Arria V Device Handbook

Volume 1: Device Interfaces and Integration

101 Innovation Drive
San Jose, CA 95134
www.altera.com

AV-5V2-2.0

Contents

Chapter 3. Variable-Precision DSP Blocks in Arria V Devices

Chapter 4. Clock Networks and PLLs in Arria V Devices

Chapter 6. High-Speed Differential I/O Interfaces and DPA in Arria V Devices

Chapter 7. External Memory Interfaces in Arria V Devices

Section III. System Integration for Arria V Devices

Chapter 8. Configuration, Design Security, and Remote System Upgrades in Arria V Devices

Chapter 9. SEU Mitigation in Arria V Devices

Chapter 10. JTAG Boundary-Scan Testing in Arria V Devices

Chapter 11. Power Management in Arria V Devices

Additional Information

Chapter Revision Dates

The chapters in this document, Volume 1: Device Interfaces and Integration, were revised on the following dates. Where chapters or groups of chapters are available separately, part numbers are listed.

Part Number:

Chapter 1. Logic Array Blocks and Adaptive Logic Modules in Arria V Devices
Revised: *June 2012*
Part Number: *AV-52001-2.0*

Chapter 2. Memory Blocks in Arria V Devices
Revised: *June 2012*
Part Number: *AV-52002-2.0*

Chapter 3. Variable-Precision DSP Blocks in Arria V Devices
Revised: *June 2012*
Part Number: *AV-52003-2.0*

Chapter 4. Clock Networks and PLLs in Arria V Devices
Revised: *June 2012*
Part Number: *AV-52004-2.0*

Chapter 5. I/O Features in Arria V Devices
Revised: *June 2012*
Part Number: *AV52005-2.0*

Chapter 6. High-Speed Differential I/O Interfaces and DPA in Arria V Devices
Revised: *June 2012*
Part Number: *AV52006-2.0*

Chapter 7. External Memory Interfaces in Arria V Devices
Revised: *June 2012*
Part Number: *AV52007-2.0*

Chapter 8. Configuration, Design Security, and Remote System Upgrades in Arria V Devices
Revised: *June 2012*
Part Number: *AV-52008-2.0*

Chapter 9. SEU Mitigation in Arria V Devices
Revised: *June 2012*
Part Number: *AV-52009-2.0*

Chapter 10. JTAG Boundary-Scan Testing in Arria V Devices
Revised: *June 2012*
Part Number: *AV-52010-2.0*

Chapter 11. Power Management in Arria V Devices
Revised: *June 2012*
Part Number: *AV-52011-2.0*

Section I. Device Core for Arria V Devices

This section provides a complete overview of all features relating to the Arria® V device family. This section includes the following chapters:

- Chapter 1, Logic Array Blocks and Adaptive Logic Modules in Arria V Devices
- Chapter 2, Memory Blocks in Arria V Devices
- Chapter 3, Variable-Precision DSP Blocks in Arria V Devices
- Chapter 4, Clock Networks and PLLs in Arria V Devices

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 this volume.

1. Logic Array Blocks and Adaptive Logic Modules in Arria V Devices

AV-52001-2.0

This chapter describes the features of the logic array block (LAB) in the Arria® V core fabric.

The LAB is composed of basic building blocks known as adaptive logic modules (ALMs) that you can configure to implement logic functions, arithmetic functions, and register functions.

You can use a quarter of the available LABs in Arria V devices as a memory LAB (MLAB).

The Quartus® II software and other supported third-party synthesis tools, in conjunction with parameterized functions such as the library of parameterized modules (LPM), automatically choose the appropriate mode for common functions such as counters, adders, subtractors, and arithmetic functions.

This chapter contains the following sections:

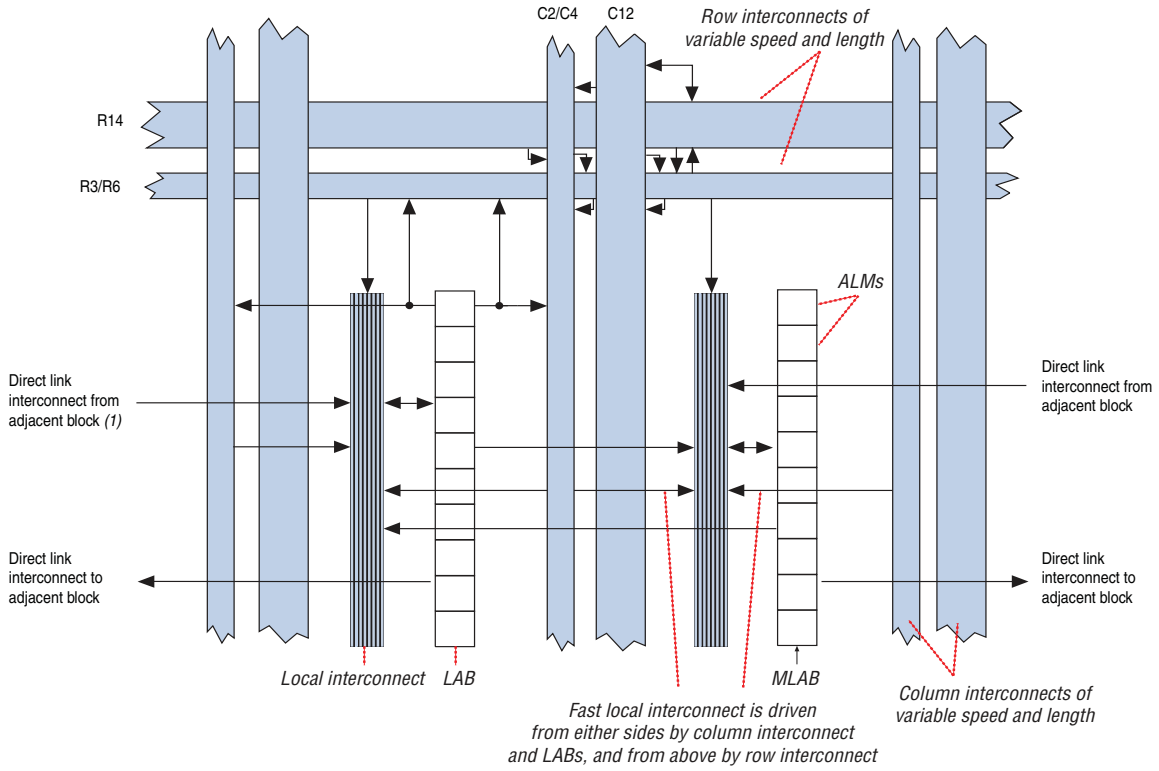
LAB

The LABs are configurable logic blocks that consist of a group of logic resources. Each LAB contains dedicated logic for driving control signals to its ALMs.

MLAB is a superset of the LAB and includes all the LAB features.

Figure 1–1 shows an overview of the Arria V LAB and MLAB structure with the LAB interconnects.

Figure 1–1. LAB Structure and Interconnects Overview in Arria V Devices



Note to Figure 1–1:

(1) Connects to adjacent LABs, memory blocks, digital signal processing (DSP) blocks, or I/O element (IOE) outputs.

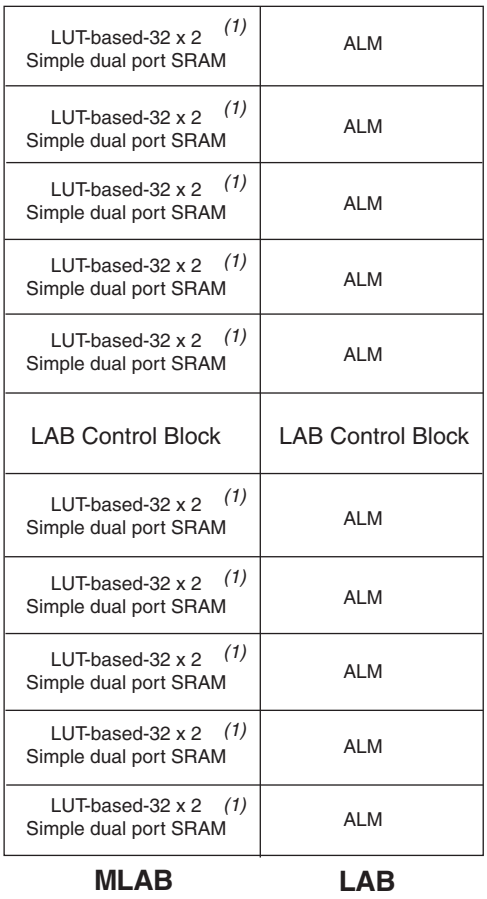
MLAB

Each MLAB supports a maximum of 640 bits of simple dual-port SRAM.

You can configure each ALM in an MLAB as a 32 x 2 memory block, resulting in a configuration of 32 x 20 simple dual-port SRAM blocks.

Figure 1–2 shows the LAB and MLAB topology.

Figure 1–2. LAB and MLAB Structure for Arria V Devices



Note to Figure 1–2:

(1) You can use an MLAB ALM as a regular LAB ALM or configure it as a dual-port SRAM.

Interconnects

Each LAB can drive 30 ALMs through fast-local and direct-link interconnects. Ten ALMs are in any given LAB and ten ALMs are in each of the adjacent LABs.

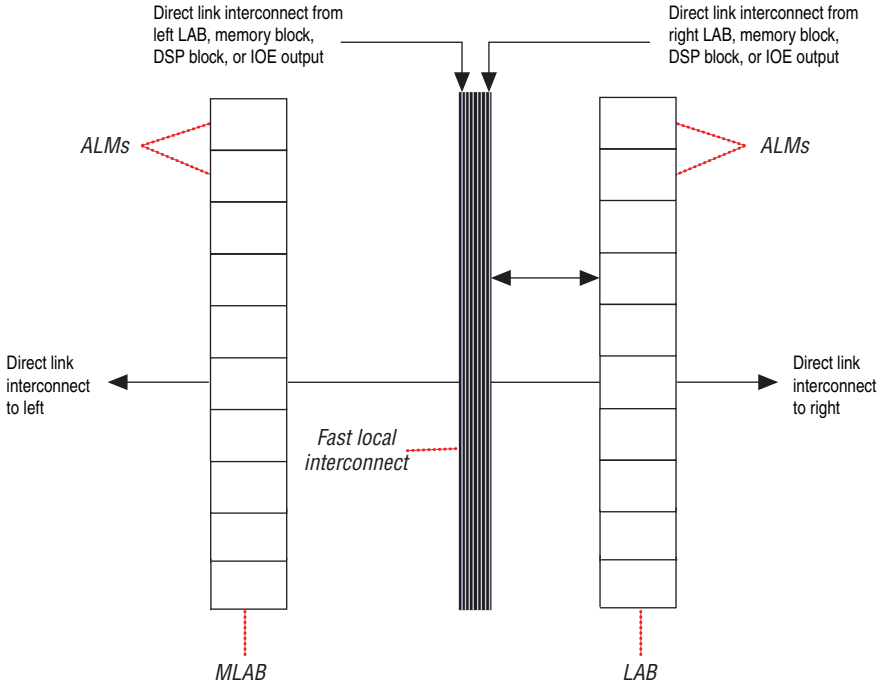
The local interconnect can drive ALMs in the same LAB using column and row interconnects and ALM outputs in the same LAB.

Neighboring LABs, MLABs, M10K blocks, or digital signal processing (DSP) blocks from the left or right can also drive the LAB's local interconnect using the direct link connection.

The direct link connection feature minimizes the use of row and column interconnects, providing higher performance and flexibility.

Figure 1–3 shows the LAB fast-local and direct-link interconnects.

Figure 1–3. Direct Link and Fast Local Interconnects for Arria V Devices



LAB Control Signals

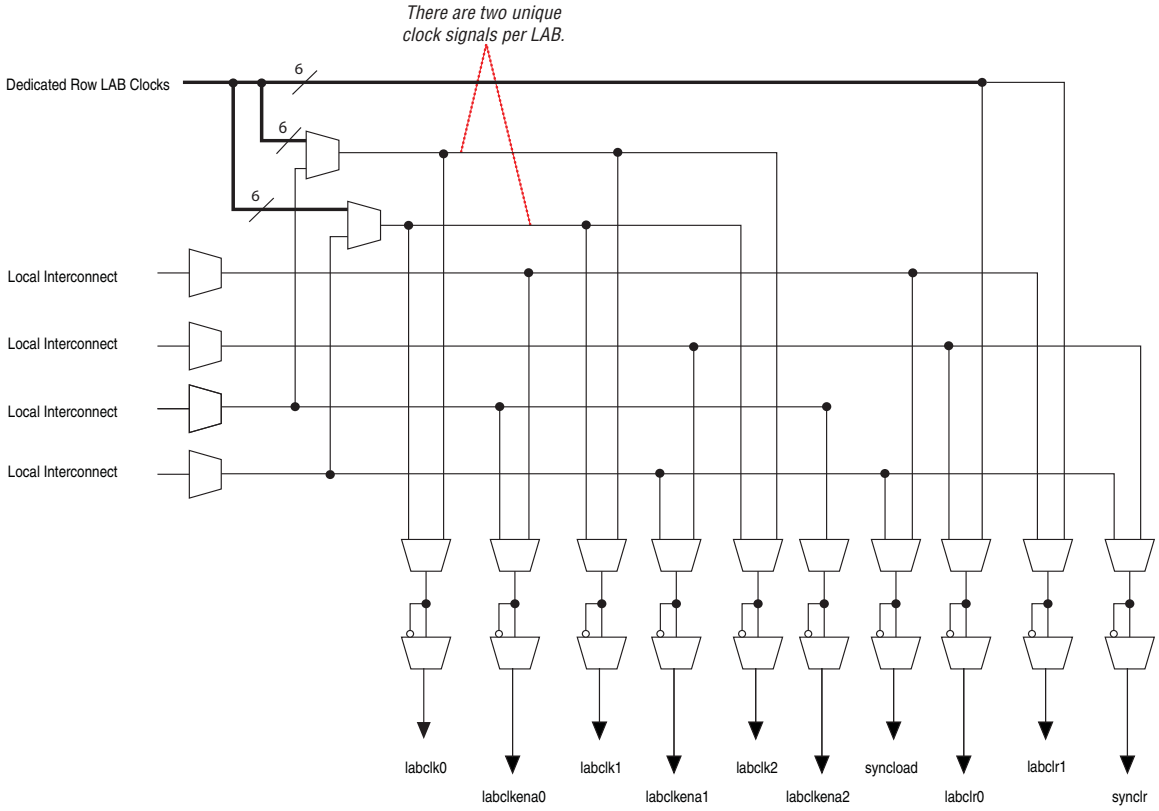
Each LAB contains dedicated logic for driving the control signals to its ALMs, and has two unique clock sources and three clock enable signals.

The LAB control block generates up to three clocks using the two clock sources and three clock enable signals. Each clock and the clock enable signals are linked.

De-asserting the clock enable signal turns off the corresponding LAB-wide clock.

Figure 1–4 shows the clock sources and clock enable signals in an LAB.

Figure 1–4. LAB-Wide Control Signals for Arria V Devices *(1)*



Note to Figure 1–4:

(1) For more information, refer to Figure 1–6 on page 1–8.

ALM Registers

One ALM contains four programmable registers. Each register has data, clock, synchronous and asynchronous clear, and synchronous load functions.

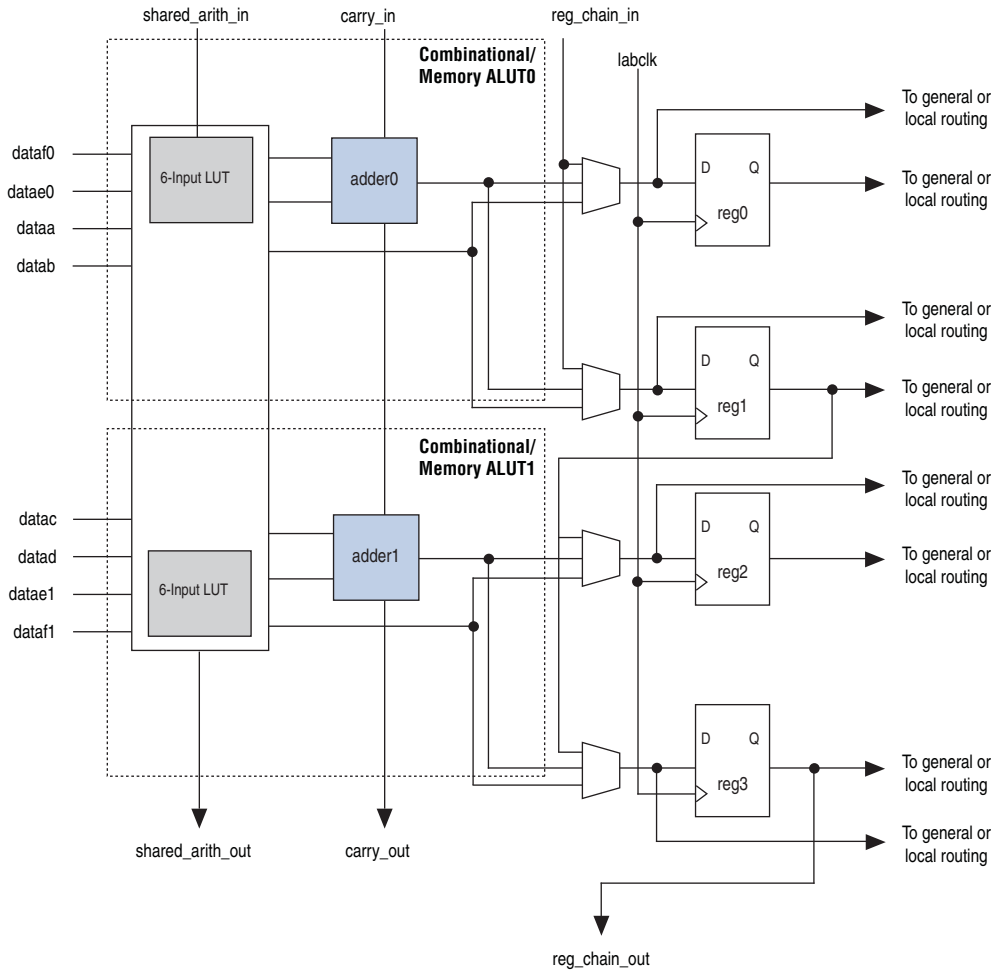
Global signals, general-purpose I/O (GPIO) pins, or any internal logic can drive the clock and clear control signals of an ALM register.

GPIO pins or internal logic drives the clock enable signal.

For combinational functions, the registers are bypassed and the output of the look-up table (LUT) drives directly to the outputs of an ALM.

Figure 1–5 shows a high-level block diagram of the Arria V ALM.

Figure 1–5. High-Level Block Diagram of the Arria V ALM



The Quartus II software automatically configures the ALMs for optimized performance.

ALM Outputs

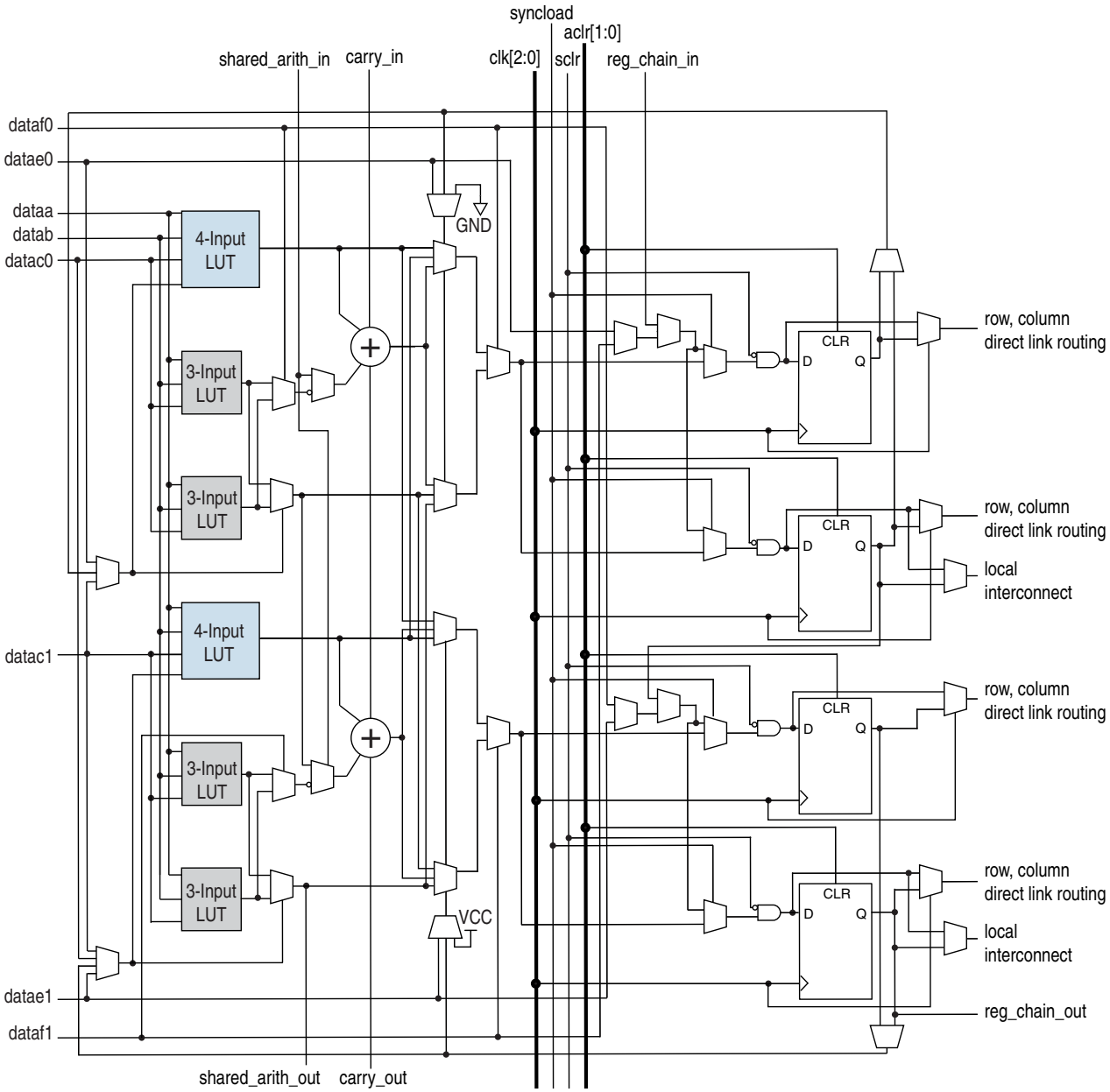
The LUT, adder, or register output can drive the ALM outputs. There are two sets of outputs—general routing outputs and register chain outputs.

For each set of output drivers, two ALM outputs can drive column, row, or direct link routing connections, and one of these ALM outputs can also drive local interconnect resources. The LUT or adder can drive one output while the register drives another output.

Register packing improves device utilization by allowing unrelated register and combinational logic to be packed into a single ALM. Another mechanism to improve fitting is to allow the register output to feed back into the look-up table (LUT) of the same ALM so that the register is packed with its own fan-out LUT. The ALM can also drive out registered and unregistered versions of the LUT or adder output.

Figure 1–6 shows a detailed view of all the connections in an ALM.

Figure 1–6. ALM Connection Details for Arria V Devices



ALM Operating Modes

The Arria V ALM operates in any of the following modes:

- Normal Mode
- Extended LUT Mode
- Arithmetic Mode
- Shared Arithmetic Mode

Normal 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 V ALM, or a single function of up to six inputs.

The ALM can support certain combinations of completely independent functions and various combinations of functions that have common inputs.

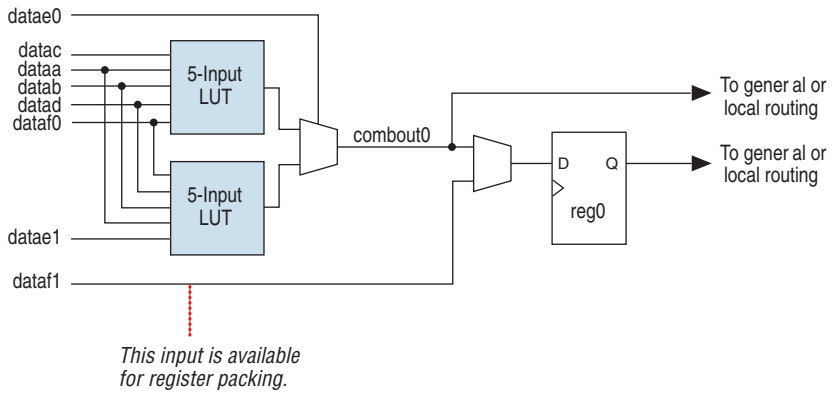
Extended LUT Mode

In this mode, if the 7-input function is unregistered, the unused eighth input is available for register packing.

Functions that fit into the template, as shown in Figure 1–7, often appear in designs as "if-else" statements in Verilog HDL or VHDL code.

Figure 1–7 shows the template of supported 7-input functions using extended LUT mode.

Figure 1–7. Template for Supported 7-Input Functions in Extended LUT Mode in Arria V Devices



Arithmetic Mode

The ALM in arithmetic mode uses two sets of two 4-input LUTs along with two dedicated full adders.

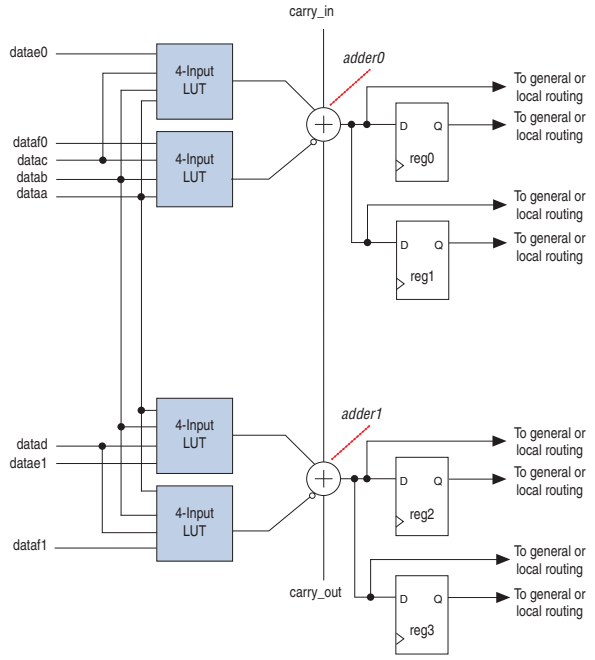
The dedicated adders allow the LUTs to perform pre-adder logic; therefore, each adder can add the output of two 4-input functions.

The ALM supports simultaneous use of the adder's carry output along with combinational logic outputs. The adder output is ignored in this operation.

Using the adder with the combinational logic output provides resource savings of up to 50% for functions that can use this mode.

Figure 1–8 shows an ALM in arithmetic mode.

Figure 1–8. ALM in Arithmetic Mode for Arria V Devices



Carry Chain

The carry chain provides a fast carry function between the dedicated adders in arithmetic or shared arithmetic mode.

The two-bit carry select feature in Arria V devices halves the propagation delay of carry chains within the ALM. 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.

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. This leaves the other half of the ALMs in the LAB available for implementing narrower fan-in functions in normal mode. Carry chains that use the top five ALMs in the first LAB carry into the top half of the ALMs in the next LAB in the column. Carry chains that use the bottom five ALMs in the first LAB carry into the bottom half of the ALMs in the next LAB within the column. You can bypass the top-half of the LAB columns and bottom-half of the MLAB columns.

The Quartus II Compiler creates carry chains longer than 20 ALMs (10 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 the TriMatrix memory and DSP blocks. A carry chain can continue as far as a full column.

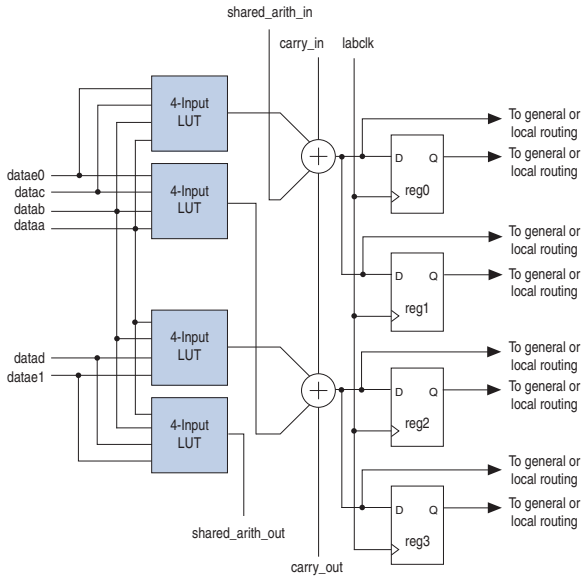
Shared Arithmetic Mode

The ALM in shared arithmetic mode can implement a 3-input add in the ALM.

This mode configures the ALM 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 using a dedicated connection called the shared arithmetic chain.

Figure 1–9 shows the ALM using this feature.

Figure 1–9. ALM in Shared Arithmetic Mode for Arria V Devices



Shared Arithmetic Chain

The shared arithmetic chain available in enhanced arithmetic mode allows the ALM to implement a 3-input adder. This significantly reduces the resources necessary to implement large adder trees or correlator functions.

Similar to carry chains, the top and bottom half of the shared arithmetic chains in alternate LAB columns can be bypassed. This capability allows the shared arithmetic chain to cascade through half of the ALMs in an LAB while leaving the other half available for narrower fan-in functionality. In every LAB column is top-half bypassable; while in MLAB columns are bottom-half bypassable.

The shared arithmetic chain can begin in either the first or sixth ALM in an LAB. The Quartus II Compiler creates shared arithmetic chains longer than 20 ALMs (10 ALMs in arithmetic or shared arithmetic mode) by linking LABs together automatically. To enhanced fitting, a long shared arithmetic chain runs vertically, allowing fast horizontal connections to the TriMatrix memory and DSP blocks. A shared arithmetic chain can continue as far as a full column.

Document Revision History

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

Table 1–1. Document Revision History

Date	Version	Changes
June 2012	2.0	Updated for the Quartus II software v12.0 release: ■ Restructured chapter. ■ Updated Figure 1–6.
November 2011	1.1	Restructured chapter.
May 2011	1.0	Initial release.

2. Memory Blocks in Arria V Devices

AV-52002-2.0

This chapter describes the embedded memory blocks in Arria® V devices.

The memory blocks in Arria V devices provide different sizes of embedded SRAM to fit your design requirements.

This chapter contains the following sections:

Memory Types

The Arria V devices contain two types of memory blocks:

- M10K blocks—10-kilobit (Kb) blocks of dedicated memory resources that you can use to create designs with large memory configurations.
- Memory logic array blocks (MLABs)—640-bit enhanced memory blocks that are configured from dual-purpose logic array blocks (LABs). The MLABs are optimized for implementation of shift registers for digital signal processing (DSP) applications, wide shallow FIFO buffers, and filter delay lines. Each MLAB is made up of ten adaptive logic modules (ALMs) that you can configure as ten 32 x 2 blocks, giving you one 32 x 20 simple dual-port SRAM block per MLAB.

For information about the embedded memory capacity available in each Arria V device, refer to the *Arria V Device Overview*.

Memory Features

Table 2–1 summarizes the features supported by the memory blocks.

Table 2–1. Memory Features in Arria V Devices

Feature	M10K	MLAB
Maximum operating frequency	400 MHz	500 MHz
Total RAM bits (including parity bits)	10,240	640
Configuration (depth × width)	256 x 32, 256 x 40, 512 x 16, 512 x 20, 1K x 8, 1K x 10, 2K x 4, 2K x 5, 4K x 2, and 8K x 1	32 x 16, 32 x 18, and 32 x 20
Parity bits	Yes	Yes
Byte enable	Yes	Yes
Packed mode	Yes	—
Address clock enable	Yes	Yes
Memory modes	■ Single-port memory ■ Simple dual-port memory ■ True dual-port memory ■ Embedded shift register ■ ROM ■ FIFO buffer	■ Single-port memory ■ Simple dual-port memory ■ Embedded shift register ■ ROM ■ FIFO buffer
Simple dual-port mixed width	Yes	—
True dual-port mixed width	Yes	—
FIFO buffer mixed width	Yes	—
Memory Initialization File (**.mif**)	Yes	Yes
Mixed-clock mode	Yes	Yes
Power-up state	Output ports are cleared.	■ Registered output ports—Cleared. ■ Unregistered output ports—Read memory contents.
Asynchronous clears	Output registers	
Write/Read operation triggering	Rising clock edges	
Same-port read-during-write	Output ports set to “new data” or “don’t care”. (The “don’t care” mode applies only for the single-port RAM mode.)	Output ports set to “don’t care”.
Mixed-port read-during-write	Output ports set to “old data” or “don’t care”.	Output ports set to “old data”, “new data”, “don’t care”, or “constrained don’t care”.
ECC support	Soft IP support using the Quartus® II software.	
Fully synchronous memory	Yes	Yes
Asynchronous memory	—	Only for flow-through read memory operations.

Memory Modes

Table 2–2 lists and describes the memory modes that are supported in the Arria V memory blocks.

To avoid corrupting the memory contents, do not violate the setup or hold time on any of the memory block input registers during read or write operations. This is applicable if you use the memory blocks in single-port RAM, simple dual-port RAM, true dual-port RAM, or ROM mode.

Table 2–2. Memory Modes Supported in the Memory Blocks

Memory Mode	Description and Additional Information
Single-port RAM	You can perform only one read or one write operation at a time. Use the read enable port to control the RAM output ports behavior during a write operation: ■ To retain the previous values that are held during the most recent active read enable—Create a read-enable port and perform the write operation with the read enable port deasserted. ■ To show the new data being written, the old data at that address, or a "Don't Care" value when read-during-write occurs at the same address location—Do not create a read-enable signal, or activate the read enable during a write operation.
Simple dual-port RAM	You can simultaneously perform one read and one write operations to different locations where the write operation happens on port A and the read operation happens on port B.
True dual-port RAM	You can perform any combination of two port operations: two reads, two writes, or one read and one write at two different clock frequencies. This mode is available only for M10K blocks.
Shift-register	You can use the memory blocks as a shift-register block to save logic cells and routing resources. This is useful in DSP applications that require local data storage such as finite impulse response (FIR) filters, pseudo-random number generators, multi-channel filtering, and auto- and cross-correlation functions. Traditionally, the local data storage is implemented with standard flip-flops that exhaust many logic cells for large shift registers.
ROM	You can use the memory blocks as ROM. ■ Initialize the ROM contents of the memory blocks using a **.mif** or .hex. ■ The address lines of the ROM are registered on M10K blocks but can be unregistered on MLABs. ■ The outputs can be registered or unregistered. ■ The output registers can be asynchronously cleared. ■ The ROM read operation is identical to the read operation in the single-port RAM configuration.
FIFO	You can use the memory blocks as FIFO buffers. Use the SCFIFO and DCFIFO megafunctions to implement single- and dual-clock asynchronous FIFO buffers in your design. For designs with many small and shallow FIFO buffers, the MLABs are ideal for the FIFO mode. However, the MLABs do not support mixed-width FIFO mode.

For more information about each memory mode, refer to the *Internal Memory (RAM and ROM) User Guide*.

For more information about implementing the shift register mode, refer to the *RAM-Based Shift Register (ALTSHIFT_TAPS) Megafunction User Guide*.

For more information about implementing FIFO buffers, refer to the *SCFIFO and DCFIFO Megafunctions User Guide*.

Mixed-Width Port Configurations

The mixed-width port configuration is supported in the simple dual-port RAM and true dual-port RAM memory modes.

For more information about dual-port mixed width support, refer to the *Internal Memory (RAM and ROM) User Guide*.

M10K Blocks Mixed-Width Configurations

Table 2–3 lists the mixed-width configurations of the M10K blocks in the simple dual-port RAM mode.

Table 2–3. M10K Block Mixed-Width Configurations (Simple Dual-Port RAM Mode)

Read Port	Write Port									
	8K x 1	4K x 2	2K x 4	2K x 5	1K x 8	1K x 10	512 x 16	512 x 20	256 x 32	256 x 40
8K x 1	Yes	Yes	Yes	—	Yes	—	Yes	—	Yes	—
4K x 2	Yes	Yes	Yes	—	Yes	—	Yes	—	Yes	—
2K x 4	Yes	Yes	Yes	—	Yes	—	Yes	—	Yes	—
2K x 5	—	—	—	Yes	—	Yes	—	Yes	—	Yes
1K x 8	Yes	Yes	Yes	—	Yes	—	Yes	—	Yes	—
1K x 10	—	—	—	Yes	—	Yes	—	Yes	—	Yes
512 x 16	Yes	Yes	Yes	—	Yes	—	Yes	—	Yes	—
512 x 20	—	—	—	Yes	—	Yes	—	Yes	—	Yes
256 x 32	Yes	Yes	Yes	—	Yes	—	Yes	—	Yes	—
256 x 40	—	—	—	Yes	—	Yes	—	Yes	—	Yes

Table 2–4 lists the mixed-width configurations of the M10K blocks in true dual-port mode.

Table 2–4. M10K Block Mixed-Width Configurations (True Dual-Port Mode) (Part 1 of 2)

Port B	Port A							
	8K x 1	4K x 2	2K x 4	2K x 5	1K x 8	1K x 10	512 x 16	512 x 20
8K x 1	Yes	Yes	Yes	—	Yes	—	Yes	—
4K x 2	Yes	Yes	Yes	—	Yes	—	Yes	—
2K x 4	Yes	Yes	Yes	—	Yes	—	Yes	—
2K x 5	—	—	—	Yes	—	Yes	—	Yes
1K x 8	Yes	Yes	Yes	—	Yes	—	Yes	—
1K x 10	—	—	—	Yes	—	Yes	—	Yes

Table 2–4. M10K Block Mixed-Width Configurations (True Dual-Port Mode) (Part 2 of 2)

Port B	Port A							
	8K x 1	4K x 2	2K x 4	2K x 5	1K x 8	1K x 10	512 x 16	512 x 20
512 x 16	Yes	Yes	Yes	—	Yes	—	Yes	—
512 x 20	—	—	—	Yes	—	Yes	—	Yes

MLABs Mixed-Width Configurations

MLABs do not have native support for mixed-width operation. However, if you select AUTO for your memory block type in the parameter editor, the Quartus II software can implement mixed-width memories in MLABs by using more than one MLAB.

Clocking Modes

This section describes the clocking modes for the Arria V memory blocks.

CAUTION To avoid corrupting the memory contents, do not violate the setup or hold time on the memory block address registers during read or write operations.

Clocking Modes for Each Memory Mode

Table 2–5 lists the memory blocks clocking modes supported in the Arria V devices and the memory modes supported by each clocking mode.

Table 2–5. Memory Blocks Clocking Modes for Each Memory Mode

Clocking Mode	Memory Mode				
	Single-Port	Simple Dual-Port	True Dual-Port	ROM	FIFO
Single clock mode	Yes	Yes	Yes	Yes	Yes
Read/write clock mode	—	Yes	—	—	Yes
Input/output clock mode	Yes	Yes	Yes	Yes	—
Independent clock mode	—	—	Yes	Yes	—

For more information about each clocking mode, refer to the *Internal Memory (RAM and ROM) User Guide*.

Asynchronous Clears

In all clocking modes, asynchronous clears are available only for output latches and output registers. For the independent clock mode, this is applicable on both ports.

Output Read Data in Simultaneous Read/Write

If you perform a simultaneous read/write to the same address location using the read/write clock mode, the output read data is unknown. If you require the output read data to be a known value, use single-clock or input/output clock mode and select the appropriate read-during-write behavior in the MegaWizard Plug-In Manager.

Independent Clock Enables

Independent clock enables are supported in the following clocking modes:

- Read/write clock mode—supported for both the read and write clocks.
- Independent clock mode—supported for the registers of both ports.

To save power, you can control the shut down of a particular register using the clock enables. For more information, refer to "Power Management" on page 2–13.

Parity Bit

Table 2–6 describes the parity bit support for the memory blocks.

Table 2–6. Parity Bit Support for the Memory Blocks

M10K	MLAB
■ The parity bit is the fifth bit associated with each 4 data bits in data widths of 5, 10, 20, and 40 (bits 4, 9, 14, 19, 24, 29, 34, and 39). ■ In non-parity data widths, the parity bits are skipped during read or write operations. ■ Parity function is not performed on the parity bit.	■ The parity bit is the ninth bit associated with each byte. ■ The ninth bit can store a parity bit or serve as an additional bit. ■ Parity function is not performed on the parity bit.

Byte Enable

The memory blocks support byte enable controls:

- The byte enable controls mask the input data so that only specific bytes of data are written. The unwritten bytes retain the values written previously.
- The write enable (`wren`) signal, together with the byte enable (`byteena`) signal, control the write operations on the RAM blocks. By default, the `byteena` signal is high (enabled) and only the `wren` signal controls the writing.
- The byte enable registers do not have a `clear` port.
- If you are using parity bits, on the M10K blocks, the byte enable function controls 8 data bits and 2 parity bits; on the MLABs, the byte enable function controls all 10 bits in the widest mode.
- The MSB and LSB of the `byteena` signal correspond to the MSB and LSB of the data bus, respectively.
- The byte enables are active high.

byteena Controls

Table 2–7 lists the byteena controls in the x20 data widths.

Table 2–7. byteena Controls in x20 Data Width

byteena[1:0]	Data Bits Written	
11 (default)	[19:10]	[9:0]
10	[19:10]	—
01	—	[9:0]

Table 2–8 lists the byteena controls in the x40 data widths.

Table 2–8. byteena Controls in x40 Data Width

byteena[3:0]	Data Bits Written			
1111 (default)	[39:30]	[29:20]	[19:10]	[9:0]
1000	[39:30]	—	—	—
0100	—	[29:20]	—	—
0010	—	—	[19:10]	—
0001	—	—	—	[9:0]

Data Byte Output

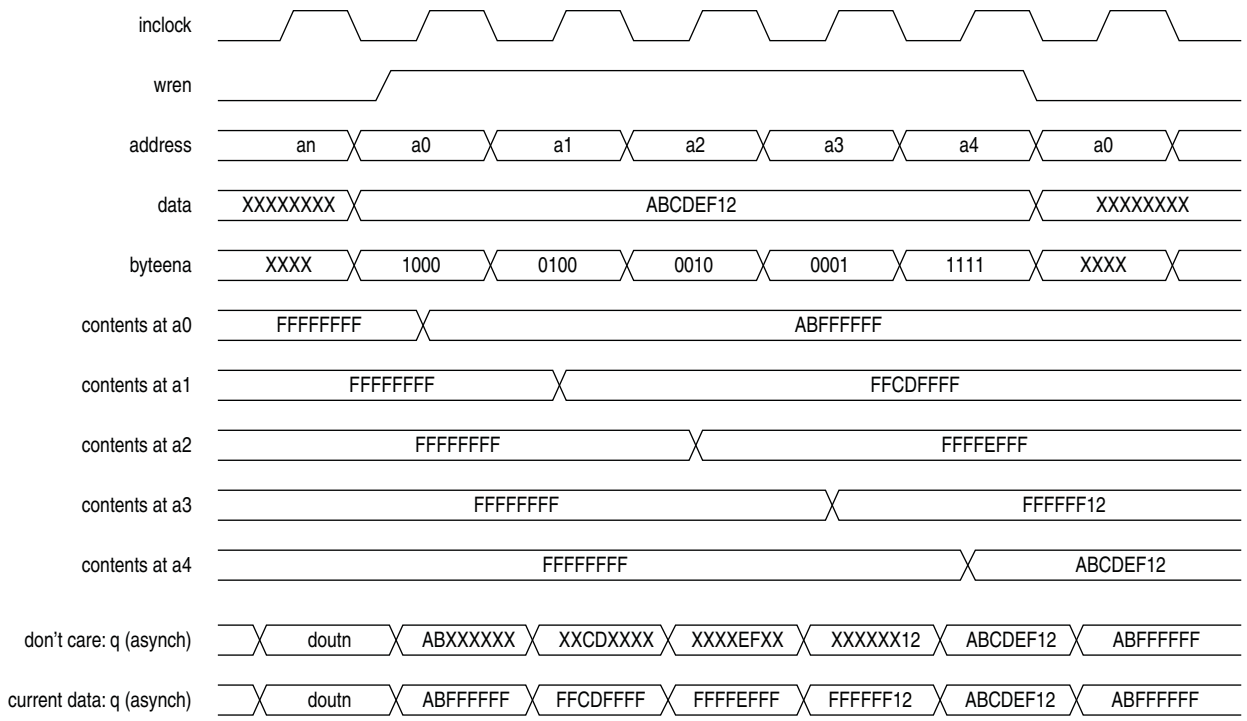
In M10K blocks, the corresponding masked data byte output appears as a “don’t care” value.

In MLABs, when a byte-enable bit is deasserted during a write cycle, the corresponding data byte output appears as either a “don’t care” value or the current data at that location. You can control the output value for the masked byte in MLABs using the Quartus II software.

RAM Blocks Operations

Figure 2–1 shows how the wren and byteena signals control the operations of the RAM blocks.

Figure 2–1. Byte Enable Functional Waveform *(1)*



Note to Figure 2–1:

(1) For the M10K blocks, the write-masked data byte output appears as a "don't care" value because the "current data" value is not supported.

Design Considerations

To ensure the success of your designs using the memory blocks in Arria V devices, take into consideration the following guidelines when you are designing with the memory blocks.

Memory Block Selection

The Quartus II software automatically partitions the user-defined memory into the memory blocks based on the speed and size constraints placed on your design. For example, the Quartus II software may spread out the memory across multiple available memory blocks to increase the performance of the design.

To assign the memory to a specific block size manually, use the RAM megafunction in the MegaWizard™ Plug-In Manager.

For the MLABs, you can implement single-port SRAM through emulation using the Quartus II software. Emulation results in minimal additional use of logic resources.

Because of the dual-purpose architecture of the MLAB, only data input registers and output registers are available in the block. The MLABs gain read address registers from the ALMs. However, the write address and read data registers are internal to the MLABs.

Conflict Resolution

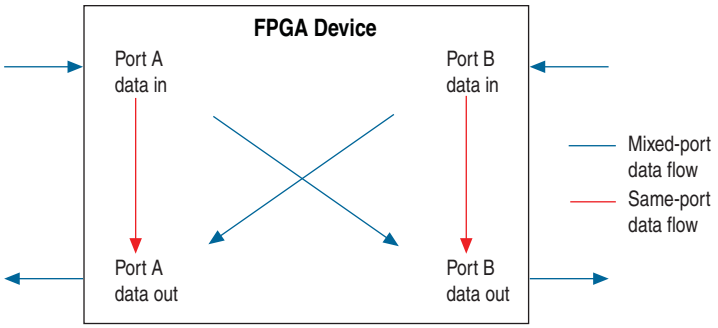
In the true dual-port RAM mode, you can perform two write operations to the same memory location. However, the memory blocks do not have internal conflict resolution circuitry. To avoid unknown data being written to the address, implement external conflict resolution logic to the memory block.

Read-During-Write Behavior

Customize the read-during-write behavior of the memory blocks to suit your design requirements.

Figure 2–2 shows the difference between the two types of read-during-write operations available—same port and mixed port.

Figure 2–2. Read-During-Write Data Flow for Arria V Devices



Same-Port Read-During-Write Mode

The same-port read-during-write mode applies to a single-port RAM or the same port of a true dual-port RAM.

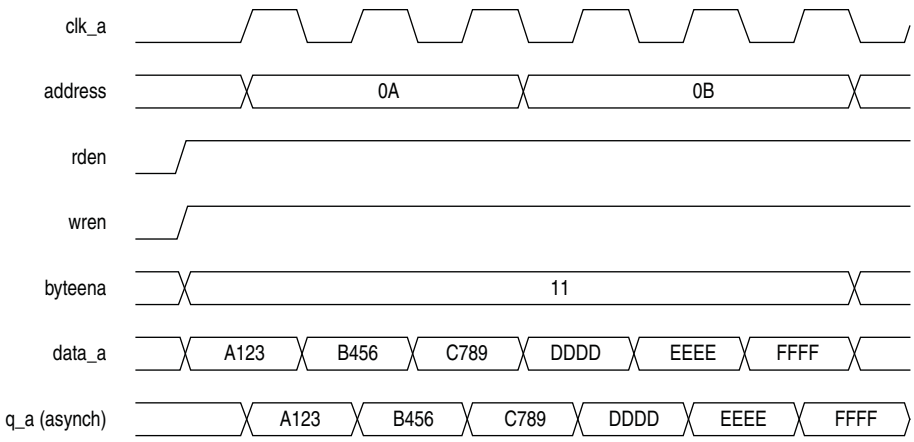
Table 2–9 lists the available output modes if you select the M10K or MLAB in the same-port read-during-write mode.

Table 2–9. Output Modes for M10K or MLAB in Same-Port Read-During-Write Mode

Output Mode	Memory Type	Description
"new data" (flow-through)	M10K	The new data is available on the rising edge of the same clock cycle on which the new data is written.
"don't care"	M10K, MLAB	The RAM outputs "don't care" values for a read-during-write operation.

Figure 2–3 shows sample functional waveforms of same-port read-during-write behavior in the "new data" mode.

Figure 2–3. Same-Port Read-During-Write: New Data Mode



Mixed-Port Read-During-Write Mode

The mixed-port read-during-write mode applies to simple and true dual-port RAM modes where two ports perform read and write operations on the same memory address using the same clock—one port reading from the address, and the other port writing to it.

Table 2–10 lists the available output modes in the mixed-port read-during-write mode.

Table 2–10. Output Modes for RAM in Mixed-Port Read-During-Write Mode

Output Mode	Memory Type	Description
"new data"	MLAB	A read-during-write operation to different ports causes the MLAB registered output to reflect the "new data" on the next rising edge after the data is written to the MLAB memory. This mode is available only if the output is registered.
"old data"	M10K, MLAB	A read-during-write operation to different ports causes the RAM output to reflect the "old data" value at the particular address. For MLAB, this mode is available only if the output is registered.
"don't care"	M10K, MLAB	The RAM outputs "don't care" or "unknown" value. For MLAB, the Quartus II software does not analyze the timing between write and read operations. To prevent metastability issue at the MLAB output, never write and read the same address at the same time. For M10K memory, the Quartus II software analyzes the timing between write and read operations.
"constrained don't care"	MLAB	The RAM outputs "don't care" or "unknown" value. The Quartus II software analyzes the timing between write and read operations in the MLAB.

Figure 2–4 shows a sample functional waveform of mixed-port read-during-write behavior for the "new data" mode.

Figure 2–4. Mixed-Port Read-During-Write: New Data Mode

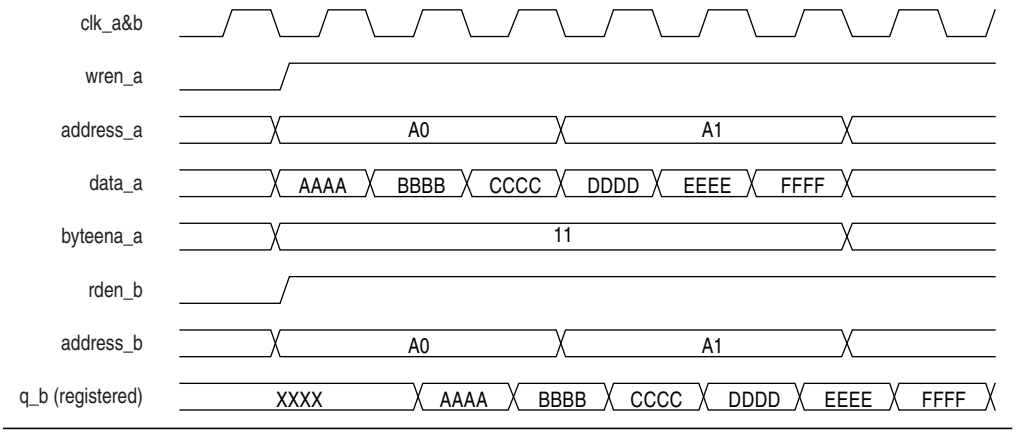


Figure 2–5 shows a sample functional waveform of mixed-port read-during-write behavior for the "old data" mode.

Figure 2–5. Mixed-Port Read-During-Write: Old Data Mode

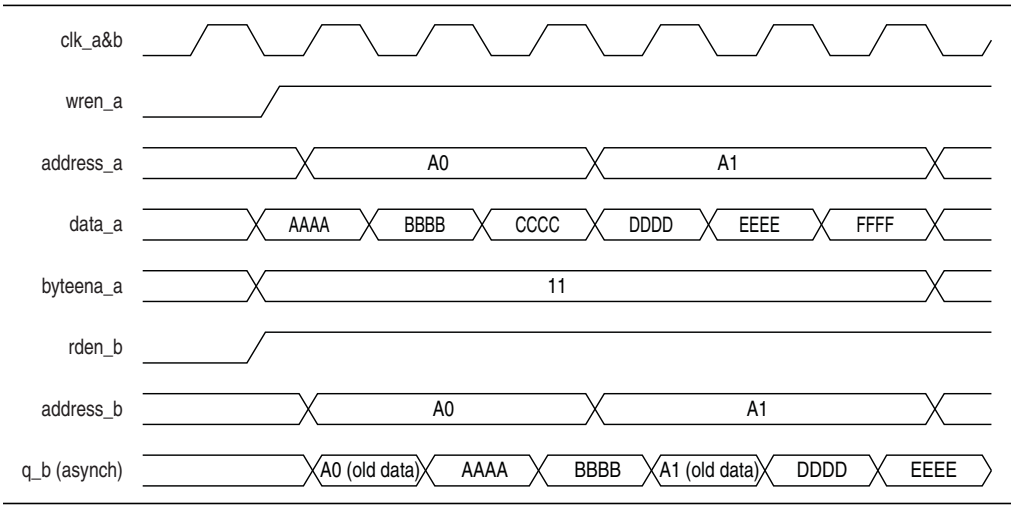
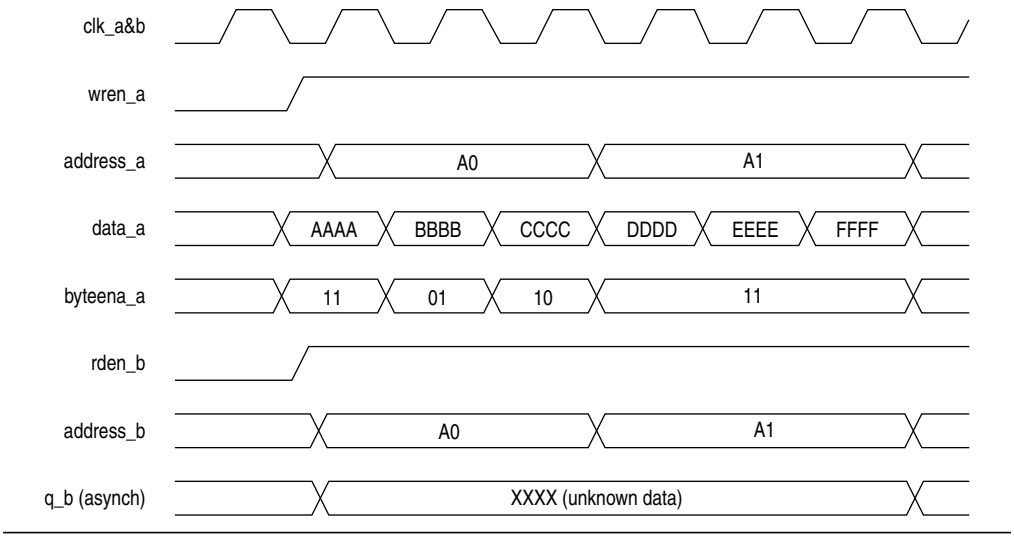


Figure 2–6 shows a sample functional waveform of mixed-port read-during-write behavior for the "don't care" or "constrained don't care" mode.

Figure 2–6. Mixed-Port Read-During-Write: Don't Care or Constrained Don't Care Mode



In the dual-port RAM mode, the mixed-port read-during-write operation is supported if the input registers have the same clock. The output value during the operation is "unknown."

For more information about the RAM megafunction that controls the read-during-write behavior, refer to the *Internal Memory (RAM and ROM) User Guide.*

Power-Up State and Memory Initialization

Consider the power up state of the different types of memory blocks if you are designing logic that evaluates the initial power-up values, as listed in Table 2–11.

Table 2–11. Initial Power-Up Values of M10K and MLAB Blocks

Memory Type	Output Registers	Power Up Value
M10K	Used	Zero (cleared)
	Bypassed	Zero (cleared)
MLAB	Used	Zero (cleared)
	Bypassed	Read memory contents

By default, the Quartus II software initializes the RAM cells in Arria V devices to zero unless you specify a **.mif**.

All memory blocks support initialization with a **.mif**. You can create **.mif** files in the Quartus II software and specify their use with the RAM megafunction when you instantiate a memory in your design. Even if a memory is pre-initialized (for example, using a **.mif**), it still powers up with its output cleared.

For more information about **.mif** files, refer to the *Internal Memory (RAM and ROM) User Guide* and the *Quartus II Handbook*.

Power Management

Reduce AC power consumption in your design by controlling the clocking of each memory block:

- Use the read-enable signal to ensure that read operations occur only when required. If your design does not require read-during-write, you can reduce your power consumption by deasserting the read-enable signal during write operations, or during the period when no memory operations occur.
- Use the Quartus II software to automatically place any unused memory blocks in low-power mode to reduce static power.

Document Revision History

Table 2–12 lists the revision history for this chapter.

Table 2–12. Document Revision History

Date	Version	Changes
June 2012	2.0	■ Restructured the chapter. ■ Updated the "Memory Modes", "Clocking Modes", and "Design Considerations" sections. ■ Updated Table 2–1. ■ Added the "Parity Bit" and "Byte Enable" sections. ■ Moved the memory capacity information to the *Arria V Device Overview*.
November 2011	1.1	■ Updated Table 2–1. ■ Restructured chapter.
May 2011	1.0	Initial release.

3. Variable-Precision DSP Blocks in Arria V Devices

AV-52003-2.0

This chapter describes how the variable-precision digital signal processing (DSP) blocks in Arria® V devices are optimized to support higher bit precision in high-performance DSP applications.

This chapter contains the following sections:

Features

Arria V variable-precision DSP blocks offer the following:

- High-performance, power-optimized, and fully registered multiplication operations
- 9-bit, 18-bit, and 27-bit word lengths
- Two 18 x 19 complex multiplications
- Built-in addition, subtraction, and dual 64-bit accumulation unit to combine multiplication results
- Cascading 19-bit or 27-bit to form the tap-delay line for filtering applications
- Cascading 64-bit output bus to propagate output results from one block to the next block without external logic support
- Hard pre-adder supported in 19-bit and 27-bit mode for symmetric filters
- Internal coefficient register bank for filter implementation
- 18-bit and 27-bit systolic finite impulse response (FIR) filters with distributed output adder

For more information about the number of multipliers in each Arria V device, refer to the *Arria V Device Overview*.

Supported Operational Modes

Table 3–1 summarizes the operational modes that are supported by Arria V variable-precision DSP blocks.

Table 3–1. Variable-Precision DSP Blocks Operational Modes for Arria V Devices

Variable-Precision DSP Block Resource	Operation Mode	Supported Instance	Pre-Adder Support	Coefficient Support	Input Cascade Support (1)	Chainout Support
1 variable-precision DSP block	Independent 9 x 9 multiplication	3	No	No	No	No
	Independent 18 x 18 multiplication	2	Yes	Yes	Yes	No
	Independent 18 x 19 multiplication	2	Yes	Yes	Yes	No
	Independent 18 x 25 multiplication	1	Yes	Yes	Yes	Yes
	Independent 20 x 24 multiplication	1	Yes	Yes	Yes	Yes
	Independent 27 x 27 multiplication	1	Yes	Yes	Yes	Yes
	Two 18 x 19 multiplier adder mode	1	Yes	Yes	Yes	Yes
	18 x 18 multiplier adder summed with 36-bit input	1	Yes	No	No	Yes
2 variable-precision DSP blocks	Complex 18 x 19 multiplication	1	No	No	Yes	No

Note to Table 3–1:

(1) When you enable the pre-adder feature, the input cascade support is not available.

Design Considerations

Operational Modes

The Quartus® II software includes megafunctions that you can use to control the operation mode of the multipliers. After entering the parameter settings with the MegaWizard™ Plug-In Manager, the Quartus II software automatically configures the variable-precision DSP block.

For more information, refer to the following user guides:

- *Introduction to Megafunction User Guide*
- *Integer Arithmetic Megafunctions User Guide*
- *Floating-Point Megafunctions User Guide*

Pre-Adder

To use the pre-adder feature, all input data and multipliers must have the same clock setting.

The input cascade support is not available when you enable the pre-adder feature.

Internal Coefficient

In both 18-bit and 27-bit modes, you can use the coefficient feature and pre-adder feature independently.

Accumulator

The accumulator supports double accumulation by enabling the 64-bit double accumulation registers located between the output register bank and the accumulator.

The double accumulation registers are set statically in the programming file.

Chainout Adder

You can use the output chaining path to add results from other DSP blocks.

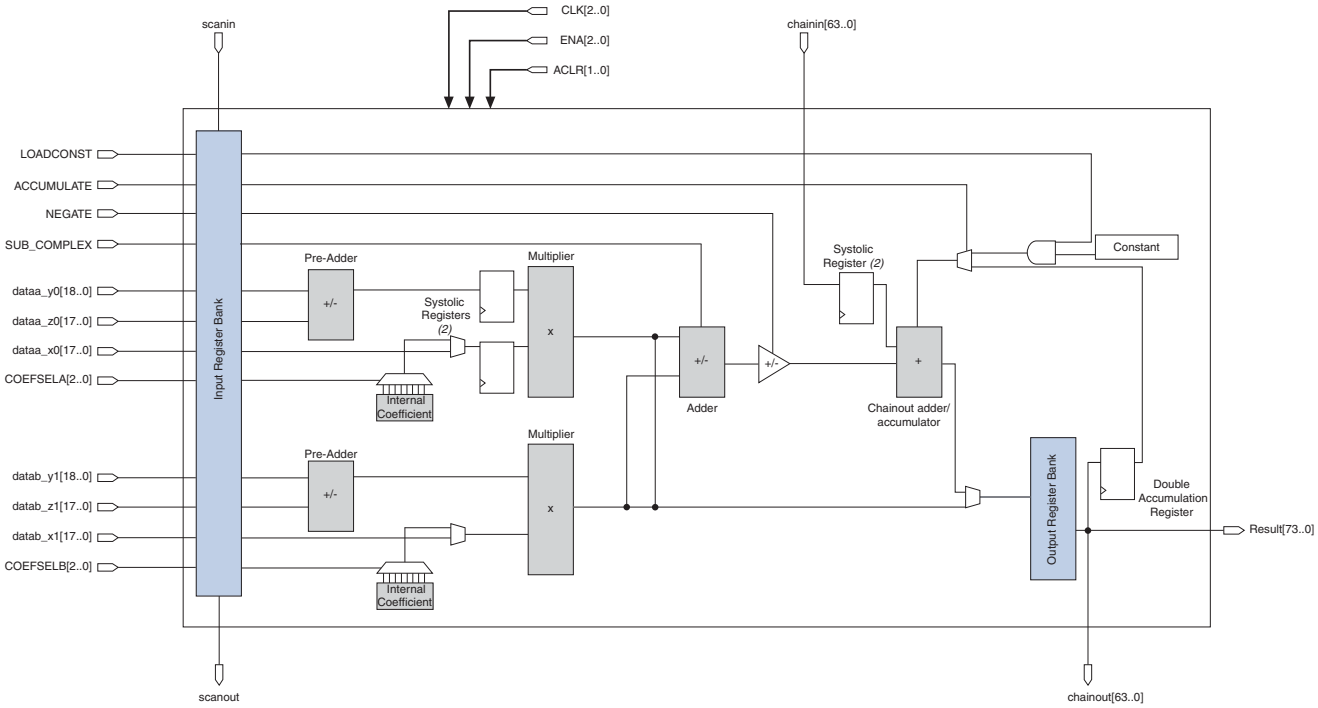
Block Architecture

The Arria V variable-precision DSP block consists of the following elements:

- Input Register Bank
- Pre-Adder
- Internal Coefficient
- Multipliers
- Adder
- Accumulator and Chainout Adder
- Systolic Registers
- Double Accumulation Register
- Output Register Bank

Figure 3–1 shows an overall architecture of the Arria V variable-precision DSP block.

Figure 3–1. Variable-Precision DSP Block Architecture for Arria V Devices *(1)*



Notes to Figure 3–1:

(1) If the variable-precision DSP block is not configured in systolic FIR mode, both systolic registers are bypassed.

(2) When enabled, systolic registers are clocked with the same clock source as the output register bank.

Input Register Bank

The input register bank consists of data, dynamic control signals, and two sets of delay registers.

All the registers in the DSP blocks are positive-edge triggered and cleared on power up. Each multiplier operand can feed an input register or a multiplier directly, bypassing the input registers.

The following variable-precision DSP block signals control the input registers within the variable-precision DSP block:

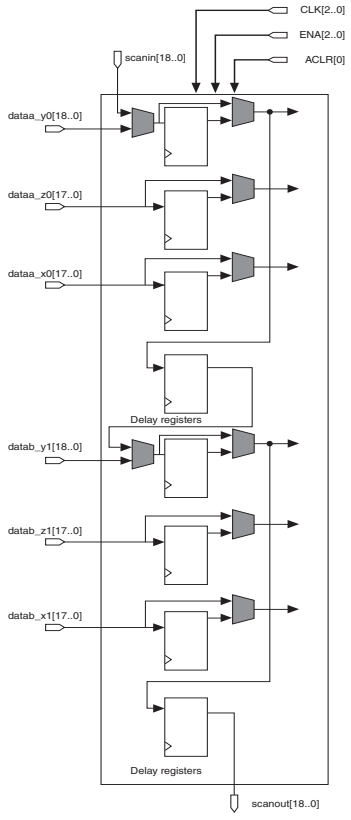
- `CLK[2..0]`
- `ENA[2..0]`
- `ACLR[0]`

In 18 x 19 mode, you can use the delay registers to balance the latency requirements when you use both the input cascade and chainout features.

The tap-delay line feature allows you to drive the top leg of the multiplier input, `dataa_y0` and `datab_y1` in 18 x 19 mode and `dataa_y0` only in 27 x 27 mode, from the general routing or cascade chain.

Figure 3–2 shows the input register for 18 x19 mode.

Figure 3–2. Input Register of a Variable-Precision DSP Block in 18 x 19 Mode *(1)*

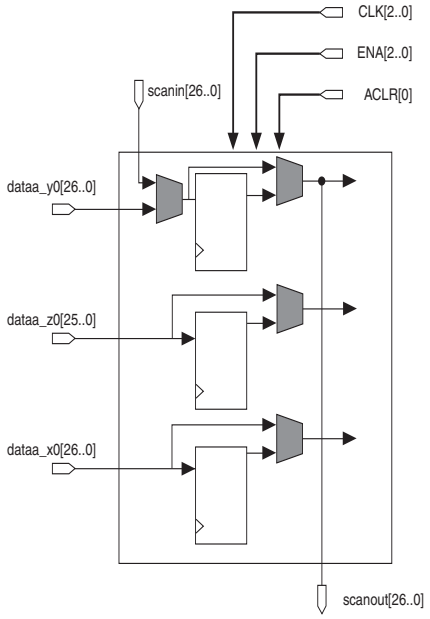


Note to Figure 3–2:

(1) This figure shows the data registers only. Registers for the control signals are not shown.

Figure 3–3 shows the input register for 27 x 27 mode.

Figure 3–3. Input Register of a Variable-Precision DSP Block in 27 x 27 Mode *(1)*



Note to Figure 3–3:

(1) This figure shows the data registers only. Registers for the control signals are not shown.

Pre-Adder

Each variable-precision DSP block has two 19-bit pre-adders. You can configure these pre-adders as two 19-bit pre-adders or one 27-bit pre-adder.

The pre-adder supports both addition and substraction in the following input configurations:

- 18-bit (signed) addition or subtraction for 18 x 19 mode
- 17-bit (unsigned) addition or subtraction for 18 x 19 mode
- 26-bit addition or subtraction for 27 x 27 mode

Internal Coefficient

The Arria V variable-precision DSP block has the flexibility of selecting the multiplicand from either the dynamic input or the internal coefficient.

The internal coefficient can support up to eight constant coefficients for the multiplicands in 18-bit and 27-bit modes. When you enable the internal coefficient feature, `COEFSELA/COEFSELB` are used to control the selection of the coefficient multiplexer.

Multipliers

A single variable-precision DSP block can perform many multiplications in parallel, depending on the data width of the multiplier.

There are two multipliers per variable-precision DSP block.

You can configure these two multipliers to the following operational modes:

- One 27 x 27 multiplier
- Two 18 (signed)/(unsigned) x 19 (signed) multipliers
- Three 9 x 9 multipliers

For more information about the operational modes of the multipliers, refer to "Operational Mode Descriptions" on page 3–9.

Adder

You can use the adder in various sizes, depending on the operational mode:

- one 64-bit adder with the 64-bit accumulator
- two 18 x 19 modes—the adder is divided into two 37-bit adders to produce the full 37-bit result of each independent 18 x 19 multiplication
- three 9 x 9 modes—you can use the adder as three 18-bit adders to produce three 9 x 9 multiplication results independently

Accumulator and Chainout Adder

The Arria V variable-precision DSP block supports a 64-bit accumulator and a 64-bit adder.

The following signals can dynamically control the function of the accumulator:

- `NEGATE`
- `LOADCONST`
- `ACCUMULATE`

The accumulator supports double accumulation by enabling the 64-bit double accumulation registers located between the output register bank and the accumulator.

The double accumulation registers are set statically in the programming file.

The accumulator and chainout adder features are not supported in two independent 18 x 19 modes and three independent 9 x 9 modes.

Table 3–2 lists the dynamic signals settings and description for each function.

Table 3–2. Dynamic Control Signals for 64-Bit Accumulator in Arria V Devices

Function	Description	NEGATE	LOADCONST	ACCUMULATE
Zeroing	Disables the accumulator.	0	0	0
Preload	Loads an initial value to the accumulator. Only one bit of the 64-bit preload value can be "1". It can be used as rounding the DSP result to any position of the 64-bit result.	0	1	0
Accumulation	Adds the current result to the previous accumulate result.	0	X (1)	1
Decimation	This function takes the current result, converts it into two's complement, and adds it to the previous result.	1	X (1)	1

Note to Table 3–2:

(1) X denotes a "don't care" value.

Systolic Registers

There are two systolic registers per variable-precision DSP block. If the variable-precision DSP block is not configured in systolic FIR mode, both systolic registers are bypassed.

The first set of systolic registers consists of 18-bit and 19-bit registers that are used to register the 18-bit and 19-bit inputs of the upper multiplier, respectively.

The second set of systolic registers are used to delay the chainout output to the next variable-precision DSP block.

You must clock all the systolic registers with the same clock source as the output register bank.

Double Accumulation Register

The double accumulation register is an extra register in the feedback path of the accumulator. Enabling the double accumulation register will cause an extra clock cycle delay in the feedback path of the accumulator.

This register has the same `CLK`, `ENA`, and `ACLR` settings as the output register bank.

By enabling this register, you can have two accumulator channels using the same number of variable precision DSP block.

Output Register Bank

The positive edge of the clock signal triggers the 64-bit bypassable output register bank and is cleared after power up.

The following variable-precision DSP block signals control the output register per variable-precision DSP block:

- `CLK[2..0]`
- `ENA[2..0]`
- `ACLR[1]`

Operational Mode Descriptions

This section describes how you can configure an Arria V variable-precision DSP block to efficiently support the following operational modes:

- Independent Multiplier Mode
- Independent Complex Multiplier Mode
- Multiplier Adder Sum Mode
- 18 x 18 Multiplication Summed with 36-Bit Input Mode
- Systolic FIR Mode

Independent Multiplier Mode

In independent input and output multiplier mode, the variable-precision DSP blocks perform individual multiplication operations for general purpose multipliers.

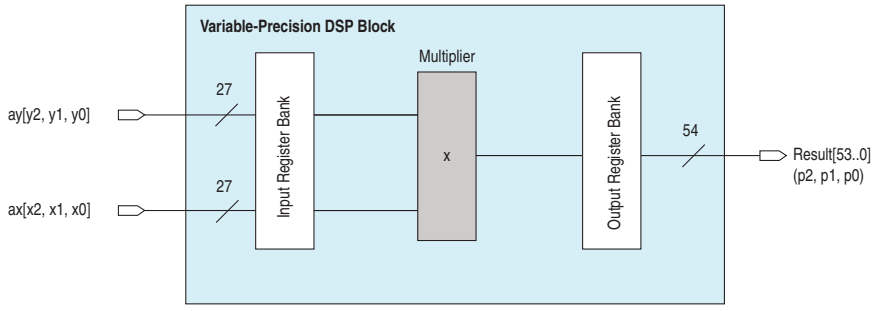
Table 3–3 lists the Arria V variable-precision DSP block multiplier configurations for the independent multiplier mode.

Table 3–3. Variable-Precision DSP Block Independent Multiplier Mode Configurations

Configuration	Multipliers per block	Description
9 x 9	3	Figure 3–4
18 (signed) x 18 (unsigned)	2	Figure 3–5
18 (unsigned) x 18 (unsigned)		
18 (signed) x 19 (signed)		
18 (unsigned) x 19 (signed)		
18 x 25	1	Figure 3–6
20 x 24	1	Figure 3–7
27 x 27	1	Figure 3–8

Figure 3–4 shows the variable-precision DSP block in 9 x 9 independent multiplier mode.

Figure 3–4. Three 9 x 9 Independent Multiplier Mode per Variable-Precision DSP Block *(1)*

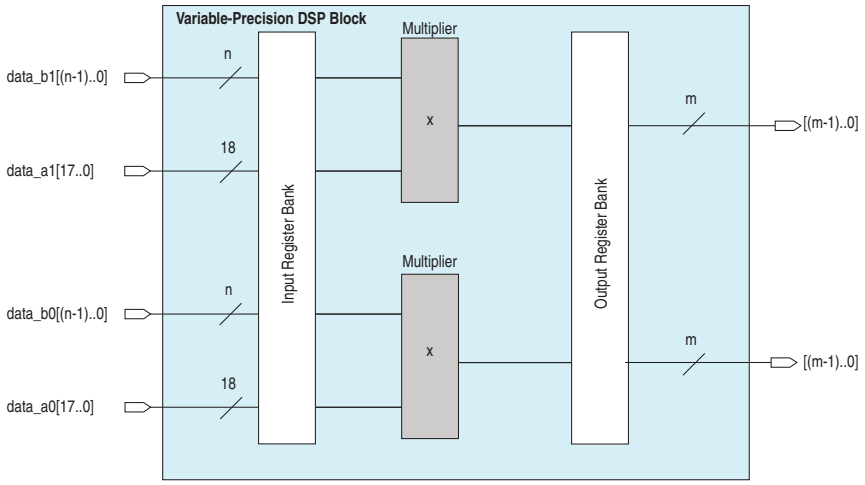


Note to Figure 3–4:

(1) Three pairs of data are packed into the `ax` and `ay` ports; `result` contains three 18-bit products.

Figure 3–5 shows the variable-precision DSP block in 18 x 18 or 18 x 19 independent multiplier mode.

Figure 3–5. Two 18 x 18 or 18 x 19 Independent Multiplier Mode per Variable-Precision DSP Block *(1), (2)*



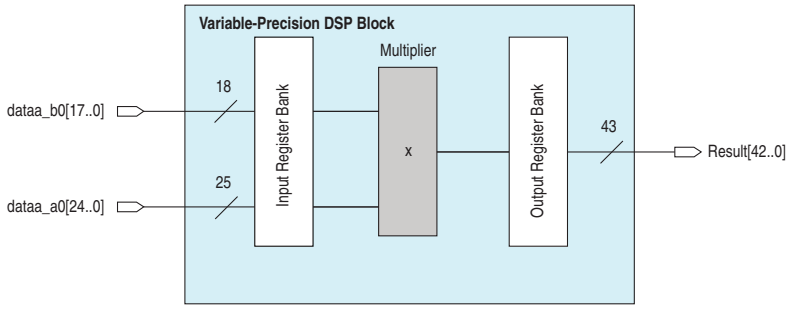
Notes to Figure 3–5:

(1) n = 19 and m = 37 for 18 x 19 mode.

(2) n = 18 and m = 36 for 18 x 18 mode.

Figure 3–6 shows the variable-precision DSP block in 18 x 25 independent multiplier mode.

Figure 3–6. One 18 x 25 Independent Multiplier Mode per Variable-Precision DSP Block *(1)*

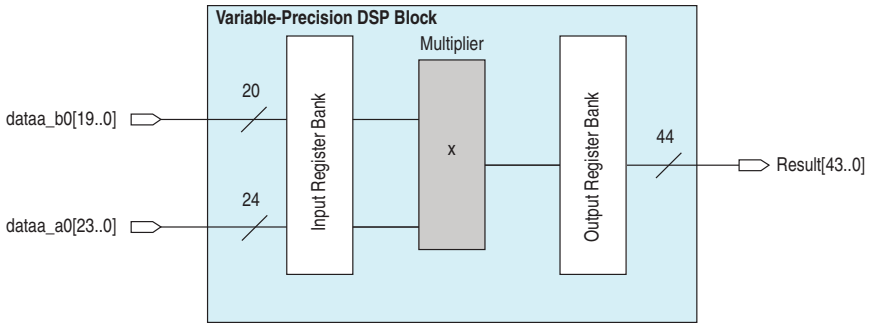


Note to Figure 3–6:

(1) The result can be up to 52 bits when combined with a chainout adder or accumulator.

Figure 3–7 shows the variable-precision DSP block in 20 x 24 independent multiplier mode.

Figure 3–7. One 20 x 24 Independent Multiplier Mode per Variable-Precision DSP Block *(1)*

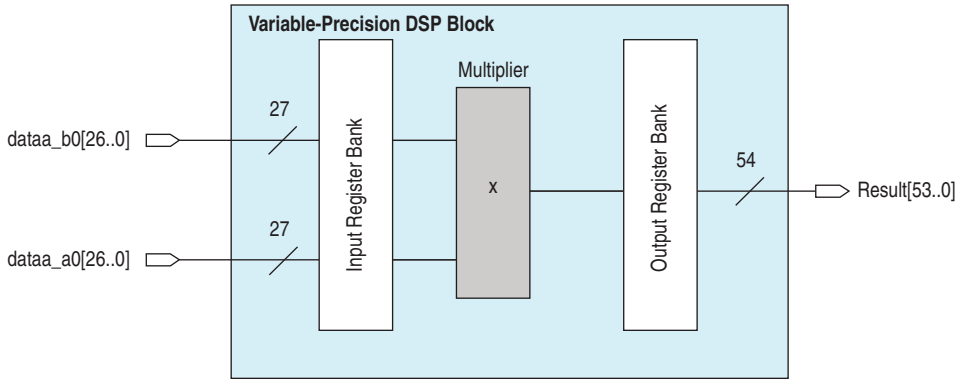


Note to Figure 3–7:

(1) The result can be up to 52 bits when combined with a chainout adder or accumulator.

Figure 3–8 shows the variable-precision DSP block in 27 x 27 independent multiplier mode.

Figure 3–8. One 27 x 27 Independent Multiplier Mode per Variable-Precision DSP Block *(1)*



Note to Figure 3–8:

(1) The result can be up to 64 bits when combined with a chainout adder or accumulator.

Independent Complex Multiplier Mode

The Arria V devices support the 18 x 19 complex multiplier mode using two Arria V variable-precision DSP blocks.

Equation 3–1 shows a sample complex multiplication equation.

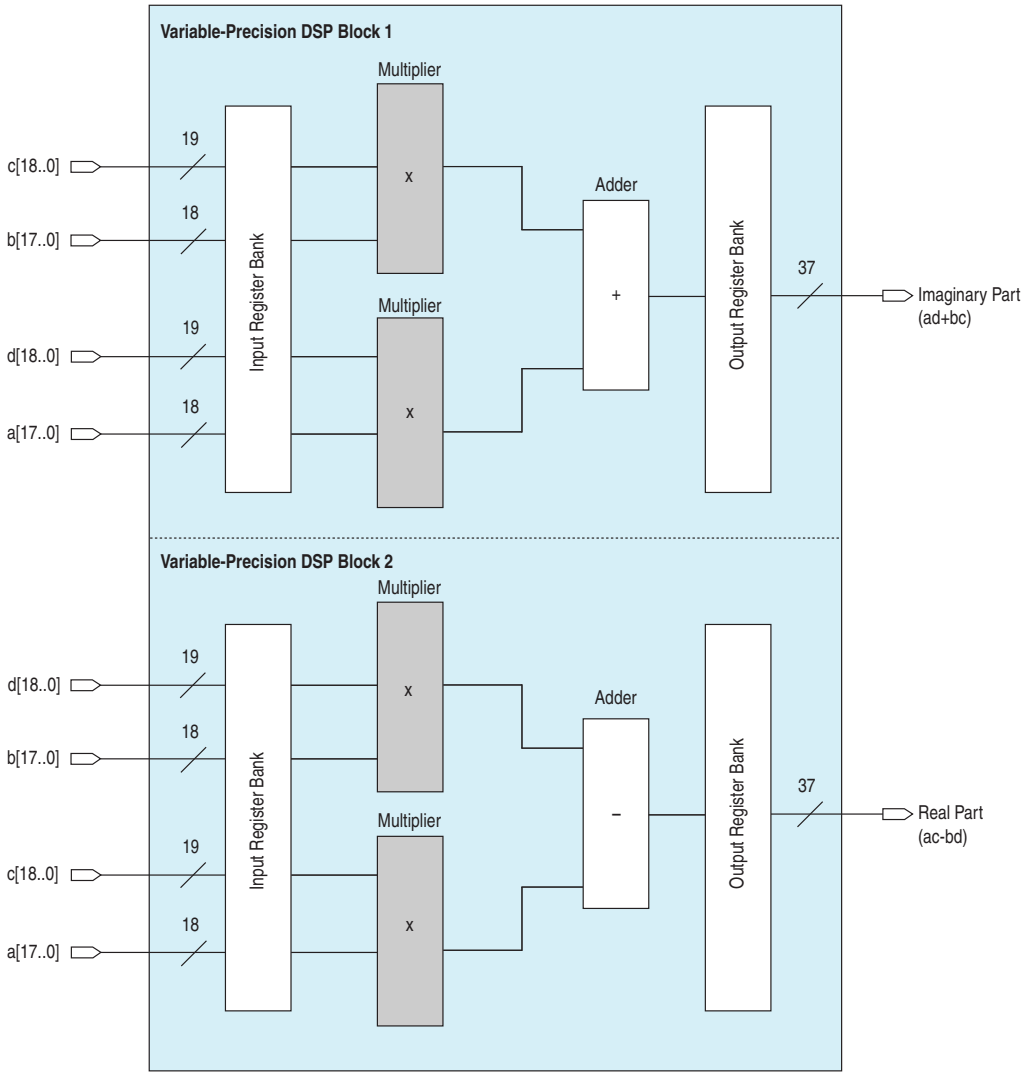
Equation 3–1. Complex Multiplication Equation

$$(a + jb) \times (c + jd) = [(a \times c) - (b \times d)] + j[(a \times d) + (b \times c)]$$

The imaginary part [(a × d) + (b × c)] is implemented in the first variable-precision DSP block, while the real part [(a × c) - (b × d)] is implemented in the second variable-precision DSP block.

Figure 3–9 shows an 18 x 19 complex multiplication.

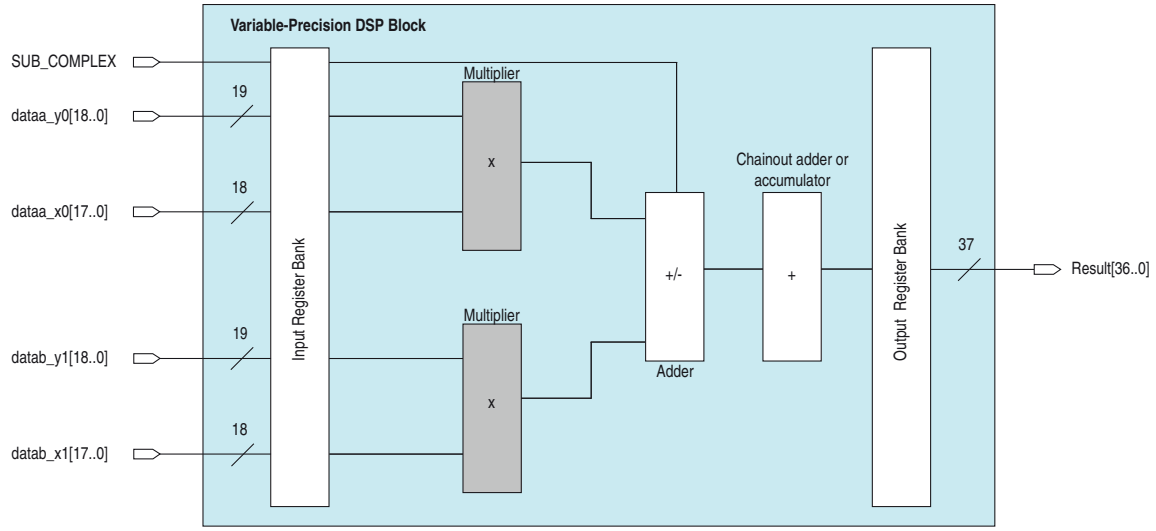
Figure 3–9. One 18 x 19 Complex Multiplier with Two Variable-Precision DSP Blocks



Multiplier Adder Sum Mode

Figure 3–10 shows the variable-precision DSP blocks in one sum of two 18 x 19 multipliers adder sum mode.

Figure 3–10. One Sum of Two 18 x 19 Multipliers with One Variable-Precision DSP Block



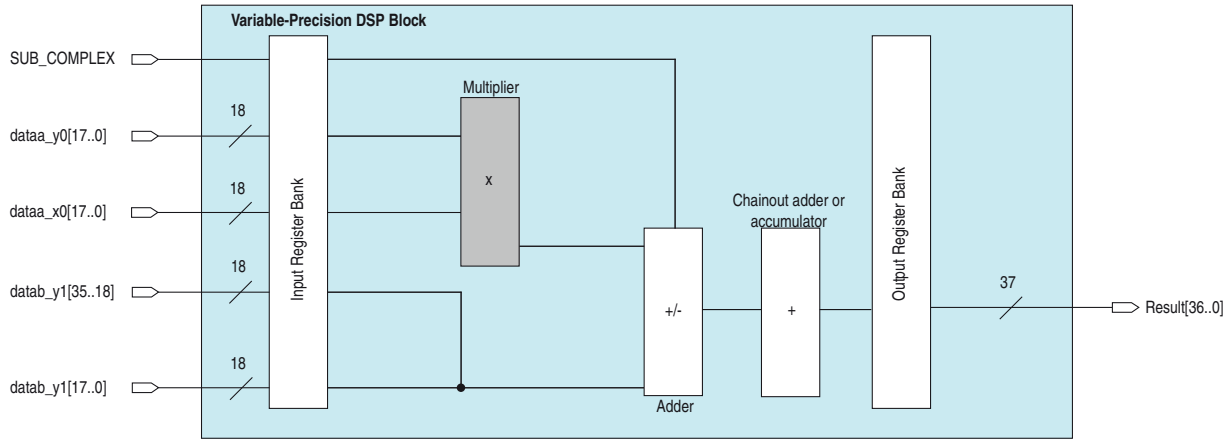
18 x 18 Multiplication Summed with 36-Bit Input Mode

Arria V variable-precision DSP blocks support one 18 x 18 multiplication summed to a 36-bit input.

Use the upper multiplier to provide the input for an 18 x 18 multiplication, while the bottom multiplier is bypassed. The `datab_y1[17..0]` and `datab_y1[35..18]` signals are concatenated to produce a 36-bit input.

Figure 3–11 shows the 18 x 18 multiplication summed with the 36-bit input mode in a variable-precision DSP block.

Figure 3–11. One 18 x 18 Multiplication Summed with 36-Bit Input Mode



Systolic FIR Mode

The basic structure of a FIR filter consists of a series of multiplications followed by an addition.

Equation 3–2 represents a FIR filter operation.

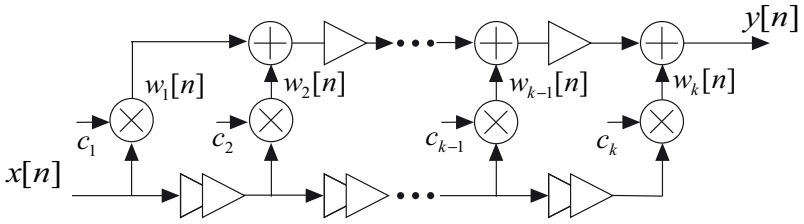
Equation 3–2. Basic FIR Filter Equation

$$y[n] = \sum_{i=1}^{k} c[i]x[n-i-1]$$

Depending on the number of taps and the input sizes, the delay through chaining a high number of adders can become quite large. To overcome the delay performance issue, the systolic form is used with additional delay elements placed per tap to increase the performance at the cost of increased latency.

Figure 3–12 shows the equivalent circuit of the FIR filter in systolic form.

Figure 3–12. Systolic FIR Filter Equivalent Circuit



Arria V variable-precision DSP blocks support 18-bit and 27-bit systolic FIR structures.

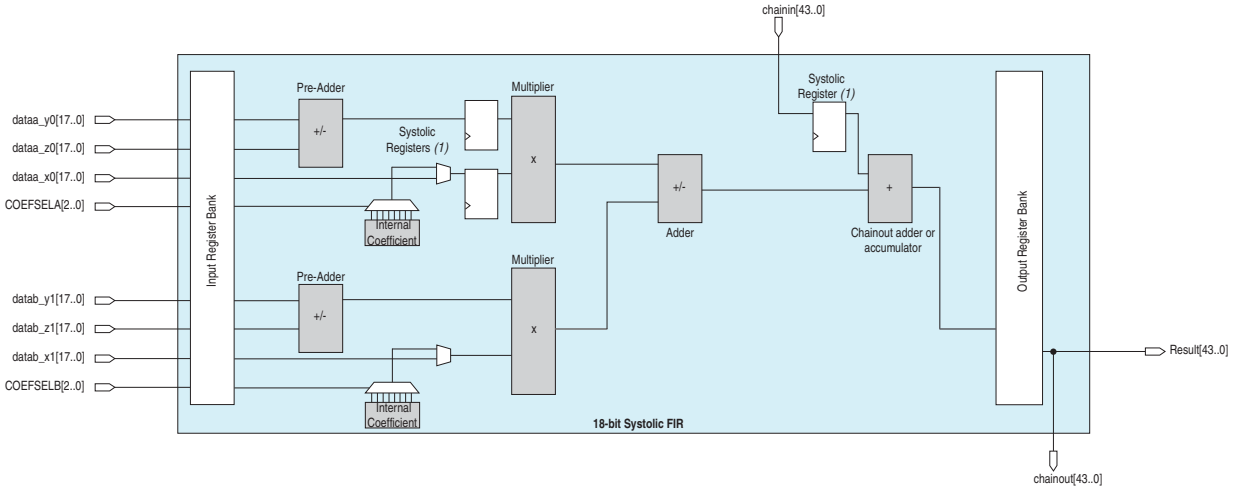
In systolic FIR mode, the input of the multiplier can come from four different sets of sources:

- two dynamic inputs
- one dynamic input and one coefficient input
- one coefficient input and one pre-adder output
- one dynamic input and one pre-adder output

Example for 18-bit systolic FIR—the adders are configured as dual 44-bit adders, thereby giving 8 bits of overhead when using an 18-bit operation (36-bit products). This allows a total of 256 multiplier products.

Figure 3–13 shows the 18-bit systolic FIR mode.

Figure 3–13. 18-Bit Systolic FIR Mode



Note to Figure 3–13:

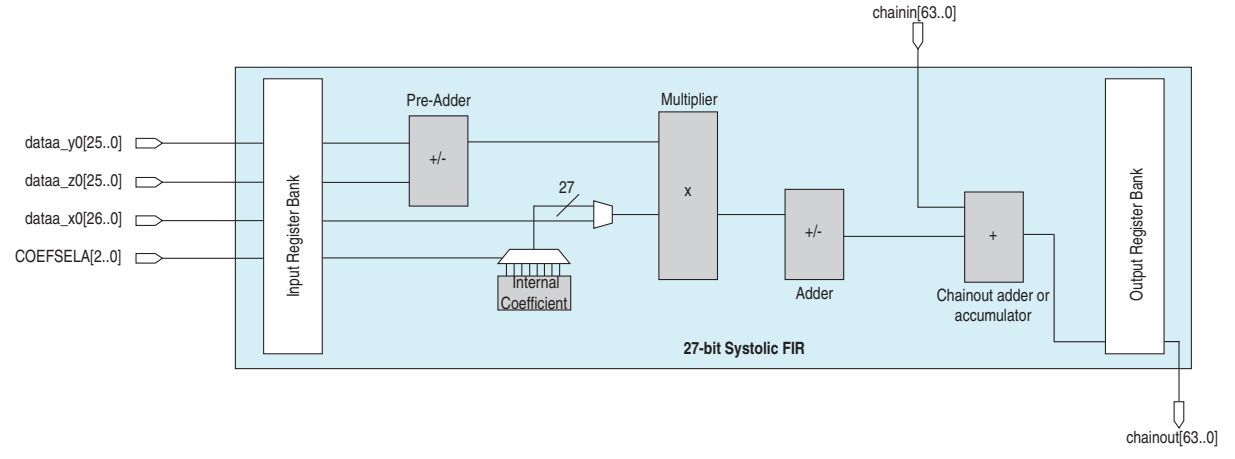
(1) The systolic registers have the same clock source as the output register bank.

Example for 27-bit systolic FIR—the chainout adder or accumulator is configured for a 64-bit operation, providing 10 bits of overhead when using a 27-bit data (54-bit products). This allows a total of 1,024 multiplier products.

The 27-bit systolic FIR mode allows the implementation of one stage systolic filter per DSP block.

Figure 3–14 shows the 27-bit systolic FIR mode.

Figure 3–14. 27-Bit Systolic FIR Mode



Document Revision History

Table 3–4 lists the revision history for this chapter.

Table 3–4. Document Revision History

Date	Version	Changes
June 2012	2.0	Updated for the Quartus II software v12.0 release: ■ Restructured chapter. ■ Added "Design Considerations", "Adder", and "Double Accumulation Register" sections. ■ Updated Figure 3–1 and Figure 3–13. ■ Added Table 3–3Table 3–3. ■ Updated "Systolic Registers" and "Systolic FIR Mode" sections. ■ Added Equation 3–2. ■ Added Figure 3–12.
November 2011	1.1	Restructured chapter.
May 2011	1.0	Initial release.

4. Clock Networks and PLLs in Arria V Devices

AV-52004-2.0

This chapter describes the advanced features of hierarchical clock networks and phase-locked loops (PLLs) in Arria® V devices. The Quartus® II software enables the PLLs and their features without external devices.

The chapter contains the following sections:

Clock Networks in Arria V Devices

Arria V devices contain the following clock networks that are organized into a hierarchical structure:

- Global clock networks (GCLKs)
- Regional clock networks (RCLKs)
- Periphery clock networks (PCLKs)

The clock networks provide up to 352 unique clock domains. Arria V devices allow up to 100 unique GCLK, RCLK, and PCLK clock sources (16 GCLKs + 22 RCLKs + 62 PCLKs) per device quadrant.

Table 4–1 lists the clock resources available in Arria V devices.

Table 4–1. Clock Resources in Arria V Devices—*Preliminary* (Part 1 of 2)

Clock Resource	Number of Resources Available	Source of Clock Resource
Clock input pins	40 Single-ended (20 Differential) (1)	CLK[0..7]p, CLK[12..23]p, CLK[0..7]n, and CLK[12..23]n pins (1)
	48 Single-ended (24 Differential) (2)	CLK[0..23]p and CLK[0..23]n pins (2)
GCLK networks	16	CLK[0..23]p and CLK[0..23]n pins, PLL clock outputs, and logic array
RCLK networks	88	CLK[0..23]p and CLK[0..23]n pins, PLL clock outputs, and logic array
PCLK networks	120,184, 224, and 248 (3)	DPA clock outputs, PLD-transceiver interface clocks, I/O pins, and logic array
GCLKs and RCLKs per quadrant	38	16 GCLKs + 22 RCLKs

Table 4–1. Clock Resources in Arria V Devices—*Preliminary* (Part 2 of 2)

Clock Resource	Number of Resources Available	Source of Clock Resource
GCLKs and RCLKs per device	104	16 GCLKs + 88 RCLKs

Notes to Table 4–1:

(1) This only applies to Arria V GX A1 and A3 devices, and Arria V GT C3 device.

(2) This applies to all Arria V devices except for Arria V GX A1 and A3 devices, and Arria V GT C3 device.

(3) There are 120 PCLKs in Arria V GX A1 and A3 devices, and Arria V GT C3 device, 184 PCLKs in Arria V GX A5 and A7 devices, and Arria V GT C7 device, 224 PCLKs in Arria V GX B1 and B3 devices, and Arria V GT D3 device, and 248 PCLKs in Arria V GX B5 and B7 devices, and Arria V GT D7 device.

Internally-generated GCLKs, RCLKs, or PCLKs cannot drive the Arria V PLLs. The input clock to the PLL must be driven by dedicated clock input pins, PLL-fed GCLKs, or PLL-fed RCLKs.

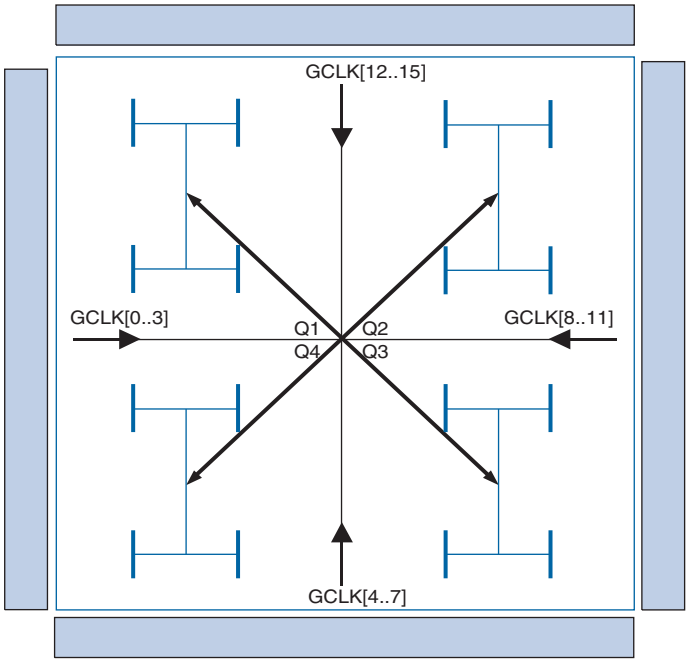
When used as single-ended clock inputs, the `CLKn` pins drive the PLLs over global or regional clock networks. The `CLKn` pins do not have dedicated routing paths to the PLLs.

For more information about the clock input pins connections, refer to *Arria V Device Family Pin Connection Guidelines*.

Global Clock Networks

Arria V devices provide up to 16 GCLKs that can drive throughout the device. The GCLKs serve as low-skew clock sources for functional blocks such as adaptive logic modules (ALMs), digital signal processing (DSP), embedded memory, and PLLs. Arria V I/O elements (IOEs) and internal logic can also drive GCLKs to create internally generated global clocks and other high fan-out control signals; such as synchronous or asynchronous clear and clock enable signals. Figure 4–1 shows the GCLK networks in Arria V devices.

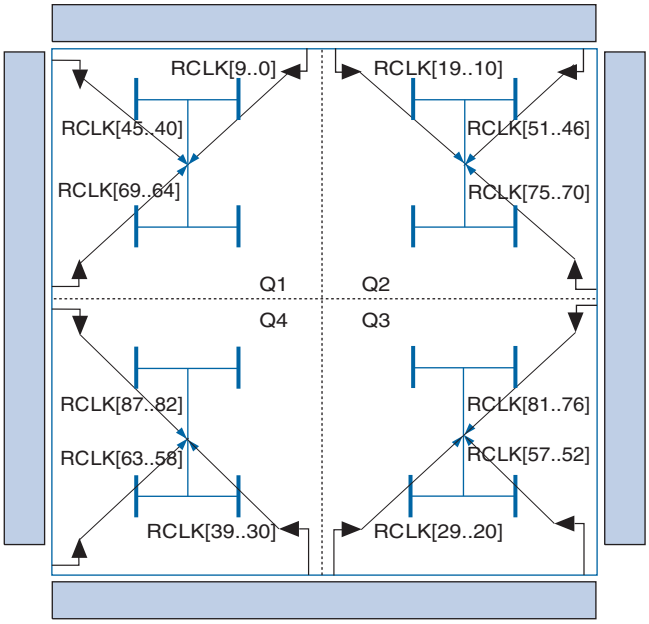
Figure 4–1. GCLK Networks in Arria V Devices



Regional Clock Networks

RCLK networks are only applicable to the quadrant they drive into. RCLK networks provide the lowest clock insertion delay and skew for logic contained within a single device quadrant. The Arria V IOEs and internal logic within a given quadrant can also drive RCLKs to create internally generated regional clocks and other high fan-out control signals; for example, synchronous or asynchronous clears and clock enables. Figure 4–2 shows the RCLK networks in Arria V devices.

Figure 4–2. RCLK Networks in Arria V Devices



Periphery Clock Networks

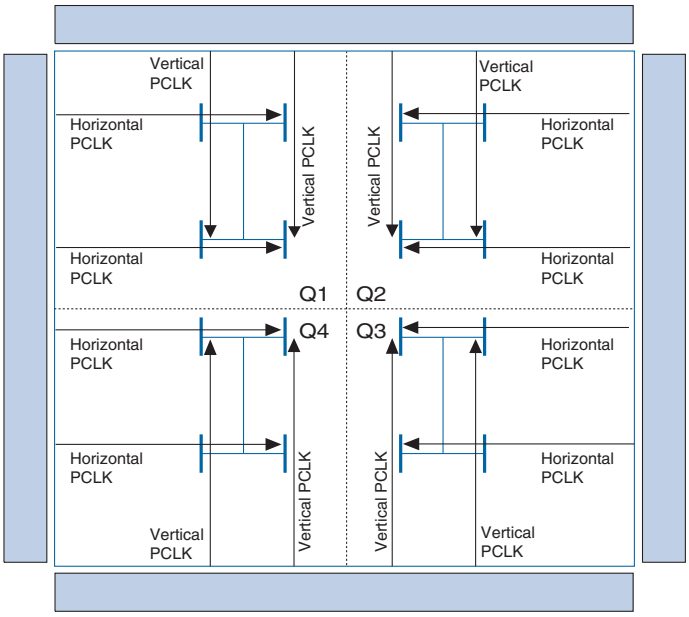
Depending on the routing direction, Arria V devices provide vertical PCLKs from the top and bottom periphery and horizontal PCLKs from the left and right periphery. Clock outputs from the dynamic phase aligner (DPA) block, programmable logic device (PLD)-transceiver interface clocks, I/O pins, and internal logic can drive the PCLK networks.

PCLKs have higher skew when compared with GCLK and RCLK networks. You can use PCLKs for general purpose routing to drive signals into and out of the Arria V device.

Legal clock sources for PCLK networks are clock outputs from the DPA block, PLD-transceiver interface clocks, horizontal I/O pins, and internal logic.

Figure 4–3 shows the PCLK networks in Arria V devices.

Figure 4–3. PCLK Networks in Arria V Devices



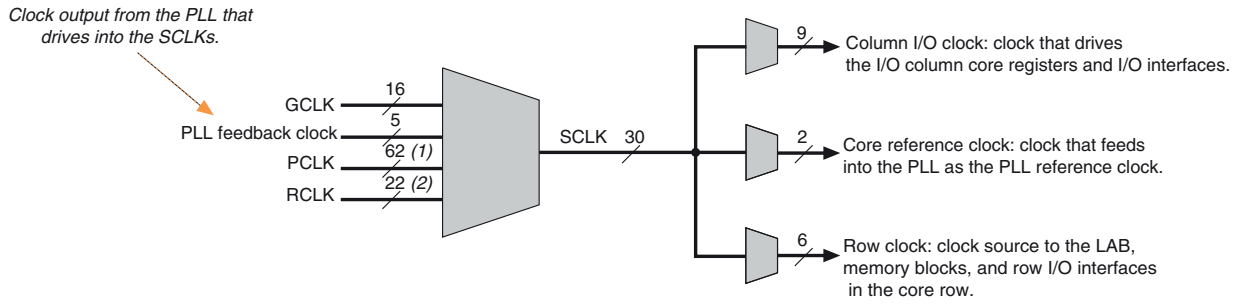
Clock Sources Per Quadrant

Arria V devices provide 30 section clock networks (SCLK) in each spine clock per quadrant that can drive six row clocks in each logic array block (LAB) row, nine column I/O clocks, and two core reference clocks. The SCLKs are the clock resources to the core functional blocks, PLLs, and I/O interfaces of the device.

A spine clock is another layer of routing between the GCLKs, RCLKs, and PCLK networks before each clock is connected to the clock routing for each LAB row. The settings for spine clocks are transparent. The Quartus II software automatically routes the spine clock based on the GCLK, RCLK, and PCLK networks.

Figure 4–4 shows SCLKs driven by the GCLK, RCLK, PCLK, or the PLL feedback clock networks in each spine clock per quadrant. The GCLK, RCLK, PCLK, and PLL feedback clocks share the same routing to the SCLKs. To ensure successful design fitting in the Quartus II software, the total number of clock resources must not exceed the SCLK limits in each region.

Figure 4–4. Hierarchical Clock Networks in Each Spine Clock Per Quadrant



Notes to Figure 4–4:

(1) There are up to 62 PCLKs that can drive the SCLKs in each spine clock per quadrant in the largest device.

(2) There are up to 22 RCLKs that can drive the SCLKs in each spine clock per quadrant in the largest device.

Clock Regions

This section describes the following types of clock regions in Arria V devices:

- Entire device clock region
- Regional clock region
- Dual-regional clock region

Entire Device Clock Region

To form the entire device clock region, a source drives a GCLK network that can be routed through the entire device. The source is not necessarily a clock signal. This clock region has the maximum insertion delay when compared with other clock regions, but allows the signal to reach every destination in the device. It is a good option for routing global reset and clear signals or routing clocks throughout the device.

Regional Clock Region

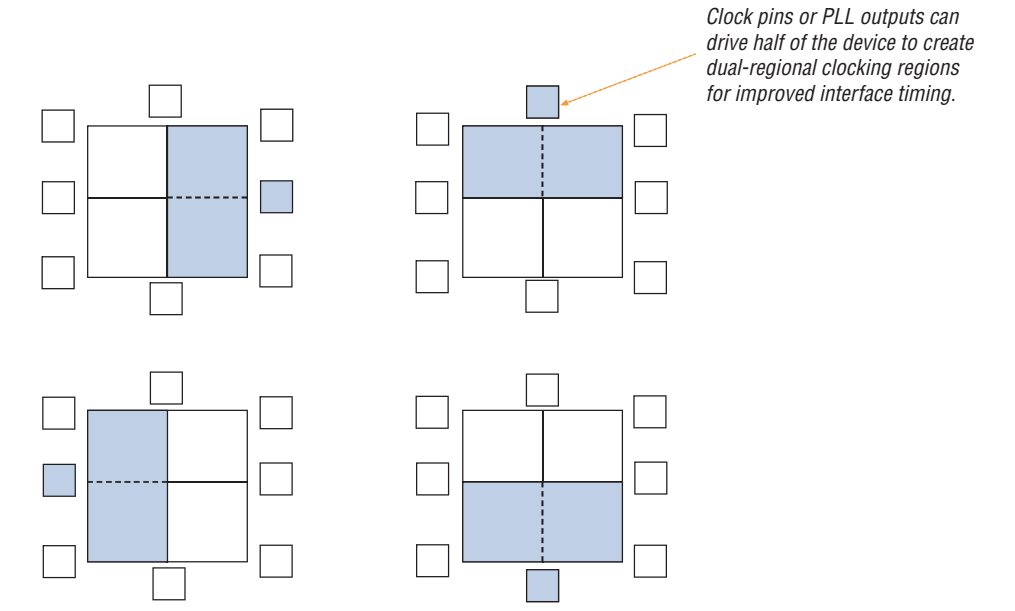
To form a regional clock region, a source drives a signal RCLK network that you can route throughout one quadrant of the device. This clock region provides the lowest skew in a quadrant. It is a good option if all the destinations are in a single quadrant.

Dual-Regional Clock Region

To form a dual-regional clock region, a single source (a clock pin or PLL output) generates a dual-regional clock by driving two RCLK networks (one from each quadrant). This technique allows destinations across two adjacent device quadrants to use the same low-skew clock. The routing of this signal on an entire side has approximately the same delay as a RCLK region. Internal logic can also drive a dual-regional clock network.

Figure 4–5 shows the dual-regional clock region.

Figure 4–5. Dual-Regional Clock Region for Arria V Devices



Clock Network Sources

In Arria V devices, clock input pins, PLL outputs, high-speed serial interface (HSSI) outputs, DPA outputs, and internal logic can drive the GCLK and RCLK networks. For connectivity between the dedicated clock pins, GCLK, and RCLK networks, refer to Table 4–2 and Table 4–3 on page 4–10.

Dedicated Clock Input Pins

CLK pins can be either differential clocks or single-ended clocks. Arria V devices support up to 24 differential clock inputs or 48 single-ended clock inputs. You can also use dedicated clock input pins CLK[23..0] for high fan-out control signals such as asynchronous clears, presets, and clock enables for protocol signals through the GCLK or RCLK networks. When used as single-ended clock inputs, the CLKn pins drive PLLs over global or regional clock networks.

Internal Logic

You can drive each GCLK, RCLK, and horizontal PCLK network using LAB-routing and row clock to enable internal logic to drive a high fan-out, low-skew signal.

Arria V PLLs cannot be driven by internally generated GCLKs, RCLKs, or horizontal PCLKs. The input clock to the PLL has to come from dedicated clock input pins or pin/PLL-fed GCLKs or RCLKs.

DPA Outputs

Every DPA generates one PCLK to the core.

HSSI Outputs

Every three HSSI outputs generate a group of six PCLKs to the core.

For more information about DPA and HSSI outputs, refer to the *High-Speed Differential I/O Interfaces with DPA in Arria V Devices* chapter.

PLL Clock Outputs

Arria V PLL clock outputs can drive both GCLK and RCLK networks.

Clock Input Pin Connections to GCLK and RCLK Networks

Table 4–2 lists the connection between the dedicated clock input pins and GCLKs.

Table 4–2. Clock Input Pin Connectivity to the GCLK Networks—*Preliminary*

Clock Resources	CLK (p/n Pins)																							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
GCLK0	Y	Y	Y	Y	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	Y	Y	Y	Y
GCLK1	Y	Y	Y	Y	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	Y	Y	Y	Y
GCLK2	Y	Y	Y	Y	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	Y	Y	Y	Y
GCLK3	Y	Y	Y	Y	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	Y	Y	Y	Y
GCLK4	—	—	—	—	Y	Y	Y	Y	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
GCLK5	—	—	—	—	Y	Y	Y	Y	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
GCLK6	—	—	—	—	Y	Y	Y	Y	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
GCLK7	—	—	—	—	Y	Y	Y	Y	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
GCLK8	—	—	—	—	—	—	—	—	Y *(1)*	Y *(1)*	Y *(1)*	Y *(1)*	Y	Y	Y	Y	—	—	—	—	—	—	—	—
GCLK9	—	—	—	—	—	—	—	—	Y *(1)*	Y *(1)*	Y *(1)*	Y *(1)*	Y	Y	Y	Y	—	—	—	—	—	—	—	—
GCLK10	—	—	—	—	—	—	—	—	Y *(1)*	Y *(1)*	Y *(1)*	Y *(1)*	Y	Y	Y	Y	—	—	—	—	—	—	—	—
GCLK11	—	—	—	—	—	—	—	—	Y *(1)*	Y *(1)*	Y *(1)*	Y *(1)*	Y	Y	Y	Y	—	—	—	—	—	—	—	—
GCLK12	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	Y	Y	Y	Y	—	—	—	—
GCLK13	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	Y	Y	Y	Y	—	—	—	—
GCLK14	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	Y	Y	Y	Y	—	—	—	—
GCLK15	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	Y	Y	Y	Y	—	—	—	—

Note to Table 4–2:

(1) This is applicable to all Arria V devices except Arria V GX A1 and A3 devices, and Arria V GT C3 device.

Table 4–3 lists the connectivity between the dedicated clock input pins and RCLKs in Arria V devices. A given clock input pin can drive two adjacent RCLK networks to create a dual-regional clock network.

Table 4–3. Clock Input Pin Connectivity to the RCLK Networks—*Preliminary* (Part 1 of 2)

Clock Resources	CLK (p/n pins)																							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
RCLK [58,59,60,61,62,63, 64,68,82,86]	Y	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
RCLK [58,59,60,61,62,63, 65,69,83,87]	—	Y	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
RCLK [58,59,60,61,62,63, 66,84]	—	—	Y	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
RCLK [58,59,60,61,62,63, 67,85]	—	—	—	Y	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
RCLK [20,24,28,30,34,38]	—	—	—	—	Y	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
RCLK [21,25,29,31,35,39]	—	—	—	—	—	Y	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
RCLK [22,26,32,36]	—	—	—	—	—	—	Y	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
RCLK [23,27,33,37]	—	—	—	—	—	—	—	Y	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
RCLK [52,53,54,55,56,57, 70,74,76,80]	—	—	—	—	—	—	—	—	Y *(1)*	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
RCLK [52,53,54,55,56,57, 71,75,77,81]	—	—	—	—	—	—	—	—	—	Y *(1)*	—	—	—	—	—	—	—	—	—	—	—	—	—	—
RCLK [52,53,54,55,56,57, 72,78]	—	—	—	—	—	—	—	—	—	—	Y *(1)*	—	—	—	—	—	—	—	—	—	—	—	—	—
RCLK [52,53,54,55,56,57, 73,79]	—	—	—	—	—	—	—	—	—	—	—	Y *(1)*	—	—	—	—	—	—	—	—	—	—	—	—
RCLK [46,47,48,49,50,51, 70,74,76,80]	—	—	—	—	—	—	—	—	—	—	—	—	Y *(2)*	—	—	—	—	—	—	—	—	—	—	—
RCLK [46,47,48,49,50,51, 71,75,77,81]	—	—	—	—	—	—	—	—	—	—	—	—	—	Y *(2)*	—	—	—	—	—	—	—	—	—	—
RCLK [46,47,48,49,50,51, 72,78]	—	—	—	—	—	—	—	—	—	—	—	—	—	—	Y *(2)*	—	—	—	—	—	—	—	—	—

Table 4–3. Clock Input Pin Connectivity to the RCLK Networks—*Preliminary* (Part 2 of 2)

Clock Resources	CLK (p/n pins)																						
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
RCLK [46,47,48,49,50,51, 73,79]	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	Y (2)	—	—	—	—	—	—	—	—
RCLK [0,4,8,10,14,18]	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	Y	—	—	—	—	—	—	—
RCLK [1,5,9,11,15,19]	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	Y	—	—	—	—	—	—
RCLK [2,6,12,16]	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	Y	—	—	—	—	—
RCLK [3,7,13,17]	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	Y	—	—	—	—
RCLK [40,41,42,43,44,45, 64,68,82,86]	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	Y	—	—	—
RCLK [40,41,42,43,44,45, 65,69,83,87]	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	Y	—	—
RCLK [40,41,42,43,44,45, 66,84]	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	Y	—
RCLK [40,41,42,43,44,45, 67,85]	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	Y

Notes to Table 4–3:

(1) This is applicable to all Arria V devices except Arria V GX A1 and A3 devices, and Arria V GT C3 device.

