words, only the non-inverted pin is required to be specified in your design, while the Quartus II software automatically generates the inverted pin for you.

Although the Quartus II software does not support pseudo-differential SSTL-2 I/O standards on the left I/O banks, you can implement these standards on these banks. You need to create two pins in the designs and configure the pins with single-ended SSTL-2 standards. However, this is limited only to pins that support the differential pin-pair I/O function and is dependent on the single-ended SSTL-2 standards support at these banks.
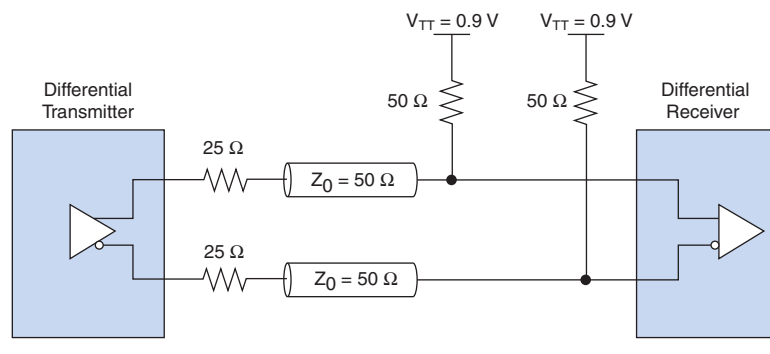
*Figure 8–10. Differential SSTL-2 Class I Termination*



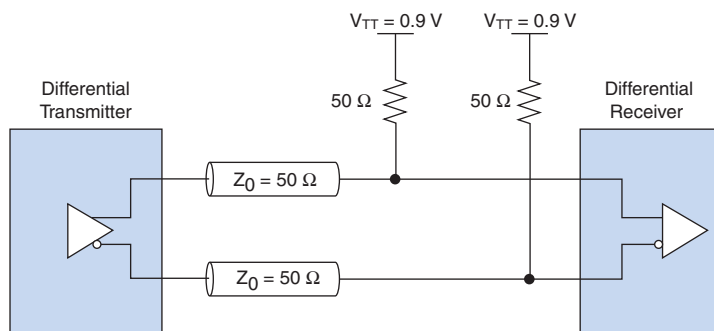*Figure 8–11. Differential SSTL-2 Class II Termination*



### Differential SSTL-18 Class I & Differential SSTL-18 Class II

The 1.8-V differential SSTL-18 standard is formulated under the JEDEC Standard, JESD8-15: Stub Series Terminated Logic for 1.8-V (SSTL_18).

The differential SSTL-18 I/O standard is a 1.8-V standard used for applications such as high-speed DDR2 SDRAM interfaces. This standard supports differential signals in systems using the SSTL-18 standard and supplements the SSTL-18 standard for differential clocks.

☞ Arria GX devices support both input and output levels operation.

Figures 8–12 and 8–13 show details about differential SSTL-18 termination. Arria GX devices support differential SSTL-18 I/O standards in pseudo-differential mode, which is implemented by using two SSTL-18 single-ended buffers.

The Quartus II software only supports pseudo-differential standards on the INCLK, FBIN, and EXTCLK ports of enhanced PLL, as well as on DQS pins when DQS megafunction (ALTDQS, Bidirectional Data Strobe) is used. Two single-ended output buffers are automatically programmed to have opposite polarity so as to implement a pseudo-differential output. A proper $V_{REF}$ voltage is required for the two single-ended input buffers to implement a pseudo-differential input. In this case, only the positive polarity input is used in the speed path while the negative input is not connected internally. In other words, only the non-inverted pin is required to be specified in your design, while the Quartus II software automatically generates the inverted pin for you.

Although the Quartus II software does not support pseudo-differential SSTL-18 I/O standards on the left and I/O banks, you can implement these standards at these banks. You need to create two pins in the designs and configure the pins with single-ended SSTL-18 standards. However, this is limited only to pins that support the differential pin-pair I/O function and is dependent on the single-ended SSTL-18 standards support at these banks.

*Figure 8–12. Differential SSTL-18 Class I Termination*

*Figure 8–13. Differential SSTL-18 Class II Termination*
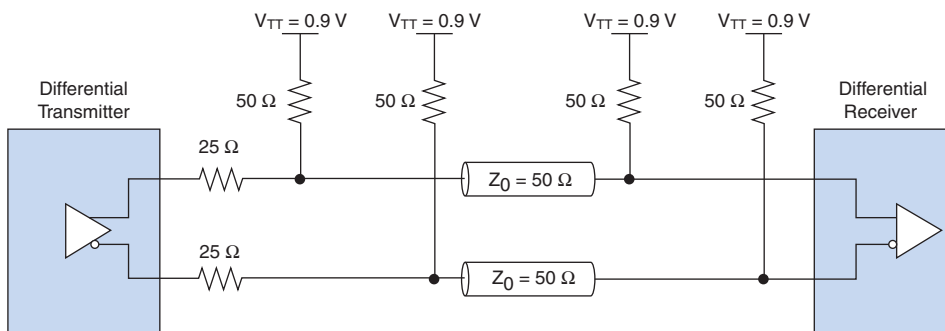


### 1.8-V Differential HSTL Class I & 1.8-V Differential HSTL Class II

The 1.8-V differential HSTL specification is the same as the 1.8-V single-ended HSTL specification. It is used for applications designed to operate in the 0.0- to 1.8-V HSTL logic switching range such as QDR memory clock interfaces. Arria GX devices support both input and output levels operation. Figures 8–14 and 8–15 show details about 1.8-V differential HSTL termination.

Arria GX devices support 1.8-V differential HSTL I/O standards in pseudo-differential mode, which is implemented by using two 1.8-V HSTL single-ended buffers.

The Quartus II software only supports pseudo-differential standards on the INCLK, FBIN, and EXTCLK ports of enhanced PLL, as well as on DQS pins when DQS megafunction (ALTDQS, Bidirectional Data Strobe) is used. Two single-ended output buffers are automatically programmed to have opposite polarity so as to implement a pseudo-differential output. A proper $V_{REF}$ voltage is required for the two single-ended input buffers to implement a pseudo-differential input. In this case, only the positive polarity input is used in the speed path while the negative input is not connected internally. In other words, only the non-inverted pin is required to be specified in your design, while the Quartus II software automatically generates the inverted pin for you.

Although the Quartus II software does not support 1.8-V pseudo-differential HSTL I/O standards on left I/O banks, you can implement these standards at these banks. You need to create two pins in the designs and configure the pins with single-ended 1.8-V HSTL standards. However, this is limited only to pins that support the differential pin-pair I/O function and is dependent on the single-ended 1.8-V HSTL standards support at these banks.

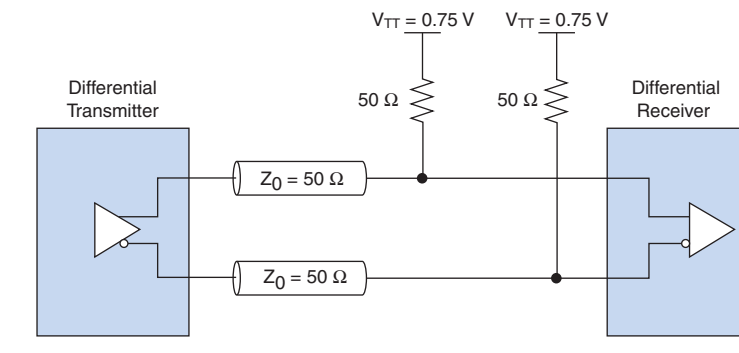*Figure 8–14. 1.8-V Differential HSTL Class I Termination*



*Figure 8–15. 1.8-V Differential HSTL Class II Termination*
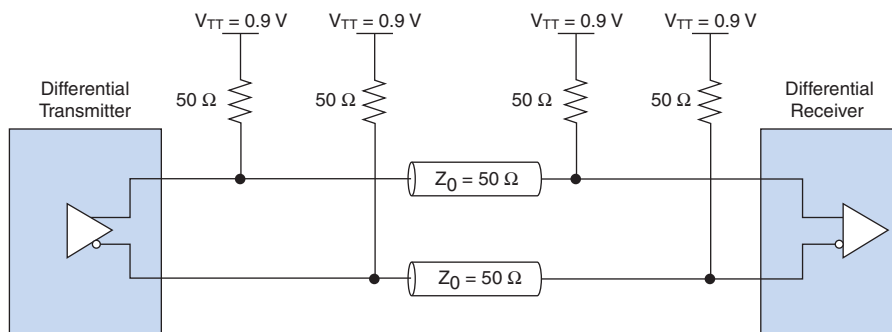


### 1.5-V Differential HSTL Class I & 1.5-V Differential HSTL Class II

The 1.5-V differential HSTL standard is formulated under the EIA/JEDEC Standard, EIA/JESD8-6: A 1.5-V Output Buffer Supply Voltage Based Interface Standard for Digital Integrated Circuits.

The 1.5-V differential HSTL specification is the same as the 1.5-V single-ended HSTL specification. It is used for applications designed to operate in the 0.0- to 1.5-V HSTL logic switching range, such as QDR memory clock interfaces. Arria GX devices support both input and output levels operation. Figures 8–16 and 8–17 show details on the 1.5-V differential HSTL termination.

Arria GX devices support 1.5-V differential HSTL I/O standards in pseudo-differential mode, which is implemented by using two 1.5-V HSTL single-ended buffers.

The Quartus II software only supports pseudo-differential standards on the INCLK, FBIN, and EXTCLK ports of enhanced PLL, as well as on DQS pins when DQS megafunction (ALTDQS, Bidirectional Data Strobe) is used. Two single-ended output buffers are automatically programmed to have opposite polarity so as to implement a pseudo-differential output. A proper $V_{REF}$ voltage is required for the two single-ended input buffers to implement a pseudo-differential input. In this case, only the positive polarity input is used in the speed path while the negative input is not connected internally. In other words, only the non-inverted pin is required to be specified in your design, while the Quartus II software automatically generates the inverted pin for you.

Although the Quartus II software does not support 1.5-V pseudo-differential HSTL I/O standards on left I/O banks, you can implement these standards at these banks. You need to create two pins in the designs and configure the pins with single-ended 1.5-V HSTL standards. However, this is limited only to pins that support the differential pin-pair I/O function and is dependent on the single-ended 1.5-V HSTL standards support at these banks.

*Figure 8–16. 1.5-V Differential HSTL Class I Termination*

*Figure 8–17. 1.5-V Differential HSTL Class II Termination*
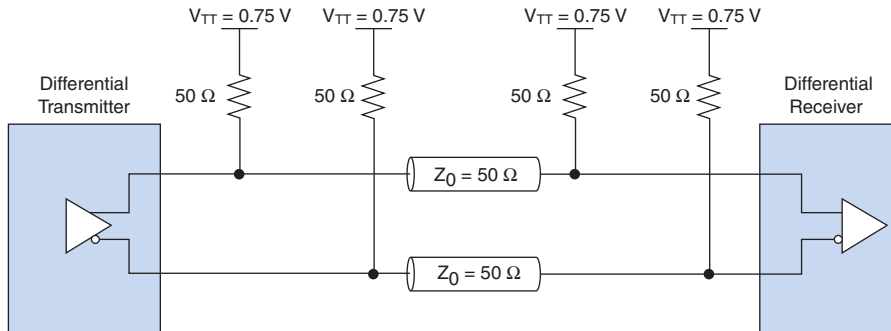


### LVDS

The LVDS standard is formulated under the ANSI/TIA/EIA Standard, ANSI/TIA/EIA-644: Electrical Characteristics of Low Voltage Differential Signaling Interface Circuits.

The LVDS I/O standard is a differential high-speed, low-voltage swing, low-power, general-purpose I/O interface standard. In Arria GX devices, the LVDS I/O standard requires a 2.5-V $V_{CCIO}$ level. However, LVDS clock output pins in the top and bottom I/O banks require a 3.3-V $V_{CCIO}$ level. This standard is used in applications requiring high-bandwidth data transfer, backplane drivers, and clock distribution. The ANSI/TIA/EIA-644 standard specifies LVDS transmitters and receivers capable of operating at recommended maximum data signaling rates of 655 Mbps. However, devices can operate at slower speeds if needed, and there is a theoretical maximum of 1.923 Gbps. Arria GX devices are capable of running at a maximum data rate of 840 Mbps and still meet the ANSI/TIA/EIA-644 standard.

Because of the low-voltage swing of the LVDS I/O standard, the electromagnetic interference (EMI) effects are much smaller than complementary metal-oxide semiconductor (CMOS), transistor-to-transistor logic (TTL), and positive (or pseudo) emitter coupled logic (PECL). This low EMI makes LVDS ideal for applications with low EMI requirements or noise immunity requirements. The LVDS standard does not require an input reference voltage. However, it does require a 100-Ω termination resistor between the two signals at the input buffer. Arria GX devices provide an optional 100-Ω differential LVDS termination resistor in the device using on-chip differential termination. Arria GX devices support both input and output levels operation.

### Differential LVPECL

The low-voltage positive (or pseudo) emitter coupled logic (LVPECL) standard is a differential interface standard requiring a 3.3-V $V_{CCIO}$. The standard is used in applications involving video graphics, telecommunications, data communications, and clock distribution. The high-speed, low-voltage swing LVPECL I/O standard uses a positive power supply and is similar to LVDS. However, LVPECL has a larger differential output voltage swing than LVDS. The LVPECL standard does not require an input reference voltage, but it does require a 100-$\Omega$ termination resistor between the two signals at the input buffer. Figures 8–18 and 8–19 show two alternate termination schemes for LVPECL.

☞ Arria GX devices support both input and output levels operation.

*Figure 8–18. LVPECL DC Coupled Termination*
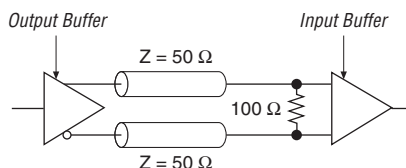


*Figure 8–19. LVPECL AC Coupled Termination*
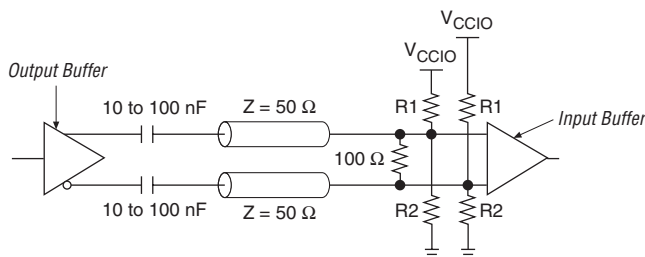
*HyperTransport Technology*

The HyperTransport standard is formulated by the HyperTransport Consortium.

The HyperTransport I/O standard is a differential high-speed, high-performance I/O interface standard requiring a 2.5- or 3.3-V $V_{CCIO}$. This standard is used in applications such as high-performance networking, telecommunications, embedded systems, consumer electronics, and Internet connectivity devices. The HyperTransport I/O standard is a point-to-point standard in which each HyperTransport bus consists of two point-to-point unidirectional links. Each link is 2 to 32 bits.

The HyperTransport standard does not require an input reference voltage. However, it does require a 100-Ω termination resistor between the two signals at the input buffer. Figure 8–20 shows HyperTransport termination. Arria GX devices include an optional 100-Ω differential HyperTransport termination resistor in the device using on-chip differential termination.

☞ Arria GX devices support both input and output levels operation.

*Figure 8–20. HyperTransport Termination*
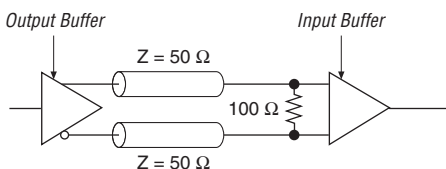


**Arria GX External Memory Interfaces**

The increasing demand for higher-performance data processing systems often requires memory-intensive applications. Arria GX devices can interface with many types of external memory.

👣 Refer to the *External Memory Interfaces in Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook* for more information about the external memory interface support in Arria GX devices.

# Arria GX I/O Banks

Arria GX devices have six general I/O banks and four enhanced phase-locked loop (PLL) external clock output banks (Figure 8–21). I/O banks 9 through 12 are enhanced PLL external clock output banks located on the top and bottom of the device.

*Figure 8–21. Arria GX I/O Banks* Note (1), (2), (3), (4), (5), (6)
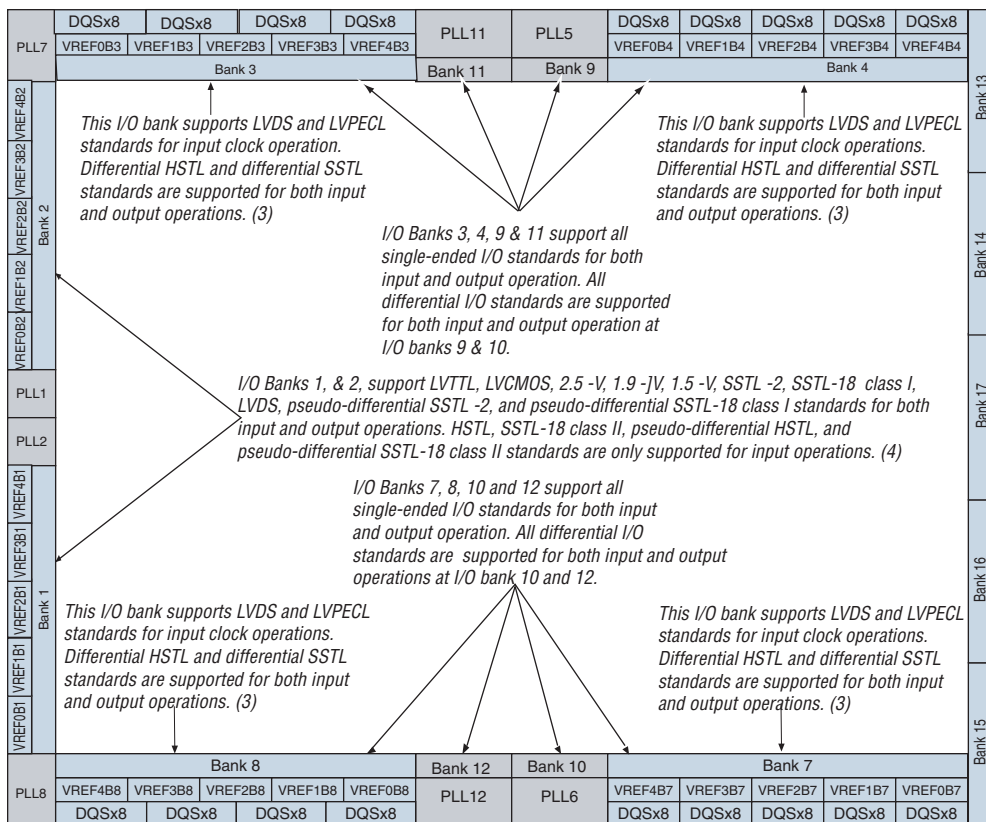


*Notes to Figure 8–21:*
(1) Figure 8–21 is a top view of the silicon die which corresponds to a reverse view for flip-chip packages. It is a graphical representation only.
(2) Depending on size of the device, different device members have different number of $V_{REF}$ groups. Refer to the pin list and the Quartus II software for exact locations.
(3) Banks 9 through 12 are enhanced PLL external clock output banks.
(4) Horizontal I/O banks feature transceiver and DPA circuitry for high speed differential I/O standards. Refer to the *High-Speed Differential I/O Interfaces in Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*, or the *Arria GX Transceiver User Guide* for more information about differential I/O standards.
(5) Quartus II software does not support differential SSTL and differential HSTL standards at left/right I/O banks. Refer to the "Differential I/O Standards" on page 8–10 if you need to implement these standards at these I/O banks.
(6) PLLs 7, 8, 11, and 12 are available only in EP1AGX50D, EP1AGX60E, and EP1AGX90E devices.

## Programmable I/O Standards

Arria GX device programmable I/O standards deliver high-speed and high-performance solutions in many complex design systems. This section discusses the I/O standard support in the I/O banks of Arria GX devices.

### Regular I/O Pins

Most Arria GX device pins are multi-function pins. These pins support regular inputs and outputs as their primary function, and offer an optional function such as DQS, differential pin-pair, or PLL external clock outputs. For example, you can configure a multi-function pin in the enhanced PLL external clock output bank as a PLL external clock output when it is not used as a regular I/O pin.

☞ I/O pins that reside in PLL banks 9 through 12 are powered by the VCC_PLL<*5, 6, 11, or 12*>_OUT pins, respectively. Some devices/packages do not support PLLs 11 and 12. Therefore, any I/O pins that reside in bank 11 are powered by the VCCIO3 pin, and any I/O pins that reside in bank 12 are powered by the VCCIO8 pin.

Table 8–2 shows the I/O standards supported when a pin is used as a regular I/O pin in the I/O banks of Arria GX devices.

*Table 8–2. Arria GX Regular I/O Standards Support  (Part 1 of 2)*

| I/O Standard | General I/O Bank (1) | | | | | | Enhanced PLL External Clock Output Bank (2) | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **7** | **8** | **9** | **10** | **11** | **12** |
| LVTTL | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| LVCMOS | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 2.5 V | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 1.8 V | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 1.5 V | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 3.3-V PCI | | | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 3.3-V PCI-X | | | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| SSTL-2 Class I | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| SSTL-2 Class II | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| SSTL-18 Class I | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| SSTL-18 Class II | (3) | (3) | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| 1.8-V HSTL Class I | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |

| *Table 8–2. Arria GX Regular I/O Standards Support   (Part 2 of 2)* | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **I/O Standard** | **General I/O Bank** *(1)* | | | | | | **Enhanced PLL External Clock Output Bank** *(2)* | | | |
| | **1** | **2** | **3** | **4** | **7** | **8** | **9** | **10** | **11** | **12** |
| 1.8-V HSTL Class II | *(3)* | *(3)* | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 1.5-V HSTL Class I | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 1.5-V HSTL Class II | *(3)* | *(3)* | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 1.2-V HSTL | | | | ✓ | ✓ | ✓ | | | | |
| Differential SSTL-2 Class I | *(4)* | *(4)* | *(5)* | *(5)* | *(5)* | *(5)* | | | | |
| Differential SSTL-2 Class II | *(4)* | *(4)* | *(5)* | *(5)* | *(5)* | *(5)* | | | | |
| Differential SSTL-18 Class I | *(4)* | *(4)* | *(5)* | *(5)* | *(5)* | *(5)* | | | | |
| Differential SSTL-18 Class II | *(4)* | *(4)* | *(5)* | *(5)* | *(5)* | *(5)* | | | | |
| 1.8-V differential HSTL Class I | *(4)* | *(4)* | *(5)* | *(5)* | *(5)* | *(5)* | | | | |
| 1.8-V differential HSTL Class II | *(4)* | *(4)* | *(5)* | *(5)* | *(5)* | *(5)* | | | | |
| 1.5-V differential HSTL Class I | *(4)* | *(4)* | *(5)* | *(5)* | *(5)* | *(5)* | | | | |
| 1.5-V differential HSTL Class II | *(4)* | *(4)* | *(5)* | *(5)* | *(5)* | *(5)* | | | | |
| LVDS | ✓ | ✓ | *(6)* | *(6)* | *(6)* | *(6)* | ✓ | ✓ | ✓ | ✓ |
| HyperTransport technology | ✓ | ✓ | | | | | | | | |
| Differential LVPECL | | | *(6)* | *(6)* | *(6)* | *(6)* | ✓ | ✓ | ✓ | ✓ |

*Notes to Table 8–2:*
(1)   Banks 5 and 6 are not available in Arria GX Devices.
(2)   A mixture of single-ended and differential I/O standards is not allowed in enhanced PLL external clock output bank.
(3)   This I/O standard is only supported for the input operation in this I/O bank.
(4)   Although the Quartus II software does not support pseudo-differential SSTL-2 I/O standards on the left and right I/O banks, you can implement these standards at these banks. Refer to "Differential I/O Standards" on page 8–10 for details.
(5)   This I/O standard is supported for both input and output operations for pins that support the DQS function. Refer to "Differential I/O Standards" on page 8–10 for details.
(6)   This I/O standard is only supported for the input operation for pins that support PLL INCLK function in this I/O bank.

### Clock I/O Pins

The PLL clock I/O pins consist of clock inputs (INCLK), external feedback inputs (FBIN), and external clock outputs (EXTCLK). Clock inputs are located on the left I/O banks (banks 1 and 2) to support fast PLLs, and at the top and bottom I/O banks (banks 3, 4, 7, and 8) to support enhanced PLLs. Both external clock outputs and external feedback inputs are located at enhanced PLL external clock output banks (banks 9, 10, 11, and 12) to support enhanced PLLs.

Table 8–3 shows the PLL clock I/O support in the I/O banks of Arria GX devices.

| Table 8–3. I/O Standards Supported for Arria GX PLL Pins | | | | |
|---|---|---|---|---|
| | Enhanced PLL (1) | | | Fast PLL |
| I/O Standard | Input | | Output | Input |
| | INCLK | FBIN | EXTCLK | INCLK |
| LVTTL | ✓ | ✓ | ✓ | ✓ |
| LVCMOS | ✓ | ✓ | ✓ | ✓ |
| 2.5 V | ✓ | ✓ | ✓ | ✓ |
| 1.8 V | ✓ | ✓ | ✓ | ✓ |
| 1.5 V | ✓ | ✓ | ✓ | ✓ |
| 3.3-V PCI | ✓ | ✓ | ✓ | |
| 3.3-V PCI-X | ✓ | ✓ | ✓ | |
| SSTL-2 Class I | ✓ | ✓ | ✓ | ✓ |
| SSTL-2 Class II | ✓ | ✓ | ✓ | ✓ |
| SSTL-18 Class I | ✓ | ✓ | ✓ | ✓ |
| SSTL-18 Class II | ✓ | ✓ | ✓ | ✓ |
| 1.8-V HSTL Class I | ✓ | ✓ | ✓ | ✓ |
| 1.8-V HSTL Class II | ✓ | ✓ | ✓ | ✓ |
| 1.5-V HSTL Class I | ✓ | ✓ | ✓ | ✓ |
| 1.5-V HSTL Class II | ✓ | ✓ | ✓ | ✓ |
| Differential SSTL-2 Class I | ✓ | ✓ | ✓ | |
| Differential SSTL-2 Class II | ✓ | ✓ | ✓ | |
| Differential SSTL-18 Class I | ✓ | ✓ | ✓ | |
| Differential SSTL-18 Class II | ✓ | ✓ | ✓ | |
| 1.8-V differential HSTL Class I | ✓ | ✓ | ✓ | |
| 1.8-V differential HSTL Class II | ✓ | ✓ | ✓ | |
| 1.5-V differential HSTL Class I | ✓ | ✓ | ✓ | |
| 1.5-V differential HSTL Class II | ✓ | ✓ | ✓ | |
| LVDS | ✓ | ✓ | ✓ | ✓ |
| HyperTransport technology | | | | ✓ |
| Differential LVPECL | ✓ | ✓ | ✓ | |

Note to Table 8–3:

(1) The enhanced PLL external clock output bank does not allow a mixture of both single-ended and differential I/O standards.

For more information, refer to the *PLLs in Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook.*

### Voltage Levels

Arria GX devices specify a range of allowed voltage levels for supported I/O standards. Table 8–4 shows only typical values for input and output $V_{CCIO}$, $V_{REF}$, as well as the board $V_{TT}$.

| *Table 8–4. Arria GX I/O Standards & Voltage Levels  (Part 1 of 2) Note (1)* | | | | | | |
|---|---|---|---|---|---|---|
| **I/O Standard** | **Arria GX** | | | | | |
| | **$V_{CCIO}$ (V)** | | | | **$V_{REF}$ (V)** | **$V_{TT}$ (V)** |
| | **Input Operation** | | **Output Operation** | | | |
| | **Top & Bottom I/O Banks** | **Left & Right I/O Banks** *(3)* | **Top & Bottom I/O Banks** | **Left & Right I/O Banks** *(3)* | **Input** | **Termination** |
| LVTTL | 3.3/2.5 | 3.3/2.5 | 3.3 | 3.3 | NA | NA |
| LVCMOS | 3.3/2.5 | 3.3/2.5 | 3.3 | 3.3 | NA | NA |
| 2.5 V | 3.3/2.5 | 3.3/2.5 | 2.5 | 2.5 | NA | NA |
| 1.8 V | 1.8/1.5 | 1.8/1.5 | 1.8 | 1.8 | NA | NA |
| 1.5 V | 1.8/1.5 | 1.8/1.5 | 1.5 | 1.5 | NA | NA |
| 3.3-V PCI | 3.3 | NA | 3.3 | NA | NA | NA |
| 3.3-V PCI-X | 3.3 | NA | 3.3 | NA | NA | NA |
| SSTL-2 Class I | 2.5 | 2.5 | 2.5 | 2.5 | 1.25 | 1.25 |
| SSTL-2 Class II | 2.5 | 2.5 | 2.5 | 2.5 | 1.25 | 1.25 |
| SSTL-18 Class I | 1.8 | 1.8 | 1.8 | 1.8 | 0.90 | 0.90 |
| SSTL-18 Class II | 1.8 | 1.8 | 1.8 | NA | 0.90 | 0.90 |
| 1.8-V HSTL Class I | 1.8 | 1.8 | 1.8 | 1.8 | 0.90 | 0.90 |
| 1.8-V HSTL Class II | 1.8 | 1.8 | 1.8 | NA | 0.90 | 0.90 |
| 1.5-V HSTL Class I | 1.5 | 1.5 | 1.5 | 1.5 | 0.75 | 0.75 |
| 1.5-V HSTL Class II | 1.5 | 1.5 | 1.5 | NA | 0.75 | 0.75 |
| 1.2-V HSTL *(4)* | 1.2 | NA | 1.2 | NA | 0.6 | NA |
| Differential SSTL-2 Class I | 2.5 | 2.5 | 2.5 | 2.5 | 1.25 | 1.25 |
| Differential SSTL-2 Class II | 2.5 | 2.5 | 2.5 | 2.5 | 1.25 | 1.25 |
| Differential SSTL-18 Class I | 1.8 | 1.8 | 1.8 | 1.8 | 0.90 | 0.90 |

*Table 8–4. Arria GX I/O Standards & Voltage Levels   (Part 2 of 2)* Note (1)

| I/O Standard | Arria GX | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| | $V_{CCIO}$ (V) | | | | $V_{REF}$ (V) | $V_{TT}$ (V) |
| | Input Operation | | Output Operation | | | |
| | Top & Bottom I/O Banks | Left & Right I/O Banks (3) | Top & Bottom I/O Banks | Left & Right I/O Banks (3) | Input | Termination |
| Differential SSTL-18 Class II | 1.8 | 1.8 | 1.8 | NA | 0.90 | 0.90 |
| 1.8-V differential HSTL Class I | 1.8 | 1.8 | 1.8 | NA | 0.90 | 0.90 |
| 1.8-V differential HSTL Class II | 1.8 | 1.8 | 1.8 | NA | 0.90 | 0.90 |
| 1.5-V differential HSTL Class I | 1.5 | 1.5 | 1.5 | NA | 0.75 | 0.75 |
| 1.5-V differential HSTL Class II | 1.5 | 1.5 | 1.5 | NA | 0.75 | 0.75 |
| LVDS (2) | 3.3/2.5/1.8/1.5 | 2.5 | 3.3 | 2.5 | NA | NA |
| HyperTransport technology | NA | 2.5 | NA | 2.5 | NA | NA |
| Differential LVPECL (2) | 3.3/2.5/1.8/1.5 | NA | 3.3 | NA | NA | NA |

*Notes to Table 8–4:*

(1)   Any input pins with PCI-clamping-diode enabled force the $V_{CCIO}$ to 3.3 V.

(2)   LVDS and LVPECL output operation in the top and bottom banks is only supported in PLL banks 9-12. The $V_{CCIO}$ level for differential output operation in the PLL banks is 3.3 V. The $V_{CCIO}$ level for output operation in the left and right I/O banks is 2.5 V.

(3)   The right I/O bank on Arria GX devices consists of transceivers.

(4)   1.2-V HSTL is only supported in I/O banks 4, 7, and 8.

Refer to the *DC & Switching Characteristics* chapter in volume 1 of the *Arria GX Device Handbook* for detailed electrical characteristics of each I/O standard.

# On-Chip Termination

Arria GX devices feature on-chip series termination to provide I/O impedance matching and termination capabilities. Apart from maintaining signal integrity, this feature also minimizes the need for external resistor networks, thereby saving board space and reducing costs.

Arria GX devices support on-chip series ($R_S$) termination for single-ended I/O standards and on-chip differential termination ($R_D$) for differential I/O standards. This section discusses the on-chip series termination support.

For more information about differential on-chip termination, refer to the *High-Speed Differential I/O Interfaces with DPA in Arria GX Devices* chapter in volume 2 of the Arria *GX Device Handbook.*

Arria GX devices support I/O driver on-chip series ($R_S$) termination through drive strength control for single-ended I/Os.

## On-Chip Series Termination without Calibration

Arria GX devices support driver impedance matching to provide the I/O driver with controlled output impedance that closely matches the impedance of the transmission line. As a result, reflections can be significantly reduced. Arria GX devices support on-chip series termination for single-ended I/O standards (see Figure 8–22). The $R_S$ shown in Figure 8–22 is the intrinsic impedance of transistors. The typical $R_S$ values are 25 $\Omega$ and 50 $\Omega$. Once matching impedance is selected, current drive strength is no longer selectable.

*Figure 8–22. Arria GX On-Chip Series Termination without Calibration*
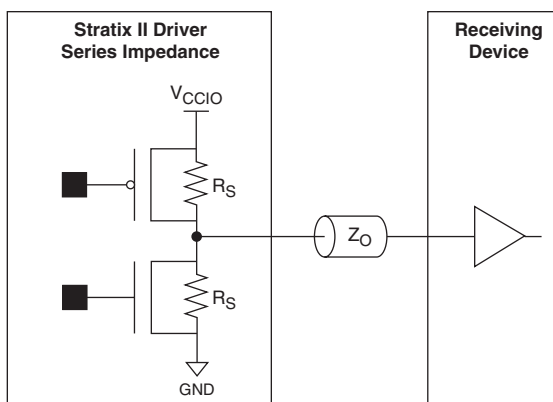
Table 8–5 shows the list of output standards that support on-chip series termination without calibration.

*Table 8–5. Selectable I/O Drivers with On-Chip Series Termination without Calibration*

| I/O Standard | On-chip Series Termination Setting | | |
|---|---|---|---|
| | Row I/O | Column I/O | Unit |
| 3.3-V LVTTL | 50 | 50 | Ω |
| | 25 | 25 | Ω |
| 3.3-V LVCMOS | 50 | 50 | Ω |
| | 25 | 25 | Ω |
| 2.5-V LVTTL | 50 | 50 | Ω |
| | 25 | 25 | Ω |
| 2.5-V LVCMOS | 50 | 50 | Ω |
| | 25 | 25 | Ω |
| 1.8-V LVTTL | 50 | 50 | Ω |
| | | 25 | Ω |
| 1.8-V LVCMOS | 50 | 50 | Ω |
| | | 25 | Ω |
| 1.5-V LVTTL | 50 | 50 | Ω |
| 1.5-V LVCMOS | 50 | 50 | Ω |
| SSTL-2 Class I | 50 | 50 | Ω |
| SSTL-2 Class II | 25 | 25 | Ω |
| SSTL-18 Class I | 50 | 50 | Ω |
| SSTL-18 Class II | | 25 | Ω |
| 1.8-V HSTL Class I | 50 | 50 | Ω |
| 1.8-V HSTL Class II | | 25 | Ω |
| 1.5-V HSTL Class I | 50 | 50 | Ω |
| 1.2-V HSTL *(1)* | | 50 | Ω |

*Note to Table 8–5:*

(1)    1.2-V HSTL is only supported in I/O banks 4, 7, and 8.

To use on-chip termination for the SSTL Class I standard, select the 50-Ω on-chip series termination setting for replacing the external 25-Ω $R_S$ (to match the 50-Ω transmission line). For the SSTL Class II standard, select the 25-Ω on-chip series termination setting (to match the 50-Ω transmission line and the near end 50-Ω pull-up to $V_{TT}$).

For more information about tolerance specifications for on-chip termination without calibration, refer to the *DC & Switching Characteristics* chapter in volume 1 of the *Arria GX Device Handbook*.

# Design Considerations

While Arria GX devices feature various I/O capabilities for high-performance and high-speed system designs, there are several other considerations that require attention to ensure the success of those designs.

## I/O Termination

I/O termination requirements for single-ended and differential I/O standards are discussed in this section.

### Single-Ended I/O Standards

Although single-ended, non-voltage-referenced I/O standards do not require termination, impedance matching is necessary to reduce reflections and improve signal integrity.

Voltage-referenced I/O standards require both an input reference voltage, $V_{REF,}$ and a termination voltage, $V_{TT}$. The reference voltage of the receiving device tracks the termination voltage of the transmitting device. Each voltage-referenced I/O standard requires a unique termination setup. For example, a proper resistive signal termination scheme is critical in SSTL standards to produce a reliable DDR memory system with superior noise margin.

Arria GX on-chip series termination provides the convenience of no external components. External pull-up resistors can be used to terminate the voltage-referenced I/O standards such as SSTL-2 and HSTL.

Refer to "Arria GX I/O Standards Support" on page 8–2 for more information about the termination scheme of various single-ended I/O standards.

### Differential I/O Standards

Differential I/O standards typically require a termination resistor between the two signals at the receiver. The termination resistor must match the differential load impedance of the bus. Arria GX devices provide an optional differential on-chip resistor when using LVDS and HyperTransport standards.

## I/O Banks Restrictions

Each I/O bank can simultaneously support multiple I/O standards. The following sections provide guidelines for mixing non-voltage-referenced and voltage-referenced I/O standards in Arria GX devices.

### Non-Voltage-Referenced Standards

Each Arria GX device I/O bank has its own $V_{CCIO}$ pins and supports only one $V_{CCIO}$, either 1.5, 1.8, 2.5, or 3.3 V. An I/O bank can simultaneously support any number of input signals with different I/O standard assignments, as shown in Table 8–6.

For output signals, a single I/O bank supports non-voltage-referenced output signals that are driving at the same voltage as $V_{CCIO}$. Since an I/O bank can only have one $V_{CCIO}$ value, it can only drive out that one value for non-voltage-referenced signals. For example, an I/O bank with a 2.5-V $V_{CCIO}$ setting can support 2.5-V standard inputs and outputs and 3.3-V LVCMOS inputs (not output or bidirectional pins).

*Table 8–6. Acceptable Input Levels for LVTTL & LVCMOS*

| Bank $V_{CCIO}$ (V) | Acceptable Input Levels (V) | | | |
|---|---|---|---|---|
| | 3.3 | 2.5 | 1.8 | 1.5 |
| 3.3 | ✓ | ✓ (1) | | |
| 2.5 | ✓ | ✓ | | |
| 1.8 | ✓ (2) | ✓ (2) | ✓ | ✓ (1) |
| 1.5 | ✓ (2) | ✓ (2) | ✓ | ✓ |

*Notes to Table 8–6:*
(1) Because the input signal does not drive to the rail, the input buffer does not completely shut off, and the I/O current is slightly higher than the default value.
(2) These input values overdrive the input buffer, so the pin leakage current is slightly higher than the default value. To drive inputs higher than $V_{CCIO}$ but less than 4.0 V, disable the PCI clamping diode and select the **Allow LVTTL and LVCMOS input levels to overdrive input buffer** option in the Quartus II software.

*Voltage-Referenced Standards*

To accommodate voltage-referenced I/O standards, each Arria GX device's I/O bank supports multiple $V_{REF}$ pins feeding a common $V_{REF}$ bus. The number of available $V_{REF}$ pins increases as device density increases. If these pins are not used as $V_{REF}$ pins, they cannot be used as generic I/O pins. However, each bank can only have a single $V_{CCIO}$ voltage level and a single $V_{REF}$ voltage level at a given time.

An I/O bank featuring single-ended or differential standards can support voltage-referenced standards as long as all voltage-referenced standards use the same $V_{REF}$ setting.

Because of performance reasons, voltage-referenced input standards use their own $V_{CCIO}$ level as the power source. For example, you can only place 1.5-V HSTL input pins in an I/O bank with a 1.5-V $V_{CCIO}$.

☞ Refer to "Arria GX I/O Banks" on page 8–20 for details about input $V_{CCIO}$ for voltage-referenced standards.

Voltage-referenced bidirectional and output signals must be the same as the I/O bank's $V_{CCIO}$ voltage. For example, you can only place SSTL-2 output pins in an I/O bank with a 2.5-V $V_{CCIO}$.

☞ Refer to "I/O Placement Guidelines" on page 8–30 for details about voltage-referenced I/O standards placement.

*Mixing Voltage-Referenced and Non-Voltage-Referenced Standards*

An I/O bank can support both non-voltage-referenced and voltage-referenced pins by applying each of the rule sets individually. For example, an I/O bank can support SSTL-18 inputs and 1.8-V inputs and outputs with a 1.8-V $V_{CCIO}$ and a 0.9-V $V_{REF}$. Similarly, an I/O bank can support 1.5-V standards, 2.5-V (inputs, but not outputs), and HSTL I/O standards with a 1.5-V $V_{CCIO}$ and 0.75-V $V_{REF}$.

## I/O Placement Guidelines

The I/O placement guidelines help to reduce noise issues that may be associated with a design such that Arria GX devices can maintain an acceptable noise level on the $V_{CCIO}$ supply. Because Arria GX devices require each bank to be powered separately for $V_{CCIO}$, these noise issues have no effect when crossing bank boundaries and, as such, these rules need not be applied.

This section provides I/O placement guidelines for the programmable I/O standards supported by Arria GX devices and includes essential information for designing systems using their devices' selectable I/O capabilities.

### $V_{REF}$ Pin Placement Restrictions

There are at least two dedicated $V_{REF}$ pins per I/O bank to drive the $V_{REF}$ bus. Larger Arria GX devices have more $V_{REF}$ pins per I/O bank. All $V_{REF}$ pins within one I/O bank are shorted together at device die level.

There are limits to the number of pins that a $V_{REF}$ pin can support. For example, each output pin adds some noise to the $V_{REF}$ level and an excessive number of outputs make the level too unstable to be used for incoming signals.

Restrictions on the placement of single-ended voltage-referenced I/O pads with respect to $V_{REF}$ pins help maintain an acceptable noise level on the $V_{CCIO}$ supply and prevent output switching noise from shifting the $V_{REF}$ rail.

**Input Pins**

Each $V_{REF}$ pin supports a maximum of 40 input pads.

**Output Pins**

When a voltage-referenced input or bidirectional pad does not exist in a bank, the number of output pads that can be used in that bank depends on the total number of available pads in that same bank. However, when a voltage-referenced input exists, a design can use up to 20 output pads per $V_{REF}$ pin in a bank.

**Bidirectional Pins**

Bidirectional pads must satisfy both input and output guidelines simultaneously. The general formulas for input and output rules are shown in Table 8–7.

*Table 8–7. Bidirectional Pin Limitation Formulas*

| Rules | Formulas |
|---|---|
| Input | <Total number of bidirectional pins> + <Total number of $V_{REF}$ input pins, if any> ≤ 40 per $V_{REF}$ pin |
| Output | <Total number of bidirectional pins> + <Total number of output pins, if any> − <Total number of pins from smallest OE group, if more than one OE groups> ≤ 20 per $V_{REF}$ pin |

- If the same output enable (OE) controls all the bidirectional pads (bidirectional pads in the same OE group are driving in and out at the same time) and there are no other outputs or voltage-referenced inputs in the bank, the voltage-referenced input is never active at the same time as an output. Therefore, the output limitation rule does not apply. However, since the bidirectional pads are linked to the same OE, the bidirectional pads will all act as inputs at the same time. Therefore, there is a limit of 40 input pads, as follows:

    *<Total number of bidirectional pins> + <Total number of $V_{REF}$ input pins>* ≤ 40 per $V_{REF}$ pin

- If any of the bidirectional pads are controlled by different OE and there are no other outputs or voltage-referenced inputs in the bank, one group of bidirectional pads can be used as inputs and another group is used as outputs. In such cases, the formula for the output rule is simplified, as follows:

    *<Total number of bidirectional pins> – <Total number of pins from smallest OE group>* ≤ 20 per $V_{REF}$ pin

- Consider a case where eight bidirectional pads are controlled by OE1, eight bidirectional pads are controlled by OE2, six bidirectional pads are controlled by OE3, and there are no other outputs or voltage-referenced inputs in the bank. While this totals 22 bidirectional pads, it is safely allowable because there would be a possible maximum of 16 outputs per $V_{REF}$ pin, assuming the worst case where OE1 and OE2 are active and OE3 is inactive. This is useful for DDR SDRAM applications.

- When at least one additional voltage-referenced input and no other outputs exist in the same $V_{REF}$ group, the bidirectional pad limitation must simultaneously adhere to the input and output limitations. The input rule becomes:

    *<Total number of bidirectional pins> + <Total number of $V_{REF}$ input pins>* ≤ 40 per $V_{REF}$ pin

    Whereas the output rule is simplified as:

    *<Total number of bidirectional pins>* 20 per $V_{REF}$ pin

- When at least one additional output exists but no voltage-referenced inputs exist, the output rule becomes:

    *<Total number of bidirectional pins> + <Total number of output pins> – <Total number of pins from smallest OE group>* ≤ 0 per $V_{REF}$ pin

■ When additional voltage-referenced inputs and other outputs exist in the same $V_{REF}$ group, the bidirectional pad limitation must again simultaneously adhere to the input and output limitations. The input rule is:

*<Total number of bidirectional pins> + <Total number of $V_{REF}$ input pins>* ≤40 per $V_{REF}$ pin

Whereas the output rule is given as:

*<Total number of bidirectional pins> + <Total number of output pins> – <Total number of pins from smallest OE group>* ≤0 per $V_{REF}$ pin

### I/O Pin Placement with Respect to High-Speed Differential I/O Pins

Regardless of whether or not the SERDES circuitry is utilized, there is a restriction on the placement of single-ended output pins with respect to high-speed differential I/O pins. As shown in Figure 8–23, all single-ended outputs must be placed at least one LAB row away from the differential I/O pins. There are no restrictions on the placement of single-ended input pins with respect to differential I/O pins. Single-ended input pins may be placed within the same LAB row as differential I/O pins. However, the single-ended input's IOE register is not available. The input must be implemented within the core logic.

This single-ended output pin placement restriction only applies when using the LVDS or HyperTransport I/O standards in the left I/O banks. There are no restrictions for single-ended output pin placement with respect to differential clock pins in the top and bottom I/O banks.

*Figure 8–23. Single-Ended Output Pin Placement with Respect to Differential I/O Pins*
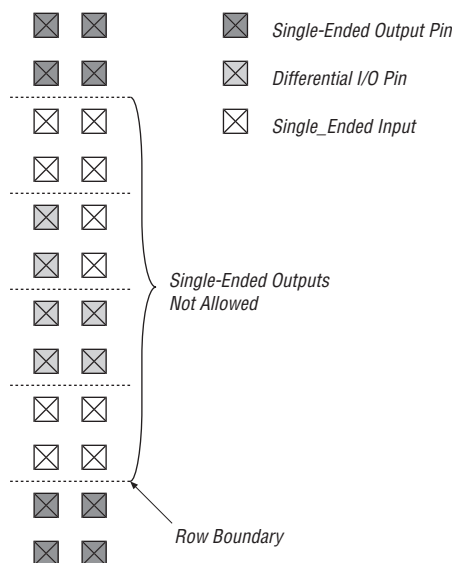


DC Guidelines

Power budgets are essential to ensure the reliability and functionality of a system application. You are often required to perform power dissipation analysis on each device in the system to come out with the total power dissipated in that system, which is composed of a static component and a dynamic component.

The static power consumption of a device is the total DC current flowing from $V_{CCIO}$ to ground.

For any ten consecutive pads in an I/O bank of an Arria GX device, Altera recommends a maximum current of 250 mA, as shown in Figure 8–24, because the placement of $V_{CCIO}$/ground (GND) bumps are regular, 10 I/O pins per pair of power pins. This limit is on the static power consumed by an I/O standard, as shown in Table 8–8. Limiting static power is a way to improve reliability over the lifetime of the device.

*Figure 8–24. DC Current Density Restriction Notes (1), (2)*
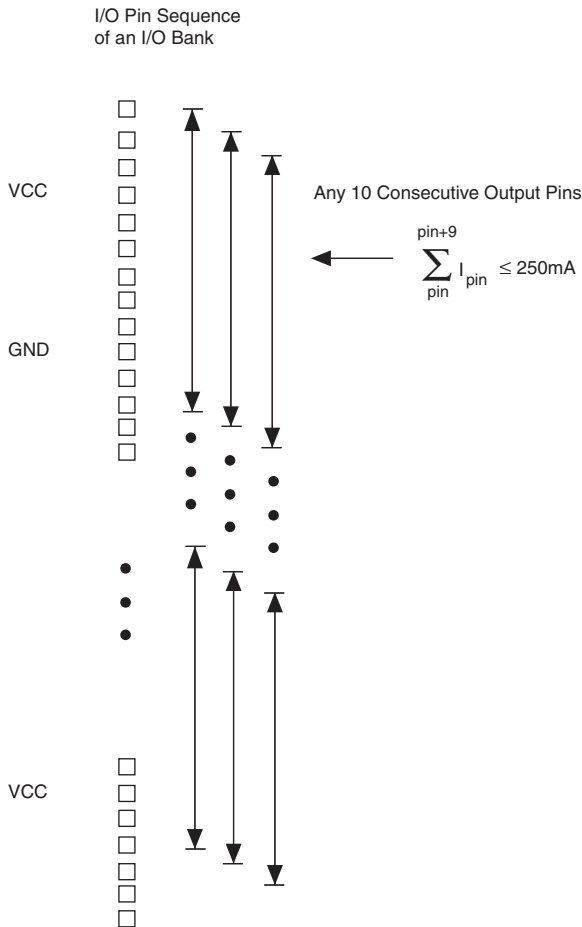


*Notes to Figure 8–24:*
(1)    The consecutive pads do not cross I/O banks.
(2)    $V_{REF}$ pins do not affect DC current calculation because there are no $V_{REF}$ pads.

Table 8–8 shows the I/O standard DC current specification.

| *Table 8–8. Arria GX I/O Standard DC Current Specification  (Part 1 of 2) Note (1)* | | |
|---|---|---|
| **I/O Standard** | **$I_{PIN}$ (mA), Top & Bottom I/O Banks** | **$I_{PIN}$ (mA), Left I/O Banks *(2)*** |
| LVTTL | *(3)* | *(3)* |
| LVCMOS | *(3)* | *(3)* |

| *Table 8–8. Arria GX I/O Standard DC Current Specification  (Part 2 of 2)* Note (1) | | |
|---|---|---|
| **I/O Standard** | **I$_{PIN}$ (mA), Top & Bottom I/O Banks** | **I$_{PIN}$ (mA), Left I/O Banks** (2) |
| 2.5 V | (3) | (3) |
| 1.8 V | (3) | (3) |
| 1.5 V | (3) | (3) |
| 3.3-V PCI | 1.5 | NA |
| 3.3-V PCI-X | 1.5 | NA |
| SSTL-2 Class I | 12 (4) | 12 (4) |
| SSTL-2 Class II | 24 (4) | 16 (4) |
| SSTL-18 Class I | 12 (4) | 10 ((4) |
| SSTL-18 Class II | 20 (4) | NA |
| 1.8-V HSTL Class I | 12 (4) | 12 |
| 1.8-V HSTL Class II | 20 (4) | NA |
| 1.5-V HSTL Class I | 12 (4) | 8 |
| 1.5-V HSTL Class II | 20 (4) | NA |
| Differential SSTL-2 Class I | 12 | 12 |
| Differential SSTL-2 Class II | 24 | 16 |
| Differential SSTL-18 Class I | 12 | 10 |
| Differential SSTL-18 Class II | 20 | NA |
| 1.8-V differential HSTL Class I | 12 | 12 |
| 1.8-V differential HSTL Class II | 20 | NA |
| 1.5-V differential HSTL Class I | 12 | 8 |
| 1.5-V differential HSTL Class II | 20 | NA |
| LVDS | 12 | 12 |
| HyperTransport technology | NA | 16 |
| Differential LVPECL | 10 | 10 |

*Notes to Table 8–8:*

(1)    The current value obtained for differential HSTL and differential SSTL standards is per pin and not per differential pair, as opposed to the per-pair current value of LVDS and HyperTransport standards.

(2)    This does not apply to the right I/O banks of Arria GX devices. Arria GX devices have transceivers on the right I/O banks.

(3)    The DC power specification of each I/O standard depends on the current sourcing and sinking capabilities of the I/O buffer programmed with that standard, as well as the load being driven. LVTTL, LVCMOS, 2.5-V, 1.8-V, and 1.5-V outputs are not included in the static power calculations because they normally do not have resistor loads in real applications. The voltage swing is rail-to-rail with capacitive load only. There is no DC current in the system.

(4)    This I$_{PIN}$ value represents the DC current specification for the default current strength of the I/O standard. The I$_{PIN}$ varies with programmable drive strength and is the same as the drive strength as set in Quartus II software. Refer to the *Arria GX Architecture* chapter in volume 1 of the *Arria GX Device Handbook* for a detailed description of the programmable drive strength feature of voltage-referenced I/O standards.

Table 8–8 only shows the limit on the static power consumed by an I/O standard. The amount of power used at any given moment could be much higher, and is based on the switching activities.

# Conclusion

Arria GX devices provide I/O capabilities that allow you to work in compliance with current and emerging I/O standards and requirements. With the Arria GX device features, such as programmable driver strength, you can reduce board design interface costs and increase the development flexibility.

# References

Refer to the following references for more information:

■ Interface Standard for Nominal 3V / 3.3-V Supply Digital Integrated Circuits, JESD8-B, Electronic Industries Association, September 1999.
■ 2.5-V +/- 0.2V (Normal Range) and 1.8-V to 2.7V (Wide Range) Power Supply Voltage and Interface Standard for Non-terminated Digital Integrated Circuits, JESD8-5, Electronic Industries Association, October 1995.
■ 1.8-V +/- 0.15 V (Normal Range) and 1.2 V - 1.95 V (Wide Range) Power Supply Voltage and Interface Standard for Non-terminated Digital Integrated Circuits, JESD8-7, Electronic Industries Association, February 1997.
■ 1.5-V +/- 0.1 V (Normal Range) and 0.9 V - 1.6 V (Wide Range) Power Supply Voltage and Interface Standard for Non-terminated Digital Integrated Circuits, JESD8-11, Electronic Industries Association, October 2000.
■ PCI Local Bus Specification, Revision 2.2, PCI Special Interest Group, December 1998.
■ PCI-X Local Bus Specification, Revision 1.0a, PCI Special Interest Group.
■ Stub Series Terminated Logic for 2.5-V (SSTL-2), JESD8-9A, Electronic Industries Association, December 2000.
■ Stub Series Terminated Logic for 1.8 V (SSTL-18), Preliminary JC42.3, Electronic Industries Association.
■ High-Speed Transceiver Logic (HSTL)—A 1.5-V Output Buffer Supply Voltage Based Interface Standard for Digital Integrated Circuits, EIA/JESD8-6, Electronic Industries Association, August 1995.
■ Electrical Characteristics of Low Voltage Differential Signaling (LVDS) Interface Circuits, ANSI/TIA/EIA-644, American National Standards Institute/Telecommunications Industry/Electronic Industries Association, October 1995.

# Referenced Documents

This chapter references the following documents:

- *Arria GX Architecture* chapter in volume 1 of the *Arria GX Device Handbook*
- *DC & Switching Characteristics* chapter in volume 1 of the *Arria GX Device Handbook*
- *External Memory Interfaces in Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*
- *High-Speed Differential I/O Interfaces with DPA in Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*
- *PLLs in Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*

# Document Revision History

Table 8–9 shows the revision history for this chapter.

*Table 8–9. Document Revision History*

| Date and Document Version | Changes Made | Summary of Changes |
|---|---|---|
| May 2008 v1.2 | Updated "1.5-V HSTL Class I & 1.5-V HSTL Class II" section. | — |
| | Minor text edits. | — |
| August 2007 v1.1 | Added the "Referenced Documents" section. | — |
| | Minor text edits. | — |
| May 2007 v1.0 | Initial release. | — |

## Introduction

The Arria™ GX device family offers up to 840-Mbps differential I/O capabilities to support source-synchronous communication protocols such as HyperTransport™ technology, Rapid I/O, XSBI, and SPI.

Arria GX devices have the following dedicated circuitry for high-speed differential I/O support:

- Differential I/O buffer
- Transmit serializer
- Receive deserializer
- Data realignment circuit
- Dynamic phase aligner (DPA)
- Synchronizer (FIFO buffer)
- Analog phased locked loop (PLLs) and fast PLLs

For high-speed differential interfaces, Arria GX devices can accommodate different differential I/O standards, including the following:

- LVDS
- HyperTransport technology
- HSTL
- SSTL
- LVPECL

☞ HSTL, SSTL, and LVPECL I/O standards can be used only for PLL clock inputs and outputs in differential mode.

This chapter contains the following sections:

# I/O Banks

Arria GX inputs and outputs are partitioned into banks located on the periphery of the die. The inputs and outputs that support LVDS and HyperTransport technology are located in row I/O banks, on the left side of the Arria GX device. LVPECL, HSTL, and SSTL standards are supported on certain top and bottom banks of the die (banks 9 to 12) when used as differential clock inputs/outputs. Differential HSTL and SSTL standards can be supported on banks 3, 4, 7, and 8 if the pins on these banks are used as DQS pins. Figure 9–1 shows where the banks and the PLLs are located on the die.

*Figure 9–1. Arria GX I/O Banks Notes (1), (2), (3), (4), (5), and (6)*
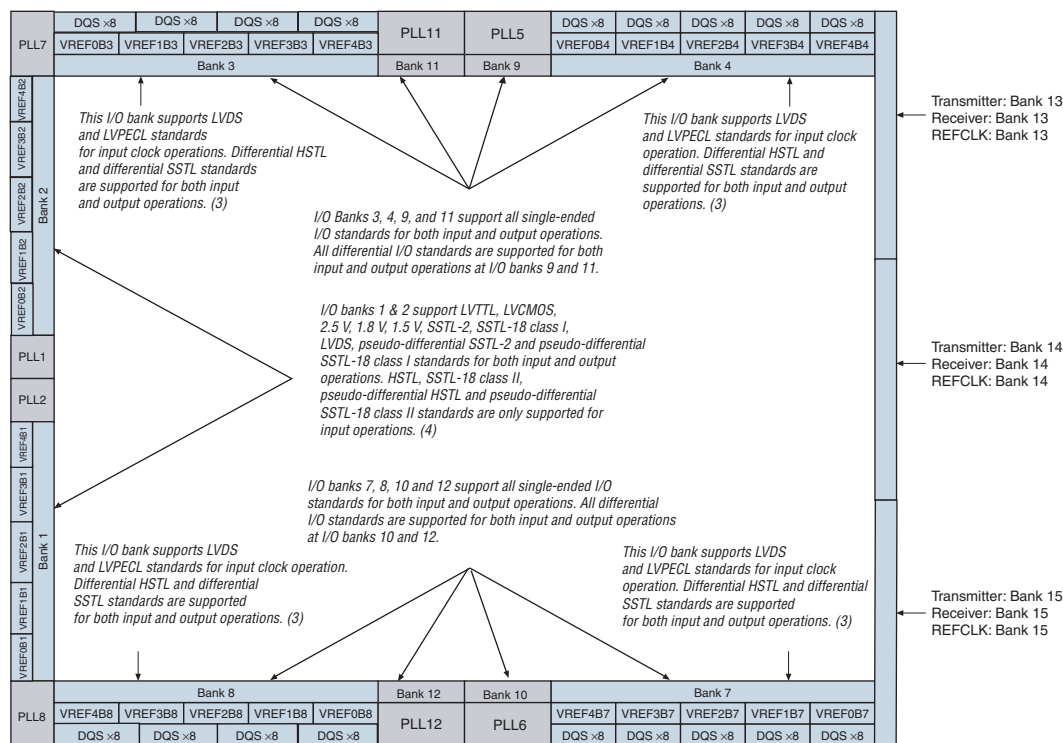


**Notes to Figure 9–1:**
(1) Figure 9–1 is a top view of the silicon die which corresponds to a reverse view for flip-chip packages. It is a graphical representation only.
(2) Depending on size of the device, different device members have different numbers of $V_{REF}$ groups. Refer to the pin list and the Quartus® II software for exact locations.
(3) Banks 9 through 12 are enhanced PLL external clock output banks.
(4) Horizontal I/O banks feature transceiver and dynamic phase alignment (DPA) circuitry for high speed differential I/O standards.
(5) Quartus II software does not support differential SSTL and differential HSTL standards at left/right I/O banks.
(6) Number of available PLLs and corresponding I/O banks vary with package options.

