1. Transceiver Architecture in Arria II Devices

AIIGX52001-4.6

This chapter describes all available modules in the Arria® II GX and GZ transceiver architecture and describes how these modules are used in the protocols shown in Table 1–1. In addition, this chapter lists the available test modes, dynamic reconfiguration, and ALTGX port names.

Arria II GX and GZ devices provide up to 24 full-duplex clock data recovery-based (CDR) transceivers with physical coding sublayer (PCS) and physical medium attachment (PMA), and support the serial protocols listed in Table 1–1 and Table 1–2.

Table 1–1 lists the serial protocols for Arria II GX devices.

Table 1–1. Serial Protocols for Arria II GX Devices

Protocol	Description
PCI Express® (PIPE) (PCIe®)	Gen1, 2.5 Gbps
Serial RapidIO® (SRIO)	1.25 Gbps, 2.5 Gbps, and 3.125 Gbps
Serial ATA (SATA)/ Serial Attached SCSI (SAS)	■ SATA I, 1.5 Gbps ■ SATA II, 3.0 Gbps ■ SATA III, 6.0 Gbps ■ SAS, 1.5 Gbps and 3.0 Gbps
Serial Digital Interface (SDI)	■ HD-SDI, 1.485 Gbps and 1.4835 Gbps ■ 3G-SDI, 2.97 Gbps and 2.967 Gbps
ASI	270 Mbps
Common Public Radio Interface (CPRI)	614.4 Mbps, 1228.8 Mbps, 2457.6 Mbps, 3072 Mbps, 4915.2 Mbps, and 6144 Mbps
OBSAI	768 Mbps, 1536 Mbps, 3072 Mbps, and 6144 Mbps
Gigabit Ethernet (GbE)	1.25 Gbps
XAUI	3.125 Gbps to 3.75 Gbps for HiGig/HiGig+ support
SONET/SDH	■ OC-12 (622 Mbps) ■ OC-48 (2.488 Gbps)
GPON	1.244 uplink and 2.488 downlink
SerialLite II	0.6 Gbps to 3.75 Gbps
Interlaken	—
CEI	—

Table 1–2 lists the serial protocols for Arria II GZ devices.

Table 1–2. Serial Protocols for Arria II GZ Devices

Protocol	Description
PCIe	Gen2, 5.0 Gbps
SRIO	1.25 Gbps, 2.5 Gbps, and 3.125 Gbps
Serial ATA (SATA)/ Serial Attached SCSI (SAS)	■ SATA I, 1.5 Gbps ■ SATA II, 3.0 Gbps ■ SATA III, 6.0 Gbps ■ SAS, 1.5 Gbps and 3.0 Gbps
Serial Digital Interface (SDI)	■ HD-SDI, 1.485 Gbps and 1.4835 Gbps ■ 3G-SDI, 2.97 Gbps and 2.967 Gbps
ASI	270 Mbps
Common Public Radio Interface (CPRI)	614.4 Mbps, 1228.8 Mbps, 2457.6 Mbps, 3072 Mbps, 4915.2 Mbps, and 6144 Mbps
OBSAI	768 Mbps, 1536 Mbps, 3072 Mbps, and 6144 Mbps
Gigabit Ethernet (GbE)	1.25 Gbps
XAUI	3.125 Gbps to 3.75 Gbps for HiGig/HiGig+ support
SONET/SDH	■ OC-12 (622 Mbps) ■ OC-48 (2.488 Gbps) ■ OC-96 (4.976 Gbps)
GPON	1.244 uplink and 2.488 downlink
Interlaken	40G with 10 channels at 6.375 Gbps
CEI	6.375 Gbps

You can implement these protocols through the ALTGX MegaWizard™ Plug-In Manager, which also offers the highly flexible Basic functional mode to implement proprietary serial protocols.

Transceiver Block Overview

Arria II GX devices offer two to four transceiver blocks per device while Arria II GZ devices offer up to six transceiver blocks. Each block consists of four fully-duplex (transmitter and receiver) channels, located on the left side of the device (in a die-top view).

Figure 1–1 shows the die-top view of the transceiver block locations in Arria II GX devices.