(2) Arria V GX A1 and A3 devices, and Arria V GT C3 device support additional RCLK clock resources, `RCLK [52..57]`.

Clock Output Connections

For Arria V PLL connectivity to GCLK and RCLK networks, refer to *PLL Connectivity to GCLK and RCLK Networks for Arria V Devices*.

Clock Control Block

Every GCLK, RCLK, and PCLK network has its own clock control block. The control block provides the following features:

- Clock source selection (dynamic selection available only for GCLKs)
- Global clock multiplexing
- Clock power down (static or dynamic clock enable or disable available only for GCLKs and RCLKs)

Table 4–4 lists the mapping between the input clock pins, PLL counter outputs, and clock control block inputs.

Table 4–4. Mapping Between the Input Clock Pins, PLL Counter Outputs, and Clock Control Block Inputs

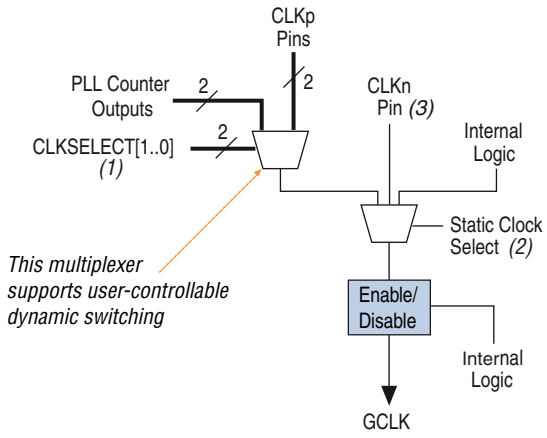
Clock	Fed by
`inclk[0]` and `inclk[1]`	Any of the four dedicated clock pins on the same side of the Arria V device.
`inclk[2]`	PLL counters C0 and C2 from the two center PLLs on the same side of the Arria V devices.
`inclk[3]`	PLL counters C1 and C3 from the two center PLLs on the same side of the Arria V devices.

CAUTION: You cannot use corner PLLs for dynamic clock control selection.

GCLK Control Block

You can select the clock source for the GCLK select block either statically or dynamically using internal logic to drive the multiplexer-select inputs. When selecting the clock source dynamically, you can select either PLL outputs (such as `C0` or `C1`) or a combination of clock pins or PLL outputs. Figure 4–6 shows the GCLK control block.

Figure 4–6. GCLK Control Block for Arria V Devices



Notes to Figure 4–6:

(1) When the device is in user mode, you can dynamically control the clock select signals through internal logic.

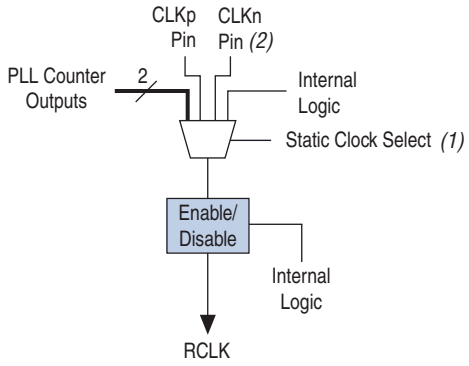
(2) When the device is in user mode, you can only set the clock select signals through a configuration file (SRAM object file [**.sof**] or programmer object file [**.pof**]) because the signals cannot be controlled dynamically.

(3) The `CLKn` pin is not a dedicated clock input when used as a single-ended PLL clock input. The `CLKn` pin can drive the PLL using the GCLK.

RCLK Control Block

You can only control the clock source selection for the RCLK select block statically using configuration bit settings in the configuration file (**.sof** or **.pof**) generated by the Quartus II software. Figure 4–7 shows the RCLK control block.

Figure 4–7. RCLK Control Block



Notes to Figure 4–7:

(1) When the device is in user mode, you can only set the clock select signals through a configuration file (**.sof** or **.pof**); they cannot be controlled dynamically.

(2) The `CLKn` pin is not a dedicated clock input when used as a single-ended PLL clock input. The `CLKn` pin can drive the PLL using the RCLK.

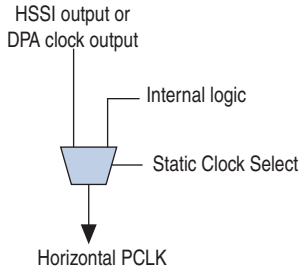
You can set the input clock sources and the `clkena` signals for the GCLK and RCLK network multiplexers through the Quartus II software using the ALTCLKCTRL megafunction.

For more information about ALTCLKCTRL megafunction, refer to *Clock Control Block (ALTCLKCTRL) Megafunction User Guide*.

PCLK Control Block

You can select the HSSI output or internal logic to drive the HSSI horizontal PCLK control block. Alternatively, you can also use the DPA clock output or internal logic to drive the DPA horizontal PCLK. You can only use the DPA clock output to generate the vertical PCLK to the core. Figure 4–8 shows the PCLK control block.

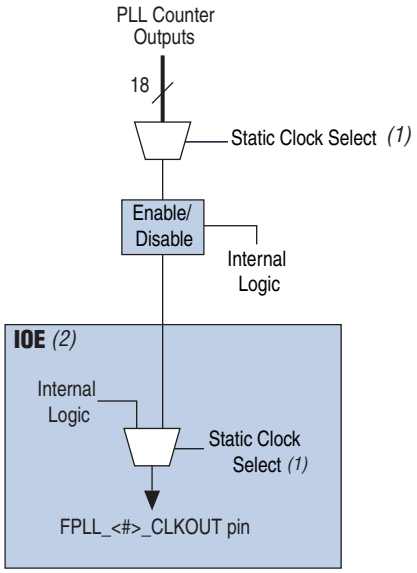
Figure 4–8. Horizontal PCLK Control Block



External PLL Clock Output Control Block

You can enable or diable the dedicated external clock output pins using the ALTCLKCTRL megafunction. Figure 4–9 shows the external PLL output clock control block.

Figure 4–9. External PLL Output Clock Control Block for Arria V Devices



Notes to Figure 4–9:

(1) When the device is in user mode, you can only set the clock select signals through a configuration file (**.sof** or **.pof**); they cannot be controlled dynamically.

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

Clock Power Down

You can power down the Arria V GCLK and RCLK clock networks using both static and dynamic approaches.

When a clock network is powered down, all the logic fed by the clock network is in off-state, thereby reducing the overall power consumption of the device. The unused GCLK, RCLK, and PCLK networks are automatically powered down through configuration bit settings in the configuration file (**.sof** or **.pof**) generated by the Quartus II software.

The dynamic clock enable or disable feature allows the internal logic to control power-up or power-down synchronously on the GCLK and RCLK networks, including dual-regional clock regions.

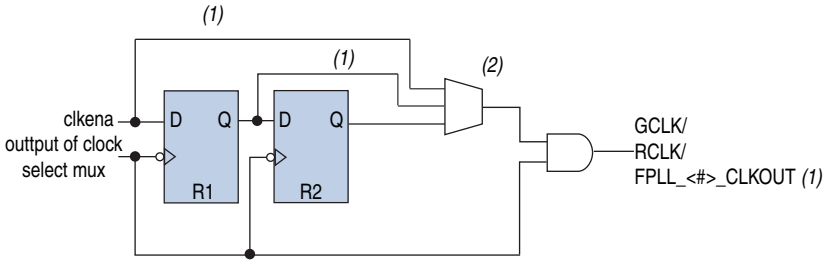
☞ You cannot dynamically enable or disable GCLK or RCLK networks that drive PLLs.

Clock Enable Signals

Figure 4–10 shows the implementation of the clock enable and disable circuit of the clock control block in Arria V devices.

You cannot use the clock enable and disable circuit of the clock control block if the GCLK or RCLK output drives the input of a PLL.

Figure 4–10. clkena Implementation



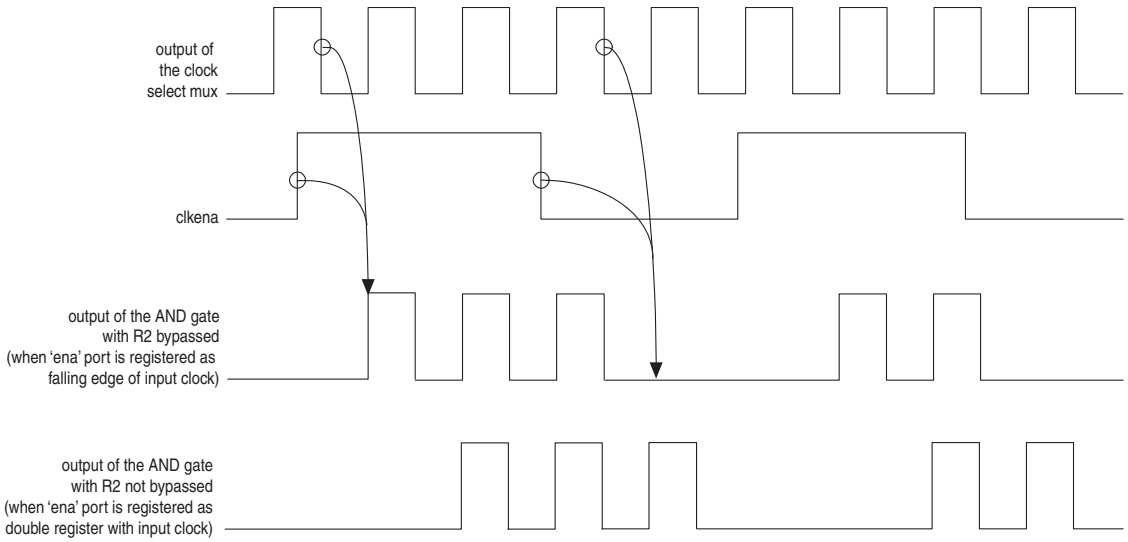
Notes to Figure 4–10:

(1) The R1 and R2 bypass paths are not available for the PLL external clock outputs.
(2) The select line is statically controlled by a bit setting in the **.sof** or **.pof**.

In Arria V devices, the `clkena` signals are supported at the clock network level instead of at the PLL output counter level. This allows you to gate off the clock even when you are not using a PLL. You can also use the `clkena` signals to control the dedicated external clocks from the PLLs.

Figure 4–11 shows a waveform example for a clock output enable. The clkena signal is synchronous to the falling edge of the clock output.

Figure 4–11. clkena Signals *(1)*



Note to Figure 4–11:

(1) Use the clkena signals to enable or disable the GCLK and RCLK networks or the FPLL_<#>_CLKOUT pins.

Arria V devices have an additional metastability register that aids in asynchronous enable and disable of the GCLK and RCLK networks. You can optionally bypass this register in the Quartus II software.

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. The clkena signal can also disable clock outputs if the system is not tolerant of frequency overshoot during resynchronization.

Arria V PLLs

PLLs provide robust clock management and synthesis for device clock management, external system clock management, and high-speed I/O interfaces.

The Arria V device family contains fractional PLLs in addition to the existing integer PLLs. Two adjacent fractional PLLs share 18 output counters that support integer or fractional frequency synthesis. Arria V devices offer up to 16 fractional PLLs in the larger densities. All Arria V fractional PLLs have the same core analog structure and features support.

Table 4–5 lists the features in Arria V PLLs.

Table 4–5. PLL Features in Arria V Devices —*Preliminary*

Feature	Support
Integer PLL	Yes
Fractional PLL	Yes
C output counters	18
M, N, C counter sizes	1 to 512
Dedicated external clock outputs	4 single-ended or 2 single-ended and 1 differential
Dedicated clock input pins	4 single-ended or 4 differential
External feedback input pin	Single-ended or differential
Spread-spectrum input clock tracking	Yes (1)
Source synchronous compensation	Yes
Direct compensation	Yes
Normal compensation	Yes
Zero-delay buffer compensation	Yes
External feedback compensation	Yes
LVDS compensation	Yes
VCO output drives the DPA clock	Yes
Phase shift resolution	78.125 ps (2)
Programmable duty cycle	Yes

Notes to Table 4–5:

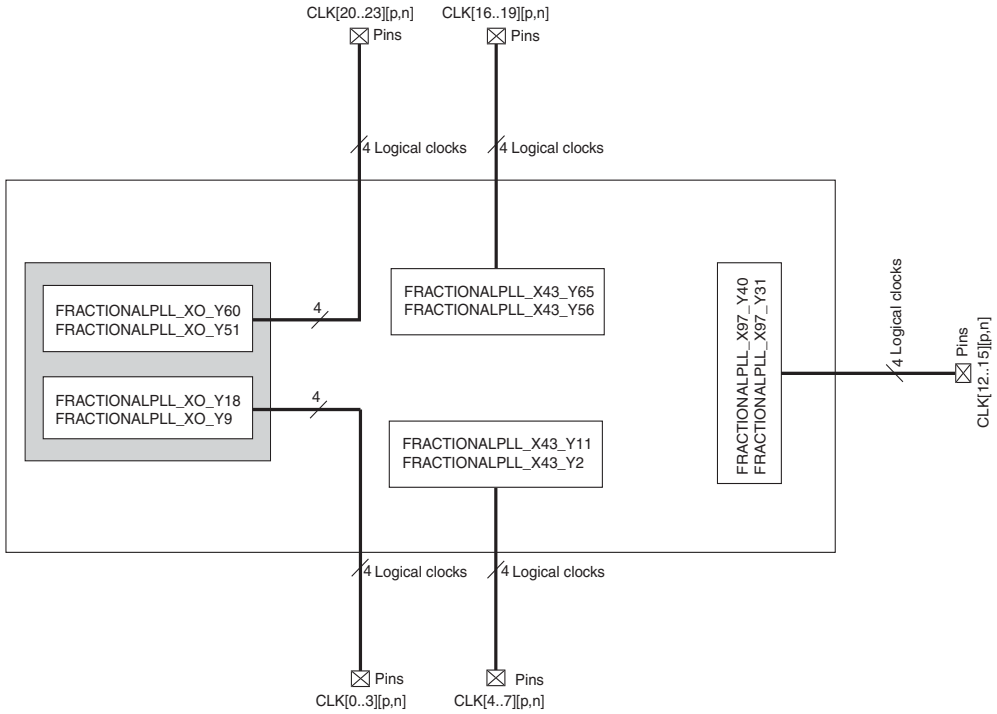
(1) Provided input clock jitter is within input jitter tolerance specifications and the modulation frequency of the input clock is below the PLL bandwidth which is specified in the Fitter report.

(2) The smallest phase shift is determined by the voltage-controlled oscillator (VCO) period divided by eight. For degree increments, the Arria V device can shift all output frequencies in increments of at least 45°. Smaller degree increments are possible depending on the frequency and divide parameters.

Figure 4–12 through Figure 4–15 on page 4–22 show the physical locations of the fractional PLLs. For single-ended clock inputs, only the CLK<#>p pin has a dedicated connection to the PLL. If you use the CLK<#>n pin, a global or regional clock is used.

Driving a PLL over a global or regional clock can lead to higher jitter at the PLL input, and the PLL will not be able to fully compensate for the global or regional clock. Altera® recommends to use the CLKp pin for optimal performance when you use single-ended clock inputs to drive the PLLs.

Figure 4–12. PLL Locations for Arria V GX A1 and A3 Devices, and Arria V GT C3 (1) Device

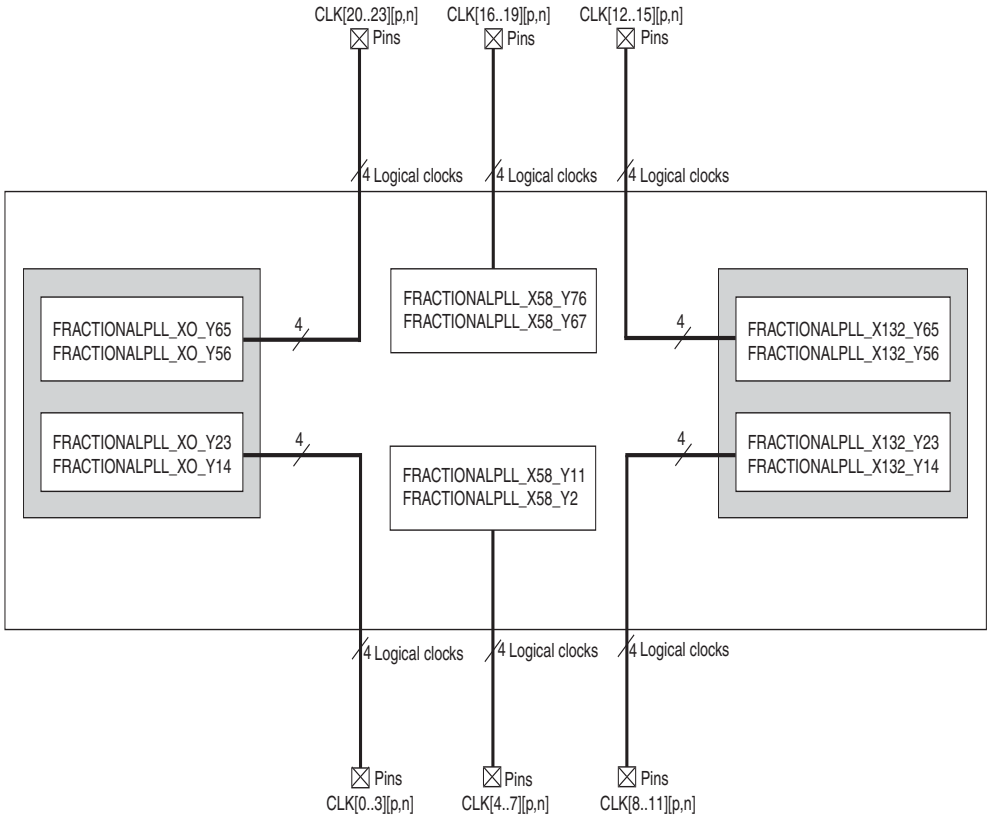


Notes to Figure 4–12:

(1) Fractional PLL coordinates for Arria V GT C3 device will be finalized in the future release of the Quartus II software.

(2) Every index represents one fractional PLL in the device. The physical locations of the fractional PLLs correspond to the coordinates in the Quartus II Chip Planner.

Figure 4–13. PLL Locations for Arria V GX A5 and A7 Devices, and Arria V GT C7 (1) Device

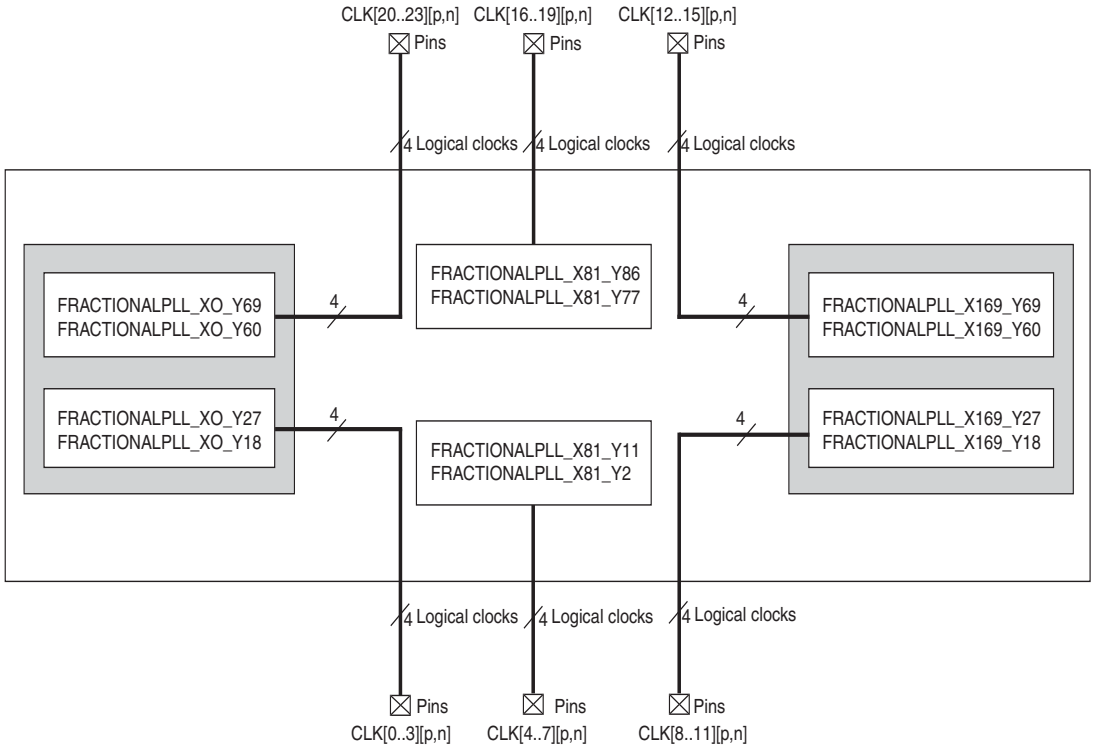


Notes to Figure 4–13:

(1) Fractional PLL coordinates for Arria V GT C7 device will be finalized in the future release of the Quartus II software.

(2) Every index represents one fractional PLL in the device. The physical locations of the fractional PLLs correspond to the coordinates in the Quartus II Chip Planner.

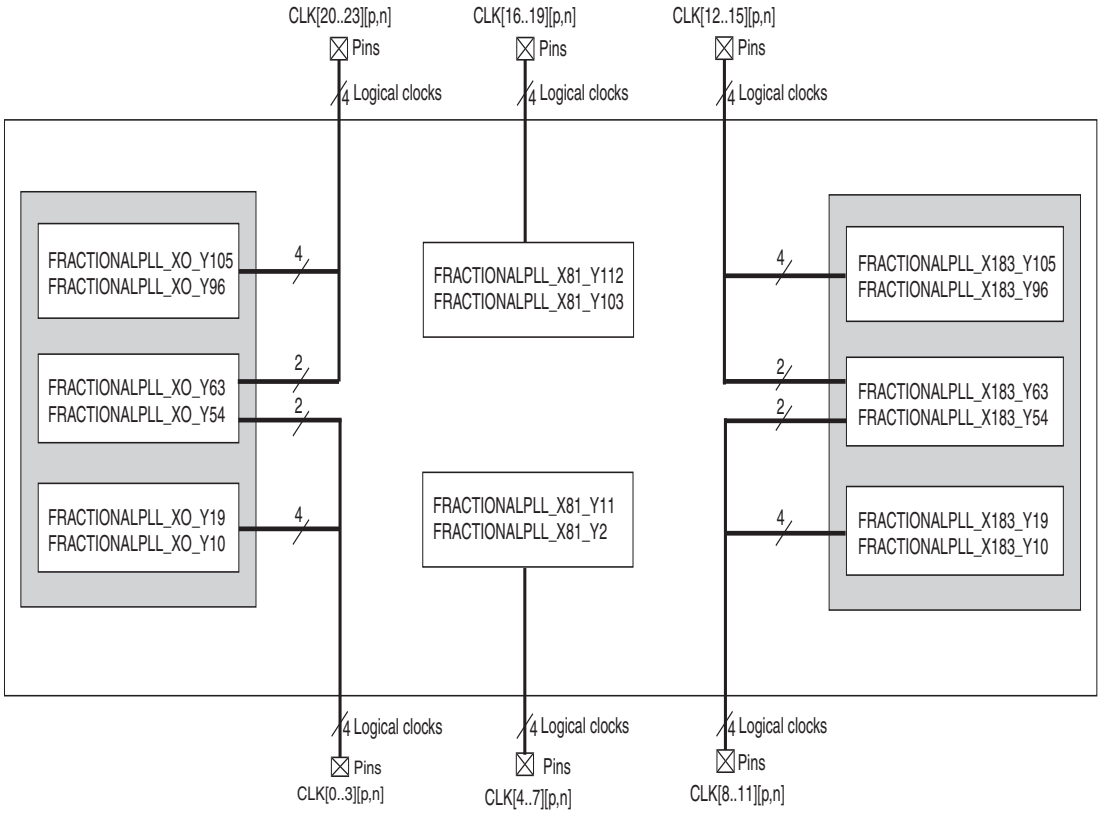
Figure 4–14. PLL Locations for Arria V GX B1 and B3 Devices, and Arria V GT D3 Device



Note to Figure 4–14:

(1) Every index represents one fractional PLL in the device. The physical locations of the fractional PLLs correspond to the coordinates in the Quartus II Chip Planner.

Figure 4–15. PLL Locations for Arria V GX B5 and B7 Devices, and Arria V GT D7 Device



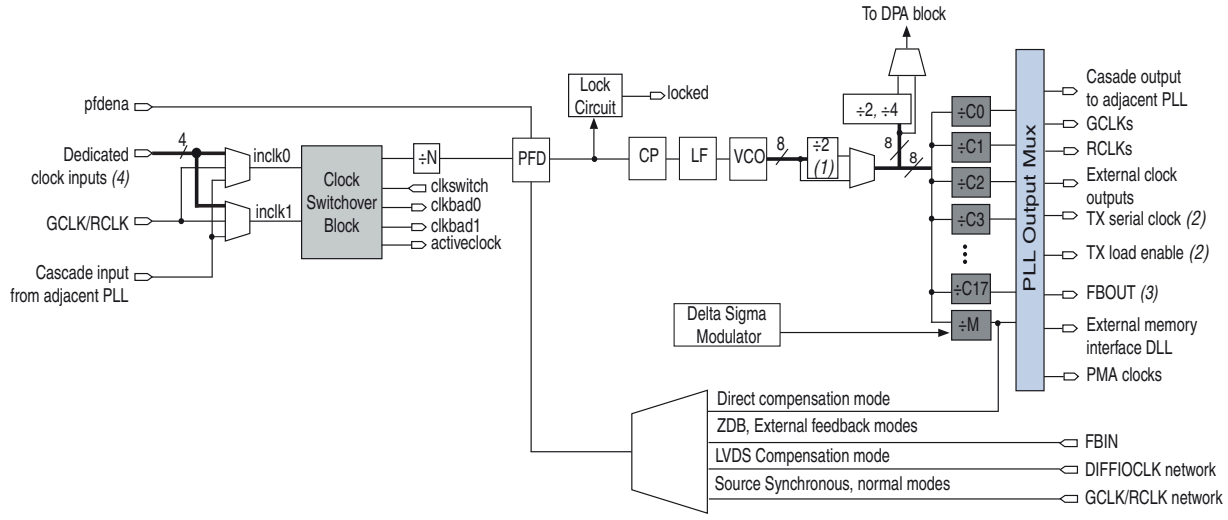
Note to Figure 4–15:

(1) Every index represents one fractional PLL in the device. The physical locations of the fractional PLLs correspond to the coordinates in the Quartus II Chip Planner.

Fractional PLL Architecture

Figure 4–16 shows the high-level block diagram of the Arria V fractional PLL.

Figure 4–16. Fractional PLL High-Level Block Diagram



Notes to Figure 4–16:

(1) This is the VCO post-scale counter `K`.

(2) Only `C0`, `C2`, `C15`, and `C17` can drive the TX serial clock and `C1`, `C3`, `C14`, and `C16` can drive the TX load enable.

(3) This `FBOUT` port is fed by the `M` counter in the Arria V PLLs.

(4) For single-ended clock inputs, only the `CLK<#>p` pin has a dedicated connection to the PLL. If you use the `CLK<#>n` pin, a global or regional clock is used.

Fractional PLL Usage

You can configure the fractional PLL to function either in the integer or in the enhanced fractional mode. One fractional PLL can use up to 18 output counters and all external clock outputs.

Fractional PLLs can be used as follows:

- Reduce the number of required oscillators on the board
- Reduce the clock pins used in the FPGA by synthesizing multiple clock frequencies from a single reference clock source
- Clock network delay compensation
- Zero delay buffering
- Transmit clocking for transceivers

PLL External Clock I/O Pins

Two adjacent fractional PLLs share four dual-purpose clock I/O pins, organized as one of the following combinations:

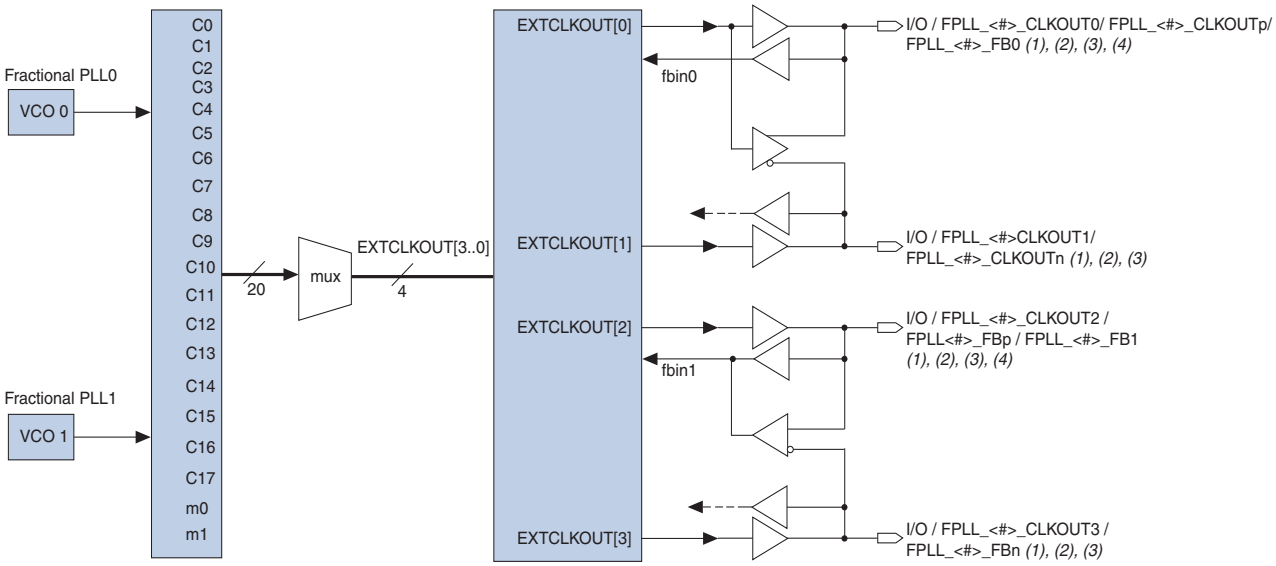
- Four single-ended clock outputs
- Two single-ended outputs and one differential clock output

- Four single-ended clock outputs and two single-ended feedback inputs within the I/O driver feedback for zero delay buffer (ZDB) mode support
- Two single-ended clock outputs and two single-ended feedback inputs for single-ended external feedback (EFB) mode support
- One differential clock output and one differential feedback input for differential EFB support (only one of the two adjacent fractional PLLs can support differential EFB while the other fractional PLL can be used for general-purpose clocking)

The center fractional PLLs on the left and right sides of Arria V GX B5 and B7 devices, and Arria V GT D7 device do not support external clock outputs.

Figure 4–17 shows the dual-purpose clock I/O pins associated with the PLL for Arria V devices.

Figure 4–17. Dual-Purpose Clock I/O Pins Associated with PLL for Arria V Devices



Notes to Figure 4–17:

(1) You can feed these clock output pins using any one of the `C[17..0]` or `M` counters. When not used as external clock outputs, these clock output pins can be used as regular user I/Os.

(2) The `FPLL_<#>_CLKOUT0`, `FPLL_<#>_CLKOUT1`, `FPLL_<#>_CLKOUT2`, and `FPLL_<#>_CLKOUT3` pins are single-ended clock output pins.

(3) The `FPLL_<#>_CLKOUTp` and `FPLL_<#>_CLKOUTn` pins are differential output pins while the `FPLL_<#>_FBp` and `FPLL_<#>_FBn` pins are differential feedback input pins to support differential EFB only in VCO 1.

(4) The `FPLL_<#>_FB0` and `FPLL_<#>_FB1` pins are single-ended feedback input pins.

Figure 4–17 shows that any of the output counters (C[17..0]) on the PLLs or the M counter can feed the dedicated external clock outputs. Therefore, one counter or frequency can drive all output pins available from a given PLL.

Each pin of a single-ended output pair can be either in-phase or 180° out-of-phase. The Quartus II software places the NOT gate in the design into the IOE to implement the 180° 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 as well as LVDS, differential high-speed transceiver logic (HSTL), and differential SSTL. Arria V PLLs can also drive out to any regular I/O pin through the GCLK or RCLK network. You can also use the external clock output pins as user I/O pins if you do not require external PLL clocking.

For more information about I/O standards supported by the PLL clock input and output pins, refer to *I/O Features in Arria V Devices* chapter.

PLL Control Signals

You can use the pfdena, areset, and locked signals to control and observe PLL operation and resynchronization.

pfdena

The pfdena signal controls the phase frequency detector (PFD) output with a programmable gate.

Use the pfdena signal to maintain the most recent locked frequency so that your system has time to store its current settings before shutting down. If you disable PFD, the VCO operates at its most recent set value of control voltage and frequency, with some long-term drift to a lower frequency. The PLL continues running even if it goes out-of-lock or the input clock is disabled.

You can use either your own control signal or the control signals available from the clock switchover circuit (activeclock, clkbad[0], or clkbad[1]) to control 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 areset is driven high, the PLL counters reset, clearing the PLL output and placing the PLL out-of-lock. The VCO is then set back to its nominal setting. When areset is driven low again, the PLL resynchronizes to its input as it re-locks.

You must assert the areset signal every time the PLL loses lock to guarantee the correct phase relationship between the PLL input and output clocks. You can set up the PLL to automatically reset (self reset) after a loss-of-lock condition using the Quartus II MegaWizard™ Plug-In Manager. You must include the areset signal if either of the following conditions is true:

- PLL reconfiguration or clock switchover is enabled in the design
- Phase relationships between the PLL input and output clocks must be maintained after a loss-of-lock condition

If the input clock to the PLL is not toggling or is unstable after power up, assert the areset signal after the input clock is stable and within specifications.

locked

The locked signal output of the PLL indicates that the PLL has locked onto the reference clock and the PLL clock outputs are operating at the desired phase and frequency set in the MegaWizard Plug-In Manager. The lock detection circuit provides a signal to the core logic that gives an indication when the feedback clock has locked onto the reference clock both in phase and frequency.

Altera recommends using the areset and locked signals in your designs to control and observe the status of your PLL.

Clock Feedback Modes

This section describes the following clock feedback modes:

- Source synchronous
- LVDS compensation
- Direct compensation
- Normal
- ZDB
- EFB

Each mode allows clock multiplication and division, phase shifting, and programmable duty cycle.

The input and output delays are fully compensated by a PLL only when using the dedicated clock input pins associated with a given PLL as the clock source.

When a RCLK or GCLK network drives the PLL or the PLL is driven by a dedicated clock pin that is not associated with the PLL, the input and output delays may not be fully compensated in the Quartus II software.

For example, when you configure a PLL in ZDB mode and the PLL input is driven by an associated dedicated clock input pin. In this configuration, a fully compensated clock path results in zero delay between the clock input and one of the clock outputs from the PLL. However, if the PLL input is fed by a non-dedicated input (using the GCLK network), the clock output may not be perfectly aligned with the input clock.

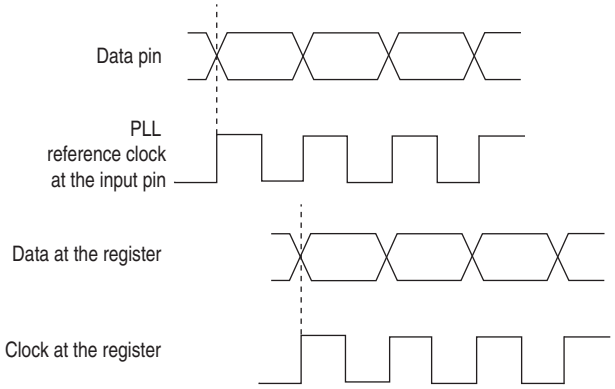
Source Synchronous Mode

If the data and clock arrive at the same time on the input pins, the same phase relationship is maintained at the clock and data ports of any IOE input register. Data and clock signals at the IOE experience similar buffer delays as long as you use the same I/O standard.

Altera recommends source synchronous mode for source-synchronous data transfers.

Figure 4–18 shows a waveform example of the clock and data in this mode.

Figure 4–18. Phase Relationship Between Clock and Data in Source Synchronous Mode



Source synchronous mode compensates for the delay of the clock network used and any difference in the delay between the following two paths:

- Data pin to the IOE register input
- Clock input pin to the PLL PFD input

The Arria V PLL can compensate multiple pad-to-input-register paths, such as a data bus when it is set to use source-synchronous compensation mode.

Use the "PLL Compensation" assignment in the Quartus II software Assignment Editor to select the input pins to be used as the PLL compensation targets. You can include your entire data bus, provided the input registers are clocked by the same output of a source-synchronous-compensated PLL. To compensate for the clock delay properly, all of the input pins must be on the same side of the device. The PLL compensates for the input pin with the longest pad-to-register delay among all input pins in the compensated bus.

If you do not make the "PLL Compensation" assignment, the Quartus II software automatically selects all of the pins driven by the compensated output of the PLL as the compensation target.

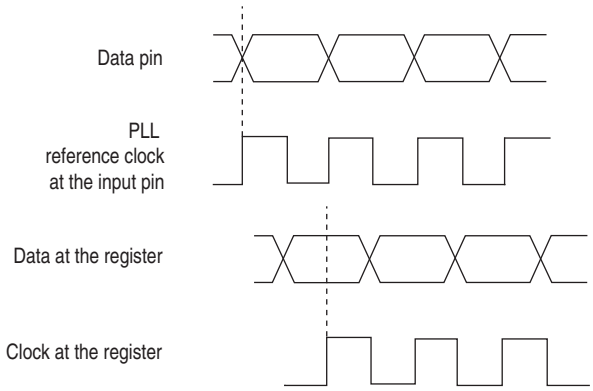
LVDS Compensation Mode

The purpose of source synchronous mode is to maintain the same data and clock timing relationship seen at the pins of the internal serializer/deserializer (SERDES) capture register, except that the clock is inverted (180° phase shift). Thus, source synchronous mode ideally compensates for the delay of the LVDS clock network including the difference in delay between the following two paths:

- Data pin-to-SERDES capture register
- Clock input pin-to-SERDES capture register. In addition, the output counter must provide the 180° phase shift

Figure 4–19 shows a waveform example of the clock and data in LVDS mode.

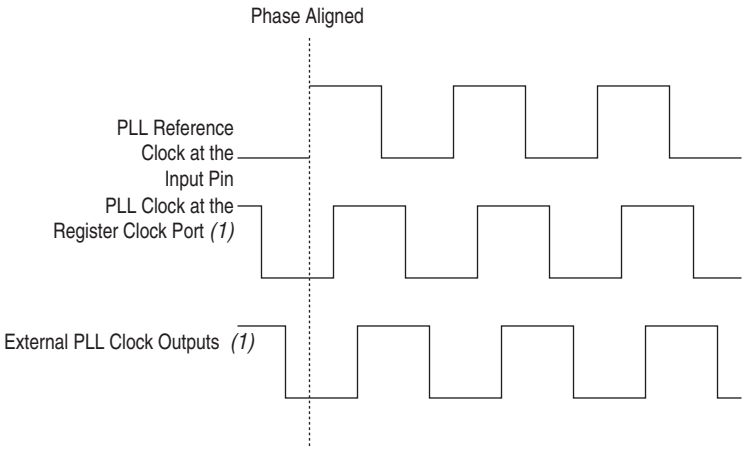
Figure 4–19. Phase Relationship Between the Clock and Data in LVDS Mode



Direct Compensation Mode

In direct compensation mode, the PLL does not compensate for any clock networks. This mode provides better jitter performance because the clock feedback into the PFD passes through less circuitry. Both the PLL internal- and external-clock outputs are phase-shifted with respect to the PLL clock input. Figure 4–20 shows a waveform example of the PLL clocks' phase relationship in direct compensation mode.

Figure 4–20. Phase Relationship Between the PLL Clocks in Direct Compensation Mode



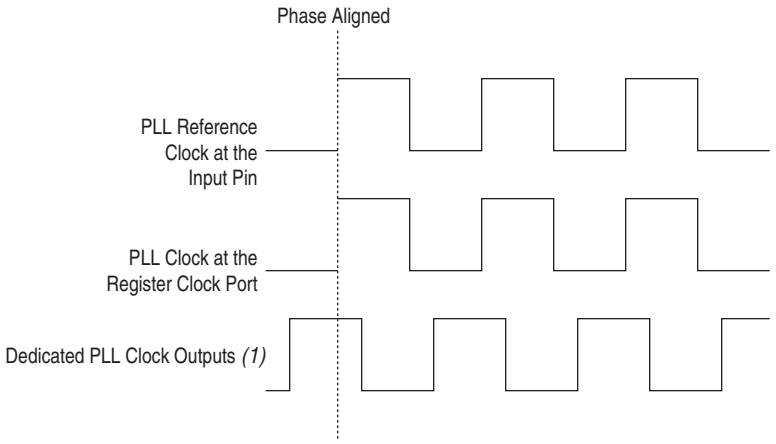
Note to Figure 4–20:

(1) The PLL clock outputs lag the PLL input clocks depending on routing delays.

Normal Mode

An internal clock in normal mode is phase-aligned to the input clock pin. The external clock-output pin has a phase delay relative to the clock input pin if connected in this mode. The Quartus II TimeQuest Timing Analyzer reports any phase difference between the two. In normal mode, the delay introduced by the GCLK or RCLK network is fully compensated. Figure 4–21 shows the phase relationship waveform example of the PLL clocks in normal mode.

Figure 4–21. Phase Relationship Between the PLL Clocks in Normal Mode



Note to Figure 4–21:

(1) The external clock output can lead or lag the PLL internal clock signals.

Zero-Delay Buffer Mode

In ZDB mode, the external clock output pin is phase-aligned with the clock input pin for zero delay through the device. This mode is supported on all Arria V PLLs.

When using this mode, you must use the same I/O standard on the input clocks and clock outputs to guarantee clock alignment at the input and output pins. You cannot use differential I/O standards on the PLL clock input or output pins. In Arria V devices, ZDB mode can support up to four single-ended clock outputs.

For more information about PLL clock outputs, refer to the "PLL External Clock I/O Pins" on page 4–23.

To ensure phase alignment between the `clk` pin and the external clock output (`CLKOUT`) pin in ZDB mode, along with single-ended I/O standards, instantiate a bidirectional I/O pin in the design to serve as the feedback path connecting the `fbout` and `fbin` ports of the PLL. The PLL uses this bidirectional I/O pin to mimic, and compensate for, the output delay from the clock output port of the PLL to the external clock output pin.

The bidirectional I/O pin that you instantiate in your design must always be assigned a single-ended I/O standard.

To avoid signal reflection when using ZDB mode, do not place board traces on the bidirectional I/O pin.

Figure 4–22 shows ZDB mode in Arria V PLLs.

Figure 4–22. ZDB Mode in Arria V PLLs

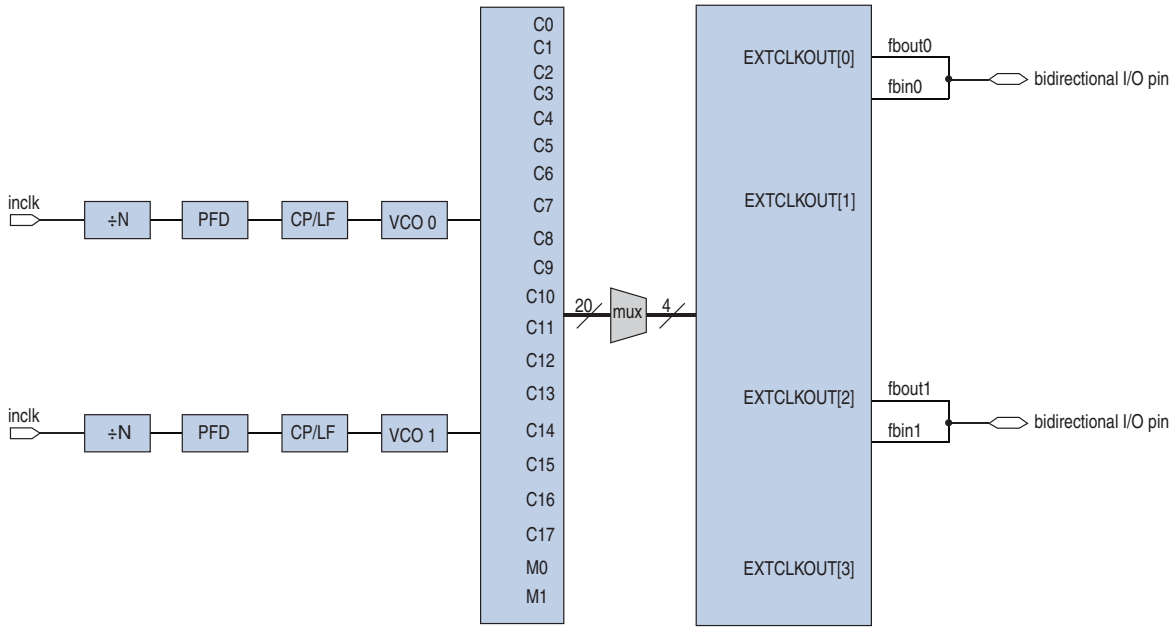
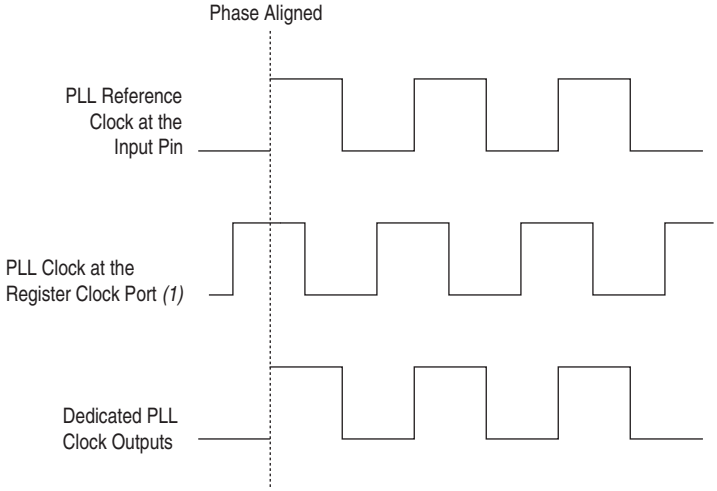


Figure 4–23 shows a waveform example of the PLL clocks' phase relationship in ZDB mode.

Figure 4–23. Phase Relationship Between the PLL Clocks in ZDB Mode



Note to Figure 4–23:

(1) The internal PLL clock output can lead or lag the external PLL clock outputs.

External Feedback Mode

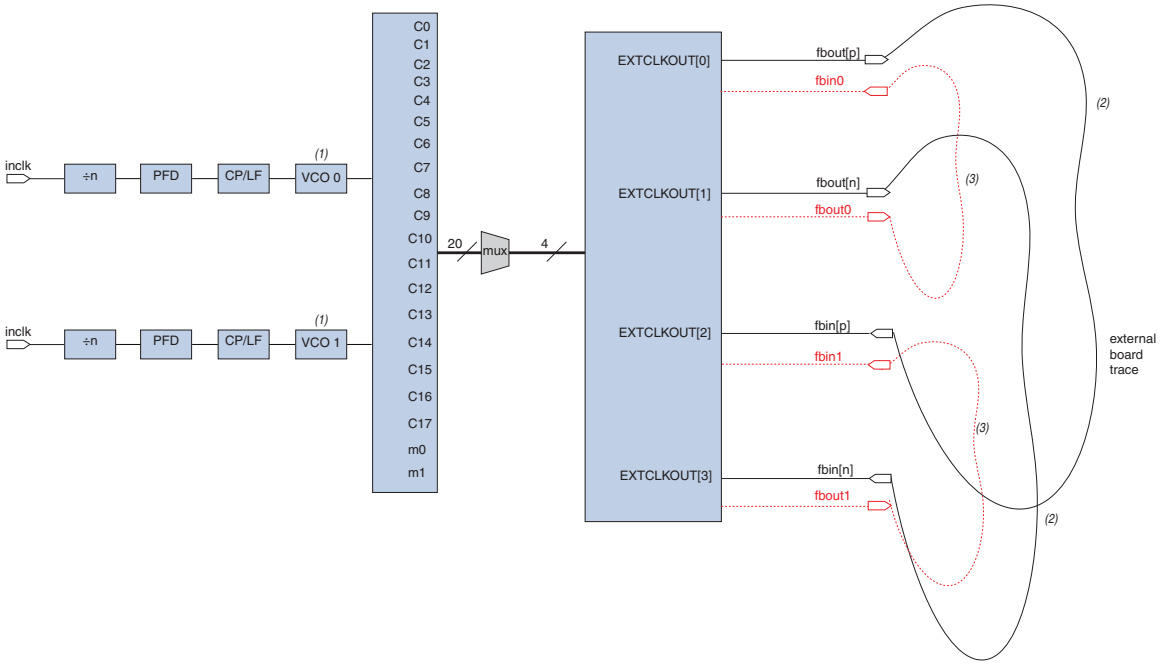
In EFB mode, the output of the M counter (fbout) feeds back to the PLL fbin input (using a trace on the board) and becomes part of the feedback loop.

One of the dual-purpose external clock outputs becomes the fbin input pin in this mode. The external feedback input pin, fbin is phase-aligned with the clock input pin. Aligning these clocks allows you to remove clock delay and skew between devices.

When using EFB mode, you must use the same I/O standard on the input clock, feedback input, and clock outputs.

Figure 4–24 shows the EFB mode in Arria V devices.

Figure 4–24. EFB Mode in Arria V Devices *(1)*

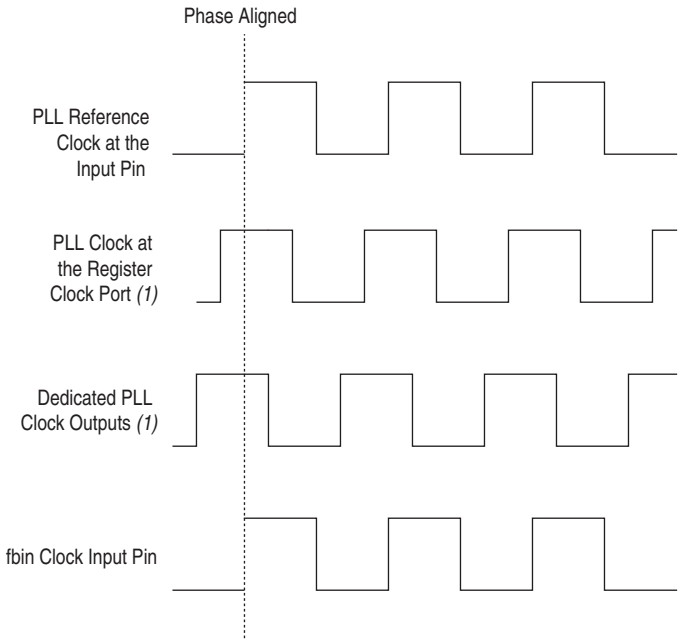


Notes to Figure 4–24:

(1) Only one of the two VCOs can support differential EFB mode at one time while you can use the other VCO for general purpose clocking.
(2) External board connection for one differential clock output and one differential feedback input for differential EFB support.
(3) External board connection for two single-ended clock outputs and two single-ended feedback inputs for single-ended EFB support.

Figure 4–25 shows a waveform example of the phase relationship between the PLL clocks in EFB mode.

Figure 4–25. Phase Relationship Between the PLL Clocks in EFB Mode *(1)*



Note to Figure 4–25:

(1) The PLL clock outputs can lead or lag the `fbin` clock input.

Clock Multiplication and Division

Each Arria V PLL provides clock synthesis for PLL output ports using the $(K \times M)/(N \times C)$ scaling factors. The input clock is divided by a pre-scale factor, `N`, and is then multiplied by the `K` and `M` feedback factors. The control loop drives the VCO to match $f_{in} \times (K \times M/N)$.

A post-scale counter, `K`, is inserted after the VCO. When you enable the VCO post-scale counter, the counter divides the VCO frequency by two. When the `K` counter is bypassed, the VCO frequency goes to the output port without being divided by two.

Each output port has a unique post-scale counter, `C`, that divides down the output from the `K` counter. 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 the output frequencies required from one PLL are 33 and 66 MHz, the Quartus II software sets the VCO to 660 MHz (the least common multiple of 33 and 66 MHz within the VCO range). Then the post-scale counters, `C`, scale down the VCO frequency for each output port.

Each PLL has one pre-scale counter, N, and one multiply counter, M, with a range of 1 to 512 for both M and N. For PLL operating in integer mode, M is an integer value. The delta-sigma modulator (DSM) is used together with M multiply counter to enable the PLL to operate in fractional mode. The DSM dynamically changes the M counter divide value on a cycle to cycle basis. The different M counter values lets the "average" M counter value to be a non-integer. In fractional mode, the M counter divide value equals to the sum of the "clock high" count, "clock low" count, and the fractional value. The fractional value is equal to K[31..0]/2^24. The PLL operates in integer mode when DSM is disabled.

The N counter does not use duty-cycle control because the only purpose of this counter is to calculate frequency division. The post-scale counters range from 1 to 512 with a 50% duty cycle setting. The high- and low-count values for each counter range from 1 to 256. The sum of the high- and low-count values chosen for a design selects the divide value for a given counter.

The Quartus II software automatically chooses the appropriate scaling factors according to the input frequency, multiplication, and division values entered into the Altera PLL megafunction.

Programmable Duty Cycle

The programmable duty cycle allows PLLs to generate clock outputs with a variable duty cycle. This feature is supported on the PLL post-scale counters.

The duty-cycle setting is achieved by a low and high time-count setting for the post-scale counters. To determine the duty cycle choices, the Quartus II software uses the frequency input and the required multiply or divide rate.

The post-scale counter value determines the precision of the duty cycle. The precision is defined as 50% divided by the post-scale counter value. For example, if the C0 counter is 10, steps of 5% are possible for duty-cycle choices from 5% to 90%.

If the PLL is in external feedback mode, 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.

Clock Switchover

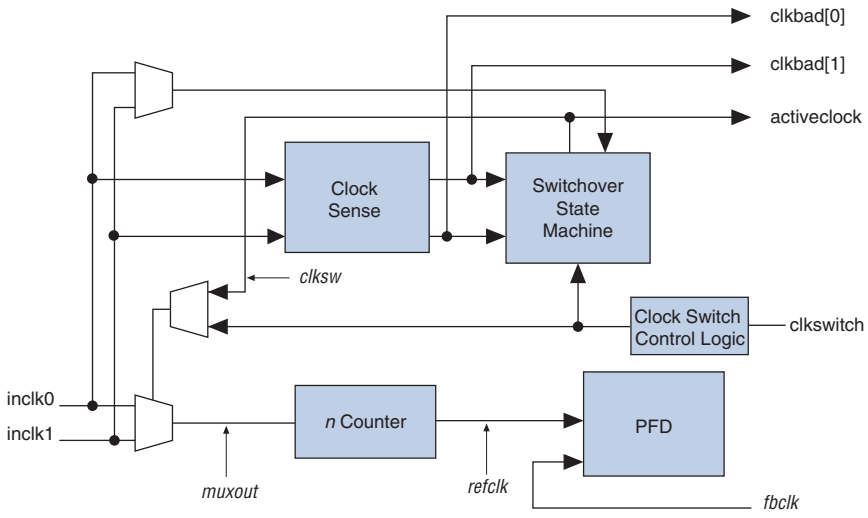
The clock switchover feature allows the PLL to switch between two reference input clocks. Use this feature for clock redundancy or for a dual-clock domain application such as in a system that turns on the redundant clock if the previous clock stops running. The design can perform clock switchover automatically when the clock is no longer toggling or based on a user control signal, clkswitch.

The following clock switchover modes are supported in Arria V PLLs:

- Automatic switchover—The clock sense circuit monitors the current reference clock and if it stops toggling, automatically switches to the other `inclk0` or `inclk1` clock.
- Manual clock switchover—Clock switchover is controlled using the `clkswitch` signal. When the `clkswitch` signal goes from logic low to logic high, and stays high for at least three clock cycles, the reference clock to the PLL is switched from `inclk0` to `inclk1`, or vice-versa.
- Automatic switchover with manual override—This mode combines automatic switchover and manual clock switchover. When the `clkswitch` signal goes high, it overrides the automatic clock switchover function. As long as the `clkswitch` signal is high, further switchover action is blocked.

Arria V PLLs support a fully configurable clock switchover capability. Figure 4–26 shows a block diagram of the automatic switchover circuit built into the PLL. When the current reference clock is not present, the clock sense block automatically switches to the backup clock for PLL reference. The clock switchover circuit also sends out three status signals—`clkbad[0]`, `clkbad[1]`, and `activeclock`—from the PLL to implement a custom switchover circuit in the logic array. You can select a clock source as the backup clock by connecting it to the `inclk1` port of the PLL in your design.

Figure 4–26. Automatic Clock Switchover Circuit Block Diagram



Automatic Clock Switchover

Use the switchover circuitry to automatically switch between `inclk0` and `inclk1` when the current reference clock to the PLL stops toggling. You can switch back and forth between `inclk0` and `inclk1` any number of times when one of the two clocks fails and the other clock is available.

For example, in applications that require a redundant clock with the same frequency as the reference clock, the switchover state machine generates a signal (`clksw`) that controls the multiplexer select input, as shown in Figure 4–26. In this case, `inclk1` becomes the reference clock for the PLL.

When using automatic clock switchover mode, the following requirements must be satisfied:

- Both clock inputs must be running
- The period of the two clock inputs can differ by no more than 20%

If the current clock input stops toggling while the other clock is also not toggling, switchover is not initiated and the `clkbad[0..1]` signals are not valid. Also, if both clock inputs are not the same frequency, but their period difference is within 20%, the clock sense block detects when a clock stops toggling, but the PLL may lose lock after the switchover is completed and needs time to relock.

Altera recommends resetting the PLL using the `areset` signal to maintain the phase relationships between the PLL input and output clocks when using clock switchover.

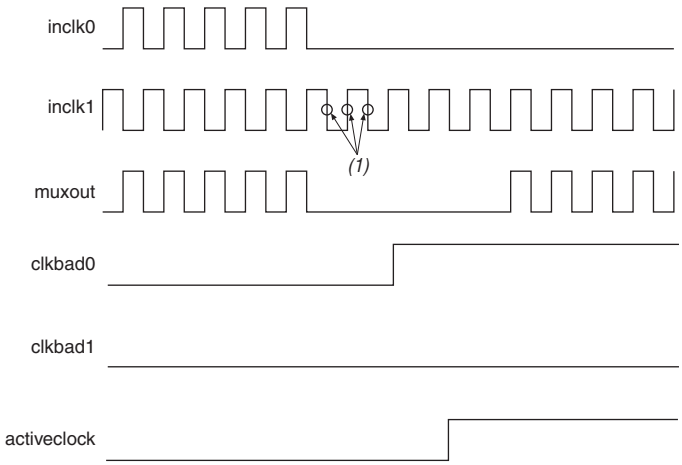
In automatic switchover mode, the `clkbad[0]` and `clkbad[1]` signals indicate the status of the two clock inputs. When they are asserted, the clock sense block has detected that the corresponding clock input has stopped toggling. These two signals are not valid if the frequency difference between `inclk0` and `inclk1` is greater than 20%.

The `activeclock` signal indicates which of the two clock inputs (`inclk0` or `inclk1`) is being selected as the reference clock to the PLL. When the frequency difference between the two clock inputs is more than 20%, the `activeclock` signal is the only valid status signal.

Glitches in the input clock may cause the frequency difference between the input clocks to be more than 20%.

Figure 4–27 shows an example waveform of the switchover feature when using automatic switchover mode. In this example, the `inclk0` signal is stuck low. After the `inclk0` signal is stuck at low for approximately two clock cycles, the clock sense circuitry drives the `clkbad[0]` signal high. Also, because the reference clock signal is not toggling, the switchover state machine controls the multiplexer through the `clkswitch` signal to switch to the backup clock, `inclk1`.

Figure 4–27. Automatic Switchover After Loss of Clock Detection



Note to Figure 4–27:

(1) Switchover is enabled on the falling edge of `inclk0` or `inclk1`, depending on which clock is available. In this figure, switchover is enabled on the falling edge of `inclk1`.

Manual Override

In automatic switchover with manual override mode, you can use the `clkswitch` input for user- or system-controlled switch conditions. You can use this mode for same-frequency switchover, or to switch between inputs of different frequencies.

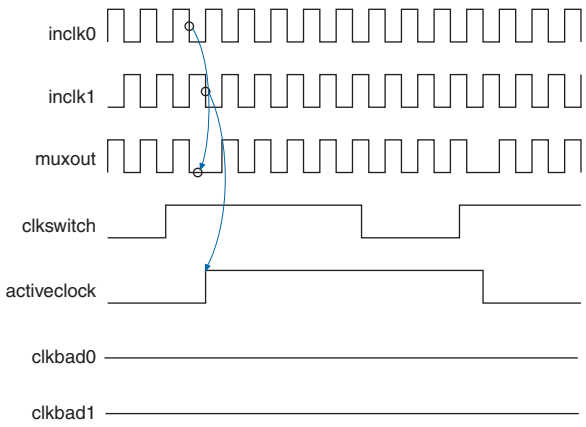
For example, if `inclk0` is 66 MHz and `inclk1` is 200 MHz, you must control switchover using `clkswitch` because the automatic clock-sense circuitry cannot monitor clock input (`inclk0` and `inclk1`) frequencies with a frequency difference of more than 100% (2×).

This feature is useful when the clock sources originate from multiple cards on the backplane, requiring a system-controlled switchover between the frequencies of operation.

You must choose the backup clock frequency and set the `M`, `N`, `C`, and `K` counters accordingly so that the VCO operates within the recommended operating frequency range. The ALTERA_PLL MegaWizard Plug-in Manager notifies you if a given combination of `inclk0` and `inclk1` frequencies cannot meet this requirement.

Figure 4–28 shows a clock switchover waveform controlled by clkswitch. In this case, both clock sources are functional and inclk0 is selected as the reference 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 clock glitching. On the falling edge of inclk1, the reference clock multiplexer switches from inclk0 to inclk1 as the PLL reference and the activeclock signal changes to indicate which clock is currently feeding the PLL.

Figure 4–28. Clock Switchover Using the clkswitch (Manual) Control *(1)*



Note to Figure 4–28:

(1) To initiate a manual clock switchover event, both inclk0 and inclk1 must be running when the clkswitch signal goes high.

In automatic override with manual switchover mode, the activeclock signal mirrors the clkswitch signal. As both clocks are still functional during the manual switch, neither clkbad signal goes high. Because the switchover circuit is positive-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.

Manual Clock Switchover

In manual clock switchover mode, the clkswitch signal controls whether inclk0 or inclk1 is selected as the input clock to the PLL. By default, inclk0 is selected.

A low-to-high transition on clkswitch and clkswitch being held high for at least three inclk cycles initiate a clock switchover event.

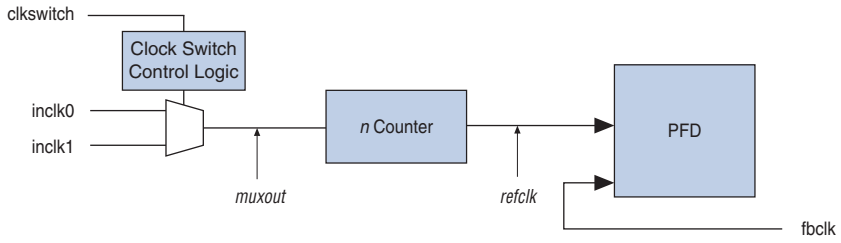
You must bring clkswitch back low again to perform another switchover event in the future. If you do not require another switchover event in the future, you can leave clkswitch in a logic high state after the initial switch.

Pulsing clkswitch high for at least three inclk cycles performs another switchover event.

If inclk0 and inclk1 are different frequencies and are always running, the clkswitch minimum high time must be greater than or equal to three of the slower frequency inclk0 or inclk1 cycles.

Figure 4–29 shows a block diagram of the manual switchover circuit.

Figure 4–29. Manual Clock Switchover Circuitry in Arria V PLLs



You can delay the clock switchover action by specifying the switchover delay in the ALTERA_PLL megafunction. When you specify the switchover delay, the clkswitch signal must be held high for at least three inclk cycles plus the number of the delay cycles that has been specified to initiate a clock switchover.

For more information about PLL software support in the Quartus II software, refer to the *Altera Phase-Locked Loop (ALTERA_PLL) Megafunction User Guide*.

Guidelines

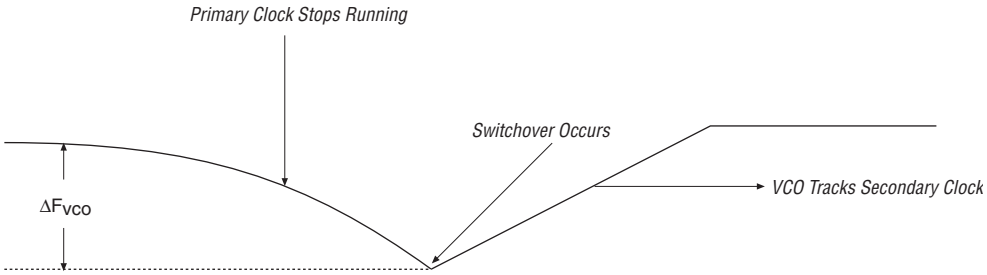
When implementing clock switchover in Arria V PLLs, use the following guidelines:

- Automatic clock switchover requires that the inclk0 and inclk1 frequencies be within 20% of each other. Failing to meet this requirement causes the clkbad[0] and clkbad[1] signals to not function properly.
- When using manual clock switchover, the difference between inclk0 and inclk1 can be more than 100% (2×). However, differences in frequency, phase, or both, of the two clock sources will likely cause the PLL to lose lock. Resetting the PLL ensures that the correct phase relationships are maintained between the input and output clocks.

  Both inclk0 and inclk1 must be running when the clkswitch signal goes high to initiate the manual clock switchover event. Failing to meet this requirement causes the clock switchover to not function properly.

- Applications that require a clock switchover feature and a small frequency drift must use a low-bandwidth PLL. The low-bandwidth PLL reacts more slowly 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 more slowly than a high-bandwidth PLL. However, be aware that the low-bandwidth PLL also increases lock time.
- After a switchover occurs, there may be a finite resynchronization period for the PLL to lock onto a new clock. The exact amount of time it takes for the PLL to relock depends on the PLL configuration.
- The phase relationship between the input clock to the PLL and the output clock from the PLL is important in your design. Assert `areset` for at least 10 ns after performing a clock switchover. Wait for the locked signal to go high and be stable before re-enabling the output clocks from the PLL.
- Figure 4–30 shows how the VCO frequency gradually decreases when the current clock is lost and then increases as the VCO locks on to the backup clock.

Figure 4–30. VCO Switchover Operating Frequency



- Disable the system during clock switchover if it is not tolerant of frequency variations during the PLL resynchronization period. You can use the `clkbad[0]` and `clkbad[1]` status signals to turn off the PFD (`PFDENA = 0`) so the VCO maintains its most recent frequency. You can also use the state machine to switch over to the secondary clock. When the PFD is re-enabled, output clock-enable signals (`clkena`) can disable clock outputs during the switchover and resynchronization period. When the lock indication is stable, the system can re-enable the output clocks.

PLL Reconfiguration and Dynamic Phase Shift

For more information about PLL reconfiguration and dynamic phase shifting, refer to *AN661: Implementing Fractional PLL Reconfiguration with ALTERA_PLL and ALTERA_PLL_RECONFIG Megafunctions*.

Document Revision History

Table 4–6 lists the revision history for this chapter.

Table 4–6. Document Revision History

Date	Version	Changes
June 2012	2.0	■ Restructured chapter. ■ Updated Figure 4–4, Figure 4–6, Figure 4–7, Figure 4–11, Figure 4–12, Figure 4–13, Figure 4–14, Figure 4–15, Figure 4–16, Figure 4–18, and Figure 4–19. ■ Updated Table 4–1, Table 4–2, Table 4–3, Table 4–4, and Table 4–5. ■ Added "Clock Regions", "Clock Network Sources", "Clock Output Connections", "Clock Enable Signals", "PLL Control Signals", "Clock Multiplication and Division", "Programmable Duty Cycle", "Clock Switchover", and "PLL Reconfiguration and Dynamic Phase Shift".
November 2011	1.1	Restructured chapter.
May 2011	1.0	Initial release.

Section II. I/O Interfaces for Arria V Devices

This section provides information about Arria® V device I/O features, external memory interfaces, and high-speed differential I/O interfaces and dynamic phase aligner (DPA). This section includes the following chapters:

- Chapter 5, I/O Features in Arria V Devices
- Chapter 6, High-Speed Differential I/O Interfaces and DPA in Arria V Devices
- Chapter 7, External Memory Interfaces in Arria V Devices

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 this volume.

5. I/O Features in Arria V Devices

AV52005-2.0

This chapter provides details about the features of the Arria® V I/O elements (IOEs) and how the IOEs work in compliance with current and emerging I/O standards and requirements.

Arria V I/Os support the following features:

- Single-ended, non-voltage-referenced, and voltage-referenced I/O standards
- Low-voltage differential signaling (**LVDS**), **RSDS**, **mini-LVDS**, **HSTL**, **HSUL**, and **SSTL** I/O standards
- Serializer/deserializer (SERDES)
- Programmable output current strength
- Programmable slew-rate
- Programmable bus-hold
- Programmable pull-up resistor
- Programmable pre-emphasis
- Programmable I/O delay
- Programmable voltage output differential (V_{OD})
- Open-drain output
- On-chip series termination (R_S OCT)
- On-chip parallel termination (R_T OCT)
- On-chip differential termination (R_D OCT)

The information in this chapter is applicable to all Arria V variants, unless noted otherwise.

This chapter contains the following sections:

I/O Standards Support

Table 5–1 lists the supported I/O standards and typical power supply values.

Table 5–1. Arria V I/O Standards and Voltage Levels (Part 1 of 2) (Part 1 of 2)

I/O Standard	Standard Support	V_{CCIO} (V)		V_{CCPD} (V) (Pre-Driver Voltage)	V_{REF} (V) (Input Ref Voltage) (1)	V_{TT} (V) (Board Termination Voltage)
		Input Operation	Output Operation			
3.3-V LVTTL/3.3-V LVCMOS (2)	JESD8-B	3.3/3.0/2.5	3.3	3.3	—	—
3.0-V LVTTL/3.0-V LVCMOS (2)	JESD8-B	3.0/2.5	3.0	3.0	—	—
2.5-V LVCMOS (2)	JESD8-5	3.0/2.5	2.5	2.5	—	—
1.8-V LVCMOS (2), (3)	JESD8-7	1.8/1.5	1.8	2.5	—	—
1.5-V LVCMOS (2)	JESD8-11	1.8/1.5	1.5	2.5	—	—
1.2-V LVCMOS	JESD8-12	1.2	1.2	2.5	—	—
3.0-V PCI	PCI Rev. 2.2	3.0	3.0	3.0	—	—
3.0-V PCI-X (4)	PCI-X Rev. 1.0	3.0	3.0	3.0	—	—
SSTL-2 Class I	JESD8-9B	(5)	2.5	2.5	1.25	1.25
SSTL-2 Class II	JESD8-9B	(5)	2.5	2.5	1.25	1.25
SSTL-18 Class I (3)	JESD8-15	(5)	1.8	2.5	0.90	0.90
SSTL-18 Class II (3)	JESD8-15	(5)	1.8	2.5	0.90	0.90
SSTL-15 Class I (3)	—	(5)	1.5	2.5	0.75	0.75
SSTL-15 Class II (3)	—	(5)	1.5	2.5	0.75	0.75
SSTL-15	JESD79-3D	(5)	1.5	2.5	0.75	(6)
SSTL-135 (3)	—	(5)	1.35	2.5	0.675	(6)
SSTL-125 (3)	—	(5)	1.25	2.5	0.625	(6)
1.8-V HSTL Class I	JESD8-6	(5)	1.8	2.5	0.90	0.90
1.8-V HSTL Class II	JESD8-6	(5)	1.8	2.5	0.90	0.90
1.5-V HSTL Class I (2)	JESD8-6	(5)	1.5	2.5	0.75	0.75
1.5-V HSTL Class II (2)	JESD8-6	(5)	1.5	2.5	0.75	0.75
1.2-V HSTL Class I	JESD8-16A	(5)	1.2	2.5	0.6	0.6
1.2-V HSTL Class II	JESD8-16A	(5)	1.2	2.5	0.6	0.6
HSUL-12 (3)	—	(5)	1.2	2.5	0.6	(6)
Differential SSTL-2 Class I	JESD8-9B	(5)	2.5	2.5	—	1.25
Differential SSTL-2 Class II	JESD8-9B	(5)	2.5	2.5	—	1.25
Differential SSTL-18 Class I	JESD8-15	(5)	1.8	2.5	—	0.90
Differential SSTL-18 Class II	JESD8-15	(5)	1.8	2.5	—	0.90
Differential SSTL-15 Class I	—	(5)	1.5	2.5	—	0.75
Differential SSTL-15 Class II	—	(5)	1.5	2.5	—	0.75
Differential 1.8-V HSTL Class I	JESD8-6	(5)	1.8	2.5	—	0.90
Differential 1.8-V HSTL Class II	JESD8-6	(5)	1.8	2.5	—	0.90
Differential 1.5-V HSTL Class I	JESD8-6	(5)	1.5	2.5	—	0.75
Differential 1.5-V HSTL Class II	JESD8-6	(5)	1.5	2.5	—	0.75

Table 5–1. Arria V I/O Standards and Voltage Levels (Part 2 of 2) (Part 2 of 2)

I/O Standard	Standard Support	V_{CCIO} (V)		V_{CCPD} (V) (Pre-Driver Voltage)	V_{REF} (V) (Input Ref Voltage) (1)	V_{TT} (V) (Board Termination Voltage)
		Input Operation	Output Operation			
Differential 1.2-V HSTL Class I	JESD8-16A	(5)	1.2	2.5	—	0.60
Differential 1.2-V HSTL Class II	JESD8-16A	(5)	1.2	2.5	—	0.60
Differential SSTL-15	JESD79-3D	(5)	1.5	2.5	—	(6)
Differential SSTL-135	—	(5)	1.35	2.5	—	(6)
Differential SSTL-125	—	(5)	1.25	2.5	—	(6)
Differential HSUL-12	—	(5)	1.2	2.5	—	(6)
LVDS	ANSI/TIA/EIA-644	(5)	2.5	2.5	—	—
RSDS	—	(5)	2.5	2.5	—	—
Mini-LVDS	—	(5)	2.5	2.5	—	—
LVPECL	—	(7)	—	2.5	—	—

Notes to Table 5–1:

(1) You cannot assign **SSTL**, **HSTL**, and **HSUL** outputs on VREF pins, even if there are no **SSTL**, **HSTL**, and **HSUL** inputs in the bank.
(2) Supported in the hard processor system (HPS) column I/Os.
(3) Supported in the HPS row I/Os.
(4) **PCI-X** does not meet the PCI-X I-V curve requirement at the linear region.
(5) Single-ended **HSTL**/**SSTL**/**HSUL**, **differential SSTL**/**HSTL**/**HSUL**, and **LVDS** input buffers are powered by V_{CCPD}.
(6) This I/O standard typically does not require board termination.
(7) The support for the **LVPECL** I/O standard is only for input clock operation. V_{CCPD} powers the differential clock input buffers.

Design Considerations

There are several considerations that require your attention to ensure the success of your designs.

I/O Bank Restrictions

The following sections provide guidelines for mixing non-voltage-referenced and voltage-referenced I/O standards in the devices.

Non-Voltage-Referenced Standards

Each Arria V I/O bank has its own VCCIO pins and supports only one V_{CCIO} (1.2, 1.25, 1.35, 1.5, 1.8, 2.5, 3.0, or 3.3 V). An I/O bank can simultaneously support any number of input signals with different I/O standard assignments if the I/O standards support the V_{CCIO} level of the I/O bank.

For output signals, a single I/O bank supports non-voltage-referenced output signals that drive at the same voltage as V_{CCIO}. Because an I/O bank can only have one V_{CCIO} value, it can only drive out the 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 **3.0-V LVCMOS** inputs only.

Voltage-Referenced Standards

To accommodate voltage-referenced I/O standards, each Arria V I/O bank contains a dedicated VREF pin. Each bank can have only a single V_{CCIO} voltage level and a single voltage reference (V_{REF}) level.

An I/O bank featuring single-ended or differential standards can support different voltage-referenced standards if the V_{CCIO} and V_{REF} are the same levels.

Voltage-referenced bidirectional and output signals must be the same as the V_{CCIO} voltage of the I/O bank.

For example, you can place only **SSTL-2** output pins in an I/O bank with a 2.5-V V_{CCIO}.

Mixing Voltage-Referenced and Non-Voltage-Referenced Standards

An I/O bank can support voltage-referenced and non-voltage-referenced pins by applying each of the rule sets individually.

First example: an I/O bank can support **SSTL-18** inputs and outputs, and 1.8-V inputs and outputs with a 1.8-V V_{CCIO} and a 0.9-V V_{REF}.

Second example: an I/O bank can support 1.5-V standards, 1.8-V inputs (but not outputs), and **1.5-V HSTL** I/O standards with a 1.5-V V_{CCIO} and 0.75-V V_{REF}.

V_{CCPD} Restriction

One VCCPD pin is shared in a group of I/O banks. If one I/O bank in a group uses 3.0-V V_{CCPD}, other I/O banks in the same group must also use 3.0-V V_{CCPD}.

The I/O banks with the same number form a group. For example, I/O banks 8A, 8B, 8C, and 8D form a group and share the same VCCPD pin. This sharing is applicable to all I/O banks except I/O banks 4A and 7A—each of these I/O banks has their own individual VCCPD pin.

V_{CCIO} Restriction

When planning the I/O bank usage, you must ensure the V_{CCIO} voltage is compatible with the V_{CCPD} voltage of the same bank. Some banks may share the same VCCPD power pin. This limits the possible V_{CCIO} voltages that can be used on banks that share VCCPD power pins.

First example: if VCCPD4BCD is connected to 2.5 V, then the VCCIO pins for banks 4B, 4C, and 4D can be connected to any of the following voltages: 1.2 V, 1.25 V, 1.35 V, 1.5 V, 1.8 V, or 2.5 V.

Second example: if VCCPD4BCD is connected to 3.0 V, then the VCCIO pins for banks 4B, 4C, and 4D must also be connected to 3.0 V.

V_{REF} Pin Restriction

You cannot assign shared VREF pins as **LVDS** or external memory interface pins.

SSTL, **HSTL**, and **HSUL** I/O standards do not support shared VREF pins.

For example, if a particular B1p or B1n pin is a shared VREF pin, the corresponding B1p/B1n pin pair do not have LVDS transmitter support.

Shared VREF pins will have reduced performance when used as normal I/Os.

You must perform signal integrity analysis using your board design when using a shared VREF pin to determine the F_{MAX} for your system. Refer to the input pin capacitance in the *Arria V Device Datasheet* for the pin capacitance of the VREF pin.

3.3-V I/O Interface

To ensure device reliability and proper operation when you use the Arria V device for 3.3-V I/O interfacing, do not violate the absolute maximum ratings of the device.

For a transmitter, use slow slew-rate and series termination to limit the overshoot and undershoot at the I/O pins.

For a receiver, use the on-chip clamp diode to limit the overshoot and undershoot voltage at the I/O pins.

For more information about absolute maximum rating and maximum allowed overshoot during transitions, refer to the *Arria V Device Datasheet*.

Altera recommends that you perform IBIS or SPICE simulations to make sure the overshoot and undershoot voltages are within the specifications.

LVDS Channels

For LVDS applications, you must use the phase-locked loops (PLLs) in integer PLL mode.

For more information about the LVDS channels in Arria V devices, refer to the *High-Speed Differential I/O Interfaces and DPA in Arria V Devices* chapter.

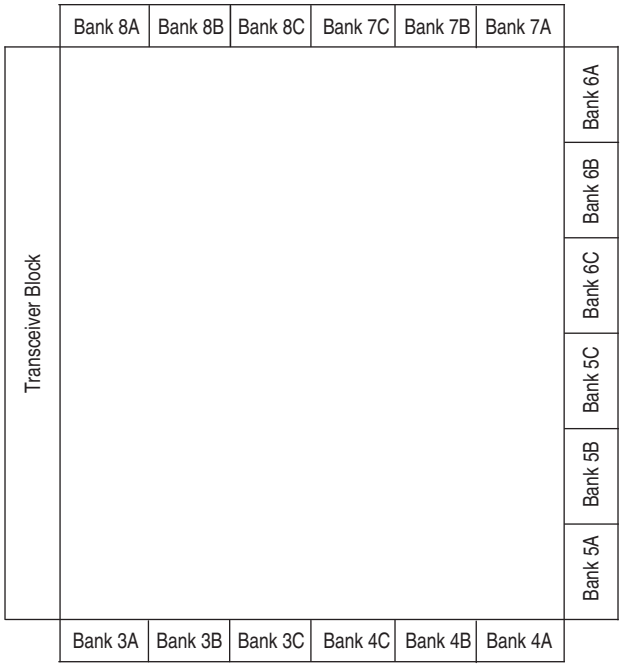
I/O Banks

The number of Arria V I/O banks in a particular device depends on the device density.

Each I/O bank can simultaneously support multiple I/O standards.

Figure 5–1 shows the I/O banks in Arria V GX A1 and A3 devices, and Arria V GT C3 devices.

Figure 5–1. I/O Banks for Arria V GX A1 and A3 Devices, and Arria V GT C3 Devices (1)—*Preliminary*

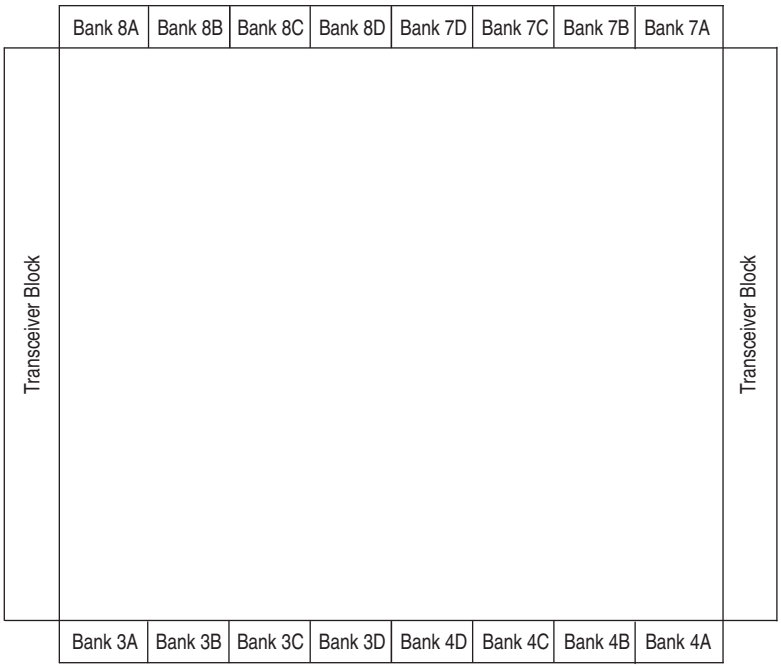


Note to Figure 5–1:

(1) This is a top view of the silicon die that corresponds to a reverse view of the device package.

Figure 5–2 shows the I/O banks in Arria V GX A5, A7, B1, B3, B5, and B7 devices, and Arria V GT C7, D3, and D7 devices.

Figure 5–2. I/O Banks for Arria V GX A5, A7, B1, B3, B5, and B7 Devices, and Arria V GT C7, D3, and D7 Devices (1)—*Preliminary*

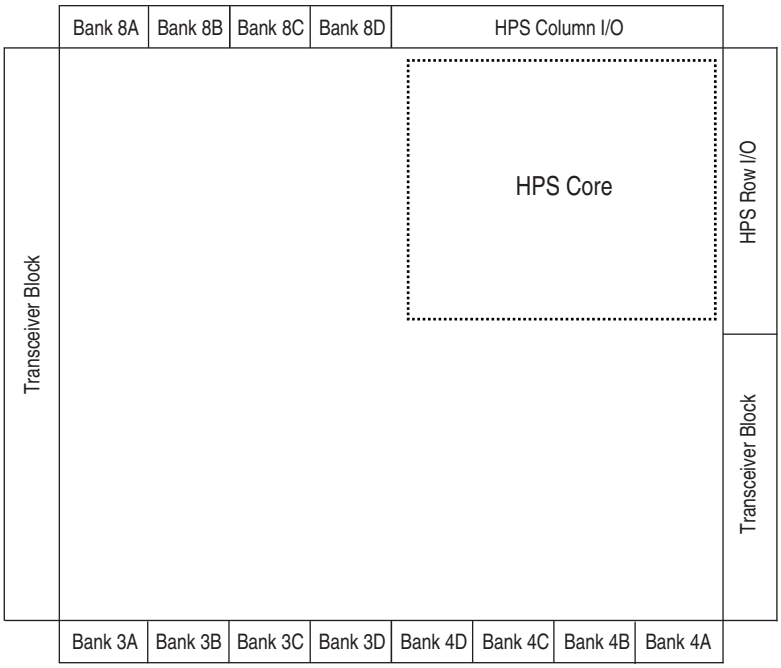


Note to Figure 5–2:

(1) This is a top view of the silicon die that corresponds to a reverse view of the device package.

Figure 5–3 shows the I/O banks in Arria V SX B3 and B5 devices, and Arria V ST D3 and D5 devices.

Figure 5–3. I/O Banks for Arria V SX B3 and B5 Devices, and Arria V ST D3 and D5 Devices (1)—*Preliminary*



Note to Figure 5–3:

(1) This is a top view of the silicon die that corresponds to a reverse view of the device package.

Modular I/O Banks

The I/O pins in Arria V devices are arranged in groups called modular I/O banks.

Table 5–2 list the modular I/O banks for Arria V GX devices.

Table 5–2. Modular I/O Banks for Arria V GX Devices (Part 1 of 2)—*Preliminary*

Member Code	Package	Bank																					Total
		3A	3B	3C	3D	4A	4B	4C	4D	5A	5B	5C	6A	6B	6C	7A	7B	7C	7D	8A	8B	8C	8D	
A1	F672	24	—	—	32	16	—	32	32	32	—	—	32	—	—	16	—	32	32	24	—	—	32	336
	F896	32	—	—	32	16	16	32	32	48	—	—	48	—	—	16	16	32	32	32	—	—	32	416
A3	F672	24	—	—	32	16	—	32	32	32	—	—	32	—	—	16	—	32	32	24	—	—	32	336
	F896	32	—	—	32	16	16	32	32	48	—	—	48	—	—	16	16	32	32	32	—	—	32	416
A5	F672	24	—	—	20	28	32	32	32	—	—	—	—	—	—	28	32	32	32	24	—	—	20	336
	F896	32	—	—	32	32	32	32	32	—	—	—	—	—	—	32	32	32	32	32	—	—	32	384
	F1152	48	32	32	32	32	32	32	32	—	—	—	—	—	—	32	32	32	32	48	32	32	32	544
A7	F672	24	—	—	32	16	—	32	32	32	—	—	32	—	—	16	—	32	32	24	—	—	32	336
	F896	32	—	—	32	16	16	32	32	48	—	—	48	—	—	16	16	32	32	32	—	—	32	416
	F1152	48	32	32	32	32	32	32	32	—	—	—	—	—	—	32	32	32	32	48	32	32	32	544

Table 5–2. Modular I/O Banks for Arria V GX Devices (Part 2 of 2)—*Preliminary*

Member Code	Package	Bank																					Total
		3A	3B	3C	3D	4A	4B	4C	4D	5A	5B	5C	6A	6B	6C	7A	7B	7C	7D	8A	8B	8C	8D	
B1	F896	32	—	—	32	32	32	32	32	—	—	—	—	—	—	32	32	32	32	32	—	—	32	384
	F1152	48	32	32	32	32	32	32	32	—	—	—	—	—	—	32	32	32	32	48	32	32	32	544
	F1517	48	32	48	48	48	48	32	48	—	—	—	—	—	—	48	48	32	48	48	32	48	48	704
B3	F896	32	—	—	32	32	32	32	32	—	—	—	—	—	—	32	32	32	32	32	—	—	32	384
	F1152	48	32	32	32	32	32	32	32	—	—	—	—	—	—	32	32	32	32	48	32	32	32	544
	F1517	48	32	48	48	48	48	32	48	—	—	—	—	—	—	48	48	32	48	48	32	48	48	704
B5	F1152	48	32	32	32	32	32	32	32	—	—	—	—	—	—	32	32	32	32	48	32	32	32	544
	F1517	48	32	48	48	48	48	32	48	—	—	—	—	—	—	48	48	32	48	48	32	48	48	704
B7	F1152	48	32	32	32	32	32	32	32	—	—	—	—	—	—	32	32	32	32	48	32	32	32	544
	F1517	48	32	48	48	48	48	32	48	—	—	—	—	—	—	48	48	32	48	48	32	48	48	704

Table 5–3 list the modular I/O banks for Arria V GT devices.

Table 5–3. Modular I/O Banks for Arria V GT Devices —*Preliminary*

Member Code	Package	Bank																					Total
		3A	3B	3C	3D	4A	4B	4C	4D	5A	5B	5C	6A	6B	6C	7A	7B	7C	7D	8A	8B	8C	8D	
C3	F672	24	—	—	32	16	—	32	32	32	—	—	32	—	—	16	—	32	32	24	—	—	32	336
	F896	32	—	—	32	16	16	32	32	48	—	—	48	—	—	16	16	32	32	32	—	—	32	416
C7	F896	32	—	—	32	16	16	32	32	48	—	—	48	—	—	16	16	32	32	32	—	—	32	416
	F1152	48	32	32	32	32	32	32	32	—	—	—	—	—	—	32	32	32	32	48	32	32	32	544
D3	F896	32	—	—	32	32	32	32	32	—	—	—	—	—	—	32	32	32	32	32	—	—	32	384
	F1152	48	32	32	32	32	32	32	32	—	—	—	—	—	—	32	32	32	32	48	32	32	32	544
	F1517	48	32	48	48	48	48	32	48	—	—	—	—	—	—	48	48	32	48	48	32	48	48	704
D7	F1152	48	32	32	32	32	32	32	32	—	—	—	—	—	—	32	32	32	32	48	32	32	32	544
	F1517	48	32	48	48	48	48	32	48	—	—	—	—	—	—	48	48	32	48	48	32	48	48	704

Table 5–4 list the modular I/O banks for Arria V SX devices.

Table 5–4. Modular I/O Banks for Arria V SX Devices —*Preliminary*

Member Code	Package	FPGA I/O Bank								HPS Row I/O Bank		HPS Column I/O Bank						FPGA I/O Bank				Total
		3A	3B	3C	3D	4A	4B	4C	4D	6A	6B	7A	7B	7C	7D	7E	7G	8A	8B	8C	8D	
B3	F896	44	28	—	13	42	—	—	—	56	44	32	22	12	20	8	—	44	28	38	13	444
	F1152	44	28	38	13	42	38	26	32	56	44	32	22	12	20	8	—	44	28	38	14	579
	F1517	48	32	48	48	48	48	32	48	56	44	32	22	12	20	8	12	48	32	48	48	734
B5	F896	44	28	—	13	42	—	—	—	56	44	32	22	12	20	8	—	44	28	38	13	444
	F1152	44	28	38	13	42	38	26	32	56	44	32	22	12	20	8	—	44	28	38	14	579
	F1517	48	32	48	48	48	48	32	48	56	44	32	22	12	20	8	12	48	32	48	48	734

Table 5–5 list the modular I/O banks for Arria V ST devices.

Table 5–5. Modular I/O Banks for Arria V ST Devices —*Preliminary*

Member Code	Package	FPGA I/O Bank								HPS Row I/O Bank		HPS Column I/O Bank						FPGA I/O Bank				Total
		3A	3B	3C	3D	4A	4B	4C	4D	6A	6B	7A	7B	7C	7D	7E	7G	8A	8B	8C	8D	
D3	F896	44	28	—	13	42	—	—	—	56	44	32	22	12	20	8	—	44	28	38	13	444
	F1152	44	28	38	13	42	38	26	32	56	44	32	22	12	20	8	—	44	28	38	14	579
	F1517	48	32	48	48	48	48	32	48	56	44	32	22	12	20	8	12	48	32	48	48	734
D5	F896	44	28	—	13	42	—	—	—	56	44	32	22	12	20	8	—	44	28	38	13	444
	F1152	44	28	38	13	42	38	26	32	56	44	32	22	12	20	8	—	44	28	38	14	579
	F1517	48	32	48	48	48	48	32	48	56	44	32	22	12	20	8	12	48	32	48	48	734

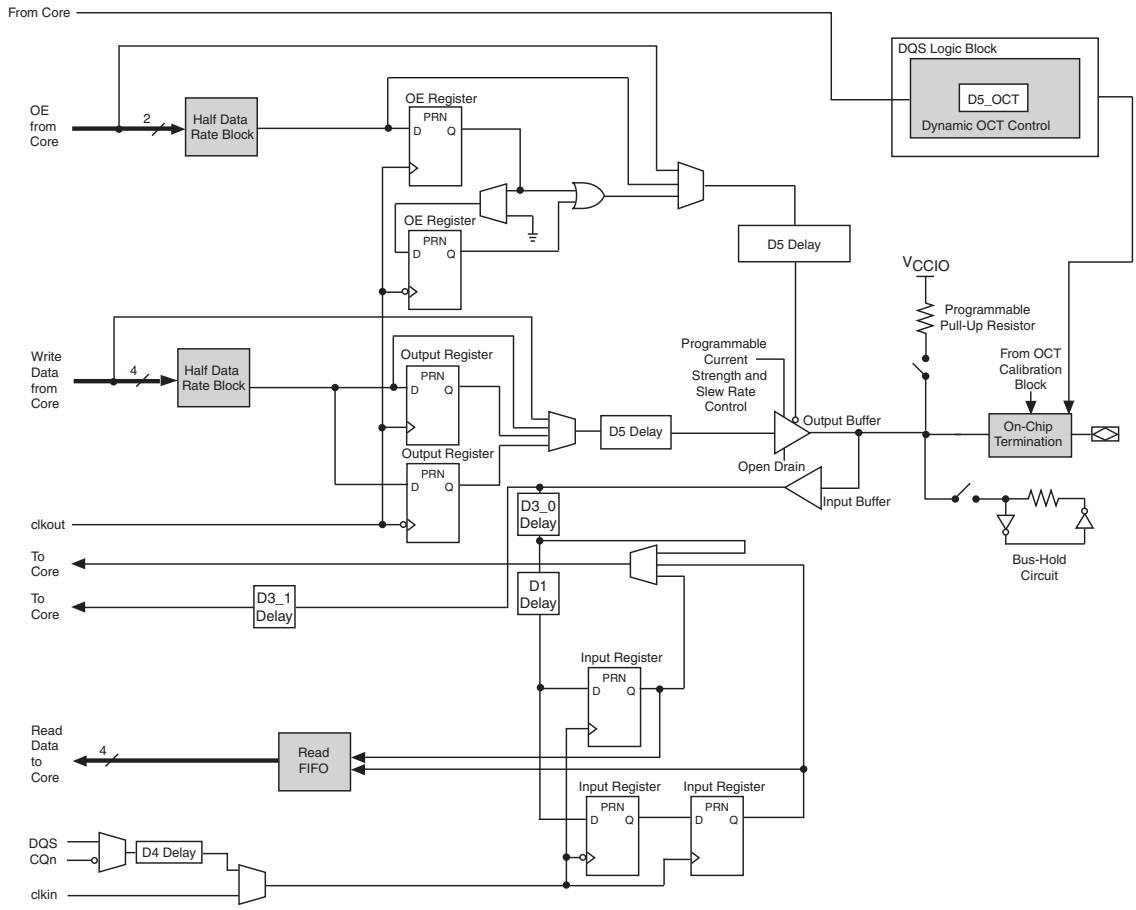
IOE Structure

The IOEs in Arria V devices contain a bidirectional I/O buffer and I/O registers to support a complete embedded bidirectional single data rate (SDR) or double data rate (DDR) transfer.

The IOEs are located in I/O blocks around the periphery of the Arria V device.

Figure 5–4 shows the Arria V IOE structure.

Figure 5–4. IOE Structure for Arria V Devices *(1), (2)*



Notes to Figure 5–4:

(1) The `D3_0` and `D3_1` delays have the same available settings in the Quartus® II software.

(2) One dynamic on-chip termination (OCT) control is available for each DQ/DQS group.

Current Strength

You can use the programmable current strength to mitigate the effects of high signal attenuation that is caused by a long transmission line or a legacy backplane.

The output buffer for each Arria V device I/O pin has a programmable current strength control for the following I/O standards.

Table 5–6 lists the programmable current strength settings for Arria V devices.

Table 5–6. Programmable Current Strength Settings

I/O Standard	I_{OH} / I_{OL} Current Strength Setting (mA) (1)
3.3-V LVTTL (2)	**8**, 4
3.3-V LVCMOS (2)	**2**
3.0-V LVTTL/3.0-V LVCMOS (2)	16, **12**, 8, 4
2.5-V LVCMOS (2)	16, **12**, 8, 4
1.8-V LVCMOS (2)	**12**, 10, 8, 6, 4, 2
1.5-V LVCMOS (2)	**12**, 10, 8, 6, 4, 2
1.2-V LVCMOS	**8**, 6, 4, 2
SSTL-2 Class I	12, 10, **8**
SSTL-2 Class II	**16**
SSTL-18 Class I (2)	12, 10, **8**, 6, 4
SSTL-18 Class II (2)	**16**
SSTL-15 Class I (2)	12, 10, **8**, 6, 4
SSTL-15 Class II (2)	**16**
1.8-V HSTL Class I	12, 10, **8**, 6, 4
1.8-V HSTL Class II	**16**
1.5-V HSTL Class I (2)	12, 10, **8**, 6, 4
1.5-V HSTL Class II (2)	**16**
1.2-V HSTL Class I	12, 10, **8**, 6, 4
1.2-V HSTL Class II	**16**

Notes to Table 5–6:

(1) The default current strength setting in the Quartus II software is the current strength shown in bold.

(2) Supported in HPS.

Altera recommends that you perform IBIS or SPICE simulations to determine the best current strength setting for your specific application.

MultiVolt I/O Interface

The MultiVolt I/O interface feature allows Arria V devices in all packages to interface with systems of different supply voltages.

Table 5–7 lists Arria V MultiVolt I/O support.

Table 5–7. MultiVolt I/O Support in Arria V Devices *(1), (2)*

V_{CCIO} (V)	Input Signal (V)								Output Signal (V)							
	1.2	1.25	1.35	1.5	1.8	2.5	3.0	3.3	1.2	1.25	1.35	1.5	1.8	2.5	3.0	3.3
1.2	Y	—	—	—	—	—	—	—	Y	—	—	—	—	—	—	—
1.25	—	Y	—	—	—	—	—	—	—	Y	—	—	—	—	—	—
1.35	—	—	Y	—	—	—	—	—	—	—	Y	—	—	—	—	—
1.5	—	—	—	Y	Y	—	—	—	—	—	—	Y	—	—	—	—
1.8	—	—	—	Y	Y	—	—	—	—	—	—	—	Y	—	—	—
2.5	—	—	—	—	—	Y	Y *(3)*	Y *(3)*	—	—	—	—	—	Y	—	—
3.0	—	—	—	—	—	Y	Y *(3)*	Y *(3)*	—	—	—	—	—	—	Y	—
3.3	—	—	—	—	—	Y	Y *(3)*	Y *(3)*	—	—	—	—	—	—	—	Y

Notes to Table 5–7:

(1) The pin current may be slightly higher than the default value. Verify that the V_{OL} maximum and V_{OH} minimum voltages of the driving device do not violate the applicable V_{IL} maximum and V_{IH} minimum voltage specifications of the Arria V device.

(2) For V_{CCIO} = 1.2, 1.25, 1.35, 1.5, 1.8, and 2.5 V, V_{CCPD} = 2.5 V. For V_{CCIO} = 3.0 V, V_{CCPD} = 3.0 V. For V_{CCIO} = 3.3 V, V_{CCPD} = 3.3 V.

(3) Altera recommends using the on-chip clamp diode on the I/O pins when the input signal is 3.0 V or 3.3 V.

Programmable IOE Features

The Arria V I/O supports programmable features, as listed in Table 5–8.

Table 5–8. Supported I/O Features and Settings *(1)*

Feature	Setting	Condition
Slew Rate Control	0 = Slow, 1 = Fast (default)	Disabled when you use the R_S OCT feature.
I/O Delay	*(1)*	—
Open-Drain Output	On, Off (default)	—
Bus-Hold	On, Off (default)	Disabled when you use the weak pull-up resistor feature.
Weak Pull-up Resistor	On, Off (default)	Disabled when you use the bus-hold feature.
Pre-Emphasis	0 = Disabled, 1 = Enabled (default)	—
Differential Output Voltage	0 = low, 1 = medium (default), 2 = high	—
On-Chip Clamp Diode *(2)*	On, Off (default)	—

Notes to Table 5–8:

(1) For information about the programmable IOE features, refer to the *Arria V Device Datasheet*.

(2) The PCI on-chip clamp diode is available on all general purpose I/O (GPIO) pins in all Arria V device variants.

Slew-Rate Control

The programmable output slew-rate control in the output buffer of each regular- and dual-function I/O pin allows you to configure the following:

- Fast slew-rate—provides high-speed transitions for high-performance systems.
- Slow slew-rate—reduces system noise and crosstalk but adds a nominal delay to the rising and falling edges.

You can specify the slew-rate on a pin-by-pin basis because each I/O pin contains a slew-rate control.

Altera recommends that you perform IBIS or SPICE simulations to determine the best slew rate setting for your specific application.

I/O Delay

The following sections describe the programmable IOE delay and the programmable output buffer delay.

Programmable IOE Delay

You can activate the programmable delays to ensure zero hold times, minimize setup times, or increase clock-to-output times.

This feature helps read and write timing margins because it minimizes the uncertainties between signals in the bus.

Each pin can have a different input delay from pin-to-input register or a delay from output register-to-output pin values to ensure that the signals within a bus have the same delay going into or out of the device.

Programmable Output Buffer Delay

The delay chains are built inside the single-ended output buffer.

There are four levels of output buffer delay settings. By default, there is no delay.

The following actions allow you to independently control the rising and falling edge delays of the output buffer:

- Adjust the output-buffer duty cycle
- Compensate channel-to-channel skew
- Reduce simultaneous switching output (SSO) noise by deliberately introducing channel-to-channel skew
- Improve high-speed memory-interface timing margins

Open-Drain Output

The optional open-drain output for each I/O pin is equivalent to an open collector output.

When configured as an open drain, the logic value of the output is either high-Z or logic low.

Use an external resistor to pull the signal to a logic high.

Bus-Hold

Each I/O pin provides an optional bus-hold feature that is active only after configuration. When the device enters user mode, the bus-hold circuit captures the value that is present on the pin by the end of the configuration.

The bus-hold circuitry uses a resistor with a nominal resistance (R_{BH}), approximately 7 kΩ, to weakly pull the signal level to the last-driven state of the pin. The bus-hold circuitry holds this pin state until the next input signal is present. Because of this, you do not require an external pull-up or pull-down resistor to hold a signal level when the bus is tri-stated.

For each I/O pin, you can individually specify that the bus-hold circuitry pulls non-driven pins away from the input threshold voltage—where noise can cause unintended high-frequency switching. To prevent over-driving signals, the bus-hold circuitry drives the voltage level of the I/O pin lower than the V_{CCIO} level.

If you enable the bus-hold feature, you cannot use the programmable pull-up option. To configure the I/O pin for differential signals, disable the bus-hold feature.

Pull-Up Resistor

The pull-up resistor weakly holds the I/O to the V_{CCIO} level.

The Arria V device supports programmable weak pull-up resistors only on user I/O pins but not on dedicated configuration pins, JTAG pins, or dedicated clock pins.

Each I/O pin provides an optional programmable pull-up resistor during user mode.

If you enable this option, you cannot use the bus-hold feature.

Pre-Emphasis

Pre-emphasis boosts the output current momentarily.

The overshoot introduced by the extra current happens only during a change of state switching to increase the output slew rate and does not ring, unlike the overshoot caused by signal reflection.

The V_{OD} setting and the output impedance of the driver set the output current limit of a high-speed transmission signal. At a high frequency, the slew rate may not be fast enough to reach the full V_{OD} level before the next edge, producing pattern-dependent jitter.

The amount of pre-emphasis required depends on the attenuation of the high-frequency component along the transmission line.

For more information about programmable pre-emphasis, refer to the *High-Speed Differential I/O Interfaces with DPA in Arria V Devices* chapter.

Differential Output Voltage

The Arria V **LVDS** transmitters support programmable V_{OD}.

The programmable V_{OD} settings allow you to adjust the output eye opening to optimize the trace length and power consumption. A higher V_{OD} swing improves voltage margins at the receiver end, and a smaller V_{OD} swing reduces power consumption.

For more information about programmable V_{OD}, refer to the *High-Speed Differential I/O Interfaces with DPA in Arria V Devices* chapter.

OCT Schemes

Dynamic R_S and R_T OCT provides I/O impedance matching and termination capabilities. OCT maintains signal quality, saves board space, and reduces external component costs.

Arria V devices support OCT in all I/O banks.

Table 5–9 lists the OCT schemes supported in Arria V devices.

Table 5–9. OCT Schemes in Arria V Devices

Direction	OCT Schemes
Output	R_S OCT with calibration (1)
	R_S OCT without calibration (1)
Input	R_T OCT with calibration (1)
	R_D OCT (**LVDS** I/O standard only)
Bidirectional	Dynamic R_S OCT and R_T OCT

Note to Table 5–9:

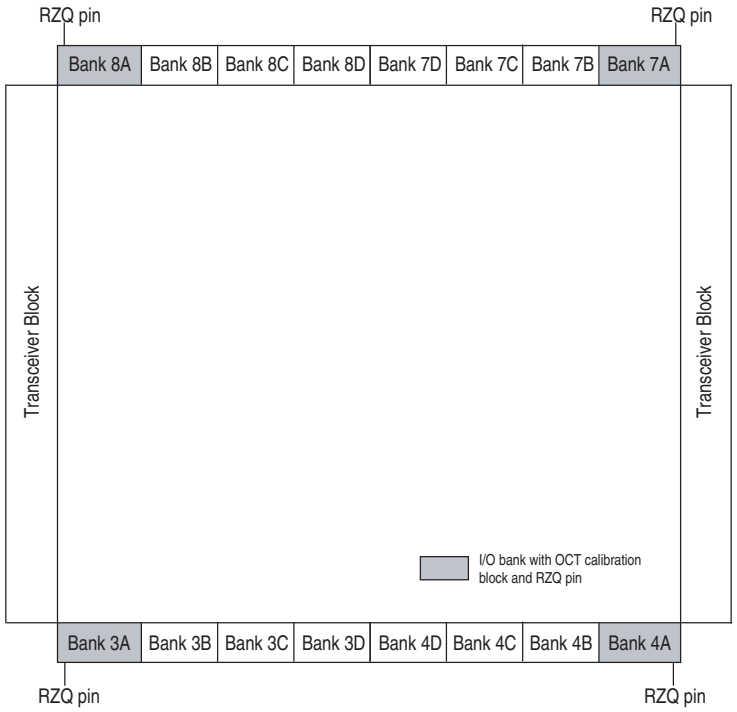
(1) For information about OCT support for the selectable I/O standards, refer to Table 5–10.

OCT Calibration Block

You can calibrate the OCT using any of the available four OCT calibration blocks for each device. Each calibration block contains one RZQ pin.

Figure 5–5 shows the location of I/O banks with OCT calibration blocks and RZQ pins.

Figure 5–5. OCT Calibration Block and RZQ Pin Location (1)—*Preliminary*



Note to Figure 5–5:

(1) This is a top view of the silicon die that corresponds to a reverse view of the device package. This figure illustrates the highest density for Arria V devices.

You can use R_S and R_T OCT in the same I/O bank for different I/O standards if the R_S and R_T OCT use the same V_{CCIO} supply voltage. You cannot configure the R_S OCT and the programmable current strength for the same I/O buffer.

Connect the RZQ pin to the GND pin through a resistor with the specified value. The RZQ pin shares the same V_{CCIO} supply voltage with the I/O bank where the pin is located.

Arria V devices support calibrated R_S and calibrated R_T on all I/O pins except for dedicated configuration pins.

Table 5–10 lists the input and output termination settings for calibrated and uncalibrated OCT on different I/O standards.

Table 5–10. Selectable I/O Standards for R_S and R_T OCT With and Without Calibration (Part 1 of 2)

I/O Standard	Output Termination			Input Termination	
	Uncalibrated OCT Setting	Calibrated OCT Setting		Calibrated OCT Setting	
	R_S (Ω)	R_S (Ω)	RZQ (Ω)	R_T (Ω)	RZQ (Ω)
3.3-V LVTTL/3.3-V LVCMOS	—	—	—	—	—
3.0-V LVVTL/3.0-V LVCMOS	25/50	25/50	100	—	—
2.5-V LVCMOS	25/50	25/50	100	—	—
1.8-V LVCMOS	25/50	25/50	100	—	—
1.5-V LVCMOS	25/50	25/50	100	—	—
1.2-V LVCMOS	25/50	25/50	100	—	—
SSTL-2 Class I	50	50	100	50	100
SSTL-2 Class II	25	25	100	50	100
SSTL-18 Class I	50	50	100	50	100
SSTL-18 Class II	25	25	100	50	100
SSTL-15 Class I	50	50	100	20, 25, 30, 40, 50, 60,120	100
SSTL-15 Class II	25	25	100	20, 25, 30, 40, 50, 60,120	100
1.8-V HSTL Class I	50	50	100	50	100
1.8-V HSTL Class II	25	25	100	50	100
1.5-V HSTL Class I	50	50	100	50	100
1.5-V HSTL Class II	25	25	100	50	100
1.2-V HSTL Class I	50	50	100	50	100
1.2-V HSTL Class II	25	25	100	50	100
SSTL-15	—	25/50	100	20, 25, 30, 40, 50, 60,120	240
		34/40	240		
SSTL-135	—	34/40	240	20, 30, 40, 60, 120	240
SSTL-125	—	34/40	240	20, 30, 40, 60, 120	240
HSUL-12	—	34/40/48/60/80	240	—	—
Differential SSTL-2 Class I	50	50	100	50	100
Differential SSTL-2 Class II	25	25	100	50	100
Differential SSTL-18 Class I	50	50	100	50	100
Differential SSTL-18 Class II	25	25	100	50	100
Differential SSTL-15 Class I	50	50	100	50	100
Differential SSTL-15 Class II	25	25	100	50	100
Differential 1.8-V HSTL Class I	50	50	100	50	100
Differential 1.8-V HSTL Class II	25	25	100	50	100

Table 5–10. Selectable I/O Standards for R_S and R_T OCT With and Without Calibration (Part 2 of 2)

I/O Standard	Output Termination			Input Termination	
	Uncalibrated OCT Setting	Calibrated OCT Setting		Calibrated OCT Setting	
	R_S (Ω)	R_S (Ω)	RZQ (Ω)	R_T (Ω)	RZQ (Ω)
Differential 1.5-V HSTL Class I	50	50	100	50	100
Differential 1.5-V HSTL Class II	25	25	100	50	100
Differential 1.2-V HSTL Class I	50	50	100	50	100
Differential 1.2-V HSTL Class II	25	25	100	50	100
Differential SSTL-15	—	25/50	100	20, 25, 30, 40, 50, 60,120	240
		34/40	240		
Differential SSTL-135	—	34/40	240	20, 30, 40, 60, 120	240
Differential SSTL-125	—	34/40	240	20, 30, 40, 60, 120	240
Differential HSUL-12	—	34/40/48/60/80	240	—	—

Sharing an OCT Calibration Block on Multiple I/O Banks

An OCT calibration block has the same V_{CCIO} as the I/O bank that contains the block. All I/O banks with the same V_{CCIO} can share one OCT calibration block, even if that particular I/O bank has an OCT calibration block.

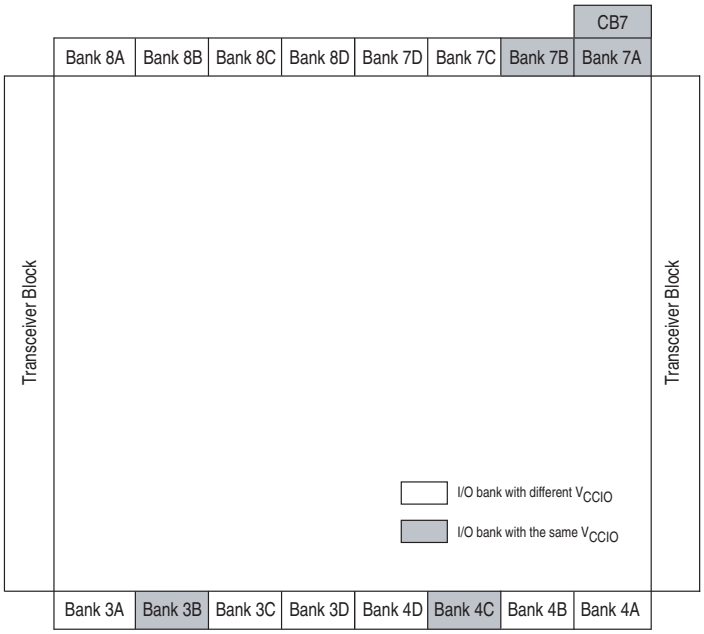
I/O banks that do not have calibration blocks share the calibration blocks in the I/O banks that have calibration blocks.

All I/O banks support OCT calibration with different V_{CCIO} voltage standards, up to the number of available OCT calibration blocks.

You can configure the I/O banks to receive calibration codes from any OCT calibration block with the same V_{CCIO}. If a group of I/O banks has the same V_{CCIO} voltage, you can use one OCT calibration block to calibrate the group of I/O banks placed around the periphery.

For example, Figure 5–6 shows a group of I/O banks that has the same V_{CCIO} voltage. This figure does not show transceiver calibration blocks.

Figure 5–6. Example of Calibrating Multiple I/O Banks with One Shared OCT Calibration Block (1)—*Preliminary*



Note to Figure 5–6:

(1) This is a top view of the silicon die that corresponds to a reverse view of the device package. This figure illustrates the highest density for Arria V devices.

Because banks 3B, 4C, and 7B have the same V_{CCIO} as bank 7A, you can calibrate all four I/O banks (3B, 4C, 7A, and 7B) with the OCT calibration block (CB7) located in bank 7A.

