LVDS SERDES Transmitter / Receiver (ALTLVDS_TX and ALTLVDS_RX) Megafunction

User Guide

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

UG-MF9504-9.0

Document last updated for Altera Complete Design Suite version: 11.0
Document publication date: February 2012

Subscribe

1. About This Megafunction

This user guide describes the features of the low-voltage differential signalling serializer or deserializer (LVDS SERDES) megafunctions—ALTLVDS_TX and ALTLVDS_RX, that you can configure through the parameter editor in the Quartus® II software. The ALTLVDS_TX and ALTLVDS_RX megafunctions implement the LVDS SERDES interfaces to transmit and receive high-speed differential data.

This user guide assumes that you are familiar with megafunctions and how to create them. If you are unfamiliar with Altera® megafunctions, refer to the *Introduction to Megafunctions User Guide*.

Features

Table 1–1 lists the features of the ALTLVDS_TX and ALTLVDS_RX megafunctions.

Table 1–1. ALTLVDS_TX and ALTLVDS_RX Features

Features	Supported devices
ALTLVDS_TX and ALTLVDS_RX	
Parameterizable data channel widths	All devices stated in the Device Support section.
Parameterizable serializer/deserializer (SERDES) factors	All devices stated in the Device Support section.
Registered input and output ports	All devices stated in the Device Support section.
Support for external phase-locked loops (PLL)	All devices stated in the Device Support section.
PLLs sharing between transmitters and receivers	All devices stated in the Device Support section.
PLL control signals	All devices stated in the Device Support section.
Choice to implement the LVDS interface in dedicated circuitry or logic cells	All devices stated in the Device Support section excluding Cyclone® series and Stratix® V.
ALTLVDS_RX Only	
Dynamic phase alignment (DPA) mode support	All devices stated in the Device Support section excluding Cyclone series.
DPA PLL calibration support	All devices stated in the Device Support section excluding Arria® II GX, Cyclone series, HardCopy® II, Stratix, Stratix GX, Stratix II, and Stratix II GX.
Soft clock data recovery (CDR) mode support	Arria II GX, Arria II GZ, HardCopy III, HardCopy IV, Stratix III, Stratix IV, and Stratix V devices.

Device Support

The ALTLVDS_TX and ALTLVDS_RX megafunctions support the following device families:

- Arria series
- Cyclone series
- HardCopy series
- Stratix series

2. Parameter Settings

This section describes the parameter settings for the ALTLVDS_TX and ALTLVDS_RX megafunctions.

You can parameterize the megafunctions using the MegaWizard™ Plug-In Manager or the command-line interface (CLI). Altera recommends that you configure the megafunctions using the MegaWizard Plug-In Manager.

MegaWizard Parameter Settings for the LVDS Transmitter

On the **General** page (page 3) of the parameter editor, depending on the device you selected, you can configure the following options:

- Implement the SERDES circuitry in LEs (logic cells) or dedicated (hard) SERDES block
- Use internal PLL or external PLL

The selections you make on the **General** page determine the features available on the remaining pages of the parameter editor.

Table 2–1 lists the parameter settings for the LVDS transmitter megafunction.

The options on pages 1 and 2a of the parameter editor are the same for all supported device families. For more information, refer to the *Introduction to Megafunctions User Guide*.

Table 2–1. ALTLVDS_TX Parameter Settings (Part 1 of 6)

Option	Description
General (page 3)	
Implement Deserializer circuitry in logic cells	Turn on this option to implement the SERDES circuitry in logic cells. The transmitter starts its operation on the first fast clock edge after the PLL is locked. This option is intended for slow speeds. The byte alignment might be different from the dedicated SERDES implementation. Turn off this option to use the dedicated SERDES circuitry in the device. When you implement the dedicated SERDES in the LVDS transmitter, the SERDES connects to the LVDS transmitter; therefore, the output of the transmitter cannot be assigned to single-ended I/O standards. This feature is supported in Arria GX, Arria II GX, Arria II GZ, HardCopy II, HardCopy III, HardCopy IV, Stratix, Stratix GX, Stratix II, Stratix II GX, Stratix III, and Stratix IV devices. This option is not supported in Stratix V devices. In Cyclone series, the SERDES circuitry is always implemented in logic cells.

Table 2–1. ALTLVDS_TX Parameter Settings (Part 2 of 6)

Option	Description
What is the number of Channels?	Number of output channels available for the LVDS transmitter. If the required number of channels is not available in the list, type the desired number. For example, if the number of channels is **44**, the port created is `tx_out[43..0]`. The legal values depend on the pins available in the device. For the legal values for your device, refer to the relevant device handbook.
What is the deserialization factor?	Determines the number of parallel bits from the core that the transmitter serializes and sends out. For example, if the deserialization factor is **10** and the number of output channels is **1**, the transmitter serializes every 10 parallel bits into a single output channel. If the deserialization factor is **10** and the number of channels is **44**, the port created is `tx_in[439..0]`. For the valid deserialization factors for your device, refer to the relevant device handbook. When the `divide_by_factor` port shown in the parameter editor is identical to the deserialization factor, the parameter editor disables the 50/50 duty cycle for x5, x7, and x9 modes.
Use External PLL	Turn on this option to use an external PLL to clock the SERDES transmitter. When you turn on this option, the options on the **Frequency/PLL settings** page are disabled. You must use a separate PLL to provide the clocking source and make the necessary connections.You must ensure your circuit has the correct input and functionality to generate an appropriate clock frequency and is correctly connected to the LVDS transmitter. For more information about instantiating the ALTPLL megafunction to generate the various clock and load enable signals, refer to "Generating Clock Signals for LVDS Interface" on page 3–18. For more information about external PLL options in Stratix II refer to *AN 409: Design Example Using the ALTLVDS Megafunction & the External PLL Option in Stratix II Devices*. When you have a deserialization factor of two, the megafunction bypasses SERDES and implements the SERDES functionality in DDR registers. Your design requires a deserialization factor of at least four to turn on the external PLL option. If you turn off this option, the megafunction automatically implements an internal PLL to clock the ALTLVDS_TX block. For Stratix and Stratix GX devices, if you implement SERDES for your LVDS transmitter using a dedicated SERDES block, you do not have the option to use an external PLL.
Use clock pin	Turn on this option to bypass PLL usage and drive the SERDES directly with a clock pin. Use this option when the incoming LVDS clock (rx_inclock) is not a continuous running clock. However, when you turn on this option, the maximum data rate for the fastest speed grade device is limited to 717 Mbps. You must create an SDC file to specify timing constraints when running the TimeQuest Timing Analyzer to ensure timing closure at the LVDS interface.
Use 'tx_data_reset' input port	This option is available when you implement the LVDS in logic cells. When you turn on this option, it adds an input port in the megafunction, which when asserted asynchronously resets all the logic in the ALTLVDS_TX megafunction excluding the PLL.

Table 2–1. ALTLVDS_TX Parameter Settings (Part 3 of 6)

Option	Description
Frequency/ PLL Settings (page 4) The options on this page are available only when you are using internal PLL	
What is the output data rate?	Specifies the data rate for the output channel of the transmitter, in Megabits per second (Mbps). For data rate ranges, refer to the *Device Data Sheet* chapter in the relevant device handbook. This option determines the legal value of the input clock rate.
Specify input clock rate by	Specifies the clock frequency (`tx_inclock` port) or the clock (`inclock_period` parameter) going into the internal PLL. The legal values depend on the output data rate selected.
What is the phase alignment of 'tx_in' with respect to the rising edge of 'tx_inclock'? (in degrees)	Determines the phase alignment of the data transmitted by the core logic array with respect to the `tx_inclock` clock. The available values are **0.00**, **22.50**, **45.00**, **67.50**, **90.00**, **112.50**, **135.00**, **157.50**, **180.00**, **202.50**, **225.00**, **247.50**, **270.00**, **292.50**, **315.00**, and **337.50.**
Use 'tx_pll_enable' input port	Turn on to control the enable port of the fast PLL that the megafunction uses with this function. If the transmitter shares the PLL with other ALTLVDS blocks, and uses the `tx_pll_enable` port, you must use this port in all the megafunction instances and tie the signals together in the design file. If you use a PLL-enabled port in one megafunction instance and not another, the PLLs are not shared, and a warning appears during compilation.
Use 'pll_areset' input port	Turn on to control the asynchronous reset port of the PLL that the megafunction uses with this function. When the transmitter shares the PLL with other ALTLVDS blocks and uses the `pll_areset` port, you must use this port in all the megafunction instances and tie the signals together in the design file. If you use the `pll_areset` port in one megafunction instance only, the PLLs are not shared and a warning appears during compilation. The PLL must be reset to set the output clock phase relationships correctly when the PLL loses lock, or if the PLL input reference clock is not stable when the device completes the configuration process. For more information, refer to "Initializing the ALTLVDS_TX and ALTLVDS_RX Megafunctions" on page 3–6.
Align clock to center of data window	Turn on this option to add a phase shift of 90° to the clock, which center-aligns the clock in the data. Turn on this option for PLL merging if you also turn on this option for the receiver. This option is available only for Arria GX, Stratix II, Stratix II GX, and HardCopy II devices when you implement the SERDES in logic cells, and for Cyclone II devices.
Enable self-reset on lost lock in PLL	Turn on this option to reset the PLL automatically whenever the PLL loses lock. This option is available only for Arria II GX, Arria II GZ, HardCopy III, HardCopy IV, Stratix III, Stratix IV, and Stratix V devices when SERDES is implemented in logic cells, and for Cyclone III and Cyclone IV devices.
Use shared PLL(s) for receivers and transmitters	Turn on this option for your LVDS receivers and transmitters to share the same PLL. Turn on this option if the LVDS receivers and transmitters use the same input clock frequency, deserialization factor, and data rates.

Table 2–1. ALTLVDS_TX Parameter Settings (Part 4 of 6)

Option	Description
Register 'tx_in' input port using	Turn on this option to specify whether input registers are clocked by the tx_inclock signal or tx_coreclock signal. When the PLLs are shared, connect the tx_inclock signal to the same reference clock as the receiver function. For example, if the tx_inclock signal is connected to a 500-MHz input reference clock, and the parallel data rate is not 500 MHz, register the parallel data using the tx_coreclock signal that runs at the output serial data rate divided by the deserialization factor. This frequency matches the parallel data rate from the FPGA core. If you turn off this option, a warning message appears that directs you to pre-register the inputs in the logic that feeds the transmitter. When you use the Cyclone series with the ALTLVDS_TX and ALTLVDS_RX megafunctions, the interface always sends the most significant bit (MSB) of your parallel data first. When you use the ALTLVDS_TX megafunction, you might get setup timing violations when you use the tx_inclock signal to register the data that feeds the SERDES blocks. The ALTLVDS_TX megafunction gives you the choice to register the tx_in[] data with either the tx_inclock or tx_coreclock signal. The default setting is tx_coreclock. Using the tx_coreclock signal to register the data before it feeds the SERDES is the better choice, because it has the optimal phase position to register the data with respect to the high-speed clock that drives the SERDES. Your setup timing violations are eliminated when you use the tx_coreclock signal instead of the tx_inclock signal to register the data in the ALTLVDS_TX megafunction. Additionally, you get better timing margins when you use the tx_coreclock signal instead of the tx_inclock signal, even if you do not have timing violations.
Transmitter Settings (page 5)	
Use 'tx_outclock' output port	The tx_outclock signal is associated with the serial transmit data stream. Every tx_outclock signal goes through the shift register logic, excluding the following parameter configurations: ■ When the outclock_divide_by signal equals to 1, or ■ When the outclock_divide_by signal equals to deserialization_factor signal (for odd factors only) and the outclock_duty_cycle signal is **50**.
What is the outclock divide factor (B)?	Specifies the frequency of the tx_outclock signal as the transmitter output data rate divided by the outclock divide factor (B). For the legal values, refer to the relevant device handbook. For a SERDES factor of **5** and **9**, the outclock divide factors available are **1**, **5**, and **9**. The divide factor of **2** is not available. For Cyclone II devices and later, when the implement_in_les parameter is ON, the outclock_duty_cycle of **50** is not supported in the following parameter configurations: ■ deserialization_factor signal is **5**, **7**, or **9** ■ outclock_divide_by signal equals to deserialization_factor ■ outclock_multiply_by is 2

Table 2–1. ALTLVDS_TX Parameter Settings (Part 5 of 6)

Option	Description
Specify phase alignment of 'tx_outclock' with respect to 'tx_out'	Specifies the phase alignment of `tx_outclock` signal with respect to the `tx_out` signal. This option is available only if you use the `tx_outclock` signal.
What is the phase alignment of 'tx_outclock' with respect to 'tx_out'?	The available values are **0.00**, **22.50**, **45.00**, **67.50**, **90.00**, **112.50**, **135.00**, **157.50**, **180.00**, **202.50**, **225.00**, **247.50**, **270.00**, **292.50**, **315.00**, and **337.50.** This option is available only when you implement the SERDES in logic cells and uses the `tx_outclock` signal.
What is the outclock duty cycle?	The default value is **50**. The `outclock_duty_cycle` of **50** is not supported when: ■ `deserialization_factor` signal is 5, 7, or 9 ■ `outclock_divide_by` signal equals to `deserialization_factor` ■ `outclock_multiply_by` is 2
Use 'tx_locked' output port	Allows you to monitor the lock status of the PLL. The status of the lock port is identical for the transmitter and receiver when the megafunction uses shared PLLs.
Use 'tx_coreclock' output port	Turn on this option to show the core clock frequency during simulation. Enables the transmitter core clock signal to the registers of all the logic that feeds the LVDS transmitter function. If any other clock feeds the transmit function, your design must implement the clock domain transfer circuitry. You must add a false path constraint from the `slow_clock` signal to the `fast_clock` signal in the ALTLVDS_TX megafunction whenever the faster `core_clock` signal implementation is used for odd deserialization factors.
What is the clock resource used for 'tx_coreclock'?	Specifies the clock resource type fed to the `tx_coreclock` signal. Allowed values are **Auto selection** (the compiler determines the type), **Global clock**, and **Regional clock**. The default value is **Auto selection**.
Simulation Model (page 6)	
Simulation Libraries	Specifies the libraries needed for functional simulation by third-party tools.
Generate netlist	Specifies whether to turn on the option to generate synthesis area and timing estimation netlist.

Table 2–1. ALTLVDS_TX Parameter Settings (Part 6 of 6)

Option	Description
Summary (page 7)	
Summary	Specifies the types of files to be generated. A gray checkmark indicates a file that is automatically generated; a green checkmark indicates an optional file. Choose from the following types of files: ■ AHDL Include file (*<function name>*.**inc**) ■ VHDL component declaration file (*<function name>*.**cmp**) ■ Quartus II symbol file (*<function name>*.**bsf**) ■ Instantiation template file (*<function name>*_**inst.v** or *<function name>*_**inst.vhd** ■ Verilog HDL block box file (*<function name>*_**bb.v**) ■ Pin Planner File (*<function name>*_**.ppf**) If you turn on the **Generate netlist** option, the file for that netlist is also available (*<function name>*_**syn.v)**.

MegaWizard Parameter Settings for the LVDS Receiver

On the **General** page (page 3) of the parameter editor, depending on the device you selected, you can configure the following options:

- Implement the SERDES circuitry in LEs (logic cells) or dedicated SERDES
- Use internal PLL or external PLL
- Use DPA mode or non-DPA mode

The selections you make on the **General** page determine the features available on the remaining pages of the parameter editor.

Table 2–2 lists the parameter settings for the LVDS receiver megafunction.

Table 2–2. ALTLVDS_RX Parameter Settings (Part 1 of 8)

Option	Description
General (page 3)	
Implement Deserializer circuitry in logic cells	Turn on this option to implement the SERDES circuitry in logic cells. The receiver starts its operation on the first fast clock edge after the PLL is locked. This option is intended for slow speeds. The byte alignment may be different from the hard SERDES implementation. Turn off this option to use the dedicated SERDES circuitry in the device. This option is supported in Arria GX, Arria II GX, Arria II GZ, HardCopy II, HardCopy III, HardCopy IV, Stratix, Stratix GX, Stratix II, Stratix II GX, Stratix III, and Stratix IV devices. This option is not supported in Stratix V devices. In Cyclone series, the SERDES circuitry is always implemented in logic cells.
Enable Dynamic Phase Alignment mode	Turn on this option to correct the skews created by the different trace lengths on the data channels routed to the device. This mode adds several ports and parameters to the megafunction instances. This option is available for Arria GX, Arria II GX, Arria II GZ, HardCopy II, HardCopy III, HardCopy IV, Stratix, Stratix GX, Stratix II, Stratix II GX, Stratix III, Stratix IV, and Stratix V devices only. Enabling the DPA mode changes the appearance of the graphic representation of the megafunction in the left-hand pane. When you turn on the DPA mode, additional ports and parameters are added to the megafunction. Depending on the selected device, the following pages are added to the parameter editor to include the additional DPA mode settings: ■ **DPA settings 1** ■ **DPA settings 2** ■ **DPA settings 3**
What is the number of channels?	The number of input channels available for the LVDS receiver. If the required number of channels is not available in the list, type the desired number in this box. For example, if the number of channels is **44**, the port created is `tx_out[43..0]`. The legal values depend on the pins available in the device. For the legal values available for your device, refer to the relevant device handbook.

Table 2–2. ALTLVDS_RX Parameter Settings (Part 2 of 8)

Option	Description
What is the deserialization factor?	Determines the number of serial input data bits that the receiver deserializes and sends to the core on a single cycle. For the valid deserialization factors for your device, refer to the relevant device handbook. For example, if the deserialization factor is **10** and the number of input channels is 1, the transmitter deserializes every 10 serial bits into 10 bits of parallel data to send to the core. If the deserialization factor is 10 and the number of channels is **44**, the port created is `rx_out[439..0]`.
Use External PLL	Turn on this option to use an external PLL to clock the SERDES receiver. When you turn on this option, the options on the **Frequency/PLL settings** page are disabled. You must use a separate PLL to provide the clocking source and make the necessary connections.You must ensure your circuit has the correct input and functionality to generate an appropriate clock frequency and is correctly connected to the LVDS receiver. For more information about instantiating the ALTPLL megafunction to generate the various clock and load enable signals, refer to "Generating Clock Signals for LVDS Interface" on page 3–18. For more information about external PLL options in Stratix II refer to *AN 409: Design Example Using the ALTLVDS Megafunction & the External PLL Option in Stratix II Devices*. When you have a deserialization factor of two, the megafunction bypasses the SERDES and implements the SERDES functionality in DDR registers. A deserialization factor of at least four is required to use the external PLL option. If you turn off this option, the megafunction automatically implements an internal PLL to clock the ALTLVDS_RX block. For Stratix and Stratix GX devices, if you implement SERDES for your LVDS transmitter using a dedicated SERDES block, you do not have the option to use an external PLL.
Use 'rx_data_reset' input port	This option is enabled when you implement the LVDS in logic cells. Turn on this option to add an input port to the megafunction. When the input port asserts, the megafunction asynchronously resets all the logic in the ALTLVDS_RX megafunction excluding the PLL.
Frequency/ PLL Settings (page 4) The options on this page are available only when you are using internal PLL	
What is the input data rate?	Specifies the data rate for the input channel of the receiver, in Mbps. For data rate ranges, refer to the specific *Device Data Sheet* chapter in the respective device handbook. This value determines the legal input clock rate values.
Specify input clock rate by	Specifies the clock frequency (`rx_inclock`) and the clock period (`inclock_period`) for the internal PLL. The legal values depend on the output data rate selected.
Use shared PLL(s) for receivers and transmitters	When you turn on this option, your LVDS receivers and transmitters can share the same PLL. Turn on this option when the LVDS receivers and transmitters use the same input clock frequency, deserialization factor, and data rates.

Table 2–2. ALTLVDS_RX Parameter Settings (Part 3 of 8)

Option	Description
Use 'pll_areset' input port	Turn on this option to control the asynchronous reset port of the PLL that the megafunction uses with this function. When other ALTLVDS blocks share the PLL with the receiver and use the `pll_areset` port, you must use this port in all megafunction instantiations and tie the signals together in the design file. If you use the `pll_areset` port only in one megafunction instance, the PLLs are not shared, and a warning appears during compilation. The PLL must be reset to set the output clock phase relationships correctly when the PLL loses lock, or if the PLL input reference clock is not stable when the device completes the configuration process. For more information, refer to "Initializing the ALTLVDS_TX and ALTLVDS_RX Megafunctions" on page 3–6.
Use 'rx_pll_enable' input port	Turn on this option to control the enable port of the fast PLL that the megafunction uses with this function. If the receiver shares the PLL with other ALTLVDS blocks, and uses the `rx_pll_enable` port, you must use this port in all megafunction instances and tie the signal together in the design file. If you use the `rx_pll_enable` port only in one megafunction instance, the PLLs are not shared and a warning appears during compilation.
Use 'rx_locked' output port	Turn on this option to monitor the lock status of the PLL. The status of the lock port is identical for the transmitter and the receiver when the megafunctions use shared PLLs. In this case, monitor the lock output from the receiver megafunction.
What is the clock resource used for 'rx_outclock'?	Specifies the clock resource type fed from the `rx_outclock` port. Legal values are **Auto selection** (the compiler determines the type), **Global clock**, and **Regional clock**. The default value is **Auto selection**.
What is the phase alignment of 'rx_in' with respect to 'rx_inclock'?	Determines the phase alignment of the data that the receiver core receives with respect to the `rx_inclock` signal. Available values are **0.00**, **22.50**, **45.00**, **67.50**, **90.00**, **112.50**, **135.00**, **157.50**, **180.00**, **202.50**, **225.00**, **247.50**, **270.00**, **292.50**, **315.00**, and **337.50**. This option is only available if you turn of the DPA mode.
Use source-synchronous mode of the PLL	Turn on this option to ensure that the megafunction instance makes the required phase adjustment to guarantee a consistent relationship between the clock and the data, at the capture register and at the pin. Always turn on this option, unless you have performed all of the necessary phase adjustments manually. Altera recommends that you turn on this option when you use non-dedicated SERDES schemes. This option is only available when you implement the SERDES in LEs.
Align clock to center of data window at capture point	Turn on this option to add a phase shift of 90° to the clock, which center-aligns the clock in the data. This option is only available for Arria GX, Cyclone II, Stratix II GX, Stratix II, and HardCopy II devices when you implement the SERDES in logic cells.
Enable self-reset on lost lock in the PLL	Turn on this option to reset the PLL automatically when the PLL loses lock. This option is only available for Arria II GX, Arria II GZ, HardCopy III, HardCopy IV, Stratix III, Stratix IV, Cyclone III and Cyclone IV devices when you implement the SERDES in logic cells.

Table 2–2. ALTLVDS_RX Parameter Settings (Part 4 of 8)

Option	Description
Enable FIFO for DPA channels	The phase-compensation FIFO buffer synchronizes parallel data to the global clock domain of the core. This option is only available in Stratix GX devices when you turn on the DPA mode.
DPA Settings 1 (page 5) The options on this page are available when you turn on the DPA mode.	
Use 'rx_divfwdclk' output port and bypass the DPA FIFO	Turn on this option to divide the DPA clock by the deserialization factor and then forward the DPA clock to the core. The DPA clock drives the bit-slip and alignment circuitry, bypassing the FIFO. Turn on this option for soft-CDR mode. This option is available in Arria II GX, Arria II GZ, HardCopy III, HardCopy IV, Stratix III, Stratix IV, and Stratix V devices only.
What is the simulated recovered clock phase drift?	Models a phase drift in the recovered clock. Clock phase drift is expressed as the equivalent number of full clock cycles of drift for every parts per million (PPM) clock cycles . The value for this option can be positive, negative or zero.
Use 'rx_dpll_enable' input port	Enables the path through the DPA circuitry. The option supports dynamic, channel-by-channel control of the DPA circuitry. To enable the DPA circuitry for a channel, set the port for the target channel to `1`. If this port is not used, the Quartus II software enables all of the channels.
Use 'rx_dpll_hold' input port	Prevents the DPA circuitry from switching to a new clock phase on the target channel. Each DPA block monitors the phase of the incoming data stream continuously and selects a new clock phase when needed. When this port is held high, the selected channels hold their current phase setting.
Use 'rx_fifo_reset' input port	Resets the FIFO buffer between the DPA circuit and the data alignment circuit. The FIFO buffer holds the data passing between the DPA and the LVDS clock domains. When this port is held high, the FIFOs in the selected channels are reset. This option is available only if you turn off the **Use 'rx_divfwdclk' output port and bypass the DPA FIFO** option. For more information, refer to "Resetting the DPA" on page 3–7.
DPA Settings 2 (page 6) The options on this page are available when you turn on the DPA mode.	
Use 'rx_reset' input port	Resets all components of the DPA circuit. You must retrain the DPA circuit after this port resets the DPA circuitry. For more information, refer to "Resetting the DPA" on page 3–7.
Automatically reset the bit serial FIFO when 'rx_dpa_locked' rises for the first time	Specifies when the bit-serial FIFO resets for DPA circuit. This option is only available in Stratix II, Arria GX, and HardCopy II devices.
User explicitly resets the bit serial FIFO through 'rx_reset'	When you turn on the `rx_reset` port, the ALTLVDS_RX parameter editor allows you to choose whether or not to automatically reset the bit-serial FIFO when `rx_dpa_locked` signal rises for the first time. This is a useful feature because it keeps the synchronizer FIFO in reset until the DPA locks. This option is only available in Stratix II, Arria GX, and HardCopy II devices.

Table 2–2. ALTLVDS_RX Parameter Settings (Part 5 of 8)

Option	Description
Use 'rx_dpa_locked' output port	The DPA block samples the data on one of eight phase clocks with a 45° resolution between phases. This port lets you monitor the status of the DPA circuit and determine when it has locked onto the phase closest to the incoming data phase. The `rx_dpa_locked` port behaves differently for various device families. After the megafunction asserts the `rx_dpa_locked signal is` upon initial lock, the `rx_dpa_locked` signal does not deassert in Stratix III, Stratix IV, Stratix V, HardCopy III, HardCopy IV, and Arria II GX unless explicitly reset using `rx_reset` or `rx_dpa_lock_reset`. In Stratix GX, Stratix II, HardCopy II, and Arria GX, the `rx_dpa_locked` signal toggles depending on how the next two settings are selected. After power up or reset, the `rx_dpa_locked` signal is asserted after the DPA circuitry acquires an initial lock to the optimum phase. You must not use the `rx_dpa_locked` signal to validate the integrity of the LVDS link. Use error checkers (for example, CRC or DIP4) to validate the integrity of the LVDS link. For more information about the device-specific `rx_dpa_locked` behavior, refer to the *Stratix III Device Family Errata Sheet* and the *Stratix IV Device Family Errata Sheet*.
When phase alignment circuitry switches to a new phase	DPA deasserts when the phase alignment circuitry switches to a new phase. This option is only available in Stratix II, HardCopy II, and Arria GX devices.
When there are two phase changes in the same direction	The `rx_dpa_locked` signal deasserts after the DPA switches two phases in the same direction. This option is only available in Stratix II, HardCopy II, and Arria GX devices.
Use 'rx_dpa_lock_reset' input port	Resets the DPA lock circuitry. For more information, refer to "Initialization and Reset" on page 3–6.
Use a DPA initial phase selection of	Turn on this option to select the initial phase setting. Specifies whether to turn on this option and its value. Simulation honors this phase selection in simulating the forwarded clock. This option is available for Arria II GX, Arria II GZ, HardCopy III, HardCopy IV, Stratix III, Stratix IV, and Stratix V devices only.
Align DPA to rising edge of data only	Turn on this option to align the DPA to the rising edge of the data only or turn of this option to align the DPA to both the rising and falling edges of the data. This option is available for Arria II GX, Arria II GZ, HardCopy III, HardCopy IV, Stratix III, Stratix IV, and Stratix V devices only.
DPA Settings 3 (page 7) The options on this page are available when you turn on the DPA mode.	
Enable PLL Calibration	Turn on this option to phase-shift the PLL outputs when the `dpa_pll_cal_busy` signal is high. The default setting is **OFF**. This option is available for Arria II GZ, HardCopy III, HardCopy IV, Stratix III, Stratix IV, and Stratix V devices only. When you enable PLL calibration, you cannot merge the PLL with other PLLs. For more information, refer to "DPA PLL Calibration" on page 3–4.

