Section I. Arria GX Transceiver User Guide

This section provides information on the configuration modes for Arria™ GX devices. It also includes information on testing, Arria GX port and parameter information, and pin constraint information.

This section includes the following chapters:

- Chapter 1, Arria GX Transceiver Architecture
- Chapter 2, Arria GX Transceiver Protocol Support and Additional Features
- Chapter 3, Arria GX ALT2GXB Megafunction User Guide
- Chapter 4, Specifications and Additional Information

Revision History

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

1. Arria GX Transceiver Architecture

AGX52001-1.2

Introduction

The Arria™ GX is a protocol-optimized FPGA family that leverages Altera®'s advanced multi-gigabit transceivers. The Arria GX transceiver blocks build on the success of the Stratix® II GX family and are optimally designed to support the following serial connectivity protocols (functional modes):

- PCI Express (PIPE)
- Gigabit Ethernet (GIGE)
- Serial RapidIO

Building Blocks

Arria GX transceivers are structured into full duplex (transmitter and receiver) four-channel groups called transceiver blocks. The Arria GX device family offers up to 12 transceiver channels (three transceiver blocks) per device. You can configure each transceiver block to one of the three supported functional modes; for example, four GIGE ports or one four-lane (×4) PCI Express (PIPE) port. In Arria GX devices that offer more than one transceiver block, you can configure each transceiver block to a different functional mode; for example, one transceiver block configured as a four-lane (×4) PCI Express (PIPE) port and the other transceiver block can be configured as four GIGE ports.

In addition to providing the physical coding sublayer (PCS) and physical media attachment (PMA) circuitry for each functional mode, Arria GX transceivers support the following protocol specific features:

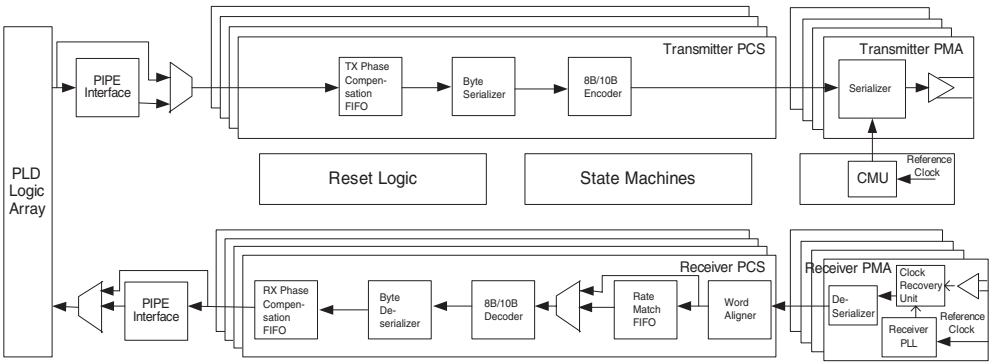
- PCI Express (PIPE)
  - 2.5 Gbps (PCI Express Generation 1) line rate
  - ×1 (Individual single-lane) and ×4 (Bonded four-lane) link widths
  - Synchronous PCI Express (rate matching FIFO bypass capability)
  - PCI Express synchronization state machine
  - Receiver detection
  - Electrical Idle generation/detection
  - Polarity inversion
  - Power state management
- GIGE
  - 1.25 Gbps line rate
  - Synchronization state machine
  - Idle sequence (/I1/, /I2/) generation

- Serial RapidIO
  - 1.25 Gbps and 2.5 Gbps line rate
  - Synchronization state machine

For a complete set of features supported in each protocol, refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*.

Figure 1–1 shows the Arria GX transceiver block diagram divided into Transmitter and Receiver Digital Circuits (PCS) and Analog Circuits (PMA).

Figure 1–1. Arria GX Gigabit Transceiver Block Diagram



Port List

You instantiate the Arria GX transceivers using the ALT2GXB megacore instance provided in the Quartus® II MegaWizard® Plug-In Manager. The ALT2GXB instance allows you to configure the transceivers for your intended protocol and select optional control and status ports to and from the instantiated transceiver channels.

Figure 1–2 shows the port diagram of all available ports in an ALT2GXB transceiver instance.

Figure 1–2. Arria GX Transceiver Port Diagram

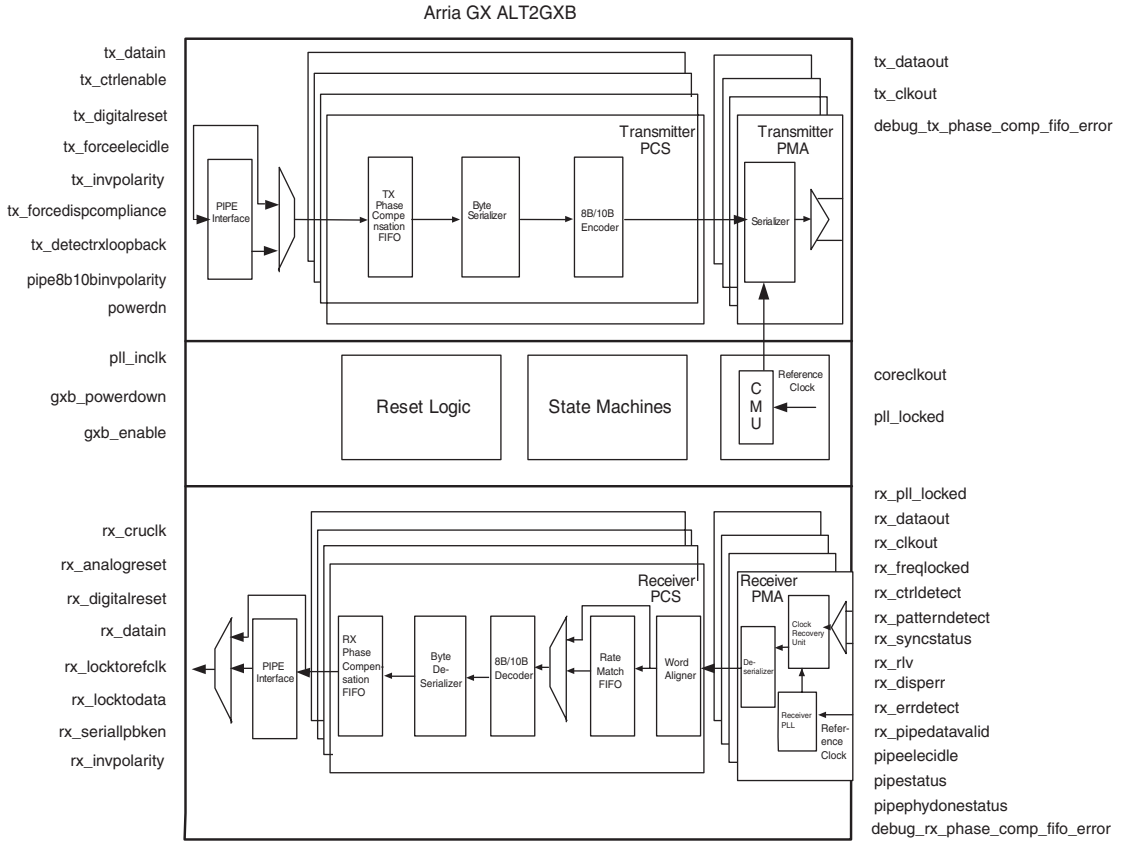


Table 1–1 provides a brief description for all available ALT2GXB ports shown in Figure 1-2.

Table 1–1. Arria GX ALT2GXB Ports (Part 1 of 5)

Port Name	Input/Output	Description	Scope
Receiver Physical Coding Sublayer (PCS) Ports			
rx_dataout	Output	Receiver parallel data output. The bus width depends on the channel width for the selected functional mode multiplied by the number of channels per instance.	
rx_clkout	Output	Recovered clock from the receiver channel.	Channel
rx_coreclk	Input	Optional read clock port for the receiver phase compensation FIFO. If not selected, Quartus II automatically selects rx_clkout/tx_clkout/coreclkout as the read clock for receiver phase compensation FIFO. If selected, you must drive this port with a clock that is frequency locked to rx_clkout/tx_clkout/coreclkout	Channel
rx_rlv	Output	Run-length violation indicator. A high signal is driven when the run length (consecutive '1's or '0's) of the received data exceeds the configured limit.	Channel
pipe8b10binvpolarity	Input	PCI Express (PIPE) polarity inversion at the 8B/10B decoder input. This port inverts the data at the input of the 8B/10B decoder.	Channel
pipestatus	Output	PCI Express (PIPE) receiver status port. In the case of multiple status signals, the lower number signal takes precedence. 000 – Received data OK 001 – 1 skip added (not supported) 010 – 1 skip removed (not supported) 011 – Receiver detected 100 – 8B/10B decoder error 101 – Elastic buffer overflow 110 – Elastic buffer underflow 111 – Received disparity error	Channel
pipephydonestatus	Output	Indicates a mode transition completion in PCI Express (PIPE) mode. On completion of a power state transition or receiver detection, a pulse is driven on this port.	Channel
rx_pipedatavalid	Output	PCI Express (PIPE) valid data indicator on the rx_dataout port.	Channel
pipeelecidle	Output	PCI Express (PIPE) signal detect.	Channel

Table 1–1. Arria GX ALT2GXB Ports (Part 2 of 5)

Port Name	Input/Output	Description	Scope
rx_digitalreset	Input	Reset port for the receiver PCS block. This port resets all the digital logic in the receiver channel. The minimum pulse width is two parallel clock cycles.	Channel
rx_ctrldetect	Output	Receiver control code group indicator port. Indicates whether the data at the output of rx_dataout is a control or data word.	Channel
rx_errdetect	Output	8B/10B code group violation signal. Indicates that the data at the output of rx_dataout has a code group violation or a disparity error. Used with disparity error signal to differentiate between a code group error and/or a disparity error.	Channel
rx_syncstatus	Output	Indicates the lane synchronization status in all functional modes. A HIGH signal is driven continuously when the synchronization state machine for the selected functional mode detects that all lane synchronization conditions are satisfied.	Channel
rx_disperr	Output	8B/10B disparity error indicator port. Indicates that the data at the output of rx_dataout has a disparity error.	Channel
rx_patterndetect	Output	Indicates when the word aligner detects the alignment pattern in the current word boundary.	Channel
rx_invpolarity	Input	Inverts the polarity of the received data at the input of the word aligner	Channel
debug_rx_phase_comp_fifo_error	Output	Indicates receiver phase compensation FIFO overrun or underrun situation	Channel
Receiver Physical Media Attachment (PMA)			
rx_pll_locked	Output	Receiver PLL locked signal. Indicates if the receiver PLL is phase locked to the CRU reference clock.	Channel
rx_analogreset	Input	Receiver analog reset. Resets all analog circuits in the receiver PMA.	Channel
rx_freqlocked	Output	CRU mode indicator port. Indicates if the CRU is locked to data mode or locked to the reference clock mode. 0 – Receiver CRU is in lock-to-reference clock mode 1 – Receiver CRU is in lock-to-data mode	Channel

Table 1–1. Arria GX ALT2GXB Ports (Part 3 of 5)

Port Name	Input/Output	Description	Scope
rx_signaldetect	Output	Signal detect port. In PCI Express (PIPE) mode, indicates if a signal that meets the specified range is present at the input of the receiver buffer. In all other modes, rx_signaldetect is forced high and must not be used as an indication of a valid signal at receiver input.	Channel
rx_seriallpbken	Input	Serial loopback control port. 0 – normal data path, no serial loopback 1 – serial loopback	Channel
rx_locktodata	Input	Lock-to-data control for the CRU. Use with rx_locktorefclk.	Channel
rx_locktorefclk	Input	Lock-to-reference lock mode for the CRU. Use with rx_locktodata. rx_locktodata/rx_locktorefclk 0/0 – CRU is in automatic mode 0/1 – CRU is in lock-to-reference clock 1/0 – CRU is in lock-to-data mode 1/1 – CRU is in lock-to-data mode	Channel
rx_cruclk	Input	Receiver PLL/CRU reference clock.	Channel
Transmitter PCS			
tx_datain	Input	Transmitter parallel data input. The bus width depends on the channel width for the selected functional mode multiplied by the number of channels per instance.	Channel
tx_clkout	Output	PLD logic array clock from the transceiver to the PLD. In an individual-channel mode (for example, GIGE, PCI Express [PIPE] ×1), there is one tx_clkout per channel.	Channel
coreclkout	Output	PLD logic array clock from the transceiver block to the PLD. In bonded-channel mode (×4 PCI Express [PIPE]), there is one coreclkout per transceiver block.	Transceiver block
tx_coreclk	Input	Optional write clock port for the transmitter phase compensation FIFO. If not selected, Quartus II automatically selects tx_clkout/coreclkout as the write clock for transmitter phase compensation FIFO. If selected, you must drive this port with a clock that is frequency locked to tx_clkout/coreclkout.	Channel

Table 1–1. Arria GX ALT2GXB Ports (Part 4 of 5)

Port Name	Input/Output	Description	Scope
tx_detectrxloopback	Input	PCI Express (PIPE) receiver detect / loopback pin. Depending on the power state (P0 or P1), the signal either activates receiver detect or loopback.	Channel
tx_forceelecidle	Input	PCI Express (PIPE) Force Electrical Idle. When asserted, it tristates the transmitter buffer in PCI Express (PIPE) mode.	Channel
tx_forcedispcompliance	Input	PCI Express (PIPE) force negative disparity for compliance pattern. The compliance pattern requires beginning with a negative disparity. Assertion of this port while transmitting the first byte ensures that the first byte is encoded with a negative disparity. This port must be de-asserted after the first byte.	Channel
powerdn	Input	PCI Express (PIPE) power mode port. This port sets the power mode of the associated PCI Express channel. The power modes are as follows: 2'b00: P0 – Normal operation 2'b01: P0s – Low recover time latency, power saving state 2'b10: P1 – Longer recovery time (64us max) latency, lower power state 2'b11: P2 – Lowest power state	Channel
tx_digitalreset	Input	Reset port for the transmitter PCS block. This port resets all the digital logic in the transmit channel. The minimum pulse width is two parallel clock cycles.	Channel
tx_ctrlenable	Input	Transmitter control code group indicator port. Indicates whether the data at the tx_datain port is a control or data word. When high, the 8-bit character is encoded as data code group (Dx.y). When low, the 8-bit character is encoded as control code group (Kx.y)	Channel
tx_invpolarity	Input	Inverts the polarity of the data to be transmitted at the transmitter PCS-PMA interface (input to the serializer).	Channel
debug_tx_phase_comp_fifo_error	Output	Indicates transmitter phase compensation FIFO overrun or underrun situation.	Channel
Transmitter PMA			
fixedclk	Input	125-MHz clock for receiver detect circuitry in PCI Express (PIPE) mode.	Channel

Table 1–1. Arria GX ALT2GXB Ports (Part 5 of 5)

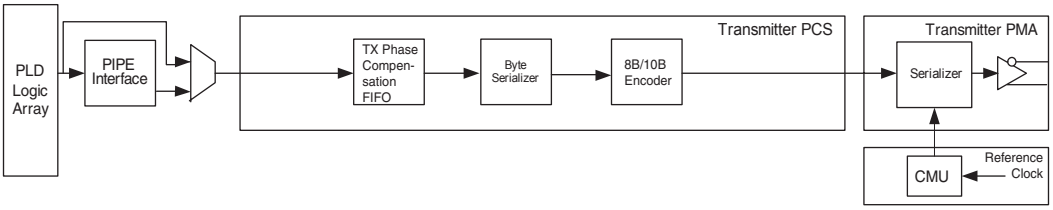
Port Name	Input/Output	Description	Scope
CMU PMA			
gxb_powerdown	Input	Transceiver block reset and power down. This resets and powers down all circuits in the transceiver block. It does not power down the REFCLK buffers and reference clock lines.	Transceiver block
gxb_enable	Input	Dedicated transceiver block enable pin. If instantiated, this port must be tied to the dedicated gigabit transceiver block enable input pin.	Transceiver block
pll_locked	Output	PLL locked indicator for the transmitter PLLs.	Transceiver block
pll_inclk	Input	Reference clocks for the transmitter PLLs.	Transceiver block
Calibration Block			
cal_blk_clk	Input	Calibration clock for the transceiver termination blocks. This clock supports frequencies from 10 MHz to 125 MHz.	Device
cal_blk_powerdown (active low)	Input	Power-down signal for the calibration block. Assertion of this signal may interrupt data transmission and reception. Use this signal to re-calibrate the termination resistors if temperature and/or voltage changes warrant it.	Device
External Signals			
tx_dataout	Output	Transmitter serial output port.	Channel
rx_datain	Input	Receiver serial input port.	Channel
rrefb *(1)*	Output	Reference resistor port. This port is always used and must be tied to a 2K-Ω resistor to ground. This port is highly sensitive to noise. There must be no noise coupled to this port.	Device
refclk *(1)*	Input	Dedicated reference clock inputs (two per transceiver block) for the transceiver. The buffer circuitry is similar to the receiver buffer, but the termination is not calibrated.	Transceiver block

Note to Table 1–1:
(1) These are dedicated pins for the transceiver and do not appear in the MegaWizard Plug-In Manager.

Transmitter Channel Architecture

This section provides a brief description about sub-blocks within the transmitter channel (Figure 1–3). The sub-blocks are described in order from the PLD-transmitter parallel interface to the serial transmitter buffer.

Figure 1–3. Arria GX Transmitter Channel Block Diagram



Clock Multiplier Unit

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

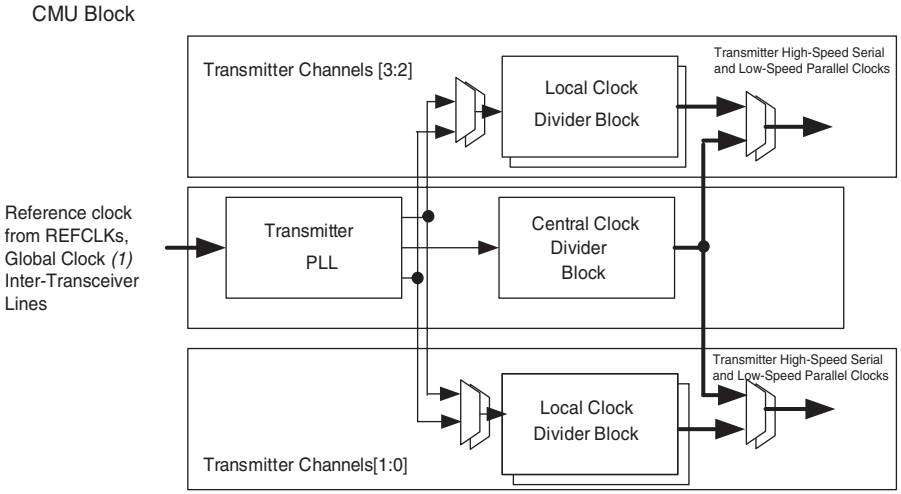
The CMU is further divided into three sub-blocks

- Transmitter PLL
- Central clock divider block
- Local clock divider block

Each transceiver block has one transmitter PLL, one central clock divider and four local clock dividers. One local clock divider is located in each transmitter channel of the transceiver block.

Figure 1–4 shows a block diagram of the CMU block within each transceiver block.

Figure 1–4. Clock Multiplier Unit Block Diagram



Note to Figure 1–4:
(1) The global clock line must be driven from an input pin only.

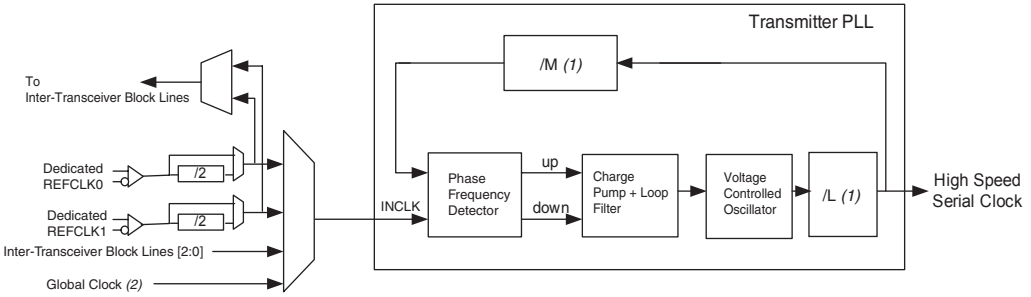
Transmitter PLL

The transmitter PLL multiplies the input reference clock to generate the high-speed serial clock required to support the intended protocol. It synthesizes a half-rate high-speed serial clock that runs at half the frequency of the serial data rate for which it is configured; for example, the transmitter PLL runs at 625 MHz when configured in 1.25 Gbps GIGE functional mode.

The transmitter PLL output feeds the central clock divider block and the local clock divider blocks. These clock divider blocks divide the high-speed serial clock to generate the low-speed parallel clock for the transceiver PCS logic and the PLD-transceiver interface clock. Depending on the functional mode for which the transceiver block is configured, either the central clock divider block or the local clock divider block is used to generate the low-speed parallel clock.

Figure 1–5 shows a block diagram of the transmitter PLL.

Figure 1–5. Transmitter PLL



Notes to Figure 1–5:

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

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

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

- One of the two available dedicated reference clock input pins (REFCLK0 or REFCLK1) of the associated transceiver block
- PLD global clock network (must be driven directly from an input clock pin and cannot be driven by user logic or enhanced PLL)
- Inter-transceiver block lines driven by reference clock input pins of other transceiver blocks

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

Dedicated Reference Clock Input Pins

Each transceiver block has two dedicated reference clock input pins (REFCLK0 and REFCLK1). The clock route from REFCLK0 and REFCLK1 pins in each transceiver block has an optional pre-divider that divides the reference clock by two before feeding it to the transmitter PLL (Figure 1–5).

The Quartus II software automatically selects the divide-by-two pre-divider for the reference clock input in PCI Express (PIPE) mode only. In GIGE and Serial Rapid IO modes, the pre-divider is not used.

Reference Clock From PLD Global Clock Network

You can drive the reference clock to the transmitter PLL from a PLD global clock network. If you choose this option, you must drive the global PLD reference clock line from a non-REFCLK FPGA input pin. You cannot use a clock generated by PLD logic or an enhanced PLL to drive the reference clock input to the transmitter PLL.

The Quartus II software requires the following setting for the non-REFCLK FPGA input pin used to drive the reference clock input:

Assignment name: *Stratix II GX/Arria GX REFCLK coupling and termination setting*

Value: *Use as regular IO*.

Inter-Transceiver Block Line Routing

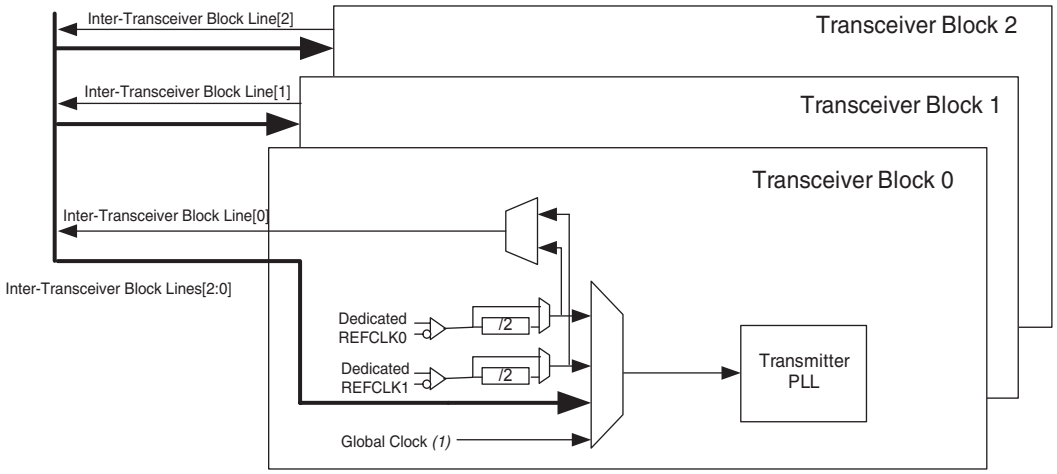
The inter-transceiver block lines allow the dedicated reference clock input pins of one transceiver block to drive the transmitter and receiver PLL of other transceiver blocks. There are a maximum of three inter-transceiver block routing lines available in the Arria GX device family. Each transceiver block can drive one inter-transceiver block line from either one of its associated reference clock pins. The inter-transceiver block lines can drive any or all of the transmitter and receiver PLLs in the device. The inter-transceiver block lines offer flexibility when multiple channels in separate transceiver blocks share a common reference clock frequency.

The inter-transceiver block lines also drive the reference clock from the REFCLK pins into the PLD fabric, which reduces the need to drive multiple clocks of the same frequency into the device. If a divide-by-two reference clock pre-divider is used, the inter-transceiver block line driven by the corresponding REFCLK pin cannot be used to clock PLD logic.

The Quartus II software automatically uses the appropriate inter-transceiver line if the transceiver block is being clocked by the dedicated reference clock (REFCLK) pin of another transceiver block.

Figure 1–6 shows the inter-transceiver block line interface to the transceivers in the gigabit transceiver blocks and to the PLD.

Figure 1–6. Inter-Transceiver Block Line Routing



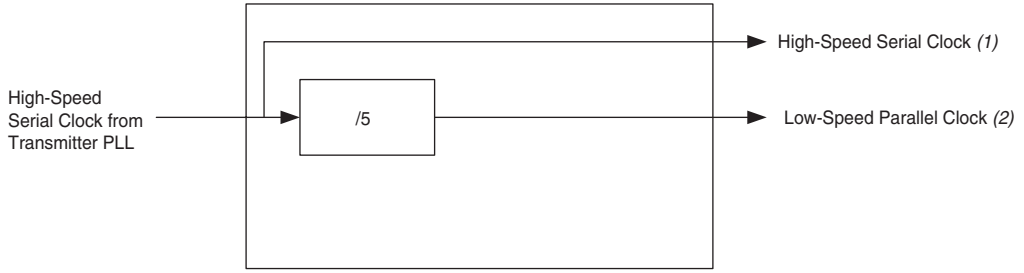
Note to Figure 1–6:
(1) The global clock line must be driven from and input pin only.

Depending on the functional mode, the Quartus II software automatically selects the appropriate transmitter PLL bandwidth.

Central Clock Divider Block

One central clock divider block is available per transceiver block. It takes in the high-speed transmitter PLL clock and divides it to generate the low-speed parallel clock for transceiver PCS logic in ×4 PCI Express (PIPE) bonded-channel mode. The Quartus II software automatically selects /5 as the division factor for all supported functional modes. The central clock divider block also forwards the high-speed serial clock from the transmitter PLL to the serializer, as shown in Figure 1–7.

Figure 1–7. Central Clock Divider Block



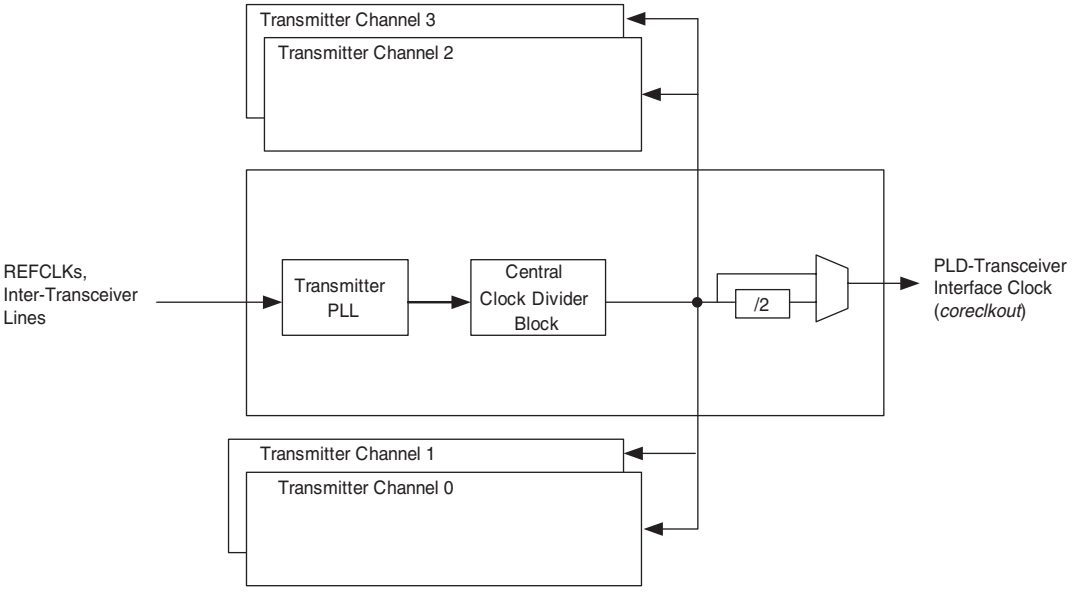
Notes to Figure 1–7:

(1) High-speed serial clock from the central clock divider block feeds the serializer in all channels within a transceiver block in bonded-channel mode (×4 PCI Express [PIPE]).

(2) Low-speed parallel clock from the central clock divider block feeds the transceiver PCS logic in all channels within a transceiver block in bonded-channel mode (×4 PCI Express [PIPE]).

The central clock divider block feeds the high-speed and low-speed clock to all the channels in the transceiver block, as shown in Figure 1–8. This ensures that the serializer in each channel outputs the same bit number at any time to minimize the channel-to-channel skew.

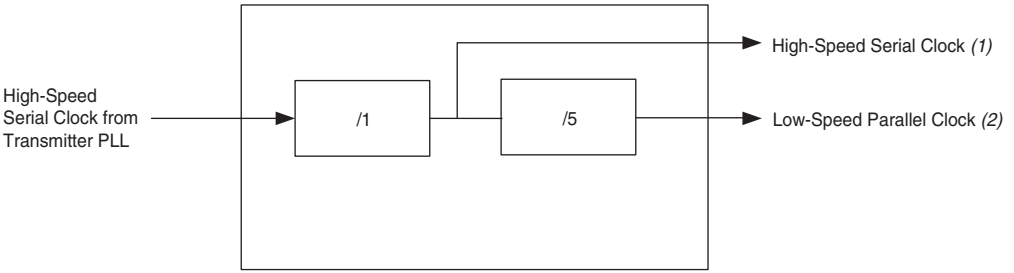
Figure 1–8. Clock Distribution in Bonded Channel Mode



Local Clock Divider Block

A local clock divider block is located in each of the four transmitter channels of a transceiver block. It takes in the high-speed transmitter PLL clock and divides it to generate the low-speed parallel clock for transceiver PCS logic in individual channel modes; for example, GIGE mode. The Quartus II software automatically selects /5 as the division factor for all supported functional modes. The local clock dividers also forward the high-speed serial clock from the transmitter PLL to the serializer as shown in Figure 1–9.

Figure 1–9. Local Clock Divider Block *Note (3)*



Notes to Figure 1–9:

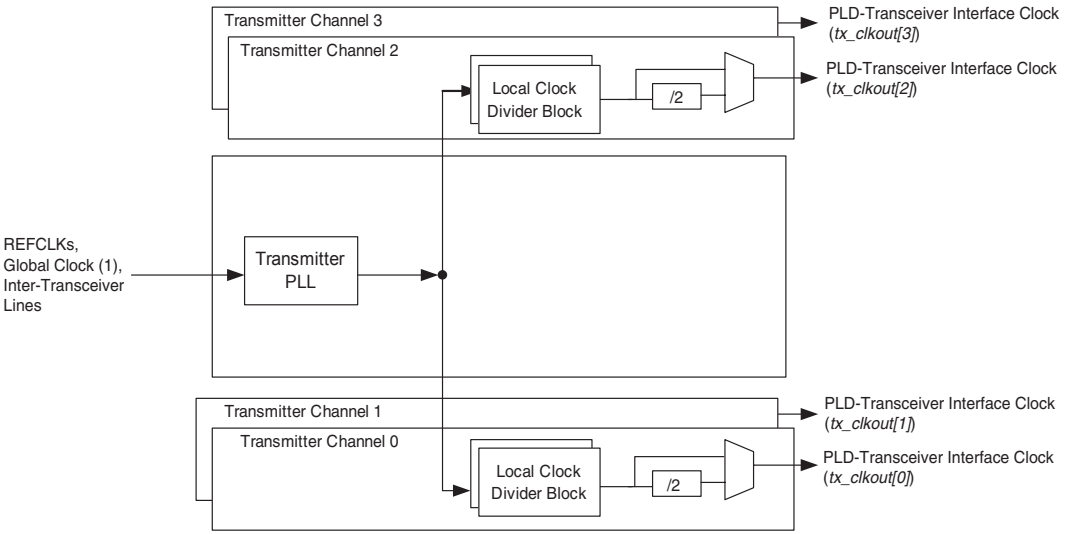
(1) High-speed serial clock from the local clock divider block feeds the serializer in its associated channel in individual-channel modes (for example, GIGE).

(2) Low-speed parallel clock from the local clock divider block feeds the transceiver PCS logic in its associated channel in individual-channel mode (for example, GIGE).

(3) Each channel within a transceiver block is fed by its own local clock divider block, as shown in Figure 1–10.

Figure 1–10 shows the clock distribution in individual channel mode.

Figure 1–10. Clock Distribution in Individual Channel Mode



Note to Figure 1–10:
(1) The global clock line must be driven from an input pin only.

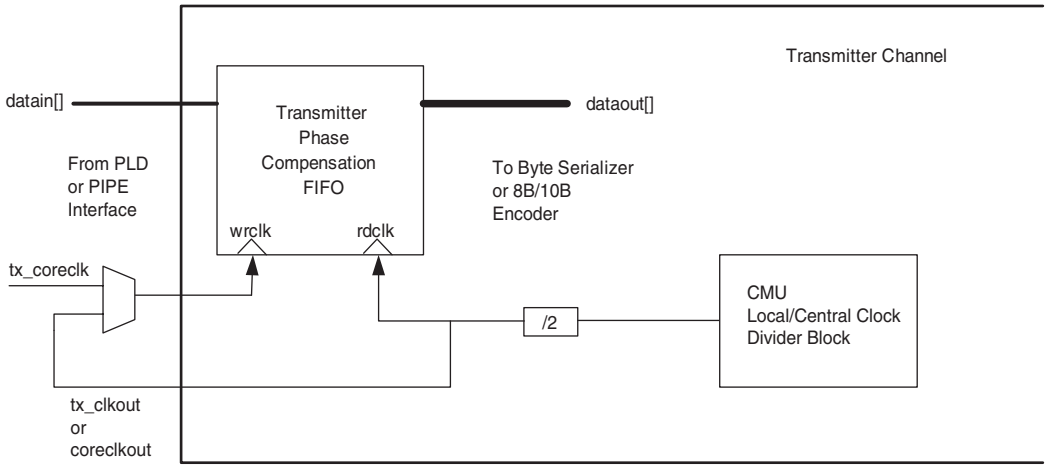
Transmitter Phase Compensation FIFO

A transmitter phase compensation FIFO (Figure 1–11) is located at each transmitter channel's logic array interface. It compensates for the phase difference between the transmitter PCS clock and the local PLD clock.

In individual channel mode (for example, GIGE and Serial RapidIO), the low-speed parallel clock (or its divide-by-two version if the byte serializer is used) from the local clock divider block of each channel clocks the read port of its transmitter phase compensation FIFO buffer. This clock is also forwarded to the logic array on `tx_clkout` port of its associated channel. If the `tx_coreclk` port is not instantiated, the clock signal on the `tx_clkout` port of Channel 0 is automatically fed back to clock the write port of the transmitter phase compensation FIFOs in all channels within the transceiver block. If the `tx_coreclk` port is instantiated, the clock signal driven on the `tx_coreclk` port clocks the write port of the transmitter phase compensation FIFO of its associated channel. You must ensure that the clock on the `tx_coreclk` port is frequency-locked to the read clock of the transmitter phase compensation FIFO. For more information about using the PLD core clock (`tx_coreclk`), refer to "PLD-Transceiver Interface Clocking" on page 1–50.

In bonded channel mode (for example, x4 PCI Express [PIPE]), the low-speed parallel clock from the central clock divider block is divided by two. This divide-by-two clock clocks the read port of the transmitter phase compensation FIFO. This clock is also forwarded to the logic array on the `coreclkout` port. If the `tx_coreclk` port is not instantiated, the clock signal on the `coreclkout` port is automatically fed back to clock the write port of transmitter phase compensation FIFO buffers in all channels within the transceiver block. If the `tx_coreclk` port is instantiated, the clock signal driven on the `tx_coreclk` port clocks the write port of the transmitter phase compensation FIFO of its associated channel. You must ensure that the clock on the `tx_coreclk` port is frequency locked to the read clock of the transmitter phase compensation FIFO. For more information about using the PLD core clock (`tx_coreclk`), refer to "PLD-Transceiver Interface Clocking" on page 1–50.

Figure 1–11. Transmitter Phase Compensation FIFO



Transmitter Phase Compensation FIFO Error Flag

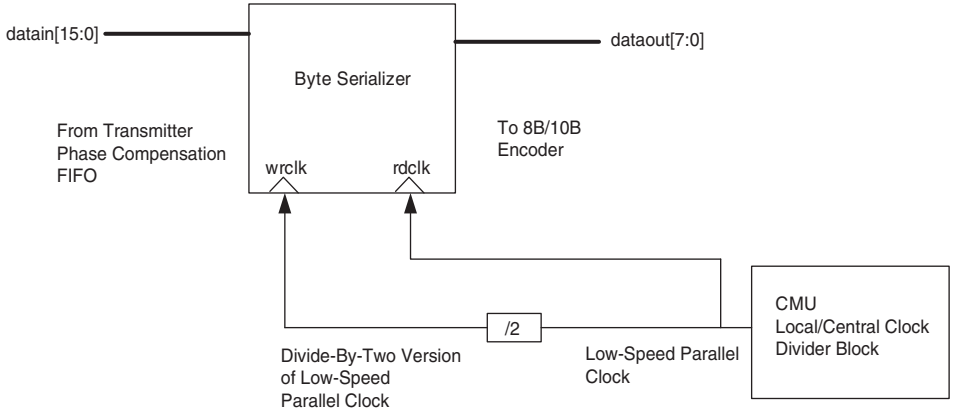
The write port of the transmitter phase compensation FIFO can be clocked by either the CMU output clock or its divide-by-two version (`tx_clkout` or `coreclkout`) or a PLD clock. The read port is always clocked by the CMU output clock or its divide-by-two version. In all configurations, the write clock and the read clock must have 0 PPM difference to avoid overrun/underflow of the phase compensation FIFO.

An optional debug_tx_phase_comp_fifo_error port is available in all modes to indicate transmitter phase compensation FIFO overrun/underflow condition. This feature should be used for debug purpose only if link errors are observed.

Byte Serializer

In PCI Express (PIPE) and Serial RapidIO functional modes, the byte serializer (Figure 1–12) takes in 16-bit wide data from the transmitter phase compensation FIFO buffer and serializes it into an 8-bit wide data at twice the speed. This allows clocking the PLD-transceiver interface at half the speed as compared to the transmitter PCS logic. The byte serializer is bypassed in GIGE mode.

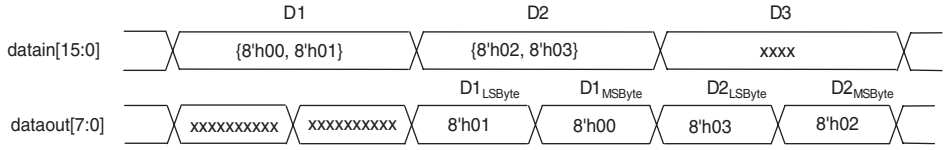
Figure 1–12. Byte Serializer



After serialization, the byte serializer transmits the least significant byte (LSByte) first and the most significant byte (MSByte) last.

Figure 1–13 shows byte serializer input and output. datain[15:0] is the input to the byte serializer from the transmitter phase compensation FIFO and dataout[7:0] is the output of the byte serializer.

Figure 1–13. Byte Serializer Operation



In Figure 1–13, the LSByte is transmitted before the MSByte from the transmitter byte serializer. For input data D1, the output data is $D1_{LSByte}$ and then $D1_{MSByte}$.

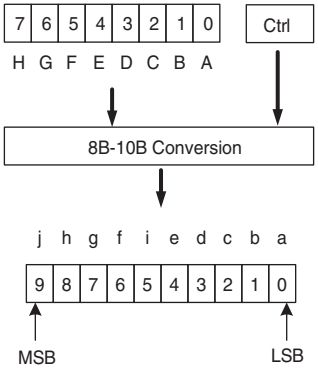
8B/10B Encoder

The 8B/10B encoder block takes in 8-bit data from the byte serializer (in PCI Express [PIPE] and Serial RapidIO modes) or transmitter phase compensation FIFO buffer (in GIGE mode). It generates a 10-bit code group with proper running disparity from the 8-bit character and a 1-bit control identifier (`tx_ctrlenable`). The 10-bit code group is fed to the serializer. The 8B/10B encoder conforms to the IEEE 802.3 1998 edition standard.

Figure 1–14 shows the 8B/10B conversion format.

For additional information about 8B/10B encoding rules, refer to the *Specifications and Additional Information* chapter in volume 2 of the *Arria GX Device Handbook*.

Figure 1–14. 8B/10B Encoder



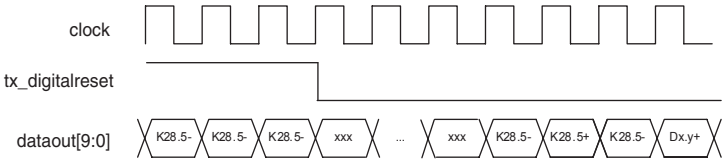
The 10-bit encoded data output from the 8B/10B encoder is fed to the serializer that transmits the data from LSB to MSB.

Reset Behavior

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

Figure 1–15 shows the 8B/10B encoder's reset behavior. When in reset (`tx_digitalreset` is high), a K28.5- (K28.5 10-bit code group from the RD- column) is sent continuously until `tx_digitalreset` is low. The transmitter channel pipelining causes some "don't cares (10'hxxx)" until the first of three K28.5 is sent. User data follows the third K28.5

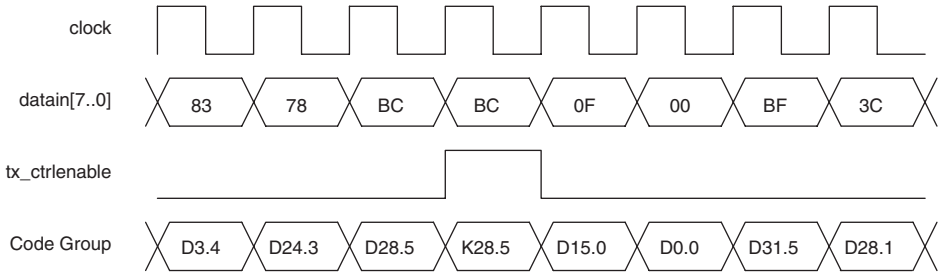
Figure 1–15. 8B/10B Encoder Output During Reset



Control Code Group Encoding

A control identifier (`tx_ctrlenable`) input signal identifies if the 8-bit input character is to be encoded as a control word (`Kx.y`) or data word (`Dx.y`). When `tx_ctrlenable` is low, the input character is encoded as data (`Dx.y`). When `tx_ctrlenable` is high, the input character is encoded as a control word (`Kx.y`). The waveform in Figure 1–16 shows that the second 0xBC character is encoded as a control word (`K28.5`). The rest of the characters are encoded as data (`Dx.y`).

Figure 1–16. Control Code Group Identification



The 8B/10B encoder does not check to see if the code group word entered is one of the 12 valid codes. If you enter an invalid control code, the resultant 10-bit code group may be encoded as an invalid code (does not map to a valid `Dx.y` or `Kx.y` code group), or unintended valid `Dx.y` code group, depending on the value entered.

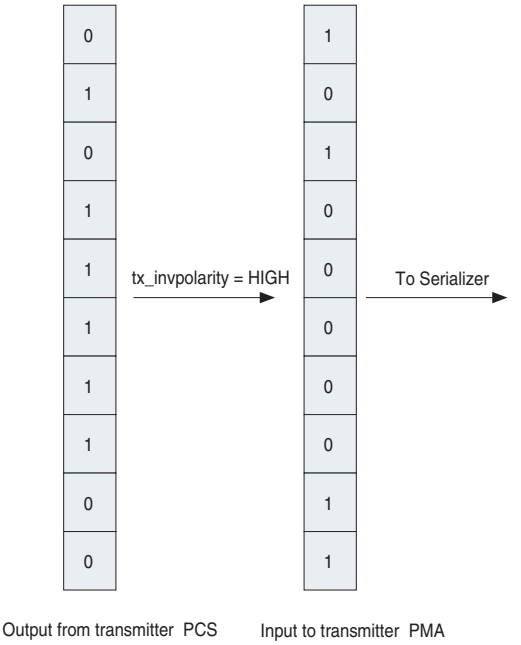
Transmitter Polarity Inversion

The positive and negative signals of a serial differential link might accidentally be swapped during board layout. Solutions such as a board re-spin or major updates to the PLD logic can prove expensive. The transmitter polarity inversion feature is provided to correct this situation.

An optional `tx_invpolarity` port is available in all modes to dynamically enable the transmitter polarity inversion feature. A high on the `tx_invpolarity` port inverts the polarity of every bit of the 10-bit input data word to the serializer in the transmitter data path. Since inverting the polarity of each bit has the same effect as swapping the positive and negative signals of the differential link, correct data is seen by the receiver. The `tx_invpolarity` is a dynamic signal and may cause initial disparity errors at the receiver of an 8B/10B encoded link. The downstream system must be able to tolerate these disparity errors.

Figure 1–17 illustrates the transmitter polarity inversion feature.

Figure 1–17. Transmitter Polarity Inversion



Serializer

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

Figure 1–18 shows the serializer block diagram.

Figure 1–18. Serializer

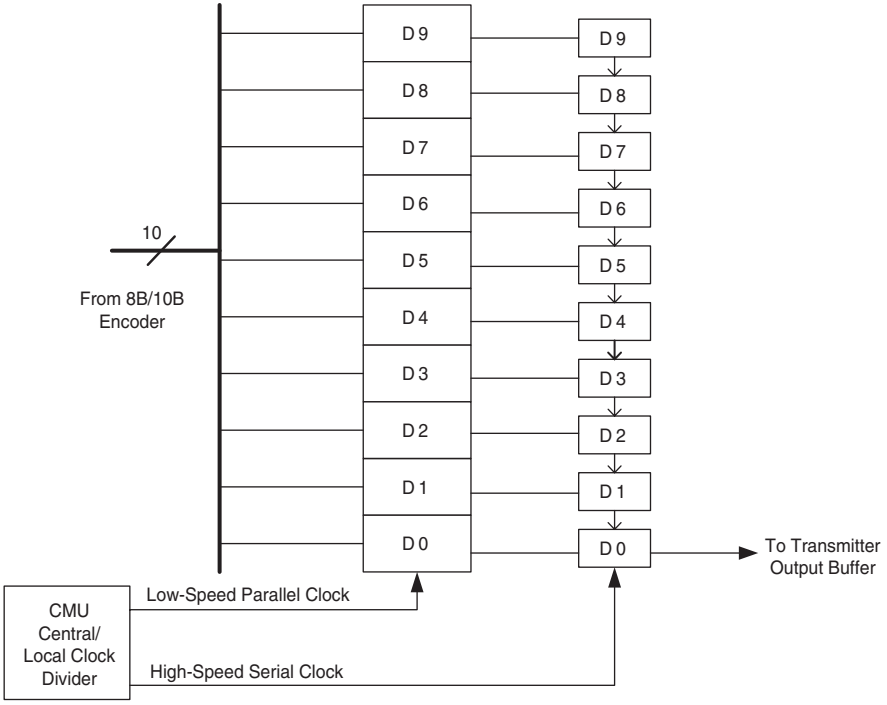
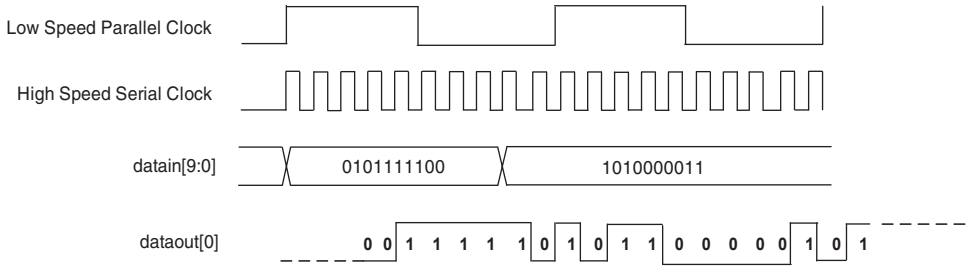


Figure 1–19 shows the serial bit order at the serializer output. In this example, 10'b17C data is serialized and transmitted from LSB to MSB.