To enable this calibration, serially shift out the R_S OCT calibration codes from the OCT calibration block in bank 7A to the I/O banks around the periphery.

For more information about the OCT calibration block, refer to the *Dynamic Calibrated On-Chip Termination (ALTOCT) Megafunction User Guide*.

R_S OCT with Calibration

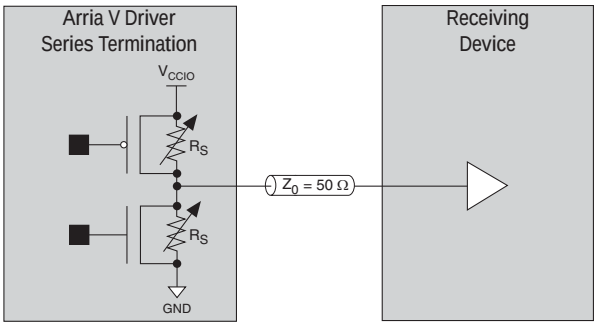
Arria V devices support R_S OCT with calibration in all banks.

The R_S OCT calibration circuit compares the total impedance of the I/O buffer to the external reference resistor connected to the RZQ pin and dynamically enables or disables the transistors until they match.

Calibration occurs at the end of device configuration. When the calibration circuit finds the correct impedance, the circuit powers down and stops changing the characteristics of the drivers.

Figure 5–7 shows the R_S as the intrinsic impedance of the output transistors.

Figure 5–7. R_S OCT with Calibration



R_S OCT Without Calibration

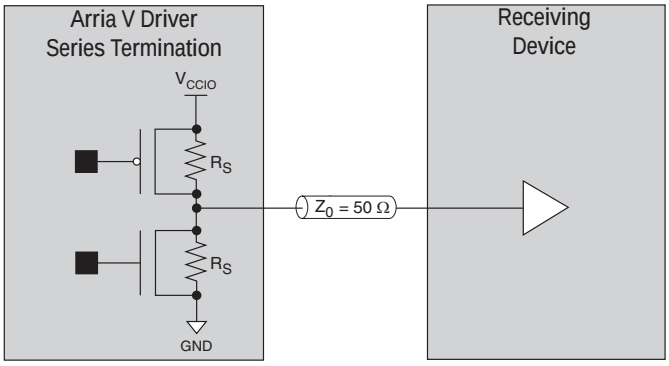
Arria V devices support R_S OCT for single-ended and voltage-referenced I/O standards.

Driver-impedance matching provides the I/O driver with controlled output impedance that closely matches the impedance of the transmission line. As a result, you can significantly reduce signal reflections on PCB traces.

When you select matching impedance, current strength is no longer selectable.

Figure 5–8 shows the R_S as the intrinsic impedance of the output transistors.

Figure 5–8. R_S OCT Without Calibration



R_T OCT with Calibration

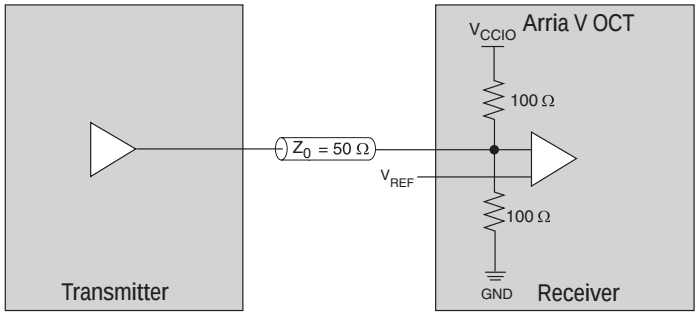
Arria V devices support R_T OCT with calibration in all banks.

The R_T OCT calibration circuit compares the total impedance of the I/O buffer to the external resistor connected to the RZQ pin. The circuit dynamically enables or disables the transistors until the total impedance of the I/O buffer matches the external resistor.

Calibration occurs at the end of the device configuration. When the calibration circuit finds the correct impedance, the circuit powers down and stops changing the characteristics of the drivers.

Figure 5–9 shows R_T OCT with calibration.

Figure 5–9. R_T OCT with Calibration



R_T OCT with calibration is available only for configuration of input and bidirectional pins. Output pin configurations do not support R_T OCT with calibration. When you use R_T OCT, the V_{CCIO} of the bank must match the I/O standard of the pin where you enable the R_T OCT.

Dynamic OCT

Dynamic OCT is useful for terminating a high-performance bidirectional path by optimizing the signal integrity depending on the direction of the data.

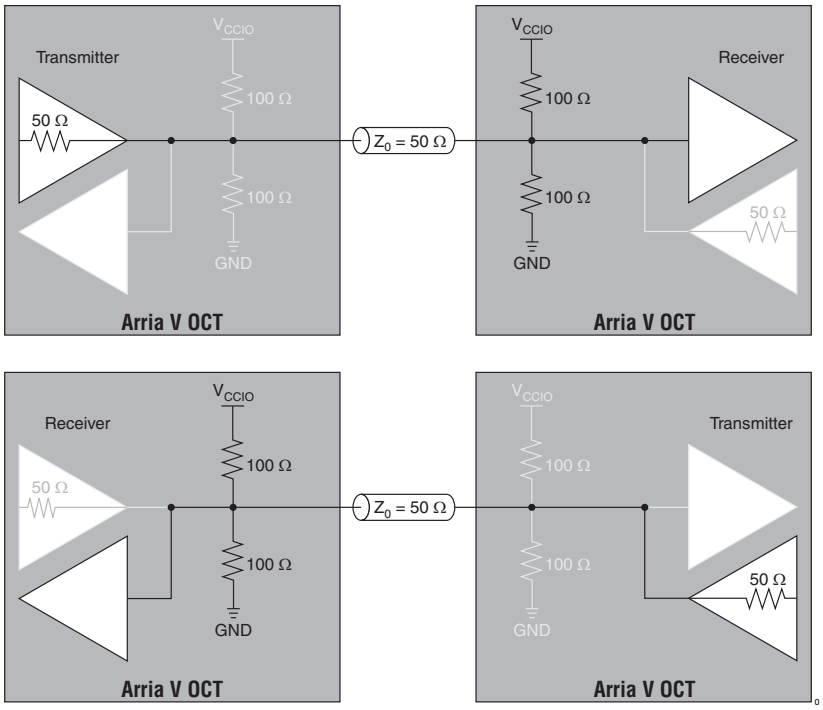
Dynamic R_T OCT or R_S OCT is enabled or disabled based on whether the bidirectional I/O acts as a receiver or driver, as listed in Table 5–11.

Table 5–11. Dynamic OCT Based on Bidirectional I/O

Dynamic OCT	Bidirectional I/O	State
Dynamic R_T OCT	Acts as a receiver	Enabled
	Acts as a driver	Disabled
Dynamic R_S OCT	Acts as a receiver	Disabled
	Acts as a driver	Enabled

Figure 5–10 shows the dynamic R_T OCT supported in the device.

Figure 5–10. Dynamic R_T OCT in Arria V Devices



Altera recommends that you use dynamic OCT for the DDR3 memory interface if you use the **SSTL-15**, **SSTL-135**, and **SSTL-125** I/O standards. These I/O standards save board space by reducing the number of external termination resistors used.

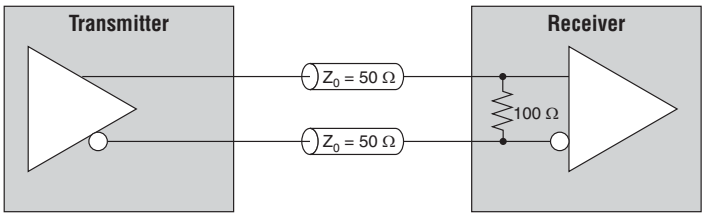
LVDS Input R_D OCT

Arria V devices support R_D OCT in all I/O banks.

You can use R_D OCT when you set the V_{CCIO} and V_{CCPD} to 2.5 V.

Arria V devices support OCT for differential **LVDS** input buffers with a nominal resistance value of 100 Ω, as shown in Figure 5–11.

Figure 5–11. Differential Input OCT



I/O Standards Termination Schemes

The following sections describe the termination schemes for the I/O standards supported in Arria V devices.

Table 5–12 lists the external termination schemes for the different I/O standards.

Table 5–12. I/O Standards External Termination Scheme (Part 1 of 2)

I/O Standard	External Termination Scheme
3.3-V LVTTL/3.3-V LVCMOS	No external termination required
3.0-V LVVTL/3.0-V LVCMOS	
2.5-V LVCMOS	
1.8-V LVCMOS	
1.5-V LVCMOS	
1.2-V LVCMOS	
3.0-V PCI	
3.0-V PCI-X	
SSTL-2 Class I	Single-Ended **SSTL** I/O Standard Termination
SSTL-2 Class II	
SSTL-18 Class I	
SSTL-18 Class II	
SSTL-15 Class I	
SSTL-15 Class II	
1.8-V HSTL Class I	Single-Ended **HSTL** I/O Standard Termination
1.8-V HSTL Class II	
1.5-V HSTL Class I	
1.5-V HSTL Class II	
1.2-V HSTL Class I	
1.2-V HSTL Class II	
SSTL-15 (1)	No external termination required
SSTL-135 (1)	
SSTL-125 (1)	
HSUL-12	
Differential SSTL-2 Class I	**Differential SSTL** I/O Standard Termination
Differential SSTL-2 Class II	
Differential SSTL-18 Class I	
Differential SSTL-18 Class II	
Differential SSTL-15 Class I	
Differential SSTL-15 Class II	

Table 5–12. I/O Standards External Termination Scheme (Part 2 of 2)

I/O Standard	External Termination Scheme
Differential 1.8-V HSTL Class I	**Differential HSTL** I/O Standard Termination
Differential 1.8-V HSTL Class II	
Differential 1.5-V HSTL Class I	
Differential 1.5-V HSTL Class II	
Differential 1.2-V HSTL Class I	
Differential 1.2-V HSTL Class II	
Differential SSTL-15 *(1)*	No external termination required
Differential SSTL-135 *(1)*	
Differential SSTL-125 *(1)*	
Differential HSUL-12	
LVDS	**LVDS** I/O Standard Termination
RSDS *(2)*	**RSDS/mini-LVDS** I/O Standard Termination
Mini-LVDS *(3)*	
LVPECL	**Differential LVPECL** I/O Standard Termination

Notes to Table 5–12:

(1) Altera recommends using dynamic OCT with these I/O standards to save board space and cost by reducing the number of external termination resistors.

(2) Arria V devices support the true **RSDS** output standard with data rates of up to 230 Mbps using true **LVDS** output buffer types on all I/O banks.

(3) Arria V devices support the true **mini-LVDS** output standard with data rates of up to 340 Mbps using true **LVDS** output buffer types on all I/O banks.

Single-Ended I/O Standard Termination

Voltage-referenced I/O standards require an input V_{REF} and a termination voltage (V_{TT}). The reference voltage of the receiving device tracks the termination voltage of the transmitting device.

The supported I/O standards such as **SSTL-15**, **SSTL-135**, **SSTL-125**, and **SSTL-12** typically do not require external board termination.

Altera recommends using dynamic OCT with these I/O standards to save board space and cost. Dynamic OCT reduces the number of external termination resistors used.

Figure 5–12 shows the details of **SSTL** I/O termination on Arria V devices.

Figure 5–12. SSTL I/O Standard Termination

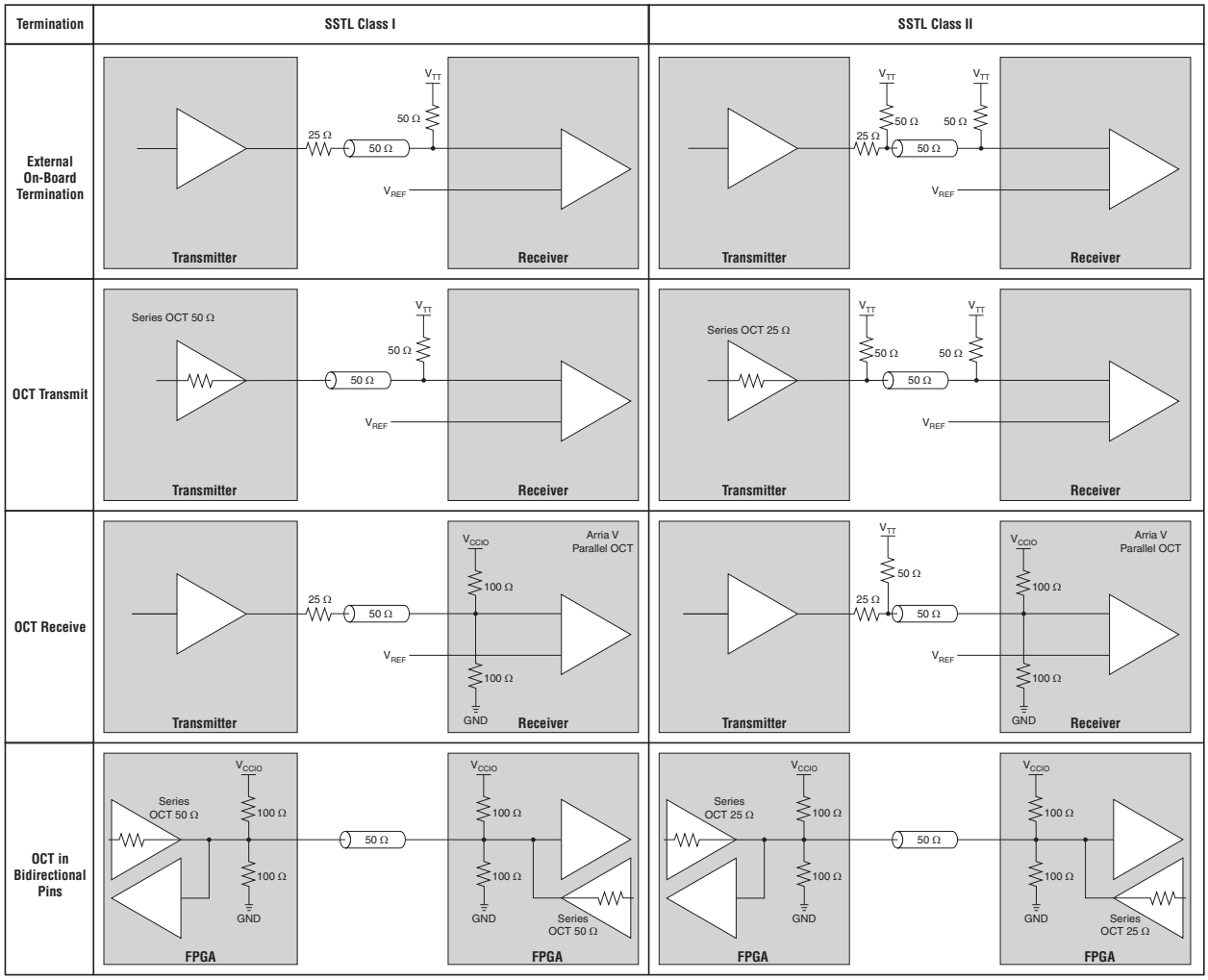
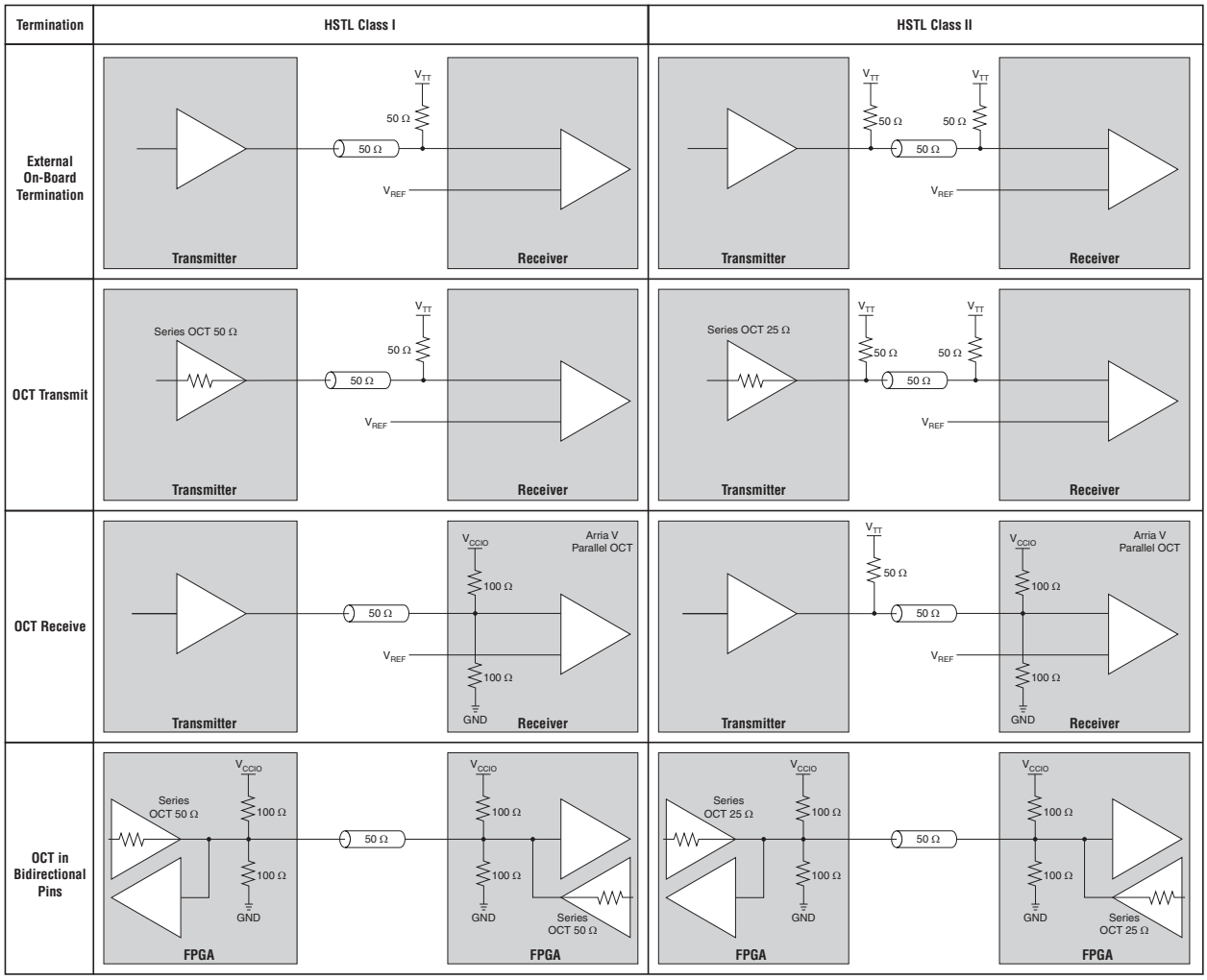


Figure 5–13 shows the details of **HSTL** I/O termination on Arria V devices.

Figure 5–13. HSTL I/O Standard Termination



You cannot use R_S and R_T OCT simultaneously. For more information, refer to "Dynamic OCT" on page 5–24.

Differential I/O Standard Termination

The I/O pins are organized in pairs to support differential I/O standards. Each I/O pin pair can support differential input and output buffers.

The supported I/O standards such as **differential SSTL-12**, **differential SSTL-15**, **differential SSTL-125**, and **differential SSTL-135** typically do not require external board termination.

Altera recommends using these I/O standards with dynamic OCT schemes to save board space and costs by reducing the number of external termination resistors used.

Differential HSTL, **SSTL**, and **HSUL** inputs use **LVDS** differential input buffers. However, R_D support is only available if the I/O standard is **LVDS**.

Differential HSTL, **SSTL**, and **HSUL** outputs are not true differential outputs. They use two single-ended outputs with the second output programmed as inverted.

Figure 5–14 shows the details of **Differential SSTL** I/O termination on Arria V devices.

Figure 5–14. Differential SSTL I/O Standard Termination

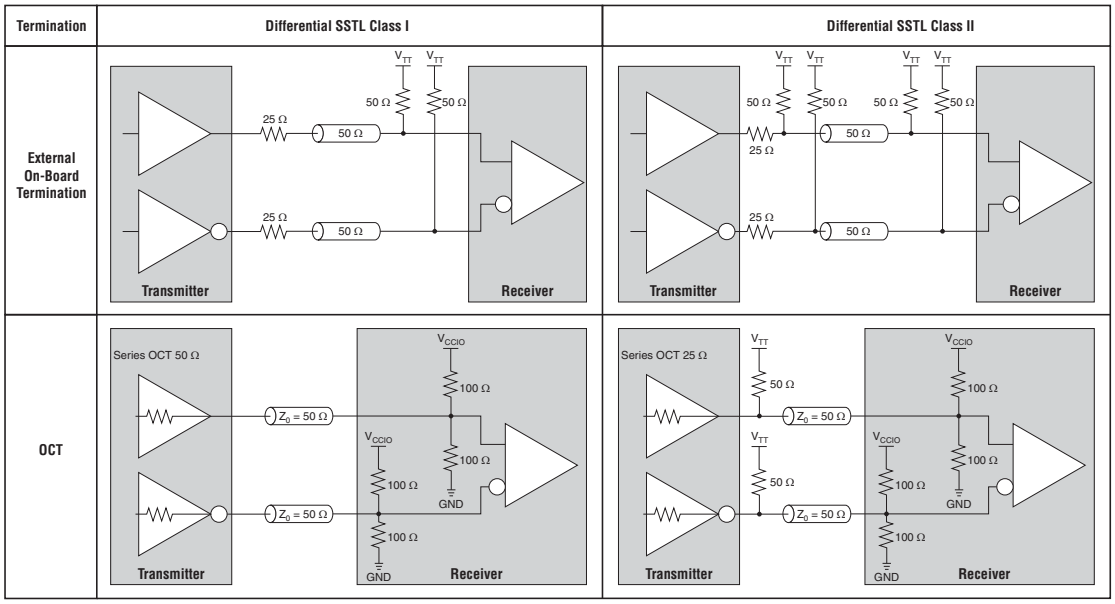
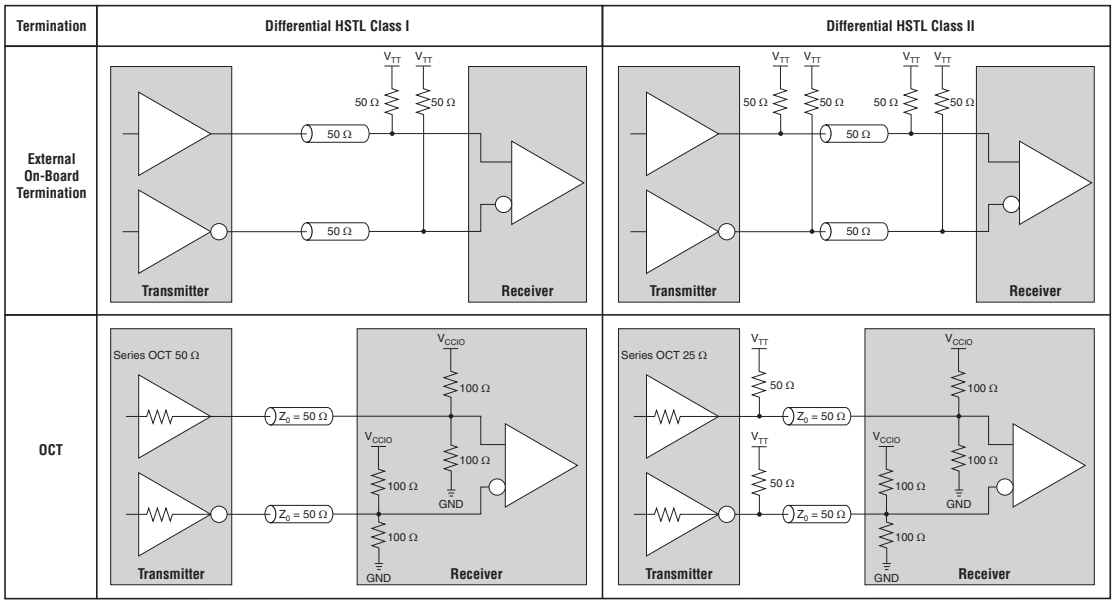


Figure 5–15 shows the details of **Differential HSTL** I/O standard termination on Arria V devices.

Figure 5–15. Differential HSTL I/O Standard Termination

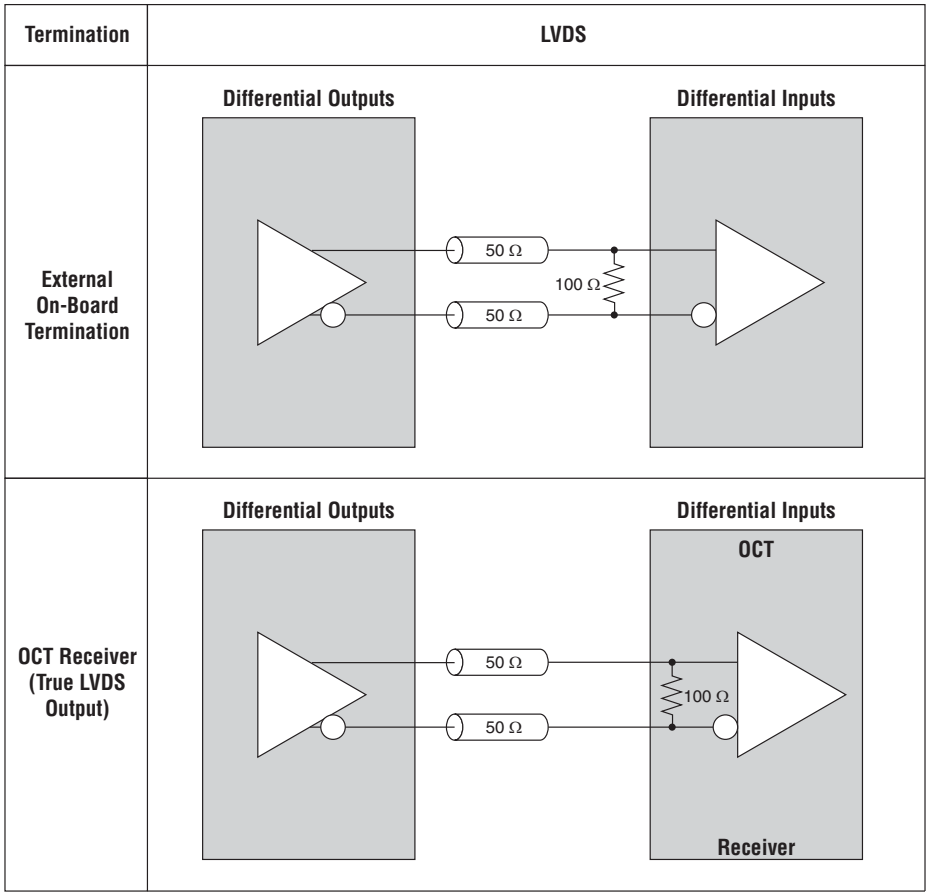


LVDS, RSDS, and Mini-LVDS I/O Standard Termination

All I/O banks have dedicated circuitry to support the true **LVDS**, **RSDS**, and **mini-LVDS** I/O standards by using true **LVDS** output buffers without resistor networks.

Figure 5–16 shows the **LVDS** I/O standard termination. The on-chip differential resistor is available in all I/O banks.

Figure 5–16. LVDS I/O Standard Termination



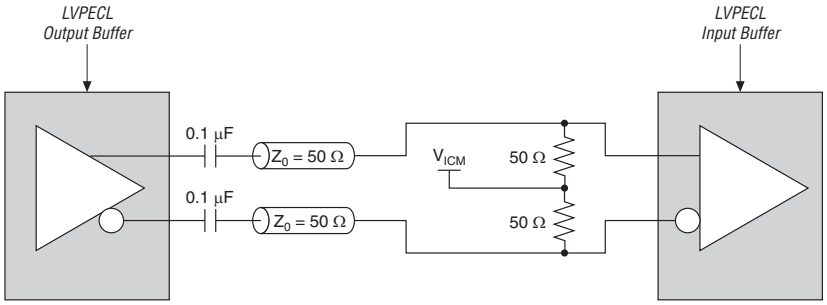
LVPECL I/O Standard Termination

Arria V devices support the **LVPECL** I/O standard on input clock pins only. **LVPECL** output operation is not supported. Use **LVDS** input buffers to support the **LVPECL** input operation.

Use AC coupling when the **LVPECL** common-mode voltage of the output buffer does not match the **LVPECL** input common-mode voltage.

Figure 5–17 shows the AC-coupled termination scheme.

Figure 5–17. LVPECL AC-Coupled Termination *(1)*

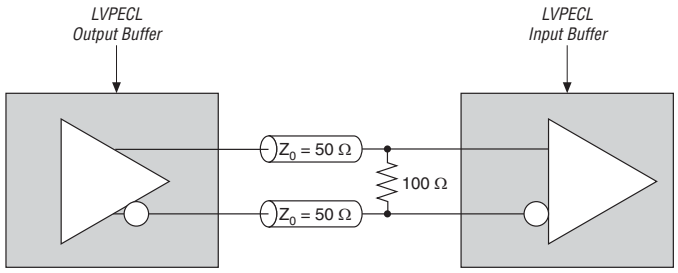


Note to Figure 5–17:

(1) The **LVPECL** AC/DC-coupled termination is applicable only when you use an Altera® FPGA transmitter.

Support for DC-coupled **LVPECL** is available if the **LVPECL** output common mode voltage is within the Arria V **LVPECL** input buffer specification, as shown in Figure 5–18.

Figure 5–18. LVPECL DC-Coupled Termination *(1)*



Note to Figure 5–18:

(1) The **LVPECL** AC/DC-coupled termination is applicable only when you use an Altera FPGA transmitter.

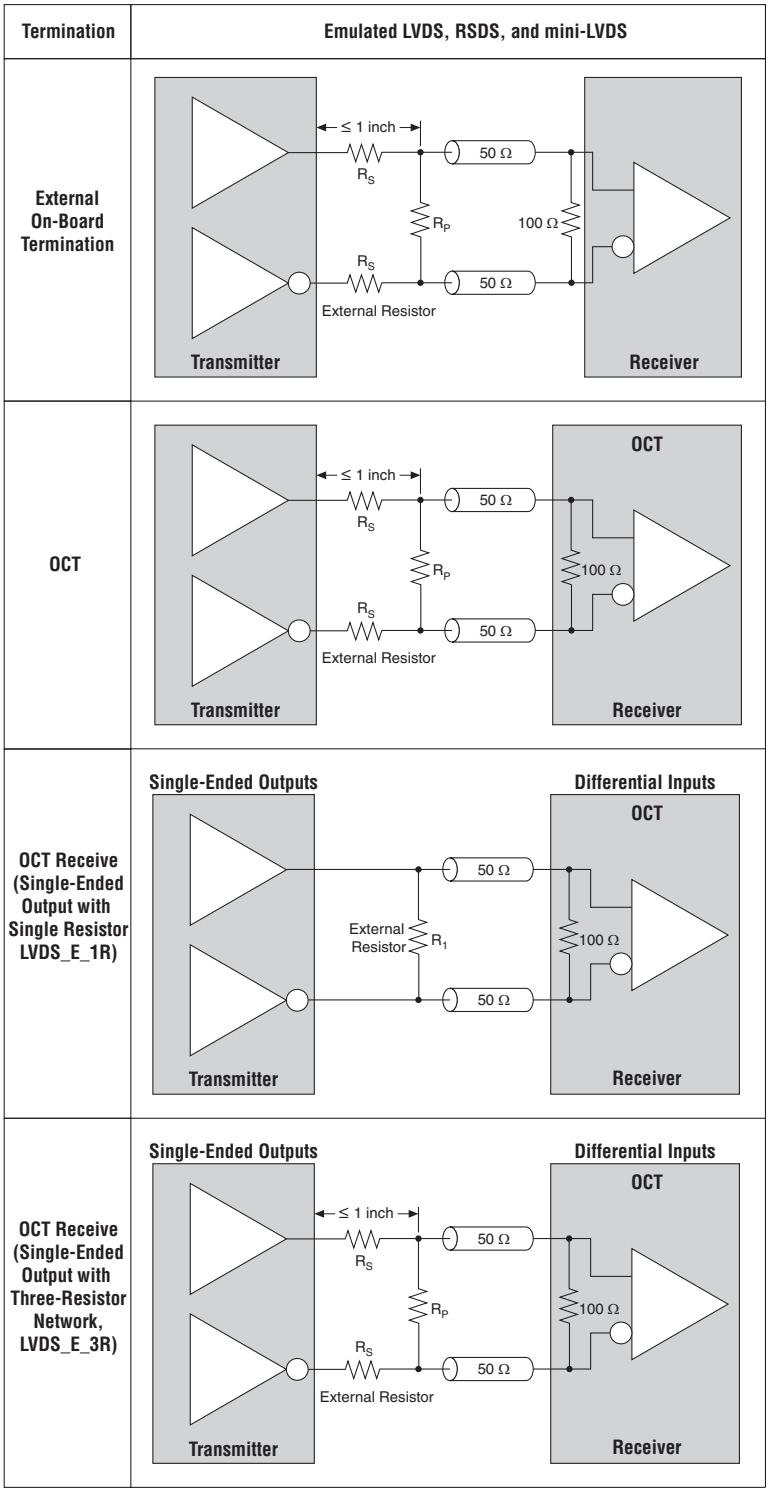
Emulated LVDS, RSDS, and Mini-LVDS I/O Standard Termination

The I/O banks also support **emulated LVDS**, **RSDS**, and **mini-LVDS** I/O standards.

Emulated **LVDS, RSDS** and **mini-LVDS** output buffers use two single-ended output buffers with either an external single resistor for data rates up to 200 Mbps or an external three-resistor network for data rates up to 1.1 Gbps, and can be tri-stated.

The output buffers are available in all I/O banks, as shown in Figure 5–19.

Figure 5–19. Emulated LVDS, RSDS, or Mini-LVDS I/O Standard Termination *(1)*



Note to Figure 5–19:

(1) The R_1, R_S, and R_P values are pending characterization.

To meet the **RSDS** or **mini-LVDS** specifications, you require a resistor network to attenuate the output-voltage swing.

You can modify the three-resistor network values to reduce power or improve the noise margin. Choose resistor values that satisfy Equation 5–1.

Equation 5–1. Resistor Network Calculation

$$\frac{R_S \times \frac{R_P}{2}}{R_S + \frac{R_P}{2}} = 50\ \Omega$$

Altera recommends that you perform additional simulations with IBIS or SPICE models to validate that the custom resistor values meet the **RSDS** or **mini-LVDS** I/O standard requirements.

For more information about the **RSDS** I/O standard, refer to the *RSDS Specification* document available on the National Semiconductor web site (www.national.com).

Document Revision History

Table 5–13 lists the revision history for this chapter.

Table 5–13. Document Revision History

Date	Version	Changes
June 2012	2.0	Updated for the Quartus II software v12.0 release: ■ Restructured chapter. ■ Added "Design Considerations", "V_{CCIO} Restriction", "LVDS Channels", "Modular I/O Banks", and "OCT Calibration Block" sections. ■ Added Figure 5–1, Figure 5–2, and Figure 5–3 ■ Updated Table 5–1, Table 5–6, and Table 5–8. ■ Updated Figure 5–19 with emulated **LVDS** with external single resistor.
February 2012	1.2	Updated Table 5–3.
November 2011	1.1	■ Restructured chapter. ■ Updated Table 5–3.
May 2011	1.0	Initial release.

6. High-Speed Differential I/O Interfaces and DPA in Arria V Devices

AV52006-2.0

This chapter describes the high-speed differential I/O interfaces and dynamic phase aligner (DPA) in Arria® V devices.

The high-speed differential I/O interfaces and DPA features in Arria V devices provide advantages over single-ended I/Os and contribute to the achievable overall system bandwidth. Arria V devices support the **LVDS**, **Mini-LVDS**, and reduced swing differential signaling (**RSDS**) differential I/O standards.

This chapter contains the following sections:

For more information about the differential I/O standards, refer to the *I/O Features in Arria V Devices* chapter.

Dedicated High-Speed I/O Circuitries

The following dedicated circuitries are available in the Arria V device family to support high-speed differential I/O:

- Differential I/O buffer
- Transmitter serializer
- Receiver deserializer
- Data realignment
- DPA
- Synchronizer (FIFO buffer)
- Fractional phase-locked loops (PLLs)

SERDES and DPA Bank Locations

The dedicated serializer/deserializer (SERDES) and DPA circuitry that supports high-speed differential I/Os is located in the top and bottom banks of the Arria V devices. Figure 6–1 through Figure 6–3 show the high-level location of SERDES/DPA in the Arria V device.

Figure 6–1. High-Speed Differential I/Os with DPA Locations for Arria V GX A1 and A3 Devices, and Arria V GT C3 Device.

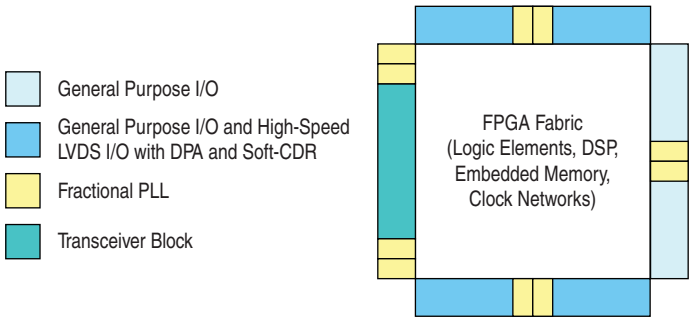


Figure 6–2. High-Speed Differential I/Os with DPA Locations for Arria V GX A5, A7, B1, and B3 Devices, and Arria V GT C7, D3, and D7 Devices

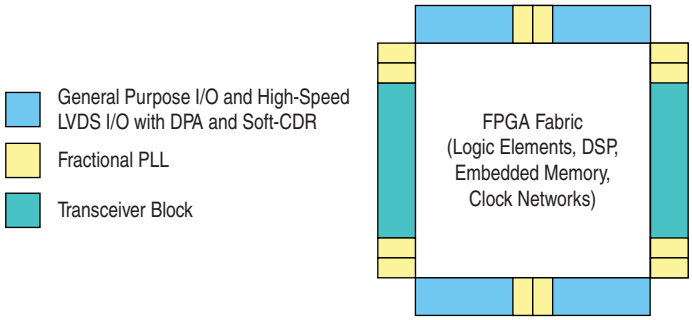
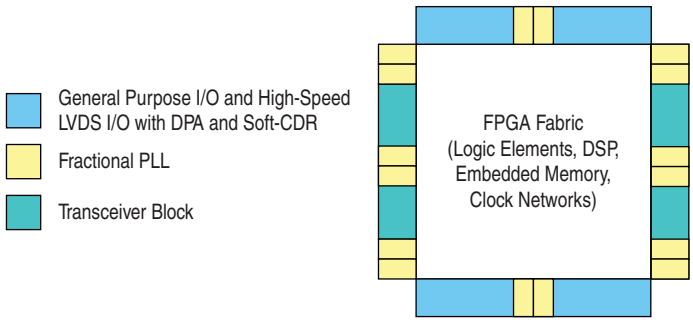


Figure 6–3. High-Speed Differential I/Os with DPA Locations for Arria V GX B5 and B7 Devices

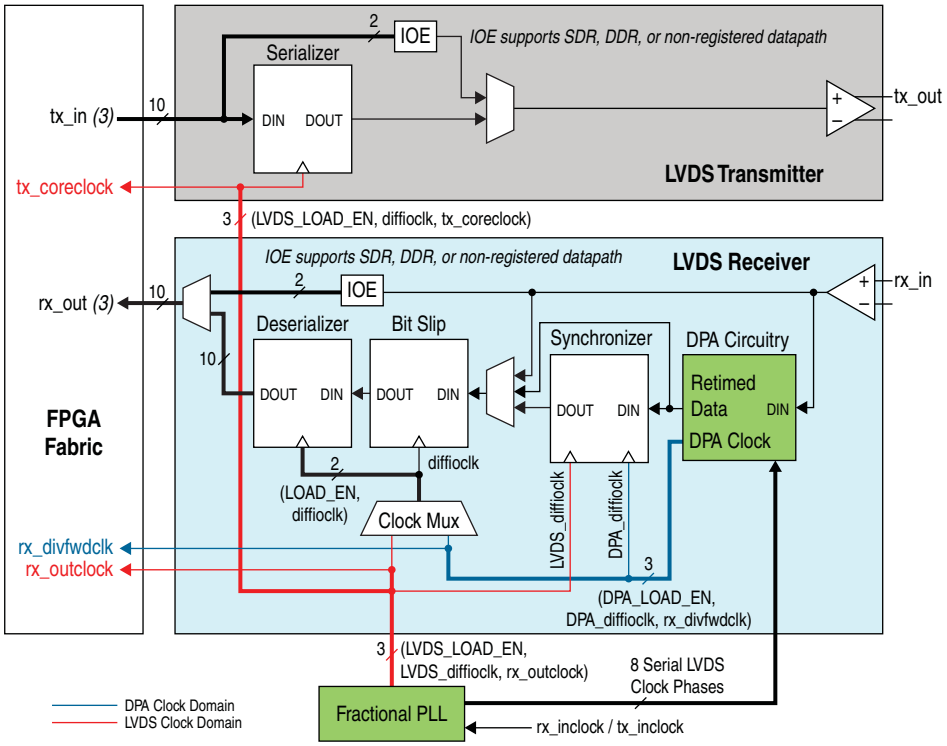


LVDS SERDES Circuitry

The SERDES circuitry supports high-speed **LVDS** interfaces at data rates of up to 1.6 Gbps. You can configure the SERDES circuitry to support source-synchronous communication protocols such as RapidIO®, XSBI, serial peripheral interface (SPI), and asynchronous protocols such as Gigabit Ethernet (GbE).

Figure 6–4 shows a transmitter and receiver block diagram for the **LVDS** SERDES circuitry. This diagram shows the interface signals of the transmitter and receiver data path.

Figure 6–4. LVDS SERDES *(1), (2)*



Notes to Figure 6–4:

(1) This diagram shows a shared PLL between the transmitter and receiver. If the transmitter and receiver do not share the same PLL, you require two fractional PLLs.

(2) In single data rate (SDR) and double data rate (DDR) modes, the data width is 1 and 2 bits, respectively.

(3) The `tx_in` and `rx_out` ports have a maximum data width of 10 bits.

For a list of the **LVDS** transmitter and receiver ports and settings using ALTLVDS, refer to *LVDS SERDES Transmitter/Receiver (ALTLVDS_RX/TX) Megafunction User Guide*.

LVDS Channels

Arria V devices support **LVDS** on all I/O banks. Both row and column I/Os support true **LVDS** input buffers with R_D OCT and true **LVDS** output buffers. Alternatively, you can configure the **LVDS** pins as emulated **LVDS** output buffers that use two single-ended output buffers with an external resistor network to support **LVDS**, **mini-LVDS**, and **RSDS** standards.

Arria V devices offer single-ended I/O reference clock support for the **LVDS** SERDES.

☞ Emulated differential output buffers support tri-state capability.

True LVDS Buffers

Table 6–1 lists the number of true **LVDS** buffers supported in Arria V devices. You can use the unutilized true **LVDS** buffers as emulated **LVDS** output buffers (eTX). For example, with the 672-pin Arria V GX A5 device, if you use 30 pairs of true **LVDS** input buffer (RX), you can use the remaining input buffer pairs as eTX.

Table 6–1. LVDS Channels Supported in Arria V Devices—*Preliminary* (1) (Part 1 of 2)

Variant	Member Code	Package	Side (2)	TX (3)	RX (3)
Arria V GX	A1 A3 (4)	672-pin FineLine BGA, Flip Chip	Top	26	34
			Bottom	26	34
		896-pin FineLine BGA, Flip Chip	Top	34	40
			Bottom	34	40
	A5 A7	672-pin FineLine BGA, Flip Chip	Top	34	44
			Bottom	34	44
		896-pin FineLine BGA, Flip Chip	Top	42	48
			Bottom	42	48
		1152-pin FineLine BGA, Flip Chip	Top	60	68
			Bottom	60	68
	B1 B3	896-pin FineLine BGA, Flip Chip	Top	42	48
			Bottom	42	48
		1152-pin FineLine BGA, Flip Chip	Top	60	68
			Bottom	60	68
		1517-pin FineLine BGA, Flip Chip	Top	80	88
			Bottom	80	88
	B5 B7	1152-pin FineLine BGA, Flip Chip	Top	60	68
			Bottom	60	68
		1517-pin FineLine BGA, Flip Chip	Top	80	88
			Bottom	80	88

Table 6–1. LVDS Channels Supported in Arria V Devices—*Preliminary* (1) (Part 2 of 2)

Variant	Member Code	Package	Side (2)	TX (3)	RX (3)
Arria V GT	C3	672-pin FineLine BGA, Flip Chip	Top	26	34
			Bottom	26	34
		896-pin FineLine BGA, Flip Chip	Top	34	40
			Bottom	34	40
	C7	896-pin FineLine BGA, Flip Chip	Top	42	48
			Bottom	42	48
		1152-pin FineLine BGA, Flip Chip	Top	60	68
			Bottom	60	68
	D3	896-pin FineLine BGA, Flip Chip	Top	42	48
			Bottom	42	48
		1152-pin FineLine BGA, Flip Chip	Top	60	68
			Bottom	60	68
		1517-pin FineLine BGA, Flip Chip	Top	80	88
			Bottom	80	88
	D7	1152-pin FineLine BGA, Flip Chip	Top	60	68
			Bottom	60	68
		1517-pin FineLine BGA, Flip Chip	Top	80	88
			Bottom	80	88
Arria V SX	B3 B5	896-pin FineLine BGA, Flip Chip	Top	42	48
			Bottom	42	48
		1152-pin FineLine BGA, Flip Chip	Top	60	68
			Bottom	60	68
		1517-pin FineLine BGA, Flip Chip	Top	80	88
			Bottom	80	88
Arria V ST	D3 D5	896-pin FineLine BGA, Flip Chip	Top	42	48
			Bottom	42	48
		1152-pin FineLine BGA, Flip Chip	Top	60	68
			Bottom	60	68
		1517-pin FineLine BGA, Flip Chip	Top	80	88
			Bottom	80	88

Notes to Table 6–1:

(1) **LVDS** channel count does not include dedicated clock pins.

(2) Dedicated SERDES and DPA is available for top and bottom banks only.

(3) You can use all unutilized TX and RX as eTX.

(4) The right side I/O banks do not contain true **LVDS** output buffer. However, emulated **LVDS** output buffer (eTX) is still supported.

Differential Transmitter

The Arria V transmitter contains dedicated circuitry to support **LVDS** signaling. The differential transmitter buffers support the following features:

- **LVDS** signaling that can drive out **LVDS**, **mini-LVDS**, and **RSDS** signals
- Programmable V_{OD} and programmable pre-emphasis

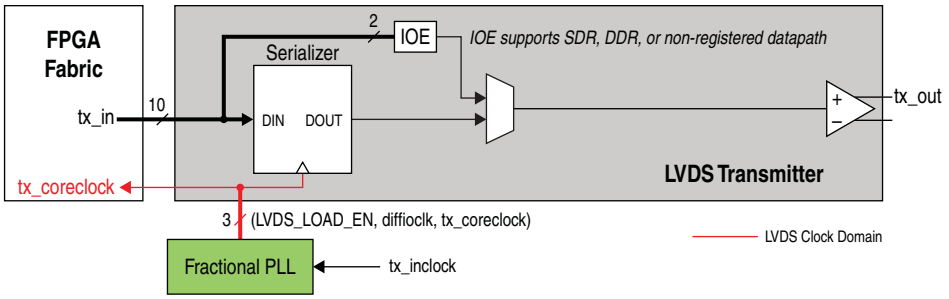
Transmitter Blocks

The dedicated circuitry consists of a true differential buffer, a serializer, and fractional PLLs that you can share between the transmitter and receiver. The serializer takes up to 10 bits wide parallel data from the FPGA fabric, clocks it into the load registers, and serializes it using shift registers that are clocked by the fractional PLL before sending the data to the differential buffer. The MSB of the parallel data is transmitted first.

To drive the **LVDS** channels, you must use the PLLs in integer PLL mode.

Figure 6–5 shows a block diagram of the Arria V transmitter.

Figure 6–5. Arria V Transmitter *(1), (2)*



Notes to Figure 6–5:

(1) In SDR and DDR modes, the data width is 1 and 2 bits, respectively.

(2) The `tx_in` port has a maximum data width of 10 bits.

Transmitter Clocking

The fractional PLL generates the load enable (`LVDS_LOAD_EN`) signal and the `diffioclk` signal (the clock running at serial data rate) that clocks the load and shift registers. You can statically set the serialization factor to ×3, ×4, ×5, ×6, ×7, ×8, ×9, or ×10 using the Quartus® II software. The load enable signal is derived from the serialization factor setting.

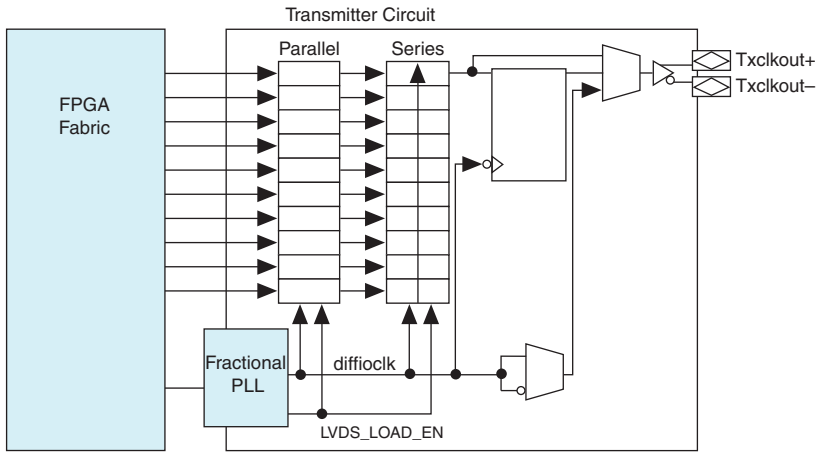
You can configure any Arria V transmitter data channel to generate a source-synchronous transmitter clock output. This flexibility allows the placement of the output clock near the data outputs to simplify board layout and reduce clock-to-data skew.

Different applications often require specific clock-to-data alignments or specific data-rate-to-clock-rate factors. You can specify these settings statically in the Quartus II MegaWizard™ Plug-In Manager:

- The transmitter can output a clock signal at the same rate as the data—with a maximum output clock frequency that each speed grade of the device supports.
- You can divide the output clock by a factor of 1, 2, 4, 6, 8, or 10, depending on the serialization factor.
- You can set the phase of the clock in relation to the data at 0° or 180° (edge- or center-aligned). The fractional PLLs provide additional support for other phase shifts in 45° increments.

Figure 6–6 shows the Arria V transmitter in clock output mode. In clock output mode, you can use an **LVDS** channel as a clock output channel.

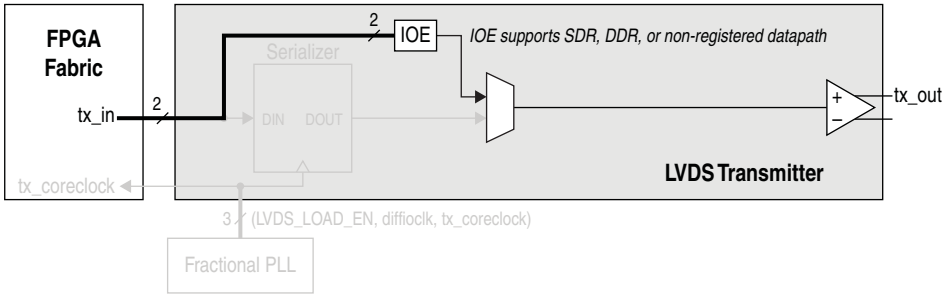
Figure 6–6. Arria V Transmitter in Clock Output Mode



Serializer Bypass for DDR and SDR Operations

You can bypass the Arria V serializer to support DDR (×2) and SDR (×1) operations to achieve a serialization factor of 2 and 1, respectively. The I/O element (IOE) contains two data output registers that can each operate in either DDR or SDR mode. Figure 6–7 shows the serializer bypass path.

Figure 6–7. Arria V Serializer Bypass *(1), (2), (3)*



Notes to Figure 6–7:

(1) All disabled blocks and signals are grayed out.
(2) In DDR mode, `tx_inclock` clocks the IOE register. In SDR mode, data is passed directly through the IOE.
(3) In SDR and DDR modes, the data width to the IOE is 1 and 2 bits, respectively.

Programmable V_{OD}

The programmable V_{OD} settings allow you to adjust the output eye opening to optimize the trace length and power consumption. A higher V_{OD} swing improves voltage margins at the receiver end, and a smaller V_{OD} swing reduces power consumption. You can statically adjust the V_{OD} of the differential signal by changing the V_{OD} settings in the Assignment Editor.

Figure 6–8 shows the V_{OD} of the differential **LVDS** output.

Figure 6–8. Differential V_{OD}

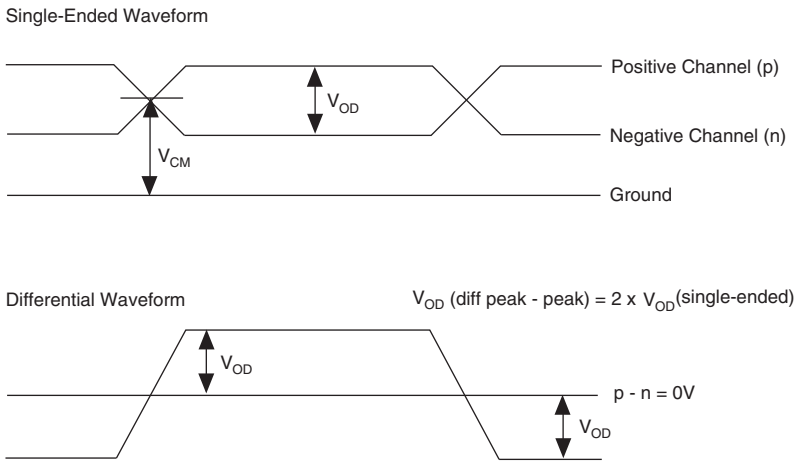


Table 6–2 lists the assignment name for programmable V_{OD} and its possible values in the Quartus II software Assignment Editor.

Table 6–2. Quartus II Software Assignment Editor—Programmable V_{OD}

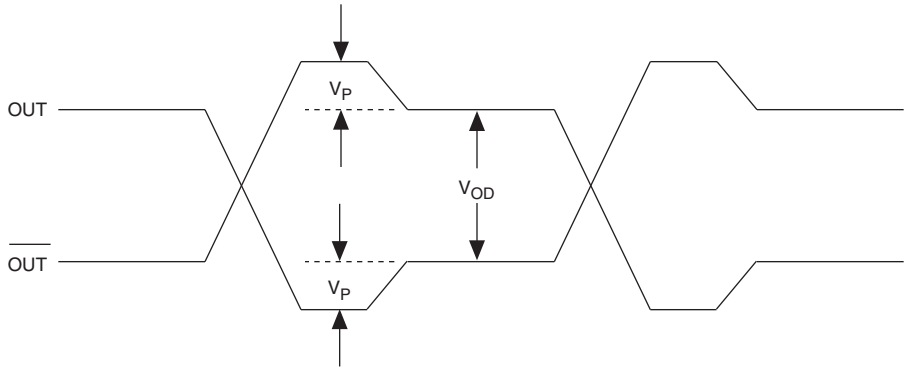
Field	Assignment
To	`tx_out`
Assignment name	Programmable Differential Output Voltage (V_{OD})
Allowed values	0, 1, 2, 3

Programmable Pre-Emphasis

The V_{OD} setting and the output impedance of the driver set the output current limit of a high-speed transmission signal. At a high frequency, the slew rate may not be fast enough to reach the full V_{OD} level before the next edge, producing pattern-dependent jitter.

Pre-emphasis increases the amplitude of the high-frequency component of the output signal, and thus helps to compensate for the frequency-dependent attenuation along the transmission line. Figure 6–9 shows the **LVDS** output with pre-emphasis.

Figure 6–9. Programmable Pre-Emphasis *(1)*



Note to Figure 6–9:

(1) V_P— voltage boost from pre-emphasis. V_{OD}— differential output voltage (peak–peak).

Table 6–3 lists the assignment name for programmable pre-emphasis and its possible values in the Quartus II software Assignment Editor.

Table 6–3. Quartus II Software Assignment Editor—Programmable Pre-Emphasis

Field	Assignment
To	`tx_out`
Assignment name	Programmable Pre-emphasis
Allowed values	0 (disable) and 1 (enable)

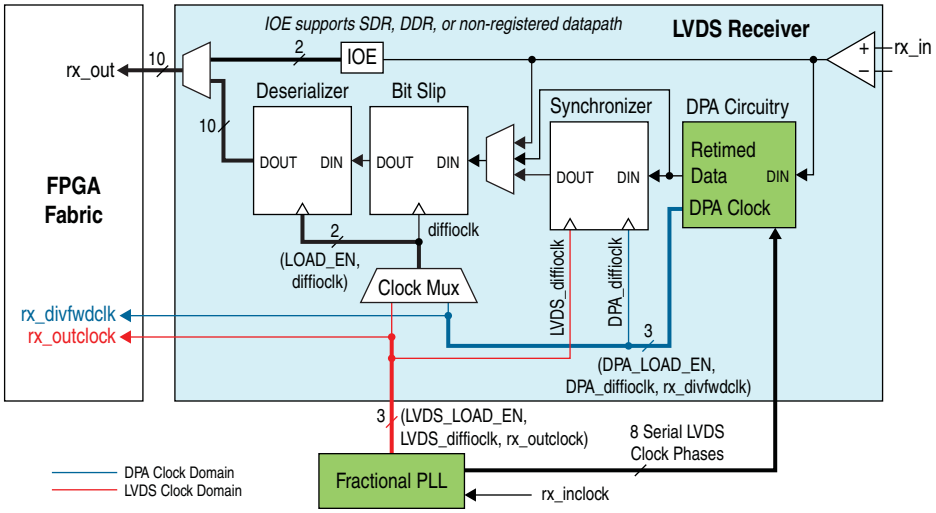
Differential Receiver

The receiver has a differential buffer and fractional PLLs that you can share among the transmitter and receiver, a DPA block, a synchronizer, a data realignment block, and a deserializer. The differential buffer can receive **LVDS**, **mini-LVDS**, and **RSDS** signal levels. You can statically set the I/O standard of the receiver pins to **LVDS**, **mini-LVDS**, or **RSDS** in the Quartus II software Assignment Editor.

Figure 6–10 shows the hardware blocks of the Arria V receiver.

To drive the **LVDS** channels, you must use the PLLs in integer PLL mode.

Figure 6–10. Receiver Block Diagram *(1), (2)*



Notes to Figure 6–10:

(1) In SDR and DDR modes, the data width from the IOE is 1 and 2 bits, respectively.

(2) The `rx_out` port has a maximum data width of 10 bits.

The deserializer includes shift registers and parallel load registers, and sends a maximum of 10 bits to the internal logic.

Receiver Blocks

The differential receiver has the following hardware blocks:

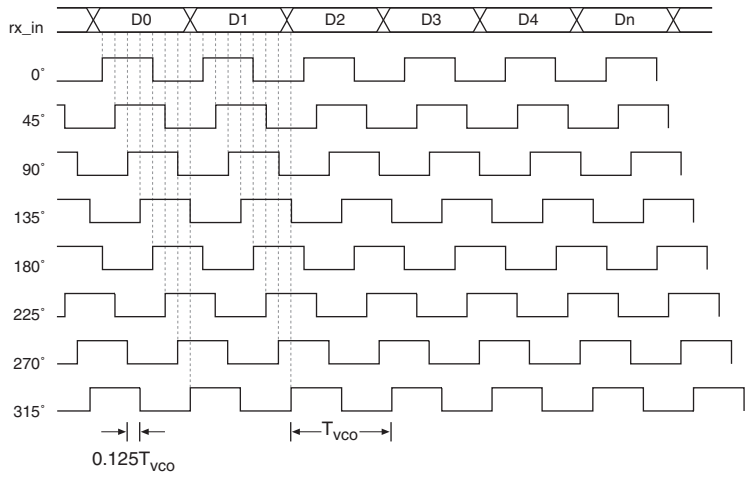
- "DPA Block" on page 6–10
- "Synchronizer" on page 6–11
- "Data Realignment Block (Bit Slip)" on page 6–12
- "Deserializer" on page 6–13

DPA Block

The DPA block takes in high-speed serial data from the differential input buffer and selects one of the eight phases that the fractional PLLs generate to sample the data. The DPA chooses a phase closest to the phase of the serial data. The maximum phase offset between the received data and the selected phase is 1/8 UI, which is the maximum quantization error of the DPA. The eight phases of the clock are equally divided, offering a 45° resolution.

Figure 6–11 shows the possible phase relationships between the DPA clocks and the incoming serial data.

Figure 6–11. DPA Clock Phase to Serial Data Timing Relationship *(1)*



Note to Figure 6–11:

(1) T_{VCO} is defined as the PLL serial clock period.

The DPA block continuously monitors the phase of the incoming serial data and selects a new clock phase if it is required. You can prevent the DPA from selecting a new clock phase by asserting the optional `RX_DPLL_HOLD` port, which is available for each channel.

DPA circuitry does not require a fixed training pattern to lock to the optimum phase out of the eight phases. After reset or power up, the DPA circuitry requires transitions on the received data to lock to the optimum phase. An optional output port, `RX_DPA_LOCKED`, is available to indicate an initial DPA lock condition to the optimum phase after power up or reset. This signal is not deasserted if the DPA selects a new phase out of the eight clock phases to sample the received data. Do not use the `rx_dpa_locked` signal to determine a DPA loss-of-lock condition. Use data checkers such as a cyclic redundancy check (CRC) or diagonal interleaved parity (DIP-4) to validate the data.

An independent reset port, `RX_RESET`, is available to reset the DPA circuitry. You must retrain the DPA circuitry after reset.

The DPA block is bypassed in non-DPA mode.

Synchronizer

The synchronizer is a 1-bit wide and 6-bit deep FIFO buffer that compensates for the phase difference between `DPA_diffioclk`—the optimal clock that the DPA block selects—and the `LVDS_diffioclk` that the fractional PLLs produce. The synchronizer can only compensate for phase differences, not frequency differences, between the data and the receiver's input reference clock.

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 data checker indicates corrupted, received data.

The synchronizer circuit is bypassed in non-DPA and soft-CDR mode.

Data Realignment Block (Bit Slip)

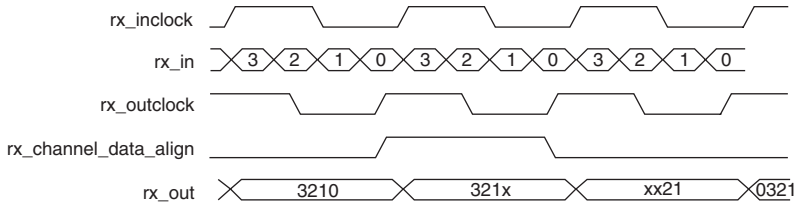
Skew in the transmitted data along with skew added by the link causes channel-to-channel skew on the received serial data streams. If you enable the DPA, the received data is captured with different clock phases on each channel. This difference may cause misalignment of the received data from channel to channel. To compensate for this channel-to-channel skew and establish the correct received word boundary at each channel, each receiver channel has a dedicated data realignment circuit that realigns the 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 on the rising edge of RX_CHANNEL_DATA_ALIGN. The requirements for the RX_CHANNEL_DATA_ALIGN signal include:

- 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 the parallel clock.
- The signal is an edge-triggered signal.
- The valid data is available two parallel clock cycles after the rising edge of RX_CHANNEL_DATA_ALIGN.

Figure 6–12 shows receiver output (RX_OUT) after one bit slip pulse with the deserialization factor set to **4**.

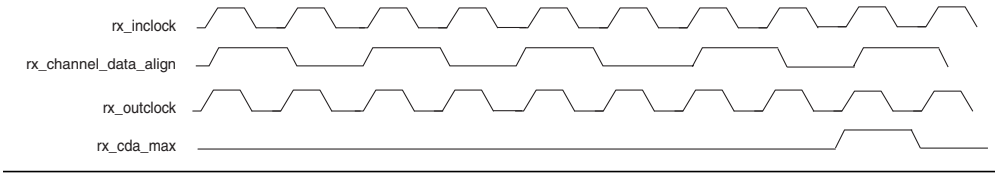
Figure 6–12. Data Realignment Timing



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. Set the programmable bit rollover point equal to, or greater than, the deserialization factor—allowing enough depth in the word alignment circuit to slip through a full word. You can set the value of the bit rollover point using the MegaWizard Plug-In Manager. An optional status port, RX_CDA_MAX, is available to the FPGA fabric from each channel to indicate the reaching of the preset rollover point.

Figure 6–13 shows a preset value of four bit-times before rollover occurs. The `rx_cda_max` signal pulses for one `rx_outclock` cycle to indicate that rollover has occurred.

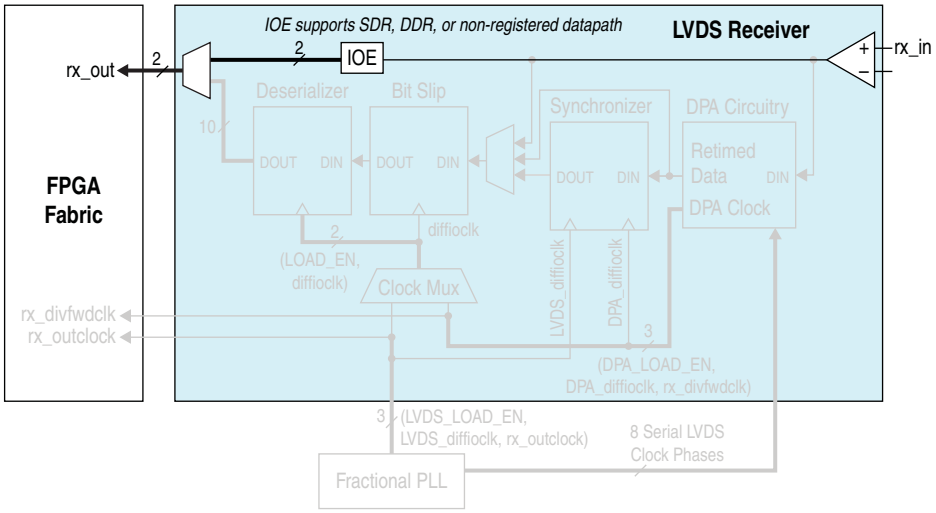
Figure 6–13. Receiver Data Realignment Rollover



Deserializer

You can statically set the deserialization factor to ×3, ×4, ×5, ×6, ×7, ×8, ×9, or ×10 by using the Quartus II software. You can bypass the Arria V deserializer in the Quartus II MegaWizard Plug-In Manager to support DDR (×2) or SDR (×1) operations, as shown Figure 6–14. You cannot use the DPA and data realignment circuit when you bypass the deserializer. The IOE contains two data input registers that can operate in DDR or SDR mode.

Figure 6–14. Deserializer Bypass *(1), (2), (3)*



Notes to Figure 6–14:

(1) All disabled blocks and signals are grayed out.
(2) In DDR mode, `rx_inclock` clocks the IOE register. In SDR mode, data is directly passed through the IOE.
(3) In SDR and DDR modes, the data width from the IOE is 1 and 2 bits, respectively.

Receiver Modes

The Arria V device family supports three different receiver modes:

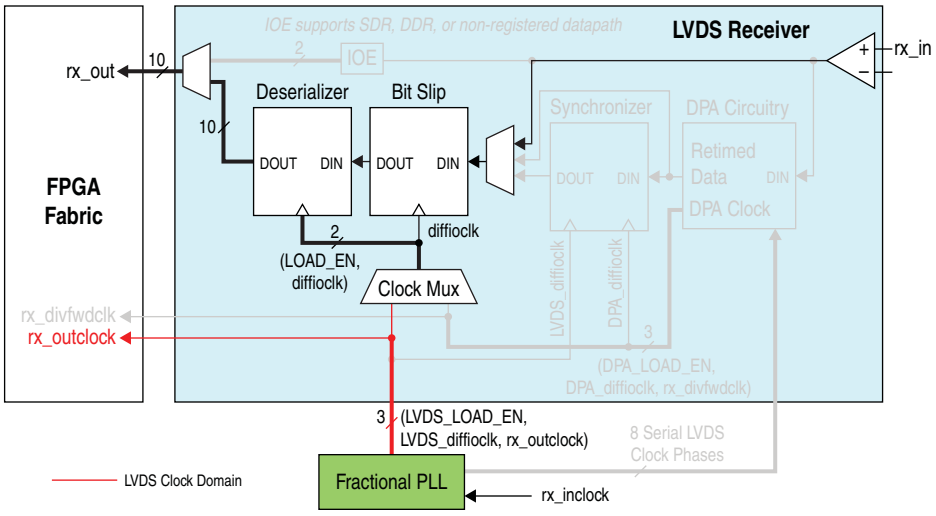
- "Non-DPA Mode" on page 6–14
- "DPA Mode" on page 6–15
- "Soft-CDR Mode" on page 6–16

Non-DPA Mode

Figure 6–15 shows the non-DPA datapath block diagram. The non-DPA mode disables the DPA and synchronizer blocks. Input serial data is registered at the rising edge of the serial `LVDS_diffioclk` clock that is produced by the left and right PLLs.

You can select the rising edge option with the Quartus II MegaWizard Plug-In Manager. The `LVDS_diffioclk` clock that is generated by the left and right PLLs clocks the data realignment and deserializer blocks.

Figure 6–15. Receiver Data Path in Non-DPA Mode *(1), (2), (3)*



Notes to Figure 6–15:

(1) All disabled blocks and signals are grayed out.

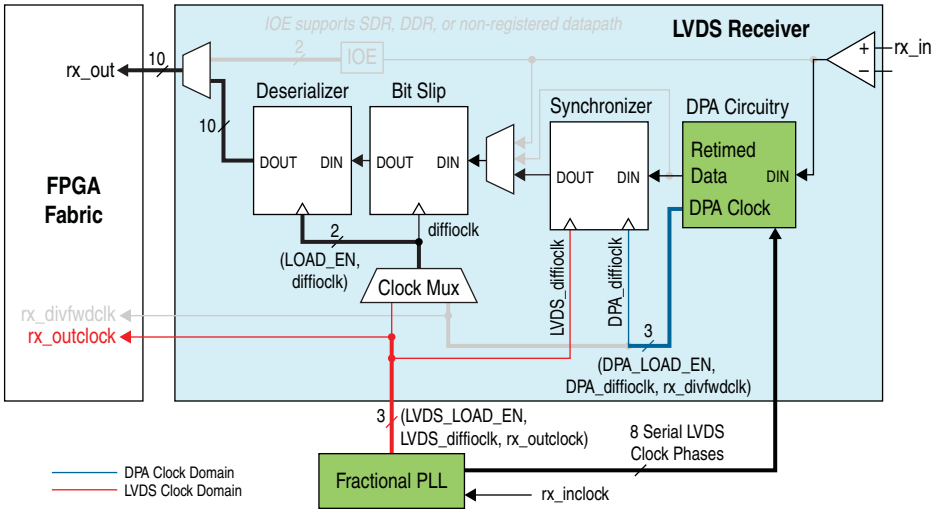
(2) In SDR and DDR modes, the data width from the IOE is 1 and 2 bits, respectively.

(3) The `rx_out` port has a maximum data width of 10 bits.

DPA Mode

Figure 6–16 shows the DPA mode datapath, where all the hardware blocks mentioned in "Receiver Blocks" on page 6–10 are active. The DPA block chooses the best possible clock (`DPA_diffioclk`) from the eight fast clocks that the fractional PLL sent. This serial `DPA_diffioclk` clock is used for writing the serial data into the synchronizer. A serial `LVDS_diffioclk` clock is used for reading the serial data from the synchronizer. The same `LVDS_diffioclk` clock is used in data realignment and deserializer blocks.

Figure 6–16. Receiver Datapath in DPA Mode *(1), (2), (3)*



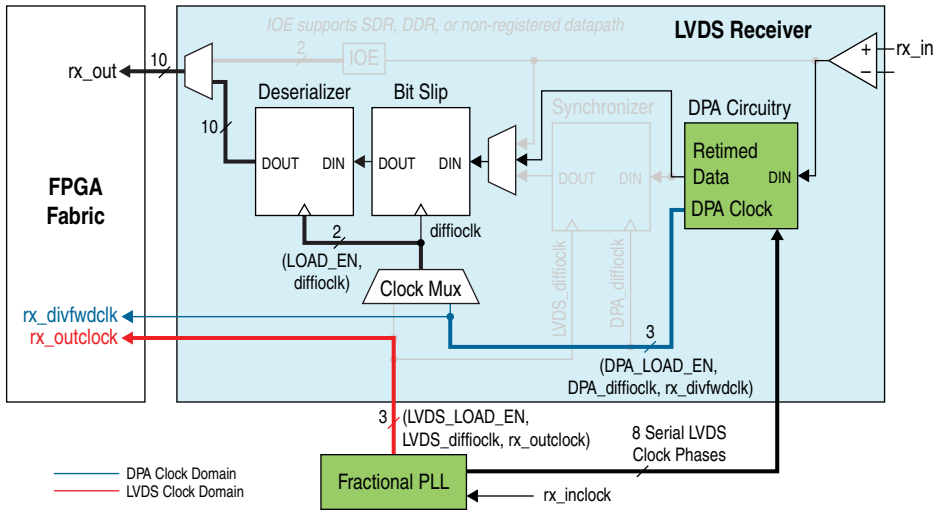
Notes to Figure 6–16:

(1) All disabled blocks and signals are grayed out.
(2) In SDR and DDR modes, the data width from the IOE is 1 and 2 bits, respectively.
(3) The `rx_out` port has a maximum data width of 10 bits.

Soft-CDR Mode

The Arria V **LVDS** channel offers the soft-CDR mode to support the GbE and SGMII protocols. A receiver PLL uses the local clock source for reference. Figure 6–17 shows the soft-CDR mode datapath.

Figure 6–17. Receiver Datapath in Soft-CDR Mode *(1), (2), (3)*



Notes to Figure 6–17:

(1) All disabled blocks and signals are grayed out.
(2) In SDR and DDR modes, the data width from the IOE is 1 and 2 bits, respectively.
(3) The `rx_out` port has a maximum data width of 10 bits.

In soft-CDR mode, the synchronizer block is inactive. The DPA circuitry selects an optimal DPA clock phase to sample the data. Use the selected DPA clock for bit-slip operation and deserialization. The DPA block also forwards the selected DPA clock, divided by the deserialization factor called `rx_divfwdclk`, to the FPGA fabric, along with the deserialized data. This clock signal is put on the periphery clock (PCLK) network.

If you use the soft-CDR mode, do not assert the `rx_reset` port after the DPA has trained. The DPA continuously chooses new phase taps from the PLL to track parts per million (PPM) differences between the reference clock and incoming data.

For more information about PCLK networks, refer to the *Clock Networks and PLLs in Arria V Devices* chapter.

You can use every **LVDS** channel in soft-CDR mode and drive the FPGA fabric using the PCLK network in the Arria V device family. The `rx_dpa_locked` signal is not valid in soft-CDR mode because the DPA continuously changes its phase to track PPM differences between the upstream transmitter and the local receiver input reference clocks. The parallel clock, `rx_outclock`, generated by the left and right PLLs, is also forwarded to the FPGA fabric.

Receiver Clocking

The fractional PLL receives the external clock input and generates different phases of the same clock. The DPA block automatically chooses one of the clocks from the fractional PLL and aligns the incoming data on each channel.

The synchronizer circuit is a 1-bit wide by 6-bit deep FIFO buffer that compensates for any phase difference between the DPA clock and the data realignment block. If necessary, the user-controlled data realignment circuitry inserts a single bit of latency in the serial bit stream to align to the word boundary.

The physical medium connecting the transmitter and receiver **LVDS** channels may introduce skew between the serial data and the source-synchronous clock. The instantaneous skew between each **LVDS** channel and the clock also varies with the jitter on the data and clock signals as seen by the receiver. The three different modes—non-DPA, DPA, and soft-CDR—provide different options to overcome skew between the source synchronous clock (non-DPA, DPA) /reference clock (soft-CDR) and the serial data.

Only the non-DPA mode requires manual skew adjustment.

Non-DPA mode allows you to statically select the optimal phase between the source synchronous clock and the received serial data to compensate skew. In DPA mode, the DPA circuitry automatically chooses the best phase to compensate for the skew between the source synchronous clock and the received serial data. Soft-CDR mode provides opportunities for synchronous and asynchronous applications for chip-to-chip and short reach board-to-board applications for SGMII protocols.

Differential I/O Termination

The Arria V device family provides a 100-Ω, on-chip differential termination option on each differential receiver channel for **LVDS** standards. On-chip termination saves board space by eliminating the need to add external resistors on the board. You can enable on-chip termination in the Quartus II software Assignment Editor.

All I/O pins and dedicated clock input pins support on-chip differential termination.

Figure 6–18 shows device on-chip termination.

Figure 6–18. On-Chip Differential I/O Termination

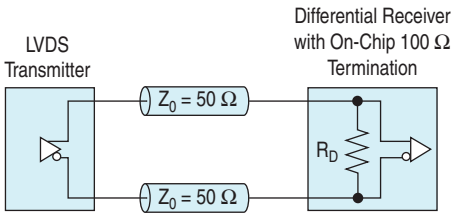


Table 6–4 lists the assignment name for on-chip differential termination in the Quartus II software Assignment Editor.

Table 6–4. Quartus II Software Assignment Editor—On-Chip Differential Termination

Field	Assignment
To	rx_in
Assignment name	Input Termination
Value	Differential

PLLs and Clocking

The Arria V device family supports fractional PLLs on each side of the device. Figure 6–1 on page 6–2 and Figure 6–2 on page 6–2 show the location of the fractional PLLs supported for the high-speed differential I/O receiver and transmitter channels to generate the parallel clocks (rx_outclock and tx_outclock) and high-speed clocks (diffioclk).

To drive the **LVDS** channels, you must use the PLLs in integer PLL mode. The center or corner PLLs can drive the **LVDS** receiver and driver channels. However, the clock tree network cannot cross over to different I/O regions. For example, the top left corner PLL cannot cross over to drive the **LVDS** receiver and driver channels on the top right I/O bank.

For more information about the fractional PLL clocking restrictions, refer to "Differential Pin Placement" on page 6–21.

The MegaWizard Plug-In Manager provides an option for implementing the **LVDS** interface with the external PLL mode. With this option enabled you can control the PLL settings, such as dynamically reconfiguring the PLL to support different data rates, dynamic phase shift, and other settings. You also must instantiate the appropriate megafunction to generate the various clock and load enable signals.

For more information about the external PLL mode, refer to "Generating Clock Signals for LVDS Interface" in the *LVDS SERDES Transmitter/Receiver (ALTLVDS_RX and ALTLVDS_TX_ Megafunction User Guide*.

Source Synchronous Timing Budget

This section describes the timing budget, waveforms, and specifications for source-synchronous signaling in the Arria V device family.

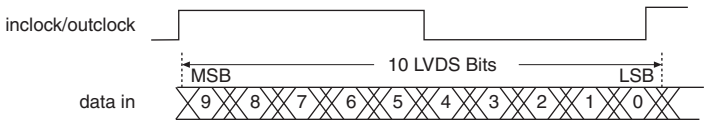
The **LVDS** I/O standards enable the transmission of data at a high speed. This high transmission rate of data results in better overall system performance. To take advantage of fast system performance, you must understand how to analyze timing for these high-speed signals. Timing analysis for the differential block is different from traditional synchronous timing analysis techniques.

The basis of the source synchronous timing analysis is the skew between the data and the clock signals instead of the clock-to-output setup times. 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 the Arria V device family, and how to use these timing parameters to determine the maximum performance of a design.

Differential Data Orientation

There is a set relationship between an external clock and the incoming data. For operations at 1 Gbps and a serialization factor of 10, the external clock is multiplied by 10. You can set phase-alignment 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 6–19 shows the data bit orientation of the x10 mode.

Figure 6–19. Bit Orientation in the Quartus II Software

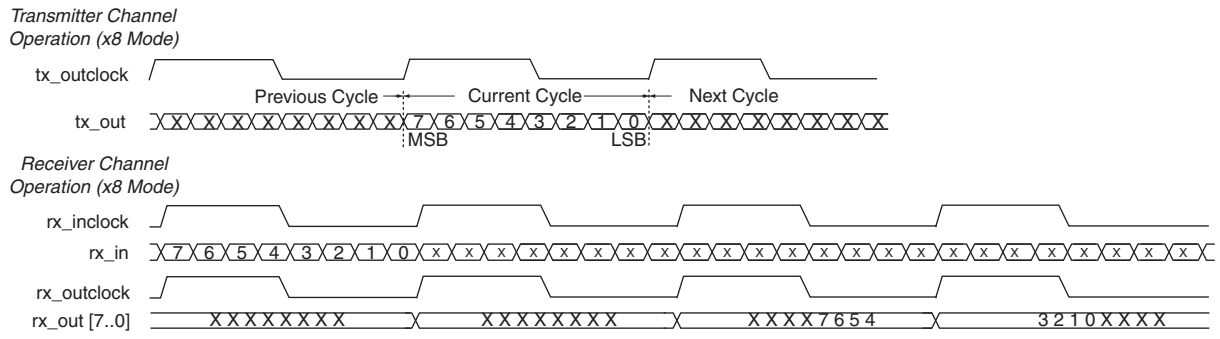


Differential I/O Bit Position

Data synchronization is necessary for successful data transmission at high frequencies. Figure 6–20 shows the data bit orientation for a channel operation and is based on the following conditions:

- The serialization factor is equal to the clock multiplication factor.
- The phase alignment uses edge alignment.
- The operation is implemented in hard SERDES.

Figure 6–20. Bit-Order and Word Boundary for One Differential Channel *(1)*



Note to Figure 6–20:

(1) These waveforms are only functional waveforms and do not convey timing information.

For other serialization factors, use the Quartus II software tools to find the bit position within the word.

Table 6–5 lists 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 6–5. Differential Bit Naming

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
12	95	88
13	103	96
14	111	104
15	119	112
16	127	120
17	135	128
18	143	136

Transmitter Channel-to-Channel Skew

The receiver skew margin calculation uses the transmitter channel-to-channel skew (TCCS)—an important parameter based on the Arria V transmitter in a source-synchronous differential interface:

- TCCS is the difference between the fastest and slowest data output transitions, including the T_{CO} variation and clock skew.
- For **LVDS** transmitters, the TimeQuest Timing Analyzer provides the TCCS value in the TCCS report (`report_TCCS`) in the Quartus II compilation report, which shows TCCS values for serial output ports.
- You can also get the TCCS value from the *Arria V Device Datasheet*.

For more information, refer to "Receiver Skew Margin for Non-DPA Mode".

Receiver Skew Margin for Non-DPA Mode

Different modes of **LVDS** receivers use different specifications, which can help in deciding the ability to sample the received serial data correctly:

- In DPA mode, use DPA jitter tolerance instead of the receiver skew margin (RSKM).
- In non-DPA LVDS mode, use RSKM, TCCS, and sampling window (SW) specifications for high-speed source-synchronous differential signals in the receiver data path.

For more information about the RSKM equation and calculation, refer to "Receiver Skew Margin for Non-DPA Mode" in the *LVDS SERDES Transmitter/Receiver (ALTLVDS_RX and ALTLVDS_TX_ Megafunction User Guide*.

For **LVDS** receivers, the Quartus II software provides an RSKM report showing the SW, TUI, and RSKM values for non-DPA LVDS mode:

- You can generate the RSKM report by executing the `report_RSKM` command in the TimeQuest Timing Analyzer. You can find the RSKM report in the Quartus II compilation report in the TimeQuest Timing Analyzer section.
- To obtain the RSKM value, assign the input delay to the **LVDS** receiver through the constraints menu of the TimeQuest Timing Analyzer. The input delay is determined according to the data arrival time at the **LVDS** receiver port, with respect to the reference clock.
- If you set the input delay in the settings parameters for the **Set Input Delay** option, set the clock name to the clock that reference the source synchronous clock that feeds the **LVDS** receiver.
- If you do not set any input delay in the TimeQuest Timing Analyzer, the receiver channel-to-channel skew (RCCS) defaults to zero.
- You can also directly set the input delay in a Synopsys Design Constraint file (**.sdc)** using the `set_input_delay` command.

For more information about **.sdc** commands and the TimeQuest Timing Analyzer, refer to the *Quartus II TimeQuest Timing Analyzer* chapter in volume 3 of the *Quartus II Development Software Handbook*.

Design Considerations

To ensure the success of your high-speed differential I/O designs, take into consideration the guidelines described in this section.

Differential Pin Placement

This section describes the pin placement guidelines with and without DPA usage.

DPA usage adds some constraints on the placement of high-speed differential channels. If DPA-enabled or DPA-disabled differential channels in the differential banks are used, you must adhere to the differential pin placement guidelines to ensure the proper high-speed operation. The Quartus II compiler automatically checks the design and issues an error message if the guidelines are not followed.

DPA-enabled differential channels refer to DPA mode or soft-CDR mode; DPA disabled channels refer to non-DPA mode.

DPA-Enabled Channels, DPA-Disabled Channels, and Single-Ended I/Os

When you enable a DPA channel in a bank, you can use both single-ended I/Os and differential I/O standards in the bank.

You can place double data rate I/O (DDIO) output pins within I/O modules that have the same pad group number as a SERDES differential channel. However, you cannot place SDR I/O output pins within I/O modules that have the same pad group number as a receiver SERDES differential channel. You must implement the input register within the FPGA fabric logic.

Guidelines for DPA-Enabled Differential Channels

The Arria V device family has differential receivers and transmitters in all I/O blocks. Each receiver has a dedicated DPA circuit to align the phase of the clock to the data phase of its associated channel. When you use DPA-enabled channels in differential banks, you must adhere to the guidelines listed in the following sections.

DPA-Enabled Channel Driving Distance

If the number of DPA-enabled channels driven by each center or corner PLL exceeds 25 LAB rows, Altera recommends implementing data realignment (bit slip) circuitry for all the DPA channels.

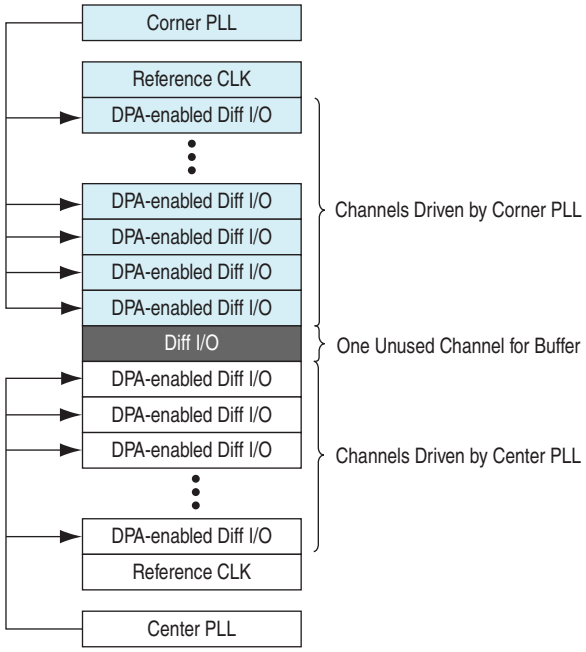
Using Center and Corner PLLs

If two PLLs drive the DPA-enabled channels in a bank—the corner and center PLL drive one group each—there must be at least one row of separation between the two groups of DPA-enabled channels, as shown in Figure 6–21. This separation prevents noise mixing because the two groups can operate at independent frequencies. No separation is necessary if a single PLL is driving both the DPA-enabled channels and DPA-disabled channels.

Figure 6–21 to Figure 6–27 show guidelines for usage of corner and center PLLs in the DPA or non-DPA LVDS channels in the high-speed **LVDS** I/O banks. These figures do not represent the exact locations of the high-speed **LVDS** I/O banks.

For information about high-speed **LVDS** I/O banks locations, refer to "SERDES and DPA Bank Locations" on page 6–2.

Figure 6–21. Center and Corner PLLs Driving DPA-enabled Differential I/Os in the Same Bank

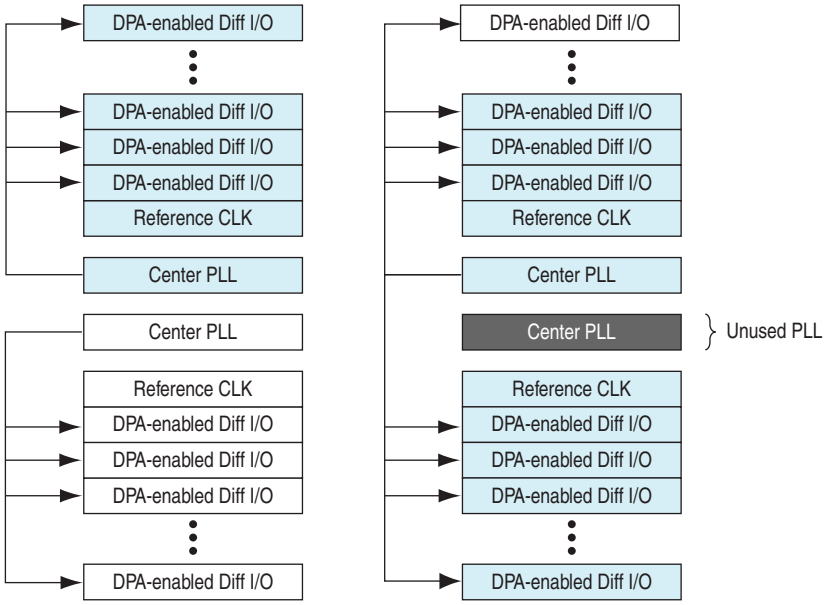


Using Both Center PLLs

You can use center PLLs to drive DPA-enabled channels simultaneously, if they drive these channels in their adjacent banks only, as shown in Figure 6–21.

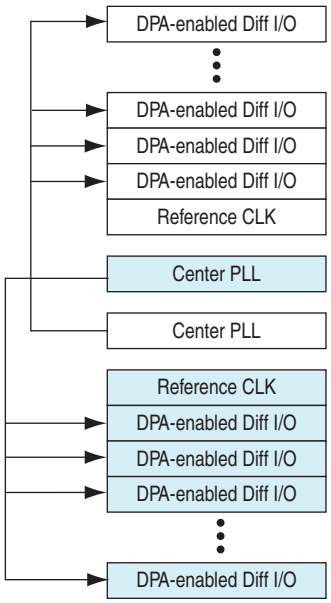
If one of the center PLLs drives the DPA-enabled channels in the upper and lower I/O banks, you cannot use the other center PLL for DPA-enabled channels, as shown in Figure 6–22.

Figure 6–22. Center PLLs Driving DPA-enabled Differential I/Os



If the upper center PLL drives the DPA-enabled channels in the lower I/O bank, the lower center PLL cannot drive the DPA-enabled channels in the upper I/O bank, and vice versa. The center PLLs cannot drive cross-banks simultaneously, as shown in Figure 6–23.

Figure 6–23. Invalid Placement of DPA-enabled Differential I/Os Driven by Both Center PLLs



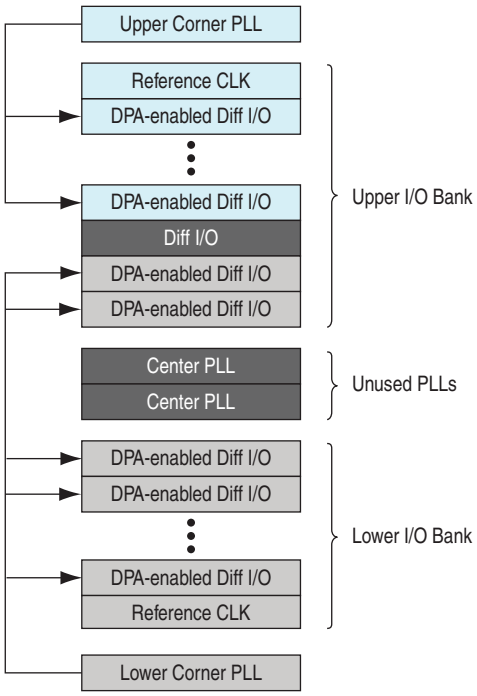
Using Both Corner PLLs

You can use both corner PLLs to drive DPA-enabled channels simultaneously, if they drive the channels in their adjacent banks only. There must be at least one row of separation between the two groups of DPA-enabled channels.

If one of the corner PLLs drives DPA-enabled channels in the upper and lower I/O banks, you cannot use the center PLLs. You can use the other corner PLL to drive DPA-enabled channels in their adjacent bank only. There must be at least one row of separation between the two groups of DPA-enabled channels.

If the upper corner PLL drives DPA-enabled channels in the lower I/O bank, the lower corner PLL cannot drive DPA-enabled channels in the upper I/O bank, and vice versa. In other words, the corner PLLs cannot drive cross-banks simultaneously, as shown in Figure 6–24.

Figure 6–24. Corner PLLs Driving DPA-enabled Differential I/Os



Guidelines for DPA-Disabled Differential Channels

When you use DPA-disabled channels, adhere to the guidelines in the following sections.

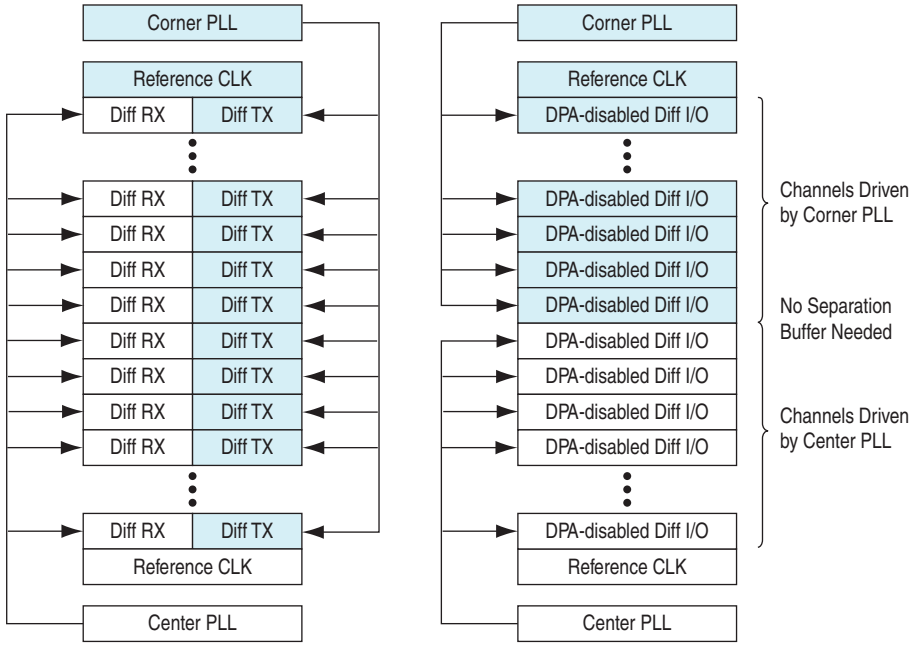
DPA-Disabled Channel Driving Distance

Each PLL can drive all the DPA-disabled channels in the entire bank.

Using Corner and Center PLLs

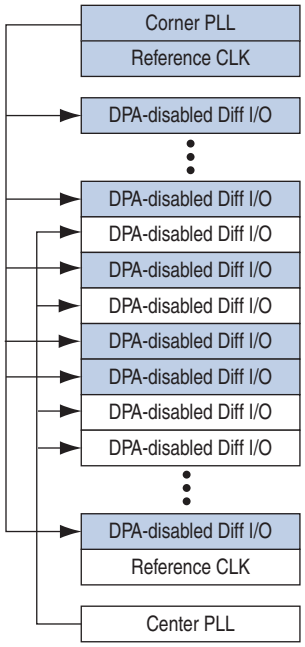
You can use a corner PLL to drive all transmitter channels and a center PLL to drive all DPA-disabled receiver channels in the same I/O bank. You can drive a transmitter channel and a receiver channel in the same LAB row by two different PLLs, as shown in Figure 6–25.

Figure 6–25. Corner and Center PLLs Driving DPA-Disabled Differential I/Os in the Same Bank



A corner PLL and a center PLL can drive duplex channels in the same I/O bank if the channels that are driven by each PLL are not interleaved. You do not require separation between the group of channels that are driven by the corner and center, left and right PLLs. Refer to Figure 6–25 and Figure 6–26.

Figure 6–26. Invalid Placement of DPA-disabled Differential I/Os Due to Interleaving of Channels Driven by the Corner and Center PLLs

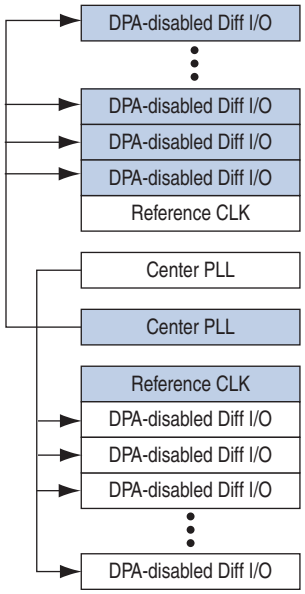


Using Both Center PLLs

You can use both center PLLs simultaneously to drive DPA-disabled channels on upper and lower I/O banks. Unlike DPA-enabled channels, the center PLLs can drive DPA-disabled channels cross-banks. For example, the upper center PLL can drive the lower I/O bank at the same time the lower center PLL is driving the upper I/O bank, and vice versa, as shown in Figure 6–27.

For information about the PLL locations available for each Arria V device, refer to "SERDES and DPA Bank Locations" on page 6–2.

Figure 6–27. Both Center PLLs Driving Cross-Bank DPA-Disabled Channels Simultaneously



Using Both Corner PLLs

You can use both corner PLLs to drive DPA-disabled channels simultaneously. You can use a corner PLL to drive all the transmitter channels and the other corner PLL to drive all the DPA-disabled receiver channels in the same I/O bank. Both corner PLLs can drive duplex channels in the same I/O bank if the channels that are driven by each PLL are not interleaved. You do not require separation between the groups of channels that are driven by both corner PLLs.

Document Revision History

Table 6–6 lists the revision history for this chapter.

Table 6–6. Document Revision History

Date	Version	Changes
June 2012	2.0	■ Restructured the chapter. ■ Updated Table 6–1. ■ Updated Figure 6–1 and Figure 6–2. ■ Added Figure 6–3. ■ Added "Design Considerations" section. ■ Updated the "Differential Pin Placement" section.
November 2011	1.1	■ Updated Table 6–1. ■ Restructured chapter.
May 2011	1.0	Initial release.

7. External Memory Interfaces in Arria V Devices

AV52007-2.0

This chapter describes external memory interfaces available with Arria® V devices, as well as the silicon capabilities of the devices to support external memory interfaces.

The following Arria V device features are used in external memory interfaces:

- Double data rate 2 (DDR2) SDRAM, DDR3 SDRAM, and low power double data rate 2 (LPDDR2) SDRAM interface support
- Quad data rate (QDR) II+ and QDR II SRAM interface support
- DQS phase-shift circuitry
- PHY Clock (PHYCLK) networks
- DQS logic block
- Dynamic on-chip termination (OCT) control
- I/O element (IOE) registers
- Delay chain
- Hard memory controllers

This chapter contains the following sections:

For more information about board design guidelines, timing analysis, simulation, and debugging information, refer to the *External Memory Interface Handbook*.

For more information about using Altera's External Memory Interface Spec Estimator tool to estimate the external memory system performance specifications, refer to the External Memory Interface Spec Estimator page of the Altera website.

Memory Interface Pin Support

This section describes the Arria V I/O pins that can be used for external memory interface. This section also provides the number of DQ groups available on each device side for all Arria V devices.

For more information about the types of memory supported by Arria V devices, refer to the *Selecting Your Memory* chapter of the *External Memory Interface Handbook*.

Arria V devices offer differential input buffers for differential read-data strobe and clock operations. In the Arria V pin tables, the DQS and DQSn pins denote the differential data strobe/clock pin pairs, while the CQ and CQn pins denote the complementary echo clock signals.

For QDR, use a single CQ pin to clock the rising and falling edges of the data, instead of two separate CQ and CQ# pins. The falling edges are used as CQ# pin replacements.

The maximum number of data pins per group listed in Table 7–1 may vary according to the following conditions:

- Single-ended DQS signaling—the maximum number of DQ pins includes parity, data mask, and QVLD pins connected to the DQS bus network.
- Differential or complementary DQS signaling—the maximum number of data pins per group decreases by one.
- DDR3 and DDR2 interfaces—the maximum number of pins is further reduced for an interface larger than x8 because you require one DQS pin for each x8/x9 group to form the x16/x18 and x32/x36 groups.

For the maximum number of DQ pins and the exact number per group for a particular Arria V device, refer to the pin table in the Arria V Device Pin-Out Files page of the Altera website.

The pin table lists the parity, DM, BWSn, NWSn, ECC, and QVLD pins as DQ pins.

Table 7–1 lists pin support per DQ/DQS bus mode, including the DQS/CQ/CQn/QK# and DQSn pins.

Table 7–1. DQ/DQS Bus Mode Pins for Arria V Devices

Mode	DQSn Support	Parity or Data Mask (Optional)	QVLD (Optional) (1)	Typical Number of Data Pins per Group	Maximum Number of Data Pins per Group
x4/x8/x9 (2)	Yes	Yes	Yes	4, 8, or 9	11
x16/x18 (3)	Yes	Yes	Yes	16 or 18	23
x32/x36 (4)	Yes	Yes	Yes	32 or 36	47

Notes to Table 7–1:

(1) The QVLD pin is not used in the UniPHY megafunction.
(2) The x4 mode uses x8/x9 groups in Arria V devices.
(3) Two x8 DQ/DQS groups are stitched to create a x16/x18 group, so there are a total of 24 pins in this group.
(4) Four x8 DQ/DQS groups are stitched to create a x32/x36 group, so there are a total of 48 pins in this group.

Table 7–2 lists the number of DQ/DQS groups available per side in each Arria V device.

Table 7–2. Number of DQ/DQS Groups in Arria V Devices per Side (1) (Part 1 of 3)

Variant	Member Code	Package	Side	x8/x9	x16/x18	x32/x36
Arria V GX	A1 A3	672-pin FineLine BGA, Flip Chip	Right	4	2	0
			Top/Bottom	8	3	0
	A1 A3	896-pin FineLine BGA, Flip Chip	Right	6	2	0
			Top/Bottom	12	6	2
	A5 A7	672-pin FineLine BGA, Flip Chip	Right	0	0	0
			Top/Bottom	8	3	1
	A5 A7 B1 B3	896-pin FineLine BGA, Flip Chip	Right	0	0	0
			Top/Bottom	12	5	1
	A5 A7 B1 B3	1152-pin FineLine BGA, Flip Chip	Right	0	0	0
			Top/Bottom	17	8	2
	B1 B3	1517-pin FineLine BGA, Flip Chip	Right	0	0	0
			Top/Bottom	22	10	4
	B5 B7	1152-pin FineLine BGA, Flip Chip	Right	0	0	0
			Top/Bottom	17	8	2
	B5 B7	1517-pin FineLine BGA, Flip Chip	Right	0	0	0
			Top/Bottom	22	10	4

Table 7–2. Number of DQ/DQS Groups in Arria V Devices per Side (1) (Part 2 of 3)

Variant	Member Code	Package	Side	x8/x9	x16/x18	x32/x36
Arria V GT	D3	896-pin FineLine BGA, Flip Chip	Right	0	0	0
			Top/Bottom	12	5	1
	D3	1152-pin FineLine BGA, Flip Chip	Right	0	0	0
			Top/Bottom	17	8	2
	D3	1517-pin FineLine BGA, Flip Chip	Right	0	0	0
			Top/Bottom	22	10	4
	D7	1152-pin FineLine BGA, Flip Chip	Right	0	0	0
			Top/Bottom	17	8	2
	D7	1517-pin FineLine BGA, Flip Chip	Right	0	0	0
			Top/Bottom	22	10	4
Arria V SX	B3 B5	896-pin FineLine BGA, Flip Chip	Right	0	0	0
			Top/Bottom	8	3	1
	B3 B5	1152-pin FineLine BGA, Flip Chip	Right	0	0	0
			Top	8	3	1
			Bottom	18	8	3
	B3 B5	1517-pin FineLine BGA, Flip Chip	Right	0	0	0
			Top	12	5	2
			Bottom	22	10	4

Table 7–2. Number of DQ/DQS Groups in Arria V Devices per Side (1) (Part 3 of 3)

Variant	Member Code	Package	Side	x8/x9	x16/x18	x32/x36
Arria V ST	D3 D5	896-pin FineLine BGA, Flip Chip	Right	0	0	0
			Top/Bottom	8	3	1
	D3 D5	1152-pin FineLine BGA, Flip Chip	Right	0	0	0
			Top	8	3	1
			Bottom	18	8	3
	D3 D5	1517-pin FineLine BGA, Flip Chip	Right	0	0	0
			Top	12	5	2
			Bottom	22	10	4

Note to Table 7–2:

(1) These numbers are preliminary until the devices are available.

For more information about which pins to use for memory clock pins and pin location requirements, refer to the *Planning Pin and FPGA Resources* chapter of the External Memory Interface Handbook.

Design Considerations

The following sections provide considerations that require your attention to ensure the success of your designs.

Memory Interface

Memory interface circuitry is available in every I/O bank that does not support transceivers.

Memory clock pins in Arria V devices are generated with double data rate input/output (DDRIO) registers.

Delay-Locked Loop

The delay-locked loop (DLL) phase comparator requires 2,560 clock cycles to lock and calculate the correct input clock period.

You can reset the DLL from either the logic array or a user I/O pin. Each time the DLL is reset, you must wait for 2,560 clock cycles for the DLL to lock before you can capture the data properly.

The DLL can shift the incoming DQS signals by 0° or 90° by using two delay cells in the DQS logic block. The shifted DQS signal is then used as the clock for the DQ IOE input registers.

DQ/DQS Pins

Arria V devices support DQ and DQS signals with DQ bus modes of x4/x8/x9, x16/x18, or x32/x36.

You can use the DQSn or CQn pins that are not used for clocking as DQ pins.

If you do not use the DQ/DQS pins for memory interfacing, you can use these pins as user I/Os.

However, unused system-on-a-chip (SOC) DQ/DQS pins cannot be used as user I/Os.

Using the RZQ Pins in a DQ/DQS Group for Memory Interfaces

You can use some of the DQ pins as RZQ pins. However, when you use these pins as RZQ pins, you cannot use them as DQ pins in an external memory interface.

You must manually assign DQ and DQS pins for x8, x16/x18, or x32/x36 DQ/DQS groups whose members are used as RZQ pins. The Quartus® II software might not be able to place DQ and DQS pins without manual pin assignments, which results in a "no-fit" error.

PHYCLK Networks

The pin placement guidelines for the PHYCLK networks are as follows:

- Two interfaces can share an I/O sub-bank (for example, sub-bank 4A) for pin placement if the two interfaces share a PLL. These two interfaces must use the same memory protocol (for example, DDR3), frequency, controller rate (for example, half rate), and phase requirements (for example, additional core-to-periphery clock phase of 90°).
- Two interfaces that do not share a PLL cannot share a sub-bank for pin placement.
- Two interfaces can share an I/O bank (for example, I/O bank 4) for pin placement regardless of whether they share a PLL or not.
- PHYCLK networks support interface at the same side of the I/O banks only.
- PHYCLK networks do not support split interface, where some pins of a memory interface are placed at the top I/O banks and some pins at the bottom I/O banks.
- For better performance, you can use the center PLL for your memory interface or you can place all pins for a memory interface in an I/O bank and use the corner PLL adjacent to that I/O bank for the memory interface.
- The memory interface pins include data, data strobe, data mask, address, command, control, and clock pins.
- To drive the external memory interfaces, you must use the PLLs in integer PLL mode.

DDR2 SDRAM Interface

Altera recommends using differential DQS signaling for DDR2 SDRAM interfaces running at 333 MHz and higher.

DDR3 SDRAM DIMM

Arria V devices do not support DDR3 SDRAM with read or write leveling for standard DDR3 SDRAM DIMMs and DDR3 SDRAM components using the standard DDR3 SDRAM fly-by address, command, and clock layout topology.

Hard Memory Controller

Bonding

Only one bonding feature is available per package, through the core fabric.

A memory interface that uses the bonding feature has higher average latency.

DDR2 and DDR3 Address Line

16th address bit is not apply to all banks for certain Arria V devices.

External Memory Interface Features

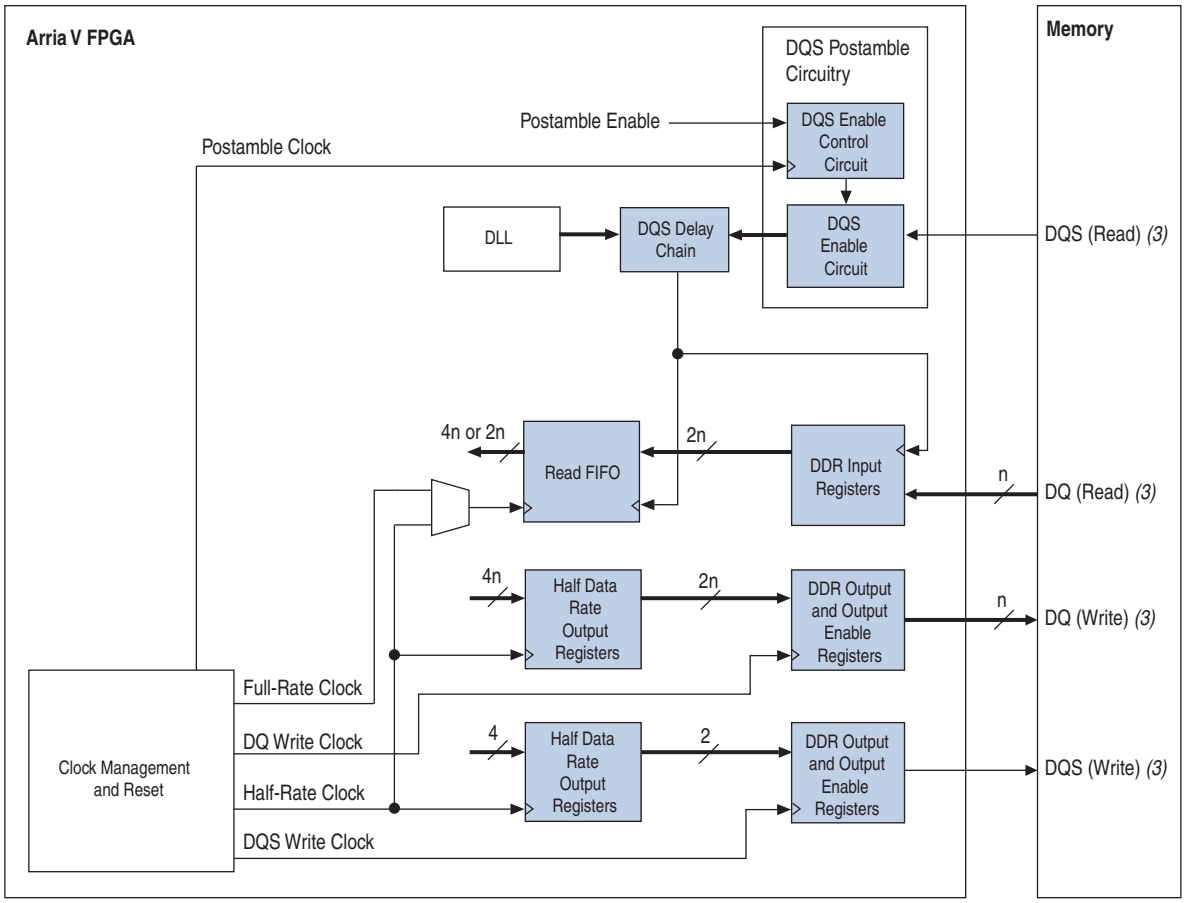
Arria V devices have features that allow robust high-performance external memory interfacing.

If you use the Altera memory controller MegaCore® functions, the UniPHY megafunction is instantiated to help you set up the physical interface (PHY) best suited for your system.

For more information about the UniPHY megafunction, refer to the *Reference Material* volume of the *External Memory Interface Handbook*.

Figure 7–1 shows an overview of the memory interface datapath that uses the Arria V IOE features.

Figure 7–1. External Memory Interface Datapath Overview for Arria V Devices *(1), (2)*



Notes to Figure 7–1:

(1) You can bypass each register block.

(2) The blocks for each memory interface may differ slightly. The shaded blocks are part of the Arria V IOE.

(3) These signals may be bidirectional or unidirectional, depending on the memory standard. When bidirectional, the signal is active during both read and write operations.

DQS Phase-Shift Circuitry

The Arria V DLL provides phase shift to the DQS/CQ/CQn/QK# pins on read transactions if the DQS/CQ/CQn/QK# pins are acting as input clocks or strobes to the FPGA.

Figure 7–2, Figure 7–3, and Figure 7–4 show how the DLLs are connected to the DQS/CQ/CQn/QK# pins in the device, where memory interfaces are supported on the right side of the Arria V device.