Table 9–1 shows the total number of differential channels available in Arria GX devices. Non-dedicated clocks in the left bank can also be used as data receiver channels. The total number of receiver channels includes these four non-dedicated clock channels. Pin migration is available for different size devices in the same package.

| Table 9–1. Differential Channels in Arria GX Devices | | | Notes (1), (2) |
|---|---|---|---|
| Device | 484-Pin FineLine BGA | 780-Pin FineLine BGA | 1,152-Pin FineLine BGA |
| EP1AGX20 | 29 transmitters 31 receivers | 29 transmitters 31 receivers | — |
| EP1AGX35 | 29 transmitters 31 receivers | 29 transmitters 31 receivers | — |
| EP1AGX50 | 29 transmitters 31 receivers | 29 transmitters 31 receivers | 42 transmitters 42 receivers |
| EP1AGX60 | 29 transmitters 31 receivers | 29 transmitters 31 receivers | 42 transmitters 42 receivers |
| EP1AGX90 | — | — | 47 transmitters 47 receivers |

*Notes to Table 9–1:*
(1)  Pin count does not include dedicated PLL input pins.
(2)  The total number of receiver channels includes the four non-dedicated clock channels that can optionally be used as data channels.

# Differential Transmitter

The Arria GX transmitter has dedicated circuitry to provide support for LVDS and HyperTransport signaling. The dedicated circuitry consists of a differential buffer, a serializer, and a shared fast PLL. The differential buffer can drive out LVDS or HyperTransport signal levels that are statically set in the Quartus II software. The serializer takes data from a parallel bus up to 10-bits wide from the internal logic, clocks it into the load registers, and serializes it using the shift registers before sending the data to the differential buffer. The most significant bit (MSB) is transmitted first. The load and shift registers are clocked by the diffioclk (a fast PLL clock running at the serial rate) and controlled by the load enable signal generated from the fast PLL. The serialization factor can be statically set to $\times 4$, $\times 5$, $\times 6$, $\times 7$, $\times 8$, $\times 9$ or $\times 10$ using the Quartus II software. The load enable signal is automatically generated by the fast PLL and is derived from the serialization factor setting. Figure 9–2 is a block diagram of the Arria GX transmitter.

*Figure 9–2. Transmitter Block Diagram*
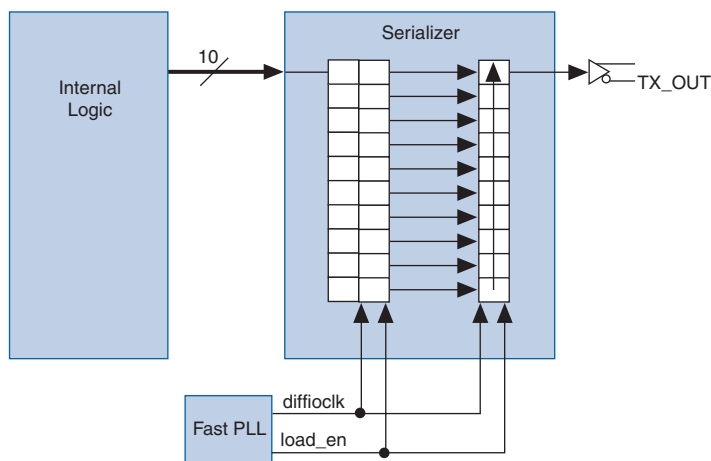


Each Arria GX transmitter data channel can be configured to operate as a transmitter clock output. This flexibility allows the designer to place the output clock near the data outputs to simplify board layout and reduce clock-to-data skew. Different applications often require specific clock-to-data alignments or specific data-rate to clock-rate factors. The transmitter can output a clock signal at the same rate as the data with a maximum frequency of 717 MHz. The output clock can also be divided by a factor of 2, 4, 8, or 10, depending on the serialization factor. The phase of the clock in relation to the data can be set at 0° or 180° (edge or center aligned). The fast PLL provides additional support for other phase shifts in 45° increments. These settings are made statically in the Quartus II MegaWizard® software. Figure 9–3 shows the transmitter in clock output mode.

*Figure 9–3. Transmitter in Clock Output Mode*
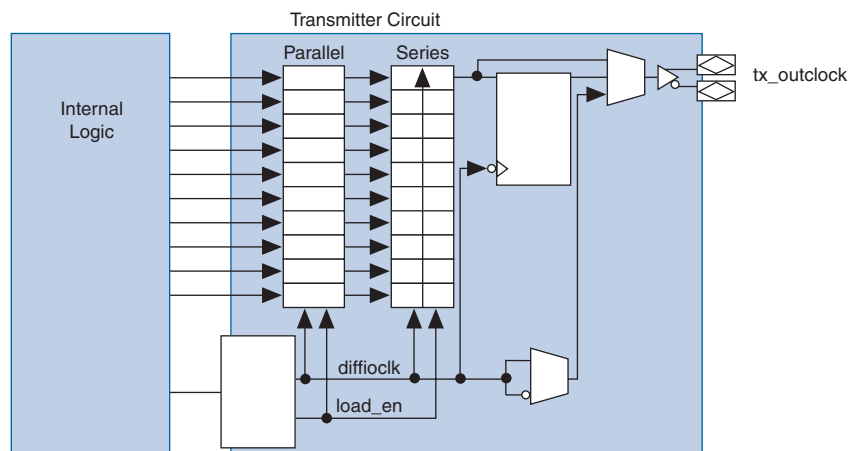


The serializer can be bypassed to support DDR (×2) and SDR (×1) operations. The I/O element (IOE) contains two data output registers that each can operate in either DDR or SDR mode. The clock source for the registers in the IOE can come from any routing resource, from the fast PLL, or from the enhanced PLL. Figure 9–4 shows the bypass path.

*Figure 9–4. Serializer Bypass*
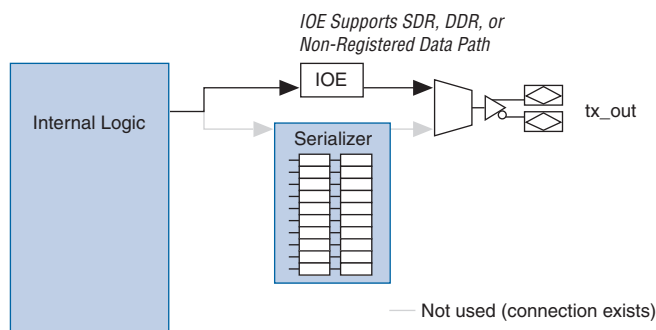
# Differential Receiver

The receiver has dedicated circuitry to support high-speed LVDS and HyperTransport signaling, along with enhanced data reception. Each receiver consists of a differential buffer, dynamic phase aligner (DPA), synchronization FIFO buffer, data realignment circuit, deserializer, and a shared fast PLL. The differential buffer receives LVDS or HyperTransport signal levels, which are statically set by the Quartus II software. The DPA block aligns the incoming data to one of eight clock phases to maximize the receiver's skew margin. The DPA circuit can be bypassed on a channel-by-channel basis if it is not needed. Set the DPA bypass statically in the Quartus II MegaWizard Plug-In Manager or dynamically by using the optional RX_DPLL_ENABLE port.

The synchronizer circuit is a 1-bit wide by 6-bit deep FIFO buffer that compensates for any phase difference between the DPA block and the deserializer. If necessary, the data realignment circuit inserts a single bit of latency in the serial bitstream to align the word boundary. The deserializer includes shift registers and parallel load registers, and sends a maximum of 10 bits to the internal logic. The data path in the receiver is clocked by either the diffioclk signal or the DPA recovered clock. The deserialization factor can be statically set to 4, 5, 6, 7, 8, 9, or 10 by using the Quartus II software. The fast PLL automatically generates the load enable signal, which is derived from the deserialization factor setting.

Figure 9–5 shows a block diagram of the receiver.

*Figure 9–5. Receiver Block Diagram*
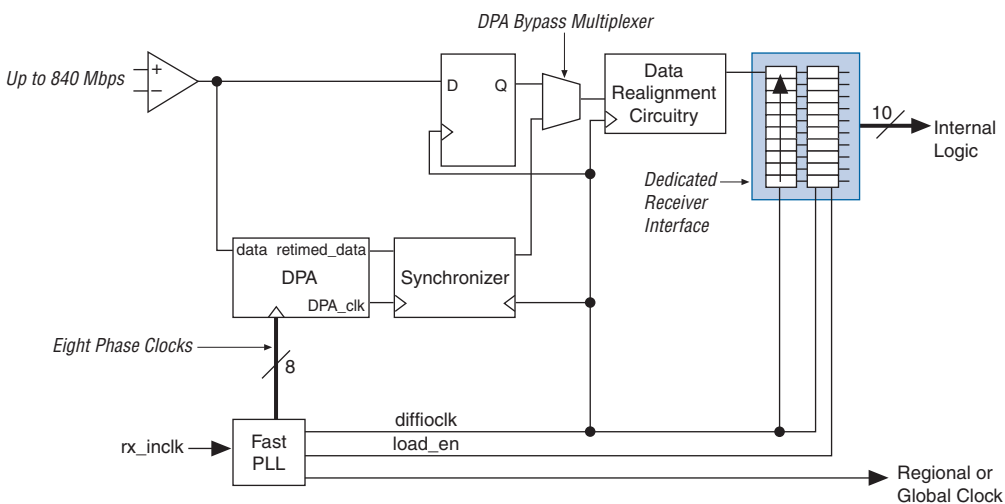
The deserializer, like the serializer, can also be bypassed to support DDR (×2) and SDR (×1) operations. The DPA and data realignment circuit cannot be used when the deserializer is bypassed. The IOE contains two data input registers that can operate in DDR or SDR mode. The clock source for the registers in the IOE can come from any routing resource, from the fast PLL, or from the enhanced PLL. Figure 9–6 shows the bypass path.

*Figure 9–6. Deserializer Bypass*
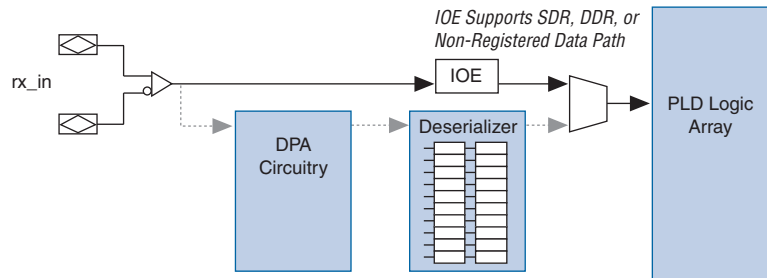


## Receiver Data Realignment Circuit

The data realignment circuit aligns the word boundary of the incoming data by inserting bit latencies into the serial stream. An optional `RX_CHANNEL_DATA_ALIGN` port controls the bit insertion of each receiver independently controlled from the internal logic. The data slips one bit for every pulse on the `RX_CHANNEL_DATA_ALIGN` port. The following are requirements for the `RX_CHANNEL_DATA_ALIGN` port:

- The minimum pulse width is one period of the parallel clock in the logic array.
- The minimum low time between pulses is one period of parallel clock.
- There is no maximum high or low time.
- Valid data is available two parallel clock cycles after the rising edge of `RX_CHANNEL_DATA_ALIGN`.

Figure 9–7 shows receiver output (RX_OUT) after one bit slip pulse with the deserialization factor set to 4.

*Figure 9–7. Data Realignment Timing*
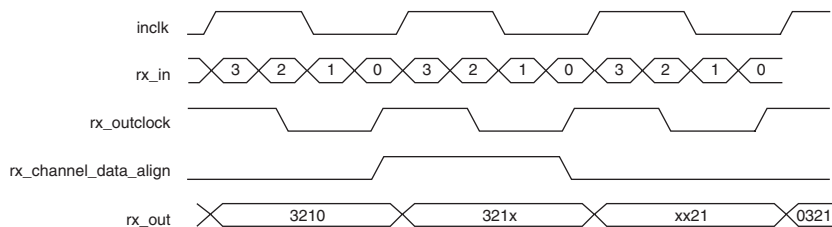


The data realignment circuit can have up to 11 bit-times of insertion before a rollover occurs. The programmable bit rollover point can be from 1 to 11 bit-times independent of the deserialization factor. An optional status port, RX_CDA_MAX, is available to the FPGA from each channel to indicate when the preset rollover point is reached.

## Dynamic Phase Aligner

The DPA block takes in high-speed serial data from the differential input buffer and selects one of eight phase clocks to sample the data. The DPA chooses a phase closest to the phase of the serial data. The maximum phase offset between the data and the phase-aligned clock is 1/8 UI, which is the maximum quantization error of the DPA. The eight phases are equally divided, giving a 45° resolution. Figure 9–8 shows the possible phase relationships between the DPA clocks and the incoming serial data.

*Figure 9–8. DPA Clock Phase to Data Bit Relationship*
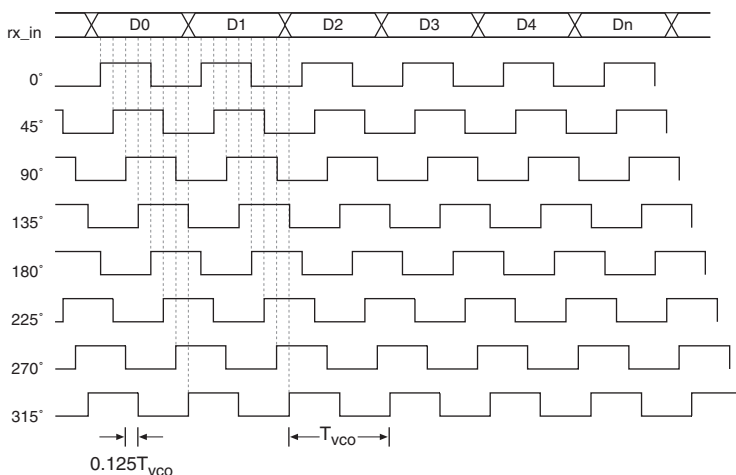


Each DPA block continuously monitors the phase of the incoming data stream and selects a new clock phase if needed. The selection of a new clock phase can be prevented by the optional RX_DPLL_HOLD port, which is available for each channel.

The DPA block requires a training pattern and a training sequence of at least 256 repetitions of the training pattern. The training pattern is not fixed, so you can use any training pattern with at least one transition on each channel. An optional output port, RX_DPA_LOCKED, is available to the internal logic, to indicate when the DPA block has settled on the closest phase to the incoming data phase. The RX_DPA_LOCKED de-asserts, depending on what is selected in the Quartus II MegaWizard Plug-In, when either a new phase is selected, or when the DPA has moved two phases in the same direction. The data may still be valid even when the RX_DPA_LOCKED is deasserted. Use data checkers to validate the data when RX_DPA_LOCKED is deasserted.

An independent reset port, RX_RESET, is available to reset the DPA circuitry. The DPA circuit must be retrained after reset.

## Synchronizer

The synchronizer is a 1-bit × 6-bit deep FIFO buffer that compensates for the phase difference between the recovered clock from the DPA circuit and the diffioclk that clocks the rest of the logic in the receiver. The synchronizer can only compensate for phase differences, not frequency differences between the data and the receiver's INCLK. An optional port,

RX_FIFO_RESET, is available to the internal logic to reset the synchronizer. The synchronizer is automatically reset when the DPA first locks to the incoming data. Altera® recommends using RX_FIFO_RESET to reset the synchronizer when the DPA signals a loss-of-lock condition beyond the initial locking condition.
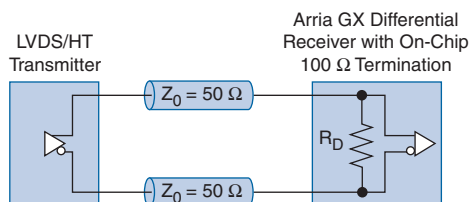
## Differential I/O Termination

Arria GX devices provide an on-chip 100-Ω differential termination option on each differential receiver channel for LVDS and HyperTransport standards. The on-chip termination (OCT) eliminates the need to supply an external termination resistor, simplifying the board design and reducing reflections caused by stubs between the buffer and the termination resistor. You can enable on-chip termination in the Quartus II assignments editor. Differential on-chip termination is supported across the full range of supported differential data rates.

For more information regarding differential on-chip termination, refer to the High-Speed I/O Specifications section of the *DC & Switching Characteristic*s chapter in volume 1 of the *Arria GX Device Handbook*.

Figure 9–9 illustrates on-chip termination.

*Figure 9–9. On-Chip Differential Termination*



On-chip differential termination is supported on all row I/O pins and on clock pins CLK[0, 2, 8, 10]. The clock pins CLK[1, 3, 9, 11], and FPLL[7..10]CLK, and the clocks in the top and bottom I/O banks (CLK[4..7, 12..15]) do not support differential on-chip termination.

## Fast PLL

The high-speed differential I/O receiver and transmitter channels use fast PLL to generate the parallel global clocks (rx- or tx- clock) and high-speed clocks (diffioclk). Figure 9–10 shows the locations of the fast PLLs. The fast PLL VCO operates at the clock frequency of the data rate. Each fast PLL offers a single serial data rate support, but up to two separate serialization and/or deserialization factors (from the C0 and C1

fast PLL clock outputs) can be used. Clock switchover and dynamic fast PLL reconfiguration is available in high-speed differential I/O support mode.

For additional information about the fast PLL, refer to the *PLLs in Arria GX Devices* chapter in volume 2 of the *Arria GX Handbook*.

Figure 9–10 shows a block diagram of the fast PLL in high-speed differential I/O support mode.

*Figure 9–10. Fast PLL Block Diagram*
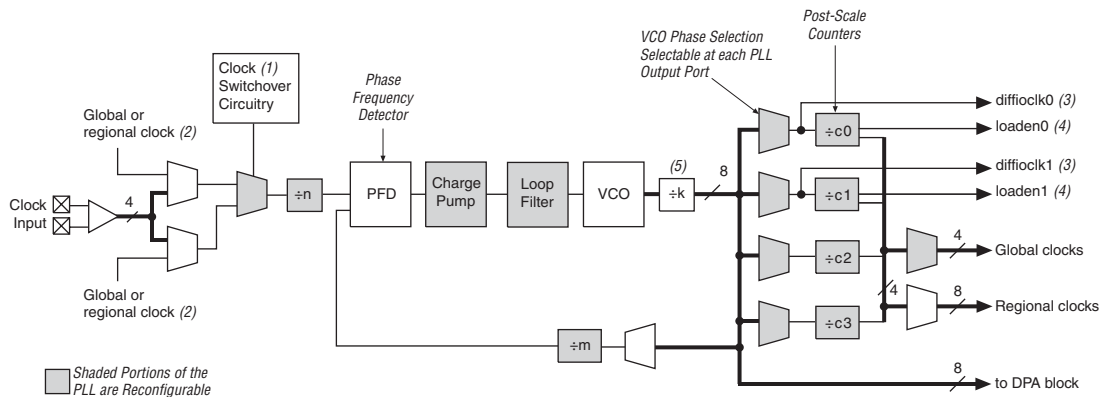


*Notes to Figure 9–10:*
(1) Arria GX fast PLLs only support manual clock switchover.
(2) The global or regional clock input can be driven by an output from another PLL, a pin-driven dedicated global or regional clock, or through a clock control block provided the clock control block is fed by an output from another PLL or pin-driven dedicated global or regional clock.
(3) In high-speed differential I/O support mode, this high-speed PLL clock feeds the SERDES. Arria GX devices only support one rate of data transfer per fast PLL in high-speed differential I/O support mode.
(4) This signal is a high-speed differential I/O support SERDES control signal.
(5) If the design enables this ÷2 counter, the device can use a VCO frequency range of 150 to 520 MHz.

**Clocking**

The fast PLLs feed in to the differential receiver and transmitter channels through the LVDS/DPA clock network. The center fast PLLs can independently feed the banks above and below them. The corner PLLs can feed only the banks adjacent to them.

Figures 9–11 and 9–12 show the Fast PLL and LVDS/DPA clock of the Arria GX devices.

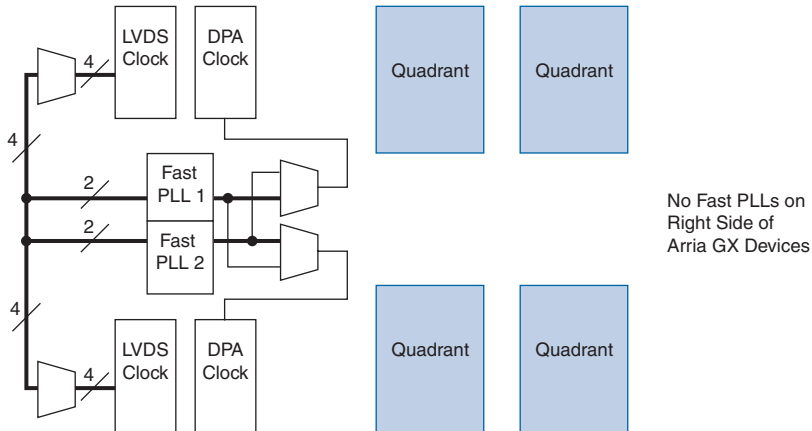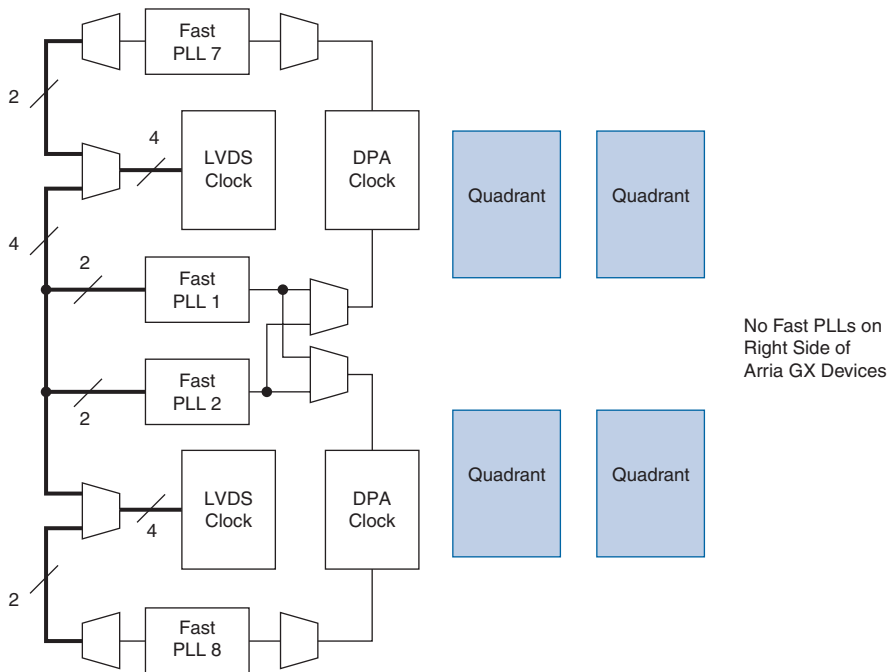*Figure 9–11. Fast PLL and LVDS/DPA Clock for EP1AGX20C, EP1AGX35C/D, EP1AGX50C/D, and EP1AGX60C/D Devices*

*Figure 9–12. Fast PLL and LVDS/DPA Clocks for EP1AGX60E and EP1AGX90E Devices*



## Source Synchronous Timing Budget

This section discusses the timing budget, waveforms, and specifications for source-synchronous signaling in Arria GX devices. LVDS and HyperTransport I/O standards enable high-speed data transmission. This high data transmission rate results in better overall system performance. To take advantage of fast system performance, it is important to understand how to analyze timing for these high-speed signals. Timing analysis for the differential block is different from traditional synchronous timing analysis techniques.

Rather than focusing on clock-to-output and setup times, source-synchronous timing analysis is based on the skew between the data and the clock signals. High-speed differential data transmission requires the use of timing parameters provided by IC vendors and is strongly influenced by board skew, cable skew, and clock jitter. This section defines the source-synchronous differential data orientation timing parameters, the timing budget definitions for Arria GX devices, and how to use these timing parameters to determine a design's maximum performance.

## Differential Data Orientation

There is a set relationship between an external clock and the incoming data. For operation at 840 Mbps and SERDES factor of 10, the external clock is multiplied by 10, and phase-alignment can be set in the PLL to coincide with the sampling window of each data bit. The data is sampled on the falling edge of the multiplied clock. Figure 9–13 shows the data bit orientation of the ×10 mode.

*Figure 9–13. Bit Orientation in the Quartus II Software*
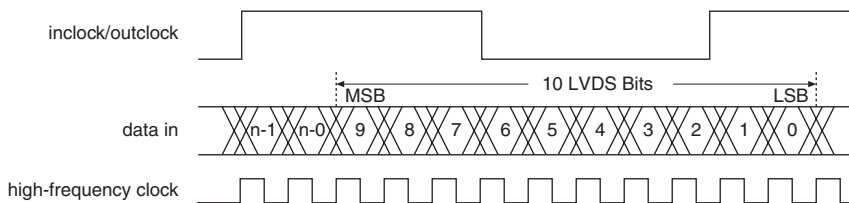


## Differential I/O Bit Position

Data synchronization is necessary for successful data transmission at high frequencies. Figure 9–14 shows the data bit orientation for a channel operation. These figures are based on the following:

- SERDES factor equals clock multiplication factor
- Edge alignment is selected for phase alignment
- Implemented in hard SERDES

For other serialization factors use the Quartus II software tools and find the bit position within the word. The bit positions after deserialization are listed in Table 9–2.

Figure 9–14 also shows a functional waveform. Timing waveforms may produce different results. Altera recommends performing a timing simulation to predict actual device behavior.

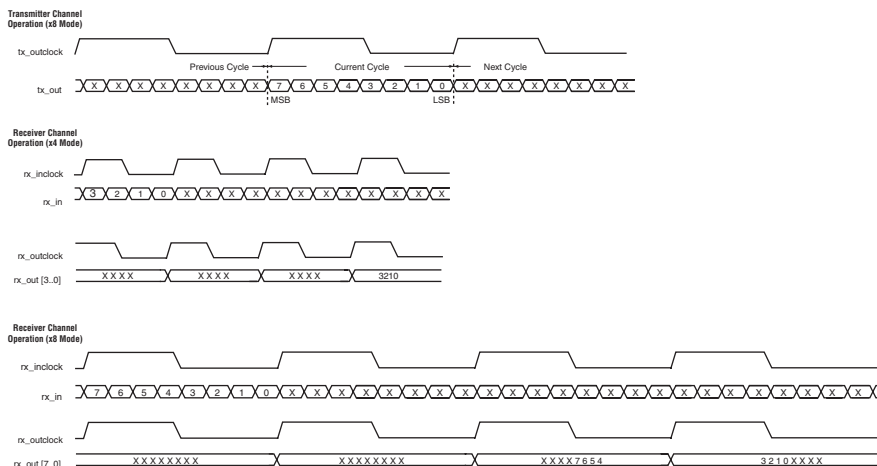*Figure 9–14. Bit Order for One Channel of Differential Data*



Table 9–2 shows the conventions for differential bit naming for 18 differential channels. The MSB and LSB positions increase with the number of channels used in a system.

| Table 9–2. LVDS Bit Naming  (Part 1 of 2) | | |
|---|---|---|
| **Receiver Channel Data Number** | **Internal 8-Bit Parallel Data** | |
| | **MSB Position** | **LSB Position** |
| 1 | 7 | 0 |
| 2 | 15 | 8 |
| 3 | 23 | 16 |
| 4 | 31 | 24 |
| 5 | 39 | 32 |
| 6 | 47 | 40 |
| 7 | 55 | 48 |
| 8 | 63 | 56 |
| 9 | 71 | 64 |
| 10 | 79 | 72 |
| 11 | 87 | 80 |

*Table 9–2. LVDS Bit Naming  (Part 2 of 2)*

| Receiver Channel Data Number | Internal 8-Bit Parallel Data | |
|:---:|:---:|:---:|
| | MSB Position | LSB Position |
| 12 | 95 | 88 |
| 13 | 103 | 96 |
| 14 | 111 | 104 |
| 15 | 119 | 112 |
| 16 | 127 | 120 |
| 17 | 135 | 128 |
| 18 | 143 | 136 |

### Receiver Skew Margin for Non-DPA

Changes in system environment, such as temperature, media (cable, connector, or PCB) loading effect, the receiver's setup and hold times, and internal skew, reduce the sampling window for the receiver. The timing margin between the receiver's clock input and the data input sampling window is called Receiver Skew Margin (RSKM). Figure 9–15 shows the relationship between the RSKM and the receiver's sampling window.

TCCS, RSKM, and the sampling window specifications are used for high-speed source-synchronous differential signals without DPA. When using DPA, these specifications are exchanged for the simpler single DPA jitter tolerance specification. For instance, the receiver skew is why each input with DPA selects a different phase of the clock, thus removing the requirement for this margin.

*Figure 9–15. Differential High-Speed Timing Diagram and Timing Budget for Non-DPA*
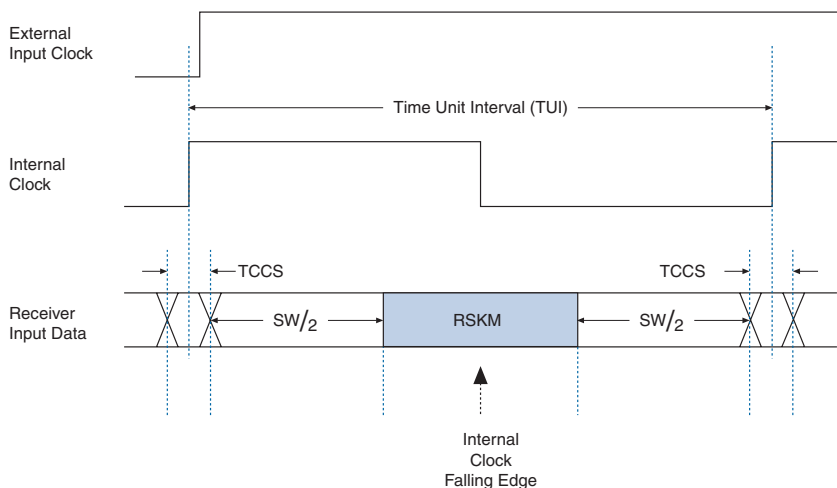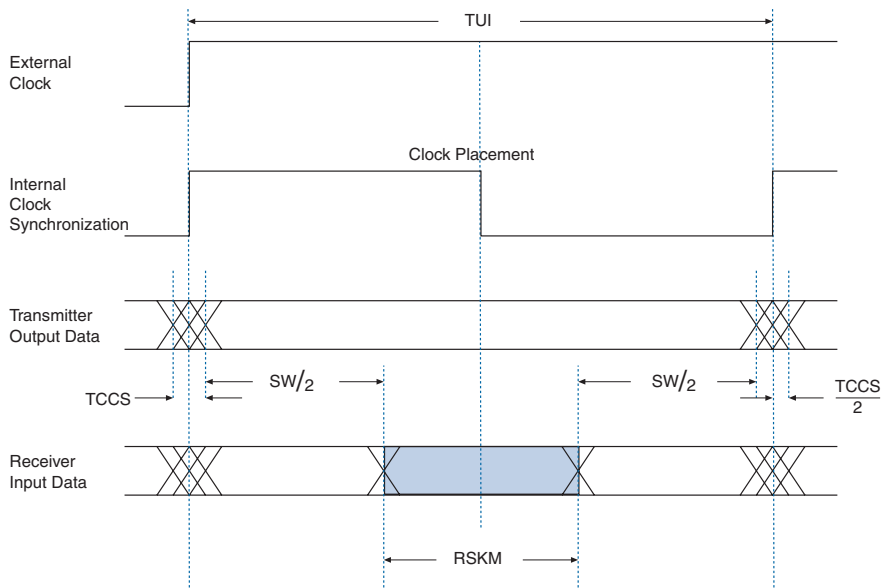
# Differential Pin Placement Guidelines

In order to ensure proper high-speed operation, differential pin placement guidelines have been established. The Quartus II compiler automatically checks that these guidelines are followed and will issue an error message if these guidelines are not met. PLL driving distance information is separated into guidelines with and without DPA usage.

## High-Speed Differential I/Os and Single-Ended I/Os

When a differential channel or channels of side banks are used (with or without DPA), you must adhere to the guidelines described in the following sections.

■ Single-ended I/Os are allowed in the same bank as the LVDS channels (with or without DPA) as long as the single-ended I/O standard uses the same $V_{CCIO}$ as the LVDS bank.

■ Single-ended inputs can be in the same logic array block (LAB) row. Outputs cannot be on the same LAB row with LVDS I/Os. If input registers are used in the I/O cell (IOC), single-ended inputs cannot be in the same LAB row as an LVDS SERDES block.

■ LVDS (non-SERDES) I/Os are allowed in the same row as LVDS SERDES but the use of IOC registers are not allowed.

■ Single-ended outputs are limited to 120 mA drive strength on LVDS banks (with or without DPA).

   ● LVTTL equation for maximum number of I/Os in an LVDS bank:

      ● 120 mA = (number of LVTTL outputs) × (drive strength of each LVTTL output)

   ● SSTL-2 equation:

      ● 120 mA = (number of SSTL-2 I/Os) × (drive strength of each output) ÷ 2

   ● LVTTL and SSTL-2 mix equation:

      ● 120 mA= (total drive strength of all LVTTL outputs) + (total drive strength of all SSTL2 outputs) ÷ 2

■ Single-ended inputs can be in the same LAB row as a differential channel using the SERDES circuitry; however, IOE input registers are not available for the single-ended I/Os placed in the same LAB row as differential I/Os. The same rule for input registers applies for non-SERDES differential inputs placed within the same LAB row as a SERDES differential channel. The input register must be implemented within the core logic. The same rule for input registers applies for non-SERDES differential inputs placed within the same LAB row as a SERDES differential channel.

■ Single-ended output pins must be at least one LAB row away from differential output pins, as shown in Figure 9–16.

*Figure 9–16. Single-Ended Output Pin Placement with Respect to Differential I/O Pins*



⊠ Single-Ended Output Pin

⊠ Differential I/O Pin

⊠ Single_Ended Input

Single-Ended Outputs
Not Allowed

Row Boundary
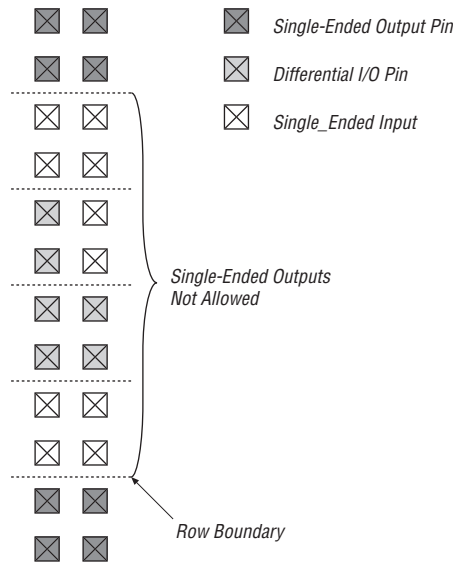
## DPA Usage Guidelines

Arria GX devices have differential receivers and transmitters on the Row banks of the device. Each receiver has a dedicated DPA circuit to align the phase of the clock to the data phase of its associated channel. When a channel or channels are used in DPA mode, the guidelines listed below must be adhered to.

### Fast PLL/DPA Channel Driving Distance

■ If the number of DPA channels driven by each center or corner fast PLL exceeds 25 LAB rows, Altera recommends you implement data realignment (bit-slip) circuitry for all the DPA channels.

■ If one center fast PLL drives DPA channels in the upper and lower banks, the other center fast PLL cannot be used for DPA.

*Using Corner and Center Fast PLLs*

■ If a differential bank is being driven by two fast PLLs, where the corner PLL is driving one group and the center fast PLL is driving another group, there must be at least one row of separation between the two groups of DPA channels (see Figure 9–17). The two groups can operate at independent frequencies. Not all the channels are bonded out of the die. Each LAB row is considered a channel, whether or not it has I/O support.

■ No separation is necessary if a single fast PLL is driving DPA channels as well as non-DPA channels as long as the DPA channels are contiguous.

*Figure 9–17. Usage of Corner and Center Fast PLLs Driving DPA Channels in a Single Bank*
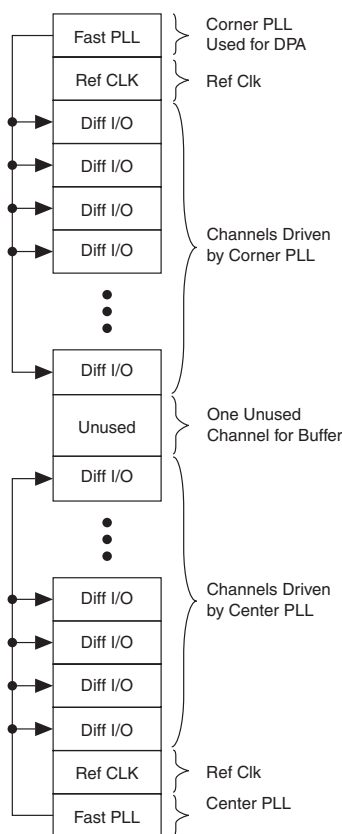
*Using Both Center Fast PLLs*

■ Both center fast PLLs can be used for DPA as long as they drive DPA channels in their adjacent quadrant only (see Figure 9–18).

■ Both center fast PLLs cannot be used for DPA if one of the fast PLLs drives the top and bottom banks, or if they are driving cross banks (for example, the lower fast PLL drives the top bank and the top fast PLL drives the lower bank).

*Figure 9–18. Center Fast PLL Usage When Driving DPA Channels*
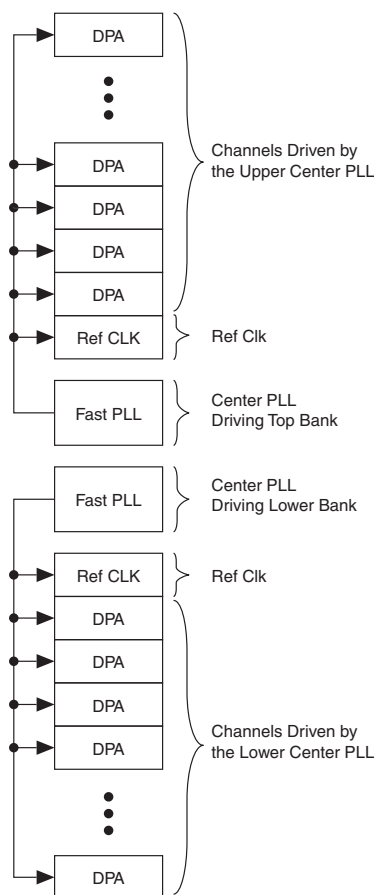
## Non-DPA Differential I/O Usage Guidelines

When a differential channel or channels of left or right banks are used in non-DPA mode, you must adhere to the guidelines in the following sections.

### Fast PLL/Differential I/O Driving Distance

■ Each fast PLL can drive all the channels in the entire bank, as shown in Figure 9–19.

*Figure 9–19. Fast PLL Driving Capability When Driving Non-DPA Differential Channels*
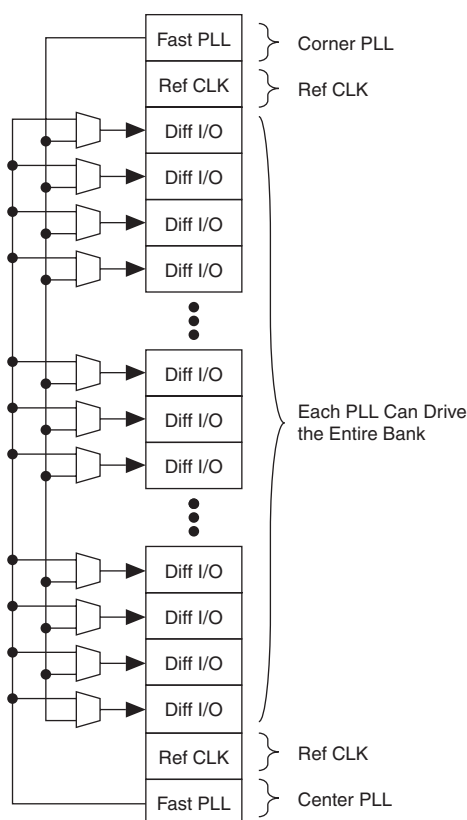
*Using Corner and Center Fast PLLs*

■ The corner and center fast PLLs can be used as long as the channels driven by separate fast PLLs do not have their transmitter or receiver channels interleaved. Figure 9–20 shows illegal placement of differential channels when using corner and center fast PLLs.

■ If one fast PLL is driving transmitter channels only, and the other fast PLL drives receiver channels only, the channels driven by those fast PLLs can overlap each other.

■ Center fast PLLs can be used for both transmitter and receiver channels.

*Figure 9–20. Illegal Placement of Interlaced Duplex Channels in an I/O Bank*
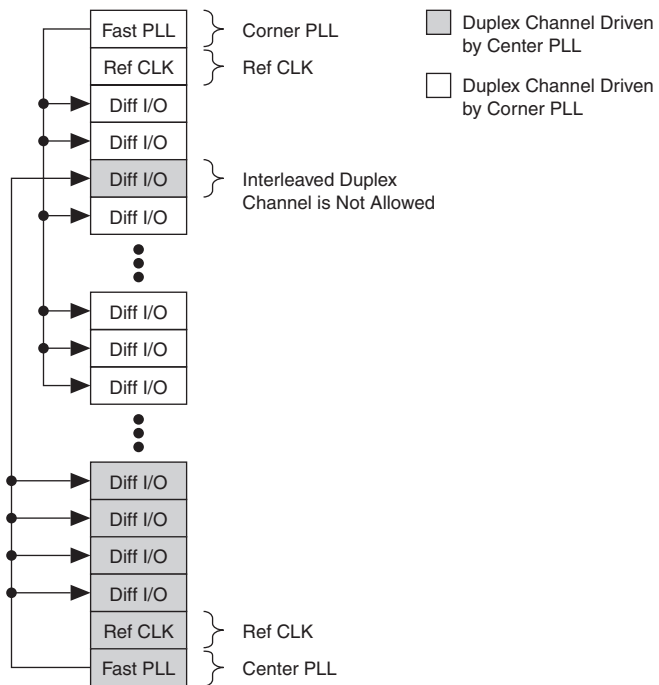


## Board Design Considerations

This section explains how to achieve the optimal performance from the Arria GX high-speed I/O block and ensure first-time success in implementing a functional design with optimal signal quality.

For more information about board layout recommendations and I/O pin terminations, refer to *AN 224: High-Speed Board Layout Guidelines*.

To achieve the best performance from the device, pay attention to the impedances of traces and connectors, differential routing, and termination techniques.

The Arria GX high-speed module generates signals that travel over the media at frequencies as high as 840 Mbps. Board designers should use the following guidelines:

■ Base board designs on controlled differential impedance. Calculate and compare all parameters such as trace width, trace thickness, and the distance between two differential traces.
■ Place external reference resistors as close to receiver input pins as possible.
■ Use surface mount components.
■ Avoid 90° or 45° corners.
■ Use high-performance connectors such as HMZD or VHDM connectors for backplane designs. Two suppliers of high-performance connectors are Teradyne Corp (www.teradyne.com) and Tyco International Ltd. (www.tyco.com).
■ Design backplane and card traces so that trace impedance matches the connector's or the termination's impedance.
■ Keep an equal number of vias for both signal traces.
■ Create equal trace lengths to avoid skew between signals. Unequal trace lengths also result in misplaced crossing points and system margins when the transmitter-channel-to-channel skew (TCCS) value increases.
■ Limit vias, because they cause impedance discontinuities.
■ Use the common bypass capacitor values such as 0.001, 0.01, and 0.1 μF to decouple the fast PLL power and ground planes. You can also use 0.0047 μF and 0.047 μF.
■ Keep switching TTL signals away from differential signals to avoid possible noise coupling.
■ Do not route transistor-to-transistor logic (TTL) clock signals to areas under or above the differential signals.
■ Route signals on adjacent layers orthogonally to each other.

## Conclusion

Arria GX high-speed differential inputs and outputs, with their DPA and data realignment circuitry, allow users to build a robust multi-Gigabit system. The DPA circuitry allows users to compensate for any timing skews resulting from physical layouts. The data realignment circuitry allows the devices to align the data packet between the transmitter and receiver. Together with the on-chip differential termination, Arria GX devices can be used as a single-chip solution for high-speed applications.

## Referenced Documents

This chapter references the following documents:

- *AN 224: High-Speed Board Layout Guidelines*
- *DC & Switching Characteristic*s chapter in volume 1 of the *Arria GX Device Handbook*
- *PLLs in Arria GX Devices* chapter in volume 2 of the *Arria GX Handbook*

## Document Revision History

Table 9–3 shows the revision history for this chapter.

*Table 9–3. Document Revision History*

| Date and Document Version | Changes Made | Summary of Changes |
|---|---|---|
| May 2008 v1.2 | Updated:<br>● "DPA Usage Guidelines"<br>● "Fast PLL/DPA Channel Driving Distance" | — |
| | Updated Figure 9–15. | — |
| | Minor text edits. | — |
| August 2007 v1.1 | Added the "Referenced Documents" section. | — |
| | Minor text edits. | — |
| May 2007 v1.0 | Initial release. | — |

# Section V. Digital Signal Processing (DSP)

This section provides information for design and optimization of digital signal processing (DSP) functions and arithmetic operations in the on-chip DSP blocks.

This section contains the following chapter:

■   Chapter 10, DSP Blocks in Arria GX Devices

## Revision History

Refer to each chapter for its own specific revision history. For information on when each chapter was updated, refer to the Chapter Revision Dates section, which appears in the full handbook.

## Introduction

Arria™ GX devices have dedicated digital signal processing (DSP) blocks optimized for DSP applications requiring high data throughput. These DSP blocks combined with the flexibility of programmable logic devices (PLDs), provide you with the ability to implement various high performance DSP functions easily. Complex systems such as CDMA2000, voice over Internet protocol (VoIP), and high-definition television (HDTV) require high performance DSP blocks to process data. These system designs typically use DSP blocks as finite impulse response (FIR) filters, complex FIR filters, fast Fourier transform (FFT) functions, discrete cosine transform (DCT) functions, and correlators.

Arria GX DSP blocks consist of a combination of dedicated blocks that perform multiplication, addition, subtraction, accumulation, and summation operations. You can configure these blocks to implement arithmetic functions like multipliers, multiply-adders and multiply-accumulators which are necessary for most DSP functions.

Along with the DSP blocks, the TriMatrix™ memory structures in Arria GX devices also support various soft multiplier implementations. The combination of soft multipliers and dedicated DSP blocks increases the number of multipliers available in Arria GX devices and provides you with a wide variety of implementation options and flexibility when designing your systems.

For more information about Arria GX devices respectively, see the *Arria GX Device Family Data Sheet* in volume 1 of the *Arria GX Device Handbook*.

This chapter contains the following sections:

## DSP Block Overview

Each Arria GX device has two to four columns of DSP blocks that efficiently implement multiplication, multiply-accumulate (MAC) and multiply-add functions. Figure 10–1 shows the arrangement of one of the DSP block columns with the surrounding LABs. Each DSP block can be configured to support:

■ Eight 9 × 9-bit multipliers
■ Four 18 × 18-bit multipliers
■ One 36 × 36-bit multiplier

*Figure 10–1. DSP Blocks Arranged in Columns with Adjacent LABs*
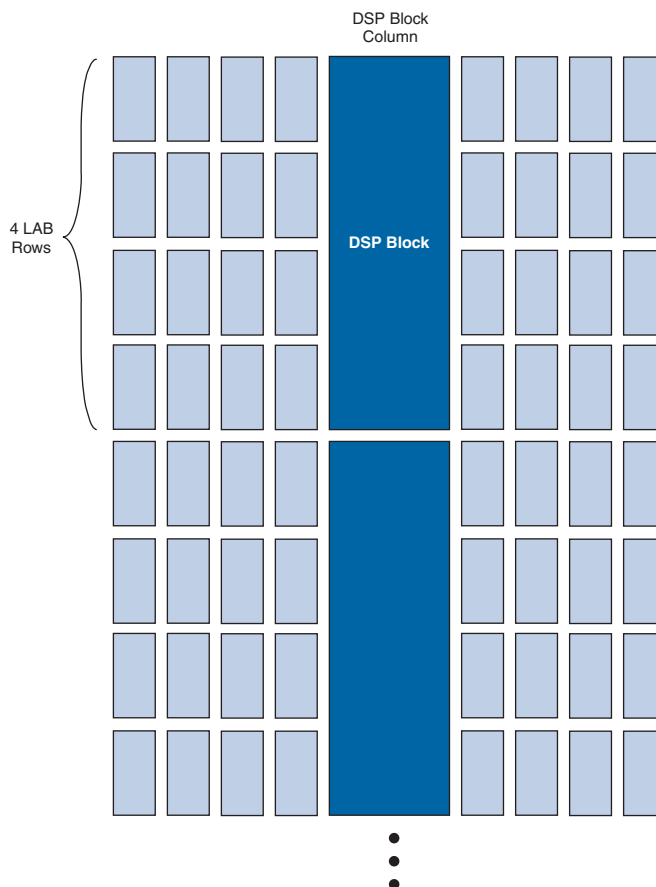
The multipliers then feed an adder or accumulator block within the DSP block. Arria GX device multipliers support rounding and saturation on Q1.15 input formats. The DSP block also has input registers that can be configured to operate in a shift register chain for efficient implementation of functions such as FIR filters. The accumulator within the DSP block can be initialized to any value and supports rounding and saturation on Q1.15 input formats to the multiplier. A single DSP block can be broken down to operate different configuration modes simultaneously.

☞ For more information on Q1.15 formatting, see the section "Saturation and Rounding" on page 10–11.

The number of DSP blocks per column and the number of columns available increases with device density.

Table 10–1 shows the number of DSP blocks in each Arria GX device and the multipliers that you can implement.

*Table 10–1. Number of DSP Blocks in Arria GX Devices* Note (1)

| Device | DSP Blocks | 9 × 9 Multipliers | 18 × 18 Multipliers | 36 × 36 Multipliers |
|---|---|---|---|---|
| EP1AGX20C | 10 | 80 | 40 | 10 |
| EP1AGX35C/D | 14 | 112 | 56 | 14 |
| EP1AGX50C/D | 26 | 208 | 104 | 26 |
| EP1AGX60C/D/E | 32 | 256 | 128 | 32 |
| EP1AGX90E | 44 | 352 | 176 | 44 |

*Note to Table 10–1:*
(1)  Each device has either the number of 9 × 9-, 18 × 18-, or 36 × 36-bit multipliers shown. The total number of multipliers for each device is not the sum of all the multipliers.

In addition to the DSP block multipliers, you can use the Arria GX device's TriMatrix memory blocks for soft multipliers. The availability of soft multipliers increases the number of multipliers available within the device. Table 10–2 shows the total number of multipliers available in Arria GX devices using DSP blocks and soft multipliers.

*Table 10–2. Number of Multipliers in Arria GX Devices*

| Device | DSP Blocks (18 × 18) | Soft Multipliers (16 × 16) (1), (2) | Total Multipliers (3), (4) |
|---|---|---|---|
| EP1AGX20C | 40 | 102 | 142 (3.55) |
| EP1AGX35C/D | 56 | 122 | 178 (3.18) |
| EP1AGX50C/D | 104 | 202 | 306 (2.94) |
| EP1AGX60C/D/E | 128 | 211 | 339 (2.65) |
| EP1AGX90E | 176 | 324 | 500 (2.84) |

*Notes to Table 10–2:*
(1) Soft multipliers implemented in sum of multiplication mode. RAM blocks are configured with 18-bit data widths and sum of coefficients up to 18-bits.
(2) Soft multipliers are only implemented in M4K and M512 TriMatrix memory blocks, not M-RAM blocks.
(3) The number in parentheses represents the increase factor, which is the total number of multipliers with soft multipliers divided by the number of 18 × 18 multipliers supported by DSP blocks only.
(4) The total number of multipliers may vary according to the multiplier mode used.

Refer to the *Arria GX Architecture* chapter in volume 1 of the *Arria GX Device Handbook* for more information about Arria GX TriMatrix memory blocks.

Refer to *AN 306: Implementing Multipliers in FPGA Devices* for more information on soft multipliers.

Figure 10–2 shows the DSP block configured for 18 × 18 multiplier mode.
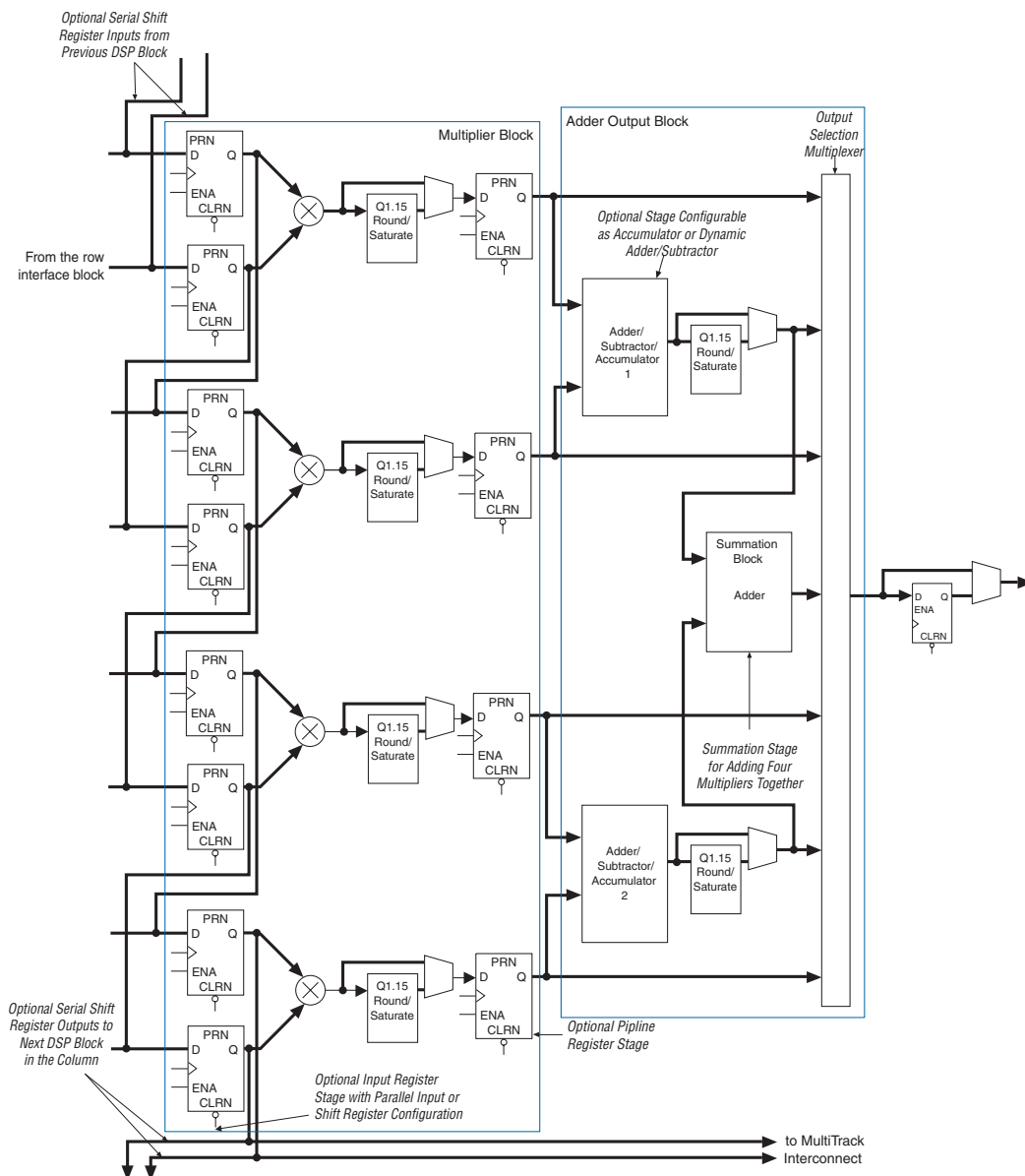
*Figure 10–2. DSP Block in 18 × 18 Mode*

Figure 10–3 shows the 9 × 9 multiplier configuration of the DSP block.

*Figure 10–3. DSP Block in 9 × 9 Mode*
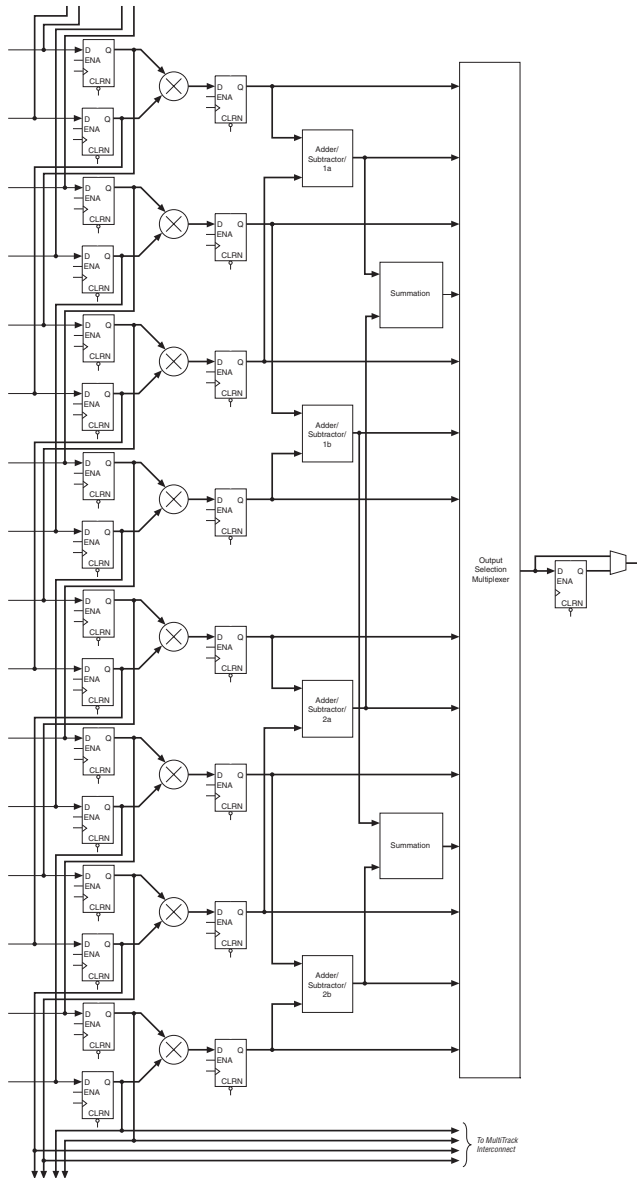
# Architecture

The DSP block consists of the following elements:

- A multiplier block
- An adder/subtractor/accumulator block
- A summation block
- Input and output interfaces
- Input and output registers

## Multiplier Block

Each multiplier block has the following elements:

- Input registers
- A multiplier block
- A rounding and/or saturation stage for Q1.15 input formats
- A pipeline output register

Figure 10–4 shows the multiplier block architecture.

*Figure 10–4. Multiplier Block Architecture*
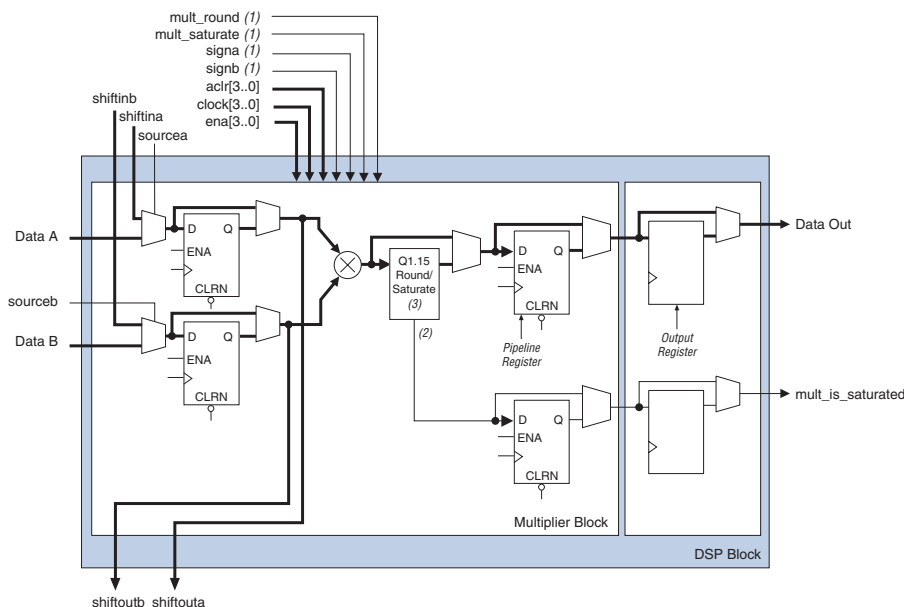


*Notes to Figure 10–4:*
(1) These signals are not registered or registered once to match the data path pipeline.
(2) You can send these signals through either one or two pipeline registers.
(3) The rounding and/or saturation is only supported in 18 × 18-bit signed multiplication for Q1.15 inputs.