Figure 1–19. Serializer Bit Order



Transmitter Buffer

The transmitter buffer takes in serial data from the serializer and drives it on the `tx_dataout` port of the associated transceiver channel.

Table 1–2 shows available transmitter buffer settings in each functional mode.

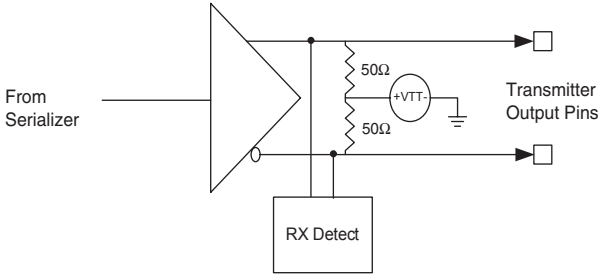
Table 1–2. Transmitter Buffer Settings

Mode	On-Chip Termination with Calibration	Supported I/O Standard	Differential Output Voltage (V_{OD})	Common Mode Voltage (V_{CM})	Pre-Emphasis
PCI Express (PIPE)	100 Ω	1.2 V-PCML	800 mV	600 mV	Enabled *(1)*
GIGE	100 Ω	1.5 V-PCML	800 mV	600 mV	Disabled
Serial RapidIO	100 Ω	1.5 V-PCML	800 mV	600 mV	Disabled

Note to Table 1–2:
(1) In PCI Express (PIPE) mode, 49% pre-emphasis is used to meet the PCI Express de-emphasis specification.

Figure 1–20 shows the transmitter buffer circuitry.

Figure 1–20. Transmitter Buffer



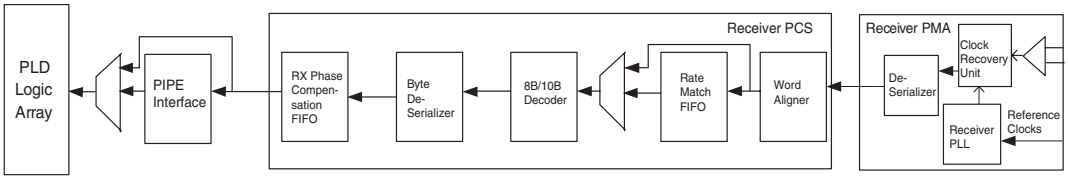
The transmitter buffer supports the Electrical Idle and Receiver Detect features in PCI Express (PIPE) mode.

Refer to the Transmitter Buffer section of the respective functional mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for more details about the supported features.

Receiver Channel Architecture

This section provides a brief description about sub-blocks within the receiver channel (Figure 1–21). The sub-blocks are described in order from the serial receiver input buffer to the receiver phase compensation FIFO buffer at the transceiver-PLD interface.

Figure 1–21. Receiver Channel Block Diagram



Receiver Buffer

The receiver buffer receives serial data from the `rx_datain` port and feeds it to the clock recovery unit (CRU).

Table 1–3 shows available receiver buffer settings in each functional mode.

Table 1–3. Receiver Buffer Settings

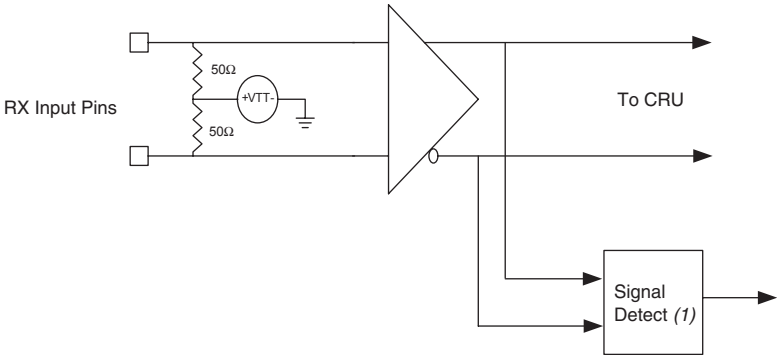
Mode	On-Chip Termination with Calibration	Supported I/O Standard	Common Mode Voltage (RX V_{CM})	Coupling
PCI Express (PIPE), GIGE, Serial RapidIO	100 Ω	1.2 V-PCML, 1.5 V-PCML, 3.3 V-PCML, Differential LVPECL, LVDS	850 mV	AC

The receiver buffer also incorporates signal threshold detection circuitry only in PCI Express (PIPE) mode.

Refer to the Receiver Buffer section of the respective functional mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for more details about the supported features.

Figure 1–22 shows the receiver buffer circuitry.

Figure 1–22. Receiver Buffer



Note to Figure 1–22:
(1) The signal detect circuitry is available only in PCI Express (PIPE) mode.

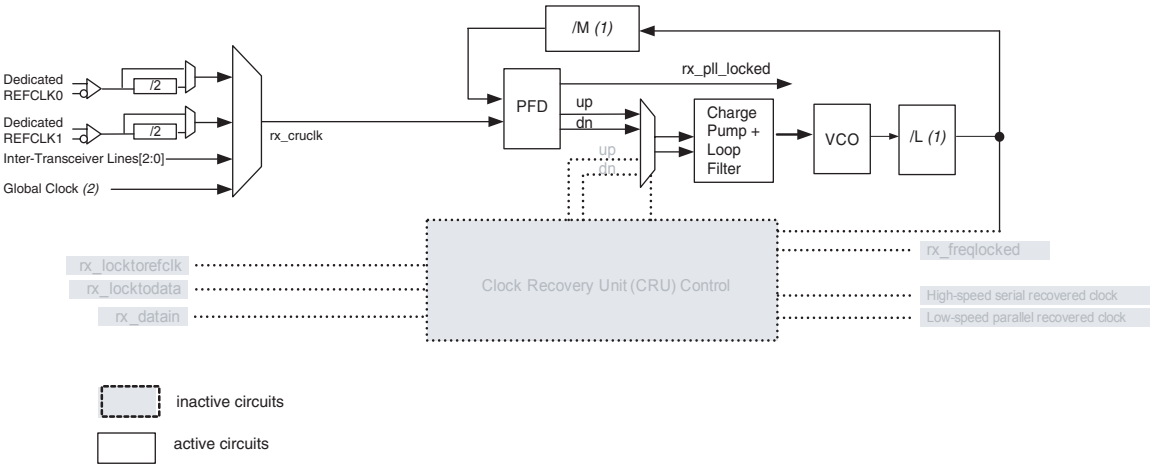
Receiver PLL

Each transceiver channel has its own receiver PLL that is fed by an input reference clock. The reference clock frequency depends on the functional mode that the transceiver channel is configured for. The clock recovery unit (CRU) controls whether the receiver PLL locks to the input reference clock (lock-to-reference mode) or the incoming serial data (lock-to-data mode). Refer to "Clock Recovery Unit (CRU)" on page 1–27 for more details on lock-to-reference and lock-to-data modes. The receiver PLL, in conjunction with the clock recovery unit, generates two clocks: a high-speed serial clock that clocks the deserializer and a low-speed parallel clock that clocks the receiver's digital logic.

This section only discusses the receiver PLL operation in lock-to-reference mode. For lock-to-data mode, refer to "Clock Recovery Unit (CRU)" on page 1–27.

Figure 1–23 shows the block diagram of the receiver PLL in lock-to-reference mode.

Figure 1–23. Receiver PLL Block Diagram



Notes to Figure 1–23:

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

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

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

- One of the two available dedicated reference clock input pins (REFCLK0 or REFCLK1) of the associated transceiver block
- PLD clock network (must be driven directly from an input clock pin and cannot be driven by user logic or enhanced PLL)
- Inter-transceiver block lines driven by reference clock input pins of other transceiver blocks

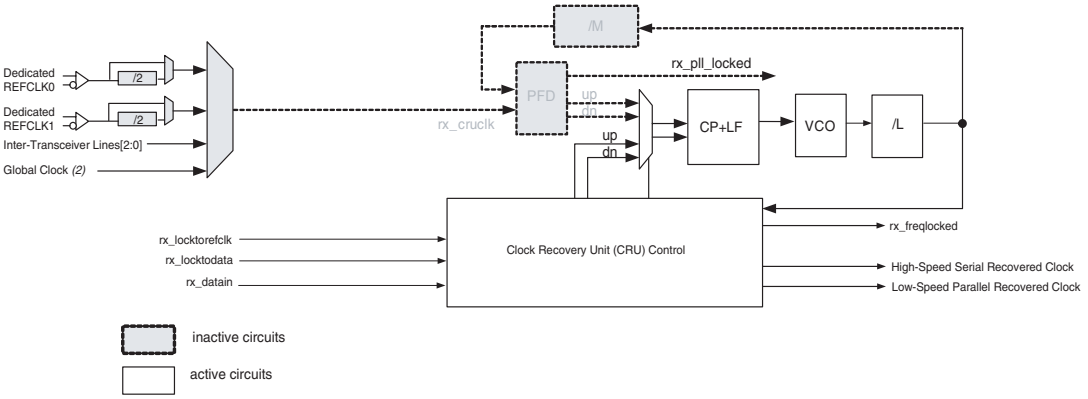
Depending on the functional mode, the Quartus II software automatically selects the appropriate receiver PLL bandwidth.

Clock Recovery Unit (CRU)

The CRU (Figure 1–24) in each transceiver channel recovers the clock from the received serial data stream. You can set the CRU to lock to the received serial data phase and frequency (lock-to-data mode) to eliminate any clock-to-data skew or to keep the receiver PLL locked to the reference clock (lock-to-reference mode). The switch between lock-to-data and lock-to-reference modes can be done automatically or manually. The CRU

in conjunction with the receiver PLL generates two clocks: a high-speed serial recovered clock that feeds the deserializer and a low-speed parallel recovered clock that feeds the receiver's digital logic.

Figure 1–24. Clock Recovery Unit



Notes to Figure 1–24:

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

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

Automatic Lock Mode

After coming out of reset in automatic lock mode, the CRU initially sets the receiver PLL to lock to the input reference clock (lock-to-reference mode). After the receiver PLL locks to the input reference clock, the CRU automatically sets it to lock to the incoming serial data (lock-to-data mode) when the following two conditions are met:

- The receiver PLL output clock is within the configured PPM frequency threshold setting with respect to its reference clock (frequency locked)
- The reference clock and receiver PLL output clock are phase matched within approximately 0.08 UI (phase locked)

When the receiver PLL and CRU are in lock-to-reference mode, the PPM detector and the phase detector circuits monitor the relationship of the reference clock to the receiver PLL VCO output. If the frequency difference is within the configured PPM setting (as set in the MegaWizard Plug-In Manager) and the phase difference is within 0.08 UI, the CRU

switches to lock-to-data mode. The switch from lock-to-reference to lock-to-data mode is indicated by the assertion of the rx_freqlocked signal.

In lock-to-data mode, the receiver PLL uses a phase detector to keep the recovered clock phase-matched to the data. If the PLL does not stay locked to data due to frequency drift or severe amplitude attenuation, the CRU switches back to lock-to-reference mode to lock the PLL to the reference clock. In automatic lock mode, the following condition forces the CRU to fall out of lock-to-data mode:

- The CRU PLL is not within the configured PPM frequency threshold setting with respect to its reference clock

The switch from lock-to-data to lock-to-reference mode is indicated by the de-assertion of rx_freqlocked signal.

When the CRU is in lock-to-data mode (rx_freqlocked is asserted), it tries to phase-match the PLL with the incoming data. As a result, the phase of the PLL output clock may differ from the reference clock due to which rx_pll_locked signal might get de-asserted. You should ignore the rx_pll_locked signal when the rx_freqlocked signal is asserted high.

Manual Lock Mode

Two optional input pins (rx_locktorefclk and rx_locktodata) allow you to control whether the CRU PLL automatically or manually switches between lock-to-reference mode and lock-to-data mode. This enables you to bypass the default automatic switchover circuitry if either rx_locktorefclk or rx_locktodata is instantiated.

When the rx_locktorefclk signal is asserted, the CRU forces the receiver PLL to lock to the reference clock. When the rx_locktodata signal is asserted, the CRU forces the receiver PLL to lock to data. When both signals are asserted, the rx_locktodata signal takes precedence over the rx_locktorefclk signal, forcing the receiver PLL to lock to data.

The PPM threshold frequency detector and phase relationship detector reaction times may be too long for some applications. You can manually control the CRU to reduce PLL lock times using the rx_locktorefclk and rx_locktodata ports. Using the manual mode may reduce the time it takes for the CRU to switch from lock-to-reference mode to lock-to-data mode. You can assert the rx_locktorefclk to initially

train the PLL to the reference clock. Once the receiver PLL locks to the reference clock, you can assert the rx_locktodata signal to force the PLL to lock to the incoming data.

When the rx_locktorefclk signal is asserted high, the rx_freqlocked signal does not have any significance and is always driven low, indicating that the CRU is in lock-to-reference mode. When the rx_locktodata signal is asserted high, the rx_freqlocked signal is always driven high, indicating that the CRU is in lock-to-data mode. If both signals are de-asserted, the CRU is in automatic lock mode.

Table 1–4 shows a summary of the control signals.

Table 1–4. CRU User Control Lock Signals

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

Deserializer

The deserializer block clocks in serial input data from the receiver buffer using the high-speed serial recovered clock and deserializes it into 10-bit parallel data using the low-speed parallel recovered clock. It feeds the de-serialized 10-bit data to the word aligner as shown in Figure 1–25.

Figure 1–25. Deserializer

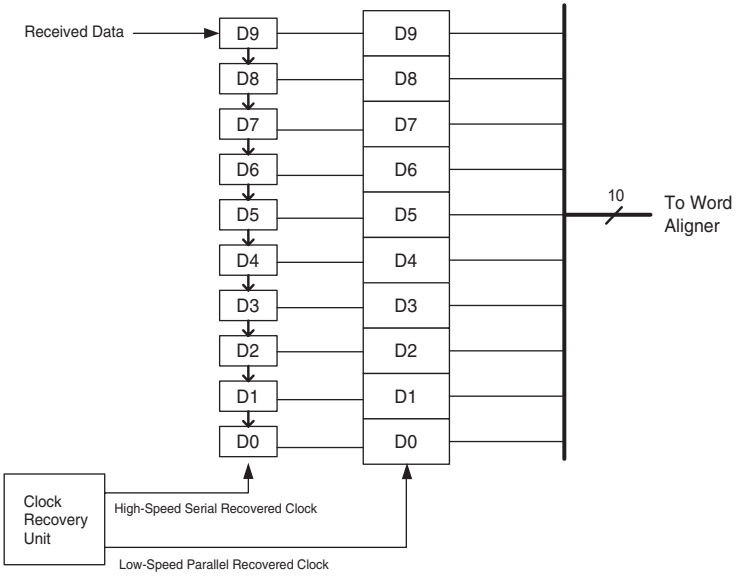
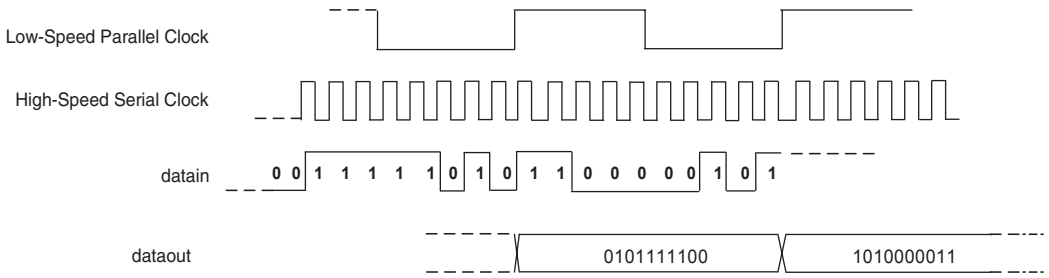


Figure 1–26 shows the serial bit order of the deserializer block input and the parallel data output of the deserializer block. A serial stream (0101111100) is de-serialized to a value 10'h17C. The serial data is assumed to be received LSB to MSB.

Figure 1–26. Deserializer Bit Order



Receiver Polarity Inversion

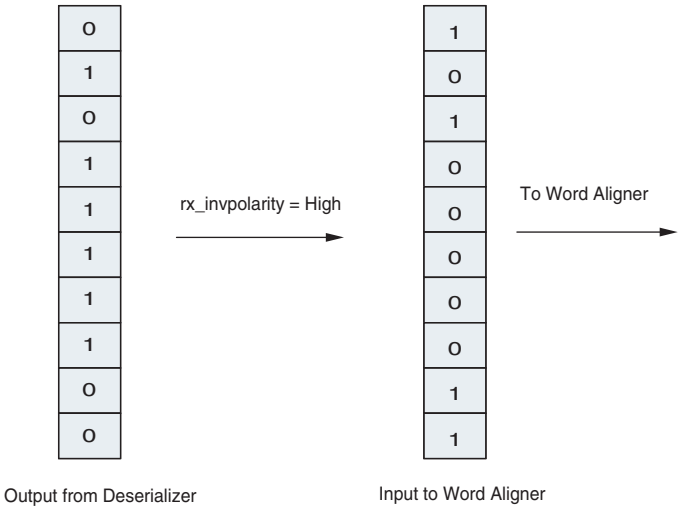
The positive and negative signals of a serial differential link might be accidentally swapped during board layout. Solutions such as a board re-spin or major updates to the PLD logic can prove expensive. The receiver polarity inversion feature is provided to correct this situation.

An optional `rx_invpolarity` port is available in all modes to dynamically enable the receiver polarity inversion feature. A high on the `rx_invpolarity` port inverts the polarity of every bit of the 10-bit input data word to the word aligner in the receiver data path. Since inverting the polarity of each bit has the same effect as swapping the positive and negative signals of the differential link, correct data is seen by the receiver. The `rx_invpolarity` is a dynamic signal and may cause initial disparity errors in an 8B/10B encoded link. The downstream system must be able to tolerate these disparity errors.

The receiver polarity inversion feature is different from the PCI Express (PIPE) 8B/10B polarity inversion feature. The receiver polarity inversion feature inverts the polarity of the data bits at the input of the word aligner. The PCI Express (PIPE) 8B/10B polarity inversion feature inverts the polarity of the data bits at the input of the 8B/10B decoder and is available only in PCI Express (PIPE) mode. Enabling the generic receiver polarity inversion and the PCI Express (PIPE) 8B/10B polarity inversion simultaneously is not allowed in PCI Express (PIPE) mode.

Figure 1–27 illustrates the receiver polarity inversion feature.

Figure 1–27. Receiver Polarity Inversion



Word Aligner

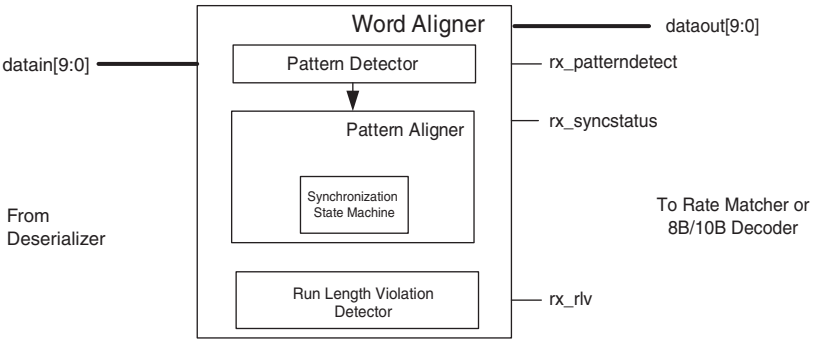
The word aligner clocks in 10-bit received data from the deserializer using the low-speed recovered clock. It restores the word boundary of the upstream transmitter based on the pre-defined word alignment character for the selected protocol. In addition to restoring the word boundary, the word aligner also implements a synchronization state machine in all functional modes to achieve lane synchronization.

The word aligner consists of three sub-modules:

- Pattern detector module
- Pattern aligner module
- Run length violation detector module

Figure 1–28 shows the block diagram for the word aligner block.

Figure 1–28. Word Aligner



Pattern Detector Module

The pattern detector looks for the configured word alignment pattern in the data clocked into the word aligner. When the pattern detector detects the word alignment pattern for the first time, it asserts the `rx_patterndetect` signal. The pattern aligner aligns the word boundary to the received word alignment pattern. Any subsequent word alignment pattern found in the same word boundary causes the `rx_patterndetect` signal to assert for one parallel clock cycle. The `rx_patterndetect` signal is not asserted if the word alignment pattern is found across the current word boundary.

All three supported functional modes, PCI Express (PIPE), GIGE, and Serial RapidIO, specify K28.5 (10'b0101111100 or 10'b1010000011) as the synchronization code group. The Quartus II software automatically programs both disparities of K28.5 control word as the word alignment pattern in all three modes.

Pattern Aligner Module

The pattern aligner module, in conjunction with the pattern detector, aligns the received data to the pre-defined word alignment pattern. The pattern aligner incorporates an automatic synchronization state machine in all supported functional modes.

Automatic Synchronization State Machine

The synchronization state machine offers automatic detection of a pre-defined number of valid alignment patterns to indicate lane synchronization and detection of code group errors for falling out of synchronization. The synchronization state machine indicates lane synchronization status on the `rx_syncstatus` port. A high signal on the `rx_syncstatus` port indicates that the lane synchronization has been achieved and a low signal indicates that the lane is not synchronized. In synchronized state (as indicated by `rx_syncstatus` asserted high), the pattern aligner does not re-align the word boundary if the pattern detector detects the word alignment pattern in the incoming data. It re-aligns the word boundary only when the synchronization state machine detects loss of synchronization state (as indicated by `rx_syncstatus` de-asserted low).

The Quartus II software automatically configures the synchronization state machine parameters; for example, number of valid synchronization characters received and number of invalid code groups received to fall out of synchronization. It selects these parameters as specified by each protocol standard (functional mode).

For more details about synchronization state machine implementation, refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*.

Run Length Violation

The programmable run-length violation circuit resides in the word aligner block and detects consecutive 1s or 0s in the data. If the data stream exceeds the preset maximum number of consecutive 1s or 0s, the violation is signified by the assertion of the `rx_rlv` signal.

The `rx_rlv` signal is not synchronized to the parallel data and appears in the logic array earlier than the run-length violation data. To ensure that the PLD can latch this signal in systems where there are frequency variations between the recovered clock and the PLD logic array clock, the `rx_rlv` signal is asserted for a minimum of two clock cycles. The `rx_rlv` signal may be asserted longer, depending on the run-length of the received data.

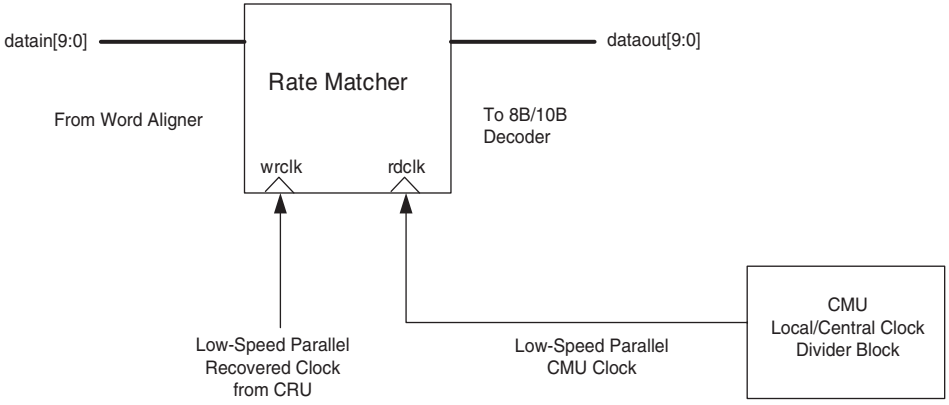
In all supported functional modes, you can set the run length violation circuitry to detect up to 160 consecutive 1s or 0s. The run length settings are available in increments of five.

Rate Matcher

In asynchronous systems, the upstream transmitter and the local receiver may be clocked with independent reference clock sources. Frequency differences in the order of a few hundred parts per million (PPM) can potentially corrupt the data at the receiver. The rate matcher compensates for small clock frequency differences between the upstream transmitter and the local receiver clocks by inserting or removing skip characters or ordered-sets from the inter-packet gap (IPG) or idle streams. It inserts a skip character or ordered-set if the local receiver is running a faster clock than the upstream transmitter. It deletes a skip character or ordered-set if the local receiver is running a slower clock than the upstream transmitter. The rate matcher is available only in PCI Express (PIPE) and GIGE functional modes. The Quartus II software automatically configures the appropriate skip character or ordered-set as specified in the Gigabit Ethernet and PCI Express Base Specification depending on the selected functional mode.

The rate matcher consists of a 20-word-deep FIFO buffer and necessary logic to detect and perform the insertion and deletion functions. The write port of the rate matcher FIFO is clocked by the low-speed parallel recovered clock. The read port is clocked by the low-speed parallel clock from the CMU central or local clock divider block (Figure 1–29).

Figure 1–29. Rate Matcher

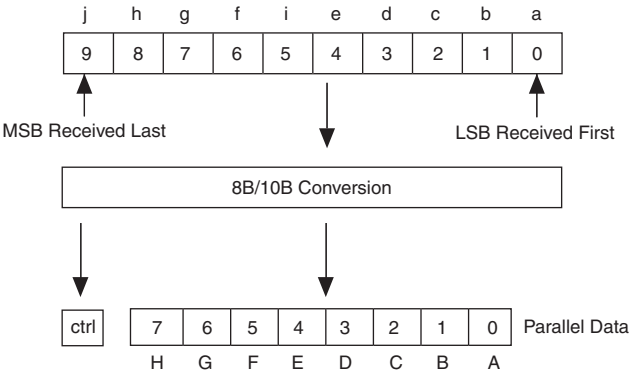


8B/10B Decoder

The 8B/10B decoder takes in 10-bit data from the rate matcher and decodes it into 8-bit data + 1-bit control identifier, thereby restoring the original transmitted data at the receiver. The decoded data is fed to the byte deserializer (in PCI Express [PIPE] and Serial RapidIO modes) or the receiver phase compensation FIFO buffer (in GIGE mode). The 8B/10B decoder conforms to IEEE 802.3 1998 edition standards.

Figure 1–30 shows a 10-bit code group decoded to an 8-bit data and a 1-bit control indicator.

Figure 1–30. 10-Bit to 8-Bit Conversion

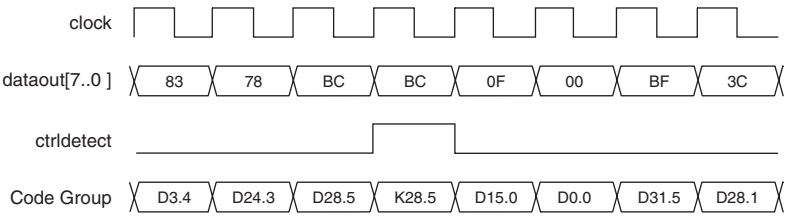


Control Code Group Detection

The 8B/10B decoder differentiates between data and control codes through the `rx_ctrldetect` port. If the received 10-bit code group is a control code group (`Kx.y`), the `rx_ctrldetect` signal is driven high, and if it is a data code group (`Dx.y`), the `rx_ctrldetect` signal is driven low.

Figure 1–31 shows an example waveform demonstrating the receipt of a K28.5 code group (BC + ctrl). The `rx_ctrldetect=1'b1` is aligned with 8'hbc, indicating that it is a control code group. The rest of the codes received are `Dx.y` code groups.

Figure 1–31. Control Code Group Detection



Code Group Error Detection

If the received 10-bit code group is not a part of valid `Dx.y` or `Kx.y` code groups, the 8B/10B decoder block asserts an error flag on `rx_errdetect` port. The error flag signal (`rx_errdetect`) has the same data path delay from the 8B/10B decoder to the PLD-transceiver interface as the invalid code group.

Disparity Error Detection

If the received 10-bit code group is detected with incorrect running disparity, the 8B/10B decoder block asserts an error flag on `rx_disperr` and `rx_errdetect` port.

Refer to the *Specifications & Additional Information* chapter in volume 2 of the *Arria GX Device Handbook* for information about the disparity calculation.

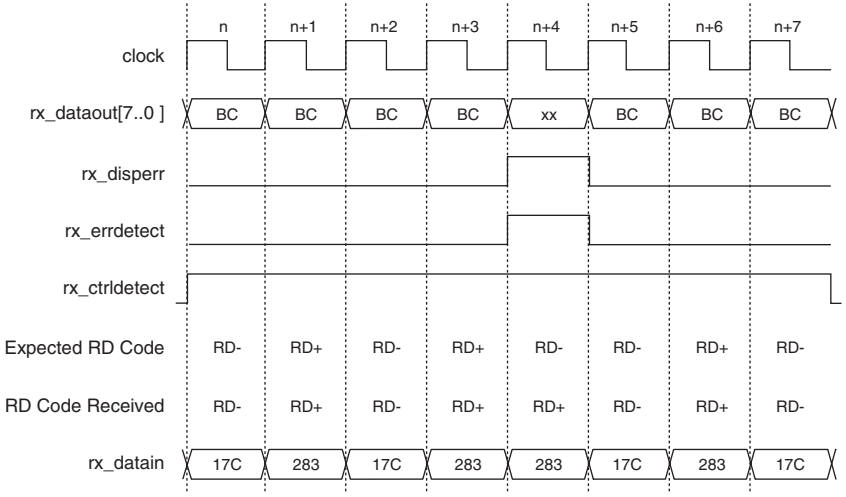
If negative disparity is calculated for the last 10-bit code group, a neutral or positive disparity 10-bit code group is expected. If the 8B/10B decoder does not receive a neutral or positive disparity 10-bit code group, the `rx_disperr` signal goes high, indicating that the code group received has a disparity error. Similarly, if a neutral or negative disparity is expected and a 10-bit code group with positive disparity is received, the `rx_disperr` signal goes high.

The detection of the disparity error might be delayed, depending on the data that follows the actual disparity error. The 8B/10B control codes terminate propagation of the disparity error. Any disparity errors propagated stop at the control code group, terminating that disparity error.

Figure 1–32 shows a case where the disparity is violated. A K28.5 code group has an 8-bit value of 8'hbc and a 10-bit value that depends on the disparity calculation at the point of the generation of the K28.5 code group. The 10-bit value is 10'b0011111010 (10'h17c) for RD– or

10'b1100000101 (10'h283) for RD+. If the running disparity at time *n* - 1 is negative, the expected code group at time must be from the RD– column. A K28.5 does not have a balanced 10-bit code group (equal number of 1s and 0s), so the expected RD code group must toggle back and forth between RD– and RD+. At time *n* + 3, the 8B/10B decoder received a RD+ K28.5 code group (10'h283), which makes the current running disparity negative. At time *n* + 4, because the current disparity is negative, a K28.5 from the RD– column is expected, but a K28.5 code group from the RD+ is received instead. This prompts *rx_disperr* to go high during time *n* + 4 to indicate that this particular K28.5 code group had a disparity error. The current running disparity at the end of time *n* + 4 is negative because a K28.5 from the RD+ column was received. Based on the current running disparity at the end of time *n* + 5, a positive disparity K28.5 code group (from the RD–) column is expected at time *n* + 5.

Figure 1–32. Disparity Error Detection



Reset Condition

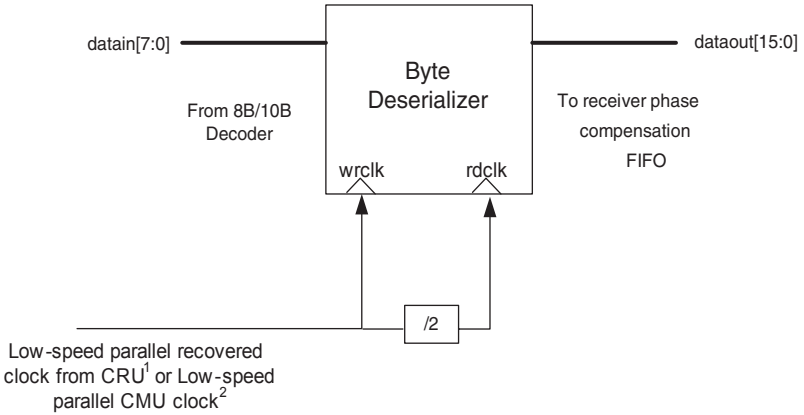
The reset for the 8B/10B decoder block is derived from the receiver digital reset (`rx_digitalreset`). When `rx_digitalreset` is asserted, the 8B/10B decoder block resets. In reset, the disparity registers are cleared and the outputs of the 8B/10B decoder block are driven low. After reset, the 8B/10B decoder starts with unknown disparity, depending on the disparity of the data it receives. The decoder calculates the initial running disparity based on the first valid code group received.

The receiver block must be word aligned after reset before the 8B/10B decoder can decode valid data or control codes. If word alignment has not been achieved, the data from the 8B/10B decoder should be discarded and considered invalid.

Byte Deserializer

In PCI Express (PIPE) and Serial RapidIO functional modes, the byte deserializer (Figure 1–33) takes in 8-bit wide data from the 8B/10B decoder and deserializes it into a 16-bit wide data at half the speed. This allows clocking the PLD-transceiver interface at half the speed as compared to the receiver PCS logic. The byte deserializer is bypassed in GIGE mode.

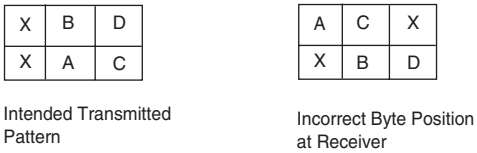
Figure 1–33. Byte Deserializer



Notes to Figure 1–33:
(1) Write port is clocked by low-speed parallel recovered clock if rate matcher is not used.
(2) Write port is clocked by low-speed parallel CMU clock if rate matcher is used.

If the byte deserializer is used, the byte ordering at the receiver output might be different than what was transmitted. Figure 1–34 shows the 16-bit transmitted data pattern with A at the lower byte, followed by B at the upper byte. C and D follow in the next lower and upper bytes, respectively. At the byte deserializer, byte A arrives when it is stuffing the upper byte instead of stuffing the lower byte. This is a non-deterministic swap because it depends on PLL lock times and link delay. You must implement byte-ordering logic in the PLD to correct this situation.

Figure 1–34. Intended Transmitted Pattern and Incorrect Byte Position at Receiver After Byte Serializer



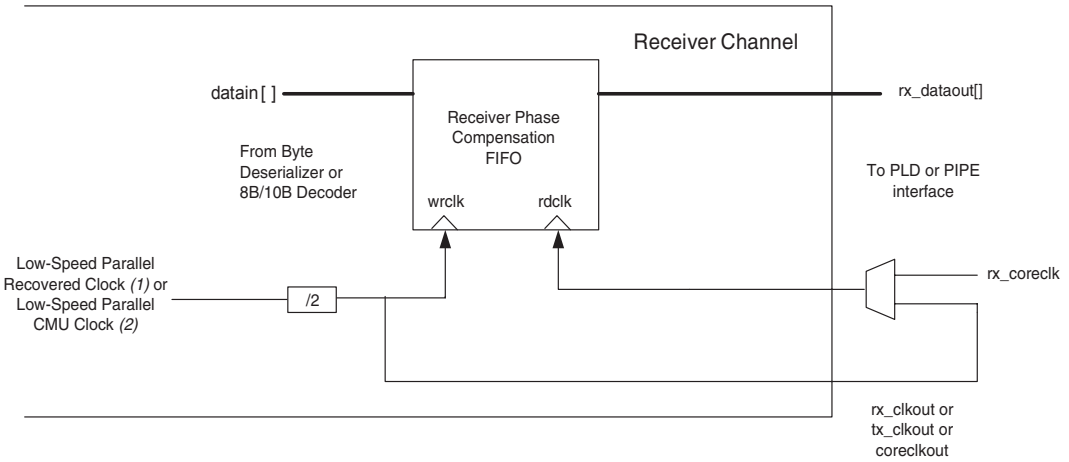
Receiver Phase Compensation FIFO buffer

A receiver phase compensation FIFO buffer (Figure 1–35) is located at each receiver channel's logic array interface. It compensates for the phase difference between the receiver PCS clock and the local PLD clock.

In individual channel mode with rate matcher (PCI Express [PIPE] ×1 and GIGE), the low-speed clock output from the local clock divider block of each channel or its divide-by-two version clocks the write port of its receiver phase compensation FIFO buffer. This clock is also forwarded to the logic array (`tx_clkout`). In individual channel mode without rate matcher (Serial RapidIO), the low-speed parallel recovered clock from the CRU is divided by two. This divide-by-two clock clocks the write port of its receiver phase compensation FIFO buffer. This clock is also forwarded to the logic array (`rx_clkout`). If the `rx_coreclk` port is instantiated, the clock signal on the `rx_coreclk` port clocks the read port of the FIFO. If the `rx_coreclk` port is not instantiated, the clock signal on `tx_clkout` (if rate matcher is used) or `rx_clkout` (if rate matcher is not used) port is automatically fed back to clock the read port of the FIFO.

In bonded channel mode (×4 PCI Express [PIPE]), the low-speed clock output from the central clock divider block is divided by two. This divide-by-two clock clocks the write port of the receiver phase compensation FIFO buffers in all channels. This clock is also forwarded to the logic array (`coreclkout`). If the `rx_coreclk` port is instantiated, the clock signal on the `rx_coreclk` port clocks the read port of the FIFO. If the `rx_coreclk` port is not instantiated, the clock signal on the `coreclkout` port is automatically fed back to clock the read port of the FIFO in all channels within the transceiver block.

Figure 1–35. Receiver Phase Compensation FIFO Buffer



Notes to Figure 1–35:
(1) Write port is clocked by low-speed parallel recovered clock when rate matcher is not used.
(2) Write port is clocked by low-speed parallel CMU clock when rate matcher is used.

Transceiver Channel Clock Distribution

This section describes clock distribution within the transceiver channel for all supported functional modes. Clock distribution can be classified into two categories:

- Individual-channel mode clocking—GIGE, Serial RapidIO, and PCI Express [PIPE] ×1 functional modes fall in this category
- Bonded-channel mode clocking—PCI Express (PIPE) ×4 functional mode falls in this category

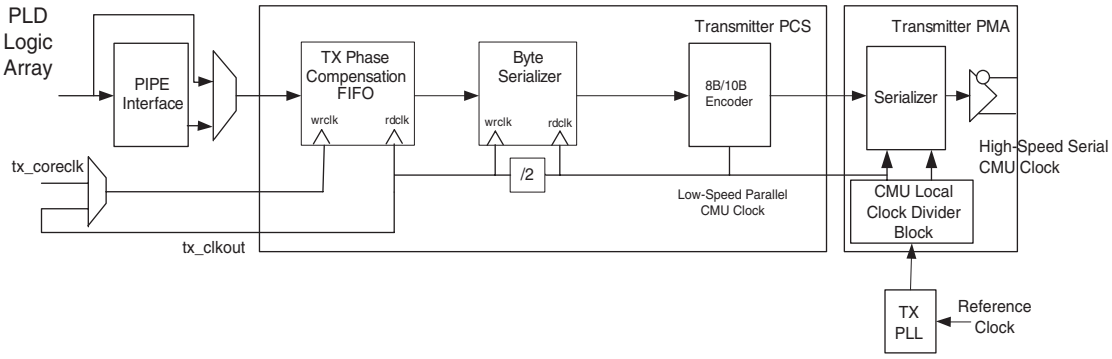
Individual-Channel Mode Clocking

In GIGE, Serial RapidIO, and PCI Express (PIPE) ×1 functional modes, each channel within the transceiver block forms an independent link. The blocks in the transmitter and receiver data paths of each channel are clocked independently and do not share any control signals. The transmitter channel-to-channel skew in individual channel modes could be relatively higher compared to bonded-channel mode.

Individual-Channel Mode Transmitter Clock Distribution

Figure 1–36 shows clock distribution within a transmitter channel when configured in one of the individual-channel modes.

Figure 1–36. Individual-Channel Mode Transmitter Clock Distribution



The transmitter PLL multiplies the input reference clock to generate a high-speed serial clock at a frequency that is half the data rate of the configured functional mode. This high-speed serial clock is fed to the local clock divider block in each channel within the transceiver block. Depending on the configured functional mode, the local clock divider block divides the high-speed serial clock to generate the low-speed parallel clock that clocks the transmitter PCS logic in the associated channel.

Table 1–5 shows high-speed serial clock frequencies, low-speed parallel clock frequencies, and PLD-transceiver interface frequencies for individual-channel functional modes.

Table 1–5. CMU Output Clock Frequencies

Functional Mode	High-Speed Serial Clock Frequency (MHz)	Low-Speed Parallel Clock Frequency (MHz)	PLD-Transceiver Interface Frequency (MHz)
PCI Express (PIPE) x1 (2.5 Gbps)	1250	250	125
GIGE (1.25 Gbps)	625	125	125
Serial RapidIO (1.25 Gbps)	625	125	62.5
Serial RapidIO (2.5 Gbps)	1250	250	125

In functional modes that use the byte serializer (PCI Express [PIPE] x1 and Serial RapidIO), the low-speed parallel clock is divided by two before being driven on the tx_clkout port. In GIGE mode that does not use the byte serializer, the low-speed parallel clock is directly driven on the tx_clkout port. If the tx_coreclk port is not instantiated, the Quartus II software automatically clocks the write port of the transmitter phase compensation FIFO in each channel with the tx_clkout signal of channel 0 (tx_clkout[0]) of that transceiver block. In PCI Express (PIPE) x1 mode, the tx_clkout signal also clocks each channel's PCI Express (PIPE) interface logic. The read port of the transmitter phase compensation FIFO is clocked by the low-speed parallel clock output (or its divide-by-two version if byte serializer is used) of the channel's local clock divider block. The read port of the byte deserializer block (if used) is clocked by the low-speed parallel clock and the write port is clocked by its divide-by-two version. The 8B/10B encoder is clocked by the low-speed parallel clock. The 10-bit input data to the serializer block is clocked by the low-speed parallel clock and the serialized data is clocked out by the high-speed serial clock.

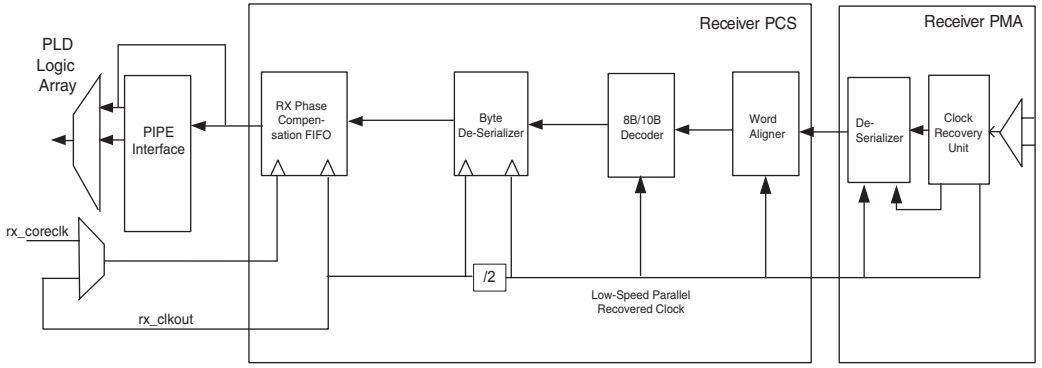
Individual-Channel Mode Receiver Clock Distribution

The clock distribution within a receiver channel varies depending on whether rate matcher is used or not. GIGE and functional modes have a rate matcher in the receiver data path to compensate for the frequency difference between the upstream transmitter and local receiver clock. Serial RapidIO and Low-Latency (Synchronous) PCI Express (PIPE) ×1 modes do not have a rate matcher in the receiver data path.

Receiver Clock Distribution Without the Rate Matcher

Figure 1–37 shows clock distribution within a receiver channel when configured in an individual-channel mode without a rate matcher (Serial RapidIO and Low-Latency [Synchronous] PCI Express [PIPE] ×1 modes).

Figure 1–37. Receiver Clock Distribution in Individual-Channel Mode Without the Rate Matcher



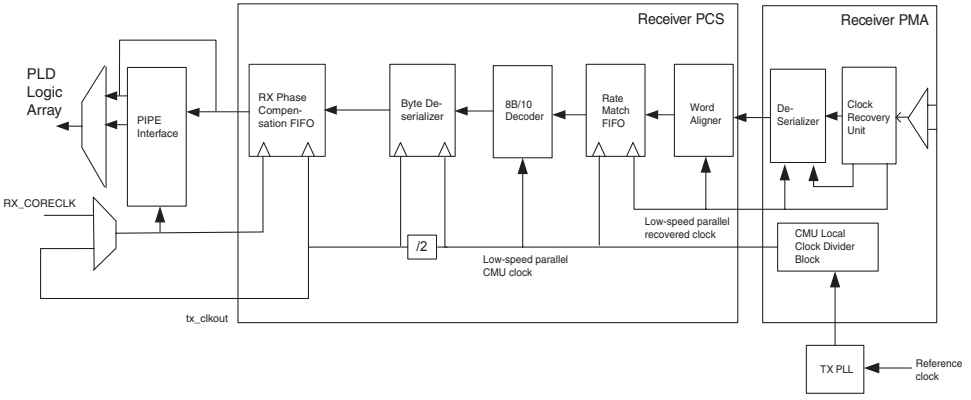
The clock recovery unit (CRU) in each channel generates a high-speed serial recovered clock and a low-speed parallel recovered clock. The de-serializer clocks in the serial received data using the high-speed serial recovered clock and clocks out the 10-bit de-serialized data using the low-speed parallel recovered clock.

The low-speed parallel recovered clock clocks the word aligner, 8B/10B decoder, and the write port of the byte deserializer. The low-speed parallel recovered clock is divided by two before being driven on the `rx_clkout` port of the corresponding channel. This divide-by-two version of the recovered clock clocks the write port of the receiver phase compensation FIFO. If the `rx_coreclk` port is not instantiated, the Quartus II software automatically clocks the read port of the receiver phase compensation FIFO in each channel with its `rx_clkout` signal.

Receiver Clock Distribution With the Rate Matcher

Figure 1–38 shows clock distribution within a receiver channel when configured in an individual-channel mode with a rate matcher (GIGE and PCI Express (PIPE) ×1 modes).

Figure 1–38. Receiver Clock Distribution in Individual-Channel Mode With the Rate Matcher



As seen in Figure 1–38, the deserializer output, the word aligner, and the write port of the rate matcher are clocked by the low-speed parallel recovered clock of its associated channel. The rate matcher clocks the data out using the low-speed parallel clock output from its associated CMU local clock divider block. The 8B/10B decoder and the write port of the byte deserializer (if used) are clocked by the low-speed parallel clock. The write port of the receiver phase compensation FIFO are clocked by the low-speed parallel clock or its divide-by-two version (if the functional mode uses the byte deserializer). This clock is also forwarded to the PLD logic array on the `tx_clkout` port. If the `rx_coreclk` port is not instantiated, the Quartus II software automatically clocks the read port of the receiver phase compensation FIFO in each channel with its associated `tx_clkout` signal.

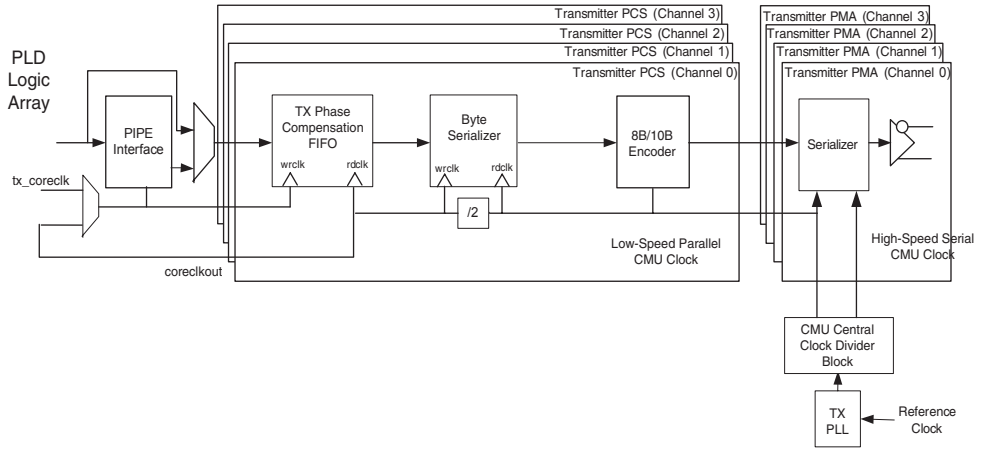
Bonded-Channel Mode Clocking

In PCI Express (PIPE) ×4 functional mode, each channel within the transceiver block is a part of a four-lane PCI Express (PIPE) link. To reduce lane-to-lane skew, the blocks in the transmitter and receiver data paths share clocks and control signals.