Figure 7–2. DQS/CQ/CQn/QK# Pins and DLLs in Arria V GX A1 and A3 Devices

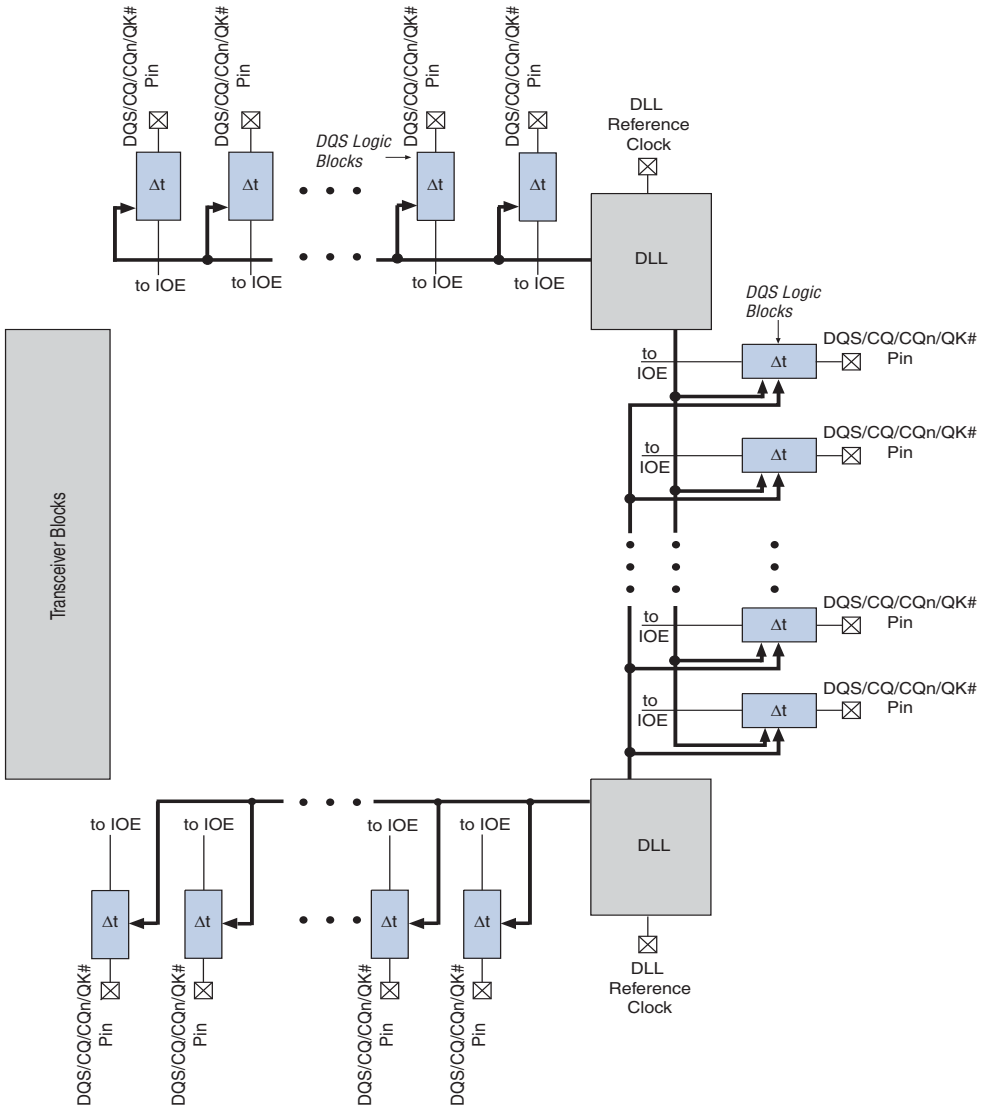


Figure 7–3. DQS/CQ/CQn/QK# Pins and DLLs in Arria V GX B1, B3, A5, A7, B5, and B7 devices, and Arria V GT D3 and D7 Devices

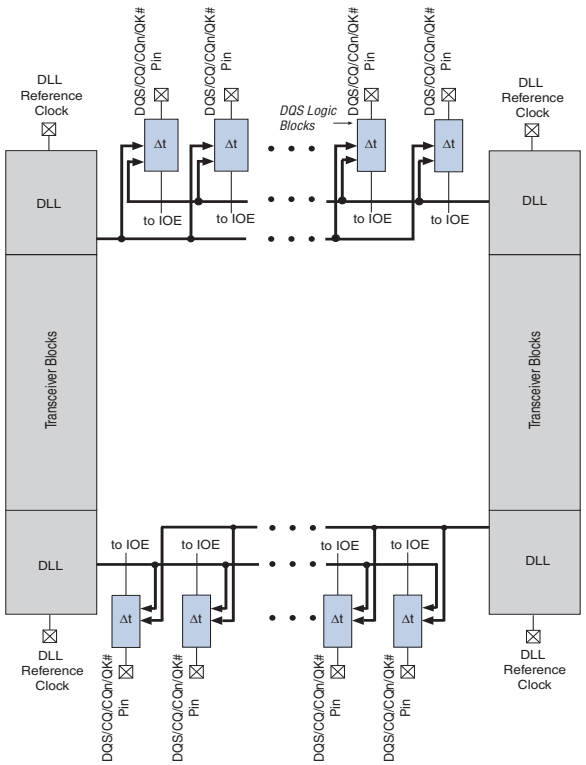
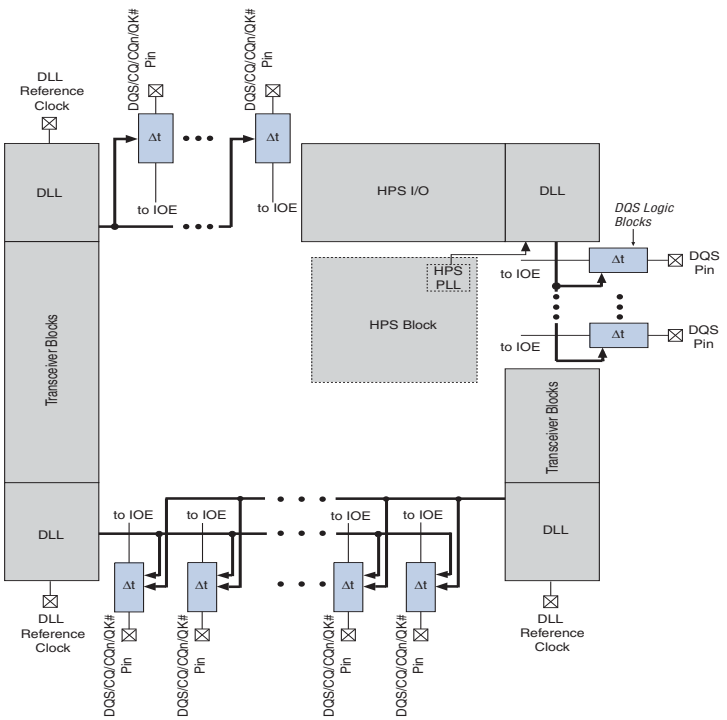


Figure 7–4. DQS/CQ/CQn/QK# Pins and DLLs in Arria V SX B3 and B5 Devices, and Arria V ST D3 and D5 Devices–*Preliminary*



Delay-Locked Loop

The DLL uses a frequency reference to dynamically generate control signals for the delay chains in each of the DQS/CQ/CQn/QK# pins, allowing the delay to compensate for process, voltage, and temperature (PVT) variations. The DQS delay settings are Gray-coded to reduce jitter if the DLL updates the settings.

There are a maximum of four DLLs, located in each corner of the Arria V devices. These four DLLs support a maximum of four unique frequencies, with each DLL running at one frequency.

The DLLs can access the two adjacent sides from its location in the device. You can have two different interfaces with the same frequency on the two sides adjacent to a DLL, where the DLL controls the DQS delay settings for both interfaces.

I/O banks between two DLLs have the flexibility to create multiple frequencies and multiple-type interfaces. These banks can use settings from either or both adjacent DLLs.

For example, `DQS1R` can get its phase-shift settings from `DLL_TR`, while `DQS2R` can get its phase-shift settings from `DLL_BR`. The reference clock for each DLL may come from the PLL output clocks or clock input pins.

DLL Phase-Shift

All DQS/CQ/CQn/QK# pins, referenced to the same DLL, can have their input signal phase shifted by a different degree amount but all must be referenced at one particular frequency.

The input reference clock goes into the DLL to a chain of up to eight delay elements. The phase comparator compares the signal coming out of the end of the delay chain block to the input reference clock. The phase comparator then issues the `upndn` signal to the Gray-code counter. This signal increments or decrements a 7-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.

The 7-bit DQS delay settings from the DLL vary with PVT to implement the phase-shift delay.

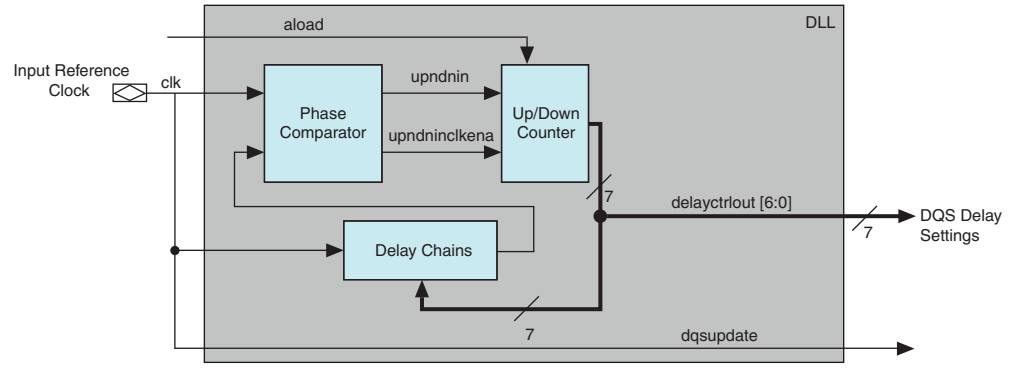
For example, with a 0° shift, the DQS/CQ/CQn/QK# signal bypasses both the DLL and DQS logic blocks. The Quartus II software automatically sets the DQ input delay chains, so that the skew between the DQ and DQS/CQ/CQn/QK# pins at the DQ IOE registers is negligible if a 0° shift is implemented. You can feed the DQS delay settings to the DQS logic block and logic array.

For SoC devices, you can feed the hard processor system (HPS) DQS delay settings to the HPS DQS logic block only.

The shifted DQS/CQ/CQn/QK# signal goes to the DQS bus to clock the IOE input registers of the DQ pins. The signal can also go into the logic array for resynchronization if you are not using IOE read FIFO for resynchronization.

Figure 7–5 shows a simple block diagram of the DLL.

Figure 7–5. Simplified Diagram of the DLL



PHY Clock (PHYCLK) Networks

The PHYCLK network is a dedicated high-speed, low-skew balanced clock tree designed for a high-performance external memory interface.

The top and bottom sides of the Arria V devices have up to four PHYCLK networks. There are up to two PHYCLK networks on the left and right side I/O banks. Each PHYCLK network spans across one I/O bank and is driven by one of the PLLs located adjacent to the I/O bank.

Figure 7–6, Figure 7–7, and Figure 7–8 show the PHYCLK networks available in the Arria V devices.

Figure 7–6. PHYCLK Networks in Arria V GX A1 and A3 Device

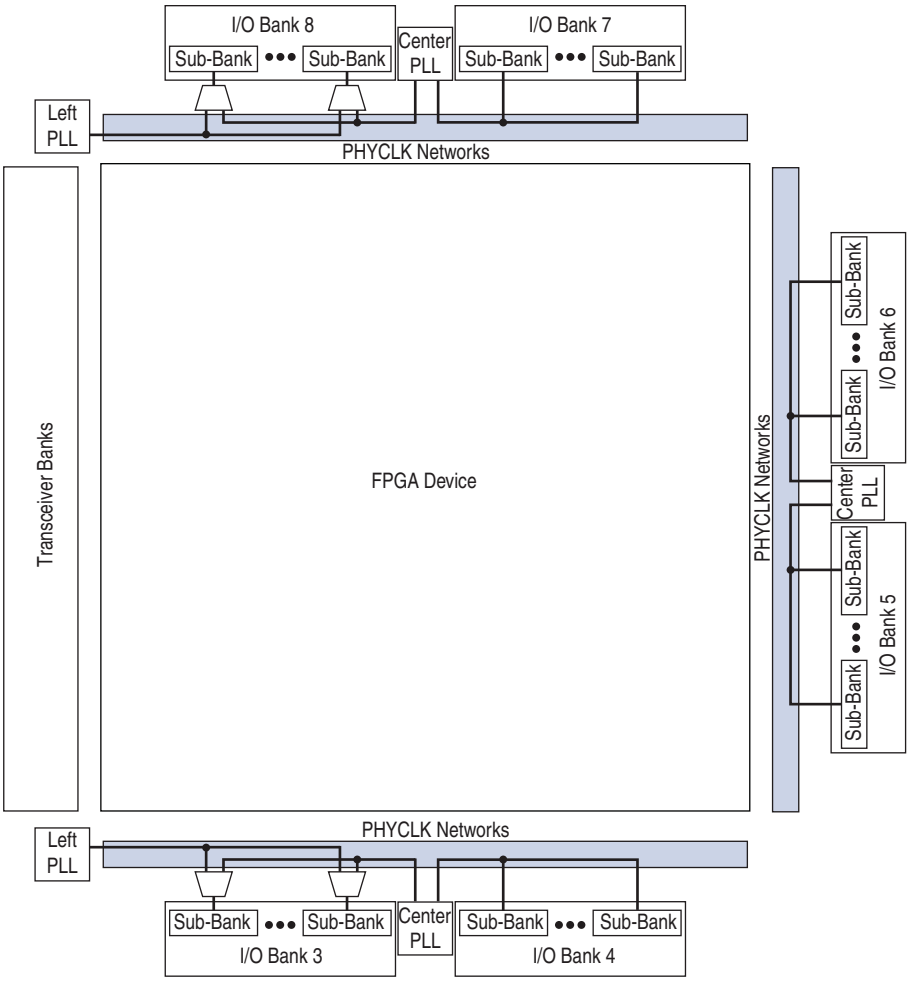


Figure 7–7. PHYCLK Networks in Arria V GX B1, B3, A5, A7, B5, and B7 Devices

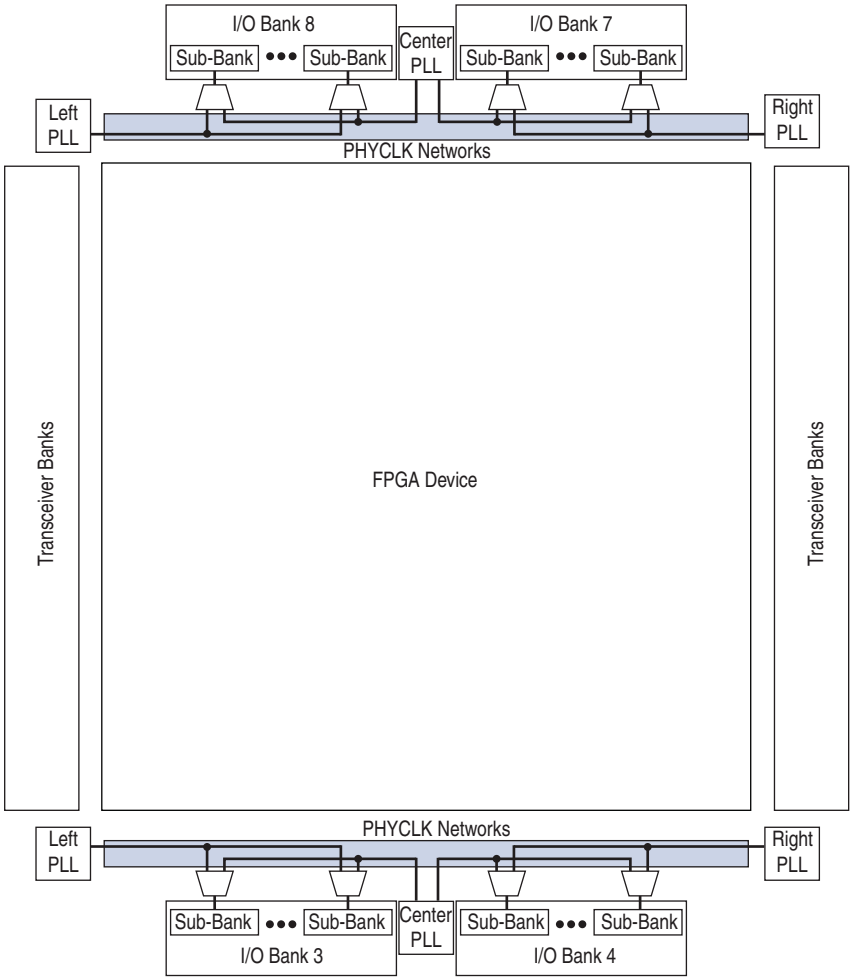
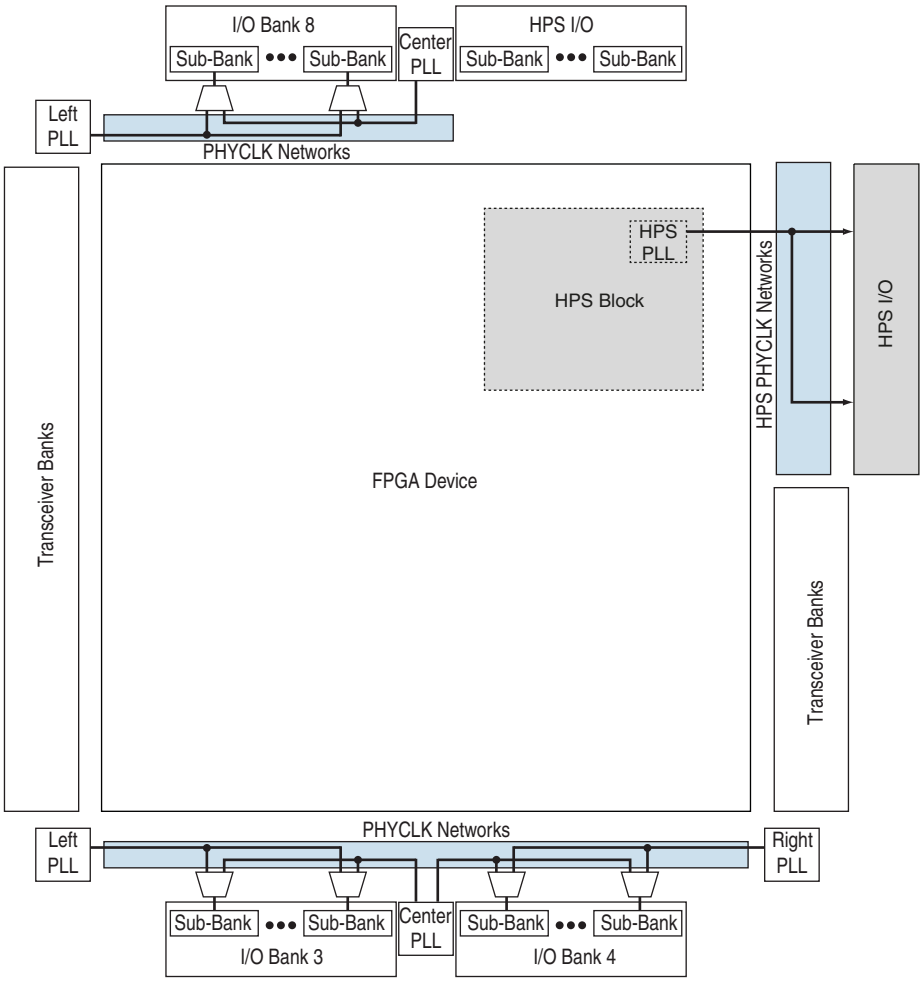


Figure 7–8. PHYCLK Networks in Arria V SX B3 and B5 Devices, and Arria V ST D3 and D5 Devices

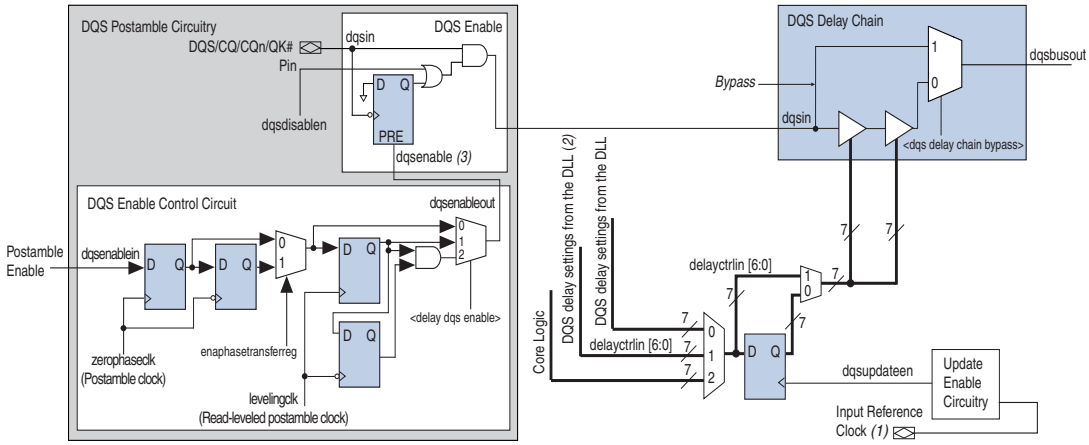


DQS Logic Block

Each DQS/CQ/CQn/QK# pin is connected to a separate DQS logic block, which consists of the update enable circuitry, DQS delay chains, and DQS postamble circuitry.

Figure 7–9 shows the DQS logic block.

Figure 7–9. DQS Logic Block in Arria V Devices



Notes to Figure 7–9:

(1) The input reference clock for the DQS phase-shift circuitry can come from a PLL output clock or an input clock pin.
(2) Only applicable if the DQS delay settings come from a side with two DLLs.
(3) The `dqsenable` signal can also come from the Arria V FPGA fabric.

Update Enable Circuitry

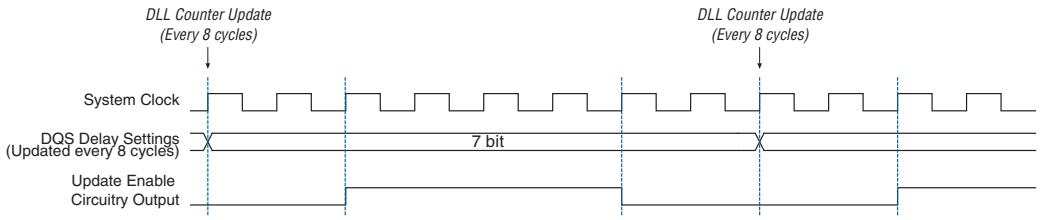
The update enable circuitry enables the registers to allow enough time for the DQS delay settings to travel from the DQS phase-shift circuitry or core logic to all the DQS logic blocks before the next change.

Both the DQS delay settings and the phase-offset settings pass through a register before going into the DQS delay chains. The registers are controlled by the update enable circuitry to allow enough time for any changes in the DQS delay setting bits to arrive at all the delay elements, which allows them to be adjusted at the same time.

The circuitry uses the input reference clock or a user clock from the core to generate the update enable output. The UniPHY intellectual property (IP) uses this circuit by default.

Figure 7–10 shows an example waveform of the update enable circuitry output.

Figure 7–10. DQS Update Enable Waveform



DQS Delay Chain

The DQS/CQ/CQn/QK# pin is shifted by the DQS delay settings.

DQS delay chains consist of a set of variable delay elements to allow the input DQS/CQ/CQn/QK# signals to be shifted by the amount specified by the DQS phase-shift circuitry or the logic array.

The SoC DQS delay chain is controlled by the DQS phase-shift circuitry only.

There are two delay elements in the DQS delay chain that have the same characteristics:

- delay elements in the DQS logic block
- delay elements in the DLL

The number of delay chains required is transparent because the UniPHY IP automatically sets it when you choose the operating frequency.

For non-SoC devices, when you do not use the DLL to control the DQS delay chains, you can input your own Gray-coded 7-bit settings using the `delayctrlin[6..0]` signals available in the UniPHY IP.

DQS Postamble Circuitry

There are preamble and postamble specifications for both read and write operations in DDR3 and DDR2 SDRAM. The DQS postamble circuitry ensures that data is not lost if there is noise on the DQS line during the end of a read operation that occurs while DQS is in a postamble state.

Arria V devices have dedicated postamble registers that you can control to ground the shifted DQS signal that is used to clock the DQ input registers at the end of a read operation.

This function ensures that any glitches on the DQS input signal during the end of a read operation and occurring while DQS is in a postamble state do not affect the DQ IOE registers.

For preamble state, the DQS is low, just after a high-impedance state.

For postamble state, the DQS is low, just before it returns to a high-impedance state.

For external memory interfaces that use a bidirectional read strobe (DDR3 and DDR2 SDRAM), the DQS signal is low before going to or coming from a high-impedance state.

HDR Block

Arria V devices have a half data rate (HDR) block in the postamble enable circuitry.

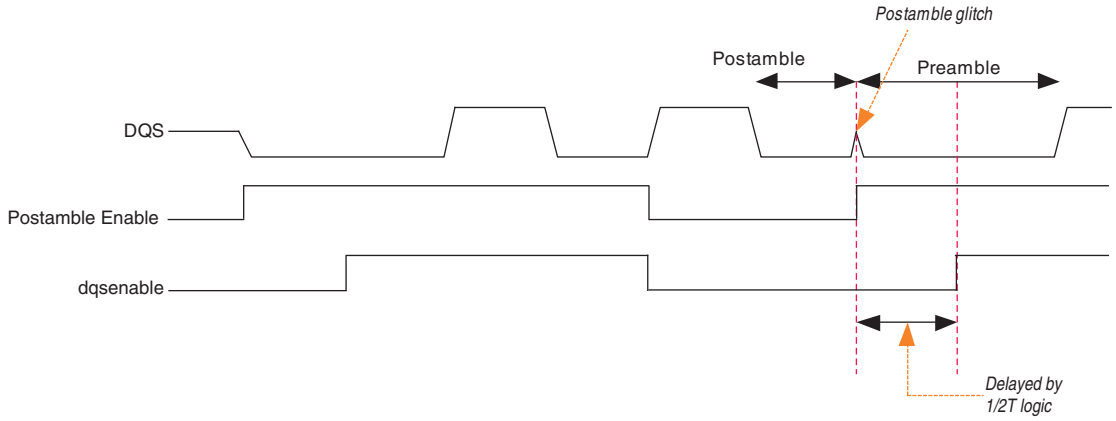
This scheme allows half-a-clock cycle latency for `dqsenable` assertion and zero latency for `dqsenable` deassertion.

The HDR block is clocked by the half-rate resynchronization clock, which is the output of the I/O clock divider circuit. There is an AND gate after the postamble register outputs to avoid postamble glitches from a previous read burst on a non-consecutive read burst.

Using the HDR block as the first stage capture register in the postamble enable circuitry block is optional. Altera recommends using these registers if the controller is running at half the frequency of the I/Os.

Figure 7–11 shows how to avoid postamble glitches using the HDR block.

Figure 7–11. Avoiding Glitch on a Non-Consecutive Read Burst Waveform

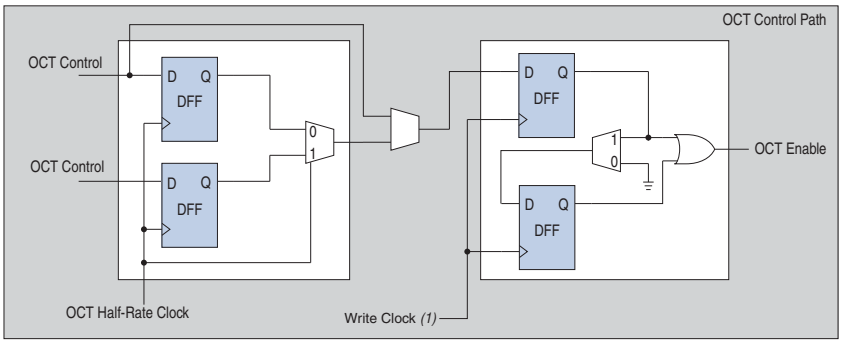


Dynamic OCT Control

The dynamic OCT control block includes all the registers that are required to dynamically turn the on-chip parallel termination (R_T OCT) on during a read and turn R_T OCT off during a write.

Figure 7–12 shows the dynamic OCT control block.

Figure 7–12. Dynamic OCT Control Block for Arria V Devices



Note to Figure 7–12:

(1) The full-rate write clock comes from the PLL. The DQ write clock and DQS write clock have a 90° offset between them.

For more information about dynamic OCT control, refer to the *I/O Features in Arria V Devices* chapter.

IOE Registers

The IOE registers are expanded to allow source-synchronous systems to have faster register-to-FIFO transfers and resynchronization. All top, bottom, and right IOEs have the same capability.

Input Registers

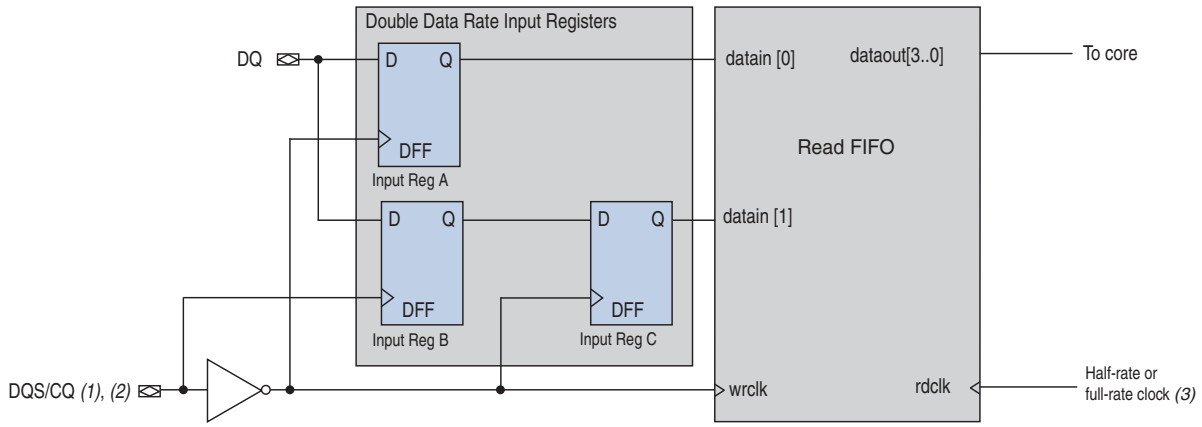
The input path consists of the DDR input registers and the read FIFO block. You can bypass each block of the input path.

There are three registers in the DDR input registers block. Registers A and B capture data on the positive and negative edges of the clock while register C aligns the captured data. Register C uses the same clock as Register A.

The read FIFO block resynchronizes the data to the system clock domain and lowers the data rate to half rate.

Figure 7–13 shows the registers available in the Arria V input path.

Figure 7–13. IOE Input Registers for Arria V Devices



Notes to Figure 7–13:

(1) The input clock can be from the DQS logic block or from a global clock line.

(2) The DQS and DQSn signals must be inverted for DDR3 and DDR2 SDRAM interfaces. When using Altera's memory interface IPs, the DQS and DQSn signals are automatically inverted.

(3) This half-rate or full-rate read clock comes from a PLL through the clock network.

Output Registers

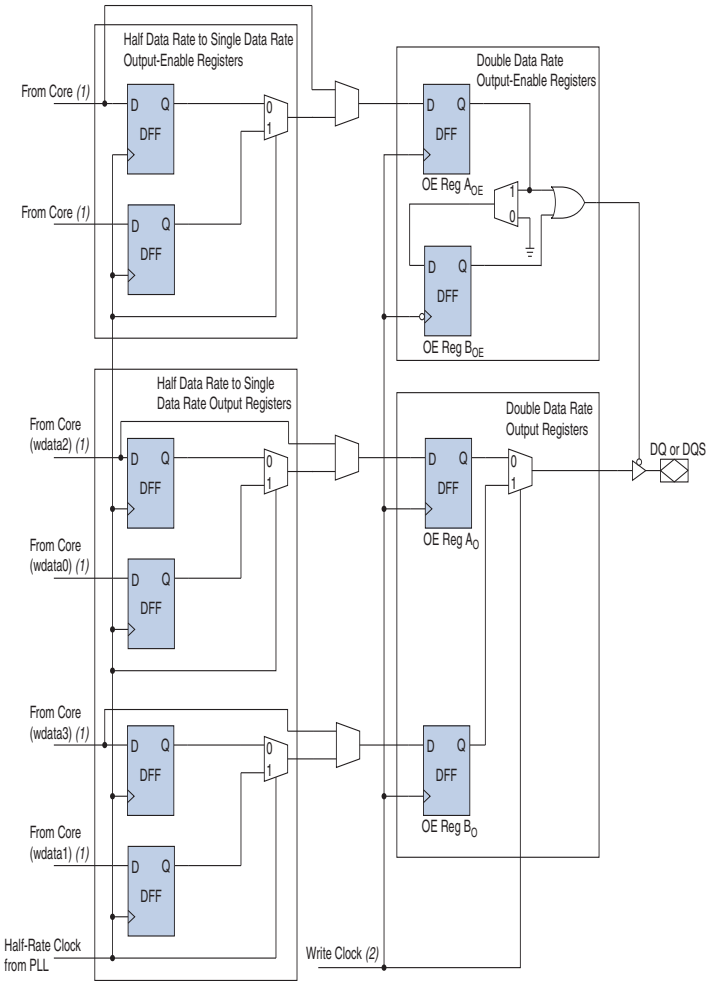
The Arria V output and output-enable path is divided into the HDR block, and output and output-enable registers. The device can bypass each block of the output and output-enable path.

The output path is designed to route combinatorial or registered single data rate (SDR) outputs and full-rate or half-rate DDR outputs from the FPGA core. Half-rate data is converted to full-rate with the HDR block, clocked by the half-rate clock from the PLL.

The output-enable path has a structure similar to the output path—ensuring that the output-enable path goes through the same delay and latency as the output path.

Figure 7–14 shows the registers available in the Arria V output and output-enable paths.

Figure 7–14. IOE Output and Output-Enable Path Registers for Arria V Devices



Notes to Figure 7–14:

(1) Data coming from the FPGA core are at half the frequency of the memory interface clock frequency in half-rate mode.

(2) The full-rate write clock can come from the PLL. The DQ write clock have a 90° offset to the DQS write clock.

Delay Chain

Arria V devices have run-time adjustable delay chains in the I/O blocks and the DQS logic blocks.

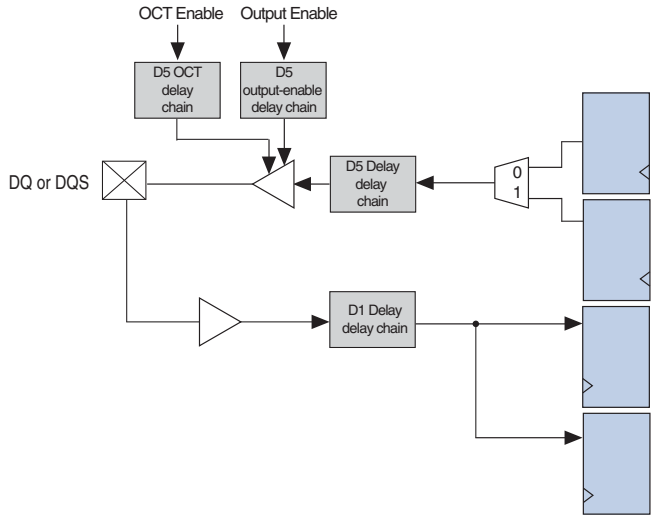
You can control the delay chain setting through the I/O or the DQS configuration block output.

Every I/O block contains a delay chain between the following elements:

- The output registers and output buffer
- The input buffer and input register
- The output enable and output buffer
- The R_T OCT enable-control register and output buffer

Figure 7–15 shows the delay chains in an I/O block.

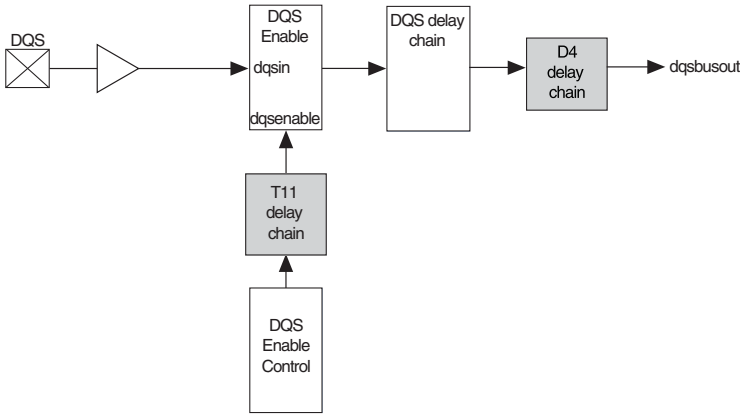
Figure 7–15. Delay Chains in an I/O Block



Each DQS logic block contains a delay chain after the `dqsbusout` output and another delay chain before the `dqsenable` input.

Figure 7–16 shows the delay chains in the DQS input path.

Figure 7–16. Delay Chains in the DQS Input Path



Hard Memory Controllers

Arria V dedicated hard memory controllers allow support for higher memory interface frequencies with shorter latency cycles in comparison to the Arria V memory controllers implemented using core logic.

The hard memory controllers in Arria V devices use dedicated I/O pins as data, address, command, control, and clock pins for the SDRAM interface. If you do not use the hard memory controllers, you can use these dedicated pins as regular I/O pins.

You can use the dedicated memory controllers that supports other features similar to the DDR2 and DDR3 SDRAM High-Performance Controller II for DDR2 and DDR3 SDRAM interfaces.

Features of the Hard Memory Controller

Table 7–3 lists the features of the hard memory controller in Arria V devices.

Table 7–3. Features of the Arria V Hard Memory Controller (Part 1 of 2)

Feature	Description
Memory Interface Data Width	■ 8-, 16-, and 32-bit data ■ 16-bit data + 8-bit ECC ■ 32-bit data + 8-bit ECC
Memory Density	The controller supports up to four gigabits density parts and two chip selects.
Memory Burst Length	■ DDR3—Burst length of 8 and burst chop of 4 ■ DDR2—Burst lengths of 4 and 8
Command and Data Reordering	The controller increases efficiency through the support for out-of-order execution of DRAM commands—with address collision detection—and in-order return of results.
Starvation Control	A starvation counter ensures that all requests are served after a predefined time-out period. This function ensures that data with low priority access are not left behind when reordering data for efficiency.

Table 7–3. Features of the Arria V Hard Memory Controller (Part 2 of 2)

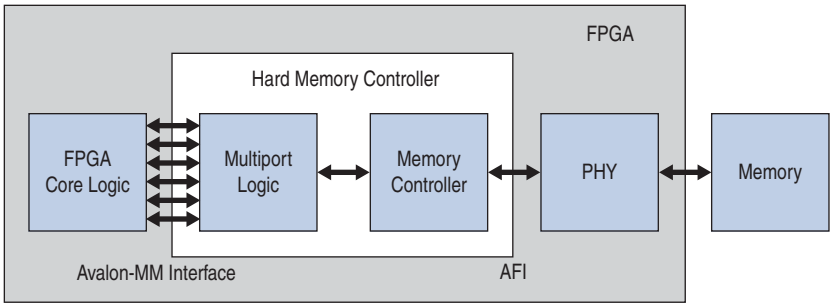
Feature	Description
User-Configurable Priority Support	When the controller detects a high priority request, it allows the request to bypass the current queuing request. This request is processed immediately and thus reduces latency.
Avalon®-MM Data Slave Local Interface	By default, the controller supports the Avalon Memory-Mapped protocol.
Bank Management	By default, the controller provides closed-page bank management on every access. The controller intelligently keeps a row open based on incoming traffic. This feature improves the efficiency of the controller especially for random traffic.
Streaming Reads and Writes	The controller can issue reads or writes continuously to sequential addresses every clock cycle if the bank is open. This function allows for very high efficiencies with large amounts of data.
Bank Interleaving	The controller can issue reads or writes continuously to 'random' addresses.
Predictive Bank Management	The controller can issue bank management commands early so that the correct row is open when the read or write occurs. This increases efficiency.
Multiport Interface	The interface allows you to connect up to six data masters to access the memory controller through the local interface. You can update the multiport scheduling configuration without interrupting traffic on a port.
Built-in Burst Adaptor	The controller can accept bursts of arbitrary sizes on its local interface and map these bursts to efficient memory commands.
Run-time Configuration of the Controller	This feature provides support for updates to the timing parameters without requiring reconfiguration of the FPGA, apart from the standard compile-time setting of the timing parameters.
On-Die Termination	The controller controls the on-die termination (ODT) in the memory, which improves signal integrity and simplifies your board design.
User-Controlled Refresh Timing	You can optionally control when refreshes occur—allowing the refreshes to avoid clashing of important reads or writes with the refresh lock-out time.
Low Power Modes	You can optionally request the controller to put the memory into the self-refresh or deep power-down modes.
Partial Array Self-Refresh	You can select the region of memory to refresh during self-refresh through the mode register to save power.
ECC	Standard Hamming single error correction, double error detection (SECDED) error correction code (ECC) support: ■ 32-bit data + 8-bit ECC ■ 16-bit data + 8-bit ECC
Additive Latency	With additive latency, the controller can issue a READ/WRITE command after the ACTIVATE command to the bank prior to t_{RCD} to increase the command efficiency. CAUTION Efficiency degradation may occur when using the additive latency feature with the hard memory controller for DDR3 SDRAM interfaces at 533 MHz.
Write Acknowledgement	The controller supports write acknowledgment on the local interface.
User Control of Memory Controller Initialization	The controller supports initialization of the memory controller under the control of user logic—for example, through the software control in the user system if a processor is present.
Controller Bonding Support	You can bond two controllers to achieve wider data width for higher bandwidth applications.

Multiport Logic

Multiport logic allows up to six local interfaces from the core logic to access a controller.

Figure 7–17 shows a simplified diagram of the Arria V hard memory controller with the multiport logic.

Figure 7–17. Simplified Diagram of the Arria V Hard Memory Interface



Bonding Support

You can bond one port of any data width (64, 128, or 256 bits) from two hard memory controllers to support wider data widths.

When you bond two hard memory controllers, the data going out of the controllers to the user logic is synchronized. However, the data going out of the controllers to the memory is not synchronized.

The bonding controllers are not synchronized and remain independent with two separate address buses and two independent command buses. These buses are calibrated separately.

Figure 7–18 shows the bonding of two opposite hard memory controllers through the core fabric.

Figure 7–18. Hard Memory Controllers in Arria V GX A1 and A3 Devices

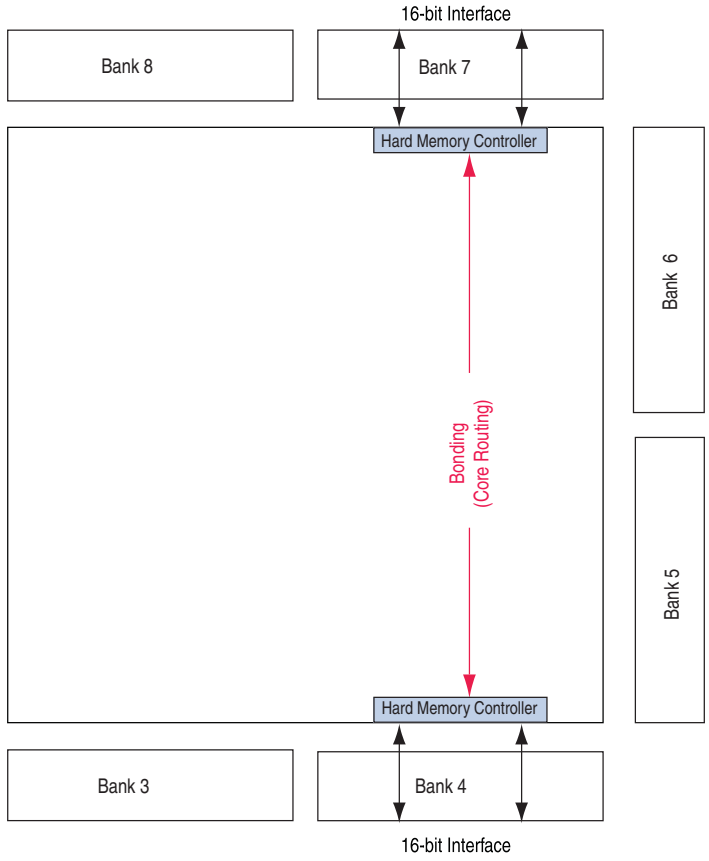
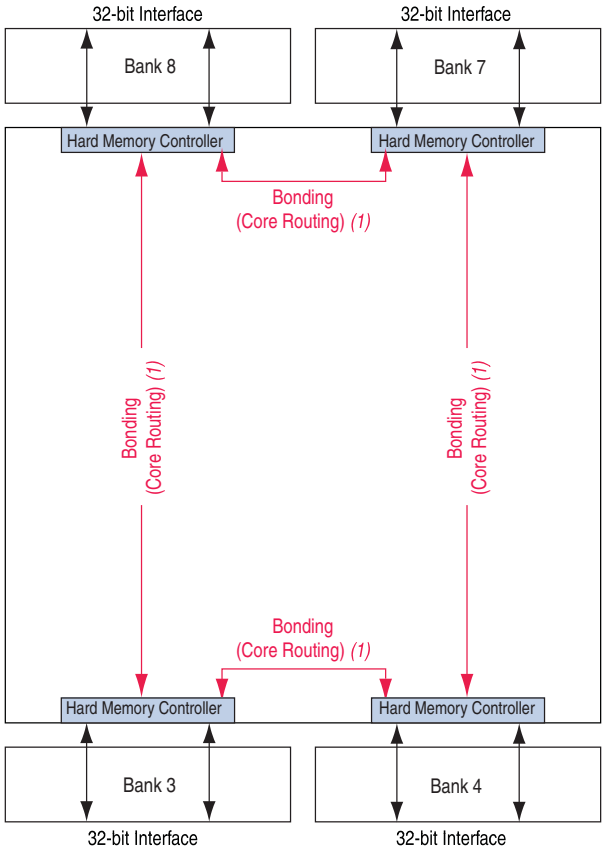


Figure 7–19 shows the bonding of opposite and same side hard memory controllers through the core fabric.

Figure 7–19. Hard Memory Controllers in Arria V GX A5, A7, B1, B3, B5, and B7 devices, and Arria V GT D3 and D7 Devices

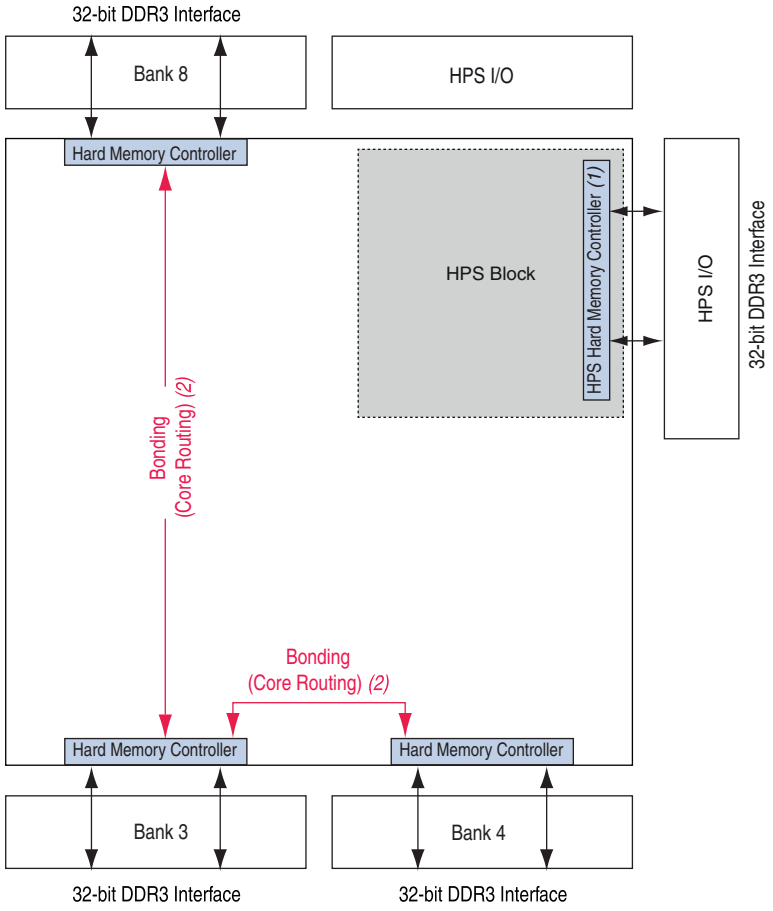


Note to Figure 7–19:

(1) Core routing is enabled only for single hard memory controller bond out per side. This bonding is available only if you do not use the hard memory controllers in banks 4 and 7 for bonding with other banks.

Figure 7–20 shows the bonding of opposite and same side hard memory controllers through the core fabric

Figure 7–20. Hard Memory Controllers in Arria V SX B3 and B5 Devices, and Arria V ST D3 and D5 Devices



Notes to Figure 7–20:

(1) There is no bonding support for the HPS hard memory controller.

(2) This is enabled only for single hard memory controller bond out per side. This bonding is available only if you do not use the hard memory controllers in bank 3 for bonding with other banks.

For more information about the dedicated pins, refer to the *Arria V Device Family Pin Connection Guidelines*.

UniPHY IP

The UniPHY IP is optimized to take advantage of the Arria V I/O structure and the Quartus II software TimeQuest Timing Analyzer. Memory controllers with UniPHY IP ensures the highest reliable frequency of operation across process, voltage, and temperature (PVT) variations.

The UniPHY IP instantiates a PLL to generate related clocks for the memory interface.

Arria V devices have built-in circuitry in the IOE to convert data from full rate (the I/O frequency) to half rate (the controller frequency) and vice versa. Memory controllers with the UniPHY IP and the Altera memory controller MegaCore functions can run at half the frequency of the I/O interface of the memory devices to allow better timing management in high-speed memory interfaces.

The UniPHY IP can also dynamically choose the number of DQS delay chains that are required for the system. The amount of delay is equal to the sum of the intrinsic delay of the delay element and the product of the number of delay steps and the value of the delay steps.

For more information about the UniPHY megafunction, refer to the *Reference Material* volume of the *External Memory Interface Handbook*.

Document Revision History

Table 7–4 lists the revision history for this chapter.

Table 7–4. Document Revision History

Date	Version	Changes
June 2012	2.0	Updated for the Quartus II software v12.0 release: ■ Restructured chapter. ■ Updated "Design Considerations", "DQS Postamble Circuitry", and "IOE Registers"sections. ■ Added SoC devices information. ■ Added Figure 7–4, Figure 7–8, and Figure 7–20.
November 2011	1.1	■ Updated Table 7–2. ■ Added "PHY Clock (PHYCLK) Networks" and "UniPHY IP" sections. ■ Restructured chapter.
May 2011	1.0	Initial release.

Section III. System Integration for Arria V Devices

This section provides information about Arria® V device configuration, design security, remote system upgrades, SEU mitigation, JTAG, and power requirements. This section includes the following chapters:

- Chapter 8, Configuration, Design Security, and Remote System Upgrades in Arria V Devices
- Chapter 9, SEU Mitigation in Arria V Devices
- Chapter 10, JTAG Boundary-Scan Testing in Arria V Devices
- Chapter 11, Power Management in Arria V Devices

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 this volume.

8. Configuration, Design Security, and Remote System Upgrades in Arria V Devices

AV-52008-2.0

This chapter describes the configuration schemes, design security, and remote system upgrade that are supported by the Arria® V devices.

This chapter contains the following sections:

For more information about the following topics, refer to the following documents:

- For the configuration features supported for each configuration scheme, refer to the *Arria V Device Overview*.
- For the Configuration via Protocol (CvP) configuration scheme, refer to the *Configuration via Protocol (CvP) Implementation in Altera FPGAs User Guide*.
- For the estimated uncompressed Raw Binary File (**.rbf**) sizes, fast passive parallel (FPP) `DCLK`-to-`DATA[]` ratio, and timing parameters, refer to the *Arria V Device Datasheet*.

MSEL Pin Settings

To select a configuration scheme, you must set the MSEL pins as specified in Table 8–1 by hardwiring the pins to V_{CCPGM} or GND without pull-up or pull-down resistors. Do not drive the MSEL pins with a microprocessor or another device.

Table 8–1. MSEL Pin Settings for Each Configuration Scheme of Arria V Devices

Configuration Scheme	Compression Feature	Design Security Feature	V_{CCPGM} (V)	POR Delay	Valid MSEL[4..0]
FPP x8	Disabled	Disabled	1.8/2.5/3.0/3.3	Fast	10100
				Standard	11000
	Disabled	Enabled	1.8/2.5/3.0/3.3	Fast	10101
				Standard	11001
	Enabled	Enabled/ Disabled	1.8/2.5/3.0/3.3	Fast	10110
				Standard	11010
FPP x16	Disabled	Disabled	1.8/2.5/3.0/3.3	Fast	00000
				Standard	00100
	Disabled	Enabled	1.8/2.5/3.0/3.3	Fast	00001
				Standard	00101
	Enabled	Enabled/ Disabled	1.8/2.5/3.0/3.3	Fast	00010
				Standard	00110
Passive serial (PS)	Enabled/ Disabled	Enabled/ Disabled	1.8/2.5/3.0/3.3	Fast	10000
				Standard	10001
Active serial (AS) (x1 and x4)	Enabled/ Disabled	Enabled/ Disabled	3.0/3.3	Fast	10010
				Standard	10011
JTAG-based configuration	Disabled	Disabled	—	—	Use any valid MSEL pin settings above

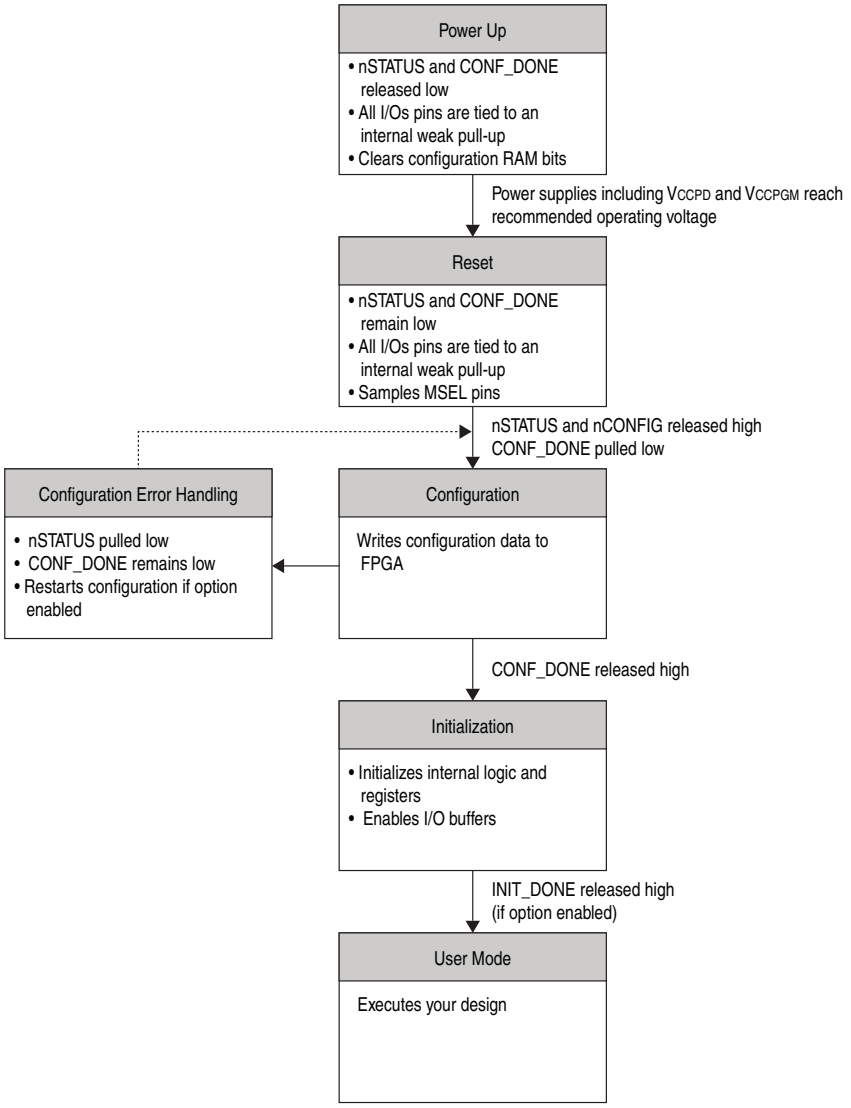
You must also select the configuration scheme in the **Configuration** page of the **Device and Pin Options** dialog box in the Quartus® II software. Based on your selection, the option bit in the programming file is set accordingly.

Configuration Sequence

This section describes the configuration sequence and each configuration stage.

Figure 8–1 shows the configuration sequence.

Figure 8–1. Configuration Sequence for Arria V Devices



You can initiate reconfiguration at any stage by pulling the nCONFIG pin low to at least the minimum t_{CFG} low-pulse width. When this pin is pulled low, the nSTATUS and CONF_DONE pins are pulled low and all I/O pins are tied to an internal weak pull-up.

Power Up

Power up all the power supplies that are monitored by the POR circuitry. All power supplies, including V_{CCPGM} and V_{CCPD}, must ramp up from 0 V to the recommended operating voltage level within the ramp-up time specification. Otherwise, hold the nCONFIG pin low until all the power supplies reach the recommended voltage level.

For more information about the ramp-up time specifications, refer to the *Power Management in Arria V Devices* chapter.

V_{CCPGM} Pin

The configuration input buffers do not have to share power lines with the regular I/O buffers in Arria V devices.

The operating voltage for the configuration input pin is independent of the I/O banks power supply, V_{CCIO}, during configuration. Therefore, Arria V devices do not require configuration voltage constraints on V_{CCIO}.

For more information about configuration pins connections, refer to the *Arria V Device Family Pin Connection Guidelines*.

V_{CCPD} Pin

Use the V_{CCPD} pin, a dedicated programming power supply, to power the I/O pre-drivers and JTAG I/O pins (TCK, TMS, TDI, and TDO). The supported configuration voltages are 2.5, 3.0, and 3.3 V.

If V_{CCIO} of the bank is set to 2.5 V or lower, V_{CCPD} must be powered up at 2.5 V. If V_{CCIO} is set greater than 2.5 V, V_{CCPD} must be greater than V_{CCIO}. For example, when V_{CCIO} is set to 3.0 V, V_{CCPD} must be set at 3.0 V or above. When V_{CCIO} is set to 3.3 V, V_{CCPD} must be set at 3.3 V.

For more information about the configuration pins, refer to "Device Configuration Pins" on page 8–6.

Reset

POR delay is the time frame between the time when all the power supplies monitored by the POR circuitry reach the recommended operating voltage and when nSTATUS is released high and the Arria V device is ready to begin configuration.

Set the POR delay using the MSEL pins. For more information about the MSEL pin settings, refer to Table 8–1 on page 8–2.

The user I/O pins are tied to an internal weak pull-up until the device is configured.

For more information about the POR delay specification, refer to the *Arria V Device Datasheet*.

Configuration

For more information about the DATA[] pins for each configuration scheme, refer to the appropriate configuration scheme.

Configuration Error Handling

To restart configuration automatically, turn on the **Auto-restart configuration after error** option in the **General** page of the **Device and Pin Options** dialog box in the Quartus II software.

If you do not turn on this option, you can monitor the nSTATUS pin to detect errors. To restart configuration, pull the nCONFIG pin high for at least the duration of t_{CFG}.

For more information about t_{STATUS} and t_{CFG} timing parameters, refer to the *Arria V Device Datasheet*.

Initialization

The initialization clock source is from the internal oscillator, CLKUSR pin, or DCLK pin. By default, the internal oscillator is the clock source for initialization. If you use the internal oscillator, the Arria V device will be provided with enough clock cycles for proper initialization.

If you use the optional CLKUSR pin as the initialization clock source and the nCONFIG pin is pulled low to restart configuration during device initialization, ensure that the CLKUSR or DCLK pin continues toggling until the nSTATUS pin goes low and then goes high again.

The CLKUSR pin provides you with the flexibility to synchronize initialization of multiple devices or to delay initialization. Supplying a clock on the CLKUSR pin during initialization does not affect configuration. After the CONF_DONE pin goes high, the CLKUSR or DCLK pin is enabled after the time specified by t_{CD2CU}. When this time period elapses, Arria V devices require a minimum number of clock cycles as specified by T_{init} to initialize properly and enter user mode as specified by the t_{CD2UMC} parameter.

For more information about t_{CD2CU}, t_{init}, and t_{CD2UMC} timing parameters, and initialization clock source, refer to the *Arria V Device Datasheet*.

User Mode

You can enable the optional INIT_DONE pin to monitor the initialization stage. After the INIT_DONE pin is pulled high, initialization completes and your design starts executing. The user I/O pins will then function as specified by your design.

Device Configuration Pins

This section lists the configuration pins for Arria V devices.

Configuration Pins Summary

Table 8–2 lists the Arria V configuration pins and their power supply.

Table 8–2. Configuration Pin Summary for Arria V Devices

Configuration Pin	Configuration Scheme	Input/Output	User Mode	Powered By
TDI	JTAG	Input	—	V_{CCPD} (1)
TMS	JTAG	Input	—	V_{CCPD} (1)
TCK	JTAG	Input	—	V_{CCPD} (1)
TDO	JTAG	Output	—	V_{CCPD} (1)
CLKUSR	All schemes	Input	I/O	V_{CCPGM}/V_{CCIO} (2)
CRC_ERROR	Optional, all schemes	Output	I/O	Pull-up
CONF_DONE	All schemes	Bidirectional	—	V_{CCPGM}/Pull-up
DCLK	FPP and PS	Input	—	V_{CCPGM}
	AS	Output	—	V_{CCPGM}
DEV_OE	Optional, all schemes	Input	I/O	V_{CCPGM}/V_{CCIO} (2)
DEV_CLRn	Optional, all schemes	Input	I/O	V_{CCPGM}/V_{CCIO} (2)
INIT_DONE	Optional, all schemes	Output	I/O	Pull-up
MSEL[4..0]	All schemes	Input	—	V_{CCPGM}
nSTATUS	All schemes	Bidirectional	—	V_{CCPGM}/Pull-up
nCE	All schemes	Input	—	V_{CCPGM}
nCEO	All schemes	Output	I/O	Pull-up
nCONFIG	All schemes	Input	—	V_{CCPGM}
DATA[15..5]	FPP	Input	I/O	V_{CCPGM}/V_{CCIO} (2)
nCSO/DATA4	AS	Output	—	V_{CCPGM}
	FPP	Bidirectional	—	V_{CCPGM}/V_{CCIO} (2)
AS_DATA[3..1]/DATA[3..1]	AS	Bidirectional	—	V_{CCPGM}
	FPP	Bidirectional	—	V_{CCPGM}/V_{CCIO} (2)
AS_DATA0/DATA0/ASDO	AS	Bidirectional	—	V_{CCPGM}
	FPP and PS	Bidirectional	—	V_{CCPGM}/V_{CCIO} (2)

Notes to Table 8–2:

(1) This pin is powered by V_{CCPD} of the bank in which the pin resides.

(2) This pin is powered by V_{CCPGM} during configuration and by V_{CCIO} of the bank in which the pin resides if you use it as a user I/O pin.

For more information about each configuration pin, refer to the *Arria V Device Family Pin Connection Guidelines*.

Configuration Pin Options in the Quartus II Software

Table 8–3 lists the dual-purpose configuration pins available in the **Device and Pin Options** dialog box in the Quartus II software.

Table 8–3. Configuration Pin Options

Configuration Pin	Category Page	Option
CLKUSR	General	Enable user-supplied start-up clock (CLKUSR)
DEV_CLRn	General	Enable device-wide reset (DEV_CLRn)
DEV_OE	General	Enable device-wide output enable (DEV_OE)
INIT_DONE	General	Enable INIT_DONE output
nCEO	General	Enable nCEO pin
CRC_ERROR	Error Detection CRC	Enable Error Detection CRC_ERROR pin
		Enable open drain on CRC_ERROR pin
		Enable internal scrubbing

For more information about the device and pin options dialog box setting, refer to the *Reviewing Printed Circuit Board Schematics with the Quartus II Software* chapter in volume 2 of the *Quartus II Handbook*.

Fast Passive Parallel Configuration

The FPP configuration scheme uses an external host, such as a microprocessor, MAX® II device, or MAX V device. This scheme is the fastest method to configure Arria V devices. The FPP configuration scheme supports 8- and 16-bits data width.

You can use an external host to control the transfer of configuration data from an external storage such as flash memory to the FPGA. The design that controls the configuration process resides in the external host. You can store the configuration data in **.rbf**, Hexadecimal (Intel-Format) File (**.hex**), or Tabular Text File (**.ttf**) formats.

You can use the parallel flash loader (PFL) megafunction with a MAX II or MAX V device to read configuration data from the flash memory device and configure the Arria V device.

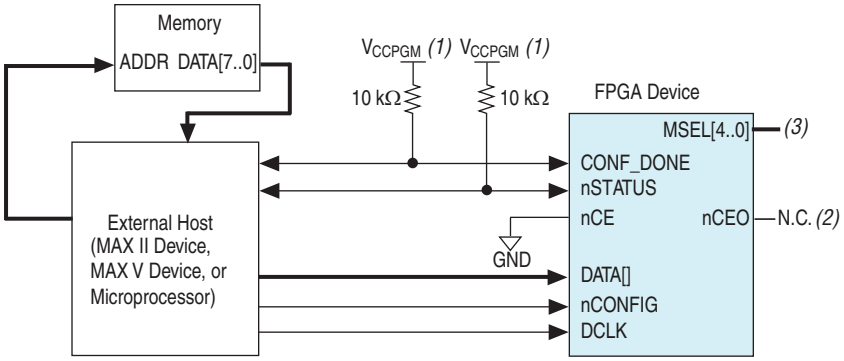
For more information about the PFL megafunction, refer to the *Parallel Flash Loader Megafunction User Guide*.

Two DCLK falling edges are required after the CONF_DONE pin goes high to begin the initialization of the device for both uncompressed and compressed configuration data in an FPP configuration.

FPP Single-Device Configuration

To configure an Arria V device, connect the device to an external host as shown in Figure 8–2.

Figure 8–2. Single Device FPP Configuration Using an External Host



Notes to Figure 8–2:

(1) Connect the resistor to a supply that provides an acceptable input signal for the Arria V device. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends powering up all configuration system I/Os with V_{CCPGM}.

(2) You can leave the nCEO pin unconnected or use it as a user I/O pin when it does not feed another device's nCE pin.

(3) For the MSEL pin settings, refer to Table 8–1 on page 8–2.

FPP Multi-Device Configuration

You can configure multiple Arria V devices that are connected in a chain.

Pin Connections and Guidelines

Observe the following pin connections and guidelines for this configuration setup:

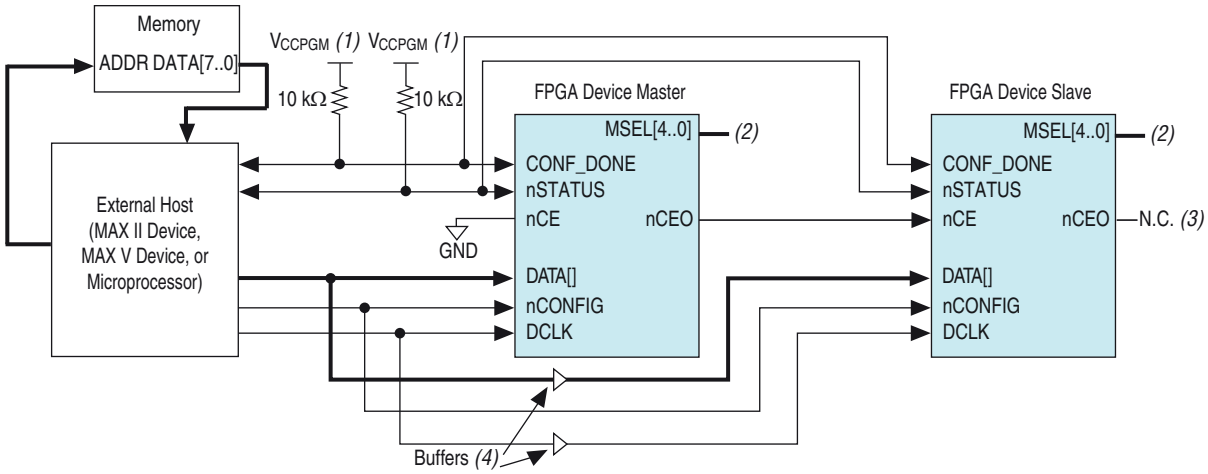
- Tie the following pins of all devices in the chain together:
  - nCONFIG
  - nSTATUS
  - DCLK
  - DATA[]
  - CONF_DONE

  By tying the CONF_DONE and nSTATUS pins together, the devices initialize and enter user mode at the same time. If any device in the chain detects an error, configuration stops for the entire chain and you must reconfigure all the devices. For example, if the first device in the chain flags an error on the nSTATUS pin, it resets the chain by pulling its nSTATUS pin low.
- Ensure that DCLK and DATA[] are buffered for every fourth device to prevent signal integrity and clock skew problems.
- All devices in the chain must use the same data width.
- If you are configuring the devices in the chain using the same configuration data, the devices must be of the same package and density.

Using Multiple Configuration Data

To configure multiple Arria V devices in a chain using multiple configuration data, connect the devices to an external host as shown in Figure 8–3.

Figure 8–3. Multiple Device FPP Configuration Using an External Host When Both Devices Receive a Different Set of Configuration Data



Notes to Figure 8–3:

(1) Connect the resistor to a supply that provides an acceptable input signal for the Arria V device. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends powering up all configuration system I/Os with V_{CCPGM}.

(2) For the MSEL pin settings, refer to Table 8–1 on page 8–2.

(3) You can leave the nCEO pin unconnected or use it as a user I/O pin when it does not feed another device's nCE pin.

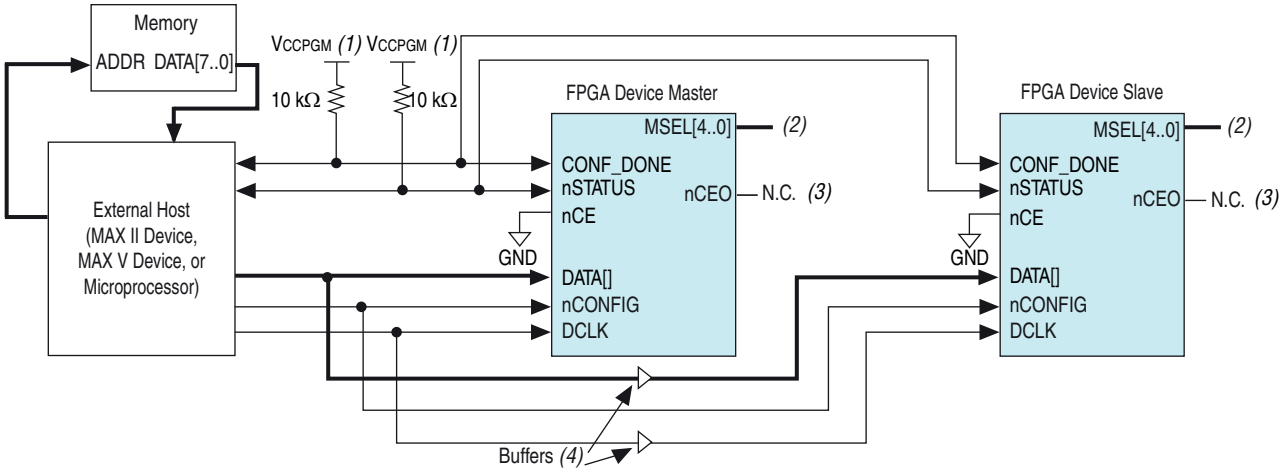
(4) Connect the repeater buffers between the Arria V master and slave device for DATA[] and DCLK for every fourth device.

When a device completes configuration, its nCEO pin is released low to activate the nCE pin of the next device in the chain. Configuration automatically begins for the second device in one clock cycle.

Configuring Multiple Devices Using One Configuration Data

To configure multiple Arria V devices in a chain using one configuration data, connect the devices to an external host as shown in Figure 8–4.

Figure 8–4. Multiple Device FPP Configuration Using an External Host When Both Devices Receive the Same Data



Notes to Figure 8–4:

(1) Connect the resistor to a supply that provides an acceptable input signal for the Arria V device. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends powering up all configuration system I/Os with V_{CCPGM}.

(2) For the MSEL pin settings, refer to Table 8–1 on page 8–2.

(3) You can leave the nCEO pin unconnected or use it as a user I/O pin when it does not feed another device's nCE pin.

(4) Connect the repeater buffers between the Arria V master and slave device for DATA[] and DCLK for every fourth device.

The nCE pins of the devices in the chain are connected to GND, allowing configuration for these devices to begin and end at the same time.

Active Serial Configuration

The AS configuration scheme supports AS x1 (1-bit data width) and AS x4 (4-bit data width) modes. The AS x4 mode provides four times faster configuration time than the AS x1 mode. In the AS configuration scheme, the Arria V device controls the configuration interface.

DATA Clock (DCLK)

Arria V devices generate the serial clock, DCLK, that provides timing to the serial interface. In the AS configuration scheme, Arria V devices drive control signals on the falling edge of DCLK and latch the configuration data on the following falling edge of this clock pin.

The maximum DCLK frequency supported by the AS configuration scheme is 100 MHz except for the AS multi-device configuration scheme. You can source DCLK using CLKUSR or the internal oscillator. If you use the internal oscillator, you can choose a 12.5, 25, 50, or 100 MHz clock under the **Device and Pins Option** dialog box, in the **Configuration** page of the Quartus II software.

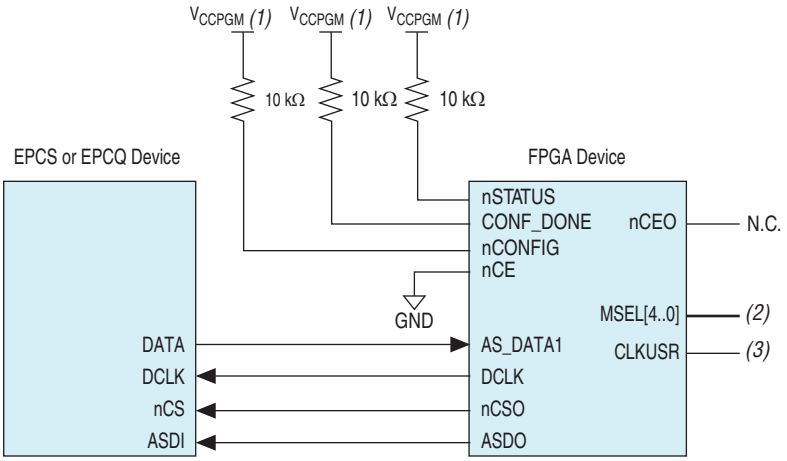
After power-up, DCLK is driven by a 12.5 MHz internal oscillator by default. The Arria V device determines the clock source and frequency to use by reading the option bit in the programming file.

For more information about DCLK frequency specification in the AS configuration scheme, refer to the *Arria V Device Datasheet*.

AS Single-Device Configuration

To configure an Arria V device, connect the device to a serial configuration (EPCS) device or quad-serial configuration (EPCQ) device, as shown in Figure 8–5 and Figure 8–6.

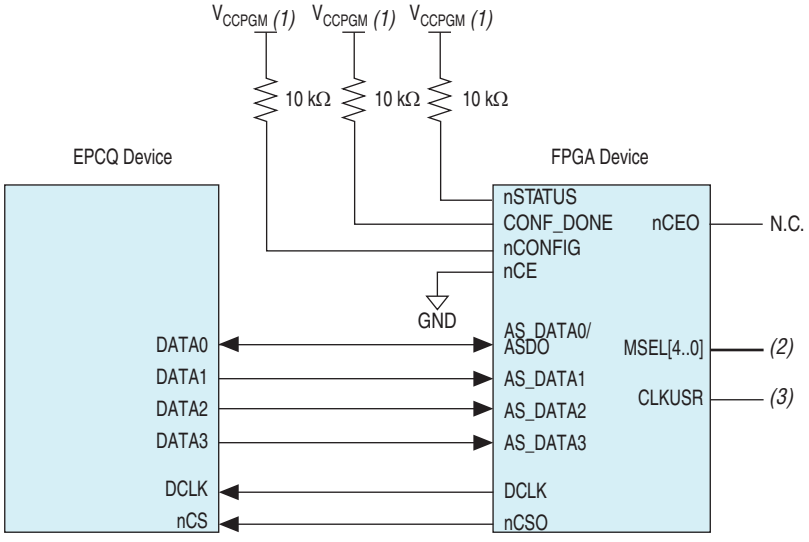
Figure 8–5. Single Device AS x1 Mode Configuration



Notes to Figure 8–5:

(1) Connect the pull-up resistors to V_{CCPGM} at a 3.0- or 3.3-V power supply.
(2) For the MSEL pin settings, refer to Table 8–1 on page 8–2.
(3) Use the CLKUSR pin to supply the external clock source to drive DCLK during configuration.

Figure 8–6. Single Device AS x4 Mode Configuration



Notes to Figure 8–6:

(1) Connect the pull-up resistors to V_{CCPGM} at a 3.0- or 3.3-V power supply.
(2) For the MSEL pin settings, refer to Table 8–1 on page 8–2.
(3) Use the CLKUSR pin to supply the external clock source to drive DCLK during configuration.

AS Multi-Device Configuration

You can configure multiple Arria V devices that are connected in a chain. Only AS x1 mode supports multi-device configuration.

The first device in the chain is the configuration master. Subsequent devices in the chain are configuration slaves.

Pin Connections and Guidelines

Observe the following pin connections and guidelines for this configuration setup:

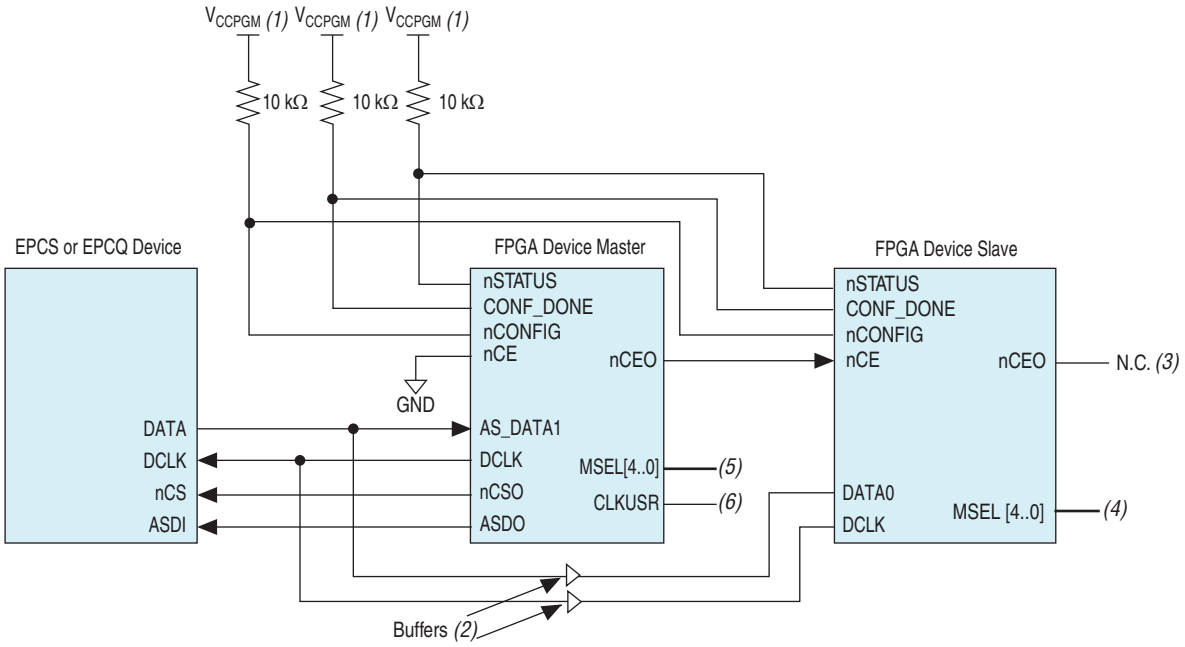
- Hardwire the MSEL pins of the first device in the chain to select the AS configuration scheme. For subsequent devices in the chain, hardwire their MSEL pins to select the PS configuration scheme. Any other Altera® devices that support the PS configuration can also be part of the chain as a configuration slaves.
- Tie the following pins of all devices in the chain together:
  - nCONFIG
  - nSTATUS
  - DCLK
  - DATA[]
  - CONF_DONE

  By tying the CONF_DONE, nSTATUS, and nCONFIG pins together, the devices initialize and enter user mode at the same time. If any device in the chain detects an error, configuration stops for the entire chain and you must reconfigure all the devices. For example, if the first device in the chain flags an error on the nSTATUS pin, it resets the chain by pulling its nSTATUS pin low.
- Ensure that DCLK and DATA[] are buffered every fourth device to prevent signal integrity and clock skew problems.

Using Multiple Configuration Data

To configure multiple Arria V devices in a chain using multiple configuration data, connect the devices to an EPCS or EPCQ device, as shown in Figure 8–7.

Figure 8–7. Multiple Device AS Configuration When Both Devices in the Chain Receive Different Sets of Configuration Data *(1)*



Notes to Figure 8–7:

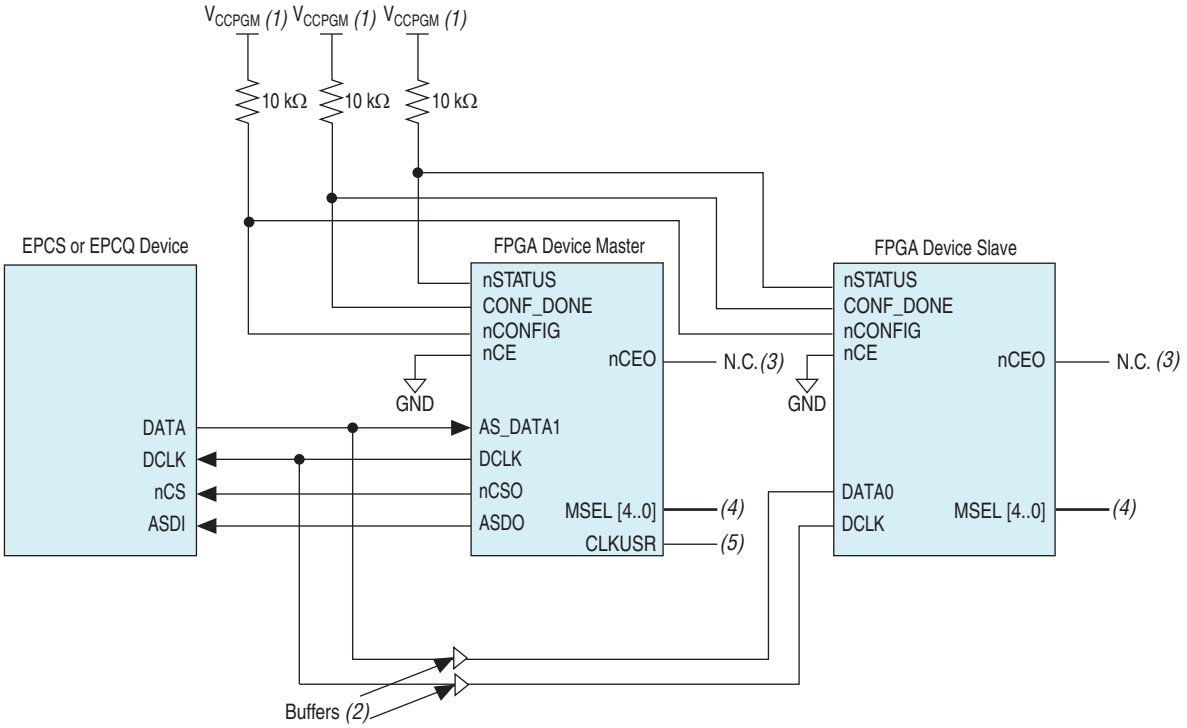
(1) Connect the pull-up resistors to V_{CCPGM} at a 3.0- or 3.3-V power supply.
(2) Connect the repeater buffers between the Arria V master and slave device for AS_DATA1 or DATA0 and DCLK for every fourth device.
(3) You can leave the nCEO pin unconnected or use it as a user I/O pin when it does not feed another device's nCE pin.
(4) For the appropriate MSEL settings based on POR delay settings, set the slave device MSEL setting to the PS scheme.
(5) For the MSEL pin settings, refer to Table 8–1 on page 8–2.
(6) Use the CLKUSR pin to supply the external clock source to drive DCLK during configuration.
(7) For 50 MHz frequency, delay DCLK in reference to DATA0 by a minimum of 5 ns and a maximum of 10 ns.

When a device completes configuration, its nCEO pin is released low to activate the nCE pin of the next device in the chain. Configuration automatically begins for the second device in one clock cycle.

Configuring Multiple Devices Using One Configuration Data

To configure multiple Arria V devices in a chain using one configuration data, connect the devices to an EPCS or EPCQ device, as shown in Figure 8–8.

Figure 8–8. Multiple Device AS Configuration When the Devices Receive the Same Data *(1)*



Notes to Figure 8–8:

(1) Connect the pull-up resistors to V_{CCPGM} at a 3.0- or 3.3-V power supply.
(2) Connect the repeater buffers between the Arria V master and slave device for `AS_DATA1` or `DATA0` and `DCLK`.
(3) You can leave the `nCEO` pin unconnected or use it as a user I/O pin when it does not feed another device's `nCE` pin.
(4) For the `MSEL` pin settings, refer to Table 8–1 on page 8–2.
(5) Use the `CLKUSR` pin to supply the external clock source to drive `DCLK` during configuration.
(6) For 50 MHz frequency, delay `DCLK` in reference to `DATA0` by a minimum of 5 ns and a maximum of 10 ns.

Estimating the AS Configuration Time

The AS configuration time is mostly the time it takes to transfer the configuration data from an EPCS or EPCQ device to the Arria V device.

Use the following equations to estimate the configuration time:

- AS x1 mode

  .rbf Size × (minimum `DCLK` period / 1 bit per `DCLK` cycle) = estimated minimum configuration time.

- AS x4 mode

  .rbf Size × (minimum `DCLK` period / 4 bits per `DCLK` cycle) = estimated minimum configuration time.

Compressing the configuration data reduces the configuration time. The amount of reduction varies depending on your design.

Using EPCS and EPCQ Devices

EPCS devices support AS x1 mode and EPCQ devices support AS x1 and AS x4 modes.

For more information about EPCS and EPCQ devices, refer to the following documents:

- *Serial Configuration (EPCS) Devices Datasheet*
- *Quad-Serial Configuration (EPCQ) Devices Datasheet*

Controlling EPCS and EPCQ Devices

During configuration, Arria V devices enable the EPCS or EPCQ device by driving its nCSO output pin low, which connects to the chip select (nCS) pin of the EPCS or EPCQ device. Arria V devices use the DCLK and ASDO pins to send operation commands and read address signals to the EPCS or EPCQ device. The EPCS or EPCQ device provides data on its serial data output (DATA[]) pin, which connects to the AS_DATA[] input of the Arria V devices.

If you wish to gain control of the EPCS pins, hold the nCONFIG pin low and pull the nCE pin high. This causes the device to reset and tri-state the AS configuration pins.

Trace Length and Loading

The maximum trace length and loading apply to both single- and multi-device AS configuration setups as listed in Table 8–4. The trace length is the length from the Arria V device to the EPCS or EPCQ device.

Table 8–4. Maximum Trace Length and Loading for AS x1 and x4 Configurations for Arria V Devices

Arria V Device AS Pins	Maximum Board Trace Length (Inches)		Maximum Board Load (pF)
	12.5/25/50 MHz	100 MHz	
DCLK	10	6	5
DATA[3..0]	10	6	10
nCSO	10	6	10

Programming EPCS and EPCQ Devices

You can program EPCS and EPCQ devices in-system using a USB-Blaster™, EthernetBlaster, EthernetBlaster II, or ByteBlaster™ II download cable. Alternatively, you can program the EPCS or EPCQ using a microprocessor with the SRunner software driver.

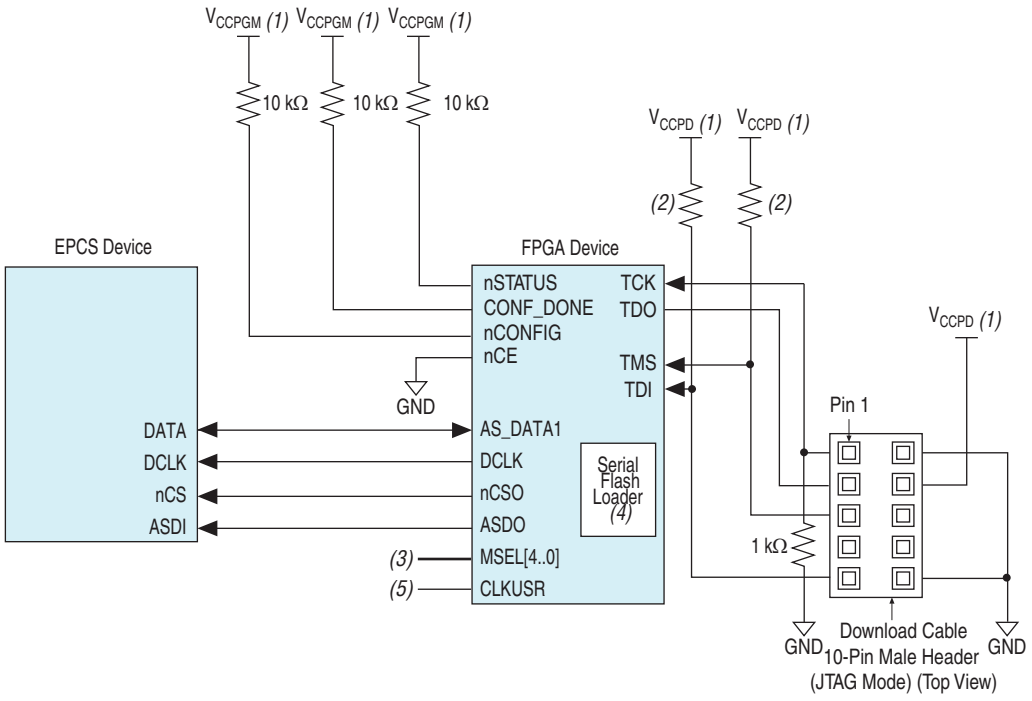
In-system programming (ISP) offers you the option to program the EPCS or EPCQ either using an AS programming interface or a JTAG interface. Using the AS programming interface, the configuration data is programmed into the EPCS by the Quartus II software or any supported third-party software. Using the JTAG interface, an Altera IP called the serial flash loader (SFL) must be downloaded into the Arria V device to form a bridge between the JTAG interface and the EPCS or EPCQ. This allows the EPCS or EPCQ to be programmed directly using the JTAG interface.

For more information about SFL and SRunner software driver, refer to the following documents:

- *AN 370: Using the Serial FlashLoader with the Quartus II Software*
- *AN 418: SRunner: An Embedded Solution for Serial Configuration Device Programming*

To program an EPCS device using the JTAG interface, connect the device as shown in Figure 8–9.

Figure 8–9. Connection Setup for Programming the EPCS Using the JTAG Interface

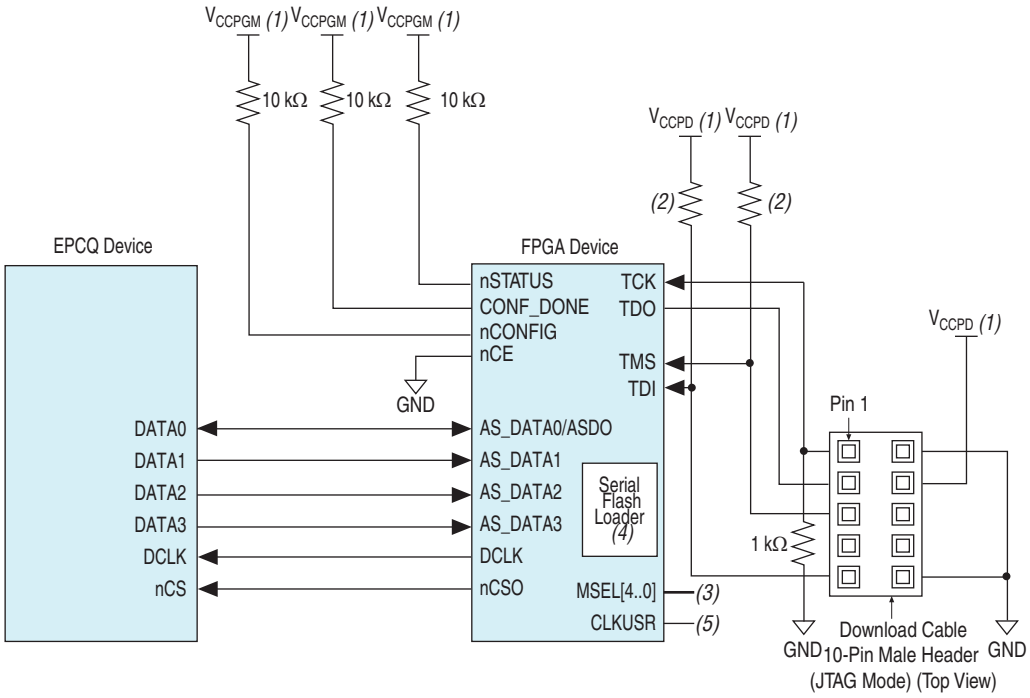


Notes to Figure 8–9:

(1) Connect the pull-up resistors to V_{CCPGM} at a 3.0- or 3.3-V power supply.
(2) The resistor value can vary from 1 kΩ to 10 kΩ. Perform signal integrity analysis to select the resistor value for your setup.
(3) For the MSEL pin settings, refer to Table 8–1 on page 8–2.
(4) Instantiate SFL in your design to form a bridge between the EPCS and the Arria V device.
(5) Use the CLKUSR pin to supply the external clock source to drive DCLK during configuration.

To program an EPCQ device using the JTAG interface, connect the device as shown in Figure 8–10.

Figure 8–10. Connection Setup for Programming the EPCQ Using the JTAG Interface

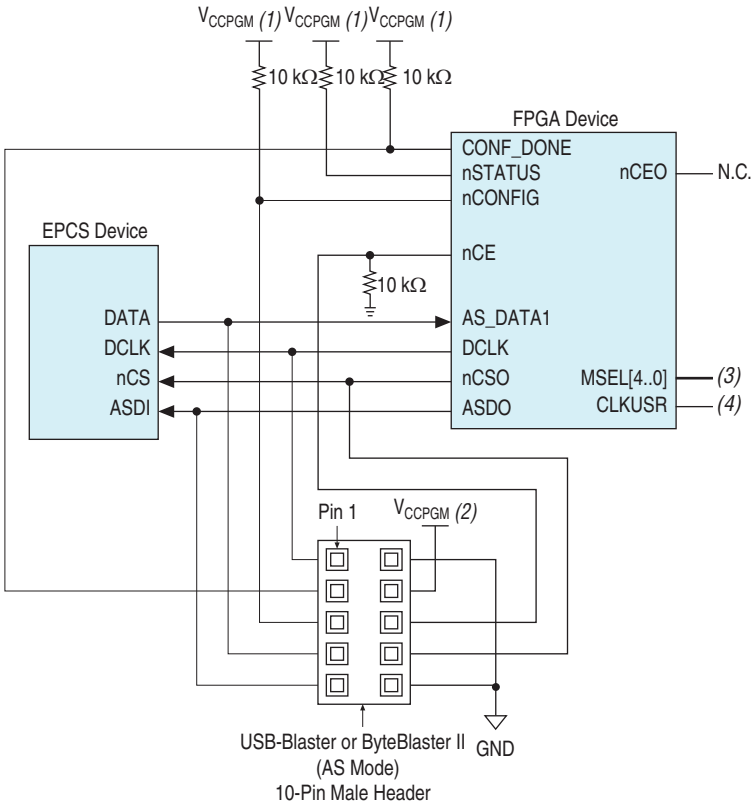


Notes to Figure 8–10:

(1) Connect the pull-up resistors to V_{CCPGM} at a 3.0- or 3.3-V power supply.
(2) The resistor value can vary from 1 kΩ to 10 kΩ. Perform signal integrity analysis to select the resistor value for your setup.
(3) For the MSEL pin settings, refer to Table 8–1 on page 8–2.
(4) Instantiate SFL in your design to form a bridge between the EPCS and the Arria V device.
(5) Use the CLKUSR pin to supply the external clock source to drive DCLK during configuration.

To program an EPCS device using the AS interface, connect the device as shown in Figure 8–11.

Figure 8–11. Connection Setup for Programming the EPCS Using the AS Interface

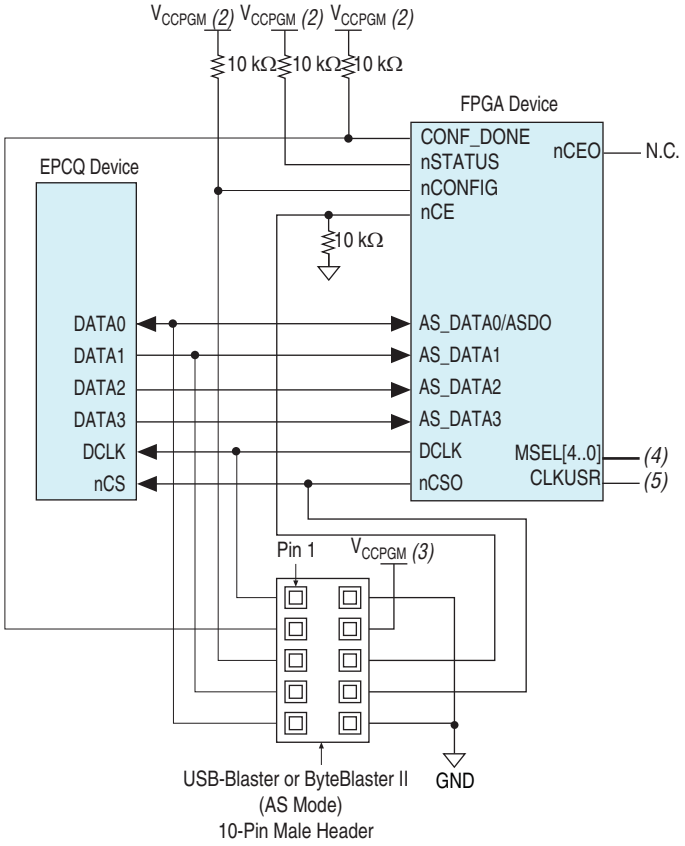


Notes to Figure 8–11:

(1) Connect the pull-up resistors to V_{CCPGM} at a 3.0- or 3.3-V power supply.
(2) Power up the USB-Blaster, ByteBlaster II, EthernetBlaster, or EthernetBlaster II cable's $V_{CC(TRGT)}$ to V_{CCPGM}.
(3) For the MSEL pin settings, refer to Table 8–1 on page 8–2.
(4) Use the CLKUSR pin to supply the external clock source to drive DCLK during configuration.

To program an EPCQ device using the AS interface, connect the device as shown in Figure 8–12.

Figure 8–12. Connection Setup for Programming the EPCQ Using the AS Interface *(1)*



Notes to Figure 8–12:

(1) Using the AS header, the programmer serially transmits the operation commands and configuration bits to the EPCQ on DATA0. This is equivalent to the programming operation for the EPCS.

(2) Connect the pull-up resistors to V_{CCPGM} at a 3.0- or 3.3-V power supply.

(3) Power up the USB-Blaster, ByteBlaster II, EthernetBlaster, or EthernetBlaster II cable's $V_{CC(TRGT)}$ to V_{CCPGM}.

(4) For the MSEL pin settings, refer to Table 8–1 on page 8–2.

(5) Use the CLKUSR pin to supply the external clock source to drive DCLK during configuration.

When programming the EPCS and EPCQ devices, the download cable disables access to the AS interface by driving the nCE pin high. The nCONFIG line is also pulled low to hold the Arria V device in the reset stage. After programming completes, the download cable releases nCE and nCONFIG, allowing the pull-down and pull-up resistors to drive the pin to GND and V_{CCPGM}, respectively.

During the EPCQ programming using the download cable, DATA0 transfers the programming data, operation command, and address information from the download cable into the EPCQ. During the EPCQ verification using the download cable, DATA1 transfers the programming data back to the download cable.

Passive Serial Configuration

The PS configuration scheme uses an external host. You can use a microprocessor, MAX II device, MAX V device, or a host PC as the external host.

You can use an external host to control the transfer of configuration data from an external storage such as flash memory to the FPGA. The design that controls the configuration process resides in the external host.

You can store the configuration data in Programmer Object File (**.pof**), **.rbf**, **.hex**, or **.ttf**. If you are using configuration data in **.rbf**, **.hex**, or **.ttf**, send the LSB of each data byte first. For example, if the **.rbf** contains the byte sequence 02 1B EE 01 FA, the serial data transmitted to the device must be 0100-0000 1101-1000 0111-0111 1000-0000 0101-1111.

You can use the PFL megafunction with a MAX II or MAX V device to read configuration data from the flash memory device and configure the Arria V device.

For more information about the PFL megafunction, refer to the *Parallel Flash Loader Megafunction User Guide*.

For a PC host, connect the PC to the device using a download cable such as the Altera USB-Blaster USB port, ByteBlaster II parallel port, EthernetBlaster, and EthernetBlaster II download cables.

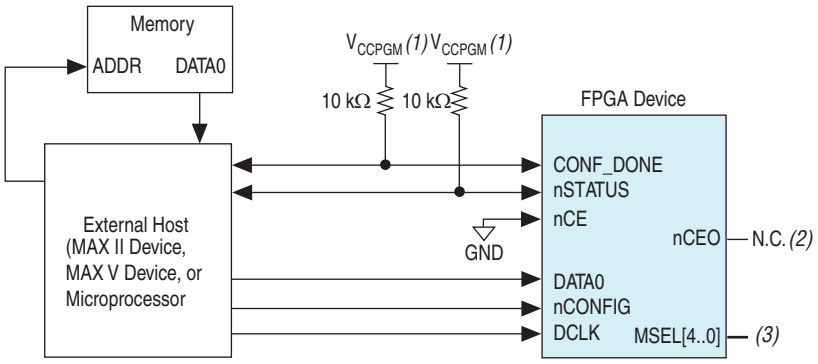
The configuration data is shifted serially into the DATA0 pin of the device.

If you are using the Quartus II programmer and the CLKUSR pin is enabled, you do not need to provide a clock source for the pin to initialize your device.

PS Single-Device Configuration

To configure an Arria V device, connect the device to an external host, as shown in Figure 8–13.

Figure 8–13. Single Device PS Configuration Using an External Host



Notes to Figure 8–13:

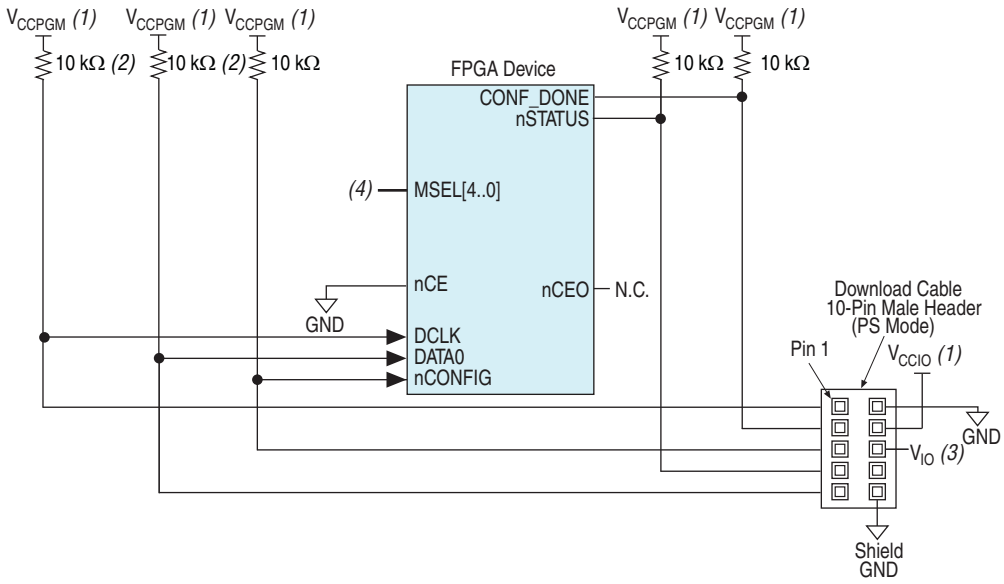
(1) Connect the resistor to a power supply that provides an acceptable input signal for the Arria V device. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends powering up all the configuration system I/Os with V_{CCPGM}.

(2) You can leave the nCEO pin unconnected or use it as a user I/O pin when it does not feed another device's nCE pin.

(3) For the MSEL pin settings, refer to Table 8–1 on page 8–2.

To configure an Arria V device, connect the device to a download cable, as shown in Figure 8–14.

Figure 8–14. Single Device PS Configuration Using an Altera Download Cable



Notes to Figure 8–14:

(1) Connect the pull-up resistor to the same supply voltage (V_{CCIO}) as the USB-Blaster, ByteBlaster II, EthernetBlaster, or EthernetBlaster II cable.

(2) You only need the pull-up resistors on DATA0 and DCLK 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 MAX II device, MAX V device, or microprocessor, you do not need the pull-up resistors on DATA0 and DCLK.

(3) In the USB-Blaster and ByteBlaster II cables, this pin is connected to nCE when you use it for AS programming. Otherwise, this pin is a no connect.

(4) For the MSEL pin settings, refer to Table 8–1 on page 8–2.

PS Multi-Device Configuration

You can configure multiple Arria V devices that are connected in a chain.

Pin Connections and Guidelines

Observe the following pin connections and guidelines for this configuration setup:

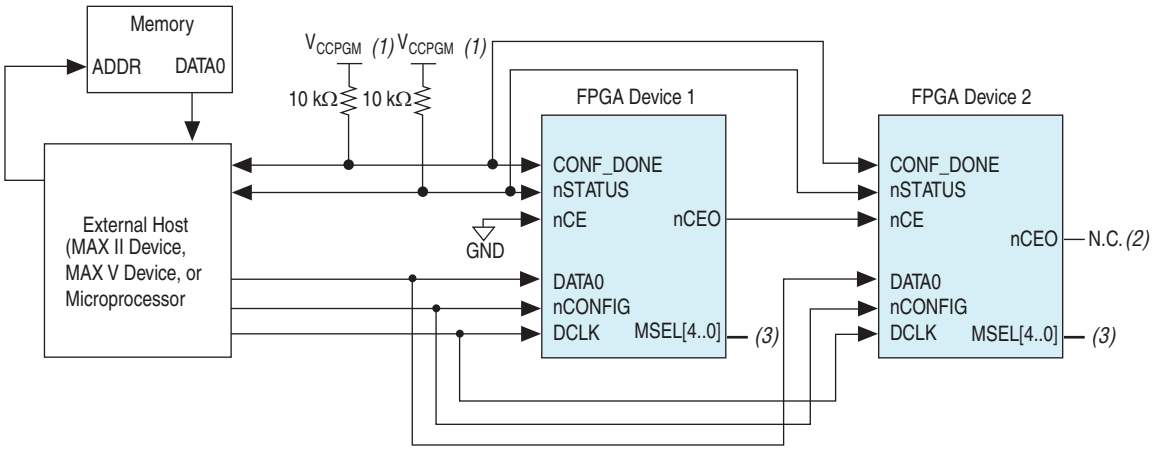
- Tie the following pins of all devices in the chain together:
  - nCONFIG
  - nSTATUS
  - DCLK
  - DATA0
  - CONF_DONE

  By tying the CONF_DONE and nSTATUS pins together, the devices initialize and enter user mode at the same time. If any device in the chain detects an error, configuration stops for the entire chain and you must reconfigure all the devices. For example, if the first device in the chain flags an error on the nSTATUS pin, it resets the chain by pulling its nSTATUS pin low.
- If you are configuring the devices in the chain using the same configuration data, the devices must be of the same package and density.

Using Multiple Configuration Data

To configure multiple Arria V devices in a chain using multiple configuration data, connect the devices to the external host as shown in Figure 8–15.

Figure 8–15. Multiple Device PS Configuration when Both Devices Receive Different Sets of Configuration Data



Notes to Figure 8–15:

(1) Connect the resistor to a power supply that provides an acceptable input signal for the Arria V device. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends powering up all the configuration system I/Os with V_{CCPGM}.

(2) You can leave the nCEO pin unconnected or use it as a user I/O pin when it does not feed another device's nCE pin.

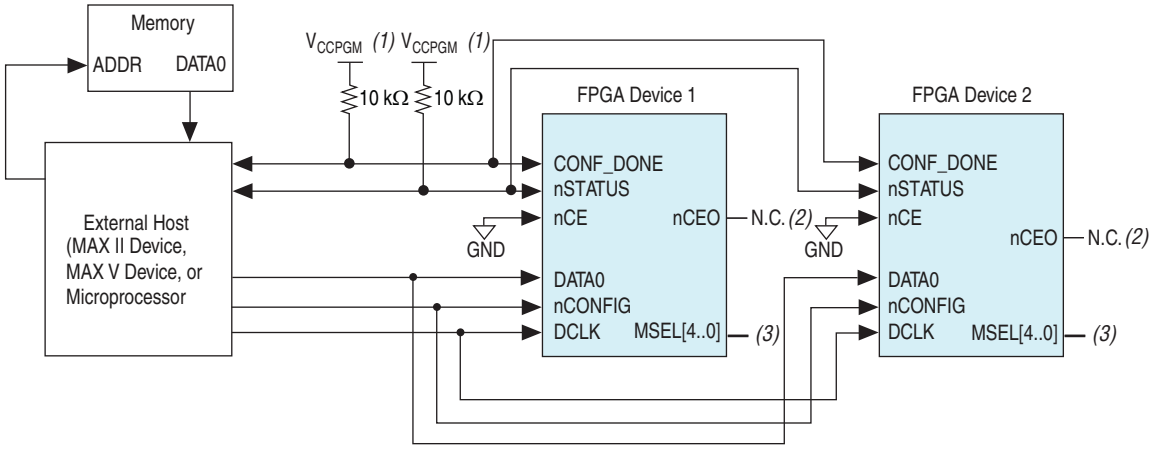
(3) For the MSEL pin settings, refer to Table 8–1 on page 8–2.

After a device completes configuration, its nCEO pin is released low to activate the nCE pin of the next device in the chain. Configuration automatically begins for the second device in one clock cycle.

Configuring Multiple Devices Using One Configuration Data

To configure multiple Arria V devices in a chain using one configuration data, connect the devices to an external host, as shown in Figure 8–16.

Figure 8–16. Multiple Device PS Configuration When Both Devices Receive the Same Set of Configuration Data



Notes to Figure 8–16:

(1) Connect the resistor to a power supply that provides an acceptable input signal for the Arria V device. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device and the external host. Altera recommends powering up all the configuration system I/Os with V_{CCPGM}.

(2) You can leave the nCEO pin unconnected or use it as a user I/O pin.

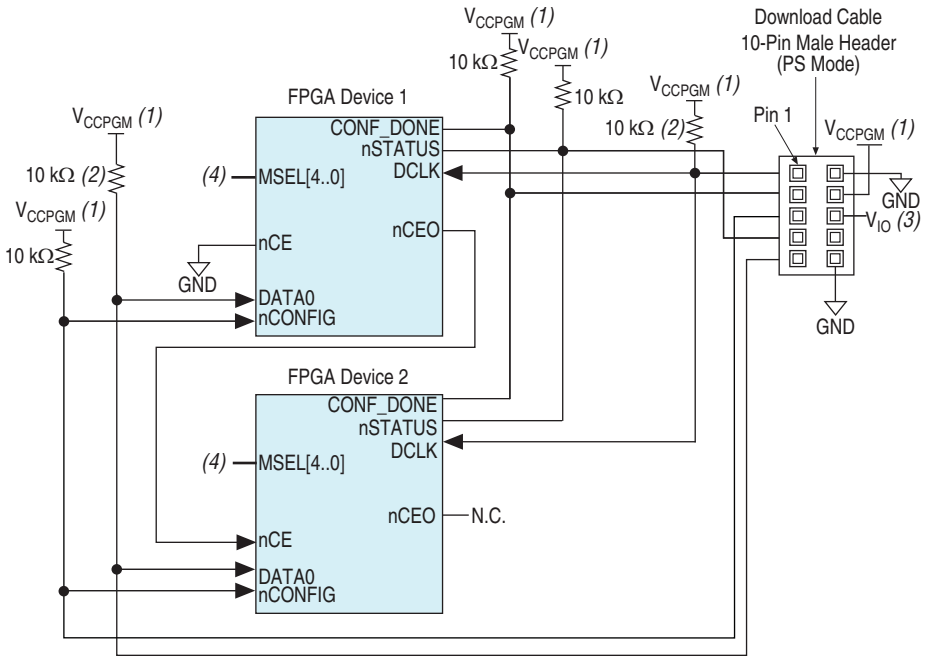
(3) For the MSEL pin settings, refer to Table 8–1 on page 8–2.

The nCE pins of the devices in the chain are connected to GND, allowing configuration for these devices to begin and end at the same time.

Using PC Host and Download Cable

To configure multiple Arria V devices, connect the devices to a download cable, as shown in Figure 8–17.

Figure 8–17. Multiple Device PS Configuration Using an Altera Download Cable



Notes to Figure 8–17:

(1) Connect the pull-up resistor to the same supply voltage (V_{CCIO}) as the USB-Blaster, ByteBlaster II, EthernetBlaster, or EthernetBlaster II cable.

(2) You only need the pull-up resistors on DATA0 and DCLK 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, you do not need the pull-up resistors on DATA0 and DCLK.

(3) In the USB-Blaster and ByteBlaster II cables, this pin is connected to nCE when you use it for AS programming. Otherwise, this pin is a no connect.

(4) For the MSEL pin settings, refer to Table 8–1 on page 8–2.

When a device completes configuration, its nCEO pin is released low to activate the nCE pin of the next device. Configuration automatically begins for the second device.

JTAG Configuration

In Arria V devices, JTAG instructions take precedence over other configuration schemes.

The Quartus II software generates an SRAM Object File (**.sof**) that you can use for JTAG configuration using a download cable in the Quartus II software programmer. Alternatively, you can use the JRunner software with **.rbf** or a JAM™ Standard Test and Programming Language (STAPL) Format File (**.jam**) or JAM Byte Code File (**.jbc**) with other third-party programmer tools.

For detailed information about JTAG configuration pins and JTAG secure mode, refer to "Device Configuration Pins" on page 8–6 and "JTAG Secure Mode" on page 8–38.

For more information about JTAG and the supported download cables, refer to the following documents:

- *AN 425: Using Command-Line Jam STAPL Solution for Device Programming*
- *JTAG Boundary-Scan Testing in Arria V Devices* chapter
- Programming Support for Jam STAPL Language page
- *USB-Blaster Download Cable User Guide*
- *ByteBlaster II Download Cable User Guide*
- *EthernetBlaster Communications Cable User Guide*
- *EthernetBlaster II Communications Cable User Guide*

JTAG Single-Device Configuration

To configure a single device in a JTAG chain, the programming software sets the other devices to the bypass mode. A device in a bypass mode transfers the programming data from the TDI pin to the TDO pin through a single bypass register. The configuration data is available on the TDO pin one clock cycle later.

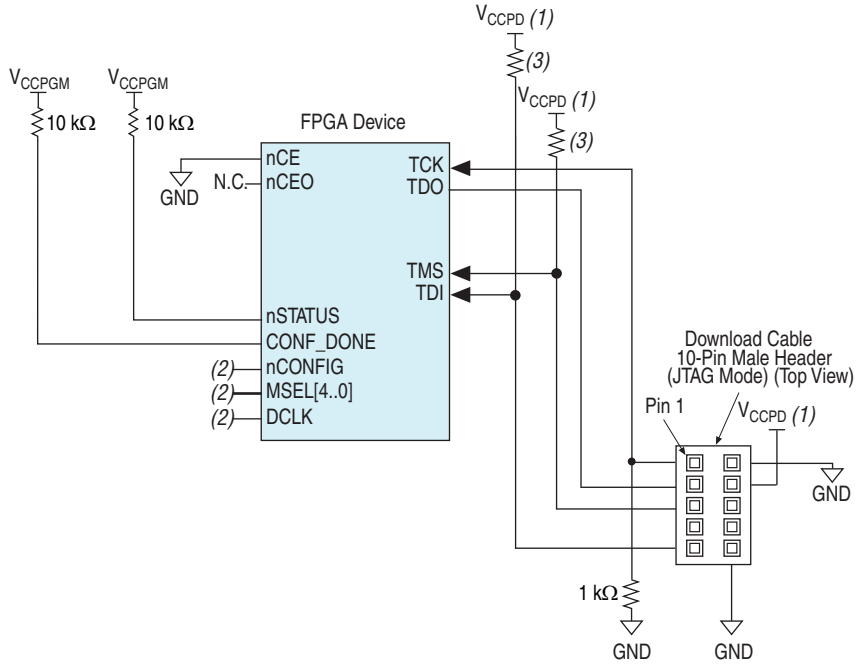
The Quartus II software can use the CONF_DONE pin to verify the completion of the configuration process through the JTAG port:

- CONF_DONE pin is low—indicates that configuration has failed.
- CONF_DONE pin is high—indicates that configuration was successful.