Figure 1–1. Transceiver Channels for Arria II GX Devices

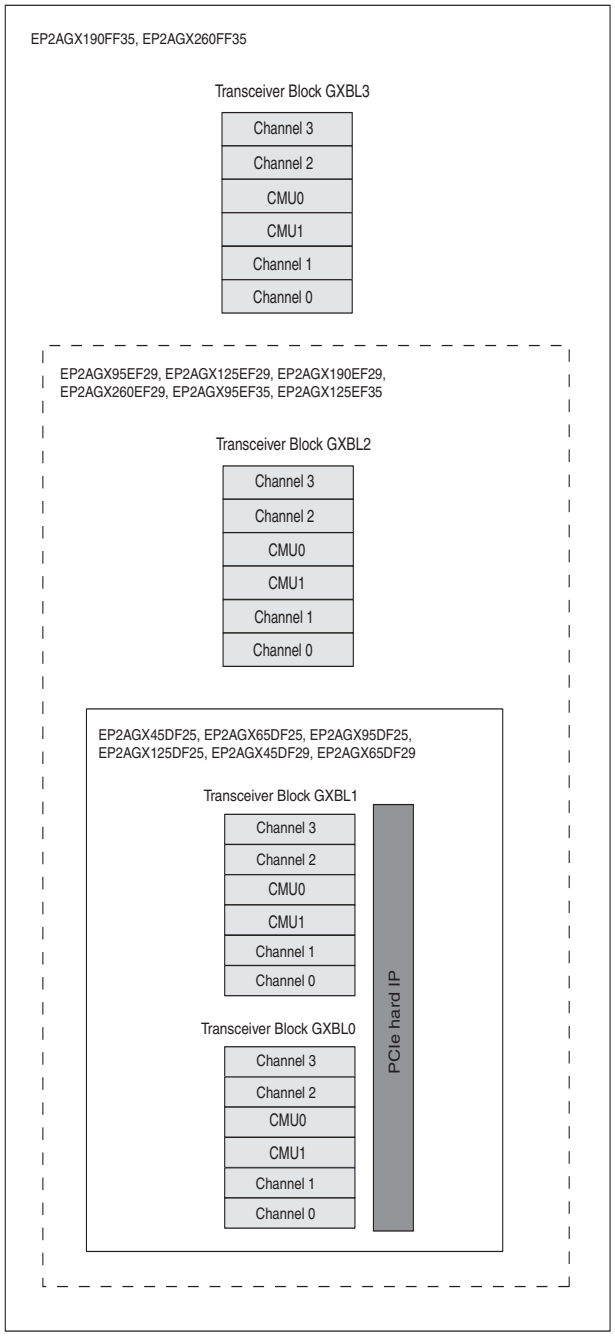


Figure 1–2 shows the die top view of the transceiver block locations in Arria II GZ devices.

Figure 1–2. Transceiver Channels for Arria II GZ Devices

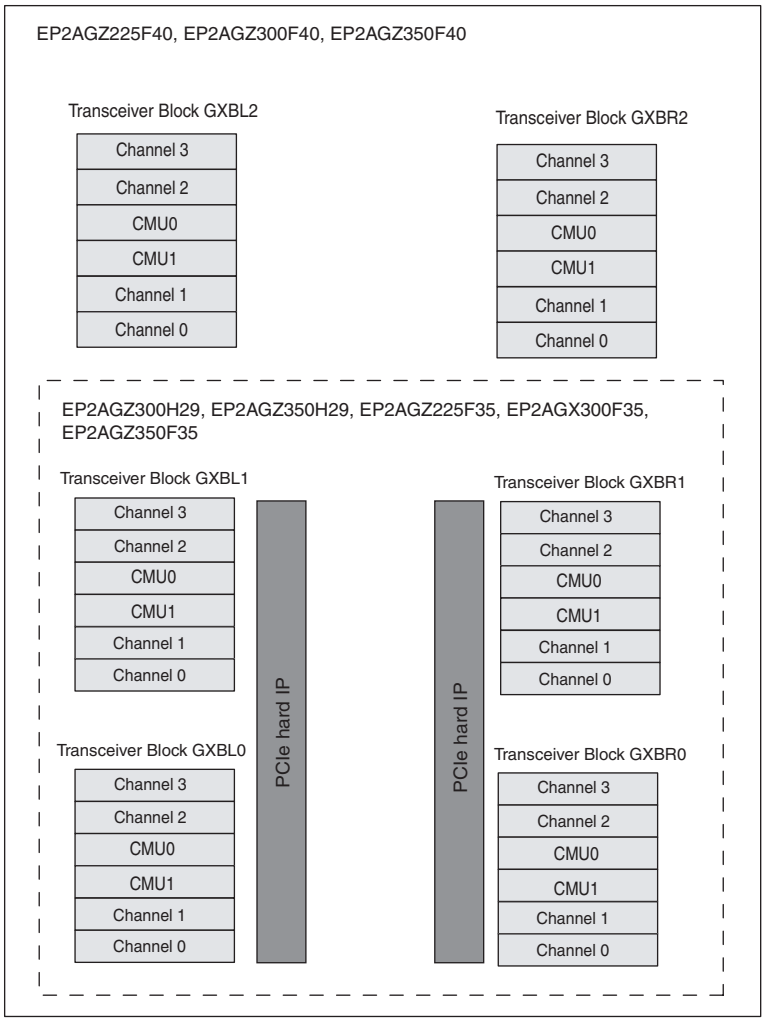
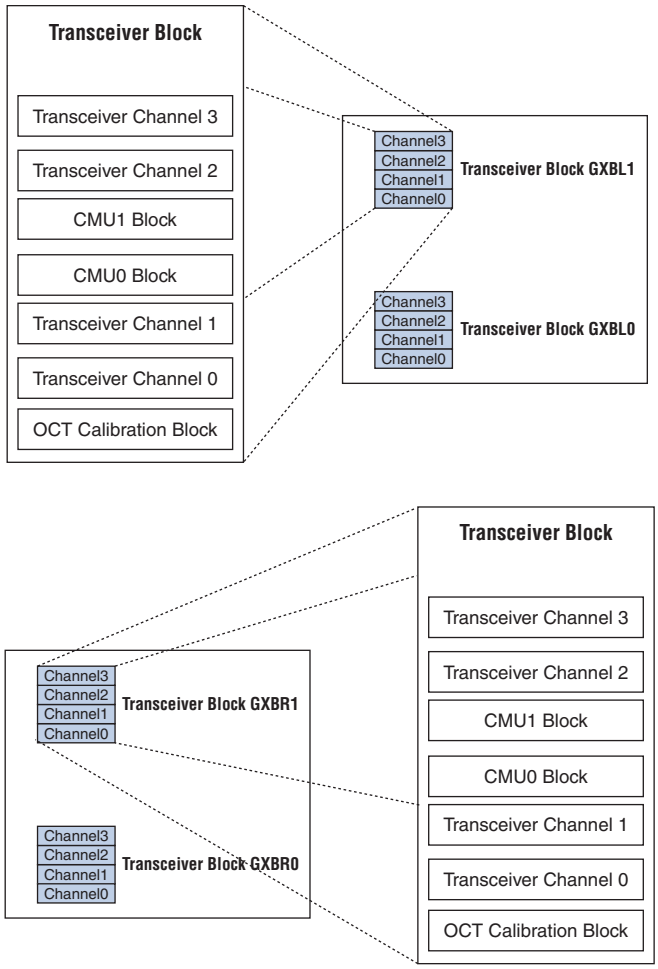