Bonded-Channel Mode Transmitter Clock Distribution

Figure 1–39 shows clock distribution within a transmitter channel when configured in a bonded-channel mode (PCI Express [PIPE] ×4).

Figure 1–39. Bonded-Channel Mode Transmitter Clock Distribution



The transmitter PLL multiplies the input reference clock to generate a high-speed serial clock at a frequency that is half the data rate of the configured functional mode. This high-speed serial clock is fed to the central clock divider block in the transceiver block. The central clock divider block divides the high-speed serial clock to generate the low-speed parallel clock that clocks the transmitter PCS logic in all channels within the transceiver block.

Table 1–6 shows high-speed serial clock and low-speed parallel clock frequencies for PCI Express (PIPE) ×4 bonded-channel mode.

Table 1–6. CMU Output Clock Frequencies

Functional Mode	High-Speed Serial Clock Frequency (MHz)	Low-Speed Parallel Clock Frequency (MHz)	PLD-Transceiver Interface Frequency (MHz)
PCI Express (PIPE) ×4 (2.5 Gbps)	1250	250	125

The low-speed parallel clock is divided by two before being driven on the coreclkout port. If the tx_coreclk port is not instantiated, the Quartus II software automatically clocks the write port of the transmitter phase compensation FIFO in all channels with the coreclkout signal. The coreclkout signal also clocks each channel's PIPE interface logic. The read port of the transmitter phase compensation FIFO is clocked by the divide-by-two version of the low-speed parallel clock output from central clock divider block. The read port of the byte deserializer block is clocked by the low-speed parallel clock and the write port is clocked by its divide-by-two version. The 8B/10B encoder is clocked by the low-speed parallel clock. The 10-bit input data to the serializer block is clocked by the low-speed parallel clock and the serialized data is clocked out by the high-speed serial clock output from the central clock divider block (refer to Figure 1–39).

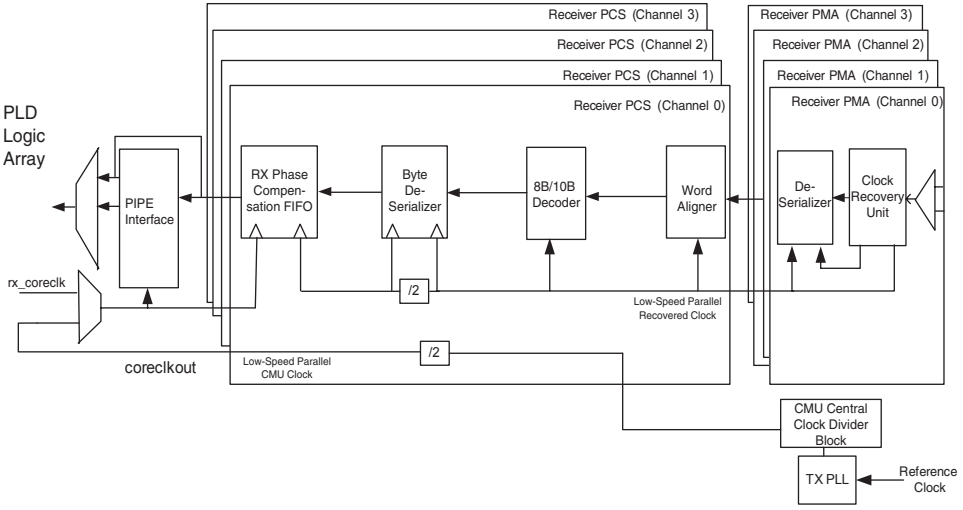
Bonded-Channel Mode Receiver Clock Distribution

In PCI Express (PIPE) ×4 mode, you can choose to implement or bypass the rate match FIFO depending on your system requirements. Bypassing the rate matcher FIFO (Low-Latency PCI Express [PIPE] mode) yields lower latency through the receiver path. The clocking within the receiver channel varies depending on whether rate matcher is used.

Bonded-Channel Mode Receiver Clock Distribution Without the Rate Matcher

Figure 1–40 shows clock distribution within a receiver channel when configured in PCI Express (PIPE) ×4 mode without the rate matcher.

Figure 1–40. Bonded Channel Mode Receiver Clock Distribution Without the Rate Matcher

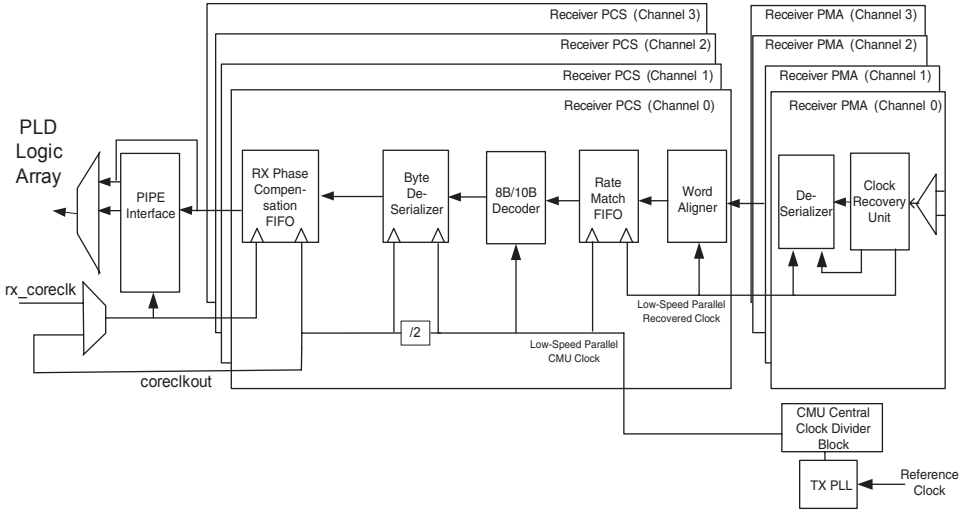


As seen in Figure 1–39, the deserializer output, the word aligner, the 8B/10B decoder, and the write port of the byte deserializer are clocked by the low-speed parallel recovered clock of its associated channel. The read port of the byte deserializer and the write port of the receiver phase compensation FIFO are clocked by the divide-by-two version of the low-speed parallel recovered clock. If the `rx_coreclk` port is not instantiated, the write port of the receiver phase compensation FIFO is automatically clocked by the `coreclkout` signal.

Bonded-Channel Mode Receiver Clock Distribution With the Rate Matcher

Figure 1–41 shows clock distribution within a receiver channel when configured in PCI Express (PIPE) x4 mode with the rate matcher.

Figure 1–41. Bonded-Channel Mode Receiver Clock Distribution With the Rate Matcher



As seen in Figure 1–40, the deserializer output, the word aligner, and the write port of the rate matcher in each of the four channels are clocked by the low-speed parallel recovered clock of its associated channel. The rate matcher in all four channels clocks the data out using the low-speed parallel clock output from the CMU central clock divider block. The 8B/10B decoder and the byte deserializer are clocked by the low-speed parallel clock. The write port of the receiver phase compensation FIFO is clocked by the divide-by-two version of the low-speed parallel clock. This clock is also forwarded to the PLD logic array on the `coreclkout` port. If the `rx_coreclk` port is not instantiated, the Quartus II software automatically clocks the read port of the receiver phase compensation FIFO in all four channels with the `coreclkout` signal.

PLD-Transceiver Interface Clocking

The transmitter phase compensation FIFO present at each channel's PLD-transmitter interface compensates for the phase difference between the PLD clock that produces the data to be transmitted and the transmitter PCS clock. The receiver phase compensation FIFO present at each channel's PLD-receiver interface compensates for the phase difference between the PLD clock that processes the received data and the receiver PCS clock.

Depending on the functional mode, the Quartus II software automatically selects appropriate clocks to clock the read port of the transmitter phase compensation FIFO and the write port of the receiver phase compensation FIFO.

The write clock of the transmitter phase compensation FIFO and the read clock of the receiver phase compensation FIFO are part of the PLD-transceiver interface clocks. Arria GX transceivers provide the following two options for selecting these PLD-transceiver interface clocks:

- Automatic Phase Compensation FIFO clock selection
- User Controlled Phase Compensation FIFO clock selection

The automatic phase compensation FIFO clock selection is a simpler option but could lead to higher clock resource utilization as compared to user controlled phase compensation FIFO clock selection. This could be critical in designs with high clock resource requirements.

Automatic Phase Compensation FIFO Clock Selection

If you do not instantiate the `tx_coreclk` and `rx_coreclk` ports for the Arria GX transceiver instance in the MegaWizard Plug-In Manager, the Quartus II software automatically selects appropriate clocks to clock the write port of the transmitter phase compensation FIFO and the read clock of the receiver phase compensation FIFO.

Table 1–7 lists the clock sources that the Quartus II software automatically selects for the transmitter and receiver phase compensation FIFOs, depending on the functional mode.

Table 1–7. Clock Sources for the Transmitter and Receiver Phase Compensation FIFOs (Part 1 of 2)

Functional Mode	Write port clock selection for Transmitter Phase Compensation FIFO	Read port clock selection for Receiver Phase Compensation FIFO
Individual-channel mode with rate matcher (PCI Express [PIPE] ×1, GIGE)	`tx_clkout[0]` from channel 0 clocks the FIFO write port in all channels in the same transceiver block.	`tx_clkout[0]` from channel 0 clocks the FIFO read port in all channels in the same transceiver block.

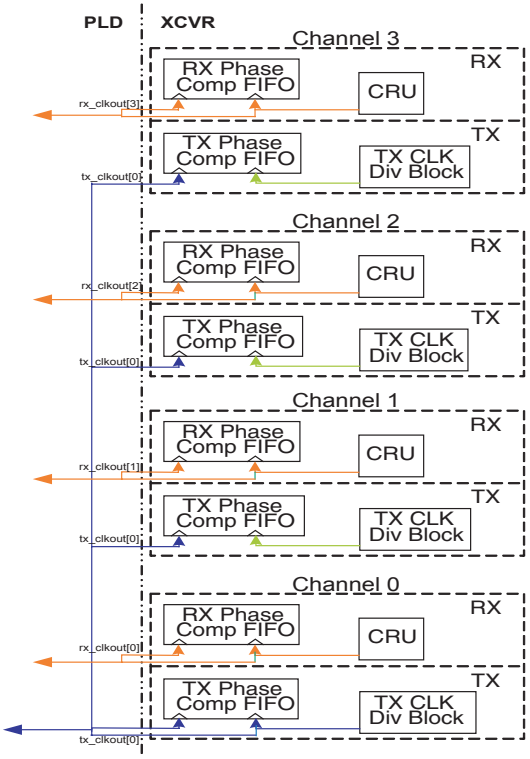
Table 1–7. Clock Sources for the Transmitter and Receiver Phase Compensation FIFOs (Part 2 of 2)

Functional Mode	Write port clock selection for Transmitter Phase Compensation FIFO	Read port clock selection for Receiver Phase Compensation FIFO
Individual-channel mode without rate matcher (Serial RapidIO, Low-latency PCI Express [PIPE] ×1)	`tx_clkout[0]` from channel 0 clocks the FIFO write port in all channels in the same transceiver block.	`rx_clkout` from each channel clocks the FIFO read port of its associated channel.
Bonded-channel mode with/without rate matcher (PCI Express [PIPE] ×4)	`coreclkout` clocks the FIFO write port in all channels in the same transceiver block.	`coreclkout` clocks the FIFO read port in all channels in the same transceiver block.

In an individual-channel mode without rate matcher (Serial RapidIO), a total of five global/regional clock resources per transceiver block are utilized by the PLD-transceiver interface clocks. Four clock resources are utilized by the `rx_clkout` signal of each channel being routed back to clock the read port of its receiver phase compensation FIFO. One clock resource is utilized by the `tx_clkout[0]` signal of channel 0 being routed back to clock the write port of all transmitter phase compensation FIFOs in the transceiver block.

Figure 1–42 shows the minimum PLD-Interface clock utilization per transceiver block when configured in individual-channel mode without the rate matcher.

Figure 1–42. Minimum PLD-Interface Clock Utilization Per Transceiver Block Without the Rate Matcher

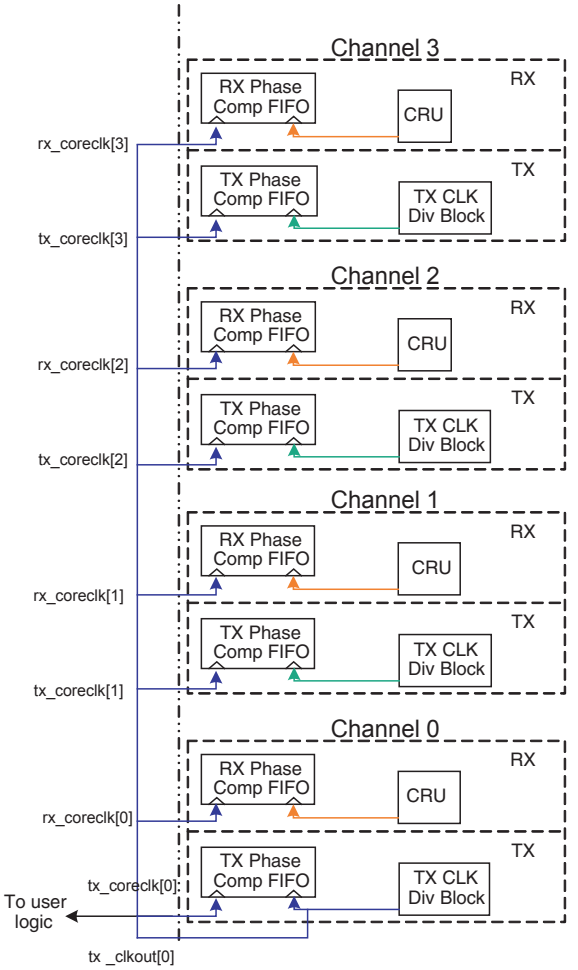


The PLD-transceiver clock utilization can be reduced by driving the transmitter and receiver phase compensation FIFOs with a single clock. This is possible only if the driving clock is frequency locked to the transceiver output clocks (`tx_clkout`, `coreclkout`, or `rx_clkout`). To control the write and read clock selection for the transmitter and receiver phase compensation FIFO, you must instantiate the `tx_coreclk` and `rx_coreclk` ports for the transceiver channels.

User Controlled Phase Compensation FIFO Clock Selection

Instead of the Quartus II software automatically selecting the write and read clocks of the transmitter and receiver phase compensation FIFOs, respectively, you can manually connect appropriate clocks by instantiating the `tx_coreclk` and `rx_coreclk` ports in the MegaWizard Plug-In Manager. For all like channels configured in the same functional mode and running off the same clock source, you can connect the `tx_coreclk` and `rx_coreclk` ports of all channels together and drive them using the same clock source. You can use a PLD clock input pin or a transceiver clock (`tx_clkout[0]/coreclkout/rx_clkout`) to clock the `tx_coreclk/rx_coreclk` ports (Figure 1–43).

Figure 1–43. User Controlled Phase Compensation FIFO Clock



If the `rx_clkout` is used as a driver, it can only drive the `rx_coreclk` ports. It cannot drive the `tx_coreclk` ports. If `tx_coreclk` and `rx_coreclk` need to be driven with the same clock, you must use the `tx_clkout` signal as the clock driver.

If the clock signal on `tx_coreclk` is used to clock the write side of the transmitter phase compensation FIFO, you must make sure that it is frequency locked to the transmitter PCS clock reading from the FIFO. If the clock signal on `rx_coreclk` is used to clock the read side of the

receiver phase compensation FIFO, you must make sure that it is frequency locked to the receiver PCS clock writing into the FIFO. Any frequency differences may cause data corruption.

To help guard against incorrect usage, the use of the `tx_coreclk` and `rx_coreclk` options requires clock assignments in the assignment organizer. If no assignments are used, the Quartus II software will issue a compilation error.

There are four settings to enable the PLD interface clocking options:

- **Stratix II GX/Arria GX GXB Shared Clock Group Setting**
- **Stratix II GX/Arria GX GXB Shared Clock Group Driver Setting**
- **Stratix II GX/Arria GX 0PPM Clock Group Setting**
- **Stratix II GX/Arria GX 0PPM Clock Group Driver Setting**

There are two main settings, Shared Clock and 0 PPM Clock, each with a driver and clock group setting. When specifying clock groups, an integer identifier is used as the group name to differentiate other clock group settings from one another.

The **Stratix II GX/Arria GX GXB Shared Clock Group Setting** is the safest assignment. The Quartus II compiler analyzes the netlist during compilation to ensure transmitter channel members are derived from the same source. The Quartus II software gives a fitting error for incompatible assignments. The software cannot check for the output of the receiver frequency locked to the driving clock as the exact frequency is dictated by the upstream transmitter's source clock. You must ensure that the `rx_coreclk` is derived from the same source clock as the upstream transmitter.

The **Stratix II GX/Arria GX GXB Shared Clock Group Driver Setting** assignment must be made to the source channel of the `tx_clkout` or `coreclkout`. Specifying anything except the transmitter channels (the source for the `tx_clkout` or `coreclkout`) results in a fitter error. If the source clock is not from `tx_clkout` or `coreclkout` (for example, the source is from `rx_clkout` or from a PLD clock input), the 0PPM setting must be used instead.

For example, in a synchronous system, the transmitter and receiver are running off the same clock. To make `tx_clkout[0]` the clock driver, the **Stratix II GX/Arria GX GXB Shared Clock Group Driver Setting** is made in the assignment editor on the `tx_dataout[0]` name. You can use a group identifier value of "1" to identify the group that this driver feeds. The **Stratix II GX/Arria GX GXB Shared Clock Group Setting** is made to all the `rx_datain` channels that the `tx_dataout[0]` output clock drives.

The other tx_dataout channels do not need an assignment as the Quartus II software automatically groups the like transmitters in a transceiver block. A group identifier value of "1" is also made to the rx_datain assignments.

The assignments in the Assignment Editor are shown in Table 1–8.

Table 1–8. Assignment Editor

To:	tx_dataout[0]
Assignment name:	Stratix II GX/Arria GX GXB Shared Clock Group Driver Setting
Value:	1
To:	rx_datain[] (note that the [] signifies the entire rx_datain group)
Assignment name:	Stratix II GX/Arria GX GXB Shared Clock Group Setting
Value:	1

The **Stratix II GX/Arria GX 0PPM Clock Group Setting** is for more advanced users that know the clocking configuration of the entire system and want to reduce the PLD global clock resource and PLD interface clock resource utilization. The Quartus II compiler does not perform any checking on the clock source. It is up to you to ensure that there is no frequency difference from the associated transceiver clock of the group and the driving clock to the tx_coreclk and rx_coreclk ports.

The **Stratix II GX/Arria GX 0PPM Clock Group Driver Setting** can be made to any of the transceiver output clocks (tx_clkout, rx_clkout, and coreclk_out) as well as any PLD clock input pins, transceiver dedicated REFCLK pin, or PLD PLL output. User logic cannot be used as a driver. As with the shared clock group setting, the driver setting for the transceiver output clocks is made to the associated channel. For example, for tx_clkout or coreclk_out, the transmitter channel name is specified. When the rx_clkout is the driver, the receiver channel name of the associated rx_clkout is specified. For the PLD input clock pins and the transceiver REFCLK pins, the name of the clock pin can be specified. For the PLL output, the PLL clock output port of the PLL can be found in the Node Finder and entered as the driver name. An integer value is specified for the group identification.

The **Stratix II GX/Arria GX 0 PPM Clock Group Setting** is made to the transmitter or receiver channel names.

The assignments in the Assignment Editor are shown in Table 1–9.

Table 1–9. Assignment Editor

To:	`tx_dataout[0]`, `pld_clk_pin_name`, `refclk_pin`, and `pll_outclk`
Assignment name:	Stratix II GX/Arria GX GXB 0 PPM Clock Group Driver Setting
Value:	1
To:	`rx_datain[]` and `tx_dataout[]`
Assignment name:	Stratix II GX/Arria GX GXB 0 PPM Clock Group Setting
Value:	1

For a complete set of features supported in each protocol, refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*.

Referenced Documents

This chapter references the following documents:

- *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
- *Specifications and Additional Information* chapter in volume 2 of the *Arria GX Device Handbook*

Document Revision History

Table 1–10 shows the revision history for this chapter.

Table 1–10. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
August 2007, v1.2	Added the "Referenced Documents" section.	—
	Minor text edits.	—
June 2007 v1.1	Added GIGE information.	—
May 2007 v1.0	Initial release.	—

2. Arria GX Transceiver Protocol Support and Additional Features

AGX52002-1.2

Introduction

Arria™ GX transceivers have dedicated physical coding sublayer (PCS) and physical media attachment (PMA) circuitry to support PCI Express (PIPE), Gigabit Ethernet (GIGE), and Serial RapidIO™ protocols.

Table 2–1 lists the Arria GX transceiver datapath modules employed in each mode.

Table 2–1. Arria GX Transceiver Datapath Modules

Functional Mode	Transmitter /Receiver Phase Compensation FIFO	Byte Serializer/ Deserializer	8B/10B Encoder/ Decoder	Word Aligner	Rate Matcher	PLD-Transceiver Interface Width (bits)	PLD-Transceiver Interface Frequency (MHz)	PCS Frequency (MHz)
PCI Express (PIPE)	✓	✓	✓	✓	✓(1)	16	125	250
GIGE	✓		✓	✓	✓	8	125	125
Serial RapidIO (1.25Gbps)	✓	✓	✓	✓		16	62.5	125
Serial RapidIO (2.5Gbps)	✓	✓	✓	✓		16	125	250

Note to Table 2–1:
(1) The rate matcher can be bypassed in low-latency (synchronous) PCI Express (PIPE) mode.

PCI Express (PIPE) Mode

PCI Express is an evolution of peripheral component interconnect (PCI). PCI is bandwidth-limited for today's applications because it relies on synchronous single-ended type signaling with a wide multi-drop data bus. Clock and data-trace matching is required with PCI. PCI Express uses differential serial signaling with an embedded clock to enable an effective data rate of 2 Gbps per lane to overcome the limitations of PCI.

Arria GX transceivers support ×1 (single-lane) and ×4 (four-lane) link widths when configured in PCI Express (PIPE) mode. The Arria GX family supports up to twelve duplex (transmitter and receiver) ×1 links and up to three ×4 links per device. Transceiver channels configured in ×4

PCI Express (PIPE) mode must be physically located in the same transceiver block with logical Lane 0 assigned to physical Channel 0, logical Lane 1 assigned to physical Channel 1 and so on.

In addition to providing the transceiver PCS and PMA circuitry, Arria GX transceivers support the following protocol-specific features:

- PCI Express synchronization state machine
- Receiver detection
- Electrical idle generation/detection
- Beacon transmission
- Polarity inversion
- Power state management

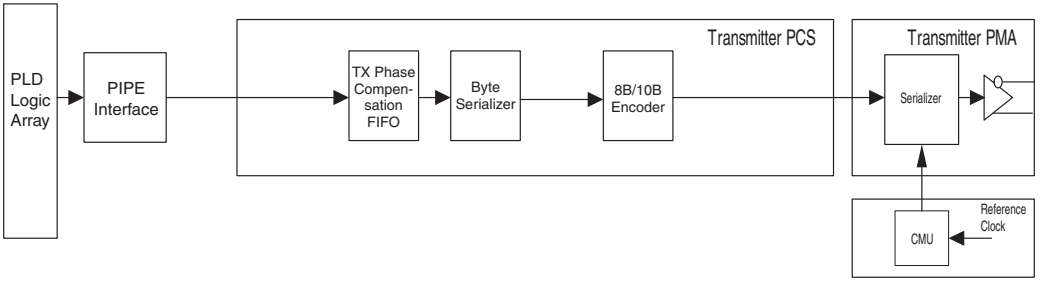
This section is organized into transmitter and receiver data path modules when configured for PCI Express (PIPE) mode. The description for each module only covers details specific to PCI Express (PIPE) functional mode support. Familiarity of PCI Express protocol and PCI Express (PIPE) specifications is assumed.

For a general description of each module, refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

PCI Express (PIPE) Mode Transmitter Architecture

This section lists sub-blocks within the transmitter channel configured in PCI Express (PIPE) mode (Figure 2–1). The sub-blocks are described in order from the PLD transceiver parallel interface to the serial transmitter buffer.

Figure 2–1. PCI Express (PIPE) Transmitter Architecture



Clock Multiplier Unit

The clock multiplier unit (CMU) takes in a reference clock and synthesizes the clocks that are used to clock the transmitter digital logic (PCS), the serializer, and the PLD-transceiver interface.

Refer to the Clock Multiplier Unit section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about CMU architecture.

In ×1 PCI Express (PIPE) mode, the CMU block consists of:

- Transmitter PLL that generates high-speed serial clock for the serializer
- Local clock divider block that generates low-speed parallel clock for transmitter digital logic and PLD-transceiver interface

In ×4 PCI Express (PIPE) mode, the CMU block consists of:

- Transmitter PLL that generates high-speed serial clock for the serializer
- Central clock divider block that generates low-speed parallel clock for transmitter digital logic and PLD-transceiver interface of each channel in the transceiver block

Input Reference Clock

In PCI Express (PIPE) mode, the only supported input reference clock frequency is 100 MHz.

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

- One of the two available dedicated reference clock input pins (`REFCLK0` or `REFCLK1`) of the associated transceiver block
- Inter-transceiver block lines driven by reference clock input pins of other transceiver blocks

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

Table 2–2 specifies the input reference clock options available in PCI Express (PIPE) mode.

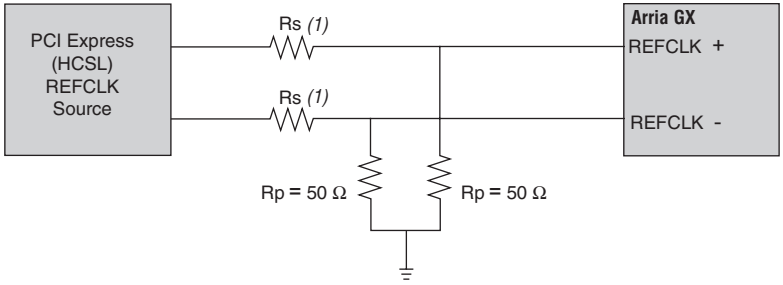
Table 2–2. PCI Express (PIPE) Mode Input Reference Clock Specifications

Frequency	I/O Standard	Coupling	Termination
100 MHz	1.2V PCML, 1.5V PCML, 3.3V PCML, Differential LVPECL, LVDS	AC	On-chip
	HCSL *(1)*	DC *(2)*	Off-chip

Notes to Table 2–2:
(1) In PCI Express (PIPE) mode, you have the option of selecting the HCSL standard for the reference clock if compliance to PCI Express is required. The Quartus® II software automatically selects DC coupling with external termination for the signal if configured as HCSL.
(2) Refer to Figure 2–2 for an example termination scheme.

Figure 2–2 shows an example termination scheme for the reference clock signal when configured as HCSL.

Figure 2–2. DC Coupling and External Termination Scheme for PCI Express Reference Clock



Note to Figure 2–2:
(1) Select resistor values as recommended by the PCI Express clock source vendor.

Clock Synthesis

In PCI Express (PIPE) mode, the reference clock pre-divider divides the 100 MHz input reference clock by two. The resulting 50 MHz clock is fed to the transmitter PLL. Since the transmitter PLL implements a half-rate VCO, it multiplies the 50 MHz input clock by 25 to generate a 1250 MHz high-speed serial clock. This high-speed serial clock feeds the central clock divider and four local clock dividers of the transceiver block.

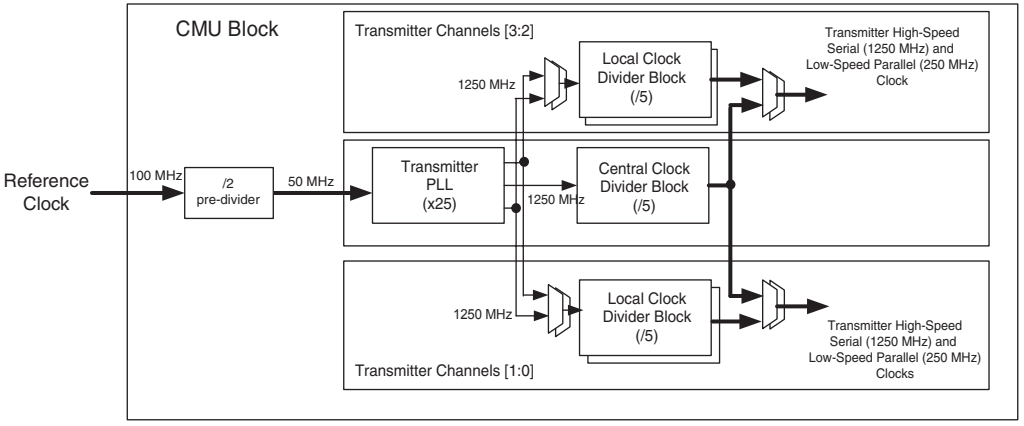
In ×4 PCI Express (PIPE) mode, the central clock divider in the transceiver block divides the 1250 MHz clock from the transmitter PLL by 5 to generate a 250 MHz parallel clock. This low-speed parallel clock output from the central clock divider block is used to clock the transmitter digital logic (PCS) in all channels of the transceiver block. The central clock divider block also forwards the high-speed serial clock from the transmitter PLL to the serializer within each channel. Since all four channels in the transceiver block are clocked with the same clock, the channel-to-channel skew is minimized.

In ×1 PCI Express (PIPE) mode, the local clock divider in each channel of the transceiver block divides the 1250 MHz clock from the transmitter PLL by 5 to generate a 250 MHz parallel clock. This low-speed parallel clock output from the local clock divider block is used to clock the transmitter digital logic (PCS) of the associated channel. The local clock divider block also forwards the high-speed serial clock from the transmitter PLL to the serializer within its associated channel.

The Quartus II software automatically selects the appropriate transmitter PLL bandwidth suited for the PCI Express (PIPE) data rate.

Figure 2–3 shows the CMU implemented in PCI Express (PIPE) mode.

Figure 2–3. PCI Express (PIPE) Mode CMU



Transmitter Phase Compensation FIFO Buffer

The transmitter phase compensation FIFO buffer compensates for the phase difference between the PLD clock that clocks in parallel data into the transmitter and the PCS clock that clocks the rest of the transmitter digital logic.

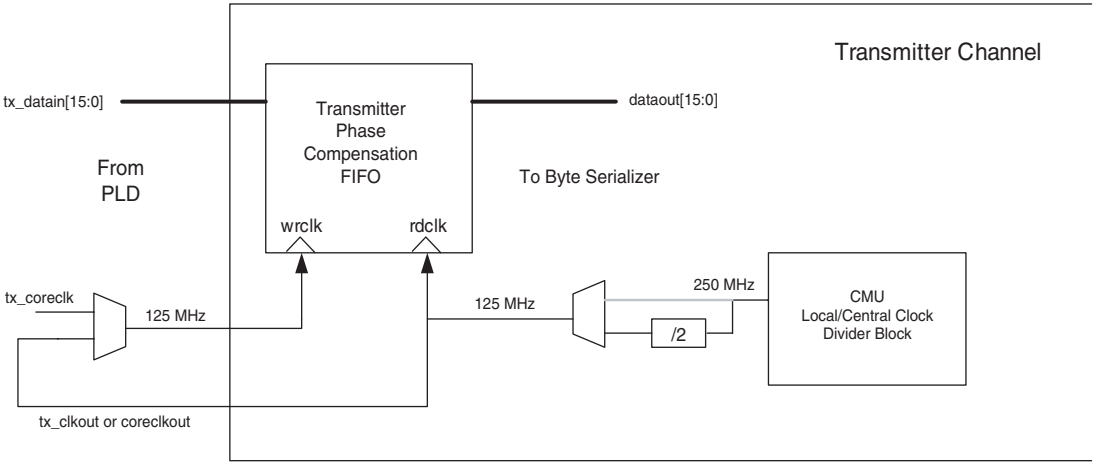
Refer to the Transmitter Phase Compensation FIFO Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about transmitter phase compensation FIFO buffer architecture.

In PCI Express (PIPE) mode, the 250 MHz clock generated by the CMU clock divider block is divided by 2. The resulting 125 MHz clock is used to clock the read port of the FIFO buffer. This 125 MHz clock is also forwarded to the PLD logic array (on the `tx_clkout` port in ×1 PCI Express (PIPE) mode or the `coreclkout` port in ×4 PCI Express (PIPE) mode). If the `tx_coreclk` port is not instantiated, the clock signal on the `tx_clkout` port of channel 0 is routed back to clock the write side of the transmitter phase compensation FIFO buffer in all channels with the transceiver block. The 16-bit PLD-transceiver interface clocked at 125 MHz results in an effective PCI Express (PIPE) data rate of 2 Gbps.

In PCI Express (PIPE) mode, the transmitter phase compensation FIFO is eight words deep. The latency through the FIFO is three to four PLD-transceiver interface clock cycles.

Figure 2–4 shows the block diagram of transmitter phase compensation FIFO in PCI Express (PIPE) mode.

Figure 2–4. TX Phase Compensation FIFO in PCI Express (PIPE) Mode



Byte Serializer

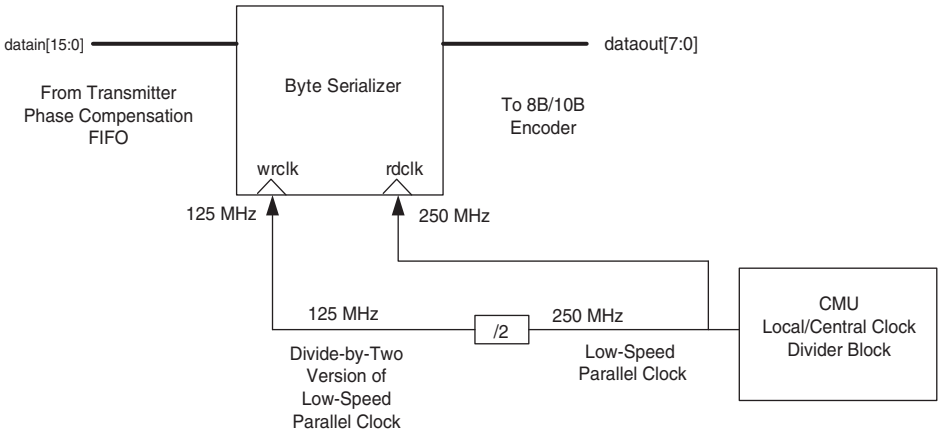
In PCI Express (PIPE) mode, the PLD-transceiver interface data is 16-bits wide and is clocked into the transmitter phase compensation FIFO at 125 MHz. The byte serializer clocks in the 16-bit wide data from the transmitter phase compensation FIFO at 125 MHz and clocks out 8-bit data to the 8B/10B encoder at 250 MHz. This allows clocking the PLD-transceiver interface at half the speed.

Refer to the Byte Serializer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about byte serializer architecture.

The write port of the byte serializer is clocked by the divide-by-two version of the low-speed parallel clock from the CMU. The read port is clocked by the low-speed parallel clock from the CMU. The byte serializer clocks out the least significant byte (LSByte) of the 16-bit data first and the most significant byte (MSByte) last.

Figure 2–5 shows the block diagram of the byte serializer in PCI Express (PIPE) mode.

Figure 2–5. Byte Serializer in PCI Express (PIPE) Mode



8B/10B Encoder

In PCI Express (PIPE) mode, the 8B/10B encoder clocks in 8-bit data and 1-bit control identifier from the byte serializer and generates 10-bit encoded data. The 10-bit encoded data is fed to the serializer.

Refer to the 8B/10B Encoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about the 8B/10B encoder functionality.

Compliance Pattern Transmission Support

PCI Express has an option to transmit a compliance pattern for testing purposes. The compliance pattern must be transmitted beginning with a negative disparity. In PCI Express (PIPE) mode, you set the negative disparity with the `tx_forcedispcompliance` port.

Asserting the `tx_forcedispcompliance` port sets the least significant byte of the 16-bit PLD-transmitter interface data to be encoded with a negative disparity. The `tx_forcedispcompliance` port must be de-asserted after the first word of the compliance pattern is clocked into the transceiver.

The compliance pattern generator is not part of the Arria GX transceiver and must be designed using the PLD logic. This feature only allows you to begin the compliance pattern with a negative disparity.

Serializer

In PCI Express (PIPE) mode, the 10-bit encoded data from the 8B/10B encoder is clocked into the 10:1 serializer with the low-speed parallel clock at 250 MHz. The 10-bit data is clocked out of the serializer LSB to MSB at both edges of the high-speed serial clock at 1250 MHz. The resulting 2.5 Gbps serial data output of the serializer is fed into the transmitter output buffer.

Refer to the Serializer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about the serializer architecture.

Transmitter Buffer

Table 2–3 shows the transmitter buffer settings when configured in PCI Express (PIPE) mode.

Table 2–3. Transmitter Buffer Settings in PCI Express (PIPE) Mode

Settings	Value
I/O Standard	1.2-V PCML *(2)*
Differential Output Voltage (V_{OD})	800 mV *(1)*
Common Mode Voltage (V_{CM})	600 mV *(1)*
Differential Termination	100 Ω *(2)*
Transmitter Pre-Emphasis	Enabled *(3)*
V_{CCH} (Transmitter Buffer Power)	1.2 V

Notes to Table 2–3:

(1) The differential output voltage (Vod) and common mode voltage (Vcm) are fixed in the MegaWizard® Plug-In Manager and cannot be changed.

(2) The I/O standard and differential termination settings are defaulted to 1.2-V PCML and 100 Ω, respectively. If you select any other setting for the I/O standard or differential termination in the Assignment Editor, the Quartus II compiler will issue an error message.

(3) The transmitter pre-emphasis setting of 49% is selected to meet the PCI Express de-emphasis specification.

Transmitter Electrical Idle

In PCI Express (PIPE) mode, you can force the transmitter into electrical idle condition during P0 and P2 power state by asserting the `tx_forceelecidle` signal high. In electrical idle state, the transmitter buffer is tri-stated. The `tx_forceelecidle` signal must always be asserted high in P0 and P1 power states. Refer to "Power State Management" on page 2–22 for more details about PCI Express (PIPE) mode power states.

Receiver Detect

PCI Express Base Specification requires the transmitter to be capable of detecting a far-end receiver before beginning link training. Arria GX transceivers have dedicated receiver detect circuitry that gets activated in PCI Express (PIPE) mode.

The receiver detect circuitry is only available in the P1 power state and is set through the `tx_detectrxloopback` port. Refer to "Power State Management" on page 2–22 for more details about PCI Express (PIPE) mode power states.

In P1 power state, the transmitter output buffer is tri-stated since the transmitter is in electrical idle. A high on the `tx_detectrxloopback` port triggers the receiver detect circuitry to alter the transmitter buffer common mode voltage. The sudden change in common mode voltage effectively appears as a step voltage at the tri-stated transmitter buffer output. If a receiver (that complies with PCI Express input impedance requirements) is present at the far end, the time constant of the step voltage is higher. If a receiver is not present or powered down, the time constant of the step voltage is lower. The receiver detect circuitry snoops the transmitter buffer output for the time constant of the step voltage to detect the presence of the receiver at the far end.

A high pulse is driven on the `pipephydonestatus` port and 3'b011 is driven on the `pipestatus` port (refer to "Receiver Status" on page 2–21) to indicate that a receiver has been detected. There is some latency after asserting the `tx_detectrxloopback` signal, before the receiver detection is indicated on the `pipephydonestatus` port.

The `tx_forceelecidle` port must be asserted at least 10 parallel clock cycles prior to the `tx_detectrxloopback` port to ensure that the transmitter buffer is tri-stated.

Beacon Transmission

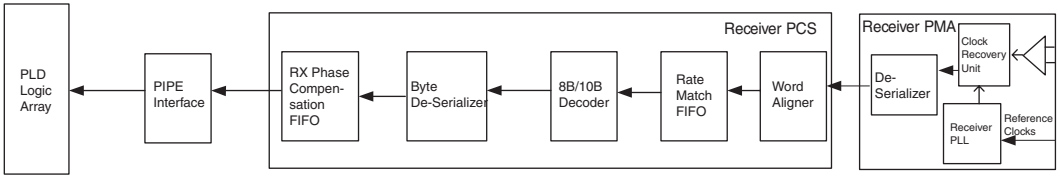
The beacon is an optional 30 kHz to 500 MHz in-band signal that wakes the receiver from a P2 power state. This signal is optional; the Arria GX device does not have dedicated beacon transmission circuitry. The Arria GX device supports the transmission of the beacon signal through

a 10-bit encoded code group that has a five 1's pulse (for example, K28.5) (10'b0101111100). Because the beacon signal is a pulse that ranges from 2 ns to 500 ns, sending out a K28.5 at 2.5 Gbps meets the lower requirement with its five 1's pulse. (Though other 8B/10B code groups might meet the beacon requirement, this document uses the K28.5 control code group as the beacon signal.) The beacon transmission takes place only in the P2 power state. The `tx_forceelecidle` port controls when the transmitter is in Electrical Idle or not. This port must be de-asserted in order to transmit the K28.5 code group for beacon transmission.

PCI Express (PIPE) Mode Receiver Architecture

This section lists sub-blocks within the receiver channel configured in PCI Express (PIPE) mode (Figure 2–6). The sub-blocks are described in order from the serial receiver input buffer to the receiver phase compensation FIFO buffer at the transceiver-PLD interface.

Figure 2–6. PCI Express (PIPE) Mode Receiver Architecture



Receiver Buffer

Table 2–4 shows the receiver buffer settings when configured in PCI Express (PIPE) mode.

Table 2–4. Receiver Buffer Settings in PCI Express (PIPE) Mode (Part 1 of 2)

Settings	Value
I/O Standard	1.2-V PCML, 1.5-V PCML, 3.3-V PCML, Differential LVPECL, LVDS
Input Common Mode Voltage (Rx V_{CM})	850 mV *(1)*
Differential Termination	100 Ω *(2)*

Table 2–4. Receiver Buffer Settings in PCI Express (PIPE) Mode (Part 2 of 2)	
Settings	**Value**
Coupling	AC

Notes to Table 2–4:
(1) The common mode voltage (Rx V_{CM}) is fixed in the MegaWizard Plug-In Manager and cannot be changed.
(2) The differential termination setting is defaulted to 100 Ω. If you select any other setting for differential termination in the Assignment Editor, the Quartus II compiler will issue an error message.

Signal Detect Threshold Circuitry
In PCI Express (PIPE) mode, the receiver buffer incorporates a signal detect threshold circuitry. The signal detect threshold circuitry senses if the specified threshold voltage level exists at the receiver buffer. This detector has a hysteresis response that filters out any high frequency ringing caused by inter symbol interference or high frequency losses in the transmission medium.

The `rx_signaldetect` signal indicates if the signal at the receiver buffer conforms to the signal detection settings. A high level on the `rx_signaldetect` port indicates that the signal conforms to the settings and a low level indicates that the signal does not conform to the settings. The Quartus II software automatically defaults to the appropriate signal detect threshold based on the PCI Express electrical idle specifications.

Receiver PLL and Clock Recovery Unit (CRU)

In PCI Express (PIPE) mode, the receiver PLL in each transceiver channel is fed by a 100 MHz input reference clock. The receiver PLL in conjunction with the clock recovery unit generates two clocks: a high-speed serial recovered clock at 1250 MHz (half-rate VCO) that feeds the deserializer, and a low-speed parallel recovered clock at 250 MHz that feeds the receiver's digital logic.

You can set the clock recovery unit in either automatic lock mode or manual lock mode. In automatic lock mode, the PPM detector and the phase detector within the receiver channel automatically switches the receiver PLL between lock-to-reference and lock-to-data modes. In manual lock mode, you can control the receiver PLL switch between lock-to-reference and lock-to-data modes via the `rx_locktorefclk` and `rx_locktodata` signals.

Refer to the Receiver PLL and Clock Recovery Unit (CRU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details on the CRU lock modes.

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

- One of the two available dedicated reference clock input pins (`REFCLK0` or `REFCLK1`) of the associated transceiver block
- Inter-transceiver block lines driven by reference clock input pins of other transceiver blocks

Deserializer

The 1:10 deserializer clocks in serial data from the receiver buffer using the high-speed recovered clock. The 10-bit de-serialized data is clocked out to the word aligner using the low-speed recovered clock at 250 MHz. The deserializer assumes that the transmission bit order is LSB to MSB; for example, the LSB of a data word is received earlier in time than its MSB.

Refer to the Deserializer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about the deserializer architecture.

Word Aligner

The word aligner clocks in the 10-bit data from the deserializer and restores the word boundary of the upstream transmitter. Besides restoring the word boundary, it also implements a synchronization state machine as specified in the PCI Express Base Specification to achieve lane synchronization.

Refer to the Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about the word aligner architecture.

In PCI Express (PIPE) mode, the word aligner comprises of the following three modules:

- Pattern detector module
- Pattern aligner module
- Run-length violation detector module

Pattern Detector

In PCI Express (PIPE) mode, the Quartus II software automatically configures 10-bit K28.5 (10'b0101111100) as the word alignment pattern. After coming out of reset (`rx_digitalreset`), when the pattern

detector detects either disparities of the K28.5 control word, it asserts the rx_patterndetect signal for one parallel clock cycle. Once the pattern aligner has aligned the incoming data to the desired word boundary, the pattern detector asserts the rx_patterndetect signal only if the word alignment pattern is found in the current word boundary.

Pattern Aligner

In PCI Express (PIPE) mode, the pattern aligner incorporates an automatic synchronization state machine. The Quartus II software automatically configures the synchronization state machine to indicate lane synchronization when the receiver receives four good /K28.5/ control code groups. Synchronization can be accomplished through the reception of four good PCI Express training sequences (TS1 or TS2) or four fast training sequences (FTS). Lane synchronization is indicated on the rx_syncstatus port of each channel. A high on the rx_syncstatus port indicates that the lane is synchronized and a low indicates that it has fallen out of synchronization.

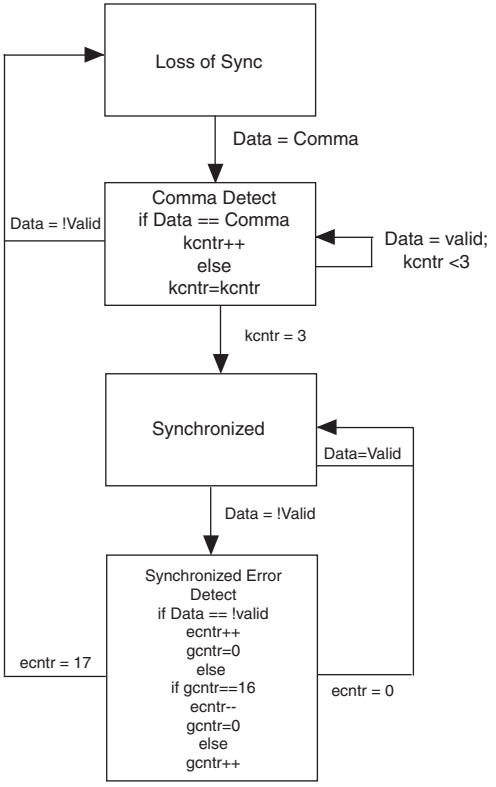
Table 2–5 lists the synchronization state machine parameters when configured in PCI Express (PIPE) mode.

Table 2–5. Synchronization State Machine Parameters in PCI Express (PIPE) Mode

Number of valid /K28.5/ code groups received to achieve synchronization (kcntr)	4
Number of errors received to lose synchronization (ecntr)	17
Number of continuous good code groups received to reduce the error count by 1 (gcntr)	16

Figure 2–7 shows a state diagram of the PCI Express (PIPE) synchronization.