*Input Registers*

Each multiplier operand can feed an input register or directly to the multiplier. The following DSP block signals control each input register within the DSP block:

- `clock[3..0]`
- `ena[3..0]`
- `aclr[3..0]`

The input registers feed the multiplier and drive two dedicated shift output lines, `shiftouta` and `shiftoutb`. The dedicated shift outputs from one multiplier block directly feed input registers of the adjacent multiplier below it within the same DSP block or the first multiplier in the next DSP block to form a shift register chain, as shown in Figure 10–5. The dedicated shift register chain spans a single column but longer shift register chains requiring multiple columns can be implemented using regular FPGA routing resources. Therefore, this shift register chain can be of any length up to 768 registers in the largest member of the Arria GX device family.

Shift registers are useful in DSP functions such as FIR filters. When implementing 9 × 9 and 18 × 18 multipliers, you do not need external logic to create the shift register chain because the input shift registers are internal to the DSP block. This implementation significantly reduces the LE resources required, avoids routing congestion, and results in predictable timing.

Arria GX DSP blocks allow you to dynamically select whether a particular multiplier operand is fed by regular data input or the dedicated shift register input using the `sourcea` and `sourceb` signals. A logic 1 value on the `sourcea` signal indicates that data A is fed by the dedicated scan-chain; a logic 0 value indicates that it is fed by regular data input. This feature allows the implementation of a dynamically loadable shift register where the shift register operates normally using the scan-chains and can also be loaded dynamically in parallel using the data input value. Figure 10–5 shows the shift register chain.

*Figure 10–5. Shift Register Chain Note (1)*
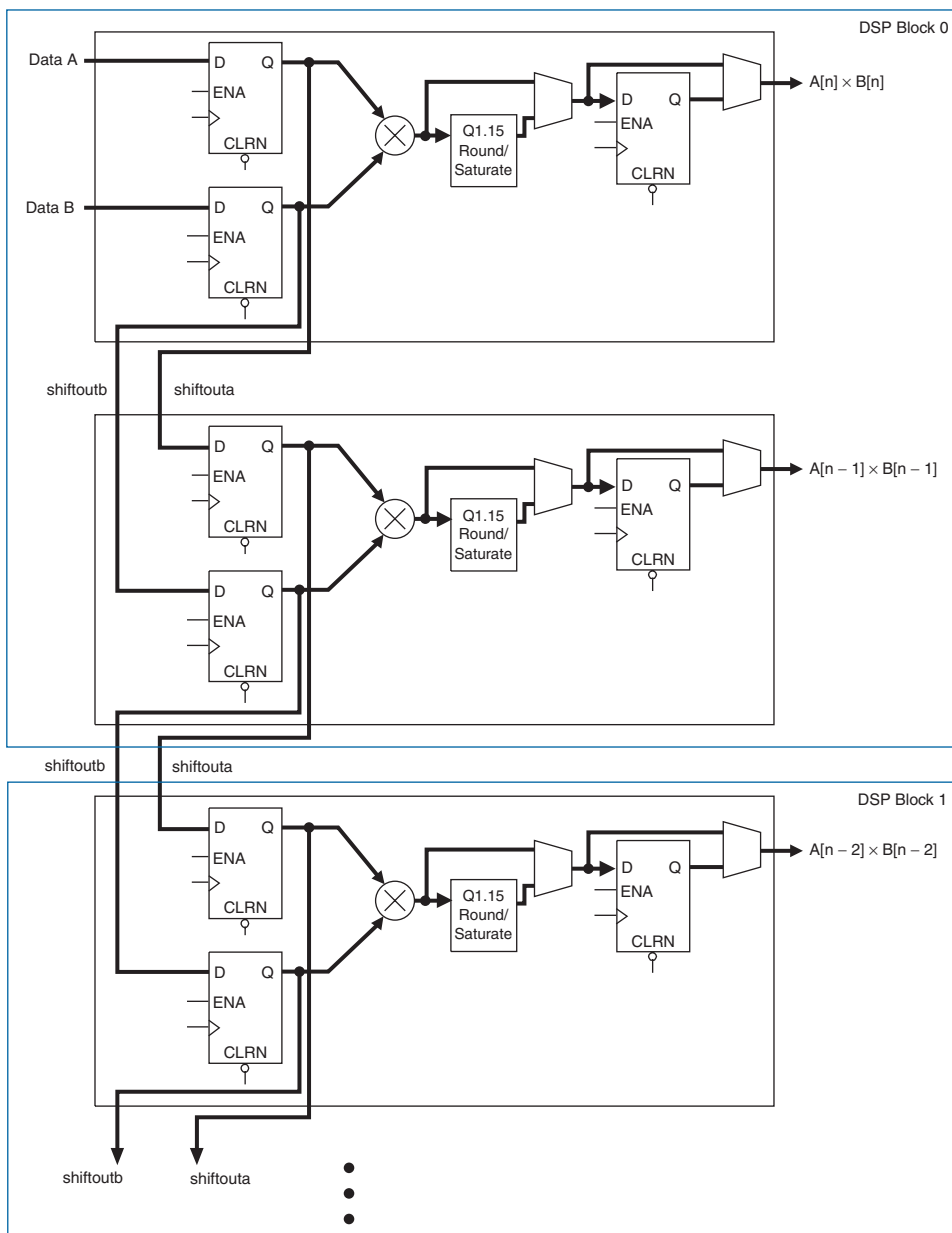


*Note to Figure 10–5:*
(1)    Either Data A or Data B input can be set to a parallel input for constant coefficient multiplication.

Table 10–3 shows the summary of input register modes for the DSP block.

| Table 10–3. Input Register Modes | | | |
|---|---|---|---|
| **Register Input Mode** | **9 × 9** | **18 × 18** | **36 × 36** |
| Parallel input | ✓ | ✓ | ✓ |
| Shift register input | ✓ | ✓ | - |

### Multiplier Stage

The multiplier stage supports 9 × 9, 18 × 18, or 36 × 36 multipliers as well as other smaller multipliers in between these configurations. See "Operational Modes" on page 10–18 for details. Depending on the data width of the multiplier, a single DSP block can perform many multiplications in parallel.

Each multiplier operand can be a unique signed or unsigned number. Two signals, signa and signb, control the representation of each operand respectively. A logic 1 value on the signa signal indicates that data A is a signed number while a logic 0 value indicates an unsigned number. Table 10–4 shows the sign of the multiplication result for the various operand sign representations. The result of the multiplication is signed if any one of the operands is a signed value.

| Table 10–4. Multiplier Sign Representation | | |
|---|---|---|
| **Data A (signa Value)** | **Data B (signb Value)** | **Result** |
| Unsigned (logic 0) | Unsigned (logic 0) | Unsigned |
| Unsigned (logic 0) | Signed (logic 1) | Signed |
| Signed (logic 1) | Unsigned (logic 0) | Signed |
| Signed (logic 1) | Signed (logic 1) | Signed |

There is only one signa and one signb signal for each DSP block. Therefore, all of the data A inputs feeding the same DSP block must have the same sign representation. Similarly, all of the data B inputs feeding the same DSP block must have the same sign representation. The multiplier offers full precision regardless of the sign representation.

☞ When the signa and signb signals are unused, the Quartus® II software sets the multiplier to perform unsigned multiplication by default.

*Saturation and Rounding*

The DSP blocks have hardware support to perform optional saturation and rounding after each 18 × 18 multiplier for Q1.15 input formats.

☞ Designs must use 18 × 18 multipliers for the saturation and rounding options because the Q1.15 input format requires 16-bit input widths.

☞ Q1.15 input format multiplication requires signed multipliers. The most significant bit (MSB) in the Q1.15 input format represents the value's sign bit. Use signed multipliers to ensure the proper sign extension during multiplication.

The Q1.15 format uses 16 bits to represent each fixed point input. The MSB is the sign bit, and the remaining 15-bits are used to represent the value after the decimal place (or the fractional value). This Q1.15 value is equivalent to an integer number representation of the 16-bits divided by $2^{15}$, as shown in the following equations.

$$-\frac{1}{2} = 1\ 100\ 0000\ 0000\ 0000 = -\frac{0x4000}{2^{15}}$$

$$\frac{1}{8} = 0\ 001\ 0000\ 0000\ 0000 = \frac{0x1000}{2^{15}}$$

All Q1.15 numbers are between –1 and 1.

When performing multiplication, even though the Q1.15 input only uses 16 of the 18 multiplier inputs, the entire 18-bit input bus is transmitted to the multiplier. This is similar to a 1.17 input, where the two least significant bits (LSBs) are always 0.

The multiplier output will be a 2.34 value (36 bits total) before performing any rounding or saturation. The two MSBs are sign bits. Since the output only requires one sign bit, you can ignore one of the two MSBs, resulting in a Q1.34 value before rounding or saturation.

When the design performs saturation, the multiplier output gets saturated to 0x7FFFFFFF in a 1.31 format. This uses bits [34..3] of the overall 36-bit multiplier output. The three LSBs are set to 0.

The DSP block obtains the `mult_is_saturated` or `accum_is_saturated` overflow signal value from the LSB of the multiplier or accumulator output. Therefore, whenever saturation occurs, the LSB of the multiplier or accumulator output sends a 1 to the

mult_is_saturated or accum_is_saturated overflow signal. At all other times, this overflow signal is 0 when saturation is enabled or reflects the value of the LSB of the multiplier or accumulator output.

When the design performs rounding, it adds 0x00008000 in 1.31 format to the multiplier output, and it only uses bits [34..15] of the overall 36-bit multiplier output. Adding 0x00008000 in 1.31 format to the 36-bit multiplier result is equivalent to adding 0x0 0004 0000 in 2.34 format. The 16 LSBs are set to 0. Figure 10–6 shows which bits are used when the design performs rounding and saturation for the multiplication.

*Figure 10–6. Rounding and Saturation Bits*

**18 × 18 Multiplication**
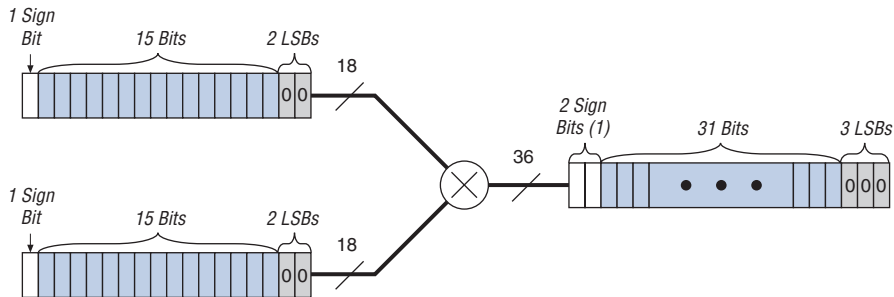


**Saturated Output Result**
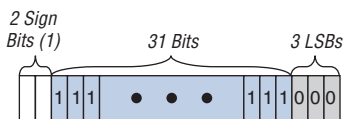


**Rounded Output Result**
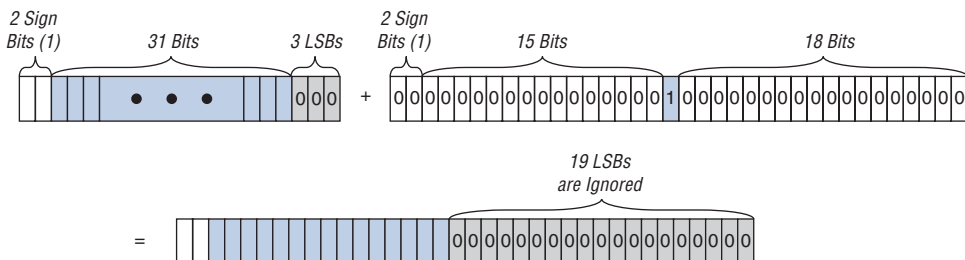


*Note to Figure 10–6:*
(1)    Both sign bits are the same. The design only uses one sign bit, and the other one is ignored.

If the design performs a `multiply_accumulate` or `multiply_add` operation, the multiplier output is input to the adder/subtractor/accumulator blocks as a 2.31 value, and the three LSBs are 0.

*Pipeline Registers*

The output from the multiplier can feed a pipeline register or this register can be bypassed. Pipeline registers may be implemented for any multiplier size and increase the DSP block's maximum performance, especially when using the subsequent DSP block adder stages. Pipeline registers split up the long signal path between the adder/subtractor/accumulator block and the adder/output block, creating two shorter paths.

## Adder/Output Block

The adder/output block has the following elements:

- An adder/subtractor/accumulator block
- A summation block
- An output select multiplexer
- Output registers

The adder/output block can be configured as:

- An output interface
- An accumulator which can be optionally loaded
- A one-level adder
- A two-level adder with dynamic addition/subtraction control on the first-level adder
- The final stage of a 36-bit multiplier, 9 × 9 complex multiplier, or 18 × 18 complex multiplier

The output select multiplexer sets the output configuration of the DSP block. The output registers can be used to register the output of the adder/output block.

☞ The adder/output block cannot be used independently from the multiplier.

Figure 10–7 shows the adder/output block architecture.

*Figure 10–7. Adder/Output Block Architecture*
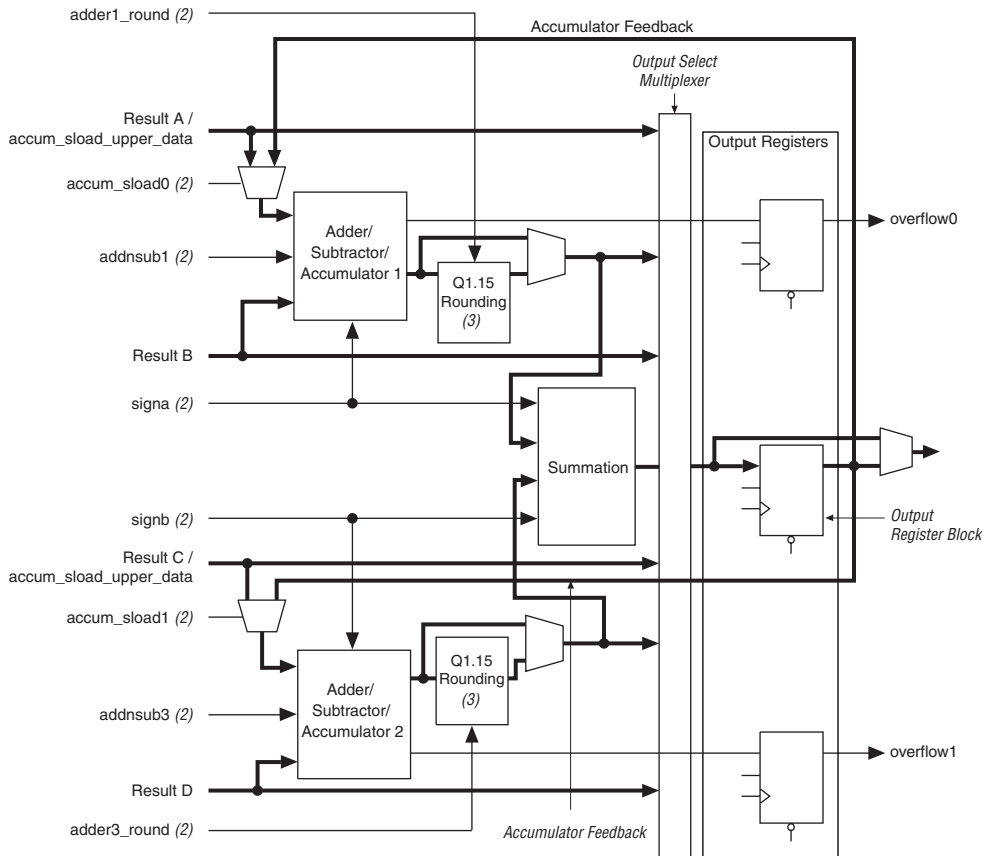


**Notes to Figure 10–7:**
(1) The adder/output block is in 18 × 18 mode. In 9 × 9 mode, there are four adder/subtractor blocks and two summation blocks.
(2) You can send these signals through a pipeline register. The pipeline length can be set to 1 or 2.
(3) Q1.15 inputs are not available in 9 × 9 or 36 × 36 modes.

### Adder/Subtractor/Accumulator Block

The adder/subtractor/accumulator block is the first level adder stage of the adder/output block. This block can be configured as an accumulator or as an adder/subtractor.

## Accumulator

When the adder/subtractor/accumulator is configured as an accumulator, the output of the adder/output block feeds back to the accumulator as shown in Figure 10–7. The accumulator can be set up to perform addition only, subtraction only or the `addnsub` signal can be used to dynamically control the accumulation direction. A logic 1 value on the `addnsub` signal indicates that the accumulator is performing addition while a logic 0 value indicates subtraction.

Each accumulator can be cleared by either clearing the DSP block output register or by using the `accum_sload` signal. The accumulator clear using the `accum_sload` signal is independent from the resetting of the output registers so the accumulation can be cleared and a new one can begin without losing any clock cycles. The `accum_sload` signal controls a feedback multiplexer that specifies that the output of the multiplier should be summed with a zero instead of the accumulator feedback path.

The accumulator can also be initialized/preloaded with a non-zero value using the `accum_sload` signal and the `accum_sload_upper_data` bus with one clock cycle latency. Preloading the accumulator is done by adding the result of the multiplier with the value specified on the `accum_sload_upper_data` bus. As in the case of the accumulator clearing, the `accum_sload` signal specifies to the feedback multiplexer that the `accum_sload_upper_data` signal should feed the accumulator instead of the accumulator feedback signal. The `accum_sload_upper_data` signal only loads the upper 36-bits of the accumulator. To load the entire accumulator, the value for the lower 16-bits must be sent through the multiplier feeding that accumulator with the multiplier set to perform a multiplication by one.

The overflow signal will go high on the positive edge of the clock when the accumulator detects an overflow or underflow. The overflow signal will stay high for only one clock cycle after an overflow or underflow is detected even if the overflow or underflow condition is still present. A latch external to the DSP block has to be used to preserve the overflow signal indefinitely or until the latch is cleared.

The DSP blocks support Q1.15 input format saturation and rounding in each accumulator. The following signals are available that can control if saturation or rounding or both is performed to the output of the accumulator:

- `accum_round`
- `accum_saturation`
- `accum_is_saturated` output

Each DSP block has two sets of `accum_round` and `accum_saturation` signals which control if rounding or saturation is performed on the accumulator output respectively (one set of signals for each accumulator). Rounding and saturation of the accumulator output is only available when implementing a 16 × 16 multiplier-accumulator to conform to the bit widths required for Q1.15 input format computation. A logic 1 value on the `accum_round` and `accum_saturation` signal indicates that rounding or saturation is performed while a logic 0 indicates that no rounding or saturation is performed. A logic 1 value on the `accum_is_saturated` output signal tells you that saturation has occurred to the result of the accumulator.

Figure 10–10 shows the DSP block configured to perform multiplier-accumulator operations.

**Adder/Subtractor**

The `addnsub1` or `addnsub3` signals specify whether you are performing addition or subtraction. A logic 1 value on the `addnsub1` or `addnsub3` signals indicates that the adder/subtractor is performing addition while a logic 0 value indicates subtraction. These signals can be dynamically controlled using logic external to the DSP block. If the first stage is configured as a subtractor, the output is A – B and C – D.

The adder/subtractor block share the same `signa` and `signb` signals as the multiplier block. The `signa` and `signb` signals can be pipelined with a latency of one or two clock cycles or not.

The DSP blocks support Q1.15 input format rounding (not saturation) after each adder/subtractor. The `addnsub1_round` and `addnsub3_round` signals determine if rounding is performed to the output of the adder/subtractor.

The `addnsub1_round` signal controls the rounding of the top adder/subtractor and the `addnsub3_round` signal controls the rounding of the bottom adder/subtractor. Rounding of the adder output is only available when implementing a 16 × 16 multiplier-adder to conform to the bit widths required for Q1.15 input format computation. A logic 1 value on the `addnsub_round` signal indicates that rounding is performed while a logic 0 indicates that no rounding is performed.

*Summation Block*

The output of the adder/subtractor block feeds an optional summation block, which is an adder block that sums the outputs of both adder/subtractor blocks. The summation block is used when more than two multiplier results are summed. This is useful in applications such as FIR filtering.

### Output Select Multiplexer

The outputs of the different elements of the adder/output block are routed through an output select multiplexer. Depending on the operational mode of the DSP block, the output multiplexer selects whether the outputs of the DSP blocks comes from the outputs of the multiplier block, the outputs of the adder/subtractor/accumulator, or the output of the summation block. The output select multiplier configuration is set automatically by software, based on the DSP block operational mode you specify.

### Output Registers

You can use the output registers to register the DSP block output. The following signals can control each output register within the DSP block:

■ `clock[3..0]`
■ `ena[3..0]`
■ `aclr[3..0]`

The output registers can be used in any DSP block operational mode.

☞ The output registers form part of the accumulator in the multiply-accumulate mode.

👣 Refer to the *Arria GX Architecture* chapter in volume 1 of the *Arria GX Device Handbook* for more information on the DSP block routing and interface.

## Operational Modes

The DSP block can be used in one of four basic operational modes, or a combination of two modes, depending on the application needs. Table 10–5 shows the four basic operational modes and the number of multipliers that can be implemented within a single DSP block depending on the mode.

| Table 10–5. DSP Block Operational Modes | | | |
|---|---|---|---|
| **Mode** | **Number of Multipliers** | | |
| | **9 × 9** | **18 × 18** | **36 × 36** |
| Simple multiplier | Eight multipliers with eight product outputs | Four multipliers with four product outputs | One multiplier |
| Multiply accumulate | - | Two 52-bit multiply-accumulate blocks | - |
| Two-multiplier adder | Four two-multiplier adder (two 9 9 complex multiply) | Two two-multiplier adder (one 18 18 complex multiply) | - |
| Four-multiplier adder | Two four-multiplier adder | One four-multiplier adder | - |

The Quartus II software includes megafunctions used to control the mode of operation of the multipliers. After you make the appropriate parameter settings using the megafunction's MegaWizard® Plug-In Manager, the Quartus II software automatically configures the DSP block.

Arria GX DSP blocks can operate in different modes simultaneously. For example, a single DSP block can be broken down to operate a $9 \times 9$ multiplier as well as an $18 \times 18$ multiplier-adder where both multiplier's input a and input b have the same sign representations. This increases DSP block resource efficiency and allows you to implement more multipliers within an Arria GX device. The Quartus II software automatically places multipliers that can share the same DSP block resources within the same block.

Additionally, you can set up each Arria GX DSP block to dynamically switch between the following three modes:

■ Up to four 18-bit independent multipliers
■ Up to two 18-bit multiplier-accumulators
■ One 36-bit multiplier

Each half of an Arria GX DSP block has separate mode control signals, which allows you to implement multiple 18-bit multipliers or multiplier-accumulators within the same DSP block and dynamically switch them independently (if they are in separate DSP block halves). If the design requires a 36-bit multiplier, you must switch the entire DSP block to accommodate it since the multiplier requires the entire DSP block. The smallest input bit width that supports dynamic mode switching is 18 bits.

## Simple Multiplier Mode

In simple multiplier mode, the DSP block performs individual multiplication operations for general-purpose multipliers and for applications such as computing equalizer coefficient updates which require many individual multiplication operations.

### 9- and 18-Bit Multipliers

Each DSP block multiplier can be configured for 9- or 18-bit multiplication. A single DSP block can support up to eight individual 9 × 9 multipliers or up to four individual 18 × 18 multipliers. For operand widths up to 9-bits, a 9 × 9 multiplier will be implemented and for operand widths from 10- to 18-bits, an 18 × 18 multiplier will be implemented. Figure 10–8 shows the DSP block in the simple multiplier operation mode.

*Figure 10–8. Simple Multiplier Mode*
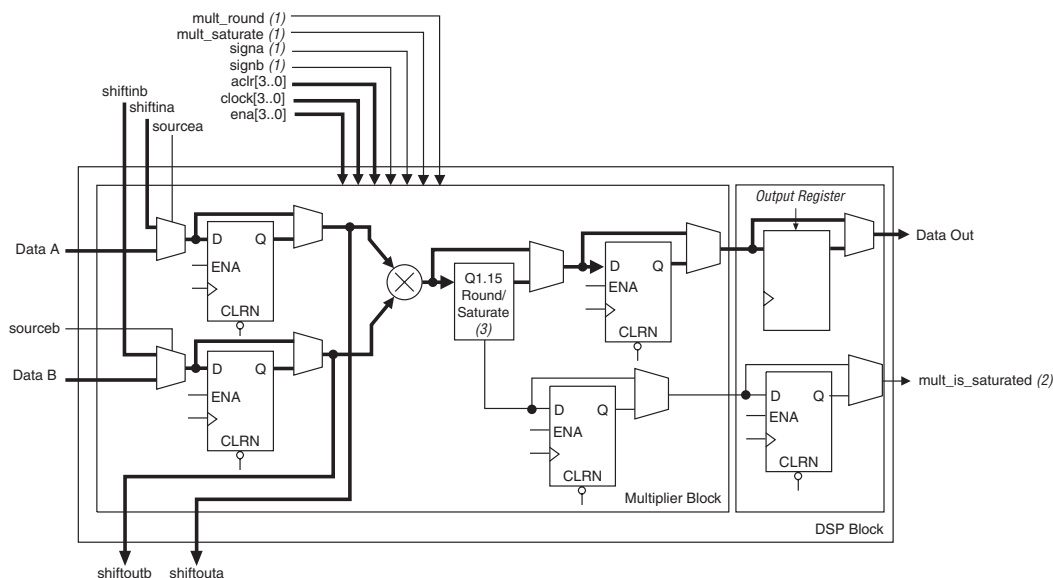


*Notes to Figure 10–8:*
(1)    These signals are not registered or registered once to match the data path pipeline.
(2)    This signal has the same latency as the data path.
(3)    The rounding and saturation is only supported in 18 × 18-bit signed multiplication for Q1.15 inputs.

The multiplier operands can accept signed integers, unsigned integers or a combination of both. The signa and signb signals can be changed dynamically and can be registered in the DSP block. Additionally, the multiplier inputs and result can be registered independently. The pipeline registers within the DSP block can be used to pipeline the multiplier result, increasing the performance of the DSP block.

### 36-Bit Multiplier

The 36-bit multiplier is also a simple multiplier mode but uses the entire DSP block, including the adder/output block to implement the 36 × 36-bit multiplication operation. The device inputs 18-bit sections of the 36-bit input into the four 18-bit multipliers. The adder/output block adds the partial products obtained from the multipliers using the summation block. Pipeline registers can be used between the multiplier stage and the summation block to speed up the multiplication. The 36 × 36-bit multiplier supports signed, unsigned as well as mixed sign multiplication. Figure 10–9 shows the DSP block configured to implement a 36-bit multiplier.
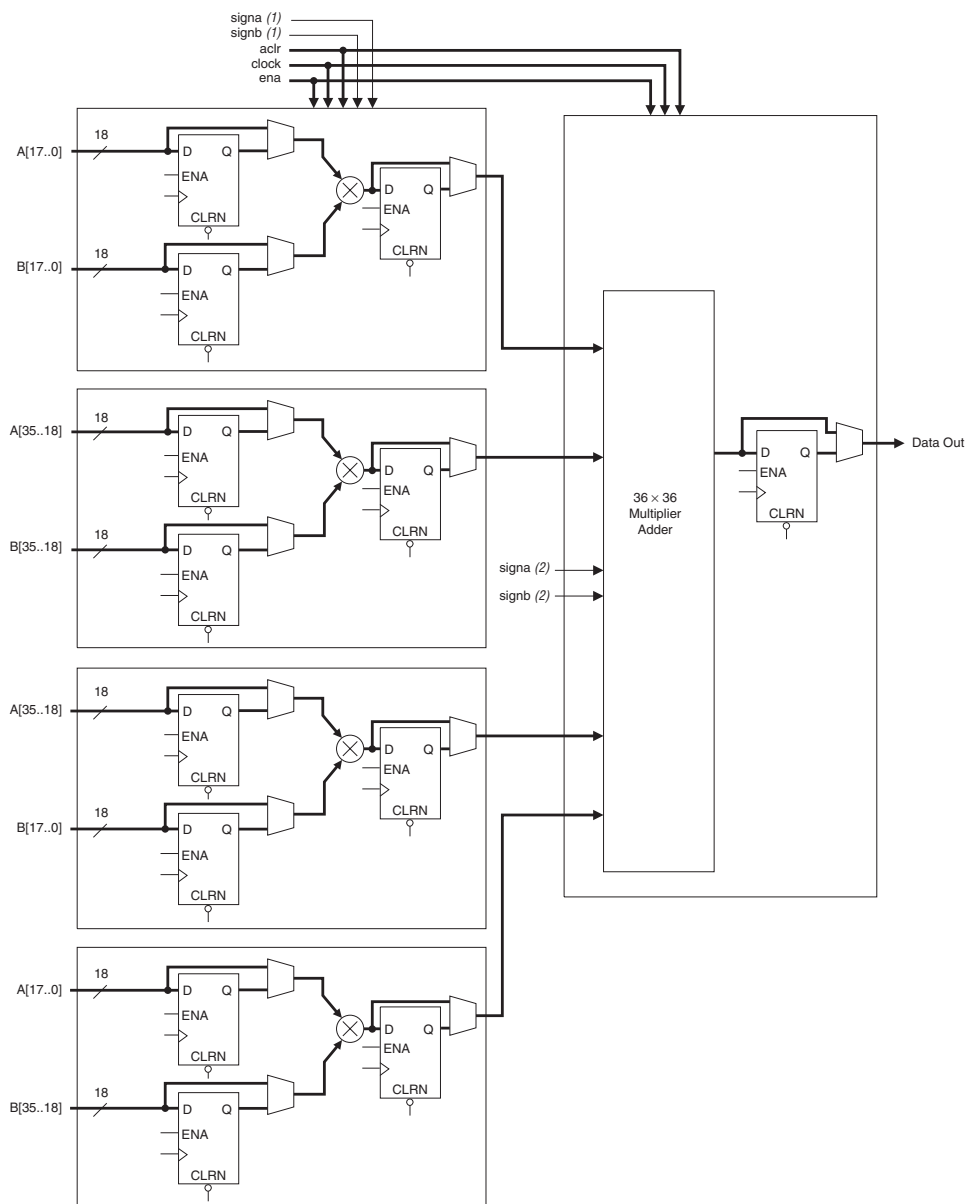
*Figure 10–9. 36-Bit Multiplier*



*Notes to Figure 10–9:*
(1)  These signals are either not registered or registered once to match the pipeline.
(2)  These signals are either not registered, registered once, or registered twice to match the data path pipeline.

The 36-bit multiplier is useful for applications requiring more than 18-bit precision, for example, for mantissa multiplication of precision floating-point arithmetic applications.

## Multiply Accumulate Mode

In multiply accumulate mode, the output of the multiplier stage feeds the adder/output block which is configured as an accumulator or subtractor. Figure 10–10 shows the DSP block configured to operate in multiply accumulate mode.

*Figure 10–10. Multiply Accumulate Mode*
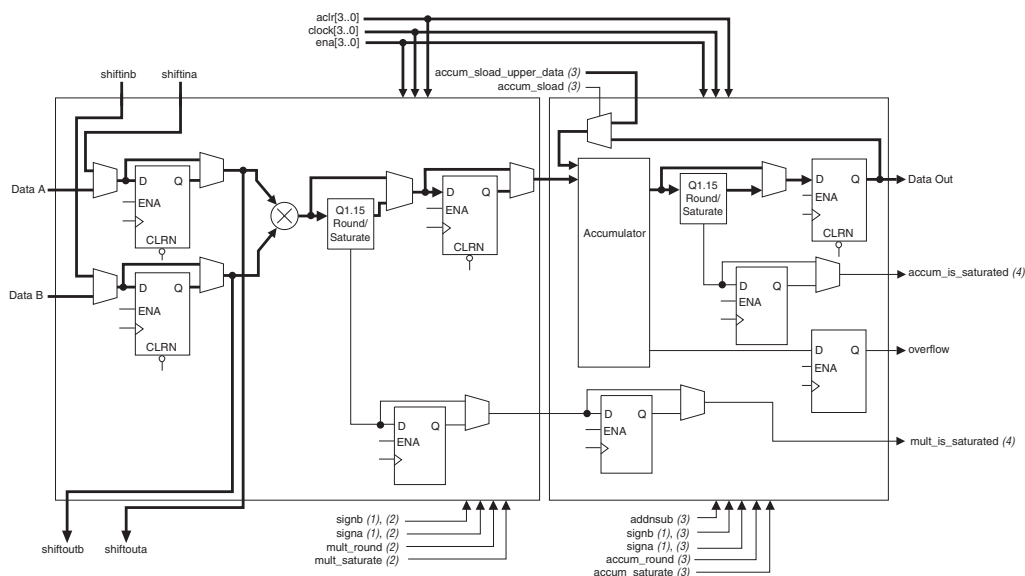


*Notes to Figure 10–10:*
(1) The `signa` and `signb` signals are the same in the multiplier stage and the adder/output block.
(2) These signals are not registered or registered once to match the data path pipeline.
(3) You can send these signals through either one or two pipeline registers.
(4) These signals match the latency of the data path.

A single DSP block can implement up to two independent 18-bit multiplier accumulators. The Quartus II software implements smaller multiplier accumulators by tying the unused lower-order bits of the 18-bit multiplier to ground.

The multiplier accumulator output can be up to 52-bits wide to account for a 36-bit multiplier result with 16-bits of accumulation. In this mode, the DSP block uses output registers and the accum_sload and overflow signals. The accum_sload signal can be used to clear the accumulator so that a new accumulation operation can begin without losing any clock cycles. This signal can be unregistered or registered once or twice. The accum_sload signal can also be used to preload the accumulator with a value specified on the accum_sload_upper_data signal with a one clock cycle penalty. The accum_sload_upper_data signal only loads the upper 36-bits (bits [51..16] of the accumulator). To load the entire accumulator, the value for the lower 16-bits (bits [15..0]) must be sent through the multiplier feeding that accumulator with the multiplier set to perform a multiplication by one. Bits [17..16] are overlapped by both the accum_sload_upper_data signal and the multiplier output. Either one of these signals can be used to load bits [17..16].

The overflow signal indicates an overflow or underflow in the accumulator. This signal gets updated every clock cycle due to a new accumulation operation every cycle. To preserve the signal, an external latch can be used. The addnsub signal can be used to specify if an accumulation or subtraction is performed dynamically.

☞ The DSP block can implement just an accumulator (without multiplication) by specifying a multiply by one at the multiplier stage followed by an accumulator to force the Quartus II software to implement the function within the DSP block.

## Multiply Add Mode

In multiply add mode, the output of the multiplier stage feeds the adder/output block which is configured as an adder or subtractor to sum or subtract the outputs of two or more multipliers. The DSP block can be configured to implement either a two-multiply add (where the outputs of two multipliers are added/subtracted together) or a four-multiply add function (where the outputs of four multipliers are added or subtracted together).

☞ The adder block within the DSP block can only be used if it follows multiplication operations.

*Two-Multiplier Adder*

In the two-multiplier adder configuration, the DSP block can implement four 9-bit or smaller multiplier adders or two 18-bit multiplier adders. The adders can be configured to take the sum of both multiplier outputs or the difference of both multiplier outputs. You have the option to vary the summation/subtraction operation dynamically. These multiply add functions are useful for applications such as FFTs and complex FIR filters. Figure 10–11 shows the DSP block configured in the two-multiplier adder mode.

*Figure 10–11. Two-Multiplier Adder Mode*
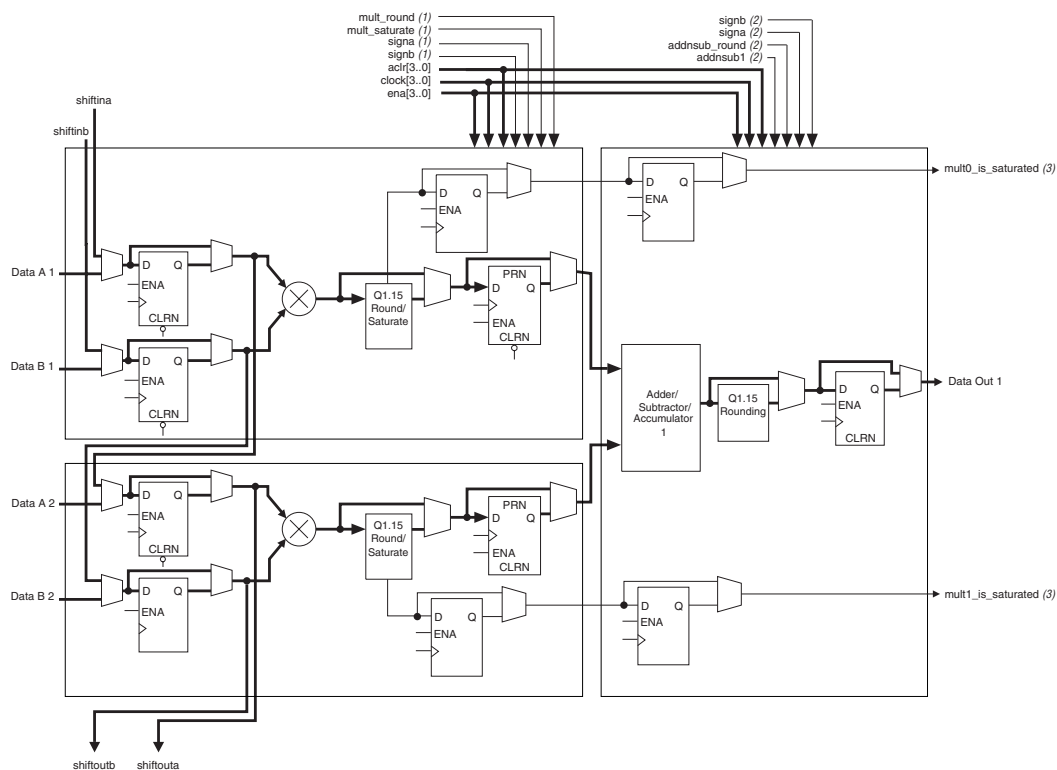


*Notes to Figure 10–11:*
(1)  These signals are not registered or registered once to match the data path pipeline.
(2)  You can send these signals through a pipeline register. The pipeline length can be set to 1 or 2.
(3)  These signals match the latency of the data path.

# Complex Multiply

The DSP block can be configured to implement complex multipliers using the two-multiplier adder mode. A single DSP block can implement one 18 × 18-bit complex multiplier or two 9 × 9-bit complex multipliers.

A complex multiplication can be written as:

$$(a + \mathrm{j}b)\,(c + \mathrm{j}d) = ((ac) - (bd)) + \mathrm{j}\,((ad) + (bc))$$

To implement this complex multiplication within the DSP block, the real part $((ac) - (bd))$ is implemented using two multipliers feeding one subtractor block while the imaginary part $((ad) + (bc))$ is implemented using another two multipliers feeding an adder block, for data up to 18-bits. Figure 10–12 shows an 18-bit complex multiplication. For data widths up to 9-bits, a DSP block can perform two separate complex multiplication operations using eight 9-bit multipliers feeding four adder/subtractor/accumulator blocks. Resources external to the DSP block must be used to route the correct real and imaginary input components to the appropriate multiplier inputs to perform the correct computation for the complex multiplication operation.

*Figure 10–12. Complex Multiplier Using Two-Multiplier Adder Mode*
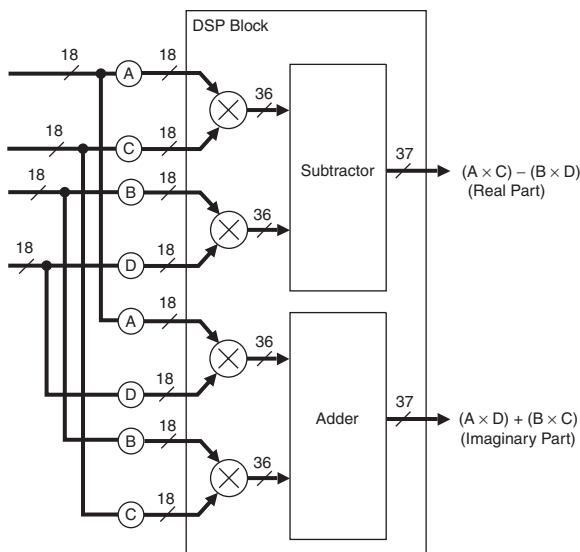
*Four-Multiplier Adder*

In the four-multiplier adder configuration, the DSP block can implement one 18 × 18 or two individual 9 × 9 multiplier adders. These modes are useful for implementing one-dimensional and two-dimensional filtering applications. The four-multiplier adder is performed in two addition stages. The outputs of two of the four multipliers are initially summed in the two first-stage adder/subtractor/accumulator blocks. The results of these two adder/subtractor/accumulator blocks are then summed in the final stage summation block to produce the final four-multiplier adder result. Figure 10–13 shows the DSP block configured in the four-multiplier adder mode.

*Figure 10–13. Four-Multiplier Adder Mode*
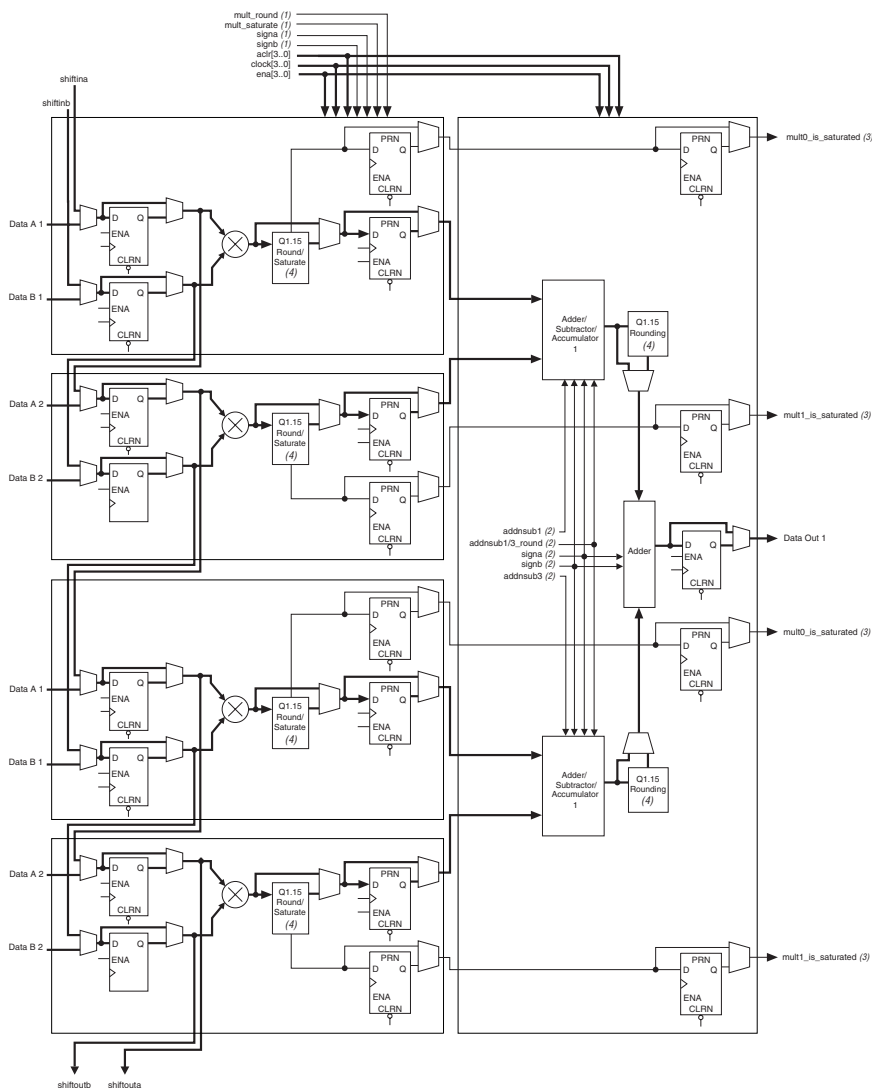


*Notes to Figure 10–13:*
(1) These signals are not registered or registered once to match the data path pipeline.
(2) You should send these signals through the pipeline register to match the latency of the data path.
(3) These signals match the latency of the data path.
(4) The rounding and saturation is only supported in 18 × 18-bit signed multiplication for Q1.15 inputs.

# FIR Filter

The four-multiplier adder mode can be used to implement FIR filter and complex FIR filter applications. To do this, the DSP block is set up in a four-multiplier adder mode with one set of input registers configured as shift registers using the dedicated shift register chain. The set of input registers configured as shift registers will contain the input data while the inputs configured as regular inputs will hold the filter coefficients. Figure 10–14 shows the DSP block configured in the four-multiplier adder mode using input shift registers.

*Figure 10–14. FIR Filter Implemented Using the Four-Multiplier Adder Mode with Input Shift Registers*
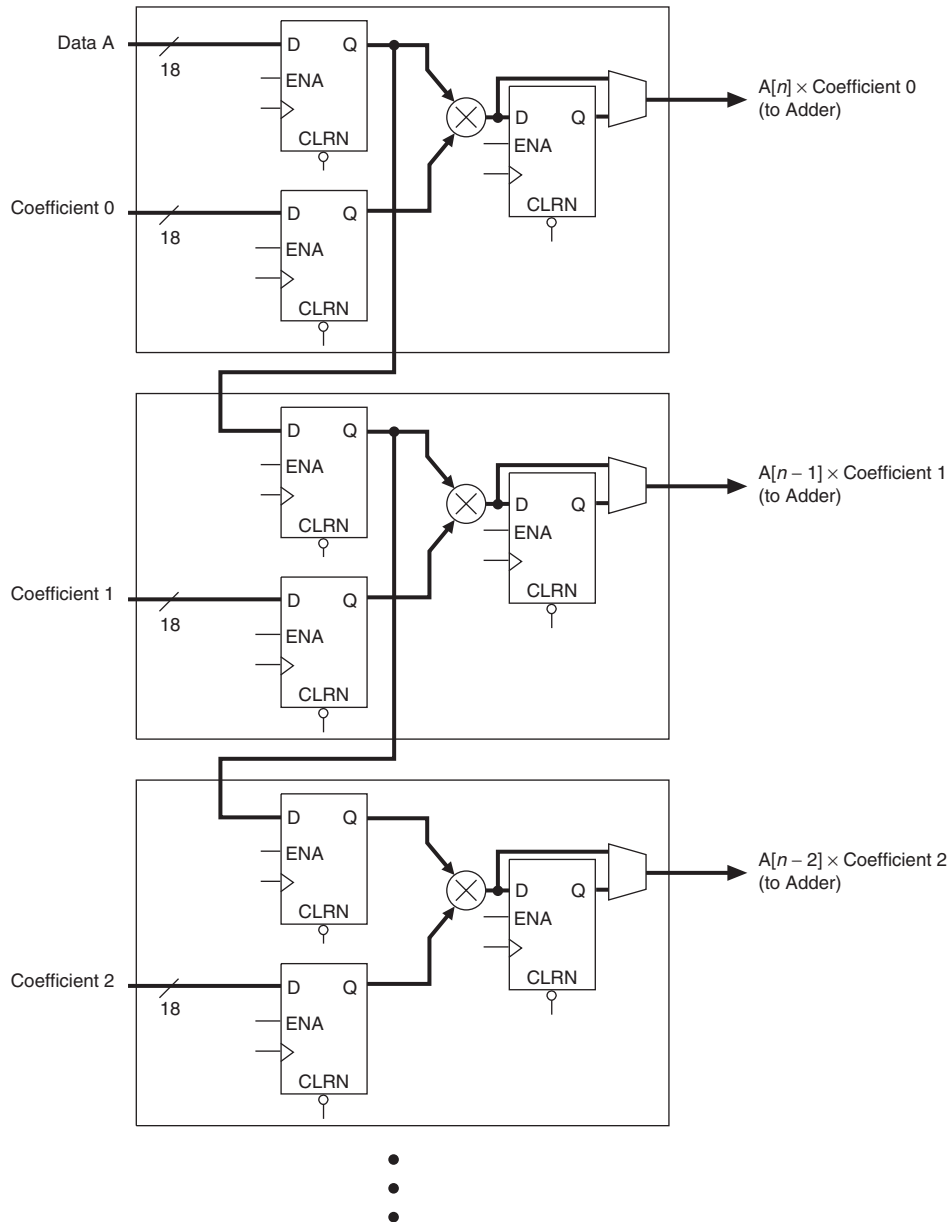
The built-in input shift register chain within the DSP block eliminates the need for shift registers externally to the DSP block in logic elements (LEs). This architecture feature simplifies the filter design and improves the filter performance because all the filter circuitry is localized within the DSP block.

☞ Input shift registers for the 36-bit simple multiplier mode have to be implemented using external registers to the DSP block.

A single DSP block can implement a four tap 18-bit FIR filter. For filters larger than four taps, the DSP blocks can be cascaded with additional adder stages implemented using LEs.

## Software Support

Altera provides two distinct methods for implementing various modes of the DSP block in your design: instantiation and inference. Both methods use the following three Quartus II megafunctions:

■ `lpm_mult`
■ `altmult_add`
■ `altmult_accum`

You can instantiate the megafunctions in the Quartus II software to use the DSP block. Alternatively, with inference, you can create a HDL design and synthesize it using a third-party synthesis tool like LeonardoSpectrum™ or Synplify or Quartus II Native Synthesis that infers the appropriate megafunction by recognizing multipliers, multiplier adders, and multiplier accumulators. Using either method, the Quartus II software maps the functionality to the DSP blocks during compilation.

👣 See Quartus II On-Line Help for instructions on using the megafunctions and the MegaWizard Plug-In Manager.

👣 For more information, see the *Design and Synthesis* section in volume 1 of the *Quartus II Development Software Handbook*.

## Conclusion

The Arria GX device DSP blocks are optimized to support DSP applications requiring high data throughput such as FIR filters, FFT functions and encoders. These DSP blocks are flexible and can be configured to implement one of several operational modes to suit a particular application. The built-in shift register chain, adder/subtractor/accumulator block and the summation block minimizes the amount of external logic required to implement these functions, resulting in efficient resource utilization and improved performance and data throughput for DSP applications. The Quartus II

software, together with the LeonardoSpectrum™ and Synplify software provide a complete and easy-to-use flow for implementing these multiplier functions in the DSP blocks.

## Referenced Documents

This chapter references the following documents:

- *AN 306: Implementing Multipliers in FPGA Devices*
- *Arria GX Architecture* chapter in volume 1 of the *Arria GX Device Handbook*
- *Arria GX Device Family Data Sheet* in volume 1 of the *Arria GX Device Handbook*
- *Design and Synthesis* section in volume 1 of the *Quartus II Development Software Handbook*

## Document Revision History

Table 10–6 shows the revision history for this chapter.

*Table 10–6. Document Revision History*

| Date and Document Version | Changes Made | Summary of Changes |
|---|---|---|
| May 2008, v1.2 | Minor text edits. | — |
| August 2007, v1.1 | Added the "Referenced Documents" section. | — |
| | Minor text edits. | — |
| May 2007, v1.0 | Initial Release | — |

# Section VI. Configuration & Remote System Upgrades

This section provides configuration information for all of the supported configuration schemes for Arria™ GX devices. These configuration schemes use either a microprocessor, configuration device, or download cable. There is detailed information on how to design with Altera enhanced configuration devices which includes information on how to manage multiple configuration files and access the on-chip FLASH memory space. The last chapter shows designers how to perform remote and local upgrades for their designs.

This section contains the following chapters:

- Chapter 11, Configuring Arria GX Devices

- Chapter 12, Remote System Upgrades with Arria GX Devices

- Chapter 13, IEEE 1149.1 (JTAG) Boundary-Scan Testing for Arria GX Devices

## Revision History

Refer to each chapter for its own specific revision history. For information on when each chapter was updated, refer to the Chapter Revision Dates section, which appears in the full handbook.

# 11. Configuring Arria GX Devices

## Introduction

Arria™ GX II devices use SRAM cells to store configuration data. Because SRAM memory is volatile, configuration data must be downloaded to Arria GX devices each time the device powers up. Arria GX devices can be configured using one of five configuration schemes: the fast passive parallel (FPP), active serial (AS), passive serial (PS), passive parallel asynchronous (PPA), and Joint Test Action Group (JTAG) configuration schemes. All configuration schemes use either an external controller (for example, a MAX® II device or microprocessor) or a configuration device.

This chapter contains the following sections:

### Configuration Devices

The Altera® enhanced configuration devices (EPC16, EPC8, and EPC4) support a single-device configuration solution for high-density devices and can be used in the FPP and PS configuration schemes. They are ISP-capable through their JTAG interface. The enhanced configuration devices are divided into two major blocks, the controller and the flash memory.

For information on enhanced configuration devices, refer to the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* and the *Altera Enhanced Configuration Devices* chapters in volume 2 of the *Configuration Handbook*.

The Altera serial configuration devices (EPCS64, EPCS16, and EPCS4) support a single-device configuration solution for Arria GX devices and are used in the AS configuration scheme. Serial configuration devices offer a low cost, low pin count configuration solution.

For information on serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* chapter in volume 2 of the *Configuration Handbook*.

The EPC2 configuration devices provide configuration support for the PS configuration scheme. The EPC2 device is ISP-capable through its JTAG interface. The EPC2 device can be cascaded to hold large configuration files.

For more information on EPC2 configuration devices, refer to the *Configuration Devices for SRAM-Based LUT Devices Data Sheet* chapter in volume 2 of the *Configuration Handbook*.

## Configuration Schemes

The configuration scheme is selected by driving the Arria GX device MSEL pins either high or low, as shown in Table 11–1. The MSEL pins are powered by the $V_{CCINT}$ power supply of the bank they reside in. The MSEL[3..0] pins have 5-kΩ internal pull-down resistors that are always active. During power-on reset (POR) and during reconfiguration, the MSEL pins have to be at LVTTL $V_{IL}$ and $V_{IH}$ levels to be considered a logic low and logic high.

☞ To avoid any problems with detecting an incorrect configuration scheme, hard-wire the MSEL[] pins to $V_{CCPD}$ and GND, without any pull-up or pull-down resistors. Do not drive the MSEL[] pins by a microprocessor or another device.

*Table 11–1. Arria GX Configuration Schemes  (Part 1 of 2)*

| Configuration Scheme | MSEL3 | MSEL2 | MSEL1 | MSEL0 |
|---|---|---|---|---|
| Fast passive parallel (FPP) | 0 | 0 | 0 | 0 |
| Passive parallel asynchronous (PPA) | 0 | 0 | 0 | 1 |
| Passive serial (PS) | 0 | 0 | 1 | 0 |
| Remote system upgrade FPP *(1)* | 0 | 1 | 0 | 0 |
| Remote system upgrade PPA *(1)* | 0 | 1 | 0 | 1 |
| Remote system upgrade PS *(1)* | 0 | 1 | 1 | 0 |
| Fast AS (40 MHz) *(2)* | 1 | 0 | 0 | 0 |
| Remote system upgrade fast AS (40 MHz) *(2)* | 1 | 0 | 0 | 1 |
| FPP with decompression feature enabled *(3)* | 1 | 0 | 1 | 1 |
| Remote system upgrade FPP with decompression feature enabled *(1)*, *(3)* | 1 | 1 | 0 | 0 |
| AS (20 MHz) *(2)* | 1 | 1 | 0 | 1 |

| Table 11–1. Arria GX Configuration Schemes  (Part 2 of 2) | | | | |
|---|---|---|---|---|
| **Configuration Scheme** | **MSEL3** | **MSEL2** | **MSEL1** | **MSEL0** |
| Remote system upgrade AS (20 MHz) *(2)* | 1 | 1 | 1 | 0 |
| JTAG-based configuration *(5)* | *(4)* | *(4)* | *(4)* | *(4)* |

Notes to Table 11–1:
(1) These schemes require that you drive the RUnLU pin to specify either remote update or local update. For more information about remote system upgrades in Arria GX devices, refer to the *Remote System Upgrades With Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook.*
(2) Only the EPCS16 and EPCS64 devices support up to a 40 MHz DCLK. Other EPCS devices support up to a 20 MHz DCLK. Refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* in volume 2 of the *Configuration Handbook* for more information.
(3) These modes are only supported when using a MAX II device or a microprocessor with flash memory for configuration. In these modes, the host system must output a DCLK that is 4× the data rate.
(4) Do not leave the MSEL pins floating. Connect them to $V_{CCPD}$ or ground. These pins support the non-JTAG configuration scheme used in production. If only JTAG configuration is used, you should connect the MSEL pins to ground.
(5) JTAG-based configuration takes precedence over other configuration schemes, which means MSEL pin settings are ignored.

Arria GX devices offer decompression and remote system upgrade features. Arria GX devices can receive a compressed configuration bitstream and decompress this data in real-time, reducing storage requirements and configuration time. You can make real-time system upgrades from remote locations of your Arria GX designs with the remote system upgrade feature.

Table 11–2 shows the uncompressed configuration file sizes for Arria GX devices.

| Table 11–2. Arria GX Uncompressed .rbf Sizes    *Note (1)* | | |
|---|---|---|
| **Device** | **Data Size (Bits)** | **Data Size (MBytes)** |
| EP1AGX20 | 9,640,672 | 1.205 |
| EP1AGX35 | 9,640,672 | 1.205 |
| EP1AGX50 | 16,951,824 | 2.119 |
| EP1AGX60 | 16,951,824 | 2.119 |
| EP1AGX90 | 25,699,104 | 3.212 |

Note to Table 11–2:
(1) **.rbf**: Raw Binary File.

Use the data in Table 11–2 to estimate the file size before design compilation. Different configuration file formats, such as a Hexidecimal (**.hex**) or Tabular Text File (**.ttf**) format, will have different file sizes. However, for any specific version of the Quartus® II software, any design targeted for the same device will have the same uncompressed configuration file size. If you are using compression, the file size can vary after each compilation because the compression ratio is dependent on the design.

This chapter explains the Arria GX device configuration features and describes how to configure Arria GX devices using the supported configuration schemes. This chapter provides configuration pin descriptions and the Arria GX device configuration file formats. In this chapter, the generic term device(s) includes all Arria GX devices.

For more information on setting device configuration options or creating configuration files, refer to the *Software Settings* section in volume 2 of the *Configuration Handbook*.

## Configuration Features

Arria GX devices offer configuration data decompression to reduce configuration file storage and remote system upgrades to allow you to remotely update your Arria GX designs. Table 11–3 summarizes which configuration features can be used in each configuration scheme.

**Table 11–3. Arria GX Configuration Features**

| Configuration Scheme | Configuration Method | Decompression | Remote System Upgrade |
|---|---|---|---|
| FPP | MAX II device or a Microprocessor with flash memory | ✔ *(1)* | ✔ |
| | Enhanced Configuration Device | ✔ *(2)* | ✔ |
| AS | Serial Configuration Device | ✔ | ✔ *(3)* |
| PS | MAX II device or a Microprocessor with flash memory | ✔ | ✔ |
| | Enhanced Configuration Device | ✔ | ✔ |
| | Download cable | ✔ | |
| PPA | MAX II device or a Microprocessor with flash memory | | ✔ |
| JTAG | MAX II device or a Microprocessor with flash memory | | |

*Notes to Table 11–3:*
(1) In these modes, the host system must send a `DCLK` that is 4× the data rate.
(2) The enhanced configuration device decompression feature is available, while the Arria GX decompression feature is not available.
(3) Only remote update mode is supported when using the AS configuration scheme. Local update mode is not supported.

## Configuration Data Decompression

Arria GX devices support configuration data decompression, which saves configuration memory space and time. This feature allows you to store compressed configuration data in configuration devices or other memory and transmit this compressed bitstream to Arria GX devices. During configuration, Arria GX devices decompress the bitstream in real time and programs its SRAM cells.

☞ Preliminary data indicates that compression typically reduces configuration bitstream size by 35 to 55%.

Arria GX devices support decompression in the FPP (when using a MAX II device/microprocessor + flash), AS, and PS configuration schemes. Decompression is not supported in the PPA configuration scheme nor in JTAG-based configuration.

☞ When using FPP mode, the intelligent host must provide a DCLK that is 4× the data rate. Therefore, the configuration data must be valid for four DCLK cycles.

The decompression feature supported by Arria GX devices is different from the decompression feature in enhanced configuration devices (EPC16, EPC8, and EPC4 devices), although they both use the same compression algorithm. The data decompression feature in the enhanced configuration devices allows them to store compressed data and decompress the bitstream before transmitting it to the target devices. When using Arria GX devices in FPP mode with enhanced configuration devices, the decompression feature is available only in the enhanced configuration device, not the Arria GX device.

In PS mode, use the Arria GX decompression feature because sending compressed configuration data reduces configuration time. Do not use both the Arria GX device and the enhanced configuration device decompression features simultaneously. The compression algorithm is not intended to be recursive and could expand the configuration file instead of compressing it further.

When you enable compression, the Quartus II software generates configuration files with compressed configuration data. This compressed file reduces the storage requirements in the configuration device or flash memory, and decreases the time needed to transmit the bitstream to the Arria GX device. The time required by an Arria GX device to decompress a configuration file is less than the time needed to transmit the configuration data to the device.

There are two ways to enable compression for Arria GX bitstreams: before design compilation (in the **Compiler Settings** menu) and after design compilation (in the **Convert Programming Files** window).