After the configuration data is transmitted serially using the JTAG TDI port, the TCK port is clocked an additional 1,222 cycles to perform device initialization.

To configure an Arria V device using a download cable, connect the device as shown in Figure 8–18.

Figure 8–18. JTAG Configuration of a Single Device Using a Download Cable



Notes to Figure 8–18:

(1) Connect the pull-up resistor V_{CCPD}.

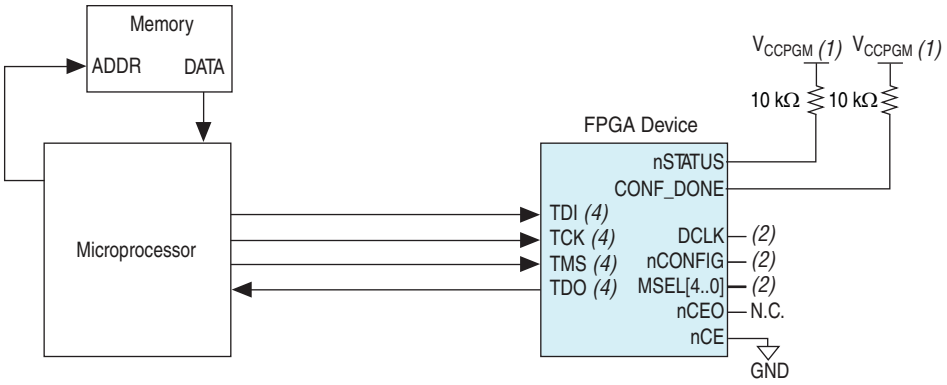
(2) If you only use the JTAG configuration, connect `nCONFIG` to V_{CCPGM} and `MSEL[4..0]` to GND. Pull `DCLK` either high or low, whichever is convenient on your board. If you are using JTAG in conjunction with another configuration scheme, connect `MSEL[4..0]`, `nCONFIG`, and `DCLK` based on the selected configuration scheme.

(3) The resistor value can vary from 1 kΩ to 10 kΩ. Perform signal integrity analysis to select the resistor value for your setup.

(4) You must connect `nCE` to GND or drive it low for successful JTAG configuration.

To configure an Arria V device using a microprocessor, connect the device as shown in Figure 8–19. You can use JRunner as your software driver.

Figure 8–19. JTAG Configuration of a Single Device Using a Microprocessor



Notes to Figure 8–19:

(1) Connect the pull-up resistor to a supply that provides an acceptable input signal for all Arria V devices in the chain. V_{CCPGM} must be high enough to meet the V_{IH} specification of the I/O on the device.

(2) If you use only the JTAG configuration, connect `nCONFIG` to V_{CCPGM} and `MSEL[4..0]` to GND. Pull `DCLK` high or low. If you are using JTAG together with another configuration scheme, set the `MSEL[4..0]` pins and tie `nCONFIG` and `DCLK` based on the selected configuration scheme.

(3) Connect `nCE` to GND or drive it low.

(4) The microprocessor must use the same I/O standard as V_{CCPD} to drive the JTAG pins.

For more information about the JRunner software driver, refer to *AN 414: The JRunner Software Driver: An Embedded Solution for PLD JTAG Configuration*.

JTAG Multi-Device Configuration

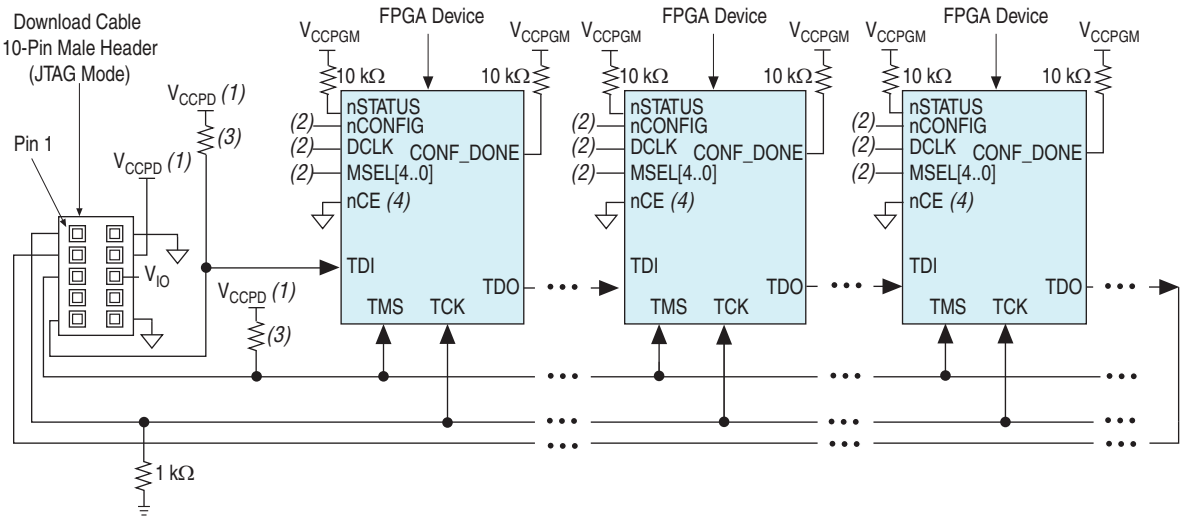
You can configure multiple devices in a JTAG chain.

Observe the following pin connections and guidelines for this configuration setup:

- Isolate the CONF_DONE and nSTATUS pins to allow each device to enter user mode independently.
- One JTAG-compatible header is connected to several devices in a JTAG chain. The number of devices in the chain is limited only by the drive capability of the download cable.
- If you have four or more devices in a JTAG chain, buffer the TCK, TDI, and TMS pins with an on-board buffer. You can also connect other Altera devices with JTAG support to the chain.
- JTAG-chain device programming is ideal when the system contains multiple devices or when testing your system using the JTAG boundary-scan testing (BST) circuitry.

Figure 8–20 shows a multi-device JTAG configuration.

Figure 8–20. JTAG Configuration of Multiple Devices Using a Download Cable



Notes to Figure 8–20:

(1) Connect the pull-up resistor V_{CCPD}.

(2) If you only use the JTAG configuration, connect nCONFIG to V_{CCPGM} and MSEL[4..0] to GND. Pull DCLK either high or low, whichever is convenient on your board. If you are using JTAG in conjunction with another configuration scheme, connect MSEL[4..0], nCONFIG, and DCLK based on the selected configuration scheme.

(3) The resistor value can vary from 1kΩ to 10 kΩ. Perform signal integrity analysis to select the resistor value for your setup.

(4) You must connect nCE to GND or drive it low for successful JTAG configuration.

For detailed information about combining the JTAG configuration with other configuration schemes, refer to *AN 656: Combining Multiple Configuration Schemes*.

CONFIG_IO JTAG Instruction

The CONFIG_IO JTAG instruction allows you to configure the I/O buffers using the JTAG port before or during device configuration. When you issue this instruction, it interrupts configuration and allows you to issue all JTAG instructions. Otherwise, you can only issue the BYPASS, IDCODE, and SAMPLE JTAG instructions.

You can use the CONFIG_IO JTAG instruction to interrupt configuration and perform board-level testing. After the board-level testing is completed, you must reconfigure your device. Use the following methods to reconfigure your device:

- JTAG interface—issue the PULSE_NCONFIG JTAG instruction.
- FPP, PS, or AS configuration scheme—pulse the nCONFIG pin low.

Configuration Data Compression

Arria V devices can receive compressed configuration bitstream and decompress the data in real-time during configuration. Preliminary data indicates that compression typically reduces the configuration file size by 30% to 55% depending on the design.

Decompression is supported in all configuration schemes except the JTAG configuration scheme.

You can enable compression before or after design compilation.

Enabling Compression Before Design Compilation

To enable compression before design compilation, follow these steps:

1. On the Assignments menu, click **Device**.
2. Select your Arria V device and then click **Device and Pin Options**.
3. In the **Device and Pin Options** window, select **Configuration** under the **Category** list and turn on **Generate compressed bitstreams**.

Enabling Compression After Design Compilation

To enable compression after design compilation, follow these steps:

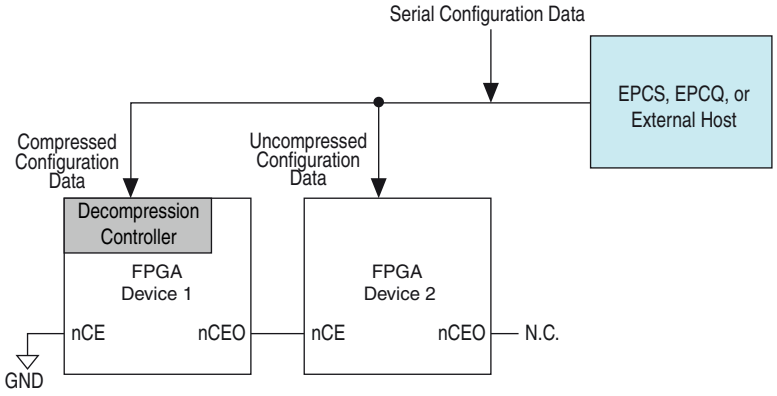
1. On the File menu, click **Convert Programming Files**.
2. Select the programming file type (**.pof**, **.sof**, **.hex**, **.hexout**, **.rbf**, or **.ttf**). For POF output files, select a configuration device.
3. Under the **Input files to convert** list, select **SOF Data**.
4. Click **Add File** and select an Arria V device **.sof**.
5. Select the name of the file you added to the **SOF Data** area and click **Properties**.
6. Turn on the **Compression** check box.

Using Compression in Multi-Device Configuration

Figure 8–21 shows a chain of two Arria V devices. Compression is only enabled for the first device.

This setup is supported by the AS or PS multi-device configuration only.

Figure 8–21. Compressed and Uncompressed Serial Configuration Data in the Same Configuration File



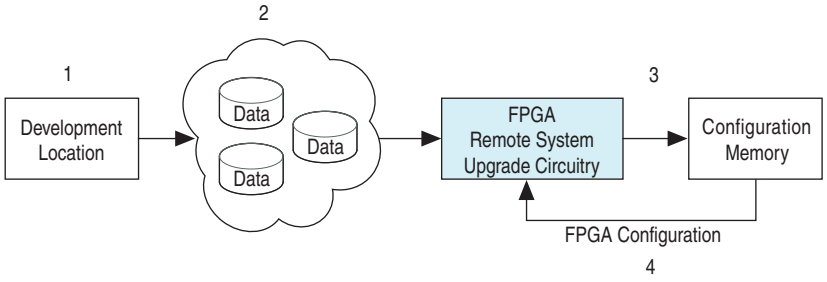
For the FPP configuration scheme, a combination of compressed and uncompressed configuration in the same multi-device configuration chain is not allowed because of the difference on the DCLK-to-DATA[] ratio.

Remote System Upgrades

Arria V devices contain dedicated remote system upgrade circuitry. You can use this feature to upgrade your system from a remote location.

Figure 8–22 shows the remote system upgrade block diagram.

Figure 8–22. Arria V Remote System Upgrade Block Diagram



You can design your system to manage remote upgrades of the application configuration images in the configuration device. The following list is the sequence of the remote system upgrade:

1. The logic (embedded processor or user logic) in the Arria V device receives a configuration image from a remote location. You can connect the device to the remote source using communication protocols such as TCP/IP, PCI, user datagram protocol (UDP), UART, or a proprietary interface.
2. The logic stores the configuration image in non-volatile configuration memory.
3. The logic starts reconfiguration cycle using the newly received configuration image.
4. When an error occurs, the circuitry detects the error, reverts to a safe configuration image, and provides error status to your design.

Configuration Images

Each Arria V device in your system requires one factory image. The factory image is a user-defined configuration image that contains logic to perform the following:

- Processes errors based on the status provided by the dedicated remote system upgrade circuitry.
- Communicates with the remote host, receive new application images, and store the images in the local non-volatile memory device.
- Determines the application image to load into the Arria V device.
- Enables or disables the user watchdog timer and load its time-out value.
- Instructs the dedicated remote system upgrade circuitry to start a reconfiguration cycle.

You can also create one or more application images for the device. An application image contains selected functionalities to be implemented in the target device.

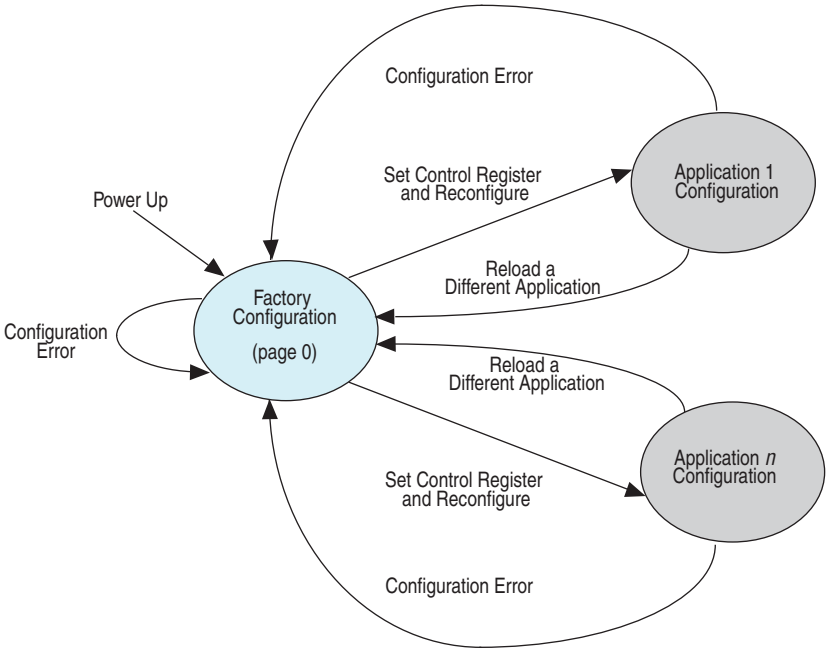
Store the images at the following locations in the EPCS or EPCQ devices:

- Factory configuration image—`PGM[23..0]` = 24'h000000 start address on the EPCS or EPCQ device.
- Application configuration image—any sector boundary. Altera recommends that you store only one image at one sector boundary.

Configuration Sequence

Figure 8–23 shows the transitions between the factory and application configurations in the remote update mode.

Figure 8–23. Transitions Between Configurations in Remote Update Mode



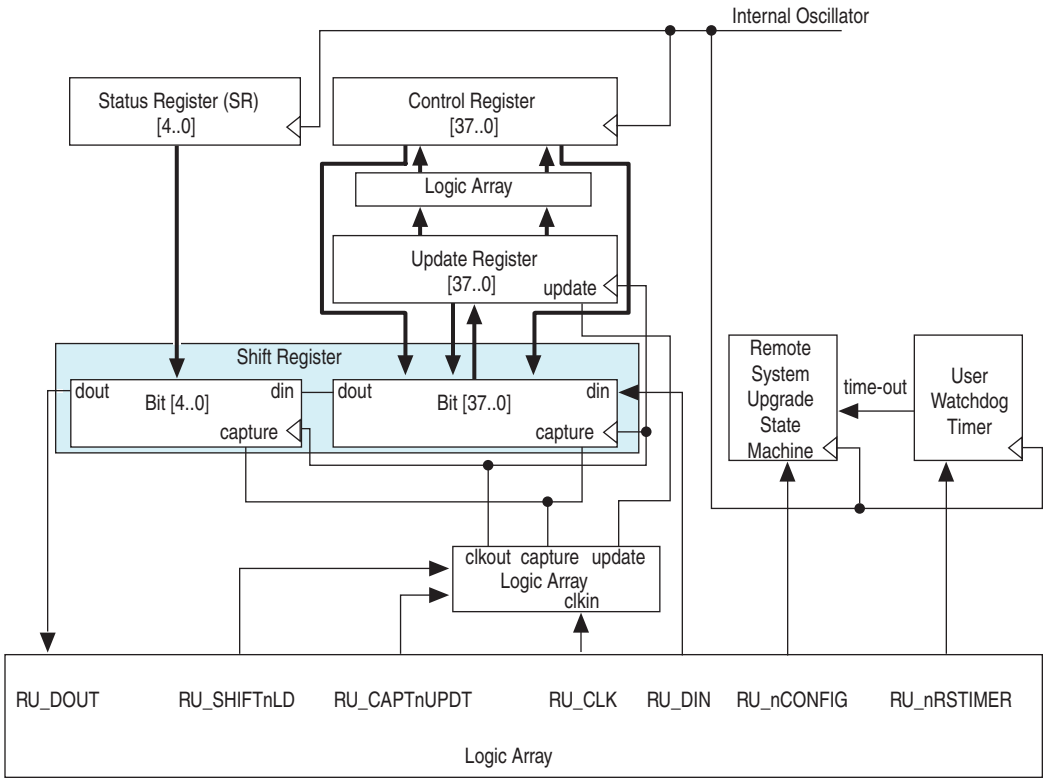
Refer to "Remote System Upgrade State Machine" on page 8–36 for a detailed description of this sequence.

Remote System Upgrade Circuitry

The remote system upgrade circuitry contains the remote system upgrade registers, watchdog timer, and a state machine that controls these components.

Figure 8–24 shows the remote system upgrade circuitry.

Figure 8–24. Remote System Upgrade Circuitry *(1)*



Note to Figure 8–24:

(1) If you are using the ALTREMOTE_UPDATE megafunction, the megafunction controls the `RU_DOUT`, `RU_SHIFTnLD`, `RU_CAPTnUPDT`, `RU_CLK`, `RU_DIN`, `RU_nCONFIG`, and `RU_nRSTIMER` signals internally to perform all the related remote system upgrade operations.

For more information about remote system upgrade circuitry timing specifications, refer to the *Arria V Device Datasheet*.

Enabling Remote System Upgrade Circuitry

To enable the remote system upgrade feature, select **Active Serial x1** or **Configuration Device** from the Configuration scheme list and **Remote** from the Configuration mode list in the **Configuration** page of the **Device and Pin Options** dialog box in the Quartus II software.

Enabling this feature automatically turns on the **Auto-restart configuration after error** option.

Altera-provided ALTREMOTE_UPDATE megafunction provides a memory-like interface to the remote system upgrade circuitry and handles the shift register read and write protocol in the Arria V device logic.

For more information about the ALTREMOTE_UPDATE megafunction, refer to the *Remote System Upgrade (ALTREMOTE_UPDATE) Megafunction User Guide*.

Remote System Upgrade Registers

Table 8–5 lists the remote system upgrade registers.

Table 8–5. Remote System Upgrade Registers

Register	Description
Shift	Accessible by the logic array and clocked by `RU_CLK`. ■ `Bits[4..0]`—Contents of the status register are shifted into these bits. ■ `Bits[37..0]`—Contents of the update and control registers are shifted into these bits.
Control	Refer to Table 8–6 for the fields description. This register is clocked by the 10-MHz internal oscillator. The contents of this register are shifted to the shift register for the user logic in the application configuration to read. When reconfiguration is triggered, this register is updated with the contents of the update register.
Update	Refer to Table 8–7 for the fields description. This register is clocked by `RU_CLK`. The factory configuration updates this register by shifting data into the shift register and issuing an update. When reconfiguration is triggered, the contents of the update register are written to the control register.
Status	After each reconfiguration, the remote system upgrade circuitry updates this register to indicate the event that triggered the reconfiguration. This register is clocked by the 10-MHz internal oscillator.

Control Register

Table 8–6 describes each field in the control register.

Table 8–6. Control Register Bits

Bit	Name	Reset Value (1)	Description
0	AnF	1'b0	Application not Factory bit. Indicates the configuration image type currently loaded in the device; **0** for factory image and **1** for application image. When this bit is **1**, the access to the control register is limited to read only and the watchdog timer is enabled. Factory configuration design must set this bit to **1** before triggering reconfiguration using an application configuration image.
1..24	PGM[0..23]	24'h000000	AS configuration start address (StAdd[23..0]).
25	Wd_en	1'b0	User watchdog timer enable bit. Set this bit to **1** to enable the watchdog timer.
26..37	Wd_timer[11..0]	12'b000000000000	User watchdog time-out value.

Note to Table 8–6:

(1) This is the default value after the device exits POR and during reconfiguration back to the factory configuration image.

Status Register

Table 8–7 describes each field in the status register.

Table 8–7. Status Register Bits

Bit	Name	Reset Value (1)	Description
0	CRC	1'b0	When set to **1**, indicates CRC error during application configuration.
1	nSTATUS	1'b0	When set to **1**, indicates that nSTATUS is asserted by an external device due to error.
2	Core_nCONFIG	1'b0	When set to **1**, indicates that reconfiguration has been triggered by the logic array of the device.
3	nCONFIG	1'b0	When set to **1**, indicates that nCONFIG is asserted.
4	Wd	1'b0	When set to **1**, indicates that the user watchdog time-out.

Note to Table 8–7:

(1) After the device exits POR and power-up, the status register content is 5'b00000.

Remote System Upgrade State Machine

The operation of the remote system upgrade state machine is as follows:

1. After power-up, the remote system upgrade registers are reset to **0** and the factory configuration image is loaded.
2. The user logic sets the AnF bit to **1** and the start address of the application image to be loaded. The user logic also writes the watchdog timer settings.
3. When the configuration reset (RU_nCONFIG) goes low, the state machine updates the control register with the contents of the update register, and triggers reconfiguration using the application configuration image.

4. If error occurs, the state machine falls back to the factory image. The control and update registers are reset to **0**, and the status register is updated with the error information.
5. After successful reconfiguration, the system stays in the application configuration.

User Watchdog Timer

The user watchdog timer prevents a faulty application configuration from stalling the device indefinitely. You can use the timer to detect functional errors when an application configuration is successfully loaded into the device. The timer is automatically disabled in the factory configuration; enabled in the application configuration.

If you do not want to use this feature in the application configuration, you need to turn off this feature by setting the `Wd_en` bit to **1'b0** in the update register during factory configuration user mode operation. You cannot disable this feature in the application 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^{17} cycles. The cycle time is based on the frequency of the user watchdog timer internal oscillator.

For more information about the operating range of the user watchdog internal oscillator's frequency, refer to the *Arria V Device Datasheet*.

The timer begins counting as soon as the application configuration enters user mode. When the timer expires, the remote system upgrade circuitry generates a time-out signal, updates the status register, and triggers the loading of the factory configuration image. To reset the time, assert `RU_nRSTIMER`.

Design Security

The Arria V design security feature supports the following capabilities:

- Enhanced built-in advanced encryption standard (AES) decryption block to support 256-bit key industry-standard design security algorithm (FIPS-197 Certified)
- Volatile and non-volatile key programming support
- Secure operation mode for both volatile and non-volatile key through tamper protection bit setting
- Limited accessible JTAG instruction during power-up in the JTAG secure mode
- Supports board-level testing
- Supports in-socket key programming for non-volatile key
- Available in all configuration schemes except JTAG
- Supports both remote system upgrades and compression features

The Arria V design security feature provides the following security protection for your designs:

- Security against copying—the security key is securely stored in the Arria V device and cannot be read out through any interface. In addition, as configuration file read-back is not supported in Arria V devices, your design information cannot be copied.
- Security against reverse engineering—reverse engineering from an encrypted configuration file is very difficult and time consuming because the Arria V configuration file formats are proprietary and the file contains millions of bits that require specific decryption.
- Security against tampering—After you set the tamper protection bit, the Arria V device can only accept configuration files encrypted with the same key. Additionally, programming through the JTAG interface and configuration interface is blocked.

When you use compression with the design security feature, the configuration file is first compressed and then encrypted using the Quartus II software. During configuration, the device first decrypts and then decompresses the configuration file.

When you use design security with Arria V devices in an FPP configuration scheme, it requires a different DCLK-to-DATA[] ratio.

JTAG Secure Mode

When you enable the tamper-protection bit, Arria V devices are in the JTAG secure mode after power-up. During this mode, many JTAG instructions are disabled. Arria V devices only allow mandatory JTAG 1149.1 instructions to be exercised. These JTAG instructions are SAMPLE/PRELOAD, BYPASS, EXTEST, and optional instructions such as IDCODE and SHIFT_EDERROR_REG.

To enable the access of other JTAG instructions such as USERCODE, HIGHZ, CLAMP, PULSE_NCONFIG, and CONFIG_IO, you must issue the UNLOCK instruction to deactivate the JTAG secure mode. You can issue the LOCK instruction to put the device back into JTAG secure mode. You can only issue both the LOCK and UNLOCK JTAG instructions during user mode.

For more information about JTAG binary instruction code related to the LOCK and UNLOCK instructions, refer to the *JTAG Boundary-Scan Testing in Arria V Devices* chapter.

Security Key Types

Arria V devices offer two types of keys—volatile and non-volatile. Table 8–8 lists the differences between the volatile key and non-volatile keys.

Table 8–8. Security Key Types

Key Types	Key Programmability	Power Supply for Key Storage	Programming Method
Volatile	■ Reprogrammable ■ Erasable	Required external battery, V_{CCBAT} (1)	On-board
Non-volatile	One-time programming	Does not require an external battery	On-board and in-socket programming (2)

Notes to Table 8–8:

(1) V_{CCBAT} is a dedicated power supply for volatile key storage. V_{CCBAT} continuously supplies power to the volatile register regardless of the on-chip supply condition.

(2) Third-party vendors offer in-socket programming.

Both non-volatile and volatile key programming offers protection from reverse engineering and copying. If you set the tamper-protection bit, the design is also protected from tampering.

You can perform key programming through the JTAG pins interface. Ensure that the nSTATUS pin is released high before any key-programming attempts.

To clear the volatile key, issue the KEY_CLR_VREG JTAG instruction. To verify the volatile key has been cleared, issue the KEY_VERIFY JTAG instruction.

For more information about the following topics, refer to the following documents:

- For detailed information about the KEY_CLR_VREG and KEY_VERIFY JTAG instructions, refer to the *JTAG Boundary-Scan Testing in Arria V Devices* chapter.
- For the V_{CCBAT} pin connection recommendations, refer to the *Arria V Device Family Pin Connection Guidelines*.
- For detailed information about battery specifications, refer to the *Arria V Device Datasheet*.

Security Modes

Table 8–9 lists the supported security modes.

Table 8–9. Supported Security Modes (Part 1 of 2)

Security Mode	Tamper protection Bit Setting	Device Accepts Unencrypted File	Device Accepts Encrypted File	Security Level
No key	—	Yes	No	—
Volatile Key	—	Yes	Yes	Secure
Volatile Key with Tamper Protection Bit Set	Set	No	Yes	Secure with tamper resistant

Table 8–9. Supported Security Modes (Part 2 of 2)

Security Mode	Tamper protection Bit Setting	Device Accepts Unencrypted File	Device Accepts Encrypted File	Security Level
Non-volatile Key	—	Yes	Yes	Secure
Non-volatile Key with Tamper Protection Bit Set	Set	No	Yes	Secure with tamper resistant

The use of unencrypted configuration bitstream in the volatile key and non-volatile key security modes is supported for board-level testing only.

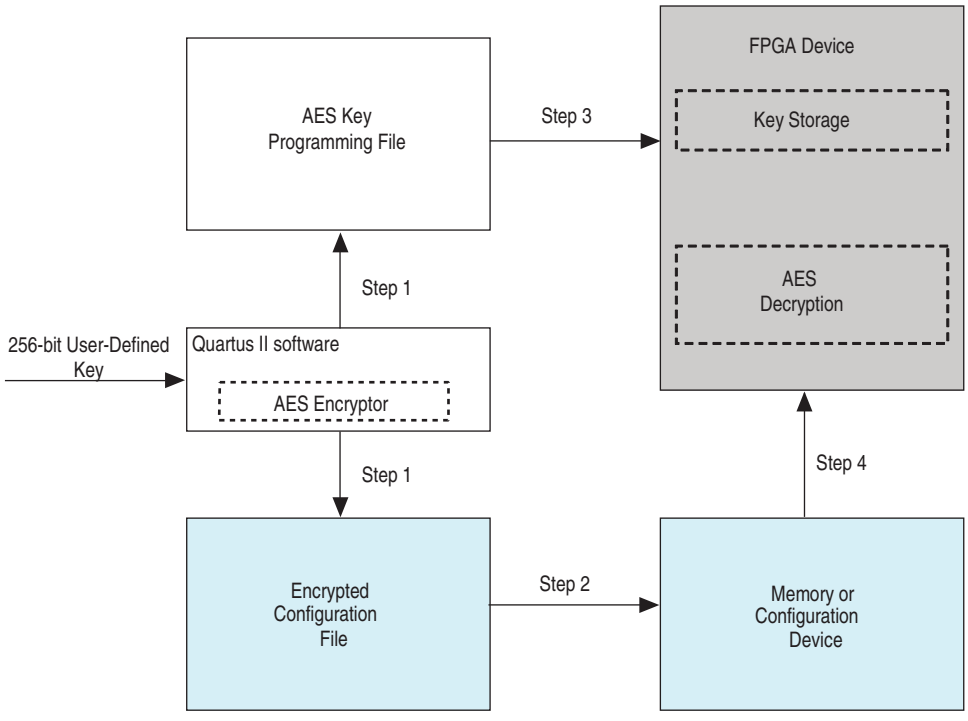
For the volatile key with tamper protection bit set security mode, Arria V devices do not accept the encrypted configuration file if the volatile key is erased. If the volatile key is erased and you want to reprogram the key, you must use the volatile key security mode.

Enabling the tamper protection bit disables the test mode in Arria V devices and disables programming through the JTAG interface. This process is irreversible and prevents Altera from carrying out failure analysis.

Design Security Implementation Steps

Figure 8–25 shows the design security implementation steps.

Figure 8–25. Design Security Implementation Steps



To carry out secure configuration, follow these steps:

1. The Quartus II software generates the design security key programming file and encrypts the configuration data using the user-defined 256-bit security key.
2. Store the encrypted configuration file in the external memory.
3. Program the AES key programming file into the Arria V device through a JTAG interface.
4. Configure the Arria V device. At the system power-up, the external memory device sends the encrypted configuration file to the Arria V device.

Document Revision History

Table 8–10 lists the revision history for this chapter.

Table 8–10. Document Revision History

Date	Version	Changes
June 2012	2.0	Restructured the chapter.
November 2011	1.1	Minor text edits.
October 2011	1.0	Initial release.

9. SEU Mitigation in Arria V Devices

AV-52009-2.0

This chapter describes the usage of the error detection cyclic redundancy check (CRC) feature in the Arria V device to mitigate single even upset (SEU) or soft errors.

This chapter contains the following topics:

Basic Description

Soft errors—changes in the CRAM's bit state caused by an ionizing particle—are uncommon in Altera devices. However, high-reliability applications that require the device to operate error-free may require that your designs account for these errors.

The error detection CRC feature in Arria V devices allows you to determine data integrity by comparing the CRC checksum values:

1. A precomputed data checksum value is stored with the data frame and transmitted together with the data.
2. Upon receiving the data frame, a new checksum value is calculated on the received data and compared against the transmitted checksum value.
3. If the compared checksum values are not identical, the system is alerted about the data transmission or data storage error.

Error Detection Features

The on-chip error detection CRC circuitry in Arria V devices allows you to perform the following operations:

- Detect and locate configuration data errors in configuration and user mode.
- Perform per frame error correction (internal scrubbing) in which the running device corrects single-bit or double-adjacent errors in the CRAM bits.
- Test the error detection functions by deliberately injecting an error through JTAG.

However, the error detection feature does not verify I/O buffers and memory blocks:

- The I/O buffers use flipflops as storage elements that are more resistant to soft errors.
- The memory logic array blocks (MLABs) and M10K memory blocks support parity bits that are used to check the contents of memory blocks for any error.
- The M10K blocks have built-in error correction code (ECC) support.

For more information about ECC support for the M10K blocks, refer to the *Internal Memory (RAM and ROM) User Guide*.

Using the error detection CRC feature does not impact the fitting or performance of the device.

Types of Error Detection

This section describes the two types of error detection CRC supported by the error detection circuitry in the Arria V devices.

Configuration Error Detection

The configuration error detection is a frame-based configuration error detection where a 16-bit precomputed CRC value is embedded in every configuration data frame.

During configuration, after a frame of data is loaded into the Arria V device, the precomputed CRC value is shifted into the CRC circuitry. A 16-bit CRC value for the loaded data frame is calculated and compared with the precomputed CRC values.

If the precomputed and calculated CRC values do not match, the `nSTATUS` pin is set to low. Unless the device detects an error, the configuration process continues until it is complete.

User Mode Error Detection

The user mode error detection detects data corruption in the configuration RAM (CRAM) cells caused by soft errors.

During configuration stage, a 32-bit CRC value is precomputed for each data frame and stored in the CRAM. In user mode, the error detection circuitry continually calculates 32-bit CRC check bits for each frame of the configured CRAM bits and compares it with the precomputed CRC values:

- If a CRAM bit error is not detected in a frame, the 32-bit signature is zero and the `CRC_ERROR` output signal is low.
- If a CRAM bit error is detected in a frame, the 32-bit signature is non-zero. The error detection circuitry pulls the `CRC_ERROR` pin high and starts searching for the error bit location.

Within a frame, the error detection circuitry can detect all single-, double-, triple-, quadruple-, and quintuple-bit errors.

The error detection circuitry reports the bit location and determines the type of error for single-bit errors or double-adjacent errors. The probability of other error patterns is very low and the reporting of bit location is not guaranteed.

The process of error detection continues until the device is reset by setting the nCONFIG signal low.

The probability of more than five CRAM bits being flipped by an SEU is very low. In general, the probability of detection for all error patterns is 99.9999%.

Error Detection Components

This section describes the error detection components in the Arria V devices.

Error Detection Pin

Whenever a CRC error is detected, the CRC_ERROR pin is driven high. Table 9–1 describes the CRC_ERROR pin.

Table 9–1. CRC_ERROR Pin Description

Pin Name	Pin Type	Description
CRC_ERROR	I/O, output, or output open-drain	This is an active-high signal indicating that the error detection circuit has detected errors in the configuration CRAM bits. This is an optional pin and is used if you enable the error detection CRC circuit. If you disable the error detection CRC circuit, it becomes a user I/O pin. When using the WYSIWYG function, you can route the crcerror port from the WYSIWYG atom to the dedicated CRC_ERROR pin or any user I/O. To route the crcerror port to a user I/O, insert a D-type flipflop in between the crcerror port and the I/O.

Error Detection Registers

This section focuses on the user mode CRC error detection.

There is one set of 32-bit registers in the error detection circuitry that stores the computed CRC signature. A non-zero value on the syndrome register causes the CRC_ERROR pin to be set high.

Figure 9–1 shows the error detection circuitry, syndrome registers, and error injection block.

Figure 9–1. Error Detection Circuitry, Syndrome Register, and Error Injection Block

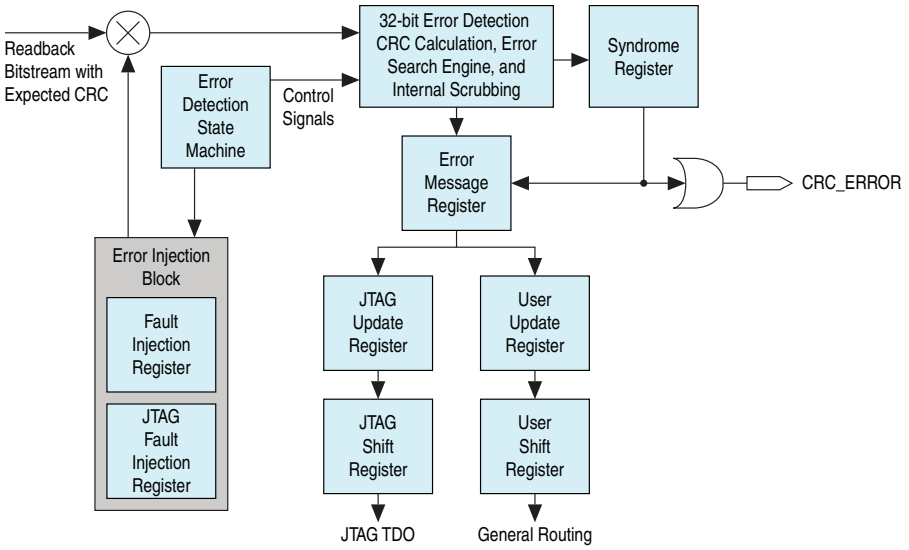


Table 9–2 lists the registers shown in Figure 9–1.

Table 9–2. Error Detection Registers (Part 1 of 2)

Register	Description
Syndrome Register	This 32-bit register contains the CRC signature of the current frame through the error detection verification cycle. The CRC_ERROR signal is derived from the contents of this register.
Error Message Register (EMR)	This 67-bit register contains information on the error type, location of the error, and the actual syndrome. The types of errors and location reported are single- and double-adjacent bit errors. The location bits for other types of errors are not identified by the EMR. The content of the register is shifted out through the SHIFT_EDERROR_REG JTAG instruction or to the core through the core interface.
JTAG Update Register	This 67-bit register is automatically updated with the contents of the EMR one cycle after this register content is validated. It includes a clock enable, which must be asserted prior to being sampled into the JTAG shift register. This requirement ensures that the JTAG update register is not being written into by the contents of the EMR at the same time that the JTAG shift register is reading its contents.
User Update Register	This 67-bit register is automatically updated with the contents of the EMR one cycle after this register content is validated. It includes a clock enable, which must be asserted prior to being sampled into the user shift register. This requirement ensures that the user update register is not being written into by the contents of the EMR at exactly the same time that the user shift register is reading its contents.
JTAG Shift Register	This 67-bit register is accessible by the JTAG interface and allows the contents of the JTAG update register to be sampled and read out by SHIFT_EDERROR_REG JTAG instruction.

Table 9–2. Error Detection Registers (Part 2 of 2)

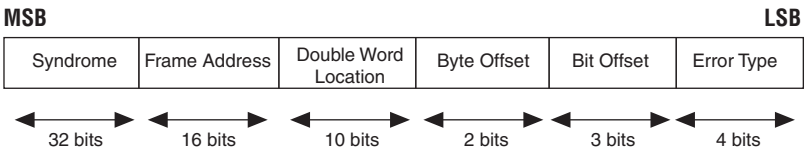
Register	Description
User Shift Register	This 67-bit register is accessible by the core logic and allows the contents of the user update register to be sampled and read by user logic.
JTAG Fault Injection Register	This 47-bit register is fully controlled by the `EDERROR_INJECT` JTAG instruction. This register holds the information of the error injection that you want in the bitstream.
Fault Injection Register	The content of the JTAG fault injection register is loaded into this 47-bit register when it is updated.

Error Message Register

The EMR stores the error type and location. The error detection circuitry updates and overwrites the EMR whenever an error is detected.

Figure 9–2 shows the structure of the EMR.

Figure 9–2. EMR Structure



The first four bits of the EMR represents the error type, as listed in Table 9–3.

Table 9–3. Error Type in the EMR

Error Type				Description
Bit 3	Bit 2	Bit 1	Bit 0	
0	0	0	0	No CRC error.
0	0	0	1	Location of a single-bit error is identified.
0	0	1	0	Location of a double-adjacent error is identified.
1	1	1	1	There is more than one error.
Other values				Reserved.

Storage Size for Error Detection

The error detection circuitry in Arria V devices uses a 32-bit CRC-ANSI standard (32-bit polynomial) as the CRC generator. The computed 32-bit CRC signature for each frame is stored in the CRAM.

Equation 9–1. Error Detection Storage Size

$$\text{Total Storage Size} = 32 \times \text{Number of Data Frames}$$

Error Detection Timing

This section describes the timing of the error detection blocks.

Minimum EMR Update Interval

The minimum interval between each update for the EMR depends on the device and the error detection clock frequency. Slowing down the error detection clock frequency slows down the error recovery time for the single event upset (SEU).

Table 9–4 lists the estimated minimum interval time between each update of the EMR in Arria V devices.

Table 9–4. Minimum EMR Update Interval in Arria V Devices —*Preliminary*

Variant	Member Code	Timing Interval (µs)
Arria V GX	A1	2.55
	A3	2.55
	A5	2.87
	A7	2.87
	B1	3.13
	B3	3.13
	B5	3.83
	B7	3.83
Arria V GT	C3	2.55
	C7	2.87
	D3	3.13
	D7	3.83
Arria V SX	B3	3.83
	B5	3.83
Arria V ST	D3	3.83
	D5	3.83

Error Detection Frequency

The error detection circuitry runs off an internal configuration oscillator with a divisor that sets the maximum frequency. Table 9–5 lists the minimum and maximum error detection frequencies in Arria V devices.

Table 9–5. Error Detection Frequencies Range in Arria V Devices

Error Detection Frequency	Maximum Error Detection Frequency	Minimum Error Detection Frequency	Valid Divisors (*n*)
100 MHz/2^n	100 MHz	390 kHz	0, 1, 2, 3, 4, 5, 6, 7, 8

You can set a lower clock frequency by specifying a division factor, ranging from 1 to 256, in the Quartus II software. Equation 9–2 shows that the division factor is 2 to the power of n, where n is from 0 to 8.

Equation 9–2. Error Detection Frequency

$$\text{Error Detection Frequency} = \frac{100\ \text{MHz}}{2^n}$$

The error detection frequency reflects the frequency of the error detection process for a frame because the CRC calculation in Arria V devices is on a per frame basis.

CRC Calculation Time

The CRC calculation time for the error detection circuitry to check all data frames depends on the device and the error detection clock frequency:

- The minimum CRC calculation time is calculated using the maximum error detection frequency with a divisor of 0.
- The maximum CRC calculation time is calculated using the minimum error detection frequency with a divisor of 8.

Table 9–6 lists the minimum and maximum estimated clock frequency time for each CRC calculation for Arria V devices.

Table 9–6. CRC Calculation Time for Arria V Devices —*Preliminary*

Variant	Member Code	Minimum Time (ms)	Maximum Time (s)
Arria V GX	A1	13.74	8.80
	A3	13.74	8.80
	A5	21.42	13.71
	A7	21.42	13.71
	B1	30.45	19.49
	B3	30.45	19.49
	B5	40.70	26.05
	B7	40.70	26.05
Arria V GT	C3	13.74	8.80
	C7	21.42	13.71
	D3	30.45	19.49
	D7	40.70	26.05
Arria V SX	B3	40.70	26.05
	B5	40.70	26.05
Arria V ST	D3	40.70	26.05
	D5	40.70	26.05

Using the Error Detection Feature

The Quartus II software supports the error detection CRC feature for Arria V devices. Enable this feature to generate the CRC_ERROR output signal to the optional, dual-purpose CRC_ERROR pin.

Enabling the User Mode Error Detection

To enable the error detection feature using CRC when the device transitions into user mode in a project that uses a Arria V device, follow these steps:

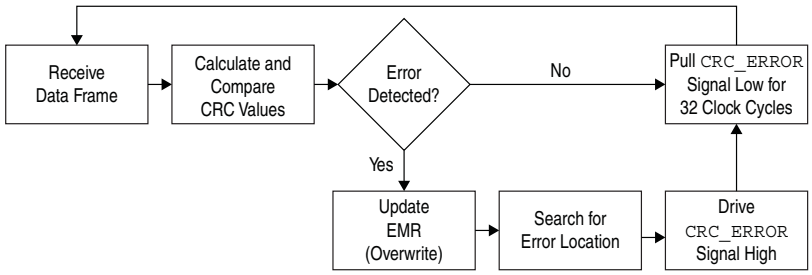
1. On the Assignments menu, click **Device**.
2. In the **Device** dialog box, click **Device and Pin Options**.
3. In the **Category** list, click **Error Detection CRC**.
4. Turn on **Enable Error Detection CRC_ERROR pin**.
5. To set the CRC_ERROR pin as output open-drain, turn on **Enable open drain on CRC_ERROR pin**. To set this pin as an output, turn this option off.
6. To enable the on-chip error correction feature, turn on **Enable internal scrubbing**.
7. In the **Divide error check frequency by** list, select a valid divisor.
8. Click **OK**.

Error Detection Process

With the error detection feature enabled, the CRC error detection process activates automatically when the Arria V device enters user mode, and continues to run until the device is reset.

If an error is detected within a frame, the error detection circuitry performs several steps, as shown in Figure 9–3.

Figure 9–3. Error Detection Process



The error detection circuitry continues to calculate the 32-bit error detection CRC value and 32-bit signatures for the next frame of data regardless of whether an error has occurred in the current frame or not. You must monitor the CRC_ERROR signal and take appropriate actions if a CRC error occurs.

Because the EMR is overwritten whenever an error is detected, you must shift out the previous error location and message from the EMR within one frame of the CRC verification. You can start to shift out the error message on each rising edge of the CRC_ERROR signal.

Reading the Error Location Bit Through JTAG

As an alternative to the core interface, you can also read the error bit location through JTAG. The JTAG interface reads the 32-bit error detection CRC result for the first frame and shifts the 32-bit error detection CRC bits to the 32-bit error detection CRC storage registers for test purposes.

To shift out the erroneous bits from the EMR, use the SHIFT_EDERROR_REG JTAG instruction, as listed in Table 9–7.

Table 9–7. SHIFT_EDERROR_REG JTAG Instruction

JTAG Instruction	Instruction Code	Description
SHIFT_EDERROR_REG	00 0001 0111	The JTAG instruction connects the EMR to the JTAG pin in the error detection block between the TDI and TDO pins.

Recovering From CRC Errors

The system that hosts the Arria V device must control device reconfiguration.

To recover from a CRC error detected on the CRC_ERROR pin, strobe the nCONFIG signal low. The system waits for a safe time and then performs reconfiguration. After the device reconfiguration rewrites the data bit with the correct value, the device functions correctly.

Testing the Error Detection Block

This section describes the method to deliberately inject errors into the configuration process for testing and verification purpose.

Error Detection Instruction

To test the capability of the error detection block, execute the EDERROR_INJECT JTAG instruction in user mode, as listed in Table 9–8. The EDERROR_INJECT JTAG instruction can change the content of the 47-bit JTAG fault injection register.

Table 9–8. EDERROR_INJECT JTAG Instruction

JTAG Instruction	Instruction Code	Description
EDERROR_INJECT	00 0001 0101	This instruction controls the 47-bit JTAG fault injection register used for error injection.

You can inject errors only in the first data frame. However, you can monitor the error information at any time.

You can introduce a single error or double errors adjacent to each other into the configuration memory. This method provides an extra way to facilitate design verification and system fault tolerance characterization.

JTAG Fault Injection Register

Use the JTAG fault injection register with the EDERROR_INJECT JTAG instruction to flip the readback bits. The Arria V device is then forced into the error test mode.

Table 9–9 lists the implementation of the fault injection register and describes the error injection.

Table 9–9. Fault Injection Register

Description	Bit[46…43]					Bit[42…32]	Bit[31…0]
	Error Type				Error Injection Type	Byte Location of the Injected Error	Error Byte Value
	Bit[46]	Bit[45]	Bit[44]	Bit[43]			
Content	0	0	0	0	No error	Depicts the location of the injected error in the first data frame.	Depicts the location of the bit error and corresponds to the error injection type selection.
	0	0	0	1	Single error		
	0	0	1	0	Double-adjacent error		
	Others				Reserved		

For more information about the JTAG fault injection register, refer to "Error Detection Registers" on page 9–3.

Altera recommends reconfiguring the device after the test is completed.

Automating the Testing Process

To automate the testing and verification process, create a Jam™ file (**.jam**).

By using the **.jam**, you can verify the CRC functionality in-system and on-the-fly without reconfiguring the device. You can then switch to the CRC circuit to check for real errors induced by an SEU.

Document Revision History

Table 9–10 lists the revision history for this chapter.

Table 9–10. Document Revision History

Date	Version	Changes
June 2012	2.0	■ Restructured the chapter. ■ Added the "Basic Description", "Error Detection Features", "Types of Error Detection", "Error Detection Components", "Using the Error Detection Feature", and "Testing the Error Detection Block" sections. ■ Updated Table 9–4, Table 9–5, and Table 9–6.
November 2011	1.1	Restructured chapter.
May 2011	1.0	Initial release.

10. JTAG Boundary-Scan Testing in Arria V Devices

AV-52010-2.0

This chapter describes the boundary-scan test (BST) features in Arria® V devices.

This chapter includes the following sections:

BST Operation Control

Arria V devices support IEEE Std. 1149.1 BST. You can perform BST on Arria V devices before, after, and during configuration.

IDCODE

The `IDCODE` is unique for each Arria V device. Use this code to identify the devices in a JTAG chain.

Table 10–1 lists the `IDCODE` information for Arria V devices.

Table 10–1. IDCODE Information for Arria V Devices (Part 1 of 2)

Variant	Member Code	IDCODE (32 Bits)			
		Version (4 Bits)	Part Number (16 Bits)	Manufacture Identity (11 Bits)	LSB (1 Bit)
Arria V GX	A1	0000	0010 1010 0001 0001	000 0110 1110	1
	A3	0000	0010 1010 0000 0001	000 0110 1110	1
	A5	0000	0010 1010 0001 0010	000 0110 1110	1
	A7	0000	0010 1010 0000 0010	000 0110 1110	1
	B1	0000	0010 1010 0001 0011	000 0110 1110	1
	B3	0000	0010 1010 0000 0011	000 0110 1110	1
	B5	0000	0010 1010 0001 0110	000 0110 1110	1
	B7	0000	0010 1010 0000 0110	000 0110 1110	1

Table 10–1. IDCODE Information for Arria V Devices (Part 2 of 2)

Variant	Member Code	IDCODE (32 Bits)			
		Version (4 Bits)	Part Number (16 Bits)	Manufacture Identity (11 Bits)	LSB (1 Bit)
Arria V GT	C3	0000	0010 1010 0000 0001	000 0110 1110	1
	C7	0000	0010 1010 0000 0010	000 0110 1110	1
	D3	0000	0010 1010 0000 0011	000 0110 1110	1
	D7	0000	0010 1010 0000 0110	000 0110 1110	1
Arria V SX	B3	0000	0010 1101 0001 0011	000 0110 1110	1
	B5	0000	0010 1101 0000 0011	000 0110 1110	1
Arria V ST	D3	0000	0010 1101 0001 0011	000 0110 1110	1
	D5	0000	0010 1101 0000 0011	000 0110 1110	1

Supported JTAG Instruction

Table 10–2 lists the JTAG instructions that are supported by Arria V devices.

Table 10–2. JTAG Instructions Supported by Arria V Devices (Part 1 of 2)

JTAG Instruction	Instruction Code	Description
SAMPLE/PRELOAD	00 0000 0101	■ Allows you to capture and examine a snapshot of signals at the device pins during normal device operation and permits an initial data pattern to be an output at the device pins. ■ Use this instruction to preload the test data into the update registers prior to loading the EXTEST instruction. ■ Used by the SignalTap™ II Embedded Logic Analyzer.
EXTEST	00 0000 1111	■ Allows you to test the external circuit and board-level interconnects by forcing a test pattern at the output pins, and capturing the test results at the input pins. Forcing known logic high and low levels on output pins allows you to detect opens and shorts at the pins of any device in the scan chain. ■ The high-impedance state of EXTEST is overridden by bus hold and weak pull-up resistor features.
BYPASS	11 1111 1111	Places the 1-bit bypass register between the TDI and TDO pins. During normal device operation, the 1-bit bypass register allows the BST data to pass synchronously through the selected devices to adjacent devices.
USERCODE	00 0000 0111	■ Examines the user electronic signature (UES) within the devices along a JTAG chain. ■ Selects the 32-bit USERCODE register and places it between the TDI and TDO pins to allow serial shifting of USERCODE out of TDO. ■ The UES value is set to default value before configuration and is only user-defined after the device is configured.

Table 10–2. JTAG Instructions Supported by Arria V Devices (Part 2 of 2)

JTAG Instruction	Instruction Code	Description
IDCODE	00 0000 0110	■ Identifies the devices in a JTAG chain. If you select IDCODE, the device identification register is loaded with the 32-bit vendor-defined identification code. ■ Selects the IDCODE register and places it between the TDI and TDO pins to allow serial shifting of IDCODE out of TDO. ■ IDCODE is the default instruction at power up and in the TAP RESET state. Without loading any instructions, you can go to the SHIFT_DR state and shift out the JTAG device ID.
HIGHZ	00 0000 1011	■ Sets all user I/O pins to an inactive drive state. ■ Places the 1-bit bypass register between the TDI and TDO pins. During normal operation, the 1-bit bypass register allows the BST data to pass synchronously through the selected devices to adjacent devices while tri-stating all I/O pins until a new JTAG instruction is executed. ■ If you are testing the device after configuration, the programmable weak pull-up resistor or the bus hold feature overrides the HIGHZ value at the pin.
CLAMP	00 0000 1010	■ Places the 1-bit bypass register between the TDI and TDO pins. During normal operation, the 1-bit bypass register allows the BST data to pass synchronously through the selected devices to adjacent devices while holding the I/O pins to a state defined by the data in the boundary-scan register. ■ If you are testing the device after configuration, the programmable weak pull-up resistor or the bus hold feature overrides the CLAMP value at the pin. The CLAMP value is the value stored in the update register of the boundary-scan cell (BSC).
PULSE_NCONFIG (1)	00 0000 0001	Emulates pulsing the nCONFIG pin low to trigger reconfiguration even though the physical pin is not affected.
CONFIG_IO (1)	00 0000 1101	Allows I/O reconfiguration (after or during reconfigurations) through the JTAG ports using I/O configuration shift register (IOCSR) for JTAG testing. You can issue the CONFIG_IO instruction only after the nSTATUS pin goes high.
LOCK (1)	01 1111 0000	Put the device in JTAG secure mode. In this mode, only BYPASS, SAMPLE/PRELOAD, EXTEST, IDCODE, and SHIFT_EDERROR_REG and UNLOCK instructions are supported. This instruction can only be accessed through JTAG core access in User mode. It cannot be accessed through external JTAG pins in Test or User mode.
UNLOCK (1)	11 0011 0001	Release the device from the JTAG secure mode to enable access to all other JTAG instructions. This instruction can only be accessed through JTAG core access in User mode. It cannot be accessed through external JTAG pins in Test or User mode.
KEY_CLR_VREG	00 0010 1001	Clears the non-volatile key.
KEY_VERIFY	00 0001 0011	Verifies the non-volatile key has been cleared.

Note to Table 10–2:

(1) For more information about this instruction mode, refer to the *Configuration, Design Security, and Remote System Upgrades in Arria V Devices* chapter.

If the device is in a reset state and the nCONFIG or nSTATUS signal is low, the device IDCODE might not be read correctly. To read the device IDCODE correctly, you must issue the IDCODE JTAG instruction only when the nCONFIG and nSTATUS signals are high.

JTAG Secure Mode

If you enable the tamper-protection bit, the Arria V device is in JTAG secure mode after power up. In the JTAG secure mode, only the BYPASS, SAMPLE/PRELOAD, EXTEST, IDCODE, SHIFT_EDERROR_REG, and UNLOCK instructions are supported by the JTAG pins. Issue the UNLOCK JTAG instruction to enable support for other JTAG instructions.

JTAG Private Instruction

CAUTION

Never invoke the following instruction codes. These instructions can damage and render the device unusable:

- 1100010000
- 0011001001
- 0000101011
- 1100010111
- 0111100000

I/O Voltage for JTAG Operation

An Arria V device operating in IEEE Std. 1149.1 BST mode uses four dedicated JTAG pins—TDI, TDO, TMS, and TCK. Arria V devices do not support the optional TRST pin. The TCK pin has an internal weak pull-down resistor, while the TDI and TMS pins have internal weak pull-up resistors. The 3.3-, 3.0-, or 2.5-V_{CCPD} supply of I/O bank 3A powers the TDO, TDO, TMS, and TCK pins. All user I/O pins are tri-stated during JTAG configuration.

The JTAG chain supports several different devices. Use the supported TDO and TDI voltage combinations listed in Table 10–3 if the JTAG chain contains devices that have different V_{CCIO} levels. The output voltage level of the TDO pin must meet the specification of the TDI pin it drives.

Table 10–3. Supported TDO and TDI Voltage Combinations (Part 1 of 2)

Device	TDI Input Buffer Power (V)	Arria V TDO V_{CCPD}		
		V_{CCPD} = 3.3 V (1)	V_{CCPD} = 3.0 V (1)	V_{CCPD} = 2.5 V (2)
Arria V	V_{CCPD} = 3.3	✓	✓	✓
	V_{CCPD} = 3.0	✓	✓	✓
	V_{CCPD} = 2.5	✓	✓	✓

Table 10–3. Supported TDO and TDI Voltage Combinations (Part 2 of 2)

Device	TDI Input Buffer Power (V)	Arria V TDO V_{CCPD}		
		V_{CCPD} = 3.3 V (1)	V_{CCPD} = 3.0 V (1)	V_{CCPD} = 2.5 V (2)
Non-Arria V	V_{CC} = 3.3	✓ (3)	✓ (4)	✓ (5)
	V_{CC} = 2.5	✓ (3)	✓ (4)	✓ (5)
	V_{CC} = 1.8	✓ (3)	✓ (4)	✓ (5)
	V_{CC} = 1.5	✓ (3)	✓ (4)	✓ (5)

Notes to Table 10–3:

(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) Input buffer must be 3.3-V tolerant.
(4) Input buffer must be 3.0-V tolerant.
(5) Input buffer must be 2.5-V tolerant.

Enabling and Disabling IEEE Std. 1149.1 BST Circuitry

The IEEE Std. 1149.1 BST circuitry is enabled after the Arria V device powers up. However for Arria V SoC FPGAs, you must power up both HPS and FPGA to perform boundary-scan testing.

To ensure that you do not inadvertently enable the IEEE Std. 1149.1 circuitry when it is not required, disable the circuitry permanently with pin connections as listed in Table 10–4.

Table 10–4. Pin Connections to Permanently Disable the IEEE Std. 1149.1 Circuitry for Arria V Devices

JTAG Pins (1)	Connection for Disabling
TMS	V_{CCPD} supply of Bank 3A
TCK	GND
TDI	V_{CCPD} supply of Bank 3A
TDO	Leave open

Note to Table 10–4:

(1) The JTAG pins are dedicated. Software option is not available to disable JTAG in Arria V devices.

Performing BST

You can issue BYPASS, IDCODE, and SAMPLE JTAG instructions before, after, or during configuration without having to interrupt configuration.

To issue other JTAG instructions, follow these guidelines:

- To perform testing before configuration, hold the nCONFIG pin low.
- To perform BST during configuration, issue CONFIG_IO JTAG instruction to interrupt configuration. While configuration is interrupted, you can issue other JTAG instructions to perform BST. After BST is completed, issue the PULSE_CONFIG JTAG instruction or pulse nCONFIG low to reconfigure the device.

The chip-wide reset (DEV_CLRn) and chip-wide output enable (DEV_OE) pins on Arria V devices do not affect JTAG boundary-scan or configuration operations. Toggling these pins does not disrupt BST operation (other than the expected BST behavior).

If you design a board for JTAG configuration of Arria V devices, consider the connections for the dedicated configuration pins.

For more information about pin connections, refer to the *Arria V Device Family Pin Connection Guidelines*.

For more information about JTAG configuration using the IEEE Std.1149.1 circuitry, refer to the *Configuration, Design Security, and Remote System Upgrades in Arria V Devices* chapter.

For more information about JTAG configuration timing, refer to the *Arria V Device Datasheet*.

Guidelines for IEEE Std. 1149.1 Boundary-Scan Testing

Consider the following guidelines when you perform BST with IEEE Std. 1149.1 devices:

- If the "10..." pattern does not shift out of the instruction register through 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 01100 code to the TMS pin.
  - Check the connections to the VCC, GND, JTAG, and dedicated configuration pins on the device.
- Perform a SAMPLE/PRELOAD test cycle before the first EXTEST test cycle to ensure that known data is present at the device pins when you enter EXTEST mode. If the OEJ update register contains 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 in-circuit reconfiguration because EXTEST is not supported during in-circuit reconfiguration. To perform testing, wait for the configuration to complete or issue the CONFIG_IO instruction to interrupt configuration.
- After configuration, you cannot test any pins in a differential pin pair. To perform BST after configuration, edit and redefine the BSC group that correspond to these differential pin pairs as an internal cell.

  For more information about the BSC group definitions, refer to the IEEE 1149.1 BSDL Files page on the Altera website.

IEEE Std. 1149.1 BST Architecture

This section describes the IEEE Std. 1149.1 BST architecture.

IEEE Std. 1149.1 BST Functionality

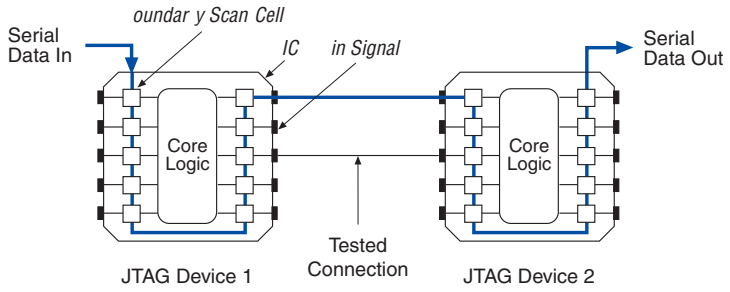
The IEEE Std. 1149.1 BST architecture tests pin connections without using physical test probes and captures functional data while a device is operating normally.

BSCs in a device can force signals onto pins or capture data from pins or logic array signals:

- Forced test data is serially shifted into the BSCs.
- Captured data is serially shifted out and externally compared with the expected results.

Figure 10–1 shows the BST architecture.

Figure 10–1. IEEE Std. 1149.1 Boundary-Scan Testing



For more information about JTAG configuration, refer to the *Configuration, Design Security, and Remote System Upgrades in Arria V Devices* chapter.

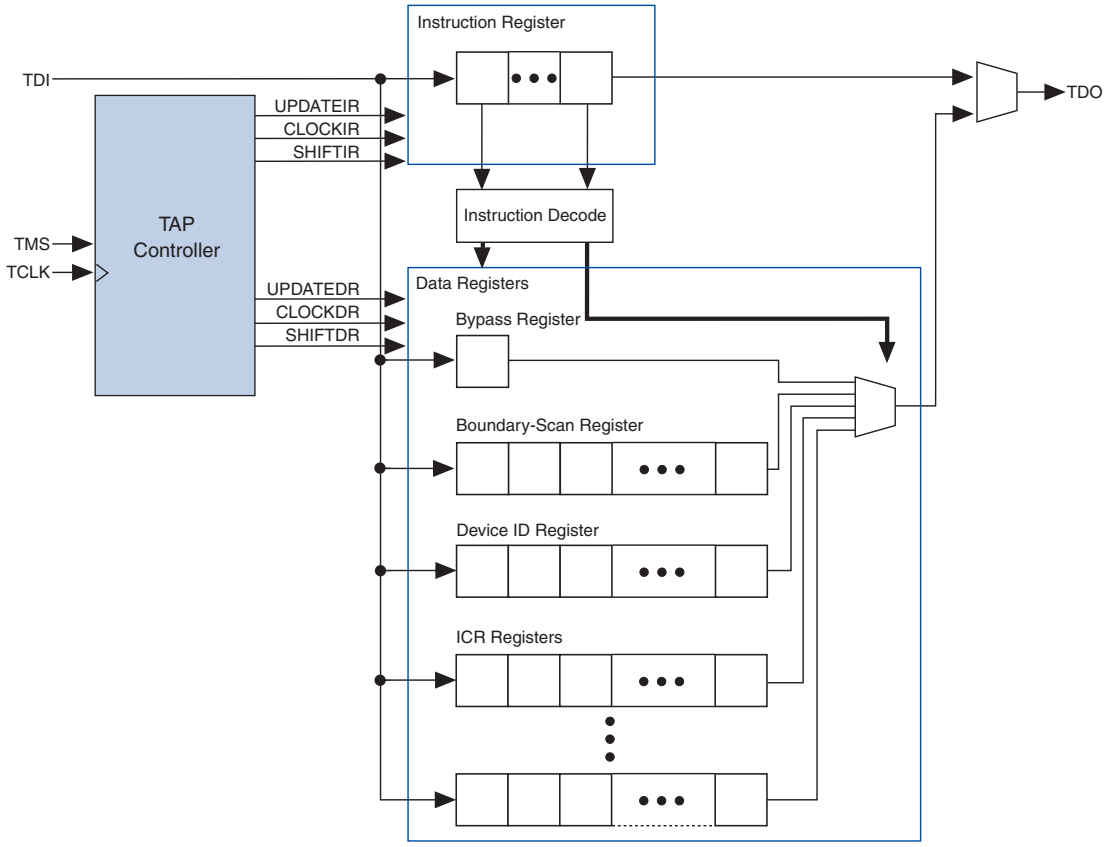
IEEE Std. 1149.1 BST Circuitry Registers

The IEEE Std. 1149.1 BST circuitry requires the following registers:

- Instruction register—determines the action to be performed and the data register to be accessed.
- Bypass register—a 1-bit-long data register that provides a minimum-length serial path between `TDI` and `TDO`.
- Boundary-scan register—a shift register composed of all the boundary-scan cells (BSCs) of the device.

Figure 10–2 shows a functional model of the IEEE Std. 1149.1 BST circuitry.

Figure 10–2. IEEE Std. 1149.1 BST Circuitry



IEEE Std. 1149.1 BST Pin Function

Table 10–5 lists the functions of the IEEE Std. 1149.1 BST pins.

Table 10–5. IEEE Std. 1149.1 Pin Descriptions

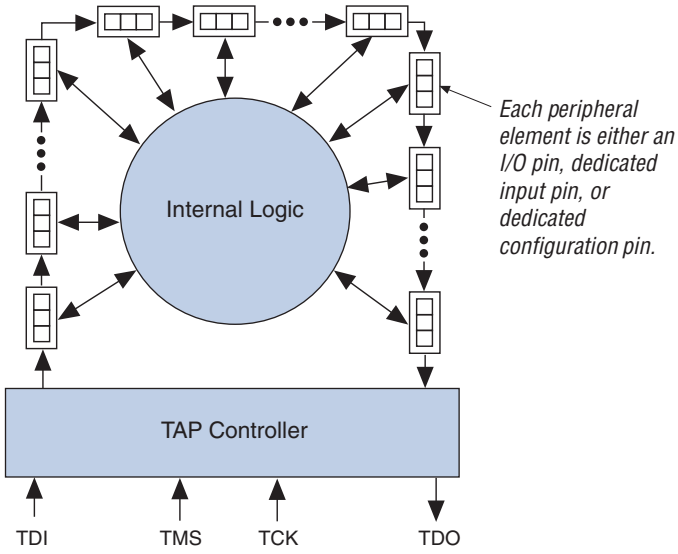
Pin	Description	Function
TDI	Test data input	Serial input pin for instructions as well as testing and programming data. Data is shifted in on the rising edge of TCK. Provides data to the instruction register and generates the control logic for the data registers.
TDO	Test data output	Serial data output pin for instructions as well as testing 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 TAP controller state machine. Transitions within the state machine occur at the rising edge of TCK. You must set up TMS 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 that operates the TAP controller. Some operations occur at the rising edge, while others occur at the falling edge.

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 V I/O pins. You can use the boundary-scan register to test external pin connections or to capture internal data.

Figure 10–3 shows how test data is serially shifted around the periphery of the IEEE Std. 1149.1 device.

Figure 10–3. Boundary-Scan Register



Boundary-Scan Cells of a Arria V Device I/O Pin

The Arria V device 3-bit BSC consists of the following registers:

- Capture registers—Connect to internal device data through the OUTJ, OEJ, and PIN_IN signals
- Update registers—Connect to external data through the PIN_OUT and PIN_OE signals.

The TAP controller generates the global control signals for the IEEE Std. 1149.1 BST registers (shift, clock, and update) internally. A decode of the instruction register generates the MODE signal.

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 10–4 shows the user I/O BSC for Arria V devices.

Figure 10–4. User I/O BSC with IEEE Std. 1149.1 BST Circuitry for Arria V Devices

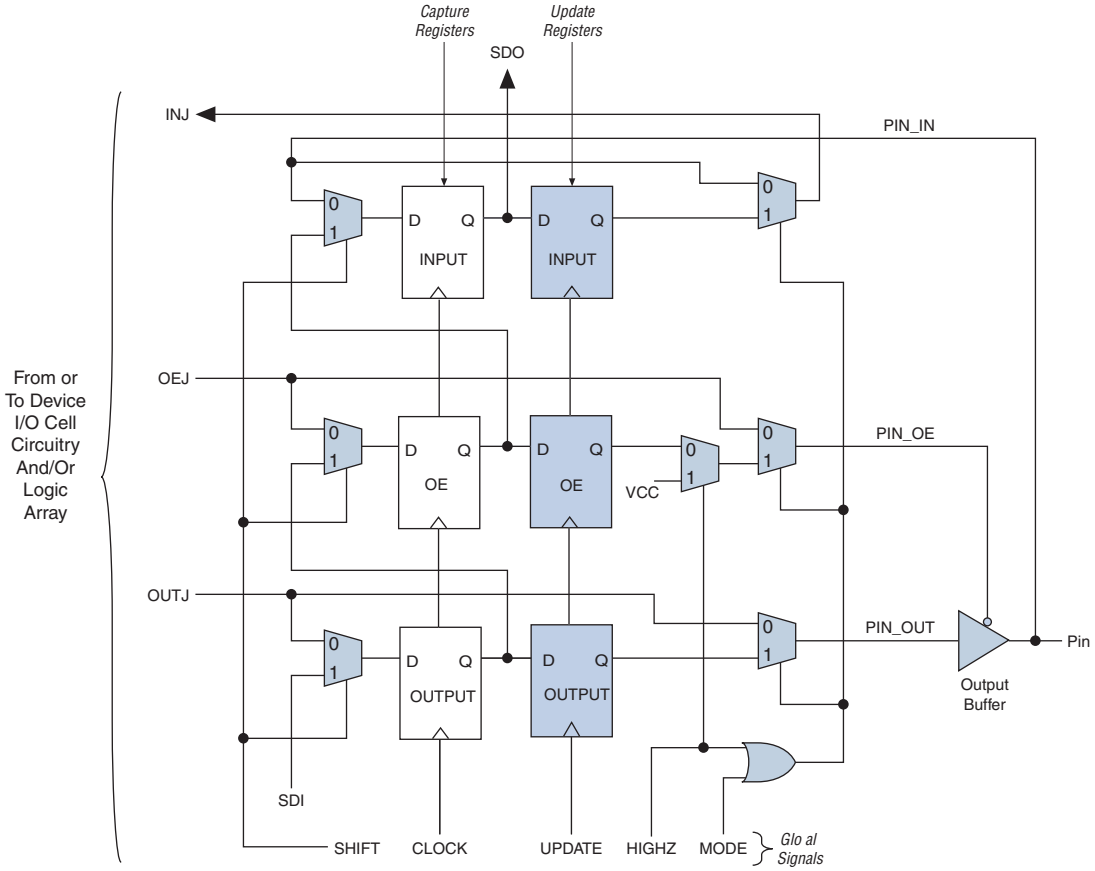


Table 10–6 lists the capture and update register capabilities of all BSCs within Arria V devices.

Table 10–6. Boundary Scan Cell Descriptions for Arria V Devices (1) (Part 1 of 2)

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 the 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 the control logic
Dedicated bidirectional (open drain) (4)	0	OEJ	PIN_IN	N.C. (2)	N.C. (2)	N.C. (2)	PIN_IN drives to the configuration control

Table 10–6. Boundary Scan Cell Descriptions for Arria V Devices (1) (Part 2 of 2)

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 (5)	OUTJ	OEJ	PIN_IN	N.C. (2)	N.C. (2)	N.C. (2)	PIN_IN drives to the configuration control and OUTJ drives to the output buffer
Dedicated output (6)	OUTJ	0	0	N.C. (2)	N.C. (2)	N.C. (2)	OUTJ drives to the output buffer

Notes to Table 10–6:

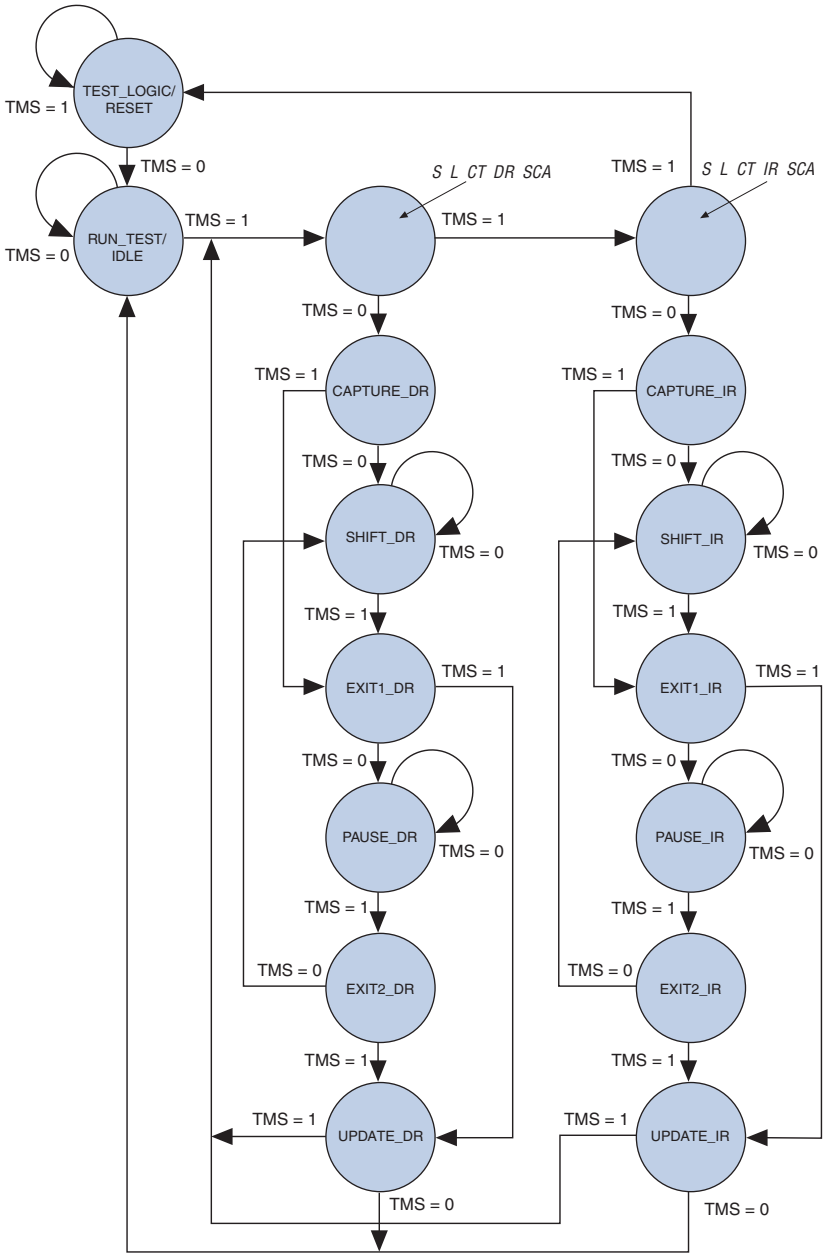
(1) TDI, TDO, TMS, TCK, all VCC and GND pin types, and VREF pins do not have BSCs.
(2) No Connect (N.C.).
(3) This includes the nCONFIG, MSEL0, MSEL1, MSEL2, MSEL3, MSEL4, and nCE pins.
(4) This includes the CONF_DONE and nSTATUS pins.
(5) This includes DCLK pin.
(6) This includes nCEO pin.

IEEE Std. 1149.1 TAP Controller

The IEEE Std. 1149.1 TAP controller—a 16-state machine clocked on the rising edge of TCK—uses the TMS pin to control IEEE Std. 1149.1 boundary-scan testing operation in the device.

Figure 10–5 shows the TAP controller state machine.

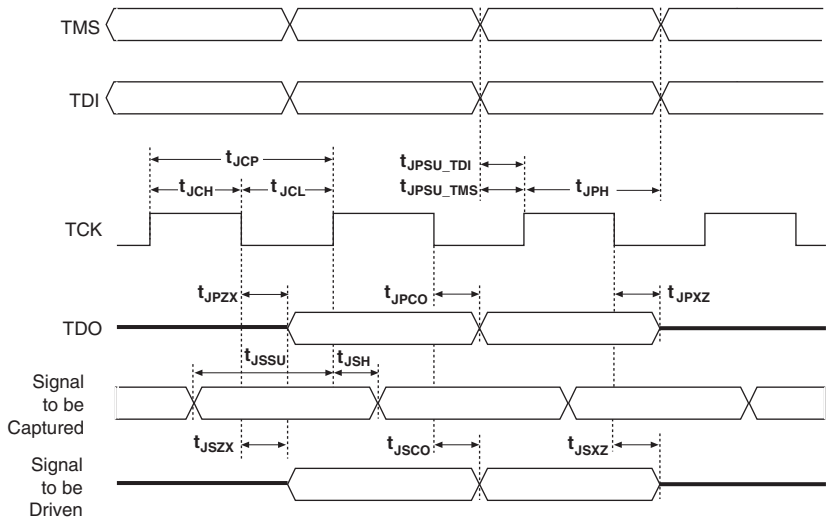
Figure 10–5. IEEE Std. 1149.1 TAP Controller State Machine



At device power up, the TAP controller starts in the TEST_LOGIC/RESET state. You can also force the TAP controller to the TEST_LOGIC/RESET state by holding TMS high for five TCK clock cycles After the TAP controller is in the TEST_LOGIC/RESET state, the TAP controller remains in this state as long as TMS is held high (while TCK is clocked)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.

Figure 10–6 shows the timing requirements for the IEEE Std. 1149.1 signals.

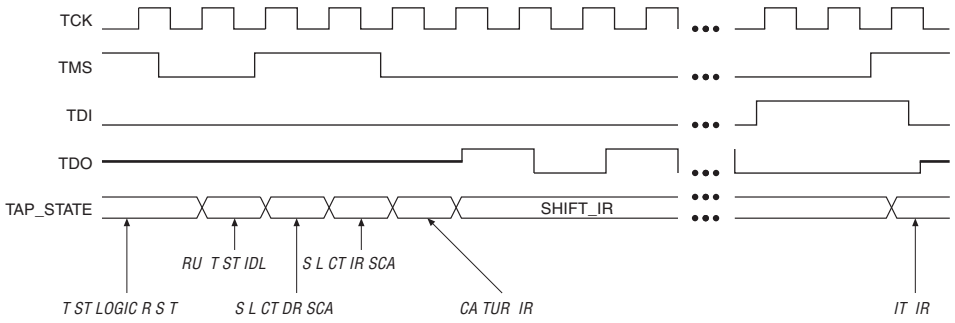
Figure 10–6. IEEE Std. 1149.1 Timing Waveforms



To start an 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.

Figure 10–7 shows the entry of the instruction code into the instruction register, the values of TCK, TMS, TDI, TDO, and the states of the TAP controller.

Figure 10–7. Selecting the Instruction Mode



From the RESET state, TMS is clocked with the pattern 01100 to advance the TAP controller to the SHIFT_IR state. 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 SHIFT_IR or SHIFT_DR state and is tri-stated at the first falling edge of TCK after leaving SHIFT_IR or SHIFT_DR state.

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, you can enter an instruction code 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. After entering the EXIT1_IR state, TDO becomes tri-stated again. TDO is always tri-stated except in the SHIFT_IR and SHIFT_DR states. The TAP controller advances to shift test data serially when one of the following instruction codes is entered correctly:

- SAMPLE/PRELOAD instruction mode
- EXTEST instruction mode
- BYPASS instruction mode

IEEE Std. 1149.1 JTAG Mandatory Instruction

This section describes the JTAG mandatory instructions.

SAMPLE/PRELOAD Instruction Mode

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 are transferred to the update registers. Use this data in EXTEST instruction mode.

Figure 10–8 shows the capture, shift, and update phases of SAMPLE/PRELOAD mode.

Figure 10–8. IEEE Std. 1149.1 BST SAMPLE/PRELOAD Mode

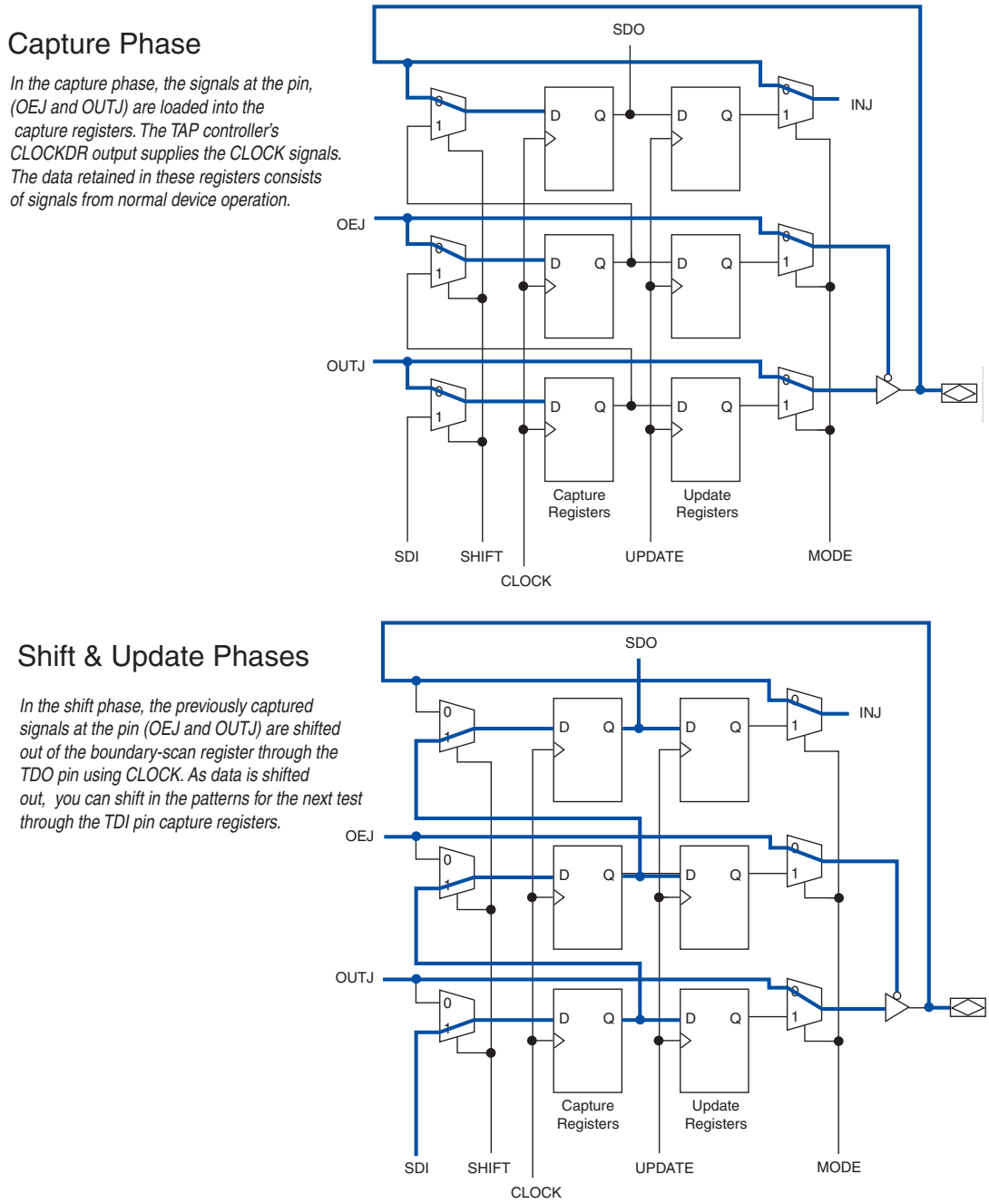
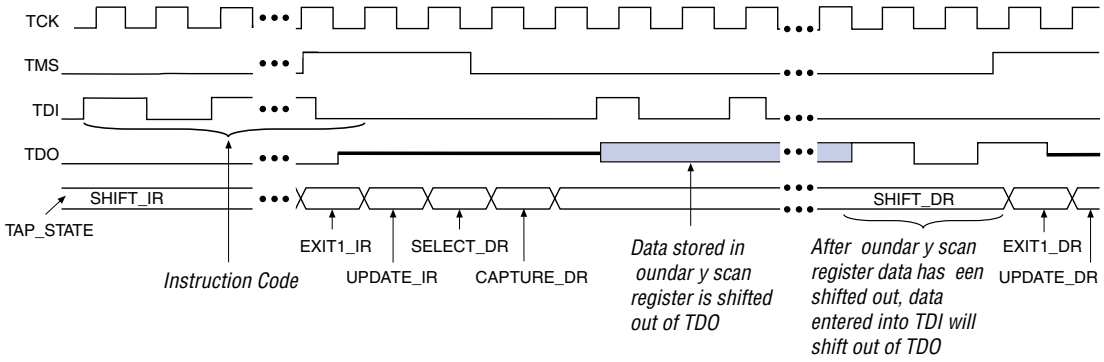


Figure 10–9 shows the SAMPLE/PRELOAD waveforms. The TDI pin shifts in the SAMPLE/PRELOAD instruction code. The TAP controller advances to the CAPTURE_DR state and then to the SHIFT_DR state, where it remains if you hold TMS low. The data that was present in the capture registers after the capture phase 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. If you hold TMS high on two consecutive TCK clock cycles, the TAP controller advances to the UPDATE_DR state for the update phase.

Figure 10–9. SAMPLE/PRELOAD Shift Data Register Waveforms



EXTEST Instruction Mode

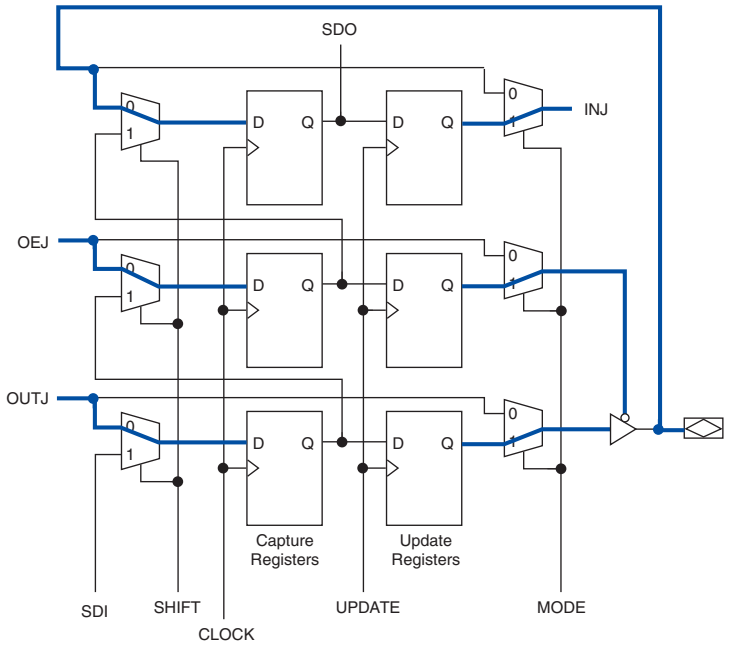
EXTEST selects data differently than SAMPLE/PRELOAD. EXTEST chooses data from the update registers as the source of the output and output enable signals. After the EXTEST instruction code is entered, the multiplexers select the update register data. Thus, you can force the data stored in these registers from a previous EXTEST or SAMPLE/PRELOAD test onto the pin signals. In the capture phase, the results of this test data is stored in the capture registers and then shifted out of TDO during the shift phase. You can then store new test data in the update registers during the update phase.

Figure 10–10 shows the capture, shift, and update phases of EXTEST mode.

Figure 10–10. 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 TAP controller's CLOCKDR output supplies the CLOCK signals. Previously retained data in the update registers drive PIN_IN and INJ, and allows the I/O pin to tri-state or drive a signal out.



Shift & Update Phases

In the shift phase, the previously captured signals at the pins (OEJ and OUTJ) are shifted out of the boundary-scan register through the TDO pin using CLOCK. As data is shifted out, you can shift in the patterns for the next test through 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 PIN_IN and INJ, and allow the I/O pin to tri-state or drive a signal out.

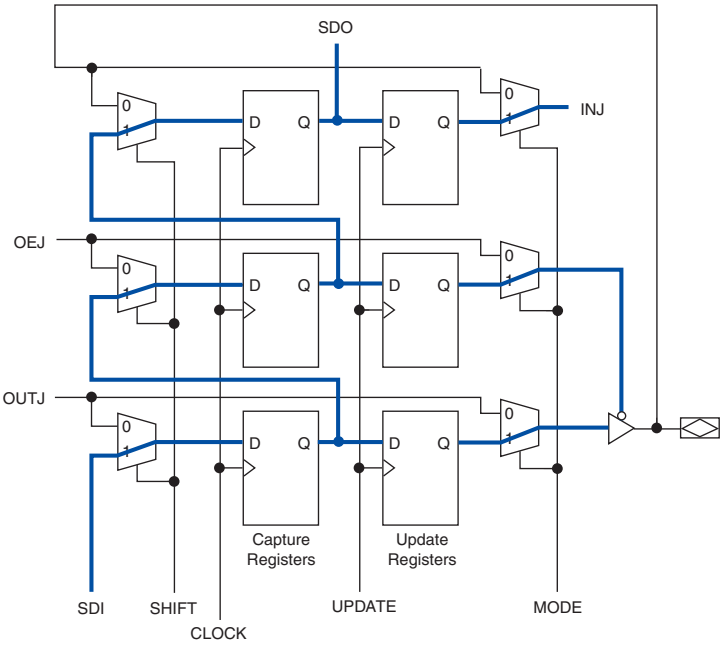
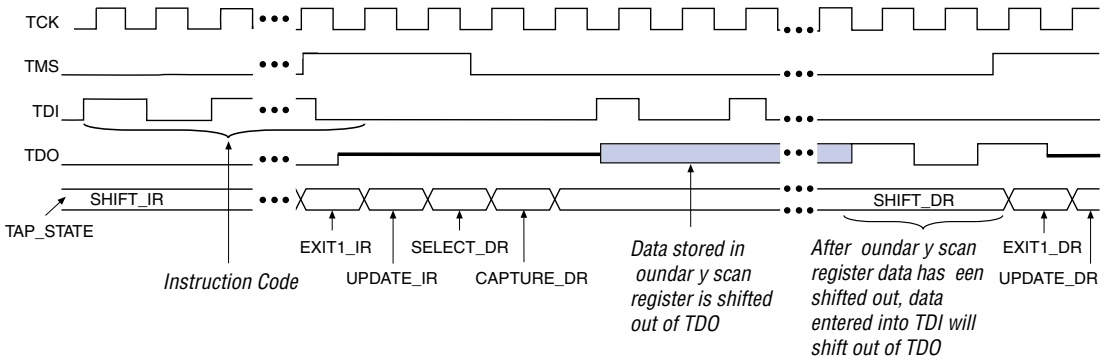


Figure 10–11 shows the EXTEST waveforms. The waveforms resemble the SAMPLE/PRELOAD waveforms, except for the instruction code. The data shifted out of TDO consists of the data that was present in the capture registers after the capture phase. New test data shifted into the TDI pin appears at the TDO pin after being clocked through the entire boundary-scan register.

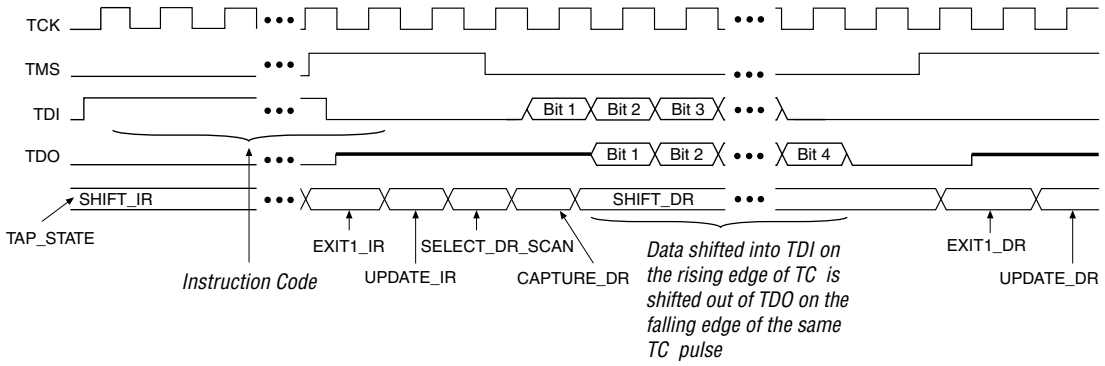
Figure 10–11. EXTEST Shift Data Register Waveforms



BYPASS Instruction Mode

BYPASS mode is activated when an instruction code of all ones is loaded in the instruction register. Figure 10–12 shows the BYPASS waveforms. When the TAP controller is in the SHIFT_DR 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 10–12. BYPASS Shift Data Register Waveforms



Document Revision History

Table 10–7 lists the revision history for this chapter.

Table 10–7. Document Revision History

Date	Version	Changes
June 2012	2.0	■ Restructured the chapter. ■ Updated Table 10–1 and Table 10–2.
February 2012	1.2	Updated Table 10–2.
November 2011	1.1	Minor text edits.
May 2011	1.0	Initial release.

11. Power Management in Arria V Devices

AV-52011-2.0

This chapter describes the hot-socketing feature, power-on reset (POR) requirements, power-up sequencing recommendation, internal temperature sensing diode (TSD), and their implementation in Arria® V devices.

This chapter contains the following sections:

For more information about the following topics, refer to the following documents:

- For information about the Quartus® II PowerPlay Power Analyzer tool, refer to the *PowerPlay Power Analysis* chapter in volume 3 of the *Quartus II Handbook*.
- For the recommended operating conditions of each power supply, refer to the *Arria V Device Datasheet*.
- For detailed information about power supply pin connection guidelines and power regulator sharing, refer to the *Arria V Device Family Pin Connection Guidelines*.
- For detailed information about power supply design requirements, refer to the Board Design Resource Center page.

Power Consumption

The total power consumption of an Arria V device consists of the following components:

- Static power—the power consumed by the configured device when it is powered up but no clocks are operating.
- Dynamic power—the power consumed by the configured device when it is operating. Equation 11–1 shows how to calculate dynamic power.

Equation 11–1. Dynamic Power *(1)*

$$P = \frac{1}{2}CV^2 \times frequency$$

Note to Equation 11–1:

(1) P = power; C = load capacitance; and V = supply voltage level.

Equation 11–1 shows that power is design-dependent and is determined by the operating frequency of your design. Arria V devices minimize static and dynamic power using advanced process optimizations. This technology allows Arria V designs to meet specific performance requirements with the lowest possible power.

Internal Temperature Sensing Diode

Arria V devices feature an internal TSD with built-in analog-to-digital converter (ADC) circuitry that self-monitors the junction temperature of the die. You can also use the TSD with external circuitry for other activities such as controlling air flow to the Arria V device.

Monitoring the junction temperature is crucial for thermal management. Junction temperature is calculated using ambient or case temperature, junction-to-ambient (ja) or junction-to-case (jc) thermal resistance, and device power consumption.

You can use the Arria V internal TSD in the following operations:

- Power-up mode—to read the temperature of the die during configuration, enable the ALTTEMP_SENSE megafunction in your design.
- User mode—to read the temperature of the die during user mode, assert the `clken` signal to the internal TSD circuitry.

To reduce power consumption, disable the Arria V internal TSD when you are not using it.

For more information about the following topics, refer to the following documents:

- For information about using the ALTTEMP_SENSE megafunction, refer to the *Temperature Sensor (ALTTEMP_SENSE) Megafunction User Guide*.
- For information about the Arria V internal TSD specification, refer to the *Arria V Device Datasheet*.

Hot-Socketing Feature

Arria V devices support hot socketing—also known as hot plug-in or hot swap. The hot-socketing circuitry monitors the V_{CCIO}, V_{CCPD}, V_{CC}, and V_{CCP} power supplies and all V_{CCIO} and V_{CCPD} banks. You can power up or power down these power supplies in any sequence.

During the hot-socketing operation, the I/O pin capacitance is less than 15 pF and the clock pin capacitance is less than 20 pF.

The hot-socketing capability removes some of the difficulty faced by designers when using the Arria V devices on PCBs that contain a mixture of devices with different voltage requirements.

The hot-socketing capability in Arria V devices provides the following advantages:

- You can drive signals into the I/O, dedicated input, and dedicated clock pins before or during power up or power down without damaging the device. External input signals to the I/O pins of the unpowered device will not power the V_{CCIO}, V_{CCPD}, V_{CC}, and V_{CCP} power supplies through internal paths within the device.
- The output buffers are tri-stated during system power up or power down. Because the Arria V device does not drive signals out before or during power up, the device does not affect the other operating buses.
- You can insert or remove an Arria V device from a powered-up system board without damaging or interfering with the operation of the system board. This capability allows you to avoid sinking current through the device signal pins to the device power supply, which can create a direct connection to GND that causes power supply failures.
- During hot socketing, Arria V devices are immune to latch up that can occur when a device is hot-socketed into an active system.

Altera uses GND as a reference for hot-socketing and I/O buffer circuitry designs. To ensure proper operation, connect GND between boards before connecting the power supplies. This prevents GND on your board from being pulled up inadvertently by a path to power through other components on your board. A pulled up GND could otherwise cause an out-of-specification I/O voltage or over current condition in the Altera® device.

For information about the Arria V hot-socketing specifications, refer to the *Arria V Device Datasheet*.

Hot-Socketing Implementation

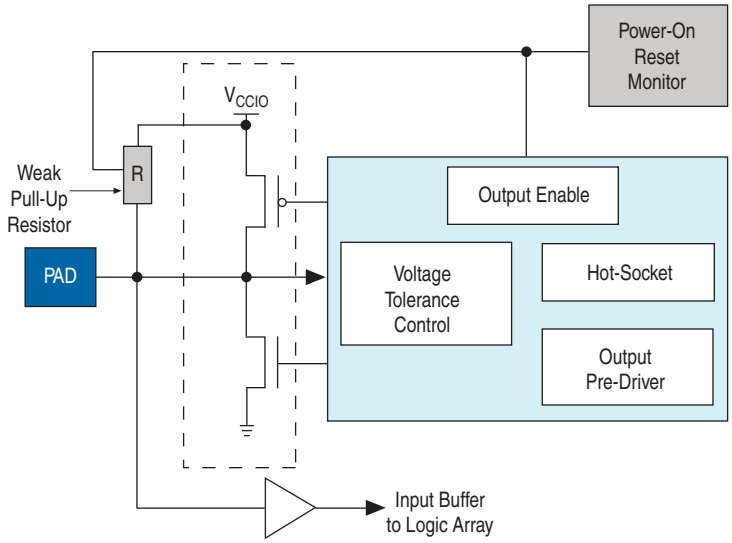
The hot-socketing feature tri-state the output buffer during power up and power down of the V_{CCIO}, V_{CCPD}, V_{CC}, and V_{CCP} power supplies. When these power supplies are below the threshold voltage, the hot-socketing circuitry generates an internal HOTSCKT signal.

Hot-socketing circuitry prevents excess I/O leakage during power up. When the voltage ramps up very slowly, I/O leakage is still relatively low, even after the POR signal is released and configuration is complete.

The output buffer cannot flip from the state set by the hot-socketing circuitry at very low voltage. To allow the CONF_DONE and nSTATUS configuration pins in bank 8A of the Arria V device to operate during configuration, the hot-socketing feature is not applied to these configuration pins. Therefore, these pins will drive out during power up and power down.

Figure 11–1 shows the I/O pin circuitry for Arria V devices.

Figure 11–1. Hot-Socketing Circuitry for Arria V Devices



The POR circuitry monitors the voltage level of the power supplies and keeps the I/O pins tri-stated until the device is in user mode. The weak pull-up resistor (R) in the Arria V input/output element (IOE) is enabled during configuration download to keep the I/O pins from floating. The 3.3-V tolerance control circuit allows the I/O pins to be driven by 3.3 V before the V_{CC}, V_{CCP}, V_{CCPD}, and V_{CCIO} power supplies are powered and prevents the I/O pins from driving out before the device enters user mode.

Power-Up Sequencing

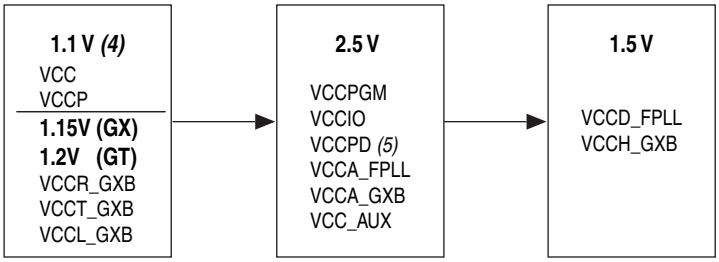
The Arria V GX B1, B3, B5, and B7 devices, and Arria V GT D3 and D7 devices are required to follow the power-up sequence in Figure 11–2.

CAUTION

Failure to follow the required power-up sequencing in the Arria V GX B1, B3, B5, and B7 devices, and Arria V GT D3 and D7 devices may result in additional current consumption or functionality issue.

For the rest of the Arria V devices, follow the power-up sequence as recommended to ensure minimum current drawn during device power-up.

Figure 11–2. Power-up Sequencing *(1), (2), (3)*



Notes to Figure 11–2:

(1) Power up V_{CCBAT} at any time.

(2) Ramp up the power rails in each group to a minimum of 80% of their full rail before the next group starts.

(3) Power up V_{CCP}, V_{CCR_GXB}, V_{CCT_GXB}, and V_{CCL_GXB} together with V_{CC}.

(4) Power up the V_{CC} and V_{CCP} with 1.15 V for the Arria V GT I3 device.

(5) An in-rush current of I_{CCPD} up to 1.22 A with a duration <30 µs may be observed during the device start-up if you do not follow the power-up sequence.

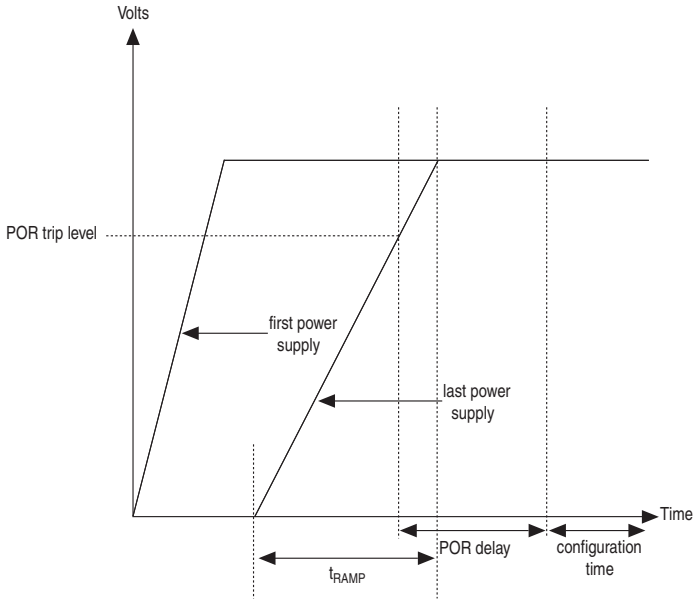
Power-On Reset Circuitry

The POR circuitry keeps the Arria V device in the reset state until the power supply outputs are within the recommended operating range.

A POR event occurs when you power up the Arria V device until the power supplies reach the recommended operating range within the maximum power supply ramp time, t_{RAMP}. If t_{RAMP} is not met, the Arria V device I/O pins and programming registers remain tri-stated, during which device configuration could fail.

Figure 11–3 shows the relationship between t_{RAMP} and POR delay.

Figure 11–3. Relationship Between t_{RAMP} and POR Delay



For more information about the POR delay specification and t_{RAMP}, refer to the *Arria V Device Datasheet*.

The Arria V POR circuitry uses an individual detecting circuitry to monitor each of the configuration-related power supplies independently. The main POR circuitry is gated by the outputs of all the individual detectors. The main POR signal is asserted when the power starts to ramp up. This signal is released after the last ramp-up power reaches the POR trip level during power up.

In user mode, the main POR signal is asserted when any of the monitored power goes below its POR trip level. Asserting the POR signal forces the device into the reset state.

The POR circuitry checks the functionality of the I/O level shifters powered by the V_{CCPD} and V_{CCPGM} power supplies during power-up mode. The main POR circuitry waits for all the individual POR circuitries to release the POR signal before allowing the control block to start programming the device.

Figure 11–4 shows a simplified POR diagram for Arria V devices.

Figure 11–4. Simplified POR Diagram for Arria V Devices

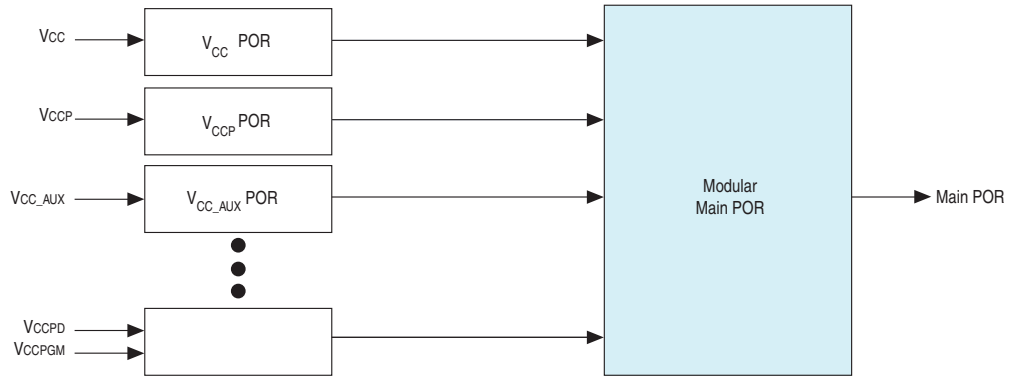


Table 11–1 lists the power supplies monitored and not monitored in the POR circuitry.

Table 11–1. Power Supplies Monitored and Not Monitored by the Arria V POR Circuitry

Power Supplies Monitored	Power Supplies Not Monitored
■ V_{CC_AUX} ■ V_{CCBAT} ■ V_{CC} ■ V_{CCP} ■ V_{CCPD} ■ V_{CCPGM} ■ V_{CC_HPS} ■ V_{CCPD_HPS} ■ $V_{CCRSTCLK\_HPS}$	■ V_{CCT_GXB} ■ V_{CCH_GXB} ■ V_{CCR_GXB} ■ V_{CCA_GXB} ■ V_{CCL_GXB} ■ V_{CCA_FPLL} ■ V_{CCD_FPLL} ■ V_{CCIO} ■ V_{CCIO_HPS} ■ V_{CCPLL_HPS}

For the device to exit POR, you must power the V_{CCBAT} power supply even if you do not use the volatile key.

For information about the `MSEL` pin settings for each POR delay, refer to the *Configuration, Design Security, and Remote System Upgrades in Arria V Devices* chapter.

Document Revision History

Table 11–2 lists the revision history for this chapter.

Table 11–2. Document Revision History

Date	Version	Changes
June 2012	2.0	■ Restructured the chapter. ■ Added the "Power-Up Sequencing" section.
February 2012	1.3	Updated V_{CCP} description.
December 2011	1.2	■ Added V_{CCP} information. ■ Updated Table 11–1.
November 2011	1.1	Restructured chapter.
May 2011	1.0	Initial release.

This chapter provides additional information about the document and Altera.

How to Contact Altera

To locate 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
Nontechnical 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

The following table shows the typographic conventions this document uses.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Indicate command names, dialog box titles, dialog box options, and other GUI labels. For example, **Save As** dialog box. For GUI elements, capitalization matches the GUI.
bold type	Indicates directory names, project names, disk drive names, file names, file name extensions, software utility names, and GUI labels. For example, **\qdesigns** directory, **D:** drive, and **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Indicate document titles. For example, *Stratix IV Design Guidelines*.
italic type	Indicates variables. For example, *n* + 1. Variable names are enclosed in angle brackets (< >). For example, *<file name>* and *<project name>***.pof** file.
Initial Capital Letters	Indicate keyboard keys and menu names. For example, the Delete key and the Options menu.
"Subheading Title"	Quotation marks indicate references to sections in 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`. The suffix `n` denotes an active-low signal. 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`).
↵	An angled arrow instructs you to press the Enter key.
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.
	A question mark directs you to a software help system with related information.
	The feet direct you to another document or website with related information.
	A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.
	A warning calls attention to a condition or possible situation that can cause you injury.
	The envelope links to the Email Subscription Management Center page of the Altera website, where you can sign up to receive update notifications for Altera documents.

Arria V Device Handbook

Volume 2: Transceivers

101 Innovation Drive
San Jose, CA 95134
www.altera.com

AV-5V3-2.0

Document last updated for Altera Complete Design Suite version: 12.0
Document publication date: June 2012

Contents

Chapter 3. Transceiver Reset Control and Power-Down in Arria V Devices

Chapter 4. Transceiver Protocol Configurations in Arria V Devices

Chapter Revision Dates

The chapters in this document, Arria V Device Handbook Volume 2: Transceivers, were revised on the following dates. Where chapters or groups of chapters are available separately, part numbers are listed.

Chapter 1. Transceiver Architecture in Arria V Devices
Revised: *June 2012*
Part Number: *AV53001-1.2*

Chapter 2. Transceiver Clocking in Arria V Devices
Revised: *June 2012*
Part Number: *AV53002-1.2*

Chapter 3. Transceiver Reset Control and Power-Down in Arria V Devices
Revised: *June 2012*
Part Number: *AV53003-1.2*

Chapter 4. Transceiver Protocol Configurations in Arria V Devices
Revised: *June 2012*
Part Number: *AV53004-1.2*

Chapter 5. Transceiver Custom Configurations in Arria V Devices
Revised: *June 2012*
Part Number: *AV53005-1.2*

Chapter 6. Transceiver Loopback Support in Arria V Devices
Revised: *June 2012*
Part Number: *AV53006-1.2*

Chapter 7. Dynamic Reconfiguration in Arria V Devices
Revised: *June 2012*
Part Number: *AV53007-1.2*

1. Transceiver Architecture in Arria V Devices

AV53001-1.2

This chapter describes the Arria® V transceiver architecture, channels, and transmitter and receiver channel datapaths.

Altera® 28-nm Arria V FPGAs provide integrated transceivers with the lowest power requirement at 10- and 6-Gigabits per second (Gbps). These transceivers comply with a wide range of protocols and data rate standards.

Arria V devices are available in two variants:

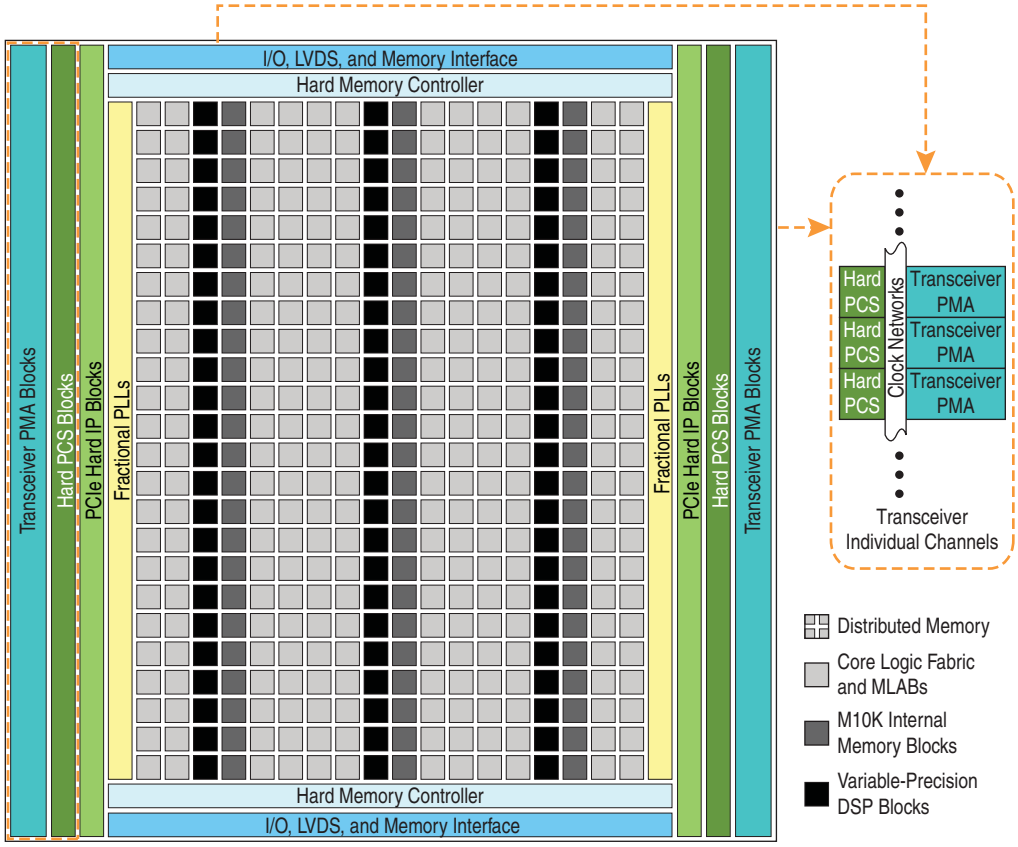
- Arria V GX with integrated backplane-capable transceivers supporting serial data rates between 611 megabits per second (Mbps) and 6.5536 Gbps.
- Arria V GT with integrated backplane-capable transceivers supporting serial data rates between 611 Mbps and 6.5536 Gbps, and has chip-to-chip and chip-to-module capabilities up to 10.3125 Gbps.

This chapter contains the following sections:

Architecture Overview

Figure 1–1 shows the columns of transceivers on the left and right sides of the Arria V device.

Figure 1–1. Basic Layout of Transceivers in Arria V Devices *(1), (2)*



Notes to Figure 1–1:

(1) Figure 1–1 represents one variant of an Arria V device. Other variants may have transceivers and PCI Express® (PCIe®) hard IP only on the left side of the device.

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

The Arria V hard IP for PCIe implements the PCIe protocol stack including the following layers:

- Physical interface/media access control (PHY/MAC) layer
- Data link layer
- Transaction layer

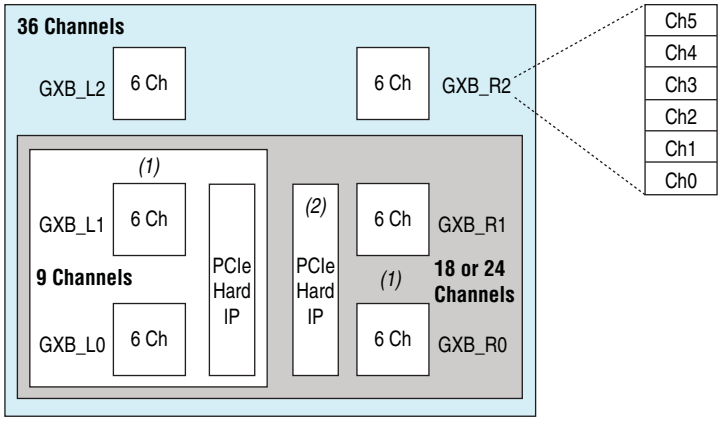
The embedded hard IP saves significant FPGA resources, reduces design risk, and reduces the time required to achieve timing closure. The hard IP complies with the PCI Express Base Specification 1.1 and 2.0 for Gen1 and Gen2 signaling data rates, respectively.