Figure 2–7. PCI-Express (PIPE) Synchronization State Machine



Tables 2–6 and 2–7 list the TS1 and TS2 training sequences, respectively. A PCI Express fast training sequence consists of a /K28.5/, followed by three /K28.1/ code groups.

Table 2–6. PCI Express TS1 Ordered Set (Part 1 of 2)

Symbol Number	Allowed Values	Encoded Values	Description
0		K28.5	Comma code group for symbol alignment
1	0–255	D0.0–D31.7, and K23.7	Link number with component

Table 2–6. PCI Express TS1 Ordered Set (Part 2 of 2)

Symbol Number	Allowed Values	Encoded Values	Description
2	0–31	D0.0–D31.0, and K23.7	Lane number within port
3	0–255	D0.0–D31.7	N_FTS. The number of fast training ordered sets required by the receiver to obtain reliable bit and symbol lock.
4	2	D2.0	Data rate identifier Bit 0–Reserved, set to 0 Bit 1 = 1, generation 1 (2.5Gbps) data rate supported Bit 2..7–Reserved, set to 0
5	Bit 0 = 0, 1 Bit 1 = 0, 1 Bit 2 = 0, 1 Bit 3 = 0, 1 Bit 4..7 = 0	D0.0, D1.0, D2.0, D4.0, and D8.0	Training control **Bit 0 – Hot reset** Bit 0 = 0, de-assert Bit 0 = 1, assert **Bit 1 – Disable link** Bit 1 = 0, de-assert Bit 1 = 1, assert **Bit 1 – Loopback** Bit 2 = 0, de-assert Bit 2 = 1, assert **Bit 3 – Disable scrambling** Bit 3 = 0, de-assert Bit 3 = 1, assert **Bit 4..7 – Reserved** Bit 0 = 0, de-assert Set to 0
6–15		D10.2	TS1 identifier

Table 2–7. PCI Express TS2 Ordered Set

Symbol Number	Allowed Values	Encoded Values	Description
0		K28.5	Comma code group for symbol alignment.
1	0–255	D0.0–D31.7, and K23.7	Link number with component.
2	0–31	D0.0–D31.0, and K23.7	Lane number within port.
3	0–255	D0.0–D31.7	N_FTS. The number of fast training ordered sets required by the receiver to obtain reliable bit and symbol lock.
4	2	D2.0	Data rate identifier Bit 0–Reserved, set to 0 Bit 1 = 1, generation 1 (2.5Gbps) data rate supported Bit 2..7–Reserved, set to 0
5	Bit 0 = 0, 1 Bit 1 = 0, 1 Bit 2 = 0, 1 Bit 3 = 0, 1 Bit 4..7 = 0	D0.0, D1.0, D2.0, D4.0, and D8.0	Training control **Bit 0 – Hot reset** Bit 0 = 0, de-assert Bit 0 = 1, assert **Bit 1 – Disable link** Bit 1 = 0, de-assert Bit 1 = 1, assert **Bit 1 – Loopback** Bit 2 = 0, de-assert Bit 2 = 1, assert **Bit 3 – Disable scrambling** Bit 3 = 0, de-assert Bit 3 = 1, assert **Bit 4..7 – Reserved** Bit 0 = 0, de-assert Set to 0
6–15		D5.2	TS2 identifier

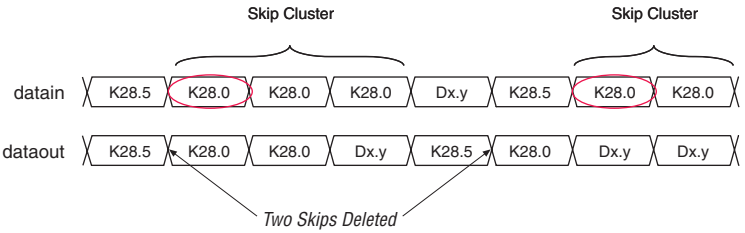
Rate Matcher

In PCI Express (PIPE) mode, the rate matcher can compensate up to ± 300 parts per million (PPM) (600 PPM total) frequency difference between the upstream transmitter and the receiver. In ×1 and ×4 PCI Express (PIPE) mode, the write port of the rate matcher FIFO in each receiver channel is clocked by its low-speed parallel recovered clock. In ×1 PCI Express (PIPE) mode, the read port is clocked by the low-speed parallel clock output of the CMU local clock divider block. In ×4 PCI Express (PIPE) mode, the read port is clocked by the low-speed parallel clock output of the CMU central clock divider block.

The rate matcher logic looks for skip ordered sets (SKP), which contains a /K28.5/ comma followed by three /K28.0/ skip characters. It deletes or inserts /K28.0/ skip characters as necessary from/to the rate matcher FIFO. The rate matcher can delete only one skip character in a consecutive cluster of skip characters and can insert only one skip character per skip cluster.

Figure 2–8 shows an example of a PCI Express (PIPE) mode rate matcher deletion of two skip characters.

Figure 2–8. PCI Express (PIPE) Mode Rate Matcher Deletion



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

8B/10B Decoder

In PCI Express (PIPE) mode, the 8B/10B decoder clocks in 10-bit data from the rate matcher and decodes it into 8-bit data + 1-bit control identifier. The 8-bit decoded data is fed to the byte deserializer.

Refer to the 8B/10B Decoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about the 8B/10B decoder functionality.

If the received 10-bit code is not a part of valid Dx.y or Kx.y code groups, the 8B/10B decoder block asserts an error flag on `rx_errdetect` port. The 8B/10B decoder replaces the invalid code group with /K30.7/ code (8'hFE + 1'b1 after decoding). The error flag signal (`rx_errdetect`) has the same data path delay from the 8B/10B decoder to the PLD-transceiver interface as the invalid code group.

If the received 10-bit code is detected with incorrect running disparity, the 8B/10B decoder block asserts an error flag on the `rx_disperr` and `rx_errdetect` ports. The error flag signal (`rx_disperr`) has the same delay from the 8B/10B decoder to the PLD-transceiver interface as the received data.

Polarity Inversion

The 8B/10B decoder supports the PCI Express (PIPE) compatible polarity inversion feature. This polarity inversion feature inverts the bits of the incoming data stream prior to the 8B/10B decoding block to fix accidental P-N polarity inversion on the differential input buffer. You use the `pipe8b10binvpolarity` port to invert the inputs to the 8B/10B decoder dynamically from the PLD.

You must not enable the receiver polarity inversion feature if you enable the PCI Express polarity inversion.

Byte Deserializer

In PCI Express (PIPE) mode, the PLD-receiver interface data is 16-bits wide and is clocked out of the receiver phase compensation FIFO at 125 MHz. The byte deserializer clocks in the 8-bit wide data from the 8B/10B decoder at 250 MHz and clocks out 16-bit wide data to the receiver phase compensation FIFO at 125 MHz. This allows clocking the PLD-transceiver interface at half the speed.

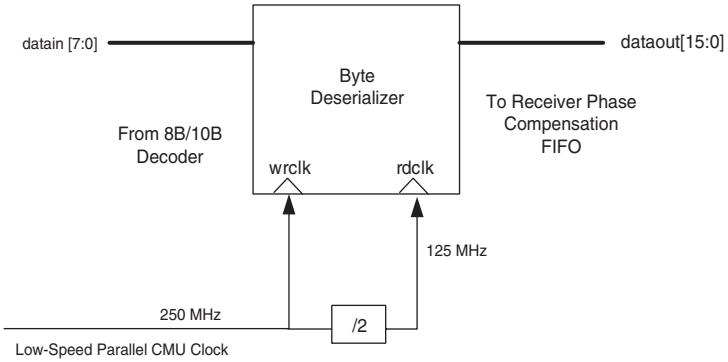
Refer to the Byte Deserializer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about byte deserializer architecture

In ×1 PCI Express (PIPE) mode, the write port of the byte deserializer is clocked by the low-speed parallel clock output from the CMU local clock divider block (`tx_clkout`) and the read port is clocked by divide-by-two version of this clock. In ×4 PCI Express (PIPE) mode, the write port of the byte deserializer is clocked by the low-speed parallel clock output from the CMU central clock divider block (`coreclkout`) and the read port is clocked by divide-by-two version of this clock.

Due to 8-bit to 16-bit byte deserialization, the byte ordering at the PLD-receiver interface might be incorrect. You implement the byte ordering logic in the PLD core to correct for this situation.

Figure 2–9 shows the block diagram of the byte serializer in PCI Express (PIPE) mode.

Figure 2–9. Byte Deserializer in PCI Express (PIPE) Mode



Receiver Phase Compensation FIFO Buffer

The receiver phase compensation FIFO buffer compensates for the phase difference between the local receiver PLD clock and the receiver PCS clock.

Refer to the Receiver Phase Compensation FIFO Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about receiver phase compensation FIFO buffer architecture.

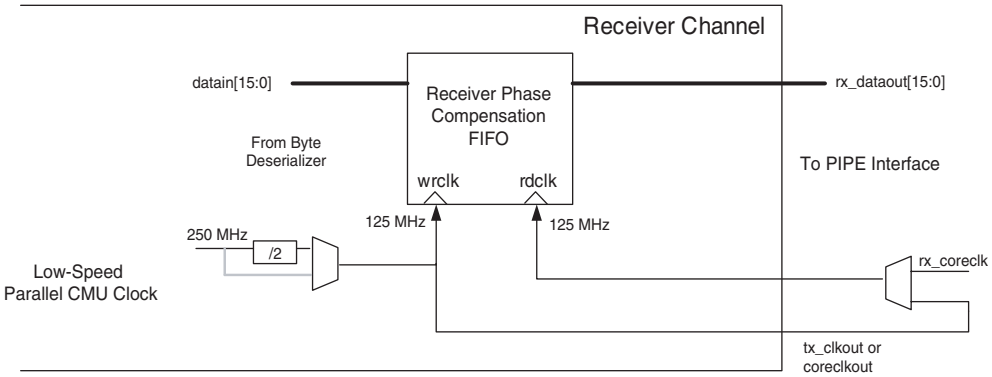
In PCI Express (PIPE) mode, the 250 MHz clock generated by the CMU clock divider block is divided by two. The resulting 125 MHz clock is used to clock the write port of the FIFO buffer. This 125 MHz clock is also forwarded to the PLD logic array (on the `tx_clkout` port in ×1 PCI Express (PIPE) mode or the `coreclkout` port in ×4 PCI Express (PIPE)

mode). If the rx_coreclk port is not instantiated, the clock signal on the tx_clkout/coreclkout port is routed back to clock the read side of the receiver phase compensation FIFO buffer. The 16-bit PLD-receiver interface, clocked at 125 MHz, results in an effective PCI Express (PIPE) data rate of 2 Gbps.

In PCI Express (PIPE) mode, the receiver phase compensation FIFO is eight words deep. The latency through the FIFO is two to three PLD-transceiver interface clock cycles.

Figure 2–10 shows the block diagram of transmitter phase compensation FIFO in PCI Express (PIPE) mode.

Figure 2–10. Receiver Phase Compensation FIFO in PCI Express (PIPE) Mode



Receiver Status

PCI Express (PIPE) specifies a receiver status indicator that reports the status of the PHY (PCS and PMA). In PCI Express (PIPE) mode, the receiver status is communicated to the PLD logic by the three-bit pipestatus port. This port reports the status, as shown in Table 2–8. If more than one event occurs at the same time, the signal is resolved with the higher priority status. The skip character added and removed flags (3'b001 and 3'b010) are not supported. The pipestatus port may be encoded to 3b'001 and 3'b010, which should be ignored. It does not indicate that a skip has been added or removed and should be considered the same as 3'b000—received data. If the upper MAC layer must know when a skip character was added or removed, Altera recommends monitoring the number of skip characters received. The transmitter should send three skip characters in a standard skip-ordered set.

Table 2–8. pipestatus Description and Priority

pipestatus	Description	Priority
3'b000	Received data	6
3'b001	One skip character added (not supported)	N/A
3'b010	One skip character removed (not supported)	N/A
3'b011	Receiver detected	1
3'b100	8B/10B decoder error	2
3'b101	Elastic buffer overflow	3
3'b110	Elastic buffer underflow	4
3'b111	Received disparity error	5

Power State Management

There are four supported power states in Arria GX transceivers when configured in PIPE mode: P0, P0s, P1, and P2. P0 is the normal power state. P0s is a low recovery time power state that is lower than P0. P1 is a lower power state than P0s and has higher latency to come out of this state. P2 is the lowest power state.

The `powerdn` port transitions the transceiver into different power states. The encoded value is shown in Table 2–9. The `pipephydonestatus` signal reacts to the `powerdn` request and pulses high for one parallel clock cycle.

There are specific functions that are performed at each of the power states. The power-down states are for PCI Express (PIPE) emulation. The transceiver does not go into actual power saving mode, with the exception of the transmitter buffer for Electrical Idle.

Table 2–9 shows each power state and its function.

Table 2–9. Power State Functions and Descriptions (Part 1 of 2)

Power State	powerdn	Function	Description
P0	2'b00	Transmits normal data, transmits Electrical Idle, or enters into loopback mode.	Normal operation mode
P0s	2'b01	Only transmits Electrical Idle.	Low recovery time power saving state

Table 2–9. Power State Functions and Descriptions (Part 2 of 2)

Power State	powerdn	Function	Description
P1	2'b10	Transmitter buffer is powered down and can do a receiver detect while in this state.	High recovery time power saving state
P2	2'b11	Transmits Electrical Idle or a beacon to wake up the downstream receiver.	Lowest power saving state

There are two signals associated with the power states: `tx_detectrxloopback` and `tx_forceelecidle`. The `tx_detectrxloopback` signal controls whether the channel goes into loopback when the power state is in P0 or receiver detect when in P1 state. This signal does not have any affect in any other power states. The `tx_forceelecidle` signal governs when the transmitter goes into an electrical idle state. The `tx_forceelecidle` signal is asserted in P0s and P1 states and de-asserted in P0 state. In P2 state, under normal conditions, the `tx_forceelecidle` signal is asserted and then de-asserted when the beacon signal must be sent out, signifying the intent to exit the P2 power-down state.

Table 2–10 shows the behavior of the `tx_detectrxloopback` and `tx_forceelecidle` signals in the power states.

Table 2–10. Power States and Functions Allowed in Each Power State

Power State	tx_detectrxloopback	tx_forceelecidle
P0	0: normal mode 1: data path in loopback mode	0: Must be de-asserted. 1: Illegal mode
P0s	Don't care	0: Illegal mode 1: Must be asserted in this state
P1	0: Electrical Idle 1: receiver detect	0: Illegal mode 1: Must be asserted in this state
P2	Don't care	De-asserted in this state for sending beacon. Otherwise asserted.

NFTS Fast Recovery IP (NFRI)

The PCI Express fast training sequences (FTS) are used for bit and byte synchronization to transition from P0s state to P0 state. The PCI Express standard specifies the required time period for this transition to be between 16 ns and 4 µs. The default PCI Express (PIPE) settings do not meet this requirement. You must enable the NFTS fast recovery IP (NFRI) for the receiver to transition from P0s to P0 within 4 µs by selecting the **Enable fast recovery mode** option in the MegaWizard Plug-In Manager.

PCI Express (PIPE) Mode Default Settings

In the PCI Express (PIPE) mode default settings (without NFRI enabled), the receiver PLL is in automatic lock mode. The PLL moves from lock-to-reference mode to lock-to-data mode based on the `rx_freqlocked` being asserted. For the `rx_freqlocked` signal to be asserted, the CRU clock should be within the PPM threshold settings of the receiver PLL reference clock. The PPM detector checks the PPM threshold settings by comparing the CRU PLL clock output with the reference clock for approximately 32768 clock cycles. For a 250 MHz PLD interface clock frequency, this comparison time period exceeds 4 µs, which violates the PCI Express specification.

The NFRI, if enabled, controls the `rx_locktorefclk` and `rx_locktodata` signals to meet the 4 µs transition time from P0s to P0 power state.

If you select the `rx_locktorefclk` and `rx_locktodata` signals in the MegaWizard Plug-In Manager (CRU Manual Lock mode), the **Enable fast recovery mode** option cannot be selected.

When you select the **Enable fast recovery mode** option, consider the following:

- NFRI is created in the PLD side for each PCI Express (PIPE) channel
- NFRI is a soft IP, so it consumes logic resources
- This block is self-contained, so no input/output ports are available to access the soft IP

Low-Latency (Synchronous) PCI Express (PIPE) Mode

The Arria GX receiver data path employs a rate match FIFO in PCI Express (PIPE) mode to compensate up to ±300 PPM difference between the upstream transmitter and the local receiver reference clock. The low-latency (synchronous) PCI Express (PIPE) mode allows bypassing the rate match FIFO in synchronous systems that derive the transmitter

and receiver reference clocks from the same source. You can bypass the rate match FIFO by not selecting the **Enable Rate Match FIFO** option in the ALT2GXB MegaWizard Plug-In Manager.

The rate match FIFO can be bypassed in both ×1 and ×4 PCI Express (PIPE) modes. In normal PCI Express (PIPE) mode, the receiver blocks following the rate match FIFO are clocked by `tx_clkout` (×1 mode) or `coreclkout` (×4 mode) of the local port. In low-latency (synchronous) PCI Express (PIPE) mode, since the rate match FIFO is bypassed, these receiver blocks are clocked by the recovered clocks of the respective channels.

Except for the rate match FIFO being bypassed and the resulting changes in transceiver internal clocking, the low-latency (synchronous) PCI Express (PIPE) mode shares the same data path and state machines as the normal PCI Express (PIPE) mode. However, some features supported in normal PCI Express (PIPE) mode are not supported in low-latency (synchronous) PCI Express (PIPE) mode.

PCI Express (PIPE) Reverse Parallel Loopback

In normal PCI Express (PIPE) mode, if the transceiver is in P0 power state, a high value on the `tx_rxdetectloop` signal forces a reverse parallel loopback, as discussed in "PCI Express (PIPE) Reverse Parallel Loopback" on page 2–57. Parallel data at the output of the receiver rate match FIFO gets looped back to the input of the transmitter serializer.

In low-latency (synchronous) PCI Express (PIPE) mode, since the rate match FIFO is bypassed, this feature is not supported. A high value on the `tx_rxdetectloop` signal when the transceiver is in P1 power state will not force it to perform reverse parallel loopback.

Link Width Negotiation

In normal ×4 PCI Express (PIPE) configuration, the receiver phase compensation FIFO control signals (write/read enable, etc.) are shared among all lanes within the link. As a result, all lanes are truly bonded and the lane-lane skew meets the PCI Express specification.

In low-latency (synchronous) PCI Express (PIPE) configuration, the receiver phase compensation FIFO of individual lanes do not share control signals. The write port of the receiver phase compensation FIFO of each lane is clocked by its recovered clock. As a result, the lanes within a link are not bonded. You should perform external lane de-skewing to ensure proper link width negotiation.

Receiver Status

Since the rate match FIFO is bypassed in low-latency (synchronous) PCI Express (PIPE) mode, status signal combinations related to the rate match FIFO on the `pipestatus[2:0]` port become irrelevant and must not be interpreted (Table 2–11).

Table 2–11. pipestatus Signal

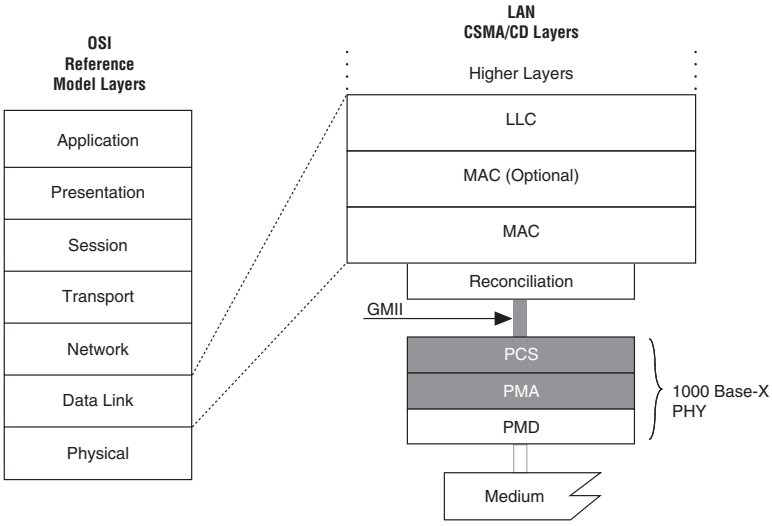
pipestatus[2:0]	Normal PIPE	Synchronous PIPE
000	Received Data OK	Received Data OK
001	Not supported	Not supported
010	Not supported	Not supported
011	Receiver Detected	Receiver Detected
100	8B/10B Decoder Error	8B/10B Decoder Error
101	Elastic Buffer Overflow	Not supported
110	Elastic Buffer Underflow	Not supported
111	Received Disparity Error	Received Disparity Error

Gigabit Ethernet (GIGE) mode

IEEE 802.3 defines the 1000 Base-X PHY as an intermediate, or transition, layer that interfaces various physical media with the media access control (MAC) in a gigabit ethernet system. It shields the MAC layer from the specific nature of the underlying medium. The 1000 Base-X PHY is divided into three sub-layers: the physical coding sublayer (PCS), the physical media attachment (PMA), and the physical medium dependent (PMD). The PCS sublayer interfaces to the MAC through the gigabit medium independent interface (GMII). The 1000 Base-X PHY defines a physical interface data rate of 1 Gbps.

Figure 2–11 shows the 1000 Base-X PHY position in a Gigabit Ethernet OSI reference model.

Figure 2–11. GIGE OSI Reference Model



Arria GX transceivers, when configured in GIGE functional mode, provide many of the PCS and PMA functions defined in the IEEE 802.3 specification, for example:

- 8B/10B encoding/decoding
- Synchronization
- Upstream transmitter and local receiver clock frequency compensation (rate matching)
- Clock recovery from the encoded data forwarded by the receiver PMD
- Serialization/deserialization

Arria GX transceivers do not have built-in support for other PCS functions; for example, auto-negotiation, collision-detect, and carrier-sense. If required, you must implement these functions in PLD logic array or external circuits.

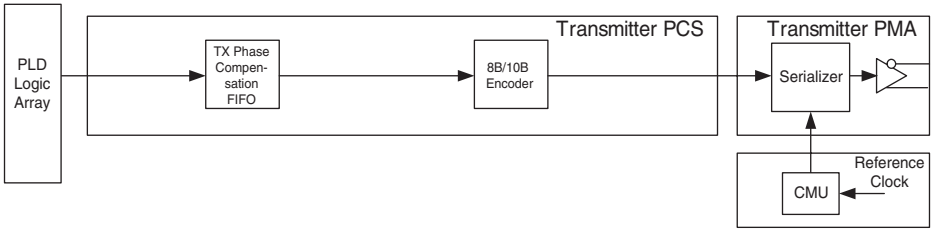
This section is organized into transmitter and receiver data path modules when configured for GIGE mode. The description for each module only covers details specific to GIGE functional mode support. Familiarity of IEEE 802.3 Ethernet specification is assumed.

For a general description of each module, refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

GIGE Mode Transmitter Architecture

This section lists sub-blocks within the transmitter channel configured in GIGE mode (Figure 2–12). The sub-blocks are described in order from the PLD-Transceiver parallel interface to the serial transmitter buffer.

Figure 2–12. GIGE Transmitter Architecture



Clock Multiplier Unit (CMU)

The clock multiplier unit takes in a reference clock and synthesizes the clocks that are used to clock the transmitter digital logic (PCS), the serializer, and the PLD-transceiver interface.

Refer to the Clock Multiplier Unit (CMU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about CMU architecture.

In GIGE mode, the CMU block consists of:

- Transmitter PLL that generates high-speed serial clock for the serializer
- Local clock divider block that generates low-speed parallel clock for transmitter digital logic and PLD-transceiver interface

Input Reference Clock
You can select either a 62.5 MHz or 125 MHz input reference clock frequency while configuring the transceiver in GIGE mode using the Quartus II MegaWizard Plug-In Manager.

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

- One of the two available dedicated reference clock input pins (REFCLK0 or REFCLK1) of the associated transceiver block
- PLD global clock network (must be driven directly from an input clock pin and cannot be driven by user logic or enhanced PLL)
- Inter-transceiver block lines driven by reference clock input pins of other transceiver blocks

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

The reference clock divide-by-two pre-divider is bypassed in GIGE mode.

Table 2–12 specifies the input reference clock options available in GIGE mode.

Table 2–12. GIGE Mode Input Reference Clock Specification

Frequency	I/O Standard	Coupling	Termination
62.5 MHz	1.2 V PCML, 1.5 V PCML, 3.3 V PCML, Differential LVPECL, LVDS	AC	On-chip
125 MHz			

Clock Synthesis

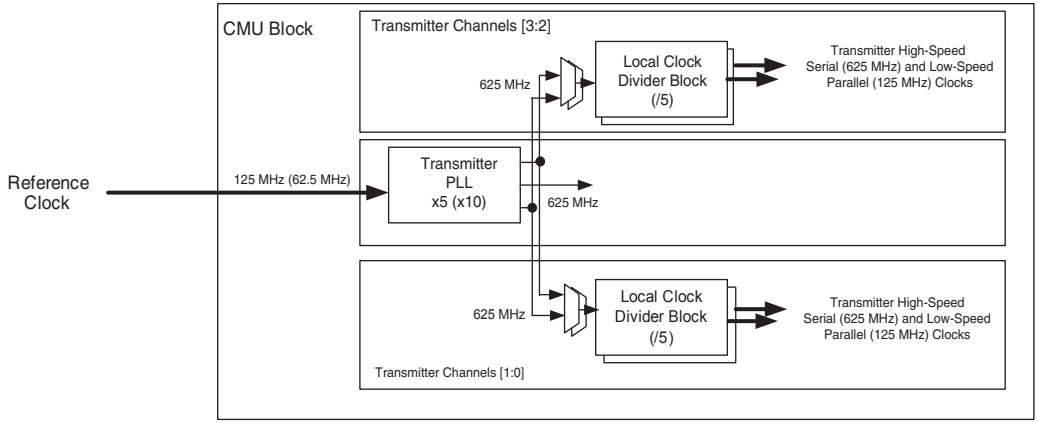
In GIGE mode, the input reference clock of 125 MHz (or 62.5 MHz) is fed to the transmitter PLL. Since the transmitter PLL implements a half-rate VCO, it multiplies the 125 MHz (or 62.5 MHz) input clock by 5 (or 10) to generate a 625 MHz high-speed serial clock. This high-speed serial clock feeds the local clock divider block in each GIGE channel instantiated within the transceiver block.

The local clock divider in each channel of the transceiver block divides the 625 MHz clock from the transmitter PLL by 5 to generate a 125 MHz parallel clock. This low-speed parallel clock output from the local clock divider block is used to clock the transmitter digital logic (PCS) of the associated channel. The local clock divider block also forwards the high-speed serial clock from the transmitter PLL to the serializer within its associated channel.

The Quartus II software automatically selects the appropriate transmitter PLL bandwidth suited for GIGE data rate.

Figure 2–13 shows the CMU implemented in GIGE mode.

Figure 2–13. GIGE Mode CMU



Transmitter Phase Compensation FIFO Buffer

The transmitter phase compensation FIFO buffer compensates for the phase difference between the PLD clock that clocks in parallel data into the transmitter and the PCS clock that clocks the rest of the transmitter digital logic.

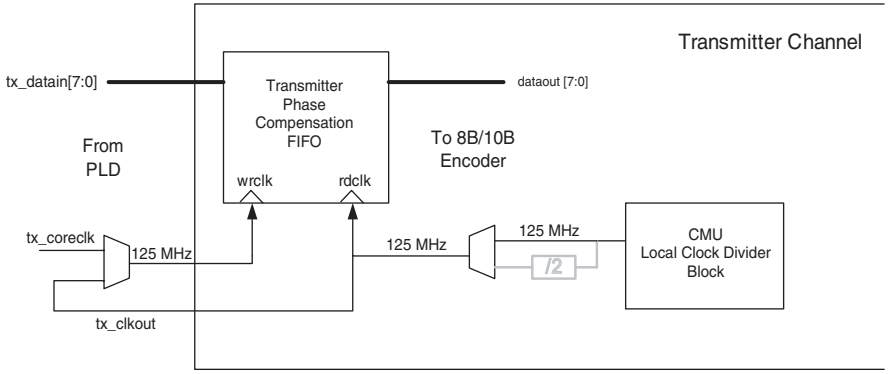
Refer to the Transmitter Phase Compensation FIFO Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about the transmitter phase compensation FIFO buffer architecture.

In GIGE mode, the 125 MHz clock generated by the CMU local clock divider is used to clock the read port of the FIFO buffer. This 125 MHz clock is also forwarded to the PLD logic array (on the `tx_clkout` port). If the `tx_coreclk` port is not instantiated, the clock signal on the `tx_clkout` port is automatically routed back to clock the write side of the transmitter phase compensation FIFO buffer. The 8-bit PLD-transceiver interface clocked at 125 MHz results into an effective GIGE data rate of 1 Gbps.

In GIGE mode, the transmitter phase compensation FIFO is four words deep. The latency through the FIFO is two to three PLD-transceiver interface clock cycles.

Figure 2–14 shows the block diagram of transmitter phase compensation FIFO in GIGE mode.

Figure 2–14. Transmitter Phase Compensation FIFO in GIGE Mode



8B/10B Encoder

In GIGE mode, the 8B/10B encoder clocks in 8-bit data and 1-bit control identifier from the transmitter phase compensation FIFO and generates a 10-bit encoded data. The 10-bit encoded data is fed to the serializer.

Refer to the *8B/10B Encoder* section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about the 8B/10B encoder functionality.

GIGE Protocol - Ordered Sets and Special Code Groups

Table 2–13 lists ordered sets and special code groups used in the GIGE functional mode.

Table 2–13. GIGE Ordered Sets (Part 1 of 2)

Code Group	Ordered Set	Number of Code Groups	Encoding
/C/	Configuration		Alternating /C1/ and /C2/
/C1/	Configuration 1	4	/K28.5/D21.5/`Config_Reg` *(1)*
/C2/	Configuration 2	4	/K28.5/D2.2/`Config_Reg` *(1)*

Table 2–13. GIGE Ordered Sets (Part 2 of 2)

Code Group	Ordered Set	Number of Code Groups	Encoding
/I/	IDLE		Correcting /I1/, Preserving /I2/
/I1/	IDLE 1	2	/K28.5/D5.6
/I2/	IDLE 2	2	/K28.5/D16.2
	Encapsulation		
/R/	Carrier_Extend	1	/K23.7/
/S/	Start_of_Packet	1	/K27.7/
/T/	End_of_Packet	1	/K29.7/
/V/	Error_Propagation	1	/K30.7/

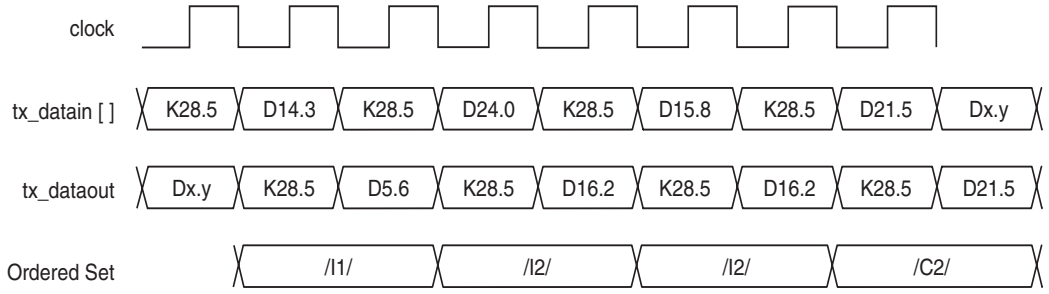
Note to Table 2–13:
(1) Two data code groups representing the Config_Reg value.

Idle Ordered-Set Generation
IEEE 802.3 requires the GIGE PHY to transmit idle ordered sets (/I/) continuously and repetitively whenever the GMII is idle. This ensures that the receiver maintains bit and word synchronization whenever there is no active data to be transmitted.

In GIGE functional mode, any /Dx.y/ following a /K28.5/ comma is replaced by the transmitter with either a /D5.6/ (/I1/ ordered set) or a /D16.2/ (/I2/ ordered set), depending on the current running disparity. The exception is when the data following the /K28.5/ is /D21.5/ (/C1/ ordered set) or /D2.2/ (/C2/) ordered set. If the running disparity before the /K28.5/ is positive, a /I1/ ordered set is generated. If the running disparity is negative, a /I2/ ordered set is generated. The disparity at the end of a /I1/ is the opposite of that at the beginning of the /I1/. The disparity at the end of a /I2/ is the same as the beginning running disparity (right before the idle code group). This ensures a negative running disparity at the end of an idle ordered set. A /Kx.y/ following a /K28.5/ is not replaced.

Figure 2–15 shows the automatic idle ordered set generation. Note that /D14.3/, /D24.0/, and /D15.8/ are replaced by /D5.6/ or /D16.2/ (for /I1/, /I2/ ordered sets). /D21.5/ (part of the /C1/ order set) is not replaced.

Figure 2–15. Idle Ordered Set Generation in GIGE Mode



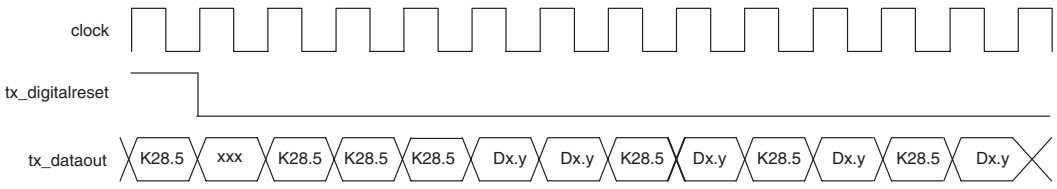
Reset Condition

After power up or reset, the GIGE transmitter outputs three /K28.5/ commas before user data can be sent. This affects the synchronization ordered set transmission.

After reset (tx_digitalreset), the 8B/10B encoder automatically sends three /K28.5/ commas. Depending on when you start outputting the synchronization sequence, there could be an even or odd number of /Dx.y/ sent as the transmitter before the synchronization sequence. The last of the three automatically sent /K28.5/and the first user-sent /Dx.y/ are treated as one idle ordered set. This can be a problem if there are an even number of /Dx.y/ transmitted before the start of the synchronization sequence.

Figure 2–16 shows an example of even numbers of /Dx.y/ between the last automatically sent /K28.5/ and the first user-sent /K28.5/. The first user-sent ordered set is ignored, so three additional ordered sets are required for proper synchronization. Figure 2–16 shows one don't care data between the tx_digitalreset signal going low and the first of three automatic K28.5, but there could be more.

Figure 2–16. GIGE Synchronization Ordered Set Considerations After Reset



Serializer

In GIGE mode, the 10-bit encoded data from the 8B/10B encoder is clocked into the 10:1 serializer with the low-speed parallel clock at 125 MHz. The 10-bit data is clocked out of the serializer LSB to MSB at the high-speed effective serial clock rate at 1250 MHz. The serial data output of the serializer is fed into the transmitter output buffer.

Refer to the Serializer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about the serializer architecture.

Transmitter Buffer

Table 2–14 shows the transmitter buffer settings when configured in GIGE mode.

Table 2–14. Transmitter Buffer Settings in GIGE Mode

Settings	Value
I/O Standard	1.5-V PCML *(2)*
Differential Output Voltage (V_{OD})	800 mV *(1)*
Common Mode Voltage (V_{CM})	600 mV *(1)*
Differential Termination	100 Ω *(2)*
V_{CCH} (Transmitter Buffer Power)	1.5 V

Notes to Table 2–14:

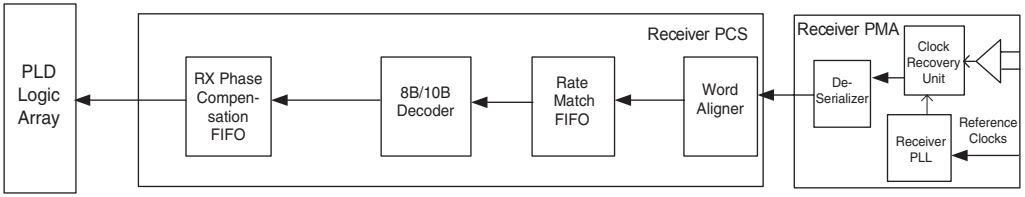
(1) The differential output voltage (Vod) and common mode voltage (Vcm) settings are fixed in the MegaWizard Plug-In Manager and cannot be changed.

(2) The I/O standard and differential termination settings are defaulted to 1.5-V PCML and 100 Ω, respectively. If you select any other setting for I/O standard or differential termination in the Assignment Editor, the Quartus II compiler will issue an error message.

GIGE Mode Receiver Architecture

This section lists sub-blocks within the receiver channel configured in GIGE mode (Figure 2–17). The sub-blocks are described in order from the serial receiver input buffer to the receiver phase compensation FIFO buffer at the transceiver-PLD interface.

Figure 2–17. GIGE Mode Receiver Architecture



Receiver Buffer

Table 2–15 shows the receiver buffer settings when configured in GIGE mode.

Table 2–15. Receiver Buffer Settings in GIGE Mode

Settings	Value
I/O Standard	1.2-V PCML, 1.5-V PCML, 3.3-V PCML, Differential LVPECL, LVDS
Input Common Mode Voltage (Rx V_{CM})	850 mV *(1)*
Differential Termination	100 Ω *(2)*
Coupling	AC

Notes to Table 2–15:

(1) The common mode voltage (Rx V_{CM}) is fixed in the MegaWizard Plug-In Manager and cannot be changed.

(2) The differential termination setting is defaulted to 100 Ω. If you select any other setting for differential termination in the Assignment Editor, the Quartus II compiler will issue an error message.

Receiver PLL and Clock Recovery Unit (CRU)

In GIGE mode, the receiver PLL in each transceiver channel is fed by a 125 MHz or a 62.5 MHz input reference clock. The receiver PLL in conjunction with the clock recovery unit (CRU) generates two clocks: a high-speed serial recovered clock at 625 MHz (half-rate PLL) that feeds the deserializer and a low-speed parallel recovered clock at 125 MHz that feeds the receiver's digital logic.

You can set the clock recovery unit in either automatic lock mode or manual lock mode. In automatic lock mode, the PPM detector and the phase detector within the receiver channel automatically switches the receiver PLL between lock-to-reference and lock-to-data modes. In

manual lock mode, you can control the receiver PLL switch between lock-to-reference and lock-to-data modes via the `rx_locktorefclk` and `rx_locktodata` signals.

Refer to the Receiver PLL and Clock Recovery Unit (CRU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about the CRU lock modes.

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

- One of the two available dedicated reference clock input pins (`REFCLK0` or `REFCLK1`) of the associated transceiver block
- PLD global clock network (must be driven directly from an input clock pin and cannot be driven by user logic or enhanced PLL)
- Inter-transceiver block lines driven by reference clock input pins of other transceiver blocks

Table 2–16 specifies the input reference clock options available in GIGE mode.

Table 2–16. GIGE Mode Input Reference Clock Specification

Frequency	I/O Standard	Coupling	Termination
125 MHz	1.2 V PCML, 1.5 V PCML, 3.3 V PCML, Differential LVPECL, LVDS	AC	On-chip
62.5 MHz			

Deserializer

The 1:10 deserializer clocks in serial data from the receiver buffer using the high-speed recovered clock. The 10-bit de-serialized data is clocked out to the word aligner using the low-speed recovered clock at 125 MHz. The deserializer assumes that the transmission bit order is LSB to MSB; for example, the LSB of a data word is received earlier in time than its MSB.

Refer to the Deserializer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about the deserializer architecture.

Word Aligner

The word aligner clocks in the 10-bit data from the deserializer and restores the word boundary of the upstream transmitter. Besides restoring the word boundary, it also implements a synchronization state machine as specified in the IEEE 802.3 specification to achieve receiver synchronization.

In GIGE mode, the word aligner comprises of the following three modules:

- Pattern detector module
- Pattern aligner module
- Run-length violation detector module

Pattern Detector
In GIGE mode, the Quartus II software automatically configures 10-bit K28.5 (10'b0101111100) as the word alignment pattern. After coming out of reset (`rx_digitalreset`), when the pattern detector detects either disparities of the K28.5 control word, it asserts the `rx_patterndetect` signal for one parallel clock cycle. Once the pattern aligner has aligned the incoming data to the desired word boundary, the pattern detector asserts the `rx_patterndetect` signal only if the word alignment pattern is found in the current word boundary.

Pattern Aligner
In GIGE mode, the pattern aligner incorporates an automatic synchronization state machine. The Quartus II software automatically configures the synchronization state machine to indicate synchronization when the receiver receives three consecutive synchronization ordered sets. An ordered set defined for synchronization is a /K28.5/ code group followed by an odd number of valid /Dx.y/ code groups. The fastest way for the receiver to achieve synchronization is to receive three continuous {/K28.5/, /Dx.y/} ordered set.

Receiver synchronization is indicated on the `rx_syncstatus` port of each channel. A high on the `rx_syncstatus` port indicates that the lane is synchronized and a low indicates that it has fallen out of synchronization. The receiver loses synchronization when it detects four invalid code groups separated by less than three valid code groups or when it is reset.

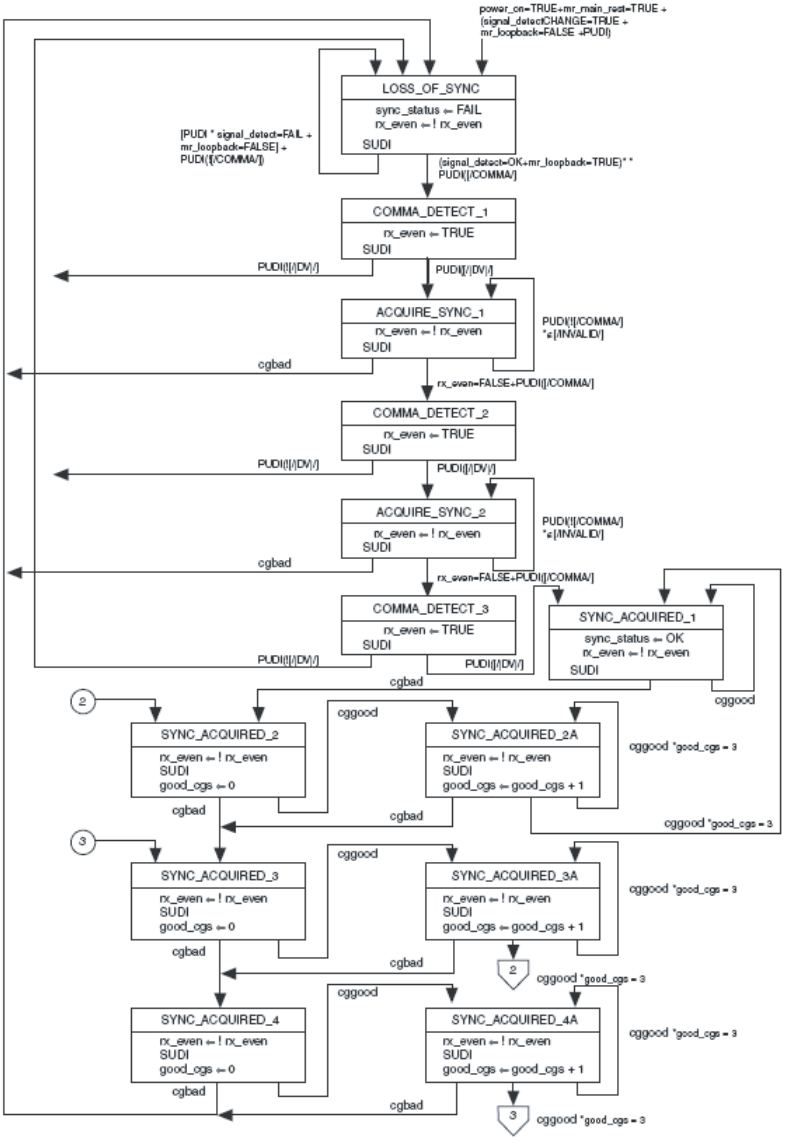
Table 2–17 lists the synchronization state machine parameters when configured in GIGE mode.

Table 2–17. Synchronization State Machine Parameters in GIGE Mode

Number of valid {/K28.5/, /Dx,y/} ordered-sets received to achieve synchronization	3
Number of errors received to lose synchronization	4
Number of continuous good code groups received to reduce the error count by 1	4

Figure 2–18 shows the synchronization state machine implemented in GIGE mode.

Figure 2–18. GIGE Synchronization State Machine



Rate Matcher

In GIGE mode, the rate matcher can compensate up to ±100 PPM (200 PPM total) frequency difference between the upstream transmitter and the receiver. The write port of the rate matcher FIFO in each receiver channel is clocked by its low-speed parallel recovered clock. The read port is clocked by the low-speed parallel clock output of the CMU local clock divider block.

The rate matcher logic inserts or deletes /I2/ idle ordered-sets to/from the rate matcher FIFO during the inter-frame or inter-packet gap (IFG or IPG). /I2/ is selected as the rate matching ordered-set since it maintains the running disparity unlike /I1/ that alters the running disparity. Since the /I2/ ordered-set contains two 10-bit code groups (/K28.5/, /D16.2/), twenty bits are inserted or deleted at a time for rate matching.

The rate matcher logic does not have capability to insert or delete /C1/ or /C2/ configuration ordered-sets.

Figure 2–19 shows an example of /I2/ deletion and Figure 2–20 shows an example of /I2/ insertion in a GIGE mode rate matcher.

Figure 2–19. GIGE Rate Matcher /I2/ Deletion

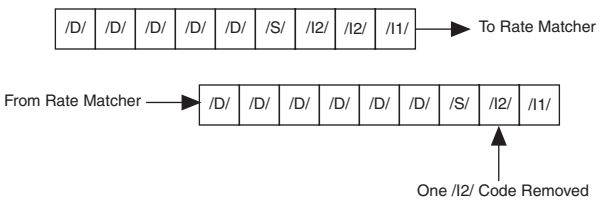
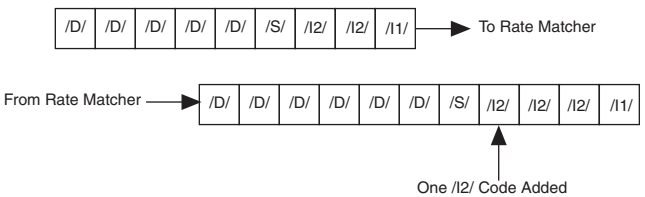


Figure 2–20. GIGE Rate Matcher /I2/ Insertion



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

8B/10B Decoder

In GIGE mode, the 8B/10B decoder clocks in 10-bit data from the rate matcher and decodes it into 8-bit data + 1-bit control identifier. The 10-bit decoded data is fed to the receiver phase compensation FIFO buffer.

Refer to the 8B/10B Decoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about the 8B/10B decoder functionality.

If the received 10-bit code group is not a part of valid Dx.y or Kx.y code groups, the 8B/10B decoder block asserts an error flag on the `rx_errdetect` port. The error flag signal (`rx_errdetect`) has the same data path delay from the 8B/10B decoder to the PLD-transceiver interface as the invalid code group.

If the received 10-bit code group is detected with incorrect running disparity, the 8B/10B decoder block asserts an error flag on the `rx_disperr` and `rx_errdetect` ports. The error flag signal (`rx_disperr`) has the same delay from the 8B/10B decoder to the PLD-transceiver interface as the received data.

Receiver Phase Compensation FIFO

The receiver phase compensation FIFO buffer compensates for the phase difference between the local receiver PLD clock and the receiver PCS clock.