Table 2–2. ALTLVDS_RX Parameter Settings (Part 6 of 8)

Option	Description
Use 'dpa_pll_recal' input port	This port recalibrates the PLL without resetting the DPA. This option is available for Arria II GZ, HardCopy III, HardCopy IV, Stratix III, Stratix IV, and Stratix V devices only.
What is the input data rate?	Specifies the data rate for the input channel of the receiver, in Mbps. For data rate ranges, refer to the specific *Device Data Sheet* chapter in the respective device handbook. This value determines the legal input clock rate values.
Receiver Settings (page 8)	
Register outputs	Turn on this option to implement soft-CDR receiver modes in standard mode. In standard mode, the outputs of the receiver are registered by the `rx_outclock` signal. Turn off this option if you do not want to register the receiver outputs. In no output register mode, you must register the output registers in the design logic that is fed by the receiver, and then specify a **Source Multiply** assignment from the receiver to the output registers with a value equal to the deserialization factor. For more information, refer to "Soft-CDR Mode" on page 3–1.
Use 'rx_cda_reset' input port	The port resets the data alignment circuitry, restoring the latency bit counter to zero. This option is available only if you turn on the **Use 'rx_channel_data_align' input port** option. This option is available only if you use dedicated SERDES block.
Use 'rx_cda_max' output port	Indicates when the rollover point is reached in the data alignment circuit. This port is available only if you turn on the **Use 'rx_channel_data_align' input port** option. This option is available only if you use a a dedicated SERDES block.
After how many pulses does the data alignment circuitry restore the serial latency back to 0?	Specifies, in pulses, when the DPA circuitry restores the serial data latency to 0. The value does not have to be the same as the deserialization factor, but set the value to the deserialization factor to make the rollover occur for every deserialization factor. The available values for this option range from 1 to 11. This option is available only if you use a dedicated SERDES block.

Table 2–2. ALTLVDS_RX Parameter Settings (Part 7 of 8)

Option	Description
Align data to the rising edge of clock	When you turn on this option, the data path is registered on the positive edge of the `diffioclk` signal (also referred to as the LVDS clock). When you turn off this option, the data path is registered on the negative edge of the `diffioclk` signal. This option is available only if you use a dedicated SERDES block, and is available only in non-DPA mode. This option changes the phase that captures the received data by 180°. Use caution when you turn off this option. The phase shift of the capture clock is automatically set according to the setting for the **What is the phase alignment of 'rx_in' with respect to the rising edge of 'rx_inclock'? (in degrees)** option. Changing the phase of the capture clock can lead to data corruption. If you turn off this option, the LVDS data is aligned to the falling edge of the clock. For an example of the usage, if you have two receivers interface with identical parameters except for the `rx_in` signal relationship to the `rx_inclock` signal, and you want to merge PLLs, one interface must have a 0° (rising edge) alignment, and the second interface must have a 180° (falling edge) alignment. You can only merge the PLLs when they have the same clock and phase settings; both must be set with the same alignment. You can set both receivers to be 0° aligned, and turn off **Align data to the rising edge of clock** on the 180° aligned interface.
Use 'rx_coreclk' input port	This option is enabled when the LVDS is implemented in logic. When you turn on this option, it adds an input port, which when asserted performs an asynchronous reset of all the logic in the ALTLVDS_RX megafunction excluding the PLL.
Use 'rx_channel_data_align' input port	Turn on this option to control bit insertion on a channel-by-channel basis to align the word boundaries of the incoming data. The data slips one bit for every pulse on the `rx_channel_data_align` port. This option is available only if you use a dedicated SERDES block. You can use control characters in the data stream so your logic can have a known pattern to search for. You can compare the data received for each channel, compare to the control character you are looking for, then pulse the `rx_channel_data_align` port as required until you successfully receive the control character. To use this port, you must meet the following requirements: ■ The minimum pulse width is one period of the parallel clock in the logic array (`rx_outclock`). ■ The minimum low time between pulses is one period of the parallel clock. ■ There is no maximum high or low time. ■ Valid data is available two parallel clock cycles after the rising edge of `rx_channel_data_align` signal. For more information, refer to "Aligning the Word Boundaries" on page 3–8.
Enable independent bitslip controls for each channel	Turn on this option to allow an independent `rx_data_align` signal for each channel that independently control the bit slip capability of each channel. This option is available if you implement the SERDES in LEs.

Table 2–2. ALTLVDS_RX Parameter Settings (Part 8 of 8)

Option	Description
Add extra register for 'rx_data_align' input port	Turn on this option to enable the synchronization register of the receiver. If you turn on this option, you can also add an extra register to register the `rx_data_align` port using the `rx_outclock` port. This option is available if you implement the SERDES in LEs.
Use 'rx_data_align_reset' input port	Turn on this option to create the reset port for the bit-slip circuitry. This option is available if you implement the SERDES in LEs.
Which output synchronization buffer implementation should be used?	Specifies where to implement the buffer. The values are **Use RAM Buffer**, **Use Multiplexer and synchronization register**, and **Use logic element based RAM buffer**. A value of **Use Multiplexer and synchronization register** implements a multiplexer instead of a buffer. A value of **Use RAM Buffer** implements a buffer in RAM blocks. A value of **Use logic element based RAM buffer** implements a buffer in logic elements. The **Use RAM Buffer** and **Use logic element based RAM buffer** values use more logic, but result in the correct word alignment. If omitted, the default value is **Use RAM Buffer**.
Simulation Model (page 9)	
Simulation Libraries	Specifies the libraries needed for functional simulation by third-party tools.
Generate netlist	Turn on this option to generate synthesis area and timing estimation netlist.
Summary (page 10)	
Summary	Specifies the types of files to be generated. A gray checkmark indicates a file that is automatically generated; a green checkmark indicates an optional file. Choose from the following types of files: ■ AHDL Include file (*<function name>*.**inc**) ■ VHDL component declaration file (*<function name>*.**cmp**) ■ Quartus II symbol file (*<function name>*.**bsf**) ■ Instantiation template file (*<function name>*_**inst.v** or *<function name>*_**inst.vhd** ■ Verilog HDL block box file (*<function name>*_**bb.v**) ■ Pin Planner File (*<function name>*_.**ppf**) If you turn on the **Generate netlist** option, the file for that netlist is also available (*<function name>*_**syn.v)**.

Command Line Interface Parameters

Expert users can choose to instantiate and parameterize the megafunction through the command-line interface using the clear box generator command. This method requires you to have command-line scripting knowledge.

For more information about using the clear box generator, refer to "Clear Box Generator" on page A–1.

Table 2–3 lists the parameters for the ALTLVDS_TX megafunction.

Table 2–3. Parameters for the ALTLVDS_TX Megafunction (Part 1 of 5)

Parameter	Type	Description
common_rx_tx_pll	String	Specifies whether the compiler uses the same PLL for both the LVDS receiver and the LVDS transmitter, or multiple LVDS receivers, or multiple LVDS transmitters, or both. You can use common PLLs if the same input clock source, same deserialization factor, same pll_areset source, and same data rates are used. The values are ON and OFF. If omitted, the default value is ON. Only available for Arria GX, Arria II GX, Arria II GZ, Cyclone, Cyclone II, Cyclone III, Cyclone IV, HardCopy II, HardCopy III, HardCopy IV, Stratix, Stratix GX, Stratix II, Stratix II GX, Stratix III, Stratix IV, and Stratix V devices.
coreclock_divide_by	Integer	Specifies the core clock output frequency to either be core clock or core clock divided by 2. The value are 1 or 2 This parameter is only available when using odd SERDES factors. When using a divide-by factor of 1, fewer device resources are used, but you may not be able to achieve timing at higher data rates. Altera recommends using a divide-by factor of two for higher data rates. This parameter is available for the Cyclone series.
deserialization_factor	Integer	Specifies the number of bits per channel. The following is the device support and its values with normal mode: ■ Arria II GX and Arria II GZ: 1 to 10 ■ Arria GX: 1, 2, 4, to 10 ■ Cyclone, Cyclone II, Cyclone III, and Cyclone IV: 1, 2, 4, to 10 ■ HardCopy II, HardCopy III, and HardCopy IV: 1, 2, 4, to 10 ■ Stratix and Stratix GX: 1, 2, 4, 7, 8, to 10 ■ Stratix II and Stratix II GX: 1, 2, 4, to 10 ■ Stratix III, Stratix IV, and Stratix V: 1 to 10 Arria GX, Arria II GX, Arria II GZ, HardCopy II, HardCopy III, HardCopy IV, Stratix, Stratix GX, Stratix II, Stratix II GX, Stratix III, Stratix IV, and Stratix V devices have the values of 1, 2, 4, to 10 with SERDES using logic cells.
enable_clk_latency	String	Specifies whether the PLLs use clock latency. The values are ON and OFF.

Table 2–3. Parameters for the ALTLVDS_TX Megafunction (Part 2 of 5)

Parameter	Type	Description
implement_in_les	String	Specifies whether to implement SERDES circuitry in logic cells, which allows the circuitry to behave similarly to Stratix LVDS circuitry. You must use the implement_in_les parameter for SERDES functions that require data rates that are lower than the dedicated circuitry. The values are ON and OFF. For Cyclone, Cyclone II, Cyclone III, and Cyclone IV devices, the value is always ON. Available for all devices except the MAX series. The ALTLVDS_TX megafunction starts its operation at the first rising edge of the fast clock, after the PLL has locked. This is intended for slow speeds and the bit alignment might be different from a dedicated SERDES implementation.
inclock_data_alignment	String	Specifies the phase alignment of the tx_in[] and tx_inclock input ports in terms of the tx_inclock frequency. The clock phase alignment for the inclock_data_alignment parameter specifies the positive phase shift needed for the clock for alignment with the data. The following are the parameter values and its values in degrees (°): ■ EDGE_ALIGNED: 0° ■ 45_DEGREES: 45° ■ 90_DEGREES: 90° ■ 135_DEGREES: 135° ■ CENTER_ALIGNED: 180° ■ 225_DEGREES: 225° ■ 270_DEGREES: 270° ■ 315_DEGREES: 315° If omitted, the default value is EDGE_ALIGNED. Available for Arria GX, Arria II GX, Arria II GZ, Cyclone, Cyclone II, Cyclone III, Cyclone IV, HardCopy II, HardCopy III, HardCopy IV, Stratix, Stratix GX, Stratix II, Stratix II GX, Stratix III, Stratix IV, and Stratix V devices.
inclock_period	Integer	Specifies the input clock either by frequency (MHz in the parameter editor) or period (ps in HDL code). This parameter is required when the external PLL option is not used.
number_of_channels	Integer	Specifies the number of LVDS channels.

Table 2–3. Parameters for the ALTLVDS_TX Megafunction (Part 3 of 5)

Parameter	Type	Description
outclock_alignment	String	Specifies the alignment of tx_outclock with respect to the V_{CO} of a fast PLL. The clock phase alignment for the outclock_alignment parameter is data leading. Values are: ■ EDGE_ALIGNED: 0° ■ 45_DEGREES: 45° ■ 90_DEGREES: 90° ■ 135_DEGREES: 135° ■ CENTER_ALIGNED:180° ■ 225_DEGREES: 225° ■ 270_DEGREES: 270° ■ 315_DEGREES: 315° If omitted, the default value is EDGE_ALIGNED. Available for all devices excluding the MAX series.
outclock_divide_by	Integer	Specifies the period of the tx_outclock port as [INCLOCK_PERIOD * OUTCLOCK_DIVIDE_BY] and the frequency of the tx_outclock port as [INCLOCK_PERIOD/OUTCLOCK_DIVIDE_BY]. The default value for this parameter is the value of the deserialization_factor parameter. Only available for Arria GX, Arria II GX, Arria II GZ, Cyclone, Cyclone II, Cyclone III, Cyclone IV, HardCopy II, HardCopy III, HardCopy IV, Stratix, Stratix GX, Stratix II, Stratix II GX, Stratix III, Stratix IV, and Stratix V devices. For more information about the DESERIALIZATION_FACTOR and outclock_divide_by values, refer to Table 2–4 on page 2–19.
outclock_duty_cycle	Integer	Specifies the external clock timing constraints. A value of 50 is not supported in the outclock_duty_cycle parameter when the following is true: ■ DESERIALIZATION_FACTOR value is 5, 7, or 9 ■ OUTCLOCK_DIVIDE_BY value is equal to the value of DESERIALIZATION_FACTOR ■ OUTCLOCK_MULTIPLY_BY value is 2 This is always true for Cyclone II, Cyclone III, Cyclone IV devices, and true for Stratix II, Stratix III, Stratix IV, and Stratix V devices when the implement_in_les parameter value is set to ON.
outclock_multiply_by	Integer	Specifies the multiplication factor. The values are 1 and 2. If omitted, the default value is 1. Only available for Cyclone, Cyclone II, Stratix, Stratix GX, and Stratix II devices.

Table 2–3. Parameters for the ALTLVDS_TX Megafunction (Part 4 of 5)

Parameter	Type	Description
outclock_phase_shift	Integer	Specifies, in picoseconds (ps), the phase shift of the output clock relative to the input clock. Phase shifts of 0.0, 0.25, 0.5, or 0.75 times the input period (0, 90, or 270°) are implemented precisely. The allowed range for the phase shift is between 0 ps and 1 input clock period. If the phase shift is outside this range, the compiler adjusts it to fall within this range. For other phase shifts, the compiler chooses the closest allowed value. If omitted, the default value is 0.
outclock_resource	String	Specifies the clock resource type to use with the tx_coreclock port. The values are AUTO, REGIONAL CLOCK, and GLOBAL CLOCK. If omitted, the default value is AUTO. Only available for Arria GX, Arria II GX, Arria II GZ, Cyclone, Cyclone II, Cyclone III, Cyclone IV, HardCopy II, HardCopy III, HardCopy IV, Stratix, Stratix GX, Stratix II, Stratix II GX, Stratix III, Stratix IV, and Stratix V devices.
output_data_rate	Integer	Specifies the data rate out of the PLL. The multiplication value for the PLL is OUTPUT_DATA_RATE/INCLOCK_PERIOD. Only available for Arria GX, Arria II GX, Arria II GZ, Cyclone, Cyclone II, Cyclone III, Cyclone IV, HardCopy II, HardCopy III, HardCopy IV, Stratix, Stratix GX, Stratix II, Stratix II GX, Stratix III, and Stratix IV devices.
pll_bandwidth_type	String	Specifies the loop filter bandwidth control setting on the PLL. The values are LOW, MEDIUM, and HIGH. This parameter is only available for the Stratix II device.
pll_self_reset_on_loss_lock	String	The values are ON and OFF. If omitted, the default value is OFF. When this parameter is enabled, the PLL is reset when it loses lock. This parameter is valid for Cyclone III, Cyclone IV, Stratix, Stratix II GX, Stratix III, and Stratix IV, and Stratix V devices when the implement_in_les parameter is set is ON.
registered_input	String	Indicates whether the tx_in[] port is registered. The values are ON, OFF, TX_INCLOCK, and TX_CORECLOCK. If omitted, the default value is ON when using the tx_coreclock port to register the data in logic elements. The TX_INCLOCK and TX_CORECLOCK values are available for Arria GX, Arria II GX, Arria II GZ, Cyclone, Cyclone II, Cyclone III, Cyclone IV, HardCopy II, HardCopy III, HardCopy IV, Stratix, Stratix GX, Stratix II, Stratix II GX, Stratix III, Stratix IV, and Stratix V devices. If the registered_input parameter is set to OFF, you must pre-register the tx_in[] port in the logic feeding the transmitter.

Table 2–3. Parameters for the ALTLVDS_TX Megafunction (Part 5 of 5)

Parameter	Type	Description
use_external_pll	String	Specifies whether the LTLVDS_TX megafunction generates a PLL or connect to a user-specified PLL. Altera recommends instantiating the external PLL with the parameter editor. Only available for Arria GX, Arria II GX, Arria II GZ, Cyclone, Cyclone II, Cyclone III, Cyclone IV, HardCopy II, HardCopy III, HardCopy IV, Stratix, Stratix GX, Stratix II, Stratix II GX, Stratix III, Stratix IV, and Stratix V devices.
use_no_phase_shift	String	When set to OFF, a phase shift of 90° is added to the clock to center the clock in the data. Use this parameter when the implement_in_les parameter value is set to ON for Cyclone II, Stratix II, Stratix III, Stratix IV, and Stratix V devices. The values are ON and OFF. If omitted, default value is ON. Altera recommends setting this parameter to OFF unless you have completed a phase adjustment.

Table 2–4 lists the deserialization factor and outclock divide by values.

Table 2–4. DESERIALIZATION_FACTOR and OUTCLOCK_DIVIDE_BY Values

Devices	DESERIALIZATION_FACTOR Value	OUTCLOCK_DIVIDE_BY Value
Arria GX, Arria II GX, Arria II GZ, HardCopy II, HardCopy III, HardCopy IV, Stratix II, Stratix II GX, Stratix III, Stratix IV, and Stratix V	4	2
		4
	5	5
	6	2
		6
	7	7
	8	2
		4
		8
	9	9
	10	2
		10
Stratix and Stratix GX	4	2
		4
	7	7
	8	2
		4
		8
	10	2
		10

Table 2–4. DESERIALIZATION_FACTOR and OUTCLOCK_DIVIDE_BY Values

Devices	DESERIALIZATION_FACTOR Value	OUTCLOCK_DIVIDE_BY Value
Cyclone, Cyclone II, Cyclone III, and Cyclone IV	4	2
		4
		8
	5	2
		5
		10
	6	2
		6
		12
	7	2
		7
		14
	8	2
		4
		8
		16
	9	2
		9
		18
	10	2
		4
		10
		20

Table 2–5 lists the parameters for the ALTLVDS_RX megafunction.

Table 2–5. Parameters for ALTLVDS_RX Megafunction (Part 1 of 6)

Parameter	Type	Description
buffer_implementation	String	Specifies where to implement the buffer. The values are MUX, RAM, and LES. A value of MUX implements a multiplexer instead of buffer implementation. A value of RAM implements a buffer in RAM blocks. A value of LES implements a buffer in logic elements. The RAM and LES values use more logic, but result in the correct word alignment. If omitted, the default value is RAM. To use the buffer_implementation parameter, the implement_in_les parameter must be turned ON. You can also use the buffer_implementation parameter with deserialization factors of 5, 7, or 9 only.
common_rx_tx_pll	String	Specifies whether the compiler uses the same PLL for both the LVDS receiver and the LVDS transmitter, or multiple LVDS receivers or multiple LVDS transmitters, or both. You can use common PLLs if the same input clock source, same deserialization factor, same pll_areset source, and same data rates are used. Values are ON and OFF. If omitted, the default value is ON.
data_align_rollover	Integer	Specifies, in pulses, when the DPA circuitry restores the serial data latency to 0. You must enable the rx_dpa_locked port and the enable_dpa_mode parameter if this parameter is specified. The legal integer value ranges from 1 to 11. If omitted, the default value is 4.

Table 2–5. Parameters for ALTLVDS_RX Megafunction (Part 2 of 6)

Parameter	Type	Description
deserialization_factor	Integer	Specifies the number of bits per channel. The values of this parameter for each supported device in normal mode are as follows: ■ Arria II GX and Arria II GZ: 1 to 10. ■ Arria GX: 1, 2, 4, to 10 ■ Cyclone series: 1, 2, 4, to 10 ■ HardCopy II, HardCopy III, and HardCopy IV: 1, 2, 4, to 10 ■ Stratix and Stratix GX: 1, 2, 4, 7, 8, to 10 ■ Stratix II and Stratix II GX: 1, 2, 4, to 10 ■ Stratix III, Stratix IV, and Stratix V: 1, 2, 3, 4, to 10 Arria GX, Arria II GX, Arria II GZ, HardCopy II, HardCopy III, HardCopy IV, Stratix, Stratix II, Stratix II GX, Stratix III, Stratix IV, and Stratix V have the values of 1, 2, 4, to 10 with SERDES using logic cells. The values of this parameter for each supported device in DPA mode are as follows: ■ Arria II GX and Arria II GZ: 1 to 10 ■ Arria GX: 1, 2, 4, to 10 ■ HardCopy II, HardCopy III, and HardCopy IV: 1, 2, 4, to 10 ■ Stratix GX: 8 and 10 ■ Stratix II and Stratix II GX: 1, 2, 4, to 10 ■ Stratix III, Stratix IV, and Stratix V: 1 to 10
dpa_initial_phase_value	Integer	Specifies the initial phase value. The values are 0 through 7. If the parameter value is set to OFF, the dpa_initial_phase_value parameter is set to 0.
enable_dpa_calibration	String	The values are ON and OFF. The default value is ON. Set this parameter to ON to phase shift the PLL outputs when the dpa_pll_cal_busy signal is high.
enable_dpa_align_to_rising_edge_only	String	Specifies that the DPA aligns to the rising edge of data only. Values are ON and OFF. If omitted, the default value is OFF. A value of OFF specifies that the DPA aligns to both the rising and falling edge of data.
enable_dpa_fifo	String	Indicates whether the DPA FIFO buffer is enabled for this channel.You must enable the rx_dpa_locked port and enable_dpa_mode parameter if this parameter is specified. The values are ON and OFF. If omitted, the default value is ON. This parameter is available for Stratix GX devices in DPA mode only.

Table 2–5. Parameters for ALTLVDS_RX Megafunction (Part 3 of 6)

Parameter	Type	Description
enable_dpa_initial_phase_selection	String	Specifies whether the dpa_initial_phase_value parameter is enabled. The values are ON and OFF. If omitted, the default value is OFF. When set to OFF, the dpa_initial_phase_value parameter value is set to 0.
enable_dpa_mode	String	Turns on DPA mode. The values are ON and OFF. If omitted, the default value is OFF.
enable_dpa_pll_calibration	String	The values are ON and OFF. The default value is OFF. Set this parameter to ON or OFF if you are instantiating the ALTLVDS_RX megafunction in DPA mode with PLL calibration.
enable_soft_cdr_mode	String	Specifies whether the rx_divfwdclk port is used. When set to ON, the rx_divfwdclk port is driven by the DPA clock, and then it is divided down by the deserialization factor. When set to ON, the DPA FIFO is bypassed and rx_fifo_reset and reset_fifo_on_first_lock are ignored. The values are ON and OFF. If omitted, the default is OFF.
implement_in_les	String	Specifies whether to implement SERDES circuitry in logic cells, which allows the circuitry to behave similar to Stratix LVDS circuitry. Use the implement_in_les parameter for SERDES functions that require data rates that are lower than the dedicated circuitry. Values are ON and OFF. Note that the receiver megafunction starts capturing the LVDS stream at the first rising edge of the fast clock, after the PLL has locked. This is intended for slow speeds and the bit alignment may be different from a hard SERDES implementation.
inclock_data_alignment	String	Specifies the phase alignment of the rx_in and rx_inclock input ports in terms of the rx_inclock frequency. The clock phase alignment for the inclock_data_alignment parameter specifies the positive phase shift needed for the clock for alignment with the data. The following are the parameter values and the corresponding phase shifts in degrees (°): ■ EDGE_ALIGNED: 0° ■ 45_DEGREES: 45° ■ 90_DEGREES: 90° ■ 135_DEGREES: 135° ■ CENTER_ALIGNED: 180° ■ 225_DEGREES: 225° ■ 270_DEGREES: 270° ■ 315_DEGREES: 315° If omitted, the default value is EDGE_ALIGNED.

Table 2–5. Parameters for ALTLVDS_RX Megafunction (Part 4 of 6)

Parameter	Type	Description
inclock_period	Integer	Specifies the period or frequency of the rx_inclock port. The default time unit is an integer in picoseconds (ps). In AHDL designs only, strings, such as 50.5 MHz, are acceptable.
inclock_phase_shift	Integer	Specifies a phase shift in 15° increments.
input_data_rate	Integer	Specifies the data rate into the PLL. The multiplication value for the PLL is INPUT_DATA_RATE/INCLOCK_PERIOD.
lose_lock_on_one_change	String	Specifies when the DPA circuitry should lose lock. You must enable the rx_dpa_locked port and the enable_dpa_mode parameter if this parameter is specified. Values are ON and OFF. If omitted, the default value is ON.
number_of_channels	Integer	Specifies the number of LVDS channels.
outclock_resource	String	Specifies the clock resource type to use with the rx_outclock port. The values are AUTO, Regional Clock, and Global Clock. If omitted, the default value is AUTO.
pll_operation_mode	String	Specifies the source synchronous mode for Cyclone II and Stratix II device LE PLLs. The values are NORMAL and SOURCE_SYNCHRONOUS. If omitted, the default value is NORMAL.
pll_self_reset_on_loss_lock	String	The values are ON and OFF. If omitted, the default value is OFF. When this parameter is enabled, the PLL is reset when it loses lock. This parameter is valid for Cyclone III, Cyclone IV, Stratix III, Stratix IV, and Stratix V devices when the implement_in_les parameter is set to ON.
port_rx_channel_data_align	String	Edge-sensitive bit-slip control signal. Each rising edge on this signal causes the data re-alignment circuitry to shift the word boundary by one bit. The minimum pulse width requirement is one parallel clock cycle. There is no maximum pulse width requirement. Determines if the rx_channel_data_align port is used or unused. The values are PORT_USED, PORT_UNUSED, and PORT_CONNECTIVITY. When set to PORT_USED, the rx_channel_data_align port is used. When set to PORT_UNUSED, the rx_channel_data_align port is unused. When set to PORT_CONNECTIVITY, the Quartus II software checks the connectivity of the rx_channel_data_align port to determine port usage. If omitted, the default value is PORT_CONNECTIVITY.

Table 2–5. Parameters for ALTLVDS_RX Megafunction (Part 5 of 6)

Parameter	Type	Description
port_rx_data_align	String	Determines if the rx_align_data_reg port is used or unused. The values are PORT_USED, PORT_UNUSED, and PORT_CONNECTIVITY. When set to PORT_USED, the rx_align_data_reg port is used. When set to PORT_UNUSED, the rx_align_data_reg port is unused. When set to PORT_CONNECTIVITY, the Quartus II software checks the connectivity of the rx_align_data_reg port to determine port usage. If omitted, the default value is PORT_CONNECTIVITY.
registered_data_align_input	String	Specifies whether the rx_align_data_reg port is registered. The values are ON and OFF. If omitted, the default is ON. Only available for Stratix and Stratix GX devices.
registered_output	String	Indicates whether the rx_out[] port should be registered. The values are ON and OFF. If omitted, the default is ON. If the registered_output parameter is set to OFF, you should pre-register the rx_out[] port in the logic feeding the receiver.
reset_fifo_at_first_lock	String	Specifies when the bit-serial FIFO resets. Normally, the bit-serial FIFO is reset when the DPA circuitry is locked or reset through the rx_reset port. The rx_dpa_locked port and the enable_dpa_mode parameter must be enabled if this parameter is specified. The values are ON and OFF. If omitted, the default value is ON. Only available for Arria GX, Arria II GX, Arria II GZ, Stratix II and Stratix II GX devices.
rx_align_data_reg	String	Controls byte alignment circuitry. If omitted, the default value is RISING_EDGE. This port is available for Stratix III devices only.
use_coreclock_input	String	Indicates whether the rx_coreclk port or the clock from PLL is used as the non-peripheral clock. You must connect the rx_coreclk port if you turn on this parameter. The values are ON and OFF. If omitted, the default value is OFF. This parameter is only available for Stratix GX devices. This parameter is available in DPA mode only.