For more information about the PCIe hard IP block architecture, refer to the *Arria V Hard IP for PCI Express User Guide*.

Transceiver Banks

The columns of Arria V transceivers are categorized in banks of six channels. The transceiver bank boundaries are important for clocking resources, bonding channels, and fitting. Figure 1–2 and Figure 1–3 show the location of the transceiver banks in Arria V devices. In some package variations, some transceiver banks are reduced to three channels.

Figure 1–2. Transceiver Bank Location for Arria V GX Devices

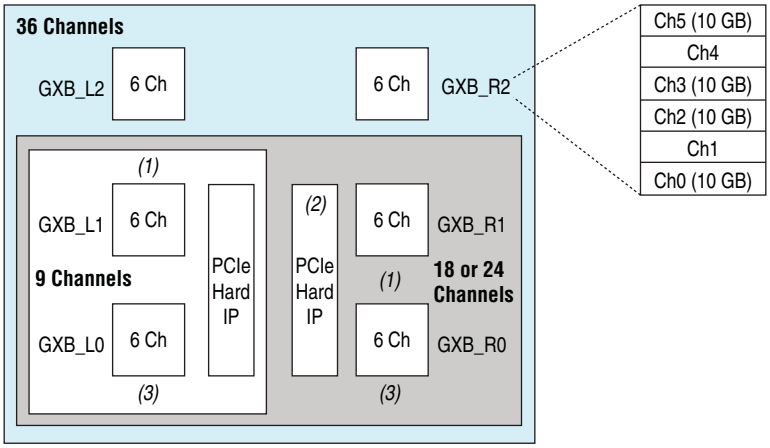


Notes to Figure 1–2:

(1) Channels 3 through 5 in this bank are not available in 9 and 18 transceiver channel devices.

(2) The PCIe hard IP spans across GXB_R0 bank only in 18 transceiver channel devices.

Figure 1–3. Transceiver Bank Location for Arria V GT Devices



Notes to Figure 1–3:

(1) Channels 3 through 5 in this bank are not available in 9 and 18 transceiver channel devices.

(2) The PCIe hard IP spans across GXB_R0 bank only in 18 transceiver channel devices.

(3) Channels 0-2 in this bank support data rate up to 6.5536 Gbps only. PMA direct mode is not supported in these channels.

(4) You can alternatively configure the two 10-Gbps channels located in channels 0 and 2, or in channels 3 and 5 as 6-Gbps channels.

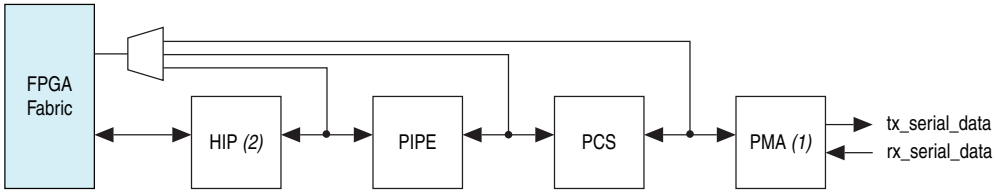
(5) 10-Gbps channels are for chip-to-chip connections only.

(6) For 10-Gbps SFF 8431 compliance, please contact Altera.

The Arria V transceivers are comprised of a transmitter and receiver that can operate individually or simultaneously—providing a full-duplex physical layer implementation for high-speed serial interfacing. Each transmitter and receiver are divided into two blocks: PMA and PCS. The PMA block connects the FPGA to the channel, generates the required clocks, and converts the data from parallel to serial or serial to parallel. The PCS block performs digital processing logic between the PMA and the FPGA core.

Figure 1–4 and Figure 1–5 below show the transceiver interfaces in Arria V GX and GT devices.

Figure 1–4. Full Duplex Channel Interfaces Up to 6.5536 Gbps in GX and GT Devices

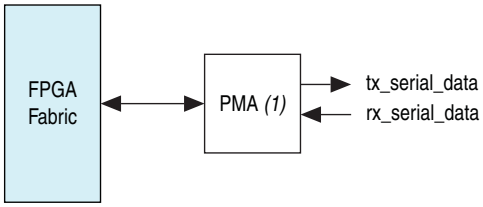


Notes to Figure 1–4:

(1) Refer to Figure 1–2 and Figure 1–3 for channels that support PMA to fabric interfaces.

(2) Refer to *Arria V Hard IP for PCI Express User Guide* for channels that support interfaces with hard IP.

Figure 1–5. Full Duplex Channel Interfaces Beyond 6.5536 Gbps in GX and GT Devices



Note to Figure 1–5:

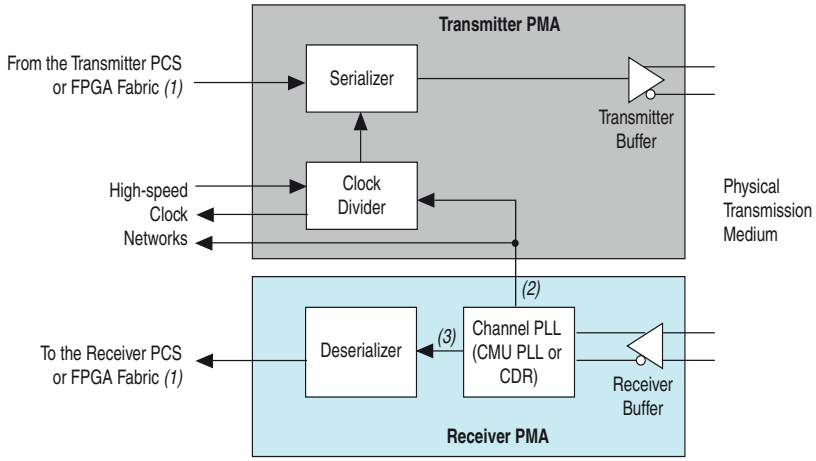
(1) Refer to Figure 1–2 and Figure 1–3 for channels that support PMA to fabric interfaces.

PMA Architecture

The PMA includes the transmitter and receiver datapaths, CMU PLL (configured from the channel PLL), and the clock divider. The analog circuitry and differential on-chip termination (OCT) in the PMA requires the calibration block to compensate for process, voltage, and temperature variations (PVT).

Figure 1–6 shows the PMA transceiver channel for Arria V devices.

Figure 1–6. Transceiver Channel PMA for Arria V Devices



Notes to Figure 1–6:

(1) The channel PLL provides the serial clock when configured as a CMU PLL.

(2) The channel PLL recovers the clock and serial data stream when configured as a CDR.

Transmitter PMA Datapath

Table 1–1 lists the blocks of the transmitter PMA datapath.

Table 1–1. Functional Blocks in the Transmitter PMA Datapath

Block	Functionality
Serializer	■ Converts the incoming low-speed parallel data from the transmitter PCS to high-speed serial data and sends the data LSB first to the transmitter buffer. ■ Supports 8-, 10-, 16-, and 20-bit serialization factors. ■ Supports the optional polarity inversion and bit reversal features. ■ Additionally supports the 64- and 80-bit serialization factor for 10-Gbps transceiver channels in Arria V GT devices.
Transmitter Buffer	■ The 1.5-V pseudo current mode logic (**PCML**) output buffer conditions the high-speed serial data for transmission into the physical medium, as shown in Figure 1–7. ■ Supports the following features: ■ Programmable differential output voltage (V_{OD}) ■ Programmable pre-emphasis ■ Programmable V_{CM} current strength ■ Programmable slew rate ■ On-chip biasing for common-mode voltage (TX V_{CM}) ■ Differential OCT (85, 100, 120 and 150 Ω) ■ Transmitter output tristate ■ Receiver detect (for the PCIe receiver detection function)

Serializer

The serializer provides parallel-to-serial data conversion and sends the data LSB first from the transmitter physical coding sublayer (PCS) to the transmitter buffer. Additionally, the serializer provides the polarity inversion and bit reversal features.

Polarity Inversion

The positive and negative signals of a serial differential link might accidentally be swapped during board layout. The polarity inversion feature of the transmitter corrects this error without requiring a board respin or major updates to the logic in the FPGA fabric. The polarity inversion feature inverts the polarity of every bit at the input to the serializer, which has the same effect as swapping the positive and negative signals of the serial differential link.

Polarity inversion is controlled dynamically with the `tx_invpolarity` register. When you enable the polarity inversion feature, it may cause initial disparity errors at the receiver with 8B/10B-coded data. The downstream system at the receiver must be able to tolerate these disparity errors.

CAUTION Enabling polarity inversion midway through a serialized word corrupts the word.

Bit Reversal

You can reverse the transmission bit order to achieve MSB-to-LSB ordering using the bit reversal feature at the transmitter. Table 1–2 lists the bit reversal serialization factors.

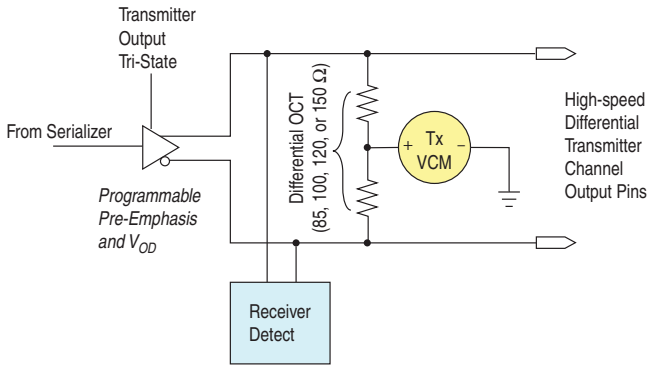
Table 1–2. Bit Reversal Feature

Bit Reversal Option	Transmission Bit Order	
	8- or 10-bit Serialization Factor	16- or 20-bit Serialization Factor
Disabled (default)	LSB to MSB	LSB to MSB
Enabled	MSB to LSB For example: 8-bit—`D[7:0]` rewired to `D[0:7]` 10-bit—`D[9:0]` rewired to `D[0:9]`	MSB to LSB For example: 16-bit—`D[15:0]` rewired to `D[0:15]` 20-bit—`D[19:0]` rewired to `D[0:19]`

Transmitter Buffer

Figure 1–7 shows the transmitter buffer in Arria V devices.

Figure 1–7. Transmitter Buffer in Arria V Devices



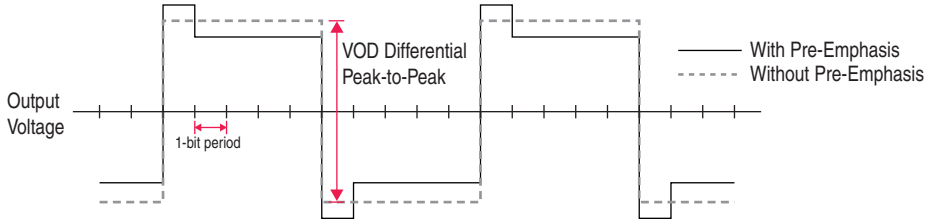
For more information about the specifications for the transmitter buffer features, refer to the *Arria V Device Datasheet*.

Table 1–3 lists the features provided by the Pseudo Current Mode Logic (PCML) output buffer to the integrated circuitry.

Table 1–3. Description of the Transmitter Buffer Features

Category	Features	Description
Improve Signal Integrity	Programmable Differential Output Voltage (V_{OD})	Controls the current mode drivers for signal amplitude to handle different trace lengths, various backplanes, and receiver requirements. The actual V_{OD} level is a function of the current setting and the transmitter termination value.
	Programmable Pre-Emphasis	Boosts the high-frequency components of the transmitted signal, which may be attenuated when propagating through the transmission medium. The physical transmission medium can be represented as a low-pass filter in the frequency domain. Variation in the signal frequency response that is caused by attenuation significantly increases the data-dependent jitter and other intersymbol interference (ISI) effects at the receiver end. Using the pre-emphasis feature maximizes the data opening at the far-end receiver. Figure 1–8 shows the signal transmission at the transmitter output with and without applying pre-emphasis post-tap for a 5 Gigabit per second (Gbps) signal with an alternating data pattern of five 1s and five 0s.
	Programmable Slew Rate	Controls the rate of change for the signal transition.
Save Board Space and Cost	On-Chip Biasing	Establishes the required transmitter common-mode voltage (TX V_{CM}) level at the transmitter output. The circuitry is available only if you enable on-chip termination (OCT). When you disable OCT, you must implement off-chip biasing circuitry to establish the required TX V_{CM} level.
	Differential OCT	The termination resistance is adjusted by the calibration circuitry, which compensates for the process, voltage, and temperature variations (PVT). You can disable OCT and use external termination. However, you must implement off-chip biasing circuitry to establish the required TX V_{CM} level. TX V_{CM} is tri-stated when using external termination.
Reduce Power	Programmable V_{CM} Current Strength	Controls the impedance of V_{CM}. A higher impedance setting reduces current consumption from the on-chip biasing circuitry.
Protocol-Specific Function	Transmitter Output Tri-State	Enables the transmitter differential pair voltages to be held constant at the same value determined by the TX V_{CM} level with the transmitter in the high impedance state. The transmitter output tri-state feature is compliant with differential and common-mode voltage levels and operation time requirements for transmitter electrical idle, as specified in the PCI Express Base Specification 2.0 for Gen1 and Gen2 signaling rates.
	Receiver Detect	Provides link partner detection capability at the transmitter end using an analog mechanism for the receiver detection sequence during link initialization in the Detect state of the PCI Express® (PCIe) Link Training and Status State Machine (LTSSM) states. The circuit detects if there is a receiver downstream by changing the transmitter V_{CM} to create a step voltage and measuring the resulting voltage rise time. For proper functionality, the series capacitor (AC-coupled link) and receiver termination values must comply with the PCI Express Base Specification 2.0. The circuit is clocked using `fixedclk` and requires that the transmitter OCT be enabled with the output tri-stated.

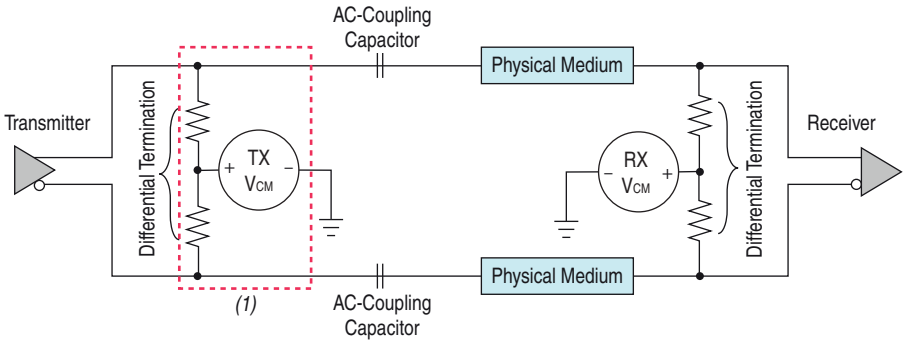
Figure 1–8. Example of Pre-Emphasis Effect on Signal Transmission at Transmitter Output



The receiver can be only AC-coupled to a transmitter. In an AC-coupled link, the AC-coupling capacitor blocks the transmitter V_{CM}. At the receiver end, the termination and biasing circuitry restores the V_{CM} level that is required by the receiver.

Figure 1–9 shows an AC-coupled link with the Arria V transmitter.

Figure 1–9. AC-Coupled Link with Arria V Transmitter



Note to Figure 1–9:

(1) When you disable OCT, you must implement external termination and off-chip biasing circuitry to establish the required TX V_{CM} level.

Receiver PMA Datapath

This section describes the receiver buffer, channel phase-locked loop (PLL) configured for clock data recovery (CDR) operation, and deserializer blocks in the receiver PMA datapath. Table 1–4 lists the blocks of the receiver PMA datapath.

Table 1–4. Functional Blocks in the Receiver PMA Datapath

Block	Functionality
Receiver Buffer (1)	■ Receives the serial data stream and feeds the stream to the channel PLL if you configure the channel PLL as a CDR, as shown in Figure 1–10. ■ Supports the following features: ■ Programmable equalization ■ Programmable DC gain ■ Programmable V_{CM} current strength ■ On-chip biasing for common-mode voltage (RX V_{CM}) ■ I/O standard (1.5 V **PCML**, 2.5 V **PCML**, **LVDS**, **LVPECL**) ■ Differential OCT (85, 100, 120 and 150 Ω) ■ Signal detect.
Channel PLL	■ Recovers the clock and serial data stream if you configure the channel PLL as a CDR. ■ Requires offset cancellation to correct the analog offset voltages. ■ If you do not use the channel PLL as a CDR, you can configure the channel PLL as a CMU PLL for clocking the transceivers. For more information about the channel PLL configured as a CMU PLL, refer to "CMU PLL" on page 1–16.
Deserializer	■ Converts the incoming high-speed serial data from the receiver buffer to low-speed parallel data for the receiver PCS. ■ Receives serial data in LSB-to-MSB order. ■ Supports 8-, 10-, 16-, and 20-bit deserialization factors. ■ Supports the optional clock-slip feature for applications with stringent latency uncertainty requirement. ■ Additionally supports the 64- and 80-bit serialization factor for 10-Gbps transceiver channels in Arria V GT devices.

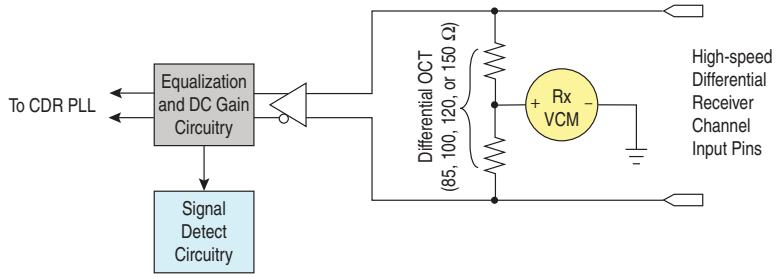
Note to Table 1–4:

(1) The receiver can be AC-coupled only. The on-chip or off-chip receiver termination and biasing circuitry automatically restores the required receiver V_{CM} level.

Receiver Buffer

Figure 1–10 shows the receiver buffer in Arria V devices.

Figure 1–10. Receiver Buffer in Arria V Devices



For more information about the specifications for the receiver buffer features, refer to the *Arria V Device Datasheet*.

Table 1–5 lists the features provided by the receiver buffer to the integrated circuitry.

Table 1–5. Description of the Receiver Buffer Features (Part 1 of 2)

Category	Features	Description
Improve Signal Integrity	Programmable Equalization	Boosts the high-frequency components of the received signal, which may be attenuated when propagating through the transmission medium. The physical transmission medium can be represented as a low-pass filter in the frequency domain. Variation in the signal frequency response that is caused by attenuation leads to data-dependent jitter and other ISI effects, causing incorrect sampling on the input data at the receiver. The amount of the high-frequency boost required at the receiver to overcome signal attenuation depends on the loss characteristics of the physical medium.
	Programmable DC Gain	Provides equal boost to the received signal across the frequency spectrum.
Save Board Space and Cost	On-Chip Biasing	Establishes the required receiver common-mode voltage (RX V_{CM}) level at the receiver input. The circuitry is available only if you enable OCT. When you disable OCT, you must implement off-chip biasing circuitry to establish the required RX V_{CM} level.
	Differential OCT	The termination resistance is adjusted by the calibration circuitry, which compensates for the PVT. You can disable OCT and use external termination. However, you must implement off-chip biasing circuitry to establish the required RX V_{CM} level. RX V_{CM} is tri-stated when using external termination.

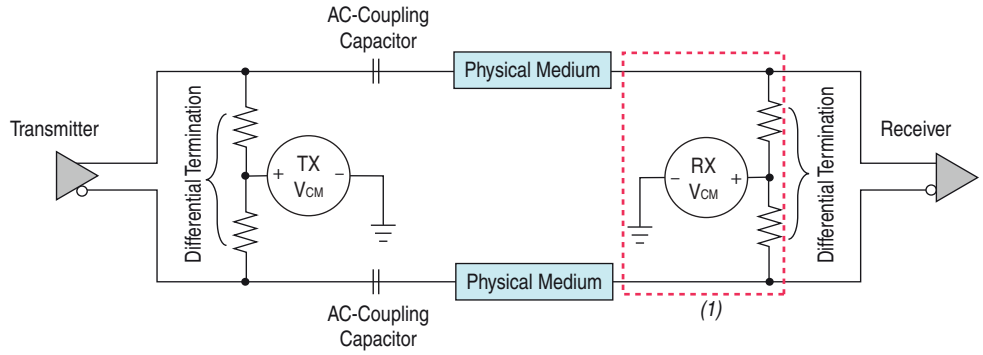
Table 1–5. Description of the Receiver Buffer Features (Part 2 of 2)

Category	Features	Description
Reduce Power	Programmable V_{CM} Current Strength	Controls the impedance of V_{CM}. A higher impedance setting reduces current consumption from the on-chip biasing circuitry.
Protocol-Specific Function	Signal Detect	Senses if the signal level present at the receiver input is above or below the threshold voltage that you specified. The detection circuitry has a hysteresis response that asserts the status signal only when a number of data pulses exceeding the threshold voltage are detected and deasserts the status signal when the signal level below the threshold voltage is detected for a number of recovered parallel clock cycles. The circuitry requires the input data stream to be 8B/10B-coded. Signal detect is compliant to the threshold voltage and detection time requirements for electrical idle detection conditions as specified in the PCI Express Base Specification 2.0 for Gen1 and Gen2 signaling rates.

The receiver can be only AC-coupled to a transmitter. In an AC-coupled link, the AC-coupling capacitor blocks the transmitter V_{CM}. At the receiver end, the termination and biasing circuitry restores the V_{CM} level that is required by the receiver.

Figure 1–11 shows an AC-coupled link with the Arria V receiver.

Figure 1–11. AC-Coupled Link with Arria V Receiver



Note to Figure 1–11:

(1) When you disable OCT, you must implement external termination and off-chip biasing circuitry to establish the required RX V_{CM} level.

Channel PLL

This section describes the architecture and operation of the channel PLL when it is configured as a CDR PLL.

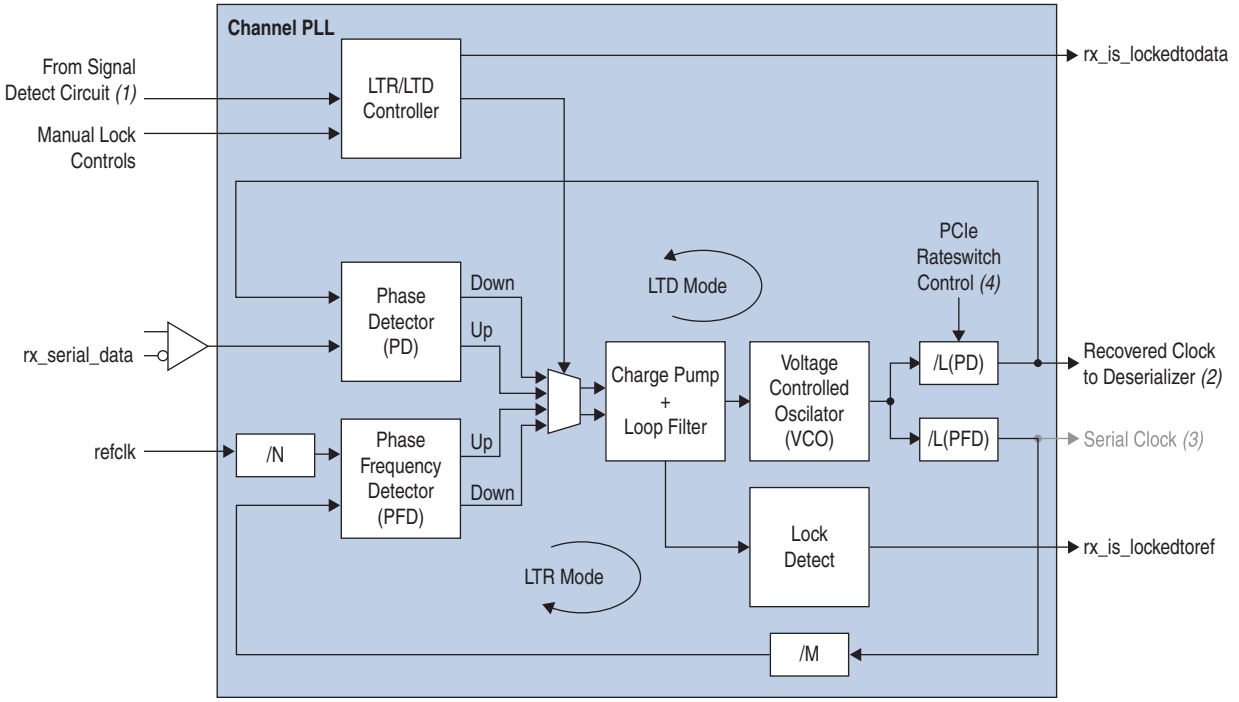
If you configure the channel PLL as a CDR PLL, the channel PLL recovers the clock and data from the serial data stream. If you do not use the channel PLL as a CDR PLL, you can configure it as a clock multiplier unit (CMU) PLL for clocking the transceivers.

For more information about the channel PLL operation when configured as a CMU PLL, refer to *"CMU PLL" on page 1–16*.

Channel PLL Architecture

Figure 1–12 shows the major components in a channel PLL. The channel PLL supports operation in either lock-to-reference (LTR) or lock-to-data (LTD) mode.

Figure 1–12. Channel PLL Block Diagram



Notes to Figure 1–12:

(1) Applicable only in a PCIe configuration.
(2) Applicable when configured as a CDR PLL.
(3) Applicable when configured as a CMU PLL.
(4) The PCIe rateswitch control allows dynamic switching between Gen2 and Gen1 line rates in a PCIe Gen2 design.

In LTR mode, the channel PLL tracks the input reference clock. The phase-frequency detector (PFD) compares the phase and frequency of the voltage controlled oscillator (VCO) output and the input reference clock. The resulting PFD output controls the VCO output frequency to half the data rate with the appropriate counter (M or L) value given an input reference clock frequency. The lock detect determines whether the PLL has achieved lock to the phase and frequency of the input reference clock.

In LTD mode, the channel PLL tracks the incoming serial data. The phase detector compares the phase of the VCO output and the incoming serial data. The resulting phase detector output controls the VCO output to continuously match the phase of the incoming serial data.

Use the LTR/LTD controller only when you configure the channel PLL as a CDR PLL.

Table 1–6 lists the counter values in the channel PLL.

Table 1–6. Counters in the Channel PLL *(1)*

Counter	Description	Values
N	Pre-scale counter to divide the input reference clock frequency to the PFD by the N factor	1, 2, 4, 8
M	Feedback loop counter to multiply the VCO frequency above the input reference frequency to the PFD by the M factor	4, 5, 8, 10, 12, 16, 20, 25
L (PFD)	VCO post-scale counter to divide the VCO output frequency by the L factor in the LTR loop	1, 2, 4, 8
L (phase detector)	VCO post-scale counter to divide the VCO output frequency by the L factor in the LTD loop	1, 2, 4, 8

Note to Table 1–6:

(1) The Quartus® II software automatically selects the appropriate counter values for each transceiver configuration.

CDR PLL Operation

The CDR PLL independently recovers the clock and data from the incoming serial data and sends the clock and data to the deserializer. The CDR PLL supports the full range of data rates.

The CDR PLL requires offset cancellation to correct the analog offset voltages that may exist from process variations between the positive and negative differential signals in the CDR circuitry.

For a detailed description of the offset cancellation process, refer to the *Dynamic Reconfiguration in Arria V Devices* chapter.

The CDR PLL operates either in LTR mode or LTD mode. After power-up or reset of the receiver PMA, the CDR PLL must first operate in LTR mode to keep the VCO output frequency close to the optimum recovered clock rate.

In LTR mode, the phase detector is not active. When the CDR PLL locks to the input reference clock, you can switch the CDR PLL to LTD mode to recover the clock and data from the incoming serial data.

In LTD mode, the PFD output is not valid and may cause the lock detect status indicator to toggle randomly. When there is no transition on the incoming serial data for an extended duration, you must switch the CDR PLL to LTR mode to wait for the real serial data.

The time needed for the CDR PLL to lock to data depends on the transition density and jitter of the incoming serial data and the parts per million (ppm) difference between the receiver input reference clock and the upstream transmitter reference clock. The receiver PCS must be held in reset until the CDR PLL locks to data and produces a stable recovered clock.

The LTR/LTD controller directs the CDR PLL transition between the LTR and LTD modes. The controller supports operation in both automatic lock mode and manual lock mode.

CDR PLL in Automatic Lock Mode

In automatic lock mode, the LTR/LTD controller directs the transition between the LTR and LTD modes when a set of conditions are met to ensure proper CDR PLL operation. The mode transitions are indicated by the `rx_is_lockedtodata` signal.

After power-up or reset of the receiver PMA, the CDR PLL is directed into LTR mode. The controller transitions the CDR PLL from LTR to LTD mode when all the following conditions are met:

- The frequency of the CDR PLL output clock and input reference clock is within the configured ppm frequency threshold setting.
- The phase of the CDR PLL output clock and input reference clock is within approximately 0.08 unit interval (UI) of difference.
- In PCIe configurations only—the signal detect circuitry must also detect the presence of the signal level at the receiver input above the threshold voltage specified in the PCI Express Base Specification 2.0.

The controller transitions the CDR PLL from LTD to LTR mode when either of the following conditions is met:

- The frequency of the CDR PLL output clock and input reference clock exceeds the configured ppm frequency threshold setting.
- In PCIe configurations only—the signal detect circuitry detects the signal level at the receiver input below the threshold voltage specified in the PCI Express Base Specification 2.0.

If there is no transition on the incoming serial data for an extended duration, the CDR output clock may drift to a frequency exceeding the configured ppm threshold when compared with the input reference clock. In such a case, the LTR/LTD controller transitions the CDR PLL from LTD to LTR mode.

CDR PLL in Manual Lock Mode

In manual lock mode, the LTR/LTD controller directs the transition between the LTR and LTD modes based on user-controlled settings in the `pma_rx_set_locktodata` and `pma_rx_set_locktoref` registers. Manual lock mode provides the flexibility to manually control the CDR PLL mode transitions bypassing the ppm detection as required by certain applications that include, but not limited to, the following:

- Link with frequency differences between the upstream transmitter and the local receiver clocks exceeding the CDR PLL ppm threshold detection capability. For example, a system with asynchronous spread-spectrum clocking (SSC) downspread of –0.5% where the SSC modulation results in a ppm difference of up to 5000.
- Link that requires a faster CDR PLL transition to LTD mode, avoiding the duration incurred by the ppm detection in automatic lock mode.

In manual lock mode, your design must include a mechanism—similar to a ppm detector—that ensures the CDR PLL output clock is kept close to the optimum recovered clock rate before recovering the clock and data. Otherwise, the CDR PLL might not achieve locking to data. If the CDR PLL output clock frequency is detected as not close to the optimum recovered clock rate in LTD mode, direct the CDR PLL to LTR mode.

For information about the proper sequence after power-up reset, refer to the *Transceiver Reset Control and Power-Down in Arria V Devices* chapter.

Deserializer

The deserializer provides serial-to-parallel data conversion and assumes the data is received LSB first from the receiver buffer. Additionally, the deserializer provides the clock-slip feature.

Clock-Slip

Word alignment in the PCS may contribute up to one parallel clock cycle of latency uncertainty. The clock-slip feature allows word alignment operation with a reduced latency uncertainty by performing the word alignment function in the deserializer. Use the clock slip feature for applications that require deterministic latency.

The deterministic latency state machine in the word aligner from the PCS automatically controls the clock-slip operation. After completing the clock-slip process, the deserialized data is word-aligned into the receiver PCS.

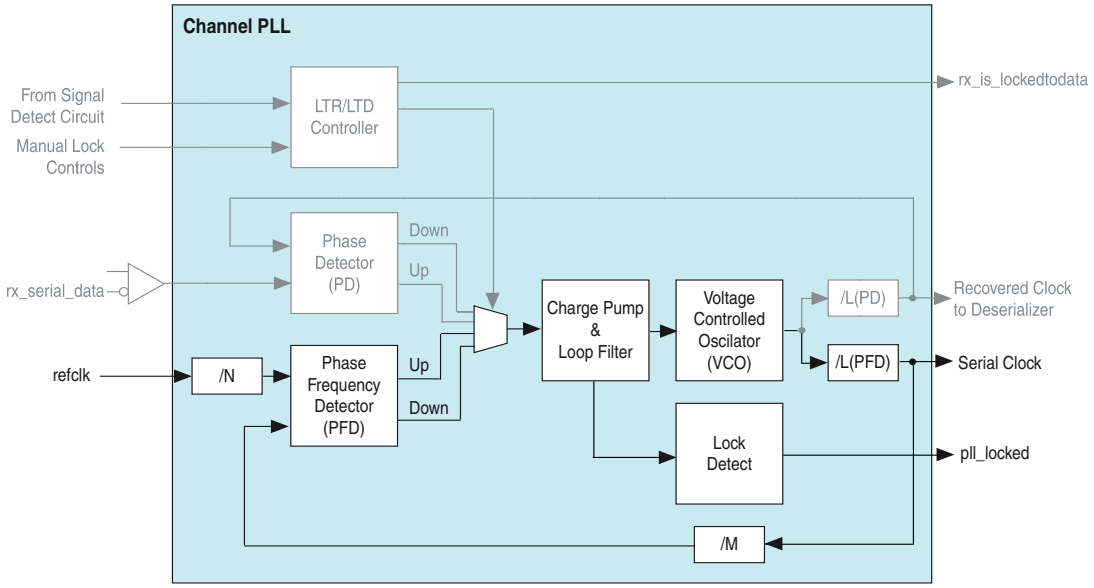
CMU PLL

In Arria V GX devices, if you do not use the channel PLL as a CDR, you can independently configure every channel PLL as a CMU PLL for clocking the transceivers. In Arria V GT devices, the PLL in 10-Gbps transceiver channels can only be configured as a CDR. The CMU PLL for clocking the 10-Gbps channels must be from channel 1 or 4.

CDR functionality for the receiver is not available when you configure the channel PLL as a CMU PLL—you can use the transceiver channel only as a transmitter.

The CMU PLL operates only in lock-to-reference (LTR) mode and supports the full range of data rates. Figure 1–13 shows the Arria V channel PLL configured as a CMU PLL.

Figure 1–13. CMU PLL in Arria V Devices



For more information about the architecture of the channel PLL, refer to the"Channel PLL" on page 1–12.

Using the input reference clock, the CMU PLL synthesizes the serial clock with a frequency that is half of the data rate. The CMU PLL output serial clock feeds the clock divider that resides in the transmitter of the same transceiver channel. Depending on the channel location in a transceiver bank, the CMU PLL of channels 1 and 4 feeds the output clock to the x1 clock lines.

For more information about the input reference clock and transmit PLL, refer to the *Transceiver Clocking in Arria V Devices* chapter.

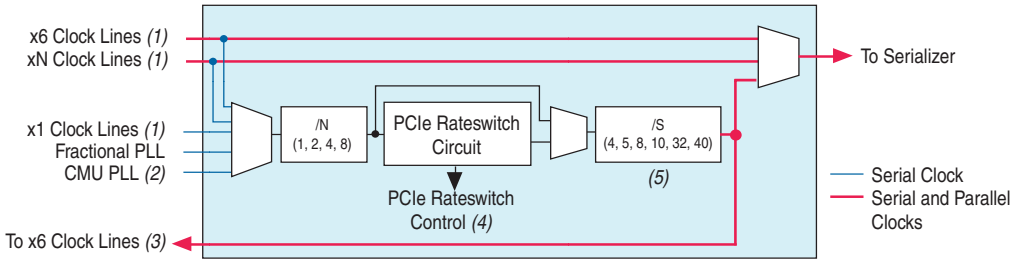
Clock Divider

Each Arria V transmitter channel has a clock divider. There are two types of clock dividers, depending on the channel location in a transceiver bank:

- Local clock divider—channels 0, 2, 3, and 5 provide serial and parallel clocks to the PMA
- Central clock divider—channels 1 and 4 can drive the x6 clock lines

Figure 1–14 shows the clock divider for a transceiver channel in Arria V devices.

Figure 1–14. Clock Divider for a Transceiver Channel in Arria V Devices



Notes to Figure 1–14:

(1) For information about the x1, x6, and xN clock lines, refer to the *Transceiver Clocking in Arria V Devices* chapter.
(2) Only from the channel PLL in the same transceiver channel configured as a CMU PLL.
(3) Applicable for central clock dividers only (clock dividers in channels 1 and 4).
(4) The PCIe rateswitch circuit allows dynamic switching between Gen2 and Gen1 line rates in a PCIe Gen2 design.
(5) The divider settings are configured automatically depending on the serialization factor. The selected divider setting is half of the serialization factor. The 32 and 40 divider settings are only available for 10-Gbps channels in Arria V GT devices.

Both types of clock dividers can divide the serial clock input to provide the parallel and serial clocks for the serializer in the channel if you use clocks from the clock lines or transmit PLLs. The central clock divider can additionally drive the x6 clock lines used to bond multiple channels.

In bonded channel configurations, both types of clock dividers can feed the serializer with the parallel and serial clocks directly, without dividing them from the x6 or xN clock lines.

In Arria V GT devices, the clock divider only receives serial clock from x1 clock line at 10-Gbps operation.

Calibration Block

Up to two calibration blocks are available for the Arria V transceiver PMA. The calibration block calibrates the differential OCT resistance and analog circuitry in the transceiver PMA to ensure the functionality is independent of PVT.

Figure 1–15 and Figure 1–16 show the calibration block location, the transceiver banks calibrated by the calibration block, and the required external connection on the RREF pin.

Figure 1–15. Calibration Block Location and Connections in Arria V Devices with Transceivers on the Left Side of the Device Only

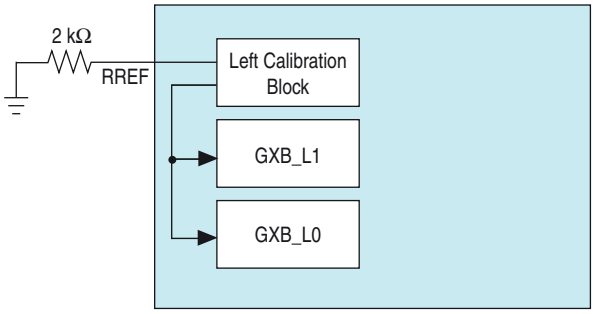
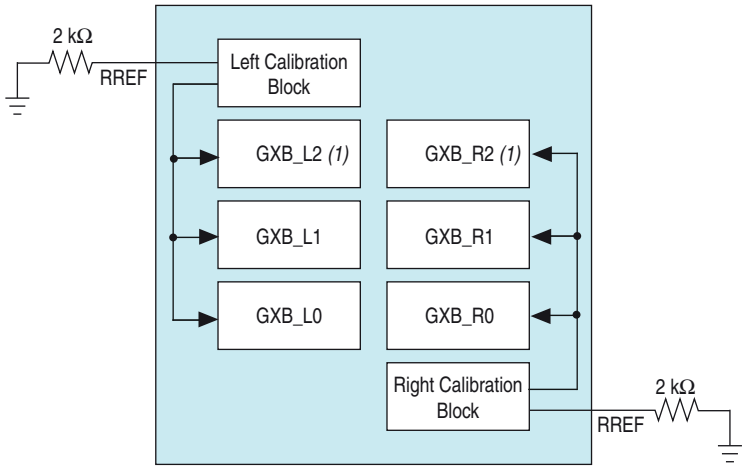


Figure 1–16. Calibration Block Location and Connections in Arria V Devices with Transceivers on Both Sides of the Device



Note to Figure 1–16:

(1) GXB_L2 and GXB_R2 banks are only available in some device variants.

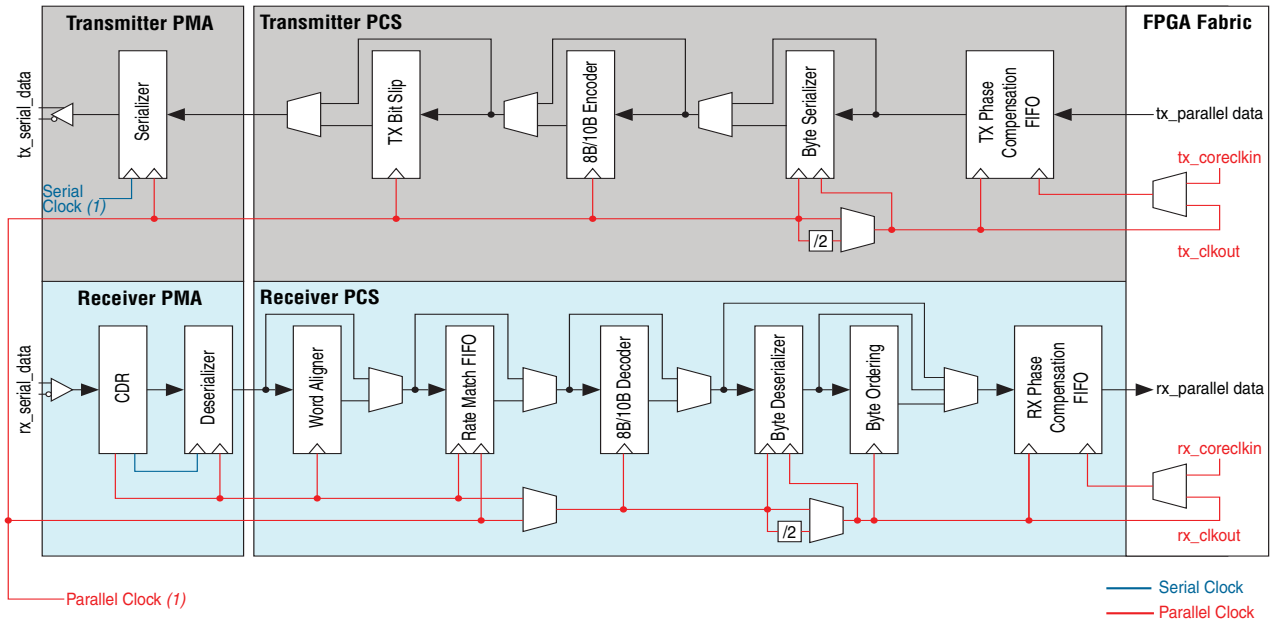
The calibration block generates a constant internal reference voltage that is independent of PVT variations using the external reference resistor. The resulting reference currents are used to calibrate the transceiver banks.

You must connect a separate 2 kΩ (tolerance maximum of ±1%) external resistor on each RREF pin to ground. To ensure the calibration block operates properly, the RREF resistor connection in the board must be free from external noise.

PCS Architecture

Figure 1–17 shows the PCS transceiver channel in Arria V devices.

Figure 1–17. Transceiver Channel PCS in Arria V Devices



Note to Figure 1–17:

(1) The serial and parallel clocks are sourced from the clock divider.

The PCS is not available when using the 10-Gbps channels; only the PMA is available. You must implement the PCS functions required for the interface using user logic in the FPGA fabric with an 80-bit FPGA fabric–transceiver interface width.

Table 1–7 lists the two transceiver channel PCS datapath configurations based on the transceiver channel PMA-PCS width (or the SERDES factor).

Table 1–7. PCS Datapath Configurations

Parameter	Single-Width	Double-Width
PMA–PCS Interface Width	8 or 10 bit	16 or 20 bit
FPGA Fabric–Transceiver Interface Width	8 or 10 bit 16 or 20 bit (1)	16 or 20 bit 32 or 40 bit (1)

Note to Table 1–7:

(1) The byte serializer and deserializer are enabled.

Transmitter PCS Datapath

This section describes the transmitter phase compensation FIFO, byte serializer, 8B/10B encoder, and transmitter bit-slip blocks in the transmitter PCS datapaths. Table 1–8 lists the blocks of the Arria V transmitter PCS datapath.

Table 1–8. Functional Blocks in the 6-Gbps Transmitter PCS Datapath

Block	Functionality
Transmitter Phase Compensation FIFO	■ Compensates for the phase difference between the low-speed parallel clock and the FPGA fabric interface clock when interfacing the transmitter PCS with the FPGA fabric directly or with the PCIe hard IP block ■ Supports operation in phase compensation and registered modes
Byte Serializer	■ Divides the FPGA fabric–transceiver interface frequency in half at the transmitter channel by doubling the transmitter input datapath width ■ Allows the transmitter channel to operate at higher data rates with the FPGA fabric–transceiver interface frequency that is within the maximum limit ■ Supports operation in double-width modes
8B/10B Encoder	■ Generates 10-bit code groups from 8-bit data and 1-bit control identifier, in compliance with Clause 36 of the IEEE 802.3 specification ■ Supports operation in single- and double-width modes, and running disparity control
Transmitter Bit-Slip	■ Enables user-controlled, bit-level delay in the data prior to serialization for serial transmission ■ Supports operation in single- and double-width modes

For more information about the transmitter PCS datapath, refer to the"Transmitter PCS Datapath" on page 1–21.

Transmitter Phase Compensation FIFO

The transmitter phase compensation FIFO is four words deep and interfaces the control and data signals between the transmitter PCS and FPGA fabric or PCIe hard IP block. The FIFO supports the following operations:

- Phase compensation mode with various clocking modes on the read clock and write clock
- Registered mode with only one clock cycle of datapath latency

Phase Compensation Mode

The transmitter phase compensation FIFO compensates any phase difference between the read and write clocks for the transmitter control and data signals. The low-speed parallel clock feeds the read clock; the FPGA fabric interface clock feeds the write clock. The clocks must have 0 ppm difference in frequency or a FIFO underrun or overflow condition may result.

The transmitter phase compensation FIFO supports various clocking modes on the read and write clocks depending on the transceiver configuration.

For a detailed description of transmitter datapath interface clocking modes when using the transmitter phase compensation FIFO, refer to the *Transceiver Clocking in Arria V Devices* chapter.

Registered Mode

To eliminate the FIFO latency uncertainty for applications with stringent datapath latency uncertainty requirements, bypass the FIFO functionality in registered mode to incur only one clock cycle of datapath latency when interfacing the transmitter channel to the FPGA fabric. Configure the FIFO to registered mode when interfacing the transmitter channel to the PCIe hard IP block to reduce datapath latency. In registered mode, the low-speed parallel clock that is used in the transmitter PCS clocks the FIFO.

Byte Serializer

The byte serializer allows the transmitter channel to operate at higher data rates in a configuration that exceeds the FPGA fabric–transceiver interface frequency limit. The byte serializer supports operation in single- and double-width modes. The datapath clock rate at the output of the byte serializer is twice the FPGA fabric–transmitter interface clock frequency.

You must use the byte serializer in configurations that exceed the maximum frequency limit of the FPGA fabric–transceiver interface.

Table 1–9 lists the byte serializer datapath conversion for the transmitter input datapath width in single- and double-width modes.

Table 1–9. Transmitter Input Datapath Conversion

Mode	Transmitter Input Datapath Width	Byte Serializer Output Datapath Width	Byte Serializer Output Ordering
Single Width	16	8	Least significant 8 bits of the 16-bit input first
	20	10	Least significant 10 bits of the 20-bit input first
Double Width	32	16	Least significant 16 bits of the 32-bit input first
	40	20	Least significant 20 bits of the 40-bit input first

8B/10B Encoder

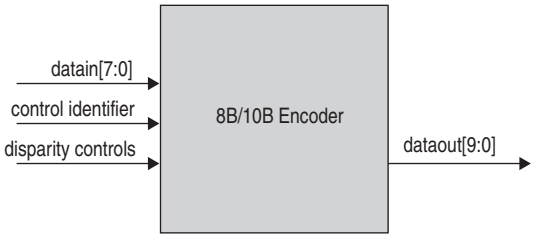
The 8B/10B encoder supports operation in single- and double-width modes with the running disparity control feature.

8B/10B Encoder in Single-Width Mode

In single-width mode, the 8B/10B encoder generates 10-bit code groups from 8-bit data and 1-bit control identifier with proper disparity according to the PCS reference diagram in Clause 36 of the IEEE 802.3 specification. The 10-bit code groups are generated as valid data code-groups (/Dx.y/) or special control code-groups (/Kx.y/), depending on the 1-bit control identifier.

Figure 1–18 shows a simplified diagram of the 8B/10B encoder in single-width mode.

Figure 1–18. 8B/10B Encoder Diagram in Single-Width Mode



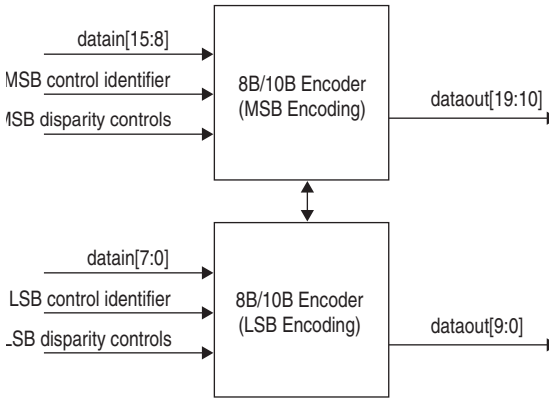
The IEEE 802.3 specification identifies only 12 sets of 8-bit characters as /Kx.y/. If other sets of 8-bit characters are set to encode as special control code-groups, the 8B/10B encoder may encode the output 10-bit code as an invalid code (it does not map to a valid /Dx.y/ or /Kx.y/ code), or unintended valid /Dx.y/ code, depending on the value entered.

8B/10B Encoder in Double-Width Mode

In double-width mode, two 8B/10B encoders are cascaded to generate two sets of 10-bit code groups from 16-bit data and two 1-bit control identifiers. When receiving the 16-bit data, the 8-bit LSByte is encoded first, followed by the 8-bit MSByte.

Figure 1–19 shows a simplified diagram of the 8B/10B encoder in double-width mode.

Figure 1–19. 8B/10B Encoder Diagram in Double-Width Mode



Running Disparity Control

The 8B/10B encoder automatically performs calculations that meet the running disparity rules when generating the 10-bit code groups. The running disparity control feature provides user-controlled signals (`tx_dispval` and `tx_forcedisp`) to manually force encoding into a positive or negative current running disparity code group. When enabled, the control overwrites the current running disparity value in the encoder based on user-controlled signals, regardless of the internally-computed current running disparity in that cycle.

Using the running disparity control may temporarily cause a running disparity error at the receiver.

Encoder Output During Reset Sequence

Figure 1–20 shows the 8B/10B encoder output during and after reset conditions in both single- and double-width modes.

Figure 1–20. 8B/10B Encoder Output During and After Reset Conditions

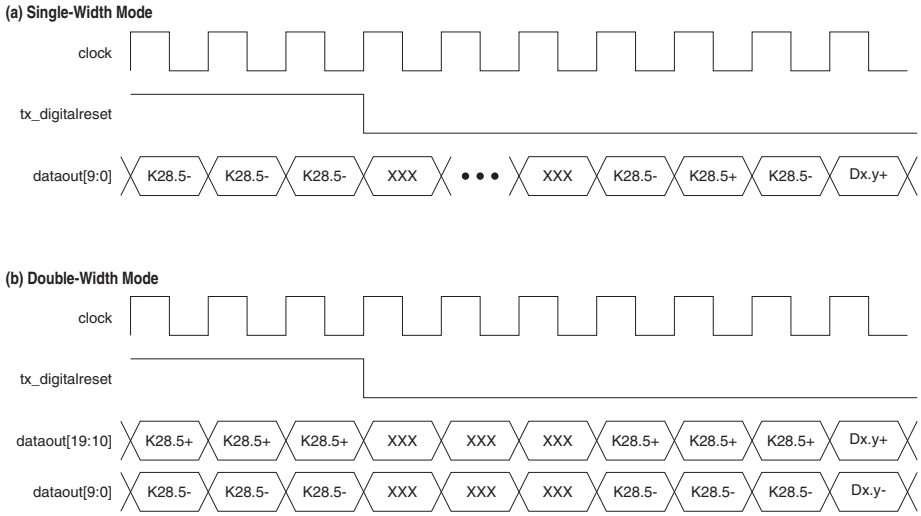


Table 1–10 lists the 8B/10B encoder output during and after reset conditions.

Table 1–10. 8B/10B Encoder Output During and After Reset Conditions

Operation Mode	During 8B/10B Reset	After 8B/10B Reset Release
Single Width	Continuously sends the /K28.5/ code from the RD– column	Some “don’t cares” due to pipelining in the transmitter channel, followed by three /K28.5/ codes with proper disparity—starts with negative disparity—before sending encoded 8-bit data at its input.
Double Width	Continuously sends the /K28.5/ code from the RD– column on LSByte and the /K28.5/ code from the RD+ column on MSByte	Some “don't cares” due to pipelining in the transmitter channel, followed by: ■ Three /K28.5/ codes from the RD– column before sending encoded 8-bit data at its input on LSByte. ■ Three /K28.5/ codes from the RD+ column before sending encoded 8-bit data at its input on MSByte.

Transmitter Bit-Slip

The transmitter bit-slip enables a bit-level delay insertion to the data prior to serialization for the serial transmission. The transmitter bit-slip supports operation in single- and double-width modes. Each bit slipped at the transmitter incurs one serial bit of datapath latency. Table 1–11 lists the number of bits allowed to be slipped with the `tx_bitslipboundaryselect` signal.

Table 1–11. Bits Slip Allowed with the tx_bitslipboundaryselect Signal

Operation Mode	Maximum Bit-Slip Setting
Singe width (8- or 10-bit)	9
Double width (16- or 20-bit)	19

Receiver PCS Datapath

This section describes the word aligner, rate match FIFO, 8B/10B decoder, byte deserializer, byte ordering, and receiver phase compensation FIFO blocks in the receiver PCS datapaths. Table 1–12 lists the blocks of the Arria V receiver PCS datapath.

Table 1–12. Functional Blocks in the 6-Gbps Receiver PCS Datapath

Block	Functionality
Word Aligner	■ Searches for a predefined alignment pattern in the deserialized data to identify the correct boundary and restores the word boundary during link synchronization ■ Supports an alignment pattern length of 7, 8, 10, 16, 20, or 32 bits ■ Supports operation in four modes—manual alignment, bit-slip, automatic synchronization state machine, and deterministic latency state machine—in single- and double-width configurations ■ Supports the optional programmable run-length violation detection, polarity inversion, bit reversal, and byte reversal features
Rate Match FIFO	■ Compensates for small clock frequency differences of up to ±300 ppm—600 ppm total—between the upstream transmitter and the local receiver clocks by inserting or deleting skip symbols when necessary ■ Supports operation that is compliant to the clock rate compensation function in supported protocols
8B/10B Decoder	■ Receives 10-bit data and decodes the data into an 8-bit data and a 1-bit control identifier—in compliance with Clause 36 of the IEEE 802.3 specification ■ Supports operation in single- and double-width modes
Byte Deserializer	■ Divides the FPGA fabric–transceiver interface frequency in half at the receiver channel by doubling the receiver output datapath width ■ Allows the receiver channel to operate at higher data rates with the FPGA fabric–transceiver interface frequency that is within the maximum limit ■ Supports operation in double-width modes
Byte Ordering	■ Searches for a predefined pattern that must be ordered to the LSByte position in the parallel data going to the FPGA fabric when you enable the byte deserializer
Receiver Phase Compensation FIFO	■ Compensates for the phase difference between the low-speed parallel clock and the FPGA fabric interface clock when interfacing the receiver PCS with the FPGA fabric directly or with the PCIe hard IP block ■ Supports operation in phase compensation and registered modes

For more information about the receiver PCS datapath, refer to the"Receiver PCS Datapath" on page 1–25.

Word Aligner

Parallel data at the input of the receiver PCS loses the word boundary of the upstream transmitter from the serial-to-parallel conversion in the deserializer. The word aligner provides word boundary restoration during link synchronization with the following four modes:

- Manual alignment—automatic synchronization to new word boundary
- Bit-slip—manual data slip control
- Automatic synchronization state machine—automatic synchronization to a new word boundary with the programmable state machine for hysteresis control
- Deterministic latency state machine—automatic synchronization to a new word boundary for applications with a stringent latency uncertainty requirement

The word aligner searches for a predefined alignment pattern in the deserialized data to identify the correct boundary and restores the word boundary during link synchronization. The alignment pattern is predefined for standard serial protocols according to the respective protocol specifications to achieve synchronization or you can specify the settings with a custom word alignment pattern for proprietary protocol implementations. Except for bit-slip mode, after completing word alignment, the deserialized data is synchronized to have the word alignment pattern at the LSB portion of the aligned data.

In addition to restoring the word boundary, the word aligner supports the optional features listed in Table 1–13.

Table 1–13. Optional Word Aligner Features

Feature	Availability
Programmable Run-Length Violation Detection	All transceiver configurations
Receiver Polarity Inversion	All transceiver configurations except PCIe
Receiver Bit Reversal	Custom single- and double-width configurations only
Receiver Byte Reversal	Custom double-width configuration only

The operation mode and alignment pattern length support varies depending on the word aligner configurations. Table 1–14 lists the operation mode and alignment pattern length support in single- and double-width configurations.

Table 1–14. Word Aligner Operation Mode and Pattern Length Support (Part 1 of 2)

PCS Mode	PMA–PCS Interface Width	Word Aligner Mode	Alignment Pattern Length
Singe Width	8 bits	Manual alignment	8 or 16 bits
		Bit-slip	16 bits
	10 bits	Manual alignment	7 or 10 bits
		Bit-slip	7 or 10 bits
		Automatic synchronization state machine	7 or 10 bits
		Deterministic latency state machine	10 bits

Table 1–14. Word Aligner Operation Mode and Pattern Length Support (Part 2 of 2)

PCS Mode	PMA–PCS Interface Width	Word Aligner Mode	Alignment Pattern Length
Double Width	16 bits	Manual alignment	8, 16, or 32 bits
		Bit-slip	8, 16, or 32 bits
	20 bits	Manual alignment	7, 10, or 20 bits
		Bit-slip	7, 10, or 20 bits
		Deterministic latency state machine	10 or 20 bits

Word Aligner in Manual Alignment Mode

In manual alignment mode, the word alignment operation is manually controlled with the `rx_enapatternalign` register. Depending on the configuration, controlling the `rx_enapatternalign` register enables the word aligner to look for the predefined word alignment pattern in the received data stream and automatically synchronizes to the new word boundary.

Table 1–15 lists the word aligner operations in manual alignment mode.

Table 1–15. Word Aligner Operations in Manual Alignment Mode (Part 1 of 2)

PCS Mode	PMA–PCS Interface Width	Word Alignment Operation
Single Width	8 bits	1. After the `rx_digitalreset` signal deasserts, a 0-to-1 transition to the `rx_enapatternalign` register triggers the word aligner to look for the predefined word alignment pattern in the received data stream and automatically synchronize to the new word boundary. 2. Any alignment pattern found thereafter in a different word boundary does not cause the word aligner to resynchronize to this new word boundary because there is a lack of a preceding 0-to-1 transition on the `rx_enapatternalign` register. 3. To resynchronize to the new word boundary, create a 0-to-1 transition to the `rx_enapatternalign` register. 4. If you set the `rx_enapatternalign` register to 1 before the deassertion of the `rx_digitalreset` signal, the word aligner updates the word boundary when the first alignment pattern is found, even though a 0-to-1 transition was not explicitly generated. 5. To resynchronize to the new word boundary, create a 0-to-1 transition to the `rx_enapatternalign` register.
	10 bits	1. After the `rx_digitalreset` signal deasserts, setting 1 to the `rx_enapatternalign` register triggers the word aligner to look for the predefined word alignment pattern, or its complement in the received data stream, and automatically synchronizes to the new word boundary. 2. Any alignment pattern found thereafter in a different word boundary causes the word aligner to resynchronize to this new word boundary if the `rx_enapatternalign` register remains set to 1. 3. If you set the `rx_enapatteralign` register to 0, the word aligner maintains the current word boundary even when it finds the alignment pattern in a new word boundary.

Table 1–15. Word Aligner Operations in Manual Alignment Mode (Part 2 of 2)

PCS Mode	PMA–PCS Interface Width	Word Alignment Operation
Double Width	16 bits	1. After the rx_digitalreset signal deasserts, regardless of the setting in the rx_enapatternalign register, the word aligner synchronizes to the first predefined alignment pattern found. 2. Any alignment pattern found thereafter in a different word boundary does not cause the word aligner to resynchronize to this new word boundary. 3. To resynchronize to the new word boundary, create a 0-to-1 transition to the rx_enapatternalign register.
	20 bits	

Bit-Slip Mode

In bit-slip mode, the word alignment is achieved by manually controlling the data slip with the rx_bitslip signal. Slipping the received data by one bit effectively shifts the word boundary by one bit. You can implement a controller in the FPGA fabric to iteratively control the rx_bitslip signal until the word aligner output matches the predefined word alignment pattern to achieve synchronization. Table 1–16 lists the word aligner operations in bit-slip mode.

Table 1–16. Word Aligner Operations in Bit-Slip Mode

PCS Mode	PMA–PCS Interface Width	Word Alignment Operation
Single Width	8 bits	1. At every rising edge to the rx_bitslip signal, the word aligner slips one bit into the received data. 2. When bit-slipping shifts a complete round of the data bus width, the word boundary is back to the original boundary. 3. Use the rx_patterndetect signal assertion or check the data output to indicate completion of the alignment process—where the word aligner output matches the predefined alignment pattern.
	10 bits	
Double Width	16 bits	
	20 bits	

For every bit slipped in the word aligner, the earliest bit received is lost.

Word Aligner in Automatic Synchronization State Machine Mode

In automatic synchronization state machine mode, a programmable state machine determines when the word aligner has either achieved synchronization or lost synchronization. You can configure the state machine to provide hysteresis control during link synchronization and throughout normal link operation. Depending on your protocol configurations, the state machine parameters are automatically configured to be compliant to the synchronization state machine specified in the respective protocol specification.

Table 1–17 lists the programmable state machine parameters for the word aligner in automatic synchronization state machine mode.

Table 1–17. State Machine Parameters for the Word Aligner in Automatic Synchronization State Machine Mode (Part 1 of 2)

Parameter	Values
Number of valid synchronization code groups or ordered sets received to achieve synchronization	1–256

Table 1–17. State Machine Parameters for the Word Aligner in Automatic Synchronization State Machine Mode (Part 2 of 2)

Parameter	Values
Number of erroneous code groups received to lose synchronization	1–64
Number of continuous good code groups received to reduce the error count by one	1–256

Table 1–18 lists the word aligner operations in automatic synchronization state machine mode.

Table 1–18. Word Aligner Operation in Automatic Synchronization State Machine Mode

PCS Mode	PMA–PCS Interface Width	Word Alignment Operation
Single Width	10 bits	1. After the rx_digitalreset signal deasserts, the word aligner starts looking for the predefined word alignment pattern, or its complement, in the received data stream and automatically aligns to the new word boundary. 2. Synchronization is achieved only after the word aligner receives the programmed number of valid synchronization code groups in the same word boundary and is indicated by the assertion of the rx_syncstatus signal. 3. After assertion and achieving synchronization, the rx_syncstatus signal remains asserted until the word aligner loses synchronization. 4. Loss of synchronization occurs when the word aligner receives the programmed number of erroneous code groups without receiving the intermediate good code groups and is indicated by the deassertion of the rx_syncstatus signal. 5. The word aligner may achieve synchronization again after receiving a new programmed number of valid synchronization code groups in the same word boundary.

Word Aligner in Deterministic Latency State Machine Mode

In deterministic latency state machine mode, word alignment is achieved by performing a clock-slip in the deserializer until the deserialized data coming into the receiver PCS is word-aligned. The state machine controls the clock-slip process in the deserializer after the word aligner has found the alignment pattern and identified the word boundary. Deterministic latency state machine mode offers a reduced latency uncertainty in the word alignment operation for applications that require deterministic latency.

Table 1–19 lists the word aligner operations in deterministic latency state machine mode.

Table 1–19. Word Aligner Operations in Deterministic Latency State Machine Mode

PCS Mode	PMA–PCS Interface Width	Word Alignment Operation
Single Width	10 bits	1. After the rx_digitalreset signal deasserts, a 0-to-1 transition to the rx_enapatternalign register triggers the word aligner to look for the predefined word alignment pattern or its complement in the received data stream. 2. After the pattern is found and the word boundary is identified, the state machine controls the deserializer to clock-slip the boundary-indicated number of serial bits. 3. When the clock-slip is complete, the deserialized data coming into the receiver PCS is word-aligned and is indicated by the assertion of the rx_syncstatus signal.
Double Width	20 bits	

Programmable Run-Length Violation Detection

The programmable run-length violation detection circuit detects if the number of consecutive 1s or 0s in the received data exceeds the user-specified threshold. Table 1–20 lists the detection capabilities of the run-length violation circuit.

Table 1–20. Detection Capabilities of the Run-Length Violation Circuit

PCS Mode	PMA–PCS Interface Width	Run-Length Violation Detector Range	
		Minimum	Maximum
Single Width	8 bits	4	128
	10 bits	5	160
Double Width	16 bits	8	512
	20 bits	10	640

Receiver Polarity Inversion

The positive and negative signals of a serial differential link might accidentally be swapped during the layout of the board. The polarity inversion feature at the receiver can correct this error without requiring board respin or major updates to the logic in the FPGA fabric. The polarity inversion feature inverts the polarity of every bit at the input to the word aligner, which has the same effect as swapping the positive and negative signals of the serial differential link.

Inversion is controlled dynamically with the `rx_invpolarity` register. Enabling the polarity inversion feature may cause initial disparity errors at the receiver with the 8B/10B-coded data. The receiver must be able to tolerate these disparity errors.

If the polarity inversion is enabled midway through a word, the word will be corrupted.

Bit Reversal

You can reverse the bit order at the output of the word aligner for receiving a MSB-to-LSB transmission using the bit reversal feature at the receiver. Table 1–21 lists the bit reversal feature options.

Table 1–21. Bit Reversal Feature

Bit Reversal Option	Received Bit Order	
	Single-Width Mode (8- or 10-bit)	Double-Width Mode (16- or 20-bit)
Disabled (default)	LSB to MSB	LSB to MSB
Enabled	MSB to LSB For example: 8-bit—`D[7:0]` rewired to `D[0:7]` 10-bit—`D[9:0]` rewired to `D[0:9]`	MSB to LSB For example: 16-bit—`D[15:0]` rewired to `D[0:15]` 20-bit—`D[19:0]` rewired to `D[0:19]`

When receiving the MSB-to-LSB transmission, the word aligner receives the data in reverse order. The word alignment pattern must be reversed accordingly to match the MSB-first incoming data ordering.

You can dynamically control the bit reversal feature using the rx_bitreversal_enable register with the word aligner in bit-slip mode. When you dynamically enable the bit reversal feature in bit-slip mode, you can ignore the pattern detection function in the word aligner because the word alignment pattern cannot be dynamically reversed to match the MSB-first incoming data ordering.

Receiver Byte Reversal

The two symbols of incoming data at the receiver in double-width mode may be accidentally swapped during transmission. For a 16-bit input data width at the word aligner, the two symbols are bits[15:8] and bits[7:0]. For a 20-bit input data width at the word aligner, the two symbols are bits[19:10] and bits[9:0]. The byte reversal feature at the word aligner output can correct this error by swapping the two symbols in double-width mode at the word aligner output, as listed in Table 1–22.

Table 1–22. Byte Reversal Feature

Byte Reversal Option	Word Aligner Output	
	16-bit Data Width	20-bit Data Width
Disabled	D[15:0]	D[19:0]
Enabled	D[7:0], D[15:8]	D[9:0], D[19:10]

The reversal is controlled dynamically using the rx_bytereversal_enable register. Enabling the byte reversal option may cause initial disparity errors at the receiver with 8B/10B-coded data. The receiver must be able to tolerate these disparity errors.

When receiving swapped symbols, the word alignment pattern must be byte-reversed to match the incoming byte-reversed data.

Rate Match FIFO

In a link where the upstream transmitter and local receiver can be clocked with independent reference clock sources, the data can be corrupted by any frequency differences (in ppm count) when crossing the data from the recovered clock domain—the same clock domain as the upstream transmitter reference clock—to the local receiver reference clock domain.

The rate match FIFO is 20 words deep, which compensates for the small clock frequency differences of up to ±300 ppm (600 ppm total) between the upstream transmitter and the local receiver clocks by performing symbol insertion or deletion, depending on the ppm difference on the clocks.

The rate match FIFO operation requires that the transceiver channel is in duplex configuration (both transmit and receive functions) and a predefined 20-bit pattern (consisting of a 10-bit control pattern and a 10-bit skip pattern). The 10-bit skip pattern must be chosen from a code group with neutral disparity.

The FIFO operates by looking for the 10-bit control pattern, followed by the 10-bit skip pattern in the data after the word aligner has restored the word boundary. After finding the pattern, the FIFO performs the following operations to ensure the FIFO does not underflow or overflow:

- Inserts the 10-bit skip pattern when the local receiver reference clock frequency is greater than the upstream transmitter reference clock frequency
- Deletes the 10-bit skip pattern when the local receiver reference clock frequency is less than the upstream transmitter reference clock frequency

The rate match FIFO supports operations in single- and double-width modes. You can define the 20-bit pattern for custom configurations. For protocol configurations, the FIFO is automatically configured to support a clock rate compensation function as required by the following specifications:

- The PCIe protocol per clock tolerance compensation requirement, as specified in the PCI Express Base Specification 2.0 for Gen1 and Gen2 signaling rates
- The Gbps Ethernet (GbE) protocol per clock rate compensation requirement using an idle ordered set, as specified in Clause 36 of the IEEE 802.3 specification

For more information about the rate match FIFO operation in custom and protocol-specific configurations, refer to the *Transceiver Custom Configurations in Arria V Devices* and the *Transceiver Protocol Configurations in Arria V Devices* chapters, respectively.

8B/10B Decoder

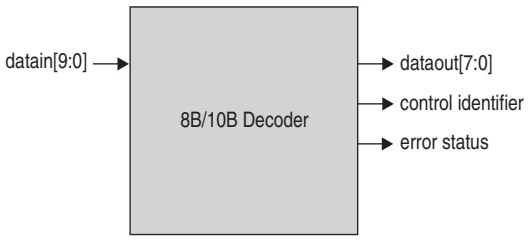
The 8B/10B decoder supports operation in single- and double-width modes.

8B/10B Decoder in Single-Width Mode

In single-width mode, the 8B/10B decoder decodes the received 10-bit code groups into an 8-bit data and a 1-bit control identifier in compliance with Clause 36 in the IEEE 802.3 specification. The 1-bit control identifier indicates if the decoded 8-bit code is a valid data or special control code.

Figure 1–21 shows a simplified 8B/10B decoder block diagram in single-width mode.

Figure 1–21. 8B/10B Decoder Block Diagram in Single-Width Mode

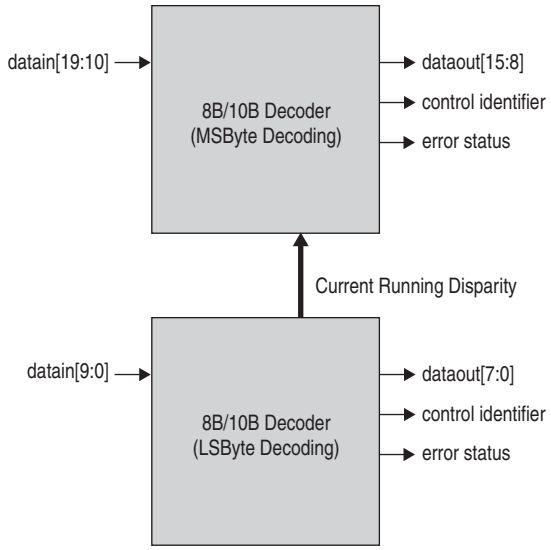


8B/10B Decoder in Double-Width Mode

In double-width mode, two 8B/10B decoders are cascaded to decode the 20-bit code groups into two sets of 8-bit data and two 1-bit control identifiers. When receiving the 20-bit code group, the 10-bit LSByte is decoded first and the ending running disparity is forwarded to the other 8B/10B decoder for decoding the 10-bit MSByte.

Figure 1–22 shows a simplified 8B/10B decoder block diagram in double-width mode.

Figure 1–22. 8B/10B Decoder Block Diagram in Double-Width Mode



Byte Deserializer

The byte deserializer allows the receiver channel to operate at higher data rates in a configuration that exceeds the FPGA fabric–transceiver interface frequency limit. The byte deserializer supports operation in single- and double-width modes. The datapath clock rate at the input of the byte deserializer is twice the FPGA fabric–receiver interface clock frequency.

You must use the byte deserializer in configurations that exceed the maximum frequency limit of the FPGA fabric–transceiver interface.

After byte deserialization, the word alignment pattern may be ordered in the MSByte or LSByte position.

For applications that require deterministic latency, you can configure the byte deserializer to order the word alignment pattern only in the LSByte position after byte deserialization. To achieve this, the data byte prior to the alignment pattern is dropped if the word alignment pattern is found in the MSByte position. In this configuration, there is no latency uncertainty in the byte deserializer operation.

Table 1–23 lists the byte deserializer conversion for the byte deserializer input datapath width in single- and double-width modes. The data is assumed to be received as LSByte first—the least significant 8 or 10 bits in single-width mode or the least significant 16 or 20 bits in double-width mode.

Table 1–23. Byte Deserializer Input Datapath Width Conversion (Part 1 of 2)

Mode	Byte Deserializer Input Datapath Width	Receiver Output Datapath Width
Single Width	8	16
	10	20

Table 1–23. Byte Deserializer Input Datapath Width Conversion (Part 2 of 2)

Mode	Byte Deserializer Input Datapath Width	Receiver Output Datapath Width
Double Width	16	32
	20	40

Byte Ordering

When you enable the byte deserializer, the output byte order may not match the originally transmitted ordering. For applications that require a specific pattern to be ordered at the LSByte position of the data, byte ordering can restore the proper byte order of the byte-deserialized data before forwarding it to the FPGA fabric.

Byte ordering operates by inserting a predefined pad pattern to the byte-deserialized data if the predefined byte ordering pattern found is not in the LSByte position.

Byte ordering requires the following:

- A receiver with the byte deserializer enabled
- A predefined byte ordering pattern that must be ordered at the LSByte position of the data
- A predefined pad

Byte ordering supports operation in single- and double-width modes. Both modes support operation in word aligner-based and manual ordering modes.

Byte Ordering in Single-Width Mode

Table 1–24 lists the byte ordering operation in single-width mode.

Table 1–24. Byte Ordering Operation in Single-Width Mode (1)

PMA–PCS Interface Width	FPGA Fabric–Transceiver Interface Width	8B/10B Decoder	Byte Ordering Pattern Length	Pad Pattern Length
8 bits	16 bits	Disabled	8 bits	8 bits
10 bits	16 bits	Enabled	9 bits (2)	9 bits (2)
	20 bits	Disabled	10 bits	10 bits

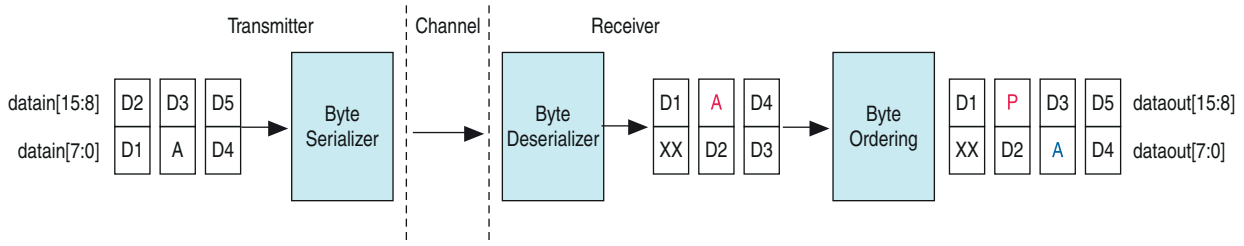
Notes to Table 1–24:

(1) Byte ordering is supported only when you enable the byte deserializer.

(2) The MSB of the 9-bit pattern represents the 1-bit control identifier of the 8B/10B-decoded data. The lower 8 bits represent the 8-bit decoded code.

Figure 1–23 shows an example of a byte ordering operation in single-width mode (8-bit PMA-PCS interface width) where A is the predefined byte ordering pattern and P is the predefined pad pattern.

Figure 1–23. Byte Ordering Operation Example in Single-Width Mode



Byte Ordering in Double-Width Mode

Table 1–25 lists the byte ordering operation in double-width mode.

Table 1–25. Byte Ordering Operation in Double-Width Mode *(1)*

PMA–PCS Interface Width	FPGA Fabric–Transceiver Interface Width	8B/10B Decoder	Byte Ordering Pattern Length	Pad Pattern Length
16 bits	32 bits	Disabled	8 or 16 bits	8 bits
20 bits	32 bits	Enabled	9 *(2)* or 18 bits *(3)*	9 bits *(2)*
	40 bits	Disabled	10 or 20 bits	10 bits

Notes to Table 1–25:

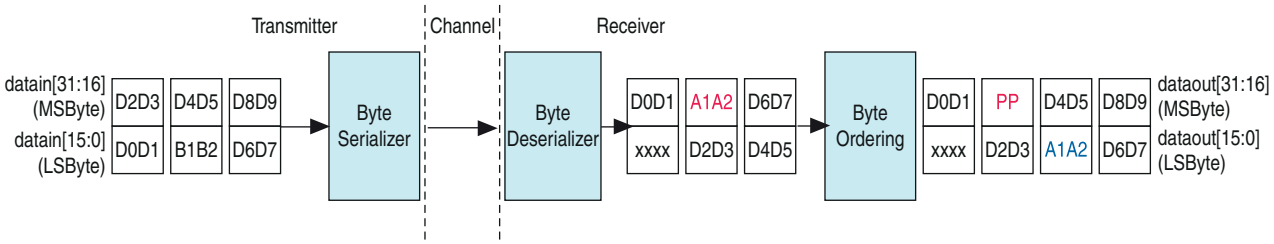
(1) Byte ordering is supported only when you enable the byte deserializer.

(2) The MSB of the 9-bit pattern represents the 1-bit control identifier of the 8B/10B-decoded data. The lower 8 bits represent the 8-bit decoded code.

(3) The 18-bit pattern consists of two sets of 9-bit patterns, individually represented as in Note *(2)*.

Figure 1–24 shows an example of a byte ordering operation in double-width mode (16-bit PMA-PCS interface width) where A1A2 is the predefined byte ordering pattern and P is the predefined pad pattern.

Figure 1–24. Byte Ordering Operation Example in Double-Width Mode



Word Aligner-Based Ordering Mode

In word aligner-based ordering mode, the byte ordering operation is controlled by the word aligner synchronization status signal, `rx_syncstatus`. After a rising edge on the `rx_syncstatus` signal, byte ordering looks for the byte ordering pattern in the byte-deserialized data.

When the first data byte that matches the byte ordering pattern is found, the byte ordering performs the following operations:

- If the pattern is not in the LSByte position—byte ordering inserts the appropriate number of pad patterns to push the byte ordering pattern to the LSByte position and indicates the byte alignment.
- If the pattern is in the LSByte position—byte ordering indicates the byte alignment.

Any byte misalignment found thereafter is ignored unless another rising edge on the `rx_syncstatus` signal, indicating resynchronization, is observed.

Manual Ordering Mode

In manual ordering mode, the byte ordering operation is controlled using the `rx_enabyteord` signal. A rising edge on the `rx_enabyteord` signal triggers byte ordering to look for the byte ordering pattern in the byte-deserialized data.

When the first data byte that matches the byte ordering pattern is found, the byte ordering performs the following operations:

- If the pattern is not in the LSByte position—byte ordering inserts the appropriate number of pad patterns to push the byte ordering pattern to the LSByte position and indicates the byte alignment.
- If the pattern is in the LSByte position—byte ordering indicates the byte alignment.

Any byte misalignment found thereafter is ignored unless another rising edge on the `rx_enabyteord` signal is observed.

Receiver Phase Compensation FIFO

The receiver phase compensation FIFO is four words deep and interfaces the status and data signals between the receiver PCS and the FPGA fabric or the PCIe hard IP block. The FIFO supports the following operations:

- Phase compensation mode with various clocking modes on the read clock and write clock
- Registered mode with only one clock cycle of datapath latency

Phase Compensation Mode

The receiver phase compensation FIFO compensates for any phase difference between the read and write clocks for the receiver status and data signals. The low-speed parallel clock feeds the write clock; the FPGA fabric interface clock feeds the read clock. The clocks must have 0 ppm difference in frequency or a FIFO underrun or overflow condition may result.

The receiver phase compensation FIFO supports various clocking modes on the read and write clocks depending on the transceiver configuration.

For a detailed description of the receiver datapath interface clocking modes when using the receiver phase compensation FIFO, refer to the *Transceiver Clocking in Arria V Devices* chapter.

Registered Mode

To eliminate the FIFO latency uncertainty for applications with stringent datapath latency uncertainty requirements, bypass the FIFO functionality in registered mode to incur only one clock cycle of datapath latency when interfacing the receiver channel to the FPGA fabric. Configure the FIFO to registered mode when interfacing the receiver channel to the PCIe hard IP block to reduce datapath latency. In registered mode, the low-speed parallel clock that is used in the receiver PCS clocks the FIFO.

Channel Bonding

The following factors contribute to the transmitter channel-to-channel skew:

- High-speed serial and low-speed parallel clock skew between channels
- Unequal latency in the transmitter phase compensation FIFO

Bonded transmitter datapath clocking provides low channel-to-channel skew when compared with non-bonded channel configurations.

For more information about the bonded and non-bonded channel clocking, refer to the *Transceiver Clocking in Arria V Devices* chapter.

Bonded Channel Configurations

In bonded channel configurations, the serial and parallel clocks are generated by the transmit PLL and the central clock divider.

There is equal latency in the transmitter phase compensation FIFO of all bonded channels because they share common pointers and control logic generated in the central clock divider.

The lower transceiver clock skew and equal latency in the transmitter phase compensation FIFO in all channels result in a lower channel-to-channel skew.

Bonded channel configurations are available only for 6-Gbps transceivers. You can bond (up to) all channels on the same side.

Non-Bonded Channel Configurations

In a non-bonded channel configuration, the parallel clock in each channel is generated independently by its local clock divider.

There may be unequal latency in the transmitter phase compensation FIFO of each channel because each channel has its own pointers and control logic. The higher transceiver clock skew and unequal latency in the transmitter phase compensation FIFO in each channel may result in a higher channel-to-channel skew.

Non-bonded channel configurations are available for 10-Gbps and 6-Gbps transceivers.

PLL Sharing

In a Quartus® II design, you can merge two different protocol configurations to share the same CMU PLL resources. These configurations must fit in the same transceiver bank. The input `refclk` and PLL output frequencies must be identical.

Document Revision History

Table 1–26 lists the revision history for this chapter.

Table 1–26. Document Revision History

Date	Version	Changes
June 2012	1.2	■ Merged information from Transceiver Basics for Arria V Devices, version 1.1 into this chapter. ■ Updated Figure 1–1. ■ Updated Table 1–8 and Table 1–12. ■ Updated "Architecture Overview", "PMA Architecture", "PCS Architecture", "Channel Bonding", "PLL Sharing" and "Document Revision History" ■ Added Figure 1–6.
November 2011	1.1	■ Updated the maximum data rate to 6.5536 Gbps. ■ Changed the "IP Compiler for PCI Express User Guide" links to the "Arria V Hard IP for PCI Express User Guide" links. ■ Updated the "Transceiver Banks", "10-Gbps Transceiver Channels", "CMU PLL", "Clock Divider", "Calibration Block" sections. ■ Updated Table 1–1, Table 1–2, and Table 1–5. ■ Updated Figure 1–1, Figure 1–2, Figure 1–3, Figure 1–4, Figure 1–5, Figure 1–7, Figure 1–9, Figure 1–13, and Figure 1–17. ■ Removed Table 1-3 and Table 1-4. ■ Minor text edits.
August 2011	1.0	Initial release.

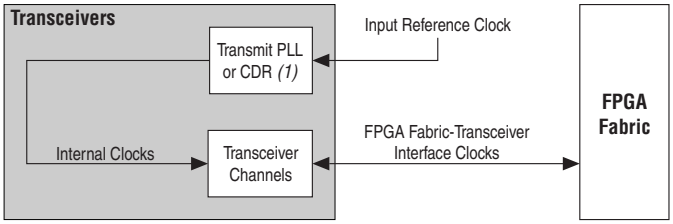
2. Transceiver Clocking in Arria V Devices

AV53002-1.2

This chapter provides information about the Arria® V transceiver clocking architecture. The chapter describes the clocks that are required for operation, internal clocking architecture, and clocking options when the transceiver interfaces with the FPGA fabric.

Figure 2–1 shows an overview of the clocking architecture.

Figure 2–1. Transceiver Clocking Architecture Overview



Note to Figure 2–1:

(1) The transmit phase-locked loop (PLL) can be a CMU PLL (channel PLL) or a fractional PLL.

This chapter contains the following sections:

Input Reference Clocking

This section describes how the reference clock for the channel PLL is provided to generate the clocks required for transceiver operation.

Each 6-Gbps transceiver channel has a channel PLL that you can configure in one of the following schemes:

- Transmitter clock multiplier unit (CMU) PLL—independently synthesizes the input reference clock to generate the high-speed serial clock for clocking the transceivers.
- Receiver clock data recovery (CDR)—independently recovers the clock and data from the incoming serial data.

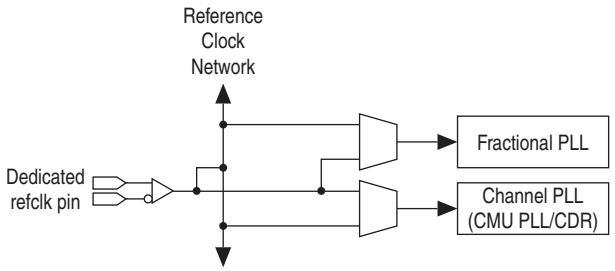
The CDR is not available when the channel PLL is configured as a CMU PLL. Without a CDR, you can use the channel only as a transmitter channel.

10-Gbps transceiver channels always support full-duplex operation. Each 10-Gbps transceiver channel configures its channel PLL as a CDR PLL for receiver operation, and uses a serial clock provided by the CMU PLL of an adjacent channel in the same triplet for transmitter operation. Fractional PLLs are not available for 10-Gbps transceivers.

For more information about configuring the channel PLL as a CMU PLL or CDR, refer to the *Transceiver Architecture in Arria V Devices* chapter.

The transceiver channel PLL and fractional PLL derive the input clock from a dedicated `refclk` pin or from the reference clock network. Figure 2–2 shows an overview of the input to the transceiver channel from the dedicated input reference clock.

Figure 2–2. Input Reference Clock Sources to Transceiver Channel



Dedicated Reference Clock Pins

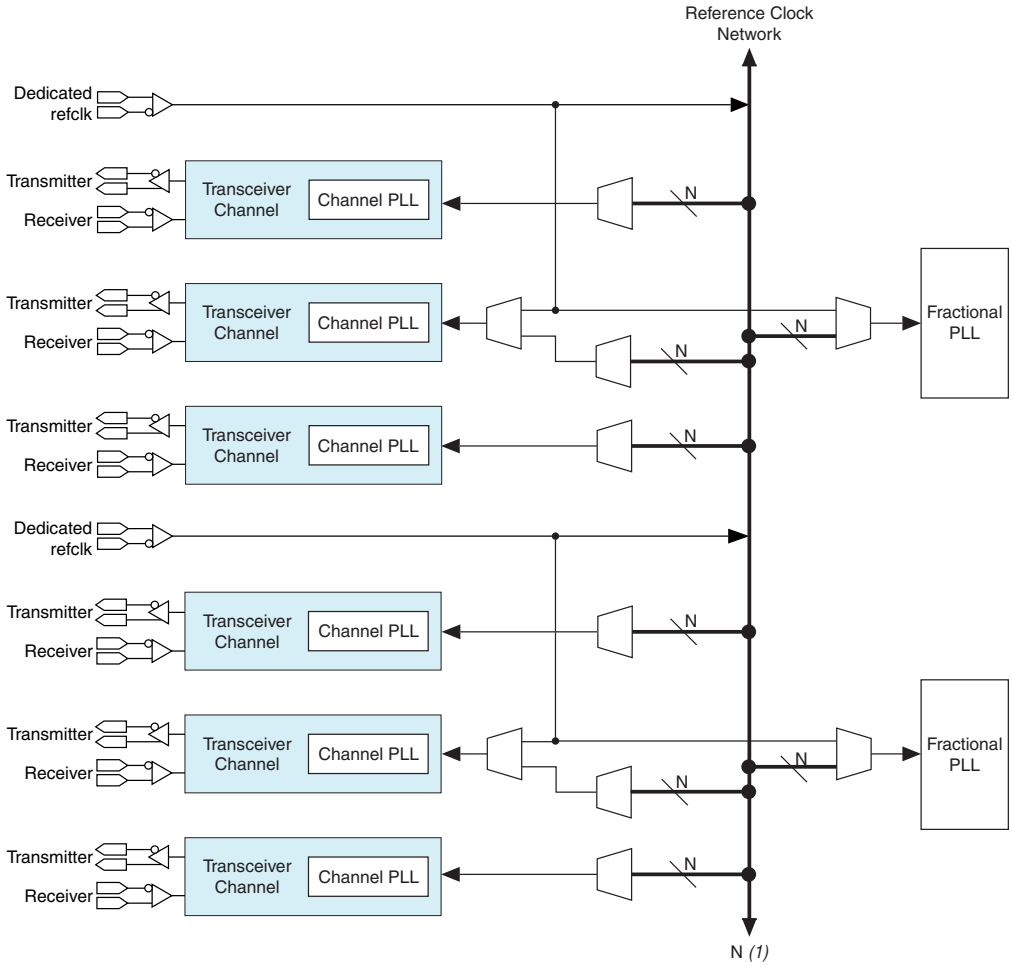
Arria V devices have one dedicated `refclk` pin for each triplet channel. Every dedicated reference clock pin drives a reference clock network that spans the side of the device. When the channels are used as 10-Gbps channel operation, the dedicated `refclk` pin only drives the 10-Gbps channels in the same triplet. For 6-Gbps transceiver channels operation, the `refclk` pin can be sourced from other triplet through the reference clock network. However, for 10-Gbps transceiver channel operation, the `refclk` pin is sourced from the same triplet.

For specifications about the input frequency supported by the `refclk` pins, refer to the *Arria V Device Datasheet*.

Dedicated Reference Clock Using the Reference Clock Network

A single dedicated `refclk` pin can provide the reference clock to multiple channel PLLs or fractional PLLs that require the same clock frequency through the reference clock network. Figure 2–3 shows the input reference clock sources for a 6-Gbps transceiver bank and two fractional PLLs.

Figure 2–3. Input Reference Clock Sources for 6-Gbps Transceiver Channels in a Transceiver Bank

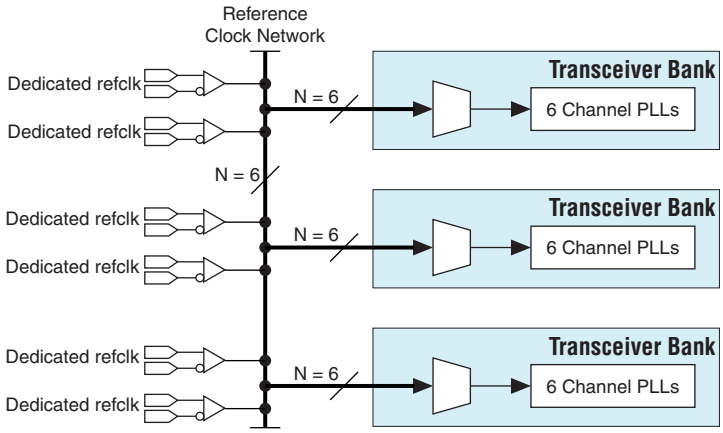


Note to Figure 2–3:

(1) `N` is the number of dedicated `refclk` pins, which is equal to the number of transceiver channels on a side divided by three.

Figure 2–4 shows the input reference clock sources for eighteen 6-Gbps channel PLLs on the left side of a 5AGXB5KF40 device. For the eighteen 6-Gbps channels, the total number of clock lines is six (N = 18/3). There are similar input reference clock resources on the right side of the device.

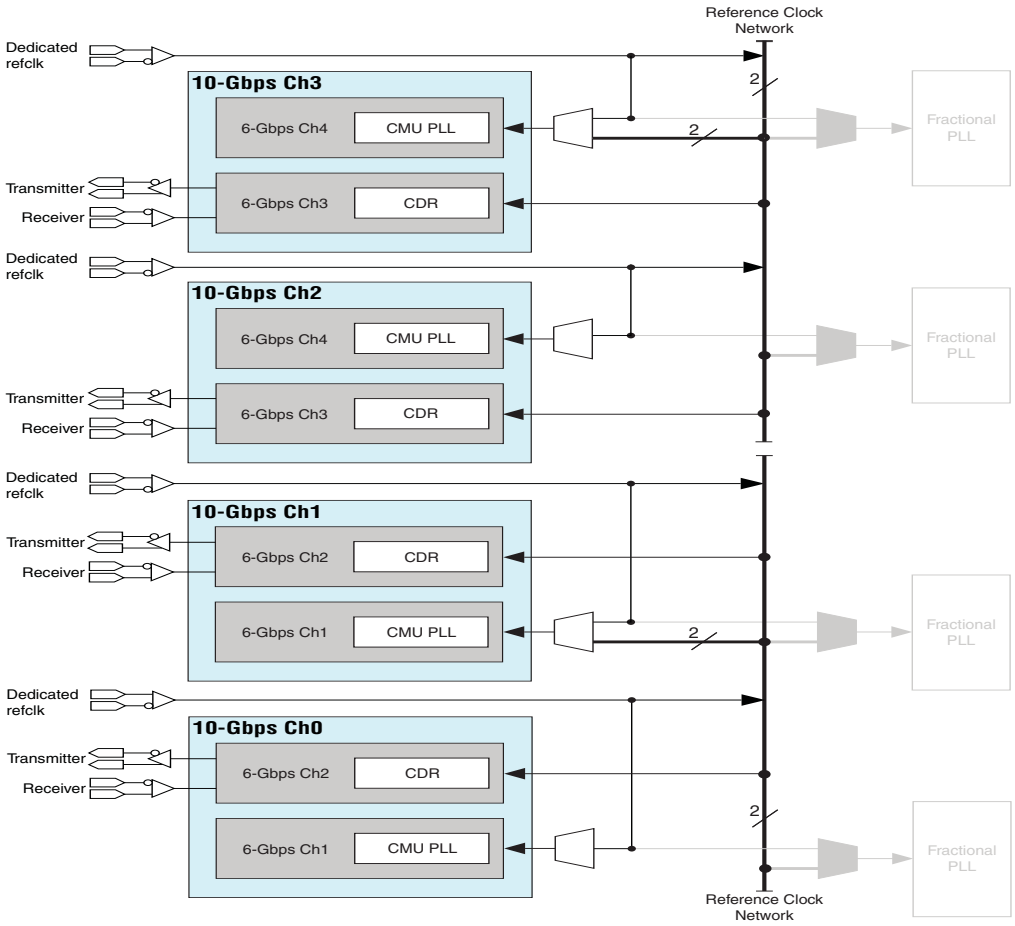
Figure 2–4. Input Reference Clock Sources in a 5AGXB5KF40 Device



Arria V GT devices use the same dedicated `refclk` input pins as the 6-Gbps transceiver channels. For 10-Gbps channel operation, each dedicated `refclk` input pin drives only the 10-Gbps channels PLLs located in the same triplet.

Figure 2–5 shows the input reference clock sources for four full-duplex 10-Gbps channels in two triplets in a quad. The 10-Gbps channel 0 is 6-Gbps channel 0, 10G-channel 1 is 6-Gbps channel 2, 10-Gbps channel 2 is 6-Gbps channel 3, and 10-Gbps channel 3 is 6-Gbps channel 5. The 10-Gbps channels configure their channel PLLs as CDRs for receiver functionality. For transmitter functionality, 10-Gbps channels 0 and 1 use the CMU PLL from 6-Gbps channel 1, while 10-Gbps channels 2 and 3 use the CMU PLL from 6-Gbps channel 4.

Figure 2–5. Input Reference Clock Sources for Four 10-Gbps Transceiver Channels in a 10 Gbps-capable Transceiver Bank *(1)*

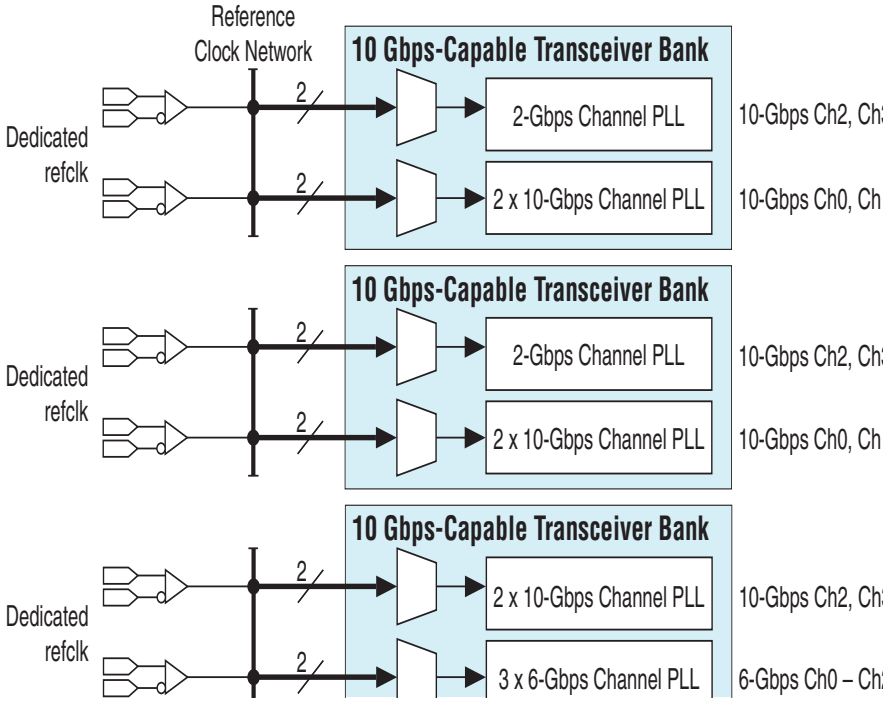


Note to Figure 2–5:

(1) You can alternatively configure the two channels as 10-Gbps or 6-Gbps channels for every triplet. You cannot mix 10-Gbps channels and 6-Gbps channels within a triplet. However, you may configure one triplet as a 10-Gbs channel and the other triplet as a 6-Gbs channel within a bank.

Figure 2–6 shows the dedicated reference clock inputs for each side (left and right) of a 5AGTD7KF40 device. For the top two banks, each reference clock input drives a 10-Gbps channel PLL. For the bottom bank, one reference clock input drives the 10-Gbps channel PLL in the upper triplet. The other reference clock input drives the three 6-Gbps channel PLL in the lower triplet.

Figure 2–6. Input Reference Clock Sources for Left or Right Side of the 5AGTD7KF40 Device



Notes to Figure 2–6:

(1) You can alternatively configure each triplet to support either 10-Gbps channels or 6-Gbps channels.
(2) The channels in this triplet can only be configured as 6-Gbs channels.

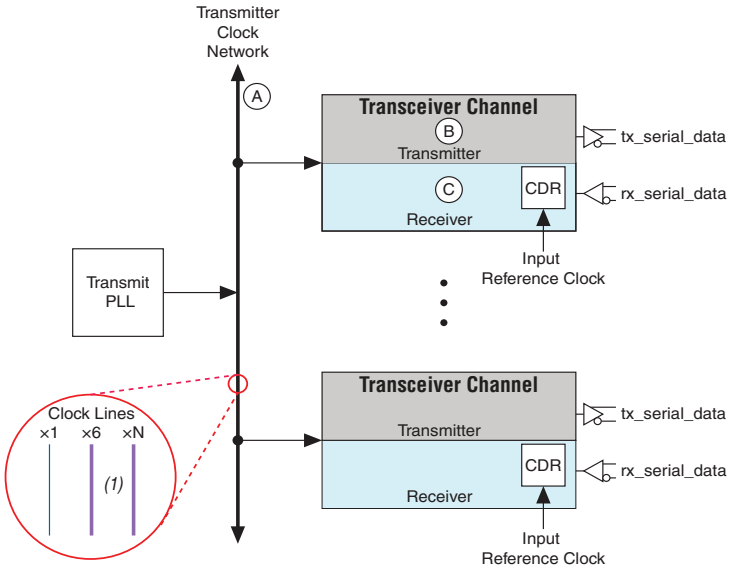
Internal Clocking

This section describes the clocking architecture internal to Arria V transceivers. Different physical coding sublayer (PCS) configurations and channel bonding options result in various transceiver clock paths.

The following labels, as shown in Figure 2–7, describe the sections of the transceiver internal clocking:

- **A**—Transmitter Clock Network
- **B**—Transmitter Clocking
- **C**—Receiver Clocking

Figure 2–7. Internal Clocking



Note to Figure 2–7:

(1) The x6 and xN clock lines are supported only by the Arria V 6-Gbps transceivers.

For the transmitter channel, the CMU PLL or fractional PLL (fractional PLLs are available only in 6-Gbps transceiver channels) use the input reference clock. The CMU PLL provides a serial clock to the transmitter clock network, which is distributed to the transceiver channels.

This section describes clocking that is internal to the transceiver. The Quartus® II software primarily performs the clock routing based on the transceiver configuration that you select.

Transmitter Clock Network

The transmitter clock network routes the clock from the transmit PLL to the transmitter channel, as shown in Figure 2–7. A clock divider provides two clocks to the transmitter channel:

- Serial clock—high-speed clock for the serializer
- Parallel clock—low-speed clock for the serializer and the PCS

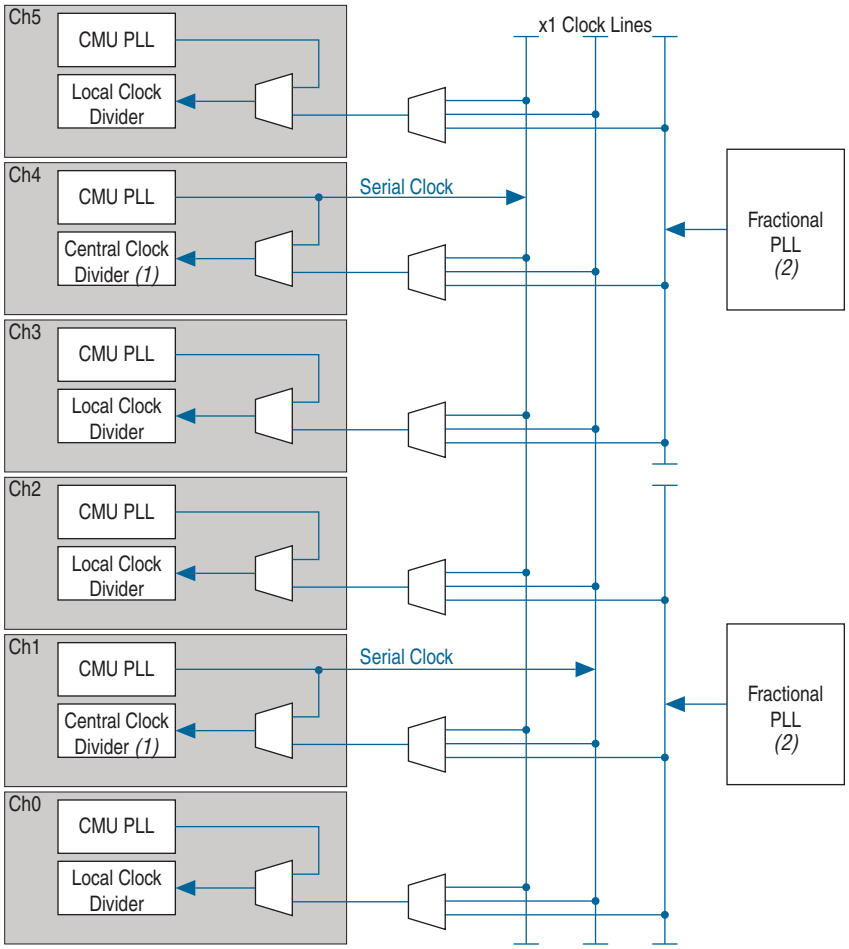
Arria V transceivers support non-bonded and bonded transceiver clocking configurations:

- Non-bonded configuration—Only the serial clock from the transmit PLL is routed to the transmitter channel. The clock divider of each channel generates the local parallel clock. The x1 clock line is used for non-bonded configurations. This configuration is available for both the 6-Gbps and 10-Gbps transceivers.
- Bonded configuration—Both the serial clock and parallel clock are routed from the central clock divider in channel 1 or 4 to the bonded transmitter channels. The x6 and xN clock lines are used for bonded configurations. This configuration is only available for 6-Gbps transceivers.

The Quartus II software performs the clock routing related to the transmitter clock network based on the transceiver configuration that you select.

Figure 2–8 shows the x1 clock lines used by a 6-Gbps transceiver bank.

Figure 2–8. x1 Clock Lines Used for Non-Bonded Configuration in a 6-Gbps Transceiver Bank



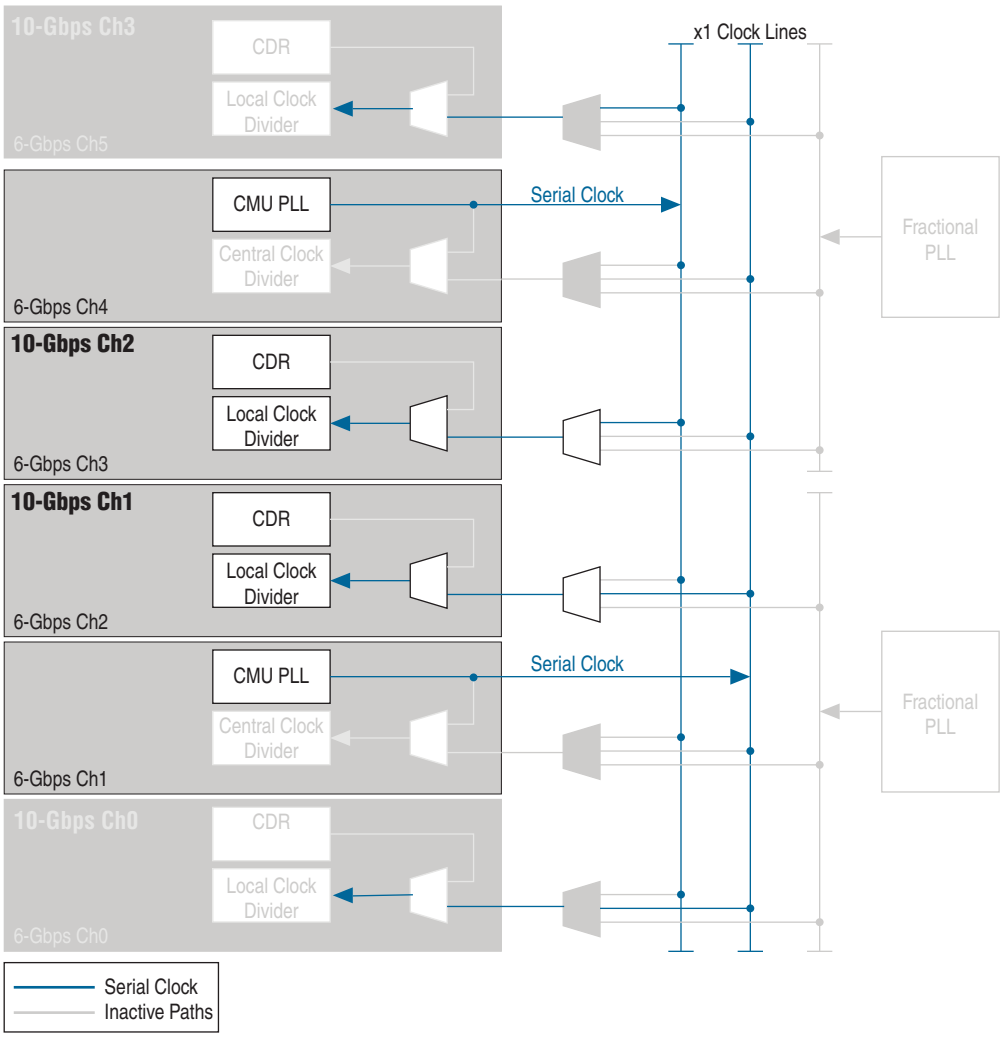
Notes to Figure 2–8:

(1) You can use the central clock divider as a local clock divider.

(2) One fractional PLL provides a clock to the x1 clock line of only three channels.

Figure 2–9 shows the x1 clock lines for a 10-Gbps transceiver bank.

Figure 2–9. 10-Gbps Transceiver Bank Operation and x1 Clock Lines Used for a Non-Bonded Configuration *(1)*



Note to Figure 2–9:

(1) This figure is applicable to the Arria V GT devices.

The x1 clock lines are driven by the following resources in a transceiver bank:

- 6-Gbps transceivers—the channel PLLs of channels 1 and 4 that are configured as CMU PLLs, or the fractional PLLs
- 10-Gbps transceivers—the channel PLLs of the 6-Gbps channels 1 and 4 that are configured as CMU PLLs

The x1 clock lines can drive the clock divider of each channel within a transceiver bank. For the 10-Gbps transceiver operation, the x1 clock lines drive only the local dividers of the 10-Gbps channels within the same triplet.

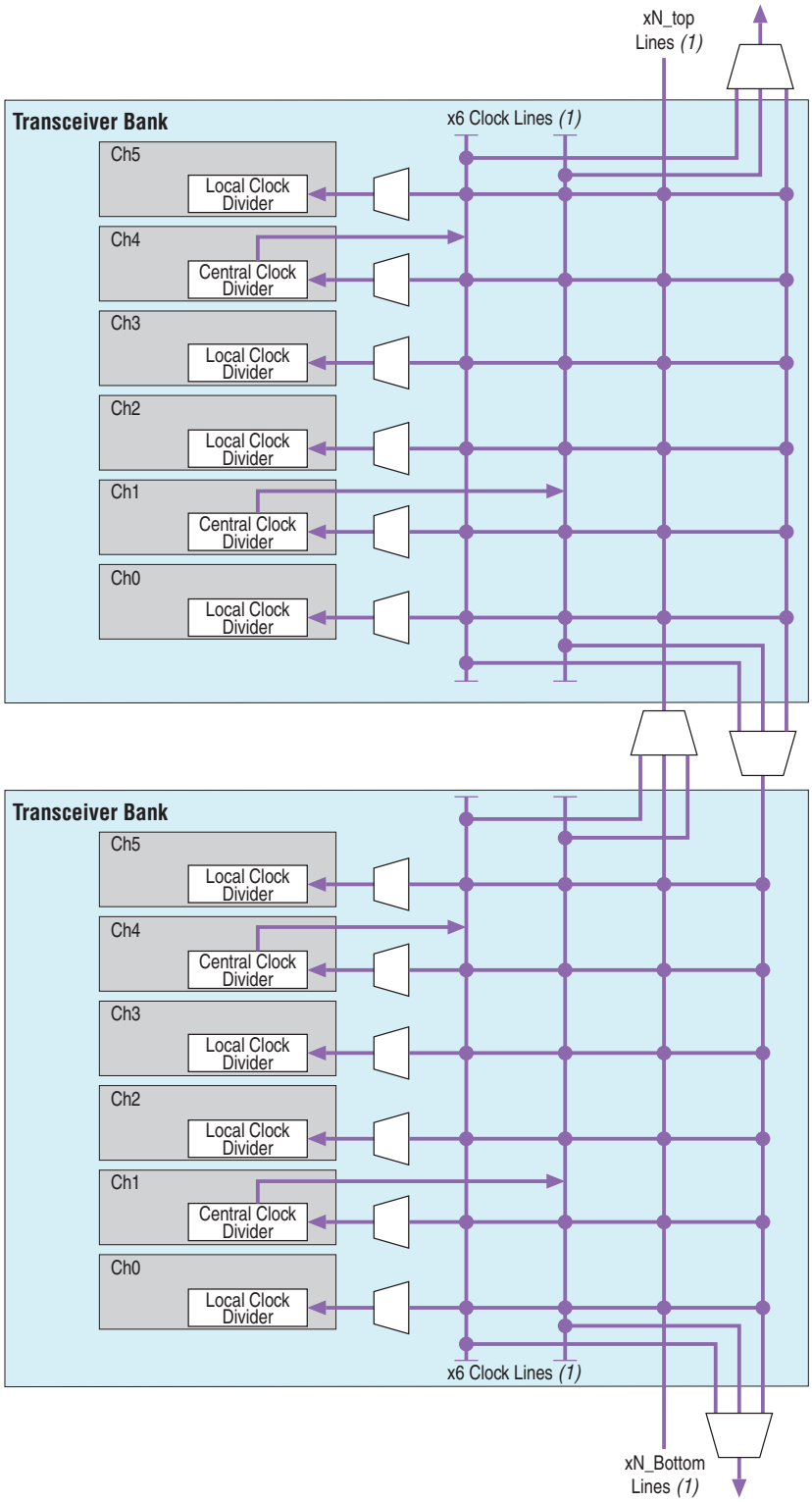
Bonded configurations use the x6 and xN clock lines. These clock lines route the serial and parallel clock from the central clock dividers of channel 1 or 4 when using 6-Gbps transceiver channels.

To conserve the number of transmit PLLs in your design, use the xN clock lines driven by the x6 clock lines, to route the serial clock from the central clock dividers in a transceiver bank to the transceiver channels in another transceiver bank for non-bonded configurations.

Figure 2–10 shows the x6 and xN clock lines for 6-Gbps transceiver banks. You can apply the following conditions to the clock lines:

- Drive the x6 clock lines with the central clock divider of channels 1 and 4 in a transceiver bank.
- Use the x6 clock lines to drive the xN clock lines.
- Use the x6 clock lines to drive any channel within a transceiver bank.
- Use the xN clock lines to drive any channel on the respective side, because the xN clock lines span the entire side of the device.

Figure 2–10. x6 and xN Clock Lines in 6-Gbps Transceivers



Note to Figure 2–10:

(1) The clock lines carry both the serial and parallel clocks.

Table 2–1 lists the span and data rates supported by the clock sources and networks in Arria V devices.