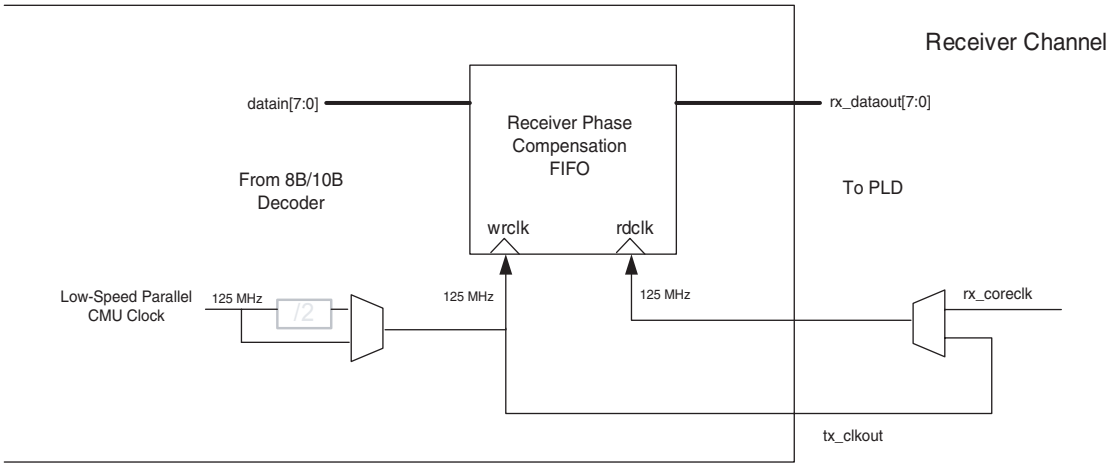
Refer to the ***Receiver Phase Compensation FIFO*** section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about the receiver phase compensation FIFO buffer architecture.

In GIGE mode, the 125 MHz clock generated by the CMU local clock divider block clocks the write port of the FIFO buffer. This 125 MHz clock is also forwarded to the PLD logic array (on the corresponding `tx_clkout` port). If the `rx_coreclk` port is not instantiated, the clock signal on the `tx_clkout` port is automatically routed back to clock the read side of the receiver phase compensation FIFO buffer. The 8-bit PLD-receiver interface clocked at 125 MHz results into an effective GIGE data rate of 1 Gbps.

In GIGE mode, the receiver phase compensation FIFO is four words deep. The latency through the FIFO is one to two PLD-transceiver interface clock cycles.

Figure 2–21 shows the block diagram of receiver phase compensation FIFO in GIGE mode.

Figure 2–21. Receiver Phase Compensation FIFO in GIGE Mode



Serial RapidIO Mode

The RapidIO™ standard is a high-performance, packet-switched interconnect technology designed to pass data and control information between microprocessors, digital signal, communications, and network processors, system memories, and peripheral devices. Serial RapidIO physical layer specification defines three line rates at 1.25 Gbps, 2.5 Gbps, and 3.125 Gbps. It also supports two link widths – single-lane (×1) and bonded four-lane (×4) at each line rate.

Arria GX transceivers support both single-lane (×1) and four-lane (×4) Serial RapidIO link widths at 1.25 Gbps and 2.5 Gbps line rates. In ×4 Serial RapidIO mode, the four transceiver channels are not bonded and are clocked independently like four individual channels.

Arria GX transceivers, when configured in Serial RapidIO functional mode, provide the following PCS and PMA functions:

- 8B/10B encoding/decoding
- Word alignment
- Lane Synchronization State Machine
- Clock recovery from the encoded data
- Serialization/deserialization

Arria GX transceivers do not have built-in support for other PCS functions; for example, clock frequency compensation between upstream transmitter clock and local receiver clock (rate matcher), idle sequence generation, and lane alignment in ×4 mode. Depending on your system requirements, you must implement these functions in the logic array or external circuits.

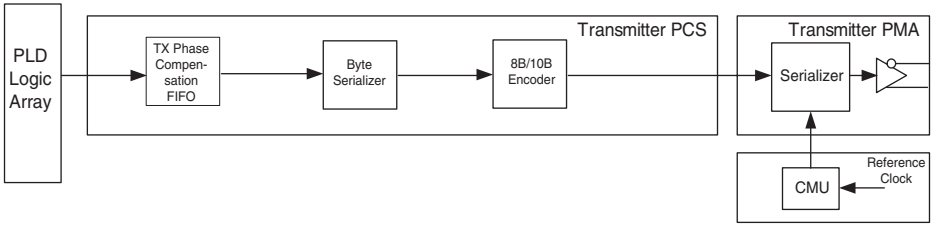
This section is organized into transmitter and receiver data path modules when configured for Serial RapidIO mode. The description for each module only covers details specific to Serial RapidIO functional mode support. Familiarity of RapidIO Interconnect Specification v1.3 is assumed.

For a general description of each module, refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*.

Serial RapidIO Mode Transmitter Architecture

This section lists sub-blocks within the transmitter channel configured in Serial RapidIO mode (Figure 2–22). The sub-blocks are described in order from the PLD-Transceiver parallel interface to the serial transmitter buffer.

Figure 2–22. Serial RapidIO Transmitter Architecture



Clock Multiplier Unit (CMU)

The clock multiplier unit takes in a reference clock and synthesizes the clocks that are used to clock the transmitter digital logic (PCS), the serializer, and the PLD-transceiver interface.

Refer to the Clock Multiplier Unit (CMU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about CMU architecture.

In Serial RapidIO mode, the CMU block consists of:

- Transmitter PLL that generates high-speed serial clock for the serializer
- Local clock divider block that generates low-speed parallel clock for transmitter digital logic and PLD-transceiver interface

Input Reference Clock

Table 2–18 lists the input reference clock frequencies allowed in Serial RapidIO mode.

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

- One of the two available dedicated reference clock input pins (REFCLK0 or REFCLK1) of the associated transceiver block
- PLD global clock network (must be driven directly from an input clock pin and cannot be driven by user logic or enhanced PLL)
- Inter-transceiver block lines driven by reference clock input pins of other transceiver blocks

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

The reference clock divide-by-two pre-divider is bypassed in serial RapidIO mode.

Table 2–18. Serial RapidIO Mode Input Reference Clock Specifications

Data Rate (Gbps)	Reference Clock Frequency (MHz)	I/O Standard	Coupling	Termination
1.25	62.5, 78.125, 125, 156.25	1.2V PCML, 1.5V PCML, 3.3V PCML, Differential LVPECL, LVDS	AC	On-chip
2.5	125, 156.25, 250, 312.5			

Clock Synthesis

In Serial RapidIO mode, the input reference clock is fed to the transmitter PLL. Since the transmitter PLL implements a half-rate VCO, it multiplies the input reference clock to generate a 625 MHz (1.25 Gbps Serial RapidIO) or 1250 MHz (2.5 Gbps Serial RapidIO) high-speed serial clock. This high-speed serial clock feeds the local clock divider block in each Serial RapidIO channel instantiated within the transceiver block. Table 2–19 lists the transmitter PLL multiplication factors that the Quartus II software automatically selects, depending on the Serial RapidIO data rate and input reference clock frequency selection.

Table 2–19. Serial RapidIO Mode Transmitter PLL Multiplication Factors

Data Rate (Gbps)	Reference Clock Frequency (MHz)	Transmitter PLL Multiplication Factor
1.25	62.5	10
	78.125	8
	125	5
	156.25	4
2.5	125	10
	156.25	8
	250	5
	312.5	4

In Serial RapidIO 1.25Gbps (2.5Gbps) mode, the local clock divider in each channel of the transceiver block divides the 625MHz (1250 MHz) clock from the transmitter PLL by 5 to generate a 125MHz (250 MHz) parallel clock. This low-speed parallel clock output from the local clock divider block is used to clock the transmitter digital logic (PCS) of the associated channel. The local clock divider block also forwards the high-speed serial clock from the transmitter PLL to the serializer within its associated channel.

The Quartus II software automatically selects the appropriate transmitter PLL bandwidth suited for Serial RapidIO data rate.

Transmitter Phase Compensation FIFO Buffer

The transmitter phase compensation FIFO buffer compensates for the phase difference between the PLD clock that clocks in parallel data into the transmitter and the PCS clock that clocks the rest of the transmitter digital logic.

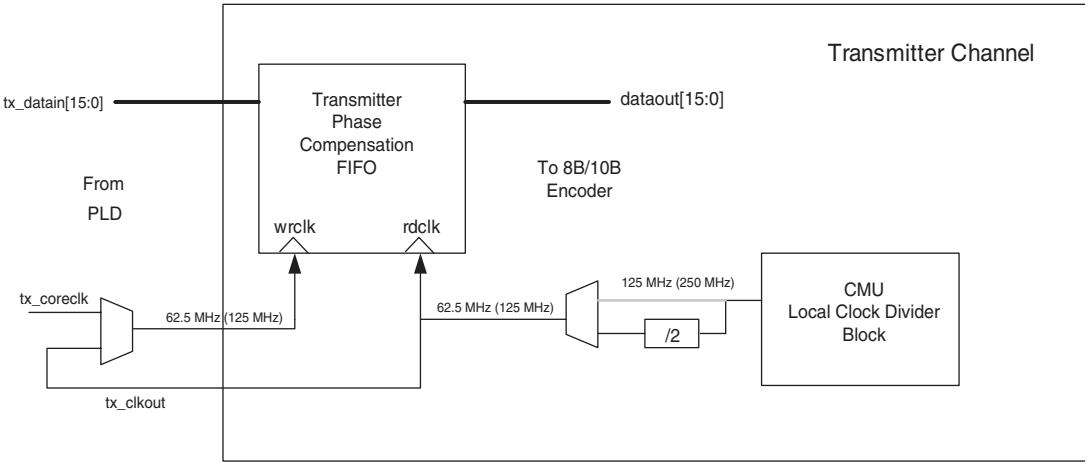
Refer to the Transmitter Phase Compensation FIFO Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about the transmitter phase compensation FIFO buffer architecture.

In Serial RapidIO 1.25 Gbps (2.5 Gbps) mode, the 125 MHz (250 MHz) clock generated by the CMU clock divider block is divided by 2. The resulting 62.5MHz (125MHz) clock is used to clock the read port of the FIFO buffer. This divide-by-two clock is also forwarded to the PLD logic array (on the `tx_clkout` port of its associated channel). If the `tx_coreclk` port is not instantiated, the clock signal on the `tx_clkout` port is automatically routed back to clock the write side of the transmitter phase compensation FIFO buffer. The 16-bit PLD-transceiver interface clocked at 62.5 MHz (125 MHz) results into an effective Serial RapidIO data rate of 1 Gbps (2 Gbps).

In Serial RapidIO mode, the transmitter phase compensation FIFO is four words deep. The latency through the FIFO is two to three PLD-transceiver interface clock cycles.

Figure 2–23 shows the block diagram of transmitter phase compensation FIFO in Serial RapidIO mode.

Figure 2–23. Transmitter Phase Compensation FIFO in Serial RapidIO Mode *Note (1)*



Note to Figure 2–23:

(1) The clock frequencies inside the parenthesis apply to 2.5 Gbps Serial RapidIO mode and the ones outside apply to 1.25 Gbps Serial RapidIO mode.

Byte Serializer

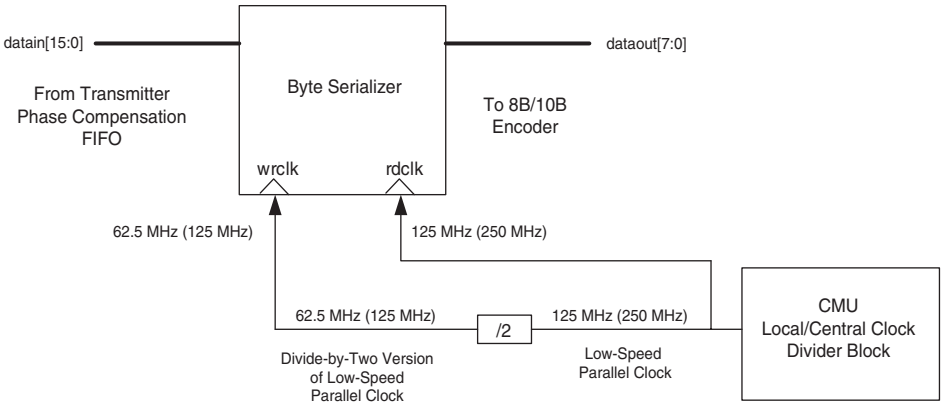
In Serial RapidIO 1.25 Gbps (2.5 Gbps) mode the PLD-transceiver interface data is 16 bits wide and is clocked into the transmitter phase compensation FIFO at 62.5 MHz (125 MHz). The byte serializer clocks in the 16-bit wide data from the transmitter phase compensation FIFO at 62.5 MHz (125 MHz) and clocks out 8-bit data to the 8B/10B encoder at 125 MHz (250 MHz). This allows clocking the PLD-transceiver interface at half the speed.

Refer to the Byte Serializer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about the byte serializer architecture.

The write port of the byte serializer is clocked by the divide-by-two version of the low-speed parallel clock from CMU. The read port is clocked by the low-speed parallel clock from CMU. The byte serializer clocks out the least significant byte of the 16-bit data first and the most significant byte last.

Figure 2–24 shows the block diagram of the byte serializer in Serial RapidIO mode.

Figure 2–24. Byte Serializer in Serial RapidIO Mode *Note (1)*



Note to Figure 2–24:

(1) The clock frequencies inside the parenthesis apply to 2.5 Gbps Serial RapidIO mode and the ones outside apply to 1.25 Gbps Serial RapidIO mode.

8B/10B Encoder

In Serial RapidIO mode, the 8B/10B encoder clocks in 8-bit data and 1-bit control identifier from the transmitter phase compensation FIFO and generates a 10-bit encoded data. The 10-bit encoded data is fed to the serializer.

Refer to the 8B/10B Encoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about the 8B/10B encoder functionality.

Serializer

In Serial RapidIO 1.25 Gbps (2.5 Gbps) mode, the 10-bit encoded data from the 8B/10B encoder is clocked into the 10:1 serializer with the low-speed parallel clock at 125 MHz (250 MHz). The 10-bit data is clocked out of the serializer LSB to MSB at the high-speed effective serial clock rate at 1250 MHz (2500 MHz). The serial data output of the serializer is fed into the transmitter output buffer.

Refer to the Serializer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*for more details about the serializer architecture.

Transmitter Buffer

Table 2–20 shows the transmitter buffer settings when configured in Serial RapidIO mode.

Table 2–20. Transmitter Buffer Settings in Serial RapidIO Mode

Settings	Value
I/O Standard	1.5-V PCML *(2)*
Differential Output Voltage (V_{OD})	800 mV *(1)*
Common Mode Voltage (V_{CM})	600 mV *(1)*
Differential Termination	100 Ω *(2)*
V_{CCH} (Transmitter Buffer Power)	1.5 V

Notes to Table 2–20:

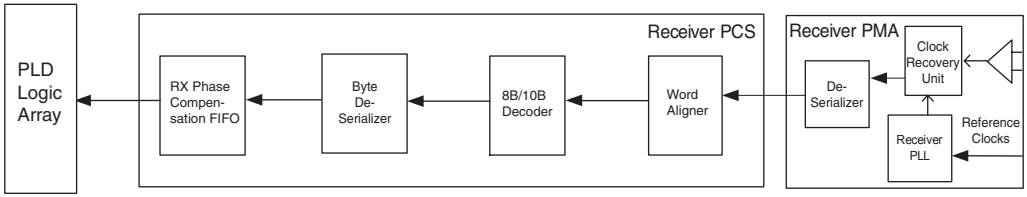
(1) The differential output voltage (Vod) and common mode voltage (Vcm) settings are fixed in the MegaWizard Plug-In Manager and cannot be changed.

(2) The I/O standard and differential termination settings are defaulted to 1.5-V PCML and 100 Ω, respectively. If you select any other setting for the I/O standard or differential termination in the Assignment Editor, the Quartus II compiler will issue an error message.

Serial RapidIO Mode Receiver Architecture

This section lists sub-blocks within the receiver channel configured in Serial RapidIO mode (Figure 2–25). The sub-blocks are described in order from the serial receiver input buffer to the receiver phase compensation FIFO buffer at the transceiver-PLD interface.

Figure 2–25. Serial RapidIO Mode Receiver Architecture



Receiver Buffer

Table 2–21 shows the receiver buffer settings when configured in Serial RapidIO mode.

Table 2–21. Receiver Buffer Settings in Serial RapidIO Mode

Settings	Value
I/O Standard	1.2-V PCML, 1.5-V PCML, 3.3-V PCML, Differential LVPECL, LVDS
Input Common Mode Voltage (Rx V_{CM})	850 mV *(1)*
Differential Termination	100 Ω *(2)*
Coupling	AC

Notes to Table 2–21:

(1) The common mode voltage (Rx V_{CM}) is fixed in the MegaWizard Plug-In Manager and cannot be changed.

(2) The differential termination setting is defaulted to 100 Ω. If you select any other setting for differential termination in the Assignment Editor, the Quartus II compiler will issue an error message.

Receiver PLL and Clock Recovery Unit (CRU)

In Serial RapidIO 1.25 Gbps (2.5 Gbps) mode, the receiver PLL in each transceiver channel is fed by an input reference clock. The receiver PLL in conjunction with the clock recovery unit generates two clocks: a half-rate

high-speed serial recovered clock at 625 MHz (1250 MHz) that feeds the deserializer and a low-speed parallel recovered clock at 125 MHz (250 MHz) that feeds the receiver's digital logic.

You can set the clock recovery unit in either automatic lock mode or manual lock mode. In automatic lock mode, the PPM detector and the phase detector within the receiver channel automatically switch the receiver PLL between lock-to-reference and lock-to-data modes. In manual lock mode, you can control the receiver PLL switch between lock-to-reference and lock-to-data modes via the `rx_locktorefclk` and `rx_locktodata` signals.

Refer to the Receiver PLL and Clock Recovery Unit (CRU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about the CRU lock modes.

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

- One of the two available dedicated reference clock input pins (`REFCLK0` or `REFCLK1`) of the associated transceiver block
- PLD global clock network (must be driven directly from an input clock pin and cannot be driven by user logic or enhanced PLL)
- Inter-transceiver block lines driven by reference clock input pins of other transceiver blocks

Table 2–22 specifies the receiver input reference clock options available in Serial RapidIO mode.

Table 2–22. Serial RapidIO Mode Input Reference Clock Specifications

Data Rate (Gbps)	Reference Clock Frequency (MHz)	I/O Standard	Coupling	Termination
1.25	62.5, 78.125, 125, 156.25	1.2 V PCML, 1.5 V PCML, 3.3 V PCML, Differential LVPECL, LVDS	AC	On-chip
2.5	125, 156.25, 250, 312.5			

Deserializer

In Serial RapidIO 1.25 Gbps (2.5 Gbps) mode, the 1:10 deserializer clocks in serial data from the receiver buffer using the high-speed serial recovered clock. The 10-bit de-serialized data is clocked out to the word aligner using the low-speed parallel recovered clock at 125 MHz

(250 MHz). The deserializer assumes that the transmission bit order is LSB to MSB; that is, the LSB of a data word is received earlier in time than its MSB.

Refer to the Deserializer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details on the deserializer architecture.

Word Aligner

The word aligner clocks in the 10-bit data from the deserializer and restores the word boundary of the upstream transmitter.

Refer to the Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about the word aligner architecture.

In Serial RapidIO mode, the word aligner comprises of the following three modules:

- Pattern detector module
- Pattern aligner module
- Run-length violation detection module

Pattern Detector
In Serial RapidIO mode, the Quartus II software automatically configures 10-bit K28.5 (10'b0101111100) as the word alignment pattern. After coming out of reset (`rx_digitalreset`), when the pattern detector detects either disparities of the K28.5 control word, it asserts the `rx_patterndetect` signal for one parallel clock cycle. Once the pattern aligner has aligned the incoming data to the desired word boundary, the pattern detector asserts `rx_patterndetect` signal only if the word alignment pattern is found in the current word boundary.

Pattern Aligner
In Serial RapidIO mode, the pattern aligner employs an automatic synchronization state machine. The Quartus II software automatically configures the synchronization state machine to indicate synchronization when the receiver receives 127 K28.5 (10'b0101111100 or 10'b1010000011) synchronization code groups without receiving an intermediate invalid code group. Once synchronized, the state machine indicates loss of synchronization when it detects three invalid code groups separated by less than 255 valid code groups or when it is reset.

Receiver synchronization is indicated on the rx_syncstatus port of each channel. A high on the rx_syncstatus port indicates that the lane is synchronized and a low indicates that it has fallen out of synchronization.

Table 2–23 lists the synchronization state machine parameters when configured in Serial RapidIO mode.

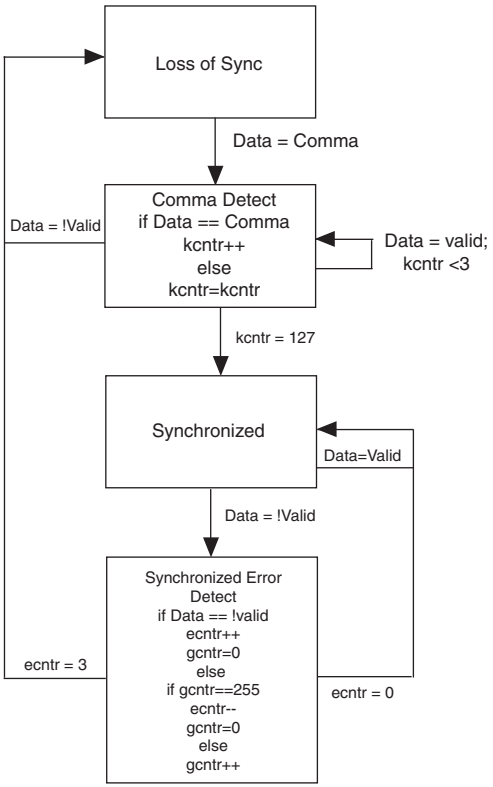
Table 2–23. Synchronization State Machine Parameters in Serial RapidIO Mode

Number of valid K28.5 code groups received to achieve synchronization	127
Number of errors received to lose synchronization	3
Number of continuous good code groups received to reduce the error count by 1	255

In an 8B/10B encoded data stream, a /K28.7/ special code group followed by any of the data code groups /D3.y/, /D11.y/, /D12.y/, /D19.y/, /D20.y/, /D28.y/ or /K28.y/ (where y ranges from 0 to 7), may cause the /K28.5/ alignment pattern to appear across the word boundary. Serial RapidIO protocol allows /K28.7/ transmission only during test and debug.

Figure 2–26 shows the synchronization state machine implemented in Serial RapidIO functional mode.

Figure 2–26. Synchronization State Machine in Serial RapidIO Mode



8B/10B Decoder

In Serial RapidIO mode, the 8B/10B decoder clocks in 10-bit data from the word aligner and decodes it into 8-bit data + 1-bit control identifier. The 8-bit decoded data is fed to the byte deserializer.

Refer to the 8B/10B Decoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about the 8B/10B decoder functionality.

If the received 10-bit code group is not a part of valid Dx.y or Kx.y code groups, the 8B/10B decoder block asserts an error flag on the rx_errdetect port. The error flag signal (rx_errdetect) has the same data path delay from the 8B/10B decoder to the PLD-transceiver interface as the invalid code group.

If the received 10-bit code group is detected with incorrect running disparity, the 8B/10B decoder block asserts an error flag on the rx_disperr and rx_errdetect ports. The error flag signal (rx_disperr) has the same delay from the 8B/10B decoder to the PLD-transceiver interface as the received data.

Byte Deserializer

In Serial RapidIO 1.25 Gbps (2.5 Gbps) mode, the PLD-receiver interface data is 16 bits wide and is clocked out of the receiver phase compensation FIFO at 62.5 MHz (125 MHz). The byte deserializer clocks in the 8-bit wide data from the 8B/10B decoder at 125 MHz (250 MHz) and clocks out 16-bit wide data to the receiver phase compensation FIFO at 62.5 MHz (125 MHz). This allows clocking the PLD-transceiver interface at half the speed.

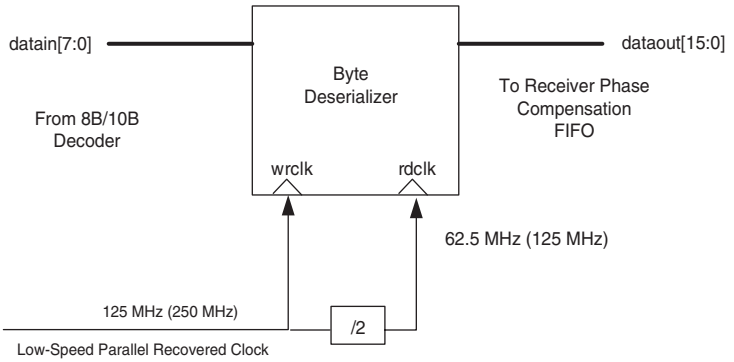
Refer to the Byte Deserializer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about byte deserializer architecture.

In Serial RapidIO mode, the write port of the byte deserializer is clocked by the low-speed parallel recovered clock and the read port is clocked by divide-by-two version of this clock.

Due to 8-bit to 16-bit byte deserialization, the byte ordering at the PLD-receiver interface might be incorrect. If required, you must implement the byte ordering logic in the PLD core to correct for this situation.

Figure 2–27 shows the block diagram of the byte de-serializer in Serial RapidIO mode.

Figure 2–27. Byte Deserializer in Serial RapidIO Mode *Note (1)*



Note to Figure 2–27:

(1) The clock frequencies inside the parenthesis apply to 2.5 Gbps Serial RapidIO mode and the ones outside apply to 1.25 Gbps Serial RapidIO mode.

Receiver Phase Compensation FIFO Buffer

The receiver phase compensation FIFO buffer compensates for the phase difference between the local receiver PLD clock and the receiver PCS clock.

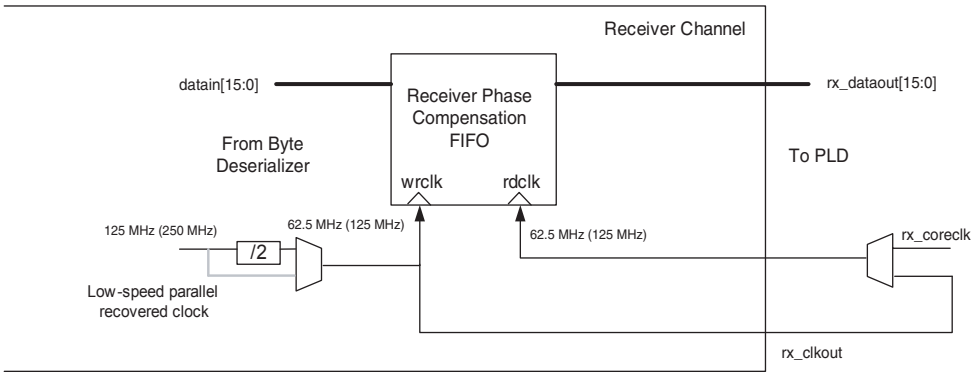
Refer to the Receiver Phase Compensation FIFO Buffer section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for more details about the receiver phase compensation FIFO buffer architecture.

In Serial RapidIO 1.25 Gbps (2.5 Gbps) mode, the 125 MHz (250 MHz) low-speed parallel recovered clock is divided by 2. The resulting 62.5 MHz (125 MHz) clock is used to clock the write port of the FIFO buffer. This divide-by-two clock is also forwarded to the PLD logic array (on the `rx_clkout` port). If the `rx_coreclk` port is not instantiated, the recovered clock signal on the `rx_clkout` port is automatically routed back to clock the read side of the receiver phase compensation FIFO buffer. The 16-bit PLD-receiver interface clocked at 62.5 MHz (125 MHz) results into an effective Serial RapidIO data rate of 1 Gbps (2 Gbps).

In Serial RapidIO mode, the receiver phase compensation FIFO is four words deep. The latency through the FIFO is one to two PLD-transceiver interface clock cycles.

Figure 2–28 shows the block diagram of receiver phase compensation FIFO in Serial RapidIO mode.

Figure 2–28. Receiver Phase Compensation FIFO in RapidIO Mode *Note (1)*



Note to Figure 2–28:

(1) The clock frequencies inside the parenthesis apply to 2.5 Gbps Serial RapidIO mode and the ones outside apply to 1.25 Gbps Serial RapidIO mode.

Loopback Modes

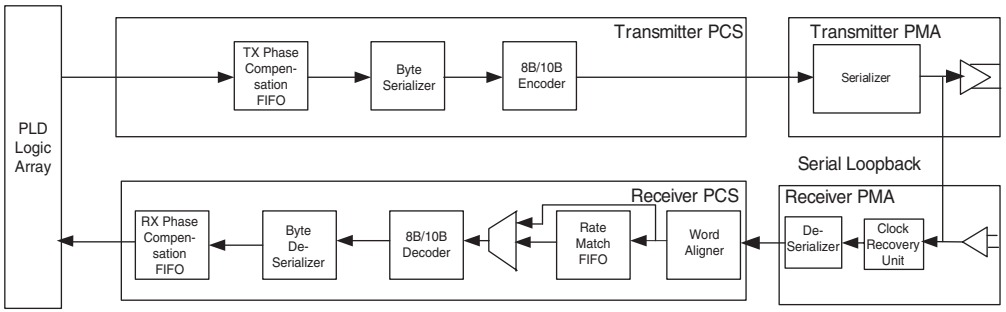
Arria GX transceivers support the following loopback configurations for diagnostic purposes:

- Serial Loopback - available in GIGE and Serial RapidIO modes
- PCI Express Reverse Parallel Loopback - available in PCI Express (PIPE) mode

Serial Loopback

Figure 2–29 shows the transceiver data path in serial loopback.

Figure 2–29. Transceiver Data Path in Serial Loopback Mode



In GIGE and Serial RapidIO modes, you can dynamically put each transceiver channel individually in serial loopback by controlling the `rx_seriallpbken` port. You instantiate the `rx_seriallpbken` port in the MegaWizard Plug-In Manager by selecting the serial loopback option. A high on the `rx_seriallpbken` port puts the transceiver into serial loopback and a low takes the transceiver out of serial loopback.

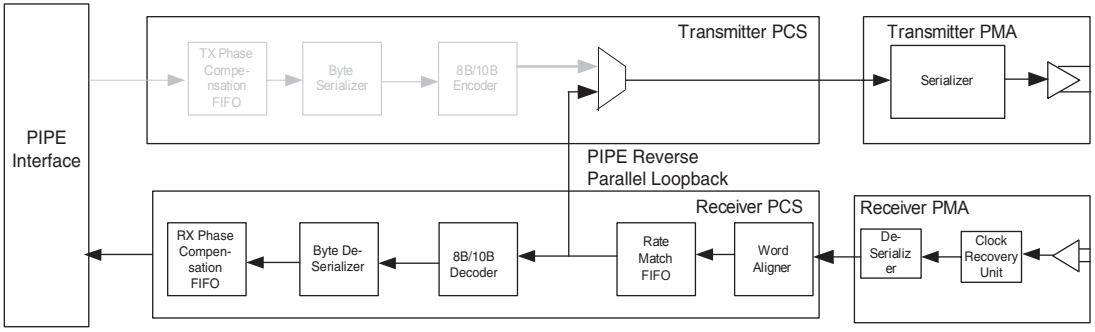
As seen in Figure 2–29, the serial data output from the transmitter serializer is looped back to the receiver clock recovery unit (CRU) in serial loopback. The transmitter data path from the PLD interface to the serializer in serial loopback is the same as in non-loopback mode. The receiver data path from the clock recovery unit to the PLD interface in serial loopback is the same as in non-loopback mode. Since the entire transceiver data path is available in serial loopback, this option is often used to diagnose the data path as a probable cause of link errors.

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

PCI Express (PIPE) Reverse Parallel Loopback

Figure 2–30 shows the data path for the PCI Express (PIPE) reverse parallel loopback. The reverse parallel loopback configuration is compliant with the PCI Express (PIPE) specification and is available only on PCI Express (PIPE) mode.

Figure 2–30. PCI Express (PIPE) Reverse Parallel Loopback



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

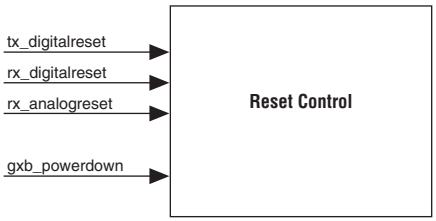
As seen in Figure 2–30, the serial data received on the `rx_datain` port in reverse parallel loopback goes through the CRU, deserializer, word aligner, and the rate matcher blocks. The parallel data at the output of the receiver rate matcher block is looped back to the input of the transmitter serializer block. The serializer converts the parallel data to serial data and feeds it to the transmitter output buffer that drives the data out on the `tx_dataout` port. The data at the output of the rate matcher also goes through the 8B/10B decoder, byte deserializer, and receiver phase compensation FIFO before being fed to the PLD on the `rx_dataout` port.

The reverse parallel loopback is not available in Low-latency (Synchronous) PCI Express (PIPE) mode as the rate matcher is bypassed.

Reset Control and Power-Down

Arria GX transceivers provide multiple reset signals to reset the analog and digital circuits in the transceiver channels. Besides individual channel resets, Arria GX transceivers also provide power-down signals that you can assert to power-down the entire transceiver block to reduce power consumption (Figure 2–31).

Figure 2–31. Reset Signals



User Reset and Power-Down Signals

Each transceiver block and each channel in the transceiver block of the Arria GX device has individual reset signals to reset the digital and analog circuits in the channel. The `tx_digitalreset`, `rx_digitalreset`, and `rx_analogreset` signals affect the channels individually. The `gxb_powerdown` signal affects the entire transceiver block.

All reset and power-down signals are optional. Altera strongly recommends using the reset and power-down signals and following the reset sequence discussed in this section.

- `tx_digitalreset`—The `tx_digitalreset` signal resets all digital logic in the transmitter. This signal operates independently from the other reset signals. The minimum pulse width is two parallel cycles.
- `rx_digitalreset`—The `rx_digitalreset` signal resets all digital logic in the receiver. This signal operates independently from the other reset signals. The minimum pulse width is two parallel cycles.
- `rx_analogreset`—The `rx_analogreset` signal resets part of the analog portion of the receiver CRU. This signal operates independently from the other reset signals. The minimum pulse width is two parallel cycles.
- `gxb_powerdown`—The `gxb_powerdown` signal powers down the entire transceiver block, including the transmitter PLL. All digital and analog circuits are also reset. This signal operates independently from the other reset signals. The minimum pulse width is 100 ns.

Table 2–24 lists the transceiver modules that get affected by each reset and power-down signal.

Table 2–24. Blocks Affected by Reset and Power-Down Signals

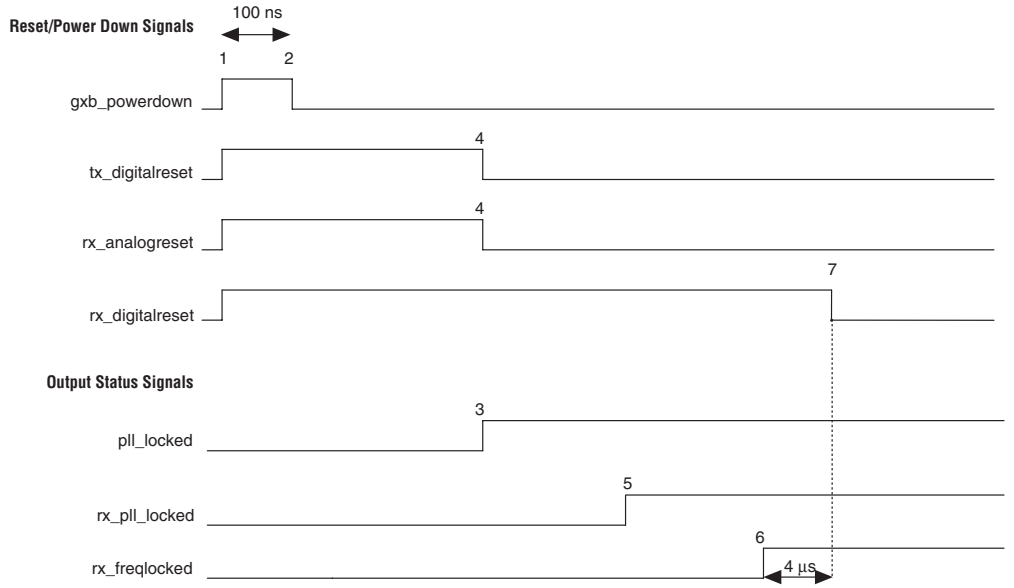
Transceiver Blocks	rx_digitalreset	rx_analogreset	tx_digitalreset	gxb_powerdown
Transmitter phase compensation FIFO buffer and byte serializer			✓	✓
Transmitter 8B/10B encoder			✓	✓
Transmitter serializer				✓
Transmitter analog circuits				✓
Transmitter PLLs				✓
Transmitter analog circuits				✓
Receiver deserializer				✓
Receiver word aligner	✓			✓
Receiver rate matcher	✓			✓
Receiver 8B/10B decoder	✓			✓
Receiver phase compensation FIFO buffer and byte deserializer	✓			✓
Receiver PLL and CRU		✓		✓
Receiver analog circuits				✓

The recommended reset sequence varies depending on whether the Clock Recovery Unit is configured in automatic lock mode or manual lock mode.

Recommended Reset Sequence for GIGE and Serial RapidIO in CRU Automatic Lock Mode

Figure 2–32 shows a sample reset sequence for GIGE and Serial RapidIO modes when the CRU is configured in automatic lock mode.

Figure 2–32. Reset Sequence for GIGE and Serial RapidIO in Automatic Mode

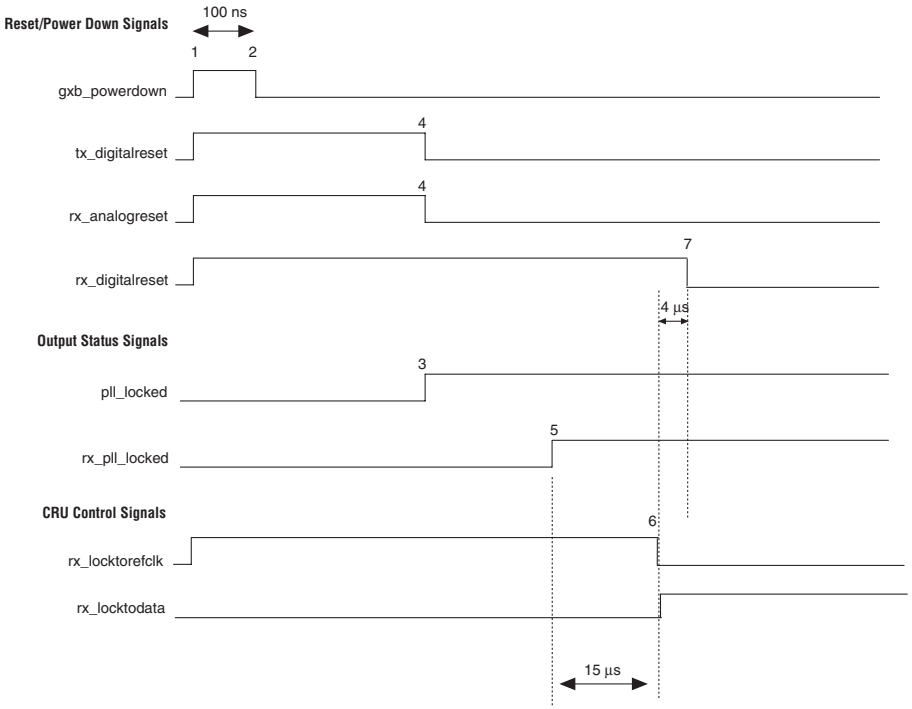


After power on, assert the gxb_powerdown port for a minimum period of 100 ns (time between markers 1 and 2). Keep the tx_digitalreset, rx_digitalreset, and rx_analogreset asserted during this time period. After you de-assert the gxb_powerdown signal, the transmitter PLL starts locking to the transmitter input reference clock. Once the transmitter PLL locks (as indicated by the pll_locked signal going high), you de-assert the tx_digitalreset signal. After you de-assert the rx_analogreset signal, the receiver PLL starts locking to the receiver input reference clock (in automatic lock mode). Once the receiver PLL locks to the input reference clock, the rx_pll_locked signal goes high. The internal PPM detector takes sometime to calculate the PPM difference between the receiver PLL output clock and the input reference clock. Once it calculates the PPM difference to be within the pre-defined limits, the rx_freqlocked signal goes high. At this point the CRU enters lock-to-data mode and the receiver PLL starts locking to the received data. You de-assert the rx_digitalreset 4 µs after the rx_freqlocked signal goes high.

Recommended Reset Sequence for GIGE and Serial RapidIO in CRU Manual Lock Mode

Figure 2–33 shows a sample reset sequence for GIGE and Serial RapidIO modes when the CRU is configured in manual lock mode.

Figure 2–33. Reset Sequence for GIGE and Serial RapidIO in Manual Mode



After power on, assert the gxb_powerdown port for a minimum period of 100 ns (time between markers 1 and 2). Keep the tx_digitalreset, rx_digitalreset, rx_analogreset, and rx_locktorefclk signals asserted during this time period. After you de-assert the gxb_powerdown signal, the transmitter PLL starts locking to the transmitter input reference clock. Once the transmitter PLL locks (as indicated by the pll_locked signal going high), you de-assert the tx_digitalreset signal. After you de-assert the rx_analogreset signal, the receiver PLL starts locking to the receiver input reference clock since rx_locktorefclk is asserted. Wait for at least 15 µs (time between markers 5 and 6) after the rx_pll_locked signal goes high and then de-assert the rx_locktorefclk signal. At the same time assert the rx_locktodata signal. At this point the CRU enters lock-to-data mode

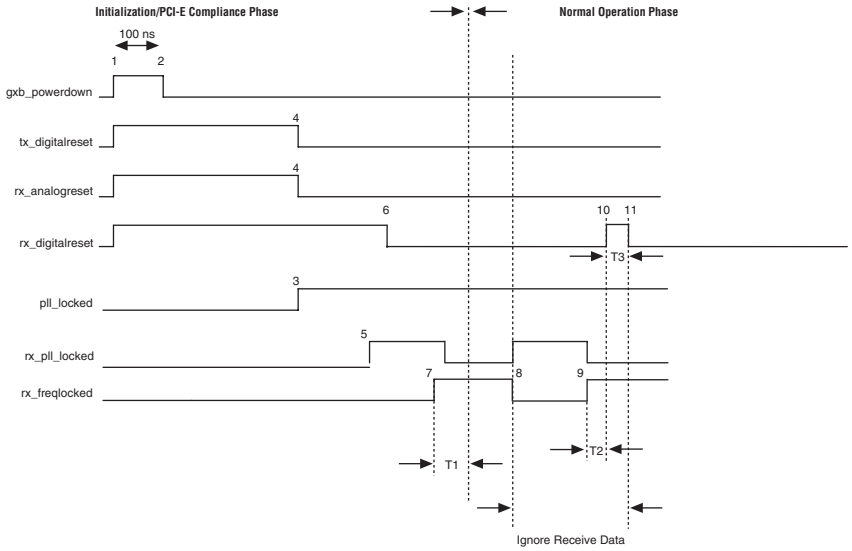
and the receiver PLL starts locking to the received data. You de-assert the rx_digitalreset at least 4 µs (time between markers 6 and 7) after asserting the rx_locktodata signal.

Recommended Reset Sequence for PCI Express (PIPE) Mode

The reset sequence used for GIGE and Serial RapidIO modes looks for the rx_freqlocked signal to de-assert rx_digitalreset. In PCI Express (PIPE) mode, the rx_freqlocked signal does not go high during the PCI Express (PIPE) compliance testing phase because of receiving Electrical Idle.

Figure 2–34 shows the reset sequence for PCI Express (PIPE) mode.

Figure 2–34. Reset Sequence for PCI Express (PIPE) Mode



Initialization and PCI Express Compliance Phase

After the device is powered up, a PCI Express compliant device may perform compliance testing. Since rx_digitalreset must be de-asserted during compliance testing, waiting for the rx_freqlocked signal to de-assert rx_digitalreset is not recommended.

You de-assert the tx_digitalreset signal after the pll_locked signal goes high. You de-assert the rx_digitalreset when the rx_pll_locked signal goes high (unlike GIGE and Serial RapidIO modes where you wait until rx_freqlocked goes high).

The parallel data sent to the PLD logic array in the receive side may not be valid until 4 µs after rx_freqlocked goes high.

Normal Operation Phase

During normal operation, the receive data is valid and the rx_freqlocked signal is high. In this situation, when rx_freqlocked is de-asserted, (marker 8 in Figure 2–34), you wait for the rx_freqlocked signal to go high again and assert rx_digitalreset (marker 10 in Figure 2–34) for two parallel receive clock cycles.

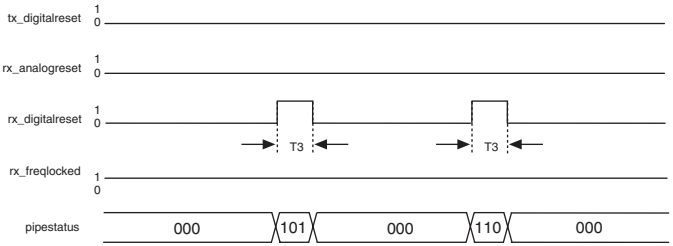
The data from the transceiver block is not valid between the time when rx_freqlocked goes low until rx_digitalreset is de-asserted. The PLD logic should ignore the data during this time period (the time period between markers 8 and 11 in Figure 2–34).

Minimum T1 and T2 period is 4 µs. Minimum T3 period is two parallel receive clock cycles.

Rate Matcher FIFO Buffer Overflow and Underflow Condition

During the normal operation phase, you monitor the overflow and underflow status of the rate matcher FIFO buffer. If there is overflow and underflow on the rate matcher FIFO buffer, you assert the rx_digitalreset signal for two receive parallel clock cycles. You can monitor the rate matcher FIFO buffer status through the pipestatus[2:0] signal from the PCI Express (PIPE) interface. This condition is shown in Figure 2–35.

Figure 2–35. PCI Express (PIPE) Mode Reset During Rate Matcher FIFO Buffer Overflow & Underflow Condition



Notes to Figure 2–35:

(1) Pipestatus = 101 represents elastic overflow (not available in Low-Latency [Synchronous] PCI Express [PIPE] mode).

(2) Pipestatus = 110 represents elastic overflow (not available in Low-Latency [Synchronous] PCI Express [PIPE] mode).

Power-Down

The Quartus II software automatically selects the power-down channel feature, which takes effect when you configure the Arria GX device. All unused transceiver channels and blocks in a design are powered down to reduce the overall power consumption.

The `gxb_powerdown` port is optional. In simulation, if the `gxb_powerdown` port is not instantiated, you must assert `tx_digitalreset`, `rx_digitalreset` and `rx_analogreset` signals appropriately for correct simulation behavior. If the `gxb_powerdown` port is instantiated and other reset signals are not used, you must assert the `gxb_powerdown` signal for at least one parallel clock cycle for correct simulation behavior. In simulation, you can de-assert the `rx_digitalreset` immediately after `rx_freqlocked` signal goes HIGH to reduce the simulation run time. It is not necessary to wait for 4 µs as suggested in the actual reset sequence.

In PCI Express (PIPE) mode simulation, you must assert the `tx_forceelecidle` signal for at least one parallel clock cycle before transmitting normal data for correct simulation behavior.

TimeQuest Timing Analyzer

Quartus II software designs targeted towards the Arria GX device family use TimeQuest Timing Analyzer for static timing analysis. In the Quartus II software versions 7.1 and 7.1 sp1, TimeQuest does not

automatically constrain the transceiver reset ports and asynchronous input/output ports. As a result, TimeQuest does not perform timing analysis on these paths.

TimeQuest reports these unconstrained paths in RED in the Timing Analyzer report. You must manually add the constraints in the Synopsys Design Constraints (**.sdc**) file for TimeQuest to analyze these paths.

Unconstrained Reset Ports

In the Quartus II software versions 7.1 and 7.1 sp1, TimeQuest does not constrain the following transceiver reset ports:

- `gxb_powerdown`
- `tx_digitalreset`
- `rx_digitalreset`
- `rx_analogreset`

Identifying Unconstrained Reset Ports
To identify the unconstrained reset/powerdown ports, follow these steps:

1. After compiling your design, select the **TimeQuest Timing Analyzer** in the **Tools** drop-down menu. This opens up the Quartus II TimeQuest Timing Analyzer window.

2. In the **Tasks** pane, execute **Report Unconstrained Paths**. This will report all unconstrained paths in RED in the **Report** pane.

3. Expand the **Unconstrained Paths** option in the **Report** pane and further expand the **Setup Analysis** or **Hold Analysis** option.

4. Under **Setup Analysis** or **Hold Analysis**, you will see **Unconstrained Input Port Paths**, **Unconstrained Output Port Paths**, or both, depending on how the reset/powerdown ports are driven.