Table 2–5. Parameters for ALTLVDS_RX Megafunction (Part 6 of 6)

Parameter	Type	Description
use_external_pll	String	Specifies whether the ALTVDS_RX megafunction generates a PLL or connect to a user-specified PLL. Altera recommends instantiating the external PLL with the parameter editor. Only available for Arria GX, Arria II GX, Arria II GZ, Cyclone, Cyclone II, Cyclone III, Cyclone IV, HardCopy II, HardCopy III, HardCopy IV, Stratix, Stratix GX. Stratix II, Stratix II GX, Stratix III, Stratix IV, and Stratix V devices. This option is not available when using deserialization factor of 1 and 2 in the Cyclone series.
use_no_phase_shift	String	The values are ON and OFF. If omitted, default value is ON. Altera recommends setting this parameter to OFF unless you have done a phase adjustment. When set to OFF, a phase shift of 90° is added to the clock to center the clock in the data. Use this parameter when the pll_operation_mode parameter value is set to SOURCE_SYNCHRONOUS for Cyclone II and Stratix II devices.

This section describes the various receiver modes and features, the functionality of the ports and the timing analysis of the megafunctions.

Receiver Modes

The ALTLVDS_RX megafunction supports the following receiver modes:

- DPA mode
- Non-DPA mode
- Soft-CDR mode

The physical medium connecting the transmitter and receiver LVDS channels may introduce a 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 receiver modes provide different options to overcome skew between the source-synchronous clock (non-DPA, DPA) /reference clock (soft-CDR) and the serial data.

DPA Mode

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.

Non-DPA Mode

Non-DPA mode allows you to statically select the optimal phase between the source-synchronous clock and the received serial data to compensate for the skew.

Soft-CDR Mode

The soft-CDR mode removes the clock from the clock-embedded data, a capability required for the serial gigabit media independent interface (SGMII) protocol. The PLL requires a reference clock, but the reference clock need not be source-synchronous with the data.

Clock Forwarding

In soft-CDR mode, the ALTLVDS_RX megafunction divides the DPA clock and the data by the deserialization factor. The newly divided clock signal, `rx_divfwdclk`, is then placed on the PCLK network, which carries the clock signal to the core. In supported devices, each LVDS channel can be in soft-CDR mode and can drive the core using the PCLK network. The clock forwarding feature is supported in Arria II GX, Arria II GZ, HardCopy III, HardCopy IV, Stratix III, Stratix IV, and Stratix V devices.

For more information about periphery clock networks for specific devices, refer to the *Clock Networks and PLLs* chapter in volume 1 of the respective device handbook.

When using soft-CDR mode, the rx_reset port must not be asserted after the rx_dpa_lock is asserted because the DPA continuously chooses new phase taps from the PLL to track parts per million (ppm) differences between the reference clock and incoming data. The parallel clock rx_outclock, generated by the left and right PLL, is also forwarded to the FPGA fabric.

For ppm tolerance specifications between the source clock and received data, refer to the appropriate device data sheet or device handbook for each device.

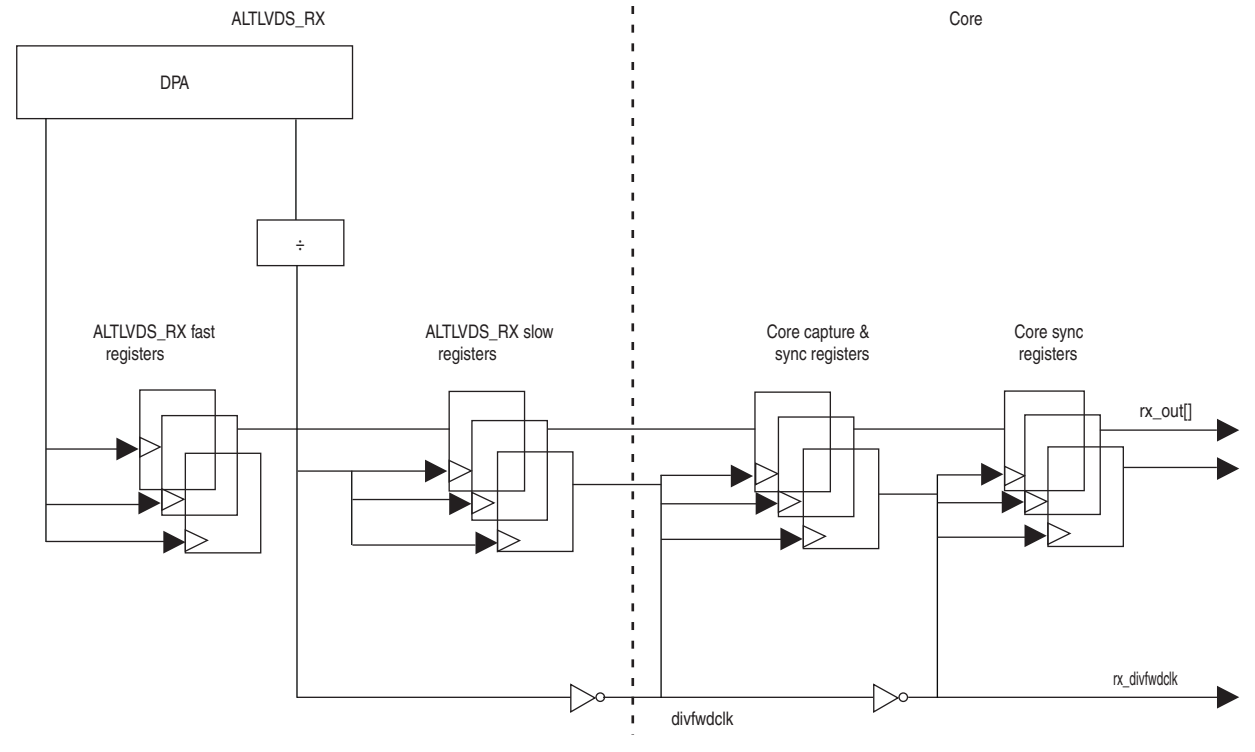
For more information about receiver modes, refer to the *High-Speed Differential I/O Interfaces* chapter in the respective device handbook.

The following two sections describe the implementation of soft -CDR mode in the ALTLVDS_RX block.

Standard Mode

Figure 3–1 shows the implementation of soft-CDR mode in standard mode. In standard mode, the first two stages of core-capture registers are created automatically by the ALTLVDS_RX parameter editor. You must clock any additional user registers from the positive edge of the rx_divfwdclk clock; using the negative edge makes it harder to meet timing, and the duty cycle is not guaranteed.

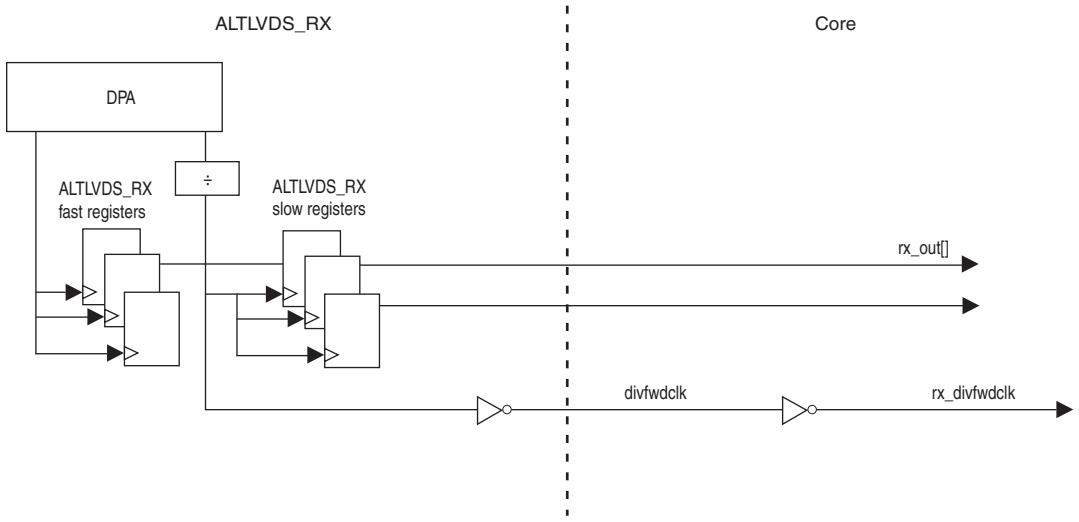
Figure 3–1. ALTLVDS_RX Block in Standard Mode



No Output Register Mode

Figure 3–2 shows the implementation of soft-CDR mode in no-output register mode. In this mode, you must create the capture registers by the user logic. To ensure even slack for both setup and hold, you must clock the first capture register stage by the falling edge of the `rx_divfwdclk` clock and clock the second stage of the registers by the rising edge of the `rx_divwdclk` clock. The register clocking method gives the equivalent implementation as the standard mode implementation, as shown in Figure 3–1.

Figure 3–2. ALTLVDS_RX Block in No Output Register Mode



DPA PLL Calibration

The following sections describe DPA PLL calibration and its effects in Stratix III, Stratix IV, Stratix IV Engineering Sample (ES), and Arria II GX devices.

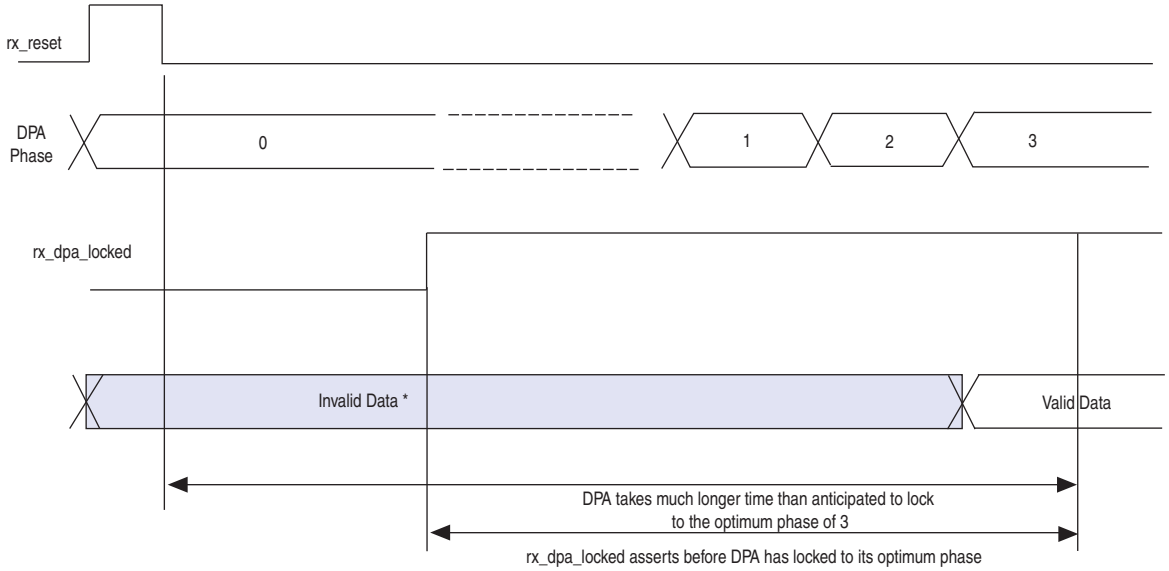
DPA PLL Calibration in Stratix III and Stratix IV ES Devices

Applications using a fixed, cyclical training pattern with sparse data transitions can cause the PLL phase to remain unchanged, which results in DPA misalignment. When DPA misaligns the DPA circuitry remains at the initial configured phase or takes a significantly longer time to lock onto the optimum phase. A non-ideal phase might result in data bit errors, even after the DPA lock signal goes high. Resetting the DPA circuit may not solve the problem.

For more information, refer to the *Stratix III DPA Misalignment* section in the *Stratix III Device Family Errata Sheet* and the *Stratix IV DPA Misalignment* section in the *Stratix IV Device Family Errata Sheet*.

Figure 3–3 shows that the DPA takes longer time to lock onto the optimum phase even after the `rx_reset` and `rx_dpa_locked` signals are asserted, resulting in data errors.

Figure 3–3. DPA Misalignment Issue



In the Quartus II software versions 9.0 and later, the DPA PLL calibration feature is added to the ALTLVDS_RX megafunction to overcome the DPA misalignment issue found in Stratix III and Stratix IV ES devices; the Stratix IV production devices are not affected. The DPA PLL calibration feature is available when the LVDS receiver is configured in DPA or soft-CDR mode. DPA PLL calibration phase-shifts the PLL outputs to induce progress in the PLL's phase-detect up and down counter and to facilitate a new phase selection.

The following events occur during the DPA PLL calibration process:

1. The ALTLVDS_RX megafunction counts 256 data transitions; the PLL calibrates the phase forward by two clocks.
2. The ALTLVDS_RX megafunction counts 256 transitions; the PLL calibrates the phase backward by two clocks so that the PLL timing returns to normal.
3. The ALTLVDS_RX megafunction counts 256 data transitions, and then asserts the `rx_dpa_locked` signal.

To enable the DPA PLL calibration feature, refer to "MegaWizard Parameter Settings for the LVDS Receiver" on page 2–7.

For more information about DPA lock time specification, refer to the *Device Data Sheet* chapter in the respective device handbook.

DPA PLL Calibration in Arria II GZ and Stratix IV Devices and Later

Starting with the Arria II GZ device and the production versions of Stratix IV devices, DPA PLL calibration is implemented for each receiver channel independently using delay elements in the LVDS receiver path. Anytime the `rx_reset` port is deasserted for a receiver channel, the DPA circuitry is reset, and the calibration and locking process begins. The DPA circuitry in an LVDS receiver can reset at anytime without impacting other LVDS receivers sharing the same PLL.

The following events occur during the DPA calibration process:

1. The ALTLVDS_RX megafunction counts 256 data transitions, then inserts delay elements on the LVDS receiver data path to skew the clock and data relationship.
2. The ALTLVDS_RX megafunction counts 256 data transitions, then removes the delay elements on the LVDS receiver data path, restoring the original clock to data relationship.
3. The ALTLVDS_RX megafunction counts 256 data transitions, and then asserts the `rx_dpa_locked` signal.

With the Stratix IV production devices, you can choose to use the DPA PLL calibration method to be backward compatible with Stratix III and Stratix IV ES devices by turning on **Enable PLL calibration** in the ALTLVDS_RX parameter editor. If you turn off **Enable PLL calibration** in the ALTLVDS_RX parameter editor, the receiver megafunction uses delay elements in the receiver data path.

Arria II GZ devices always use the DPA calibration method using delay elements in the receiver data path.

Effects of DPA PLL Calibration

There are two notable effects when DPA PLL calibration is enabled: effect on the timing of the logic clocked by the PLL, and effect related to the merging PLLs.

During PLL phase calibration, the I/O timing is pulled in by quarter of the voltage-controlled oscillator (VCO) period. All outputs of the PLL, including the slow clock, are affected. All HSIO TX data from interfaces, clocked by the affected PLL, clocks out quarter of the VCO period earlier. Likewise, all HSIO RX data clocks quarter cycle out of phase with the VCO but has less time to be sampled. For the slow clock that drives the core and the system, there is a loss of quarter of the VCO period on internal timing, across clock domain transfers in the core. The quarter period-pull greatly affects a design that has cross-clock transfer without using a FIFO, and the two clocks are not from the same PLL.

If DPA PLL calibration is enabled, PLLs, between receiver and transmitter instances or multiple receiver instances, do not merge even if the **Share PLLs for receivers and transmitters** setting is enabled. To force merging of such PLLs, use `FORCE_MERGE_PLLS=ON` setting in the Quartus II Settings File (**.qsf**).

For more information about the `FORCE_MERGE_PLL` command, refer to the *Quartus II Settings File Manual*.

Initialization and Reset

This section describes the initialization and reset aspects, using control characters. This section also provides a recommended initialization and reset flow for the ALTLVDS_TX and ALTLVDS_RX megafunctions.

Initializing the ALTLVDS_TX and ALTLVDS_RX Megafunctions

With the ALTLVDS_TX and ALTLVDS_RX megafunctions, the PLL is locked to the reference clock prior to implementing the SERDES blocks for data transfer. The PLL starts to lock to the reference clock during device initialization. The PLL is operational when the PLL achieves lock during user mode. If the clock reference is not stable during device initialization, the PLL output clock phase shifts becomes corrupted.

When the PLL output clock phase shifts are not set correctly, the data transfer between the high-speed LVDS domain and the low-speed parallel domain might not be successful, which leads to data corruption. Assert the `pll_areset` port for at least 10 ns, and then deassert the `pll_areset` port and wait until the PLL lock becomes stable. After the PLL lock port asserts and is stable, the SERDES blocks are ready for operation.

When using DPA, further steps are required for initialization and reset recovery. The DPA circuit samples the incoming data and finds the optimal phase tap from the PLL to capture the data on a receiver channel-by-channel basis. If the PLL has not locked to a stable clock source, the DPA circuit might lock pre-maturely to a non-ideal phase tap. Use the `rx_reset` port to keep the DPA in reset until the PLL lock signal is asserted and stable.

In Stratix GX, Stratix II, Stratix II GX, HardCopy II, and Arria GX devices, when using the rx_reset port, the ALTLVDS_RX parameter editor allows you to choose whether or not to automatically reset the bit serial FIFO when the rx_dpa_locked signal asserts for the first time. This is a useful feature because it keeps the synchronizer FIFO in reset until the DPA locks. To provide optimal timing between the DPA domain, it is important to keep the FIFO in reset until the DPA locks.

With Stratix III, HardCopy III, Arria II GX, Arria II GZ devices and later generations of these devices, the rx_dpa_lock signal asserts only after a specific number of transitions are detected in the parallel data stream. You must not assert rx_fifo_reset port until the rx_dpa_lock signal asserts, otherwise, there will be no data transitions in the parallel data, and the rx_dpa_lock signal will never assert. Altera recommends asserting the rx_fifo_reset port after the rx_dpa_locked signal asserts, and then deassert the rx_fifo_reset port to begin receiving data.

Each time the DPA shifts the phase taps during normal operation to track variations between the relationship of the reference clock source and the data, the timing margin for the data transfer between clock domains is reduced.

For Stratix GX, Stratix II, Stratix II GX, HardCopy II, and Arria GX devices, when the ALTLVDS_RX deasserts the rx_dpa_locked port to indicate that the DPA has selected a new phase tap to capture the data. You can choose the options in the ALTLVDS_RX parameter editor if you want the DPA lock signal to deassert after one phase step, or after two phase steps in the same direction (check device family availability for this option).

With Stratix III, HardCopy III, Arria II GX, Arria II GZ devices and later generations of these devices, the ALTLVDS_RX asserts the rx_dpa_locked port upon initial DPA lock. This port remains asserted throughout the operation until the ALTLVDS_RX asserts the rx_reset or rx_dpa_lock_reset ports. The rx_dpa_locked port does not indicate if the DPA has selected a new phase. Altera recommends using the data checkers to ensure data accuracy.

Resetting the DPA

When the data becomes corrupted, you must reset the DPA circuitry using the rx_reset port and rx_fifo_reset port.

Assert the rx_reset port to reset the entire DPA block. This requires the DPA to be trained before it is ready for data capture. Altera recommends using the option to automatically reset the bit serial FIFO when the rx_dpa_locked signal rises for the first time, if available for your device family; otherwise, toggle the rx_fifo_reset port after rx_dpa_locked is asserted. This option ensures the synchronization FIFO is set with the optimal timing to transfer data between the DPA and high-speed LVDS clock domains.

Assert the rx_fifo_reset port to reset only the synchronization FIFO. This allows you to continue system operation without having to re-train the DPA. Using this port can fix data corruption because it resets the FIFO; however, it does not reset the DPA circuit. In Stratix GX, Stratix II, Stratix II GX, HardCopy II, and Arria GX devices, the rx_dpa_locked port remains in its previous state; if it was deasserted, it remains deasserted and you are not be able to use it to know when the DPA is using the ideal phase tap for data capture.

When the DPA is locked, the ALTLVDS block is ready to capture data. The DPA finds the optimal sample location to capture each bit. The next step is to set up the word boundary using custom logic to control the `rx_channel_data_align` port on a channel-by-channel basis.

The word aligner or the bit-slip circuit can be reset using the `rx_cda_reset` port. This circuit can be reset anytime and is not dependent on the PLL or DPA circuit operation. For more information about how to use the `rx_channel_data_align` port, refer to Table 3–5 on page 3–23.

Aligning the Word Boundaries

To align the word boundaries, it is useful to have control characters in the data stream so that your logic can have a known pattern to search for. You can compare the data received for each channel, compare to the control character you are looking for, then pulse the `rx_channel_data_align` port as required until you successfully receive the control character. Altera recommends setting the `rx_cda_max[]` port to the deserialization factor or higher, which allows enough depth in the bit slip circuit to roll through an entire word if required.

If you do not have control characters in the received data, you need a deterministic relationship between the reference clock and data to predict the word boundary using timing simulation or laboratory measurements. The only way to ensure a deterministic relationship on the default word position in the SERDES when the device powers up, or anytime the PLL is reset, is to have a reference clock equal to the data rate divided by the deserialization factor. For example, if the data rate is 800 Mbps, and the deserialization factor is 8, the PLL requires a 100-MHz reference clock. This is important because the PLL locks to the rising edge of the reference clock. If you have one rising edge on the reference clock per serial word received, the deserializer always starts at the same position. Using timing simulation, or lab measurements, monitor the parallel words received and determine how many pulses are required on the `rx_channel_data_align` port to set your word boundaries. You can create a simple state machine to apply the required number of pulses when you enter user mode, or anytime you reset the PLL and DPA blocks.

Recommended Initialization and Reset Flow

Altera recommends that you follow these steps to initialize and reset the ALTLVDS megafunctions:

1. During entry into user mode, or anytime in user mode operation when the interface requires a reset, assert the `pll_areset` and `rx_reset` ports.
2. Deassert the `pll_areset` port and monitor the `rx_locked` port (`rx_locked` is the PLL lock indicator).
3. Deassert the `rx_reset` port after the `rx_locked` port becomes asserted and stable.
4. Apply the DPA training pattern and allow the DPA circuit to lock. (If a training pattern is not available, any data with transitions is required to allow the DPA to lock.) Refer to the respective device data sheet for DPA lock time specifications.
5. Wait for the `rx_dpa_locked` port to assert.
6. Beginning with Stratix III, HardCopy III, Arria II GX, and Arria II GZ devices, assert `rx_fifo_reset` for at least one parallel clock cycle, and then de-assert `rx_fifo_reset`.
7. Assert the `rx_cda_reset` port for at least one parallel clock cycle, and then deassert the `rx_cda_reset` port.
8. Begin word alignment by applying pulses as required to the `rx_channel_data_align` port.
9. When the word boundaries are established on each channel, the interface is ready for operation.

Source-Synchronous Timing Analysis and Timing Constraints

This section defines the source-synchronous differential data orientation timing parameters, the timing budget definitions, and how to use these timing parameters to determine a design's maximum performance.

Different modes of LVDS receivers use different specifications in deciding the ability to sample the received serial data correctly.

Dedicated SERDES

The LVDS receiver megafunction implemented in a dedicated SERDES and using the DPA mode are characterized and guaranteed to function correctly within the LVDS system. Refer to the respective device handbook for details about whether dedicated SERDES and DPA are supported for the device family. The Quartus II compiler automatically ensures the associated delay chain settings are set correctly for the data path at the LVDS receiver that uses the source-synchronous compensation mode of PLL operation.

You can optionally add false path constraints to the asynchronous input and output ports to avoid unconstrained path warnings. For non-DPA mode, you can optionally constrain the synchronous input ports to improve the accuracy of the receiver skew margin analysis.

SERDES in LEs

For receiver designs that are using the SERDES in LEs, you must ensure proper timing constraints for the TimeQuest timing analyzer tool in the Quartus II software to indicate whether the SERDES captures the data as expected or otherwise.

For dedicated SERDES and SERDES in LEs, you can set the timing constraints using the following methods:

- Setting timing constraints using the TimeQuest Timing Analyzer GUI
- Setting timing constraints manually in the Synopsys Design Constraints (**.sdc**) file.

The TimeQuest Timing Analyzer automatically adds the required multicycle path. false path, and clock uncertainty constraints to analyze timing for the dedicated SERDES if you add `derive_pll_clocks` to your **.sdc** file.

Receiver Skew Margin and Transmitter Channel-to-Channel Skew

Changes in system environment, such as temperature, media (cable, connector, or PCB), and loading, affect the receiver's setup and hold times; internal skew affects the sampling ability of the receiver.