Table 2–1. Data Rates and Spans Supported Using Arria V Clock Sources and Networks

Transceiver	Clock Network	Clock Source	Maximum Data Rate (1)	Bonding	Span
6-Gbps	x1	Ch1 or Ch4 CMU PLL in a transceiver bank	6.5536 Gbps	No	Within transceiver bank
		Fractional PLLs in a transceiver bank	3.125 Gbps		
	x6	Central clock dividers in a transceiver bank (in Ch1 or Ch4 only)	6.5536 Gbps	Yes	Within transceiver bank
	xN	Central clock dividers in a transceiver bank through x6 clock lines (in Ch1 or Ch4 only)	3.125 Gbp	Yes	Side wide
10-Gbps	x1	Inactive 6-Gbps Ch1 and Ch4 used as a dedicated CMU PLL in the transceiver bank	10.3125 Gbps	No	Specific to 10-Gbps Channel

Note to Table 2–1:

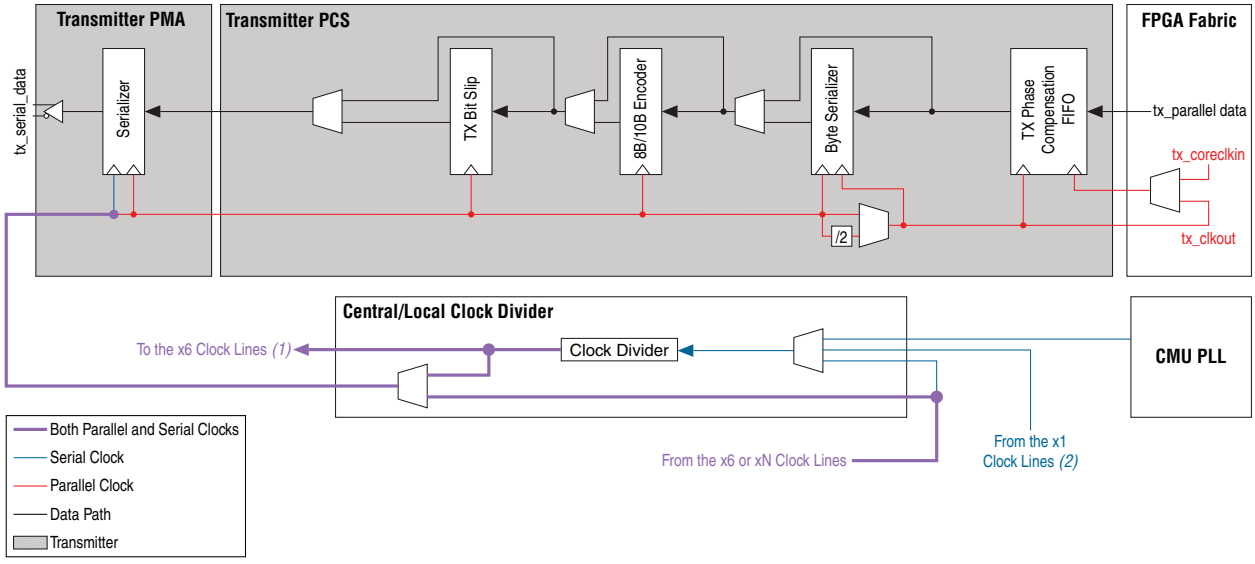
(1) Applies only to the fastest speed grade. For other speed grades, refer to the *Arria V Device Datasheet*.

Transmitter Clocking

Transmitter clocking refers to the clocking architecture that is internal to the transmitter channel of a transceiver.

Figure 2–11 shows the transmitter PCS and physical medium attachment (PMA) clocking for 6-Gbps transceivers.

Figure 2–11. Transmitter PCS Clocking for 6-Gbps Transceiver



Notes to Figure 2–11:

(1) Only available in the central clock dividers of channel 1 and channel 4 in a transceiver bank.

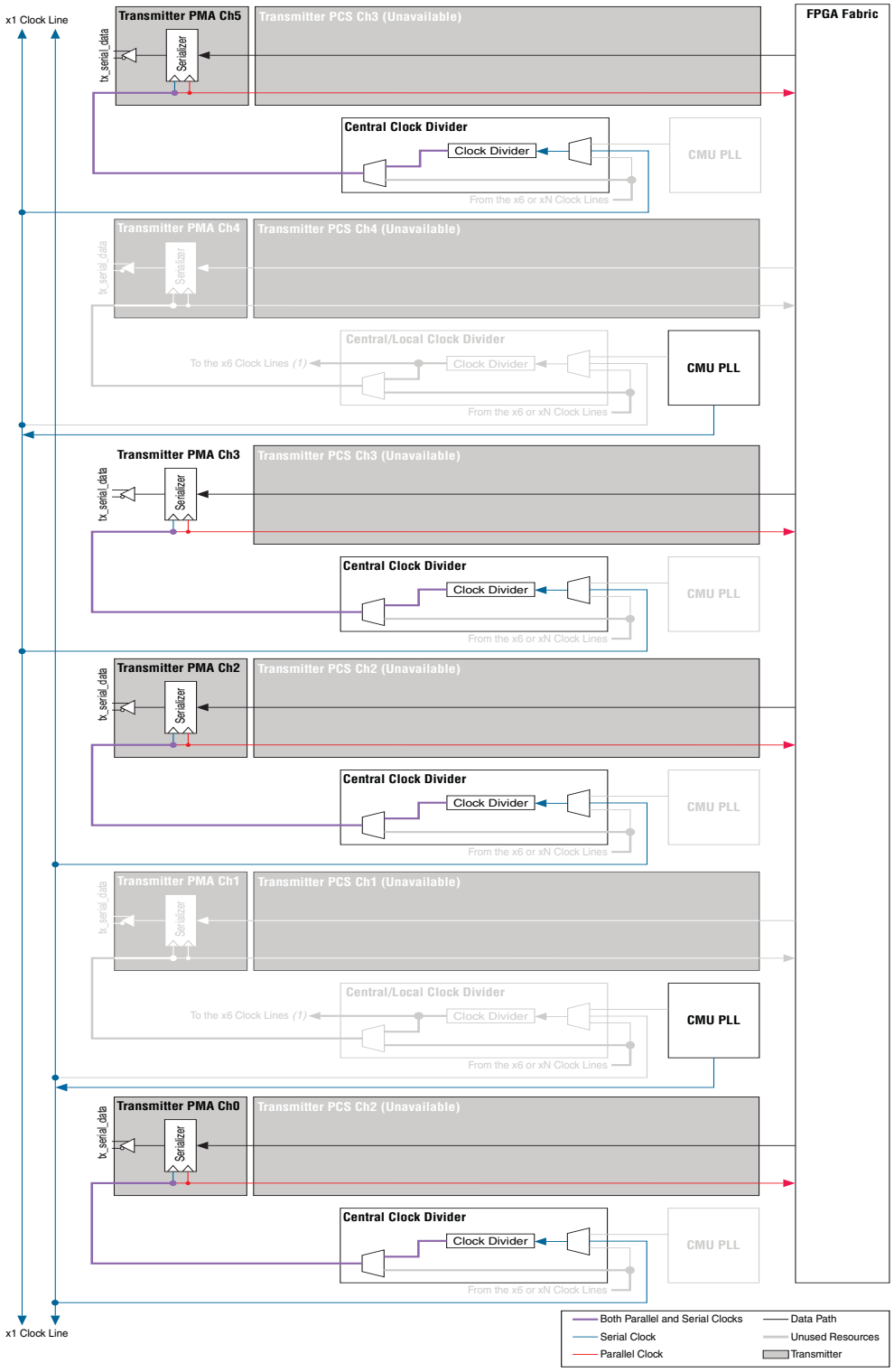
(2) You can drive x1 clock lines of the 6-Gbps transceiver with a CMU PLL. A fractional PLL only supports a 3-Gbps transceiver.

As shown in Figure 2–11, the clock divider block provides the serial and parallel clock to the serializer of the transmitter PMA, and the parallel clock to the transmitter PCS. The parallel clock clocks all the blocks up to the read side of the transmitter (TX) phase compensation FIFO any time the byte serializer is not used.

For configurations with the byte serializer block, the clock is divided by a factor of two for the byte serializer and the read side of the TX phase compensation FIFO. The read side clock of the TX phase compensation FIFO is also forwarded to the FPGA fabric to interface the FPGA fabric with the transceiver.

Figure 2–12 shows transmitter PMA clocking for 10-Gbps transceivers.

Figure 2–12. Transmitter PMA Direct Clocking for 10-Gbps Transceiver



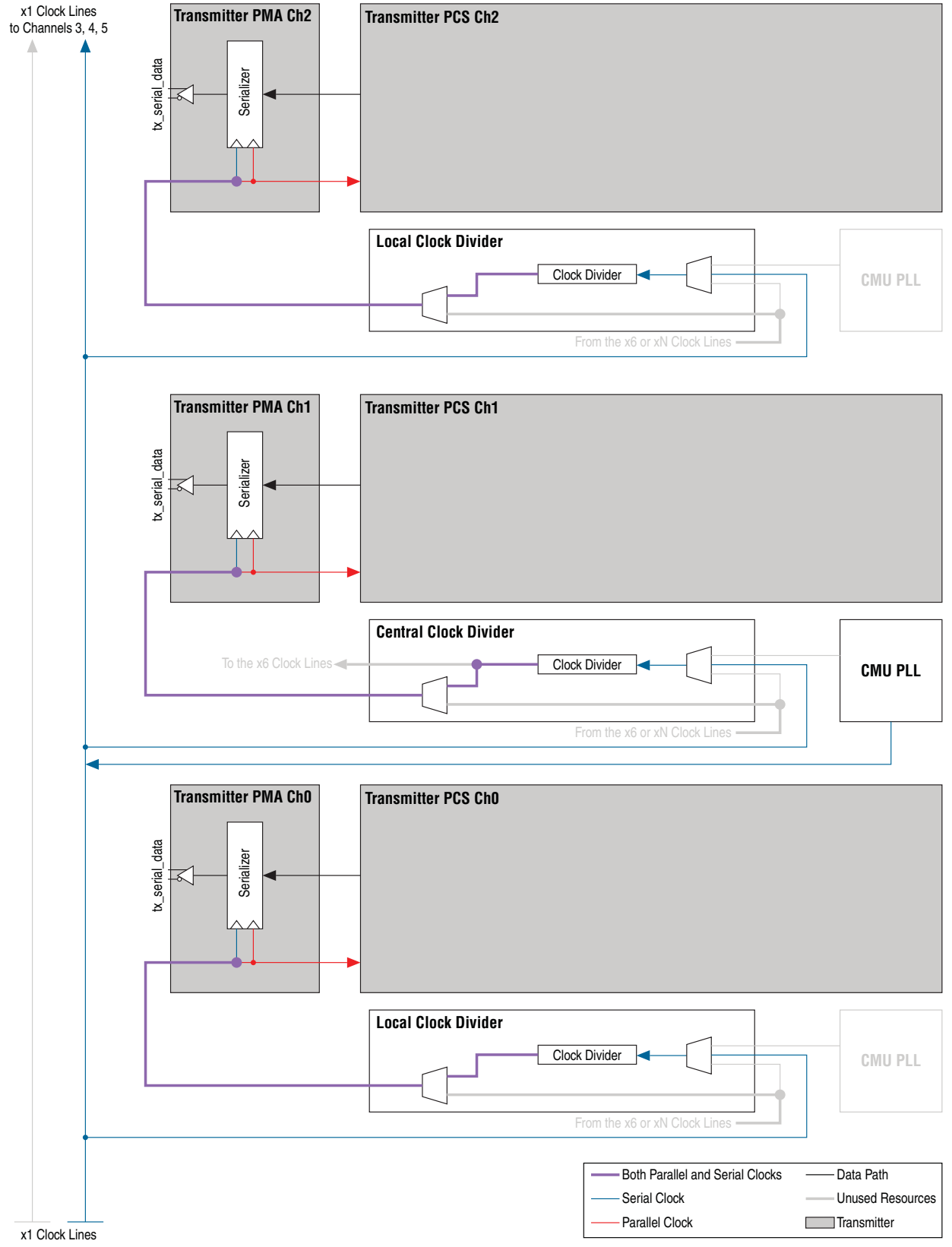
As shown in Figure 2–12, the clock divider block provides the serial and parallel clock to the serializer of the transmitter PMA. The parallel clock is driven directly to the FPGA fabric because the PCS is unavailable in 10-Gbps transceivers. All PCS functions, such as encoding and bit slipping, must be implemented in the core IP.

For more about clocking schemes used in different configurations, refer to the *Transceiver Protocol Configurations in Arria V Devices* and *Transceiver Custom Configurations in Arria V Devices* chapters.

Non-Bonded Channel Configurations

In non-bonded configurations for 6-Gbps transceivers, the clock divider of individual channels generates the parallel clock. Figure 2–13 shows three transmitter channels in a non-bonded configuration driven by the CMU PLL of channel 1 and driving the x1 clock line. The clock divider block of each channel generates its own parallel clock by dividing the serial clock from the x1 clock line.

Figure 2–13. Three 6-Gbps Transmitter Channels in Non-Bonded Configuration



In non-bonded configurations for 10-Gbps transceivers, the local clock divider of the individual channel generates the parallel clock for the transmitter. In Figure 2–12 on page 2–14, four 10-Gbps transmit channels function independently and are driven by their respective CMU PLL from the inactive 6-Gbps channels in the same triplet. The clock divider of each 10-Gbps channel generates its own parallel clock by dividing the serial clock coming from the CMU PLL of the same triplet.

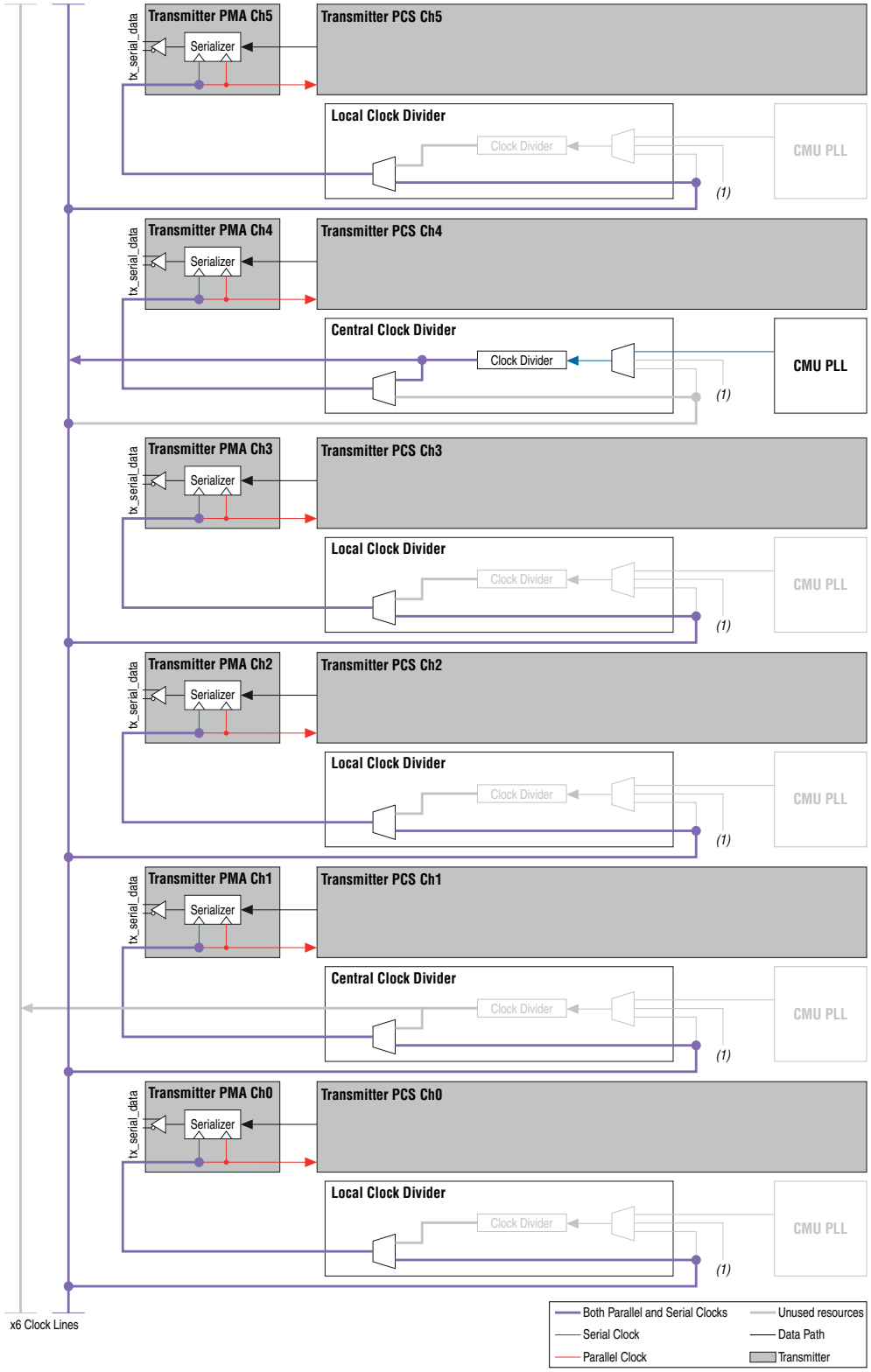
Bonded Channel Configurations

In bonded configurations, both the parallel clock and serial clock are sourced from the x6 or xN clock line. The central clock dividers use the serial clock from a transmit PLL from the same transceiver bank using the x1 clock line. The central clock divider generates the parallel clock and drives both the serial clock and parallel clock on the x6 clock line. By default, the Quartus II software configures channel 4 (for example) as a CMU PLL, and uses its central clock divider to generate clocks on the x6 clock line, instead of going through x1 clock line.

The bonded configuration is only applicable to the 6-Gbps transceiver, and channels should be contiguous

Figure 2–14 shows six transmit-only channels configured in a bonded configuration driven by the channel PLL of channel 4, which is configured as a CMU PLL. The central clock divider of channel 4 generates a parallel clock and drives both the serial clock and parallel clock on the x6 clock line. All bonded channels get both serial and parallel clocks from the x6 clock line.

Figure 2–14. Six Transmitter Channels Configured in Bonded Configuration



Note to Figure 2–14:

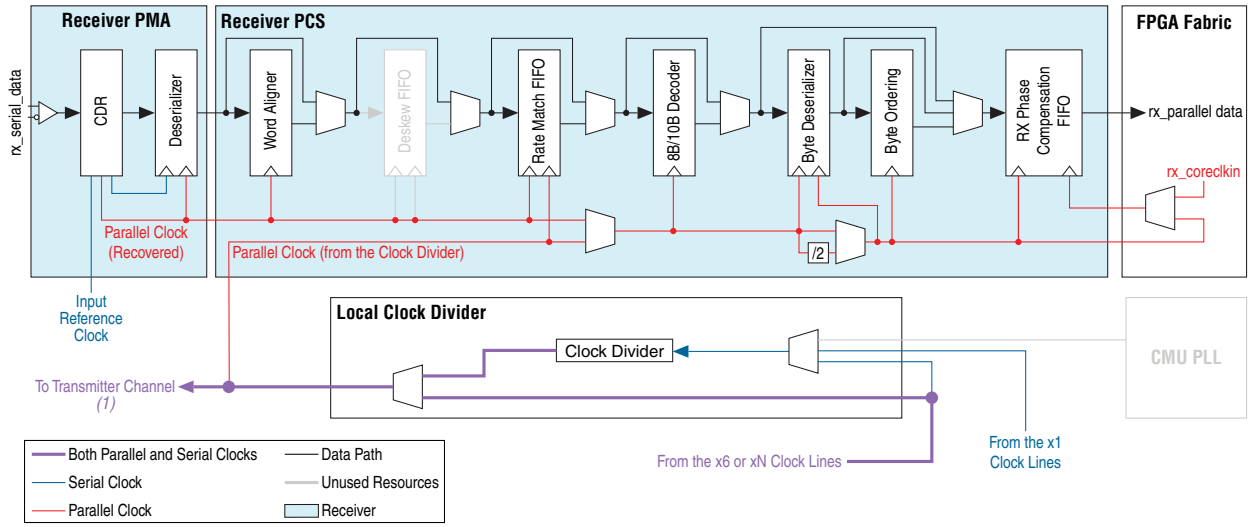
(1) Serial clock from the x1 clock lines. The x1 clock lines are inactive because all 6 transceiver channels in the bank are bonded.

For an example of using the xN clock lines, refer to the PCIe® x8 configuration in the *Transceiver Protocol Configurations in Arria V Devices* and *Transceiver Custom Configurations in Arria V Devices* chapters.

Receiver Clocking

Receiver clocking refers to the clocking architecture internal to the receiver channel of a transceiver. Figure 2–15 shows 6-Gbps receiver PCS and PMA clocking.

Figure 2–15. Receiver PCS Clocking for 6-Gbps Transceivers



Note to Figure 2–15:

(1) Only available in the central clock dividers of channel 1 and channel 4 in a transceiver bank.

The CDR in the PMA of each channel recovers the serial clock from the incoming data and generates the parallel clock (recovered) by dividing the serial clock (recovered). The deserializer uses both clocks. The receiver PCS can use the following clocks depending on the configuration of the receiver channel:

- Parallel clock (recovered) from the CDR in the PMA
- Parallel clock from the clock divider that is used by the channel's transmitter PCS

Table 2–2 lists the different clock sources available for each block in the receiver PCS.

Table 2–2. Clock Sources for All Receiver PCS Blocks (Part 1 of 2)

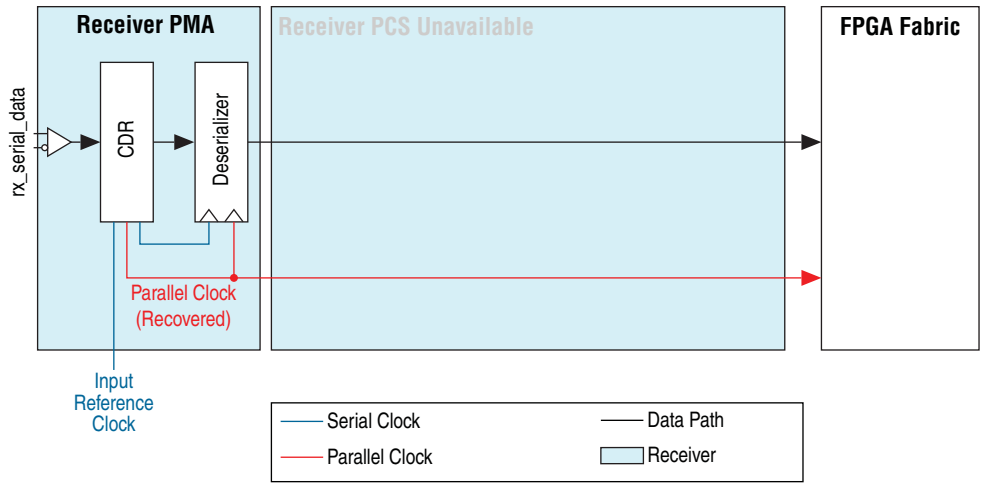
Block	Side	Clock Source
Word aligner	—	Parallel clock (recovered)
Rate match FIFO	Write	Parallel clock (recovered)
	Read	Parallel clock from the clock divider
8B/10B decoder	—	■ Rate match FIFO is not used—Parallel clock (recovered) ■ Rate match FIFO is used—Parallel clock from the clock divider

Table 2–2. Clock Sources for All Receiver PCS Blocks (Part 2 of 2)

Block	Side	Clock Source
Byte deserializer	Write	■ Rate match FIFO is not used—Parallel clock (recovered) ■ Rate match FIFO is used—Parallel clock from the clock divider
	Read	Divided down version of the write side clock depending on the deserialization factor of 1 or 2, also called the parallel clock (divided)
Byte ordering	—	Parallel clock (divided)
Receiver (RX) phase compensation FIFO	Write	Parallel clock (divided). This clock is also forwarded to the FPGA fabric.
	Read	Clock sourced from the FPGA fabric

Figure 2–16 shows clocking for the receiver PMA in a 10-Gbps transceiver.

Figure 2–16. Receiver PMA Direct Clocking for a 10-Gbps Transceiver



As shown in Figure 2–16, the parallel recovered clock from the CDR and deserializer is directly connected to the FPGA fabric because the PCS is unavailable in 10-Gbps transceivers. All PCS functions, such as word alignment, rate matching, decoding, and byte ordering, must be implemented in the core IP.

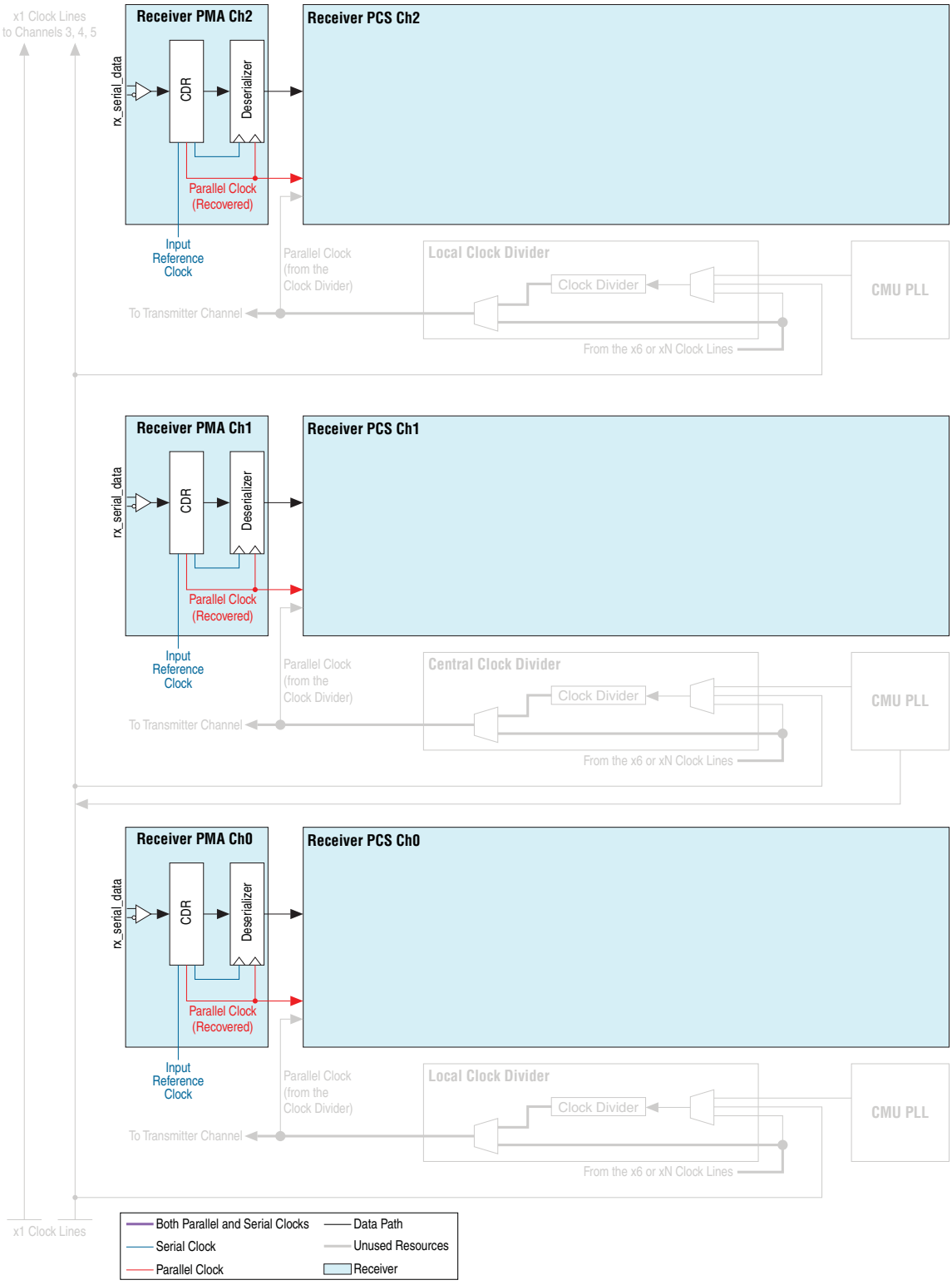
Non-Bonded Channel Configurations

In non-bonded configurations, the 6-Gbps receiver PCS requires the parallel clock (recovered) and depending on the configuration you use, may also require the parallel clock from the clock divider used by the transmitter.

For 10-Gbps transceivers, the FPGA fabric requires the parallel clock (recovered) and depending on the configuration you use, may also require the parallel clock from the clock divider used by the transmitter.

Figure 2–17 shows three 6-Gbps receiver channels in a non-bonded configuration when the rate match FIFO is not used.

Figure 2–17. Three 6-Gbps Transceiver Channels in a Non-Bonded Configuration



Bonded Channel Configurations

Bonded channel configurations are only supported in 6-Gbps transceiver channels. In bonded configurations, the receiver PCS requires the parallel clock (recovered) and, depending on your configuration, may require the parallel clock from the central clock divider in channel 1 or 4.

Figure 2–18 shows five bonded channels in a transceiver bank. Because the channel PLL in channel 4 is configured as a CMU PLL, you cannot use the receiver CDR. Therefore, you can only use channel 4 as a transmitter.

In Figure 2–19, all six channels in the transceiver bank are in a bonded configuration, as opposed to a maximum of five in Figure 2–18. This configuration is possible because the fractional PLL is used as a transmit PLL instead of a channel PLL in the transceiver bank. Using the fractional PLL enables you to configure the channel PLLs of both channels 1 and 4 as CDRs to perform receiver operations.

Fractional PLLs can only support 3-Gbps transceiver channels.

For more information about the clocking scheme used in different configurations, refer to the *Transceiver Protocol Configurations in Arria V Devices* and *Transceiver Custom Configurations in Arria V Devices* chapters.

Figure 2–18. Five Channels Configured in Bonded Configuration

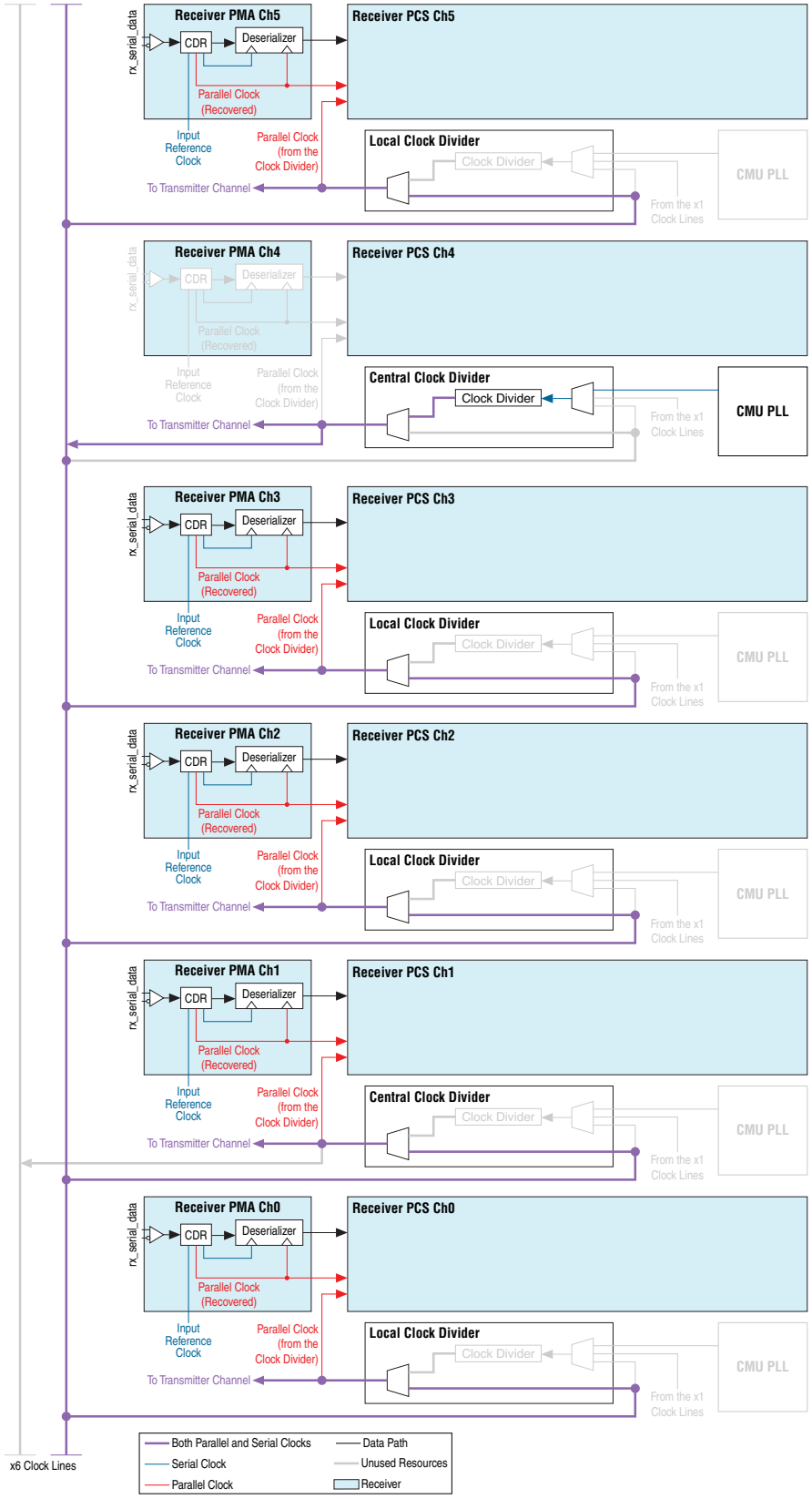
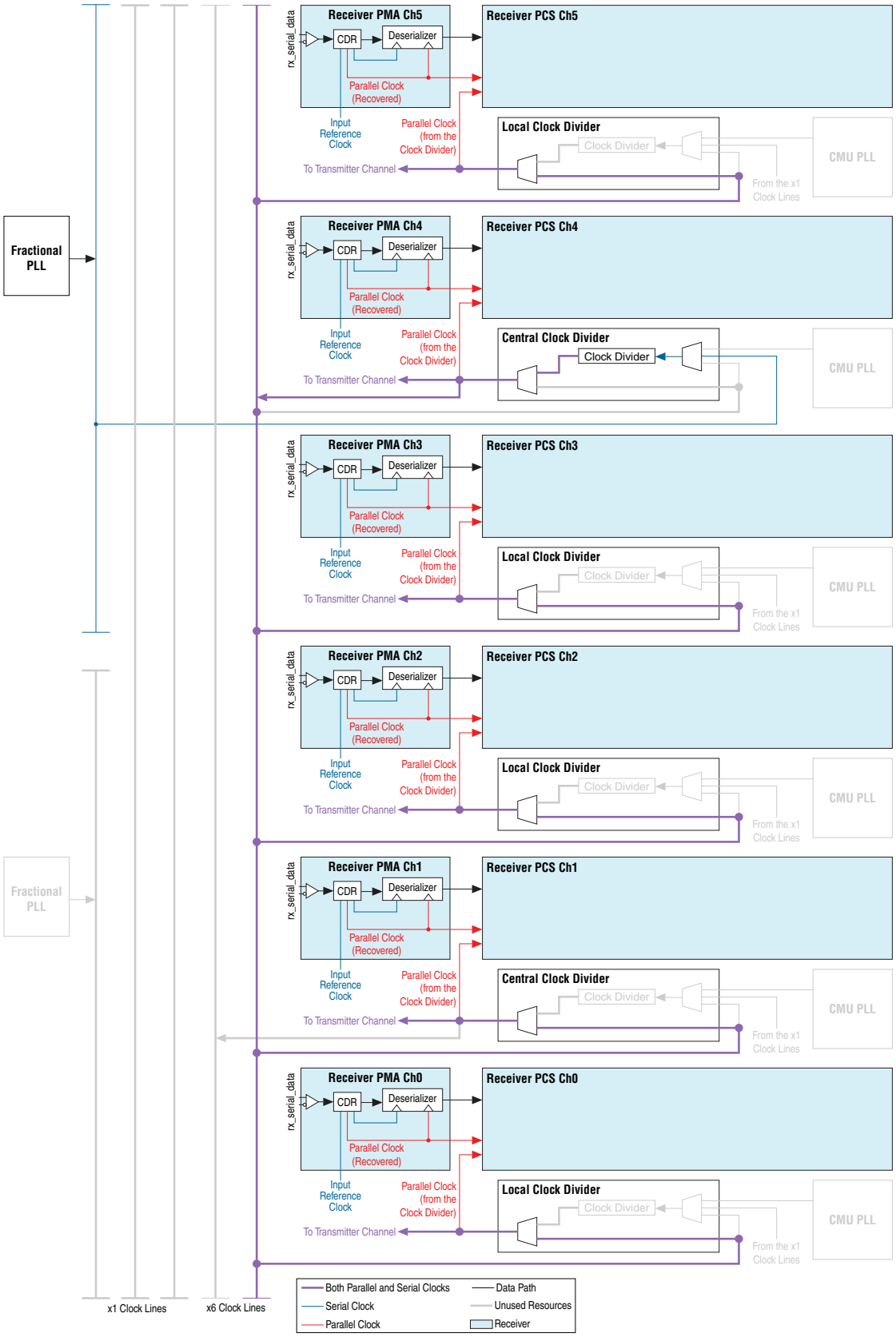


Figure 2–19. Six Channels Configured in Bonded Configuration Using Fractional PLL



FPGA Fabric–Transceiver Interface Clocking

The FPGA fabric–transceiver interface clocks can be subdivided into the following three categories:

- Input reference clocks—Can be an FPGA fabric–transceiver interface clock. This may occur when the FPGA fabric-transceiver interface clock is forwarded to the FPGA fabric, where it can then clock logic.

  The input reference clock can only be routed into the FPGA fabric if a transceiver is also instantiated.

- Transceiver datapath interface clocks—Used to transfer data, control, and status signals between the FPGA fabric and the transceiver channels. The transceiver channel forwards the `tx_clkout` signal to the FPGA fabric to clock the data and control signals into the transmitter. The transceiver channel also forwards the recovered `rx_clkout` clock (in configurations without the rate matcher) or the `tx_clkout` clock (in configurations with the rate matcher) to the FPGA fabric to clock the data and status signals from the receiver into the FPGA fabric.
- Other transceiver clocks—The following transceiver clocks form a part of the FPGA fabric–transceiver interface clocks:
  - `mgmt_clk`—Avalon®-MM interface clock used for controlling the transceivers, dynamic reconfiguration, and calibration
  - `fixed_clk`—the 125 MHz fixed-rate clock used in the PCIe (PIPE) receiver detect circuitry

Table 2–3 shows the FPGA fabric–transceiver interface clocks.

Table 2–3. FPGA Fabric–Transceiver Interface Clocks *(1)*

Clock Name	Clock Description	Interface Direction	FPGA Fabric Clock Resource Utilization
`pll_ref_clk`	Input reference clock used for clocking logic in the FPGA fabric	Transceiver-to-FPGA fabric	GCLK, RCLK, PCLK
`tx_clkout`	Clock forwarded by the transceiver for clocking the transceiver datapath interface		
`rx_clkout`	Clock forwarded by the receiver for clocking the receiver datapath interface		
`tx_coreclkin`	User-selected clock for clocking the transmitter datapath interface	FPGA fabric-to-transceiver	
`rx_coreclkin`	User-selected clock for clocking the receiver datapath interface		
`fixed_clk`	PCIe receiver detect clock		
`mgmt_clk` *(2)*	Avalon-MM interface management clock		

Notes to Table 2–3:

(1) For more information about the GCLK, RCLK, and PCLK resources available in each device, refer to the *Clock Networks and PLLs in Arria V Devices* chapter.

(2) The `mgmt_clk` is a free-running clock that is not derived from the transceiver blocks.

Table 2–4 lists the port names for tx_clkout and rx_clkout.

Table 2–4. Configuration Specific Port Names for tx_clkout and rx_clkout

Configuration	Port Name for tx_clkout	Port Name for rx_clkout
Custom	tx_clkout	rx_clkout
Interlaken		
Low Latency		
PCIe	pipe_pclk	pipe_pclk
XAUI	xgmii_tx_clk	xgmii_rx_clk

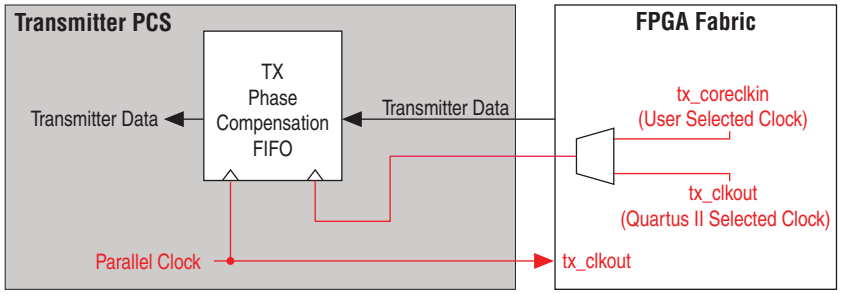
Transmitter Datapath Interface Clocking

For the 6-Gbps transceivers, the write side of the transmitter phase compensation FIFO makes up the transmitter datapath interface. This interface is clocked with the transmitter datapath interface clock.

Figure 2–20 shows the 6-Gbps transmitter datapath interface clocking. The transmitter PCS forwards the following clocks to the FPGA fabric:

- tx_clkout—for each transmitter channel in a non-bonded configuration
- tx_clkout[0]—for all transmitter channels in a bonded configuration

Figure 2–20. Transmitter Datapath Interface Clocking for 6-Gbps Transceivers

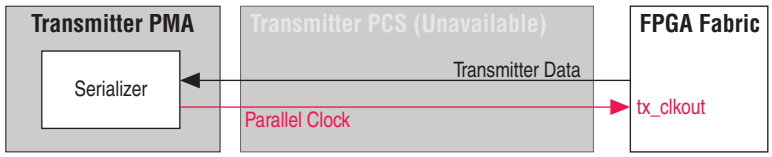


All configurations that use the PCS channel must have a 0 parts per million (ppm) difference between write and read clocks of the transmitter phase compensation FIFO.

For the 10-Gbps transceivers, there are no PCS blocks. The only transmit datapath available is a direct connection from the FPGA fabric to the serializer of the transmitter PMA.

Figure 2–21 shows the 10-Gbps transmitter datapath interface clocking. For each transmitter channel in a non-bonded configuration, the FPGA fabric forwards the following tx_clkout clock to the transmitter PMA.

Figure 2–21. Transmitter Datapath Interface Clocking for 10-Gbps Transceivers



For more information about interface clocking for each configuration, refer to the *Transceiver Custom Configuration in Arria V Devices* and *Transceiver Protocol Configurations in Arria V Devices* chapters.

You can clock the transmitter datapath interface by one of the following options:

- The Quartus II-selected transmitter datapath interface clock
- The user-selected transmitter datapath interface clock

To reduce GCLK, RCLK, and PCLK resource utilization in your design, you can select the user-selection option to share the transceiver datapath interface clocks.

Quartus II-Selected Transmitter Datapath Interface Clock

The Quartus II software automatically picks the appropriate clock from the FPGA fabric to clock the transmitter datapath interface. Figure 2–22 shows the transmitter datapath interface of two 6-Gbps transceiver non-bonded channels clocked by their respective transmitter PCS clocks, which are forwarded to the FPGA fabric.

Figure 2–22. 6-Gbps Transmitter Datapath Interface Clocking for Non-Bonded Channels

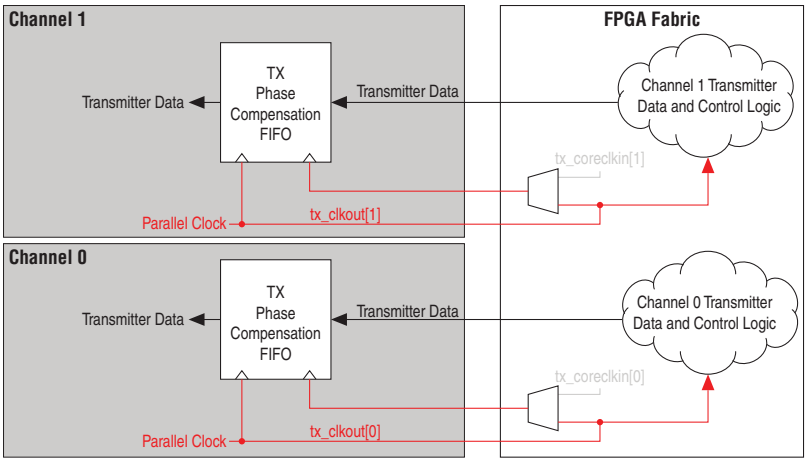


Figure 2–23 shows the transmitter datapath interface of two 10-Gbps transceiver non-bonded channels clocked by their respective transmitter PMA clocks, which are forwarded to the FPGA fabric.

Figure 2–23. 10-Gbps Transmitter Datapath Interface Clocking for Non-Bonded Channels

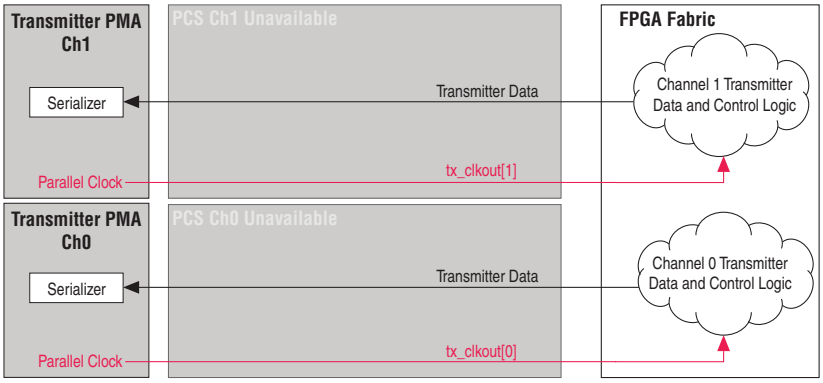
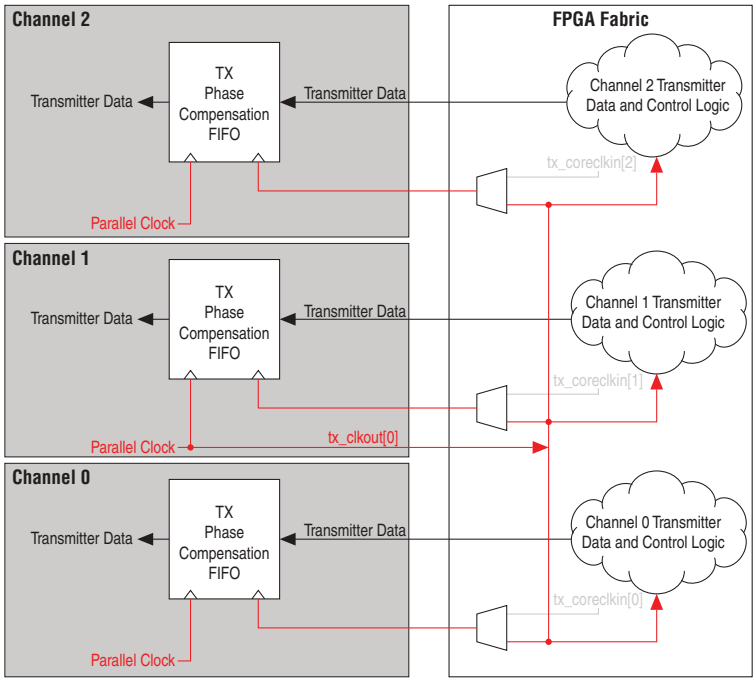


Figure 2–24 shows the 6-Gbps transmitter datapath interface of three bonded channels clocked by the `tx_clkout[0]` clock. The `tx_clkout[0]` clock is derived from the central clock divider of channel 1 or 4 in a transceiver bank.

Figure 2–24. 6-Gbps Transmitter Datapath Interface Clocking for Three Bonded Channels



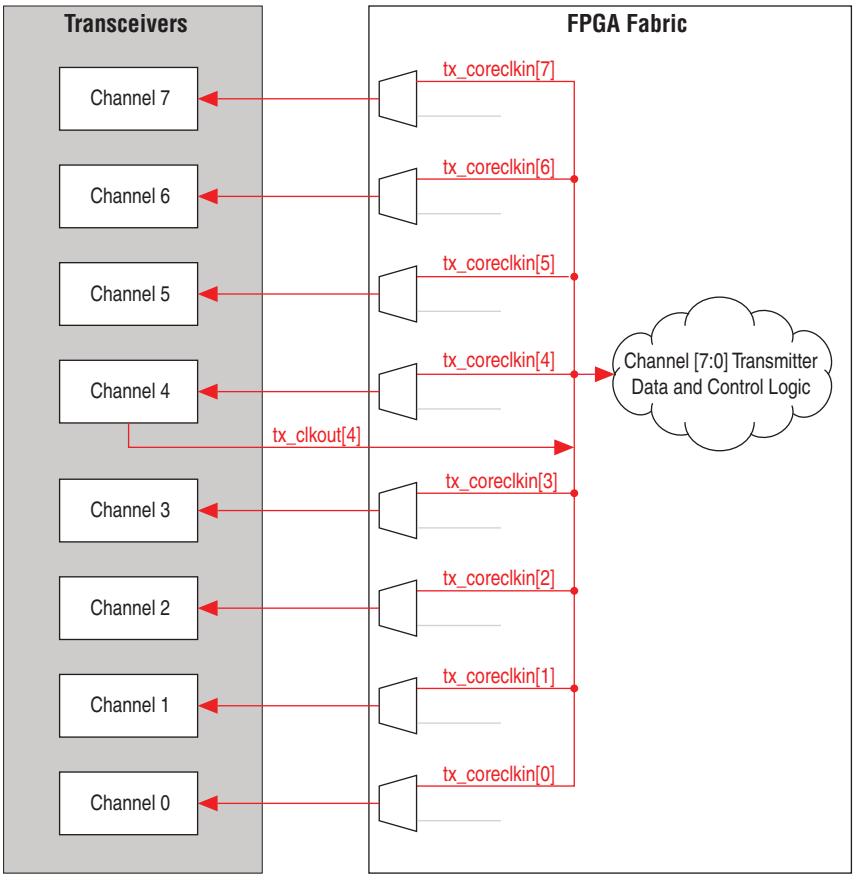
User-Selected Transmitter Datapath Interface Clock

Multiple transmitter channels that are non-bonded lead to high utilization of GCLK, RCLK, and PCLK resources (one clock resource per channel as shown in Figure 2–22). You can significantly reduce GCLK, RCLK, and PCLK resource use for transmitter datapath clocks if the transmitter channels are identical.

Identical transmitter channels have the same input reference clock source, transmit PLL configuration, transmitter PMA, and PCS configuration, but may have different analog settings, such as transmitter voltage output differential (V_{OD}), transmitter common-mode voltage (V_{CM}), or pre-emphasis.

To achieve the clock resource savings, select a common clock driver for the transmitter datapath interface of all identical transmitter channels. Figure 2–25 shows eight identical channels clocked by a single clock (`tx_clkout` of channel 4).

Figure 2–25. Eight Identical Channels with a Single User-Selected Transmitter Interface Clock



To clock eight identical channels with a single clock, perform these steps:

- Instantiate the `tx_coreclkin` port for all the identical transmitter channels (`tx_coreclkin[7:0]`).
- Connect `tx_clkout[4]` to the `tx_coreclkin[7:0]` ports.
- Connect `tx_clkout[4]` to the transmitter data and control logic for all eight channels.

Resetting or powering down channel 4 causes a loss of the clock for all eight channels.

The common clock must have a 0 ppm difference for the read side of the transmitter phase compensation FIFO of all the identical channels. A frequency difference causes the FIFO to under run or overflow, depending on whether the common clock is slower or faster, respectively.

You can drive the 0 ppm common clock by one of the following sources:

- `tx_clkout` of any channel in non-bonded channel configurations
- `tx_clkout[0]` in bonded channel configurations
- Dedicated `refclk` pins

The Quartus II software does not allow gated clocks or clocks that are generated in the FPGA logic to drive the `tx_coreclkin` ports.

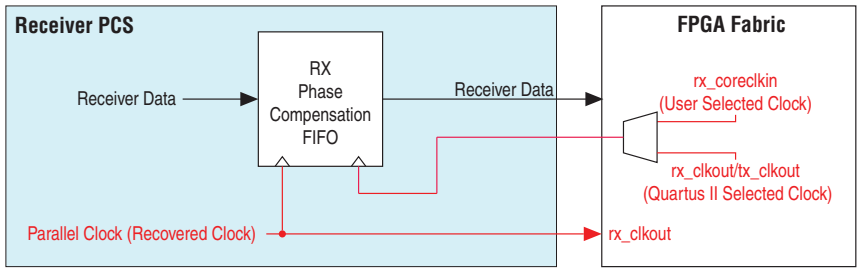
You must ensure a 0 ppm difference. The Quartus II software is unable to ensure a 0 ppm difference because it allows you to use external pins, such as dedicated `refclk` pins.

Receiver Datapath Interface Clock

The read side of the RX phase compensation FIFO makes up the 6-Gbps receiver datapath interface. The receiver datapath interface clock clocks this interface. Figure 2–26 shows the receiver datapath interface clocking. The receiver PCS forwards the following clocks to the FPGA fabric:

- `rx_clkout`—for each receiver channel in a non-bonded configuration when you do not use a rate matcher
- `tx_clkout`—for each receiver channel in a non-bonded configuration when you use a rate matcher
- single `rx_clkout[0]`—for all receiver channels in a bonded configuration

Figure 2–26. 6-Gbps Receiver Datapath Interface Clocking

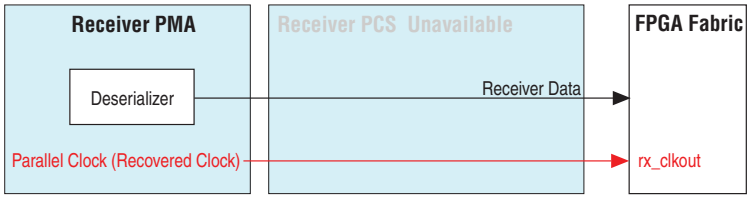


All configurations that use the PCS channel must have a 0 ppm difference between the receiver datapath interface clock and the read side clock of the RX phase compensation FIFO.

For the 10-Gbps transceivers, there are no PCS blocks. The only receiver datapath available is a direct connection from the receiver PMA deserializer to the FPGA fabric.

Figure 2–27 shows the 10-Gbps receiver datapath interface clocking. For each receiver channel in a non-bonded configuration, the receiver PMA forwards the `rx_clkout` clock to the FPGA fabric.

Figure 2–27. 10-Gbps Receiver Datapath Interface Clocking



For more information about interface clocking for each configuration, refer to the *Transceiver Custom Configuration in Arria V Devices* and *Transceiver Protocol Configurations in Arria V Devices* chapters.

You can clock the receiver datapath interface by one of the following options:

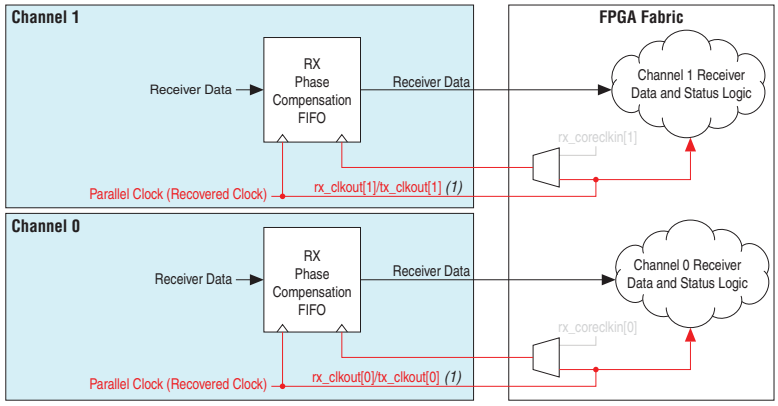
- The Quartus II-selected receiver datapath interface clock
- The user-selected receiver datapath interface clock

To reduce GCLK, RCLK, and PCLK resource utilization in your design, you can select the user-selection option to share the transceiver datapath interface clocks.

Quartus II Software-Selected Receiver Datapath Interface Clock

The Quartus II software automatically picks the appropriate clock from the FPGA fabric to clock the receiver datapath interface. Figure 2–28 shows the receiver datapath interface of two 6-Gbps transceiver non-bonded channels clocked by their respective receiver PCS clocks, which are forwarded to the FPGA fabric.

Figure 2–28. 6-Gbps Receiver Datapath Interface Clocking for Non-Bonded Channels



Note to Figure 2–28:

(1) If you use a rate matcher, the `tx_clkout` clock is used.

Figure 2–29 shows the receiver datapath interface of two 10-Gbps transceiver non-bonded channels clocked by their respective receiver CDR recovered PMA clocks, which are forwarded to the FPGA fabric.

Figure 2–29. 10-Gbps Receiver Datapath Interface Clocking for Non-Bonded Channels

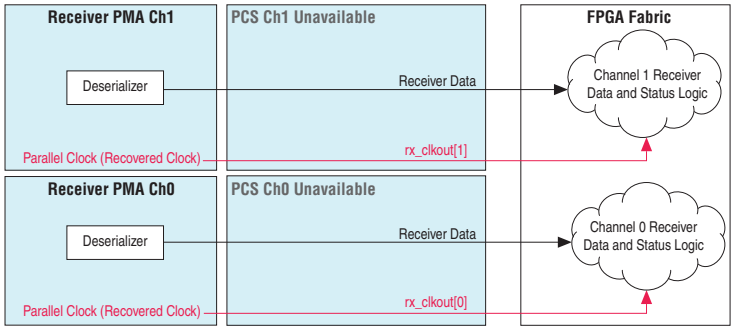
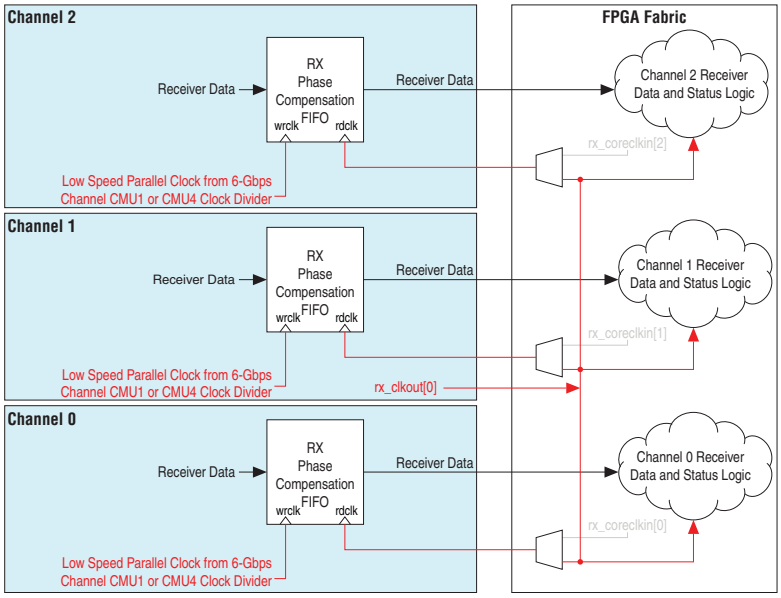


Figure 2–30 shows the 6-Gbps receiver datapath interface of three bonded channels clocked by the `tx_clkout[0]` clock. The `tx_clkout[0]` clock is derived from the central clock divider of channel 1 or 4 in a transceiver bank.

Figure 2–30. 6-Gbps Receiver Datapath Interface Clocking for Three Bonded Channels



User-Selected Receiver Datapath Interface Clock

Non-bonded multiple receiver channels lead to high utilization of GCLK, RCLK, and PCLK resources—one clock resource per channel, as shown in Figure 2–28. You can significantly reduce GCLK, RCLK, and PCLK resource use for the receiver datapath clocks if the receiver channels are identical.

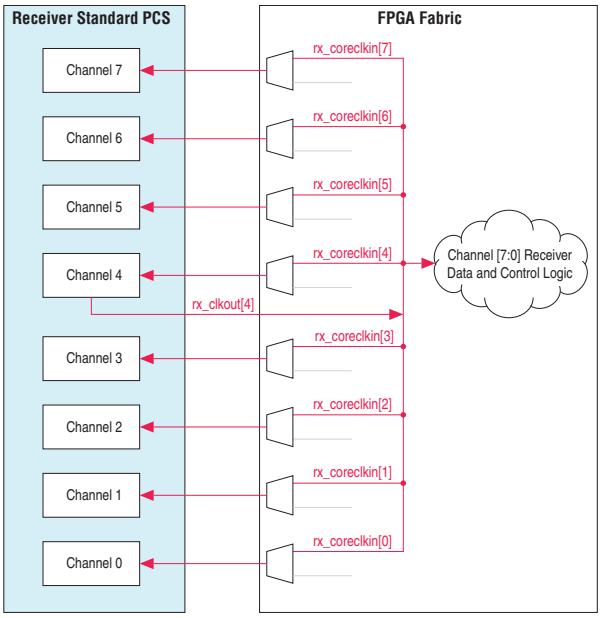
Identical receiver channels are defined as channels that have the same input reference clock source for the CDR and the same receiver PMA and PCS configuration. These channels may have different analog settings, such as receiver common mode voltage (V_{ICM}), equalization, or DC gain setting.

To achieve clock resource savings, select a common clock driver for the receiver datapath interface of all identical receiver channels. To select a common clock driver, perform these steps:

- Instantiate the `rx_coreclkin` port for all the identical receiver channels
- Connect the common clock driver to their receiver datapath interface, and receiver data and control logic.

Figure 2–31 shows eight identical channels that are clocked by a single clock (`rx_clkout` of channel 4).

Figure 2–31. Eight Identical Channels with a Single User-Selected Receiver Interface Clock



To clock eight identical channels with a single clock, perform these steps:

- Instantiate the `rx_coreclkin` port for all the identical receiver channels (`rx_coreclkin[7:0]`).
- Connect `rx_clkout[4]` to the `rx_coreclkin[7:0]` ports.
- Connect `rx_clkout[4]` to the receiver data and control logic for all eight channels.

Resetting or powering down channel 4 leads to a loss of the clock for all eight channels.

The common clock must have a 0 ppm difference for the write side of the RX phase compensation FIFO of all the identical channels. A frequency difference causes the FIFO to under run or overflow, depending on whether the common clock is faster or slower, respectively.

You can drive the 0 ppm common clock driver by one of the following sources:

- `tx_clkout` of any channel in non-bonded receiver channel configurations with the rate matcher
- `rx_clkout` of any channel in non-bonded receiver channel configurations without the rate matcher
- `tx_clkout[0]` in bonded receiver channel configurations
- Dedicated `refclk` pins

The Quartus II software does not allow gated clocks or clocks generated in the FPGA logic to drive the `rx_coreclkin` ports.

You must ensure a 0 ppm difference. The Quartus II software is unable to ensure a 0 ppm difference because it allows you to use external pins, such as dedicated `refclk` pins.

Document Revision History

Table 2–5 lists the revision history for this chapter.

Table 2–5. Document Revision History

Date	Version	Changes
June 2012	1.2	■ Updated Figure 2–5, Figure 2–6, and Figure 2–12. ■ Updated for the Quartus II software version12.0 ■ Added basic clocking information from obsoleted “basics” chapter.
November 2011	1.1	Updated chapter for clarity and Quartus II 11.1 update.
August 2011	1.0	Initial release

3. Transceiver Reset Control and Power-Down in Arria V Devices

AV53003-1.2

This chapter provides information about the transceiver reset control and transceiver power-down support in Arria® V devices.

You can implement the transceiver reset using the embedded reset controller in the PHY IP MegaCore® function. Use the transceiver reset controller to initialize the physical coding sublayer (PCS) and physical medium attachment (PMA) blocks. Use the reset sequence recommended in this chapter to ensure a reliable link initialization after the initial power-up and for reestablishing the link.

This chapter contains the following sections:

Transceiver Reset Controller

The embedded reset controller in the PHY IP MegaCore function provides an option to implement an automatic reset sequence, simplifying your transceiver-based design. The embedded reset controller requires only one control input to initiate the automatic reset sequence. There is only one embedded reset controller for all the channels in a PHY IP instance.

Figure 3–1 shows the embedded reset controller in the PHY IP MegaCore function.

Figure 3–1. Embedded Reset Controller in Arria V Devices

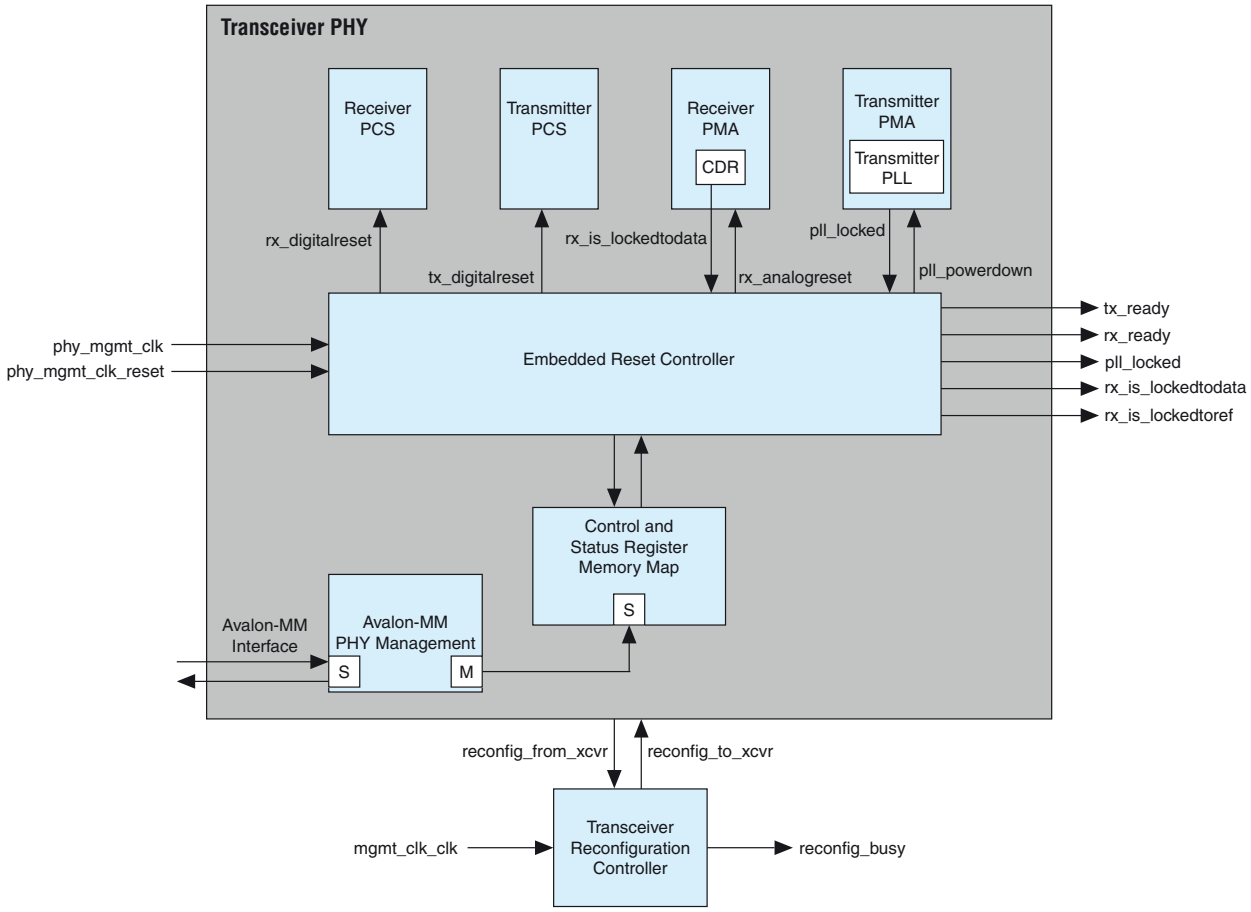


Table 3–1 lists the control inputs, status outputs, and internal signals.

Table 3–1. Transceiver Reset Control and Status Signals (Part 1 of 2)

Signal Name	Signal Type	Description
phy_mgmt_clk	Control Input	Clock for the embedded reset controller.
phy_mgmt_clk_reset	Control Input	A high-to-low transition of this asynchronous reset signal initiates the automatic reset sequence control.
reconfig_busy	Status Output	An output from the Transceiver Reconfiguration Controller block indicates the status of the dynamic reconfiguration controller. At the first mgmt_clk_clk clock cycle after power-up, reconfig_busy remains low. This signal is asserted from the second mgmt_clk_clk clock cycle to indicate that the offset cancellation process is active on clock data recovery (CDR). When the offset cancellation process is completed, the reconfig_busy signal is deasserted. This signal is also routed to the embedded reset controller by the Quartus® II software by embedding the signal with the reconfig_to_xcvr bus between the PHY IP and the ALT_XCVR_RECONFIG block.

Table 3–1. Transceiver Reset Control and Status Signals (Part 2 of 2)

Signal Name	Signal Type	Description
tx_ready	Status Output	A low-to-high transition of this signal indicates that the transmitter (TX) channel is out of reset and is ready for data transmission. This signal is synchronous to phy_mgmt_clk.
rx_ready	Status Output	A low-to-high transition of this signal indicates that the receiver (RX) channel is out of reset and is ready for data reception. This signal is synchronous to phy_mgmt_clk.
pll_powerdown	Internal Control	When this signal is asserted, the TX phase-locked loop (PLL) and all blocks in the TX PMA are reset. The pll_powerdown signal can only be controlled by using the embedded reset controller. For user control, refer to "User-Controlled Reset Controller" on page 3–4.
tx_digitalreset	Internal Control	When this signal is asserted, all blocks in the TX PCS are reset.
rx_analogreset	Internal Control	When this signal is asserted, the RX CDR and all blocks in the RX PMA are reset. After reconfig_busy is low, this signal can be deasserted.
rx_digitalreset	Internal Control	When this signal is asserted, all blocks in the RX PCS are reset.
pll_locked	Internal Status	This signal is asserted to indicate that the TX PLL achieves lock to the input reference clock (phy_mgmt_clk). When this signal is asserted high, the embedded reset controller deasserts the tx_digitalreset signal.
	Status Output	
rx_is_lockedtodata	Internal Status	When this signal is asserted, the embedded reset controller deasserts the rx_digitalreset signal. When this signal is deasserted, the embedded reset controller asserts the rx_digitalreset signal.
	Status Output	This signal is an optional output status port. When asserted, this signal indicates that the CDR is locked to the RX data and the CDR has changed from lock-to-reference (LTR) to lock-to-data (LTD) mode.
rx_is_lockedtoref	Status Output	This is an optional output status port. When asserted, this signal indicates that the CDR is locked to the reference clock.

You cannot control the pll_powerdown signal using memory map registers.

Table 3–2 lists the memory map registers for CDR lock mode and channel reset.

Table 3–2. Transceiver Manual Reset Control Using Memory Map Registers

Register Name	Description
pma_rx_set_locktodata	This register is for CDR manual lock mode. When you set the register to high, the RX CDR PLL locks to the incoming data.
pma_rx_set_locktoref	This register is for CDR manual lock mode. When you set the register to high, the RX CDR PLL locks to the reference clock.
reset_tx_digital	When you set this register to high, the tx_digitalreset signal is asserted in every channel that is enabled for reset control. To deassert the tx_digitalreset signal, set the reset_tx_digital register to low.
reset_rx_analog	When you set this register to high, the rx_analogreset signal is asserted in every channel that is enabled for reset control. To deassert the rx_analogreset signal, set the reset_rx_analog register to low.
reset_rx_digital	When you set this register to high, the rx_digitalreset signal is asserted in every channel that is enabled for reset control. To deassert the rx_digitalreset signal, set the reset_rx_digital register to low.

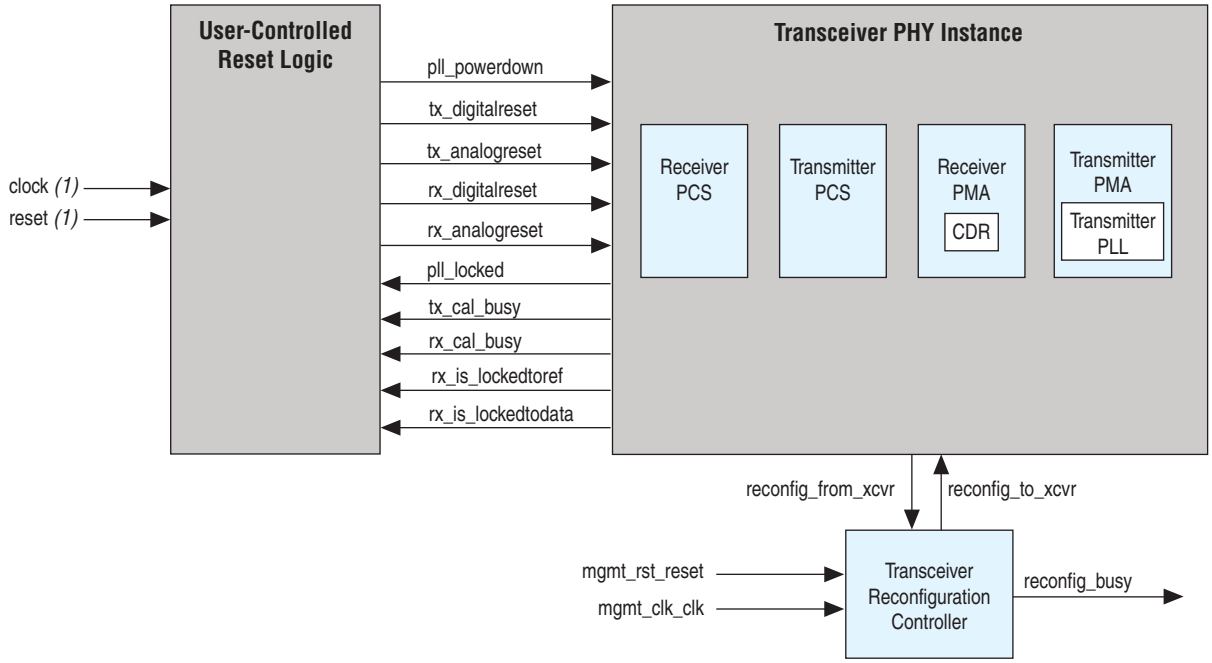
If you set the pma_rx_set_locktodata or pma_rx_set_locktoref register to high, the CDR is placed in manual lock mode.

The reset_ch_bitmask registers provide an option to enable or disable certain channels in a PHY IP instance for reset control. By default, all channels in a PHY IP instance are enabled for reset control.

User-Controlled Reset Controller

Figure 3–2 shows a block diagram of the transceiver PHY instance and the Transceiver Reconfiguration Controller interacting with the user-controlled reset controller.

Figure 3–2. Interaction Between the Transceiver PHY Instance, Transceiver Reconfiguration Controller, and the User-Controlled Reset Controller



Note to Figure 3–2:

(1) You can use phy_mgmt_clk and phy_mgmt_clk_reset as the clock and reset to the user-controlled reset logic.

Table 3–3 lists the signals used by the transceiver PHY instance, Transceiver Reconfiguration Controller, and user-controlled reset controller.

Table 3–3. Signals Used by the Transceiver PHY instance, Transceiver Reconfiguration Controller, and User-Controlled Reset Controller (1) (Part 1 of 2)

Signal Name	Signal Type	Description
mgmt_clk_clk	Clock	Clock for the Transceiver Reconfiguration Controller
mgmt_rst_reset	Reset	Reset for the Transceiver Reconfiguration Controller
pll_powerdown	Control	Resets the TX PLL when asserted high
tx_analogreset	Control	Resets the TX PMA when asserted high
tx_digitalreset	Control	Resets the TX PCS when asserted high

Table 3–3. Signals Used by the Transceiver PHY instance, Transceiver Reconfiguration Controller, and User-Controlled Reset Controller (1) (Part 2 of 2)

Signal Name	Signal Type	Description
rx_analogreset	Control	Resets the RX PMA when asserted high
rx_digitalreset	Control	Resets the RX PCS when asserted high
reconfig_busy	Status	A high on this signal indicates that reconfiguration is active
tx_cal_busy	Status	A high on this signal indicates that TX calibration is active
rx_cal_busy	Status	A high on this signal indicates that RX calibration is active
pll_locked	Status	A high on this signal indicates that the TX PLL is locked
rx_is_lockedtoref	Status	A high on this signal indicates that the RX CDR is in the LTR mode
rx_is_lockedtodata	Status	A high on this signal indicates that the RX CDR is in the LTD mode

Note to Table 3–3:

(1) Some of the signal names may be slightly different from the register or port names in the Quartus II software.

If you disable the embedded reset controller, you must implement your own reset controller logic. You must consider the following:

- The user-controlled reset controller is level sensitive (active high)
- The user-controlled reset controller does not depend on phy_mgmt_clk_reset
- You are responsible for providing a clock and reset to the reset controller logic
- You can hold the transceiver channels in reset by asserting the appropriate reset control signals
- You can create a reset controller to reset individual channels independently. Alternatively, you can create an instance of the Transceiver PHY Reset Controller.

  For more information, refer to the *Altera Transceiver PHY IP Core User Guide*.

- The tx_ready and rx_ready status signals are not available in the user-controlled reset controller mode

The embedded reset controller can only be disabled for Custom PHY IP and the Deterministic Latency PHY IP.

The Native PHY IP has only the user-controlled reset controller option.

CDR is set to automatic lock mode by default. To apply CDR manual lock mode, refer to the descriptions in Table 3–2 on page 3–3.

Transceiver Reset Sequence

After device power-up and phy_mgmt_clk_reset is set to low, the embedded reset controller automatically performs the reset sequence for the transmitter and receiver.

If phy_mgmt_clk_reset is set to high after device power-up, the embedded reset controller holds the transmitter and receiver in reset and the calibration process is on hold. After phy_mgmt_clk_reset is deasserted, the embedded reset controller performs an automatic reset sequence for the transmitter and receiver and the calibration process starts.

During device operation, you can perform the reset sequence with the following options:

- Automatic reset sequence using the embedded reset controller for both the transmitter and receiver.
- Manual reset sequence using the memory map registers for the receiver. This option allows you to perform the reset sequence for the receiver only while in duplex mode.

The transceiver reset control can use the CDR in automatic or manual lock mode. The mode is determined by the memory map registers `pma_rx_set_locktoref` and `pma_rx_set_locktodata`. By default, CDR is in automatic lock mode.

After the transceiver dynamic reconfiguration completes, as indicated by the deassertion of the `reconfig_busy` signal, you must repeat the entire reset sequence.

Each reset sequence described in this section is the full reset sequence for a duplex channel unless otherwise stated. For a transmitter- or receiver-only channel, only the signals related to the transmitter or receiver are used.

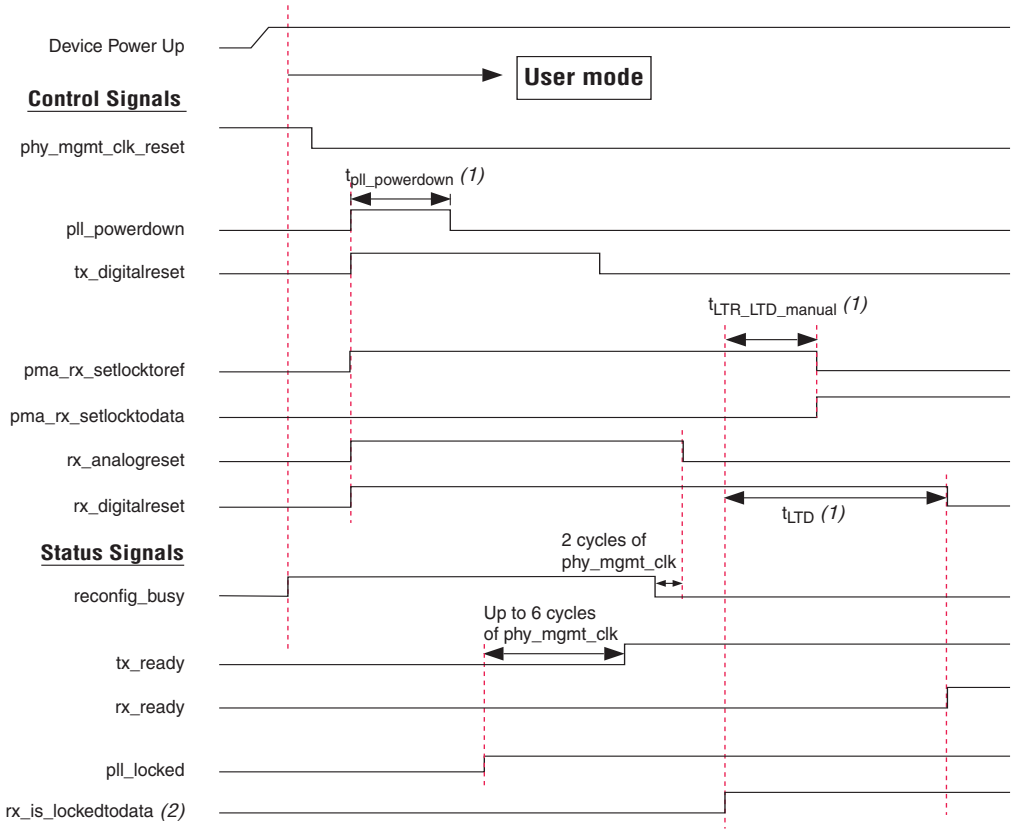
Automatic Reset Sequence Using the Embedded Reset Controller

The recommended transceiver reset sequence is different for non-PCI Express® (PCIe®) and PCIe configurations.

Reset Sequence in Non-PCIe Configurations

Figure 3–3 shows the transceiver automatic reset sequence timing diagram for non-PCIe configurations using the embedded reset controller with CDR manual lock mode.

Figure 3–3. Timing Diagram for Transceiver Automatic Reset Sequence in Non-PCIe Configurations with CDR Manual Lock Mode



Notes to Figure 3–3:

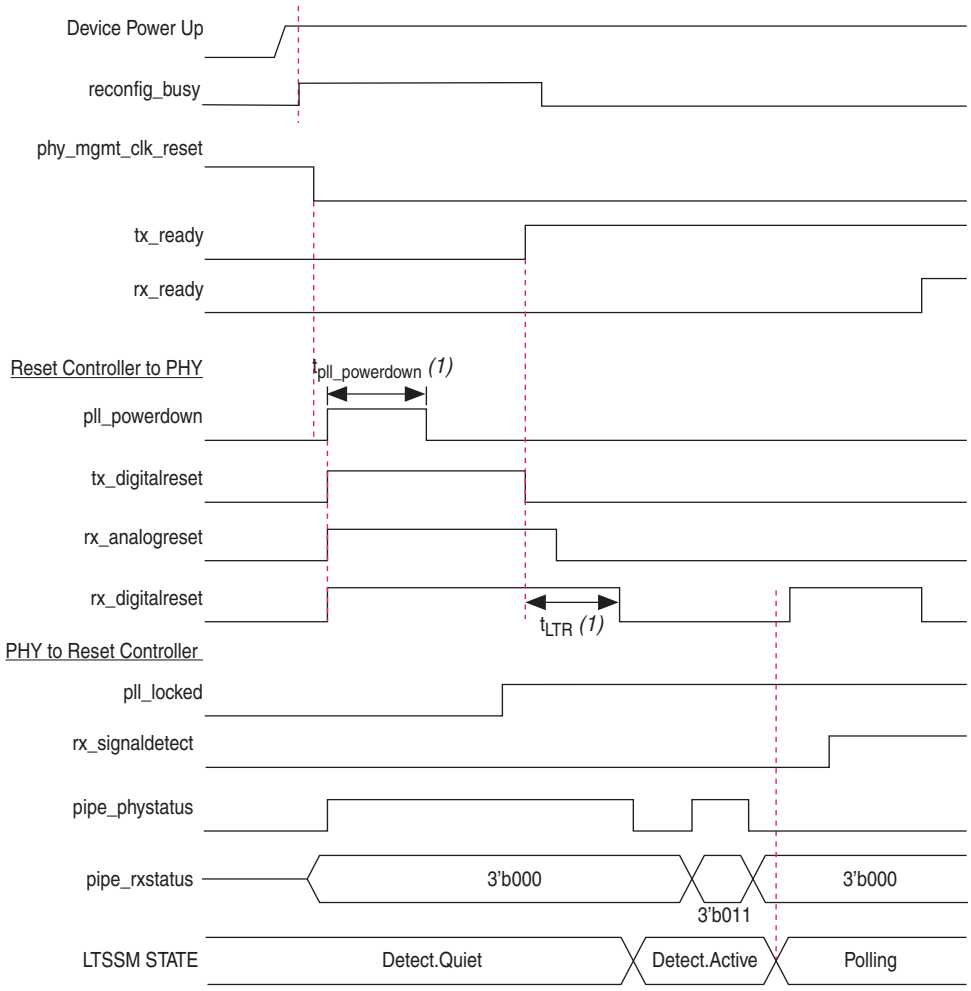
(1) $t_{pll\_powerdown}$, $t_{LTR\_LTD\_manual}$, and t_{LTD} are pending characterization.
(2) In bonded mode configurations, the `rx_is_lockedtodata` signal shown in this figure is the logical AND of the `rx_is_lockedtodata` signals from all channels.

In Figure 3–3, $t_{LTR\_LTD\_manual}$ is the timing requirement when using CDR manual lock mode. t_{LTD} is the timing requirement when using CDR automatic lock mode.

Reset Sequence in PCIe Configuration

Figure 3–4 shows the transceiver reset sequence timing diagram for PCIe configurations using the embedded reset controller in the PHY IP MegaCore function.

Figure 3–4. Timing Diagram for Transceiver Reset Sequence in PCIe Configurations



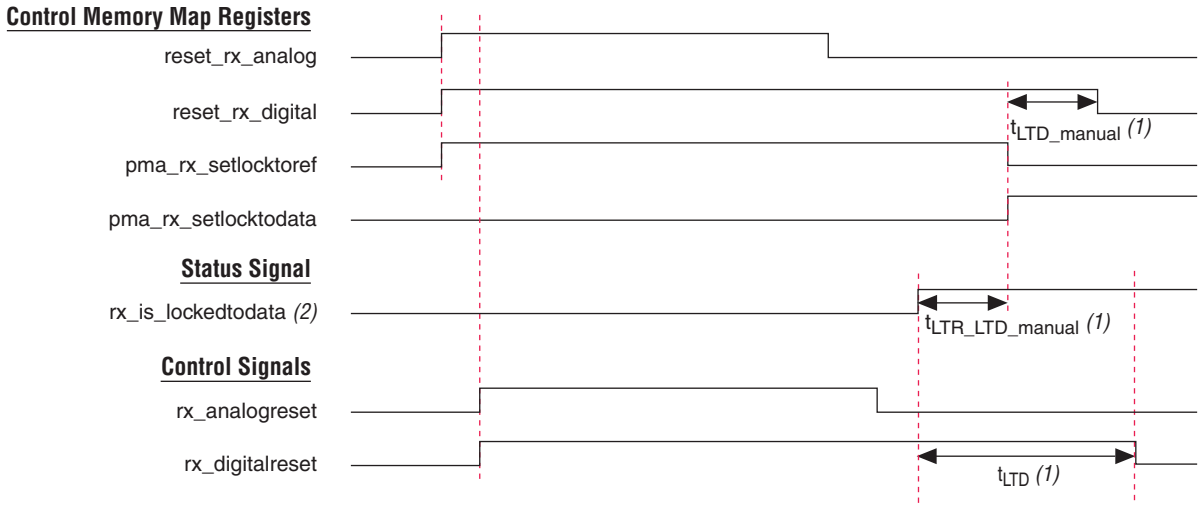
Note to Figure 3–4:

(1) $t_{pll\_powerdown}$ and t_{LTR} are pending characterization.

Manual Reset Sequence

Figure 3–5 shows the timing diagram for the receiver manual reset sequence during the device operation for non-PCIe configurations with CDR manual lock mode. This option allows you to perform the reset sequence for only the receiver in a duplex channel configuration.

Figure 3–5. Timing Diagram for Receiver Manual Reset Sequence During Device Operation for Non-PCIe Configurations with CDR Manual Lock Mode



Notes to Figure 3–5:

(1) $t_{LTR\_LTD\_manual}$, t_{LTD_manual}, and t_{LTD} are pending characterization.

(2) In bonded mode configurations, the `rx_is_lockedtodata` signal shown in this figure is the logical AND of the `rx_is_lockedtodata` signals from all channels.

In Figure 3–5, $t_{LTR\_LTD\_manual}$ and t_{LTD_manual} are the timing requirements when using CDR manual lock mode. t_{LTD} is the timing requirement when using CDR automatic lock mode.

Transceiver Power-Down

To maximize power savings, enable PMA hard power-down across all channels on a side of the device where you do not use the transceivers.

The power granularity control of the transceiver PMA is per side. To enable PMA hard power-down on the left or right side of the device, ground the transceiver power supply of the respective side.

To power up the PMA on the device side, supply power to the side and perform the initialization according to the recommended reset sequence, as shown in Figure 3–3 on page 3–7 and Figure 3–4 on page 3–8.

When you configure the Arria V device, the Quartus II software automatically selects the power-down channel feature. All unused transceiver channels and blocks are powered down to reduce overall power consumption.

For information about the transceiver power supply operating conditions of the left and right side of Arria V devices, refer to the *Arria V Device Datasheet*.

Document Revision History

Table 3–4 lists the revision history for this chapter.

Table 3–4. Document Revision History

Date	Version	Changes
June 2012	1.2	■ Added "User-Controlled Reset Controller" section. ■ Updated Figure 3–1 and Table 3–1.
November 2011	1.1	■ Updated all figures and tables. ■ Reorganized and updated the "Transceiver Reset Sequence" section.
August 2011	1.0	Initial release.

4. Transceiver Protocol Configurations in Arria V Devices

AV53004-1.2

This chapter provides the transceiver channel datapath, clocking guidelines, channel placement guidelines, and a brief description of protocol features supported in each transceiver configuration for Arria® V devices.

This chapter contains the following sections:

Transceiver PCS Features

Arria V devices have dedicated transceiver physical coding sublayer (PCS) and physical medium attachment (PMA) circuitry to support the communication protocols listed in Table 4–1.

Table 4–1. Transceiver PCS Features for Arria V Devices (Part 1 of 2) (1)

PCS Support	Data Rates (Gbps)	Transmitter Datapath	Receiver Datapath
PCI Express® (PCIe®) Gen1 (x1, x2, x4, and x8)	2.5	The same as custom single- and double-width modes, plus the PHY interface for PCI Express (PIPE) 2.0 interface to the core logic	The same as custom single- and double-width modes, plus the rate match FIFO and PIPE 2.0 interface to the core logic
Gbps Ethernet (GbE)	1.25, 3.125	The same as custom single- and double-width modes	The same as custom single- and double-width modes, plus the rate match FIFO
Serial Digital Interface (SDI)	0.27 (2), 1.485, and 2.97	Phase compensation FIFO and byte serializer	Phase compensation FIFO and byte deserializer
SATA	1.5, 3.0, and 6.0	Phase compensation FIFO, byte serializer, and 8B/10B encoder	Phase compensation FIFO, byte deserializer, word aligner, and 8B/10B decoder

Table 4–1. Transceiver PCS Features for Arria V Devices (Part 2 of 2) (1)

PCS Support	Data Rates (Gbps)	Transmitter Datapath	Receiver Datapath
Common Public Radio Interface (CPRI)	0.6144, 1.2288, 2.4576, 3.072, 4.9152, 6.144, 9.8304 (3)	The same as custom single- and double-width modes, plus the transmitter (TX) deterministic latency	The same as custom single- and double-width modes, plus the receiver (RX) deterministic latency
OBSAI	0.768, 1.536, 3.072, 6.144	The same as custom single- and double-width modes, plus the TX deterministic latency	The same as custom single- and double-width modes, plus the RX deterministic latency
Serial RapidIO® (SRIO)	1.25, 2.5, 3.125	The same as custom single- and double-width modes	The same as custom single- and double-width modes

Notes to Table 4–1:

(1) Not yet supported by the Quartus® II software version 12.0.

(2) The 0.27 gigabits per second (Gbps) data rate is supported using oversampling user logic that must be implemented in the FPGA core.

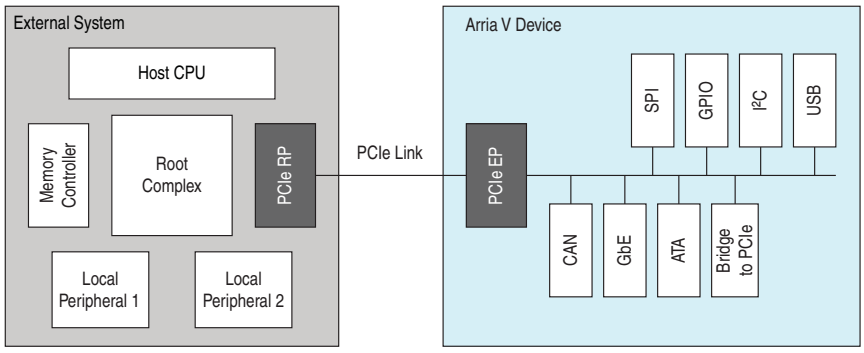
(3) For the 9.8304 Gbps CPRI implementation (Arria GT devices only), there is no hard PCS. To interface with the PMA in a PMA Direct configuration, soft PCS is required.

For a complete list of transceiver protocols supported by Arria V devices, refer to the *Upcoming Arria V Device Features* document. For a complete list of serial protocols supported by Arria V devices, refer to the *Arria V Device Overview*.

PCI Express

The Arria V devices have PCIe hard IP—consisting of the media access control (MAC) lane, data link, and transaction layers—that is designed for performance, ease-of-use, and increased functionality. The PCIe hard IP supports the PCIe Gen1 end point and root port up to x8 lane configurations. The PCIe endpoint support includes multifunction support for up to eight functions and Gen2 x4 lane configurations, as shown in Figure 4–1.

Figure 4–1. PCIe Multifunction for Arria V Devices



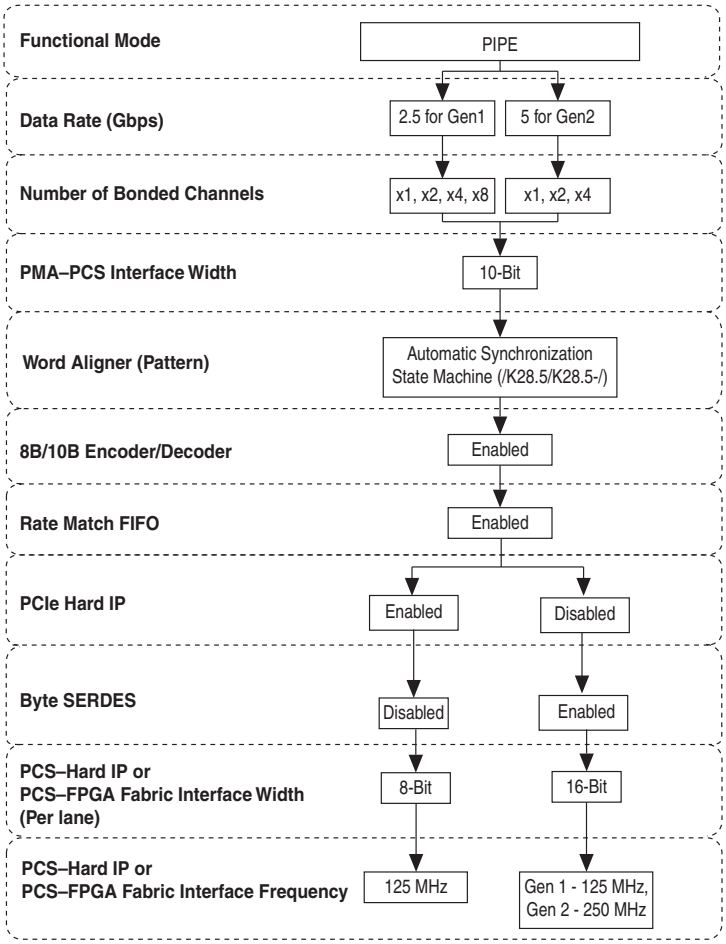
The Arria V PCIe hard IP operates independently from the core logic, which allows the PCIe link to wake up and complete link training in less than 100 ms while the Arria V device completes loading the programming file for the rest of the device.

In addition, the Arria V device PCIe hard IP has improved end-to-end datapath protection using error correction code (ECC).

Transceiver Datapath

Figure 4–2 shows the transceiver configurations allowed in a PIPE configuration.

Figure 4–2. Arria V Transceivers in a PIPE Configuration *(1)*



Note to Figure 4–2:

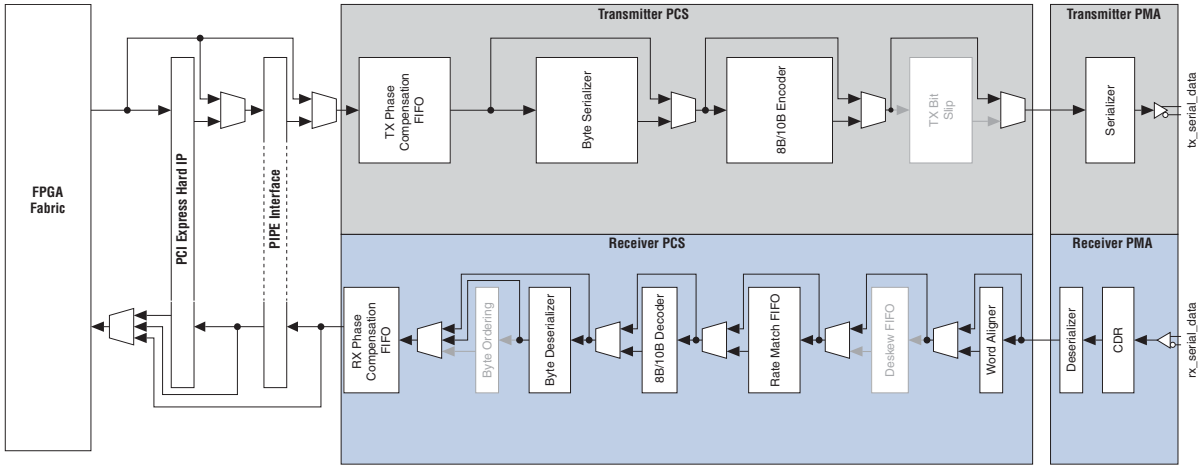
(1) The PCIe Gen2 (up to x4) is supported only through the PCS–hard IP interface.

The transceiver datapath clocking varies between non-bonded (x1) and bonded (x2, x4, and x8) configurations. For more information about transceiver datapath clocking in different PIPE configurations, refer to "Transceiver Clocking" on page 4–10.

Transceiver Channel Datapath

Figure 4–3 shows the Arria V transmitter and receiver channel datapath in a PIPE configuration.

Figure 4–3. Arria V Transmitter Channel Datapath in a PIPE Configuration



For more information about the blocks in the transmitter datapath, refer to the *Transceiver Architecture in Arria V Devices* chapter.

Supported Features

The PIPE configuration for the 2.5 Gbps (Gen1) data rate supports these features:

- PCIe-compliant synchronization state machine
- ±300 parts per million (ppm)—total 600 ppm—clock rate compensation
- 8-bit FPGA fabric–transceiver interface
- 16-bit FPGA fabric–transceiver interface
- Transmitter buffer electrical idle
- Receiver detection
- 8B/10B encoder disparity control when transmitting compliance pattern
- Power state management (Electrical Idle only)
- Receiver status encoding

PIPE Interface

In a PIPE configuration, each channel has a PIPE interface block that transfers data, control, and status signals between the PHY-MAC layer and the transceiver channel PCS and PMA blocks. The PIPE interface block complies with version 2.0 of the PIPE specification. If you use the PIPE hard IP block, the PHY-MAC layer is implemented in the hard IP block. Otherwise, you can implement the PHY-MAC layer using soft IP in the FPGA fabric.

The PIPE interface block is only used in a PIPE configuration and cannot be bypassed.

In addition to transferring data, control, and status signals between the PHY-MAC layer and the transceiver, the PIPE interface block implements the following functions that are required in a PCIe-compliant physical layer device:

- Forces the transmitter buffer to an electrical idle state
- Initiates the receiver detect sequence
- Controls the 8B/10B encoder disparity when transmitting a compliance pattern
- Manages the PCIe power states
- Indicates the completion of various PHY functions, such as receiver detection and power state transitions on the `pipe_phystatus` signal
- Encodes the receiver status and error conditions on the `pipe_rxstatus[2:0]` signal, as specified in the PCIe specification

Transmitter Electrical Idle Generation

The PIPE interface block in Arria V devices places the channel transmitter buffer in an electrical idle state when the electrical idle input signal is asserted. During electrical idle, the transmitter buffer differential and common configuration output voltage levels are compliant to the PCIe Base Specification 2.0 for the PCIe Gen1 data rate.

The PCIe specification requires that the transmitter buffer be placed in electrical idle in certain power states. For more information about input signal levels required in different power states, refer to the "Power State Management" section.

Power State Management

The PCIe specification defines four power states—P0, P0s, P1, and P2—that the physical layer device must support to minimize power consumption:

- P0 is the normal operating state during which packet data is transferred on the PCIe link.
- P0s, P1, and P2 are low-power states into which the physical layer must transition as directed by the PHY-MAC layer to minimize power consumption.

The PIPE interface in Arria V transceivers provides an input port for each transceiver channel configured in a PIPE configuration.

When transitioning from the P0 power state to lower power states (P0s, P1, and P2), the PCIe specification requires that the physical layer device implements power saving measures. The Arria V transceivers do not implement these power saving measures except to place the transmitter buffer in electrical idle in the lower power states.

8B/10B Encoder Usage for Compliance Pattern Transmission Support

The PCIe transmitter transmits a compliance pattern when the LTSSM state machine enters a polling compliance substate. The polling compliance substate assesses if the transmitter is electrically compliant with the PCIe voltage and timing specifications.

Receiver Electrical Idle Inference

The PCIe protocol allows inferring the electrical idle condition at the receiver instead of detecting the electrical idle condition using analog circuitry.

In all PIPE configurations, each receiver channel PCS has an optional Electrical Idle Inference module designed to implement the electrical idle inference conditions specified in the PCIe Base Specification 2.0.

Receiver Status

The PCIe specification requires that the PHY encode the receiver status on a 3-bit status signal (`pipe_rxstatus[2:0]`). This status signal is used by the PHY-MAC layer for its operation. The PIPE interface block receives the status signals from the transceiver channel PCS and PMA blocks, and encodes the status on the `pipe_rxstatus[2:0]` signal to the FPGA fabric. The encoding of the status signals on the `pipe_rxstatus[2:0]` signal is compliant with the PCIe specification.

Receiver Detection

The PIPE interface block in Arria V transceivers provides an input signal (`pipe_txdetectrx_loopback`) for the receiver detect operation that is required by the PCIe protocol during the detect substate of the LTSSM.

When the `pipe_txdetectrx_loopback` signal is asserted in the P1 power state, the PCIe interface block sends a command signal to the transmitter buffer in that channel to initiate a receiver detect sequence. In the P1 power state, the transmitter buffer must always be in the electrical idle state.

After receiving this command signal, the receiver detect circuitry creates a step voltage at the output of the transmitter buffer. If an active receiver that complies with the PCIe input impedance requirements is present at the far end, the time constant of the step voltage on the trace is higher than if the receiver is not present. The receiver detect circuitry monitors the time constant of the step signal that is seen on the trace to determine if a receiver was detected. The receiver detect circuitry monitor requires a 125-MHz clock for operation that you must drive on the `fixedclk` port.

For the receiver detect circuitry to function reliably, the AC-coupling capacitor on the serial link and the receiver termination values used in your system must be compliant with the PCIe Base Specification 2.0.

The PCI Express PHY (PIPE) IP core provides a 1-bit PHY status (`pipe_phystatus`) and a 3-bit receiver status signal (`pipe_rxstatus[2:0]`) to indicate whether a receiver was detected or not, in accordance to the PIPE 2.0 specifications.

Clock Rate Compensation Up to ±300 ppm

In compliance with the PCIe protocol, the Arria V receiver channels are equipped with a rate match FIFO to compensate for the small clock frequency differences of up to ±300 ppm between the upstream transmitter and local receiver clocks.

For more information about the rate match FIFO, refer to the *Transceiver Architecture for Arria V Devices* chapter.

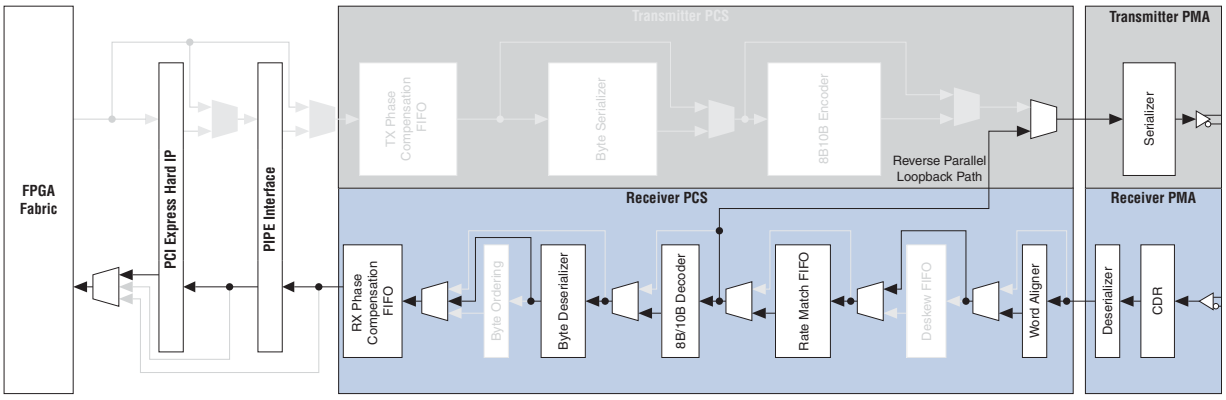
PCIe Reverse Parallel Loopback

The PCIe reverse parallel loopback is only available in the PCIe functional configuration for the Gen1 data rate. The received serial data passes through the receiver CDR, deserializer, word aligner, and rate matching FIFO buffer, as shown in Figure 4–4. It is then looped back to the transmitter serializer and transmitted out through the transmitter buffer. The received data is also available to the FPGA fabric through the port. This loopback mode is compliant with the PCIe specification 2.0.

Arria V devices provide the `pipe_txdetectrx_loopback` input signal to enable this loopback mode. If the `pipe_txdetectrx_loopback` signal is asserted in the P1 power state, receiver detection is performed. If the signal is asserted in the P0 power state, reverse parallel loopback is performed.

The PCIe reverse parallel loopback is the only loopback option that is supported in PIPE configurations.

Figure 4–4. PIPE Reverse Parallel Loopback Mode Datapath *(1)*



Note to Figure 4–4:

(1) The grayed out blocks are inactive.

Transceiver Channel Placement Guidelines

Table 4–2 lists the physical placement of PIPE channels in x1, x2, x4, and x8 bonding configurations.

Table 4–2. PIPE Channel Placement for PCIe Gen1

Configuration	Data Channel Placement	Minimum Channel Utilization (1)
x1	Any channel	2 (1 data channel, 1 clock channel)
x2	2 contiguous channels	3 (2 data channels, 1 clock channel)
x4	4 contiguous channels	5 (4 data channels, 1 clock channel)
x8	8 contiguous channels	9 (8 data channels, 1 clock channel)

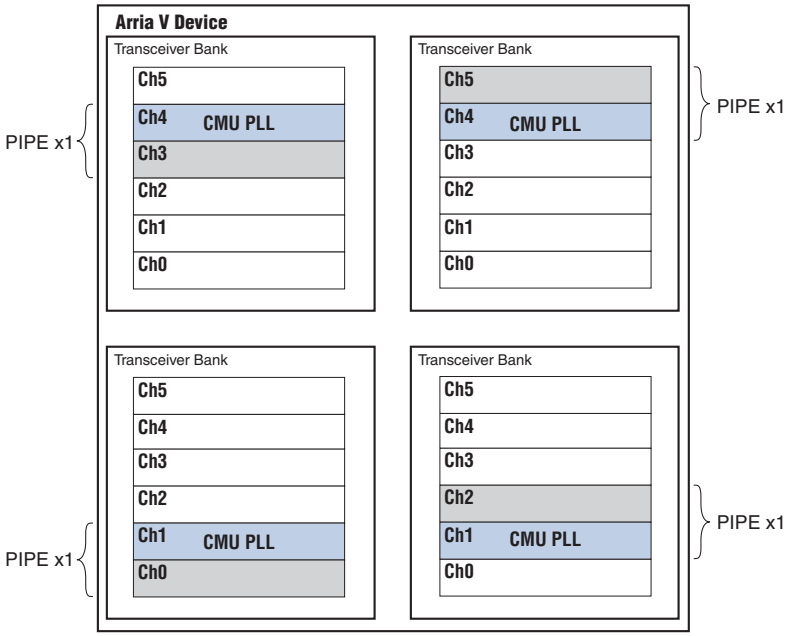
Note to Table 4–2:

(1) Placement by the Quartus II software may vary with design—resulting in higher channel utilization.

For more information about the hard IP implementation of PCIe and restrictions, refer to the "Transceiver Banks" section of the *Transceiver Architecture in Arria V Devices*.

Figure 4–5, Figure 4–6, and Figure 4–7 show examples of channel placement for PIPE x1, x2, x4, and x8 configurations.

Figure 4–5. Example of PIPE x1 Channel Placement (1), (2), (3)



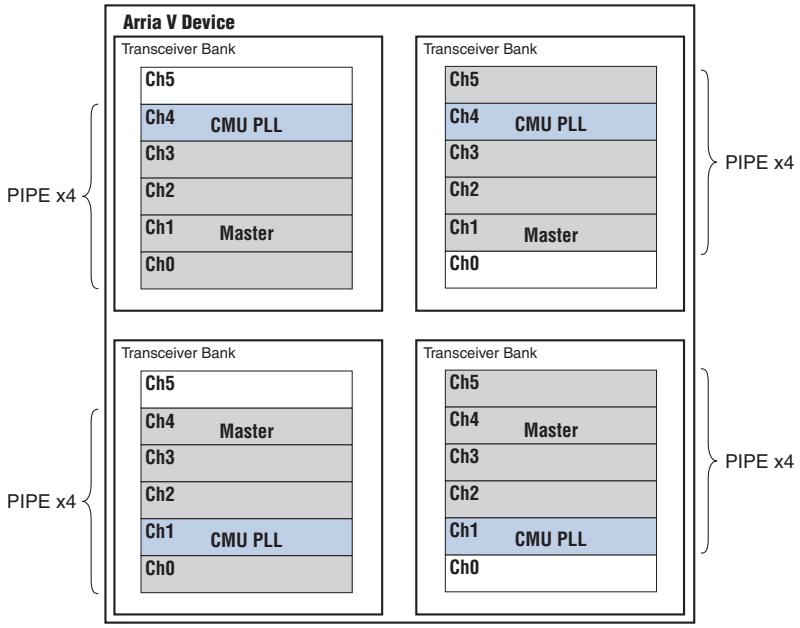
Notes to Figure 4–5:

(1) Channels shaded in blue provide the high-speed serial clock.

(2) Channels shaded in gray are data channels.

(3) You can place the PIPE data channels in any available channel in the transceiver bank.

Figure 4–6. Example of PIPE x2 and x4 Channel Placement *(1), (2), (3), (4)*



Notes to Figure 4–6:

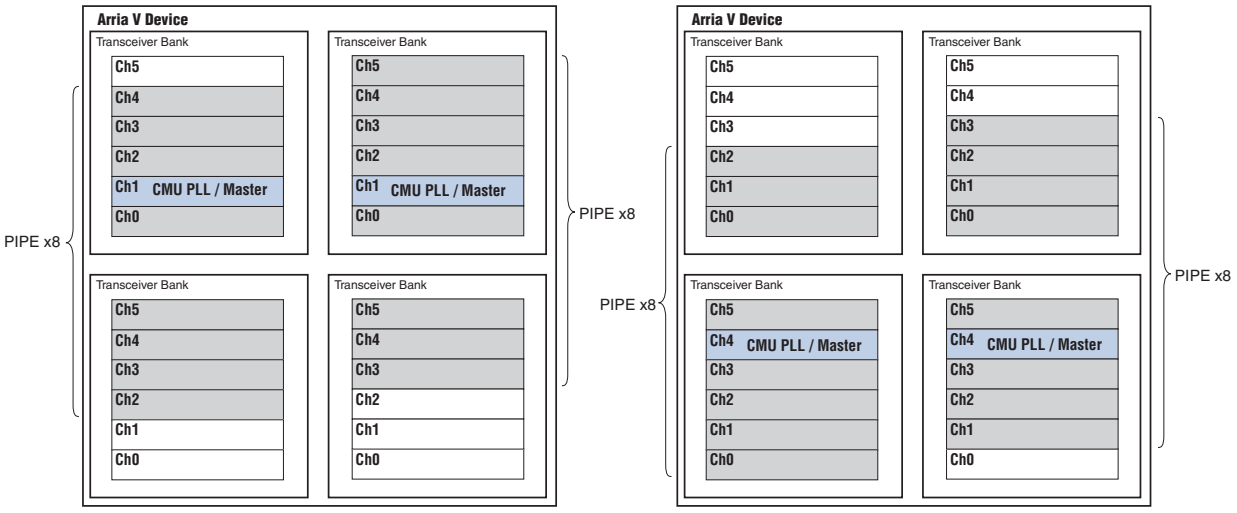
(1) Channels shaded in blue provide the high-speed serial clock.

(2) Channels shaded in gray are data channels.

(3) The Quartus II software automatically places the clock generator and master in either channels 1 or 4 within a transceiver bank. The master is referring to the transceiver channel with the central clock divider.

(4) The x2 channel placement for the clock generator and master channel is the same as the x4 channel placement. The other channel for the x2 configuration can be placed in any of the channels shaded in gray.

Figure 4–7. Example of PIPE x8 Channel Placement *(1), (2)*



Notes to Figure 4–7:

(1) Channels shaded in blue provide the high-speed serial clock.

(2) Channels shaded in gray are data channels.

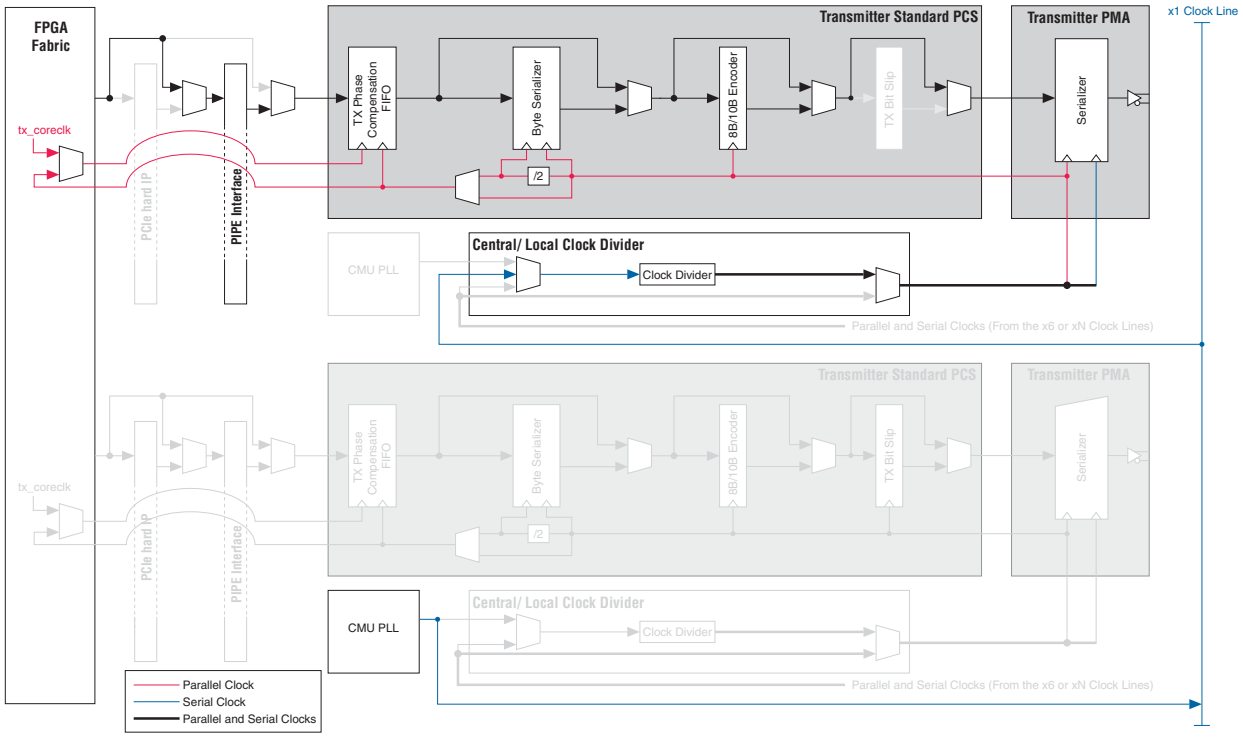
Transceiver Clocking

This section describes transceiver clocking for PIPE configurations.