To enable compression in the project's compiler settings, select **Device** under the **Assignments** menu to bring up the **Settings** window. After selecting your Arria GX device, open the **Device & Pin Options** window, and in the **General** settings tab, enable the check box for **Generate compressed bitstreams** (as shown in Figure 11–1).

*Figure 11–1. Enabling Compression for Arria GX Bitstreams in Compiler Settings*
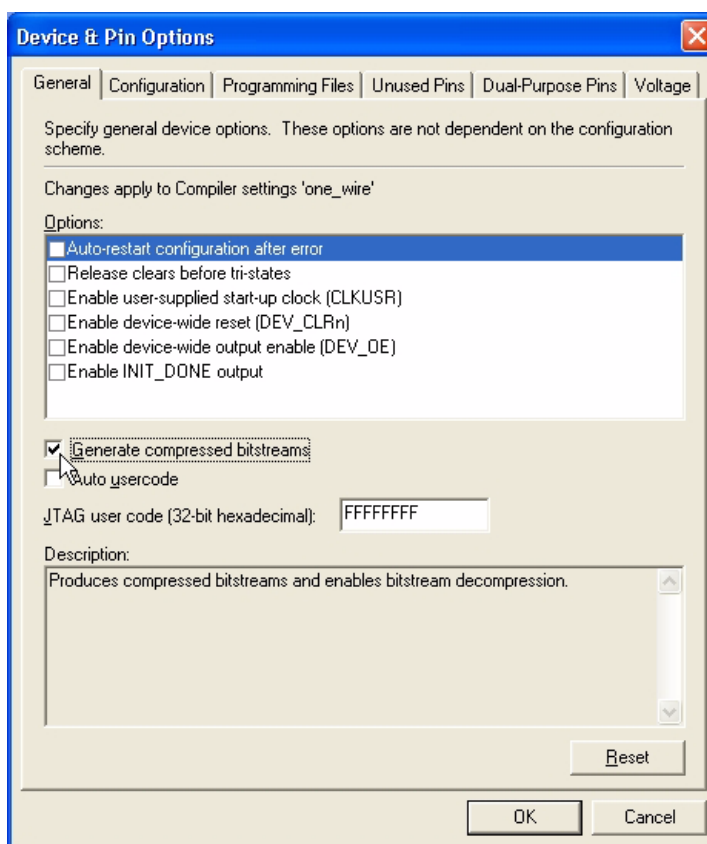
Compression can also be enabled when creating programming files from the **Convert Programming Files** window by following these steps:

1. Click **Conv ert Programming Files** (File menu).

2. Select the programming file type (POF, SRAM HEXOUT, RBF, or TTF).

3. For POF output files, select a configuration device.

4. In the **Input files to convert** box, select **SOF Data**.

5. Select **Add File** and add an Arria GX device SOF(s).

6. Select the name of the file you added to the **SOF Data** area and click **Properties**.

7. Check the **Compression** check box.

When multiple Arria GX devices are cascaded, you can selectively enable the compression feature for each device in the chain if you are using a serial configuration scheme. Figure 11–2 depicts a chain of two Arria GX devices. The first Arria GX device has compression enabled and receives a compressed bitstream from the configuration device. The second Arria GX device has the compression feature disabled and receives uncompressed data.

In a multi-device FPP configuration chain, all Arria GX devices in the chain must either enable of disable the decompression feature. You can not selectively enable the compression feature for each device in the chain because of the DATA and DCLK relationship.
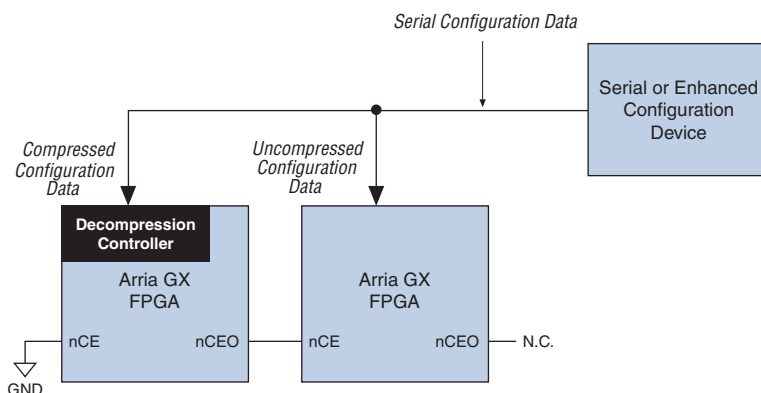
You can generate programming files for this setup from the **Convert Programming Files** window (File menu) in the Quartus II software.

## Remote System Upgrade

Arria GX devices feature remote and local update.

For more information about this feature, refer to the *Remote System Upgrades with Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*.

## Power-On Reset Circuit

The POR circuit keeps the entire system in reset until the power supply voltage levels have stabilized on power-up. Upon power-up, the device does not release nSTATUS until $V_{CCINT}$, $V_{CCPD}$, and $V_{CCIO}$ of banks 3, 4, 7, and 8 are above the device's POR trip point. On power down, $V_{CCINT}$ is monitored for brown-out conditions.

The passive serial mode (MSEL[3..0] = 0010) and the Fast passive parallel mode (MSEL[3..0] = 0000) always enable bank 3 to use the lower POR trip point consistent with 1.8- and 1.5-V signaling, regardless of the VCCSEL setting. For all other configuration modes, VCCSEL selects the POR trip point level. Refer to "VCCSEL Pin" on page 11–9 for more details.

In Arria GX devices, the pin-selectable option `PORSEL` allows you to select between a typical POR time setting of 12 ms or 100 ms. In both cases, you can extend the POR time by using an external component to assert the `nSTATUS` pin low.

## $V_{CCPD}$ Pins

Arria GX devices also offer a new power supply, $V_{CCPD}$, which must be connected to 3.3-V in order to power the 3.3-V/2.5-V buffer available on the configuration input pins and JTAG pins. $V_{CCPD}$ applies to all the JTAG input pins (`TCK`, `TMS`, `TDI`, and `TRST`) and the configuration pins when `VCCSEL` is connected to ground. Refer to Table 11–4 for information on the pins affected by `VCCSEL`.

☞      $V_{CCPD}$ must ramp-up from 0-V to 3.3-V within 100 ms. If $V_{CCPD}$ is not ramped up within this specified time, your Arria GX device will not configure successfully. If your system does not allow for a $V_{CCPD}$ ramp-up time of 100 ms or less, you must hold `nCONFIG` low until all power supplies are stable.

## VCCSEL Pin

The `VCCSEL` pin selects the type of input buffer used on configuration input pins and it selects the POR trip point voltage level for $V_{CCIO}$ bank 3 powered by VCCIO3 pins.

The configuration input pins and the `PLL_ENA` pin (Table 11–4) have a dual buffer design. These pins have a 3.3-V/2.5-V input buffer and a 1.8-V/1.5-V input buffer. The `VCCSEL` input pin selects which input buffer is used during configuration. The 3.3-V/2.5-V input buffer is powered by $V_{CCPD}$, while the 1.8-V/1.5-V input buffer is powered by

$V_{CCIO}$. After configuration, the dual-purpose configuration pins are powered by the $V_{CCIO}$ pins. Table 11–4 shows the pins affected by VCCSEL.

*Table 11–4. Pins Affected by the Voltage Level at VCCSEL*

| Pin | VCCSEL = LOW (connected to GND) | VCCSEL = HIGH (connected to $V_{CCPD}$) |
|---|---|---|
| nSTATUS (when used as an input) | 3.3/2.5-V input buffer is selected. Input buffer is powered by $V_{CCPD}$. | 1.8/1.5-V input buffer is selected. Input buffer is powered by $V_{CCIO}$ of the I/O bank. These input buffers are 3.3-V tolerant. |
| nCONFIG | | |
| CONF_DONE (when used as an input) | | |
| DATA[7..0] | | |
| nCE | | |
| DCLK (when used as an input) | | |
| CS | | |
| nWS | | |
| nRS | | |
| nCS | | |
| CLKUSR | | |
| DEV_OE | | |
| DEV_CLRn | | |
| RUnLU | | |
| PLL_ENA | | |

VCCSEL is sampled during power-up. Therefore, the VCCSEL setting cannot change on-the-fly or during a reconfiguration. The VCCSEL input buffer is powered by $V_{CCINT}$ and has an internal 5-kΩ pull-down resistor that is always active.

☞ VCCSEL must be hardwired to $V_{CCPD}$ or GND.

A logic high selects the 1.8-V/1.5-V input buffer, and a logic low selects the 3.3-V/2.5-V input buffer. VCCSEL should be set to comply with the logic levels driven out of the configuration device or MAX II device or a microprocessor with flash memory.

VCCSEL also sets the POR trip point for I/O bank 3 to ensure that this I/O bank has powered up to the appropriate voltage levels before configuration begins. For passive serial (PS) mode (MSEL[3..0] = 0010) and for Fast passive parallel (FPP) mode (MSEL[3..0] = 0000)

the POR circuitry selects the trip point associated with 1.5/1.8-V signaling. For all other configuration modes defined by `MSEL[3..0]` settings other than `00X0` (`MSEL[1] = X, don't care`), `VCCSEL=GND` selects the higher I/O Bank 3 POR trip point for 2.5V/3.3V signaling and `VCCSEL=V`$_{CCPD}$ selects the lower I/O Bank 3 POR trip point associated with 1.5V/1.8V signaling.

For all configuration modes with `MSEL[3..0]` not equal to `00X0` (`MSEL[1] = X, don't care`), if $V_{CCIO}$ of configuration bank 3 is powered by 1.8-V or 1.5-V and `VCCSEL = GND`, the voltage supplied to this I/O bank(s) may never reach the POR trip point, which prevents the device from beginning configuration.

☞ The fast passive parallel (FPP) and passive serial (PS) modes always enable bank 3 to use the POR trip point to be consistent with 1.8- and 1.5-V signaling, regardless of the `VCCSEL` setting.

If the $V_{CCIO}$ of I/O bank 3 is powered by 1.5 or 1.8-V and the configuration signals used require 3.3- or 2.5-V signaling, you should set `VCCSEL` to $V_{CCPD}$ to enable the 1.8/1.5-V input buffers for configuration. The 1.8-V/1.5-V input buffers are 3.3-V tolerant.

Table 11–5 shows how you should set the `VCCSEL`, depending on the configuration mode, the voltage level on VCCIO3 pins that power bank 3, and the supported configuration input voltages.

*Table 11–5. Supported V$_{CCSEL}$ Setting based on Mode, VCCIO3, and Input Configuration Voltage*

| Configuration Mode | V$_{CCIO}$ (Bank 3) | Configuration Input Signaling Voltage | V$_{CCSEL}$ |
|---|---|---|---|
| All modes | 3.3-V/2.5-V | 3.3-V/2.5-V | GND |
| All modes | 1.8-V/1.5-V | 3.3-V/2.5-V | V$_{CCPD}$ *(1)* |
| All modes | 1.8-V/1.5-V | 1.8-V/1.5-V | V$_{CCPD}$ |
| - | 3.3-V/2.5-V | 1.8-V/1.5-V | Not Supported |

*Note to Table 11–5:*

(1) The `VCCSEL` pin can also be connected to GND for PS (`MSEL[3..0]=0010`) and FPP (`MSEL[3..0]=0000`) modes.

The key is to ensure the $V_{CCIO}$ voltage of bank 3 is high enough to trip VCCIO3 POR trip point on power-up. Also, to make sure the configuration device meets the $V_{IH}$ for the configuration input pins based on the selected input buffer.

Table 11–6 shows the configuration mode support for banks 4, 7, and 8.

| Table 11–6. Arria GX Configuration Mode Support for Banks 4, 7, & 8 | | | |
|---|---|---|---|
| **Configuration Mode** | **Configuration Voltage/V$_{CCIO}$ Support for Banks 4, 7, & 8** | | |
| | **3.3/3.3** | **1.8/1.8** | **3.3/1.8** |
| | **VCCSEL = GND** | **VCCSEL = VCCPD** | **VCCSEL = GND** |
| Fast passive parallel | Y | Y | Y |
| Passive parallel asynchronous | Y | Y | Y |
| Passive serial | Y | Y | Y |
| Remote system upgrade FPP | Y | Y | Y |
| Remote system upgrade PPA | Y | Y | Y |
| Remote system upgrade PS | Y | Y | Y |
| Fast AS (40 MHz) | Y | Y | Y |
| Remote system upgrade fast AS (40 MHz) | Y | Y | Y |
| FPP with decompression | Y | Y | Y |
| Remote system upgrade FPP with decompression feature enabled | Y | Y | Y |
| AS (20 MHz) | Y | Y | Y |
| Remote system upgrade AS (20 MHz) | Y | Y | Y |

You must verify the configuration output pins for your chosen configuration modes meet the V$_{IH}$ of the configuration device. Refer to Table 11–22 for a consolidated list of configuration output pins.

The V$_{IH}$ of 3.3 or 2.5 V configuration devices will not be met when the V$_{CCIO}$ of the output configuration pins is 1.8 V or 1.5 V. Level shifters will be required to meet the input high level voltage threshold VIH.

Note that AS mode is only applicable for 3.3-V configuration. If I/O bank 3 is less than 3.3V then level shifters are required on the output pins (DCLK, nCSO, ASDO) from the Arria GX device back to the EPCS device.

The VCCSEL signal does not control TDO or nCEO. During configuration, these pins drive out voltage levels corresponding to the V$_{CCIO}$ supply voltage that powers the I/O bank containing the pin.

For more information on multi-volt support, including information on using TDO and nCEO in multi-volt systems, refer to the *Arria GX Architecture* chapter in volume 1 of the *Arria GX Device Handbook*.

# Fast Passive Parallel Configuration

Fast passive parallel (FPP) configuration in Arria GX devices is designed to meet the continuously increasing demand for faster configuration times. Arria GX devices are designed with the capability of receiving byte-wide configuration data per clock cycle. Table 11–7 shows the MSEL pin settings when using the FFP configuration scheme.

*Table 11–7. Arria GX MSEL Pin Settings for FPP Configuration Schemes*

| Configuration Scheme | MSEL3 | MSEL2 | MSEL1 | MSEL0 |
|---|---|---|---|---|
| FPP when not using remote system upgrade or decompression feature | 0 | 0 | 0 | 0 |
| FPP when using remote system upgrade *(1)* | 0 | 1 | 0 | 0 |
| FPP with decompression feature enabled *(2)* | 1 | 0 | 1 | 1 |
| FPP when using remote system upgrade and decompression feature *(1)*, *(2)* | 1 | 1 | 0 | 0 |

*Notes to Table 11–7:*

(1)   These schemes require that you drive the RUnLU pin to specify either remote update or local update. For more information about remote system upgrade in Arria GX devices, refer to the *Remote System Upgrades with Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*.

(2)   These modes are only supported when using a MAX II device or a microprocessor with flash memory for configuration. In these modes, the host system must output a DCLK that is 4× the data rate.

FPP configuration of Arria GX devices can be performed using an intelligent host, such as a MAX II device, a microprocessor, or an Altera enhanced configuration device.

## FPP Configuration Using a MAX II Device as an External Host

FPP configuration using compression and an external host provides the fastest method to configure Arria GX devices. In the FPP configuration scheme, a MAX II device can be used as an intelligent host that controls the transfer of configuration data from a storage device, such as flash memory, to the target Arria GX device. Configuration data can be stored in RBF, HEX, or TTF format. When using the MAX II devices as an intelligent host, a design that controls the configuration process, such as fetching the data from flash memory and sending it to the device, must be stored in the MAX II device.

☞   If you are using the Arria GX decompression feature, the external host must be able to send a DCLK frequency that is 4× the data rate.

The 4× DCLK signal does not require an additional pin and is sent on the DCLK pin. The maximum DCLK frequency is 100 MHz, which results in a maximum data rate of 200 Mbps. If you are not using the Arria GX decompression feature, the data rate is 8× the DCLK frequency.

Figure 11–3 shows the configuration interface connections between the Arria GX device and a MAX II device for single device configuration.

*Figure 11–3. Single Device FPP Configuration Using an External Host*
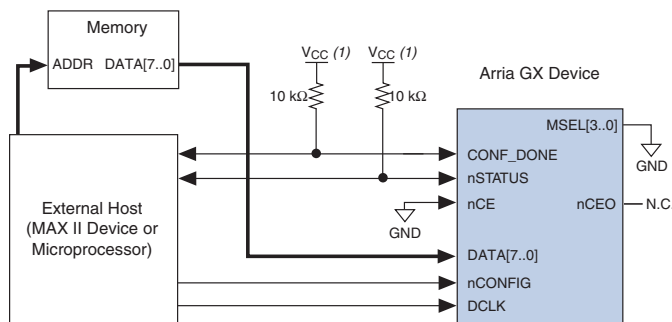


*Note to Figure 11–3:*
(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for the device. $V_{CC}$ should be high enough to meet the $V_{IH}$ specification of the I/O on the device and the external host.

Upon power-up, the Arria GX devices go through a power-on reset (POR). The POR delay is dependent on the PORSEL pin setting: when PORSEL is driven low, the POR time is approximately 100 ms; when PORSEL is driven high, the POR time is approximately 12 ms. During POR, the device resets, holds nSTATUS low, and tri-states all user I/O pins. Once the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins have weak pull-up resistors, which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *DC & Switching Characteristics* chapter in volume 1 of the *Arria GX Device Handbook*.

The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in the reset stage. To initiate configuration, the MAX II device must drive the nCONFIG pin from low to high.

$V_{CCINT}$, $V_{CCIO}$, and $V_{CCPD}$ of the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels in order to begin the configuration process.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 10-kΩ pull-up resistor. Once nSTATUS is released, the device is ready to receive configuration data and the configuration stage begins. When nSTATUS is pulled high, the MAX II device places the configuration data, one byte at a time, on the DATA[7..0] pins.

☞ Arria GX devices receive configuration data on the DATA[7..0] pins and the clock is received on the DCLK pin. Data is latched into the device on the rising edge of DCLK. If you are using the Arria GX decompression feature, configuration data is latched on the rising edge of every fourth DCLK cycle. After the configuration data is latched in, it is processed during the following three DCLK cycles.

Data is continuously clocked into the target device until CONF_DONE goes high. The CONF_DONE pin goes high one byte early in parallel configuration (FPP and PPA) modes. The last byte is required for serial configuration (AS and PS) modes. After the device has received the next to last byte of the configuration data successfully, it releases the open-drain CONF_DONE pin, which is pulled high by an external 10-kΩ pull-up resistor. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize.

In Arria GX devices, the initialization clock source is either the internal oscillator (typically 10 MHz) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the Arria GX device provides itself with enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device. Driving DCLK to the device after configuration is complete does not affect device operation.

You can also synchronize initialization of multiple devices or to delay initialization with the CLKUSR option. The **Enable user-supplied start-up clock** (**CLKUSR**) option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR does not affect the configuration process. The CONF_DONE pin goes high one byte early in parallel configuration (FPP and PPA) modes. The last byte is required for serial configuration (AS and PS) modes. After the CONF_DONE pin transitions high, CLKUSR is enabled after the time specified as $t_{CD2CU}$. After this time period elapses, Arria GX devices require 299 clock cycles to initialize properly and enter user mode. Arria GX devices support a CLKUSR $f_{MAX}$ of 100 MHz.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. This **Enable INIT_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the INIT_DONE pin is used, it is high because of an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high. The MAX II device must be able to detect this low-to-high transition, which signals the device has entered user mode. When initialization is complete, the device enters user mode. In user-mode, the user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

To ensure DCLK and DATA[7..0] are not left floating at the end of configuration, the MAX II device must drive them either high or low, whichever is convenient on your board. The DATA[7..0] pins are available as user I/O pins after configuration. When you select the FPP scheme in the Quartus II software, as a default, these I/O pins are tri-stated in user mode. To change this default option in the Quartus II software, select the **Pins** tab of the **Device & Pin Options** dialog box.

The configuration clock (DCLK) speed must be below the specified frequency to ensure correct configuration. No maximum DCLK period exists, which means you can pause configuration by halting DCLK for an indefinite amount of time.

☞ If you are using the Arria GX decompression feature and need to stop DCLK, it can only be stopped three clock cycles after the last data byte was latched into the Arria GX device.

By stopping DCLK, the configuration circuit allows enough clock cycles to process the last byte of latched configuration data. When the clock restarts, the MAX II device must provide data on the DATA[7..0] pins prior to sending the first DCLK rising edge.

If an error occurs during configuration, the device drives its nSTATUS pin low, resetting itself internally. The low signal on the nSTATUS pin also alerts the MAX II device that there is an error. If the **Auto-restart configuration after error** option (available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box) is turned on, the device releases nSTATUS after a reset time-out period (maximum of 100 µs). After nSTATUS is released and pulled high by a pull-up resistor, the MAX II device can try to reconfigure the target device

without needing to pulse nCONFIG low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2 µs) on nCONFIG to restart the configuration process.

The MAX II device can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. The CONF_DONE pin must be monitored by the MAX II device to detect errors and determine when programming completes. If all configuration data is sent, but the CONF_DONE or INIT_DONE signals have not gone high, the MAX II device will reconfigure the target device.

☞ If the optional CLKUSR pin is used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure CLKUSR continues toggling during the time nSTATUS is low (maximum of 100 µs).

When the device is in user-mode, initiating a reconfiguration is done by transitioning the nCONFIG pin low-to-high. The nCONFIG pin should be low for at least 2 µs. When nCONFIG is pulled low, the device also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Once nCONFIG returns to a logic high level and nSTATUS is released by the device, reconfiguration begins.

Figure 11–4 shows how to configure multiple devices using a MAX II device. This circuit is similar to the FPP configuration circuit for a single device, except the Arria GX devices are cascaded for multi-device configuration.

*Figure 11–4. Multi-Device FPP Configuration Using an External Host*
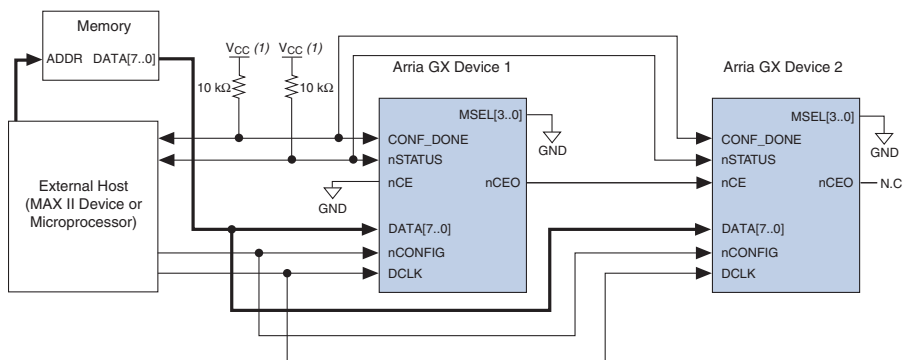


*Note to Figure 11–4:*
(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain. $V_{CC}$ should be high enough to meet the $V_{IH}$ specification of the I/O standard on the device and the external host.

In multi-device FPP configuration, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration within one clock cycle; therefore, the transfer of data destinations is transparent to the MAX II device. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. The configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

All nSTATUS and CONF_DONE pins are tied together and if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single device detecting an error.

If the **Auto-restart configuration after error** option is turned on, the devices release their nSTATUS pins after a reset time-out period (maximum of 100 μs). After all nSTATUS pins are released and pulled high, the MAX II device can try to reconfigure the chain without pulsing nCONFIG low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2 μs) on nCONFIG to restart the configuration process.

In a multi-device FPP configuration chain, all Arria GX devices in the chain must either enable or disable the decompression feature. You can not selectively enable the decompression feature for each device in the chain because of the DATA and DCLK relationship.

If a system has multiple devices that contain the same configuration data, tie all device nCE inputs to GND, and leave nCEO pins floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. Configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices start and complete configuration at the same time. Figure 11–5 shows multi-device FPP configuration when both Arria GX devices are receiving the same configuration data.

*Figure 11–5. Multiple-Device FPP Configuration Using an External Host When Both Devices Receive the Same Data*
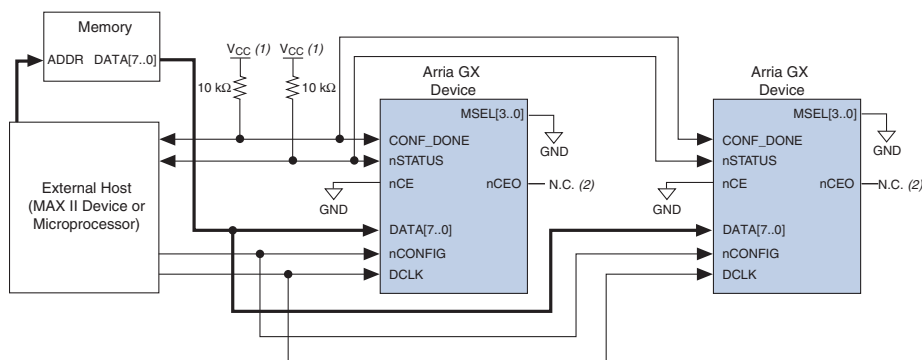


*Notes to Figure 11–5:*

(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain. $V_{CC}$ should be high enough to meet the $V_{IH}$ specification of the I/O on the device and the external host.

(2) The nCEO pins of both Arria GX devices are left unconnected when configuring the same configuration data into multiple devices.

You can use a single configuration chain to configure Arria GX devices with other Altera devices that support FPP configuration, such as Stratix® devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, tie all of the device CONF_DONE and nSTATUS pins together.

For more information about configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains* chapter in volume 2 of the *Configuration Handbook.*

### FPP Configuration Timing

Figure 11–6 shows the timing waveform for FPP configuration when using a MAX II device as an external host. This waveform shows the timing when the decompression feature is not enabled.

*Figure 11–6. FPP Configuration Timing Waveform* *Notes (1)*, *(2)*
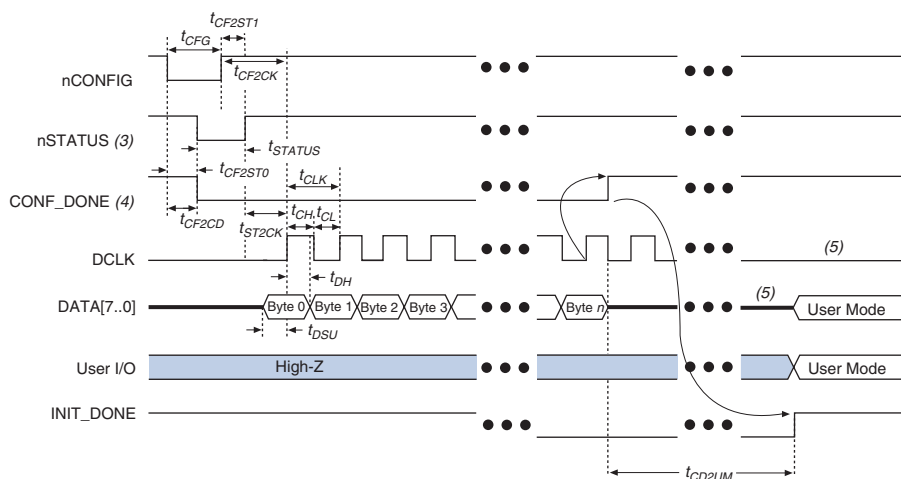


*Notes to Figure 11–6:*
(1) This timing waveform should be used when the decompression feature is not used.
(2) The beginning of this waveform shows the device in user-mode. In user-mode, nCONFIG, nSTATUS, and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.
(3) Upon power-up, the Arria GX device holds nSTATUS low for the time of the POR delay.
(4) Upon power-up, before and during configuration, CONF_DONE is low.
(5) DCLK should not be left floating after configuration. It should be driven high or low, whichever is more convenient.
(6) DATA[7..0] are available as user I/O pins after configuration and the state of these pins depends on the dual-purpose pin settings.

Table 11–8 defines the timing parameters for Arria GX devices for FPP configuration when the decompression feature is not enabled.

*Table 11–8. FPP Timing Parameters for Arria GX Devices  (Part 1 of 2)*   *Notes (1)*, *(2)*

| Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|
| $t_{CF2CD}$ | nCONFIG low to CONF_DONE low | | 800 | ns |
| $t_{CF2ST0}$ | nCONFIG low to nSTATUS low | | 800 | ns |
| $t_{CFG}$ | nCONFIG low pulse width | 2 | | µs |
| $t_{STATUS}$ | nSTATUS low pulse width | 10 | 100 *(3)* | µs |
| $t_{CF2ST1}$ | nCONFIG high to nSTATUS high | | 100 *(3)* | µs |
| $t_{CF2CK}$ | nCONFIG high to first rising edge on DCLK | 100 | | µs |
| $t_{ST2CK}$ | nSTATUS high to first rising edge of DCLK | 2 | | µs |
| $t_{DSU}$ | Data setup time before rising edge on DCLK | 5 | | ns |

| Symbol | Parameter | Min | Max | Units |
|--------|-----------|-----|-----|-------|
| $t_{DH}$ | Data hold time after rising edge on DCLK | 0 | | ns |
| $t_{CH}$ | DCLK high time | 4 | | ns |
| $t_{CL}$ | DCLK low time | 4 | | ns |
| $t_{CLK}$ | DCLK period | 10 | | ns |
| $f_{MAX}$ | DCLK frequency | | 100 | MHz |
| $t_R$ | Input rise time | | 40 | ns |
| $t_F$ | Input fall time | | 40 | ns |
| $t_{CD2UM}$ | CONF_DONE high to user mode *(4)* | 20 | 100 | μs |
| $t_{CD2CU}$ | CONF_DONE high to CLKUSR enabled | $4 \times$ maximum DCLK period | | |
| $t_{CD2UMC}$ | CONF_DONE high to user mode with **CLKUSR** option on | $t_{CD2CU} + (299 \times$ CLKUSR period) | | |

*Table 11–8. FPP Timing Parameters for Arria GX Devices (Part 2 of 2)* *Notes (1), (2)*

*Notes to Table 11–8:*
(1) This information is preliminary.
(2) These timing parameters should be used when the decompression feature is not used.
(3) This value is obtainable if you do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.
(4) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device.

Figure 11–7 shows the timing waveform for FPP configuration when using a MAX II device as an external host. This waveform shows the timing when the decompression feature is enabled.

*Figure 11–7. FPP Configuration Timing Waveform With Decompression Feature Enabled*    *Notes (1)*, *(2)*
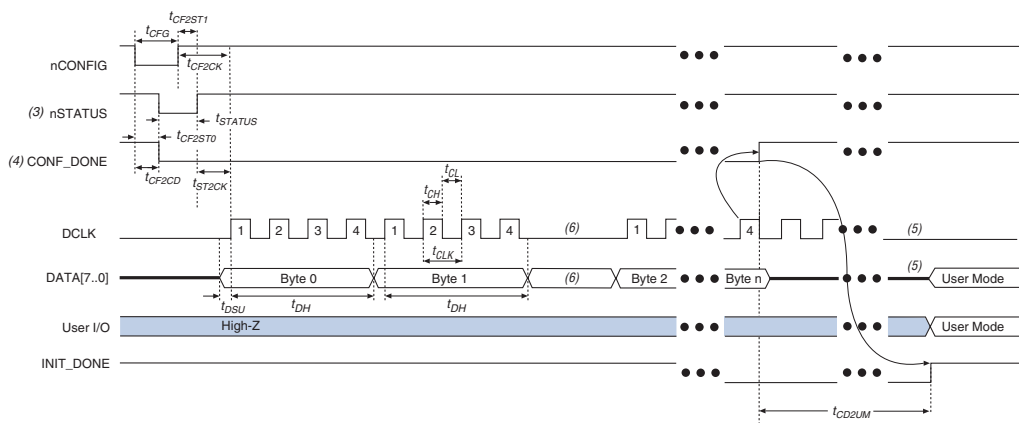


**Notes to Figure 11–7:**
(1) This timing waveform should be used when the decompression feature is used.
(2) The beginning of this waveform shows the device in user-mode. In user-mode, nCONFIG, nSTATUS and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.
(3) Upon power-up, the Arria GX device holds nSTATUS low for the time of the POR delay.
(4) Upon power-up, before and during configuration, CONF_DONE is low.
(5) DCLK should not be left floating after configuration. It should be driven high or low, whichever is more convenient.
(6) DATA[7..0] are available as user I/O pins after configuration and the state of these pins depends on the dual-purpose pin settings. If needed, DCLK can be paused by holding it low. When DCLK restarts, the external host must provide data on the DATA[7..0] pins prior to sending the first DCLK rising edge.

Table 11–9 defines the timing parameters for Arria GX devices for FPP configuration when the decompression feature is enabled.

**Table 11–9. FPP Timing Parameters for Arria GX Devices With Decompression Feature Enabled** *Notes (1)*, *(2)*

| Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|
| $t_{CF2CD}$ | nCONFIG low to CONF_DONE low | | 800 | ns |
| $t_{CF2ST0}$ | nCONFIG low to nSTATUS low | | 800 | ns |
| $t_{CFG}$ | nCONFIG low pulse width | 2 | | μs |
| $t_{STATUS}$ | nSTATUS low pulse width | 10 | 100 *(3)* | μs |
| $t_{CF2ST1}$ | nCONFIG high to nSTATUS high | | 100 *(3)* | μs |
| $t_{CF2CK}$ | nCONFIG high to first rising edge on DCLK | 100 | | μs |
| $t_{ST2CK}$ | nSTATUS high to first rising edge of DCLK | 2 | | μs |
| $t_{DSU}$ | Data setup time before rising edge on DCLK | 5 | | ns |
| $t_{DH}$ | Data hold time after rising edge on DCLK | 30 | | ns |
| $t_{CH}$ | DCLK high time | 4 | | ns |
| $t_{CL}$ | DCLK low time | 4 | | ns |
| $t_{CLK}$ | DCLK period | 10 | | ns |
| $f_{MAX}$ | DCLK frequency | | 100 | MHz |
| $t_{DATA}$ | Data rate | | 200 | Mbps |
| $t_R$ | Input rise time | | 40 | ns |
| $t_F$ | Input fall time | | 40 | ns |
| $t_{CD2UM}$ | CONF_DONE high to user mode *(4)* | 20 | 100 | μs |
| $t_{CD2CU}$ | CONF_DONE high to CLKUSR enabled | 4 × maximum DCLK period | | |
| $t_{CD2UMC}$ | CONF_DONE high to user mode with **CLKUSR** option on | $t_{CD2CU}$ + (299 × CLKUSR period) | | |

*Notes to Table 11–9:*
(1) This information is preliminary.
(2) These timing parameters should be used when the decompression feature is used.
(3) This value is obtainable if users do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.
(4) The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device.

Device configuration options and how to create configuration files are discussed further in the *Software Settings* section in the *Configuration Handbook*.

## FPP Configuration Using a Microprocessor

In the FPP configuration scheme, a microprocessor can control the transfer of configuration data from a storage device, such as flash memory, to the target Arria GX device.

☞ All information in "FPP Configuration Using a MAX II Device as an External Host" on page 11–13 is also applicable when using a microprocessor as an external host. Refer to that section for all configuration and timing information.

## FPP Configuration Using an Enhanced Configuration Device

In the FPP configuration scheme, an enhanced configuration device sends a byte of configuration data every DCLK cycle to the Arria GX device. Configuration data is stored in the configuration device.

☞ When configuring your Arria GX device using FPP mode and an enhanced configuration device, the enhanced configuration device decompression feature is available while the Arria GX decompression feature is not.

Figure 11–8 shows the configuration interface connections between a Arria GX device and the enhanced configuration device for single device configuration.

☞ The figures in this chapter only show the configuration-related pins and the configuration pin connections between the configuration device and the device.

⬥⬥ For more information on the enhanced configuration device and flash interface pins, such as PGM[2..0], EXCLK, PORSEL, A[20..0], and DQ[15..0], refer to the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* in the *Configuration Handbook.*

*Figure 11–8. Single Device FPP Configuration Using an Enhanced Configuration Device*
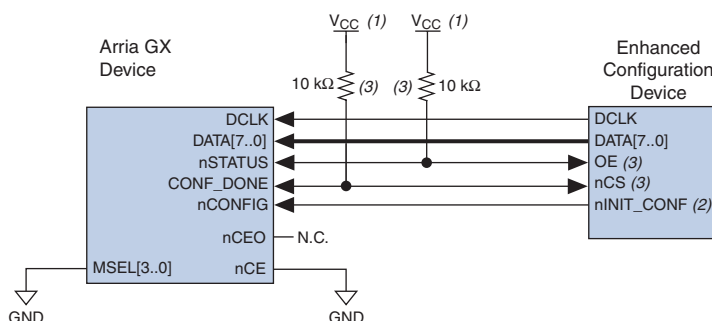


*Notes to Figure 11–8:*
(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.
(2) The nINIT_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active. This means an external pull-up resistor should not be used on the nINIT_CONF-nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to $V_{CC}$ either directly or through a resistor. If reconfiguration is required, a resistor is necessary.
(3) The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

The value of the internal pull-up resistors on the enhanced configuration devices can be found in the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* in the *Configuration Handbook.*

When using enhanced configuration devices, you can connect the device's nCONFIG pin to nINIT_CONF pin of the enhanced configuration device, which allows the INIT_CONF JTAG instruction to initiate device configuration. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to $V_{CC}$ either directly or through a resistor. An internal pull-up resistor on the nINIT_CONF pin is always active in the enhanced configuration devices, which means an external pull-up resistor should not be used if nCONFIG is tied to nINIT_CONF.

Upon power-up, the Arria GX device goes through a POR. The POR delay is dependent on the PORSEL pin setting: when PORSEL is driven low, the POR time is approximately 100 ms; when PORSEL is driven high, the POR time is approximately 12 ms. During POR, the device will reset, hold

nSTATUS low, and tri-state all user I/O pins. The configuration device also goes through a POR delay to allow the power supply to stabilize. The POR time for enhanced configuration devices can be set to either 100 ms or 2 ms, depending on its PORSEL pin setting. If the PORSEL pin is connected to GND, the POR delay is 100 ms. If the PORSEL pin is connected to $V_{CC}$, the POR delay is 2 ms. During this time, the configuration device drives its OE pin low. This low signal delays configuration because the OE pin is connected to the target device's nSTATUS pin.

☞ When selecting a POR time, you need to ensure that the device completes power-up before the enhanced configuration device exits POR. Altera recommends that you use a 12-ms POR time for the Arria GX device, and use a 100-ms POR time for the enhanced configuration device.

When both devices complete POR, they release their open-drain OE or nSTATUS pin, which is then pulled high by a pull-up resistor. Once the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors, which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

👣 The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *Arria GX Device Handbook*.

When the power supplies have reached the appropriate operating voltages, the target device senses the low-to-high transition on nCONFIG and initiates the configuration cycle. The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. The beginning of configuration can be delayed by holding the nCONFIG or nSTATUS pin low.

☞ $V_{CCINT}$, $V_{CCIO}$, and $V_{CCPD}$ of the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels in order to begin the configuration process.

When nCONFIG goes high, the device comes out of reset and releases the nSTATUS pin, which is pulled high by a pull-up resistor. Enhanced configuration devices have an optional internal pull-up resistor on the OE pin. This option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If this internal pull-up resistor is not used, an external 10-kΩ pull-up resistor on the OE-nSTATUS line is required. Once nSTATUS is released, the device is ready to receive configuration data and the configuration stage begins.

When nSTATUS is pulled high, the configuration device's OE pin also goes high and the configuration device clocks data out to the device using the Arria GX device's internal oscillator. The Arria GX devices receive configuration data on the DATA[7..0] pins and the clock is received on the DCLK pin. A byte of data is latched into the device on each rising edge of DCLK.

After the device has received all configuration data successfully, it releases the open-drain CONF_DONE pin which is pulled high by a pull-up resistor. Because CONF_DONE is tied to the configuration device's nCS pin, the configuration device is disabled when CONF_DONE goes high. Enhanced configuration devices have an optional internal pull-up resistor on the nCS pin. This option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If this internal pull-up resistor is not used, an external 10-kΩ pull-up resistor on the nCS-CONF_DONE line is required. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin.

In Arria GX devices, the initialization clock source is either the internal oscillator (typically 10 MHz) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the Arria GX device provides itself with enough clock cycles for proper initialization. You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the CLKUSR option. The **Enable user-supplied start-up clock** (**CLKUSR**) option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR will not affect the configuration process. After all configuration data has been accepted and CONF_DONE goes high, CLKUSR will be enabled after the time specified as $t_{CD2CU}$. After this time period elapses, Arria GX devices require 299 clock cycles to initialize properly and enter user mode. Arria GX devices support a CLKUSR $f_{MAX}$ of 100 MHz.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. The **Enable INIT_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the INIT_DONE pin is used, it will be high due to an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin will go low. When initialization is complete, the INIT_DONE pin will be released and pulled high. In user-mode, the user

I/O pins will no longer have weak pull-up resistors and will function as assigned in your design. The enhanced configuration device will drive DCLK low and DATA[7..0] high at the end of configuration.

If an error occurs during configuration, the device drives its nSTATUS pin low, resetting itself internally. Because the nSTATUS pin is tied to OE, the configuration device will also be reset. If the **Auto-restart configuration after error** option (available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box) is turned on, the device will automatically initiate reconfiguration if an error occurs. The Arria GX device releases its nSTATUS pin after a reset time-out period (maximum of 100 μs). When the nSTATUS pin is released and pulled high by a pull-up resistor, the configuration device reconfigures the chain. If this option is turned off, the external system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 2 μs to restart configuration. The external system can pulse nCONFIG if nCONFIG is under system control rather than tied to V$_{CC}$.

In addition, if the configuration device sends all of its data and then detects that CONF_DONE has not gone high, it recognizes that the device has not configured successfully. Enhanced configuration devices wait for 64 DCLK cycles after the last configuration bit was sent for CONF_DONE to reach a high state. In this case, the configuration device pulls its OE pin low, which in turn drives the target device's nSTATUS pin low. If the Auto-restart configuration after error option is set in the software, the target device resets and then releases its nSTATUS pin after a reset time-out period (maximum of 100 μs). When nSTATUS returns to a logic high level, the configuration device will try to reconfigure the device.

When CONF_DONE is sensed low after configuration, the configuration device recognizes that the target device has not configured successfully. Therefore, your system should not pull CONF_DONE low to delay initialization. Instead, you should use the CLKUSR option to synchronize the initialization of multiple devices that are not in the same configuration chain. Devices in the same configuration chain will initialize together if their CONF_DONE pins are tied together.

☞ If the optional CLKUSR pin is used and nCONFIG is pulled low to restart configuration during device initialization, ensure CLKUSR continues toggling during the time nSTATUS is low (maximum of 100 μs).

When the device is in user-mode, a reconfiguration can be initiated by pulling the nCONFIG pin low. The nCONFIG pin should be low for at least 2 μs. When nCONFIG is pulled low, the device also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Because CONF_DONE is

pulled low, this activates the configuration device because it sees its `nCS` pin drive low. Once `nCONFIG` returns to a logic high level and `nSTATUS` is released by the device, reconfiguration begins.

Figure 11–9 shows how to configure multiple Arria GX devices with an enhanced configuration device. This circuit is similar to the configuration device circuit for a single device, except the Arria GX devices are cascaded for multi-device configuration.

*Figure 11–9. Multi-Device FPP Configuration Using an Enhanced Configuration Device*



*Notes to Figure 11–9:*
(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.
(2) The `nINIT_CONF` pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active. This means an external pull-up resistor should not be used on the `nINIT_CONF`-`nCONFIG` line. The `nINIT_CONF` pin does not need to be connected if its functionality is not used. If `nINIT_CONF` is not used, `nCONFIG` must be pulled to $V_{CC}$ either directly or through a resistor.
(3) The enhanced configuration devices' `OE` and `nCS` pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-up resistors on configuration device** option when generating programming files.

☞ Enhanced configuration devices cannot be cascaded.

When performing multi-device configuration, you must generate the configuration device's POF from each project's SOF. You can combine multiple SOFs using the **Convert Programming Files** window in the Quartus II software.

For more information on how to create configuration files for multi-device configuration chains, refer to the *Software Settings* section in volume 2 of the *Configuration Handbook*.

In multi-device FPP configuration, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. Pay special attention to the configuration signals because they may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device.

When configuring multiple devices, configuration does not begin until all devices release their OE or nSTATUS pins. Similarly, because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Because all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This low signal drives the OE pin low on the enhanced configuration device and drives nSTATUS low on all devices, which causes them to enter a reset state. This behavior is similar to a single device detecting an error.

If the **Auto-restart configuration after error** option is turned on, the devices will automatically initiate reconfiguration if an error occurs. The devices will release their nSTATUS pins after a reset time-out period (maximum of 100 μs). When all the nSTATUS pins are released and pulled high, the configuration device tries to reconfigure the chain. If the **Auto-restart configuration after error** option is turned off, the external system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 2 μs to restart configuration. The external system can pulse nCONFIG if nCONFIG is under system control rather than tied to $V_{CC}$.

Your system may have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA[7..0], and CONF_DONE) are connected to every device in the chain. Configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices will start and complete configuration at the same time. Figure 11–10 shows multi-device FPP configuration when both Arria GX devices are receiving the same configuration data.

*Figure 11–10. Multiple-Device FPP Configuration Using an Enhanced Configuration Device When Both Devices Receive the Same Data*
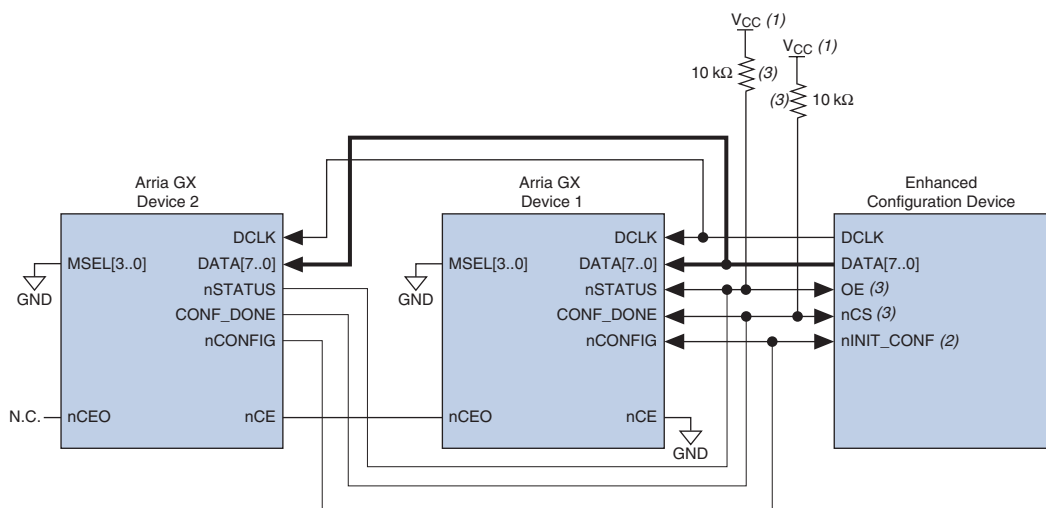


**Notes to Figure 11–10:**

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active. This means an external pull-up resistor should not be used on the nINIT_CONF-nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to $V_{CC}$ either directly or through a resistor.

(3) The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

(4) The nCEO pins of both devices are left unconnected when configuring the same configuration data into multiple devices.

You can use a single enhanced configuration chain to configure multiple Arria GX devices with other Altera devices that support FPP configuration, such as Arria GX devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, all of the device CONF_DONE and nSTATUS pins must be tied together.

For more information about configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains* chapter in volume 2 of the *Configuration Handbook*.

Figure 11–11 shows the timing waveform for the FPP configuration scheme using an enhanced configuration device.

*Figure 11–11. Arria GX FPP Configuration Using an Enhanced Configuration Device Timing Waveform*
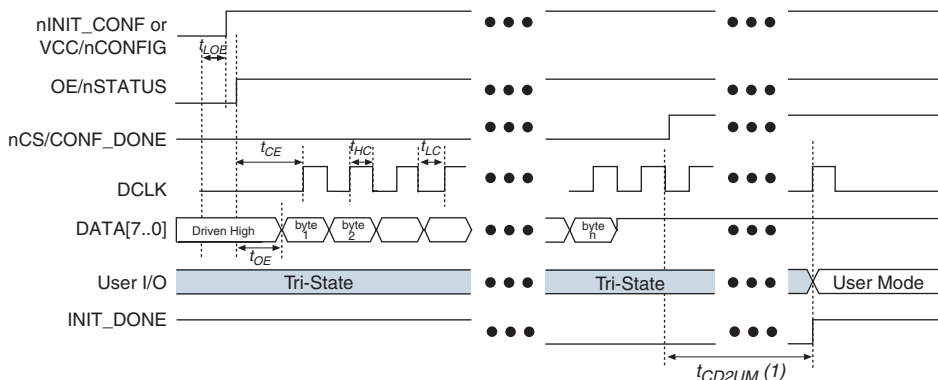


*Note to Figure 11–11:*
(1)    The initialization clock can come from the Arria GX device's internal oscillator or the CLKUSR pin.

For timing information, refer to the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* in volume 2 of the *Configuration Handbook*.

Device configuration options and how to create configuration files are discussed further in the *Software Settings* section of the *Configuration Handbook*.

# Active Serial Configuration (Serial Configuration Devices)

In the AS configuration scheme, Arria GX devices are configured using a serial configuration device. These configuration devices are low-cost devices with non-volatile memory that feature a simple four-pin interface and a small form factor. These features make serial configuration devices an ideal low-cost configuration solution.

For more information on serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* in volume 2 of the *Configuration Handbook*.

Serial configuration devices provide a serial interface to access configuration data. During device configuration, Arria GX devices read configuration data via the serial interface, decompresses data if necessary, and configures their SRAM cells. This scheme is referred to as the AS

configuration scheme because the device controls the configuration interface. This scheme contrasts with the PS configuration scheme, where the configuration device controls the interface.

☞ The Arria GX decompression feature is fully available when configuring your Arria GX device using AS mode.

Table 11–10 shows the MSEL pin settings when using the AS configuration scheme.

| *Table 11–10. Arria GX MSEL Pin Settings for AS Configuration Schemes* | | | | |
|---|---|---|---|---|
| **Configuration Scheme** | **MSEL3** | **MSEL2** | **MSEL1** | **MSEL0** |
| Fast AS (40 MHz) *(1)* | 1 | 0 | 0 | 0 |
| Remote system upgrade fast AS (40 MHz) *(1)* | 1 | 0 | 0 | 1 |
| AS (20 MHz) *(1)* | 1 | 1 | 0 | 1 |
| Remote system upgrade AS (20 MHz) *(1)* | 1 | 1 | 1 | 0 |

*Note to Table 11–10:*

(1) Only the EPCS16 and EPCS64 devices support a DCLK up to 40 MHz clock; other EPCS devices support a DCLK up to 20 MHz. Refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* in volume 2 of the *Configuration Handbook* for more information.

Serial configuration devices have a four-pin interface: serial clock input (DCLK), serial data output (DATA), AS data input (ASDI), and an active-low chip select (nCS). This four-pin interface connects to Arria GX device pins, as shown in Figure 11–12.

*Figure 11–12. Single Device AS Configuration*
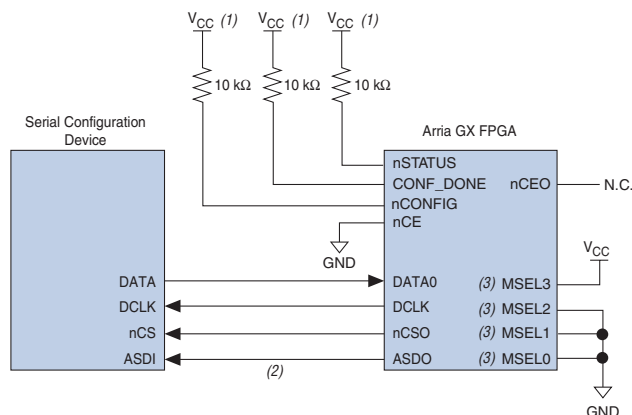


Notes to *Figure 11–12:*
(1) Connect the pull-up resistors to a 3.3-V supply.
(2) Arria GX devices use the ASDO to ASDI path to control the configuration device.
(3) If using an EPCS4 device, MSEL[3..0] should be set to **1101**. Refer to Table 11–10 for more details.

Upon power-up, Arria GX devices go through a POR. The POR delay is dependent on the PORSEL pin setting. When PORSEL is driven low, the POR time is approximately 100 ms. If PORSEL is driven high, the POR time is approximately 12 ms. During POR, the device will reset, hold nSTATUS and CONF_DONE low, and tri-state all user I/O pins. Once the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

☞ The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *DC & Switching Characteristics* chapter in volume 1 of the *Arria GX Device Handbook.*

The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. After POR, the Arria GX devices release nSTATUS, which is pulled high by an external 10-kΩ pull-up resistor, and enters configuration mode.

☞ To begin configuration, power the $V_{CCINT}$, $V_{CCIO}$, and $V_{CCPD}$ voltages (for the banks where the configuration and JTAG pins reside) to the appropriate voltage levels.

The serial clock (DCLK) generated by Arria GX devices controls the entire configuration cycle and provides the timing for the serial interface. Arria GX devices use an internal oscillator to generate DCLK. Using the MSEL [] pins, you can select to use either a 40- or 20-MHz oscillator.

☞ Only the EPCS16 and EPCS64 devices support a DCLK up to 40-MHz clock; other EPCS devices support a DCLK up to 20-MHz.

🐾 Refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* in volume 2 of the *Configuration Handbook* for more information.

The EPCS4 device only supports the smallest Arria GX (EP2S15) device, which is when the SOF compression is enabled. Because of its insufficient memory capacity, the EPCS1 device does not support any Arria GX devices.

Table 11–11 shows the active serial DCLK output frequencies.

*Table 11–11. Active Serial DCLK Output Frequency   Note (1)*

| Oscillator | Minimum | Typical | Maximum | Units |
|------------|---------|---------|---------|-------|
| 40 MHz *(2)* | 20 | 26 | 40 | MHz |
| 20 MHz | 10 | 13 | 20 | MHz |

*Notes to Table 11–11:*
(1)  These values are preliminary.
(2)  Only the EPCS16 and EPCS64 devices support a DCLK up to 40-MHz clock; other EPCS devices support a DCLK up to 20-MHz. Refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* in volume 2 of the *Configuration Handbook* for more information.

In both AS and fast AS configuration schemes, the serial configuration device latches input and control signals on the rising edge of DCLK and drives out configuration data on the falling edge. Arria GX devices drive out control signals on the falling edge of DCLK and latch configuration data on the falling edge of DCLK.

In configuration mode, Arria GX devices enable the serial configuration device by driving the nCSO output pin low, which connects to the chip select (nCS) pin of the configuration device. Arria GX devices use the serial clock (DCLK) and serial data output (ASDO) pins to send operation commands and/or read address signals to the serial configuration device. The configuration device provides data on its serial data output (DATA) pin, which connects to the DATA0 input of the Arria GX devices.

After all configuration bits are received by the Arria GX device, it releases the open-drain CONF_DONE pin, which is pulled high by an external 10-kΩ resistor. Initialization begins only after the CONF_DONE signal reaches a logic high level. All AS configuration pins (DATA0, DCLK, nCSO, and ASDO) have weak internal pull-up resistors that are always active. After configuration, these pins are set as input tri-stated and are driven high by the weak internal pull-up resistors. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize.

In Arria GX devices, the initialization clock source is either the 10-MHz (typical) internal oscillator (separate from the active serial internal oscillator) or the optional **CLKUSR** pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the Arria GX device provides itself with enough clock cycles for proper initialization. You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the CLKUSR option. The **Enable user-supplied start-up clock** (**CLKUSR**) option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. When you **Enable** the **user supplied start-up clock** option, the CLKUSR pin is the initialization clock source. Supplying a clock on CLKUSR will not affect the configuration process. After all configuration data has been accepted and CONF_DONE goes high, CLKUSR is enabled after 600 ns. After this time period elapses, Arria GX devices require 299 clock cycles to initialize properly and enter user mode. Arria GX devices support a CLKUSR $f_{MAX}$ of 100 MHz.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. The **Enable INIT_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the INIT_DONE pin is used, it will be high due to an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high. This low-to-high transition signals that the device has entered user mode. When initialization is complete, the device enters user mode. In user mode, the user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

If an error occurs during configuration, Arria GX devices assert the nSTATUS signal low, indicating a data frame error, and the CONF_DONE signal stays low. If the **Auto-restart configuration after error** option (available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box) is turned on, the Arria GX device resets the configuration device by pulsing nCSO, releases nSTATUS after a reset

time-out period (maximum of 100 μs), and retries configuration. If this option is turned off, the system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 2 μs to restart configuration.

When the Arria GX device is in user mode, you can initiate reconfiguration by pulling the nCONFIG pin low. The nCONFIG pin should be low for at least 2 μs. When nCONFIG is pulled low, the device also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Once nCONFIG returns to a logic high level and nSTATUS is released by the Arria GX device, reconfiguration begins.

You can configure multiple Arria GX devices using a single serial configuration device. You can cascade multiple Arria GX devices using the chip-enable (nCE) and chip-enable-out (nCEO) pins. The first device in the chain must have its nCE pin connected to ground. You must connect its nCEO pin to the nCE pin of the next device in the chain. When the first device captures all of its configuration data from the bitstream, it drives the nCEO pin low, enabling the next device in the chain. You must leave the nCEO pin of the last device unconnected. The nCONFIG, nSTATUS, CONF_DONE, DCLK, and DATA0 pins of each device in the chain are connected (refer to Figure 11–13).

This first Arria GX device in the chain is the configuration master and controls configuration of the entire chain. You must connect its MSEL pins to select the AS configuration scheme. The remaining Arria GX devices are configuration slaves and you must connect their MSEL pins to select the PS configuration scheme. Any other Altera device that supports PS configuration can also be part of the chain as a configuration slave. Figure 11–13 shows the pin connections for this setup.

*Figure 11–13. Multi-Device AS Configuration*
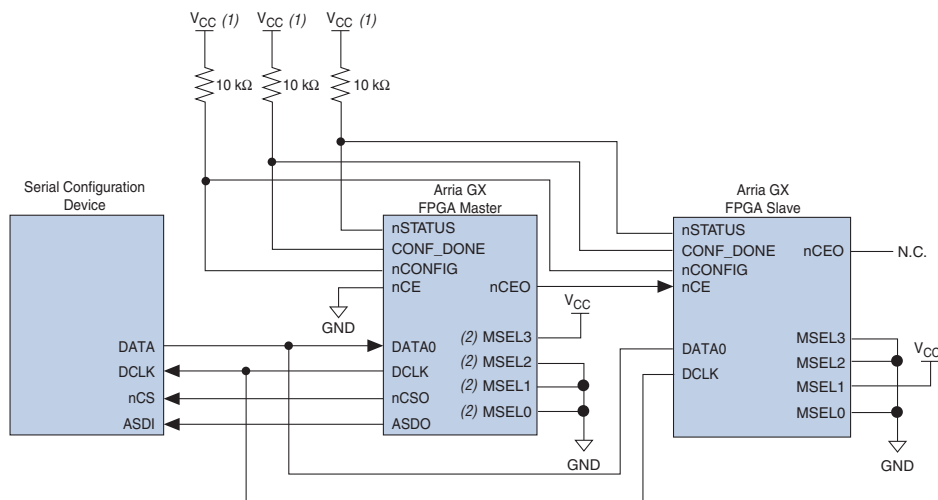


Notes to *Figure 11–13*:
(1)   Connect the pull-up resistors to a 3.3-V supply.
(2)   If using an EPCS4 device, `MSEL[3..0]` should be set to **1101**. Refer to Table 11–10 on page 11–33 for more details.

As shown in Figure 11–13, the `nSTATUS` and `CONF_DONE` pins on all target devices are connected together with external pull-up resistors. These pins are open-drain bidirectional pins on the devices. When the first device asserts `nCEO` (after receiving all of its configuration data), it releases its `CONF_DONE` pin. But the subsequent devices in the chain keep this shared `CONF_DONE` line low until they have received their configuration data. When all target devices in the chain have received their configuration data and have released `CONF_DONE`, the pull-up resistor drives a high level on this line and all devices simultaneously enter initialization mode.

If an error occurs at any point during configuration, the `nSTATUS` line is driven low by the failing device. If you enable the Auto-restart configuration after error option, reconfiguration of the entire chain begins after a reset time-out period (a maximum of 100 µs). If the **Auto-restart configuration after error** option is turned off, the external system must monitor `nSTATUS` for errors and then pulse `nCONFIG` low to restart configuration. The external system can pulse `nCONFIG` if it is under system control rather than tied to $V_{CC}$.

☞   While you can cascade Arria GX devices, serial configuration devices cannot be cascaded or chained together.

If the configuration bitstream size exceeds the capacity of a serial configuration device, you must select a larger configuration device and/or enable the compression feature. When configuring multiple devices, the size of the bitstream is the sum of the individual devices' configuration bitstreams.

A system may have multiple devices that contain the same configuration data. In active serial chains, this can be implemented by storing two copies of the SOF in the serial configuration device. The first copy would configure the master Arria GX device; the second copy would configure all remaining slave devices concurrently. All slave devices must be the same density and package. The setup is similar to Figure 11–13, where the master is set up in active serial mode and the slave devices are set up in passive serial mode.