In non-DPA mode, use receiver skew margin (RKSM), transmitter channel-to-channel skew (TCCS), and sampling window (SW) specifications to analyze the timing for high-speed source-synchronous differential signals in the receiver data path. The relationship between RSKM, TCCS, and SW is expressed by the RSKM equation shown in Equation 3–1:

Equation 3–1. RSKM

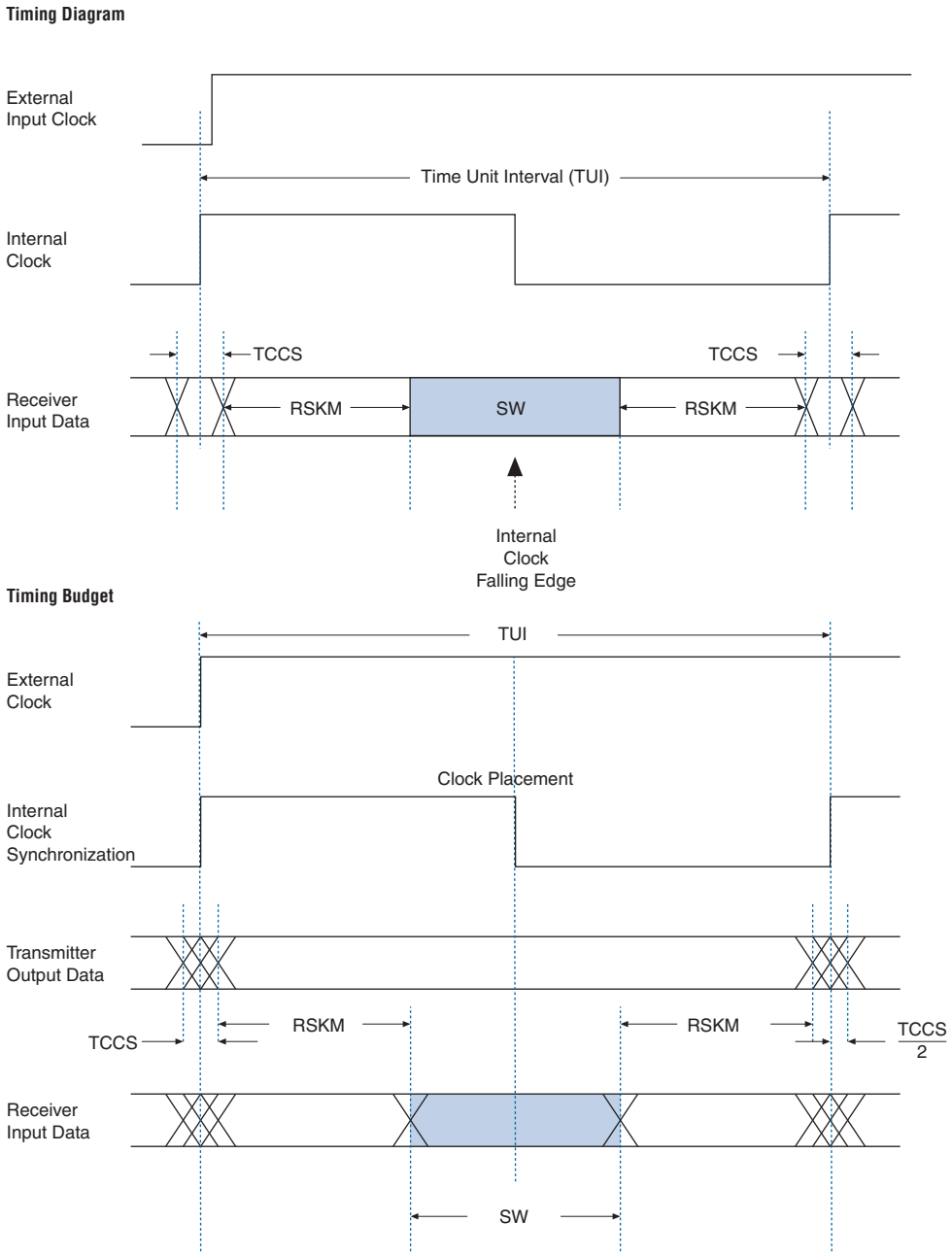
$$RSKM = \frac{TUI - SW - TCCS}{2}$$

Where:

- RSKM—is the timing margin between the receiver's clock input and the data input SW.
- Time unit interval (TUI)—is the time period of the serial data ($1/f_{MAX}$). Also known as the LVDS period in the **TimeQuest Timing Analyzer** section in the Quartus II Compilation Report.
- SW—is the period of time that the input data must be stable to ensure that data is successfully sampled by the LVDS receiver. The SW is a device property and varies with device speed grade.
- TCCS—is the timing difference between the fastest and slowest data output transitions, including the t_{CO} variations and clock skew.

Figure 3–4 shows the relationship between the RSKM, TCCS, and SW.

Figure 3–4. Differential High-Speed Timing Diagram and Timing Budget for Non-DPA Mode



You must calculate the RSKM value to decide whether you can properly sample the data by the LVDS receiver with the given data rate and device. A positive RSKM value indicates the LVDS receiver can properly sample the data; a negative RSKM value indicates the receiver cannot properly sample the data.

Example 3–1 shows the RSKM calculation.

Example 3–1. RSKM

Data Rate: 1 Gbps, Board channel-to-channel skew = **200 ps**

For Stratix IV devices:

TCCS = **100 ps** (pending characterization)
SW = **300 ps** (pending characterization)

TUI = **1000 ps**
Total RCCS = TCCS + Board channel-to-channel skew= 100 ps + 200 ps
= **300 ps**

RSKM= **TUI - SW - RCCS**
= **1000 ps - 300 ps - 300 ps**
= **400 ps > 0**

Because the RSKM > 0 ps, receiver non-DPA mode must work correctly.

For LVDS transmitters, the TimeQuest Timing Analyzer provides a TCCS report, which shows TCCS values for serial output ports. To obtain the TCCS report (`report_TCCS`), follow these steps:

1. In the Quartus II software, under the Tools menu, click **TimeQuest Timing Analyzer.**
2. From the **TimeQuest Timing Analyzer,** under **Reports,** select **Device Specific** and click **Report TCCS**.

For LVDS receivers, the Quartus II software provides the RSKM report showing SW, TUI or LVDS period, and RSKM values for non-DPA mode.You can generate the RSKM report by executing the `report_rskm` command in the TimeQuest Timing Analyzer.

To obtain the RSKM report, follow these steps:

1. In the Quartus II software, under the Tools menu, click **TimeQuest Timing Analyzer.**
2. From the **TimeQuest Timing Analyzer,** under **Reports,** select **Device Specific** and click **Report RSKM**.

In the TimeQuest timing analyzer tool, the `report_TCCS` and `report_rskm` commands are not available when you are using SERDES in LEs. The commands are only available for transmitter and receiver with dedicated SERDES.

For more information about the `report_TCCS` and `report_rskm` commands, refer to *The Quartus II TimeQuest Timing Analyzer* chapter in volume 3 of the *Quartus II Handbook*.

Setting Timing Constraints Using the TimeQuest Timing Analyzer GUI

Timing constraints for the LVDS receiver are needed only for the input clock ports and the synchronous input ports. The synchronous output ports and the asynchronous input and output ports are set to false path.

Constraining the Input Clock Signal

To constrain the input clock signal in the TimeQuest Timing Analyzer, follow these steps:

1. Run full compilation for the LVDS design. Ensure that the timing analysis tool is set to **TimeQuest Timing Analyzer**.
2. After full compilation completes, on the Tools menu, select **TimeQuest Timing Analyzer** to launch the TimeQuest analyzer window.
3. In the **Tasks** list, under **Diagnostic**, click **Report Unconstrained Paths** to view the list of unconstrained paths and ports of the LVDS design.
4. In the **Report** list, under **Unconstrained Paths**, click **Clock Status Summary** to view the clock that requires constraints. The default setting for all unconstrained clocks is 1 GHz. To constrain the clock signal, right-click the clock name and select **Edit Clock Constraint**.
5. In the **Create Clock** dialog box, set the period and the clock rising and falling edge (duty cycle of the clock) constraint. Refer to Table 3–1 on page 3–16 for timing constraints options and descriptions.
6. Click **Run**.

Constraining the Synchronous Input Ports

Constrain the synchronous input signals for non-DPA mode SERDES to allow the TimeQuest Timing Analyzer to consider your board channel-to-channel skew in the RSKM report. Without these constraints, you need to subtract the board channel-to-channel skew from the RSKM value reported by the TimeQuest Timing Analyzer.

To constrain the synchronous input signals in the TimeQuest Timing Analyzer, follow these steps:

1. Run full compilation for the LVDS design. Ensure that the timing analysis tool is set to **TimeQuest Timing Analyzer**.
2. After full compilation completes, on the Tools menu, select **TimeQuest Timing Analyzer** to launch the TimeQuest analyzer window.
3. In the **Tasks** list, under **Diagnostic**, double-click **Report Unconstrained Paths** to view the list of unconstrained paths and ports of the LVDS design.
4. In the **Report** list, under **Unconstrained Paths** category, expand the **Setup Analysis** folder, and then click **Unconstrained Input Ports**.

5. Set constraints for all the receiver synchronous input ports in the **From** list. To set input delay, perform the following steps:
   a. Right-click on the synchronous input port and select **Set Input Delay**.
   b. The **Set Input Delay** dialog box appears.
   c. Select the desired clock using the pull down menu. The clock name must reference the source synchronous clock that feeds the LVDS receiver.
   d. Set the appropriate values for **Input Delay** and **Delay**. Refer to Table 3–1 on page 3–16 for timing constraints options and descriptions.
   e. Click **Run** to incorporate these values in the TimeQuest Timing Analyzer.


If no input delay is set in the TimeQuest Timing Analyzer, the receiver channel-to-channel skew (RCCS) defaults to zero.

Setting False Path for the Asynchronous Input and Output Ports

All asynchronous input and output ports are excluded from the timing analysis of the LVDS core because the signals on these ports are not synchronous to a megafunction clock source. The internal structure of the LVDS megafunction handles the metastability of these asynchronous signals. Therefore these asynchronous signals are set to false path.

To exclude asynchronous input and output ports from the timing analysis, perform the following steps:

1. Run full compilation for the LVDS design. Ensure that the timing analysis tool is set to **TimeQuest Timing Analyzer**.
2. After full compilation completes, on the Tools menu, select **TimeQuest Timing Analyzer** to launch the TimeQuest analyzer window.
3. In the **Tasks** list, under **Diagnostic**, double-click **Report Unconstrained Paths** to view the list of unconstrained paths and ports of the LVDS design.
4. In the **Report** list, under **Unconstrained Paths** category, expand the **Setup Analysis** folder.
5. Click **Unconstrained Input Port Paths** to view the unconstrained input ports or click **Unconstrained Output Port Paths** to view the unconstrained output ports.
6. Right-click on an ansynchronous input or output port, and select **Set False Path**.


After you specify all timing constraint settings for the clock signal, on the **Constraints** menu, click **Write SDC File** to write all the constraints to a specific **.sdc**. Then, run full compilation for the LVDS design again.

Setting Timing Constraints Manually in the Synopsys Design Constraint File

You can also set timing constraints manually using SDC commands in an **.sdc** file, and include the .**sdc** file into your Quartus II design file.

Example 3–2 shows a coding example for a simple source-synchronous interface where the data is aligned with respect to the falling edge of the clock.

Example 3–2. SDC Example

```
#**************************************************************

# Create Clock

#**************************************************************

create_clock -name virtual_clock_lvds -period 25

create_clock -name {rx_inclock} -period 25.000 -waveform { 0.000 12.500
} [get_ports {rx_inclock}] -add

#**************************************************************

# Create Generated Clock

#**************************************************************

derive_pll_clocks

**************************************************************

# Set Input Delay

#**************************************************************

set_input_delay -clock [get_clocks virtual_clock_lvds] -clock_fall -max
0.200 [get_ports rx_in*] -add_delay

set_input_delay -clock [get_clocks virtual_clock_lvds] -clock_fall -min
-0.200 [get_ports rx_in*] -add_delay
```

To add the **.sdc** file into your Quartus II design file, follow these steps:

1. In the Quartus II software, click on the Assignments menu, and select **Settings**.
2. On the **Settings** page, under **Category**, select **TimeQuest Timing Analyzer**.
3. On the **TimeQuest Timing Analyzer** sub-window, browse to the **.sdc** file, and click **Add**.
4. Click **OK**.

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*.

Table 3–1 lists the LVDS timing constraints options and descriptions.

Table 3–1. LVDS Timing Constraints Options and Descriptions

Port Name	Constraint Type	Option		Description
		GUI Setting	SDC command	
Input Clock Constraints				
rx_inclock	create_clock	Clock name	-name	Specifies the name of the LVDS input clock.
		Period	-period	Specifies the clock period (1/fmax).
		Rising, Falling	-waveform	Specifies the clock's rising and falling edges or the the duty cycle of the clock. For example, a 10 ns period where the first rising edge occurs at 0 ns and the first falling edge occurs at 5 ns would be written as waveform {0 5}. The difference must be within one period unit, and the rise edge must come before the fall edge. The default edge list is {0 <period>/2}, or a 50 percent duty cycle.
		Target	[get_ports {*<port name>*}]	Specifies the clock input port name connected to rx_inclock.
Synchronous Input Port Constraints				
rx_in	set_input_delay	Minimum, Maximum	-max -min	Specifies the maximum and minimum delay for the data input to the FPGA.
		Rise, Fall, Both	-clock fall -clock rise	Specifies the clock's rising and falling edges or the the duty cycle of the clock.
		Delay	-*<delay value>*	Specifies the data to clock skew in ns.
		Target	[get_ports {*<port name>*}]	Specifies the data input port name connected to rx_in.

Arria II GX and Stratix V LVDS Package Skew Compensation Report Panel

This section describes the LVDS package skew compensation report panel for the transmitter and non-DPA receiver of the Arria II GX and Stratix V device families.

The report panel contains details about the package trace delay compensation needed between the LVDS pins on the device to meet your timing budget. You can find the report panel in the Quartus II Fitter report under **Resource Section**. The report panel is called **LVDS Receiver Package Skew Compensation**, and **LVDS Transmitter Package Skew Compensation** for the LVDS receiver and LVDS transmitter respectively. The report panel is triggered in the Quartus II software when your design uses a non-DPA receiver, and with an input data rate higher than 840 Mbps.

Figure 3–5 shows the LVDS Transmitter Package Skew Compensation report panel, and Figure 3–6 shows the LVDS Receiver Package Skew Compensation report panel.

Figure 3–5. LVDS Transmitter Package Skew Compensation

LVDS Transmitter Package Skew Compensation

	Name	Pin	Recommended Trace Delay Addition	Estimated TCCS Reduction
1	lvds1:inst2|altlvds_tx:altlvds_tx_component|lvds1_lvds_tx1:auto_generated|wire_tx_dataout[0]	pin_name7[0]	67ps	
2	lvds1:inst2|altlvds_tx:altlvds_tx_component|lvds1_lvds_tx1:auto_generated|wire_tx_dataout[1]	pin_name7[1]	35ps	
3	lvds1:inst2|altlvds_tx:altlvds_tx_component|lvds1_lvds_tx1:auto_generated|wire_tx_dataout[2]	pin_name7[2]	54ps	
4	lvds1:inst2|altlvds_tx:altlvds_tx_component|lvds1_lvds_tx1:auto_generated|wire_tx_dataout[3]	pin_name7[3]	34ps	
5				33ps

Figure 3–6. LVDS Receiver Package Skew Compensation

LVDS Receiver Package Skew Compensation

	Name	Pin	Recommended Trace Delay Addition	Estimated Sampling Window Reduction
1	lvds1:inst2|altlvds_rx:altlvds_rx_component|lvds1_lvds_rx:auto_generated|wire_rx_dataout[12]	pin_name3[3]	63ps	
2	lvds1:inst2|altlvds_rx:altlvds_rx_component|lvds1_lvds_rx:auto_generated|wire_rx_dataout[8]	pin_name3[2]	79ps	
3	lvds1:inst2|altlvds_rx:altlvds_rx_component|lvds1_lvds_rx:auto_generated|wire_rx_dataout[4]	pin_name3[1]	52ps	
4	lvds1:inst2|altlvds_rx:altlvds_rx_component|lvds1_lvds_rx:auto_generated|wire_rx_dataout[0]	pin_name3[0]	65ps	
5	lvds1:inst2|altlvds_rx:altlvds_rx_component|lvds1_lvds_rx:auto_generated|wire_pll_clk[0]	pin_name4	57ps	
6				16ps

The **Recommended Trace Delay Addition** column in the report panel displays the recommended amount of trace delay that you must add to each trace of the corresponding LVDS pins, which reduces the channel-to-channel skew between the LVDS channels. For example, in Figure 3–5, the recommended trace delay addition for `pin_name7[0]` is 67 ps. This means you must manually adjust the PCB trace for `pin_name7[0]` to have a delay addition of 67 ps. The corresponding pin is listed in the **Pin** column, in the report panel.

The report panel also shows the total estimated TCCS and SW reductions when the recommended trace delay values are added to the PCB trace.

Generating Clock Signals for LVDS Interface

The ALTLVDS MegaWizard Plug-In Manager software provides an option for implementing the LVDS interface with the **Use External PLL** option. 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 an ALTPLL megafunction to generate the various clock and load enable signals.

When you enable the **Use External PLL** option with the ALTLVDS transmitter and receiver, the following signals are required from the ALTPLL megafunction:

- Serial clock input to the SERDES of the ALTLVDS transmitter and receiver
- Load enable to the SERDES of the ALTLVDS transmitter and receiver
- Parallel clock used to clock the transmitter FPGA fabric logic and parallel clock used for the receiver `rx_syncclock` port and receiver FPGA fabric logic
- Asynchronous PLL reset port of the ALTLVDS receiver

The following section describes how the serial clock output, load enable output, and the parallel clock output are generated on ports `c0`, `c1`, and `c2`, respectively, along with the locked signal of the ALTPLL instance. You can choose any of the PLL output clock ports to generate the interface clocks.

The high-speed clock generated from the PLL is intended to clock the LVDS SERDES circuitry only. Do not use the high-speed clock to drive other logic because the allowed frequency to drive the core logic is restricted by the PLL F_{OUT} specification. For more information about the F_{OUT} specification, refer to the *DC and Switching Characteristics for Stratix IV Devices* chapter.

Table 3–2 lists the signal interface between the output ports of the ALTPLL megafunction and the input ports of the ALTLVDS transmitter and receiver.

Table 3–2. Signal Interface Between ALTPLL and ALTLVDS Megafunctions

From the ALTPLL Megafunction	To the ALTLVDS Transmitter	To the ALTLVDS Receiver
Serial clock output (c0) *(1)*	`tx_inclock` (serial clock input to the transmitter)	`rx_inclock` (serial clock input)
Load enable output (c1)	`tx_enable` (load enable to the transmitter)	`rx_enable` (load enable for the deserializer)
Parallel clock output (c2)	Parallel clock used inside the transmitter core logic in the FPGA fabric	`rx_syncclock` (parallel clock input) and parallel clock used inside the receiver core logic in the FPGA fabric
~(locked)	—	`pll_areset` (asynchronous PLL reset port) *(2)*

Notes to Table 3–2:

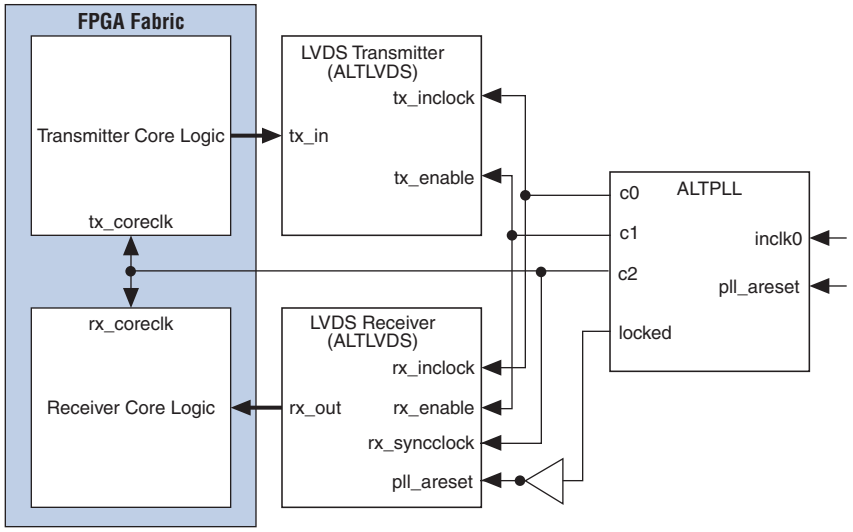
(1) The serial clock output (c0) can only drive `tx_inclock` on the ALTLVDS transmitter and `rx_inclock` on the ALTLVDS receiver. This clock cannot drive the core logic.

(2) The `pll_areset` signal is automatically enabled for the LVDS receiver in external PLL mode. This signal does not exist for LVDS transmitter instantiation when the external PLL option is enabled.

The rx_syncclock port is automatically enabled in an LVDS receiver in external PLL mode. The Quartus II compiler errors out if this port is not connected, as shown in Figure 3–7.

When generating the ALTPLL megafunction, the **Left/Right PLL** option sets up the PLL in LVDS mode. Figure 3–7 shows the connection between the ALTPLL and ALTLVDS megafunctions.

Figure 3–7. LVDS Interface with the ALTPLL Megafunction *(1)*



Note to Figure 3–7:

(1) Instantiation of pll_areset is optional for the ALTPLL instantiation.

Example 3–3 shows the parameter values that you can set in the ALTPLL parameter editor to generate three output clocks.

Example 3–3. Generating Three Output Clocks Using an ALTPLL Megafunction

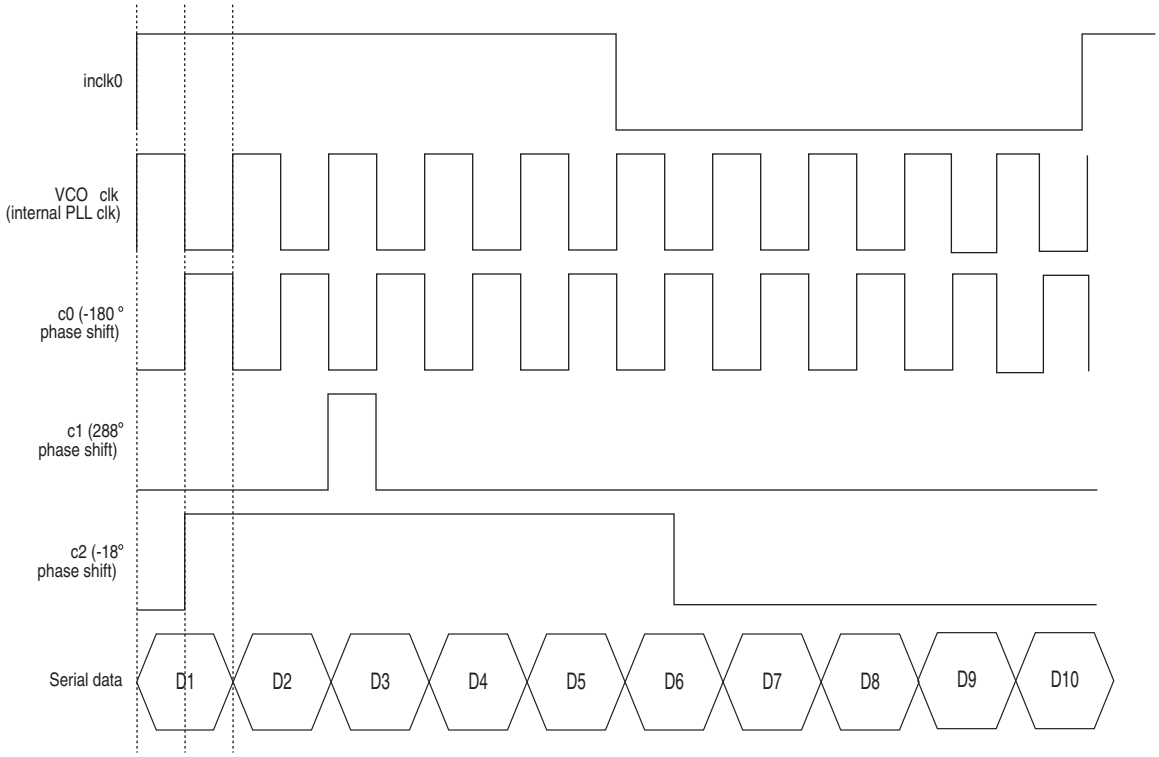
LVDS data rate = 1 Gbps; serialization factor = 10; input reference clock = 100 MHz

The following settings are used when generating the three output clocks using an ALTPLL megafunction. The serial clock must be **1000 MHz** and the parallel clock must be **100 MHz** (serial clock divided by the serialization factor):

- c0
  - Frequency = **1000 MHz** (multiplication factor = 10 and division factor = 1)
  - Phase shift = **–180°** with respect to the voltage-controlled oscillator (VCO) clock
  - Duty cycle = **50%**
- c1
  - Frequency = (1000/10) = **100 MHz** (multiplication factor = 1 and division factor = 1)
  - Phase shift = (10 - 2) × 360/10 = **288°** [(deserialization factor - 2)/deserialization factor] × 360°
  - Duty cycle = (100/10) = **10%** (100 divided by the serialization factor)
- c2
  - Frequency = (1000/10) = **100 MHz** (multiplication factor = 1 and division factor = 1)
  - Phase shift = (–180/10) = **–18°** (c0 phase shift divided by the serialization factor)
  - Duty cycle = **50%**

Phase shift calculattions using Equation 3–1 on page 3–10 assume that the input clock and serial data are edge aligned. Figure 3–8 shows that by introducing a phase shift of –180° to sampling clock (c0) ensures that the input data is center-aligned with respect to the c0.

Figure 3–8. Phase Relationship for External PLL Interface Signals



Resource Utilization and Performance

The Quartus II software configures the PLL according to the settings you apply in the ALTLVDS_RX and ALTVDS_TX parameter editor. All supported devices provide the option to use an external PLL, which requires you to enter the appropriate PLL parameters.

When the ALTLVDS_TX and ALTLVDS_RX megafunctions are instantiated without the external PLL option, they use one PLL per instance. During compilation, if directed to do so, the compiler tries to merge PLLs whenever possible to minimize resource usage.

For more information about the PLL parameters, refer to the *ALTPLL Megafunction User Guide*.

The Arria, Cyclone, Hardcopy, and Stratix series support the **Use Shared PLL(s) for Receiver and Transmitter** option to allow both the ALTLVDS_TX and ALTLVDS_RX megafunctions to share a PLL. The Quartus II software lets the transmitter and receiver share the same PLL when both use identical input clock sources, identical `pll_areset` sources, identical deserialization factors, and identical output settings. For example, the Quartus II software displays the following message when the PLL merges successfully:

```
Info: Receiver fast PLL <lvds_rx PLL name> and transmitter fast PLL
<lvds_tx PLL name> are merged together
```

The Quartus II software displays the following message when it cannot merge the PLLs for the LVDS transmitter and receiver pair in the design:

```
Warning: Can't merge transmitter-only fast PLL <lvds_tx PLL name> and
receiver-only fast PLL <lvds_rx PLL name>
```

One cause for the warning message is that PLLs that are driven by different clocks cannot be merged. For PLL merging to happen, the input clocks and the settings on the outputs must be identical.

To use the LVDS I/O standard in the I/O Bank 1 of Cyclone III and Cyclone IV E devices, ensure that you set the **Configuration device I/O voltage** to 2.5 V, or Auto in the **Device and Pin Options** dialog box of the Quartus II software.

For the Stratix series, the side I/O banks contain dedicated SERDES circuitry, which includes the PLLs, serial shift registers, and parallel registers. The transmit and receive functions use varying numbers of LEs depending on the number of channels, serialization, and deserialization factors. For best performance, manually place these LEs in columns as close as possible to the SERDES circuitry and LVDS pins. By default, the Quartus II software places these LEs automatically during placement and routing.

When dedicated SERDES is implemented in LVDS transmitter, the SERDES is directly connected to the LVDS transmitter; therefore, the output of the transmitter cannot be assigned to single-ended I/O standards.

The Quartus II software reports the number of LEs used per ALTLVDS block in the **Fitter Resource Utilization by Entity** section in the **Resource** section of the **Compilation Report**.

The Cyclone series uses DDIO registers as part of the SERDES interface. Because data is clocked on both the rising edge and falling edge, the clock frequency must be half the data rate; therefore, the PLL runs at half the frequency of the data rate. The core clock frequency for the transmitter is data rate divided by serialization factor (J). For the odd serialization factors, depending on the output clock-divide factor (B) and device family, an optional core clock frequency of data rate divided by two times the serialization factor (J) is also available.

Use Table 3–3 and Table 3–4 to determine the clock and data rate relationships.

Table 3–3. Cyclone Series ALTLVDS Receiver Clock Relationships

Clock Type	J = Even	J = Odd
Fast Clock	Data Rate / 2	Data Rate / 2
Slow Clock (outclock)	Data Rate / J	Data Rate / J

Table 3–4. Cyclone Series ALTLVDS Transmitter Clock Relationships

Clock Type	J = Even	J = Odd
Fast Clock	Data Rate / 2	Data Rate / 2
Slow Clock (outclock)	Data Rate / 2 * B	Data Rate / 2 * B
Core Clock	Data Rate / J	Data Rate / J

ALTLVDS_TX and ALTLVDS_RX Ports

This section describes the ports for the ALTLVDS_TX and ALTLVDS_RX megafunctions.