Figure 1–3 shows the block diagram of the transceiver block architecture for Arria II GX and GZ devices.

Figure 1–3. Top-Level View of a Transceiver Block for Arria II GX and GZ Devices



The following sections describe all the modules of the transceiver blocks. The input and output ports of these modules are described in the module sections, and are listed in the "Transceiver Port List" on page 1–94.

Clock Multiplier Units (CMU)

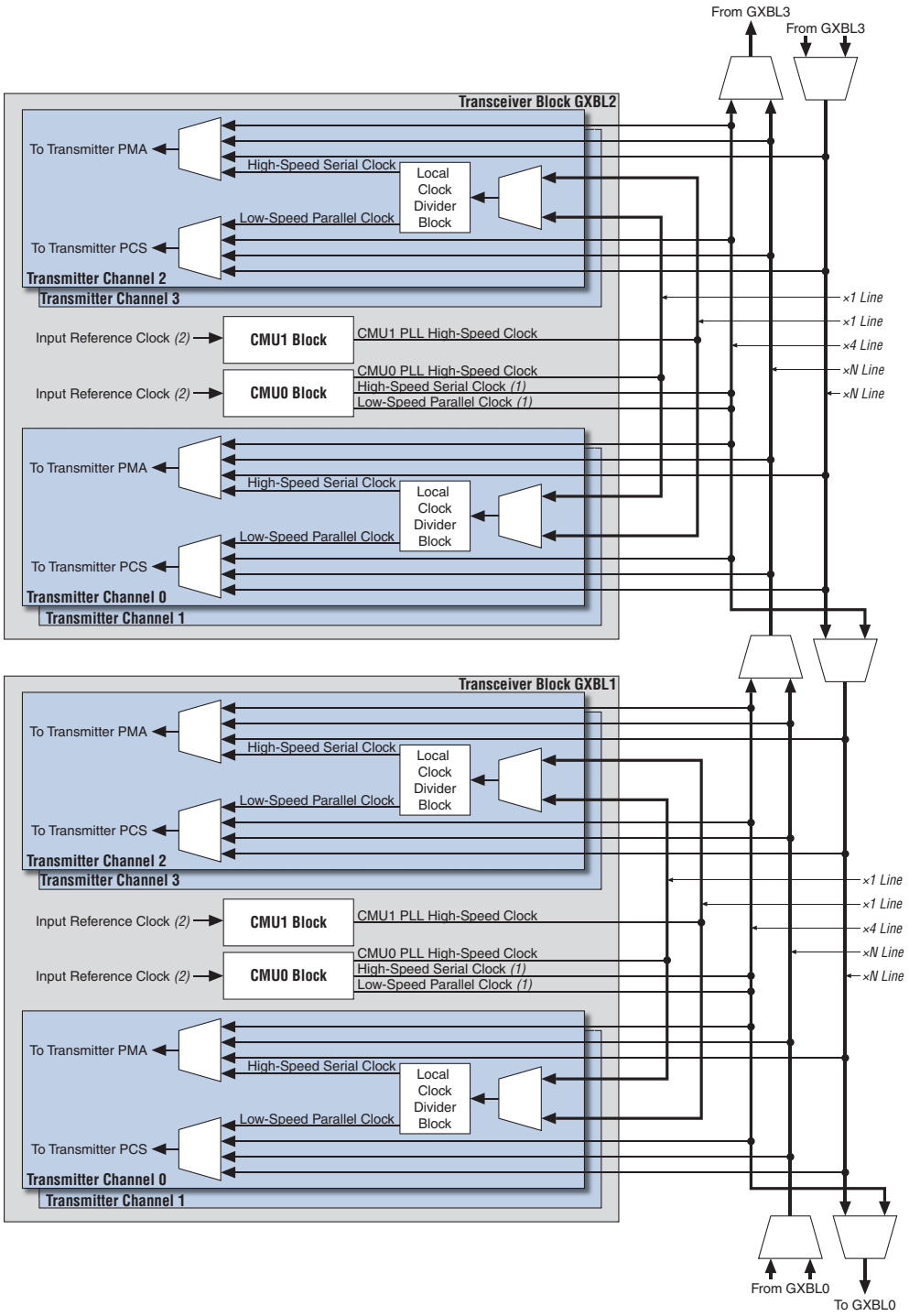
Each transceiver block contains two CMU blocks, which contain a CMU phase-locked loop (PLL) that provides clocks to all the transmitter channels in the same transceiver block. These two CMU blocks can provide two independent high-speed clocks per transceiver block.