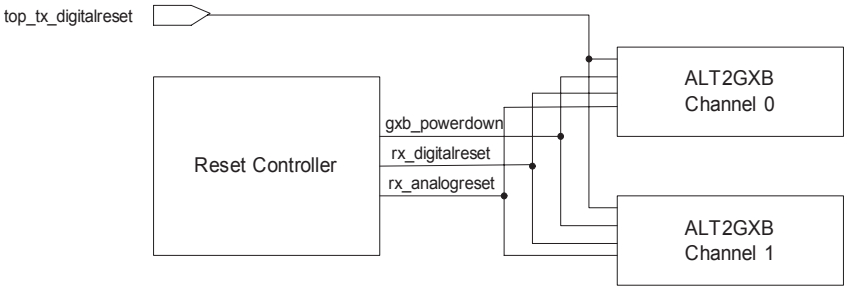
    a. If a reset/powerdown port is driven by an input pin, it will be listed in the **Unconstrained Input Port Paths** report.

    b. If a reset/powerdown port is driven by synchronous logic, it will be listed in the **Unconstrained Output Port Paths** report.

5. In the **Unconstrained Input Port Paths** and **Unconstrained Output Port Paths** reports, the unconstrained reset/powerdown ports of your ALT2GXB instances are listed under the **To** column.

Consider the design example in Figure 2–36.

Figure 2–36. Example Design for TimeQuest Timing Analyzer Constraints



In the design example in Figure 2–36, all reset/powerdown ports except the `tx_digitalreset` port for the two channels are driven by the reset controller. The `tx_digitalreset` port is driven from an input pin.

Figures 2–37 and 2–38 show the TimeQuest Timing Analyzer Report for **Unconstrained Input Port Paths** and **Unconstrained Output Port Paths**, respectively.

Figure 2–37. Unconstrained Input Port Paths

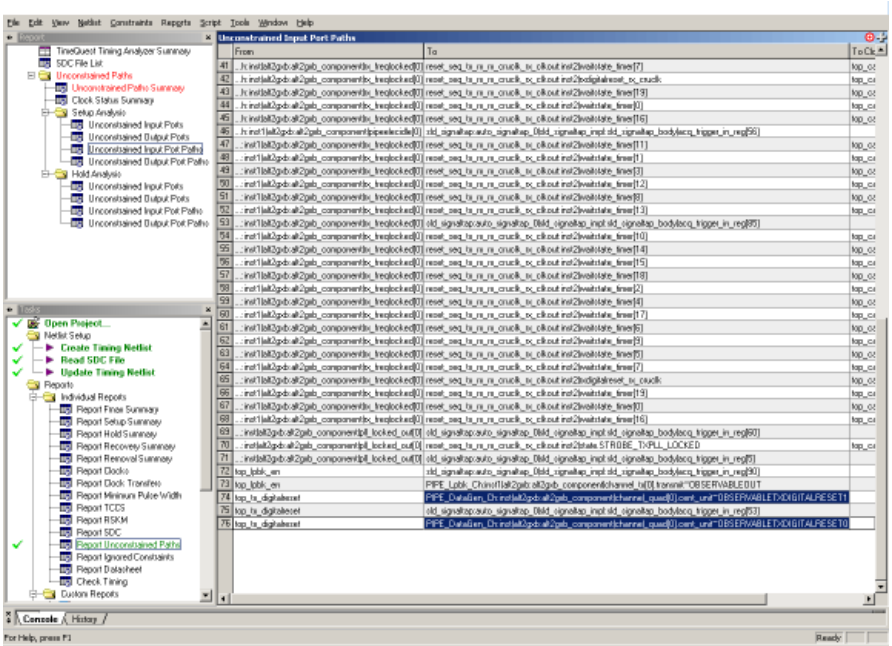
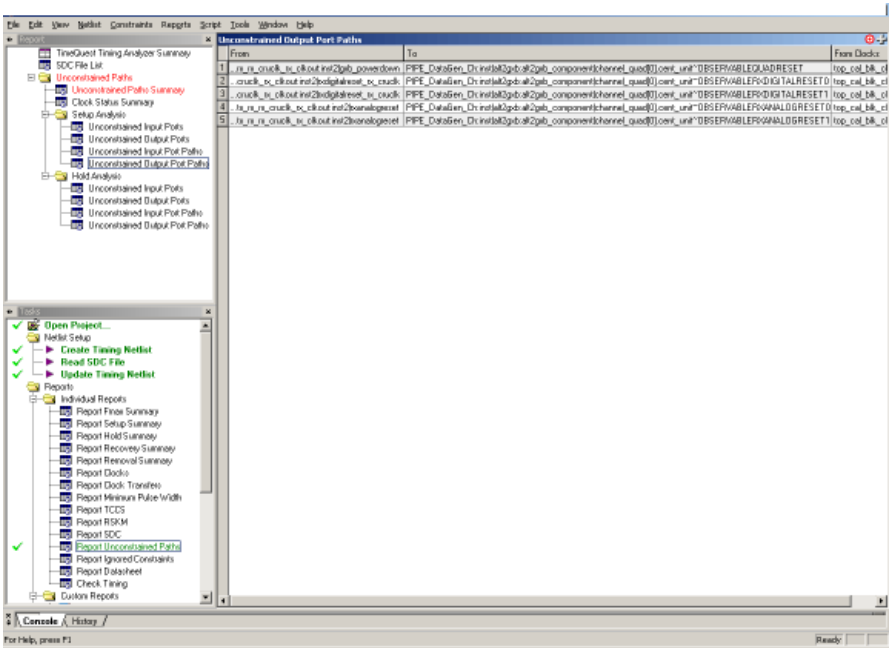


Figure 2–38. Unconstrained Output Port Paths



Having identified the unconstrained reset/powerdown ports in the design, the next step is to constrain these ports.

Setting Reset/Powerdown Port Timing Constraints

You must add the reset/powerdown port timing constraints either directly in the SDC file or through the TimeQuest Timing Analyzer GUI.

To add the timing constraints using the TimeQuest GUI, follow these steps:

1. Locate the reset/powerdown ports in either the **Unconstrained Input Port Paths** or **Unconstrained Output Port Paths** report.

2. Right click on the reset/powerdown port in the **To** column and select **Set Max Delay**. On the resulting window, enter an initial **Delay Value** of **4** ns.

3. Right click on the reset/powerdown port in the **To** column again and select **Set Min Delay**. On the resulting window, enter an initial **Delay Value** of **1.2** ns.

The difference between the maximum delay and minimum delay is set to **2.8** ns which is the maximum skew allowed on reset/powerdown ports.

4. Similarly, set the maximum and minimum delay for all transceiver reset/powerdown ports in your design.

5. Execute **Update Timing Netlist** and **Write SDC File** by double-clicking these options in the **Tasks** pane of the TimeQuest Timing Analyzer window. Confirm that the above timing constraints were added to the SDC file linked with your design.

6. Run the Quartus II Fitter.

7. After the Quartus II Fitter operation completes, execute **Update Timing Netlist** by double-clicking this option in the **Tasks** pane of TimeQuest Timing Analyzer window.

8. Execute **Report Top Failing Paths** by double-clicking this option in the **Tasks** pane of the TimeQuest Timing Analyzer window.

9. Assuming all other paths in your design meet timing, one or more of the paths involving reset/powerdown ports might report timing violations. This is because the design is not able to meet the preliminary timing constraints of 4 ns (maximum delay) and 1.2 ns (minimum delay).

10. Note the slack in the timing report for all failing paths and adjust the maximum delay and the minimum delay values in the SDC file. Maintain a difference of 2.8 ns between the maximum delay and the minimum delay for each reset/powerdown port.

11. After adjusting the delay values, execute **Update Timing Netlist** and run the Quartus II Fitter again.

12. After the Quartus II Fitter operation completes, execute **Update Timing Netlist**.

13. Execute **Report Top Failing Paths** once again. If there are any failing paths involving the reset/powerdown ports, adjust the delay values in the SDC file and repeat the procedure until no failing paths are reported.

Consider the previous design example in which all unconstrained ports were identified. The following example shows how to set the constraints for the gxb_powerdown port. The same procedure must be followed for all other reset ports.

After setting the maximum and minimum delay for the gxb_powerdown port, the SDC file should have the following constraints:

```
#****************************************************
# Set Maximum Delay
#****************************************************

set_max_delay -from  [get_keepers
{reset_seq_tx_rx_rx_cruclk_rx_clkout:inst2|gxb_powerd
own}]  -to  [get_ports
{PIPE_DataGen_Ch:inst|alt2gxb:alt2gxb_component|chann
el_quad[0].cent_unit~OBSERVABLEQUADRESET}] 4.000

#****************************************************
# Set Minimum Delay
#****************************************************

set_min_delay -from  [get_keepers
{reset_seq_tx_rx_rx_cruclk_rx_clkout:inst2|gxb_powerd
own}]  -to  [get_ports
{PIPE_DataGen_Ch:inst|alt2gxb:alt2gxb_component|chann
el_quad[0].cent_unit~OBSERVABLEQUADRESET}] 1.200
```

After running the Quartus II fitter with the above timing constraints for the gxb_powerdown port, the following slack is reported on this path after executing **Report Top Failing Paths** (Figure 2–39).

Figure 2–39. Slack Reported for the gxb_powerdown Port

SLACK: -0.798 ns (VIOLATED)

Path Summary

	Property	Value
1	To Node	PIPE_DataGen_Ch:inst|alt2gxb:alt2gxb_component|channel_quad[0].cent_unit~OBSERVABLEQUADRESET
2	Slack	-0.798 (VIOLATED)
3	Launch Clock	cal_blk_clk
4	Latch Clock	n/a
5	From Node	reset_seq_tx_rx_rx_cruclk_rx_clkout:inst2|gxb_powerdown
6	Data Required Time	4.000
7	Data Arrival Time	4.798

Since the data arrival time is later than the data required time by 0.798 ns, the maximum delay and minimum delay should both be incremented by 0.8 ns in the SDC file. The new SDC file should have the following modified constraints for the gxb_powerdown port.

```
#**************************************************
# Set Maximum Delay
#**************************************************

set_max_delay -from  [get_keepers
{reset_seq_tx_rx_rx_cruclk_rx_clkout:inst2|gxb_powerd
own}]  -to  [get_ports
{PIPE_DataGen_Ch:inst|alt2gxb:alt2gxb_component|chann
el_quad[0].cent_unit~OBSERVABLEQUADRESET}] 4.8

#**************************************************
# Set Minimum Delay
#**************************************************

set_min_delay -from  [get_keepers
{reset_seq_tx_rx_rx_cruclk_rx_clkout:inst2|gxb_powerd
own}]  -to  [get_ports
{PIPE_DataGen_Ch:inst|alt2gxb:alt2gxb_component|chann
el_quad[0].cent_unit~OBSERVABLEQUADRESET}] 2.000
```

After modifying the SDC file and running the Quartus II Fitter, the **Update Timing Netlist** option should be executed, followed by **Report Top Failing Paths**. If the gxb_powerdown port still shows in the failing paths, modify the slack appropriately in the SDC file and repeat the procedure until timing is met on this path.

Follow the same procedure to set timing constraints on all transceiver reset/powerdown ports in your design.

You should set constraints and meet timing for both fast and slow timing models. The same maximum and minimum delay constraints might not be able to meet timing for both timing models. This is acceptable as long as the skew is within the specified period (2.8 ns) for each path in the SDC file for each timing model.

Unconstrained Asynchronous ALT2GXB Ports

In the Quartus II software versions 7.1 and 7.1 sp1, TimeQuest does not automatically constrain transceiver asynchronous input/output ports. These ports are listed in Table 2–25.

Table 2–25. TImeQuest Port Names Versus ALT2GXB Port Names

TimeQuest Port Name	ALT2GXB Port Name
ala2size	rx_ala2size
enapatternalign	rx_enapatternalign
bitslip	rx_bitslip
rlv	rx_rlv
invpol	rx_invpolarity
enabyteord	rx_enabyteord
pipe8b10binvpolarity	pipe8b10binvpolarity
revbitorderwa	rx_revbitorderwa
bisterr	rx_bisterr
bistdone	rx_bitstdone
phaselockloss	rx_pll_locked
freqlock	rx_freqlocked
seriallpbkben	rx_seriallpbken

You must manually add the timing constraints in the SDC file for TimeQuest to analyze these paths. For these asynchronous ports, you only need to set a maximum delay constraint of **10** ns in the SDC file.

To identify all unconstrained ALT2GXB asynchronous ports, execute **Report Unconstrained Paths** in TimeQuest Timing Analyzer after running the Quartus II Fitter. Set a maximum delay of **10** ns for all such ports in the SDC file.

For example, if the `rx_invpolarity` signal is driven by the signal `top_rx_invpolarity` on an input pin, the SDC file constraint for this port should be set as:

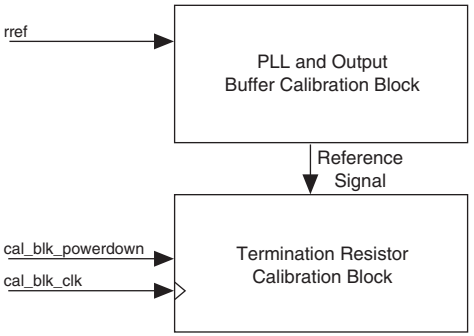
```
set_max_delay -from  [get_ports {top_rx_invpolarity}]
-to  [get_keepers
{xcvr_inst.receive~OBSERVABLEINVPOL}] 10.000
```

Follow the same procedure to constrain all asynchronous ALT2GXB ports in your design before closing timing analysis for your design.

Calibration Blocks

The Arria GX transceiver block contains calibration circuits to calibrate the on-chip termination, the PLLs, and the output buffers. The calibration circuits are divided into two main blocks: the PLL and output buffer calibration block and the termination resistor calibration block (refer to Figure 2–40). Each transceiver block contains a PLL and output buffer calibration block that calibrates the PLLs and output buffers within that particular transceiver block. Each device contains one termination resistor calibration block that calibrates all the termination resistors in the transceiver channels of the entire device.

Figure 2–40. Calibration Blocks



PLL and Output Buffer Calibration Block

Each Arria GX transceiver block contains a PLL and output buffer calibration circuit to counter the effects of PVT (process, voltage, and temperature) on the PLL and output buffer. Each transceiver block's calibration circuit uses a voltage reference derived from an external reference resistor. One reference resistor must be connected to the RREFB pin of each active transceiver block in the Arria GX device. Reference resistor pins (RREFB) of unused transceiver blocks (except the transceiver blocks feeding the termination resistor calibration block) can be left unconnected or tied to a stable power supply.

Termination Resistor Calibration Block

The Arria GX transceiver on-chip termination resistors in the transceiver channels of the entire device are calibrated by a single calibration block. This block ensures that process, voltage, and temperature variations do not have an impact on the termination resistor value.

A 2 KΩ reference resistor must be connected to the RREFB pins of transceiver banks 13 and 14 to ensure proper operation of the termination resistor calibration block.

The termination resistor calibration circuit requires a calibration clock (cal_blk_clk) with frequency between 10 MHz and 125 MHz. Any clock resource can be used to provide the calibration clock.

For multiple ALT2GXB instances in the design, if all the instances have the calibration block instantiated, the cal_blk_clk port of all instances must be tied to a common clock. The Quartus II design compilation provides an error message if the cal_blk_clk ports of multiple ALT2GXB instances in the design are tied to different clock sources.

The calibration block can be powered down through the optional cal_blk_powerdown port. The cal_blk_powerdown is an active low signal. Powering down the calibration block during operations may yield transmit and receive data errors. You can use this port to reset the calibration block to initiate a re-calibration of the termination resistors to account for variations in temperature or voltage.

Referenced Document

This chapter references the following document:

- *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*

Document Revision History

Table 2–26 shows the revision history for this chapter.

Table 2–26. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
August 2007, v1.2	Added the "Referenced Document" section.	—
	Minor text edits.	—
June 2007 v1.1	Added "TimeQuest Timing Analyzer" section.	—
	Added GIGE information.	—
May 2007 v1.0	Initial release.	—

3. Arria GX ALT2GXB Megafunction User Guide

AGX52003-1.2

Introduction

The MegaWizard® Plug-In Manager in the Quartus® II software creates or modifies design files that contain custom megafunction variations that can then be instantiated in a design file. The MegaWizard Plug-In Manager provides a wizard that allows you to specify options for the ALT2GXB megafunction. You can use the wizard to set the ALT2GXB megafunction features in the design.

Start the MegaWizard Plug-In Manager using one of the following methods:

- Choose the **MegaWizard Plug-In Manager** command (Tools menu).
- When working in the Block Editor, click **MegaWizard Plug-In Manager** in the **Symbol** dialog box (Edit menu).
- Start the stand-alone version of the MegaWizard Plug-In Manager by typing the following command at the command prompt: qmegawiz.

The ALT2GXB MegaWizard Plug-In Manager allows you to configure one or more transceiver channels.

Figure 3–1 shows the first page of the MegaWizard Plug-In Manager. To generate an ALT2GXB custom megafunction variation, select **Create a new custom megafunction variation**.

Figure 3–1. MegaWizard Plug-In Manager (Page 1)

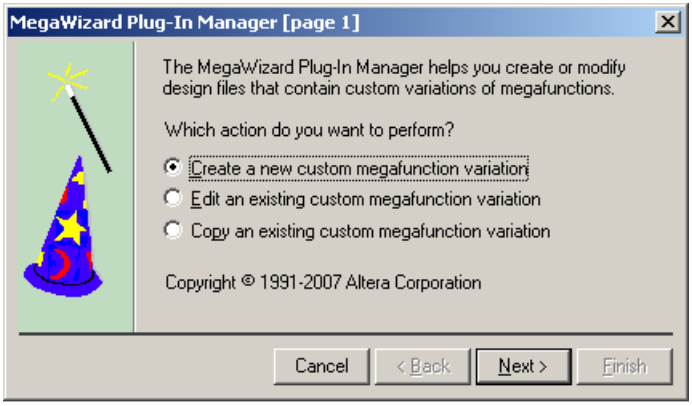
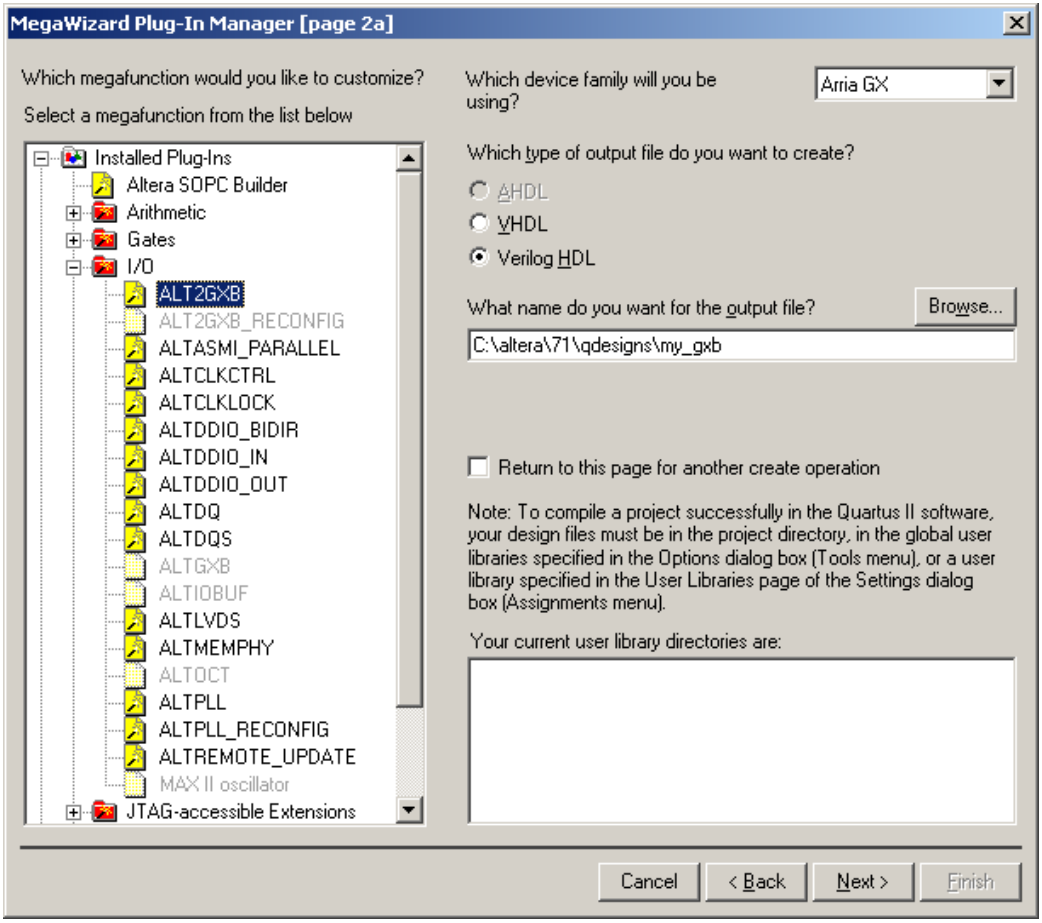


Figure 3–2 shows the second page of the MegaWizard Plug-In Manager. Select the Arria™ GX device as the device family.

Figure 3–2. MegaWizard Plug-In Manager (Page 2)



PCI Express (PIPE) Mode

This section provides descriptions of the options available on the individual pages of the ALT2GXB MegaWizard Plug-In Manager for the PCI Express (PIPE) mode. The MegaWizard Plug-In Manager provides a warning if any of the settings you choose are illegal.

Figure 3–3 shows page 3 of the ALT2GXB MegaWizard Plug-In Manager for PCI Express (PIPE) mode.

Figure 3–3. MegaWizard Plug-In Manager - ALT2GXB (General)

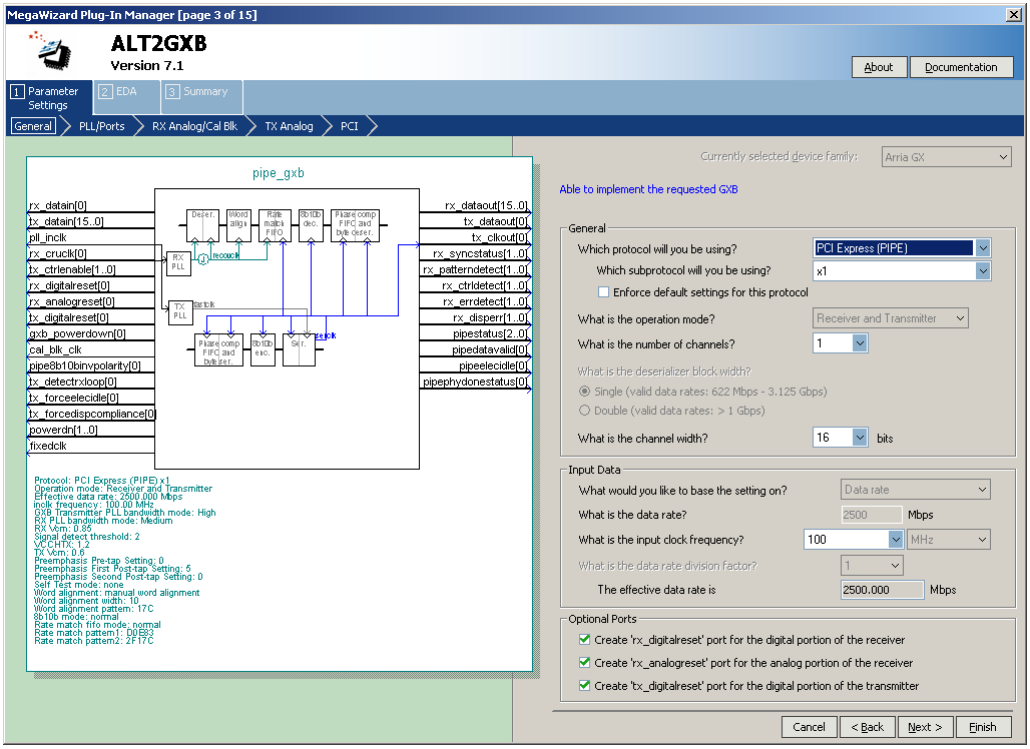


Table 3–1 describes the available options on page 3 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

Table 3–1. MegaWizard Plug-In Manager Options (Page 3 for PCI Express [PIPE] Mode) (Part 1 of 2)

ALT2GXB Setting	Description	Reference
Which protocol will you be using?	Determines the specific protocol or modes under which the transceiver operates. The possible selections are PCI Express (PIPE), GIGE, and Serial RapidIO™. For the PCI Express (PIPE) mode, you must select the PCI Express (PIPE) protocol.	—
Which subprotocol will you be using?	In PCI Express (PIPE) mode, the subprotocols are the supported link widths:×1 or ×4	PCI Express (PIPE) Mode section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Enforce default settings for this protocol	Selecting this option skips the **PCI** page in the PCI Express (PIPE) MegaWizard Plug-In Manager. The **PCI** page allows you to select the PCI Express (PIPE) specific ports for your design. If you select this option, all PCI Express (PIPE) specific ports are used.	—
What is the operation mode?	Only the receiver and transmitter (full duplex) mode is allowed in the PCI Express (PIPE) mode. Receiver-only and transmitter only modes are not allowed.	—
What is the number of channels?	This determines how many duplicate channels this ALT2GXB instance contains. In a x4 subprotocol, the number of channels increments by 4.	—
What is the deserializer block width?	This option is unavailable in PCI Express (PIPE) ALT2GXB instance.	—
What is the channel width?	This option determines the PLD-transceiver interface width. Only 16-bit interface width is supported.	Byte Serializer and Byte Deserializer sections in the *Arria GX Architecture* chapter in volume 1 of the *Arria GX Device Handbook*
What would you like to base the setting on?	This option is unavailable because the data rate is fixed at 2500 Mbps for PCI Express (PIPE) mode.	—
What is the data rate?	This option is unavailable because the data rate is fixed at 2500 Mbps for PCI Express (PIPE) mode.	—

Table 3–1. MegaWizard Plug-In Manager Options (Page 3 for PCI Express [PIPE] Mode) (Part 2 of 2)

ALT2GXB Setting	Description	Reference
What is the input clock frequency?	Determines the input reference clock frequency for the transceiver. In PCI Express (PIPE) mode, only 100 MHz is allowed.	PCI Express (PIPE) Mode section in the *Arria GX Architecture* chapter in volume 1 of the *Arria GX Device Handbook*
What is the data rate division factor?	This option is unavailable in PCI Express (PIPE) `ALT2GXB` instance.	—
Create `rx_digitalreset` port for the digital portion of the receiver	Receiver digital reset port. Resets the PCS logic of the receiver. Altera® recommends using this port to implement the recommended reset sequence.	Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create `rx_analogreset` port for the analog portion of the receiver	Receiver analog reset port.	Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create `tx_digitalreset` port for the digital portion of the transmitter	Transmitter digital reset port. Resets the PCS logic of the transmitter. Altera recommends using this port to implement the recommended reset sequence.	Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*

Figure 3–4 shows page 4 of the ALT2GXB MegaWizard Plug-In Manager for PCI Express (PIPE) mode.

Figure 3–4. MegaWizard Plug-In Manager - ALT2GXB (PLL/Ports)

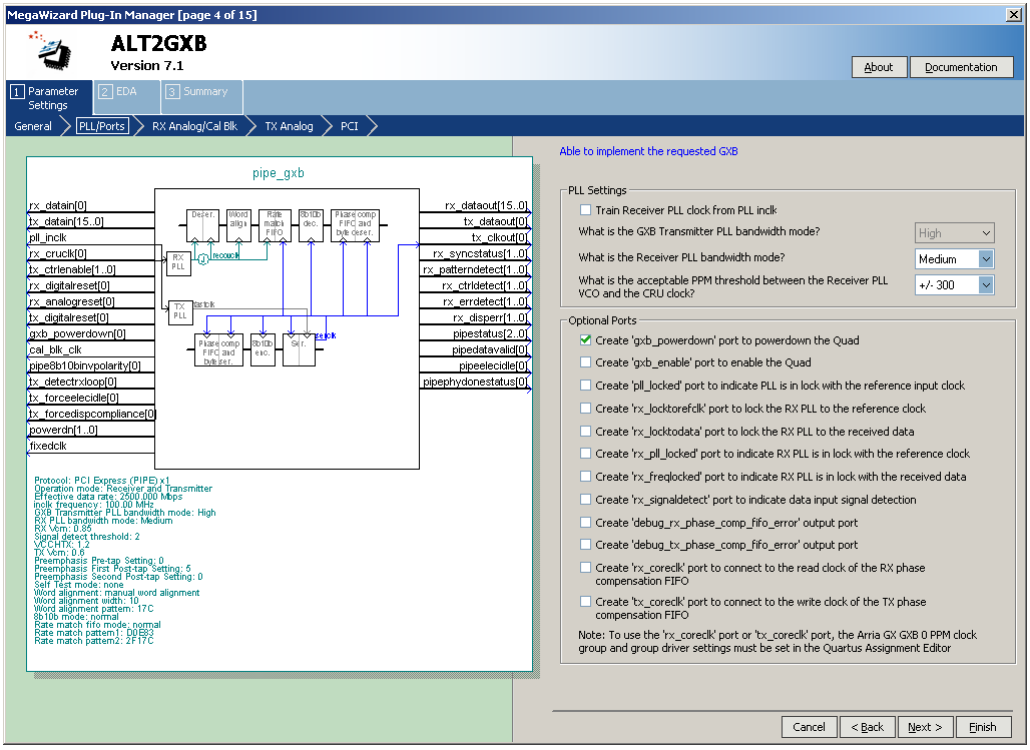


Table 3–2 describes the available options on page 4 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

Table 3–2. MegaWizard Plug-In Manager Options (Page 4 for PCI Express [PIPE] Mode) (Part 1 of 3)

ALT2GXB Setting	Description	Reference
Train Receiver PLL clock from PLL_inclk	If you select this option, the transmitter input reference clock (pll_inclk) drives the receiver PLL input reference clock also. If you do not select this option, the signal on the rx_cruclk port drives the receiver PLL input reference clock.	—
What is the GXB Transmitter PLL bandwidth mode?	This option is not available in PCI Express (PIPE) mode because the transmitter PLL bandwidth is fixed at high.	—
What is the Receiver PLL bandwidth mode?	This option is not available in PCI Express (PIPE) mode because the receiver PLL bandwidth is fixed at medium.	—
What is the acceptable PPM threshold between the Receiver PLL VCO and the CRU clock?	This option determines the PPM difference that affects the automatic receiver clock recovery unit (CRU) switchover between lock-to-data and lock-to-reference. (There are additional factors that affect the CRU's transition.)	Clock Recovery Unit (CRU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
Create gxb_powerdown port to power down the Quad	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create gxb_enable port to enable the Quad	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create pll_locked port to indicate PLL is in lock with the reference input clock	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create rx_locktorefclk port to lock the RX PLL to the reference clock	Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Clock Recovery Unit (CRU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*

Table 3–2. MegaWizard Plug-In Manager Options (Page 4 for PCI Express [PIPE] Mode) (Part 2 of 3)

ALT2GXB Setting	Description	Reference
Create `rx_locktodata` port to lock the RX PLL to the received data	Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Clock Recovery Unit (CRU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
Create `rx_pll_locked` port to indicate RX PLL is in lock with the reference clock	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Reset Control and Power Down section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
Create `rx_freqlocked` port to indicate RX PLL is in lock with the received data	Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Clock Recovery Unit (CRU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
Create `rx_signaldetect` port to indicate data input signal detection	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Receiver Buffer section under PCI Express (PIPE) mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create `debug_rx_phase_comp_fifo_error` output port	This optional output port indicates Receiver Phase Compensation FIFO overflow/underrun condition. Note that no ppm difference is allowed between FIFO read and write clocks. This port should be used for debugging purposes only.	Receiver Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
Create `debug_tx_phase_comp_fifo_error` output port	This optional output port indicates Transmitter Phase Compensation FIFO overflow/underrun condition. Note that no ppm difference is allowed between FIFO read and write clocks. This port should be used for debug purpose only.	Transmitter Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*

Table 3–2. MegaWizard Plug-In Manager Options (Page 4 for PCI Express [PIPE] Mode) (Part 3 of 3)

ALT2GXB Setting	Description	Reference
Create `rx_coreclk` port to connect to the read clock of the RX phase compensation FIFO	This optional input port allows you to clock the read side of the Receiver Phase Compensation FIFO with a non-transceiver PLD clock.	PLD-Transceiver Interface Clocking section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
Create `tx_coreclk` port to connect to the write clock of the TX phase compensation FIFO	This optional input port allows you to clock the write side of the Transmitter Phase Compensation FIFO with a non-transceiver PLD clock.	PLD-Transceiver Interface Clocking section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*

Figure 3–3 shows page 5 of the ALT2GXB MegaWizard Plug-In Manager for PCI Express (PIPE) mode.

Figure 3–5. MegaWizard Plug-In Manager - ALT2GXB (RX Analog/Cal Blk)

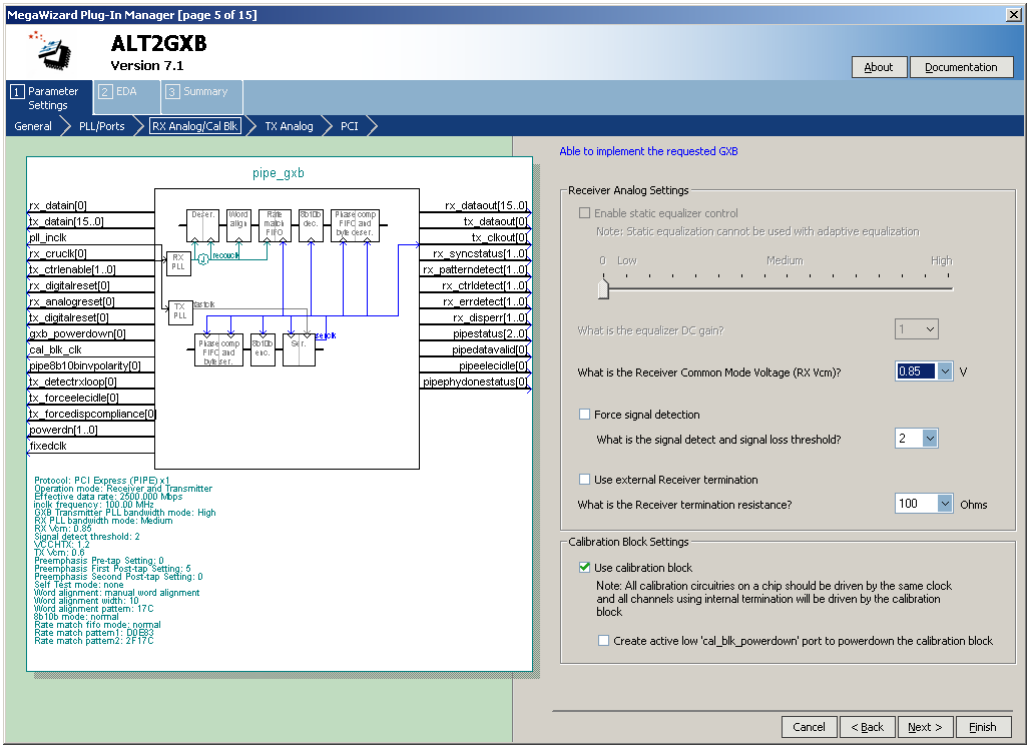


Table 3–3 describes the available options on page 5 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

Table 3–3. MegaWizard Plug-In Manager Options (Page 5 for PCI Express [PIPE] Mode)

ALT2GXB Setting	Description	Reference
What is the Receiver Common Mode Voltage (RX Vcm)?	The receiver common mode voltage is set to 0.85 V.	Receiver Buffer section under PCI Express (PIPE) mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Force signal detection	This option disables the signal detect circuit. You must not select this option as signal detect circuitry is required for electrical idle detection at the receiver.	Receiver Buffer section under PCI Express (PIPE) mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
What is the signal detect and signal loss threshold?	This option sets the trip point of the signal detect circuit. You must select a threshold level of **2** in PCI Express (PIPE) mode.	Receiver Buffer section under PCI Express (PIPE) mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Use external receiver termination	This option is available if you want to use an external termination resistor instead of the on-chip termination (OCT). If checked, this option turns off the receiver OCT.	—
What is the receiver termination resistance?	In PCI Express (PIPE) mode, the only supported receiver termination resistance is 100 Ω.	Receiver Buffer section under PCI Express (PIPE) mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*

Table 3–3. MegaWizard Plug-In Manager Options (Page 5 for PCI Express [PIPE] Mode)

ALT2GXB Setting	Description	Reference
Use calibration block	This option allows you to select which instance of ALT2GXB instantiates the calibration block. Only one instance of ALT2GXB is required to instantiate the calibration block.	Calibration Block section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create active low cal_blk_powerdown to power down the calibration block	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Calibration Block section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*

Figure 3–6 shows page 6 of the ALT2GXB MegaWizard Plug-In Manager for PCI Express (PIPE) mode.

Figure 3–6. MegaWizard Plug-In Manager - ALT2GXB (TX Analog)

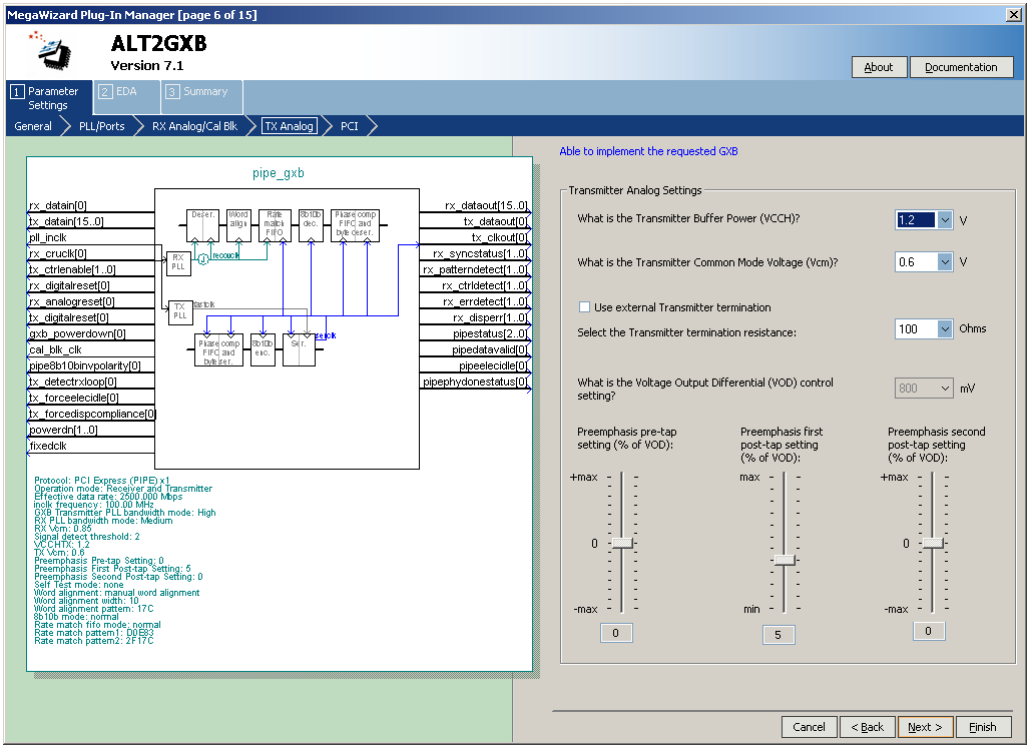


Table 3–4 describes the available options on page 6 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

Table 3–4. MegaWizard Plug-In Manager Options (Page 6 for PCI Express [PIPE] Mode)

ALT2GXB Setting	Description	Reference
What is the Transmitter Buffer Power (VCCH)?	In PCI Express (PIPE) mode, the transmitter buffer power is fixed at 1.2 V. You must connect the VCCH power pins of a PCI Express (PIPE) transceiver bank to a 1.2-V power supply. You must select 1.2-V PCML I/O standard for the transmitter data output pins.	Transmitter Buffer section under PCI Express (PIPE) Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
What is the Transmitter Common Mode Voltage (Vcm)?	In PCI Express (PIPE) mode, the transmitter common mode voltage is fixed at 0.6V.	Transmitter Buffer section under PCI Express (PIPE) Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Use external Transmitter termination	This option is available if you want to use an external termination resistor instead of the on-chip termination (OCT). Checking this option turns off the transmitter OCT.	—
Select the Transmitter termination resistance	In PCI Express (PIPE) mode, the only supported receiver termination resistance is 100 Ω .	Transmitter Buffer section under PCI Express (PIPE) Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
What is the Voltage Output Differential (VOD) control setting?	In PCI Express (PIPE) mode, the only supported peak-peak differential output voltage is 800 mV.	Transmitter Buffer section under PCI Express (PIPE) Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Pre-emphasis pre-tap setting (% of VOD)	In PCI Express (PIPE) mode, this option is unavailable and is fixed to 0.	—

Table 3–4. MegaWizard Plug-In Manager Options (Page 6 for PCI Express [PIPE] Mode)

ALT2GXB Setting	Description	Reference
Pre-emphasis first post-tap setting (% of VOD)	In PCI Express (PIPE) mode, this option is unavailable and is fixed to 5 (to meet the de-emphasis specification required by PCI Express).	Transmitter Buffer section under PCI Express (PIPE) Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Pre-emphasis second post-tap setting (% of VOD)	In PCI Express (PIPE) mode, this option is unavailable and is fixed to 0.	—

Figure 3–7 shows page 13 of the ALT2GXB MegaWizard Plug-In Manager for PCI Express (PIPE) mode. If **Enforce default settings for this protocol** option is selected, this page does not appear in the MegaWizard.

Figure 3–7. MegaWizard Plug-In Manager - ALT2GXB (PCI)

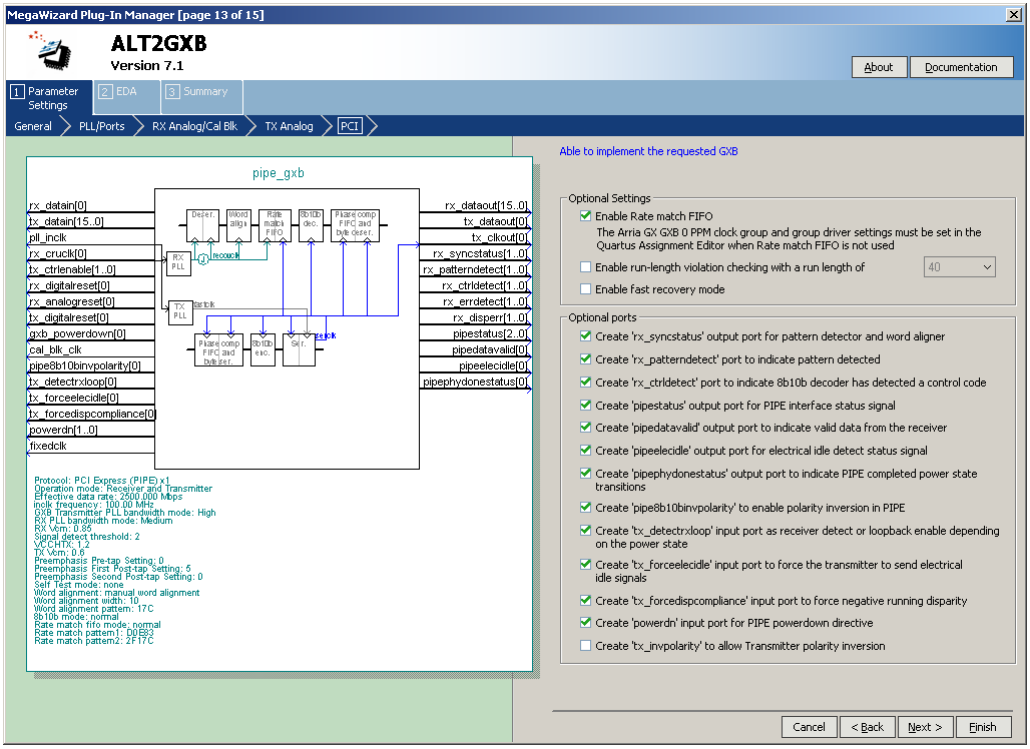


Table 3–5 describes the available options on page 13 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

Table 3–5. MegaWizard Plug-In Manager Options (Page 13 for PCI Express [PIPE] Mode) (Part 1 of 3)

ALT2GXB Setting	Description	Reference
Enable Rate match FIFO	This option enables bypassing of the rate match FIFO in the receiver data path (Low-latency [Synchronous] PCI Express [PIPE] mode)	Low-latency (Synchronous) PCI Express (PIPE) Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Enable run-length violation checking with a run length of	This option activates the run length violation circuit. You can program the run length at which the circuit triggers the rx_rlv signal.	Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
Enable fast recovery mode	This option creates the NFTS fast recovery IP required to meet the PCI Express specification in the PLD logic array.	NFTS Fast Recovery IP (NFRI) section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create rx_syncstatus output port for pattern detector and word aligner	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Word Aligner section under PCI Express (PIPE) Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create rx_patterndetect output port to indicate pattern detected	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Word Aligner section under PCI Express (PIPE) Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create rx_ctrldetect output port to indicate 8B/10B decoder has detected a control code	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	8B/10B Decoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*

Table 3–5. MegaWizard Plug-In Manager Options (Page 13 for PCI Express [PIPE] Mode) (Part 2 of 3)

ALT2GXB Setting	Description	Reference
Create `pipestatus` output port for PIPE interface status signal	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Receiver Status section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create `pipedatavalid` output port to indicate valid data from the receiver	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	PCI Express (PIPE) Mode section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create `pipeelecidle` output port for Electrical Idle detect status signal	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	PCI Express (PIPE) Mode section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create `pipephydonestatus` output port to indicate PIPE completed power state transitions	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	PCI Express (PIPE) Mode section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create `pipe8b/10binvpolarity` to enable polarity inversion in PIPE	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	PCI Express (PIPE) Mode section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create `tx_detectrxloop` input port as receiver detect or loopback enable, depending on the power state	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	PCI Express (PIPE) Mode section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*

Table 3–5. MegaWizard Plug-In Manager Options (Page 13 for PCI Express [PIPE] Mode) (Part 3 of 3)

ALT2GXB Setting	Description	Reference
Create `tx_forceelecidle` input port to force the transmitter to send Electrical Idle signals	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	PCI Express (PIPE) Mode section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create `tx_forcedispcompliance` input port to force negative running disparity	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook 2* for information about this port.	PCI Express (PIPE) Mode section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create `powerdn` input port for PIPE powerdown directive	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	PCI Express (PIPE) Mode section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create `rx_invpolarity` to enable word aligner polarity inversion	This optional port allows you to dynamically reverse the polarity of the received data at the input of the word aligner. This feature must not be enabled when `pipe8b10binvpolarity` is enabled	Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
Create `tx_invpolarity` to allow Transmitter polarity inversion	This optional port allows you to dynamically reverse the polarity of the data to be transmitted at the transmitter PCS-PMA interface.	8B/10B Encoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*\

Figure 3–8 shows page 14 of the MegaWizard Plug-In Manager for the PCI Express (PIPE) protocol selection. The **Generate simulation model** option creates a behavioral model (**.vo** or **.vho**) of the transceiver instance for third-party simulators. The **Generate a netlist for synthesis area and timing estimation** option creates a netlist file (**.syn**) for third-party synthesis tools.