ALTLVDS_RX Megafunction Ports

Table 3–5 lists the input and output ports for the ALTLVDS_RX megafunction.

n is the number of channels. *m* is the `deserialization_factor` × `number_of_channels`.

Table 3–5. ALTLVDS_RX Megafunction Input and Output Ports (Part 1 of 4)

Port Name	Direction	Width (Bit)	Description
`dpa_pll_recal`	Input	1	Enables dynamic recalibration without resetting the DPA circuitry or the PLL. Only available in DPA mode when PLL calibration is enabled.
`pll_areset`	Input	1	Asynchronously resets all counters to initial values. The minimum pulse width requirement for this signal is 10 ns.
`pll_phasedone`	Input	1	Specifies whether dynamic phase reconfiguration is complete. Only available when using an external PLL when PLL calibration is enabled.
`rx_cda_reset`	Input	*n*	Asynchronous reset to the data realignment circuitry. The minimum pulse width requirement for this reset is one parallel clock cycle. This signal resets the data realignment block.
`rx_channel_data_align`	Input	*n*	Controls byte alignment circuitry.
`rx_coreclk`	Input	*n*	LVDS reference input clock. Replaces the non-peripheral clock from the PLL. One clock for each channel.
`rx_data_align`	Input	1	Controls byte alignment circuitry. You can register this port using the `rx_outclock` port. This port is available when `implement_in_les` parameter is set to `ON` and can be implemented using flexible LVDS.

Table 3–5. ALTLVDS_RX Megafunction Input and Output Ports (Part 2 of 4)

Port Name	Direction	Width (Bit)	Description
rx_data_align_reset	Input	1	Resets the byte alignment circuitry. Use the rx_data_align_reset input port when you need to reset the PLL during device operation and when you need to re-establish the word alignment. This port is available when implement_in_les parameter is set to ON.
rx_data_reset	Input	*n*	Asynchronous reset for all channels, excluding the PLL.
rx_deskew	Input	1	Specifies whether to activate calibration mode.
rx_dpa_lock_reset	Input	*n*	Forces the rx_dpa_locked port to low and forces the lock counter to start counting again.
rx_dpll_enable	Input	*n*	Enables the data path that flows through the DPA circuit. This port is available only when DPA mode is enabled. This port is supported in Arria GX, HardCopy II, Stratix II, and Stratix II GX devices only.
rx_dpll_hold	Input	*n*	Prevents the DPA circuitry from switching to a new phase. When low, the DPA tracks any dynamic phase variations between the clock and data. When high, the DPA holds the last locked phase and does not track any dynamic phase variations between the clock and data. This port is not available in non-DPA mode.
rx_dpll_reset	Input	*n*	Asynchronous reset for all channels.
rx_enable	Input	1	Enables external PLL usage. When the rx_enable port is specified, it must connect to the enable0 or enable1 port of an ALTPLL megafunction instance configured in LVDS mode.
rx_fifo_reset	Input	*n*	Asynchronous reset to the FIFO between the DPA and the data realignment circuits. The synchronizer block must be reset after a DPA loses lock condition and the data checker shows corrupted received data. The minimum pulse width requirement for this reset is one parallel clock cycle. This signal resets the FIFO block. Only available when DPA mode is enabled.
rx_in[]	Input	*n*	LVDS serial data input port. After deserialization, rx_in[n-1] is the first bit received and rx_in[0] is the last bit received for channel 1; for channel 2, rx_in[2n-1] is the first bit received and rx_in[n] is the last bit received.

Table 3–5. ALTLVDS_RX Megafunction Input and Output Ports (Part 3 of 4)

Port Name	Direction	Width (Bit)	Description
rx_inclock	Input	1	LVDS reference input clock. The allowed frequency range for the reference clock is between 5 MHz and 625 MHz and must satisfy the following condition: 5 Mbps < ([Input Clock Frequency] × [PLL Multiplication Factor]) < 1600 Mbps. The MegaWizard Plug-In Manager software automatically selects the appropriate PLL multiplication factor based on the data rate and reference clock frequency selection. When using Stratix II devices in external PLL mode, connect the rx_inclock port to the sclkout0 or sclkout1 port. When using Cyclone and Cyclone II devices in external PLL mode, connect the rx_inclock port to other clocks.
rx_pll_enable	Input	1	Enables control for the LVDS PLL.
rx_readclock	Input	1	Clock input port for reading operation.
rx_reset	Input	n	Asynchronous reset to the DPA circuitry and FIFO. The minimum pulse width requirement for this reset is one parallel clock cycle. This signal resets DPA and FIFO blocks. You can connect this port if the enable_dpa_mode parameter is turned on.
rx_syncclock	Input	1	Slow clock input port.
dpa_pll_cal_busy	Output	1	Busy signal that is asserted high when PLL calibration occurs. PLL clock signals are phase adjusted for two fast clock cycles ahead. Available only when DPA mode with PLL calibration is enabled.
pll_phasecounterselect	Output	1	Specifies the PLL counter select. Available only when DPA mode with PLL calibration is enabled.
pll_phasestep	Output	1	Specifies dynamic phase shifting. Available only when DPA mode with PLL calibration is enabled.
pll_phaseupdown	Output	1	Specifies dynamic phase adjustment. Available only when DPA mode with PLL calibration is enabled.
pll_scanclk	Output	1	Clock signal for the serial scan chain. Available only when DPA mode with PLL calibration is enabled.
rx_cda_max	Output	n	Data re-alignment (bit slip) roll-over signal. When high for one parallel clock cycle, this signal indicates that the user-programmed number of bits for the word boundary to roll-over have been slipped. Indicates when the next rx_channel_data_align pulse restores the serial data latency back to 0.

Table 3–5. ALTLVDS_RX Megafunction Input and Output Ports (Part 4 of 4)

Port Name	Direction	Width (Bit)	Description
rx_divfwdclk	Output	n	Parallel DPA clock to the FPGA fabric logic array. The parallel receiver output data to the FPGA fabric logic array is synchronous to this clock in soft-CDR mode. This signal is not available in non-DPA and DPA modes. Divides and forwards the clock to the source from the DPA block of the clock channel. When the enable_soft_cdr_mode parameter is set to ON, the rx_divfwdclk port is used. When set to ON, the rx_divfwdclk port clocks the synchronization registers.
rx_dpa_locked	Output	n	Indicates whether the channel is locked to DPA mode. This signal only indicates 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. You must not use the rx_dpa_locked signal to determine a DPA loss-of-lock condition.
rx_locked	Output	1	Provides the LVDS PLL status. Stays high when the PLL is locked to rx_inclock, and stays low when the PLL fails to lock.
rx_out	Output	m	Receiver parallel data output. The data bus width per channel is the same as the deserialization factor (DF). The output data is synchronous to the rx_outclock signal in non-DPA and DPA modes. It is synchronous to the rx_divfwdclk signal in soft-CDR mode.
rx_outclock	Output	1	Parallel output clock from the receiver PLL. The parallel data output from the receiver is synchronous to this clock in non-DPA and DPA modes. This port is not available when you turn on the **Use External PLL** option in the parameter editor. The FPGA fabric-receiver interface clock must be driven by the PLL instantiated through the ALTPLL parameter editor.

ALTLVDS_TX Megafunction Ports

Table 3–6 lists the input and output ports for the ALTLVDS_TX megafunction.

☞ *n* is the number of channels. *m* is the deserialization_factor × number_of_channels.

Table 3–6. ALTLVDS_TX Megafunction Input and Output Ports (Part 1 of 2)

Port Name	Direction	Width (Bit)	Description
pll_areset	Input	1	Asynchronously resets all counters to the initial values.
sync_inclock	Input	1	Optional clock for the input registers.

Table 3–6. ALTLVDS_TX Megafunction Input and Output Ports (Part 2 of 2)

Port Name	Direction	Width (Bit)	Description
tx_data_reset	Input	n	Asynchronous reset for the shift registers, capture registers, and synchronization registers for all channels. This port is available only when implement_in_les parameter is set to ON. This port does not affect the data realignment block or the PLL.
tx_enable	Input	1	Enables external PLL usage. When the tx_enable port is specified, connect the port to the enable0 or enable1 port of an ALTPLL megafunction.
tx_in[]	Input	m	This is parallel data which needs to be serially transmitted by the megafunction. Input data must be synchronous to the tx_coreclock signal. The data bus width per channel is the same as the serialization factor (SF)
tx_inclock	Input	1	Reference clock input for the transmitter PLL. The allowed frequency range for the reference clock is between 5 MHz and 625 MHz and must satisfy the following condition: 5 Mbps < ([Input Clock Frequency] × [PLL Multiplication Factor]) < 1600 Mbps. The MegaWizard Plug-In Manager software automatically selects the appropriate PLL multiplication factor based on the data rate and reference clock frequency selection. When using Stratix II devices in external PLL mode, connect the tx_inclock port to the sclkout0 or sclkout1 port. When using Cyclone and Cyclone II devices in external PLL mode, connect the tx_inclock port to other clocks.
tx_pll_enable	Input	1	Enables control for the LVDS PLL.
tx_syncclock	Input	1	Slow clock input port. In the Quartus II software version 8.0 or later, the tx_syncclock port is necessary for even deserialization factors in external PLL mode.
tx_coreclock	Output	1	Output clock used to feed non-peripheral logic. FPGA fabric-transmitter interface clock. The parallel transmitter data generated in the FPGA fabric must be clocked with this clock.
tx_locked	Output	1	Provides the LVDS PLL status. This port stays high when the PLL is locked to the input reference clock, and stays low when the PLL fails to lock.
tx_out[]	Output	n	Serialized LVDS data signal. The tx_out[] is clocked by a serial clock generated by the left and right PLL. After deserialization, tx_out[n-1] is the first bit transmitted and tx_out[0] is the last bit transmitted for channel one; for channel two, tx_out[2n-1] is the first bit transmitted and tx_out[n] is the last bit transmitted.
tx_outclock	Output	1	External reference clock. The frequency of this clock is programmable to be the same as the data rate (up to 717 MHz), half the data rate, or one-fourth the data rate. The phase offset of this clock, with respect to the serial data, is programmable in increments of 45°.

Design Example Files

The design examples for the ALTLVDS megafunctions in this user guide use the MegaWizard Plug-In Manager in the Quartus II software. The design example files are available for download from the following locations:

- On the Quartus II Development Software Literature page, expand the **Using Megafunctions** section and then expand the **I/O** section.
- On the Literature: User Guide section of the Altera website.

In the following design examples, create two files using the parameter editor: one for an ALTLVDS receiver and one for an ALTLVDS transmitter. Both files are created either in VHDL (**.vhd**) or Verilog (**.v**) and are included in your project directory when completed. The PLL block that is used in certain design examples is already customized in the design.

The designs are simulated in the ModelSim®-Altera software to generate a waveform display of the device behavior. You should be familiar with the ModelSim-Altera software before using the design examples. To get started with the ModelSim-Altera software, refer to the ModelSim-Altera Software Support page on the Altera website. The support page includes links to such topics as installation, usage, and troubleshooting

Design Example 1: Stratix II LVDS-to-LVDS Bridge Using Different Clock Frequencies

With the inclusion of DPA circuitry, Stratix II devices offers enhanced support for source-synchronous protocols. The enhanced source-synchronous channels on Stratix II devices support 1-Gbps data transfer, and the dedicated DPA circuitry simplifies PCB design by eliminating signal-alignment issues introduced by clock-to-channel and channel-to-channel skew. Stratix II devices support a wide array of high-speed protocols and can be used to bridge high-speed interfaces.

This design example uses a Stratix II device to bridge a 1-Gbps LVDS interface using a 500-MHz reference clock to a 1-Gbps LVDS interface using a 250-MHz reference clock. The ALTLVDS receiver function uses the DPA circuitry. The transmitter and receiver share one fast PLL.

The focus of this design is to illustrate the features available in the parameter editor. No user logic is shown between the receiver and transmitter blocks, and all ports are connected to input or output pins. Most designs use custom logic for many of the control ports within the FPGA. However, for this design example, all such parameters are controlled outside of the Stratix II device.

In this example, the following tasks are completed:

- Generate a high-speed differential receiver in DPA mode using the ALTLVDS megafunctions and the parameter editor.
- Generate a high-speed differential transmitter using the ALTLVDS megafunctions and the parameter editor.
- Implement the receiver and transmitter functions in the device by adding your custom megafunctions to the project and compiling the project.

- Simulate the high-speed differential interface design using the ModelSim®-Altera software.

Generate an ALTLVDS Receiver and ALTLVDS Transmitter

To generate the LVDS receiver, follow these steps:

1. Open the **ALTLVDS_DesignExample.zip** file and extract the Quartus II archive project **ALTLVDS_stratixII.qar**.
2. In the Quartus II software, open **ALTLVDS_stratixII.qar** and restore the archive file into your working directory.
3. Open the top-level block editor file **ALTLVDS_stratixII.bdf**. The file **ALTLVDS_stratixII.bdf** is an incomplete file that you need to complete in the course of this example. The ALTLVDS megafunction created in this example are added to the top-level file.
4. Double-click anywhere in the white space in the block editor file. The Symbol window appears.
5. In the Symbol window, click **MegaWizard Plug-In Manager**. Page 1 of the parameter editor appears.
6. Select **Create a new custom megafunction variation**.
7. Click **Next**. Page 2a of the parameter editor appears.
8. In the MegaWizard Plug-In Manager pages, select or verify the configuration settings shown in Table 3–7. Click **Next** to advance from one page to the next.

Table 3–7. Configuration Settings for Design Example 1 (LVDS Receiver) (Part 1 of 3)

MegaWizard Plug-In Manager Page	MegaWizard Plug-In Manager Configuration Setting	Value
2a	**Device family**	Stratix II
	Output file type	Verilog HDL
	Output file name	‘stratixiv_rx’
	Return to this page for another create operation	Selected
3	**Currently selected device family**	Stratix IV
	Match project/default	Selected
	Implement Deserializer circuitry in logic cells	Not selected
	Enable Dynamic Phase Alignment mode	Selected
	What is the number of channels?	8
	What is the deserialization factor?	8
	Use external PLL	Not selected
	Use clock pin	Not selected
	Use ‘rx_data_reset’ input port	Not selected

Table 3–7. Configuration Settings for Design Example 1 (LVDS Receiver) (Part 2 of 3)

MegaWizard Plug-In Manager Page	MegaWizard Plug-In Manager Configuration Setting	Value
4	**What is the input data rate?**	1 Gbps (1,000 Mbps)
	Specify input clock rate by	Clock frequency
	Clock frequency	500 MHz
	Clock period	—
	Use shared PLL(s) for receivers and transmitters	Selected
	Use 'pll_areset' input port	Selected. (The PLL must be reset for the output clock phase relationships to be set correctly when the PLL loses lock, or if the PLL input reference clock is not stable when the device completes the configuration process.)
	Use 'rx_pll_enable' input port	—
	Use 'rx_locked' output port	Not selected
	What is the clock resource used for 'rx_outclock'?	Auto selection
	What is the phase alignment of 'rx_in' with respect to 'rx_inclock' (in degrees)	—
	Use source-synchronous mode of the PLL	—
	Align clock to center of data window at capture point	—
	Enable self-reset on lost lock in PLL	—
	Enable FIFO for DPA channels	—
5	**Use 'rx_divfwdclk' output port and bypass the DPA FIFO**	Not selected
	What is the simulated recovered clock phase drift?	—
	Use 'rx_dpll_enable' input port	—
	Use 'rx_dpll_hold' input port	Selected
	Use 'rx_fifo_reset' input port	Selected
6	**Use 'rx_reset' input port**	Selected
	Automatically reset the bit serial FIFO when 'rx_dpa_locked' rises for the the first time	—
	User explicitly resets the bit serial FIFO through 'rx_reset'	—
	Use 'rx_dpa_locked' output port	Selected
	When should the 'rx_dpa_locked' fall low?	—
	Use 'rx_dpa_lock_reset" input port	—
	Use a DPA initial phase selection of (in degrees)	—
	Align DPA to rising edge of data only	—

Table 3–7. Configuration Settings for Design Example 1 (LVDS Receiver) (Part 3 of 3)

MegaWizard Plug-In Manager Page	MegaWizard Plug-In Manager Configuration Setting	Value
7	Enable PLL calibration	Not selected
	Use 'dpa_pll_recal' input port	—
	What is the input data rate (in Mbps)	—
8	Register outputs	Selected
	Use 'rx_cda_reset' input port	Selected
	Use 'rx_cda_max' output port	Selected
	After how many pulses does the data alignment circuitry restore the serial data latency back to 0?	8
	Align data to rising edge of clock	—
	Use 'rx_coreclk' input port	—
	Use 'rx_channel_data_align' input port	Selected
	Enable independent bitslip control for each channel	—
	Add extra register for 'rx_data_align' input port	—
	Use 'rx_data_align_reset' input port	Selected
	Use RAM buffer	—
	Use a multiplexer and synchronization register	—
	Use logic element based RAM buffer	—
9	Generate netlist	Not selected
10	Variation file	Selected
	Quartus II IP file	Selected
	Quartus II symbol file (.bsf)	Selected
	Instantiation template file	Selected
	Verilog HDL black box file (_bb.v)	Selected
	AHDL Include file (.inc)	Selected
	VHDL component declaration file (.cmp)	Selected
	PinPlanner ports file (.PPF)	Selected

9. Click **Finish**. The 'stratixii_rx' module is built.
10. Click **OK**. The MegaWizard Plug-In Manager resets to page 2a to allow you to create a new custom megafunction variation for the LVDS transmitter.
11. In the MegaWizard Plug-In Manager pages, select or verify the configuration settings shown in Table 3–8.

Table 3–8. Configuration Settings for Design Example 1 (LVDS Transmitter) (Part 1 of 3)

MegaWizard Plug-In Manager Page	MegaWizard Plug-In Manager Configuration Setting	Value
2a	Device family	Stratix IV
	Output file type	Verilog HDL
	Output file name	'stratixii_tx'
	Return to this page for another create operation	Not selected

Table 3–8. Configuration Settings for Design Example 1 (LVDS Transmitter) (Part 2 of 3)

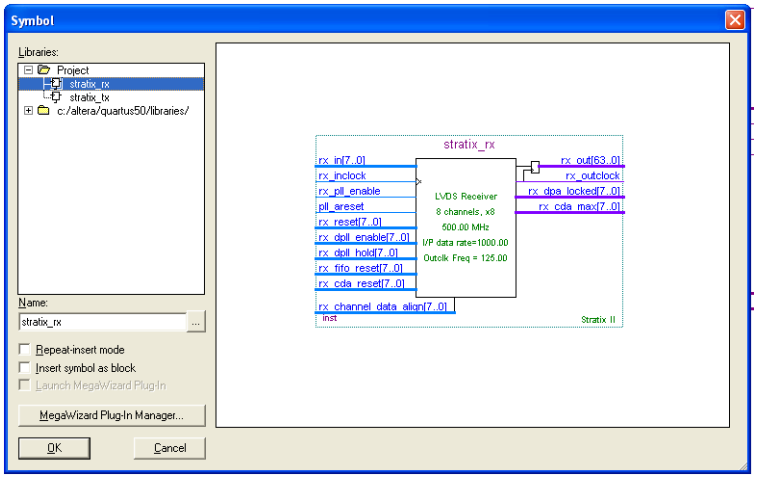
MegaWizard Plug-In Manager Page	MegaWizard Plug-In Manager Configuration Setting	Value
3	Currently selected device family	Stratix Ii
	Match project/default	Selected
	Implement Serializer/Deserializer circuitry in logic cells	Not selected
	What is the number of channels?	8
	What is the deserialization factor?	8
	Use external PLL	Not selected
	Use clock pin	Not selected
	Use ‘tx_data_reset’ input port	Not selected
4	What is the output data rate?	1 Gbps (1,000 Mbps)
	Specify input clock rate by	Clock frequency
	Clock frequency	500 MHz
	Clock period	—
	What is the phase alignment of ‘tx_in’ with respect to the rising edge of ‘tx_inclock’ (in degrees)	0
	Use ‘tx_pll_enable’ input port	—
	Use ‘pll_areset’ input port	Selected (The PLL must be reset for the output clock phase relationships to be set correctly when the PLL loses lock, or if the PLL input reference clock is not stable when the device completes the configuration process.)
	Align clock to center of data window	—
	Enable self-reset on loss lock in the PLL	—
	Use shared PLL(s) for receivers and transmitters	Selected
	Register ‘tx_in’ input port using	tx_coreclock
5	Use ‘tx_outclock’ output port	Selected
	What is the outclock divide factor (B)?	4
	Specify phase alignment of ‘tx_outclock’ with respect to ‘tx_out’	—
	What is the phase alignment of ‘tx_outclock’ with respect to ‘tx_out’ (in degrees)	0
	What is the outclock duty cycle?	—
	Use ‘tx_locked’ output port	Selected
	Use ‘tx_coreclock’ output port	Selected
	What is the clock resource used for ‘tx_coreclock’?	Auto selection
6	Generate netlist	Not selected

Table 3–8. Configuration Settings for Design Example 1 (LVDS Transmitter) (Part 3 of 3)

MegaWizard Plug-In Manager Page	MegaWizard Plug-In Manager Configuration Setting	Value
7	**Variation file (.v)**	Selected
	Quartus II IP file (.qip)	Selected
	Quartus II symbol file (.bsf)	Selected
	Instantiation template file (_inst.v)	Selected
	Verilog HDL black box file (_bb.v)	Selected
	AHDL Include file (.inc)	Selected
	VHDL component declaration file (.cmp)	Selected
	PinPlanner ports file (.PPF)	Selected

12. Click **Finish**. The 'stratixii_tx' module is built.
13. Click **OK**.
14. Place the **stratixii_tx** symbol in the **altlvds_stratixii** block editor design file under the text **INSERT STRATIXII_TX HERE**, aligning the input and output ports with the signals already present in the design file.
15. Double-click anywhere in the white space of the design file. The Symbol window appears.
16. Choose **stratixii_rx** from the **Project** library list and click **OK** (Figure 1). You must expand the **Project** folder to see which megafunction it contains.

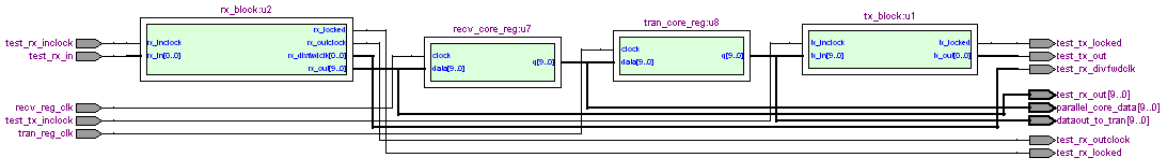
Figure 1. Design Example 1 Symbol View of stratixii_rx Module



17. Place the **stratixii_rx** symbol in the **altlvds_stratixII** block editor design file under the text **INSERT STRATIXII_RX HERE** and align the input and output ports with the signals already present in the design file.

    The top-level schematic looks similar to Figure 3–9 on page 3–34.

Figure 3–9. Design Example 1 Top-Level Schematic



18. On the File menu, click **Save**.
19. On the Processing menu, click **Start Compilation**.
20. When the **Full Compilation was successful** message box appears, click **OK**.

Functional Results—Simulate the ALTLVDS Receiver/Transmitter Design in the ModelSim-Altera Software

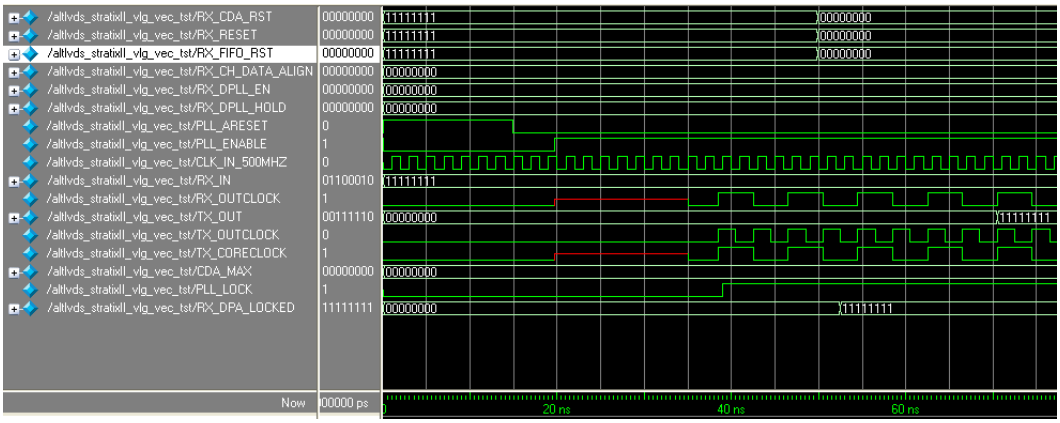
Simulate the design in the ModelSim-Altera software to generate a waveform display of the device behavior.

To set up the ModelSim-Altera Software, follow these steps:

1. Unzip the **altlvds_ex1_msim.zip** file to any working directory on your PC.
2. Start the ModelSim-Altera software.
3. On the File menu, click **Change Directory**.
4. Select the folder in which you unzipped the files. Click **OK**.
5. On the Tools menu, point to **TCL** and click **Execute Macro**. The **Execute Do File** dialog box appears.
6. Select the **altlvds_ex1_msim.do** file and click **Open**. This is a script file for the ModelSim-Altera software that automates all the necessary settings for the simulation.
7. Verify the results shown in the waveform viewer window.

You can rearrange signals, remove signals, add signals, and change the radix by modifying the script in **altlvds_ex1_msim.do**. Figure 3–10 shows the expected simulation results in the ModelSim-Altera software.

Figure 3–10. Design Example 1— ModelSim-Altera Simulation Results



This waveform file contains only the input and output pins that are shown. You can add internal post-compilation nodes with the **Node Finder** to view all of the data paths in the design. You can verify that `tx_outclock` signal has a 4-ns period, which corresponds to a 250-MHz clock rate, as specified in the design.

The `rx_in` input serial data is edge-aligned to the `clk_in_500MHz` reference clock, and the `tx_out` output serial data is edge-aligned to `tx_outclock` signal, as specified in the ALTLVDS_TX megafunction.

This simulation output verifies that 1-Gbps data can be successfully received by and transmitted from the Stratix II device.

You can change the input vectors in the waveform editor to experiment with the other features, such as data alignment, to view the functionality, and help you understand the ALTLVDS megafunction.

Design Example 2: Cyclone II ALTLVDS Using External PLL Option

The following design example illustrates the connection scheme used in Cyclone II devices for an ALTLVDS receiver and transmitter using a common PLL. The ALTLVDS_RX and ALTLVDS_TX megafunctions are set up using the external PLL option. This example shows a step-by-step approach to set up the ALTLVDS_RX and ALTLVDS_TX megafunctions. Note that the ALTPLL megafunction has been set up in the design file.

For PLL-specific information and examples of PLL clock relationships, refer to the *ALTPLL Megafunction User Guide*.

You can use the external PLL option in the ALTLVDS megafunctions to give you direct access to all of the PLL clocks that are not accessible when you allow the ALTLVDS megafunctions to infer the PLL for you. This option gives you the ability to use the PLL output clocks throughout your design for other functions besides the serialization and deserialization of data.

This example shows how to design a 600-Mbps receiver with a deserialization factor of 8 and a 600-Mbps transmitter with the same deserialization factor of 8. The input reference clock frequency is 75 MHz.