To configure four identical Arria GX devices with the same SOF, you could set up the chain similar to the example shown in Figure 11–14. The first device is the master device and its MSEL pins should be set to select AS configuration. The other three slave devices are set up for concurrent configuration and its MSEL pins should be set to select PS configuration. The nCEO pin from the master device drives the nCE input pins on all three slave devices, and the DATA and DCLK pins connect in parallel to all four devices. During the first configuration cycle, the master device reads its configuration data from the serial configuration device while holding nCEO high. After completing its configuration cycle, the master drives nCE low and transmits the second copy of the configuration data to all three slave devices, configuring them simultaneously.

*Figure 11–14. Multi-Device AS Configuration When devices Receive the Same Data*
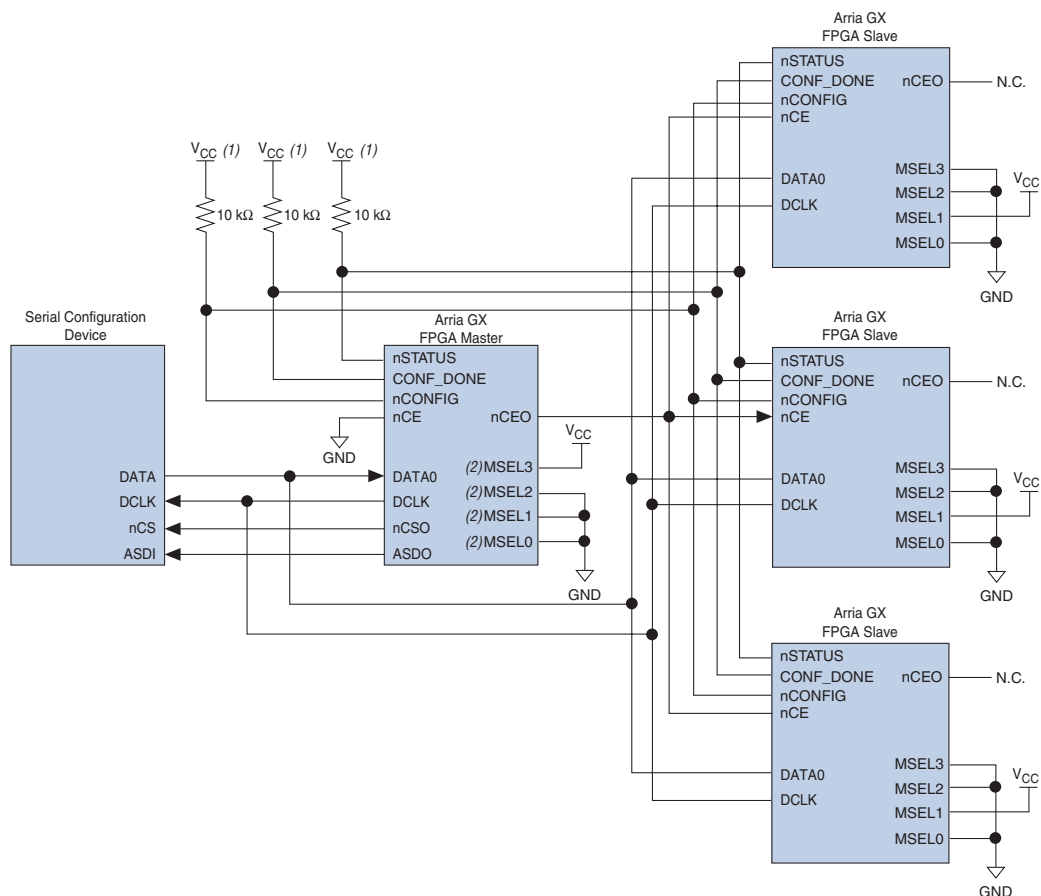


*Notes to Figure 11–14:*

(1)  Connect the pull-up resistors to a 3.3-V supply.

(2)  If using an EPCS4 device, MSEL[3..0] should be set to **1101**. Refer to Table 11–10 on page 11–33 for more details.

### Estimating Active Serial Configuration Time

Active serial configuration time is dominated by the time it takes to transfer data from the serial configuration device to the Arria GX device. This serial interface is clocked by the Arria GX DCLK output (generated from an internal oscillator). As listed in Table 11–11 on page 11–35, the DCLK minimum frequency when choosing to use the 40-MHz oscillator is 20 MHz (50 ns). Therefore, the maximum configuration time estimate for an EP2S15 device (5 MBits of uncompressed data) is:

RBF Size (minimum DCLK period / 1 bit per DCLK cycle) = estimated maximum configuration time

5 Mbits × (50 ns / 1 bit) = 250 ms

To estimate the typical configuration time, use the typical DCLK period as listed in Table 11–11. With a typical DCLK period of 38.46 ns, the typical configuration time is 192 ms. Enabling compression reduces the amount of configuration data that is transmitted to the Arria GX device, which also reduces configuration time. On average, compression reduces configuration time by 50%.

### Programming Serial Configuration Devices

Serial configuration devices are non-volatile, flash-memory-based devices. You can program these devices in-system using the USB-Blaster™ or ByteBlaster™ II download cable. Alternatively, you can program them using the Altera Programming Unit (APU), supported third-party programmers, or a microprocessor with the SRunner software driver.

You can perform in-system programming of serial configuration devices via the AS programming interface. During in-system programming, the download cable disables device access to the AS interface by driving the nCE pin high. Arria GX devices are also held in reset by a low level on nCONFIG. After programming is complete, the download cable releases nCE and nCONFIG, allowing the pull-down and pull-up resistors to drive GND and $V_{CC}$, respectively. Figure 11–15 shows the download cable connections to the serial configuration device.

For more information about the USB Blaster download cable, refer to the *USB-Blaster USB Port Download Cable User Guide*. For more information about the ByteBlaster II cable, refer to the *ByteBlaster II Download Cable User Guide*.

*Figure 11–15. In-System Programming of Serial Configuration Devices*
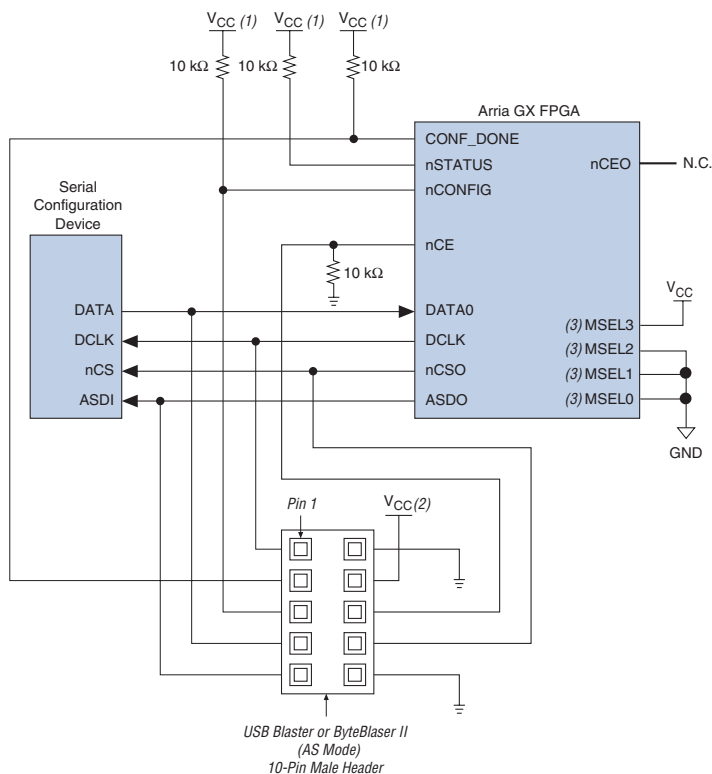


*Notes to Figure 11–15:*
(1) Connect these pull-up resistors to 3.3-V supply.
(2) Power up the ByteBlaster II cable's $V_{CC}$ with a 3.3-V supply.
(3) If using an EPCS4 device, `MSEL[3..0]` should be set to **1101**. Refer to Table 11–10 on page 11–33 for more details.

You can program serial configuration devices with the Quartus II software with the Altera programming hardware and the appropriate configuration device programming adapter. The EPCS1 and EPCS4 devices are offered in an eight-pin small outline integrated circuit (SOIC) package.

In production environments, serial configuration devices can be programmed using multiple methods. Altera programming hardware or other third-party programming hardware can be used to program blank serial configuration devices before they are mounted onto printed circuit

boards (PCBs). Alternatively, you can use an on-board microprocessor to program the serial configuration device in-system using C-based software drivers provided by Altera.

A serial configuration device can be programmed in-system by an external microprocessor using SRunner. SRunner is a software driver developed for embedded serial configuration device programming, which can be easily customized to fit in different embedded systems. SRunner is able to read a raw programming data (**.rpd**) file and write to the serial configuration devices. The serial configuration device programming time using SRunner is comparable to the programming time with the Quartus II software.

For more information about SRunner, refer to *AN 418: SRunner: An Embedded Solution for Serial Configuration* and the source code on the Altera web site at www.altera.com.

For more information on programming serial configuration devices, refer to the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* in volume 2 of the *Configuration Handbook*.

Figure 11–16 shows the timing waveform for the AS configuration scheme using a serial configuration device.

*Figure 11–16. AS Configuration Timing*
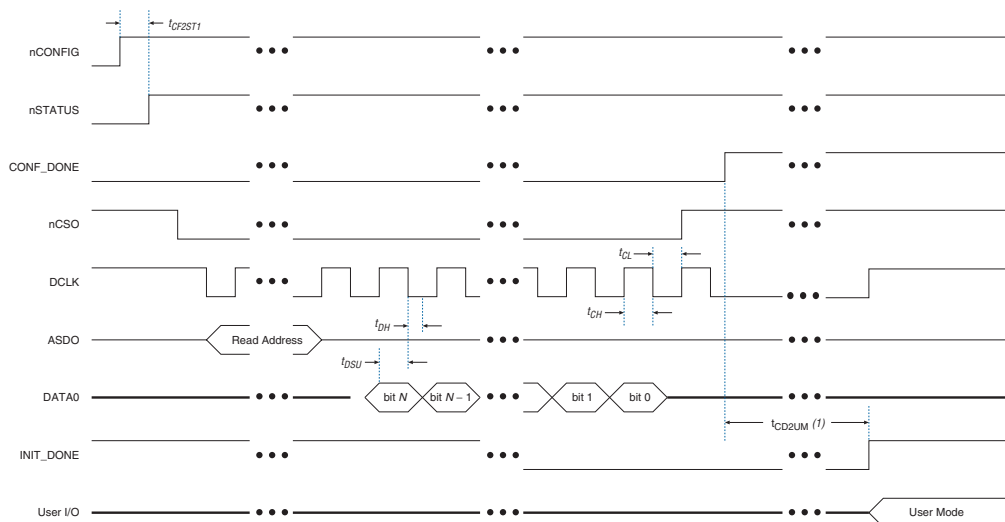


Note to *Figure 11–16*:
(1)    The initialization clock can come from the Arria GX device's internal oscillator or the CLKUSR pin.

Table 11–12 shows the AS timing parameters for Arria GX devices.

**Table 11–12. AS Timing Parameters for Arria GX Devices**

| Symbol | Parameter | Condition | Minimum | Typical | Maximum |
|--------|-----------|-----------|---------|---------|---------|
| $t_{CF2ST1}$ | nCONFIG high to nSTATUS high | | | | 100 |
| $t_{DSU}$ | Data setup time before falling edge on DCLK | | 7 | | |
| $t_{DH}$ | Data hold time after falling edge on DCLK | | 0 | | |
| $t_{CH}$ | DCLK high time | | 10 | | |
| $t_{CL}$ | DCLK low time | | 10 | | |
| $t_{CD2UM}$ | CONF_DONE high to user mode | | 20 | | 100 |

# Passive Serial Configuration

PS configuration of Arria GX devices can be performed using an intelligent host, such as a MAX II device or microprocessor with flash memory, an Altera configuration device, or a download cable. In the PS scheme, an external host (MAX II device, embedded processor, configuration device, or host PC) controls configuration. Configuration data is clocked into the target Arria GX device via the DATA0 pin at each rising edge of DCLK.

☞ The Arria GX decompression feature is fully available when configuring your Arria GX device using PS mode.

Table 11–13 shows the MSEL pin settings when using the PS configuration scheme.

**Table 11–13. Arria GX MSEL Pin Settings for PS Configuration Schemes**

| Configuration Scheme | MSEL3 | MSEL2 | MSEL1 | MSEL0 |
|----------------------|-------|-------|-------|-------|
| PS | 0 | 0 | 1 | 0 |
| PS when using Remote System Upgrade *(1)* | 0 | 1 | 1 | 0 |

*Note to Table 11–13:*
(1) This scheme requires that you drive the RUnLU pin to specify either remote update or local update. For more information about remote system upgrade in Arria GX devices, refer to the *Remote System Upgrades with Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*.

## PS Configuration Using a MAX II Device as an External Host

In the PS configuration scheme, a MAX II device can be used as an intelligent host that controls the transfer of configuration data from a storage device, such as flash memory, to the target Arria GX device. Configuration data can be stored in RBF, HEX, or TTF format. Figure 11–17 shows the configuration interface connections between a Arria GX device and a MAX II device for single device configuration.

*Figure 11–17. Single Device PS Configuration Using an External Host*
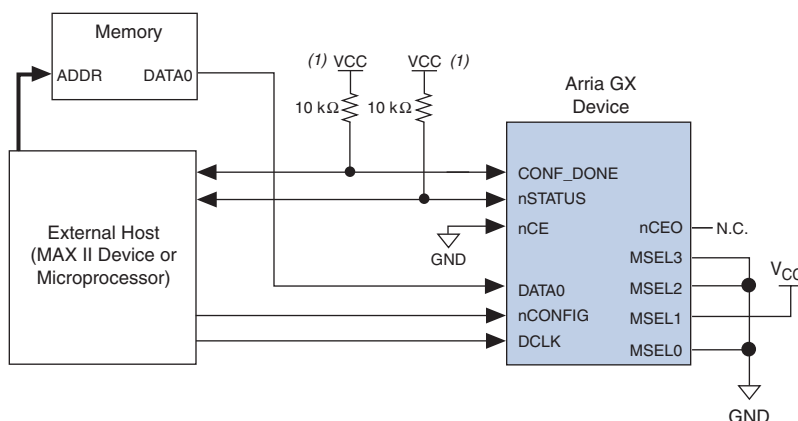


*Note to Figure 11–17:*
(1)   Connect the pull-up resistor to a supply that provides an acceptable input signal for the device. $V_{CC}$ should be high enough to meet the $V_{IH}$ specification of the I/O on the device and the external host.

Upon power-up, Arria GX devices go through a POR. The POR delay is dependent on the PORSEL pin setting: when PORSEL is driven low, the POR time is approximately 100 ms; when PORSEL is driven high, the POR time is approximately 12 ms. During POR, the device resets, holds nSTATUS low, and tri-states all user I/O pins. Once the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *Arria GX Device Handbook*.

The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. To initiate configuration, the MAX II device must generate a low-to-high transition on the nCONFIG pin.

☞ $V_{CCINT}$, $V_{CCIO}$, and $V_{CCPD}$ of the banks where the configuration and JTAG pins reside need to be fully powered to the appropriate voltage levels in order to begin the configuration process.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 10-kΩ pull-up resistor. Once nSTATUS is released, the device is ready to receive configuration data and the configuration stage begins. When nSTATUS is pulled high, the MAX II device should place the configuration data, one bit at a time, on the DATA0 pin. If you are using configuration data in RBF, HEX, or TTF format, you must send the least significant bit (LSB) of each data byte first. For example, if the RBF contains the byte sequence 02 1B EE 01 FA, the serial bitstream you should transmit to the device is 0100-0000 1101-1000 0111-0111 1000-0000 0101-1111.

Arria GX devices receive configuration data on the DATA0 pin and the clock is received on the DCLK pin. Data is latched into the device on the rising edge of DCLK. Data is continuously clocked into the target device until CONF_DONE goes high. After the device has received all configuration data successfully, it releases the open-drain CONF_DONE pin, which is pulled high by an external 10-kΩ pull-up resistor. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize.

In Arria GX devices, the initialization clock source is either the internal oscillator (typically 10 MHz) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the Arria GX device provides itself with enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device. Driving DCLK to the device after configuration is complete does not affect device operation.

You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the CLKUSR option. The **Enable user-supplied start-up clock** (**CLKUSR**) option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR will not affect the configuration process. After all configuration data has been accepted and CONF_DONE

goes high, CLKUSR will be enabled after the time specified as $t_{CD2CU}$. After this time period elapses, Arria GX devices require 299 clock cycles to initialize properly and enter user mode. Arria GX devices support a CLKUSR $f_{MAX}$ of 100 MHz.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. The **Enable INIT_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the INIT_DONE pin is used, it will be high due to an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin will go low. When initialization is complete, the INIT_DONE pin will be released and pulled high. The MAX II device must be able to detect this low-to-high transition which signals the device has entered user mode. When initialization is complete, the device enters user mode. In user-mode, the user I/O pins will no longer have weak pull-up resistors and will function as assigned in your design.

To ensure DCLK and DATA0 are not left floating at the end of configuration, the MAX II device must drive them either high or low, whichever is convenient on your board. The DATA[0] pin is available as a user I/O pin after configuration. When the PS scheme is chosen in the Quartus II software, as a default, this I/O pin is tri-stated in user mode and should be driven by the MAX II device. To change this default option in the Quartus II software, select the **Dual-Purpose Pins** tab of the **Device & Pin Options** dialog box.

The configuration clock (DCLK) speed must be below the specified frequency to ensure correct configuration. No maximum DCLK period exists, which means you can pause configuration by halting DCLK for an indefinite amount of time.

If an error occurs during configuration, the device drives its nSTATUS pin low, resetting itself internally. The low signal on the nSTATUS pin also alerts the MAX II device that there is an error. If the **Auto-restart configuration after error** option (available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box) is turned on, the Arria GX device releases nSTATUS after a reset time-out period (maximum of 100 μs). After nSTATUS is released and pulled high by a pull-up resistor, the MAX II device can try to reconfigure the target device without needing to pulse nCONFIG low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2 μs) on nCONFIG to restart the configuration process.

The MAX II device can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. The CONF_DONE pin must be monitored by the MAX II device to detect errors and determine when programming completes. If all configuration data is sent, but CONF_DONE or INIT_DONE have not gone high, the MAX II device must reconfigure the target device.

☞ If the optional CLKUSR pin is being used and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure that CLKUSR continues toggling during the time nSTATUS is low (maximum of 100 μs).

When the device is in user-mode, you can initiate a reconfiguration by transitioning the nCONFIG pin low to high. The nCONFIG pin must be low for at least 2 μs. When nCONFIG is pulled low, the device also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Once nCONFIG returns to a logic high level and nSTATUS is released by the device, reconfiguration begins.

Figure 11–18 shows how to configure multiple devices using a MAX II device. This circuit is similar to the PS configuration circuit for a single device, except Arria GX devices are cascaded for multi-device configuration.

*Figure 11–18. Multi-Device PS Configuration Using an External Host*
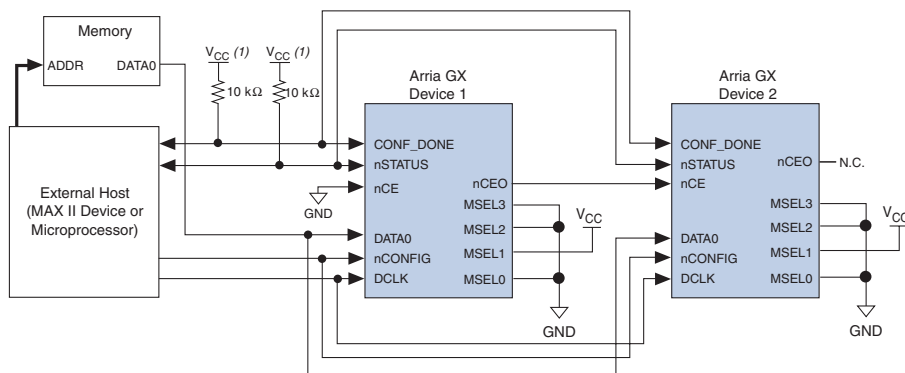


*Note to Figure 11–18:*
(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain. $V_{CC}$ should be high enough to meet the $V_{IH}$ specification of the I/O on the device and the external host.

In multi-device PS configuration, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration within one clock cycle. Therefore, the transfer of data destinations is transparent to the MAX II device. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. Configuration signals can require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Because all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single device detecting an error.

If the **Auto-restart configuration after error** option is turned on, the devices release their nSTATUS pins after a reset time-out period (maximum of 100 μs). After all nSTATUS pins are released and pulled high, the MAX II device can try to reconfigure the chain without needing to pulse nCONFIG low. If this option is turned off, the MAX II device must generate a low-to-high transition (with a low pulse of at least 2 μs) on nCONFIG to restart the configuration process.

In your system, you can have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. Configuration signals can require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices will start and complete configuration at the same time. Figure 11–19 shows multi-device PS configuration when both Arria GX devices are receiving the same configuration data.

*Figure 11–19. Multiple-Device PS Configuration When Both Devices Receive the Same Data*
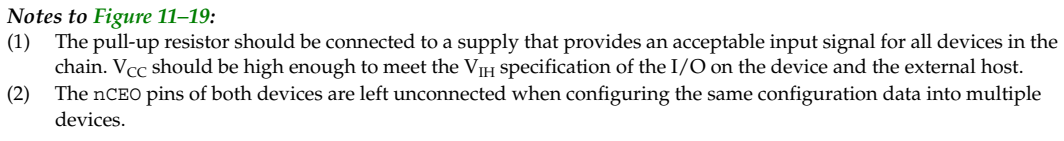


*Notes to Figure 11–19:*
(1)   The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain. $V_{CC}$ should be high enough to meet the $V_{IH}$ specification of the I/O on the device and the external host.
(2)   The nCEO pins of both devices are left unconnected when configuring the same configuration data into multiple devices.

You can use a single configuration chain to configure Arria GX devices with other Altera devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, all of the device CONF_DONE and nSTATUS pins must be tied together.

For more information about configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains* chapter in volume 2 of the *Configuration Handbook*.

*PS Configuration Timing*

Figure 11–20 shows the timing waveform for PS configuration when using a MAX II device as an external host.

*Figure 11–20. PS Configuration Timing Waveform*     Note (1)
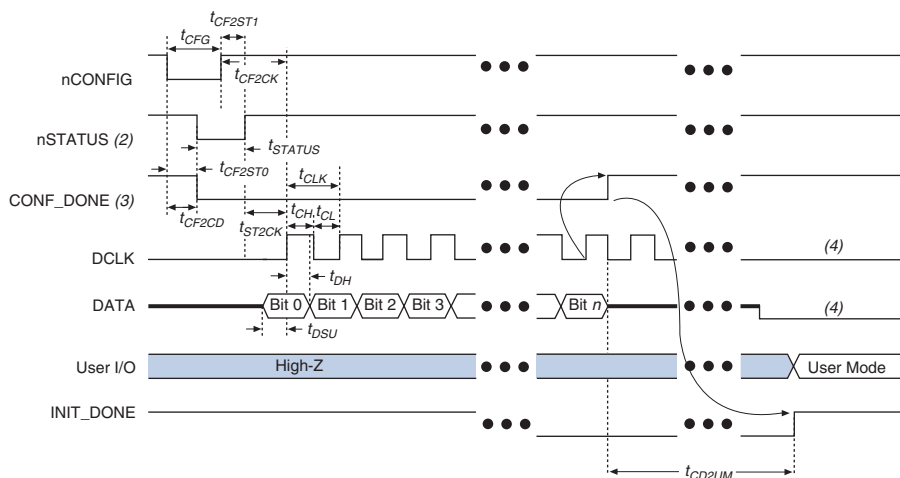


*Notes to Figure 11–20:*
(1) The beginning of this waveform shows the device in user-mode. In user-mode, nCONFIG, nSTATUS, and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.
(2) Upon power-up, the Arria GX device holds nSTATUS low for the time of the POR delay.
(3) Upon power-up, before and during configuration, CONF_DONE is low.
(4) DCLK should not be left floating after configuration. It should be driven high or low, whichever is more convenient. DATA[0] is available as a user I/O pin after configuration and the state of this pin depends on the dual-purpose pin settings.

Table 11–14 defines the timing parameters for Arria GX devices for PS configuration.

*Table 11–14. PS Timing Parameters for Arria GX Devices  (Part 1 of 2)*     Note (1)

| Symbol | Parameter | Min | Max | Units |
|--------|-----------|-----|-----|-------|
| $t_{CF2CD}$ | nCONFIG low to CONF_DONE low | | 800 | ns |
| $t_{CF2ST0}$ | nCONFIG low to nSTATUS low | | 800 | ns |
| $t_{CFG}$ | nCONFIG low pulse width | 2 | | µs |
| $t_{STATUS}$ | nSTATUS low pulse width | 10 | 100 (2) | µs |
| $t_{CF2ST1}$ | nCONFIG high to nSTATUS high | | 100 (2) | µs |
| $t_{CF2CK}$ | nCONFIG high to first rising edge on DCLK | 100 | | µs |
| $t_{ST2CK}$ | nSTATUS high to first rising edge of DCLK | 2 | | µs |
| $t_{DSU}$ | Data setup time before rising edge on DCLK | 5 | | ns |
| $t_{DH}$ | Data hold time after rising edge on DCLK | 0 | | ns |

*Table 11–14. PS Timing Parameters for Arria GX Devices  (Part 2 of 2)    Note (1)*

| Symbol | Parameter | Min | Max | Units |
|--------|-----------|-----|-----|-------|
| $t_{CH}$ | DCLK high time | 4 | | ns |
| $t_{CL}$ | DCLK low time | 4 | | ns |
| $t_{CLK}$ | DCLK period | 10 | | ns |
| $f_{MAX}$ | DCLK frequency | | 100 | MHz |
| $t_R$ | Input rise time | | 40 | ns |
| $t_F$ | Input fall time | | 40 | ns |
| $t_{CD2UM}$ | CONF_DONE high to user mode (3) | 20 | 100 | µs |
| $t_{CD2CU}$ | CONF_DONE high to CLKUSR enabled | $4 \times$ maximum DCLK period | | |
| $t_{CD2UMC}$ | CONF_DONE high to user mode with CLKUSR option on | $t_{CD2CU}$ + (299 × CLKUSR period) | | |

*Notes to Table 11–14:*

(1)   This information is preliminary.

(2)   This value is applicable if users do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.

(3)   The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting the device.

Device configuration options and how to create configuration files are discussed further in the *Software Settings* section in volume 2 of the *Configuration Handbook*.

An example PS design that uses a MAX II device as the external host for configuration will be available when devices are available.

## PS Configuration Using a Microprocessor

In the PS configuration scheme, a microprocessor can control the transfer of configuration data from a storage device, such as flash memory, to the target Arria GX device.

☞   All information in the "PS Configuration Using a MAX II Device as an External Host" on page 11–45 section is also applicable when using a microprocessor as an external host. Refer to that section for all configuration and timing information.

## PS Configuration Using a Configuration Device

You can use an Altera configuration device, such as an enhanced configuration device or EPC2 device, to configure Arria GX devices using a serial configuration bitstream. Configuration data is stored in the configuration device. Figure 11–21 shows the configuration interface connections between an Arria GX device and a configuration device.

☞ The figures in this chapter only show the configuration-related pins and the configuration pin connections between the configuration device and the device.

👣 For more information on the enhanced configuration device and flash interface pins (such as PGM[2..0], EXCLK, PORSEL, A[20..0], and DQ[15..0]), refer to the *Enhanced Configuration Devices (EPC4, EPC8, & EPC16) Data Sheet* chapter in volume 2 of the *Configuration Handbook*.

*Figure 11–21. Single Device PS Configuration Using an Enhanced Configuration Device*
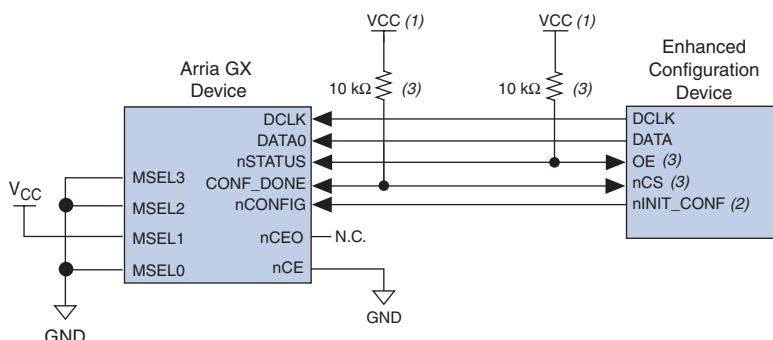


Notes to Figure 11–21:
(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.
(2) The nINIT_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active, meaning an external pull-up resistor should not be used on the nINIT_CONF-nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to $V_{CC}$ either directly or through a resistor.
(3) The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

👣 The value of the internal pull-up resistors on the enhanced configuration devices and EPC2 devices can be found in the Operating Conditions table of the *Enhanced Configuration Devices (EPC4, EPC8, & EPC16) Data Sheet* chapter or the *Configuration Devices for SRAM-based LUT Devices Data Sheet* chapter in volume 2 of the *Configuration Handbook.*

When using enhanced configuration devices or EPC2 devices, `nCONFIG` of the device can be connected to `nINIT_CONF` of the configuration device, which allows the `INIT_CONF` JTAG instruction to initiate device configuration. The `nINIT_CONF` pin does not need to be connected if its functionality is not used. An internal pull-up resistor on the `nINIT_CONF` pin is always active in enhanced configuration devices and EPC2 devices, which means an external pull-up resistor should not be used if `nCONFIG` is tied to `nINIT_CONF`.

Upon power-up, the Arria GX devices go through a POR. The POR delay is dependent on the `PORSEL` pin setting. When `PORSEL` is driven low, the POR time is approximately 100 ms. If `PORSEL` is driven high, the POR time is approximately 12 ms. During POR, the device will reset, hold `nSTATUS` low, and tri-state all user I/O pins. The configuration device also goes through a POR delay to allow the power supply to stabilize. The POR time for EPC2 devices is 200 ms (maximum). The POR time for enhanced configuration devices can be set to either 100 ms or 2 ms, depending on its `PORSEL` pin setting. If the `PORSEL` pin is connected to GND, the POR delay is 100 ms. If the `PORSEL` pin is connected to $V_{CC}$, the POR delay is 2 ms. During this time, the configuration device drives its `OE` pin low. This low signal delays configuration because the `OE` pin is connected to the target device's `nSTATUS` pin.

☞   When selecting a POR time, you need to ensure that the device completes power-up before the enhanced configuration device exits POR. Altera recommends that you choose a POR time for the Arria GX device of 12 ms, while selecting a POR time for the enhanced configuration device of 100 ms.

When both devices complete POR, they release their open-drain `OE` or `nSTATUS` pin, which is then pulled high by a pull-up resistor. Once the device successfully exits POR, all user I/O pins continue to be tri-stated. If `nIO_pullup` is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors which are on (after POR) before and during configuration. If `nIO_pullup` is driven high, the weak pull-up resistors are disabled.

✏   The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *DC & Switching Characteristics* chapter in volume 1 of the *Arria GX Device Handbook*.

When the power supplies have reached the appropriate operating voltages, the target device senses the low-to-high transition on `nCONFIG` and initiates the configuration cycle. The configuration cycle consists of three stages: reset, configuration, and initialization. While `nCONFIG` or `nSTATUS` are low, the device is in reset. The beginning of configuration can be delayed by holding the `nCONFIG` or `nSTATUS` pin low.

☞ To begin configuration, power the $V_{CCINT}$, $V_{CCIO}$, and $V_{CCPD}$ voltages (for the banks where the configuration and JTAG pins reside) to the appropriate voltage levels.

When nCONFIG goes high, the device comes out of reset and releases the nSTATUS pin, which is pulled high by a pull-up resistor. Enhanced configuration and EPC2 devices have an optional internal pull-up resistor on the OE pin. This option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If this internal pull-up resistor is not used, an external 10-kΩ pull-up resistor on the OE-nSTATUS line is required. Once nSTATUS is released, the device is ready to receive configuration data and the configuration stage begins.

When nSTATUS is pulled high, OE of the configuration device also goes high and the configuration device clocks data out serially to the device using the Arria GX device's internal oscillator. Arria GX devices receive configuration data on the DATA0 pin and the clock is received on the DCLK pin. Data is latched into the device on the rising edge of DCLK.

After the device has received all the configuration data successfully, it releases the open-drain CONF_DONE pin, which is pulled high by a pull-up resistor. Because CONF_DONE is tied to the configuration device's nCS pin, the configuration device is disabled when CONF_DONE goes high. Enhanced configuration and EPC2 devices have an optional internal pull-up resistor on the nCS pin. This option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If this internal pull-up resistor is not used, an external 10-kΩ pull-up resistor on the nCS-CONF_DONE line is required. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin.

In Arria GX devices, the initialization clock source is either the internal oscillator (typically 10 MHz) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If you are using internal oscillator, the Arria GX device supplies itself with enough clock cycles for proper initialization. You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the CLKUSR option. You can turn on the **Enable user-supplied start-up clock** (**CLKUSR**) option in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR will not affect the configuration process. After all configuration data has been accepted and CONF_DONE goes high, CLKUSR will be enabled after the time specified as $t_{CD2CU}$. After this time period elapses, the Arria GX devices require 299 clock cycles to initialize properly and enter user mode. Arria GX devices support a CLKUSR $f_{MAX}$ of 100 MHz.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. The **Enable INIT_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If you are using the INIT_DONE pin, it will be high due to an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high. This low-to-high transition signals that the device has entered user mode. In user-mode, the user I/O pins will no longer have weak pull-up resistors and will function as assigned in your design. Enhanced configuration devices and EPC2 devices drive DCLK low and DATA0 high at the end of configuration.

If an error occurs during configuration, the device drives its nSTATUS pin low, resetting itself internally. Because the nSTATUS pin is tied to OE, the configuration device will also be reset. If the **Auto-restart configuration after error** option, available in the Quartus II software, from the **General** tab of the **Device & Pin Options** dialog box is turned on, the device automatically initiates reconfiguration if an error occurs. The Arria GX devices release the nSTATUS pin after a reset time-out period (maximum of 100 µs). When the nSTATUS pin is released and pulled high by a pull-up resistor, the configuration device reconfigures the chain. If this option is turned off, the external system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 2 µs to restart configuration. The external system can pulse nCONFIG if nCONFIG is under system control rather than tied to $V_{CC}$.

In addition, if the configuration device sends all of its data and then detects that CONF_DONE has not gone high, it recognizes that the device has not configured successfully. Enhanced configuration devices wait for 64 DCLK cycles after the last configuration bit was sent for CONF_DONE to reach a high state. EPC2 devices wait for 16 DCLK cycles. In this case, the configuration device pulls its OE pin low, driving the target device's nSTATUS pin low. If the **Auto-restart configuration after error** option is set in the software, the target device resets and then releases its nSTATUS pin after a reset time-out period (maximum of 100 µs). When nSTATUS returns to a logic high level, the configuration device tries to reconfigure the device.

When CONF_DONE is sensed low after configuration, the configuration device recognizes that the target device has not configured successfully. Therefore, your system should not pull CONF_DONE low to delay initialization. Instead, use the CLKUSR option to synchronize the

initialization of multiple devices that are not in the same configuration chain. Devices in the same configuration chain will initialize together if their CONF_DONE pins are tied together.

☞      If you are using the optional CLKUSR pin and nCONFIG is pulled low to restart configuration during device initialization, you need to ensure that CLKUSR continues toggling during the time nSTATUS is low (maximum of 100 μs).

When the device is in user-mode, pulling the nCONFIG pin low initiates a reconfiguration. The nCONFIG pin should be low for at least 2 μs. When nCONFIG is pulled low, the device also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Because CONF_DONE is pulled low, this activates the configuration device because it sees its nCS pin drive low. Once nCONFIG returns to a logic high level and nSTATUS is released by the device, reconfiguration begins.

Figure 11–22 shows how to configure multiple devices with an enhanced configuration device. This circuit is similar to the configuration device circuit for a single device, except Arria GX devices are cascaded for multi-device configuration.

*Figure 11–22. Multi-Device PS Configuration Using an Enhanced Configuration Device*



*Notes to Figure 11–22:*
(1)     The pull-up resistor should be connected to the same supply voltage as the configuration device.
(2)     The nINIT_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active, meaning an external pull-up resistor should not be used on the nINIT_CONF-nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to V_CC either directly or through a resistor.
(3)     The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

☞     Enhanced configuration devices cannot be cascaded.

When performing multi-device configuration, you must generate the configuration device's POF from each project's SOF. You can combine multiple SOFs using the **Convert Programming Files** window in the Quartus II software.

✎     For more information about creating configuration files for multi-device configuration chains, refer to the *Software Settings* section in volume 2 of the *Configuration Handbook.*

In multi-device PS configuration, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, prompting the second device to begin configuration. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. Configuration signals can require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device.

When configuring multiple devices, configuration does not begin until all devices release their OE or nSTATUS pins. Similarly, because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Because all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This low signal drives the OE pin low on the enhanced configuration device and drives nSTATUS low on all devices, causing them to enter a reset state. This behavior is similar to a single device detecting an error.

If the **Auto-restart configuration after error** option is turned on, the devices will automatically initiate reconfiguration if an error occurs. The devices will release their nSTATUS pins after a reset time-out period (maximum of 100 µs). When all the nSTATUS pins are released and pulled high, the configuration device tries to reconfigure the chain. If the **Auto-restart configuration after error** option is turned off, the external system must monitor nSTATUS for errors and then pulse nCONFIG low for at least 2 µs to restart configuration. The external system can pulse nCONFIG if nCONFIG is under system control rather than tied to $V_{CC}$.

The enhanced configuration devices also support parallel configuration of up to eight devices. The n-bit ($n$ = 1, 2, 4, or 8) PS configuration mode allows enhanced configuration devices to concurrently configure devices or a chain of devices. In addition, these devices do not have to be the same device family or density as they can be any combination of Altera devices. An individual enhanced configuration device DATA line is available for each targeted device. Each DATA line can also feed a daisy chain of devices. Figure 11–23 shows how to concurrently configure multiple devices using an enhanced configuration device.

*Figure 11–23. Concurrent PS Configuration of Multiple Devices Using an Enhanced Configuration Device*
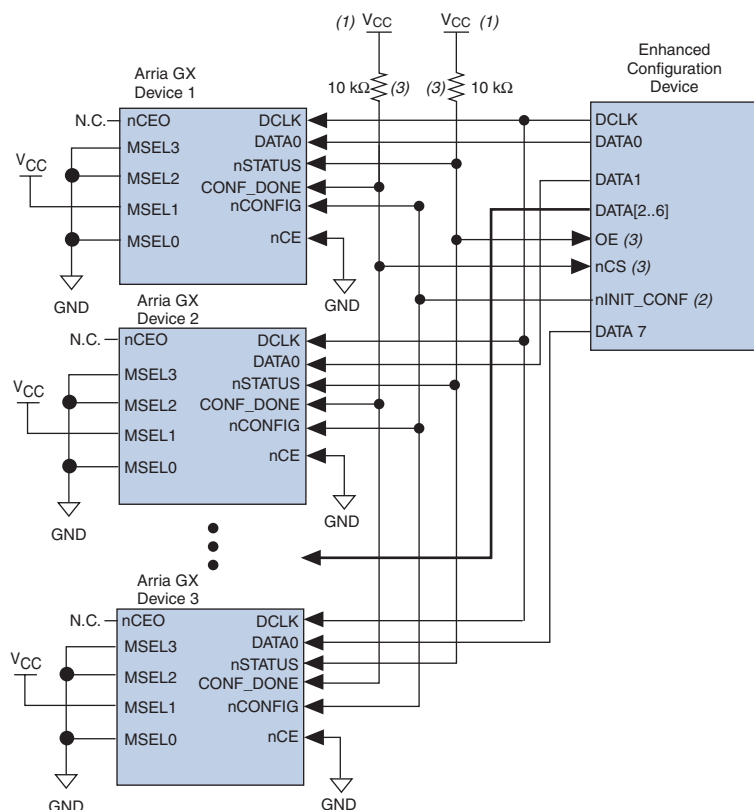


**Notes to Figure 11–23:**

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active, meaning an external pull-up resistor should not be used on the nINIT_CONF-nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to $V_{CC}$ either directly or through a resistor.

(3) The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

The Quartus II software only allows the selection of n-bit PS configuration modes, where n must be 1, 2, 4, or 8. However, you can use these modes to configure any number of devices from 1 to 8. When configuring SRAM-based devices using n-bit PS modes, use Table 11–15 to select the appropriate configuration mode for the fastest configuration times.

| Table 11–15. Recommended Configuration Using n-Bit PS Modes | |
|---|---|
| **Number of Devices** *(1)* | **Recommended Configuration Mode** |
| 1 | 1-bit PS |
| 2 | 2-bit PS |
| 3 | 4-bit PS |
| 4 | 4-bit PS |
| 5 | 8-bit PS |
| 6 | 8-bit PS |
| 7 | 8-bit PS |
| 8 | 8-bit PS |

*Note to Table 11–15:*
(1)   Assume that each DATA line is only configuring one device, not a daisy chain of devices.

For example, if you configure three devices, you would use the 4-bit PS mode. For the DATA0, DATA1, and DATA2 lines, the corresponding SOF data is transmitted from the configuration device to the device. For DATA3, you can leave the corresponding Bit3 line blank in the Quartus II software. On the PCB, leave the DATA3 line from the enhanced configuration device unconnected.

Alternatively, you can daisy chain two devices to one DATA line while the other DATA lines drive one device each. For example, you could use the 2-bit PS mode to drive two devices with DATA Bit0 (two EP2S15 devices) and the third device (EP2S30 device) with DATA Bit1. This 2-bit PS configuration scheme requires less space in the configuration flash memory, but can increase the total system configuration time.

A system may have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) are connected to every device in the chain. Configuration signals can require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DCLK and DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices will start and complete

configuration at the same time. Figure 11–24 shows multi-device PS configuration when the Arria GX devices are receiving the same configuration data.

*Figure 11–24. Multiple-Device PS Configuration Using an Enhanced Configuration Device When Devices Receive the Same Data*
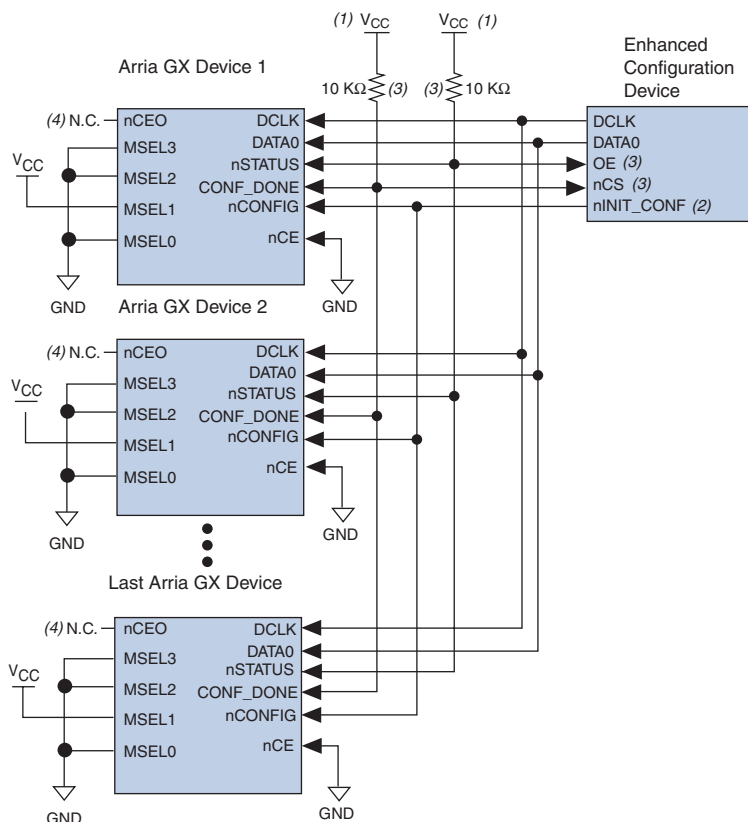


*Notes to Figure 11–24:*
(1)   The pull-up resistor should be connected to the same supply voltage as the configuration device.
(2)   The nINIT_CONF pin is available on enhanced configuration devices and has an internal pull-up resistor that is always active, meaning an external pull-up resistor should not be used on the nINIT_CONF-nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used. If nINIT_CONF is not used, nCONFIG must be pulled to V_CC either directly or through a resistor.
(3)   The enhanced configuration devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.
(4)   The nCEO pins of all devices are left unconnected when configuring the same configuration data into multiple devices.

You can cascade several EPC2 devices to configure multiple Arria GX devices. The first configuration device in the chain is the master configuration device, while the subsequent devices are the slave devices. The master configuration device sends DCLK to the Arria GX devices and to the slave configuration devices. The first EPC device's nCS pin is connected to the CONF_DONE pins of the devices, while its nCASC pin is connected to nCS of the next configuration device in the chain. The last device's nCS input comes from the previous device, while its nCASC pin is left floating. When all data from the first configuration device is sent, it drives nCASC low, which in turn drives nCS on the next configuration device. A configuration device requires less than one clock cycle to activate a subsequent configuration device, so the data stream is uninterrupted.

☞   Enhanced configuration devices cannot be cascaded.

Because all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, the master configuration device stops configuration for the entire chain and the entire chain must be reconfigured. For example, if the master configuration device does not detect CONF_DONE going high at the end of configuration, it resets the entire chain by pulling its OE pin low. This low signal drives the OE pin low on the slave configuration device(s) and drives nSTATUS low on all devices, causing them to enter a reset state. This behavior is similar to the device detecting an error in the configuration data.

Figure 11–25 shows how to configure multiple devices using cascaded EPC2 devices.

*Figure 11–25. Multi-Device PS Configuration Using Cascaded EPC2 Devices*



*Notes to Figure 11–25:*

(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.

(2) The nINIT_CONF pin (available on enhanced configuration devices and EPC2 devices only) has an internal pull-up resistor that is always active, meaning an external pull-up resistor should not be used on the nINIT_CONF-nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used.

(3) The enhanced configuration devices' and EPC2 devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external 10-kΩ pull-up resistors should not be used. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-ups on configuration device** option when generating programming files.

When using enhanced configuration devices or EPC2 devices, nCONFIG of the device can be connected to nINIT_CONF of the configuration device, allowing the INIT_CONF JTAG instruction to initiate device configuration. The nINIT_CONF pin does not need to be connected if its functionality is not used. An internal pull-up resistor on the nINIT_CONF pin is always active in the enhanced configuration devices and the EPC2 devices, which means that you shouldn't be using an external pull-up resistor if nCONFIG is tied to nINIT_CONF. If you are using multiple EPC2 devices to configure a Arria GX device(s), only the first EPC2 has its nINIT_CONF pin tied to the device's nCONFIG pin.

You can use a single configuration chain to configure Arria GX devices with other Altera devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, all of the device CONF_DONE and nSTATUS pins must be tied together.

For more information on configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains* chapter in volume 2 of the *Configuration Handbook*.

Figure 11–26 shows the timing waveform for the PS configuration scheme using a configuration device.

*Figure 11–26. Arria GX PS Configuration Using a Configuration Device Timing Waveform*



*Note to Figure 11–26:*
(1)    The initialization clock can come from the Arria GX device's internal oscillator or the **CLKUSR** pin.

For timing information, refer to the *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* chapter or the *Configuration Devices for SRAM-Based LUT Devices Data Sheet* chapter in volume 2 of the *Configuration Handbook*.

Device configuration options and how to create configuration files are discussed further in the *Software Settings* chapter of the *Configuration Handbook*.

## PS Configuration Using a Download Cable

In this section, the generic term "download cable" includes the Altera USB-Blaster™ universal serial bus (USB) port download cable, MasterBlaster™ serial/USB communications cable, ByteBlaster™ II parallel port download cable, and the ByteBlaster MV parallel port download cable.

In PS configuration with a download cable, an intelligent host (such as a PC) transfers data from a storage device to the device via the USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV cable.

Upon power-up, the Arria GX devices go through a POR. The POR delay is dependent on the PORSEL pin setting. When PORSEL is driven low, the POR time is approximately 100 ms. If PORSEL is driven high, the POR time is approximately 12 ms. During POR, the device will reset, hold nSTATUS low, and tri-state all user I/O pins. Once the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

☞ The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *DC & Switching Characteristics* chapter in volume 1 of the *Arria GX Device Handbook.*

The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. To initiate configuration in this scheme, the download cable generates a low-to-high transition on the nCONFIG pin.

☞ To begin configuration, power the $V_{CCINT}$, $V_{CCIO}$, and $V_{CCPD}$ voltages (for the banks where the configuration and JTAG pins reside) to the appropriate voltage levels.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 10-kΩ pull-up resistor. Once nSTATUS is released the device is ready to receive configuration data and the configuration stage begins. The programming hardware or download cable then places the configuration data one bit at a time on the device's DATA0 pin. The configuration data is clocked into the target device until CONF_DONE goes high. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize.

When using a download cable, setting the **Auto-restart configuration after error** option does not affect the configuration cycle because you must manually restart configuration in the Quartus II software when an error occurs. Additionally, the **Enable user-supplied start-up clock** (**CLKUSR**) option has no affect on the device initialization because this option is disabled in the SOF when programming the device using the Quartus II programmer and download cable. Therefore, if you turn on the CLKUSR option, you do not need to provide a clock on CLKUSR when you are configuring the device with the Quartus II programmer and a download cable. Figure 11–27 shows PS configuration for Arria GX devices using a USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV cable.

*Figure 11–27. PS Configuration Using a USB Blaster, MasterBlaster, ByteBlaster II or ByteBlasterMV Cable*
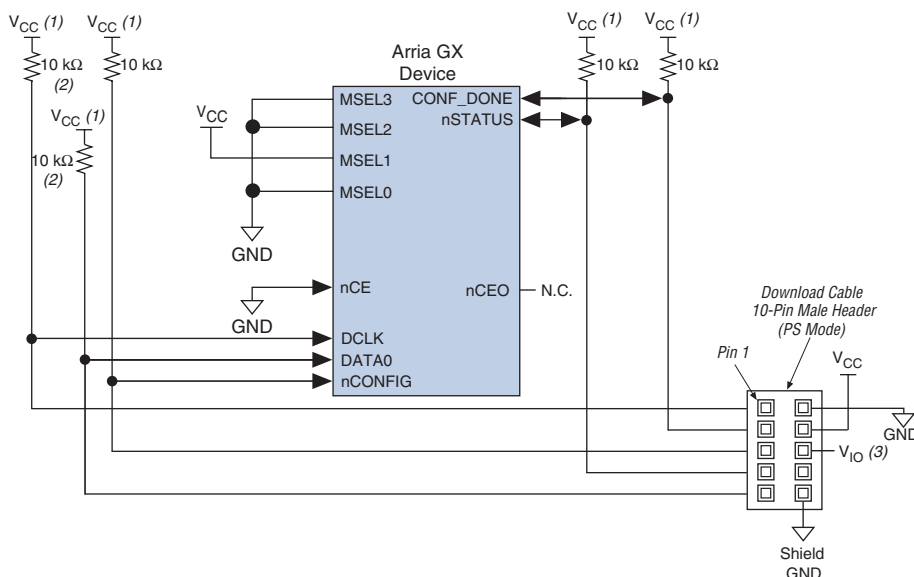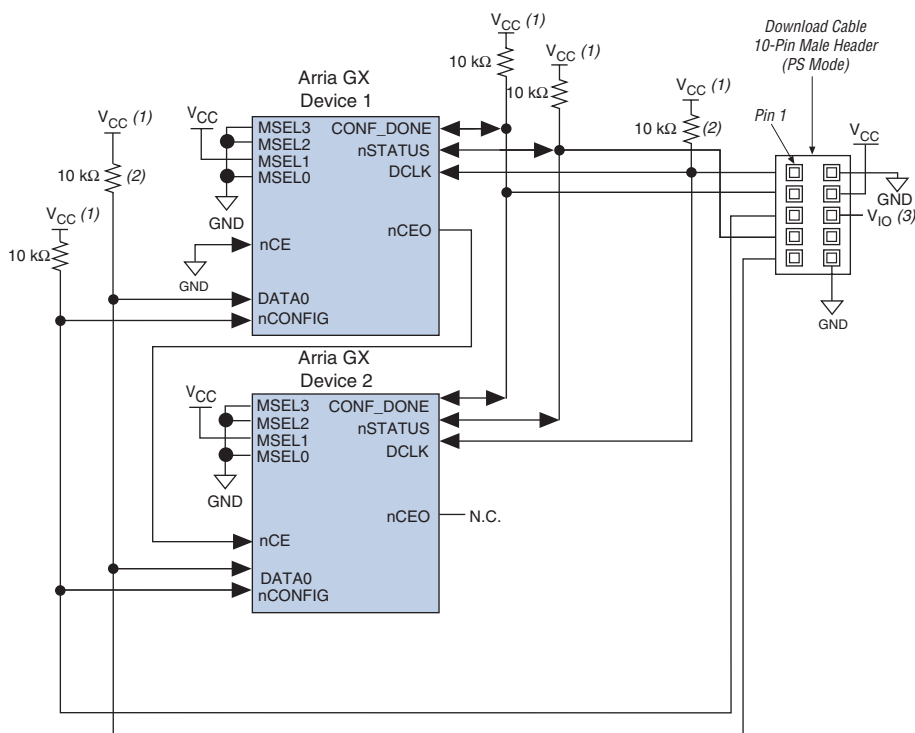


*Notes to Figure 11–27:*

(1) The pull-up resistor should be connected to the same supply voltage as the USB Blaster, MasterBlaster ($V_{IO}$ pin), ByteBlaster II, or ByteBlasterMV cable.

(2) The pull-up resistors on DATA0 and DCLK are only needed if the download cable is the only configuration scheme used on your board. This ensures that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a configuration device, the pull-up resistors on DATA0 and DCLK are not needed.

(3) Pin 6 of the header is a $V_{IO}$ reference voltage for the MasterBlaster output driver. $V_{IO}$ should match the device's $V_{CCIO}$. Refer to the *MasterBlaster Serial/USB Communications Cable User Guide* for this value. In the ByteBlasterMV cable, this pin is a no connect. In the USB Blaster and ByteBlaster II cables, this pin is connected to nCE when it is used for active serial programming, otherwise it is a no connect.

You can use a download cable to configure multiple Arria GX devices by connecting each device's nCEO pin to the subsequent device's nCE pin. The first device's nCE pin is connected to GND while its nCEO pin is connected to the nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. All other configuration pins, nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE are connected to every device in the chain. Because all CONF_DONE pins are tied together, all devices in the chain initialize and enter user mode at the same time.

In addition, because the nSTATUS pins are tied together, the entire chain halts configuration if any device detects an error. The **Auto-restart configuration after error** option does not affect the configuration cycle because you must manually restart configuration in the Quartus II software when an error occurs.

Figure 11–28 shows how to configure multiple Arria GX devices with a download cable.

*Figure 11–28. Multi-Device PS Configuration using a USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV Cable*



*Notes to Figure 11–28:*
(1) The pull-up resistor should be connected to the same supply voltage as the USB Blaster, MasterBlaster ($V_{IO}$ pin), ByteBlaster II, or ByteBlasterMV cable.
(2) The pull-up resistors on DATA0 and DCLK are only needed if the download cable is the only configuration scheme used on your board. This is to ensure that DATA0 and DCLK are not left floating after configuration. For example, if you are also using a configuration device, the pull-up resistors on DATA0 and DCLK are not needed.
(3) Pin 6 of the header is a $V_{IO}$ reference voltage for the MasterBlaster output driver. $V_{IO}$ should match the device's $V_{CCIO}$. Refer to the *MasterBlaster Serial/USB Communications Cable User Guide* for this value. In the ByteBlasterMV cable, this pin is a no connect. In the USB Blaster and ByteBlaster II cables, this pin is connected to nCE when it is used for active serial programming, otherwise it is a no connect.

If you are using a download cable to configure device(s) on a board that also has configuration devices, electrically isolate the configuration device from the target device(s) and cable. One way of isolating the configuration device is to add logic, such as a multiplexer, that can select between the configuration device and the cable. The multiplexer chip allows bidirectional transfers on the nSTATUS and CONF_DONE signals. Another option is to add switches to the five common signals (nCONFIG, nSTATUS, DCLK, DATA0, and CONF_DONE) between the cable and the configuration device. The last option is to remove the configuration device from the board when configuring the device with the cable. Figure 11–29 shows a combination of a configuration device and a download cable to configure an device.

*Figure 11–29. PS Configuration with a Download Cable and Configuration Device Circuit*
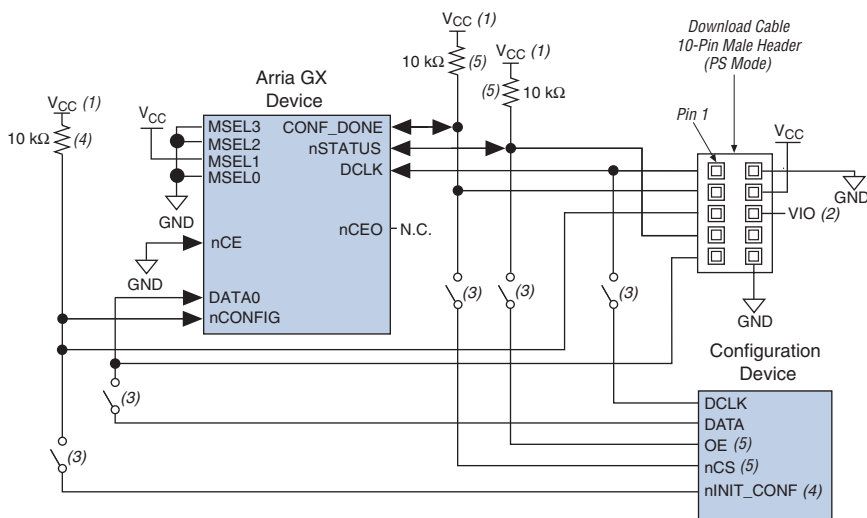


*Notes to Figure 11–29:*
(1) The pull-up resistor should be connected to the same supply voltage as the configuration device.
(2) Pin 6 of the header is a $V_{IO}$ reference voltage for the MasterBlaster output driver. $V_{IO}$ should match the device's $V_{CCIO}$. Refer to the *MasterBlaster Serial/USB Communications Cable User Guide* for this value. In the ByteBlasterMV cable, this pin is a no connect. In the USB Blaster and ByteBlaster II cables, this pin is connected to nCE when it is used for active serial programming, otherwise it is a no connect.
(3) You should not attempt configuration with a download cable while a configuration device is connected to an Arria GX device. Instead, you should either remove the configuration device from its socket when using the download cable or place a switch on the five common signals between the download cable and the configuration device.
(4) The nINIT_CONF pin (available on enhanced configuration devices and EPC2 devices only) has an internal pull-up resistor that is always active. This means an external pull-up resistor should not be used on the nINIT_CONF-nCONFIG line. The nINIT_CONF pin does not need to be connected if its functionality is not used.
(5) The enhanced configuration devices' and EPC2 devices' OE and nCS pins have internal programmable pull-up resistors. If internal pull-up resistors are used, external pull-up resistors should not be used on these pins. The internal pull-up resistors are used by default in the Quartus II software. To turn off the internal pull-up resistors, check the **Disable nCS and OE pull-up resistors on configuration device** option when generating programming files.

For more information on how to use the USB Blaster, MasterBlaster, ByteBlaster II or ByteBlasterMV cables, refer to the following data sheets:

■ *USB-Blaster Download Cable User Guide*
■ *MasterBlaster Serial/USB Communications Cable User Guide*
■ *ByteBlaster II Download Cable User Guide*
■ *ByteBlasterMV Download Cable User Guide*

# Passive Parallel Asynchronous Configuration

Passive parallel asynchronous (PPA) configuration uses an intelligent host, such as a microprocessor, to transfer configuration data from a storage device, such as flash memory, to the target Arria GX device.

Configuration data can be stored in RBF, HEX, or TTF format. The host system outputs byte-wide data and the accompanying strobe signals to the device. When using PPA, pull the DCLK pin high through a 10-kΩ pull-up resistor to prevent unused configuration input pins from floating.

☞ You cannot use the Arria GX decompression feature if you are configuring your Arria GX device using PPA mode.

Table 11–16 shows the MSEL pin settings when using the PS configuration scheme.