The CMU PLL is also known as the TX PLL.

The CMU PLLs in `CMU0` and `CMU1` are identical and each transmitter channel in the transceiver block can receive a high-speed clock from either of the two CMU PLLs. However, the `CMU0` block has an additional clock divider after the `CMU0` PLL to support bonded functional modes where multiple channels share a common clock to reduce skew between the channels. With the ALTGX MegaWizard Plug-In Manager, you can select the bonded functional modes used in ×4 Basic, PCIe, and XAUI.

Figure 1–4 shows a top-level block diagram of the connections between the CMU blocks and the transceiver channels.

Figure 1–4. Top-Level Diagram of CMU Block Connections in a Transceiver Block



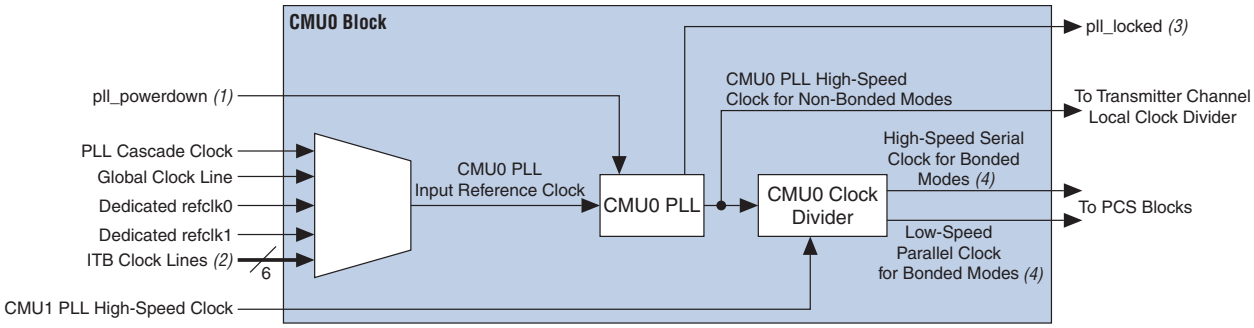
Notes to Figure 1–4:

(1) Clocks provided to support bonded channel functional mode.

(2) For more information, refer to the *Transceiver Clocking for Arria II Devices* chapter.

Figure 1–5 and Figure 1–6 show the top-level block diagram of CMU0 and CMU1 blocks, respectively.