In this example, the following tasks are completed:

- Generate a high-speed differential receiver using the ALTLVDS megafunctions and the parameter editor.
- Generate a high-speed differential transmitter using the ALTLVDS megafunctions and the parameter editor.
- View descriptions of the external PLL settings used to clock the transmitter and receiver blocks.
- Implement the receiver, transmitter, and PLL in the device by adding your custom megafunctions to the project and compiling the project.
- Simulate the high-speed differential interface design.

Generate an ALTLVDS Receiver and ALTLVDS Transmitter

To generate the LVDS receiver, follow these steps:

1. Open the **altlvds_DesignExample_ex2.zip** file and extract the Quartus II archive project **cii_ALTLVDS_extpll.qar.**
2. In the Quartus II software, open **cii_altlvds_extpll.qar** and restore the archive file into your working directory.
3. Open the top-level block editor file **cii_altlvds_extpll.bdf**.
4. The **cii_altlvds_extpll.bdf** is an incomplete file that you need to complete in the course of this example. The ALTLVDS megafunctions created in this example are added to the top-level file.
5. Double-click anywhere in the white space in the block editor file. The Symbol window appears.
6. In the Symbol window, click **MegaWizard Plug-In Manager**. Page 1 of the parameter editor appears.
7. Select **Create a new custom megafunction variation**.
8. Click **Next**. Page 2a of the parameter editor appears.
9. In the MegaWizard Plug-In Manager pages, select or verify the configuration settings shown in Table 3–9. Click **Next** to advance from one page to the next.

Table 3–9. Configuration Settings for Design Example 2 (LVDS Receiver) (Part 1 of 3)

MegaWizard Plug-In Manager Page	MegaWizard Plug-In Manager Configuration Setting	Value
2a	**Megafunction**	Under **I/O**, select **ALTLVDS_RX**
	Device family	Cyclone II
	Output file type	VHDL
	Output file name	‘rx_block’
	Return to this page for another create operation	Selected

Table 3–9. Configuration Settings for Design Example 2 (LVDS Receiver) (Part 2 of 3)

MegaWizard Plug-In Manager Page	MegaWizard Plug-In Manager Configuration Setting	Value
3	**Currently selected device family**	Cyclone II
	Match project / default	Selected
	Implement Deserializer circuity in logic cells	Selected
	Enable Dynamic Phase Alignment mode	—
	What is the number of channels?	4
	What is the deserialization factor?	8
	Use external PLL	Selected (The PLL must be reset for the output clock phase relationships to be set correctly when the PLL loses lock, or if the PLL input reference clock is not stable when the device completes the configuration process.)
	Use clock pin	—
	Use 'rx_data_reset' input port	Not selected
4	**What is the input data rate?**	—
	Specify input clock rate by	—
	Clock frequency	—
	Clock period	—
	Use shared PLLs (s) for receivers and transmitters	—
	Use 'pll_areset' input port	—
	Use 'rx_pll_enable' input port	—
	Use 'rx_locked' output port	—
	What is the clock resource used for 'rx_outclock'?	—
	What is the phase alignment of 'rx_in' with respect to 'rx_inclock'	—
	Use source-synchronous mode of the PLL	—
	Align clock to center of data window at capture point	—
	Enable self-reset on lost lock in PLL	—
	Enable FIFO for DPA channels	—
5	**Use 'rx_divfwdclk' output port and bypass the DPA FIFO**	—
	What is the simulated recovered clock phase shift?	—
	Use 'rx_dpll_enable' input port	—
	Use 'rx_dpll_hold' input port	—
	Use 'rx_fifo_reset' input port	—

Table 3–9. Configuration Settings for Design Example 2 (LVDS Receiver) (Part 3 of 3)

MegaWizard Plug-In Manager Page	MegaWizard Plug-In Manager Configuration Setting	Value
6	Use 'rx_reset' input port	—
	Automatically reset the bit serial FIFO when 'rx_dpa_locked' rises for the first time	—
	User explicitly resets the bit serial FIFO through 'rx_reset'	—
	Use 'rx_dpa_locked' output port	—
	When phase alignment circuitry switches to a new phase	—
	When there are two phase changes in the same direction	—
	Use 'rx_dpa_lock_reset" input port	—
	Use a DPA initial phase selection of	—
	Align DPA to rising edge of data only	—
7	Enable PLL calibration	—
	Use 'dpa_pll_recal' input port	—
	What is the input data rate? (Mbps)	—
8	Register outputs	Selected
	Use 'rx_cda_reset' input port	—
	Use 'rx_cda_max' output port	—
	After how many pulses does the data alignment circuitry restore the serial data latency back to 0?	—
	Align data to rising edge of clock	—
	Use 'rx_coreclk' input port	—
	Use 'rx_data_align' input port	—
	Enable independent bitslip controls for each channel	—
	Add extra register for 'rx_data_align ' input port	—
	Use 'rx_data_align_reset' input port	—
	Use RAM buffer	—
	Use a multiplexer and synchronization register	—
	Use logic element based RAM buffer	—
9	Generate netlist	Not selected
10	Variation file (.v)	Selected
	Quartus II IP file	Selected
	Quartus II symbol file (.bsf)	Selected
	Instantiation template file (_inst.v)	Selected
	Verilog HDL black box file (_bb.v)	Selected
	AHDL Include file (.inc)	Selected
	VHDL component declaration file (.cmp)	Selected
	PinPlanner ports file (.PPF)	Selected

10. Click **Finish**. The 'rx_block' module is built.

11. Click **OK**. The MegaWizard Plug-In Manager resets to page 2a to allow you to create a new custom megafunction variation for the LVDS transmitter.
12. In the MegaWizard Plug-In Manager pages, select or verify the configuration settings shown in Table 3–10.

Table 3–10. Configuration Settings for Design Example 2 (LVDS Transmitter) (Part 1 of 2)

MegaWizard Plug-In Manager Page	MegaWizard Plug-In Manager Configuration Setting	Value
2a	**Megafunction**	Under **I/O**, select **ALTLVDS**
	Device family	Cyclone II
	Output file type	VHDL
	Output file name	'tx_block'
	Return to this page for another create operation	Selected
3	**Currently selected device family**	Cyclone II
	Match project/default	Selected
	Implement SERDES in logic cells	Selected
	What is the number of channels?	4
	What is the deserialization factor?	8
	Use external PLL	Selected (The PLL must be reset for the output clock phase relationships to be set correctly when the PLL loses lock, or if the PLL input reference clock is not stable when the device completes the configuration process.)
	Use clock pin	—
	Use 'tx_data_reset' input port	Not selected
4	**What is the output data rate?**	—
	Specify input clock rate by	—
	Clock frequency	—
	Clock period	—
	What is the phase alignment of 'tx_in' with respect to the rising edge of 'tx_inclock' (in degrees)	—
	Use 'tx_pll_enable' input port	—
	Use 'pll_areset' input port	—
	Align clock to center of data window	—
	Enable self-reset on loss lock in the PLL	—
	Use shared PLL(s) for receivers and transmitters	—
	Register 'tx_in' input port using	—

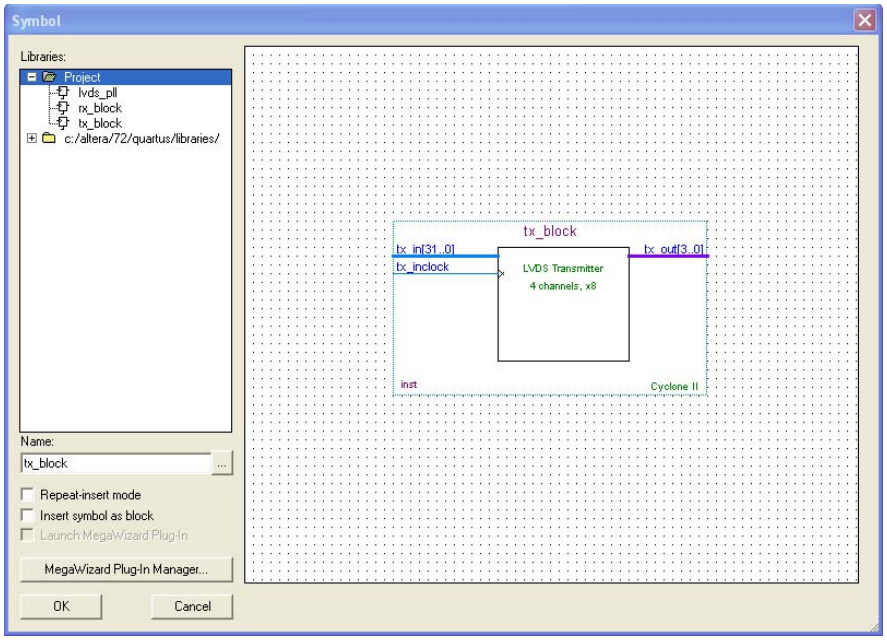
Table 3–10. Configuration Settings for Design Example 2 (LVDS Transmitter) (Part 2 of 2)

MegaWizard Plug-In Manager Page	MegaWizard Plug-In Manager Configuration Setting	Value
5	Use 'tx_outclock' output port	—
	What is the outclock divide factor (B)?	—
	Specify phase alignment of 'tx_outclock' with respect to 'tx_out'	—
	What is the phase alignment of 'tx_outclock' with respect to 'tx_out' (in degrees)	—
	What is the outclock duty cycle?	—
	Use 'tx_locked' output port	—
	Use 'tx_coreclock' output port	—
	What is the clock resource used for 'tx_coreclock'?	—
6	Generate netlist	Not selected
7	Variation file (.v)	Selected
	Quartus II IP file (.qip)	Selected
	Quartus II symbol file (.bsf)	Selected
	Instantiation template file (_inst.v)	Selected
	AHDL Include file (.inc)	Selected
	VHDL component declaration file (.cmp)	Selected
	PinPlanner ports file (.PPF)	Selected

13. Click **Finish**. The 'tx_block' module is built.

The symbol for the ALTLVDS transmitter function you just created appears in the Symbol window (Figure 3–11 on page 3–40).

Figure 3–11. Design Example 2— Symbol View of tx_block Module



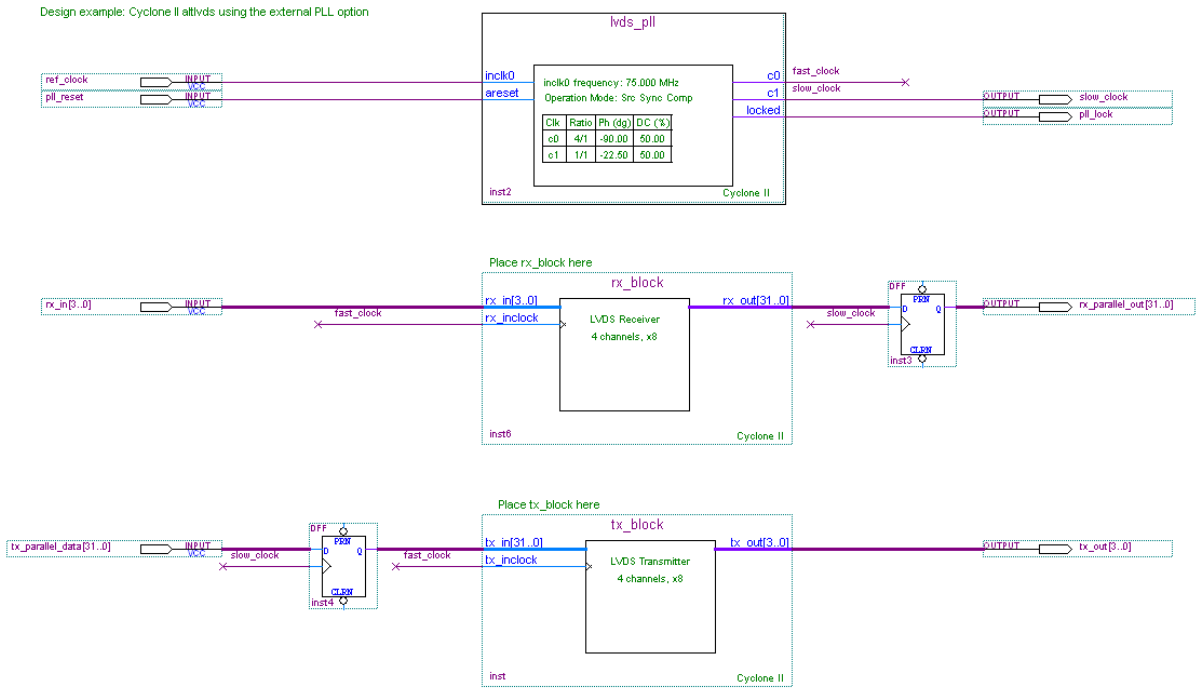
Integrating the ALTLVDS Receiver and Transmitter in the Design

To integrate the ALTLVDS receiver and transmitter in the final design, follow these steps:

1. After the ALTLVDS transmitter function you just created appears in the Symbol window, click **OK** to return to the block design editor with the `tx_block` attached to your cursor.
2. Place the **rx_block** representation in the top-level file under the text **Place tx_block here**, aligning the input and output ports to the existing connections. Left-click to drop the `tx_block` in place.
3. Double-click anywhere in the white space of the design file. The Symbol window appears.
4. Select **rx_block** from the **Project** library list and click **OK**. You need to expand the **Project** folder to see the megafunctions it contains.
5. Place the **rx_block** symbol in the block editor design file under the text **Place rx_block here** and align the input and output ports with the signals already present in the design file.

The rest of the components for this design are already in the top-level file. They include two instantiations of DFF and one instantiation of the ALTPLL megafunction named `lvds_pll` in this example. After you place the `rx_block` and `tx_block` modules, the top level should resemble Figure 3–12 on page 3–41.

Figure 3–12. Design Example 2 —Top-Level Schematic



If you do not want to create the rx_block and tx_block modules, they are located in the backup folder in the project directory. You can move them to the top-level directory and insert them directly into the top-level design.

Parameters Used by the ALTPLL Megafunction

When you instantiate the ALTLVDS megafunctions in external PLL mode, Altera recommends setting up the data rate and clocking with the ALTPLL megafunction. For Cyclone devices, you must select **In normal mode**. For Cyclone II devices, you can select either **In normal mode** or **In source-synchronous compensation mode**. The source-synchronous compensation mode is recommended for Cyclone II devices.

For more information about PLL compensation modes, refer to the *PLL* chapter of the relevant device handbook.

The LVDS receiver and transmitter in this design example operate at a 600-Mbps data rate with deserialization and serialization factors of 8. The reference clock frequency is equal to the data rate divided by the deserialization factor, which is 75 MHz.

Cyclone and Cyclone II devices use DDIO registers as part of the SERDES interface. Because the data is clocked on both the rising and falling edge, the clock frequency must be half the data rate; therefore, the fast clock from the PLL runs at half the data rate. The core clock frequency or slow clock is the data rate divided by the serialization factor (J). In this example, the slow clock is the same frequency as the reference clock, but is phase-aligned to the fast clock from the PLL.

The PLL in this design example is in the source-synchronous compensation mode, using the following selections for the ALTPLL megafunction:

- Select C0 for **Which output clock will be compensated for?**
- Type 75 MHz for **What is the frequency of the inclock0 input?**
- Turn on ports for the asynchronous reset function and locked output.

C0 is the high-speed clock. The data rate for this design example is 600 Mbps. The C0 signal must have a frequency of one half the data rate, so the output frequency must be set to 300 MHz. The phase you select depends on the reference clock to data relationship at the pins of the device. The source-synchronous compensation mode maintains the clock to data relationship to the I/O element capture registers on the receiver.

In this design example, the reference clock and data are rising edge-aligned (the bit boundary is synchronous to the rising edge of the reference clock, and 8 bits are received in every clock period). The C0 signal must be phase-shifted to properly capture the data internally at the registers.

For edge-aligned interfaces, you would normally phase-shift your high speed clock by –180° to center-align the clock and data relationship at the capture register. However, because Cyclone and Cyclone II devices use a half-rate high-speed clock, due to the DDIO implementation, a –180° phase-shift would simply switch the rising edge with the falling edge. To move the capture clock with respect to the DDIO data, a –90° phase shift is required.

C1 is the low-speed clock. For even SERDES factors, it must be the data rate divided by the SERDES factor (J). In this example, the **C1** signal is 75 MHz (600 Mbps/8). The phase-shift on the **C1** clock must correspond to the phase-shift on the C0 signal. The data is edge-aligned with respect to the reference clock at the input pins, so the clocks have to phase-shift to center-align the rising edge in the core. The high-speed-to-low-speed data transfer does not use the DDIO circuitry, so you can calculate –180°/J to determine the correct phase shift for this clock to achieve the desired data alignment. In this example, the result is –22.5°, which you use for the C1 phase shift.

In this example, the receiver and transmitter use the same PLL. The phase shifts are optimized to capture data at the receiver, but they also determine a fixed relationship of clock and data at the transmitter. If the slow clock is forwarded with the transmit data, it is center-aligned. If a different clock frequency and phase are desired, you can enable the C2 output port of the PLL (in Cyclone devices, the third port of the PLL is E0). The frequency and phase are restricted to the possibilities available for the ALTPLL megafunction parameters.

Functional Results—Simulate the ALTLVDS Receiver/Transmitter Design in the ModelSim-Altera Software

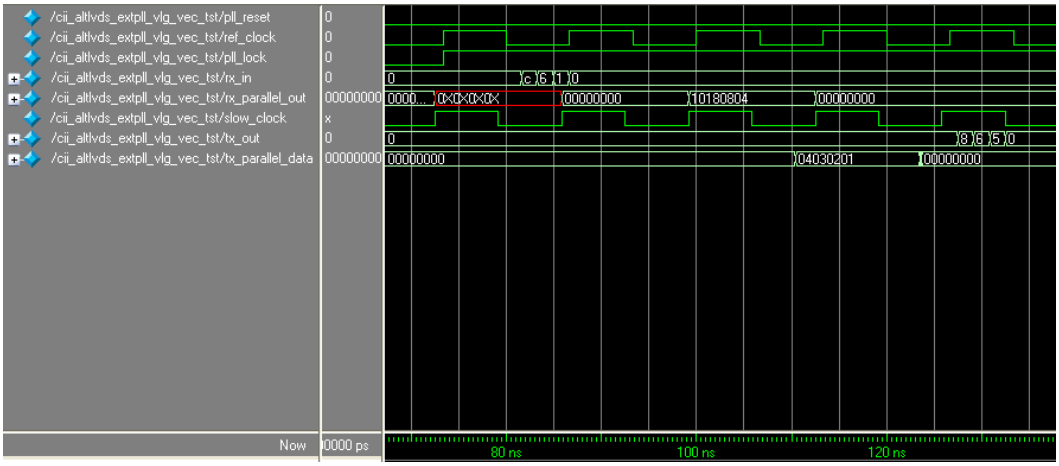
Simulate the design in the ModelSim-Altera software to compare the results of both simulators.

To set up the ModelSim-Altera software, follow these steps:

1. Unzip the **altlvds_ex2_msim.zip** file to any working directory on your PC.
2. Start the ModelSim-Altera software.
3. On the File menu, click **Change Directory**.
4. Select the folder in which you unzipped the files. Click **OK**.
5. On the Tools menu, point to **TCL** and click **Execute Macro**. The **Execute Do File** dialog box appears.
6. Select the **altlvds_ex2_msim.do** file and click **Open**. This is a script file for the ModelSim-Altera software that automates all the necessary settings for the software.
7. Verify the results shown in the Waveform Viewer window.

You can rearrange signals, remove signals, add signals, and change the radix by modifying the script in **altlvds_ex2_msim.do**. Figure 3–13 shows the expected simulation results in the ModelSim-Altera software.

Figure 3–13. Design Example 2—ModelSim-Altera Software Results



The receiver channels bring the serial data into the device, edge-aligned with the reference clock (`ref_clock`). The `rx_in0` signal receives a data value of `1`, the `rx_in1` signal receives a data value of `2`, the `rx_in2` signal receives a data value of `3`, and the `rx_in3` signal receives a data value of `4`. You can expand the parallel receiver channels (`rx_parallel_out`) in the waveform editor to see the word alignment position. The parallel data is shifted two bits toward the MSB in each channel.

The transmitter channel sends data from the device center-aligned with the `slow_clock` output pin. The `tx_out0` signal sends a data value of `1`, the `tx_out1` signal sends a data value of `2`, the `tx_out2` signal sends a data value of `3`, and the `tx_out3` signal sends a data value of `4`. Depending on your system requirements, you can add the remaining output clock port of the PLL and forward a clock that is aligned differently with respect to the data. The timing simulation shows the transmitted data is shifted three bit positions towards the MSB with respect to the `slow_clock` signal.

Design Example 3: Stratix III Soft Clock Data Recovery

With the inclusion of soft CDR circuitry for the CDR feature, Stratix III devices offer support for the widely-used Gigabit Ethernet/SGMII protocol. CDR removes the clock from the clock-embedded data, a capability required for SGMII support. Stratix III devices support a wide array of high-speed protocols and are used to bridge high-speed interfaces.

This design example uses a Stratix III device to receive a single, 1-Gbps LVDS channel using a 50-MHz reference clock to the FPGA core. This design example uses an LVDS transmitter and an LVDS receiver. The transmitter and receiver use separate PLLs. The core clock frequency is 100 MHz. The deserialization factor in both the receiver and the transmitter is 10. The LVDS receiver function uses soft CDR circuitry.

The focus of this design is to illustrate the recovered clock—the clock that is recovered and forwarded to the core by the LVDS receiver.

In this example, the following tasks are completed:

- Generate a high-speed differential receiver in DPA mode and soft CDR mode using the ALTLVDS megafunctions and the parameter editor.
- Generate a high-speed differential transmitter using the ALTLVDS megafunctions and the parameter editor.
- Implement the receiver and transmitter functions in the device by adding your custom megafunctions to the project and compiling the project.
- Simulate the high-speed differential interface design using the ModelSim-Altera software.

Generate an ALTLVDS Receiver and ALTLVDS Transmitter

To generate the LVDS receiver, follow these steps:

1. Open the **altlvds_DesignExample_ex3.zip** file and extract the Quartus II archive project **altlvds_s3_serial_link.qar**.
2. In the Quartus II software, open **altlvds_s3_serial_link.qar** and restore the archive file into your working directory.
3. Open the top-level file **altlvds_s3_serial_link.v**.
4. On the Tools menu, click **MegaWizard Plug-In Manager**. Page 1 of the parameter editor appears.
5. Select **Create a new custom megafunction variation**.
6. Click **Next**. Page 2a of the parameter editor appears.
7. In the MegaWizard Plug-In Manager pages, select or verify the configuration settings shown in Table 3–11. Click **Next** to advance from one page to the next.

Table 3–11. Configuration Settings for Design Example 3 (LVDS Receiver) (Part 1 of 3)

MegaWizard Plug-In Manager Page	MegaWizard Plug-In Manager Configuration Setting	Value
2a	**Megafunction**	Under **I/O**, select **ALTLVDS_RX**
	Device family	Stratix III
	Output file type	Verilog HDL
	Output file name	‘rx_block’
	Return to this page for another create operation	Selected
3	**Currently selected device family**	Stratix III
	Match project/default	Selected
	This module acts as an	LVDS receiver
	Implement SERDES in logic cells	Not selected
	Enable Dynamic Phase Alignment mode (receiver only)	Selected
	What is the number of channels?	1
	What is the deserialization factor?	10
	Use external PLL	Not selected
	Use clock pin	Not selected
	Use ‘rx_data_reset’ input port	Not selected

Table 3–11. Configuration Settings for Design Example 3 (LVDS Receiver) (Part 2 of 3)

MegaWizard Plug-In Manager Page	MegaWizard Plug-In Manager Configuration Setting	Value
4	What is the input data rate?	1 Gbps (1,000 Mbps)
	Specify input clock rate by	Clock frequency
	Clock frequency	50 MHz
	Clock period	—
	Use shared PLL(s) for receivers and transmitters	Not Selected
	Use 'pll_areset' input port	Not selected
	Use 'rx_pll_enable' input port	—
	Use 'rx_locked' output port	Selected
	What is the clock resource used for 'rx_outclock'?	Auto selection
	What is the phase alignment of 'rx_in' with respect to 'rx_inclock'	—
	Use source-synchronous mode of the PLL	—
	Align clock to center of data window at capture point	—
	Enable self-reset on lost lock in PLL	—
	Enable FIFO for DPA channels	—
5	Use 'rx_divfwdclk' output port and bypass the DPA FIFO	Selected
	What is the simulated recovered clock phase drift?	0 PPM
	Use 'rx_dpll_enable' input port	—
	Use 'rx_dpll_hold' input port	—
	Use 'rx_fifo_reser' input port	—
6	Use 'rx_reset' input port	—
	Automatically reset the bit serial FIFO when 'rx_dpa_locked' rises for the first time	—
	User explicitly resets the bit serial FIFO through 'rx_reset'	—
	Use 'rx_dpa_locked' output port	—
	When phase alignment circuitry switches to a new phase	—
	When there are two phase changes in same direction	—
	Use 'rx_dpa_lock_reset" input port	—
	Use a DPA initial phase selection of	—
	Align DPA to rising edge of data only	—
7	Enable PLL calibration	—
	Use 'dpa_pll_recal' input port	—
	What is the input data rate? (Mbps)	—

Table 3–11. Configuration Settings for Design Example 3 (LVDS Receiver) (Part 3 of 3)

MegaWizard Plug-In Manager Page	MegaWizard Plug-In Manager Configuration Setting	Value
8	Register outputs	Selected
	Use 'rx_cda_reset' input port	—
	Use 'rx_cda_max' output port	—
	After how many pulses does the data alignment circuitry restore the serial data latency back to 0?	—
	Align data to rising edge of clock	Selected
	Use 'rx_coreclk' input port	—
	Use 'rx_channel_data_align' input port	Not selected
	Enable independent bitslip control	—
	Add extra register for 'rx_data_align' input port	—
	Use 'rx_data_align_reset' input port	—
	Use RAM buffer	—
	Use a multiplexer and synchronization register	—
	Use logic element based RAM buffer	—
9	Generate netlist	Not selected
10	Variation file (.v)	Selected
	Quartus II IP file	Selected
	Quartus II symbol file (.bsf)	Selected
	Instantiation template file (_inst.v)	Selected
	Verilog HDL black box file (_bb.v)	Selected
	AHDL Include file (.inc)	Selected
	VHDL component declaration file (.cmp)	Selected
	PinPlanner ports file (.PPF)	Selected

8. Click **Finish**. The 'rx_block' module is built.
9. Click **OK**. The MegaWizard Plug-In Manager resets to page 2a to allow you to create a new custom megafunction variation for the LVDS transmitter.
10. In the MegaWizard Plug-In Manager pages, select or verify the configuration settings shown in Table 3–12.