PIPE x1 Configuration

Figure 4–8 shows the transceiver clocking configuration in a PIPE x1 configuration. The serial clock is provided by the CMU PLL in a channel different from that of the data channel. The local clock divider block in the data channel generates a parallel clock from this high-speed clock and distributes both clocks to the PMA and PCS of the data channel.

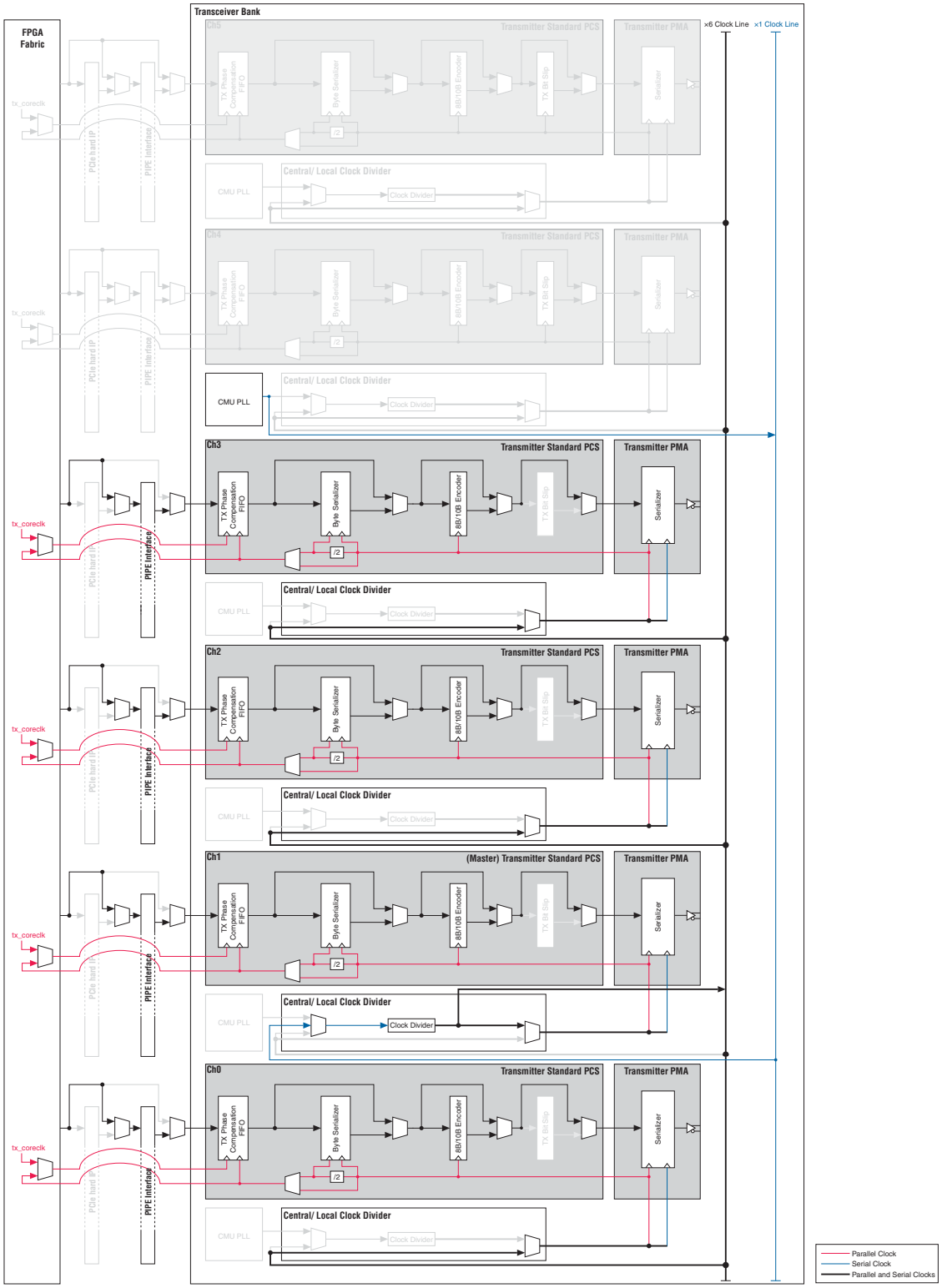
Figure 4–8. Transceiver Clocking Configuration in a PIPE x1 Configuration



PIPE x4 Configuration

Figure 4–9 shows the transmitter clocking for a PIPE x4 bonded configuration. Clocking is independent for the receiver channels. The clocking and control signals are bonded only for the transmitter channels.

Figure 4–9. Transceiver Clocking Configuration in a PIPE x4 Configuration (1)



Note to Figure 4–9:

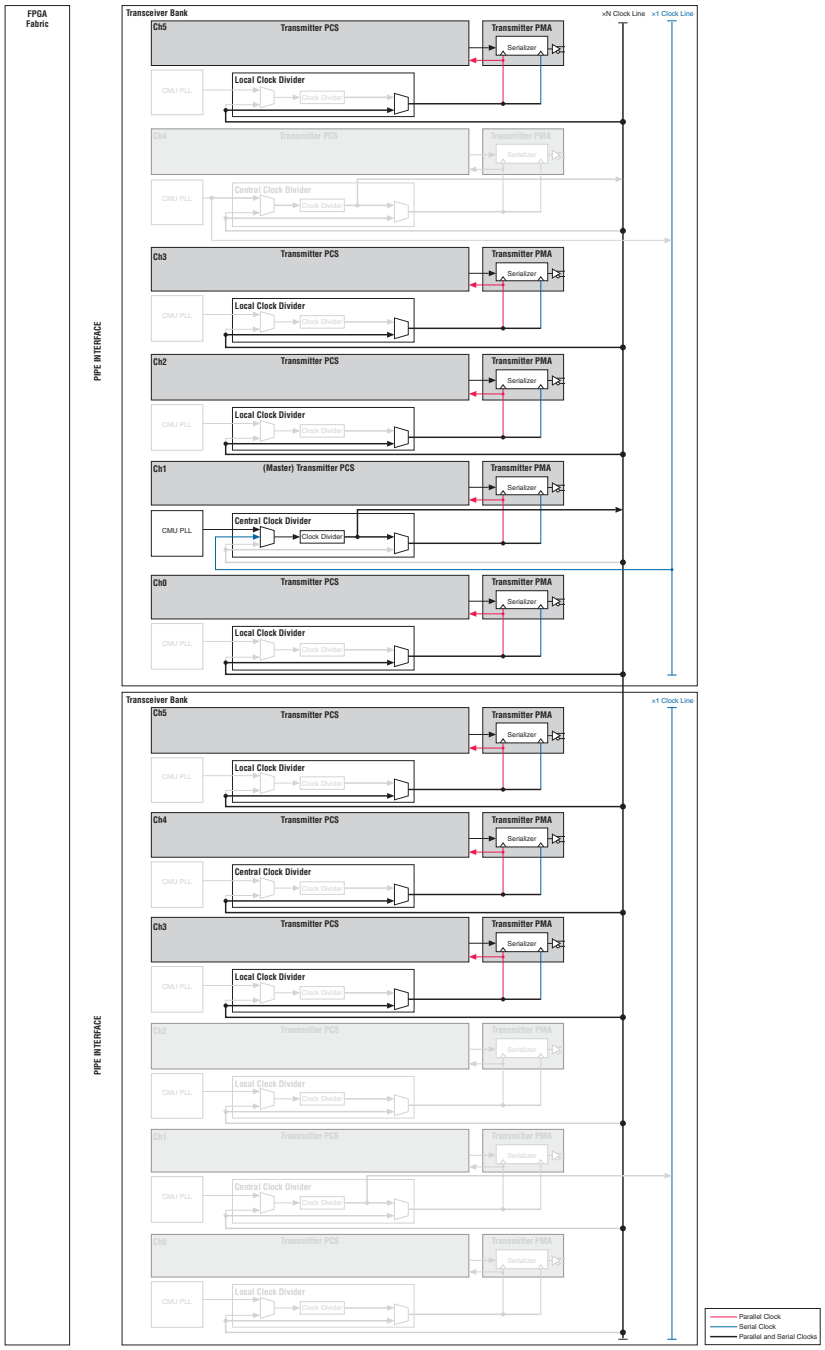
(1) The x2 channel placement for the clock generator (CMU PLL) and master channel is the same as the x4 channel placement. The other channel for the x2 configuration can be placed in either of the channels shaded in gray.

PIPE x8 Configuration

Figure 4–10 shows the clocking for the PMA and PCS blocks in the PIPE x8 bonded configuration. The clocking is independent for the receiver channels. The clocking and control signals are bonded only for the transmitter channels.

For more information about clocking in Arria V devices, refer to the *Transceiver Clocking in Arria V Devices* chapter.

Figure 4–10. Transceiver Clocking Configuration in a PIPE x8 Configuration



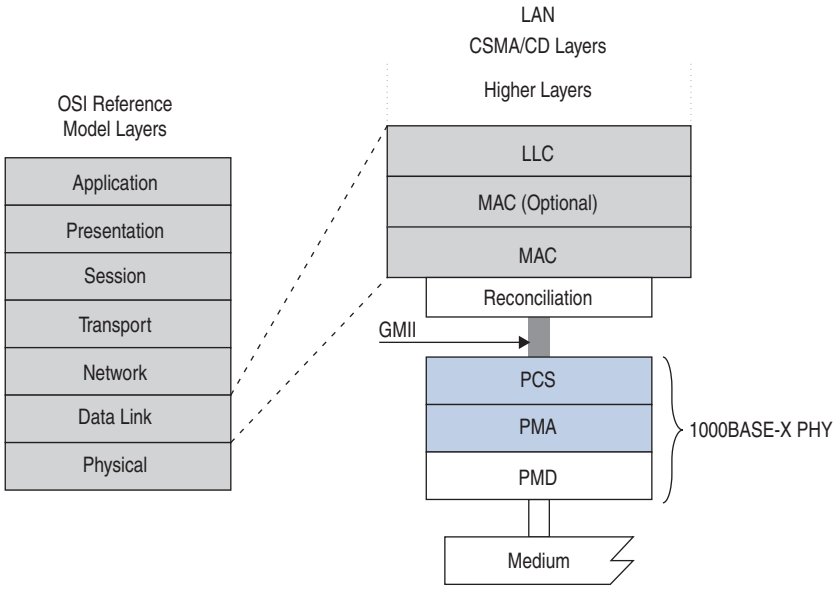
For more information about PCI Express PHY IP implementation, refer to the *"PCI Express PHY IP Core"* chapter in the *Altera Transceiver PHY IP Core User Guide*.

Gigabit Ethernet

The IEEE 802.3 specification defines the 1000BASE-X PHY as an intermediate, or transition layer that interfaces various physical media with the MAC in a gigabit ethernet (GbE) system, shielding the MAC layer from the specific nature of the underlying medium. The 1000BASE-X PHY is divided into the PCS, PMA, and PMD sublayers.

The PCS sublayer interfaces with the MAC through the gigabit media independent interface (GMII). The 1000BASE-X PHY defines a physical interface data rate of 1 Gbps. Figure 4–11 shows the 1000BASE-X PHY position in a GbE Open Systems Interconnection (OSI) reference model.

Figure 4–11. 1000BASE-X PHY in a GbE OSI Reference Model



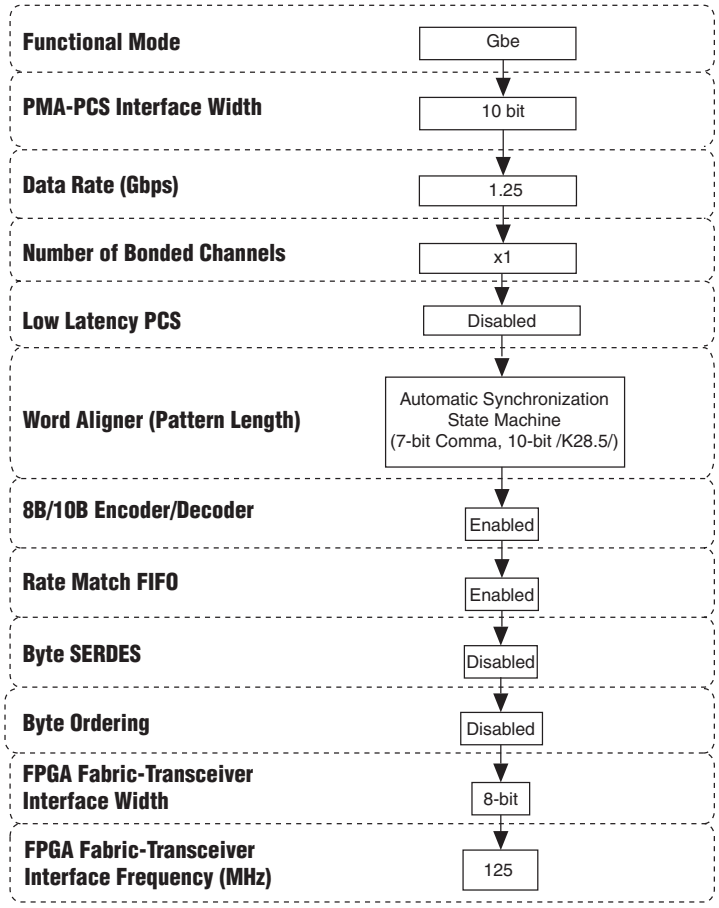
Arria V transceivers, when configured in GbE functional mode, have built-in circuitry to support the following PCS and PMA functions, as defined in the IEEE 802.3 specification:

- 8B/10B encoding and decoding (excluding PMD implementation)
- Synchronization
- Upstream transmitter and local receiver clock frequency compensation (rate matching)
- Clock recovery from the encoded data forwarded by the receiver PMD
- Serialization and deserialization

Arria V transceivers do not have built-in support for other PCS functions, such as the autonegotiation state machine, collision-detect, and carrier-sense functions. If you require these functions, implement them in the FPGA fabric or in external circuits.

Figure 4–12 shows the transceiver blocks that are enabled in a GbE configuration.

Figure 4–12. Arria V GX GbE Configuration



Transceiver Datapath

Figure 4–13 shows the transceiver datapath when configured in the GbE functional mode.

Figure 4–13. GbE Mode Datapath

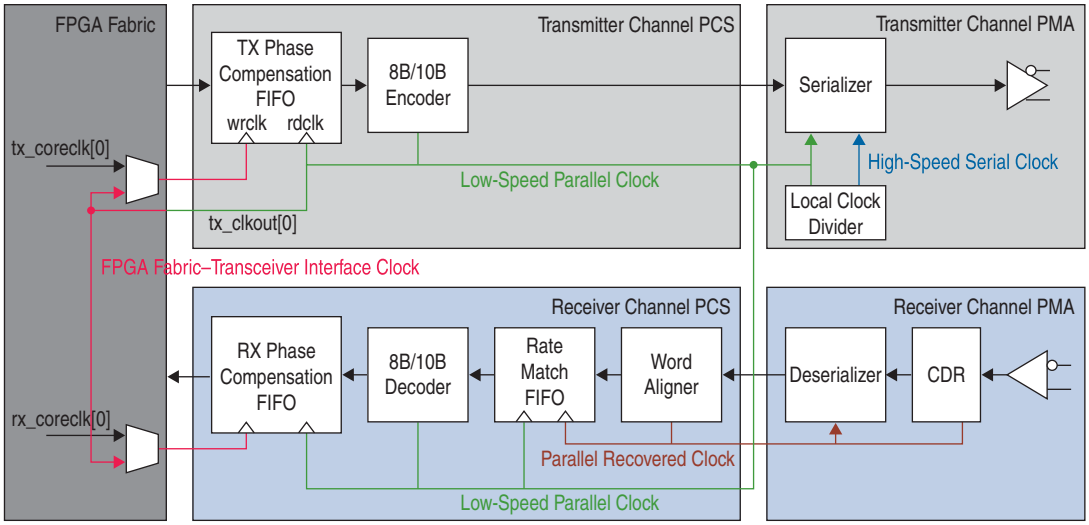


Table 4–3 lists the transceiver datapath clock frequencies in GbE functional mode.

Table 4–3. Transceiver Datapath Clock Frequencies in GbE Mode

Functional Mode	Data Rate	High-Speed Serial Clock Frequency	Parallel Recovered Clock and Low-Speed Parallel Clock Frequency	FPGA Fabric–Transceiver Interface Clock Frequency
GbE	1.25 Gbps	625 MHz	125 MHz	125 MHz

8B/10B Encoder

In GbE mode, the 8B/10B encoder clocks in 8-bit data and 1-bit control identifiers from the transmitter phase compensation FIFO and generates 10-bit encoded data. The 10-bit encoded data is fed to the serializer.

For more information about the 8B/10B encoder functionality, refer to the *Transceiver Architecture for Arria V Devices* chapter.

Rate Match FIFO

In GbE mode, the rate match FIFO is capable of compensating for up to ±100 ppm (200 ppm total) difference between the upstream transmitter and the local receiver reference clock. The GbE protocol requires that the transmitter send idle ordered sets /I1/ (/K28.5/D5.6/) and /I2/ (/K28.5/D16.2/) during interpacket gaps, adhering to the rules listed in the IEEE 802.3 specification.

The rate match operation begins after the synchronization state machine in the word aligner indicates that the synchronization is acquired—by driving the rx_syncstatus signal high. The rate matcher deletes or inserts both symbols (/K28.5/ and /D16.2/) of the /I2/ ordered sets, even if it requires deleting only one symbol to prevent the rate match FIFO from overflowing or underrunning. The rate matcher can insert or delete as many /I2/ ordered sets as necessary to perform the rate match operation.

Two flags are forwarded to the FPGA fabric:

- rx_rmfifodatadeleted—Asserted for two clock cycles for each deleted /I2/ ordered set to indicate the rate match FIFO deletion event
- rx_rmfifodatainserted—Asserted for two clock cycles for each inserted /I2/ ordered set to indicate the rate match FIFO insertion event

For more information about the rate match FIFO, refer to the *Transceiver Architecture for Arria V Devices* chapter.

GbE Protocol-Ordered Sets and Special Code Groups

Table 4–4 lists the ordered sets and special code groups, as specified in the IEEE 802.3-2008 specification.

Table 4–4. GIGE Ordered Sets

Code	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)
/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 4–4:

(1) Two data code groups representing the Config_Reg value.

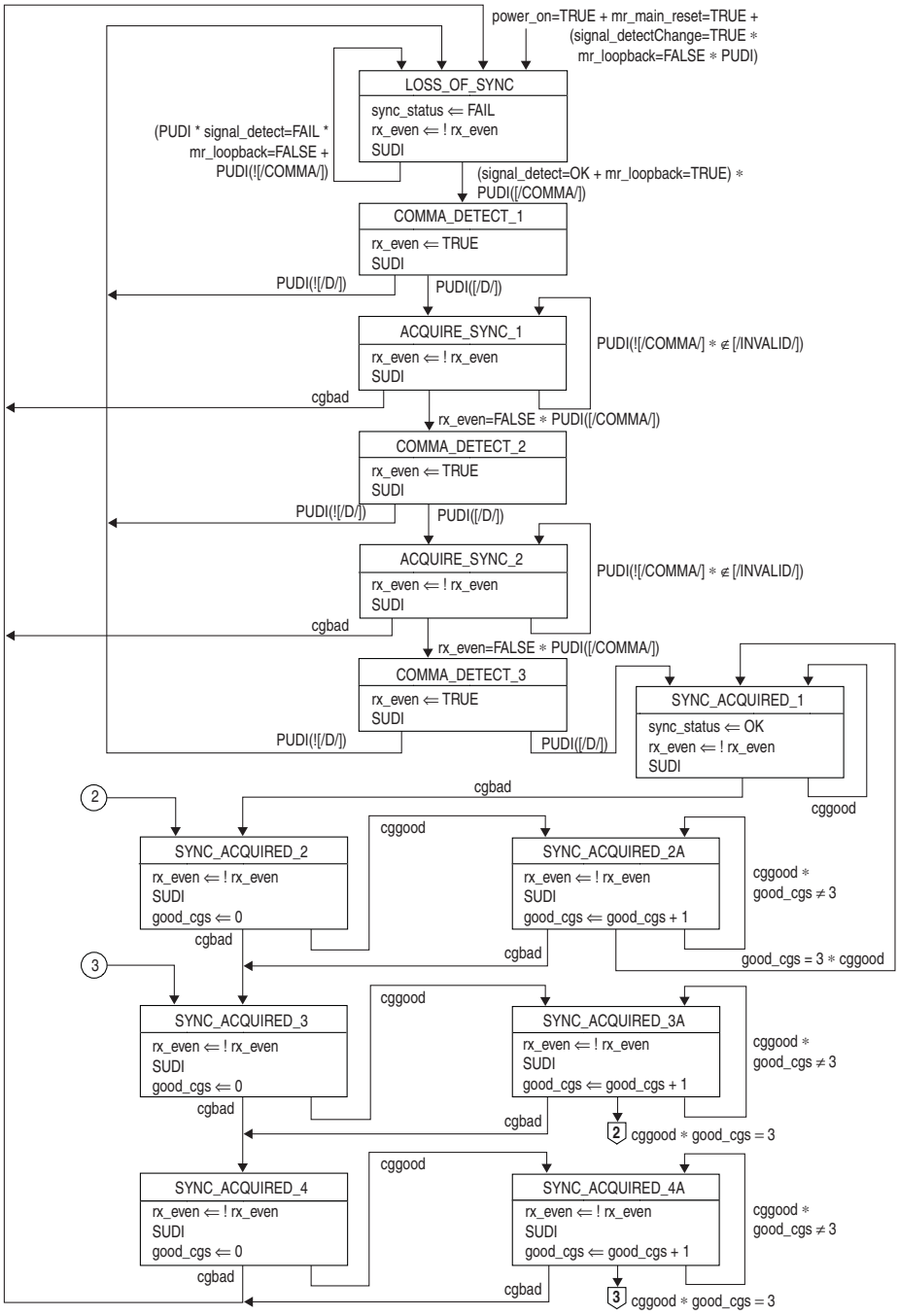
Table 4–5 lists the synchronization state machine parameters in GbE functional mode.

Table 4–5. Synchronization State Machine Parameters in GbE Mode

Synchronization State Machine Parameters	Setting
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 4–14 shows the synchronization state machine implemented in GbE mode.

Figure 4–14. Synchronization State Machine in GbE Mode *(1)*



Note to Figure 4–14:

(1) This figure is from "Figure 36–9" in the IEEE 802.3-2008 specification. For more details about the 1000BASE-X implementation, refer to Clause 36 of the IEEE 802.3-2008 specification.

For more information about Gigabit Ethernet implementation in a Custom PHY IP, refer to the *Custom PHY IP Core* chapter in the *Altera Transceiver PHY IP Core User Guide*.

Serial Digital Interface

The Society of Motion Picture and Television Engineers (SMPTE) defines various Serial Digital Interface (SDI) standards for transmission of uncompressed video.

The following SMPTE standards are popular in video broadcasting applications:

- SMPTE 259M standard—more popularly known as the standard-definition (SD) SDI; defined to carry video data at 270 Mbps
- SMPTE 292M standard—more popularly known as the high-definition (HD) SDI; 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; defined to carry video data at either 2970 Mbps or 2967 Mbps

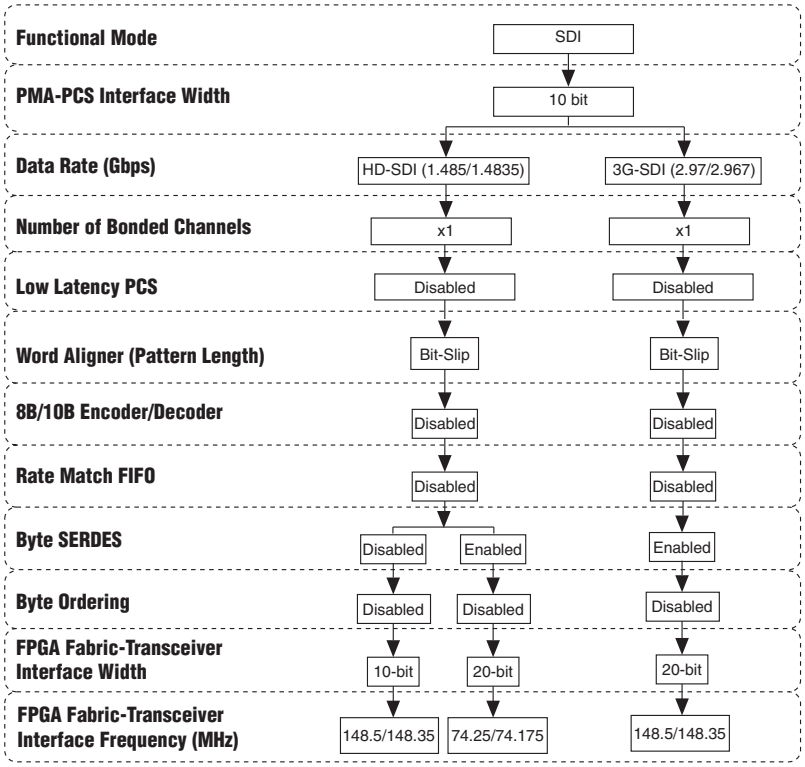
Table 4–6 lists the configurations supported by Arria V transceivers in SDI mode.

Table 4–6. Configurations in SDI Mode

Configuration	Data Rate (Mbps)	REFCLK Frequencies (MHz)	FPGA Fabric-Transceiver Interface Width
HD	1485	74.25, 148.5	10 bit and 20 bit
	1483.5	74.175, 148.35	10 bit and 20 bit
3G	2970	148.5, 297	Only 20-bit interfaces allowed in 3G
	2967	148.35, 296.7	Only 20-bit interfaces allowed in 3G

Figure 4–15 shows SDI mode configurations supported in Arria V devices.

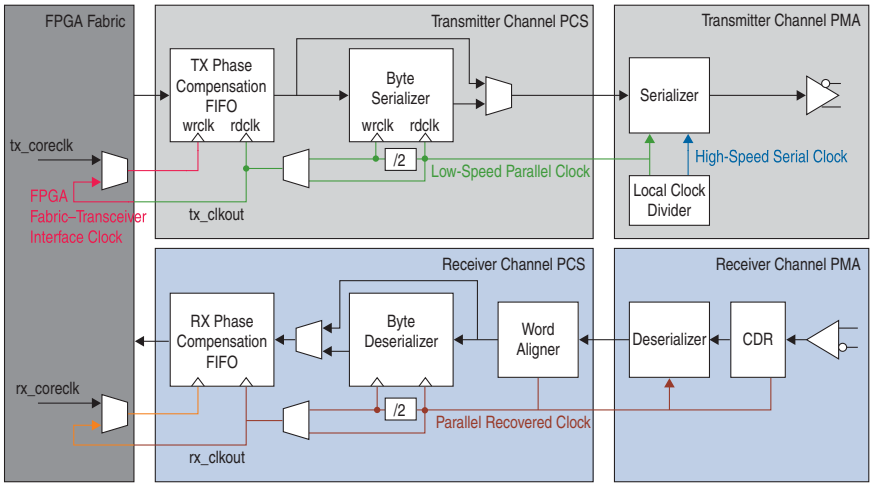
Figure 4–15. SDI Mode for Arria V Configurations



Transceiver Datapath

Figure 4–16 shows the transceiver datapath when configured in SDI mode.

Figure 4–16. SDI Mode Datapath



Transmitter Datapath

The transmitter datapath in the HD-SDI configuration with a 10-bit wide FPGA fabric–transceiver interface consists of the transmitter phase compensation FIFO and the 10:1 serializer. In HD-SDI and 3G-SDI configurations with 20-bit wide FPGA fabric–transceiver interface, the transmitter datapath also includes the byte serializer.

In SDI mode, the transmitter is purely a parallel-to-serial converter. You must implement the SDI transmitter functions, such as the scrambling and cyclic redundancy check (CRC) code generation, in the FPGA logic array.

Receiver Datapath

In the 10-bit channel width SDI configuration, the receiver datapath consists of the clock recovery unit (CRU), 1:10 deserializer, word aligner in bit-slip mode, and receiver phase compensation FIFO. In the 20-bit channel width SDI configuration, the receiver datapath also includes the byte deserializer.

You must implement the SDI receiver functions, such as descrambling, framing, and CRC checker, in the FPGA logic array.

Receiver Word Alignment and Framing

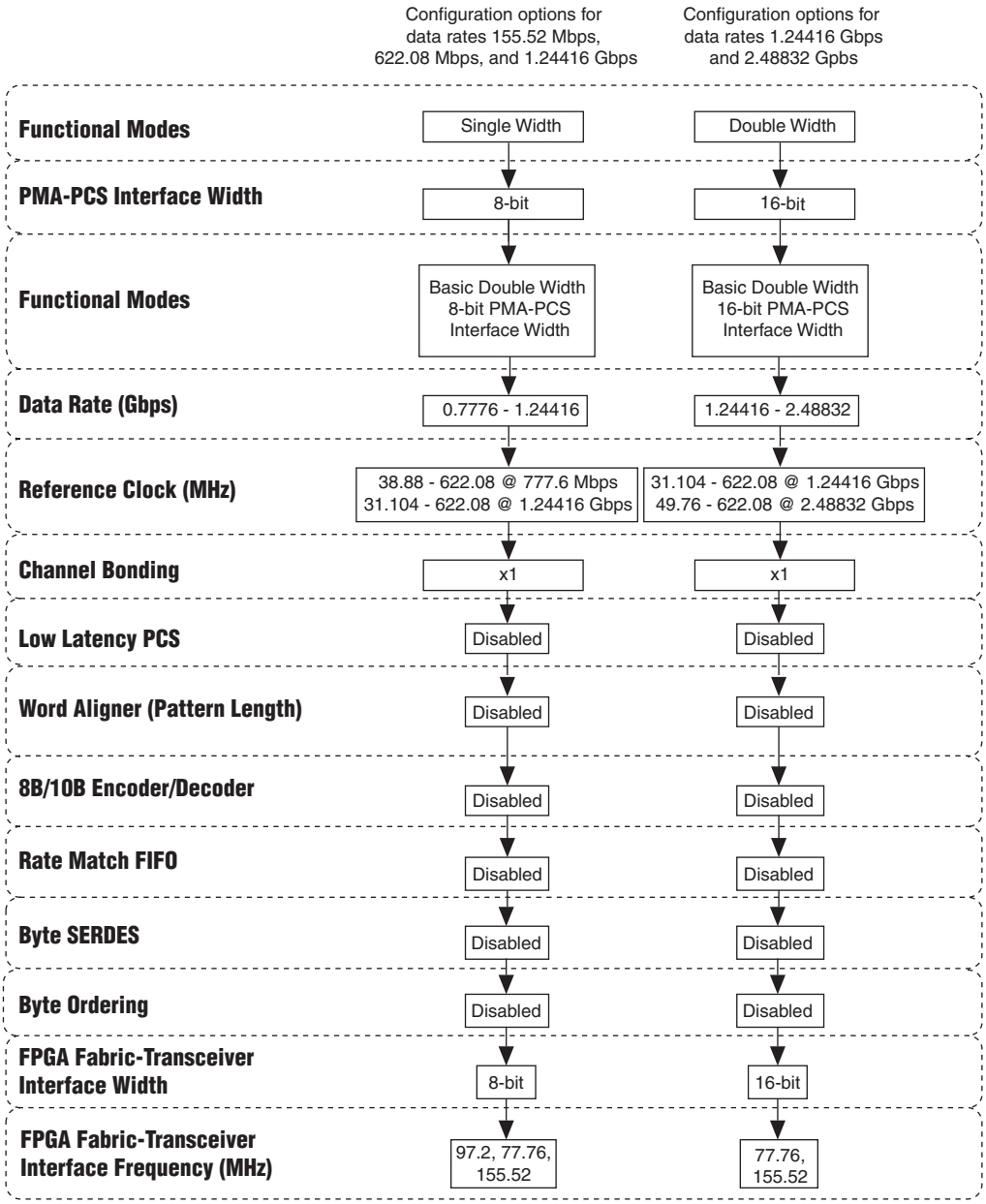
In SDI systems, the word aligner in the receiver datapath is not useful because the word alignment and framing happen after descrambling. Altera recommends that you drive the `rx_bitslip` of the PHY MegaWizard™ signal low to avoid having the word aligner insert bits in the received data stream.

Gigabit-Capable Passive Optical Network (GPON)

The GPON protocol network provides optical fiber cabling and signals to the home and office using a point-to-multipoint scheme. Arria V devices support GPON data rates of 155.52 Mbps, 622.08 Mbps, 1.24416 Gbps, and 2.48832 Gbps, with a reference clock of 155.52 MHz. The minimum supported data rate of Arria V device is 600 Mbps, so a 5x over-sampling factor is used for the GPON data rate of 155.52 Mbps, resulting in a data rate of 777.6 Mbps.

Figure 4–17 shows the recommended configurations for implementing the GPON protocol in Arria V devices.

Figure 4–17. GPON for Arria V Configurations

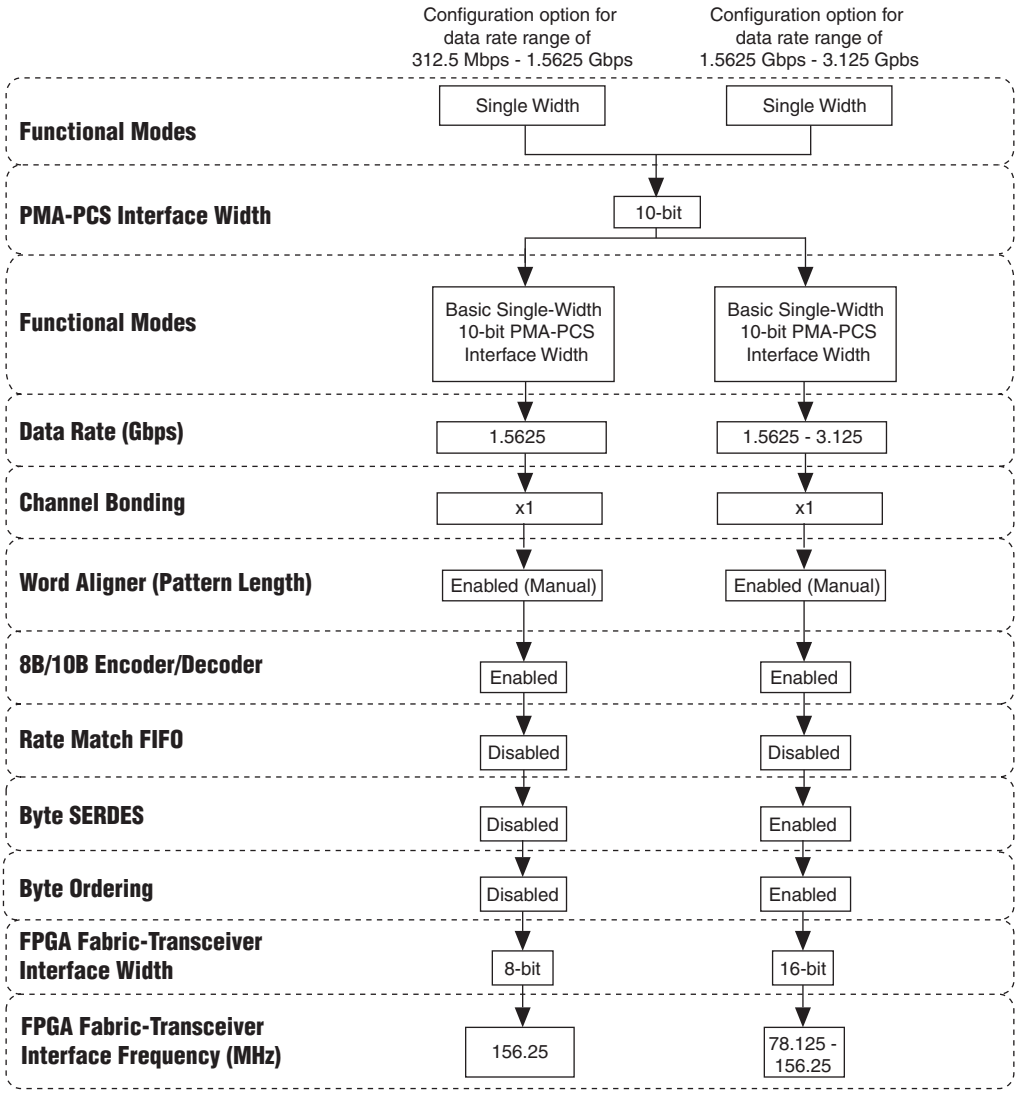


Serial Data Converter (SDC) JESD204

The SDC (JESD204) protocol conforms to JESD204, a JEDEC standard that enables a high-speed serial connection between analog-to-digital converters and logic devices using only a two-wire high-speed serial interface. Arria V devices support an SDC (JESD204) data rate range of 312.5 Mbps to 3.125 Gbps. The minimum supported data rate in Arria V devices is 611 Mbps, so a 5x over-sampling factor is used for the SDC (JESD204) data rate of 312.5 Mbps, resulting in a data rate of 1.5625 Gbps.

Figure 4–18 shows the recommended configurations for implementing the SDC protocol in Arria V devices.

Figure 4–18. SDC (JESD204) for Arria V Configurations



SATA and SAS Protocols

Serial ATA (SATA) and Serial Attached SCSI (SAS) are data storage protocol standards that have the primary function of transferring data (directly or otherwise) between the host system and mass storage devices, such as hard disk drives, optical drives, and solid-state disks. These serial storage protocols offer several advantages over older parallel storage protocol (ATA and SCSI) interfaces:

- Faster data transfer
- Hot swapping (when supported by the operating system)
- Thinner cables for more efficient air cooling
- Increased operation reliability

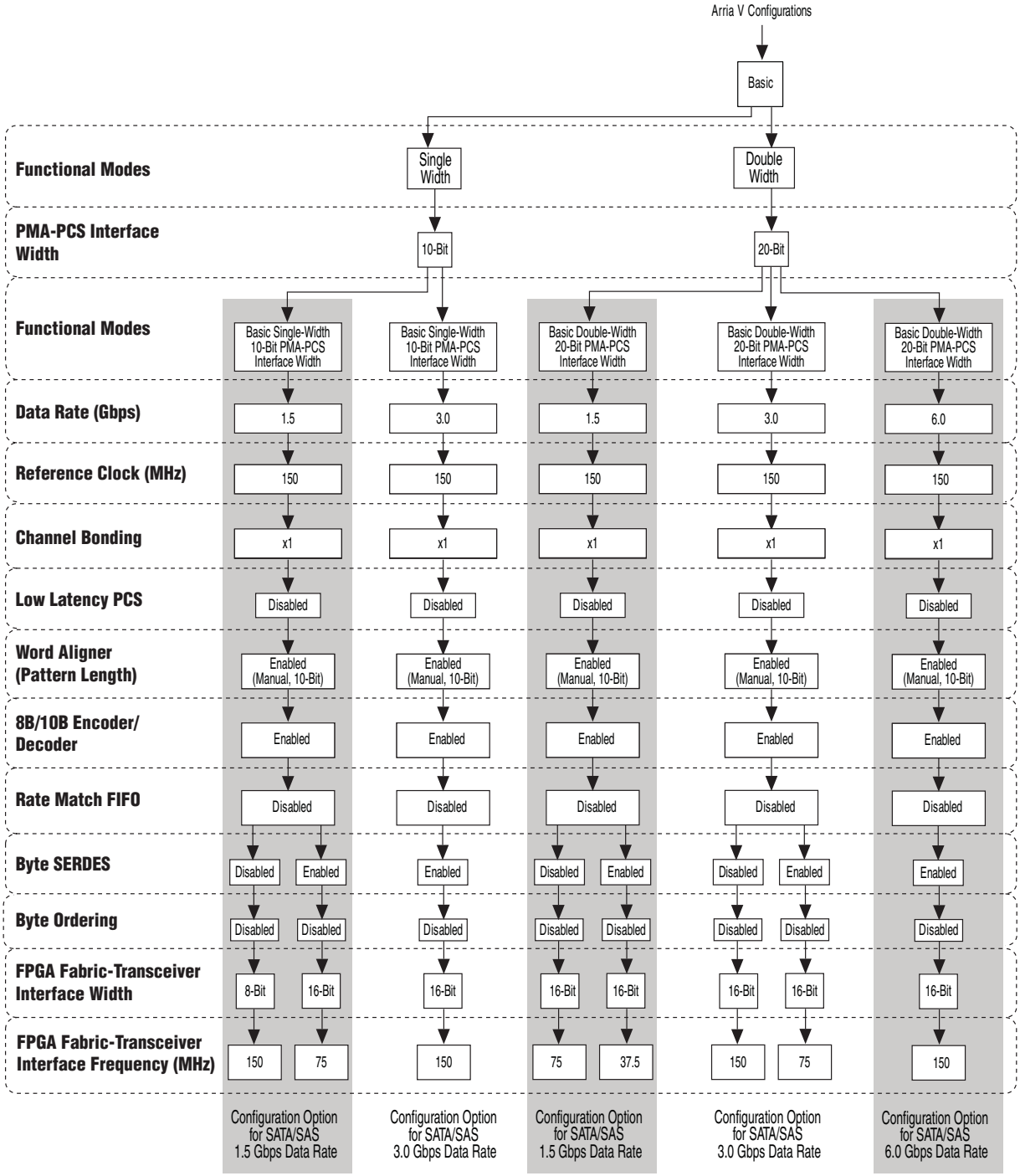
Table 4–7 lists the serial data rates supported by Arria V devices.

Table 4–7. Serial Data Rates for SATA and SAS Protocols

Protocol	SATA (Gbps)	SAS (Gbps)
Gen1	1.5	3.0
Gen2	3.0	6.0
Gen3	6.0	—

Figure 4–19 shows the recommended configurations for implementing the SATA and SAS protocols in Arria V devices.

Figure 4–19. SATA and SAS for Arria V Configurations

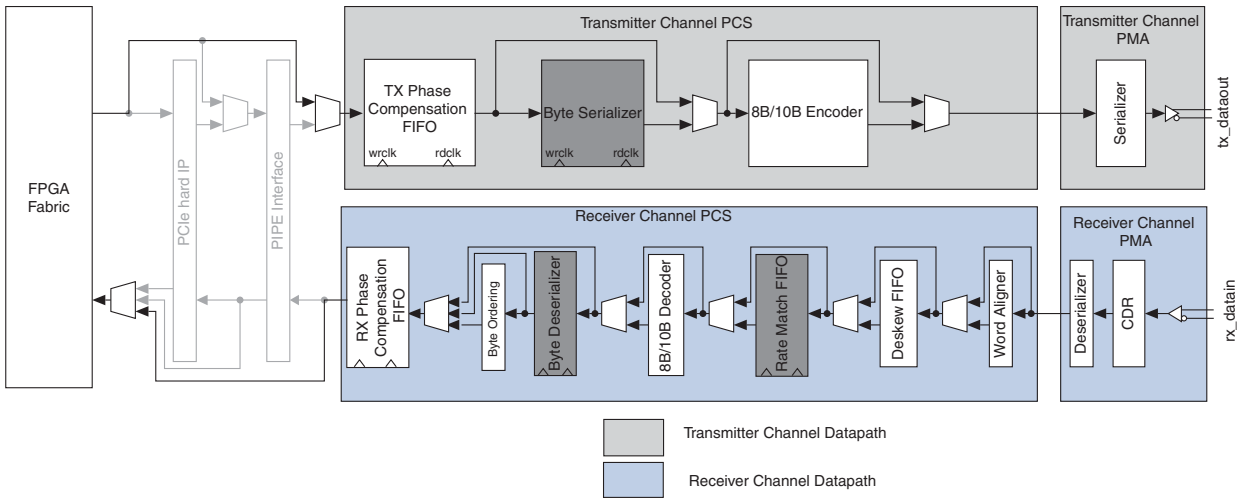


Deterministic Latency Protocols—CPRI and OBSAI

Arria V devices have a deterministic latency option available for use in high-speed serial interfaces such as the Common Public Radio Interface (CPRI) and OBSAI Reference Point 3 (OBSAI RP3). Both CPRI and OBSAI RP3 protocols place stringent requirements on the amount of latency variation that is permissible through a link that implements these protocols. Arria V GT devices also support 9.8304 Gbps CPRI with PMA direct configuration; PCS is implemented in soft logic.

Figure 4–20 shows the transceiver datapath when using deterministic latency mode.

Figure 4–20. Transceiver Datapath in Deterministic Latency Mode



Receiver Phase Compensation FIFO in Register Mode

To remove the latency uncertainty through the receiver's phase compensation FIFO, the receiver and transmitter phase compensation FIFOs are always set to register mode. In register mode, the phase compensation FIFO acts as a register and thereby removes the uncertainty in latency. The latency through the phase compensation FIFO in register mode is one clock cycle.

The following options are available:

- Single-width mode with 8-bit channel width and 8B/10B encoder enabled or 10-bit channel width with 8B/10B disabled
- Double-width mode with 16-bit channel width and 8B/10B encoder enabled or 20-bit channel width with 8B/10B disabled

Transmitter Phase Compensation FIFO in Register Mode

In register mode, the phase compensation FIFO acts as a register and thereby removes the uncertainty in latency. The latency through the transmitter and receiver phase compensation FIFO in register mode is one clock cycle.

Channel PLL Feedback

To implement the deterministic latency functional mode, the phase relationship between the low-speed parallel clock and channel PLL input reference clock must be deterministic. A feedback path is enabled to ensure a deterministic relationship between the low-speed parallel clock and channel PLL input reference clock.

To achieve deterministic latency through the transceiver, the reference clock to the channel PLL must be the same as the low-speed parallel clock. For example, if you need to implement a data rate of 1.2288 Gbps for the CPRI protocol, which places stringent requirements on the amount of latency variation, you must choose a reference clock of 122.88 MHz to allow the usage of a feedback path from the channel PLL. This feedback path reduces the variations in latency.

When you select this option, provide an input reference clock to the channel PLL that is of the same frequency as the low-speed parallel clock.

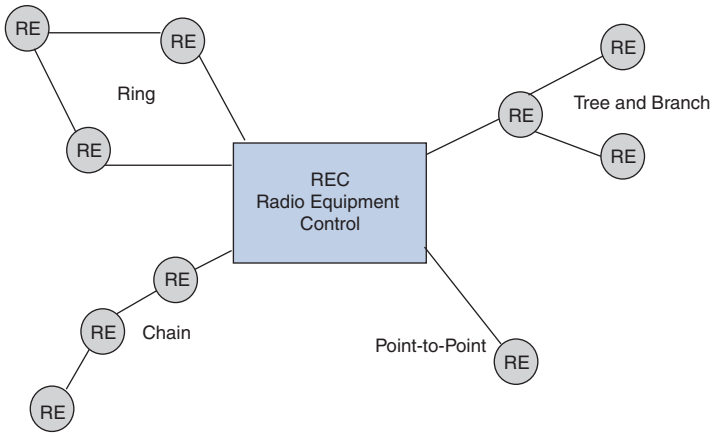
CPRI and OBSAI

You can use the deterministic latency functional mode to implement protocols such as CPRI and OBSAI.

The CPRI interface defines a digital point-to-point interface between the Radio Equipment Control (REC) and the Radio Equipment (RE), allowing flexibility in either co-locating the REC and the RE, or a remote location of the RE.

Figure 4–21 shows various CPRI topologies. In most cases, CPRI links are between REC and RE modules or between two RE modules in a chain configuration.

Figure 4–21. CPRI Topologies



If the destination for the high-speed serial data that leaves the REC is the first RE, it is a single-hop connection. If the serial data from the REC must traverse through multiple REs before reaching the destination RE, it is a multi-hop connection.

Remotely locating the RF transceiver from the main base station introduces a complexity with overall system delay. The CPRI specification requires that the accuracy of measurement of roundtrip delay on single-hop and multi-hop connections be within ±16.276 ns to properly estimate the cable delay.

For a single-hop system, this allows a variation in roundtrip delay of up to ±16.276 ns. However, for multi-hop systems, the allowed delay variation is divided among the number of hops in the connection—typically, equal to ±16.276 ns/(the number of hops) but not always equally divided among the hops.

Deterministic latency on a CPRI link also enables highly accurate triangulation of the location of the caller.

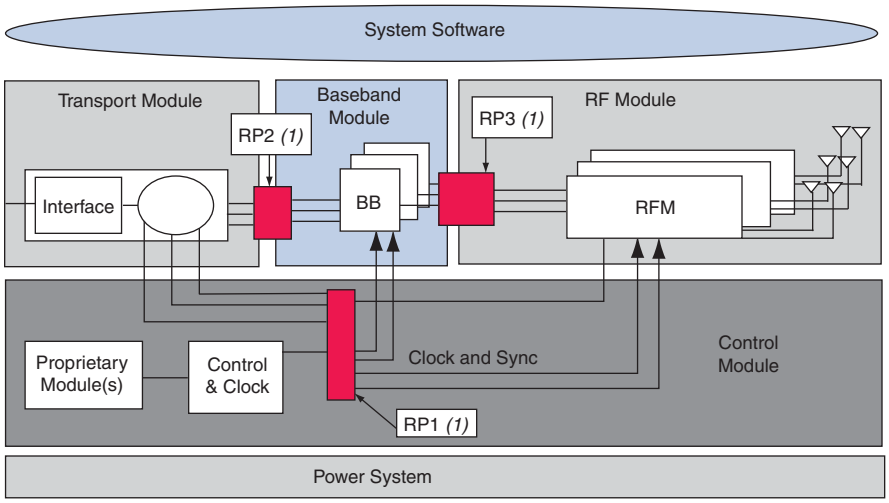
OBSAI was established by several OEMs to develop a set of specifications that can be used for configuring and connecting common modules into base transceiver stations (BTS).

The BTS has four main modules:

- Radio frequency (RF)
- Baseband
- Control
- Transport

Figure 4–22 shows a typical BTS. The radio frequency module (RFM) receives signals using portable devices and converts the signals to digital data. The baseband module processes the encoded signal and brings it back to the baseband before transmitting it to the terrestrial network using the transport module. A control module maintains the coordination between these three functions.

Figure 4–22. Example of the OBSAI BTS Architecture



Note to Figure 4–22:

(1) "RP" means Reference Point.

Using the deterministic latency option, you can implement the CPRI data rates in the following modes:

- Single-width mode—with 8/10-bit channel width
- Double-width mode—with 16/20-bit channel width

Table 4–8 lists sample channel width options, and CPRI and OBSAI data rates in the deterministic latency mode.

Table 4–8. Sample Channel Width Options for Supported Serial Data Rates

Serial Data Rate (Mbps)	Channel Width (FPGA-PCS Fabric)			
	Single Width		Double-Width	
	8-Bit	16-Bit	16-Bit	32-Bit
614.4	Yes	Yes	No	No
1228.8	Yes	Yes	Yes	Yes
2457.6	No	Yes	Yes	Yes
3072	No	Yes (1)	Yes (1)	Yes
4915.2	No	No	No	Yes (1)
6144	No	No	No	Yes (1)
9830.4	N/A (2)	N/A (2)	N/A (2)	N/A (2)

Notes to Table 4–8:

(1) Applicable to C4, C5, and I5 speed grades only.

(2) CPRI 9830.4 Mbps uses PMA direct mode with 80-bits PMA-PLD data width.

CPRI Enhancements in Arria V Devices

Enhancements in Arria V transceivers over Stratix IV transceivers support the deterministic latency requirements in the CPRI specification. The deterministic latency state machine in the word aligner reduces the known delay variation from the word alignment process and automatically synchronizes and aligns the word boundary by slipping a clock cycle in the deserializer. Incoming data to the word aligner is aligned to the boundary of the word alignment pattern (K28.5). User logic is not required to manipulate the TX bit slipper for constant round-trip delay. In manual mode, the TX bit slipper is able to compensate one unit interval (UI). Figure 4–23 shows the deterministic latency state machine.

The word alignment pattern (K28.5) position varies in byte deserialized data. Delay variation is up to ½ parallel clock cycle. You must add in extra user logic to manually check the K28.5 position in byte deserialized data for the actual latency.

Figure 4–23. Deterministic Latency State Machine in the Word Aligner

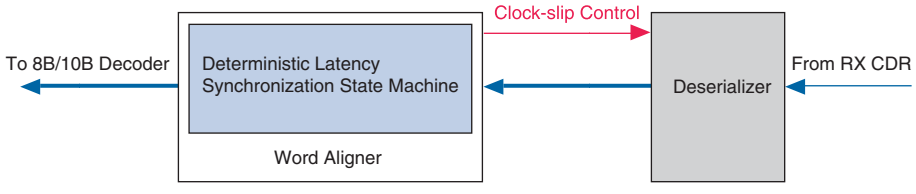


Table 4–9 lists the existing and new methods to achieve the deterministic latency mode in Arria V devices.

Table 4–9. Methods to Achieve Deterministic Latency Mode in Arria V Devices

Existing Feature (1)		Enhanced Feature (2)	
Description	**Requirement**	**Description**	**Requirement**
Manual alignment with bit position indicator provides deterministic latency. Delay variation up to 1 parallel clock cycle.	Extra user logic to manipulate the TX bit slipper with a bit position indicator from the word aligner for constant total round-trip delay.	Deterministic latency state machine alignment reduces the known delay variation in word alignment operation.	None.

Notes to Table 4–9:

(1) Backward compatible with CPRI designs in Arria II devices.

(2) Enhanced deterministic latency feature in Arria V devices.

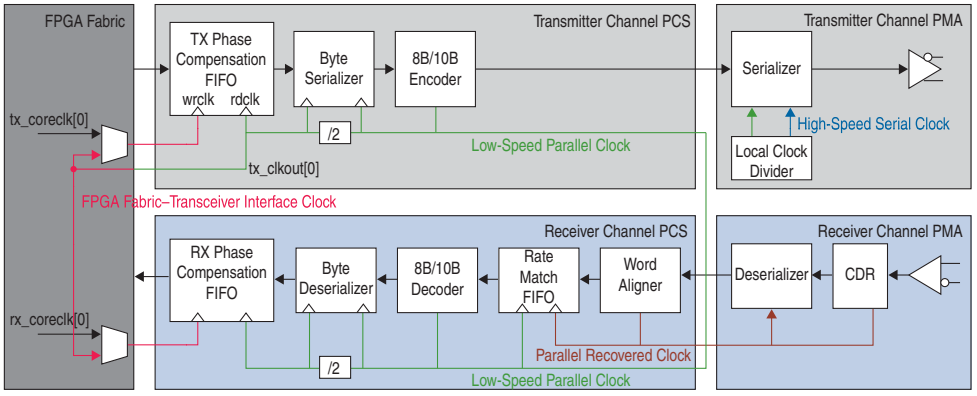
For more information about deterministic latency PHY IP implementation, refer to the *Deterministic Latency PHY IP Core* chapter in the *Altera Transceiver PHY IP Core User Guide*.

Serial RapidIO

The RapidIO Trade Association defines a high-performance, packet-switched interconnect standard to pass data and control information between microprocessors, digital signal, communications, and network processors, system memories, and peripheral devices.

Figure 4–24 shows the transceiver datapath when configured in the Serial RapidIO (SRIO) mode.

Figure 4–24. Serial RapidIO Mode Datapath



Arria V transceivers support SRIO physical layer specifications, versions 1.3 and 2.1, from 1.25 Gbps to 6.25 Gbps. The transceivers are compliant with the x2 and x4 channel bonding, deskew state machine, and rate match FIFO.

Synchronization State Machine

The word aligner has a synchronization state machine that handles the receiver lane synchronization.

The synchronization state machine indicates synchronization when the receiver receives 127 K28.5 (10'b0101111100 or 10'b1010000011) synchronization code groups without receiving an intermediate invalid code group. After synchronization, the state machine indicates loss of synchronization when it detects three invalid code groups separated by less than 255 valid code groups, or when it is reset.

The `rx_syncstatus` port of each channel indicates the receiver synchronization:

- High—the lane is synchronized
- Low—the lane has fallen out of synchronization

Table 4–10 lists the synchronization state machine parameters when configured in SRIO mode.

Table 4–10. Synchronization State Machine Parameters in Serial RapidIO Mode

Parameters	Number
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 one	255

Rate Match FIFO

In SRIO mode, the rate match FIFO is capable of compensating for up to ±100 ppm (200 ppm total) difference between the upstream transmitter and the local receiver reference clock.

The rate match FIFO operation begins after the word aligner synchronization status, `rx_syncstatus`, goes high. When the rate matcher receives either of the two 10-bit control patterns followed by the respective 10-bit skip pattern, it inserts or deletes the 10-bit skip pattern as necessary to avoid the rate match FIFO from overflowing or under-running.

In SRIO mode, the rate match FIFO can delete or insert a maximum of one skip pattern from a cluster.

For more details, refer to "Chapter 4 PCS and PMA Layers" of *Part 6: LP-Serial Physical Layer Specification* in the *RapidIO Interconnect Specification*.

Document Revision History

Table 4–11 lists the revision history for this chapter.

Table 4–11. Document Revision History

Date	Version	Changes
June 2012	1.2	■ Updated for the Quartus II software version 12.0. ■ Added the "Serial Digital Interface" section. ■ Added the "Gigabit-Capable Passive Optical Network (GPON)" section. ■ Added the "Serial Data Converter (SDC) JESD204" section. ■ Added the "SATA and SAS Protocols" section. ■ Updated Figure 4–2 and Figure 4–18. ■ Added Figure 4–19. ■ Updated Table 4–1, Table 4–8, and Table 4–9. ■ Updated the "CPRI Enhancements in Arria V Devices" section. ■ Added the "Serial RapidIO" section.
November 2011	1.1	Updated for the Quartus II software version 11.1.
August 2011	1.0	Initial release.

5. Transceiver Custom Configurations in Arria V Devices

AV53005-1.2

This chapter describes the custom transceiver datapath configuration for customized and proprietary high-speed serial interface (HSSI) implementations in Arria® V devices.

For integration with the FPGA fabric, the full-duplex transceiver channel supports custom configuration with physical medium attachment (PMA), physical coding sublayer (PCS), and low latency custom configurations with PMA and low latency PCS.

The 10-Gbps transceiver channel in Arria V GT devices supports PMA Direct configurations enabling data rates above 6.5536 gigabits per second (Gbps).

This chapter contains the following sections:

PCS Datapath Latency

Table 5–1 compares the latency incurred in each available PCS block between custom and low latency custom configurations.

Table 5–1. PCS Datapath Latency (Part 1 of 2) (1)

Datapath	PCS Block	Custom Configuration		Low Latency Custom Configuration	
		Enabled	Disabled	Enabled	Disabled
Transmitter (TX)	TX Phase Compensation FIFO	3 – 4	1 (1)	3 – 4	—
	Byte Serializer	1 – 2	1	1 – 2	0
	8B/10B Encoder	1	1	—	1
	Transmitter Bit-Slip	0 – 1	0	—	1

Table 5–1. PCS Datapath Latency (Part 2 of 2) (1)

Datapath	PCS Block	Custom Configuration		Low Latency Custom Configuration	
		Enabled	Disabled	Enabled	Disabled
Receiver (RX)	Word Aligner	4 – 8	—	—	1
	Rate Match FIFO	11 – 14	0	—	0
	8B/10B Decoder	1	1	—	0
	Byte Deserializer	1 – 2	1	1 – 2	0
	Byte Ordering	1 – 3	1	—	0
	RX Phase Compensation FIFO	2 – 3	1 (1)	2 – 3	—

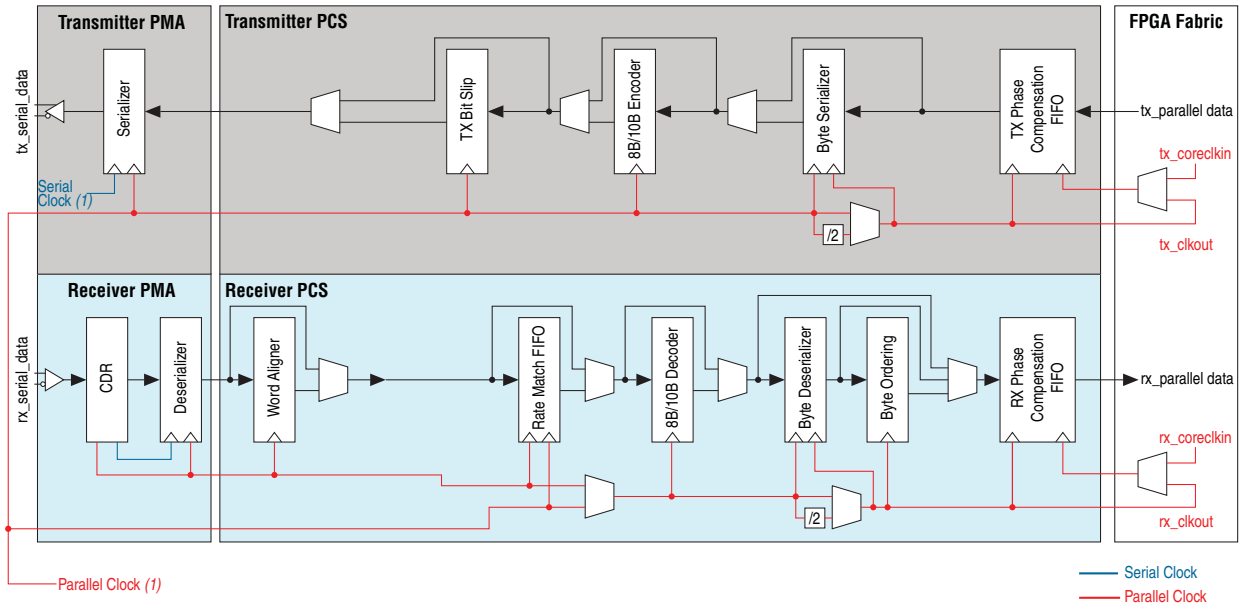
Note to Table 5–1:

(1) The registered mode FIFO is supported in deterministic latency configurations only.

Custom Configuration

In a custom configuration, you can customize the transceiver channel to include a PMA and PCS with functions that your application requires. The transceiver channel interfaces with the FPGA fabric through the PCS. Based on your application requirements, you can enable, modify, or disable the blocks, except the deskew FIFO block, as shown in Figure 5–1.

Figure 5–1. Complete Datapath in a Custom Configuration



Note to Figure 5–1:

(1) The serial and parallel clocks are sourced from the clock divider.

The maximum supported data rate varies depending on the customization. Table 5–2 lists an example of the maximum supported data rate in various custom configurations with the PMA and PCS using the fastest speed grade device.

Table 5–2. Maximum Supported Data Rate for Fastest Speed Grade Device (–4 Commercial) in Custom Configuration

Datapath Configuration	PMA–PCS Interface Width	PCS–FPGA Fabric Interface Width (1)		Maximum Data Rate (Mbps) (2), (3)
		8B/10B Enabled	8B/10B Disabled	
Single-width	8	—	8, 16	2500
	10	8, 16	10, 20	3125
Double-width	16	—	16, 32	5242.88
	20	16, 32	20, 40	6553.6

Notes to Table 5–2:

(1) The supported interface width varies depending on the usage of the byte serializer/deserializer (SERDES), and the 8B/10B encoder or decoder.

(2) The byte serializer or deserializer is assumed to be enabled. Otherwise, the maximum data rate supported is half of the specified value.

(3) Data rates over 6553.6 Mbps are supported in PMA Direct mode. Refer to "PMA Direct Configuration" on page 5–13 for more information.

In all the supported configuration options of the channel, the transmitter bit-slip function is optional, as shown in Figure 5–2 through Figure 5–5.

Figure 5–2. Configuration Options for Custom Single-Width Mode (8-bit PMA–PCS Interface Width)

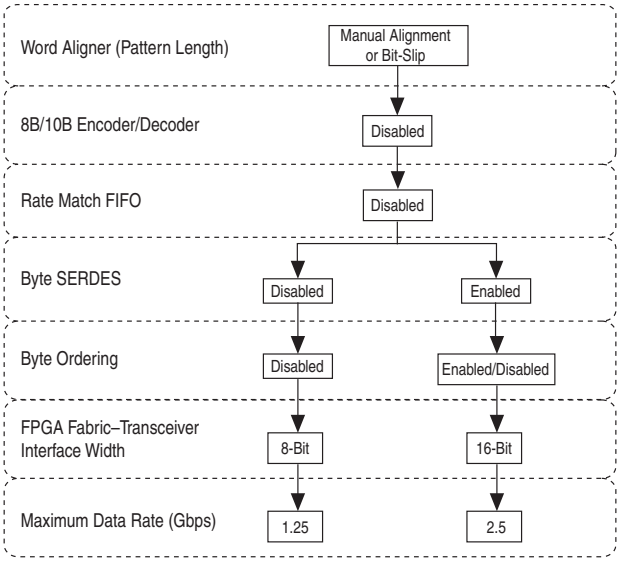


Figure 5–3. Configuration Options for Custom Single-Width Mode (10-bit PMA–PCS Interface Width)

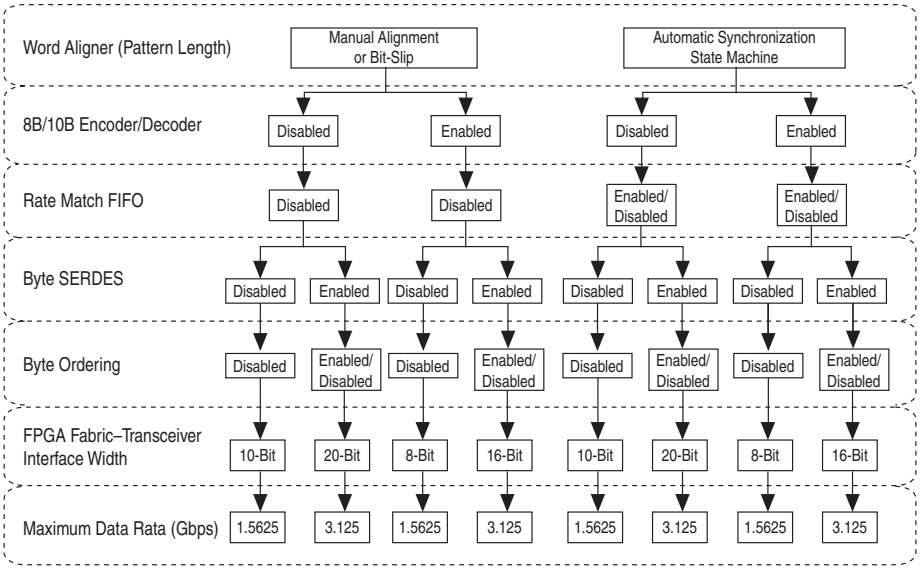


Figure 5–4. Configuration Options for Custom Double-Width Mode (16-bit PMA–PCS Interface Width)

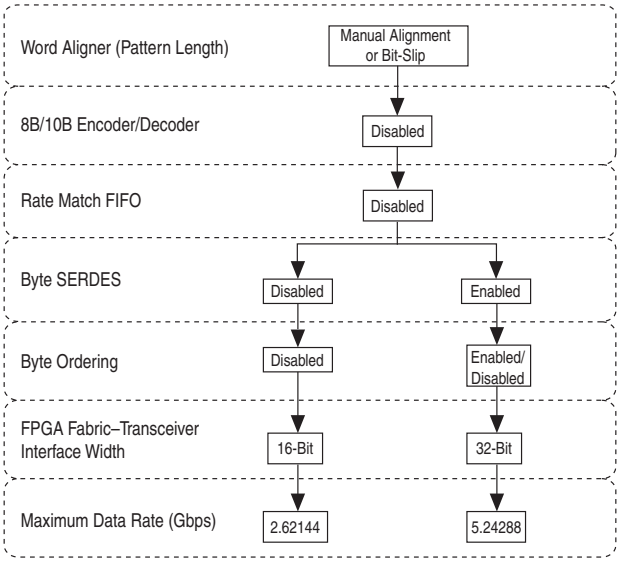
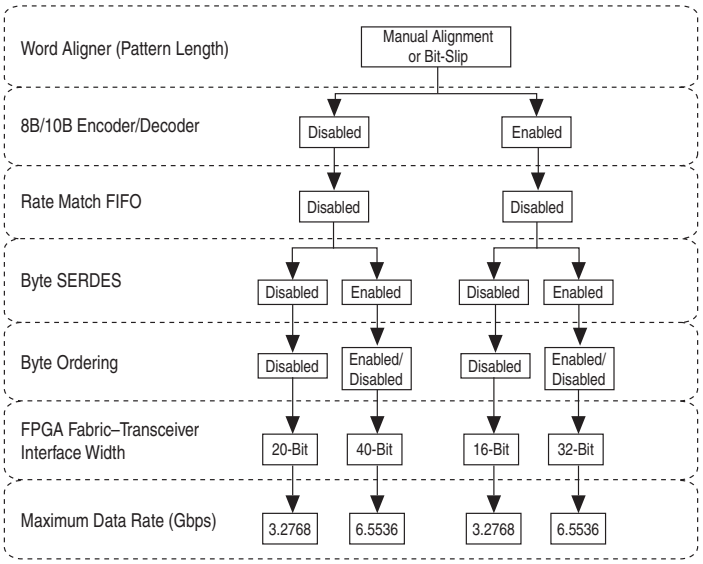


Figure 5–5. Configuration Options for Custom Double-Width Mode (20-bit PMA–PCS Interface Width)



For information about the operation of the blocks available in a custom configuration, excluding the rate match FIFO, refer to the *Transceiver Architecture in Arria V Devices* chapter.

In a custom configuration, the 20-bit pattern for the rate match FIFO is user-defined. The FIFO operates by looking for the 10-bit control pattern followed by the 10-bit skip pattern in the data, after the word aligner restores the word boundary. After finding the pattern, the FIFO performs a skip pattern insertion or deletion to ensure that the FIFO does not underflow or overflow a given part per million (ppm) difference between the clocks.

Table 5–3 lists the rate match FIFO behavior in a custom single-width mode configuration.

Table 5–3. Rate Match FIFO Behaviors in Custom Single-Width Mode (10-bit PMA–PCS Interface Width)

Operation	Behavior
Symbol Insertion	Inserts a maximum of four skip patterns in a cluster, only if there are no more than five skip patterns in the cluster after the symbol insertion.
Symbol Deletion	Deletes a maximum of four skip patterns in a cluster, only if there is one skip pattern left in the cluster after the symbol deletion.
Full Condition	Deletes the data byte that causes the FIFO to go full.
Empty Condition	Inserts a /K30.7/ (9'h1FE) after the data byte that caused the FIFO to go empty.

Table 5–4 lists the rate match FIFO behaviors in custom a double-width mode configuration.

Table 5–4. Rate Match FIFO Behaviors in Custom Double-Width Mode (20-bit PMA–PCS Interface Width) *(1)*

Operation	Behavior
Symbol Insertion	Inserts as many pairs (10-bit skip patterns at the LSByte and MSByte of the 20-bit word at the same clock cycle) of skip patterns as needed.
Symbol Deletion	Deletes as many pairs (10-bit skip patterns at the LSByte and MSByte of the 20-bit word at the same clock cycle) of skip patterns as needed.
Full Condition	Deletes the pair (20-bit word) of data bytes that causes the FIFO to go full.
Empty Condition	Inserts a pair of /K30.7/ ({9'h1FE, 9'h1FE}) after the data byte that causes the FIFO to go empty.

Note to Table 5–4:

(1) The rate match FIFO operation requires 8B/10B-coded data.

Figure 5–6 shows an example of rate match FIFO deletion in the case where three skip patterns are required to be deleted. In this example, /K28.5/ is the control pattern and neutral disparity /K28.0/ is the skip pattern. The first skip cluster has a /K28.5/ control pattern followed by two /K28.0/ skip patterns. The second skip cluster has a /K28.5/ control pattern followed by four /K28.0/ skip patterns. The rate match FIFO deletes only one /K28.0/ skip pattern from the first skip cluster to maintain at least one skip pattern in the cluster after deletion. Two /K28.0/ skip patterns are deleted from the second cluster for the three skip pattern deletion requirement.

Figure 5–6. Rate Match Deletion in Custom Single-Width Mode

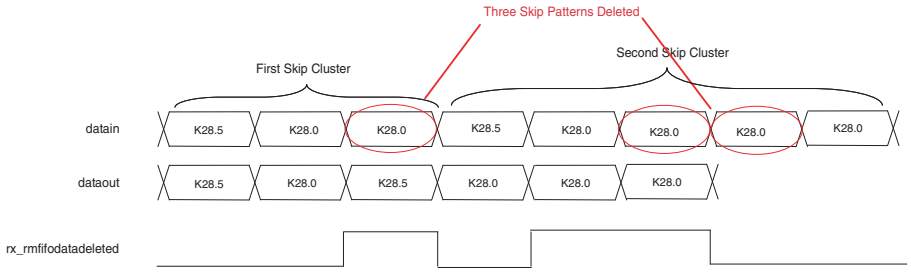


Figure 5–7 shows an example of rate match FIFO insertion in the case where three skip patterns are required to be inserted. In this example, /K28.5/ is the control pattern and neutral disparity /K28.0/ is the skip pattern. The first skip cluster has a /K28.5/ control pattern followed by three /K28.0/ skip patterns. The second skip cluster has a /K28.5/ control pattern followed by one /K28.0/ skip pattern. The rate match FIFO inserts only two /K28.0/ skip patterns into the first skip cluster to maintain a maximum of five skip patterns in the cluster after insertion. One /K28.0/ skip pattern is inserted into the second cluster for a total of three skip patterns to meet the insertion requirement.

Figure 5–7. Rate Match Insertion in Custom Single-Width Mode

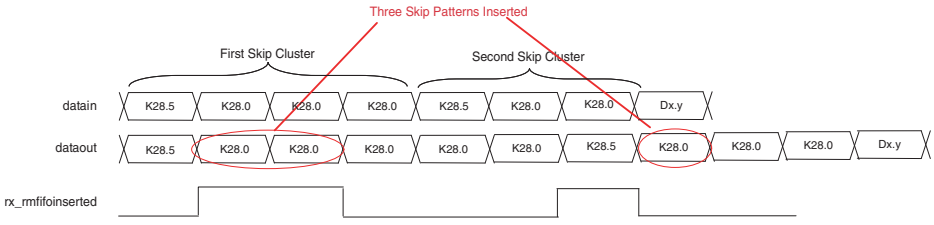


Figure 5–8 shows the rate match FIFO full condition in custom single-width mode. The rate match FIFO becomes full after receiving data byte D4.

Figure 5–8. Rate Match FIFO Full Condition in Custom Single-Width Mode

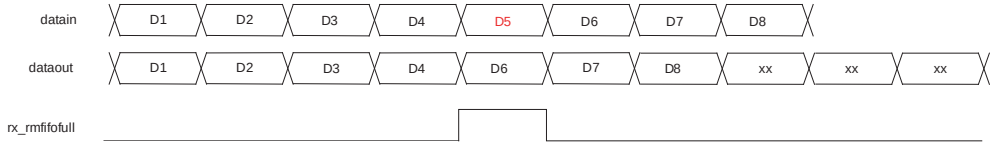


Figure 5–9 shows the rate match FIFO empty condition in custom single-width mode. The rate match FIFO becomes empty after reading out data byte D3.

Figure 5–9. Rate Match FIFO Empty Condition in Custom Single-Width Mode

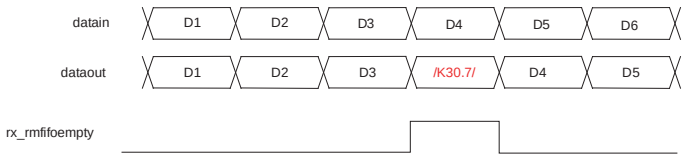


Figure 5–10 shows an example of rate match FIFO deletion in the case where three skip patterns are required to be deleted. In this example, /K28.5/ is the control pattern and neutral disparity /K28.0/ is the skip pattern. The first skip cluster has a /K28.5/ control pattern in the LSByte and /K28.0/ skip pattern in the MSByte of a clock cycle followed by one /K28.0/ skip pattern in the LSByte of the next clock cycle. The rate match FIFO cannot delete the two skip patterns in this skip cluster because they do not appear in the same clock cycle. The second skip cluster has a /K28.5/ control pattern in the MSByte of a clock cycle followed by two pairs of /K28.0/ skip patterns in the next two cycles. The rate match FIFO deletes both pairs of /K28.0/ skip patterns (for a total of four skip patterns deleted) from the second skip cluster to meet the three skip pattern deletion requirement.

Figure 5–10. Rate Match Deletion in Custom Double-Width Mode

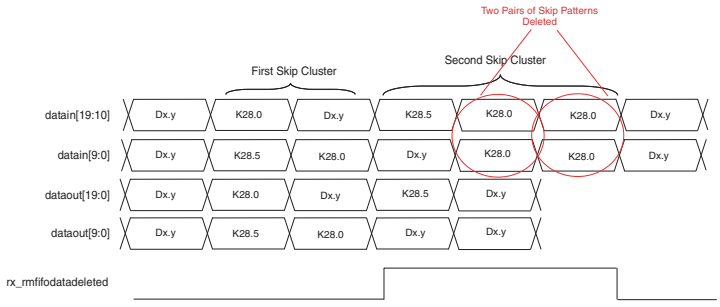


Figure 5–11 shows an example of rate match FIFO insertion in the case where three skip patterns are required to be inserted. In this example, /K28.5/ is the control pattern and neutral disparity /K28.0/ is the skip pattern. The first skip cluster has a /K28.5/ control pattern in the LSByte and /K28.0/ skip pattern in the MSByte of a clock cycle followed by one /K28.0/ skip pattern in the LSByte of the next clock cycle. The rate match FIFO inserts pairs of skip patterns in this skip cluster to meet the three skip pattern insertion requirement.

Figure 5–11. Rate Match Insertion in Custom Double-Width Mode

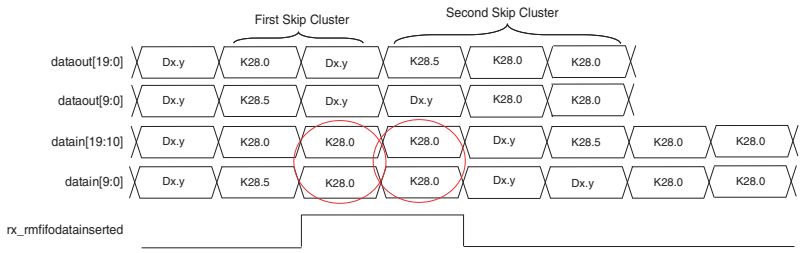


Figure 5–12 shows the rate match FIFO full condition in custom double-width mode. The rate match FIFO becomes full after receiving the 20-bit word D5D6.

Figure 5–12. Rate Match FIFO Full Condition in Custom Double-Width Mode

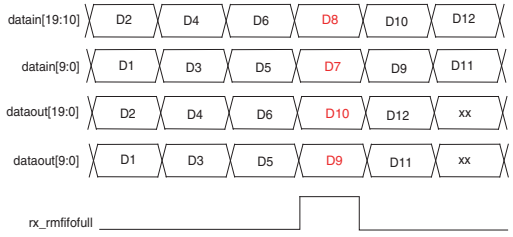
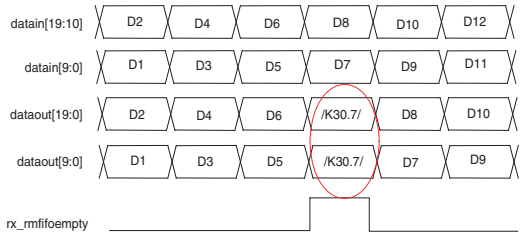


Figure 5–13 shows the rate match FIFO empty condition in custom double-width mode. The rate match FIFO becomes empty after reading out the 20-bit word D5D6.

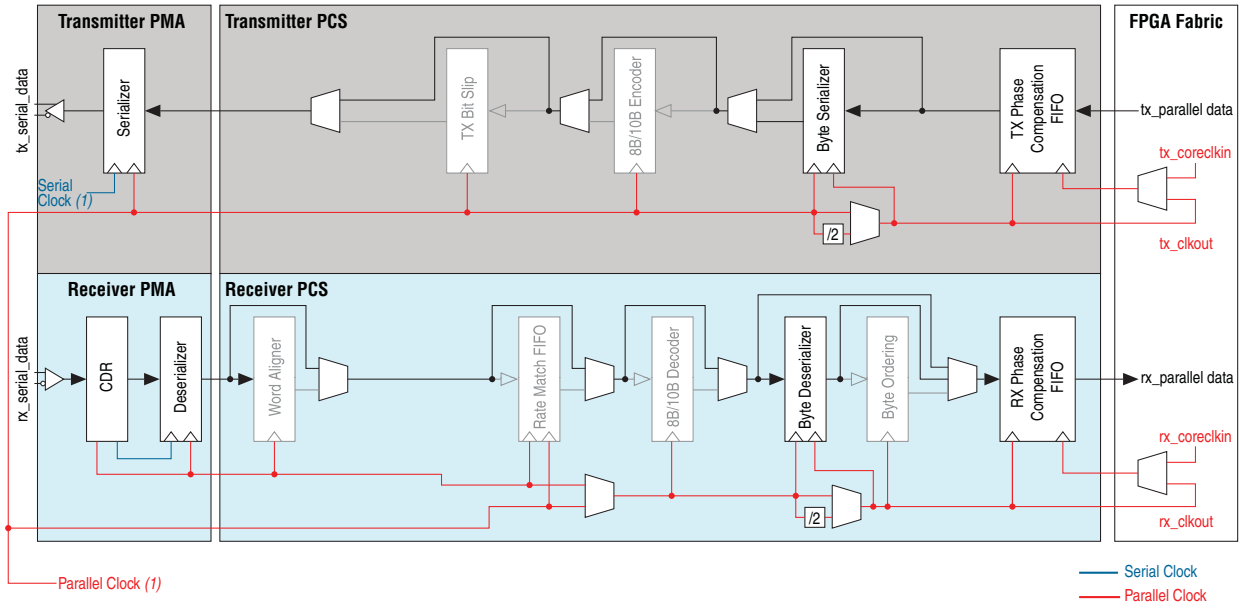
Figure 5–13. Rate Match FIFO Empty Condition in Custom Double-Width Mode



Low Latency Custom Configuration

In a low latency custom configuration, you can customize the transceiver channel to include a PMA and PCS that bypasses most of the PCS logical functionality for a low latency datapath. To provide a low latency datapath, the PCS includes only the phase compensation FIFO in phase compensation mode, and optionally, the byte serializer and byte deserializer blocks, as shown in Figure 5–14. The transceiver channel interfaces with the FPGA fabric through the PCS.

Figure 5–14. Complete Datapath in Low Latency Custom Configuration



Note to Figure 5–14:

(1) The serial and parallel clocks are sourced from the clock divider.

The maximum supported data rate varies depending on the customization and is identical to the custom configuration except that the 8B/10B block is disabled, as listed in Table 5–2 on page 5–3.

The supported configuration options of the channel are shown in Figure 5–15 through Figure 5–18, where:

- The blocks shown as "Disabled" are not used but incur latency.
- The blocks shown as "Bypassed" are not used and do not incur any latency.
- The transmitter bit-slip is disabled.

Figure 5–15. Configuration Options for Low Latency Custom Single-Width Mode (8-bit PMA–PCS Interface Width)

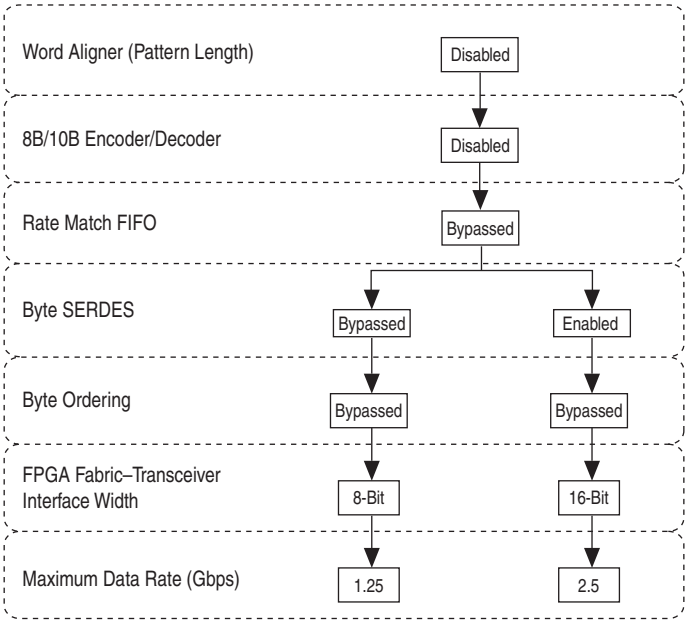


Figure 5–16. Configuration Options for Low Latency Custom Single-Width Mode (10-bit PMA–PCS Interface Width)

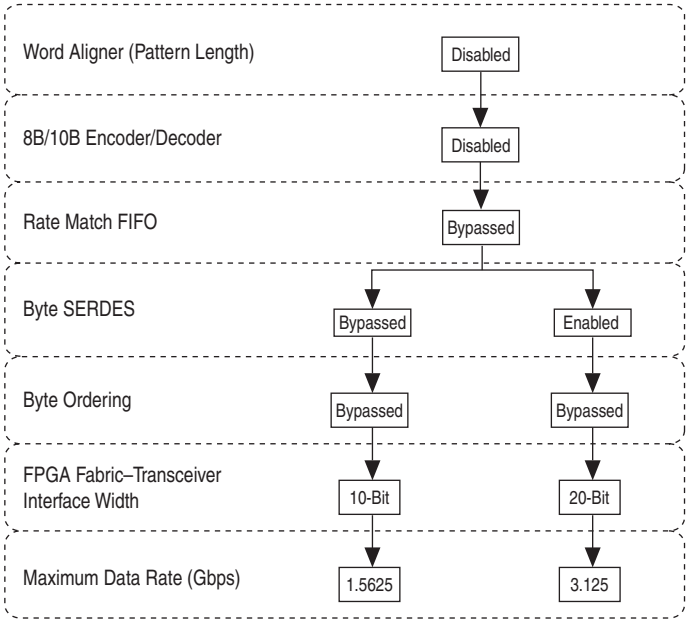


Figure 5–17. Configuration Options for Low Latency Custom Double-Width Mode (16-bit PMA–PCS Interface Width)

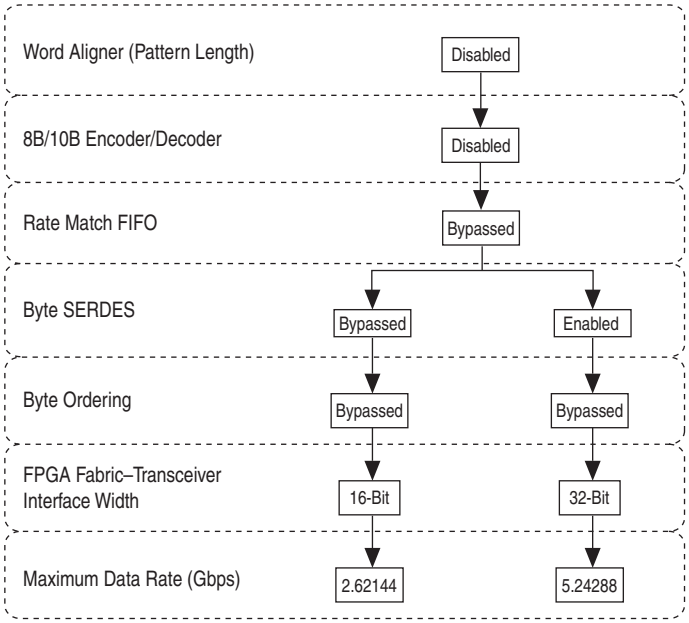
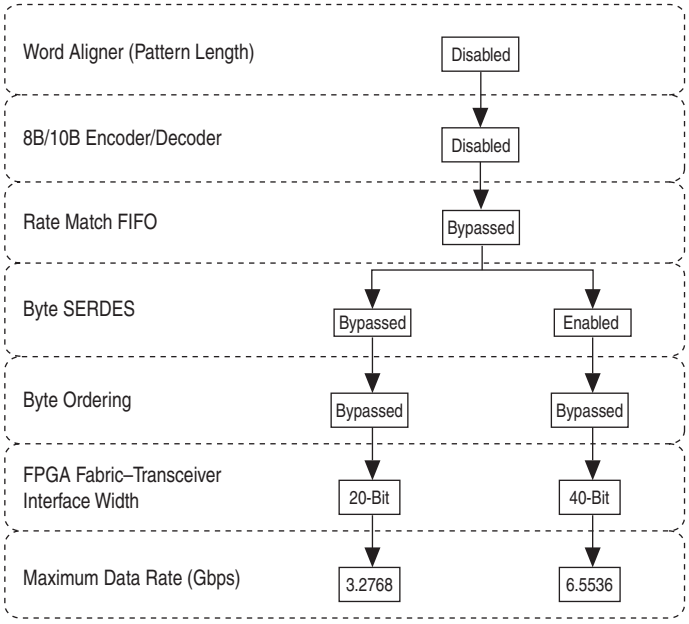


Figure 5–18. Configuration Options for Low Latency Custom Double-Width Mode (20-bit PMA–PCS Interface Width)



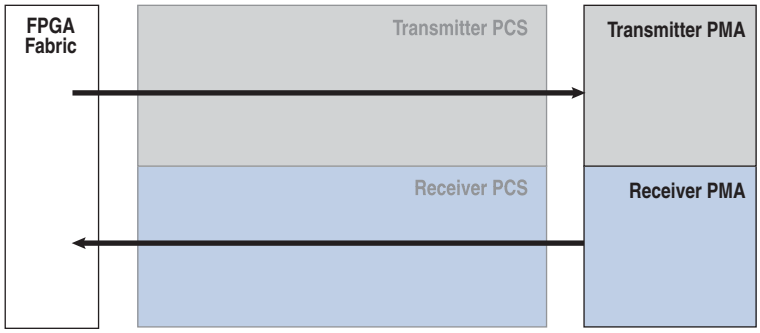
For information about the operation of the blocks available in a low latency custom configuration, refer to the *Transceiver Architecture in Arria V Devices* chapter.

PMA Direct Configuration

In a PMA Direct configuration, you can customize the transceiver channel to include only the PMA. In this configuration, the serializer and deserializer interface directly to the FPGA fabric, bypassing the PCS. The serializer and deserializer support 8-, 10-, 16-, 20- and 80-bit configurations. You must implement the PCS functions in the FPGA fabric.

Figure 5–19 shows the transceiver datapath in a PMA Direct configuration.

Figure 5–19. Transceiver Datapath in a PMA Direct Configuration



Document Revision History

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

Table 5–5. Document Revision History

Date	Version	Changes
June 2012	1.2	■ Updated for the Quartus II software version 12.0. ■ Updated the "PCS Datapath Latency" section. ■ Updated the "PMA Direct Configuration" section. ■ Updated Figure 5–1 and Figure 5–14.
November 2011	1.1	Updated for the Quartus II software version 11.1.
August 2011	1.0	Initial release.

6. Transceiver Loopback Support in Arria V Devices

AV53006-1.2

This chapter describes the loopback options available for Arria® V devices, which allow you to verify how different functional blocks work in the transceiver.

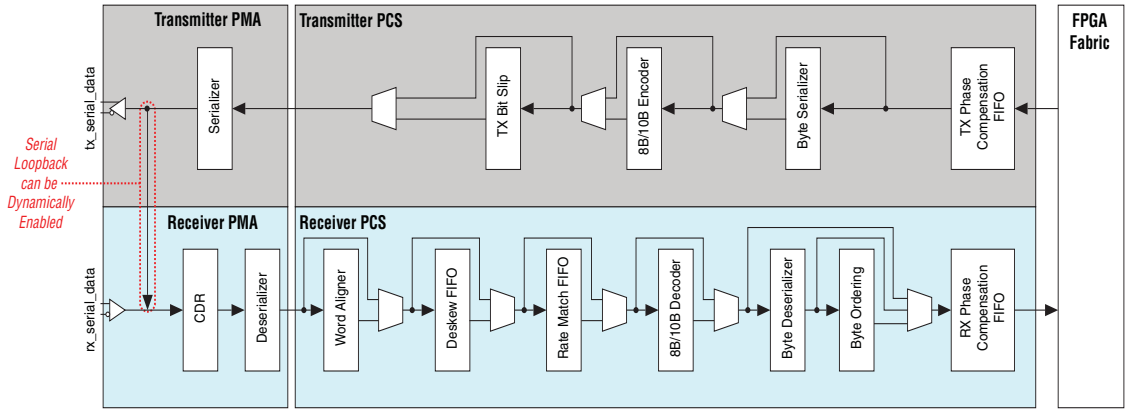
The available loopback options are:

Serial Loopback

Serial loopback is available for all transceiver configurations except the PIPE mode. You can use serial loopback as a debugging aid to ensure that the enabled physical coding sublayer (PCS) and physical media attachment (PMA) blocks in the transmitter and receiver channels are functioning correctly. Furthermore, you can dynamically enable serial loopback on a channel-by-channel basis.

Figure 6–1 shows the datapath for serial loopback. The data from the FPGA fabric passes through the transmitter channel and is looped back to the receiver channel, bypassing the receiver buffer. The received data is available to the FPGA logic for verification.

Figure 6–1. Serial Loopback Datapath



When you enable serial loopback, the transmitter channel sends data to both the `tx_serial_data` output port and to the receiver channel. The differential output voltage on the `tx_serial_data` port is based on the selected differential output voltage (V_{OD}) settings.

The looped-back data is forwarded to the receiver clock data recovery (CDR). You must provide an alignment pattern for the word aligner to enable the receiver channel to retrieve the byte boundary.

If the device is not in the serial loopback configuration and is receiving data from a remote device, the recovered clock from the receiver CDR is locked to the data from the remote source.

If the device is placed in the serial loopback configuration, the data source to the receiver changes from the remote device to the local transmitter channel—prompting the receiver CDR to start tracking the phase of the new data source. During this time, the recovered clock from the receiver CDR may be unstable. Because the receiver PCS is running off of this recovered clock, you must place the receiver PCS under reset by asserting the `rx_digitalreset` signal during this period.

When moving into or out of serial loopback, you must assert the `rx_digitalreset` signal for a minimum of two parallel clock cycles.

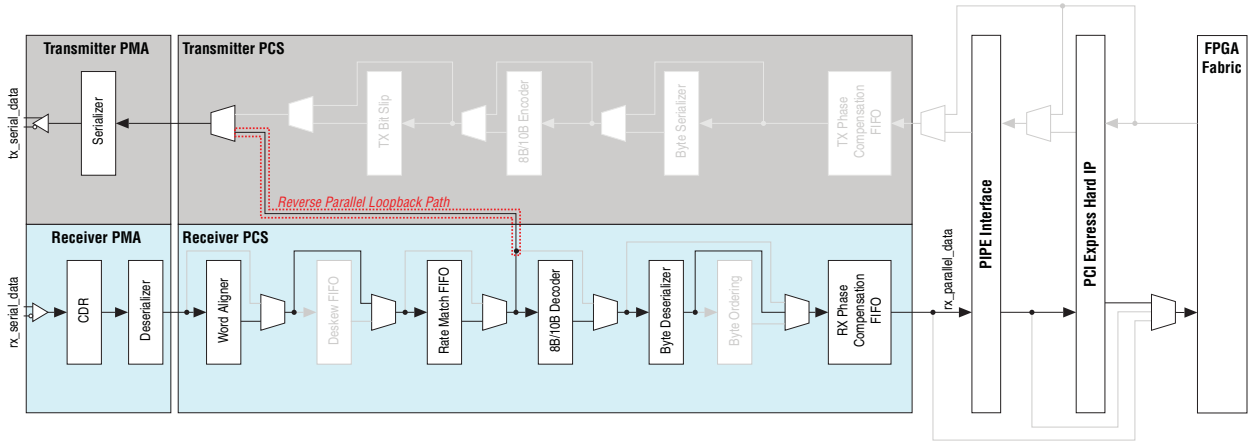
PIPE Reverse Parallel Loopback

PIPE reverse parallel loopback is only available in the PCIe® configuration for Gen1 and Gen2 data rates. Figure 6–2 shows the received serial data passing through the receiver CDR, deserializer, word aligner, and rate match FIFO buffer. The parallel data from the rate match FIFO is then looped back to the transmitter serializer and transmitted out through the `tx_serial_data` port. The received data is also available to the FPGA fabric through the `rx_parallel_data` signal.

PIPE reverse parallel loopback is compliant with the PCIe 2.0 specification. To enable this loopback configuration, assert the `pipe_txdetectrx_loopback` signal.

PIPE reverse parallel loopback is the only loopback option supported in the PCIe configuration.

Figure 6–2. PIPE Reverse Parallel Loopback Configuration Datapath *(1)*



Note to Figure 6–2:

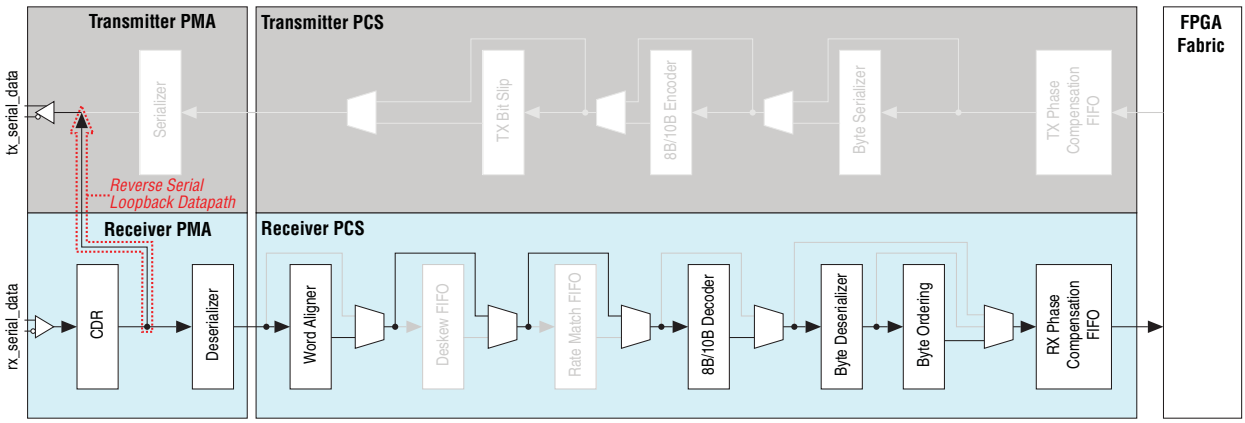
(1) Grayed-out blocks are not active when the PIPE reverse parallel loopback is enabled.

Reverse Serial Loopback

Reverse serial loopback is available as a subprotocol under custom configuration. In reverse serial loopback, the data is received through the `rx_serial_data` port, retimed through the receiver CDR, and sent to the `tx_serial_data` port. The received data is also available to the FPGA logic. No dynamic pin control is available to select or deselect reverse serial loopback. Figure 6–3 shows the transceiver channel datapath for reverse serial loopback mode.

The transmitter buffer is the only active block in the transmitter channel. You can change the VOD and the pre-emphasis first post tap values on the transmitter buffer through the ALTGX MegaWizard™ Plug-In Manager or through the dynamic reconfiguration controller. Reverse serial loopback is often implemented when using a bit error rate tester (BERT) on the upstream transmitter.

Figure 6–3. Reverse Serial Loopback Datapath *(1)*



Note to Figure 6–3:

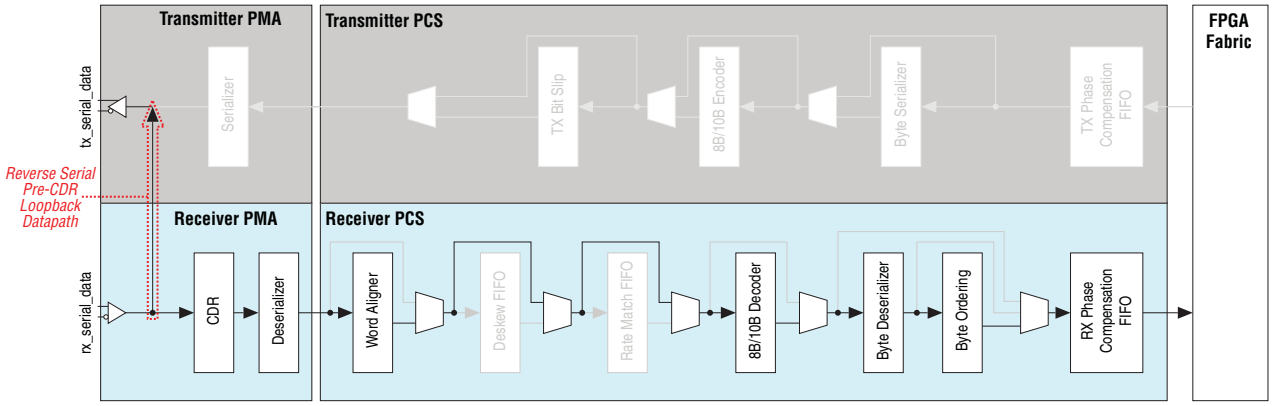
(1) Grayed-out blocks are not active when the reverse serial loopback is enabled.

Reverse Serial Pre-CDR Loopback

Reverse serial pre-CDR loopback is available as a subprotocol under custom configuration. In reverse serial pre-CDR loopback, the data received through the `rx_serial_data` port is looped back to the `tx_serial_data` port before the receiver CDR. The received data is also available to the FPGA logic. Figure 6–4 shows the transceiver channel datapath for reverse serial pre-CDR loopback mode.

The transmitter buffer is the only active block in the transmitter channel. You can change the VOD on the transmitter buffer through the ALTGX MegaWizard Plug-In Manager. The pre-emphasis settings for the transmitter buffer cannot be changed in this configuration.

Figure 6–4. Reverse Serial Pre-CDR Loopback Datapath *(1)*



Note to Figure 6–3:

(1) Grayed-out blocks are not active when the reverse serial pre-CDR loopback is enabled.

Document Revision History

Table 6–1 lists the revision history for this chapter.

Table 6–1. Document Revision History

Date	Version	Changes
June 2012	1.2	Minor updates for the Quartus II software version 12.0
November 2011	1.1	■ Added the following two loopback options: ■ "Reverse Serial Loopback" ■ "Reverse Serial Pre-CDR Loopback"
August 2011	1.0	Initial release.

7. Dynamic Reconfiguration in Arria V Devices

AV53007-1.2

This chapter describes the dynamic reconfiguration features available in Arria® V transceivers. Table 7–1 shows the reconfiguration applications, the blocks that are being reconfigured, and the reconfiguration modes that are supported by the Arria V transceivers.

Table 7–1. Arria V Reconfiguration Applications

Reconfiguration Application	Affected Blocks	Reconfiguration Mode
Offset Cancellation—Counter offset variations due to process operation for the analog circuit. This feature is mandatory if you use receivers.	CDR	Transceiver Calibration Functions
DCD Calibration—Compensates for the duty cycle distortion caused by clock network skew.	TX buffer and clock network skew	Transceiver Calibration Functions
Analog Controls Reconfiguration—Fine-tune the signal integrity by adjusting the transmitter (TX) and receiver (RX) analog settings while bringing up a link	Analog circuit of TX and RX buffer	PMA Controls Reconfiguration Mode
Loopback Modes—Enable or disable Pre- and Post-CDR Reverse Serial Loopback dynamically	PMA	PMA Controls Reconfiguration Mode
Data Rate Change Applications: ■ Increase or decrease the data rate (/1, /2, /4, /8) for autonegotiation purposes such as CPRI and SATA/SAS applications ■ Reconfigure the TX PLL settings for protocols with multi-data rate support such as CPRI ■ Switch between multiple TX PLLs for multi-data rate support ■ Channel reconfiguration—Reconfigure the RX CDR PLL from one data rate to another data rate	■ TX Local clock dividers ■ TX PLL ■ TX PLL within the same transceiver banks ■ RX CDR PLL	■ Transceiver PLL Reconfiguration Mode ■ Transceiver Channel and Interface Reconfiguration Mode

The transceiver reconfiguration controller offers several different reconfiguration modes. You can choose the appropriate reconfiguration modes that best suit your application needs. All the reconfiguration applications are done through the transceiver reconfiguration PHY IP.

For more information about the transceiver reconfiguration PHY IP, refer to the *Altera Transceiver PHY IP Core User Guide*.

This chapter contains the following sections:

Offset Cancellation and TX Duty Cycle Distortion Calibration

Every transceiver channel in Arria V devices has an offset cancellation circuitry to compensate for the offset variations that are caused by process operations. The offset cancellation circuitry is controlled by the offset cancellation control logic IP within the transceiver reconfiguration controller.

The TX clocks that are generated by the CMU and travel across the clock network may introduce duty cycle distortions (DCD). The DCD can be reduced with the DCD calibration IP that is integrated in the transceiver reconfiguration controller.

These IPs are active only during device power up and they automatically perform the offset cancellation process and DCD calibration. The DCD calibration is enabled based on your design need. When the offset cancellation and DCD calibration are completed, a `reconfig_busy` status signal from the reconfiguration controller is deasserted to indicate the completion of the processes. For transceiver applications that operate at 5 Gigabits per second (Gbps) or higher, you must turn on the DCD calibration for better TX jitter performance. If the transceiver application is operating below 5 Gbps, you do not have to turn on the DCD calibration feature.

In Arria V devices the embedded reset controller automatically performs a reset sequence on the transceiver's channels after the offset cancellation process and DCD calibration are complete. After the reset sequence is completed, the transceiver reconfiguration controller is ready for user-controlled operations.

PMA Controls Reconfiguration

After offset cancellation is complete and the required transceiver reset sequence has been performed automatically by the embedded reset controller, you can dynamically reconfigure the analog controls setting. You can continue with the subsequent reconfigurations of the analog controls when the `reconfig_busy` status signal is low. A high on the `reconfig_busy` signal indicates that the reconfiguration operation is in progress.

You can reconfigure the following transceiver analog controls:

- Transmitter pre-emphasis
- Differential output voltage (V_{OD})
- Receiver equalizer control
- Direct-current (DC) gain settings

To reconfigure the analog control settings, perform `read` and `write` operations to the PMA analog settings reconfiguration control IP within the reconfiguration controller. For more information about the `read` and `write` operations, refer to the *Altera Transceiver PHY IP Core User Guide*.

Enabling and Disabling Loopback Modes

The following loopback paths are available:

- **Post-CDR reverse serial loopback path**—The RX captures the input data and feeds it into the CDR. The recovered data from the CDR output feeds into the TX driver and sends to the TX pins through the TX driver. For this path, the RX and CDR can be tested. For this path, the TX driver can be programmed to use either the main tap only or the main tap and the pre-emphasis first post-tap. Enabling or disabling the post-CDR reverse serial loopback modes is done through the PMA Analog Reconfiguration IP in the Transceiver Reconfiguration PHY IP.
- **Pre-CDR reverse serial loopback path**—The RX captures the input data and feeds it back to the TX driver through a buffer. With this path, you can perform a quick check for the quality of the RX and TX buffers. Enabling or disabling the pre-CDR reverse serial loopback mode is done through the analog IP in the Transceiver Reconfiguration PHY IP.

For implementation details about the different loopback modes, refer to "Loopback Modes" in the *Transceiver Reconfiguration Controller* chapter of the *Altera Transceiver PHY IP Core User Guide*.

Transceiver PLL Reconfiguration

The Arria V Transceiver Reconfiguration Controller supports PLL reconfiguration. You can reconfigure the PLL connected to the transceiver channel. For example, you can change the data rate from 2.5G to 5G.

The Transceiver Reconfiguration PHY IP in Arria V devices provides an Avalon®-MM user interface to perform PLL reconfiguration.

For more information about performing PLL reconfiguration, refer to the "PLL Reconfiguration" section in the *Transceiver Reconfiguration Controller* chapter of the *Altera Transceiver PHY IP Core User Guide*.

Transceiver Channel and Interface Reconfiguration

The Arria V Transceiver Reconfiguration Controller supports channel and interface reconfiguration.

The blocks that are reconfigured by this dynamic reconfiguration mode are the PCS and PMA blocks of a transceiver channel, the clock divider settings of the transmitter, and the CDR.

Transceiver Channel Reconfiguration

You can reconfigure the channels in the following ways:

- Reconfigure the CDR of the receiver channel.
- Enable and disable all static PCS sub-blocks.
- Select an alternate PLL within the transceiver block to supply a different clock to the transceiver clock generation block.
- Reconfigure the TX local clock divider with a 1, 2, 4, or 8 division factor.

Every transmitter channel has a clock divider. When you reconfigure these clock dividers, ensure that the functional mode of the transceiver channel supports the reconfigured data rate.

For example, you can change the data rate from 6.25 Gbps to 3.125 Gbps by reconfiguring the clock divider.

Transceiver Interface Reconfiguration

You can reconfigure the transceiver interfaces by reconfiguring the FPGA fabric-transceiver channel data width that includes PCS-PLD and PMA-PCS interfaces.

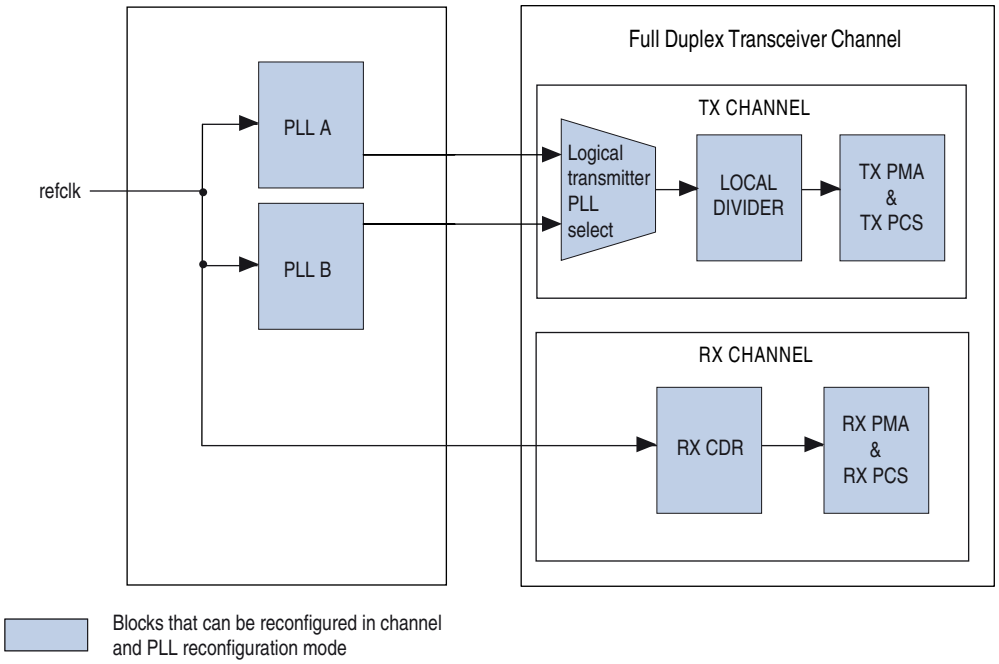
For example, you can reconfigure the custom PHY IP to enable or disable the 8B/10B encoder/decoder. There is no limit to the number of functional modes you can reconfigure the transceiver channel to if the various clocks involved support the transition. When you switch the custom PHY IP from one function mode to a different function mode, you may need to reconfigure the FPGA fabric-transceiver channel data width, enable or disable PCS sub-blocks, or both, to comply with the protocol requirements.

Channel reconfiguration only affects the channel involved in the reconfiguration (the transceiver channel specified by the unique logical channel address), without affecting the remaining transceiver channels controlled by the same Transceiver Reconfiguration Controller. PLL reconfiguration affects all channels that are currently using that PLL for transmission.

Channel reconfiguration from either a transmitter-only configuration to a receiver-only configuration or vice versa is not allowed.

Figure 7–1 shows the functional blocks that you can dynamically reconfigure using transceiver channel and PLL reconfiguration mode.

Figure 7–1. Transceiver Channel and PLL Reconfiguration in a Transceiver Block



For more information about the transceiver channel and PLL reconfiguration, refer to the "Channel and PLL Reconfiguration" section in the *Transceiver Reconfiguration Controller* chapter of the *Altera Transceiver PHY IP Core User Guide*.

Document Revision History

Table 7–2 lists the revision history for this chapter.

Table 7–2. Document Revision History

Date	Version	Changes
June 2012	1.2	■ Added "Transceiver PLL Reconfiguration" and "Transceiver Channel and Interface Reconfiguration" sections. ■ Updated the "Offset Cancellation and TX Duty Cycle Distortion Calibration" section. ■ Updated Table 7–1.
November 2011	1.1	■ Added Table 7–1. ■ Deleted "Transceiver Reconfiguration Controller" and "Channel and PLL Reconfiguration" sections. ■ Updated "Offset Cancellation" and "PMA Analog Settings Reconfiguration" sections ■ Added "Enabling and Disabling Loopback Modes" section. ■ Minor text edits.
August 2011	1.0	Initial release.

This chapter provides additional information about the document and Altera.

How to Contact Altera

To locate 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

The following table shows the typographic conventions this document uses.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Indicate command names, dialog box titles, dialog box options, and other GUI labels. For example, **Save As** dialog box. For GUI elements, capitalization matches the GUI.
bold type	Indicates directory names, project names, disk drive names, file names, file name extensions, software utility names, and GUI labels. For example, **\qdesigns** directory, **D:** drive, and **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Indicate document titles. For example, *Stratix IV Design Guidelines*.
italic type	Indicates variables. For example, *n* + 1. Variable names are enclosed in angle brackets (< >). For example, *<file name>* and *<project name>***.pof** file.
Initial Capital Letters	Indicate keyboard keys and menu names. For example, the 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`. The suffix `n` denotes an active-low signal. 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`).
↵	An angled arrow instructs you to press the Enter key.
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.
	A question mark directs you to a software help system with related information.
	The feet direct you to another document or website with related information.
	A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.
	A warning calls attention to a condition or possible situation that can cause you injury.
	The envelope links to the Email Subscription Management Center page of the Altera website, where you can sign up to receive update notifications for Altera documents.