Figure 3–8. MegaWizard Plug-In Manager - ALT2GXB (EDA)

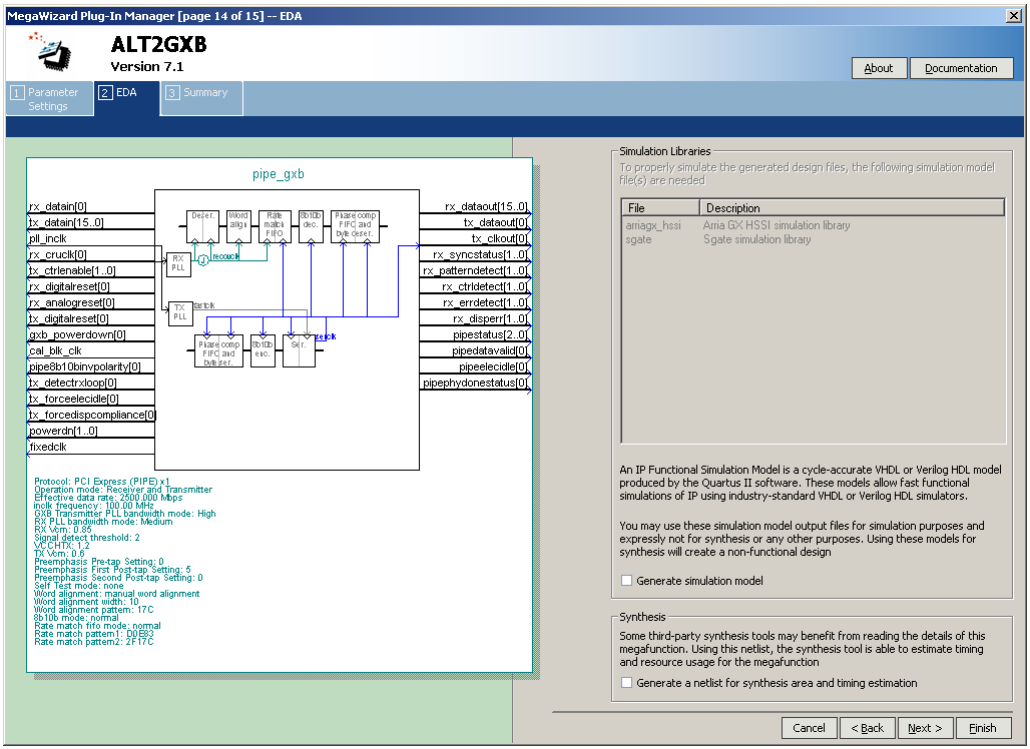
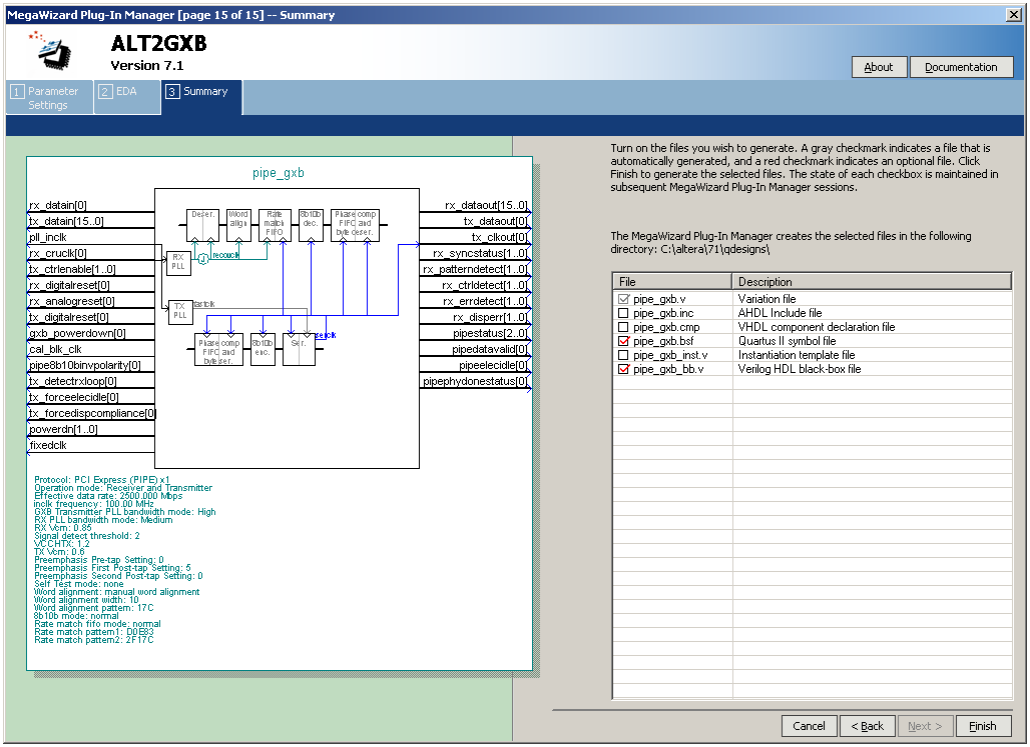


Figure 3–9 shows page 15 (the last page) of the MegaWizard Plug-In Manager for the PCI Express (PIPE) protocol set up. You can select optional files on this page. After you make your selections, click **Finish** to generate the files.

Figure 3–9. MegaWizard Plug-In Manager - ALT2GXB (Summary)



GIGE Mode

This section provides descriptions of the options available on the individual pages of the ALT2GXB MegaWizard Plug-In Manager for the GIGE mode. The MegaWizard Plug-In Manager provides a warning if any of the settings you choose are illegal.

Figure 3–10 shows page 3 of the ALT2GXB MegaWizard Plug-In Manager for GIGE mode.

Figure 3–10. MegaWizard Plug-In Manager - ALT2GXB (page 3)

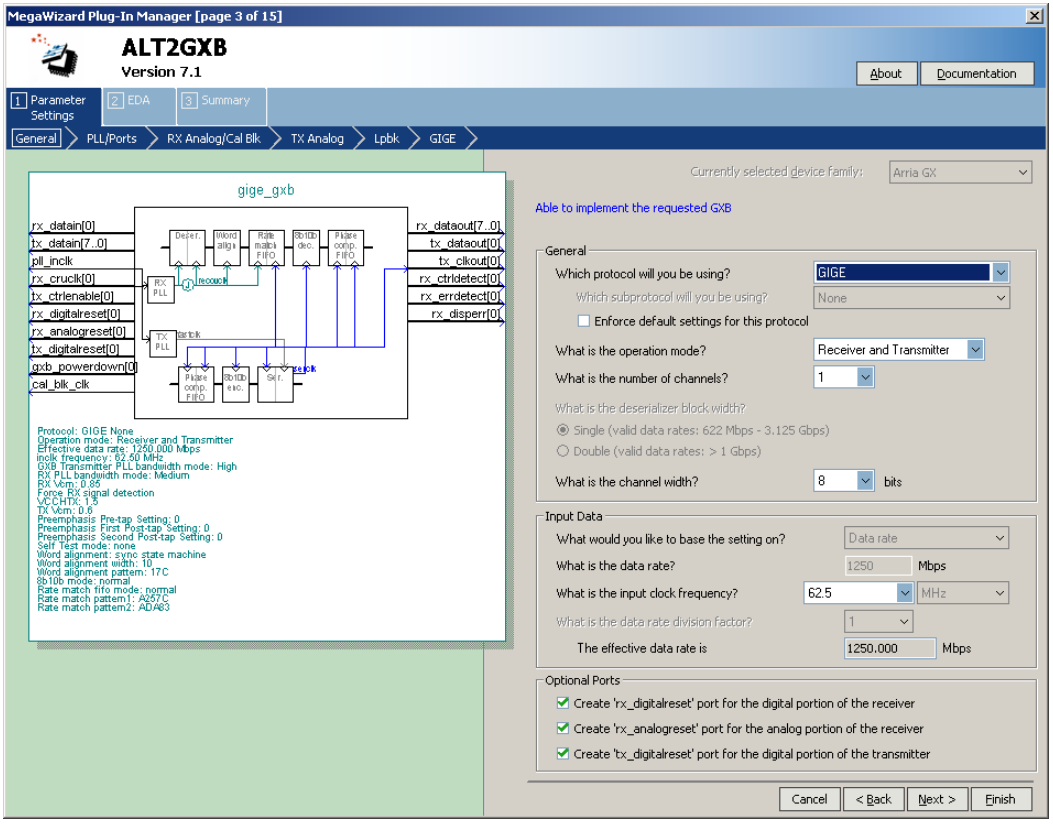


Table 3–6 describes the available options on page 3 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

Table 3–6. MegaWizard Plug-In Manager Options (Page 3 for GIGE Mode) (Part 1 of 2)

ALT2GXB Setting	Description	Reference
Which protocol will you be using?	Determines the specific protocol or modes under which the transceiver operates. The possible selections are PCI Express (PIPE), GIGE, and Serial RapidIO. For the GIGE mode, you must select the GIGE protocol.	—
Which subprotocol will you be using?	Not applicable to GIGE mode.	—
Enforce default settings for this protocol	Selecting this option skips the **GIGE** page of the GIGE MegaWizard Plug-In Manager. The **GIGE** page allows you to select the GIGE-specific ports for your design. If you select this option, all GIGE-specific ports are used.	—
What is the operation mode?	The transmitter only and receiver and transmitter (full duplex) modes are allowed in GIGE protocol. The receiver only mode is not available.	—
What is the number of channels?	This selects how many duplicate channels this ALT2GXB instance contains. In GIGE mode, the number of channels increments by 1.	—
What is the deserializer block width?	This option is unavailable in GIGE ALT2GXB instance.	—
What is the channel width?	This option determines the PLD-transceiver interface width. Only 8-bit interface width is supported.	Byte Serializer and Byte Deserializer sections in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
What would you like to base the setting on?	This option is unavailable because the data rate is fixed at 1250 Mbps for GIGE mode.	—
What is the data rate?	This option is unavailable because the data rate is fixed at 1250 Mbps for GIGE mode.	—
What is the input clock frequency?	Determines the input reference clock frequency for the transceiver. In GIGE mode, input reference clock frequencies of 62.5 MHz and 125 MHz are supported.	GIGE Mode section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
What is the data rate division factor?	This option is unavailable in GIGE ALT2GXB instance.	—

Table 3–6. MegaWizard Plug-In Manager Options (Page 3 for GIGE Mode) (Part 2 of 2)

ALT2GXB Setting	Description	Reference
Create `rx_digitalreset` port for the digital portion of the receiver	Receiver digital reset port. Resets the PCS logic of the receiver. Altera recommends using this port to implement the recommended reset sequence.	Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create `rx_analogreset` port for the analog portion of the receiver	Receiver analog reset port.	Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create `tx_digitalreset` port for the digital portion of the transmitter	Transmitter digital reset port. Resets the PCS logic of the transmitter. Altera recommends using this port to implement the recommended reset sequence.	Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*

Figure 3–11 shows page 4 of the ALT2GXB MegaWizard Plug-In Manager for GIGE mode.

Figure 3–11. MegaWizard Plug-In Manager - ALT2GXB (PLL/Ports)

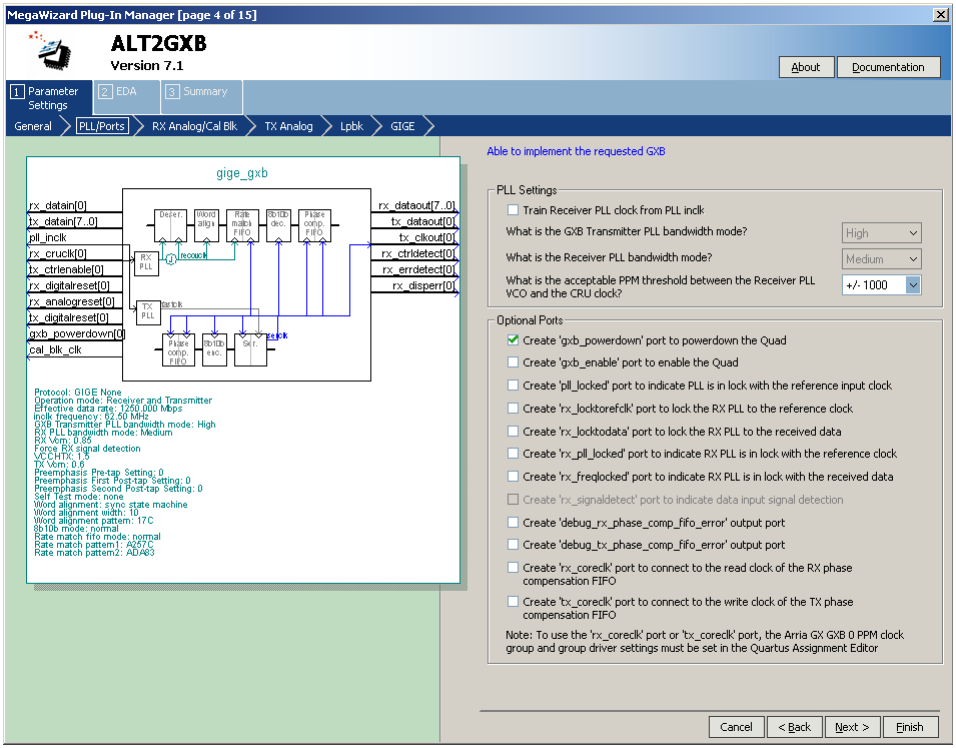


Table 3–7 describes the available options on page 4 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

Table 3–7. MegaWizard Plug-In Manager Options (Page 4 for GIGE Mode) (Part 1 of 3)

ALT2GXB Setting	Description	Reference
Train Receiver PLL clock from PLL_inclk	If you select this option, the transmitter input reference clock (pll_inclk) drives the receiver PLL input reference clock also. If you do not select this option, the signal on the rx_cruclk port drives the receiver PLL input reference clock.	—
What is the GXB Transmitter PLL bandwidth mode?	This option is not available in GIGE mode because the transmitter PLL bandwidth is fixed at high.	—

Table 3–7. MegaWizard Plug-In Manager Options (Page 4 for GIGE Mode) (Part 2 of 3)

ALT2GXB Setting	Description	Reference
What is the Receiver PLL bandwidth mode?	This option is not available in GIGE mode because the receiver PLL bandwidth is fixed at medium.	—
What is the acceptable PPM threshold between the Receiver PLL VCO and the CRU clock?	This option determines the PPM difference that affects the automatic receiver CRU switchover between lock-to-data and lock-to-reference. (There are additional factors that affect the CRU's transition.)	Clock Recovery Unit (CRU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
Create `gxb_powerdown` port to power down the Quad	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create `gxb_enable` port to enable the Quad	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create `pll_locked` port to indicate PLL is in lock with the reference input clock	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create `rx_locktorefclk` port to lock the RX PLL to the reference clock	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Clock Recovery Unit (CRU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
Create `rx_locktodata` port to lock the RX PLL to the received data	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Clock Recovery Unit (CRU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*

Table 3–7. MegaWizard Plug-In Manager Options (Page 4 for GIGE Mode) (Part 3 of 3)

ALT2GXB Setting	Description	Reference
Create `rx_pll_locked` port to indicate RX PLL is in lock with the reference clock	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create `rx_freqlocked` port to indicate RX PLL is in lock with the received data	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Clock Recovery Unit (CRU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
Create `rx_signaldetect` port to indicate data input signal detection	This option is unavailable in GIGE mode.	—
Create `debug_rx_phase_comp_fifo_error` output port	This optional output port indicates Receiver Phase Compensation FIFO overflow/underrun condition. Note that no ppm difference is allowed between FIFO read and write clocks. This port should be used for debug purpose only.	Receiver Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
Create `debug_tx_phase_comp_fifo_error` output port	This optional output port indicates Transmitter Phase Compensation FIFO overflow/underrun condition. Note that no ppm difference is allowed between FIFO read and write clocks. This port should be used for debug purpose only.	Transmitter Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
Create `rx_coreclk` port to connect to the read clock of the RX phase compensation FIFO	This optional input port allows you to clock the read side of the Receiver Phase Compensation FIFO with a non-transceiver PLD clock.	PLD-Transceiver Interface Clocking section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
Create `tx_coreclk` port to connect to the write clock of the TX phase compensation FIFO	This optional input port allows you to clock the write side of the Transmitter Phase Compensation FIFO with a non-transceiver PLD clock.	PLD-Transceiver Interface Clocking section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*

Figure 3–12 shows page 5 of the ALT2GXB MegaWizard Plug-In Manager for GIGE mode.

Figure 3–12. MegaWizard Plug-In Manager - ALT2GXB (RX Analog/Cal Blk)

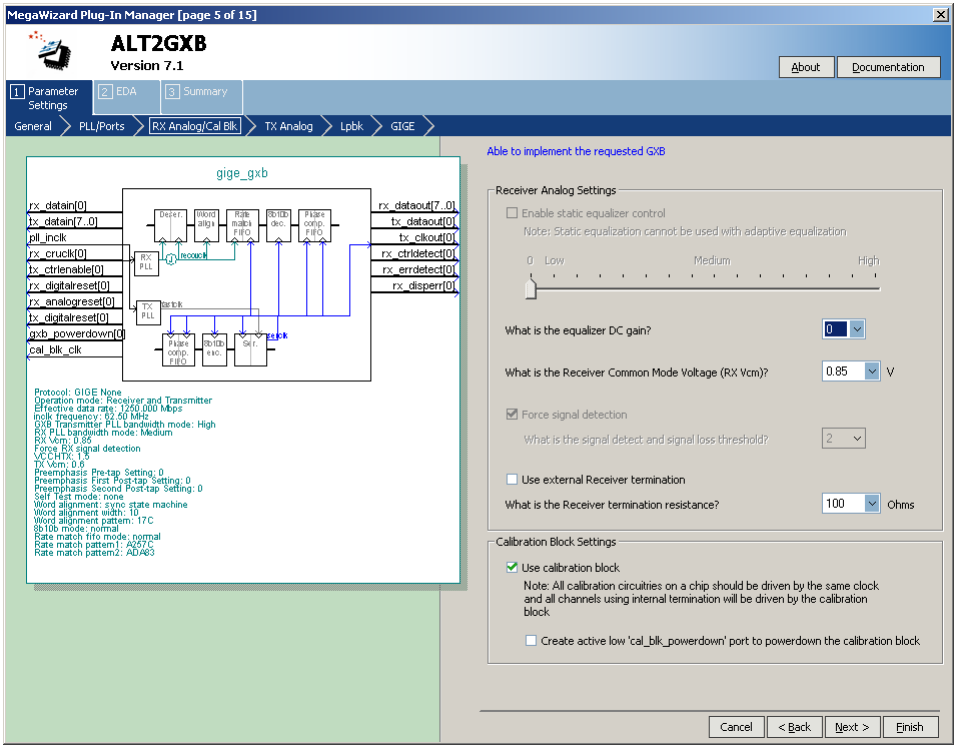


Table 3–8 describes the available options on page 5 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

Table 3–8. MegaWizard Plug-In Manager Options (Page 5 for GIGE Mode)

ALT2GXB Setting	Description	Reference
What is the Receiver Common Mode Voltage (RX Vcm)?	The receiver common mode voltage is set to 0.85 V.	Receiver Buffer section under GIGE Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Force signal detection	This option is unavailable in GIGE mode and is always forced selected.	—
What is the signal detect and signal loss threshold?	This option is unavailable in GIGE mode as signal detection is forced.	—
Use external receiver termination	This option is available if you want to use an external termination resistor instead of the on-chip termination (OCT). If checked, this option turns off the receiver OCT.	—
What is the receiver termination resistance?	In GIGE mode, the only supported receiver termination resistance is 100 Ω.	Receiver Buffer section under GIGE Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Use calibration block	This option allows you to select which instance of ALT2GXB instantiates the calibration block. Only one instance of ALT2GXB is required to instantiate the calibration block.	Calibration Block section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create active low cal_blk_powerdown to power down the calibration block	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Calibration Block section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*

Figure 3–13 shows page 6 of the ALT2GXB MegaWizard Plug-In Manager for GIGE mode.

Figure 3–13. MegaWizard Plug-In Manager - ALT2GXB (TX Analog)

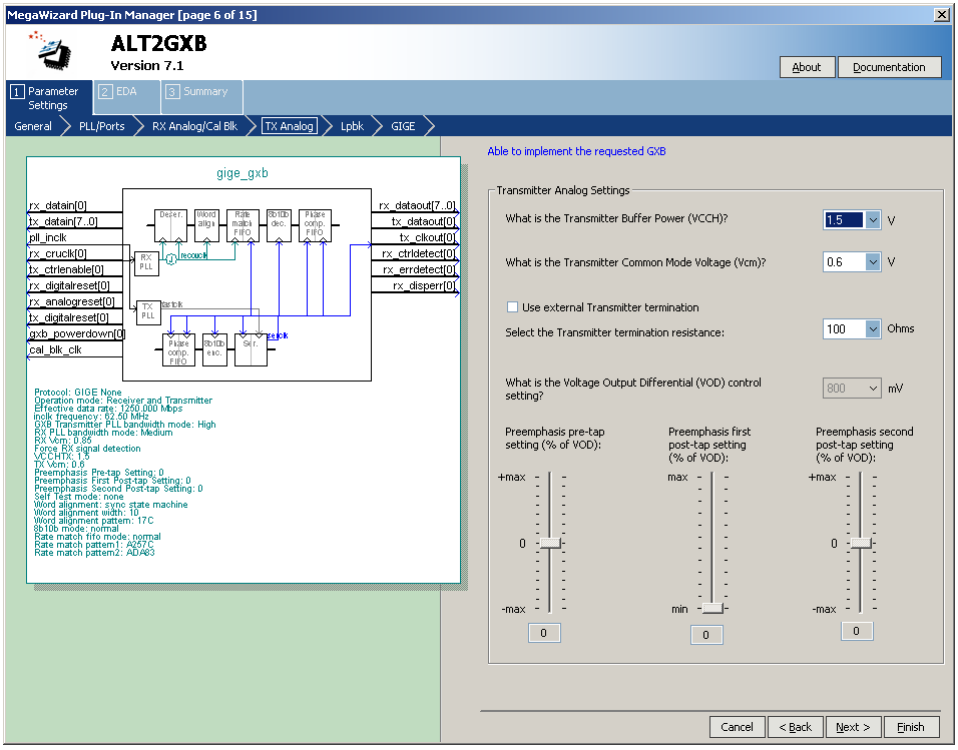


Table 3–9 describes the available options on page 6 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

Table 3–9. MegaWizard Plug-In Manager Options (Page 6 for GIGE Mode)

ALT2GXB Setting	Description	Reference
What is the Transmitter Buffer Power (VCCH)?	In GIGE mode, the transmitter buffer power is fixed at 1.5 V. You must connect the VCCH power pins of a GIGE transceiver bank to a 1.5V power supply. You must select 1.5-V PCML I/O standard for the transmitter data output pins.	Transmitter Buffer section under GIGE Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
What is the Transmitter Common Mode Voltage (Vcm)?	In GIGE mode, the transmitter common mode voltage is fixed at 0.6 V.	Transmitter Buffer section under GIGE Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Use external Transmitter termination	This option is available if you want to use an external termination resistor instead of the on-chip termination (OCT). Checking this option turns off the transmitter OCT.	—
Select the Transmitter termination resistance	In GIGE mode, the only supported receiver termination resistance is 100 Ω.	Transmitter Buffer section under GIGE Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
What is the Voltage Output Differential (VOD) control setting?	In GIGE mode, the only supported peak-peak differential output voltage is 800 mV.	Transmitter Buffer section under GIGE Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Pre-emphasis pre-tap setting (% of VOD)	In GIGE mode, this option is unavailable and is fixed to 0.	—

Table 3–9. MegaWizard Plug-In Manager Options (Page 6 for GIGE Mode)

ALT2GXB Setting	Description	Reference
Pre-emphasis first post-tap setting (% of VOD)	In GIGE mode, this option is unavailable and is fixed to 0.	Transmitter Buffer section under GIGE Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Pre-emphasis second post-tap setting (% of VOD)	In GIGE mode, this option is unavailable and is fixed to 0.	—

Figure 3–14 shows page 11 of the ALT2GXB MegaWizard Plug-In Manager for GIGE mode.

Figure 3–14. MegaWizard Plug-In Manager - ALT2GXB (Loopback)

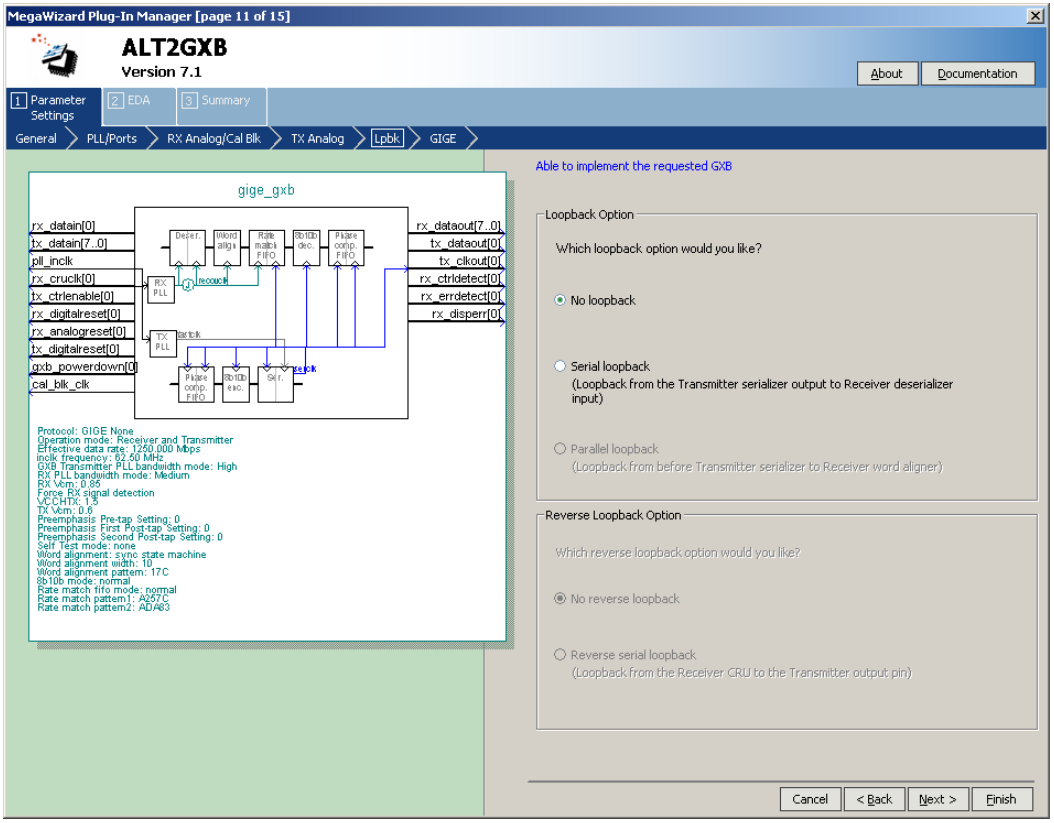


Table 3–10 describes the available options on page 11 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

Table 3–10. MegaWizard Plug-In Manager Options (Page 11 for GIGE Mode)

ALT2GXB Setting	Description	Reference
Which loopback option would you like?	No loopback and serial loopback options are available in GIGE mode. No loopback is the default mode. If you select serial loopback, the rx_seriallpbken port is available to control the serial loopback feature dynamically. A 1'b1 enables serial loopback and a 1'b0 disables loopback on a channel-by-channel basis.	Loopback Modes section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*

Figure 3–15 shows page 13 of the ALT2GXB MegaWizard Plug-In Manager for GIGE mode. If **Enforce default settings for this protocol** option is selected, this page does not appear in the MegaWizard.

Figure 3–15. MegaWizard Plug-In Manager - ALT2GXB (GIGE)

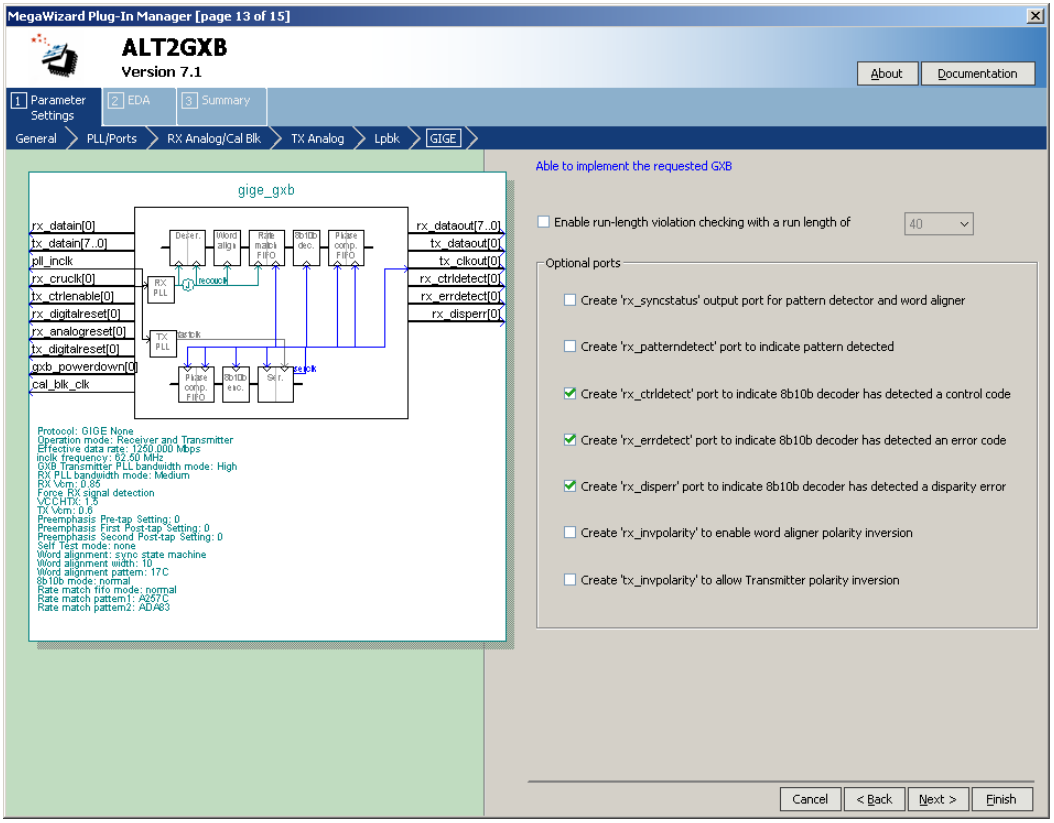


Table 3–11 describes the available options on page 13 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

Table 3–11. MegaWizard Plug-In Manager Options (Page 13 for GIGE Mode)

ALT2GXB Setting	Description	Reference
Enable run-length violation checking with a run length of	This option activates the run-length violation circuit. You can program the run length at which the circuit triggers the rx_rlv signal.	Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
Create rx_syncstatus output port for pattern detector and word aligner	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Word Aligner section under GIGE mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create rx_patterndetect output port to indicate pattern detected	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Word Aligner section under GIGE Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create rx_ctrldetect output port to indicate 8B/10B decoder has detected a control code	Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	8B/10B Decoder section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
Create rx_errdetect port to indicate 8B/10B decoder has detected an error code	Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	8B/10B Decoder section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create rx_disperr port to indicate 8B/10B decoder has detected a disparity error	Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	8B/10B Decoder section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*

Table 3–11. MegaWizard Plug-In Manager Options (Page 13 for GIGE Mode)

ALT2GXB Setting	Description	Reference
Create rx_invpolarity to enable word aligner polarity inversion	This optional port allows you to dynamically reverse the polarity of the received data at the input of the word aligner.	Word Aligner section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
Create tx_invpolarity to allow Transmitter polarity inversion	This optional port allows you to dynamically reverse the polarity of the data to be transmitted at the transmitter PCS-PMA interface.	8B/10B Encoder section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*

Figure 3–16 shows page 14 of the MegaWizard Plug-In Manager for the GIGE protocol selection. The **Generate simulation model** option creates a behavioral model (**.vo** or **.vho**) of the transceiver instance for third-party simulators. The **Generate a netlist for synthesis area and timing estimation** option creates a netlist file (**.syn**) for third-party synthesis tools.

Figure 3–16. MegaWizard Plug-In Manager - ALT2GXB (EDA)

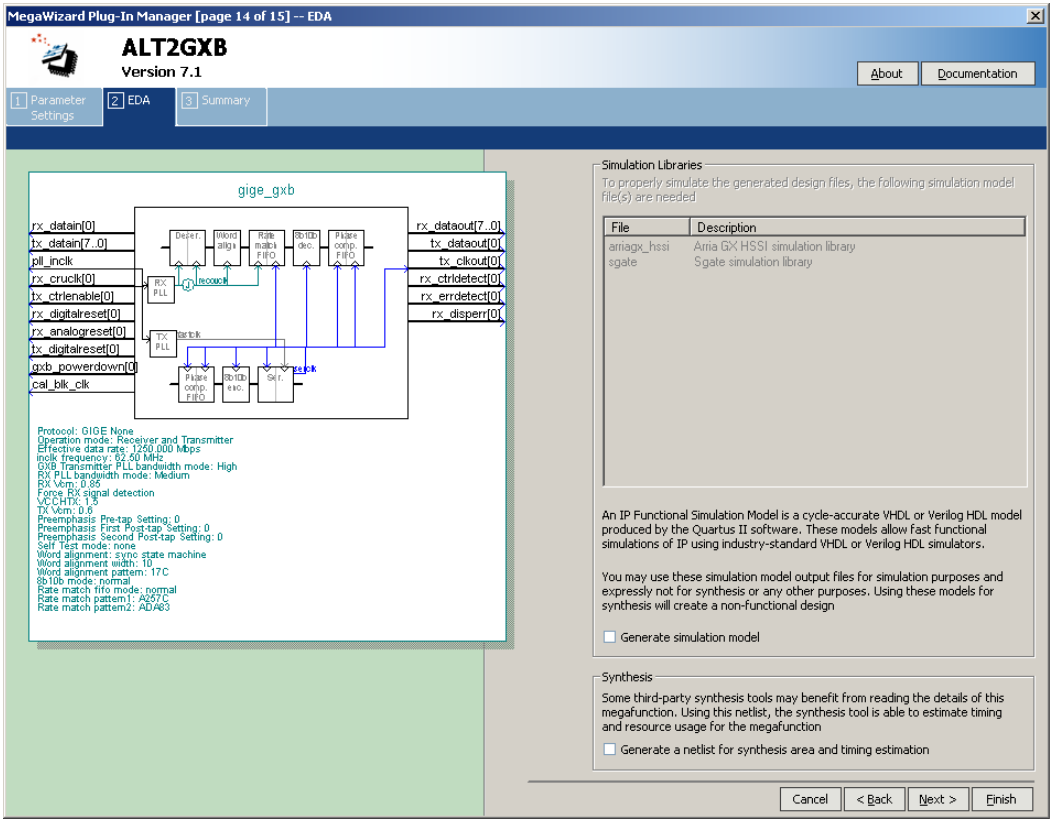
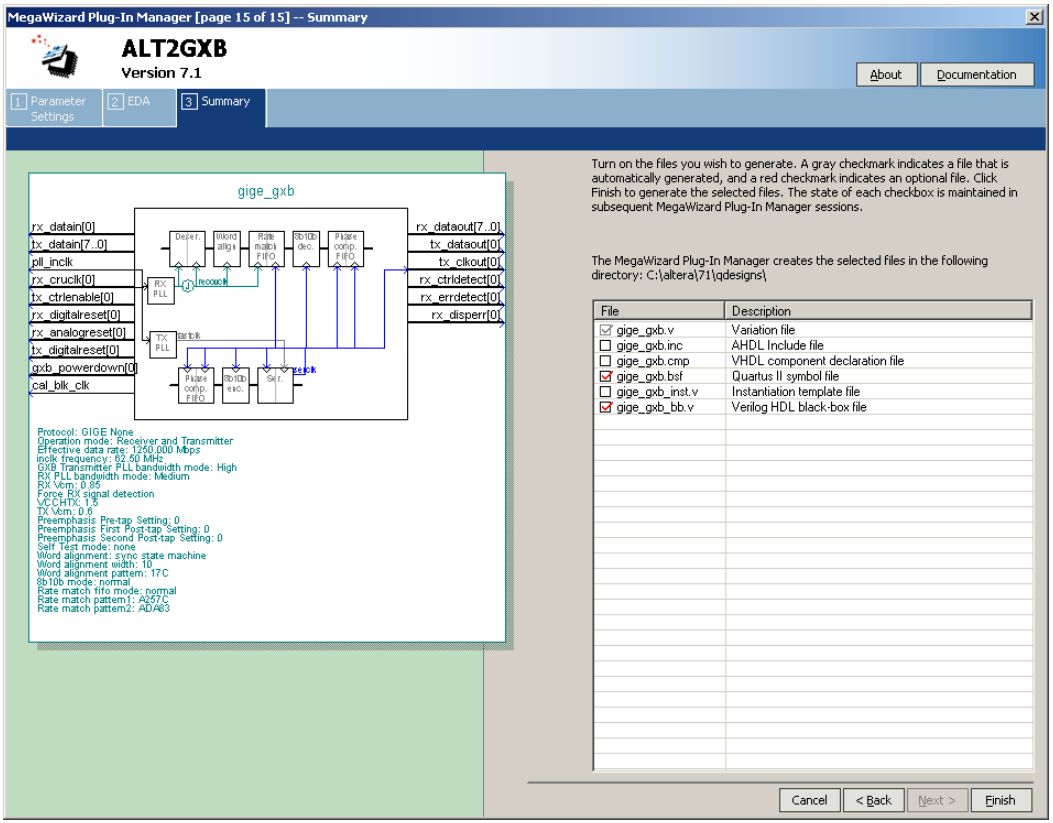


Figure 3–17 shows page 15 (the last page) of the MegaWizard Plug-In Manager for the GIGE protocol set up. You can select optional files on this page. After you make your selections, click **Finish** to generate the files.

Figure 3–17. MegaWizard Plug-In Manager - ALT2GXB (Summary)



Serial RapidIO Mode

This section provides descriptions of the options available on the individual pages of the ALT2GXB MegaWizard Plug-In Manager for Serial RapidIO mode.

The MegaWizard Plug-In Manager provides a warning if any of the settings you choose are illegal.

Figure 3–18 shows page 3 of the ALT2GXB MegaWizard Plug-In Manager in Serial RapidIO mode.

Figure 3–18. MegaWizard Plug-In Manager - ALT2GXB (General)

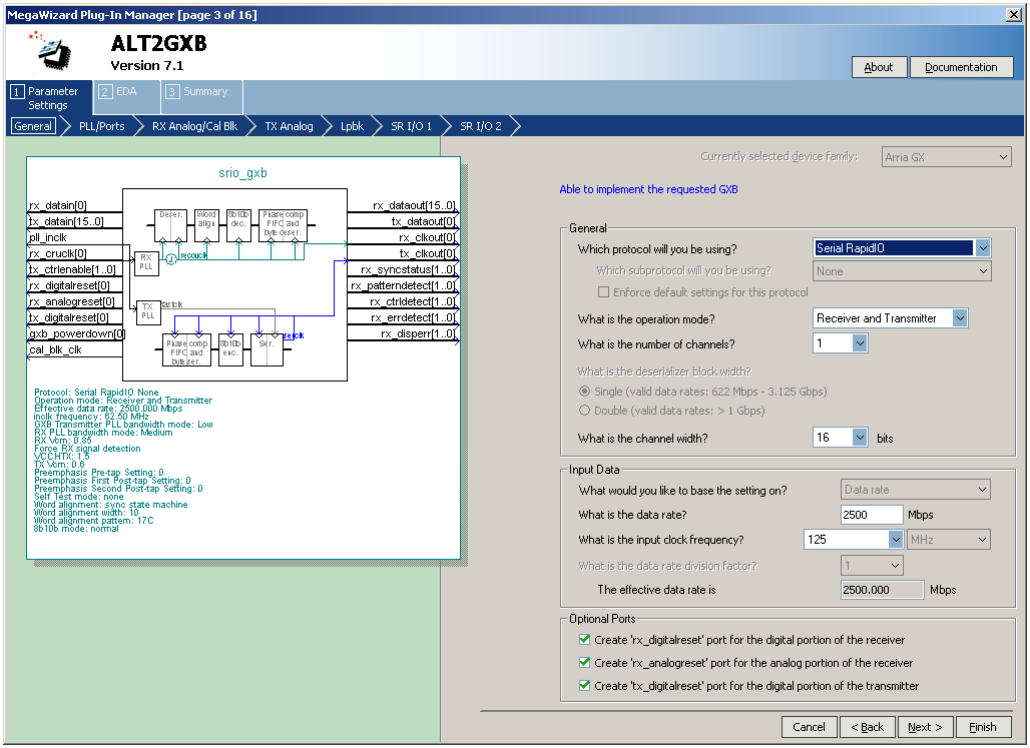


Table 3–12 describes the available options on page 3 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

Table 3–12. MegaWizard Plug-In Manager Options (Page 3 for Serial RapidIO Mode) (Part 1 of 3)

ALT2GXB Setting	Description	Reference
Which protocol will you be using?	Determines the specific protocol or modes under which the transceiver operates. The possible selections are PCI Express (PIPE), GIGE, and Serial RapidIO. For the Serial RapidIO mode, you must select the Serial RapidIO protocol.	—
Which subprotocol will you be using?	Not applicable to Serial RapidIO mode.	—

Table 3–12. MegaWizard Plug-In Manager Options (Page 3 for Serial RapidIO Mode) (Part 2 of 3)

ALT2GXB Setting	Description	Reference
Enforce default settings for this protocol	Not applicable to Serial RapidIO mode.	—
What is the operation mode?	The available operation modes are receiver only, transmitter only, and receiver and transmitter.	—
What is the number of channels?	This option determines how many duplicate channels this ALT2GXB instance contains.	—
What is the deserializer block width?	This option is unavailable in Serial RapidIO ALT2GXB instance.	—
What is the channel width?	This option determines the PLD-transceiver interface width. Only 16-bit interface width is supported.	Byte Serializer and Byte Deserializer sections in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
What would you like to base the setting on?	This option is unavailable in Serial RapidIO mode.	—
What is the data rate?	In Serial RapidIO mode, data rates of 1250 Mbps and 2500 Mbps are supported.	—
What is the input clock frequency?	Determines the input reference clock frequency for the transceiver. The following input reference clock frequencies are supported for each data rate option: 1250 Mbps: 62.5 MHz, 78.125 MHz, 125 MHz, 156.25 MHz 2500 Mbps: 125 MHz, 156.25 MHz, 250 MHz, 312.5 MHz	Serial RapidIO Mode section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
What is the data rate division factor?	This option is unavailable in Serial RapidIO ALT2GXB instance.	—
Create rx_digitalreset port for the digital portion of the receiver	Receiver digital reset port. Resets the PCS logic of the receiver. Altera recommends using this port to implement the recommended reset sequence.	Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*

Table 3–12. MegaWizard Plug-In Manager Options (Page 3 for Serial RapidIO Mode) (Part 3 of 3)

ALT2GXB Setting	Description	Reference
Create `rx_analogreset` port for the analog portion of the receiver	Receiver analog reset port.	Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create `tx_digitalreset` port for the digital portion of the transmitter	Transmitter digital reset port. Resets the PCS logic of the transmitter. Altera recommends using this port to implement the recommended reset sequence.	Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*

Figure 3–19 shows page 4 of the ALT2GXB MegaWizard Plug-In Manager for Serial RapidIO mode.

Figure 3–19. MegaWizard Plug-In Manager - ALT2GXB (PLL/Ports)

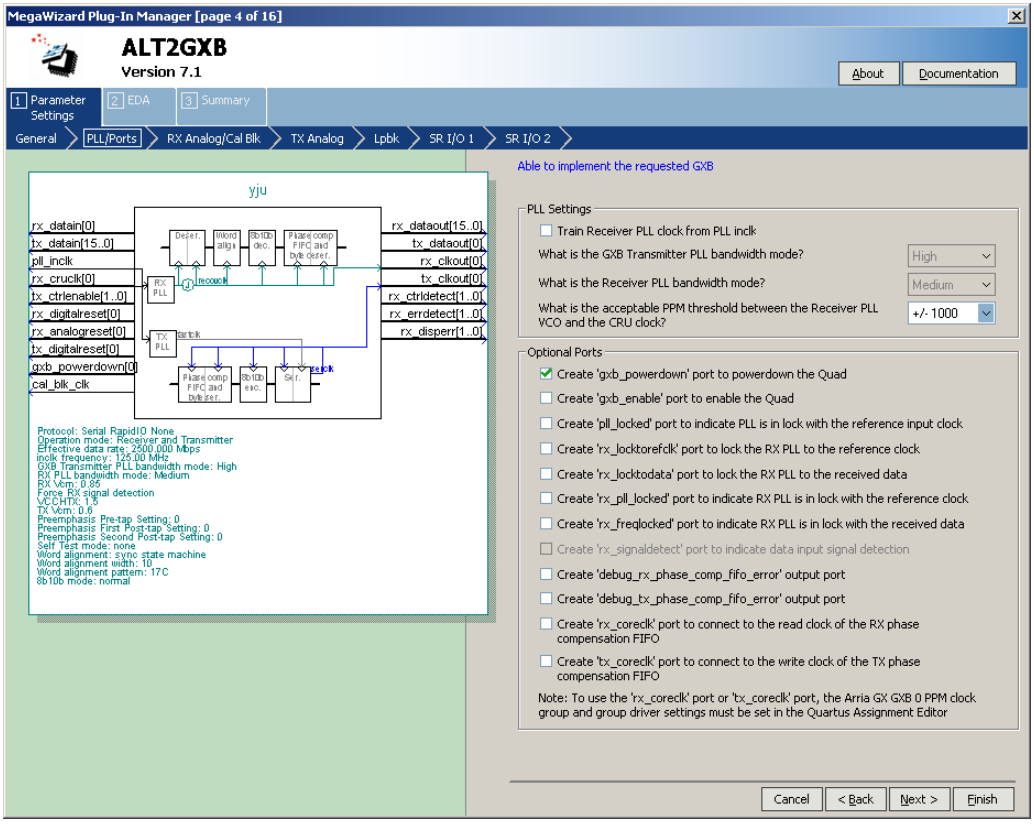


Table 3–13 describes the available options on page 4 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

Table 3–13. MegaWizard Plug-In Manager Options (Page 4 for Serial RapidIO Mode)

ALT2GXB Setting	Description	Reference
Train Receiver PLL clock from PLL inclk	If you select this option, the transmitter input reference clock (pll_inclk) drives the receiver PLL input reference clock also. If you do not select this option, the signal on the rx_cruclk port drives the receiver PLL input reference clock.	—
What is the GXB Transmitter PLL bandwidth mode?	This option is not available in Serial RapidIO mode because the transmitter PLL bandwidth is fixed at high.	—
What is the Receiver PLL bandwidth mode?	This option is not available in Serial RapidIO mode because the receiver PLL bandwidth is fixed at medium.	—
What is the acceptable PPM threshold between the Receiver PLL VCO and the CRU clock?	This option determines the PPM difference that affects the automatic receiver CRU switchover between lock-to-data and lock-to-reference. (There are additional factors that affect the CRU's transition.)	Clock Recovery Unit (CRU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
Create gxb_powerdown port to power down the Quad	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create gxb_enable port to enable the Quad	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create pll_locked port to indicate PLL is in lock with the reference input clock	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*

Table 3–13. MegaWizard Plug-In Manager Options (Page 4 for Serial RapidIO Mode)

ALT2GXB Setting	Description	Reference
Create `rx_locktorefclk` port to lock the RX PLL to the reference clock	Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Clock Recovery Unit (CRU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
Create `rx_locktodata` port to lock the RX PLL to the received data	Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Clock Recovery Unit (CRU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
Create `rx_pll_locked` port to indicate RX PLL is in lock with the reference clock	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Reset Control and Power Down section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create `rx_freqlocked` port to indicate RX PLL is in lock with the received data	Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Clock Recovery Unit (CRU) section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
Create `rx_signaldetect` port to indicate data input signal detection	This option is unavailable in Serial RapidIO mode.	—
Create debug_rx_phase_comp_fifo_error output port	This optional output port indicates Receiver Phase Compensation FIFO overflow/underrun condition. Note that no ppm difference is allowed between FIFO read and write clocks. This port should be used for debug purpose only.	Receiver Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
Create debug_tx_phase_comp_fifo_error output port	This optional output port indicates Transmitter Phase Compensation FIFO overflow/underrun condition. Note that no ppm difference is allowed between FIFO read and write clocks. This port should be used for debug purpose only.	Transmitter Phase Compensation FIFO section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*

Table 3–13. MegaWizard Plug-In Manager Options (Page 4 for Serial RapidIO Mode)

ALT2GXB Setting	Description	Reference
Create rx_coreclk port to connect to the read clock of the RX phase compensation FIFO	This optional input port allows you to clock the read side of the Receiver Phase Compensation FIFO with a non-transceiver PLD clock.	PLD-Transceiver Interface Clocking section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
Create tx_coreclk port to connect to the write clock of the TX phase compensation FIFO	This optional input port allows you to clock the write side of the Transmitter Phase Compensation FIFO with a non-transceiver PLD clock.	PLD-Transceiver Interface Clocking section in the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*

Figure 3–20 shows page 5 of the ALT2GXB MegaWizard Plug-In Manager for Serial RapidIO mode.