*Table 11–16. Arria GX MSEL Pin Settings for PPA Configuration Schemes*

| Configuration Scheme | MSEL3 | MSEL2 | MSEL1 | MSEL0 |
|---|---|---|---|---|
| PPA | 0 | 0 | 0 | 1 |
| Remote System Upgrade PPA *(1)* | 0 | 1 | 0 | 1 |

*Note to Table 11–16:*

(1)  This scheme requires that you drive the RUnLU pin to specify either remote update or local update. For more information about remote system upgrades in Arria GX devices, refer to the *Remote System Upgrades with Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook.*

Figure 11–30 shows the configuration interface connections between the device and a microprocessor for single device PPA configuration. The microprocessor or an optional address decoder can control the device's chip select pins, nCS and CS. The address decoder allows the microprocessor to select the Arria GX device by accessing a particular address, which simplifies the configuration process. Hold the nCS and CS pins active during configuration and initialization.

*Figure 11–30. Single Device PPA Configuration Using a Microprocessor*
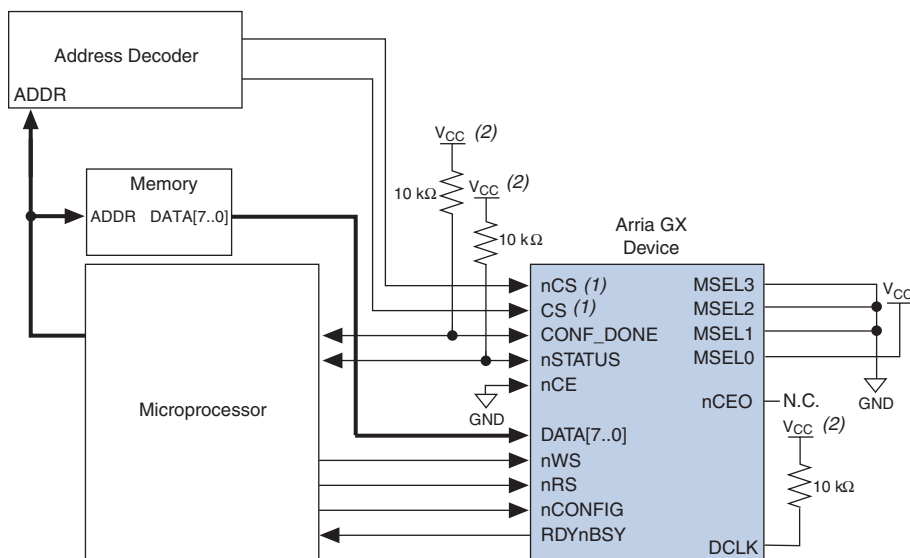


*Notes to Figure 11–30:*

(1)    If not used, the CS pin can be connected to V<sub>CC</sub> directly. If not used, the nCS pin can be connected to GND directly.
(2)    The pull-up resistor should be connected to a supply that provides an acceptable input signal for the device. V<sub>CC</sub> should be high enough to meet the V<sub>IH</sub> specification of the I/O on the device and the external host.

During PPA configuration, it is only required to use either the nCS or CS pin. Therefore, if you are using only one chip-select input, the other must be tied to the active state. For example, nCS can be tied to ground while CS is toggled to control configuration. The device's nCS or CS pins can be toggled during PPA configuration if the design meets the specifications set for $t_{CSSU}$, $t_{WSP}$, and $t_{CSH}$ listed in Table 11–17 on page 11–80.

Upon power-up, the Arria GX devices go through a POR. The POR delay is dependent on the PORSEL pin setting. When PORSEL is driven low, the POR time is approximately 100 ms. If PORSEL is driven high, the POR time is approximately 12 ms. During POR, the device will reset, hold nSTATUS low, and tri-state all user I/O pins. Once the device successfully exits POR, all user I/O pins continue to be tri-stated. If nIO_pullup is driven low during power-up and configuration, the user I/O pins and dual-purpose I/O pins will have weak pull-up resistors which are on (after POR) before and during configuration. If nIO_pullup is driven high, the weak pull-up resistors are disabled.

The value of the weak pull-up resistors on the I/O pins that are on before and during configuration can be found in the *DC & Switching Characteristics* chapter in volume 1 of the *Arria GX Device Handbook*.

The configuration cycle consists of three stages: reset, configuration, and initialization. While nCONFIG or nSTATUS are low, the device is in reset. To initiate configuration, the microprocessor must generate a low-to-high transition on the nCONFIG pin.

☞    To begin configuration, power the $V_{CCINT}$, $V_{CCIO}$, and $V_{CCPD}$ voltages (for the banks where the configuration and JTAG pins reside) to the appropriate voltage levels.

When nCONFIG goes high, the device comes out of reset and releases the open-drain nSTATUS pin, which is then pulled high by an external 10-kΩ pull-up resistor. Once nSTATUS is released, the device is ready to receive configuration data and the configuration stage begins. When nSTATUS is pulled high, the microprocessor should then assert the target device's nCS pin low and/or CS pin high. Next, the microprocessor places an 8-bit configuration word (one byte) on the target device's DATA[7..0] pins and pulses the nWS pin low.

On the rising edge of nWS, the target device latches in a byte of configuration data and drives its RDYnBSY signal low, which indicates it is processing the byte of configuration data. The microprocessor can then perform other system functions while the Arria GX device is processing the byte of configuration data.

During the time RDYnBSY is low, the Arria GX device internally processes the configuration data using its internal oscillator (typically 100 MHz). When the device is ready for the next byte of configuration data, it will drive RDYnBSY high. If the microprocessor senses a high signal when it polls RDYnBSY, the microprocessor sends the next byte of configuration data to the device.

Alternatively, the nRS signal can be strobed low, causing the RDYnBSY signal to appear on DATA7. Because RDYnBSY does not need to be monitored, this pin doesn't need to be connected to the microprocessor. Do not drive data onto the data bus while nRS is low because it will cause contention on the DATA7 pin. If you are not using the nRS pin to monitor configuration, it should be tied high.

To simplify configuration and save an I/O port, the microprocessor can wait for the total time of $t_{BUSY}$ (max) + $t_{RDY2WS}$ + $t_{W2SB}$ before sending the next data byte. In this set-up, nRS should be tied high and RDYnBSY does not need to be connected to the microprocessor. The $t_{BUSY}$, $t_{RDY2WS}$, and $t_{W2SB}$ timing specifications are listed in

Next, the microprocessor checks nSTATUS and CONF_DONE. If nSTATUS is not low and CONF_DONE is not high, the microprocessor sends the next data byte. However, if nSTATUS is not low and all the configuration data has been received, the device is ready for initialization. The CONF_DONE pin will go high one byte early in parallel configuration (FPP and PPA) modes. The last byte is required for serial configuration (AS and PS) modes. A low-to-high transition on CONF_DONE indicates configuration is complete and initialization of the device can begin. The open-drain CONF_DONE pin is pulled high by an external 10-kΩ pull-up resistor. The CONF_DONE pin must have an external 10-kΩ pull-up resistor in order for the device to initialize.

In Arria GX devices, the initialization clock source is either the internal oscillator (typically 10 MHz) or the optional CLKUSR pin. By default, the internal oscillator is the clock source for initialization. If the internal oscillator is used, the Arria GX device provides itself with enough clock cycles for proper initialization. Therefore, if the internal oscillator is the initialization clock source, sending the entire configuration file to the device is sufficient to configure and initialize the device.

You also have the flexibility to synchronize initialization of multiple devices or to delay initialization with the CLKUSR option. The **Enable user-supplied start-up clock** (**CLKUSR**) option can be turned on in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. Supplying a clock on CLKUSR does not affect the configuration process. After CONF_DONE goes high, CLKUSR is enabled after the time specified as $t_{CD2CU}$. After this time period elapses, Arria GX devices require 299 clock cycles to initialize properly and enter user mode. Arria GX devices support a CLKUSR $f_{MAX}$ of 100 MHz.

An optional INIT_DONE pin is available, which signals the end of initialization and the start of user-mode with a low-to-high transition. This **Enable INIT_DONE Output** option is available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box. If the INIT_DONE pin is used, it is high because of an external 10-kΩ pull-up resistor when nCONFIG is low and during the beginning of configuration. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin goes low. When initialization is complete, the INIT_DONE pin is released and pulled high. The microprocessor must be able to detect this low-to-high transition that signals the device has entered user mode. When initialization is complete, the device enters user mode. In user-mode, the user I/O pins no longer have weak pull-up resistors and function as assigned in your design.

To ensure DATA[7..0] is not left floating at the end of configuration, the microprocessor must drive them either high or low, whichever is convenient on your board. After configuration, the nCS, CS, nRS, nWS, RDYnBSY, and DATA[7..0] pins can be used as user I/O pins. When choosing the PPA scheme in the Quartus II software as a default, these I/O pins are tri-stated in user mode and should be driven by the microprocessor. To change this default option in the Quartus II software, select the **Dual-Purpose Pins** tab of the **Device & Pin Options** dialog box.

If an error occurs during configuration, the device drives its nSTATUS pin low, resetting itself internally. The low signal on the nSTATUS pin also alerts the microprocessor that there is an error. If the **Auto-restart configuration after error** option-available in the Quartus II software from the **General** tab of the **Device & Pin Options** dialog box-is turned on, the device releases nSTATUS after a reset time-out period (maximum of 100 µs). After nSTATUS is released and pulled high by a pull-up resistor, the microprocessor can try to reconfigure the target device without needing to pulse nCONFIG low. If this option is turned off, the microprocessor must generate a low-to-high transition (with a low pulse of at least 2 µs) on nCONFIG to restart the configuration process.

The microprocessor can also monitor the CONF_DONE and INIT_DONE pins to ensure successful configuration. To detect errors and determine when programming completes, monitor the CONF_DONE pin with the microprocessor. If the microprocessor sends all configuration data but CONF_DONE or INIT_DONE has not gone high, the microprocessor must reconfigure the target device.

☞ If you are using the optional CLKUSR pin and nCONFIG is pulled low to restart configuration during device initialization, ensure CLKUSR continues toggling during the time nSTATUS is low (maximum of 100 µs).

When the device is in user-mode, a reconfiguration can be initiated by transitioning the nCONFIG pin low-to-high. The nCONFIG pin should go low for at least 2 µs. When nCONFIG is pulled low, the device also pulls nSTATUS and CONF_DONE low and all I/O pins are tri-stated. Once nCONFIG returns to a logic high level and nSTATUS is released by the device, reconfiguration begins.

Figure 11–31 shows how to configure multiple Arria GX devices using a microprocessor. This circuit is similar to the PPA configuration circuit for a single device, except the devices are cascaded for multi-device configuration.

*Figure 11–31. Multi-Device PPA Configuration Using a Microprocessor*
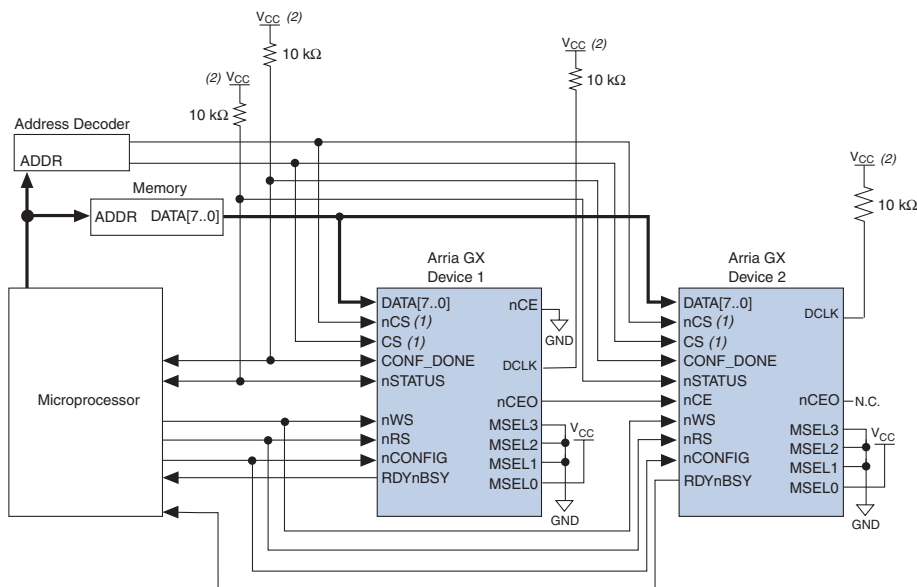


*Notes to Figure 11–31:*
(1)   If not used, the CS pin can be connected to $V_{CC}$ directly. If not used, the nCS pin can be connected to GND directly.
(2)   The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain. $V_{CC}$ should be high enough to meet the $V_{IH}$ specification of the I/O on the device and the external host.

In the multi-device PPA configuration, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. The second device in the chain begins configuration within one clock cycle. Therefore, the transfer of data destinations is transparent to the microprocessor.

Each device's RDYnBSY pin can have a separate input to the microprocessor. Alternatively, if the microprocessor is pin limited, all the RDYnBSY pins can feed an AND gate and the output of the AND gate can feed the microprocessor. For example, if you have two devices in a PPA configuration chain, the second device's RDYnBSY pin will be high during the time that the first device is being configured. When the first device has been successfully configured, it will drive nCEO low to activate the next device in the chain and drive its RDYnBSY pin high. Therefore, because

RDYnBSY signal is driven high before configuration and after configuration before entering user-mode, the device being configured will govern the output of the AND gate.

The nRS signal can be used in the multi-device PPA chain because the Arria GX devices tri-state the DATA[7..0] pins before configuration and after configuration before entering user-mode to avoid contention. Therefore, only the device that is currently being configured responds to the nRS strobe by asserting DATA7.

All other configuration pins (nCONFIG, nSTATUS, DATA[7..0], nCS, CS, nWS, nRS, and CONF_DONE) are connected to every device in the chain. It is not necessary to tie nCS and CS together for every device in the chain, as each device's nCS and CS input can be driven by a separate source. Configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DATA lines are buffered for every fourth device. Because all device CONF_DONE pins are tied together, all devices initialize and enter user mode at the same time.

Because all nSTATUS and CONF_DONE pins are tied together, if any device detects an error, configuration stops for the entire chain and the entire chain must be reconfigured. For example, if the first device flags an error on nSTATUS, it resets the chain by pulling its nSTATUS pin low. This behavior is similar to a single device detecting an error.

If the **Auto-restart configuration after error** option is turned on, the devices release their nSTATUS pins after a reset time-out period (maximum of 100 μs). After all nSTATUS pins are released and pulled high, the microprocessor can try to reconfigure the chain without needing to pulse nCONFIG low. If this option is turned off, the microprocessor must generate a low-to-high transition (with a low pulse of at least 2 μs) on nCONFIG to restart the configuration process.

In your system, you may have multiple devices that contain the same configuration data. To support this configuration scheme, all device nCE inputs are tied to GND, while nCEO pins are left floating. All other configuration pins (nCONFIG, nSTATUS, DATA[7..0], nCS, CS, nWS, nRS, and CONF_DONE) are connected to every device in the chain. Configuration signals may require buffering to ensure signal integrity and prevent clock skew problems. Ensure that the DATA lines are buffered for every fourth device. Devices must be the same density and package. All devices start and complete configuration at the same time. Figure 11–32 shows multi-device PPA configuration when both devices are receiving the same configuration data.

*Figure 11–32. Multiple-Device PPA Configuration Using a Microprocessor When Both Devices Receive the Same Data*
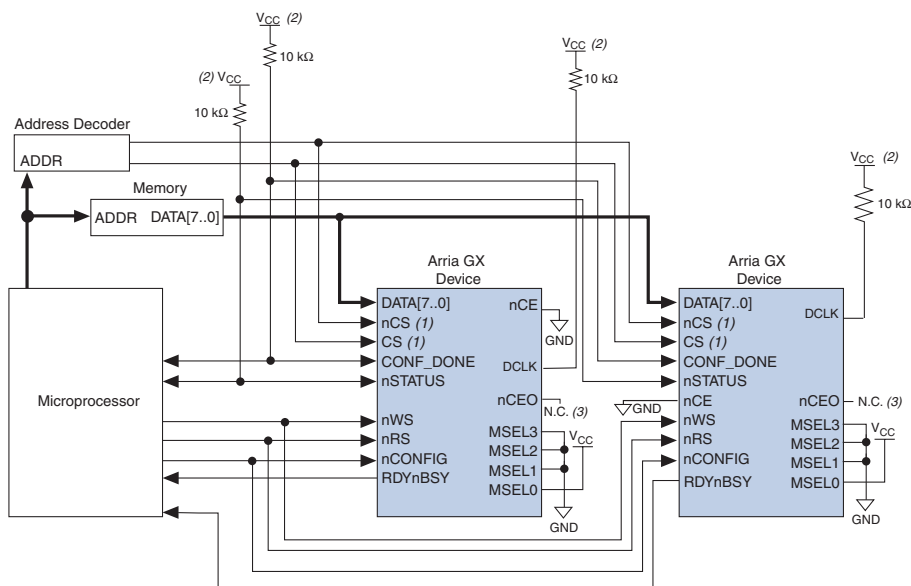


*Notes to Figure 11–32:*

(1)    If not used, the CS pin can be connected to $V_{CC}$ directly. If not used, the nCS pin can be connected to GND directly.
(2)    The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain. $V_{CC}$ should be high enough to meet the $V_{IH}$ specification of the I/O on the device and the external host.
(3)    The nCEO pins of both devices are left unconnected when configuring the same configuration data into multiple devices.

You can use a single configuration chain to configure Arria GX devices with other Altera devices that support PPA configuration, such as Stratix, Mercury™, APEX™ 20K, ACEX® 1K, and FLEX® 10KE devices. To ensure that all devices in the chain complete configuration at the same time or that an error flagged by one device initiates reconfiguration in all devices, all of the device CONF_DONE and nSTATUS pins must be tied together.

For more information about configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains* chapter in volume 2 of the *Configuration Handbook*.

*PPA Configuration Timing*

Figure 11–33 shows the timing waveform for the PPA configuration scheme using a microprocessor.

*Figure 11–33. Arria GX PPA Configuration Timing Waveform Using nWS*    *Note (1)*
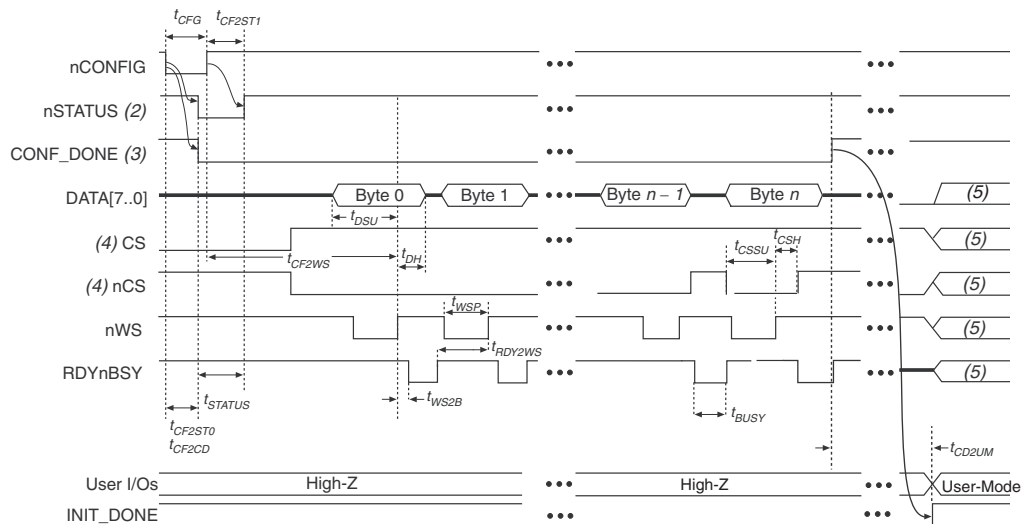


**Notes to Figure 11–33:**
(1) The beginning of this waveform shows the device in user-mode. In user-mode, nCONFIG, nSTATUS, and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.
(2) Upon power-up, Arria GX devices hold nSTATUS low for the time of the POR delay.
(3) Upon power-up, before and during configuration, CONF_DONE is low.
(4) The user can toggle nCS or CS during configuration if the design meets the specification for $t_{CSSU}$, $t_{WSP}$, and $t_{CSH}$.
(5) DATA[7..0], CS, nCS, nWS, nRS, and RDYnBSY are available as user I/O pins after configuration and the state of theses pins depends on the dual-purpose pin settings.

Figure 11–34 shows the timing waveform for the PPA configuration scheme when using a strobed nRS and nWS signal.

*Figure 11–34. Arria GX PPA Configuration Timing Waveform Using nRS & nWS    Note (1)*
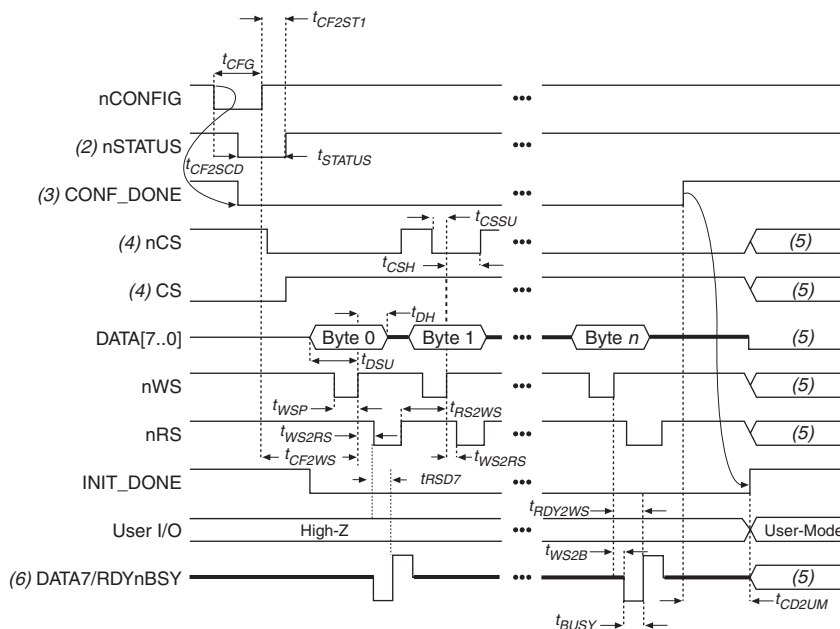


Notes to *Figure 11–34*:
(1)    The beginning of this waveform shows the device in user-mode. In user-mode, nCONFIG, nSTATUS, and CONF_DONE are at logic high levels. When nCONFIG is pulled low, a reconfiguration cycle begins.
(2)    Upon power-up, Arria GX devices hold nSTATUS low for the time of the POR delay.
(3)    Upon power-up, before and during configuration, CONF_DONE is low.
(4)    The user can toggle nCS or CS during configuration if the design meets the specification for $t_{CSSU}$, $t_{WSP}$, and $t_{CSH}$.
(5)    DATA[7..0], CS, nCS, nWS, nRS, and RDYnBSY are available as user I/O pins after configuration and the state of theses pins depends on the dual-purpose pin settings.
(6)    DATA7 is a bidirectional pin. It is an input for configuration data input, but it is an output to show the status of RDYnBSY.

Table 11–17 defines the timing parameters for Arria GX devices for PPA configuration.

*Table 11–17. PPA Timing Parameters for Arria GX Devices  (Part 1 of 2)*    *Note (1)*

| Symbol | Parameter | Min | Max | Units |
|--------|-----------|-----|-----|-------|
| $t_{CF2CD}$ | nCONFIG low to CONF_DONE low | | 800 | ns |
| $t_{CF2ST0}$ | nCONFIG low to nSTATUS low | | 800 | ns |

*Table 11–17. PPA Timing Parameters for Arria GX Devices  (Part 2 of 2)*     *Note (1)*

| Symbol | Parameter | Min | Max | Units |
|--------|-----------|-----|-----|-------|
| t$_{CFG}$ | nCONFIG low pulse width | 2 | | µs |
| t$_{STATUS}$ | nSTATUS low pulse width | 10 | 100 *(2)* | µs |
| t$_{CF2ST1}$ | nCONFIG high to nSTATUS high | | 100 *(2)* | µs |
| t$_{CSSU}$ | Chip select setup time before rising edge on nWS | 10 | | ns |
| t$_{CSH}$ | Chip select hold time after rising edge on nWS | 0 | | ns |
| t$_{CF2WS}$ | nCONFIG high to first rising edge on nWS | 100 | | µs |
| t$_{ST2WS}$ | nSTATUS high to first rising edge on nWS | 2 | | µs |
| t$_{DSU}$ | Data setup time before rising edge on nWS | 10 | | ns |
| t$_{DH}$ | Data hold time after rising edge on nWS | 0 | | ns |
| t$_{WSP}$ | nWS low pulse width | 15 | | ns |
| t$_{WS2B}$ | nWS rising edge to RDYnBSY low | | 20 | ns |
| t$_{BUSY}$ | RDYnBSY low pulse width | 7 | 45 | ns |
| t$_{RDY2WS}$ | RDYnBSY rising edge to nWS rising edge | 15 | | ns |
| t$_{WS2RS}$ | nWS rising edge to nRS falling edge | 15 | | ns |
| t$_{RS2WS}$ | nRS rising edge to nWS rising edge | 15 | | ns |
| t$_{RSD7}$ | nRS falling edge to DATA7 valid with RDYnBSY signal | | 20 | ns |
| t$_R$ | Input rise time | | 40 | ns |
| t$_F$ | Input fall time | | 40 | ns |
| t$_{CD2UM}$ | CONF_DONE high to user mode *(3)* | 20 | 100 | µs |
| t$_{CD2CU}$ | CONF_DONE high to CLKUSR enabled | 40 | | ns |
| t$_{CD2UMC}$ | CONF_DONE high to user mode with **CLKUSR** option on | t$_{CD2CU}$ + (299 × CLKUSR period) | | |

*Notes to Table 11–17:*
(1)  This information is preliminary.
(2)  This value is obtainable if users do not delay configuration by extending the nCONFIG or nSTATUS low pulse width.
(3)  The minimum and maximum numbers apply only if the internal oscillator is chosen as the clock source for starting up the device.

Device configuration options and how to create configuration files are discussed further in the *Software Settings* section of the *Configuration Handbook*.

# JTAG Configuration

The JTAG has developed a specification for boundary-scan testing (BST). This boundary-scan test architecture offers the capability to efficiently test components on PCBs with tight lead spacing. The BST architecture can test pin connections without using physical test probes and capture functional data while a device is operating normally. The JTAG circuitry can also be used to shift configuration data into the device. The Quartus II software automatically generates SOFs that can be used for JTAG configuration with a download cable in the Quartus II software programmer.

For more information on JTAG boundary-scan testing, refer to the following documents:

■ *IEEE 1149.1 (JTAG) Boundary-Scan Testing for Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*
■ *Jam Programming Support - JTAG Technologies*

Arria GX devices are designed such that JTAG instructions have precedence over any device configuration modes. Therefore, JTAG configuration can take place without waiting for other configuration modes to complete. For example, if you attempt JTAG configuration of Arria GX devices during PS configuration, PS configuration is terminated and JTAG configuration begins.

☞ You cannot use the Arria GX decompression feature if you are configuring your Arria GX device when using JTAG-based configuration.

A device operating in JTAG mode uses four required pins, TDI, TDO, TMS, and TCK, and one optional pin, TRST. The TCK pin has an internal weak pull-down resistor, while the TDI, TMS, and TRST pins have weak internal pull-up resistors (typically 25 kΩ). The TDO output pin is powered by $V_{CCIO}$ in I/O bank 4. All of the JTAG input pins are powered by the 3.3-V $V_{CCPD}$ pin. All user I/O pins are tri-stated during JTAG configuration. Table 11–18 explains each JTAG pin's function.

☞ The TDO output is powered by the $V_{CCIO}$ power supply of I/O bank 4.

For recommendations on how to connect a JTAG chain with multiple voltages across the devices in the chain, refer to the *IEEE 1149.1 (JTAG) Boundary-Scan Testing for Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*.

*Table 11–18. Dedicated JTAG Pins*

| Pin Name | Pin Type | Description |
|----------|----------|-------------|
| TDI | Test data input | Serial input pin for instructions as well as test and programming data. Data is shifted in on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to $V_{CC}$. |
| TDO | Test data output | Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by leaving this pin unconnected. |
| TMS | Test mode select | Input pin that provides the control signal to determine the transitions of the TAP controller state machine. Transitions within the state machine occur on the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to $V_{CC}$. |
| TCK | Test clock input | The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to GND. |
| TRST | Test reset input (optional) | Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to IEEE Std. 1149.1. If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to GND. |

During JTAG configuration, data can be downloaded to the device on the PCB through the USB Blaster, MasterBlaster, ByteBlaster II, or ByteBlasterMV download cable. Configuring devices through a cable is similar to programming devices in-system, except the TRST pin should be connected to $V_{CC}$. This ensures that the TAP controller is not reset. Figure 11–35 shows JTAG configuration of a single Arria GX device.

*Figure 11–35. JTAG Configuration of a Single Device Using a Download Cable*
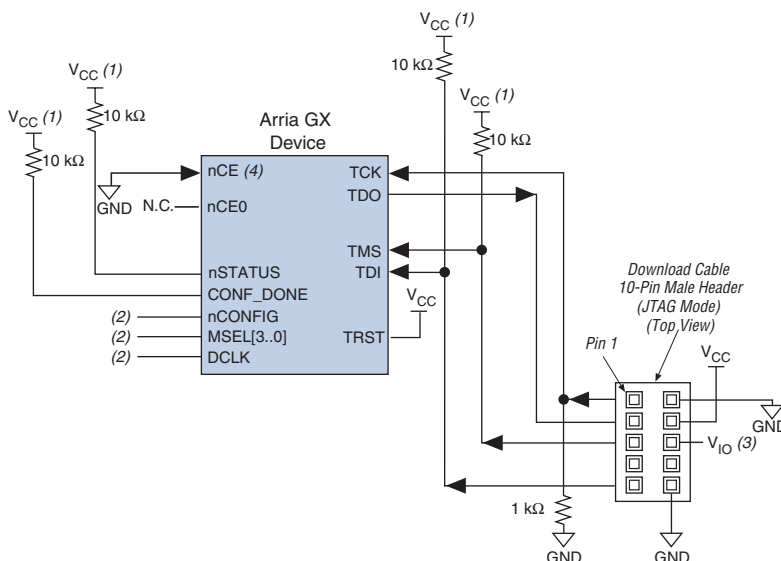


*Notes to Figure 11–35:*

(1) The pull-up resistor should be connected to the same supply voltage as the USB Blaster, MasterBlaster ($V_{IO}$ pin), ByteBlaster II, or ByteBlasterMV cable.

(2) The nCONFIG, MSEL[3..0] pins should be connected to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect nCONFIG to $V_{CC}$, and MSEL[3..0] to ground. Pull DCLK either high or low, whichever is convenient on your board.

(3) Pin 6 of the header is a $V_{IO}$ reference voltage for the MasterBlaster output driver. $V_{IO}$ should match the device's $V_{CCIO}$. Refer to the *MasterBlaster Serial/USB Communications Cable User Guide* for this value. In the ByteBlasterMV cable, this pin is a no connect. In the USB Blaster and ByteBlaster II cables, this pin is connected to nCE when it is used for active serial programming, otherwise it is a no connect.

(4) nCE must be connected to GND or driven low for successful JTAG configuration.

To configure a single device in a JTAG chain, the programming software places all other devices in bypass mode. In bypass mode, devices pass programming data from the TDI pin to the TDO pin through a single bypass register without being affected internally. This scheme enables the programming software to program or verify the target device. Configuration data driven into the device appears on the TDO pin one clock cycle later.

The Quartus II software verifies successful JTAG configuration upon completion. At the end of configuration, the software checks the state of CONF_DONE through the JTAG port. When Quartus II generates a (.**jam**) file for a multi-device chain, it contains instructions so that all the devices in the chain will be initialized at the same time. If CONF_DONE is not high, the Quartus II software indicates that configuration has failed. If

CONF_DONE is high, the software indicates that configuration was successful. After the configuration bitstream is transmitted serially via the JTAG TDI port, the TCK port is clocked an additional 299 cycles to perform device initialization.

Arria GX devices have dedicated JTAG pins that always function as JTAG pins. Not only can you perform JTAG testing on Arria GX devices before and after, but also during configuration. While other device families do not support JTAG testing during configuration, Arria GX devices support the bypass, idcode, and sample instructions during configuration without interrupting configuration. All other JTAG instructions may only be issued by first interrupting configuration and reprogramming I/O pins using the CONFIG_IO instruction.

The CONFIG_IO instruction allows I/O buffers to be configured via the JTAG port and when issued, interrupts configuration. This instruction allows you to perform board-level testing prior to configuring the Arria GX device or waiting for a configuration device to complete configuration. Once configuration has been interrupted and JTAG testing is complete, the part must be reconfigured via JTAG (PULSE_CONFIG instruction) or by pulsing nCONFIG low.

The chip-wide reset (DEV_CLRn) and chip-wide output enable (DEV_OE) pins on Arria GX devices do not affect JTAG boundary-scan or programming operations. Toggling these pins does not affect JTAG operations (other than the usual boundary-scan operation).

When designing a board for JTAG configuration of Arria GX devices, consider the dedicated configuration pins. Table 11–19 shows how these pins should be connected during JTAG configuration.

When programming a JTAG device chain, one JTAG-compatible header is connected to several devices. The number of devices in the JTAG chain is limited only by the drive capability of the download cable. When four or more devices are connected in a JTAG chain, Altera recommends buffering the TCK, TDI, and TMS pins with an on-board buffer.

*Table 11–19. Dedicated Configuration Pin Connections During JTAG Configuration*

| Signal | Description |
|--------|-------------|
| nCE | On all Arria GX devices in the chain, nCE should be driven low by connecting it to ground, pulling it low via a resistor, or driving it by some control circuitry. For devices that are also in multi-device FPP, AS, PS, or PPA configuration chains, the nCE pins should be connected to GND during JTAG configuration or JTAG configured in the same order as the configuration chain. |
| nCEO | On all Arria GX devices in the chain, nCEO can be left floating or connected to the nCE of the next device. |
| MSEL | These pins must not be left floating. These pins support whichever non-JTAG configuration is used in production. If only JTAG configuration is used, tie these pins to ground. |
| nCONFIG | Driven high by connecting to $V_{CC}$, pulling up via a resistor, or driven high by some control circuitry. |
| nSTATUS | Pull to $V_{CC}$ via a 10-k$\Omega$ resistor. When configuring multiple devices in the same JTAG chain, each nSTATUS pin should be pulled up to $V_{CC}$ individually. |
| CONF_DONE | Pull to $V_{CC}$ via a 10-k$\Omega$ resistor. When configuring multiple devices in the same JTAG chain, each CONF_DONE pin should be pulled up to $V_{CC}$ individually. CONF_DONE going high at the end of JTAG configuration indicates successful configuration. |
| DCLK | Should not be left floating. Drive low or high, whichever is more convenient on your board. |

JTAG-chain device programming is ideal when the system contains multiple devices, or when testing your system using JTAG BST circuitry. Figure 11–36 shows multi-device JTAG configuration.

*Figure 11–36. JTAG Configuration of Multiple Devices Using a Download Cable*
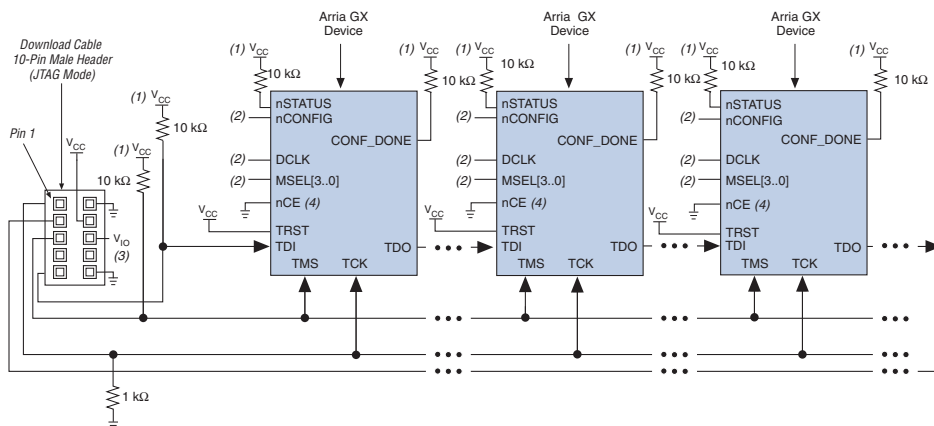


*Notes to Figure 11–36:*
(1) The pull-up resistor should be connected to the same supply voltage as the USB Blaster, MasterBlaster ($V_{IO}$ pin), ByteBlaster II, or ByteBlasterMV cable.
(2) The nCONFIG and MSEL[3..0] pins should be connected to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect nCONFIG to $V_{CC}$, and MSEL[3..0] to ground. Pull DCLK either high or low, whichever is convenient on your board.
(3) Pin 6 of the header is a $V_{IO}$ reference voltage for the MasterBlaster output driver. $V_{IO}$ should match the device's $V_{CCIO}$. Refer to the *MasterBlaster Serial/USB Communications Cable User Guide* for this value. In the ByteBlasterMV cable, this pin is a no connect. In the USB Blaster and ByteBlaster II cables, this pin is connected to nCE when it is used for active serial programming, otherwise it is a no connect.
(4) nCE must be connected to GND or driven low for successful JTAG configuration.

The nCE pin must be connected to GND or driven low during JTAG configuration. In multi-device FPP, AS, PS, and PPA configuration chains, the first device's nCE pin is connected to GND while its nCEO pin is connected to nCE of the next device in the chain. The last device's nCE input comes from the previous device, while its nCEO pin is left floating. In addition, the CONF_DONE and nSTATUS signals are all shared in multi-device FPP, AS, PS, or PPA configuration chains so the devices can enter user mode at the same time after configuration is complete. When the CONF_DONE and nSTATUS signals are shared among all the devices, every device must be configured when JTAG configuration is performed.

If you only use JTAG configuration, Altera recommends that you connect the circuitry as shown in Figure 11–36, where each of the CONF_DONE and nSTATUS signals are isolated, so that each device can enter user mode individually.

After the first device completes configuration in a multi-device configuration chain, its nCEO pin drives low to activate the second device's nCE pin, which prompts the second device to begin configuration. Therefore, if these devices are also in a JTAG chain, make sure the nCE pins are connected to GND during JTAG configuration or that the devices are JTAG configured in the same order as the configuration chain. As long as the devices are JTAG configured in the same order as the multi-device configuration chain, the nCEO of the previous device will drive nCE of the next device low when it has successfully been JTAG configured.

Other Altera devices that have JTAG support can be placed in the same JTAG chain for device programming and configuration.

☞    Stratix, Arria GX, Cyclone®, and Cyclone II devices must be within the first 17 devices in a JTAG chain. All of these devices have the same JTAG controller. If any of the Stratix, Arria GX, Cyclone, and Cyclone II devices are in the 18th or after they will fail configuration. This does not affect SignalTap® II.

For more information on configuring multiple Altera devices in the same configuration chain, refer to the *Configuring Mixed Altera FPGA Chains* chapter in volume 2 of the *Configuration Handbook.*

Figure 11–37 shows JTAG configuration of a Arria GX device with a microprocessor.

*Figure 11–37. JTAG Configuration of a Single Device Using a Microprocessor*
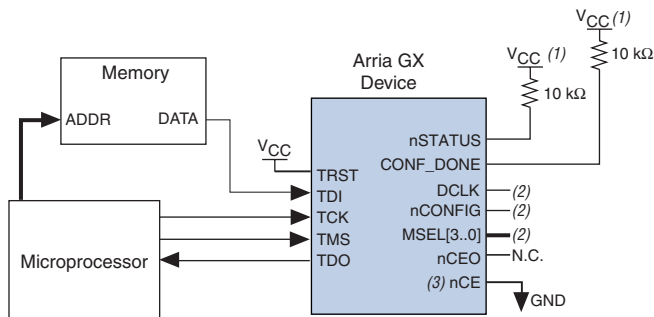


*Notes to Figure 11–37:*
(1) The pull-up resistor should be connected to a supply that provides an acceptable input signal for all devices in the chain. $V_{CC}$ should be high enough to meet the $V_{IH}$ specification of the I/O on the device.
(2) The nCONFIG, MSEL[3..0] pins should be connected to support a non-JTAG configuration scheme. If only JTAG configuration is used, connect nCONFIG to $V_{CC}$, and MSEL[3..0] to ground. Pull DCLK either high or low, whichever is convenient on your board.
(3) nCE must be connected to GND or driven low for successful JTAG configuration.

## Jam STAPL

Jam STAPL, JEDEC standard JESD-71, is a standard file format for in-system programmability (ISP) purposes. Jam STAPL supports programming or configuration of programmable devices and testing of electronic systems, using the IEEE 1149.1 JTAG interface. Jam STAPL is a freely licensed open standard.

The Jam Player provides an interface for manipulating the IEEE Std. 1149.1 JTAG TAP state machine.

For more information on JTAG and Jam STAPL in embedded environments, refer to *AN 122: Using Jam STAPL for ISP & ICR via an Embedded Processor*.

# Device Configuration Pins

Table 11–20 describes the connections and functionality of all the configuration related pins on Arria GX devices and summarizes the Arria GX pin configuration.

| Table 11–20. Arria GX Configuration Pin Summary  (Part 1 of 2)   *Note (1)* | | | | | |
|---|---|---|---|---|---|
| Bank | Description | Input/Output | Dedicated | Powered By | Configuration Mode |
| 3 | PGM[2..0] | Output | | *(2)* | PS, FPP, PPA, RU, LU |
| 3 | ASDO | Output | | *(2)* | AS |
| 3 | nCSO | Output | | *(2)* | AS |
| 3 | CRC_ERROR | Output | | *(2)* | Optional, all modes |
| 3 | DATA0 | Input | | *(3)* | All modes except JTAG |
| 3 | DATA[7..1] | Input | | *(3)* | FPP, PPA |
| 3 | DATA7 | Bidirectional | | *(2)*, *(3)* | PPA |
| 3 | RDYnBSY | Output | | *(2)* | PPA |
| 3 | INIT_DONE | Output | | Pull-up | Optional, all modes |
| 3 | nSTATUS | Bidirectional | Yes | Pull-up | All modes |
| 3 | nCE | Input | Yes | *(3)* | All modes |
| 3 | DCLK | Input | Yes | *(3)* | PS, FPP |
| | | Output | | *(2)* | AS |
| 3 | CONF_DONE | Bidirectional | Yes | Pull-up | All modes |
| 8 | TDI | Input | Yes | VCCPD | JTAG |
| 8 | TMS | Input | Yes | VCCPD | JTAG |
| 8 | TCK | Input | Yes | VCCPD | JTAG |
| 8 | TRST | Input | Yes | VCCPD | JTAG |
| 8 | nCONFIG | Input | Yes | *(3)* | All modes |
| 8 | VCCSEL | Input | Yes | VCCINT | All modes |
| 8 | CS | Input | | *(3)* | PPA |
| 8 | CLKUSR | Input | | *(3)* | Optional |
| 8 | nWS | Input | | *(3)* | PPA |
| 8 | nRS | Input | | *(3)* | PPA |
| 8 | RUnLU | Input | | *(3)* | PS, FPP, PPA, RU, LU |
| 8 | nCS | Input | | *(3)* | PPA |
| 7 | PORSEL | Input | Yes | VCCINT | All modes |
| 7 | nIO_PULLUP | Input | Yes | VCCINT | All modes |
| 7 | PLL_ENA | Input | Yes | *(3)* | Optional |

*Table 11–20. Arria GX Configuration Pin Summary  (Part 2 of 2)*     *Note (1)*

| Bank | Description | Input/Output | Dedicated | Powered By | Configuration Mode |
|------|-------------|--------------|-----------|------------|--------------------|
| 7 | nCEO | Output | Yes | *(2)*, *(4)* | All modes |
| 4 | MSEL[3..0] | Input | Yes | VCCINT | All modes |
| 4 | TDO | Output | Yes | *(2)*, *(4)* | JTAG |

*Notes to Table 11–20:*
(1)  Total number of pins is 41, total number of dedicated pins is 19.
(2)  All outputs are powered by VCCIO except as noted.
(3)  All inputs are powered by VCCIO or VCCPD, based on the VCCSEL setting, except as noted.
(4)  An external pull-up resistor may be required for this configuration pin because of the multivolt I/O interface. Refer to the *Arria GX Architecture* chapter in volume 1 of the *Arria GX Device Handbook* for pull-up or level shifter recommendations for nCEO and TDO.
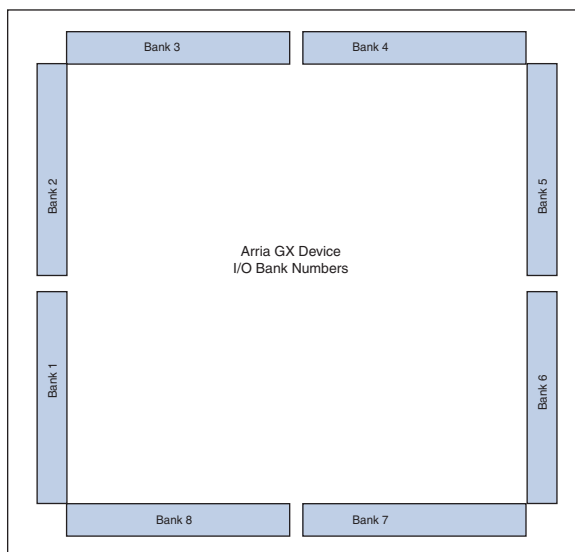
Figure 11–38 shows the I/O bank locations.

*Figure 11–38. Arria GX I/O Bank Numbers*

Table 11–21 describes the dedicated configuration pins, which are required to be connected properly on your board for successful configuration. Some of these pins may not be required for your configuration schemes.

| Table 11–21. Dedicated Configuration Pins on the Arria GX Device  (Part 1 of 10) | | | | |
|---|---|---|---|---|
| **Pin Name** | **User Mode** | **Configuration Scheme** | **Pin Type** | **Description** |
| V$_{CCPD}$ | N/A | All | Power | Dedicated power pin. This pin is used to power the I/O pre-drivers, the JTAG input pins, and the configuration input pins that are affected by the voltage level of VCCSEL.<br><br>This pin must be connected to 3.3-V. V$_{CCPD}$ must ramp-up from 0-V to 3.3-V within 100 ms. If V$_{CCPD}$ is not ramped up within this specified time, your Arria GX device will not configure successfully. If your system does not allow for a V$_{CCPD}$ ramp-up time of 100 ms or less, you must hold nCONFIG low until all power supplies are stable. |
| VCCSEL | N/A | All | Input | Dedicated input that selects which input buffer is used on the PLL_ENA pin and the configuration input pins; nCONFIG, DCLK (when used as an input), nSTATUS (when used as an input), CONF_DONE (when used as an input), DEV_OE, DEV_CLRn, DATA[7..0], RUnLU, nCE, nWS, nRS, CS, nCS, and CLKUSR. The 3.3-V/2.5-V input buffer is powered by V$_{CCPD}$, while the 1.8-V/1.5-V input buffer is powered by V$_{CCIO}$.<br><br>The VCCSEL input buffer has an internal 5-kΩ pull-down resistor that is always active. The VCCSEL input buffer is powered by V$_{CCINT}$ and must be hardwired to V$_{CCPD}$ or ground. A logic high selects the 1.8-V/1.5-V input buffer, and a logic low selects the 3.3-V/2.5-V input buffer. For more information, refer to the "VCCSEL Pin" section. |

| Table 11–21. Dedicated Configuration Pins on the Arria GX Device  (Part 2 of 10) | | | | |
|---|---|---|---|---|
| **Pin Name** | **User Mode** | **Configuration Scheme** | **Pin Type** | **Description** |
| PORSEL | N/A | All | Input | Dedicated input which selects between a POR time of 12 ms or 100 ms. A logic high (1.5 V, 1.8 V, 2.5 V, 3.3 V) selects a POR time of about 12 ms and a logic low selects POR time of about 100 ms.<br><br>The PORSEL input buffer is powered by $V_{CCINT}$ and has an internal 5-kΩ pull-down resistor that is always active. The PORSEL pin should be tied directly to $V_{CCPD}$ or GND. |
| nIO_PULLUP | N/A | All | Input | Dedicated input that chooses whether the internal pull-up resistors on the user I/O pins and dual-purpose I/O pins (nCSO, nASDO, DATA[7..0], nWS, nRS, RDYnBSY, nCS, CS, RUnLU, PGM[], CLKUSR, INIT_DONE, DEV_OE, DEV_CLR) are on or off before and during configuration. A logic high (1.5 V, 1.8 V, 2.5 V, 3.3 V) turns off the weak internal pull-up resistors, while a logic low turns them on.<br><br>The nIO-PULLUP input buffer is powered by $V_{CCPD}$ and has an internal 5-kΩ pull-down resistor that is always active. The nIO-PULLUP can be tied directly to $V_{CCPD}$ or use a 1-kΩ pull-up resistor or tied directly to GND. |
| MSEL[3..0] | N/A | All | Input | 4-bit configuration input that sets the Arria GX device configuration scheme. Refer to Table 11–1 for the appropriate connections.<br><br>These pins must be hard-wired to $V_{CCPD}$ or GND.<br><br>The MSEL[3..0] pins have internal 5-kΩ pull-down resistors that are always active. |

| Table 11–21. Dedicated Configuration Pins on the Arria GX Device (Part 3 of 10) | | | | |
|---|---|---|---|---|
| **Pin Name** | **User Mode** | **Configuration Scheme** | **Pin Type** | **Description** |
| nCONFIG | N/A | All | Input | Configuration control input. Pulling this pin low during user-mode will cause the device to lose its configuration data, enter a reset state, tri-state all I/O pins. Returning this pin to a logic high level will initiate a reconfiguration.<br><br>If your configuration scheme uses an enhanced configuration device or EPC2 device, nCONFIG can be tied directly to $V_{CC}$ or to the configuration device's nINIT_CONF pin. |
| nSTATUS | N/A | All | Bidirectional open-drain | The device drives nSTATUS low immediately after power-up and releases it after the POR time.<br><br>Status output. If an error occurs during configuration, nSTATUS is pulled low by the target device.<br><br>Status input. If an external source drives the nSTATUS pin low during configuration or initialization, the target device enters an error state.<br><br>Driving nSTATUS low after configuration and initialization does not affect the configured device. If a configuration device is used, driving nSTATUS low will cause the configuration device to attempt to configure the device, but because the device ignores transitions on nSTATUS in user-mode, the device does not reconfigure. To initiate a reconfiguration, nCONFIG must be pulled low.<br><br>The enhanced configuration devices' and EPC2 devices' OE and nCS pins have optional internal programmable pull-up resistors. If internal pull-up resistors on the enhanced configuration device are used, external 10-kΩ pull-up resistors should not be used on these pins. The external 10-kΩ pull-up resistor should be used only when the internal pull-up resistor is not used. |

| Table 11–21. Dedicated Configuration Pins on the Arria GX Device  (Part 4 of 10) | | | | |
|---|---|---|---|---|
| Pin Name | User Mode | Configuration Scheme | Pin Type | Description |
| nSTATUS (continued) | | | | If VCCPD and VCCIO are not fully powered up, the following could occur:<br>● VCCPD and VCCIO are powered high enough for the nSTATUS buffer to function properly, and nSTATUS is driven low. When VCCPD and VCCIO are ramped up, POR trips, and nSTATUS is released after POR expires.<br>● VCCPD and VCCIO are not powered high enough for the nSTATUS buffer to function properly. In this situation, nSTATUS might appear logic high, triggering a configuration attempt that would fail because POR did not yet trip. When VCCPD and VCCIO are powered up, nSTATUS is pulled low because POR did not yet trip. When POR trips after VCCPD and VCCIO are powered up, nSTATUS is released and pulled high. At that point, reconfiguration is triggered and the device is configured. |

| Table 11–21. Dedicated Configuration Pins on the Arria GX Device  (Part 5 of 10) | | | | |
|---|---|---|---|---|
| **Pin Name** | **User Mode** | **Configuration Scheme** | **Pin Type** | **Description** |
| CONF_DONE | N/A | All | Bidirectional open-drain | Status output. The target device drives the CONF_DONE pin low before and during configuration. Once all configuration data is received without error and the initialization cycle starts, the target device releases CONF_DONE.<br><br>Status input. After all data is received and CONF_DONE goes high, the target device initializes and enters user mode. The CONF_DONE pin must have an external 10-k$\Omega$ pull-up resistor in order for the device to initialize.<br><br>Driving CONF_DONE low after configuration and initialization does not affect the configured device.<br><br>The enhanced configuration devices' and EPC2 devices' OE and nCS pins have optional internal programmable pull-up resistors. If internal pull-up resistors on the enhanced configuration device are used, external 10-k$\Omega$ pull-up resistors should not be used on these pins. The external 10-k$\Omega$ pull-up resistor should be used only when the internal pull-up resistor is not used. |
| nCE | N/A | All | Input | Active-low chip enable. The nCE pin activates the device with a low signal to allow configuration. The nCE pin must be held low during configuration, initialization, and user mode. In single device configuration, it should be tied low. In multi-device configuration, nCE of the first device is tied low while its nCEO pin is connected to nCE of the next device in the chain.<br><br>The nCE pin must also be held low for successful JTAG programming of the device. |

| Table 11–21. Dedicated Configuration Pins on the Arria GX Device  (Part 6 of 10) | | | | |
|---|---|---|---|---|
| **Pin Name** | **User Mode** | **Configuration Scheme** | **Pin Type** | **Description** |
| nCEO | N/A | All | Output | Output that drives low when device configuration is complete. In single device configuration, this pin is left floating. In multi-device configuration, this pin feeds the next device's nCE pin. The nCEO of the last device in the chain is left floating. The nCEO pin is powered by $V_{CCIO}$ in I/O bank 7. For recommendations on how to connect nCEO in a chain with multiple voltages across the devices in the chain, refer to the *Arria GX Architecture* chapter in volume 1 of the *Arria GX Device Handbook*. |
| ASDO | N/A in AS mode I/O in non-AS mode | AS | Output | Control signal from the Arria GX device to the serial configuration device in AS mode used to read out configuration data. In AS mode, ASDO has an internal pull-up resistor that is always active. |
| nCSO | N/A in AS mode I/O in non-AS mode | AS | Output | Output control signal from the Arria GX device to the serial configuration device in AS mode that enables the configuration device. In AS mode, nCSO has an internal pull-up resistor that is always active. |

*Table 11–21. Dedicated Configuration Pins on the Arria GX Device  (Part 7 of 10)*

| Pin Name | User Mode | Configuration Scheme | Pin Type | Description |
|---|---|---|---|---|
| DCLK | N/A | Synchronous configuration schemes (PS, FPP, AS) | Input (PS, FPP) Output (AS) | In PS and FPP configuration, DCLK is the clock input used to clock data from an external source into the target device. Data is latched into the device on the rising edge of DCLK.<br><br>In AS mode, DCLK is an output from the Arria GX device that provides timing for the configuration interface. In AS mode, DCLK has an internal pull-up resistor (typically 25 kΩ) that is always active.<br><br>In PPA mode, DCLK should be tied high to $V_{CC}$ to prevent this pin from floating.<br><br>After configuration, this pin is tri-stated. In schemes that use a configuration device, DCLK will be driven low after configuration is done. In schemes that use a control host, DCLK should be driven either high or low, whichever is more convenient. Toggling this pin after configuration does not affect the configured device. |
| DATA0 | I/O | PS, FPP, PPA, AS | Input | Data input. In serial configuration modes, bit-wide configuration data is presented to the target device on the DATA0 pin.<br><br>The $V_{IH}$ and $V_{IL}$ levels for this pin are dependent on the input buffer selected by the VCCSEL pin. Refer to the section "VCCSEL Pin" on page 11–9 for more information.<br><br>In AS mode, DATA0 has an internal pull-up resistor that is always active.<br><br>After configuration, DATA0 is available as a user I/O pin and the state of this pin depends on the **Dual-Purpose Pin** settings.<br><br>After configuration, EPC1 and EPC1441 devices tri-state this pin, while enhanced configuration and EPC2 devices drive this pin high. |

| *Table 11–21. Dedicated Configuration Pins on the Arria GX Device  (Part 8 of 10)* | | | | |
|---|---|---|---|---|
| **Pin Name** | **User Mode** | **Configuration Scheme** | **Pin Type** | **Description** |
| DATA[7..1] | I/O | Parallel configuration schemes (FPP and PPA) | Inputs | Data inputs. Byte-wide configuration data is presented to the target device on DATA[7..0].<br><br>The $V_{IH}$ and $V_{IL}$ levels for this pin are dependent on the input buffer selected by the VCCSEL pin. Refer to the section "VCCSEL Pin" on page 11–9 for more information.<br><br>In serial configuration schemes, they function as user I/O pins during configuration, which means they are tri-stated.<br><br>After PPA or FPP configuration, DATA[7..1] are available as user I/O pins and the state of these pin depends on the **Dual-Purpose Pin** settings. |
| DATA7 | I/O | PPA | Bidirectional | In the PPA configuration scheme, the DATA7 pin presents the RDYnBSY signal after the nRS signal has been strobed low.<br><br>The $V_{IH}$ and $V_{IL}$ levels for this pin are dependent on the input buffer selected by the VCCSEL pin. Refer to the section "VCCSEL Pin" on page 11–9 for more information.<br><br>In serial configuration schemes, it functions as a user I/O pin during configuration, which means it is tri-stated.<br><br>After PPA configuration, DATA7 is available as a user I/O and the state of this pin depends on the **Dual-Purpose Pin** settings. |
| nWS | I/O | PPA | Input | Write strobe input. A low-to-high transition causes the device to latch a byte of data on the DATA[7..0]  pins.<br><br>In non-PPA schemes, it functions as a user I/O pin during configuration, which means it is tri-stated.<br><br>After PPA configuration, nWS is available as a user I/O pins and the state of this pin depends on the **Dual-Purpose Pin** settings. |

| *Table 11–21. Dedicated Configuration Pins on the Arria GX Device  (Part 9 of 10)* | | | | |
|---|---|---|---|---|
| **Pin Name** | **User Mode** | **Configuration Scheme** | **Pin Type** | **Description** |
| nRS | I/O | PPA | Input | Read strobe input. A low input directs the device to drive the RDYnBSY signal on the DATA7 pin.<br><br>If the nRS pin is not used in PPA mode, it should be tied high. In non-PPA schemes, it functions as a user I/O during configuration, which means it is tri-stated.<br><br>After PPA configuration, nRS is available as a user I/O pin and the state of this pin depends on the **Dual-Purpose Pin** settings. |
| RDYnBSY | I/O | PPA | Output | Ready output. A high output indicates that the target device is ready to accept another data byte. A low output indicates that the target device is busy and not ready to receive another data byte.<br><br>In PPA configuration schemes, this pin will drive out high after power-up, before configuration and after configuration before entering user-mode. In non-PPA schemes, it functions as a user I/O pin during configuration, which means it is tri-stated.<br><br>After PPA configuration, RDYnBSY  is available as a user I/O pin and the state of this pin depends on the **Dual-Purpose Pin** settings. |

*Table 11–21. Dedicated Configuration Pins on the Arria GX Device  (Part 10 of 10)*

| Pin Name | User Mode | Configuration Scheme | Pin Type | Description |
|---|---|---|---|---|
| nCS/CS | I/O | PPA | Input | Chip-select inputs. A low on nCS and a high on CS select the target device for configuration. The nCS and CS pins must be held active during configuration and initialization.<br><br>During the PPA configuration mode, it is only required to use either the nCS or CS pin. Therefore, if only one chip-select input is used, the other must be tied to the active state. For example, nCS can be tied to ground while CS is toggled to control configuration.<br><br>In non-PPA schemes, it functions as a user I/O pin during configuration, which means it is tri-stated.<br><br>After PPA configuration, nCS and CS are available as user I/O pins and the state of these pins depends on the **Dual-Purpose Pin** settings. |
| RUnLU | N/A if using Remote System Upgrade I/O if not | Remote System Upgrade in FPP, PS, or PPA | Input | Input that selects between remote update and local update. A logic high (1.5-V, 1.8-V, 2.5-V, 3.3-V) selects remote update and a logic low selects local update.<br><br>When not using remote update or local update configuration modes, this pin is available as general-purpose user I/O pin.<br><br>When using remote system upgrade in AS mode, set the RUnLU pin to high because AS does not support local update |
| PGM[2..0] | N/A if using Remote System Upgrade I/O if not using | Remote System Upgrade in FPP, PS, or PPA | Output | These output pins select one of eight pages in the memory (either flash or enhanced configuration device) when using a remote system upgrade mode.<br><br>When not using remote update or local update configuration modes, these pins are available as general-purpose user I/O pins. |

Table 11–22 describes the optional configuration pins. If these optional configuration pins are not enabled in the Quartus II software, they are available as general-purpose user I/O pins. Therefore, during configuration, these pins function as user I/O pins and are tri-stated with weak pull-up resistors.