Figure 1–5. CMU0 Block Diagram



Notes to Figure 1–5:

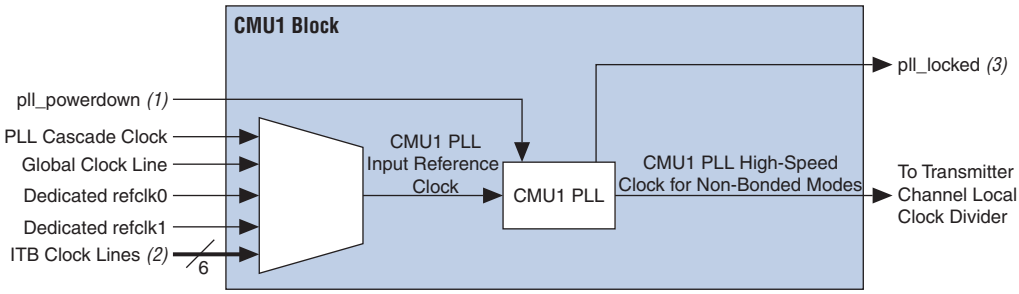
(1) Although each CMU PLL has its own pll_powerdown port, the ALTGX MegaWizard Plug-In Manager instantiation provides only one port per transceiver block. This port power downs one or both CMU PLLs (if used).

(2) The inter-transceiver block (ITB) clock lines shown are the maximum value. The actual number of ITB lines in your device depends on the number of transceiver blocks on one side of the device.

(3) There is one pll_locked signal per CMU PLL.

(4) Used in ×4, ×8, and XAUI functional modes. In ×8 functional mode, only the CMU0 channel of the master transceiver block provides clock output to all eight transceiver channels configured in PCIe functional mode.

Figure 1–6. CMU1 Block Diagram



Notes to Figure 1–6:

(1) Although each CMU PLL has its own pll_powerdown port, the ALTGX MegaWizard Plug-In Manager instantiation provides only one port per transceiver block. This port power downs one or both CMU PLLs (if used).

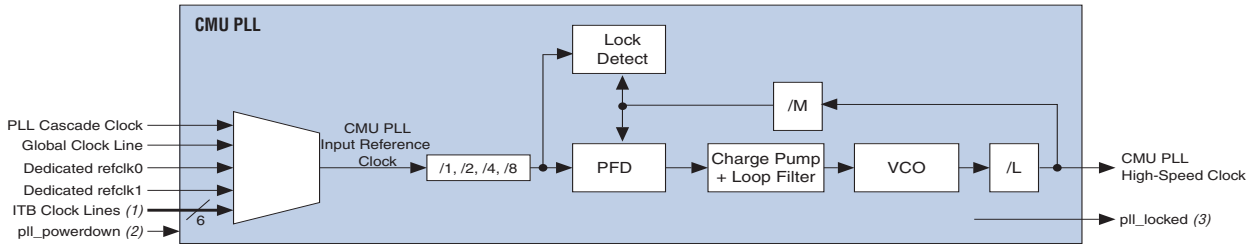
(2) The ITB clock lines shown are the maximum value. The actual number of ITB lines in your device depends on the number of transceiver blocks on one side of the device.

(3) There is one pll_locked signal per CMU PLL.

CMU PLL

Figure 1–7 shows the block diagram of the CMU PLL.

Figure 1–7. Diagram of the CMU PLL



Notes to Figure 1–7:

(1) The ITB clock lines shown are the maximum value. The actual number of ITB lines in your device depends on the number of transceiver blocks on one side of the device.

(2) Although each CMU PLL has its own `pll_powerdown` port, the ALTGX MegaWizard Plug-In Manager instantiation provides only one port per transceiver block. This port power downs one or both CMU PLLs (if used).

(3) There is one `pll_locked` signal per CMU PLL.

For more information about input reference clocks, refer to the "CMU PLL and Receiver CDR Input Reference Clocks" section of the *Transceiver Clocking in Arria II Devices* chapter.

The phase frequency detector (PFD) in the CMU PLL tracks the voltage-controlled oscillator (VCO) output with the input reference clock. This VCO runs at half the serial data rate. The CMU PLL generates the high-speed clock from the input reference clock through the two divider blocks (/M and /L) in the feedback path. Table 1–3 lists the available /M and /L settings, which are set automatically in the Quartus® II software, based on the input reference clock frequency and serial data rate.

Table 1–3. Multiplier Block Heading to Clock Divider for Arria II Devices

Multiplier Block	Available Values
/M	1, 4, 5, 8, 10, 16, 20, 25
/L	1, 2, 4

You can set the PLL bandwidth in the ALTGX megafunction.

The high-speed clock output from the CMU PLL is forwarded to the `CMU0` clock divider block in bonded functional modes and the transmitter channel local clock divider block in non-bonded functional modes. The output of either clock divider block provides clocks for the PCS and PMA blocks.