Table 3–12. Configuration Settings for Design Example 3 (LVDS Transmitter) (Part 1 of 3)

MegaWizard Plug-In Manager Page	MegaWizard Plug-In Manager Configuration Setting	Value
2a	Megafunction	Under **I/O**, select **ALTLVDS_TX**
	Device family	Stratix III
	Output file type	Verilog HDL
	Output file name	'tx_block'
	Return to this page for another create operation	Not selected

Table 3–12. Configuration Settings for Design Example 3 (LVDS Transmitter) (Part 2 of 3)

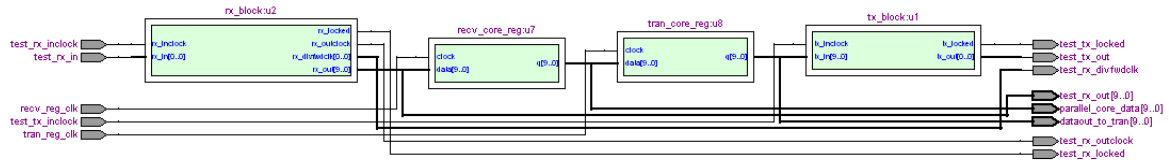
MegaWizard Plug-In Manager Page	MegaWizard Plug-In Manager Configuration Setting	Value
3	Currently selected device family	Stratix III
	Match project/default	Selected
	Implement SERDES in logic cells	Not selected
	Enable Dynamic Phase Alignment mode (receiver only)	Not available
	What is the number of channels?	1
	What is the deserialization factor	10
	Use external PLL	Not selected
	Use clock pin	—
	Use 'tx_data_reset' input port	Not selected
4	What is the output data rate?	1 Gbps (1,000 Mbps)
	Specify input clock rate by	Clock frequency
	Clock frequency	50 MHz
	Clock period	—
	What is the phase alignment of 'tx_in' with respect to the rising edge of 'tx_inclock'?	0
	Use 'tx_pll_enable' input port	—
	Use 'pll_areset' input port	Not selected
	Align clock to center of data window	Selected
	Enable self-reset on loss lock in the PLL	—
	Use shared PLL(s) for receivers and transmitters	Not selected
	Register 'tx_in' input port using	'tx_coreclock'
5	Use 'tx_outclock' output port	Not selected
	What is the outclock divide factor (B)?	—
	Specify phase alignment of 'tx_outclock' with respect to 'tx_out'	—
	What is the phase alignment of 'tx_outclock' with respect to 'tx_out' (in degrees)	0
	What is the outclock duty cycle?	—
	Use 'tx_locked' output port	Selected
	Use 'tx_coreclock' output port	Not selected
	What is the clock resource used for 'tx_coreclock'?	—
6	Generate netlist	Not selected

Table 3–12. Configuration Settings for Design Example 3 (LVDS Transmitter) (Part 3 of 3)

MegaWizard Plug-In Manager Page	MegaWizard Plug-In Manager Configuration Setting	Value
7	**Variation file (.v)**	Selected
	Quartus II IP file	Selected
	Quartus II symbol file (.bsf)	Selected
	Instantiation template file (_inst.v)	Selected
	Verilog HDL black box file (_bb.v)	Selected
	AHDL Include file (.inc)	Selected
	VHDL component declaration file (.cmp)	Selected
	PinPlanner ports file (.PPF)	Selected

11. Click **Finish**. The `tx_block` module is built.
12. On the File menu, click **Save Project**.
13. On the Processing menu, click **Start Compilation**.
14. When the **Full Compilation was successful** message box appears, click **OK**.
15. On the Tools menu, select **Netlist Viewers** and click **RTL Viewer** to view the design schematic in the **RTL Viewer**. The block diagram file should resemble Figure 3–14.

Figure 3–14. Design Example 3 in RTL Viewer



16. On the Processing menu, click **Compilation Report** to view the resource usage of this design.

Functional Results—Simulate the ALTLVDS Receiver/Transmitter Design in the ModelSim-Altera Software

Simulate the design in the ModelSim-Altera software to generate a waveform display of the device behavior.

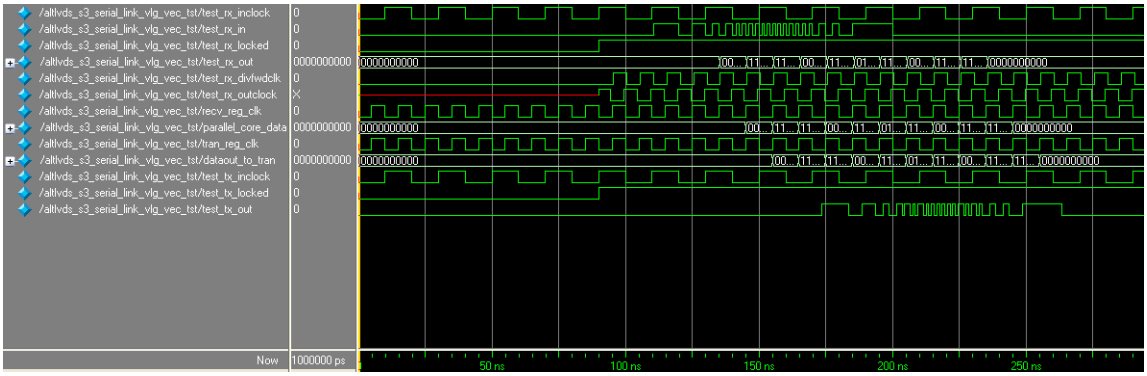
To set up the ModelSim-Altera software, follow these steps:

1. Unzip the **altlvds_ex3_msim.zip** file to any working directory on your PC.
2. Start the ModelSim-Altera software.
3. On the File menu, click **Change Directory**.
4. Select the folder in which you unzipped the files. Click **OK**.
5. On the Tools menu, point to **TCL** and click **Execute Macro**. The **Execute Do File** dialog box appears.

6. Select the **altlvds_ex3_msim.do** file and click **Open**. This is a script file for the ModelSim-Altera software that automates all the necessary settings for the simulation.
7. Verify the results shown in the Waveform Viewer window.

You can rearrange signals, remove signals, add signals, and change the radix by modifying the script in **altlvds_ex1_msim.do**. Figure 3–15 on page 3–50 shows the expected simulation results in the ModelSim-Altera software.

Figure 3–15. Design Example 3 ModelSim-Altera Software Results



This waveform file contains only the input and output pins that are shown. You can add internal post-compilation nodes with the **Node Finder** to view all data paths in the design, including the soft CDR data path.

This software output confirms that 1-Gbps data can be successfully received using the recovered clock from the soft CDR and then transmitted from the Stratix III device. You can observe this by comparing the values on the `text_rx_in` and `text_tx_out` ports.

Design Example 4: Stratix III ALTLVDS Receiver with DPA PLL Calibration

In Quartus II software version 9.0 and later, a new feature called DPA PLL calibration is added to the ALTLVDS megafunctions to overcome the DPA misalignment issue found in the supported devices when using the previous version of the Quartus II software.

For details about the DPA misalignment issue, refer to "DPA PLL Calibration in Stratix III and Stratix IV ES Devices" on page 3–4.

This design example uses a Stratix III device to receive a single, 1,250-Mbps LVDS channel using a 125-MHz reference clock from an external PLL. This design contains only the LVDS receiver module with a deserialization factor of 10.

The focus of this design example is to illustrate the settings needed to use the DPA PLL calibration feature in the Stratix III device.

In this example, the following tasks are completed:

- Generate a high-speed differential receiver in DPA mode and with DPA PLL calibration enabled using the ALTLVDS_RX/_TX parameter editor.

- Implement the receiver in the device by adding your custom megafunctions to the project and compiling the project.
- Simulate the high-speed differential interface design using the ModelSim-Altera software.

Generate an ALTLVDS Receiver

To generate the LVDS receiver, perform the following steps:

1. Open the **altlvds_DesignExample_ex4.zip file** and extract the Quartus II archive project **altlvds_pll_calib.qar**.
2. In the Quartus II software, open **altlvds_pll_calib.qar** and restore the archive file into your working directory.
3. Open the top-level file **altlvds_pll_calib.bdf**.
4. The file **altlvds_pll_calib.bdf** is an incomplete file that you have to complete in this example. The ALTLVDS megafunctions created in this example is added to the top-level file.
5. Double-click anywhere in the white space in the block editor file. The Symbol window appears.
6. In the Symbol window, click on the **MegaWizard Plug-In Manager**. Page 1 of the parameter editor appears.
7. Select **Create a new custom megafunction variation**.
8. Click **Next**. Page 2a of the parameter editor appears.
9. In the parameter editor pages, select or verify the configuration settings shown in Table 3–13. Click **Next** to advance from one page to the next.

Table 3–13. Configuration Settings for Design Example 4 (LVDS Receiver) (Part 1 of 3)

Parameter Editor Page	Parameter Editor Configuration Setting	Value
2a	**Megafunction**	Under **I/O**, select **ALTLVDS_RX**
	Device family	Stratix III
	Output file type	Verilog HDL
	Output file name	rx_block
	Return to this page for another create operation	Not selected

Table 3–13. Configuration Settings for Design Example 4 (LVDS Receiver) (Part 2 of 3)

Parameter Editor Page	Parameter Editor Configuration Setting	Value
3	Currently selected device family	Stratix III
	Match project/default	Selected
	This module acts as an	LVDS receiver
	Implement SERDES in logic cells	Not selected
	Enable Dynamic Phase Alignment mode (receiver only)	Selected
	What is the number of channels?	1
	What is the deserialization factor?	10
	Use external PLL	Selected
	Use clock pin	—
	Use 'rx_data_reset' input port	—
4	What is the input data rate?	1,250 Mbps
	Specify input clock rate by	Clock frequency
	Clock frequency	500 MHz
	Clock period	—
	Use shared PLLs (s) for receivers and transmitters	—
	Use 'pll_areset' input port	—
	Use 'rx_pll_enable' input port	—
	Use 'rx_locked' output port	—
	What is the clock resource for 'rx_outclock'?	Auto selection
	What is the phase alignment of 'rx_in' with respect to 'rx_inclock'	—
	Use source-synchronous mode of the PLL	Selected
	Align clock to center of data window at capture point	Selected
	Enable self-reset on lost lock in PLL	—
	Enable FIFO for DPA channels	—
5	Use 'rx_divfwdclk' output port and bypass the DPA FIFO	Not selected
	What is the simulated recovered clock phase shift?	—
	Use 'rx_dpll_enable' input port	—
	Use 'rx_dpll_hold' input port	Not selected
	Use 'rx_fifo_reset' input port	Not selected

Table 3–13. Configuration Settings for Design Example 4 (LVDS Receiver) (Part 3 of 3)

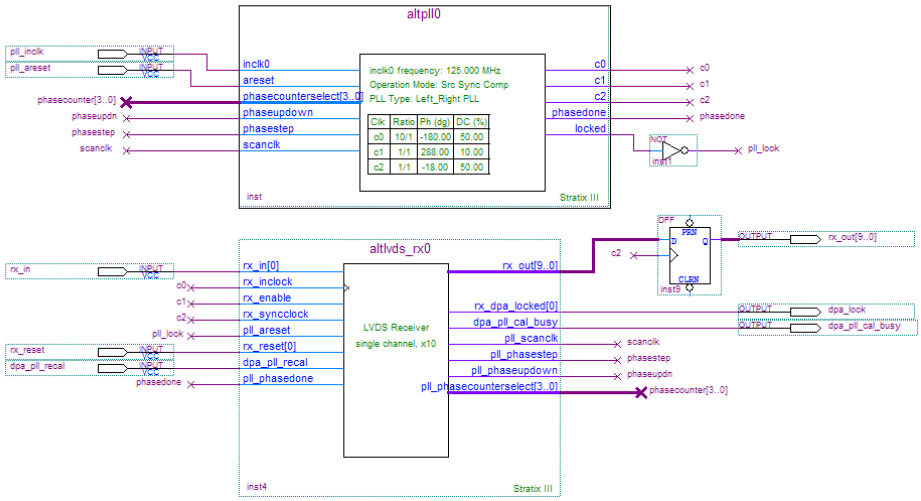
Parameter Editor Page	Parameter Editor Configuration Setting	Value
6	Use 'rx_reset' input port	Selected
	Automatically reset the bit serial FIFO when 'rx_dpa_locked' rises for the first time	Selected
	User explicitly resets the bit serial FIFO through 'rx_reset'	—
	Use 'rx_dpa_locked' output port	Selected
	When phase alignment circuitry switches to new phase	—
	When there are two phase change…	—
	Use 'rx_dpa_lock_reset" input port	Not selected
	Use a DPA initial phase selection of	Not selected
	Align DPA to rising edge of data only	Not selected
7	Enable PLL calibration	Selected
	Use 'dpa_pll_recal' input port	Selected
	What is the input data rate? (Mbps)	—
8	Register outputs	—
	Use 'rx_cda_reset' input port	—
	Use 'rx_cda_max' output port	—
	After how many pulses does the data alignment circuitry restore the serial data latency back to 0?	—
	Align data to rising edge of clock	—
	Use 'rx_coreclk' input port	—
	Use 'rx_channel_data_align' input port	Not selected
	Enable independent bitslip controls for each channel	—
	Add extra register for 'rx_data_align' input port	—
	Use 'rx_data_align_reset' input port	—
	Use RAM buffer	—
	Use a multiplexer and synchronization register	—
	Use logic element based RAM buffer	—
9	Generate netlist	Not selected
10	Variation file (.v)	Selected
	Quartus II IP file	Selected
	Quartus II symbol file (.bsf)	Selected
	Instantiation template file (_inst.v)	Selected
	Verilog HDL black box file (_bb.v)	Selected
	AHDL Include file (.inc)	Selected
	VHDL component declaration file (.cmp)	Selected
	PinPlanner ports file (.PPF)	Selected

10. Click **Finish**. The `rx_block` module is built. The symbol for the ALTLVDS receiver you just created appears in the Symbol window.
11. Click **OK** to return to the block design editor with the `rx_block` module attached to your cursor.
12. Place the `rx_block` module in the top-level file under the text **Place rx_block here**, aligning the input and output ports to the existing connections. Left-click to drop the `rx_block` module in place.

The ALTPLL instantiation that is used as the external PLL for this design is already in the top-level file. For details about the PLL settings, refer to "Settings for the External PLL" on page 3–54.

Figure 3–16 on page 3–54 shows a completed top-level schematic with the `rx_block` module connected to the external PLL.

Figure 3–16. Design Example 4—Top-Level Schematic



Settings for the External PLL

The following section lists the details of the external PLL settings for this design example.

Parameter Settings:

- Set the device speed grade to 3.
- Frequency of the `inclock0` input is 125 MHz.
- Select `Left_Right` PLL for the PLL type.
- Select feedback path inside the PLL in Source-Synchronous Compensation Mode.
- Leave other parameters as default.

PLL Reconfiguration:

Select optional inputs for dynamic phase reconfiguration without enabling the phase-shift step resolution edit.

Output Clocks

- Clk c0:
  - This is the high-speed serial clock (fast clock) you must connect to the `rx_inclock` of the ALTLVDS receiver megafunction
  - Output frequency: `data_rate` = 1,250 MHz
  - Phase shift: -180°
  - Duty cycle: 50%
  - Select **Use these clock settings for DPA clock** option. You must turn on this option when DPA is used in the ALTLVDS receiver

    For the Quartus II 8.0 software or later, select **Use these clock settings for DPA clock** option in the **Output Clocks** setting tab of the ALTPLL megafunction. Apply this setting on the output clock, which is used as the high-speed serial clock or fast clock.

- Clk c1:
  - This is the load-enable signal that you need to connect to the `rx_enable` of the ALTLVDS receiver megafunction
  - Output frequency: `data_rate/deserialization_factor` = 1,250/10 = 125 MHz
  - Phase shift: `(deserialization_factor - 2)/deserialization_factor` * 360° = (10 - 2)/10 * 360 = 288°
  - Duty cycle: 50%
- Clk c2:
  - This is the clock signal to be connected to the `rx_synclock` of the ALTLVDS receiver megafunction and for the synchronization register.
  - Output frequency: `data_rate/deserialization_factor` = 1250/10 = 125 MHz
  - Phase shift: `-180/deserialization_factor` = -180/10 = -18‘°
  - Duty cycle: 50%

Functional Results—Simulate the ALTLVDS Receiver Design in the ModelSim-Altera Software

Simulate the design in the ModelSim-Altera software to generate a waveform display of the device behavior.

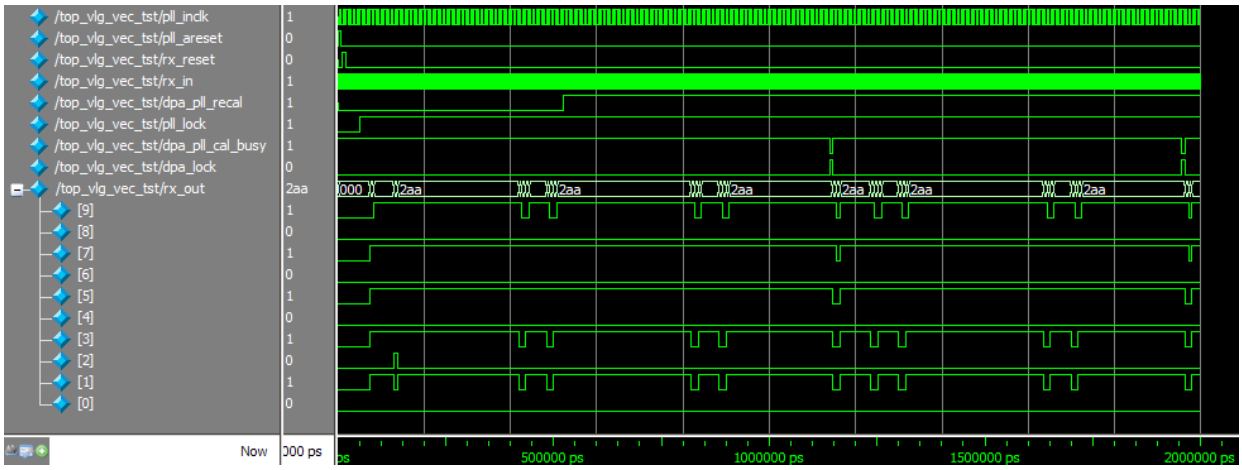
To set up the ModelSim-Altera software, follow these steps:

1. Unzip the **altlvds_ex4_msim.zip** file to any working directory on your PC.
2. Start the ModelSim-Altera software.
3. On the File menu, click **Change Directory**.
4. Select the folder in which you unzipped the files. Click **OK**.

5. On the Tools menu, point to **TCL** and click **Execute Macro**. The **Execute Do File** dialog box appears.
6. Select the **altlvds_ex4_msim.do** file and click **Open**. This is a script file for the ModelSim-Altera software that automates all necessary settings for the simulation.
7. Verify the results shown in the Waveform Viewer window.

You can rearrange signals, remove signals, add signals, and change the radix by modifying the script in **altlvds_ex4_msim.do**. Figure 3–17 shows the expected simulation results in the ModelSim-Altera software.

Figure 3–17. Design Example 4—Modelsim-Altera Simulation Results



This waveform figure contains only the input and output pins. You can add internal post-compilation nodes with the **Node Finder** to view all of the data paths in the design. The simulation result shows the top level of the DPA PLL calibration process. Once the PLL is reset (`pll_areset` asserted), the PLL calibration process starts (`dpa_pll_cal_busy` goes high during calibration). The LVDS receiver monitors the `rx_in` port for 256 data transitions, and then shifts the PLL phase forward by two fast clock phases. The LVDS receivers waits for another 256 data transitions, and then shifts the PLL phase backward by two fast clock phases so that the PLL timing returns to normal. Finally, the LVDS receiver waits for another 256 data transitions, then locks onto the optimum phase. Observe that the `dpa_lock` port is asserted once the DPA locks onto the optimum phase, and the `dpa_pll_cal_busy` signal is deasserted (means that no PLL calibration is in progress).

Design Example 5: Implementing Force Merging of PLLs in Stratix III Device

This design example uses a Stratix III device to transmit and receive two 1,250-Mbps LVDS channels with an input clock of 125 MHz, and deserialization factor of 10. The receiver block has the DPA PLL calibration feature turned on.

This example is a step-by-step guide to create an LVDS receiver and transmitter using the ALTLVDS parameter editor, and then using the Assignment Editor to merge the internal PLLs together to save device resources.

In this example, the following tasks are completed:

- Generate a high-speed differential transmitter using the ALTLVDS parameter editor.
- Generate a high-speed differential receiver in DPA mode and with DPA PLL Calibration enabled using the ALTLVDS parameter editor.
- Merge the PLLs together, compile the design, and analyze the resource usage report.

Restoring the Quartus II Project Archive

To restore the project archive of the design example, perform the following steps:

1. Open the **altlvds_DesignExample_ex5.zip** file and extract the Quartus II archive project **altlvds_pll_merge.qar**.
2. In the Quartus II software, open **altlvds_pll_merge.qar** file and restore the archive file into your working directory.
3. Open the top-level file **altlvds_pll_merge.bdf**.

The file **altlvds_pll_merge.bdf** is an incomplete file that you need to complete in this example. The LVDS receiver and transmitter blocks that you are going to create in this example are added to the top-level file.

Generate an ALTLVDS Transmitter

To generate the ALTLVDS transmitter block, perform the following steps:

1. Double click anywhere in the white space in the top-level file **altlvds_pll_merge.bdf**. The Symbol window appears.
2. In the Symbol window, click on the MegaWizard Plug-In Manager. Page 1 of the MegaWizard Plug-In Manager appears.
3. Select **Create a new custom megafunction variation**.
4. Click **Next**. Page 2a of the MegaWizard Plug-In Manager appears.

5. In the MegaWizard Plug-In Manager pages, select or verify the configuration settings shown in Table 20. Click **Next** to advance from one page to the next.

Table 3–14. Configuration Settings for Design Example 3 (LVDS Transmitter) (Part 1 of 2)

MegaWizard Plug-In Manager Page	MegaWizard Plug-In Manager Configuration Setting	Value
2a	**Megafunction**	Under **I/O**, select **ALTLVDS_TX**
	Device family	Stratix III
	Output file type	Verilog HDL
	Output file name	'tx_block'
	Return to this page for another create operation	Not selected
3	**Currently selected device family**	Stratix III
	Match project/default	Selected
	Implement SERDES in logic cells	Not selected
	What is the number of channels?	1
	What is the deserialization factor	10
	Use external PLL	Not selected
	Use clock pin	Not selected
	Use 'tx_data_reset' input port	Not selected
4	**What is the output data rate?**	1,250 Mbps
	Specify input clock rate by	Clock frequency
	Clock frequency	125 MHz
	Clock period	—
	What is the phase alignment of 'tx_in' with respect to the rising edge of 'tx_inclock'?	0
	Use 'tx_pll_enable' input port	—
	Use 'pll_areset' input port	Selected
	Align clock to center of data window	Selected
	Enable self-reset on loss lock in the PLL	—
	Use shared PLL(s) for receivers and transmitters	Selected
	Register 'tx_in' input port using	'tx_coreclock'
5	**Use 'tx_outclock' output port**	Not selected
	What is the outclock divide factor (B)?	—
	Specify phase alignment of 'tx_outclock' with respect to 'tx_out'	—
	What is the phase alignment of 'tx_outclock' with respect to 'tx_out' (in degrees)	0
	What is the outclock duty cycle?	—
	Use 'tx_locked' output port	Not selected
	Use 'tx_coreclock' output port	Not selected
	What is the clock resource used for 'tx_coreclock'?	—
6	**Generate netlist**	Not selected

Table 3–14. Configuration Settings for Design Example 3 (LVDS Transmitter) (Part 2 of 2)

MegaWizard Plug-In Manager Page	MegaWizard Plug-In Manager Configuration Setting	Value
7	**Variation file (.v)**	Selected
	Quartus II IP file	Selected
	Quartus II symbol file (.bsf)	Selected
	Instantiation template file (_inst.v)	Selected
	Verilog HDL black box file (_bb.v)	Selected
	AHDL Include file (.inc)	Selected
	VHDL component declaration file (.cmp)	Selected
	PinPlanner ports file (.PPF)	Selected

6. Click **Finish**. The 'lvds_tx' module is built.
7. Click OK.
8. Place the **lvds_tx** symbol in the top-level **altlvds_pll_merge.bdf** under the text **INSERT LVDS_TX HERE**, aligning the input and output ports with the signals already present in the design file. For a completed design schematic, refer to Figure 3–18 on page 3–62.
9. On the File menu, Click **Save**.

Generate an ALTLVDS Receiver

To generate the LVDS receiver, perform the following steps:

1. Double-click anywhere in the white space in the top-level file, **altlvds_pll_merge.bdf**. The Symbol window appears.
2. In the Symbol window, click on the **MegaWizard Plug-In Manager**. Page 1 of the parameter editor appears.
3. Select **Create a new custom megafunction variation**.
4. Click **Next**. Page 2a of the parameter editor appears.
5. In the parameter editor pages, select or verify the configuration settings shown in Table 3–13. Click **Next** to advance from one page to the next.