Figure 3–20. MegaWizard Plug-In Manager - ALT2GXB (RX Analog/Cal Blk)

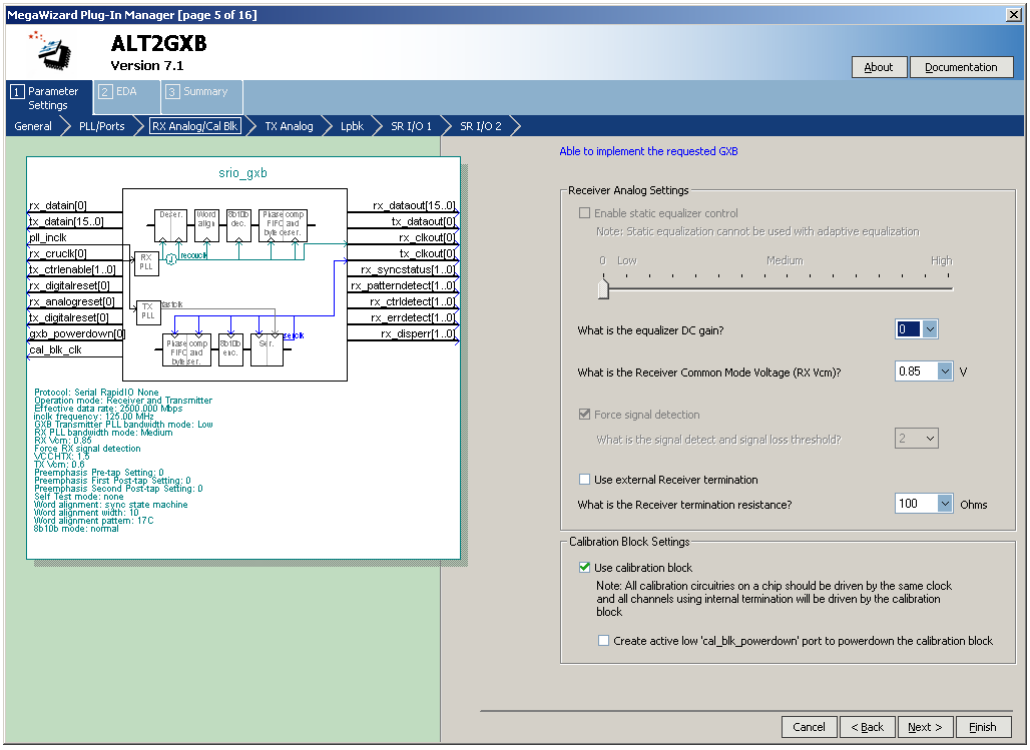


Table 3–14 describes the available options on page 5 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

Table 3–14. MegaWizard Plug-In Manager Options (Page 5 for Serial RapidIO Mode)

ALT2GXB Setting	Description	Reference
What is the Receiver Common Mode Voltage (RX Vcm)?	The receiver common mode voltage is set to 0.85 V.	Receiver Buffer section under Serial RapidIO Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Force signal detection	This option is unavailable in Serial RapidIO mode and is always forced selected.	—
What is the signal detect and signal loss threshold?	This option is unavailable in Serial RapidIO mode as signal detection is forced.	—
Use external receiver termination	This option is available if you want to use an external termination resistor instead of the on-chip termination (OCT). If checked, this option turns off the receiver OCT.	—
What is the receiver termination resistance?	In Serial RapidIO mode, the only supported receiver termination resistance is 100 Ω .	Receiver Buffer section under Serial RapidIO Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Use calibration block	This option allows you to select which instance of ALT2GXB instantiates the calibration block. Only one instance of ALT2GXB is required to instantiate the calibration block.	Calibration Block section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create active low cal_blk_powerdown to power down the calibration block	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in the *Arria GX Device Handbook, volume 2* for information about this port.	Calibration Block section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*

Figure 3–21 shows page 6 of the ALT2GXB MegaWizard Plug-In Manager for Serial RapidIO mode.

Figure 3–21. MegaWizard Plug-In Manager - ALT2GXB (TX Analog)

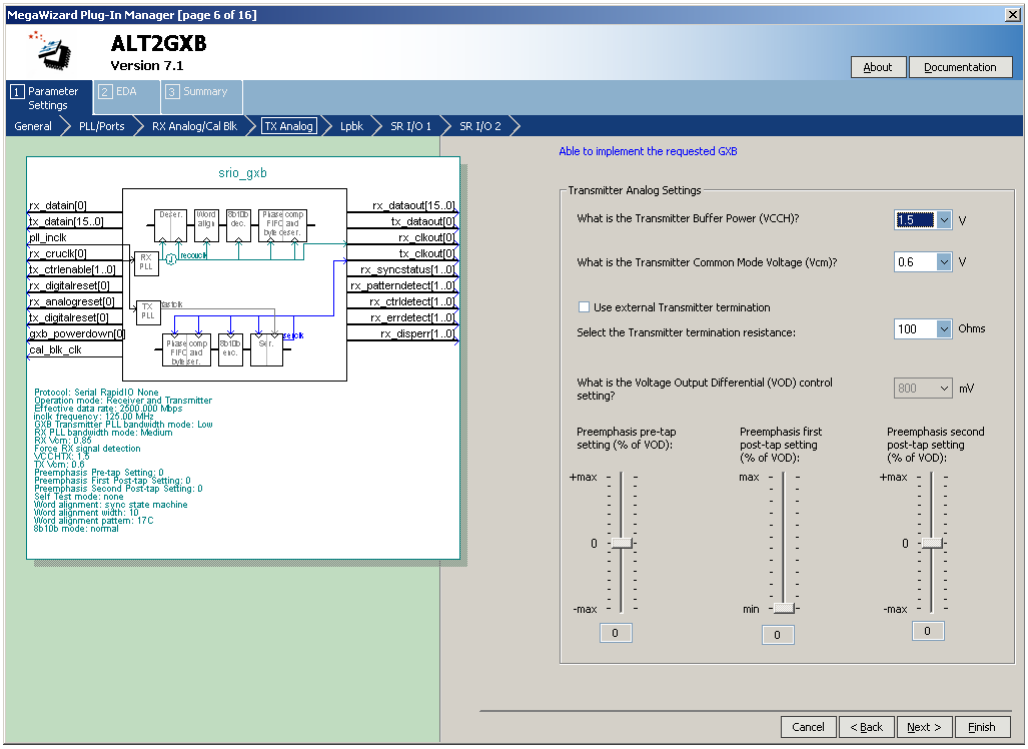


Table 3–15 describes the available options on page 6 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

Table 3–15. MegaWizard Plug-In Manager Options (Page 6 for Serial RapidIO Mode)

ALT2GXB Setting	Description	Reference
What is the Transmitter Buffer Power (VCCH)?	In Serial RapidIO mode, the transmitter buffer power is fixed at 1.5 V. You must connect the VCCH power pins of Serial RapidIO transceiver bank to a 1.5-V power supply. You must select 1.5-V PCML I/O standard for the transmitter data output pins.	Transmitter Buffer section under Serial RapidIO Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
What is the Transmitter Common Mode Voltage (Vcm)?	In Serial RapidIO mode, the transmitter common mode voltage is fixed at 0.6 V.	Transmitter Buffer section under Serial RapidIO mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Use external Transmitter termination	This option is available if you want to use an external termination resistor instead of the on-chip termination (OCT). Checking this option turns off the transmitter OCT.	—
Select the Transmitter termination resistance	In Serial RapidIO mode, the only supported receiver termination resistance is 100 Ω.	Transmitter Buffer section under Serial RapidIO mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
What is the Voltage Output Differential (VOD) control setting?	In Serial RapidIO mode, the only supported peak differential output voltage is 800 mV.	Transmitter Buffer section under Serial RapidIO Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Pre-emphasis pre-tap setting (% of VOD)	In Serial RapidIO mode, this option is unavailable and is fixed to 0.	—

Table 3–15. MegaWizard Plug-In Manager Options (Page 6 for Serial RapidIO Mode)		
ALT2GXB Setting	**Description**	**Reference**
Pre-emphasis first post-tap setting (% of VOD)	In Serial RapidIO mode, this option is unavailable and is fixed to 0.	Transmitter Buffer section under Serial RapidIO Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Pre-emphasis second post-tap setting (% of VOD)	In Serial RapidIO mode, this option is unavailable and is fixed to 0.	—

Figure 3–22 shows page 11 of the ALT2GXB MegaWizard Plug-In Manager for Serial RapidIO mode.

Figure 3–22. MegaWizard Plug-In Manager - ALT2GXB (Loopback)

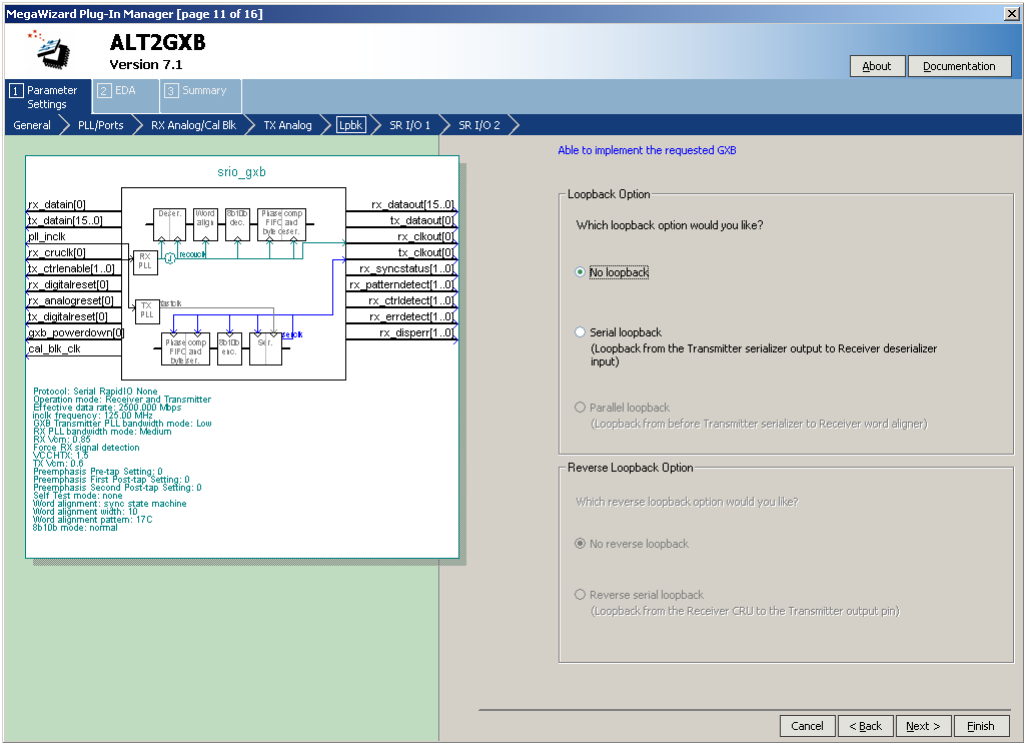


Table 3–16 describes the available options on page 11 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

Table 3–16. MegaWizard Plug-In Manager Options (Page 11 for Serial RapidIO Mode)

ALT2GXB Setting	Description	Reference
Which loopback option would you like?	No loopback and serial loopback options are available in Serial RapidIO mode. No loopback is the default mode. If you select serial loopback, the rx_seriallpbken port is available to control the serial loopback feature dynamically. A 1'b1 enables serial loopback and a 1'b0 disables loopback on a channel-by-channel basis.	Loopback Modes section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*

Figure 3–23 shows page 13 of the ALT2GXB MegaWizard Plug-In Manager for Serial RapidIO mode.

Figure 3–23. MegaWizard Plug-In Manager - ALT2GXB (SR I/O 1)

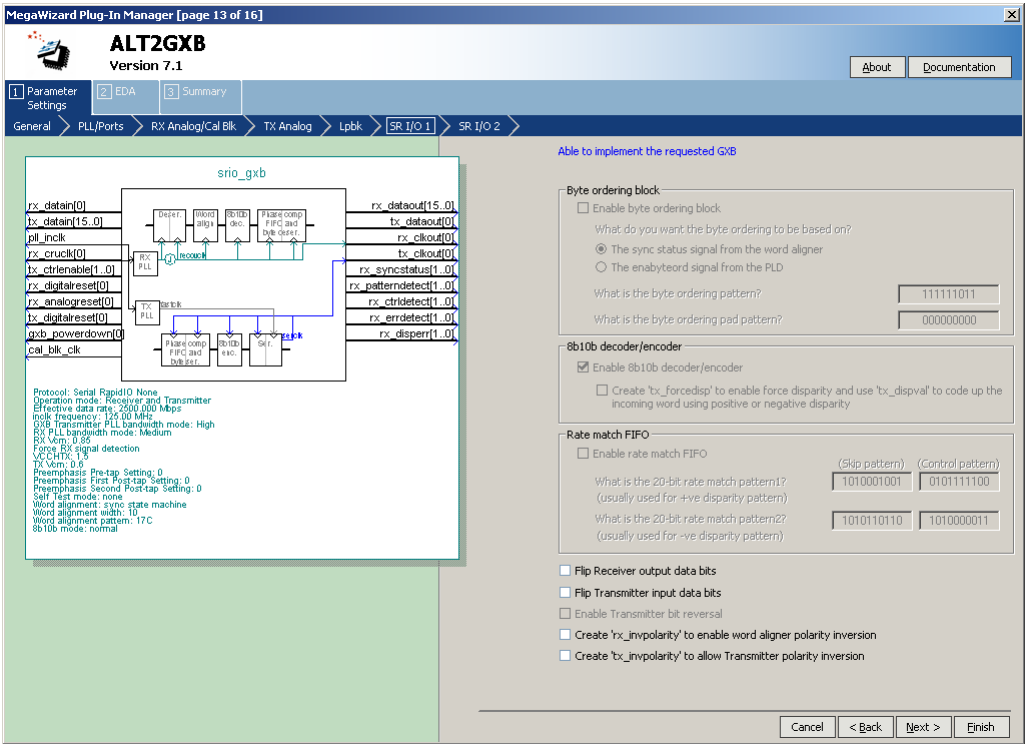


Table 3–17 describes the available options on page 13 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

Table 3–17. MegaWizard Plug-In Manager Options (Page 13 for Serial RapidIO Mode)

ALT2GXB Setting	Description	Reference
Enable byte ordering block	This option is unavailable in Serial RapidIO mode.	—
Enable 8B/10B decoder/encoder	This option is unavailable in Serial RapidIO mode and is always forced selected to enable 8B/10B decoder/encoder.	—
Create tx_forcedisp to enable Force disparity and use tx_dispval to code up the incoming word using positive or negative disparity	This option is unavailable in Serial RapidIO mode.	—
Enable rate match FIFO	This option is unavailable in Serial RapidIO mode as the rate matcher is not supported.	—
Flip Receiver output data bits	This option reverses the bit order of the data at the receiver-PLD interface at a byte level.	—
Flip Transmitter input data bits	This option reverses the bit order of the data bits at the input of the transmitter at a byte level.	—
Enable Transmitter bit reversal	This option is unavailable in Serial RapidIO mode.	—
Create rx_invpolarity to enable word aligner polarity inversion	This optional port allows you to dynamically reverse the polarity of the received data at the input of the word aligner.	Word Aligner section n the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
Create tx_invpolarity to allow Transmitter polarity inversion	This optional port allows you to dynamically reverse the polarity of the data to be transmitted at the transmitter PCS-PMA interface.	8B/10B Encoder section n the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*

Figure 3–24 shows page 14 of the MegaWizard Plug-In Manager for the Serial RapidIO protocol set up.

Figure 3–24. MegaWizard Plug-In Manager - ALT2GXB (SR I/O 2)

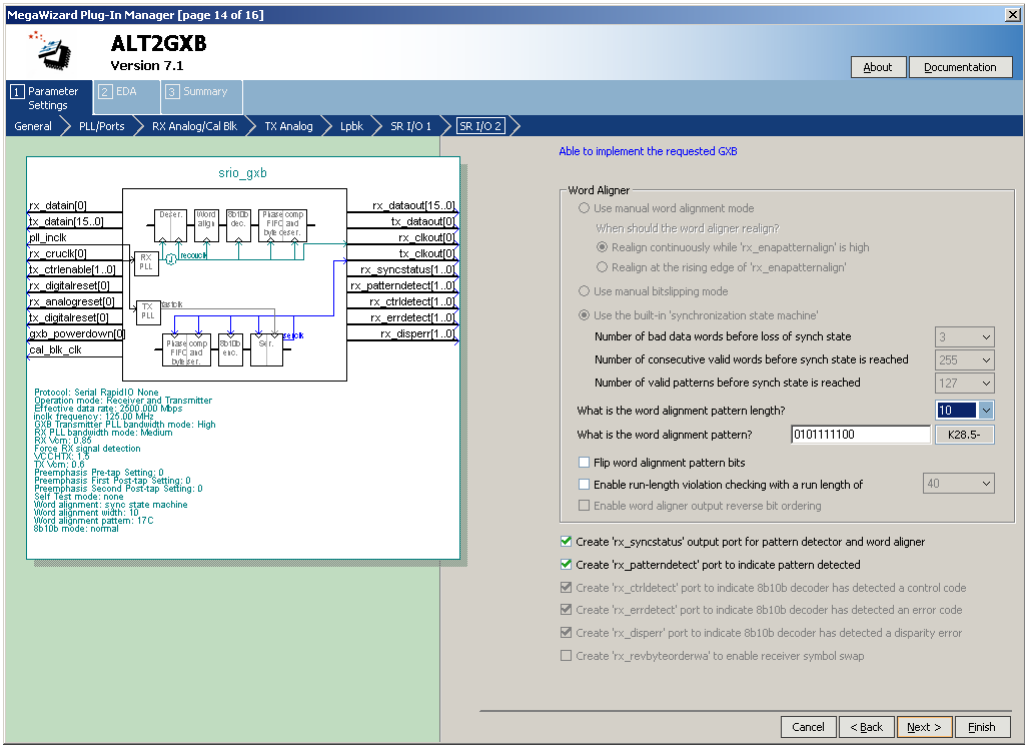


Table 3–18 describes the available options on page 14 of the MegaWizard Plug-In Manager for your ALT2GXB custom megafunction variation.

Table 3–18. MegaWizard Plug-In Manager Options (Page 14 for Serial RapidIO Mode) (Part 1 of 3)

ALT2GXB Setting	Description	Reference
Use manual word alignment mode	This option is unavailable in Serial RapidIO mode.	—
Use manual bit slipping mode.	This option is unavailable in Serial RapidIO mode.	—

Table 3–18. MegaWizard Plug-In Manager Options (Page 14 for Serial RapidIO Mode) (Part 2 of 3)

ALT2GXB Setting	Description	Reference
Use the built-in 'synchronization state machine'	This option is forced selected in Serial RapidIO mode	Word Aligner section under Serial RapidIO Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Number of bad data words before loss of synch state	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Word Aligner section under Serial RapidIO Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Number of consecutive valid words before synch state is reached	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Word Aligner section under Serial RapidIO mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Number of valid patterns before synch state is reached	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Word Aligner section under Serial RapidIO Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
What is the word alignment pattern length?	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Word Aligner section under Serial RapidIO mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
What is the word alignment pattern?	Refer to the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Word Aligner section under Serial RapidIO Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*

Table 3–18. MegaWizard Plug-In Manager Options (Page 14 for Serial RapidIO Mode) (Part 3 of 3)

ALT2GXB Setting	Description	Reference
Enable run-length violation checking with a run length of	This option activates the run-length violation circuit. You can program the run length at which the circuit triggers the `rx_rlv` signal.	Word Aligner section n the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
Enable word aligner output reverse bit ordering	This option is unavailable in Serial RapidIO mode.	—
Create `rx_syncstatus` output port for pattern detector and word aligner	Refer to the *AArria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Word Aligner section under Serial RapidIO Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create `rx_patterndetect` output port to indicate pattern detected	Refer to the *AArria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	Word Aligner section under Serial RapidIO Mode in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create `rx_ctrldetect` output port to indicate 8B/10B decoder has detected a control code	Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	8B/10B Decoder section n the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
Create `rx_errdetect` port to indicate 8B/10B decoder has detected an error code	Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	8B/10B Decoder section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create `rx_disperr` port to indicate 8B/10B decoder has detected a disparity error	Refer to the *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook* for information about this port.	8B/10B Decoder section in the *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*
Create `rx_revbyteorderwa` to enable receiver symbol swap	This option is unavailable in Serial RapidIO mode.	—

Figure 3–25 shows page 15 of the MegaWizard Plug-In Manager for the Serial RapidIO protocol selection. The **Generate simulation model** option creates a behavioral model (**.vo** or **.vho**) of the transceiver instance for third-party simulators. The **Generate a netlist for synthesis area and timing estimation** option creates a netlist file (**.syn**) for third-party synthesis tools.

Figure 3–25. MegaWizard Plug-In Manager - ALT2GXB (EDA)

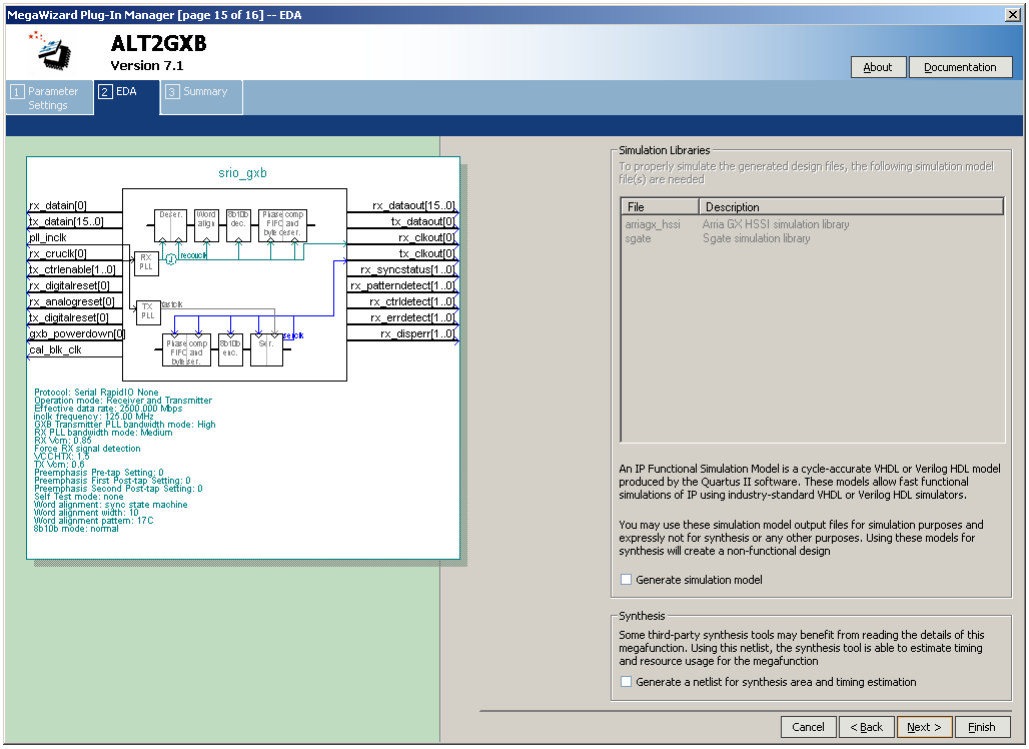
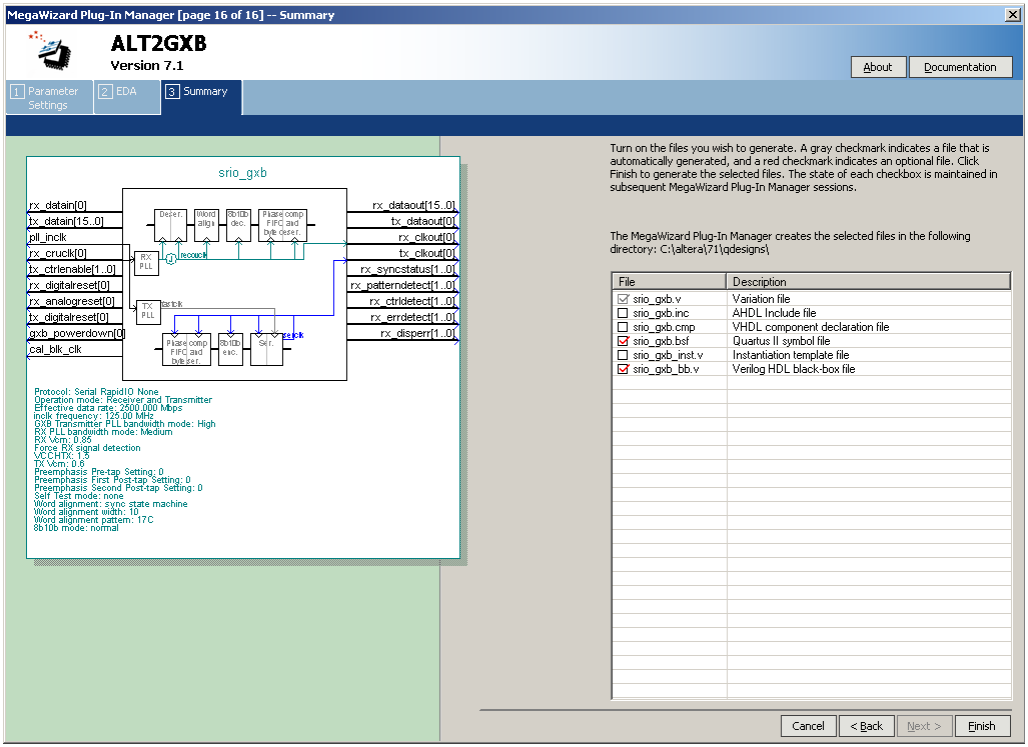


Figure 3–26 shows page 16 (last page) of the MegaWizard Plug-In Manager for the Serial RapidIO protocol set up. You can select optional files on this page. After you make your selections, click **Finish** to generate the files.

Figure 3–26. MegaWizard Plug-In Manager - ALT2GXB (Summary)



Referenced Documents

This chapter references the following documents:

- *Arria GX Architecture* chapter in volume 1 of the *Arria GX Device Handbook*
- *Arria GX Transceiver Architecture* chapter in volume 2 of the *Arria GX Device Handbook*
- *Arria GX Transceiver Protocol Support and Additional Features* chapter in volume 2 of the *Arria GX Device Handbook*

Document Revision History

Table 3–19 shows the revision history for this chapter.

Table 3–19. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
August 2007, v1.2	Added the "Referenced Documents" section.	—
	Minor text edits.	—
June 2007, v1.1	Added GIGE information.	—
May 2007, v1.0	Initial Release.	—

4. Specifications and Additional Information

AGX52004-1.0

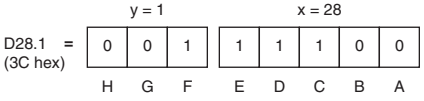
8B/10B Code

This section provides information about the data and control codes for Arria™ GX devices.

Code Notation

The 8B/10B data and control codes are referred to as Dx.y and Kx.y, respectively. The 8-bit byte – H G F E D C B A, where H is the most significant bit (MSB) and A is the significant bit (LSB) – is broken up into two groups, x and y, where x is the five lower bits (E D C B A) and y is the three upper bits (H G F). Figure 4–1 shows the designation for 3C hex.

Figure 4–1. Sample Notation for 3C hex



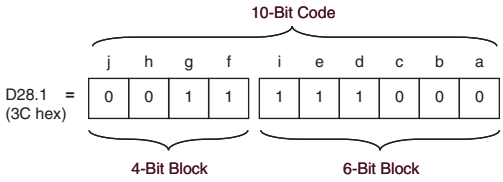
There are 256 Dx.y and 12 Kx.y valid 8-bit codes. These codes have two 10-bit equivalent codes associated with each 8-bit code. The 10-bit codes have either a neutral disparity or a non-neutral disparity. With neutral disparity, two neutral disparity 10-bit codes are associated with an 8-bit code. With non-neutral disparity 10-bit code, a positive and a negative disparity code are associated with the 8-bit code.

The positive disparity 10-bit code is associated in the RD– column. The negative disparity 10-bit code is associated in the RD+ column.

Disparity Calculation

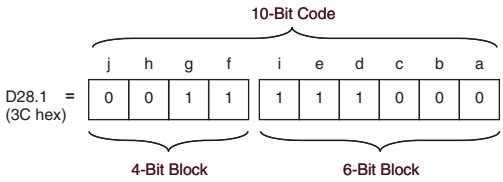
Running disparity is calculated based on the sub-blocks of the 10-bit code. The 10-bit code is divided into two sub-blocks, a 6-bit sub-block (abcdei) and a 4-bit sub-block (fghj), as shown in Figure 4–2.

Figure 4–2. 10-Bit Grouping of 6-bit & 4-Bit Sub-Blocks



The running disparity at the beginning of the 6-bit sub-block is the running disparity at the end of the previous 10-bit code. The running disparity of the 4-bit sub-block is the running disparity at the end of the 6-bit sub-block. The running disparity at the end of the 4-bit sub-block is the running disparity of the 10-bit code (refer to Figure 4–3).

Figure 4–3. Running Disparity Between Sub-Blocks



The running disparity calculation rules are as follows:

- The current running disparity at the end of a sub-block is positive if any of the following is true:
  - The sub-block contains more ones than zeros
  - The 6-bit sub-block is 6'b000111
  - The 4-bit sub-block is 4'b0011
- The current running disparity at the end of a sub-block is negative if any of the following is true:
  - The sub-block contains more zeros than ones
  - The 6-bit sub-block is 6'b111000
  - The 4-bit sub-block is 4'b1100

If those conditions are not met, the running disparity at the end of the sub-block is the same as at the beginning of the sub-block.

Supported Codes

The 8B/10B scheme defines the 12 control codes listed in Table 4–1 for synchronization, alignment, and general application purposes.

Table 4–1. Supported K Codes

K Code	Octal Value	8-Bit Code	10-Bit Code RD–	10-Bit Code RD+
		HGF_EDCBA	abcdei_fghj	
K28.0	1C	8'b000_11100	10'b001111_0100	10'b110000_1011
K28.1	3C	8'b001_11100	10'b001111_1001	10'b110000_0110
K28.2	5C	8'b010_11100	10'b001111_0101	10'b110000_1010
K28.3	7C	8'b011_11100	10'b001111_0011	10'b110000_1100
K28.4	9C	8'b100_11100	10'b001111_0010	10'b110000_1101
K28.5 *(1)*	BC	8'b101_11100	10'b001111_1010	10'b110000_0101
K28.6	DC	8'b110_11100	10'b001111_0110	10'b110000_1001
K28.7	FC	8'b111_11100	10'b001111_1000	10'b110000_0111
K23.7	F7	8'b111_10111	10'b111010_1000	10'b000101_0111
K27.7	FB	8'b111_11011	10'b110110_1000	10'b001001_0111
K29.7	FD	8'b111_11101	10'b101110_1000	10'b010001_0111
K30.7	FE	8'b111_11110	10'b011110_1000	10'b100001_0111

Note to Table 4–1:
(1) K28.5 is a comma code used for word alignment and indicates an IDLE state.

Table 4–2 shows the valid data code-groups.

Table 4–2. Valid Data Code-Groups (Part 1 of 9)

Code-Group Name	Octet Value	Octet Bits	Current RD–	Current RD+
		HGF EDCBA	abcdei fghj	
D0.0	00	000 00000	100111 0100	011000 1011
D1.0	01	000 00001	011101 0100	100010 1011
D2.0	02	000 00010	101101 0100	010010 1011
D3.0	03	000 00011	110001 1011	110001 0100
D4.0	04	000 00100	110101 0100	001010 1011
D5.0	05	000 00101	101001 1011	101001 0100
D6.0	06	000 00110	011001 1011	011001 0100
D7.0	07	000 00111	111000 1011	000111 0100

Table 4–2. Valid Data Code-Groups (Part 2 of 9)

Code-Group Name	Octet Value	Octet Bits	Current RD–	Current RD+
		HGF EDCBA	abcdei fghj	
D8.0	08	000 01000	111001 0100	000110 1011
D9.0	09	000 01001	100101 1011	100101 0100
D10.0	0A	000 01010	010101 1011	010101 0100
D11.0	0B	000 01011	110100 1011	110100 0100
D12.0	0C	000 01100	001101 1011	001101 0100
D13.0	0D	000 01101	101100 1011	101100 0100
D14.0	0E	000 01110	011100 1011	011100 0100
D15.0	0F	000 01111	010111 0100	101000 1011
D16.0	10	000 10000	011011 0100	100100 1011
D17.0	11	000 10001	100011 1011	100011 0100
D18.0	12	000 10010	010011 1011	010011 0100
D19.0	13	000 10011	110010 1011	110010 0100
D20.0	14	000 10100	001011 1011	001011 0100
D21.0	15	000 10101	101010 1011	101010 0100
D22.0	16	000 10110	011010 1011	011010 0100
D23.0	17	000 10111	111010 0100	000101 1011
D24.0	18	000 11000	110011 0100	001100 1011
D25.0	19	000 11001	100110 1011	100110 0100
D26.0	1A	000 11010	010110 1011	010110 0100
D27.0	1B	000 11011	110110 0100	001001 1011
D28.0	1C	000 11100	001110 1011	001110 0100
D29.0	1D	000 11101	101110 0100	010001 1011
D30.0	1E	000 11110	011110 0100	100001 1011
D31.0	1F	000 11111	101011 0100	010100 1011
D0.1	20	001 00000	100111 1001	011000 1001
D1.1	21	001 00001	011101 1001	100010 1001
D2.1	22	001 00010	101101 1001	010010 1001
D3.1	23	001 00011	110001 1001	110001 1001
D4.1	24	001 00100	110101 1001	001010 1001
D5.1	25	001 00101	101001 1001	101001 1001
D6.1	26	001 00110	011001 1001	011001 1001
D7.1	27	001 00111	111000 1001	000111 1001
D8.1	28	001 01000	111001 1001	000110 1001

Table 4–2. Valid Data Code-Groups (Part 3 of 9)

Code-Group Name	Octet Value	Octet Bits	Current RD–	Current RD+
		HGF EDCBA	abcdei fghj	
D9.1	29	001 01001	100101 1001	100101 1001
D10.1	2A	001 01010	010101 1001	010101 1001
D11.1	2B	001 01011	110100 1001	110100 1001
D12.1	2C	001 01100	001101 1001	001101 1001
D13.1	2D	001 01101	101100 1001	101100 1001
D14.1	2E	001 01110	011100 1001	011100 1001
D15.1	2F	001 01111	010111 1001	101000 1001
D16.1	30	001 10000	011011 1001	100100 1001
D17.1	31	001 10001	100011 1001	100011 1001
D18.1	32	001 10010	010011 1001	010011 1001
D19.1	33	001 10011	110010 1001	110010 1001
D20.1	34	001 10100	001011 1001	001011 1001
D21.1	35	001 10101	101010 1001	101010 1001
D22.1	36	001 10110	011010 1001	011010 1001
D23.1	37	001 10111	111010 1001	000101 1001
D24.1	38	001 11000	110011 1001	001100 1001
D25.1	39	001 11001	100110 1001	100110 1001
D26.1	3A	001 11010	010110 1001	010110 1001
D27.1	3B	001 11011	110110 1001	001001 1001
D28.1	3C	001 11100	001110 1001	001110 1001
D29.1	3D	001 11101	101110 1001	010001 1001
D30.1	3E	001 11110	011110 1001	100001 1001
D31.1	3F	001 11111	101011 1001	010100 1001
D0.2	40	010 00000	100111 0101	011000 0101
D1.2	41	010 00001	011101 0101	100010 0101
D2.2	42	010 00010	101101 0101	010010 0101
D3.2	43	010 00011	110001 0101	110001 0101
D4.2	44	010 00100	110101 0101	001010 0101
D5.2	45	010 00101	101001 0101	101001 0101
D6.2	46	010 00110	011001 0101	011001 0101
D7.2	47	010 00111	111000 0101	000111 0101
D8.2	48	010 01000	111001 0101	000110 0101
D9.2	49	010 01001	100101 0101	100101 0101

Table 4–2. Valid Data Code-Groups (Part 4 of 9)

Code-Group Name	Octet Value	Octet Bits	Current RD–	Current RD+
		HGF EDCBA	abcdei fghj	
D10.2	4A	010 01010	010101 0101	010101 0101
D11.2	4B	010 01011	110100 0101	110100 0101
D12.2	4C	010 01100	001101 0101	001101 0101
D13.2	4D	010 01101	101100 0101	101100 0101
D14.2	4E	010 01110	011100 0101	011100 0101
D15.2	4F	010 01111	010111 0101	101000 0101
D16.2	50	010 10000	011011 0101	100100 0101
D17.2	51	010 10001	100011 0101	100011 0101
D18.2	52	010 10010	010011 0101	010011 0101
D19.2	53	010 10011	110010 0101	110010 0101
D20.2	54	010 10100	001011 0101	001011 0101
D21.2	55	010 10101	101010 0101	101010 0101
D22.2	56	010 10110	011010 0101	011010 0101
D23.2	57	010 10111	111010 0101	000101 0101
D24.2	58	010 11000	110011 0101	001100 0101
D25.2	59	010 11001	100110 0101	100110 0101
D26.2	5A	010 11010	010110 0101	010110 0101
D27.2	5B	010 11011	110110 0101	001001 0101
D28.2	5C	010 11100	001110 0101	001110 0101
D29.2	5D	010 11101	101110 0101	010001 0101
D30.2	5E	010 11110	011110 0101	100001 0101
D31.2	5F	010 11111	101011 0101	010100 0101
D0.3	60	011 00000	100111 0011	011000 1100
D1.3	61	011 00001	011101 0011	100010 1100
D2.3	62	011 00010	101101 0011	010010 1100
D3.3	63	011 00011	110001 1100	110001 0011
D4.3	64	011 00100	110101 0011	001010 1100
D5.3	65	011 00101	101001 1100	101001 0011
D6.3	66	011 00110	011001 1100	011001 0011
D7.3	67	011 00111	111000 1100	000111 0011
D8.3	68	011 01000	111001 0011	000110 1100
D9.3	69	011 01001	100101 1100	100101 0011
D10.3	6A	011 01010	010101 1100	010101 0011

Table 4–2. Valid Data Code-Groups (Part 5 of 9)

Code-Group Name	Octet Value	Octet Bits	Current RD–	Current RD+
		HGF EDCBA	abcdei fghj	
D11.3	6B	011 01011	110100 1100	110100 0011
D12.3	6C	011 01100	001101 1100	001101 0011
D13.3	6D	011 01101	101100 1100	101100 0011
D14.3	6E	011 01110	011100 1100	011100 0011
D15.3	6F	011 01111	010111 0011	101000 1100
D16.3	70	011 10000	011011 0011	100100 1100
D17.3	71	011 10001	100011 1100	100011 0011
D18.3	72	011 10010	010011 1100	010011 0011
D19.3	73	011 10011	110010 1100	110010 0011
D20.3	74	011 10100	001011 1100	001011 0011
D21.3	75	011 10101	101010 1100	101010 0011
D22.3	76	011 10110	011010 1100	011010 0011
D23.3	77	011 10111	111010 0011	000101 1100
D24.3	78	011 11000	110011 0011	001100 1100
D25.3	79	011 11001	100110 1100	100110 0011
D26.3	7A	011 11010	010110 1100	010110 0011
D27.3	7B	011 11011	110110 0011	001001 1100
D28.3	7C	011 11100	001110 1100	001110 0011
D29.3	7D	011 11101	101110 0011	010001 1100
D30.3	7E	011 11110	011110 0011	100001 1100
D31.3	7F	011 11111	101011 0011	010100 1100
D0.4	80	100 00000	100111 0010	011000 1101
D1.4	81	100 00001	011101 0010	100010 1101
D2.4	82	100 00010	101101 0010	010010 1101
D3.4	83	100 00011	110001 1101	110001 0010
D4.4	84	100 00100	110101 0010	001010 1101
D5.4	85	100 00101	101001 1101	101001 0010
D6.4	86	100 00110	011001 1101	011001 0010
D7.4	87	100 00111	111000 1101	000111 0010
D8.4	88	100 01000	111001 0010	000110 1101
D9.4	89	100 01001	100101 1101	100101 0010
D10.4	8A	100 01010	010101 1101	010101 0010
D11.4	8B	100 01011	110100 1101	110100 0010

Table 4–2. Valid Data Code-Groups (Part 6 of 9)

Code-Group Name	Octet Value	Octet Bits	Current RD–	Current RD+
		HGF EDCBA	abcdei fghj	
D12.4	8C	100 01100	001101 1101	001101 0010
D13.4	8D	100 01101	101100 1101	101100 0010
D14.4	8E	100 01110	011100 1101	011100 0010
D15.4	8F	100 01111	010111 0010	101000 1101
D16.4	90	100 10000	011011 0010	100100 1101
D17.4	91	100 10001	100011 1101	100011 0010
D18.4	92	100 10010	010011 1101	010011 0010
D19.4	93	100 10011	110010 1101	110010 0010
D20.4	94	100 10100	001011 1101	001011 0010
D21.4	95	100 10101	101010 1101	101010 0010
D22.4	96	100 10110	011010 1101	011010 0010
D23.4	97	100 10111	111010 0010	000101 1101
D24.4	98	100 11000	110011 0010	001100 1101
D25.4	99	100 11001	100110 1101	100110 0010
D26.4	9A	100 11010	010110 1101	010110 0010
D27.4	9B	100 11011	110110 0010	001001 1101
D28.4	9C	100 11100	001110 1101	001110 0010
D29.4	9D	100 11101	101110 0010	010001 1101
D30.4	9E	100 11110	011110 0010	100001 1101
D31.4	9F	100 11111	101011 0010	010100 1101
D0.5	A0	101 00000	100111 1010	011000 1010
D1.5	A1	101 00001	011101 1010	100010 1010
D2.5	A2	101 00010	101101 1010	010010 1010
D3.5	A3	101 00011	110001 1010	110001 1010
D4.5	A4	101 00100	110101 1010	001010 1010
D5.5	A5	101 00101	101001 1010	101001 1010
D6.5	A6	101 00110	011001 1010	011001 1010
D7.5	A7	101 00111	111000 1010	000111 1010
D8.5	A8	101 01000	111001 1010	000110 1010
D9.5	A9	101 01001	100101 1010	100101 1010
D10.5	AA	101 01010	010101 1010	010101 1010
D11.5	AB	101 01011	110100 1010	110100 1010
D12.5	AC	101 01100	001101 1010	001101 1010

Table 4–2. Valid Data Code-Groups (Part 7 of 9)

Code-Group Name	Octet Value	Octet Bits	Current RD–	Current RD+
		HGF EDCBA	abcdei fghj	
D13.5	AD	101 01101	101100 1010	101100 1010
D14.5	AE	101 01110	011100 1010	011100 1010
D15.5	AF	101 01111	010111 1010	101000 1010
D16.5	B0	101 10000	011011 1010	100100 1010
D17.5	B1	101 10001	100011 1010	100011 1010
D18.5	B2	101 10010	010011 1010	010011 1010
D19.5	B3	101 10011	110010 1010	110010 1010
D20.5	B4	101 10100	001011 1010	001011 1010
D21.5	B5	101 10101	101010 1010	101010 1010
D22.5	B6	101 10110	011010 1010	011010 1010
D23.5	B7	101 10111	111010 1010	000101 1010
D24.5	B8	101 11000	110011 1010	001100 1010
D25.5	B9	101 11001	100110 1010	100110 1010
D26.5	BA	101 11010	010110 1010	010110 1010
D27.5	BB	101 11011	110110 1010	001001 1010
D28.5	BC	101 11100	001110 1010	001110 1010
D29.5	BD	101 11101	101110 1010	010001 1010
D30.5	BE	101 11110	011110 1010	100001 1010
D31.5	BF	101 11111	101011 1010	010100 1010
D0.6	C0	110 00000	100111 0110	011000 0110
D1.6	C1	110 00001	011101 0110	100010 0110
D2.6	C2	110 00010	101101 0110	010010 0110
D3.6	C3	110 00011	110001 0110	110001 0110
D4.6	C4	110 00100	110101 0110	001010 0110
D5.6	C5	110 00101	101001 0110	101001 0110
D6.6	C6	110 00110	011001 0110	011001 0110
D7.6	C7	110 00111	111000 0110	000111 0110
D8.6	C8	110 01000	111001 0110	000110 0110
D9.6	C9	110 01001	100101 0110	100101 0110
D10.6	CA	110 01010	010101 0110	010101 0110
D11.6	CB	110 01011	110100 0110	110100 0110
D12.6	CC	110 01100	001101 0110	001101 0110
D13.6	CD	110 01101	101100 0110	101100 0110

Table 4–2. Valid Data Code-Groups (Part 8 of 9)

Code-Group Name	Octet Value	Octet Bits	Current RD–	Current RD+
		HGF EDCBA	abcdei fghj	
D14.6	CE	110 01110	011100 0110	011100 0110
D15.6	CF	110 01111	010111 0110	101000 0110
D16.6	D0	110 10000	011011 0110	100100 0110
D17.6	D1	110 10001	100011 0110	100011 0110
D18.6	D2	110 10010	010011 0110	010011 0110
D19.6	D3	110 10011	110010 0110	110010 0110
D20.6	D4	110 10100	001011 0110	001011 0110
D21.6	D5	110 10101	101010 0110	101010 0110
D22.6	D6	110 10110	011010 0110	011010 0110
D23.6	D7	110 10111	111010 0110	000101 0110
D24.6	D8	110 11000	110011 0110	001100 0110
D25.6	D9	110 11001	100110 0110	100110 0110
D26.6	DA	110 11010	010110 0110	010110 0110
D27.6	DB	110 11011	110110 0110	001001 0110
D28.6	DC	110 11100	001110 0110	001110 0110
D29.6	DD	110 11101	101110 0110	010001 0110
D30.6	DE	110 11110	011110 0110	100001 0110
D31.6	DF	110 11111	101011 0110	010100 0110
D0.7	E0	111 00000	100111 0001	011000 1110
D1.7	E1	111 00001	011101 0001	100010 1110
D2.7	E2	111 00010	101101 0001	010010 1110
D3.7	E3	111 00011	110001 1110	110001 0001
D4.7	E4	111 00100	110101 0001	001010 1110
D5.7	E5	111 00101	101001 1110	101001 0001
D6.7	E6	111 00110	011001 1110	011001 0001
D7.7	E7	111 00111	111000 1110	000111 0001
D8.7	E8	111 01000	111001 0001	000110 1110
D9.7	E9	111 01001	100101 1110	100101 0001
D10.7	EA	111 01010	010101 1110	010101 0001
D11.7	EB	111 01011	110100 1110	110100 1000
D12.7	EC	111 01100	001101 1110	001101 0001
D13.7	ED	111 01101	101100 1110	101100 1000
D14.7	EE	111 01110	011100 1110	011100 1000

Table 4–2. Valid Data Code-Groups (Part 9 of 9)

Code-Group Name	Octet Value	Octet Bits	Current RD–	Current RD+
		HGF EDCBA	abcdei fghj	
D15.7	EF	111 01111	010111 0001	101000 1110
D16.7	F0	111 10000	011011 0001	100100 1110
D17.7	F1	111 10001	100011 0111	100011 0001
D18.7	F2	111 10010	010011 0111	010011 0001
D19.7	F3	111 10011	110010 1110	110010 0001
D20.7	F4	111 10100	001011 0111	001011 0001
D21.7	F5	111 10101	101010 1110	101010 0001
D22.7	F6	111 10110	011010 1110	011010 0001
D23.7	F7	111 10111	111010 0001	000101 1110
D24.7	F8	111 11000	110011 0001	001100 1110
D25.7	F9	111 11001	100110 1110	100110 0001
D26.7	FA	111 11010	010110 1110	010110 0001
D27.7	FB	111 11011	110110 0001	001001 1110
D28.7	FC	111 11100	001110 1110	001110 0001
D29.7	FD	111 11101	101110 0001	010001 1110
D30.7	FE	111 11110	011110 0001	100001 1110
D31.7	FF	111 11111	101011 0001	010100 1110

Document Revision History

Table 4–3 shows the revision history for this document.

Table 4–3. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
May 2007 v1.0	Initial Release	N/A