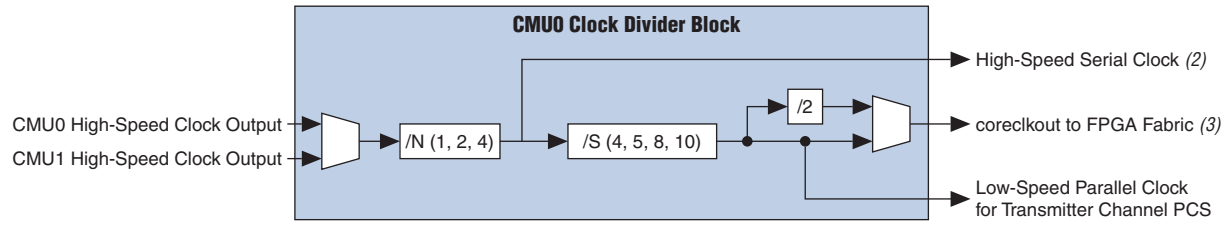
For more information about using two CMU PLLs to configure multiple transmitter channels, refer to the *Configuring Multiple Protocols and Data Rates in Arria II Devices* chapter.

CMU0 Clock Divider

The clock divider is only available only in the CMU0 block and is used in bonded functional modes.

Figure 1–8 shows a diagram of the CMU0 clock divider block.

Figure 1–8. CMU0 Clock Divider Block ***(Note 1)***



Notes to Figure 1–8:

(1) The Quartus II software automatically selects all the divider settings based on the input clock frequency, data rate, deserialization width, and channel width settings.

(2) The high-speed serial clock is available to all the transmitter channels in the transceiver block. In a ×8 configuration, only the CMU0 clock divider of the master transceiver block provides the high-speed serial clock to all eight channels.

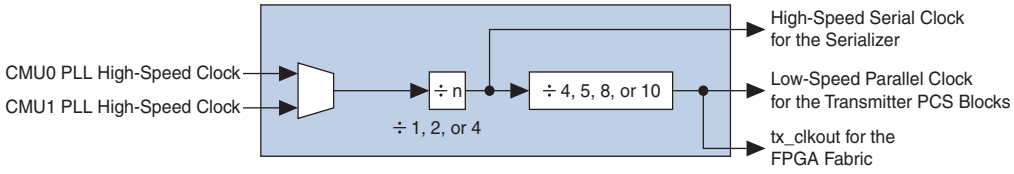
(3) If the byte serializer block is enabled in bonded channel modes, the coreclkout clock output is half the frequency of the low-speed parallel clock. Otherwise, the coreclkout clock output is the same frequency as the low-speed parallel clock.

Transmitter Channel Local Clock Divider Block

Each transmitter channel contains a local clock divider block used automatically by the Quartus II software for non-bonded functional modes (for example, ×1 PCIe, Gbps Ethernet (GbE), SONET/SDH, and SDI mode). This block allows each transmitter channel to run at /1, /2, or /4 of the CMU PLL output data rate.

Figure 1–9 shows the transmitter local clock divider block.

Figure 1–9. Transmitter Local Clock Divider Block

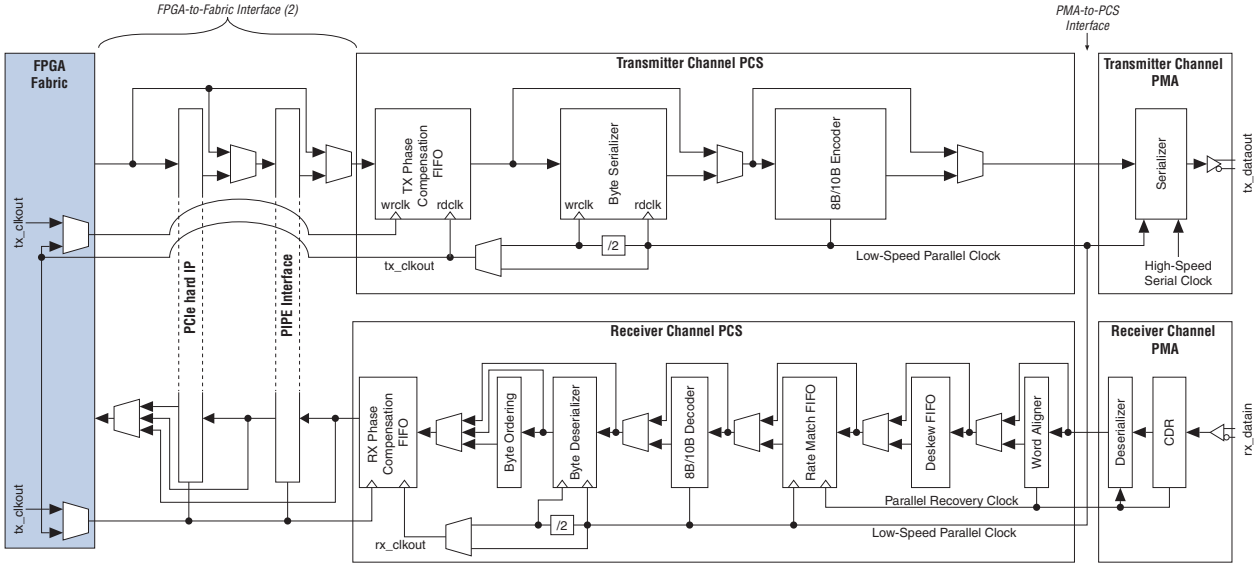


For more information about transceiver channel local clock divider block clocking, refer to the "Transceiver Channel Datapath Clocking" section in the *Transceiver Clocking in Arria II Devices* chapter.