Table 3–15. Configuration Settings for Design Example 5 (LVDS Receiver) (Part 1 of 3)

Parameter Editor Page	Parameter Editor Configuration Setting	Value
2a	**Megafunction**	Under **I/O**, select **ALTLVDS_RX**
	Device family	Stratix III
	Output file type	Verilog HDL
	Output file name	rx_block
	Return to this page for another create operation	Not selected

Table 3–15. Configuration Settings for Design Example 5 (LVDS Receiver) (Part 2 of 3)

Parameter Editor Page	Parameter Editor Configuration Setting	Value
3	Currently selected device family	Stratix III
	Match project/default	Selected
	This module acts as an	LVDS receiver
	Implement SERDES in logic cells	Not selected
	Enable Dynamic Phase Alignment mode (receiver only)	Selected
	What is the number of channels?	2
	What is the deserialization factor?	10
	Use external PLL	Not selected
	Use clock pin	—
	Use 'rx_data_reset' input port	—
4	What is the input data rate?	1,250 Mbps
	Specify input clock rate by	Clock frequency
	Clock frequency	125 MHz
	Clock period	—
	Use shared PLLs (s) for receivers and transmitters	Selected
	Use 'pll_areset' input port	Not selected
	Use 'rx_pll_enable' input port	—
	Use 'rx_locked' output port	Not selected
	What is the clock resource for 'rx_outclock'?	Auto selection
	What is the phase alignment of 'rx_in' with respect to 'rx_inclock'	—
	Use source-synchronous mode of the PLL	Selected
	Align clock to center of data window at capture point	Selected
	Enable self-reset on lost lock in PLL	—
	Enable FIFO for DPA channels	—
5	Use 'rx_divfwdclk' output port and bypass the DPA FIFO	Not selected
	What is the simulated recovered clock phase shift?	—
	Use 'rx_dpll_enable' input port	—
	Use 'rx_dpll_hold' input port	—
	Use 'rx_fifo_reset' input port	—

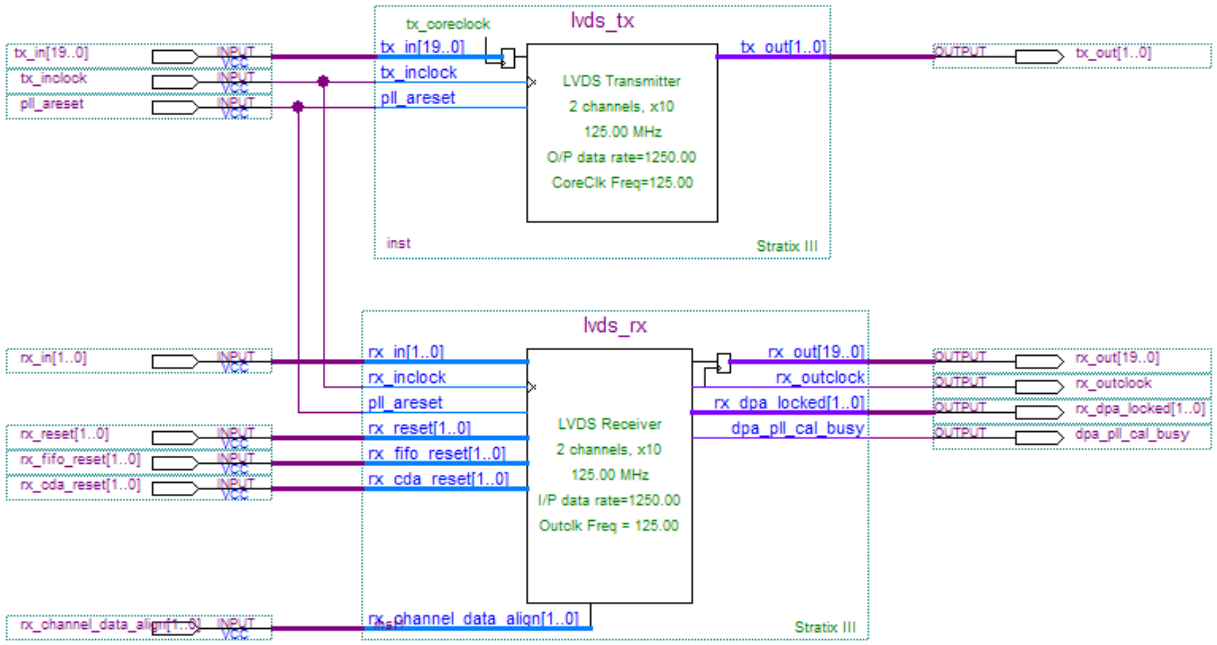
Table 3–15. Configuration Settings for Design Example 5 (LVDS Receiver) (Part 3 of 3)

Parameter Editor Page	Parameter Editor Configuration Setting	Value
6	Use 'rx_reset' input port	Selected
	Automatically reset the bit serial FIFO when 'rx_dpa_locked' rises for the first time	Selected
	User explicitly resets the bit serial FIFO through 'rx_reset'	—
	Use 'rx_dpa_locked' output port	Selected
	When phase alignment circuitry switches to new phase	—
	When there are two phase change…	—
	Use 'rx_dpa_lock_reset" input port	Not selected
	Use a DPA initial phase selection of	Not selected
	Align DPA to rising edge of data only	Not selected
7	Enable PLL calibration	Selected
	Use 'dpa_pll_recal' input port	Not selected
	What is the input data rate? (Mbps)	—
8	Register outputs	Selected
	Use 'rx_cda_reset' input port	Selected
	Use 'rx_cda_max' output port	Not selected
	After how many pulses does the data alignment circuitry restore the serial data latency back to 0?	10
	Align data to rising edge of clock	Selected
	Use 'rx_coreclk' input port	—
	Use 'rx_channel_data_align' input port	Not selected
	Enable independent bitslip controls for each channel	—
	Add extra register for 'rx_data_align' input port	—
	Use 'rx_data_align_reset' input port	—
	Use RAM buffer	—
	Use a multiplexer and synchronization register	—
	Use logic element based RAM buffer	—
9	Generate netlist	Not selected
10	Variation file (.v)	Selected
	Quartus II IP file	Selected
	Quartus II symbol file (.bsf)	Selected
	Instantiation template file (_inst.v)	Selected
	Verilog HDL black box file (_bb.v)	Selected
	AHDL Include file (.inc)	Selected
	VHDL component declaration file (.cmp)	Selected
	PinPlanner ports file (.PPF)	Selected

6. Click **Finish**. The `rx_block` module is built. The symbol for the ALTLVDS receiver you just created appears in the Symbol window.
7. Click **OK** to return to the block design editor with the `rx_block` module attached to your cursor.
8. Place the `lvds_rx` module in the top-level **altlvds_pll_merge.bdf** file under the text **INSERT LVDS_RX HERE**, aligning the input and output ports to the existing connections. Left-click to drop the `rx_block` module in place. For a completed design schematic, refer to Figure 3–18 on page 3–62.
9. On the File menu, click **Save**.

Figure 3–18 shows a completed top-level schematic.

Figure 3–18. Completed Design Schematic



Merging the PLLs

To merge the PLLs, follow these steps:

1. Ensure that you have completed the steps for top-level **altlvds_pll_merge.bdf** file as shown in "Restoring the Quartus II Project Archive" on page 3–57 through "Generate an ALTLVDS Receiver" on page 3–59.
2. On the Processing menu, point to **Start**, and click **Start Analysis & Elaboration**.
3. When the **Analysis & Elaboration was successful** message box appears, click **OK**.
4. On the Assignments menu, click **Assignment Editor**. The Assignment Editor tool appears.
5. On the **Category** bar of the Assignment Editor tool, select **All** from the drop down menu.

6. In the Assignment Editor spreadsheet, right-click the first row in the **From** column, and click **Node Finder**. The Node Finder window appears.
7. Click **Post-Synthesis** in the **Filter** drop-down menu.
8. In the **Look in** field, browse the **lvds_tx:inst hierarchy** and select **lvds_tx_lvds_tx:auto_generated**. Click **OK**.
9. In the **Nodes Found** list, select **pll**, and click on the  button to copy the node to the **Selected Nodes** list. Click **OK**.
10. In the Assignment Editor spreadsheet, right-click the first row in the **To** column, and click **Node Finder**. The Node Finder window appears.
11. Select **Post-Synthesis** in the **Filter** drop-down menu.
12. In the **Look in** field, browse the **lvds_rx:inst1** hierarchy and select **lvds_rx_lvds_rx:auto_generated**. Click **OK**.
13. In the **Nodes Found list**, select **lvdspll0**, and click on the  button to copy the node to the **Selected Nodes** list. Click **OK**.
14. Double click the first row in the **Assignment Name** column. Select **Force Merging of PLLs**.
15. Set the first row in the **Value** column to **On**.
16. Set the first row in the **Enabled** column to **Yes**.
17. On the File menu, click **Save**.

Running Full Compilation

To run full compilation, follow these steps:

1. On the Processing menu, click **Start Compilation**.
2. When the **Full Compilation was successful** message box appears, click **OK**.

Analyzing the Resource Usage Report

To see how many PLLs are used in your design, follow these steps:

1. Make sure you have run a full compilation on your design.
2. In the Processing menu, click **Compilation Report**. The Compilation Report window appears.
3. Click the **Fitter** folder in the report. The Fitter summary appears. Observe that your design only uses one PLL (refer to Figure 3–19 on page 3–64).

Figure 3–19. Fitter Summary (with the Force Merging of PLLs Assignment)

Revision Name	altlvds_merge
Top-level Entity Name	altlvds_merge
Family	Stratix III
Device	EP3SE50F484I3
Timing Models	Final
Met timing requirements	N/A
Logic utilization	< 1 %
Combinational ALUTs	117 / 38,000 (< 1 %)
Memory ALUTs	0 / 19,000 (0 %)
Dedicated logic registers	166 / 38,000 (< 1 %)
Total registers	166
Total pins	63 / 296 (21 %)
Total virtual pins	0
Total block memory bits	0 / 5,455,872 (0 %)
DSP block 18-bit elements	0 / 384 (0 %)
Total PLLs	1 / 4 (25 %)
Total DLLs	0 / 4 (0 %)

4. Disable the **Force Merging of PLLs** assignment, and repeat step in "Running Full Compilation" on page 3–63 through "Analyzing the Resource Usage Report" on page 3–63. Observe that your design uses two PLLs instead of one (refer to Figure 3–20).

Figure 3–20. Fitter Summary (with the Force Merging of PLLs Assignment)

Revision Name	altlvds_merge
Top-level Entity Name	altlvds_merge
Family	Stratix III
Device	EP3SE50F484I3
Timing Models	Final
Logic utilization	< 1 %
Combinational ALUTs	116 / 38,000 (< 1 %)
Memory ALUTs	0 / 19,000 (0 %)
Dedicated logic registers	165 / 38,000 (< 1 %)
Total registers	165
Total pins	63 / 296 (21 %)
Total virtual pins	0
Total block memory bits	0 / 5,455,872 (0 %)
DSP block 18-bit elements	0 / 384 (0 %)
Total PLLs	2 / 4 (50 %)
Total DLLs	0 / 4 (0 %)

For more information about the PLL parameter settings, refer to the **PLL Summary report** in the Resource Section folder, in the Fitter folder.

Prototypes and Component Declarations

This section describes the prototypes and component declarations of the ALTLVDS_TX and ALTLVDS_RX megafunctions.

1. Verilog HDL Prototype for ALTLVDS_RX Megafunction

The following Verilog HDL prototype is located in the Verilog Design File (.**v**) **altera_mf.v** in the <*Quartus II installation directory*>**\eda\synthesis** directory.

```
module altlvds_rx
#( parameter    buffer_implementation = "RAM",
   parameter    cds_mode = "UNUSED",
   parameter    common_rx_tx_pll = "ON",
   parameter    data_align_rollover = 4,
   parameter    deserialization_factor = 4,
   parameter    intended_device_family = "unused",
   parameter    dpa_initial_phase_value = 0,
   parameter    dpll_lock_count = 0,
   parameter    dpll_lock_window = 0,
   parameter    enable_dpa_align_to_rising_edge_only = "OFF",
   parameter    enable_dpa_calibration = "ON",
   parameter    enable_dpa_fifo = "OFF",
   parameter    enable_dpa_initial_phase_selection = "OFF",
   parameter    enable_dpa_mode = "OFF",
   parameter    enable_dpa_pll_calibration = "OFF",
   parameter    enable_soft_cdr_mode = "OFF",
   parameter    implement_in_les = "OFF",
   parameter    inclock_boost = 0,
   parameter    inclock_data_alignment = "EDGE_ALIGNED",
   parameter    inclock_period = 0,
   parameter    inclock_phase_shift = 0,
   parameter    input_data_rate = 0,
   parameter    lose_lock_on_one_change = "OFF",
   parameter    number_of_channels = 1,
   parameter    outclock_resource = "AUTO",
   parameter    pll_operation_mode = "NORMAL",
   parameter    pll_self_reset_on_loss_lock = "OFF",
   parameter    port_rx_channel_data_align = "PORT_CONNECTIVITY",
   parameter    port_rx_data_align = "PORT_CONNECTIVITY",
   parameter    registered_data_align_input = "ON",
   parameter    registered_output = "ON",
   parameter    reset_fifo_at_first_lock = "ON",
   parameter    rx_align_data_reg = "RISING_EDGE",
```

```
    parameter    sim_dpa_is_negative_ppm_drift = "OFF",
    parameter    sim_dpa_net_ppm_variation = 0,
    parameter    sim_dpa_output_clock_phase_shift = 0,
    parameter    use_coreclock_input = "OFF",
    parameter    use_dpll_rawperror = "OFF",
    parameter    use_external_pll = "OFF",
    parameter    use_no_phase_shift = "ON",
    parameter    x_on_bitslip = "ON",
    parameter    lpm_type = "altlvds_rx",
    parameter    lpm_hint = "unused")
(   outputwire   dpa_pll_cal_busy,
    inputwire    dpa_pll_recal,
    inputwire    pll_areset,
    outputwire   [3:0]pll_phasecounterselect,
    inputwire    pll_phasedone,
    outputwire   pll_phasestep,
    outputwire   pll_phaseupdown,
    outputwire   pll_scanclk,
    outputwire   [number_of_channels-1:0]rx_cda_max,
    inputwire    [number_of_channels-1:0]rx_cda_reset,
    inputwire    [number_of_channels-1:0]rx_channel_data_align,
    inputwire    [number_of_channels-1:0]rx_coreclk,
    inputwire    rx_data_align,
    inputwire    rx_data_align_reset,
    inputwire    rx_data_reset,
    inputwire    rx_deskew,
    outputwire   [number_of_channels-1:0]rx_divfwdclk,
    inputwire    [number_of_channels-1:0]_reset,
    outputwire   [number_of_channels-1:0]rx_dpa_locked,
    inputwire    [number_of_channels-1:0]rx_dpll_enable,
    inputwire    [number_of_channels-1:0]rx_dpll_hold,
    inputwire    [number_of_channels-1:0]rx_dpll_reset,
    inputwire    rx_enable,
    inputwire    [number_of_channels-1:0]rx_fifo_reset,
    inputwire    [number_of_channels-1:0]rx_in,
    inputwire    rx_inclock,
    outputwire   rx_locked,
    outputwire   [deserialization_factor*number_of_channels-1:0]
rx_out,
    outputwire   rx_outclock,
    inputwire    rx_pll_enable,
```

```
    inputwire     rx_readclock,
    inputwire     [number_of_channels-1:0]rx_reset,
    inputwire     rx_syncclock);

endmodule
```

Verilog HDL Prototype for ALTLVDS_TX Megafunction

The following Verilog HDL prototype is located in the Verilog Design File (.**v**) **altera_mf.v** in the <*Quartus II installation directory*>**\eda\synthesis** directory.

```
module altlvds_tx
#(  parameter center_align_msb = "UNUSED",
    parameter common_rx_tx_pll = "ON",
    parameter coreclock_divide_by = 2,
    parameter deserialization_factor = 4,
    parameter intended_device_family = "unused",
    parameter differential_drive = 0,
    parameter implement_in_les = "OFF",
    parameter inclock_boost = 0,
    parameter inclock_data_alignment = "EDGE_ALIGNED",
    parameter inclock_period = 0,
    parameter inclock_phase_shift = 0,
    parameter multi_clock = "OFF",
    parameter number_of_channels = 1,
    parameter outclock_alignment = "EDGE_ALIGNED",
    parameter outclock_divide_by = 1,
    parameter outclock_duty_cycle = 50,
    parameter outclock_multiply_by = 1,
    parameter outclock_phase_shift = 0,
    parameter outclock_resource = "AUTO",
    parameter output_data_rate = 0,
    parameter pll_self_reset_on_loss_lock = "OFF",
    parameter preemphasis_setting = 0,
    parameter registered_input = "ON",
    parameter use_external_pll = "OFF",
    parameter use_no_phase_shift = "ON",
    parameter vod_setting = 0,
    parameter lpm_type = "altlvds_tx",
    parameter lpm_hint = "unused")
(   inputwire pll_areset,
    inputwire sync_inclock,
    outputwire    tx_coreclock,
```

```
    inputwire    tx_data_reset,
    inputwire    tx_enable,
    inputwire    [deserialization_factor*number_of_channels-1:0]tx_in,
    inputwire    tx_inclock,
    outputwire   tx_locked,
    outputwire   [number_of_channels-1:0]tx_out,
    outputwire   tx_outclock,
    inputwire    tx_pll_enable,
    inputwire    tx_syncclock);

endmodule
```

VHDL Component Declaration for ALTLVDS_RX Megafunction

The following VHDL component declaration is located in the VHDL Design File (**.vhd**) **altera_mf_components.vhd** in the <*Quartus II installation directory*>**\libraries\vhdl\altera_mf** directory.

```
component altlvds_rx
    generic (
        buffer_implementation:string := "RAM";
        cds_mode:string := "UNUSED";
        common_rx_tx_pll:string := "ON";
        data_align_rollover:natural := 4;
        deserialization_factor:natural := 4;
        intended_device_family:string := "unused";
        dpa_initial_phase_value:natural := 0;
        dpll_lock_count:natural := 0;
        dpll_lock_window:natural := 0;
        enable_dpa_align_to_rising_edge_only:string := "OFF";
        enable_dpa_calibration:string := "ON";
        enable_dpa_fifo:string := "OFF";
        enable_dpa_initial_phase_selection:string := "OFF";
        enable_dpa_mode:string := "OFF";
        enable_dpa_pll_calibration:string := "OFF";
        enable_soft_cdr_mode:string := "OFF";
        implement_in_les:string := "OFF";
        inclock_boost:natural := 0;
        inclock_data_alignment:string := "EDGE_ALIGNED";
        inclock_period:natural := 0;
        inclock_phase_shift:natural := 0;
        input_data_rate:natural := 0;
        lose_lock_on_one_change:string := "OFF";
```

```
        number_of_channels:natural;
        outclock_resource:string := "AUTO";
        pll_operation_mode:string := "NORMAL";
        pll_self_reset_on_loss_lock:string := "OFF";
        port_rx_channel_data_align:string := "PORT_CONNECTIVITY";
        port_rx_data_align:string := "PORT_CONNECTIVITY";
        registered_data_align_input:string := "ON";
        registered_output:string := "ON";
        reset_fifo_at_first_lock:string := "ON";
        rx_align_data_reg:string := "RISING_EDGE";
        sim_dpa_is_negative_ppm_drift:string := "OFF";
        sim_dpa_net_ppm_variation:natural := 0;
        sim_dpa_output_clock_phase_shift:natural := 0;
        use_coreclock_input:string := "OFF";
        use_dpll_rawperror:string := "OFF";
        use_external_pll:string := "OFF";
        use_no_phase_shift:string := "ON";
        x_on_bitslip:string := "ON";
        lpm_hint:string := "UNUSED";
        lpm_type:string := "altlvds_rx"
    );
    port(
        dpa_pll_cal_busy:out std_logic;
        dpa_pll_recal:in std_logic := '0';
        pll_areset:in std_logic := '0';
        pll_phasecounterselect:out std_logic_vector(3 downto 0);
        pll_phasedone:in std_logic := '1';
        pll_phasestep:out std_logic;
        pll_phaseupdown:out std_logic;
        pll_scanclk:out std_logic;
        rx_cda_max:out std_logic_vector(number_of_channels-1 downto 0);
        rx_cda_reset:in std_logic_vector(number_of_channels-1 downto 0)
:= (others => '0');
        rx_channel_data_align:in std_logic_vector(number_of_channels-1
downto 0) := (others => '0');
        rx_coreclk:in std_logic_vector(number_of_channels-1 downto 0) :=
(others => '1');
        rx_data_align:in std_logic := '0';
        rx_data_align_reset:in std_logic := '0';
        rx_data_reset:in std_logic := '0';
        rx_deskew:in std_logic := '0';
```

```
        rx_divfwdclk:out std_logic_vector(number_of_channels-1 downto
0);

        _reset:in std_logic_vector(number_of_channels-1 downto 0) :=
(others => '0');

        rx_dpa_locked:out std_logic_vector(number_of_channels-1 downto
0);

        rx_dpll_enable:in std_logic_vector(number_of_channels-1 downto
0) := (others => '1');

        rx_dpll_hold:in std_logic_vector(number_of_channels-1 downto 0)
:= (others => '0');

        rx_dpll_reset:in std_logic_vector(number_of_channels-1 downto 0)
:= (others => '0');

        rx_enable:in std_logic := '1';

        rx_fifo_reset:in std_logic_vector(number_of_channels-1 downto 0)
:= (others => '0');

        rx_in: in std_logic_vector(number_of_channels-1 downto 0);

        rx_inclock:in std_logic;

        rx_locked:out std_logic;

        rx_out:out
std_logic_vector(deserialization_factor*number_of_channels-1 downto
0);

        rx_outclock:out std_logic;

        rx_pll_enable:in std_logic := '1';

        rx_readclock:in std_logic := '0';

        rx_reset:in std_logic_vector(number_of_channels-1 downto 0) :=
(others => '0');

        rx_syncclock:in std_logic := '0'

    );

end component;
```

VHDL Component Declaration for ALTLVDS_TX Megafunction

The following VHDL component declaration is located in the VHDL Design File (**.vhd**) **altera_mf_components.vhd** in the <*Quartus II installation directory*>**\libraries\vhdl\altera_mf** directory.

```
component altlvds_tx
    generic (
        center_align_msb:string := "UNUSED";
        common_rx_tx_pll:string := "ON";
        coreclock_divide_by:natural := 2;
        deserialization_factor:natural := 4;
        intended_device_family:string := "unused";
        differential_drive:natural := 0;
        implement_in_les:string := "OFF";
        inclock_boost:natural := 0;
```

```
        inclock_data_alignment:string := "EDGE_ALIGNED";
        inclock_period:natural := 0;
        inclock_phase_shift:natural := 0;
        multi_clock:string := "OFF";
        number_of_channels:natural;
        outclock_alignment:string := "EDGE_ALIGNED";
        outclock_divide_by:natural := 1;
        outclock_duty_cycle:natural := 50;
        outclock_multiply_by:natural := 1;
        outclock_phase_shift:natural := 0;
        outclock_resource:string := "AUTO";
        output_data_rate:natural := 0;
        pll_self_reset_on_loss_lock:string := "OFF";
        preemphasis_setting:natural := 0;
        registered_input:string := "ON";
        use_external_pll:string := "OFF";
        use_no_phase_shift:string := "ON";
        vod_setting:natural := 0;
        lpm_hint:string := "UNUSED";
        lpm_type:string := "altlvds_tx"
    );
    port(
        pll_areset:in std_logic := '0';
        sync_inclock:in std_logic := '0';
        tx_coreclock:out std_logic;
        tx_data_reset:in std_logic := '0';
        tx_enable:in std_logic := '1';
        tx_in: in
std_logic_vector(deserialization_factor*number_of_channels-1 downto
0);
        tx_inclock:in std_logic;
        tx_locked:out std_logic;
        tx_out:out std_logic_vector(number_of_channels-1 downto 0);
        tx_outclock:out std_logic;
        tx_pll_enable:in std_logic := '1';
        tx_syncclock:in std_logic := '0'
    );
end component;
```

VHDL LIBRARY-USE Declaration

The VHDL LIBRARY-USE declaration is not required if you use the VHDL Component Declaration.

```
LIBRARY altera_mf;
USE altera_mf.altera_mf_components.all;
```

Using Clear Box Generator

You can use clear box generator, a command-line executable, to configure parameters that are in the ALTLVDS_TX and ALTLVDS_RX parameter editors. The clear box generator creates or modifies custom megafunction variations, which you can instantiate in a design file. The clear box generator generates megafunction variation file in Verilog HDL or VHDL format.

To generate the ALTLVDS_TX and ALTLVDS_RX megafunctions using the clear box generator, perform the following steps:

1. Create a text file (**.txt**) that contains your clear box ports and parameter settings in your working directory.
2. Access the command prompt of your operating system, and change the current directory to your working directory by typing the following command:

   ```
   cd c:\altera\11.0\quartus\work\
   ```

   The clear box executable file name is **clearbox.exe**.

   When you install the Quartus II software, the %QUARTUS_ROOTDIR%\bin is added into your system's environment variables. Therefore, you can run the clear box command from any directory.

3. To view the available ports and parameters for this megafunction, type the following command at the command prompt of your operating system:

   ```
   clearbox altlvds_tx -h
   ```

   or

   ```
   clearbox altlvds_rx -h
   ```

4. To generate the ALTLVDS_TX and ALTLVDS_RX megafunctions variation file based on the ports and parameter settings in the text file, type the following command:

   `clearbox altlvds_tx -f *.txt` (for ALTLVDS_TX)

   or

   `clearbox altlvds_rx -f *.txt` (for ALTLVDS_RX)

   For example, `clearbox altlvds_tx -f sample_param_test.txt`

5. After the clear box generator generates the megafunction variation files, you can instantiate the megafunction module in a HDL file or a block diagram file in the Quartus II software.

6. To view the estimated hardware resources that the ALTLVDS_TX and ALTLVDS_RX megafunctions use, type the following command:

```
clearbox altlvds_tx -f sample_param_test.txt -resc_count
```

or

```
clearbox altlvds_rx -f sample_param_test.txt -resc_count
```

This command does not generate a HDL file.

For more information about the clear box parameters, refer to "Parameter Settings" on page 2–1.

For more information about the ports, refer to "ALTLVDS_TX and ALTLVDS_RX Ports" on page 3–23.

Document Revision History

Table A–1 displays the revision history for this user guide.

Table A–1. Document Revision History

Date	Document Version	Changes Made
February 2012	v9.0	■ Updated "Source-Synchronous Timing Analysis and Timing Constraints" section. ■ Added design examples. ■ Updated "Parameter Settings" chapter to include "Use Clock Pin" parameter.
June 2011	v.8.0	Updated the following items for the Quartus II software 11.0: ■ Reorganized the document format. ■ Added "Source-Synchronous Timing Analysis and Timing Constraints" section. ■ Added "Generating Clock Signals for LVDS Interface" section. ■ Updated the timing diagram in the "Receiver Skew Margin and Transmitter Channel-to-Channel Skew" section. ■ Updated "Parameter Settings" chapter. ■ Added "Using Clear Box Generator" section.
August 2010	v.7.0	Updated for the Quartus II software 10.0 and Cyclone IV and Stratix V devices: ■ Updated "DPA PLL Calibration in Stratix III and Stratix IV E Devices" section. ■ Added Verilog HDL prototypes. ■ Added VHDL LIBRARY-USE declaration. ■ Added VHDL Component Declarations. ■ Added new ports and parameters. ■ Added new parameter settings. ■ Removed Design Examples for this release.
November 2009	v6.1	Added "Arria II GX and Stratix V LVDS Package Skew Compensation Report Panel".

Table A–1. Document Revision History

Date	Document Version	Changes Made
September 2009	v6.0	Updated for the Quartus II software 9.1: ■ Added "Device Support". ■ Updated "Specifications" section to include "Ports and Parameters in ALTLVDS_RX Megafunction" and "Ports and Parameters in ALTLVDS_TX Megafunction". ■ Added "Specifications".
March 2009	v5.0	Updated for the Quartus II software 9.0: ■ Updated Table 4, and Table 12. ■ Added DPA Misalignment Issue, Figure 3, and "DPA PLL Calibration", Figure 20 and Figure 21. ■ Added Table 11 ALTLVDS Receiver DPA settings 3 option (page 7) and Table 19 Configuration Settings for Design Example 4 (LVDS Receiver). ■ Added description about "Design Example 4: Stratix III ALTLVDS Receiver with DPA PLL Calibration.
December 2008	v4.0	Updated for the Quartus II software 8.1: ■ Removed figures. ■ Added Stratix IV to Device Family Support. ■ Updated Table 3, Table 4, Table 5, Table 6, Table 7, Table 8, Table 12, Table 13, Table 15,Table 3-1, Table 3-2, Table 3-3, Table 3-4, and Table 3-6. ■ Added Enable bitslip control, Enable independent bitslip controls for each channel, and Register the bitslip control input using 'rx_outclock' parameters and descriptions Table 11. ■ Updated steps in Functional Results—Simulate the ALTLVDS Receiver/Transmitter Design in the ModelSim-Altera Software, Functional Results—Simulate the ALTLVDS Receiver/Transmitter Design in the Quartus II Software, "Functional Results—Simulatethe ALTLVDS Receiver/Transmitter Design in the ModelSim-Altera Software". ■ Added `tx_syncclock` and descriptions in Table 3–1. ■ Added `rx_data_align` and `rx_syncclock` in Table 3–4. ■ Updated descriptions in Table 3–6.
May 2008	v3.4	Small changes to Table 2–7 on page 2–27 and Table 2–9 on page 2–32.
November 2007	v3.3	Updated for the Quartus® II software v7.2, including: ■ Added soft-CDR mode. ■ Added description of new receiver output port `rx_divfwdclk[]`. ■ Added description of new receiver parameters `enable_soft_cdr`, `is_negative_ppm_drift`,`net_ppm_variation`, `enable_dpa_align_to_ rising_edge_only`, ■ `dpa_initial_phase_ value`, and `enable_dpa_initial_phase _selection`. ■ Updated two design examples. ■ Added third design example using soft-CDR mode.
March 2007	v3.2	Updated for Quartus II software 7.0, including Cyclone® III information.
December 2006	v3.1	Updated Table 1-1 to include Stratix® III information.

Table A–1. Document Revision History

Date	Document Version	Changes Made
November 2006	v3.0	Updated for the Quartus II software 6.1.
June 2006	v2.0	Updated for the Quartus II software 6.0.
August 2005	v1.1	Minor content changes.
December 2004	v1.0	Initial release.