*Table 11–22. Optional Configuration Pins*

| Pin Name | User Mode | Pin Type | Description |
|----------|-----------|----------|-------------|
| CLKUSR | N/A if option is on. I/O if option is off. | Input | Optional user-supplied clock input synchronizes the initialization of one or more devices. This pin is enabled by turning on the **Enable user-supplied start-up clock** (**CLKUSR**) option in the Quartus II software. |
| INIT_DONE | N/A if option is on. I/O if option is off. | Output open-drain | Status pin can be used to indicate when the device has initialized and is in user mode. When nCONFIG is low and during the beginning of configuration, the INIT_DONE pin is tri-stated and pulled high due to an external 10-kΩ pull-up resistor. Once the option bit to enable INIT_DONE is programmed into the device (during the first frame of configuration data), the INIT_DONE pin will go low. When initialization is complete, the INIT_DONE pin will be released and pulled high and the device enters user mode. Thus, the monitoring circuitry must be able to detect a low-to-high transition. This pin is enabled by turning on the **Enable INIT_DONE output** option in the Quartus II software. |
| DEV_OE | N/A if option is on. I/O if option is off. | Input | Optional pin that allows the user to override all tri-states on the device. When this pin is driven low, all I/O pins are tri-stated; when this pin is driven high, all I/O pins behave as programmed. This pin is enabled by turning on the **Enable device-wide output enable** (**DEV_OE**) option in the Quartus II software. |
| DEV_CLRn | N/A if option is on. I/O if option is off. | Input | Optional pin that allows you to override all clears on all device registers. When this pin is driven low, all registers are cleared; when this pin is driven high, all registers behave as programmed. This pin is enabled by turning on the **Enable device-wide reset** (**DEV_CLRn**) option in the Quartus II software. |

Table 11–23 describes the dedicated JTAG pins. JTAG pins must be kept stable before and during configuration to prevent accidental loading of JTAG instructions. The TDI, TMS, and TRST have weak internal pull-up resistors (typically 25 kΩ) while TCK has a weak internal pull-down resistor. If you plan to use the SignalTap embedded logic array analyzer, you need to connect the JTAG pins of the Arria GX device to a JTAG header on your board.

*Table 11–23. Dedicated JTAG Pins*

| Pin Name | User Mode | Pin Type | Description |
|---|---|---|---|
| TDI | N/A | Input | Serial input pin for instructions as well as test and programming data. Data is shifted in on the rising edge of TCK. The TDI pin is powered by the 3.3-V $V_{CCPD}$ supply.<br><br>If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to $V_{CC}$. |
| TDO | N/A | Output | Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. The TDO pin is powered by $V_{CCIO}$ in I/O bank 4. For recommendations on connecting a JTAG chain with multiple voltages across the devices in the chain, refer to the chapter *IEEE 1149.1 (JTAG) Boundary Scan Testing in Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*.<br><br>If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by leaving this pin unconnected. |
| TMS | N/A | Input | Input pin that provides the control signal to determine the transitions of the TAP controller state machine. Transitions within the state machine occur on the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. The TMS pin is powered by the 3.3-V $V_{CCPD}$ supply.<br><br>If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting this pin to $V_{CC}$. |
| TCK | N/A | Input | The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. The TCK pin is powered by the 3.3-V $V_{CCPD}$ supply.<br><br>If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting TCK to GND. |
| TRST | N/A | Input | Active-low input to asynchronously reset the boundary-scan circuit. The TRST pin is optional according to IEEE Std. 1149.1. The TRST pin is powered by the 3.3-V $V_{CCPD}$ supply.<br><br>If the JTAG interface is not required on the board, the JTAG circuitry can be disabled by connecting the TRST pin to GND. |

## Conclusion

Arria GX devices can be configured in a number of different schemes to fit your system's need. In addition, configuration data decompression and remote system upgrade support supplement the Arria GX configuration solution.

## Referenced Documents

This chapter references the following documents:

- *Altera Enhanced Configuration Devices* chapter in volume 2 of the *Configuration Handbook*
- *AN 122: Using Jam STAPL for ISP & ICR via an Embedded Processor*
- *AN 418: SRunner: An Embedded Solution for Serial Configuration*
- *Arria GX Architecture* chapter in volume 1 of the *Arria GX Device Handbook*
- *Arria GX Device Handbook*
- *ByteBlaster II Download Cable User Guide*
- *ByteBlasterMV Download Cable User Guide*
- *Configuration Devices for SRAM-Based LUT Devices Data Sheet* chapter in volume 2 of the *Configuration Handbook*
- *Configuring Mixed Altera FPGA Chains* chapter in volume 2 of the *Configuration Handbook*
- *DC & Switching Characteristics* chapter in volume 1 of the *Arria GX Device Handbook*
- *Enhanced Configuration Devices (EPC4, EPC8 & EPC16) Data Sheet* chapter in volume 2 of the *Configuration Handbook*
- *IEEE 1149.1 (JTAG) Boundary-Scan Testing for Arria GX Devices*
- Jam Programming Support - JTAG Technologies
- *MasterBlaster Serial/USB Communications Cable User Guide*
- *Remote System Upgrades With Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*
- *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* chapter in volume 2 of the *Configuration Handbook*
- *Software Settings* section in volume 2 of the *Configuration Handbook*
- *USB-Blaster USB Port Download Cable User Guide*

# Document Revision History

Table 11–24 shows the revision history for this chapter.

| Table 11–24. Document Revision History | | |
|---|---|---|
| **Date and Document Version** | **Changes Made** | **Summary of Changes** |
| May 2008 v1.3 | Updated:<br>● Table 11–2<br>● Figure 11–6<br>● Figure 11–7 | — |
| | Minor text edits. | — |
| August 2007 v1.2 | Added the "Referenced Documents" section. | — |
| | Minor text edits. | — |
| June 2007 v1.1 | Updated $t_{CF2CK}$ in Figures 11–6 and 11–7. | — |
| May 2007 v1.0 | Initial Release | — |

# 12. Remote System Upgrades with Arria GX Devices

## Introduction

System designers today face difficult challenges such as shortened design cycles, evolving standards, and system deployments in remote locations. Arria™ GX FPGAs help overcome these challenges with their inherent re-programmability and dedicated circuitry to perform remote system upgrades. Remote system upgrades help deliver feature enhancements and bug fixes without costly recalls, reduce time-to-market, and extend product life.

Arria GX FPGAs feature dedicated remote system upgrade circuitry. Soft logic (either the Nios® embedded processor or user logic) implemented in an Arria GX device can download a new configuration image from a remote location, store it in configuration memory, and direct the dedicated remote system upgrade circuitry to initiate a reconfiguration cycle. The dedicated circuitry performs error detection during and after the configuration process, recovers from any error condition by reverting back to a safe configuration image, and provides error status information. This dedicated remote system upgrade circuitry is unique to Stratix®, Stratix II, Stratix II GX, and Arria GX FPGAs and helps to avoid system downtime.

Remote system upgrade is supported in all Arria GX configuration schemes: fast passive parallel (FPP), active serial (AS), passive serial (PS), and passive parallel asynchronous (PPA). Remote system upgrade can also be implemented in conjunction with advanced Arria GX features such as real-time decompression of configuration data for efficient field upgrades.

This chapter describes the functionality and implementation of the dedicated remote system upgrade circuitry. It also defines several concepts related to remote system upgrade, including factory configuration, application configuration, remote update mode, local update mode, the user watchdog timer, and page mode operation. Additionally, this chapter provides design guidelines for implementing remote system upgrade with the various supported configuration schemes.

This chapter contains the following sections:

# Functional Description

The dedicated remote system upgrade circuitry in Arria GX FPGAs manages remote configuration and provides error detection, recovery, and status information. User logic or a Nios processor implemented in the FPGA logic array provides access to the remote configuration data source and an interface to the system's configuration memory.

Arria GX FPGA's remote system upgrade process involves the following steps:

1. A Nios processor (or user logic) implemented in the FPGA logic array receives new configuration data from a remote location. The connection to the remote source is a communication protocol such as the transmission control protocol/Internet protocol (TCP/IP), peripheral component interconnect (PCI), user datagram protocol (UDP), universal asynchronous receiver/transmitter (UART), or a proprietary interface.

2. The Nios processor (or user logic) stores this new configuration data in non-volatile configuration memory. The non-volatile configuration memory can be any standard flash memory used in conjunction with an intelligent host (for example, a MAX® device or microprocessor), the serial configuration device, or the enhanced configuration device.

3. The Nios processor (or user logic) initiates a reconfiguration cycle with the new or updated configuration data.

4. The dedicated remote system upgrade circuitry detects and recovers from any error(s) that might occur during or after the reconfiguration cycle, and provides error status information to the user design.

Figure 12–1 shows the steps required for performing remote configuration updates. (The numbers in the figure below coincide with the steps above.)

*Figure 12–1. Functional Diagram of Aria GX Remote System Upgrade*
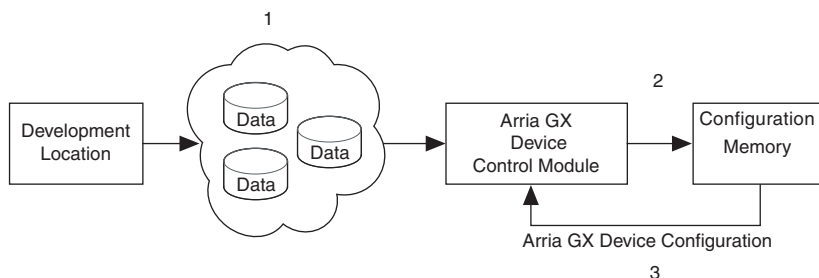


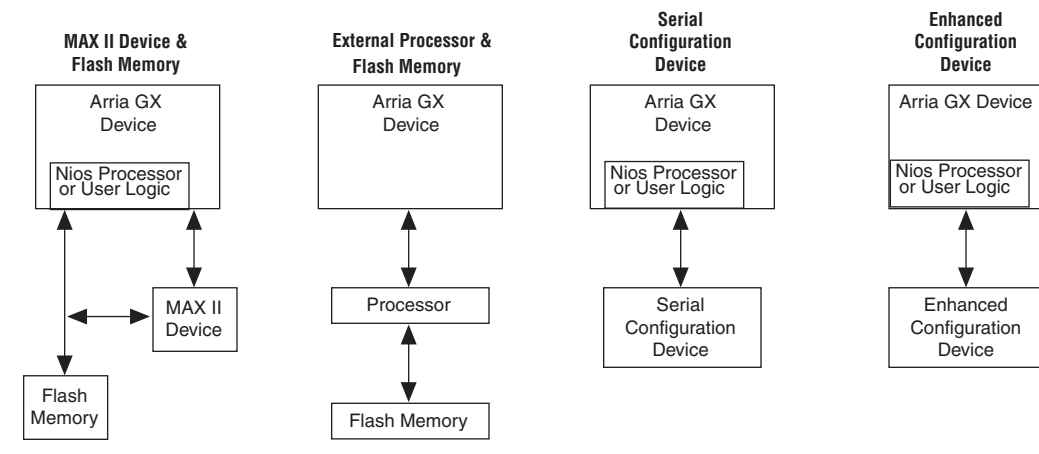Arria GX FPGAs support remote system upgrade in the FPP, AS, PS, and PPA configuration schemes.

- Serial configuration devices use the AS scheme to configure Arria GX FPGAs.
- A MAX II device (or microprocessor and flash configuration schemes) uses FPP, PS, or PPA schemes to configure Arria GX FPGAs.
- Enhanced configuration devices use the FPP or PS configuration schemes to configure Arria GX FPGAs.

☞ The JTAG-based configuration scheme does not support remote system upgrade.

Figure 12–2 shows the block diagrams for implementing remote system upgrade with the various Arria GX configuration schemes.

*Figure 12–2. Remote System Upgrade Block Diagrams for Various Arria GX Configuration Schemes*



**MAX II Device &
Flash Memory**

**External Processor &
Flash Memory**

**Serial
Configuration
Device**

**Enhanced
Configuration
Device**

Arria GX
Device

Arria GX
Device

Arria GX
Device

Arria GX Device

Nios Processor
or User Logic

Nios Processor
or User Logic

Nios Processor
or User Logic

MAX II
Device

Processor

Serial
Configuration
Device

Enhanced
Configuration
Device

Flash
Memory

Flash Memory

☞ For active serial configuration scheme, remote system upgrade only supports single device configuration.

You must set the mode select pins (MSEL[3..0]) and the RUnLU pin to select the configuration scheme and remote system upgrade mode best suited for your system. Table 12–1 lists the pin settings for Arria GX FPGAs. Standard configuration mode refers to normal FPGA configuration mode with no support for remote system upgrades, and the remote system upgrade circuitry is disabled. The following sections describe the local update and remote update remote system upgrade modes.

For more information on standard configuration schemes supported in Arria GX FPGAs, see the *Configuring Arria GX Devices* chapter of the *Arria GX Handbook*.

*Table 12–1. Arria GX Remote System Upgrade Modes  (Part 1 of 2)*

| Configuration Scheme | MSEL[3..0] | RUnLU | Remote System Upgrade Mode |
|---|---|---|---|
| FPP | 0000 | — | Standard |
| | 0100 *(1)* | 0 | Local update |
| | 0100 *(1)* | 1 | Remote update |

*Table 12–1. Arria GX Remote System Upgrade Modes  (Part 2 of 2)*

| Configuration Scheme | MSEL[3..0] | RUnLU | Remote System Upgrade Mode |
|---|---|---|---|
| FPP with decompression feature enabled *(2)* | 1011 | — | Standard |
| | 1100 *(1)* | 0 | Local update |
| | 1100 *(1)* | 1 | Remote update |
| Fast AS (40 MHz) *(3)* | 1000 | — | Standard |
| | 1001 | 1 | Remote update |
| AS (20 MHz) *(3)* | 1101 | — | Standard |
| | 1110 | 1 | Remote update |
| PS | 0010 | — | Standard |
| | 0110 *(1)* | 0 | Local update |
| | 0110 *(1)* | 1 | Remote update |
| PPA | 0001 | — | Standard |
| | 0101 *(1)* | 0 | Local update |
| | 0101 *(1)* | 1 | Remote update |

*Notes to Table 12–1:*

(1)    These schemes require that you drive the RUnLU pin to specify either remote update or local update mode. AS schemes only support the remote update mode.

(2)    These modes are only supported when using a MAX II device or microprocessor and flash for configuration. In these modes, the host system must output a DCLK that is 4x the data rate.

(3)    The EPCS16 and EPCS64 serial configuration devices support a DCLK up to 40 MHz; other EPCS devices support a DCLK up to 20 MHz. See the *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* in volume 2 of the *Configuration Handbook* for more information.

## Configuration Image Types & Pages

When using remote system upgrade, FPGA configuration bitstreams are classified as factory configuration images or application configuration images. An image, also referred to as a configuration, is a design loaded into the FPGA that performs certain user-defined functions. Each FPGA in your system requires one factory image and one or more application images. The factory image is a user-defined fall-back, or safe, configuration and is responsible for administering remote updates in conjunction with the dedicated circuitry. Application images implement user-defined functionality in the target FPGA.

A remote system update involves storing a new application configuration image or updating an existing one via the remote communication interface. After an application configuration image is stored or updated remotely, the user design in the FPGA initiates a reconfiguration cycle with the new image. Any errors during or after this cycle are detected by

the dedicated remote system upgrade circuitry and cause the FPGA to automatically revert to the factory image. The factory image then performs error processing and recovery. While error processing functionality is limited to the factory configuration, both factory and application configurations can download and store remote updates and initiate system reconfiguration.

Arria GX FPGAs select between the different configuration images stored in the system configuration memory using the page address pins or start address registers. A page is a section of the configuration memory space that contains one configuration image for each FPGA in the system. One page stores one system configuration, regardless of the number of FPGAs in the system.

Page address pins select the configuration image within an enhanced configuration device or flash memory (MAX II device or microprocessor setup). Page start address registers are used when Arria GX FPGAs are configured in AS mode with serial configuration devices. Figure 12–3 illustrates page mode operation in Arria GX FPGAs.

*Figure 12–3. Page Mode Operation in Arria GX FPGAs*
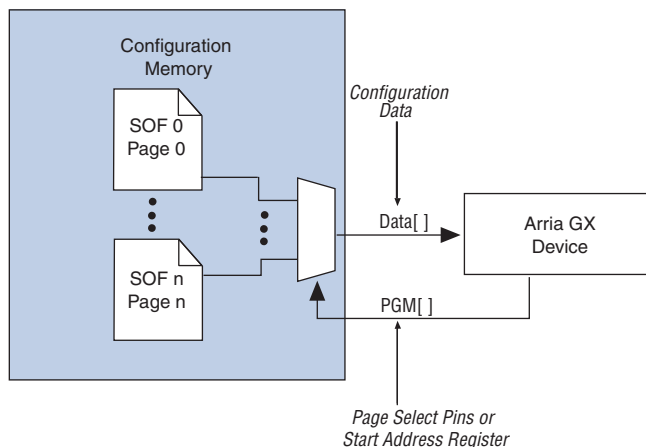


Arria GX devices drive out three page address pins, PGM[2..0], to the MAX II device or microprocessor or enhanced configuration device. These page pins select between eight configuration pages. Page zero (PGM[2..0] = 000) must contain the factory configuration, and the other seven pages are application configurations. The PGM[] pins are pointers to the start address and length of each page, and the MAX II device, microprocessor, and enhanced configuration devices perform this translation.

☞ When implementing remote system upgrade with an intelligent-host-based configuration, your MAX II device or microprocessor should emulate the page mode feature supported by the enhanced configuration device, which translates PGM pointers to a memory address in the configuration memory. Your MAX II device or microprocessor must provide a similar translation feature.

For more information about the enhanced configuration device page mode feature, refer to the Dynamic Configuration (Page Mode) Implementation section of the *Enhanced Configuration Devices (EPC4, EPC8, & EPC16) Data Sheet* chapter in volume 2 of the *Configuration Handbook*.

When implementing remote system upgrade with AS configuration, a dedicated 7-bit page start address register inside Arria GX device determines the start addresses for configuration pages within the serial configuration device. The PGM[6..0] registers form bits [22..16] of the 24-bit start address while the other 17 bits are set to zero: StAdd[23..0] = {1'b0, PGM[6..0], 16'b0}. During AS configuration, Arria GX devices use this 24-bit page start address to obtain configuration data from the serial configuration devices.

# Remote System Upgrade Modes

Remote system upgrade has two modes of operation: remote update mode and local update mode. The remote and local update modes allow you to determine the functionality of your system upon power up and offer different features. The RUnLU input pin selects between the remote update (logic high) and local update (logic low) modes.

## Overview

In remote update mode, Arria GX devices load the factory configuration image upon power up. The user-defined factory configuration should determine which application configuration is to be loaded and trigger a reconfiguration cycle. Remote update mode allows up to eight configuration images (one factory plus seven application images) when used with the MAX II device or microprocessor and flash-based configuration or an enhanced configuration device.

When used with serial configuration devices, the remote update mode allows an application configuration to start at any flash sector boundary. This translates to a maximum of 128 pages in the EPCS64 and 32 pages in the EPCS16 device, where the minimum size of each page is 512 KBits. Additionally, the remote update mode features a user watchdog timer that can detect functional errors in an application configuration.

Local update mode is a simplified version of the remote update mode. In this mode, Arria GX FPGAs directly load the application configuration, bypassing the factory configuration. This mode is useful if your system is required to boot into user mode with minimal startup time. It is also useful during system prototyping, as it allows you to verify functionality of the application configuration.

In local update mode, a maximum of two configuration images or pages is supported: one factory configuration, located at page address PGM[2..0] = 000, and one application configuration, located at page address PGM[2..0] = 001. Because the page address of the application configuration is fixed, the local update mode does not require the factory configuration image to determine which application is to be loaded. If any errors are encountered while loading the application configuration, Arria GX FPGAs revert to the factory configuration. The user watchdog timer feature is not supported in this mode.

☞ Also, local update mode does not support AS configuration with the serial configuration devices because these devices don't support a dynamic pointer to page 001 start address location.

Table 12–2 details the differences between remote and local update modes.

*Table 12–2. Differences Between Remote & Local Update Modes  (Part 1 of 2)*

| Features | Remote Update Mode | Local Update Mode |
|---|---|---|
| RUnLU input pin setting | 1 | 0 |
| Page selection upon power up | PGM[2..0]  = 000 (Factory) | PGM[2..0]  = 001 (Application) |
| Supported configurations | MAX II device or microprocessor-based configuration, serial configuration, and enhanced configuration devices (FPP, PS, AS, PPA) | MAX II device or microprocessor-based configuration and enhanced configuration devices (FPP, PS, PPA) |
| Number of pages supported | Eight pages for external host or controller based configuration; up to 128 pages (512 KBits/page) for serial configuration device | Two pages |

*Table 12–2. Differences Between Remote & Local Update Modes  (Part 2 of 2)*

| Features | Remote Update Mode | Local Update Mode |
|---|---|---|
| User watchdog timer | Available | Disabled |
| Remote system upgrade control and status register | Read/write access allowed in factory configuration. Read access in application configuration | Only status register read access allowed in local update mode (factory and application configurations). Write access to control register is disabled |

## Remote Update Mode

When Arria GX FPGAs are first powered up in remote update mode, it loads the factory configuration located at page zero (page address pins PGM[2..0] = "000"; page registers PGM[6..0] = "0000000"). You should always store the factory configuration image for your system at page address zero. A factory configuration image is a bitstream for the FPGA(s) in your system that is programmed during production and is the fall-back image when errors occur. This image is stored in non-volatile memory and is never updated or modified using remote access. This corresponds to PGM[2..0] = 000 of the enhanced configuration device or standard flash memory, and start address location 0x000000 in the serial configuration device.

The factory image is user designed and contains soft logic to:

■ Process any errors based on status information from the dedicated remote system upgrade circuitry
■ Communicate with the remote host and receive new application configurations, and store this new configuration data in the local non-volatile memory device
■ Determine which application configuration is to be loaded into the FPGA
■ Enable or disable the user watchdog timer and load its time-out value (optional)
■ Instruct the dedicated remote system upgrade circuitry to initiate a reconfiguration cycle

Figure 12–4 shows the transitions between the factory and application configurations in remote update mode.

*Figure 12–4. Transitions Between Configurations in Remote Update Mode*
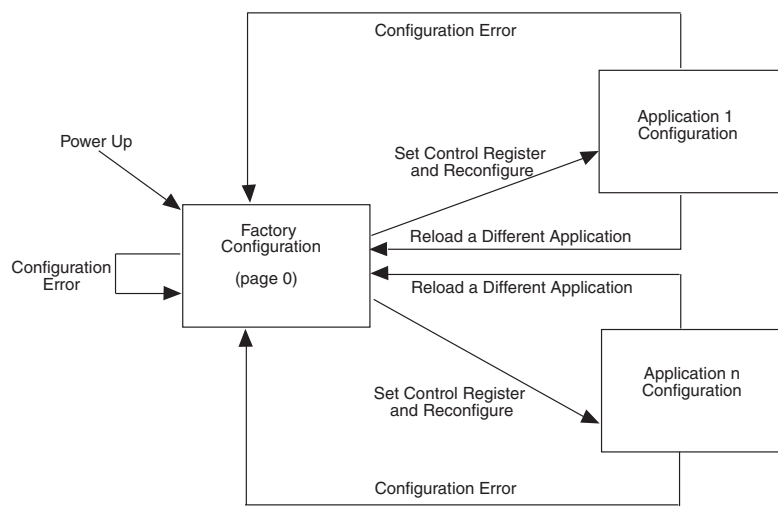


After power up or a configuration error, the factory configuration logic should write the remote system upgrade control register to specify the page address of the application configuration to be loaded. The factory configuration should also specify whether or not to enable the user watchdog timer for the application configuration and, if enabled, specify the timer setting.

The user watchdog timer ensures that the application configuration is valid and functional. After confirming the system is healthy, the user-designed application configuration should reset the timer periodically during user-mode operation of an application configuration. This timer reset logic should be a user-designed hardware and/or software health monitoring signal that indicates error-free system operation. If the user application configuration detects a functional problem or if the system hangs, the timer is not reset in time and the dedicated circuitry updates the remote system upgrade status register, triggering the device to load the factory configuration. The user watchdog timer is automatically disabled for factory configurations.

☞ Only valid application configurations designed for remote update mode include the logic to reset the timer in user mode.

For more information about the user watchdog timer, see "User Watchdog Timer" on page 12–19.

If there is an error while loading the application configuration, the remote system upgrade status register is written by the Arria GX FPGA's dedicated remote system upgrade circuitry, specifying the cause of the reconfiguration. Actions that cause the remote system upgrade status register to be written are:

■ nSTATUS driven low externally
■ Internal CRC error
■ User watchdog timer time out
■ A configuration reset (logic array nCONFIG signal or external nCONFIG pin assertion)

Arria GX FPGAs automatically load the factory configuration located at page address zero. This user-designed factory configuration should read the remote system upgrade status register to determine the reason for reconfiguration. The factory configuration should then take appropriate error recovery steps and write to the remote system upgrade control register to determine the next application configuration to be loaded.

When Arria GX devices successfully load the application configuration, they enter into user mode. In user mode, the soft logic (Nios processor or state machine and the remote communication interface) assists the Arria GX device in determining when a remote system update is arriving. When a remote system update arrives, the soft logic receives the incoming data, writes it to the configuration memory device, and triggers the device to load the factory configuration. The factory configuration reads the remote system upgrade status register, determines the valid application configuration to load, writes the remote system upgrade control register accordingly, and initiates system reconfiguration.

Arria GX FPGAs support the remote update mode in the AS, FPP, PS, and PPA configuration schemes. In the FPP, PS, and PPA schemes, the MAX II device, microprocessor, or enhanced configuration device should sample the PGM[2..0] outputs from the Arria GX FPGA and transmit the appropriate configuration image. In the AS scheme, the Arria GX device uses the page addresses to read configuration data out of the serial configuration device.

## Local Update Mode

Local update mode is a simplified version of the remote update mode. This feature allows systems to load an application configuration immediately upon power up without loading the factory configuration first. Local update mode does not require the factory configuration to

determine which application configuration to load, because only one application configuration is allowed (at page address one (PGM [2..0] = 001). You can update this application configuration remotely. If an error occurs while loading the application configuration, the factory configuration is automatically loaded.

Upon power up or nCONFIG assertion, the dedicated remote system upgrade circuitry drives out "001" on the PGM[] pins selecting the application configuration stored in page one. If the device encounters any errors during the configuration cycle, the remote system upgrade circuitry retries configuration by driving PGM[2..0] to zero (PGM[2..0] = 000) to select the factory configuration image. The error conditions that trigger a return to the factory configuration are:

■ An internal CRC error
■ An external error signal (nSTATUS detected low)

When the remote system upgrade circuitry detects an external configuration reset (nCONFIG pulsed low) or internal configuration reset (logic array nCONFIG assertion), the device attempts to reload the application configuration from page one.

Figure 12–5 shows the transitions between configurations in local update mode.

*Figure 12–5. Transitions Between Configurations in Local Update Mode*
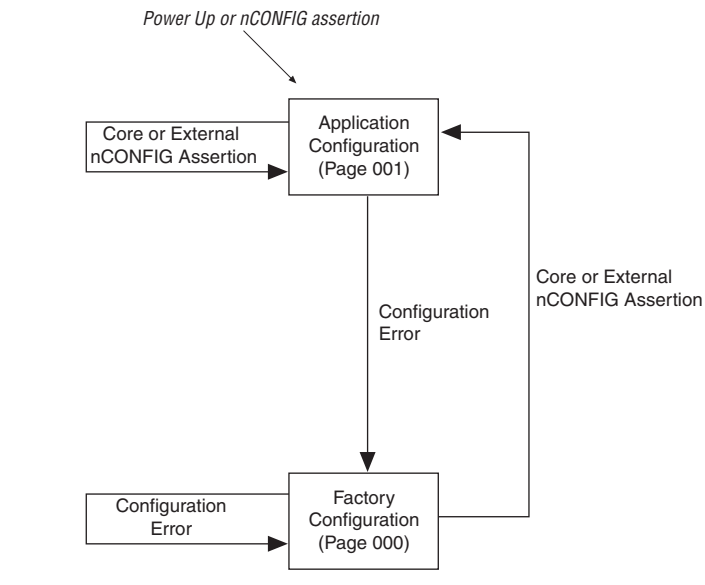
Arria GX FPGAs support local update mode in the FPP, PS, and PPA configuration schemes. In these schemes, the MAX II device, microprocessor, or enhanced configuration device should sample the `PGM[2..0]` outputs from the Arria GX FPGA and transmit the appropriate configuration image.

Local update mode is not supported with the AS configuration scheme, (or serial configuration device), because the Arria GX FPGA cannot determine the start address of the application configuration page upon power up. While the factory configuration is always located at memory address 0x000000, the application configuration can be located at any other sector boundary within the serial configuration device. The start address depends on the size of the factory configuration and is user selectable. Hence, only remote update mode is supported in the AS configuration scheme.

☞ Local update mode is not supported in the AS configuration scheme (with a serial configuration device).

Local update mode supports read access to the remote system upgrade status register. The factory configuration image can use this error status information to determine if a new application configuration must be downloaded from the remote source. After a remote update, the user design should assert the logic array configuration reset (nCONFIG) signal to load the new application configuration.

The device does not support write access to the remote system upgrade control register in local update mode. Write access is not required because this mode only supports one application configuration (eliminating the need to write in a page address) and does not support the user watchdog timer (eliminating the need to enable or disable the timer or specify its time-out value).

☞ The user watchdog timer is disabled in local update mode.

☞ Write access to the remote system upgrade control register is disabled in local update mode. However, the device supports read access to obtain error status information.

## Dedicated Remote System Upgrade Circuitry

This section explains the implementation of the Arria GX remote system upgrade dedicated circuitry. The remote system upgrade circuitry is implemented in hard logic. This dedicated circuitry interfaces to the user-defined factory application configurations implemented in the FPGA logic array to provide the complete remote configuration solution. The remote system upgrade circuitry contains the remote system upgrade

registers, a watchdog timer, and a state machine that controls those components. Figure 12–6 shows the remote system upgrade block's data path.

*Figure 12–6. Remote System Upgrade Circuit Data Path*
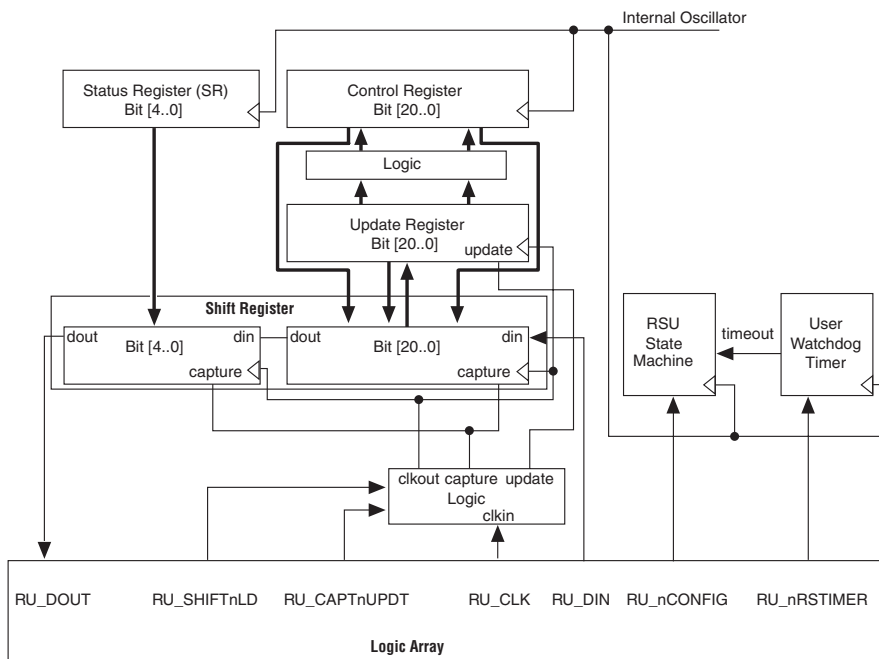
## Remote System Upgrade Registers

The remote system upgrade block contains a series of registers that store the page addresses, watchdog timer settings, and status information. These registers are detailed in Table 12–3.

| Table 12–3. Remote System Upgrade Registers | |
|---|---|
| **Register** | **Description** |
| Shift register | This register is accessible by the logic array and allows the update, status, and control registers to be written and sampled by user logic. Write access is enabled in remote update mode for factory configurations to allow writes to the update register. Write access is disabled in local update mode and for all application configurations in remote update mode. |
| Control register | This register contains the current page address, the user watchdog timer settings, and one bit specifying whether the current configuration is a factory configuration or an application configuration. During a read operation in an application configuration, this register is read into the shift register. When a reconfiguration cycle is initiated, the contents of the update register are written into the control register. |
| Update register | This register contains data similar to that in the control register. However, it can only be updated by the factory configuration by shifting data into the shift register and issuing an update operation. When a reconfiguration cycle is triggered by the factory configuration, the control register is updated with the contents of the update register. During a read in a factory configuration, this register is read into the shift register. |
| Status register | This register is written to by the remote system upgrade circuitry on every reconfiguration to record the cause of the reconfiguration. This information is used by the factory configuration to determine the appropriate action following a reconfiguration. During a capture cycle, this register is read into the shift register. |

The remote system upgrade control and status registers are clocked by the 10-MHz internal oscillator (the same oscillator that controls the user watchdog timer). However, the remote system upgrade shift and update registers are clocked by the user clock input (RU_CLK).

### Remote System Upgrade Control Register

The remote system upgrade control register stores the application configuration page address and user watchdog timer settings. The control register functionality depends on the remote system upgrade mode selection. In remote update mode, the control register page address bits are set to all zeros ($7'b0 = 0000\_000$) at power up in order to load the factory configuration. However, in local update mode the control register page address bits power up as ($7'b1 = 0000\_001$) in order to select the application configuration. Additionally, the control register cannot be updated in local update mode, whereas a factory configuration in remote update mode has write access to this register.

The control register bit positions are shown in Figure 12–7 and defined in Table 12–4. In the figure, the numbers show the bit position of a setting within a register. For example, bit number 8 is the enable bit for the watchdog timer.

*Figure 12–7. Remote System Upgrade Control Register*

| 20 19 18 17 16 15 14 13 12 11 10 9 | 8 | 7 6 5 4 | 3 2 1 | 0 |
|---|---|---|---|---|
| Wd_timer[11..0] | Wd_en | PGM[6..3] | PGM[2..0] | AnF |

The application-not-factory (AnF) bit indicates whether the current configuration loaded in the Arria GX device is the factory configuration or an application configuration. This bit is set high at power up in local update mode, and is set low by the remote system upgrade circuitry when an error condition causes a fall-back to factory configuration. When the AnF bit is high, the control register access is limited to read operations. When the AnF bit is low, the register allows write operations and disables the watchdog timer.

☞ In remote update mode, factory configuration design should set this bit high (1'b1) when updating the contents of the update register with application page address and watchdog timer settings.

*Table 12–4. Remote System Upgrade Control Register Contents (Part 1 of 2)*

| Control Register Bit | Remote System Upgrade Mode | Value | Definition |
|---|---|---|---|
| AnF *(1)* | Local update<br>Remote update | 1'b1<br>1'b0 | Application not factory |
| PGM[2..0] | Local update<br>Remote update (FPP, PS, PPA) | 3'b001<br>3'b000 | Page mode select |
| | Remote update (AS) | 3'b000 | AS configuration start address (StAdd[18..16]) |
| PGM[6..3] | Local update<br>Remote update (FPP, PS, PPA) | 4'b0000<br>4'b0000 | Not used |
| | Remote update (AS) | 4'b0000 | AS configuration start address (StAdd[22..19]) |
| Wd_en | Remote update | 1'b0 | User watchdog timer enable bit |

*Table 12–4. Remote System Upgrade Control Register Contents  (Part 2 of 2)*

| Control Register Bit | Remote System Upgrade Mode | Value | Definition |
|---|---|---|---|
| `Wd_timer[11..0]` | Remote update | 12'b000000000000 | User watchdog time-out value (most significant 12 bits of 29-bit count value: $\{$`Wd_timer[11..0]`, `17'b0`$\}$) |

Note to Table 12–4:
(1)  In remote update mode, the remote configuration block does not update the `AnF` bit automatically (you can update it manually). In local update mode, the remote configuration updates the `AnF` bit with `0` in the factory page and `1` in the application page.

### Remote System Upgrade Status Register

The remote system upgrade status register specifies the reconfiguration trigger condition. The various trigger and error conditions include:

- CRC (cyclic redundancy check) error during application configuration
- `nSTATUS` assertion by an external device due to an error
- FPGA logic array triggered a reconfiguration cycle, possibly after downloading a new application configuration image
- External configuration reset (`nCONFIG`) assertion
- User watchdog timer time out

Figure 12–8 and Table 12–5 specify the contents of the status register. The numbers in the figure show the bit positions within a 5-bit register.

*Figure 12–8. Remote System Upgrade Status Register*

| 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|
| Wd | nCONFIG | Core_nCONFIG | nSTATUS | CRC |

*Table 12–5. Remote System Upgrade Status Register Contents*

| Status Register Bit | Definition | POR Reset Value |
|---|---|---|
| CRC (from configuration) | CRC error caused reconfiguration | 1 bit '0' |
| nSTATUS | nSTATUS caused reconfiguration | 1 bit '0' |
| CORE *(1)* CORE_nCONFIG | Device logic array caused reconfiguration | 1 bit '0' |
| nCONFIG | nCONFIG caused reconfiguration | 1 bit '0' |
| Wd | Watchdog timer caused reconfiguration | 1 bit '0' |

*Note to Table 12–5:*
(1) Logic array reconfiguration forces the system to load the application configuration data into the Arria GX device. This occurs after the factory configuration specifies the appropriate application configuration page address by updating the update register.

## Remote System Upgrade State Machine

The remote system upgrade control and update registers have identical bit definitions, but serve different roles (see Table 12–3 on page 12–15). While both registers can only be updated when the FPGA is loaded with a factory configuration image, the update register writes are controlled by the user logic, and the control register writes are controlled by the remote system upgrade state machine.

In factory configurations, the user logic should send the AnF bit (set high), the page address, and watchdog timer settings for the next application configuration bit to the update register. When the logic array configuration reset (RU_nCONFIG) goes high, the remote system upgrade

state machine updates the control register with the contents of the update register and initiates system reconfiguration from the new application page.

In the event of an error or reconfiguration trigger condition, the remote system upgrade state machine directs the system to load a factory or application configuration (page zero or page one, based on mode and error condition) by setting the control register accordingly. Table 12–6 lists the contents of the control register after such an event occurs for all possible error or trigger conditions.

The remote system upgrade status register is updated by the dedicated error monitoring circuitry after an error condition but before the factory configuration is loaded.

*Table 12–6. Control Register Contents After an Error or Reconfiguration Trigger Condition*

| Reconfiguration Error/Trigger | Control Register Setting | |
|---|---|---|
| | Remote Update | Local Update |
| nCONFIG reset | All bits are 0 | PGM[6..0] = 7'b0000001<br>AnF = 1<br>All other bits are 0 |
| nSTATUS error | All bits are 0 | All bits are 0 |
| CORE triggered reconfiguration | Update register | PGM[6..0] = 7'b0000001<br>AnF = 1<br>All other bits are 0 |
| CRC error | All bits are 0 | All bits are 0 |
| Wd time out | All bits are 0 | All bits are 0 |

Read operations during factory configuration access the contents of the update register. This feature is used by the user logic to verify that the page address and watchdog timer settings were written correctly. Read operations in application configurations access the contents of the control register. This information is used by the user logic in the application configuration.

## User Watchdog Timer

The user watchdog timer prevents a faulty application configuration from stalling the device indefinitely. The system uses the timer to detect functional errors after an application configuration is successfully loaded into the FPGA.

The user watchdog timer is a counter that counts down from the initial value loaded into the remote system upgrade control register by the factory configuration. The counter is 29-bits-wide and has a maximum count value of $2^{29}$. When specifying the user watchdog timer value, specify only the most significant 12 bits. The granularity of the timer setting is $2^{15}$ cycles. The cycle time is based on the frequency of the 10-MHz internal oscillator. Table 12–7 specifies the operating range of the 10-MHz internal oscillator.

*Table 12–7. 10-MHz Internal Oscillator Specifications Note (1)*

| Minimum | Typical | Maximum | Units |
|---------|---------|---------|-------|
| 5 | 6.5 | 10 | MHz |

*Note to Table 12–7:*
(1)    These values are preliminary.

The user watchdog timer begins counting once the application configuration enters FPGA user mode. This timer must be periodically reloaded or reset by the application configuration before the timer expires by asserting RU_nRSTIMER. If the application configuration does not reload the user watchdog timer before the count expires, a time-out signal is generated by the remote system upgrade dedicated circuitry. The time-out signal tells the remote system upgrade circuitry to set the user watchdog timer status bit (Wd) in the remote system upgrade status register and reconfigures the device by loading the factory configuration.

The user watchdog timer is not enabled during the configuration cycle of the FPGA. Errors during configuration are detected by the CRC engine. Also, the timer is disabled for factory configurations. Functional errors should not exist in the factory configuration since it is stored and validated during production and is never updated remotely.

☞    The user watchdog timer is disabled in factory configurations and during the configuration cycle of the application configuration. It is enabled after the application configuration enters user mode.

## Interface Signals between Remote System Upgrade Circuitry & FPGA Logic Array

The dedicated remote system upgrade circuitry drives (or receives) seven signals to (or from) the FPGA logic array. The FPGA logic array uses these signals to read and write the remote system upgrade control, status, and

update registers using the remote system upgrade shift register. Table 12–8 lists each of these seven signals and describes their functionality.

Except for `RU_nRSTIMER` and `RU_CAPTnUPDT`, the logic array signals are enabled for both remote and local update modes and for both factory and application configurations. `RU_nRSTIMER` is only valid for application configurations in remote update mode, since local update configurations and factory configurations have the user watchdog timer disabled. When `RU_CAPTnUPDT` is low, the device can write to the update register only for factory configurations in remote update mode, since this is the only case where the update register is written to by the user logic. When the `RU_nCONFIG` signal goes high, the contents of the update register are written into the control register for controlling the next configuration cycle.

*Table 12–8. Interface Signals between Remote System Upgrade Circuitry & FPGA Logic Array  (Part 1 of 2)*

| Signal Name | Signal Direction | Description |
|---|---|---|
| RU_nRSTIMER | Input to remote system upgrade block (driven by FPGA logic array) | Request from the application configuration to reset the user watchdog timer with its initial count. A falling edge of this signal triggers a reset of the user watchdog timer. |
| RU_nCONFIG | Input to remote system upgrade block (driven by FPGA logic array) | When driven low, this signal triggers the device to reconfigure. If asserted by the factory configuration in remote update mode, the application configuration specified in the remote update control register is loaded. If requested by the application configuration in remote update mode, the factory configuration is loaded. In the local updated mode, the application configuration is loaded whenever this signal is asserted. |
| RU_CLK | Input to remote system upgrade block (driven by FPGA logic array) | Clocks the remote system upgrade shift register and update register so that the contents of the status, control, and update registers can be read, and so that the contents of the update register can be loaded. The shift register latches data on the rising edge of this clock signal. |

| Table 12–8. Interface Signals between Remote System Upgrade Circuitry & FPGA Logic Array  (Part 2 of 2) | | |
|---|---|---|
| **Signal Name** | **Signal Direction** | **Description** |
| RU_SHIFTnLD | Input to remote system upgrade block (driven by FPGA logic array) | This pin determines if the shift register contents are shifted over during the next clock edge or loaded in/out.<br><br>When this signal is driven high (1'b1), the remote system upgrade shift register shifts data left on each rising edge of RU_CLK.<br><br>When RU_SHIFTnLD is driven low (1'b0) and RU_CAPTnUPDT is driven low (1'b0), the remote system upgrade update register is updated with the contents of the shift register on the rising edge of RU_CLK.<br><br>When RU_SHIFTnLD is driven low (1'b0) and RU_CAPTnUPDT is driven high (1'b1), the remote system upgrade shift register captures the status register and either the control or update register (depending on whether the current configuration is application or factory, respectively) on the rising edge of RU_CLK. |
| RU_CAPTnUPDT | Input to remote system upgrade block (driven by FPGA logic array) | This pin determines if the contents of the shift register are captured or updated on the next clock edge.<br><br>When the RU_SHIFTnLD signal is driven high (1'b1), this input signal has no function.<br><br>When RU_SHIFTnLD is driven low (1'b0) and RU_CAPTnUPDT is driven high (1'b1), the remote system upgrade shift register captures the status register and either the control or update register (depending on whether the current configuration is application or factory, respectively) on the rising edge of RU_CLK.<br><br>When RU_SHIFTnLD is driven low (1'b0) and RU_CAPTnUPDT is driven low (1'b0), the remote system upgrade update register is updated with the contents of the shift register on the rising edge of RU_CLK.<br><br>In local update mode, a low input on RU_CAPTnUPDT has no function, because the update register cannot be updated in this mode. |
| RU_DIN | Input to remote system upgrade block (driven by FPGA logic array) | Data to be written to the remote system upgrade shift register on the rising edge of RU_CLK. To load data into the shift register, RU_SHIFTnLD must be asserted. |
| RU_DOUT | Output from remote system upgrade block (driven to FPGA logic array) | Output data from the remote system upgrade shift register to be read by logic array logic. New data arrives on each rising edge of RU_CLK. |

### Remote System Upgrade Pin Descriptions

Table 12–9 describes the dedicated remote system upgrade configuration pins. For descriptions of all the configuration pins, refer to the *Configuring Arria GX Devices* chapter in volume 2 of the *Arria GX Handbook*.

| *Table 12–9. Arria GX Remote System Upgrade Pins* | | | | |
|---|---|---|---|---|
| **Pin Name** | **User Mode** | **Configuration Scheme** | **Pin Type** | **Description** |
| RUnLU | N/A if using remote system upgrade in FPP, PS, AS, or PPA modes. I/O if not using these modes. | Remote configuration in FPP, PS, or PPA | Input | Input that selects between remote update and local update. A logic high (1.5-V, 1.8-V, 2.5-V, 3.3-V) selects remote update, and a logic low selects local update. When not using remote update or local update configuration modes, this pin is available as a general-purpose user I/O pin. When using remote configuration in AS mode, set the RUnLU pin to high because AS does not support local update. |
| PGM[2..0] | N/A if using remote system upgrade in FPP, PS, or PPA modes. I/O if not using these modes. | Remote configuration in FPP, PS or PPA | Output | These output pins select one of eight pages in the memory (either flash or enhanced configuration device) when using remote update mode. When not using remote update or local update configuration modes, these pins are available as general-purpose user I/O pins. |

## Quartus II Software Support

Implementation in your design requires a remote system upgrade interface between the FPGA logic array and remote system upgrade circuitry. You also need to generate configuration files for production and remote programming of the system configuration memory. The Quartus® II software provides these features.

The two implementation options, altremote_update megafunction and remote system upgrade atom, are for the interface between the remote system upgrade circuitry and the FPGA logic array interface.

### altremote_update Megafunction

The `altremote_update` megafunction provides a memory-like interface to the remote system upgrade circuitry and handles the shift register read/write protocol in FPGA logic. This implementation is suitable for designs that implement the factory configuration functions using a Nios processor in the FPGA.

Tables 12–10 and 12–11 describe the input and output ports available on the `altremote_update` megafunction. Table 12–12 shows the `param[2..0]` bit settings.

| *Table 12–10. Input Ports of the altremote_update Megafunction    (Part 1 of 2)* | | | |
|---|---|---|---|
| **Port Name** | **Required** | **Source** | **Description** |
| clock | Y | Logic Array | Clock input to the `altremote_update` block. All operations are performed with respects to the rising edge of this clock. |
| reset | Y | Logic Array | Asynchronous reset, which is used to initialize the remote update block. To ensure proper operation, the remote update block must be reset before first accessing the remote update block. This signal is not affected by the busy signal and will reset the remote update block even if busy is logic high. This means that if the reset signal is driven logic high during writing of a parameter, the parameter will not be properly written to the remote update block. |
| reconfig | Y | Logic Array | When driven logic high, reconfiguration of the device is initiated using the current parameter settings in the remote update block. If busy is asserted, this signal is ignored. This is to ensure all parameters are completely written before reconfiguration begins. |
| reset_timer | N | Logic Array | This signal is required if you are using the watchdog timer feature. A logic high resets the internal watchdog timer. This signal is not affected by the busy signal and can reset the timer even when the remote update block is busy. If this port is left connected, the default value is 0. |
| read_param | N | Logic Array | Once `read_param` is sampled as a logic high, the busy signal is asserted. While the parameter is being read, the busy signal remains asserted, and inputs on `param[]` are ignored. Once the busy signal is deactivated, the next parameter can be read. If this port is left unconnected, the default value is 0. |

### Table 12–10. Input Ports of the altremote_update Megafunction    (Part 2 of 2)

| Port Name | Required | Source | Description |
|---|---|---|---|
| write_param | N | Logic Array | This signal is required if you intend on writing parameters to the remote update block. When driven logic high, the parameter specified on the param[] port should be written to the remote update block with the value on data_in[]. The number of valid bits on data_in[] is dependent on the parameter type. This signal is sampled on the rising edge of clock and should only be asserted for one clock cycle to prevent the parameter from being re-read on subsequent clock cycles. Once write_param is sampled as a logic high, the busy signal is asserted. While the parameter is being written, the busy signal remains asserted, and inputs on param[] and data_in[] are ignored. Once the busy signal is deactivated, the next parameter can be written. This signal is only valid when the Current_Configuration parameter is factory since parameters cannot be written in application configurations. If this port is left unconnected, the default value is 0. |
| param[2..0] | N | Logic Array | 3-bit bus that selects which parameter should be read or written. If this port is left unconnected, the default value is 0. |
| data_in[11..0] | N | Logic Array | This signal is required if you intend on writing parameters to the remote update block 12-bit bus used when writing parameters, which specifies the parameter value. The parameter value is requested using the param[] input and by driving the write_param signal logic high, at which point the busy signal goes logic high and the value of the parameter is captured from this bus. For some parameters, not all 12 bits are used, in which case only the least significant bits are used. This port is ignored if the Current_Configuration parameter is set to an application configuration since writing of parameters is only allowed in the factory configuration. If this port is left unconnected, the default value is 0. |

*Note to Table 12–10:*

(1)    Logic array source means that you can drive the port from internal logic or any general-purpose I/O pin.

**Table 12–11. Output Ports of the altremote_update Megafunction**

| Port Name | Required | Destination | Description |
|-----------|----------|-------------|-------------|
| `busy` | Y | Logic Array | When this signal is a logic high, the remote update block is busy either reading or writing a parameter. When the remote update block is busy, it ignores its `data_in[]`, `param[]`, and `reconfig` inputs. This signal goes high when `read_param` or `write_param` is asserted and remains asserted until the operation is complete. |
| `pgm_out[2..0]` | Y | `PGM[2..0]` pins | 3-bit bus that specifies the page pointer of the configuration data to be loaded when the device is reconfigured. This port must be connected to the `PGM[]` output pins, which should be connected to the external configuration device. |
| `data_out[11..0]` | N | Logic Array | 12-bit bus used when reading parameters, which reads out the parameter value. The parameter value is requested using the `param[]` input and by driving the `read_param` signal logic high, at which point the busy signal goes logic high. When the busy signal goes low, the value of the parameter is driven out on this bus. The `data_out[]` port is only valid after a `read_param` has been issued and once the busy signal is deasserted. At any other time, its output values are invalid. For example, even though the `data_out[]` port may toggle during a writing of a parameter, these values are not a valid representation of what was actually written to the remote update block. For some parameters, not all 12 bits are used, in which case only the least significant bits are used. |

*Note to Table 12–11:*
(1)    Logic array destination means that you can drive the port to internal logic or any general-purpose I/O pin.

**Table 12–12. Parameter Settings for the altremote_update Megafunction    (Part 1 of 2)**

| Selected Parameter | param[2..0] Bit Setting | Width of Parameter Value | POR Reset Value | Description |
|--------------------|-------------------------|--------------------------|-----------------|-------------|
| Status Register Contents | 000 | 5 | 5 bit '0 | Specifies the reason for re-configuration, which could be caused by a CRC error during configuration, `nSTATUS` being pulled low due to an error, the device core caused an error, `nCONFIG` pulled low, or the watchdog timer timed-out. This parameter can only be read. |
| Watchdog Timeout Value | 010 | 12 | 12 bits '0 | User watchdog timer time-out value. Writing of this parameter is only allowed when in the factory configuration. |

*Table 12–12. Parameter Settings for the altremote_update Megafunction   (Part 2 of 2)*

| Selected Parameter | param[2..0] Bit Setting | Width of Parameter Value | POR Reset Value | Description |
|---|---|---|---|---|
| Watchdog Enable | 011 | 1 | 1 bit '0 | User watchdog timer enable. Writing of this parameter is only allowed when in the factory configuration |
| Page select | 100 | 3 (FPP, PS, PPA) | 3 bit '001' - Local configuration | Page mode selection. Writing of this parameter is only allowed when in the factory configuration. |
| | | | 3 bit '000' - Remote configuration | |
| | | 7 (AS) | 7 bit '0000000' - Remote configuration | |
| Current configuration (AnF) | 101 | 1 | 1 bit '0' - Factory | Specifies whether the current configuration is factory or and application configuration. This parameter can only be read. |
| | | | 1 bit '1' - Application | |
| Illegal values | 001 | — | — | — |
| | 110 | — | — | — |
| | 111 | — | — | — |

### Remote System Upgrade Atom

The remote system upgrade atom is a WYSIWYG atom or primitive that can be instantiated in your design. The primitive is used to access the remote system upgrade shift register, logic array reset, and watchdog timer reset signals. The ports on this primitive are the same as those listed in Table 12–8. This implementation is suitable for designs that implement the factory configuration functions using state machines (without a processor).

## System Design Guidelines

The following general guidelines are applicable when implementing remote system upgrade in Arria GX FPGAs. Guidelines for specific configuration schemes are also discussed in this section.

■    After downloading a new application configuration, the soft logic implemented in the FPGA can validate the integrity of the data received over the remote communication interface. This optional step helps avoid configuration attempts with bad or incomplete configuration data. However, in the event that bad or incomplete configuration data is sent to the FPGA, it detects the data corruption using the CRC signature attached to each configuration frame.

■ The auto-reconfigure on configuration error option bit is ignored when remote system upgrade is enabled in your system. This option is always enabled in remote configuration designs, allowing your system to return to the safe factory configuration in the event of an application configuration error or user watchdog timer time-out.

## Remote System Upgrade With Serial Configuration Devices

Remote system upgrade support in the AS configuration scheme is similar to support in other schemes, with the following exceptions:

■ The remote system upgrade block provides the AS configuration controller inside the Arria GX FPGA with a 7-bit page start address (PGM[6..0]) instead of driving the 3-bit page mode pins (PGM[2..0]) used in FPP, PS, and PPA configuration schemes. This 7-bit address forms the 24-bit configuration start address (StAdd[23..0]). Table 12–13 illustrates the start address generation using the page address registers.

■ The configuration start address for factory configuration is always set to 24'b0.

■ PGM[2..0] pins on Arria GX devices are not used in AS configuration schemes and can be used as regular I/O pins.

■ The Nios ASMI peripheral can be used to update configuration data within the serial configuration device.

| *Table 12–13. AS Configuration Start Address Generation* | | | | |
|---|---|---|---|---|
| **Serial Configuration Device** | **Serial Configuration Device Density (MB)** | **Add[23]** | **PGM[6..0] (Add[22..16])** | **Add[15..0]** |
| EPCS64 | 64 | 0 | MSB[6..0] | All 0s |
| EPCS16 | 16 | 0 | 00, MSB[4..0] | All 0s |
| EPCS4 | 4 | 0 | 0000, MSB[2..0] | All 0s |

## Remote System Upgrade With a MAX II Device or Microprocessor & Flash Device

This setup requires the MAX II device or microprocessor to support page addressing. MAX II or microprocessor devices implementing remote system upgrade should emulate the enhanced configuration device page mode feature. The PGM[2..0] output pins from the Arria GX device must be sampled to determine which configuration image is to be loaded into the FPGA.

If the FPGA does not release CONF_DONE after all data has been sent, the MAX II microprocessor should reset the FPGA back to the factory image by pulsing its nSTATUS pin low.

The MAX II device or microprocessor and flash configuration can use FPP, PS, or PPA. Decompression is supported in the FPP (requires 4× DCLK) and PS modes only. Figure 12–9 shows a system block diagram for remote system upgrade with the MAX II device or microprocessor and flash.

*Figure 12–9. System Block Diagram for Remote System Upgrade With MAX II Device or Microprocessor & Flash Device*
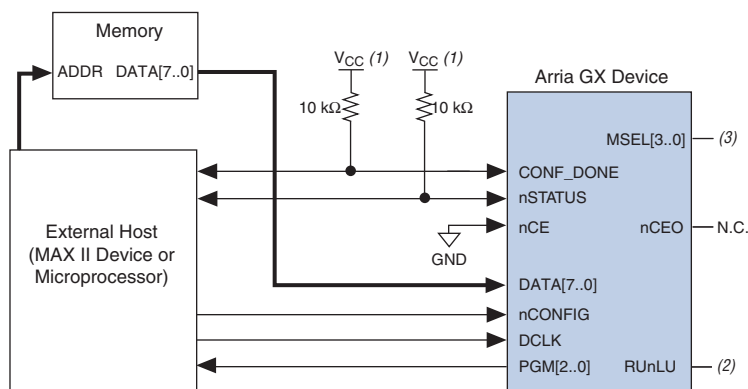


Notes to Figure 12–9:
(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for the device.
(2) Connect RUnLU to GND or $V_{CC}$ to select between remote and local update modes.
(3) Connect MSEL[3..0] to 0100 to enable remote update remote system upgrade mode.

## Remote System Upgrade with Enhanced Configuration Devices

■ Enhanced Configuration devices support remote system upgrade with FPP or PS configuration schemes. The Arria GX decompression feature is only supported in the PS mode. The enhanced configuration device's decompression feature is supported in both PS and FPP schemes.
■ In remote update mode, neither the factory configuration nor the application configurations should alter the enhanced configuration device's option bits or the page 000 factory configuration data. This ensures that an error during remote update can always be resolved by reverting to the factory configuration located at page 000.

■   The enhanced configuration device features an error checking mechanism to detect instances when the FPGA fails to detect the configuration preamble. In these instances, the enhanced configuration device pulses the nSTATUS signal low, and the remote system upgrade circuitry attempts to load the factory configuration. Figure 12–10 shows a system block diagram for remote system upgrade with enhanced configuration devices.

*Figure 12–10. System Block Diagram for Remote System Upgrade with Enhanced Configuration Devices*
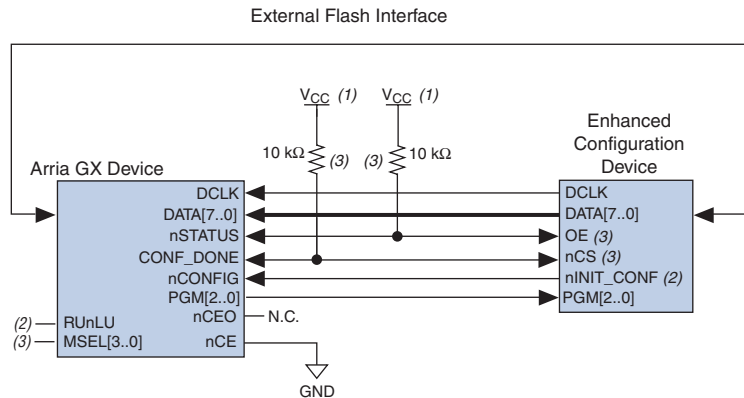


*Notes to Figure 12–10:*
(1)   Connect the pull-up resistor to a supply that provides an acceptable input signal for the device.
(2)   Connect RUnLU to GND or $V_{CC}$ to select between remote and local update modes.
(3)   Connect MSEL[3..0] to 0100 to enable remote update remote system upgrade mode.

**Conclusion**

Arria GX devices offer remote system upgrade capability, where you can upgrade a system in real-time through any network. Remote system upgrade helps to deliver feature enhancements and bug fixes without costly recalls, reduces time to market, and extends product life cycles. The dedicated remote system upgrade circuitry in Arria GX devices provides error detection, recovery, and status information to ensure reliable reconfiguration.

## Referenced Documents

This chapter references the following documents:

■ *Configuring Arria GX Devices* chapter of the *Arria GX Handbook*
■ *Enhanced Configuration Devices (EPC4, EPC8, & EPC16) Data Sheet* chapter in volume 2 of the *Configuration Handbook*
■ *Serial Configuration Devices (EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128) Data Sheet* in volume 2 of the *Configuration Handbook*

## Document Revision History

Table 12–14 shows the revision history for this chapter.

| *Table 12–14. Document Revision History* | | |
|---|---|---|
| **Date and Document Version** | **Changes Made** | **Summary of Changes** |
| May 2008, v1.2 | Minor text edits. | — |
| August 2007, v1.1 | Added the "Referenced Documents" section. | — |
| | Minor text edits. | — |
| May 2007, v1.0 | Initial Release | N/A |

# 13. IEEE 1149.1 (JTAG) Boundary-Scan Testing for Arria GX Devices

**Introduction**

As printed circuit boards (PCBs) become more complex, the need for thorough testing becomes increasingly important. Advances in surface-mount packaging and PCB manufacturing have resulted in smaller boards, making traditional test methods (such as; external test probes and "bed-of-nails" test fixture) harder to implement. As a result, cost savings from PCB space reductions increases the cost for traditional testing methods.