Transceiver Channel Architecture

Each transceiver channel consists of a transmitter channel and a receiver channel. Each transmitter or receiver channel comprises the channel PCS and channel PMA blocks. Figure 1–10 shows the Arria II GX and GZ transceiver channel architecture.

Figure 1–10. Transceiver Channel Architecture for Arria II GX and GZ Devices *(Note 1)*



Notes to Figure 1–10:

(1) Shaded boxes are in the FPGA; unshaded boxes are in the I/O periphery.

(2) The PCIe hard IP block and PIPE interface are used only when the FPGA design includes the PCIe megafunction. For more information about the use of these two blocks, refer to the *PCI Express Compiler User Guide*.

The FPGA fabric-to-transceiver interface and the PMA-to-PCS interface can support an 8, 10, 16, or 20 bit-width data bus.

The transceiver channel is available in two modes:

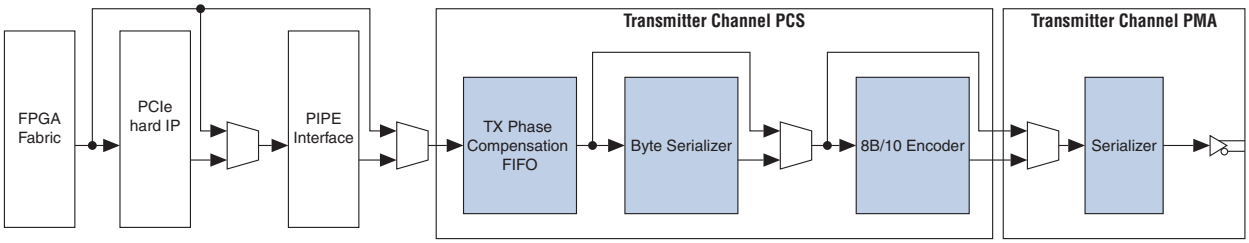
- Single-width mode—In this mode, the PMA-to-PCS interface uses an 8- or 10-bit wide data bus. The FPGA fabric-to-transceiver interface supports an 8- or 10-bit wide data bus, with the byte serializer/deserializer disabled. When the byte serializer/deserializer is enabled, the FPGA fabric-to-transceiver interface supports a 16 or 20 bit-width data bus.
- Double-width mode—In this mode, both the PMA-to-PCS interface and the FPGA fabric-to-transceiver uses an 16- and 20-bit wide data bus. The byte serializer/deserializer is supported in Arria II GZ devices, but not in Arria II GX devices. This mode is only supported for BASIC or Deterministic Latency protocol, used for CPRI and OBSAI interfaces.

Transmitter Channel Datapath

This section describes the Arria II GX and GZ transmitter channel datapath architecture. The sub-blocks in the transmitter datapath are described in order from the transmitter (TX) phase compensation FIFO buffer at the FPGA fabric-to-transceiver interface to the transmitter input buffer.

Figure 1–11 shows the transmitter channel datapath.

Figure 1–11. Transmitter Channel Datapath



Transmitter PCS

This section describes the transmitter PCS modules, which consists of the TX phase compensation FIFO, byte serializer, and 8B/10B encoder.

The `tx_digitalreset` signal resets all modules in the transmitter PCS block.

For more information about the `tx_digitalreset` signal, refer to the *Reset Control and Power Down in Arria II Devices* chapter.

TX Phase Compensation FIFO

This FIFO compensates for the phase difference between the low-speed parallel clock and the FPGA fabric interface clock. Table 1–4 lists the available modes for the TX phase compensation FIFO.

Table 1–4. Transmitter Phase Compensation FIFO Modes for Arria II Devices

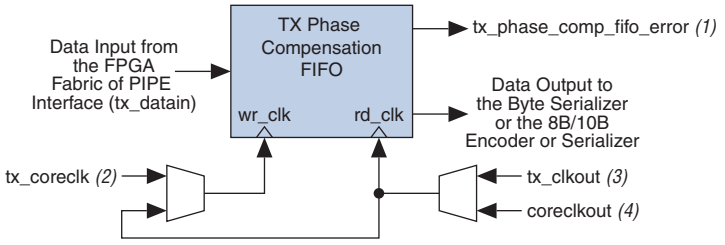
Mode	FIFO Depth	Latency Through FIFO	Applicable Functional Modes *(1)*
Low Latency	4-words deep	2-to-3 parallel clock cycles	All functional modes except PCIe and Deterministic Latency
High Latency	8-words deep	4-to-5 parallel clock cycles	PCIe
Register	—	1	Deterministic Latency

Note to Table 1–4:

(1) Automatically set when you select a protocol in the ALTGX MegaWizard Plug-In Manager.