In the 1980s, the Joint Test Action Group (JTAG) developed a specification for boundary-scan testing that was later standardized as the IEEE Std. 1149.1 specification. This Boundary-Scan Test (BST) architecture offers the capability to test efficiently components on PCBs with tight lead spacing.

This BST architecture tests pin connections without using physical test probes and captures functional data while a device is operating normally. Boundary-scan cells in a device can force signals onto pins or capture data from pin or logic array signals. Forced test data is serially shifted into the boundary-scan cells. Captured data is serially shifted out and externally compared to expected results. Figure 13–1 illustrates the concept of BST.

*Figure 13–1. IEEE Std. 1149.1 Boundary-Scan Testing*
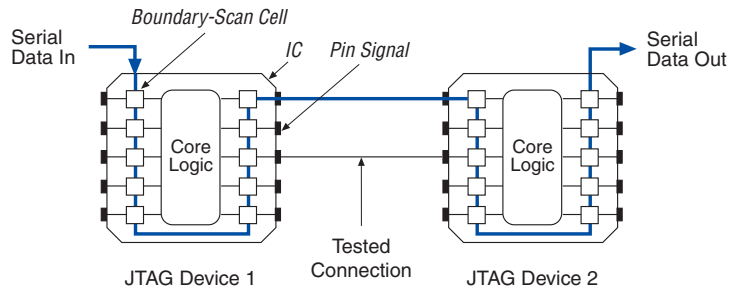
This chapter discusses how to use the IEEE Std. 1149.1 BST circuitry in Arria™ GX devices, including:

- IEEE Std. 1149.1 BST architecture
- IEEE Std. 1149.1 boundary-scan register
- IEEE Std. 1149.1 BST operation control
- I/O Voltage Support in JTAG Chain
- IEEE Std. 1149.1 BST circuitry utilization
- IEEE Std. 1149.1 BST circuitry disabling
- IEEE Std. 1149.1 BST guidelines
- Boundary-Scan Description Language (BSDL) support

In addition to BST, you can use the IEEE Std. 1149.1 controller for Arria GX device in-circuit reconfiguration (ICR). However, this chapter only discusses the BST feature of the IEEE Std. 1149.1 circuitry.

For information on configuring Arria GX devices via the IEEE Std. 1149.1 circuitry, refer to the *Configuring Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*.

☞ When configuring via JTAG make sure that Arria GX, Stratix II, Stratix II GX, Stratix®, Cyclone® II, and Cyclone devices are within the first 17 devices in a JTAG chain. All of these devices have the same JTAG controller. If any of the Arria GX, Stratix II, Stratix II GX, Stratix, Cyclone II, and Cyclone devices are in the 18th or further position, configuration fails. This does not affect SignalTap® II or boundary-scan testing.

This chapter contains the following sections:

# IEEE Std. 1149.1 BST Architecture

An Arria GX device operating in IEEE Std. 1149.1 BST mode uses four required pins, TDI, TDO, TMS, and TCK, and one optional pin, TRST. The TCK pin has an internal weak pull-down resistor, while the TDI, TMS, and TRST pins have weak internal pull-ups. The TDO output pin is powered by $V_{CCIO}$ in I/O bank 4. All of the JTAG input pins are powered by the 3.3-V $V_{CCPD}$ supply. All user I/O pins are tri-stated during JTAG configuration.

☞ For recommendations on how to connect a JTAG chain with multiple voltages across the devices in the chain, refer to "I/O Voltage Support in JTAG Chain" on page 13–17.

Table 13–1 summarizes the functions of each of these pins.

*Table 13–1. IEEE Std. 1149.1 Pin Descriptions*

| Pin | Description | Function |
|-----|-------------|----------|
| TDI | Test data input | Serial input pin for instructions as well as test and programming data. Data is shifted in on the rising edge of TCK. |
| TDO | Test data output | Serial data output pin for instructions as well as test and programming data. Data is shifted out on the falling edge of TCK. The pin is tri-stated if data is not being shifted out of the device. |
| TMS | Test mode select | Input pin that provides the control signal to determine the transitions of the Test Access Port (TAP) controller state machine. Transitions within the state machine occur at the rising edge of TCK. Therefore, TMS must be set up before the rising edge of TCK. TMS is evaluated on the rising edge of TCK. |
| TCK | Test clock input | The clock input to the BST circuitry. Some operations occur at the rising edge, while others occur at the falling edge. |
| TRST | Test reset input (optional) | Active-low input to asynchronously reset the boundary-scan circuit. This pin should be driven low when not in boundary-scan operation and for non-JTAG users the pin should be permanently tied to GND. |

The IEEE Std. 1149.1 BST circuitry requires the following registers:

■ The instruction register determines the action to be performed and the data register to be accessed.
■ The bypass register is a 1-bit-long data register that provides a minimum-length serial path between TDI and TDO.
■ The boundary-scan register is a shift register composed of all the boundary-scan cells of the device.

Figure 13–2 shows a functional model of the IEEE Std. 1149.1 circuitry.

*Figure 13–2. IEEE Std. 1149.1 Circuitry*
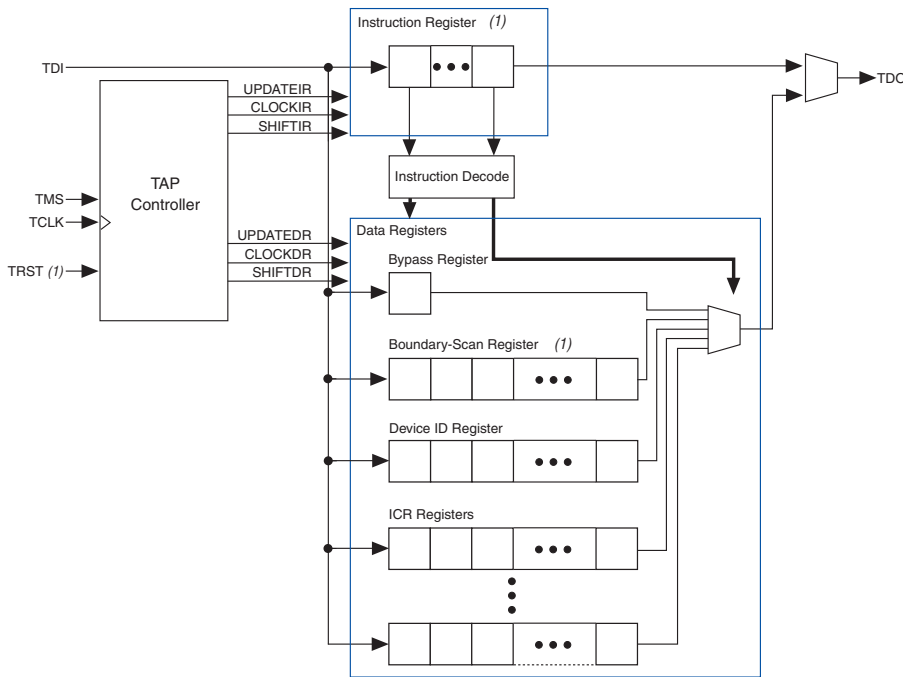


*Note to Figure 13–2:*
(1)    Refer to the appropriate device data sheet for register lengths.

IEEE Std. 1149.1 boundary-scan testing is controlled by a test access port (TAP) controller. For more information on the TAP controller, refer to "IEEE Std. 1149.1 BST Operation Control" on page 13–7. The `TMS` and `TCK` pins operate the TAP controller, and the `TDI` and `TDO` pins provide the serial path for the data registers. The `TDI` pin also provides data to the instruction register, which then generates control logic for the data registers.

## IEEE Std. 1149.1 Boundary-Scan Register

The boundary-scan register is a large serial shift register that uses the `TDI` pin as an input and the `TDO` pin as an output. The boundary-scan register consists of 3-bit peripheral elements that are associated with Arria GX I/O pins. You can use the boundary-scan register to test external pin connections or to capture internal data.

Refer to the the *Configuration & Testing* chapter in volume 1 of the *Arria GX Device Handbook* for the Arria GX family device boundary-scan register lengths.

Figure 13–3 shows how test data is serially shifted around the periphery of the IEEE Std. 1149.1 device.

*Figure 13–3. Boundary-Scan Register*



Each peripheral element is either an I/O pin, dedicated input pin, or dedicated configuration pin.
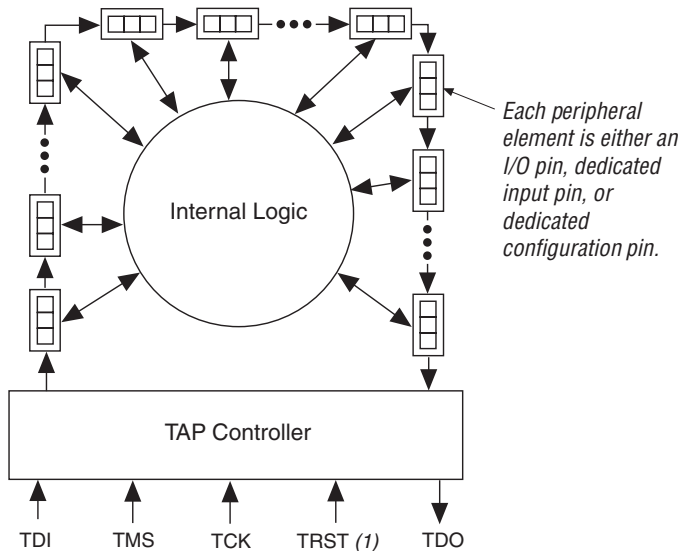
## Boundary-Scan Cells of a Arria GX Device I/O Pin

The Arria GX device 3-bit boundary-scan cell (BSC) consists of a set of capture registers and a set of update registers. The capture registers can connect to internal device data via the OUTJ, OEJ, and PIN_IN signals, while the update registers connect to external data through the PIN_OUT, and PIN_OE signals. The global control signals for the IEEE Std. 1149.1 BST registers (such as shift, clock, and update) are generated internally by the TAP controller. The MODE signal is generated by a decode of the instruction register. The data signal path for the boundary-scan register runs from the serial data in (SDI) signal to the serial data out (SDO) signal. The scan register begins at the TDI pin and ends at the TDO pin of the device.

Figure 13–4 shows the Arria GX device's user I/O boundary-scan cell.

*Figure 13–4. Arria GX Device's User I/O BSC with IEEE Std. 1149.1 BST Circuitry*
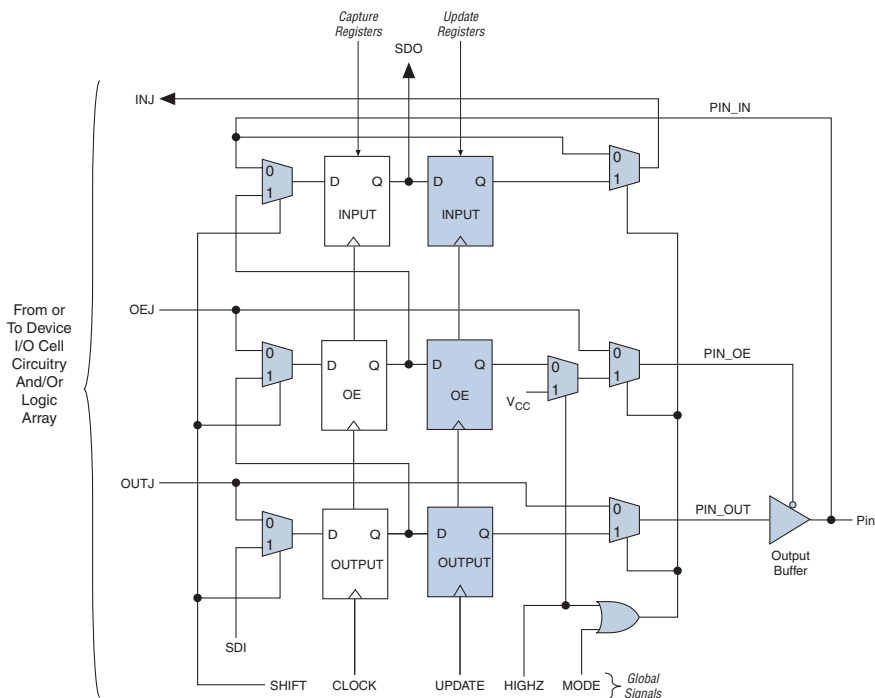


Table 13–2 describes the capture and update register capabilities of all boundary-scan cells within Arria GX devices.

*Table 13–2. Arria GX Device Boundary Scan Cell Descriptions* **(Part 1 of 2)** *Note (1)*

| Pin Type | Captures | | | Drives | | | Comments |
|---|---|---|---|---|---|---|---|
| | Output Capture Register | OE Capture Register | Input Capture Register | Output Update Register | OE Update Register | Input Update Register | |
| User I/O pins | OUTJ | OEJ | PIN_IN | PIN_OUT | PIN_OE | INJ | NA |
| Dedicated clock input | 0 | 1 | PIN_IN | N.C. *(2)* | N.C. *(2)* | N.C. *(2)* | PIN_IN drives to clock network or logic array |
| Dedicated input *(3)* | 0 | 1 | PIN_IN | N.C. *(2)* | N.C. *(2)* | N.C. *(2)* | PIN_IN drives to control logic |

*Table 13–2. Arria GX Device Boundary Scan Cell Descriptions*  *(Part 2 of 2)*  *Note (1)*

| Pin Type | Captures | | | Drives | | | Comments |
|---|---|---|---|---|---|---|---|
| | Output Capture Register | OE Capture Register | Input Capture Register | Output Update Register | OE Update Register | Input Update Register | |
| Dedicated bidirectional (open drain) *(4)* | 0 | `OEJ` | `PIN_IN` | N.C. *(2)* | N.C. *(2)* | N.C. *(2)* | `PIN_IN` drives to configuration control |
| Dedicated bidirectional *(5)* | `OUTJ` | `OEJ` | `PIN_IN` | N.C. *(2)* | N.C. *(2)* | N.C. *(2)* | `PIN_IN` drives to configuration control and `OUTJ` drives to output buffer |
| Dedicated output *(6)* | `OUTJ` | 0 | 0 | N.C. *(2)* | N.C. *(2)* | N.C. *(2)* | `OUTJ` drives to output buffer |

*Notes to Table 13–2:*

(1)   `TDI`, `TDO`, `TMS`, `TCK`, all $V_{CC}$ and GND pin types, `VREF`, and `TEMP_DIODE` pins do not have BSCs.
(2)   No Connect (N.C.).
(3)   This includes pins `PLL_ENA`, `nCONFIG`, `MSEL0`, `MSEL1`, `MSEL2`, `MSEL3`, `nCE`, `VCCSEL`, `PORSEL`, and `nIO_PULLUP`.
(4)   This includes pins `CONF_DONE` and `nSTATUS`.
(5)   This includes pin `DCLK`.
(6)   This includes pin `nCEO`.

# IEEE Std. 1149.1 BST Operation Control

Arria GX devices implement the following IEEE Std. 1149.1 BST instructions:

■   `SAMPLE/PRELOAD` instruction mode is used to take snapshot of the device data without interrupting normal device operations

■   `EXTEST` instruction mode is used to check external pin connections between devices

■   `BYPASS` instruction mode is used when an instruction code consisting of all ones is loaded into the instruction register

■   `IDCODE` instruction mode is used to identify the devices in an IEEE Std. 1149.1 chain

■   `USERCODE` instruction mode is used to examine the user electronic signature within the device along an IEEE Std. 1149.1 chain.

■   `CLAMP` instruction mode is used to allow the state of the signals driven from the pins to be determined from the boundary-scan register while the bypass register is selected as the serial path between the TDI and TDO ports

■   `HIGHZ` instruction mode sets all of the user I/O pins to an inactive drive state

The BST instruction length is 10 bits. These instructions are described later in this chapter.

For summaries of the BST instructions and their instruction codes, refer to the *Configuration and Testing* chapter in volume 1 of the *Arria GX Device Handbook*.

The IEEE Std. 1149.1 TAP controller, a 16-state state machine clocked on the rising edge of TCK, uses the TMS pin to control IEEE Std. 1149.1 operation in the device. Figure 13–5 shows the TAP controller state machine.
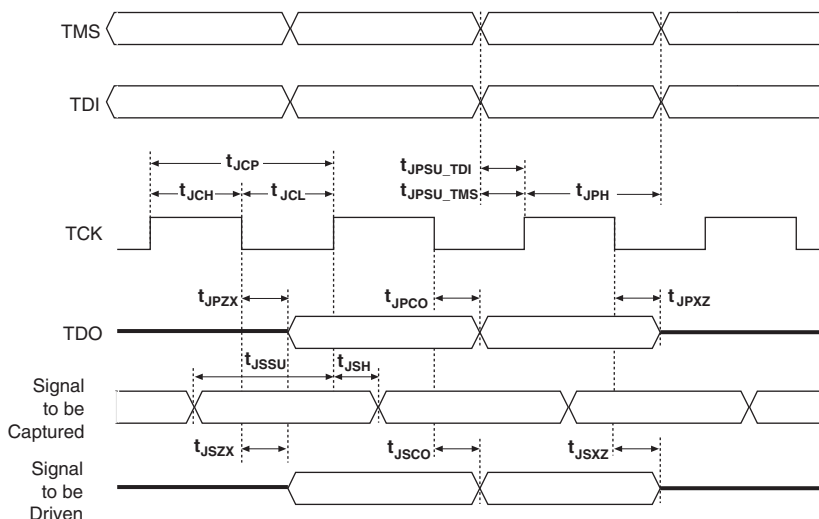
*Figure 13–5. IEEE Std. 1149.1 TAP Controller State Machine*
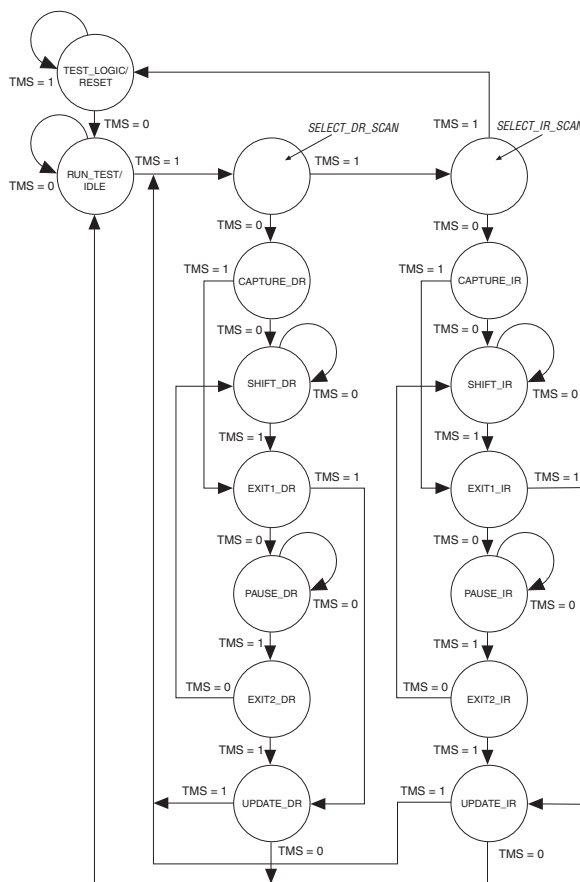
When the TAP controller is in the TEST_LOGIC/RESET state, the BST circuitry is disabled, the device is in normal operation, and the instruction register is initialized with IDCODE as the initial instruction. At device power-up, the TAP controller starts in this TEST_LOGIC/RESET state. In addition, forcing the TAP controller to the TEST_LOGIC/RESET state is done by holding TMS high for five TCK clock cycles or by holding the TRST pin low. Once in the TEST_LOGIC/RESET state, the TAP controller remains in this state as long as TMS is held high (while TCK is clocked) or TRST is held low.

Figure 13–6 shows the timing requirements for the IEEE Std. 1149.1 signals.

*Figure 13–6. IEEE Std. 1149.1 Timing Waveforms*

To start IEEE Std. 1149.1 operation, select an instruction mode by advancing the TAP controller to the shift instruction register (`SHIFT_IR`) state and shift in the appropriate instruction code on the `TDI` pin. The waveform diagram in Figure 13–7 represents the entry of the instruction code into the instruction register. Figure 13–7 shows the values of `TCK`, `TMS`, `TDI`, `TDO`, and the states of the TAP controller. From the `RESET` state, `TMS` is clocked with the pattern `01100` to advance the TAP controller to `SHIFT_IR`.

*Figure 13–7. Selecting the Instruction Mode*
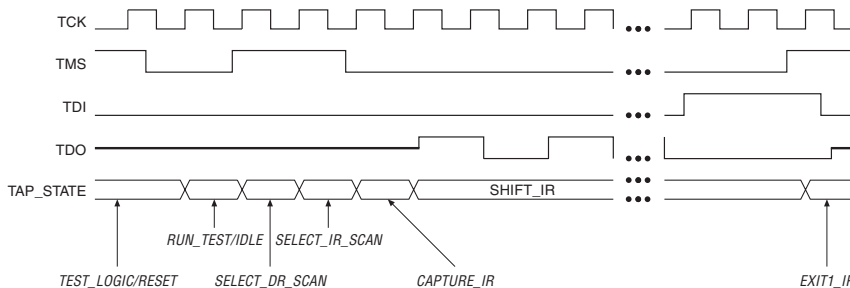


The `TDO` pin is tri-stated in all states except in the `SHIFT_IR` and `SHIFT_DR` states. The `TDO` pin is activated at the first falling edge of `TCK` after entering either of the shift states and is tri-stated at the first falling edge of `TCK` after leaving either of the shift states.

When the `SHIFT_IR` state is activated, `TDO` is no longer tri-stated, and the initial state of the instruction register is shifted out on the falling edge of `TCK`. `TDO` continues to shift out the contents of the instruction register as long as the `SHIFT_IR` state is active. The TAP controller remains in the `SHIFT_IR` state as long as `TMS` remains low.

During the `SHIFT_IR` state, an instruction code is entered by shifting data on the `TDI` pin on the rising edge of `TCK`. The last bit of the instruction code is clocked at the same time that the next state, `EXIT1_IR`, is activated. Set `TMS` high to activate the `EXIT1_IR` state. Once in the `EXIT1_IR` state, `TDO` becomes tri-stated again. `TDO` is always tri-stated except in the `SHIFT_IR` and `SHIFT_DR` states. After an instruction code is entered correctly, the TAP controller advances to serially shift test data in one of three modes. The three serially shift test data instruction modes are discussed on the following pages:

■ "SAMPLE/PRELOAD Instruction Mode" on page 13–11
■ "EXTEST Instruction Mode" on page 13–13
■ "BYPASS Instruction Mode" on page 13–15

## SAMPLE/PRELOAD Instruction Mode

The SAMPLE/PRELOAD instruction mode allows you to take a snapshot of device data without interrupting normal device operation. However, this instruction is most often used to preload the test data into the update registers prior to loading the EXTEST instruction. Figure 13–8 shows the capture, shift, and update phases of the SAMPLE/PRELOAD mode.

*Figure 13–8. IEEE Std. 1149.1 BST SAMPLE/PRELOAD Mode*

## Capture Phase

*In the capture phase, the signals at the pin, OEJ and OUTJ, are loaded into the capture registers. The CLOCK signals is supplied by the TAP controller's CLOCKDR output. The data retained in these registers consists of signals from normal device operation.*

## Shift & Update Phases

*In the shift phase, the previously captured signals at the pin, OEJ and OUTJ, are shifted out of the boundary-scan register via the TDO pin using CLOCK. As data is shifted out, the patterns for the next test can be shifted in via the TDI pin.*

*In the update phase, data is transferred from the capture to the UPDATE registers using the UPDATE clock. The data stored in the UPDATE registers can be used for the EXTEST instruction.*
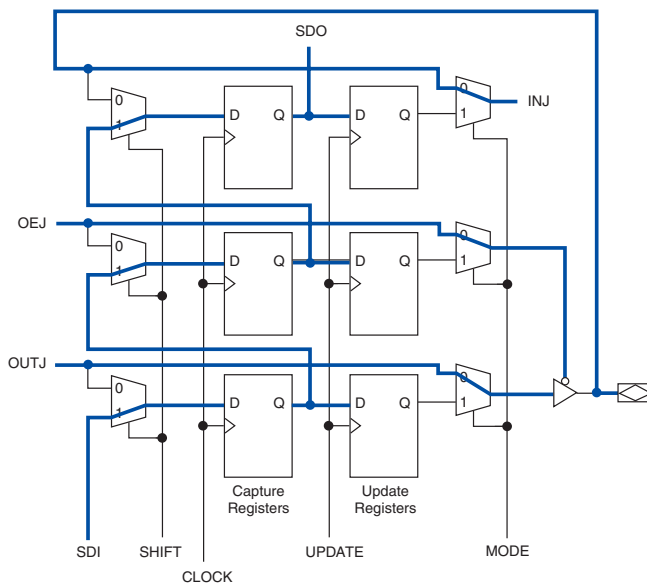
During the capture phase, multiplexers preceding the capture registers select the active device data signals. This data is then clocked into the capture registers. The multiplexers at the outputs of the update registers also select active device data to prevent functional interruptions to the device. During the shift phase, the boundary-scan shift register is formed by clocking data through capture registers around the device periphery and then out of the TDO pin. The device can simultaneously shift new test data into TDI and replace the contents of the capture registers. During the update phase, data in the capture registers is transferred to the update registers. This data can then be used in the EXTEST instruction mode. Refer to "EXTEST Instruction Mode" on page 13–13 for more information.

Figure 13–9 shows the SAMPLE/PRELOAD waveforms. The SAMPLE/PRELOAD instruction code is shifted in through the TDI pin. The TAP controller advances to the CAPTURE_DR state and then to the SHIFT_DR state, where it remains if TMS is held low. The data that was present in the capture registers after the capture phase is shifted out of the TDO pin. New test data shifted into the TDI pin appears at the TDO pin after being clocked through the entire boundary-scan register.
Figure 13–9 shows that the instruction code at TDI does not appear at the TDO pin until after the capture register data is shifted out. If TMS is held high on two consecutive TCK clock cycles, the TAP controller advances to the UPDATE_DR state for the update phase.

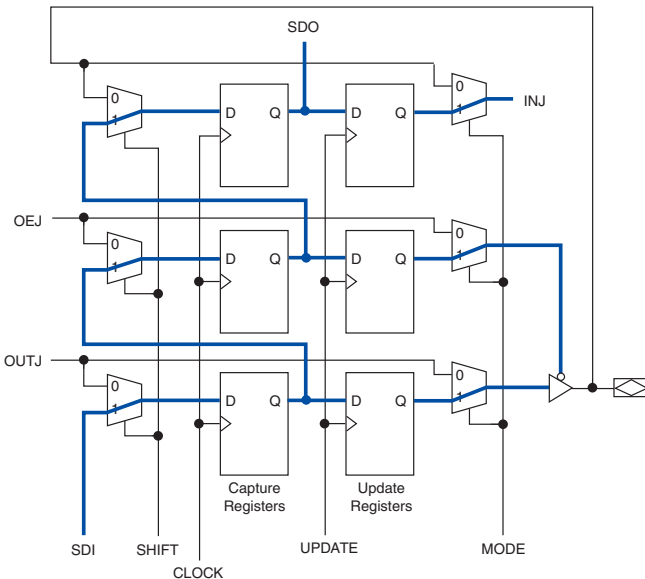*Figure 13–9. SAMPLE/PRELOAD Shift Data Register Waveforms*
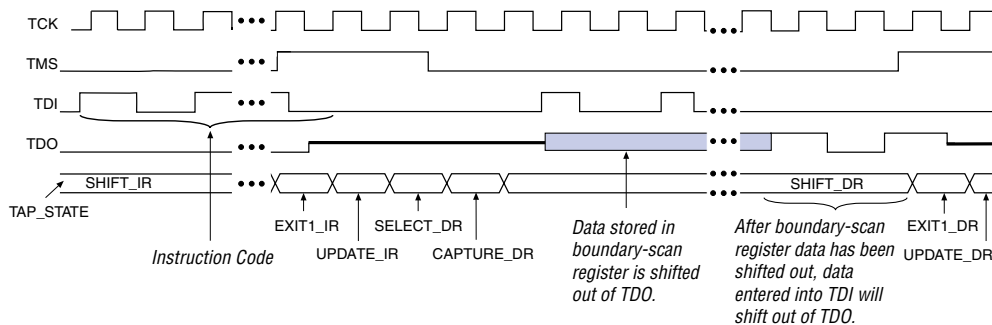


## EXTEST Instruction Mode

The EXTEST instruction mode is used primarily to check external pin connections between devices. Unlike the SAMPLE/PRELOAD mode, EXTEST allows test data to be forced onto the pin signals. By forcing known logic high and low levels on output pins, opens and shorts can be detected at pins of any device in the scan chain.

Figure 13–10 shows the capture, shift, and update phases of the EXTEST mode.

*Figure 13–10. IEEE Std. 1149.1 BST EXTEST Mode*

## Capture Phase

*In the capture phase, the signals at the pin, OEJ and OUTJ, are loaded into the capture registers. The CLOCK signals is supplied by the TAP controller's CLOCKDR output. Previously retained data in the update registers drive the PIN_IN, INJ, and allows the I/O pin to tri-state or drive a signal out.*

*A "1" in the OEJ update register tri-states the output buffer.*

## Shift & Update Phases

*In the shift phase, the previously captured signals at the pin, OEJ and OUTJ, are shifted out of the boundary-scan register via the TDO pin using CLOCK. As data is shifted out, the patterns for the next test can be shifted in via the TDI pin.*

*In the update phase, data is transferred from the capture registers to the update registers using the UPDATE clock. The update registers then drive the PIN_IN, INJ, and allow the I/O pin to tri-state or drive a signal out.*

EXTEST selects data differently than SAMPLE/PRELOAD. EXTEST chooses data from the update registers as the source of the output and output enable signals. Once the EXTEST instruction code is entered, the multiplexers select the update register data. Thus, data stored in these registers from a previous EXTEST or SAMPLE/PRELOAD test cycle can be forced onto the pin signals. In the capture phase, the results of this test data are stored in the capture registers and then shifted out of TDO during the shift phase. New test data can then be stored in the update registers during the update phase.

The EXTEST waveform diagram in Figure 13–11 resembles the SAMPLE/PRELOAD waveform diagram, except for the instruction code. The data shifted out of TDO consists of the data that was present in the capture registers after the capture phase. New test data shifted into the TDI pin appears at the TDO pin after being clocked through the entire boundary-scan register.

*Figure 13–11. EXTEST Shift Data Register Waveforms*
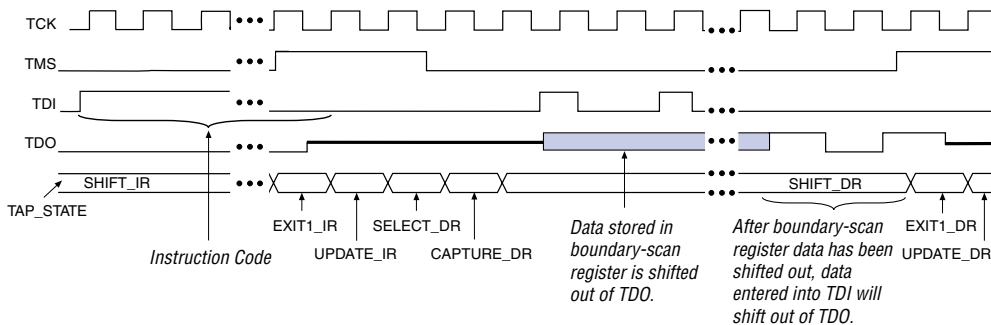


## BYPASS Instruction Mode

The BYPASS mode is activated when an instruction code of all ones is loaded in the instruction register. The waveforms in Figure 13–12 show how scan data passes through a device once the TAP controller is in the SHIFT_DR state. In this state, data signals are clocked into the bypass register from TDI on the rising edge of TCK and out of TDO on the falling edge of the same clock pulse.

*Figure 13–12. BYPASS Shift Data Register Waveforms*
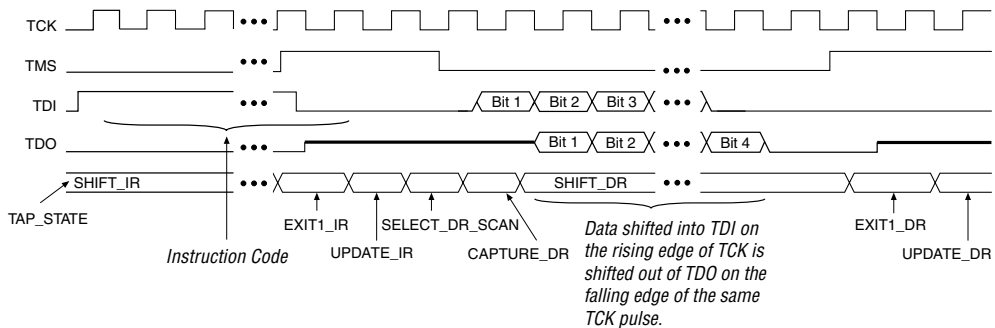


## IDCODE Instruction Mode

The IDCODE instruction mode is used to identify the devices in an IEEE Std. 1149.1 chain. When IDCODE is selected, the device identification register is loaded with the 32-bit vendor-defined identification code. The device ID register is connected between the TDI and TDO ports, and the device IDCODE is shifted out.

For more information on the IDCODE for Arria GX devices refer to the *Configuration and Testing* chapter in volume 1 of the *Arria GX Device Handbook*.

## USERCODE Instruction Mode

The USERCODE instruction mode is used to examine the user electronic signature (UES) within the devices along an IEEE Std. 1149.1 chain. When this instruction is selected, the device identification register is connected between the TDI and TDO ports. The user-defined UES is shifted into the device ID register in parallel from the 32-bit USERCODE register. The UES is then shifted out through the device ID register.

☞ The UES value is not user defined until after the device is configured. Before configuration, the UES value is set to the default value.

### CLAMP Instruction Mode

The CLAMP instruction mode is used to allow the state of the signals driven from the pins to be determined from the boundary-scan register while the bypass register is selected as the serial path between the TDI and TDO ports. The state of all signals driven from the pins are completely defined by the data held in the boundary-scan register.

☞      If you are testing after configuring the device, the programmable weak pull-up resister or the bus hold feature overrides the CLAMP value (the value stored in the update register of the boundary-scan cell) at the pin.

### HIGHZ Instruction Mode

The HIGHZ instruction mode sets all of the user I/O pins to an inactive drive state. These pins are tri-stated until a new JTAG instruction is executed. When this instruction is loaded into the instruction register, the bypass register is connected between the TDI and TDO ports.

☞      If you are testing after configuring the device, the programmable weak pull-up resistor or the bus hold feature overrides the HIGHZ value at the pin.

## I/O Voltage Support in JTAG Chain

The JTAG chain supports several devices. However, you should use caution if the chain contains devices that have different $V_{CCIO}$ levels. The output voltage level of the TDO pin must meet the specifications of the TDI pin it drives. The TDI pin is powered by $V_{CCPD}$ (3.3 V). For Arria GX devices, the $V_{CCIO}$ power supply of bank 4 powers the TDO pin. Table 13–3 shows board design recommendations to ensure proper JTAG chain operation.

You can interface the TDI and TDO lines of the devices that have different $V_{CCIO}$ levels by inserting a level shifter between the devices. If possible, you should build the JTAG chain in such a way that a device with a higher $V_{CCIO}$ level drives to a device with an equal or lower $V_{CCIO}$ level. This way, a level shifter is used only to shift the TDO level to a level acceptable to the JTAG tester. Figure 13–13 shows the JTAG chain of mixed voltages and how a level shifter is inserted in the chain.

*Table 13–3. Supported TDO/TDI Voltage Combinations*

| Device | TDI Input Buffer Power | Arria GX TDO $V_{CCIO}$ Voltage Level in I/O Bank 4 | | | |
|---|---|---|---|---|---|
| | | $V_{CCIO}$ = 3.3 V | $V_{CCIO}$ = 2.5 V | $V_{CCIO}$ = 1.8 V | $V_{CCIO}$ = 1.5 V |
| Arria GX | Always $V_{CCPD}$ (3.3V) | ✓ (1) | ✓ (2) | ✓ (3) | level shifter required |
| Non-Arria GX | VCC = 3.3 V | ✓ (1) | ✓ (2) | ✓ (3) | level shifter required |
| | VCC = 2.5 V | ✓ (1), (4) | ✓ (2) | ✓ (3) | level shifter required |
| | VCC = 1.8 V | ✓ (1), (4) | ✓ (2), (5) | ✓ | level shifter required |
| | VCC = 1.5 V | ✓ (1), (4) | ✓ (2), (5) | ✓ (6) | ✓ |

*Notes to Table 13–3:*
(1)  The TDO output buffer meets $V_{OH}$ (MIN) = 2.4 V.
(2)  The TDO output buffer meets $V_{OH}$ (MIN) = 2.0 V.
(3)  An external 250-Ω pull-up resistor is not required, but recommended if signal levels on the board are not optimal.
(4)  Input buffer must be 3.3-V tolerant.
(5)  Input buffer must be 2.5-V tolerant.
(6)  Input buffer must be 1.8-V tolerant.

*Figure 13–13. JTAG Chain of Mixed Voltages*
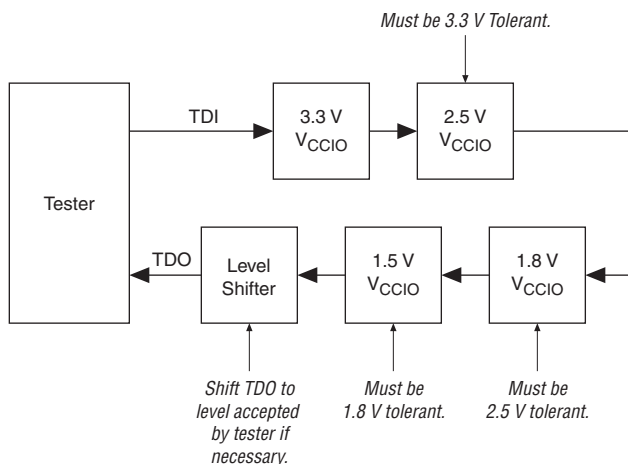
## Using IEEE Std. 1149.1 BST Circuitry

Arria GX devices have dedicated JTAG pins and the IEEE Std. 1149.1 BST circuitry is enabled upon device power-up. Not only can you perform BST on Arria GX FPGAs before and after, but also during configuration. Arria GX FPGAs support the BYPASS, IDCODE and SAMPLE instructions during configuration without interrupting configuration. To send all other JTAG instructions, you must interrupt configuration using the CONFIG_IO instruction.

The CONFIG_IO instruction allows you to configure I/O buffers via the JTAG port, and when issued, interrupts configuration. This instruction allows you to perform board-level testing prior to configuring the Arria GX FPGA or you can wait for the configuration device to complete configuration. Once configuration is interrupted and JTAG BST is complete, you must reconfigure the part via JTAG (PULSE_CONFIG instruction) or by pulsing nCONFIG low.

☞ When you perform JTAG boundary-scan testing before configuration, the nCONFIG pin must be held low.

The chip-wide reset (DEV_CLRn) and chip-wide output enable (DEV_OE) pins on Arria GX devices do not affect JTAG boundary-scan or configuration operations. Toggling these pins does not disrupt BST operation (other than the expected BST behavior).

When you design a board for JTAG configuration of Arria GX devices, you need to consider the connections for the dedicated configuration pins.

For more information on using the IEEE Std.1149.1 circuitry for device configuration, refer to the *Configuring Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*.

## BST for Configured Devices

For a configured device, the input buffers are turned off by default for I/O pins that are set as output only in the design file. You cannot sample on the configured device output pins with the default BSDL file when the input buffers are turned off. You can set the Quartus II software to always enable the input buffers on a configured device so it behaves the same as an unconfigured device for boundary-scan testing, allowing sample function on output pins in the design. This aspect can cause slight increase in standby current because the unused input buffer is always on. In the Quartus II software, do the following:

1. Choose **Settings** (Assignments menu).

2. Click **Assembler**.

3.   Turn on **Always Enable Input Buffers**.

## Disabling IEEE Std. 1149.1 BST Circuitry

The IEEE Std. 1149.1 BST circuitry for Arria GX devices is enabled upon device power-up. Because the IEEE Std. 1149.1 BST circuitry is used for BST or in-circuit reconfiguration, you must enable the circuitry only at specific times as mentioned in, "Using IEEE Std. 1149.1 BST Circuitry" on page 13–19.

☞   If you are not using the IEEE Std. 1149.1 circuitry in Arria GX, then you should permanently disable the circuitry to ensure that you do not inadvertently enable when it is not required.

Table 13–4 shows the pin connections necessary for disabling the IEEE Std. 1149.1 circuitry in Arria GX devices.

*Table 13–4. Disabling IEEE Std. 1149.1 Circuitry*

| JTAG Pins *(1)* | Connection for Disabling |
|---|---|
| TMS | $V_{CC}$ |
| TCK | GND |
| TDI | $V_{CC}$ |
| TDO | Leave open |
| TRST | GND |

*Note to Table 13–4:*
(1)   There is no software option to disable JTAG in Arria GX devices. The JTAG pins are dedicated.

## Guidelines for IEEE Std. 1149.1 Boundary-Scan Testing

Use the following guidelines when performing boundary-scan testing with IEEE Std. 1149.1 devices:

■   If the "10..." pattern does not shift out of the instruction register via the TDO pin during the first clock cycle of the SHIFT_IR state, the TAP controller did not reach the proper state. To solve this problem, try one of the following procedures:

●   Verify that the TAP controller has reached the SHIFT_IR state correctly. To advance the TAP controller to the SHIFT_IR state, return to the RESET state and send the code 01100 to the TMS pin.
●   Check the connections to the $V_{CC}$, GND, JTAG, and dedicated configuration pins on the device.

■ Perform a SAMPLE/PRELOAD test cycle prior to the first EXTEST test cycle to ensure that known data is present at the device pins when you enter the EXTEST mode. If the OEJ update register contains a 0, the data in the OUTJ update register is driven out. The state must be known and correct to avoid contention with other devices in the system.

■ Do not perform EXTEST testing during ICR. This instruction is supported before or after ICR, but not during ICR. Use the CONFIG_IO instruction to interrupt configuration and then perform testing, or wait for configuration to complete.

■ If performing testing before configuration, hold nCONFIG pin low.

■ After configuration, any pins in a differential pin pair cannot be tested. Therefore, performing BST after configuration requires editing of BSC group definitions that correspond to these differential pin pairs. The BSC group should be redefined as an internal cell.

☞ Refer to the Boundary-Scan Description Language (BSDL) file for more information on editing.

■ The following private instructions must not be used as invoking, such instructions potentially damage the device rendering the device useless:

1100010000
0011001001
0000101001
0000010000

☞ You should take precaution not to invoke such instructions at any instance. Contact Altera® Applications if you need to use these instructions.

🐾 For more information on boundary scan testing, contact Altera Applications Group.

# Boundary-Scan Description Language (BSDL) Support

The Boundary-Scan Description Language (BSDL), a subset of VHDL, provides a syntax that allows you to describe the features of an IEEE Std. 1149.1 BST-capable device that can be tested. Test software development systems then use the BSDL files for test generation, analysis, and failure diagnostics.

🐾 For more information, or to receive BSDL files for IEEE Std. 1149.1-compliant Arria GX devices, visit the Altera web site at www.altera.com.

## Conclusion

The IEEE Std. 1149.1 BST circuitry available in Arria GX devices provides a cost-effective and efficient way to test systems that contain devices with tight lead spacing. Circuit boards with Altera and other IEEE Std. 1149.1-compliant devices can use the EXTEST, SAMPLE/PRELOAD, and BYPASS modes to create serial patterns that internally test the pin connections between devices and check device operation.

## References

Bleeker, H., P. van den Eijnden, and F. de Jong. *Boundary-Scan Test: A Practical Approach*. Eindhoven, The Netherlands: Kluwer Academic Publishers, 1993.

Institute of Electrical and Electronics Engineers, Inc. *IEEE Standard Test Access Port and Boundary-Scan Architecture* (IEEE Std 1149.1-2001). New York: Institute of Electrical and Electronics Engineers, Inc., 2001.

Maunder, C. M., and R. E. Tulloss. *The Test Access Port and Boundary-Scan Architecture*. Los Alamitos: IEEE Computer Society Press, 1990.

## Referenced Documents

This chapter references the following documents:

■ *Configuring Arria GX Devices* chapter in volume 2 of the *Arria GX Device Handbook*
■ *Configuration & Testing* chapter in volume 1 of the *Arria GX Device Handbook*

## Document Revision History

Table 13–5 shows the revision history for this chapter.

| Table 13–5. Document Revision History | | |
|---|---|---|
| **Date and Document Version** | **Changes Made** | **Summary of Changes** |
| May 2008, v1.2 | Minor text edits. | — |
| August 2007, v1.1 | Added the "Referenced Documents" section. | — |
| May 2007, v1.0 | Initial Release | N/A |

# Section VII. PCB Layout Guidelines

This section provides information for board layout designers to successfully layout their boards for Arria™ GX devices. These chapters contain the required PCB layout guidelines and package specifications.

This section contains the following chapters:

■ Chapter 14, Package Information for Arria GX Devices

## Revision History

Refer to each chapter for its own specific revision history. For information on when each chapter was updated, refer to the Chapter Revision Dates section, which appears in the full handbook.

# 14. Package Information for Arria GX Devices

## Introduction

This chapter provides package information for Altera® Arria™ GX devices, including:

■ Device and package cross reference
■ Thermal resistance values
■ Package outlines

Tables 14–1 shows which Altera Arria GX devices, respectively, are available in FineLine BGA® (FBGA) packages.

| Table 14–1. Arria GX Devices in FBGA Packages | | |
|---|---|---|
| **Device** | **Package** | **Pins** |
| EP1AGX20 | Flip-chip FBGA | 484 |
| | Flip-chip FBGA | 780 |
| EP1AGX35 | Flip-chip FBGA | 484 |
| | Flip-chip FBGA | 780 |
| EP1AGX50 | Flip-chip FBGA | 484 |
| | Flip-chip FBGA | 780 |
| | Flip-chip FBGA | 1152 |
| EP1AGX60 | Flip-chip FBGA | 484 |
| | Flip-chip FBGA | 780 |
| | Flip-chip FBGA | 1152 |
| EP1AGX90 | Flip-chip FBGA | 1152 |

## Thermal Resistance

Thermal resistance values for Arria GX devices are provided for a board that meets JDEC specifications and for a typical board. The following values are provided:

- $\theta_{JA}$ (°C/W) still air—Junction-to-ambient thermal resistance with no air flow when a heat sink is not used.
- $\theta_{JA}$ (°C/W) 100 ft./min.—Junction-to-ambient thermal resistance with 100 ft./min. airflow when a heat sink is not used.
- $\theta_{JA}$ (°C/W) 200 ft./min.—Junction-to-ambient thermal resistance with 200 ft./min. airflow when a heat sink is not used.
- $\theta_{JA}$ (°C/W) 400 ft./min.—Junction-to-ambient thermal resistance with 400 ft./min. airflow when a heat sink is not used.
- $\theta_{JC}$—Junction-to-case thermal resistance for device.
- $\theta_{JB}$—Junction-to-board thermal resistance for device.

Table 14–2 provides $\theta_{JA}$ (junction-to-ambient thermal resistance), $\theta_{JC}$ (junction-to-case thermal resistance), and $\theta_{JB}$ (junction-to-board thermal resistance) values for Arria GX devices.

*Table 14–2. Arria GX GX Device Thermal Resistance*

| Device | Pin Count | Package | $\theta_{JA}$ (° C/W) Still Air | $\theta_{JA}$ (° C/W) 100 ft./min. | $\theta_{JA}$ (° C/W) 200 ft./min. | $\theta_{JA}$ (° C/W) 400 ft./min. | $\theta_{JC}$ (° C/W) | $\theta_{JB}$ (° C/W) |
|---|---|---|---|---|---|---|---|---|
| EP1AGX20 | 484 | FBGA | 12.8 | 10.3 | 8.7 | 7.5 | 0.3 | 3.14 |
| | 780 | FBGA | 11.1 | 8.6 | 7.2 | 6.0 | 0.24 | 3.14 |
| EP1AGX35 | 484 | FBGA | 12.8 | 10.3 | 8.7 | 7.5 | 0.3 | 3.14 |
| | 780 | FBGA | 11.1 | 8.6 | 7.2 | 6.0 | 0.24 | 3.14 |
| EP1AGX50 | 484 | FBGA | 12.7 | 10.2 | 8.6 | 7.3 | 0.2 | 2.86 |
| | 780 | FBGA | 10.9 | 8.4 | 6.9 | 5.8 | 0.15 | 2.84 |
| | 1152 | FBGA | 9.9 | 7.5 | 6.1 | 5.0 | 0.15 | 2.5 |
| EP1AGX60 | 484 | FBGA | 12.7 | 10.2 | 8.6 | 7.3 | 0.2 | 2.86 |
| | 780 | FBGA | 10.9 | 8.4 | 6.9 | 5.8 | 0.15 | 2.84 |
| | 1152 | FBGA | 9.9 | 7.5 | 6.1 | 5.0 | 0.15 | 2.5 |
| EP1AGX90 | 1152 | FBGA | 9.6 | 7.3 | 5.9 | 4.9 | 0.11 | 2.33 |

This chapter contains the following section:

- "Package Outlines" on page 14–3

# Package Outlines

The package outlines are listed in order of ascending pin count. Altera package outlines meet the requirements of *JEDEC Publication No. 95.*

## 484-Pin FBGA - Flip Chip

■ All dimensions and tolerances conform to ASME Y14.5M – 1994.
■ Controlling dimension is in millimeters.
■ Pin A1 may be indicated by an ID dot, or a special feature, in its proximity on the package surface.

Tables 14–3 and 14–4 show the package information and package outline figure references, respectively, for the 484-pin FBGA packaging.

| *Table 14–3. 484-Pin FBGA Package Information* | |
|---|---|
| **Description** | **Specification** |
| Ordering code reference | F |
| Package acronym | FBGA |
| Substrate material | BT |
| Solder ball composition | Regular: 63Sn:37Pb (Typ.) Pb-free: Sn:3Ag:0.5Cu (Typ.) |
| JEDEC outline reference | MS-034 variation: AAJ-1 |
| Maximum lead coplanarity | 0.008 inches (0.20 mm) |
| Weight | 7.1 g |
| Moisture sensitivity level | Printed on moisture barrier bag |

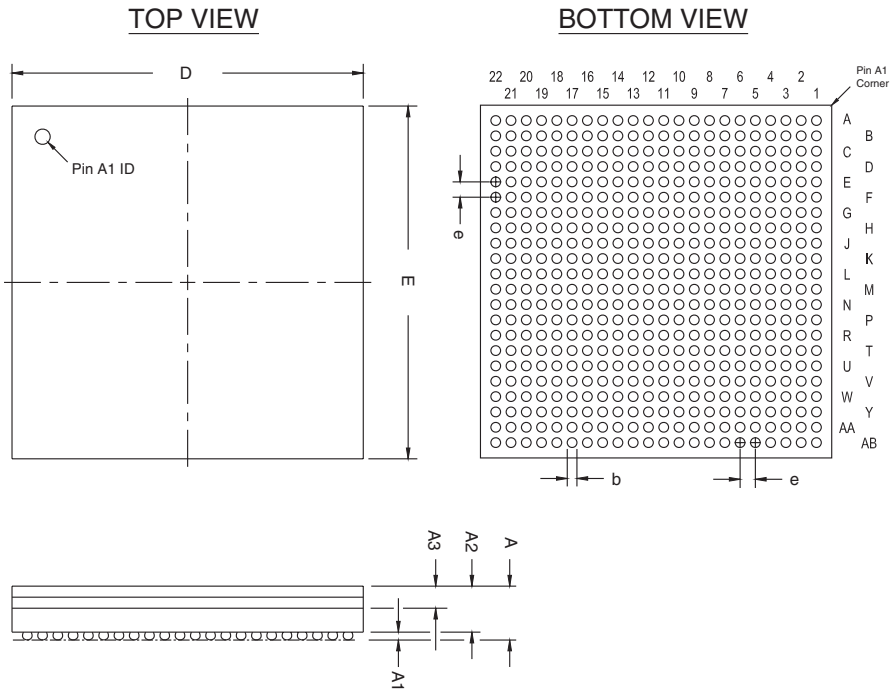| *Table 14–4. 484-Pin FBGA Package Outline Dimensions  (Part 1 of 2)* | | | |
|---|---|---|---|
| **Symbol** | **Millimeter** | | |
| | **Min.** | **Nom.** | **Max.** |
| A | – | – | 3.50 |
| A1 | 0.30 | – | – |
| A2 | 0.25 | – | 3.00 |
| A3 | – | – | 2.50 |
| D | 23.00 BSC | | |
| E | 23.00 BSC | | |

| Table 14–4. 484-Pin FBGA Package Outline Dimensions (Part 2 of 2) | | | |
|---|---|---|---|
| **Symbol** | **Millimeter** | | |
| | **Min.** | **Nom.** | **Max.** |
| b | 0.50 | 0.60 | 0.70 |
| e | 1.00 BSC | | |

Figure 14–1 shows a package outline for the 484-pin FineLine BGA packaging.

*Figure 14–1. 484-Pin FBGA Package Outline*

## 780-Pin FBGA - Flip Chip

■ Arria GX Device Handbook, Volume 2All dimensions and tolerances conform to ASME Y14.5M - 1994.
■ Controlling dimension is in millimeters.
■ Pin A1 may be indicated by an ID dot, or a special feature, in its proximity on package surface.

Tables 14–5 and 14–6 show the package information and package outline figure references, respectively, for the 780-pin FBGA packaging.

**Table 14–5. 780-Pin FBGA Package Information**

| Description | Specification |
|---|---|
| Ordering code reference | F |
| Package acronym | FBGA |
| Substrate material | BT |
| Solder ball composition | Regular:  63Sn:37Pb (Typ.)<br>Pb-free:  Sn:3Ag:0.5Cu  (Typ.) |
| JEDEC outline reference | MS-034    variation: AAM-1 |
| Maximum lead coplanarity | 0.008 inches (0.20 mm) |
| Weight | 11.9 g |
| Moisture Sensitivity Level | Printed on moisture barrier bag |

**Table 14–6. 780-Pin FBGA Package Outline Dimensions**

| Symbol | Millimeters | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | – | – | 3.50 |
| A1 | 0.30 | – | – |
| A2 | 0.25 | – | 3.00 |
| A3 | – | – | 2.50 |
| D | 29.00 BSC | | |
| E | 29.00 BSC | | |
| b | 0.50 | 0.60 | 0.70 |
| e | 1.00 BSC | | |

Figure 14–2 shows a package outline for the 780-pin FineLine BGA packaging.

*Figure 14–2. 780-Pin FBGA Package Outline*
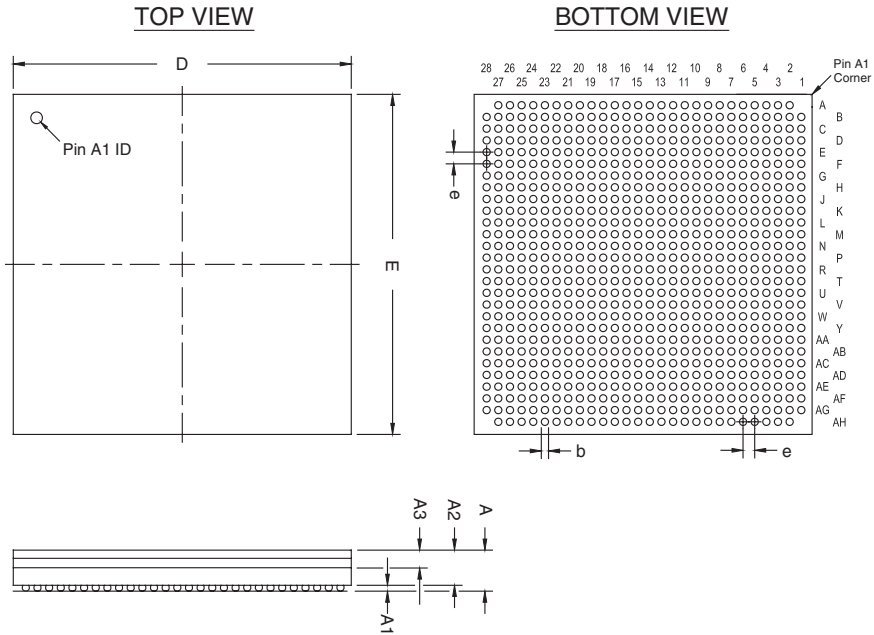
## 1,152-Pin FBGA - Flip Chip

■ All dimensions and tolerances conform to ASME Y14.5M - 1994.
■ Controlling dimension is in millimeters.
■ Pin A1 may be indicated by an ID dot, or a special feature, in its proximity on package surface.

Tables 14–7 and 14–8 show the package information and package outline figure references, respectively, for the 1,152-pin FBGA packaging.

| Table 14–7. 1,152-Pin FBGA Package Information | |
|---|---|
| **Description** | **Specification** |
| Ordering code reference | F |
| Package acronym | FBGA |
| Substrate material | BT |
| Solder ball composition | Regular: 63Sn:37Pb (Typ.)<br>Pb-free: Sn:3Ag:0.5Cu (Typ.) |
| JEDEC outline reference | MS-034    variation: AAR-1 |
| Maximum lead coplanarity | 0.008 inches (0.20 mm) |
| Weight | 15.8 g |
| Moisture sensitivity level | Printed on moisture barrier bag |

| Table 14–8. 1,152-Pin FBGA Package Outline Dimensions | | | |
|---|---|---|---|
| **Symbol** | **Millimeters** | | |
| | **Min.** | **Nom.** | **Max.** |
| A | – | – | 3.50 |
| A1 | 0.30 | – | – |
| A2 | 0.25 | – | 3.00 |
| A3 | – | – | 2.50 |
| D | 35.00 BSC | | |
| E | 35.00 BSC | | |
| b | 0.50 | 0.60 | 0.70 |
| e | 1.00 BSC | | |

Figure 14–3 shows a package outline for the 1,152-pin FineLine BGA packaging.

*Figure 14–3. 1,152-Pin FBGA Package Outline*
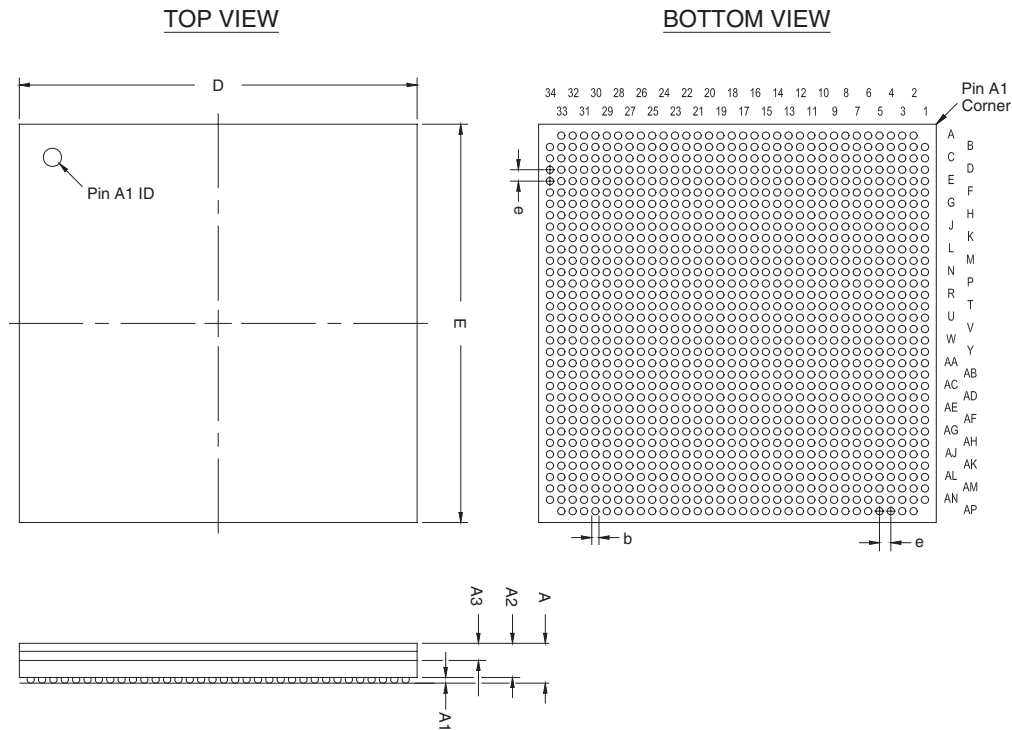


## Document Revision History

Table 14–9 shows the revision history for this document.

*Table 14–9. Document Revision History*

| Date and Document Version | Changes Made | Summary of Changes |
|---|---|---|
| May 2007, v1.1 | Minor text edits. | — |
| May 2007, v1.0 | Initial Release | N/A |