Figure 1–12 shows the datapath and clocking of the TX phase compensation FIFO.

Figure 1–12. TX Phase Compensation FIFO



Notes to Figure 1–12:

(1) The `tx_phase_comp_fifo_error` is optional and available in all functional modes. This signal is asserted high to indicate an overflow or underflow condition.

(2) Use this optional clock for the FIFO write clock if you instantiate the `tx_coreclk` port in the ALTGX MegaWizard Plug-In Manager, regardless of the channel configurations. Otherwise, the same clock used for the read clock is also used for the write clock. Ensure that there is 0 parts per million (PPM) frequency difference between the `tx_coreclk` clock and the read clock of the FIFO.

(3) The `tx_clkout` low-speed parallel clock is from the local clock divider from the associated transmitter channel and is used in non-bonded configurations.

(4) The `coreclkout` clock is from the `CMU0` block of the associated transceiver block or the master transceiver block for ×4 bonded or ×8 bonded channel configurations, respectively.

For more information about TX phase compensation FIFO clocking, refer to the "Limitation of the Quartus II Software-Selected Transmitter Phase Compensation FIFO Write (or Read) Clocks" section in the *Transceiver Clocking in Arria II Devices* chapter.

An optional `tx_phase_comp_fifo_error` port is available in all functional modes and is asserted high in an overflow or underflow condition. If this signal is asserted, ensure that there is 0 PPM difference between the TX phase compensation FIFO read and write clocks.

The output of this block can go to any of the following blocks:

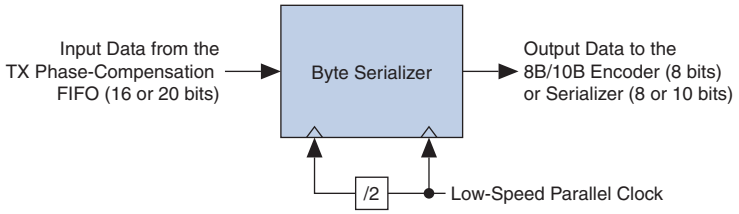
- Byte serializer—If you enable this block.
- 8B/10B encoder—If you disable the byte serializer, but enable the 8B/10B encoder and your channel width is either 8 or 16 bits.
- Serializer—If you disable both the byte serializer and the 8B/10B encoder, or if you use low-latency PCS bypass mode.

Byte Serializer

In Arria II GX devices, you cannot enable the byte serializer in double-width mode. However, in Arria II GZ devices, you can enable both double-width and the byte serializer to achieve a 32- or 40-bit PCS-FPGA interface.

Figure 1–13 shows the byte serializer datapath for Arria II GX devices.

Figure 1–13. Byte Serializer Datapath for Arria II GX Devices



The byte serializer divides the input datapath width by two. This allows you to run the transceiver channel at higher data rates while keeping the FPGA fabric frequency within the maximum limit. This module is required in configurations that exceed the FPGA fabric-to-transceiver interface clock upper frequency limit. It is optional in configurations that do not exceed the FPGA fabric-to-transceiver interface clock upper frequency limit.

For example, if you want to run the transceiver channel at 3.125 Gbps, without the byte serializer, the FPGA fabric interface clock frequency must be 312.5 MHz (3.125 Gbps/10), which violates the FPGA fabric interface frequency limit. When you use the byte serializer, the FPGA fabric interface frequency is 156.25 MHz (3.125 Gbps/20).

For more information about the maximum frequency limit for the FPGA fabric-to-transceiver interface, refer to the *Device Datasheet for Arria II Devices*.

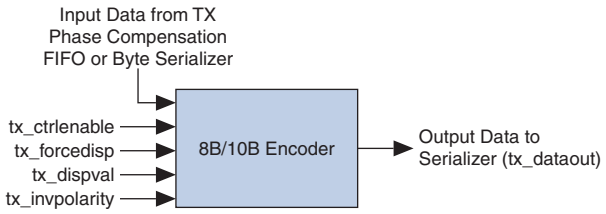
The byte serializer forwards the data from the TX phase compensation FIFO LSByte first. For example, assuming a channel width of 20 bits, the byte serializer sends out the least significant word `datain[9:0]` of the parallel data from the FPGA fabric, followed by `datain[19:10]`.

The data from the byte serializer is forwarded to the 8B/10B encoder if the module is enabled and the input data width is 16 bits. Otherwise, the output is forwarded to the serializer module in the transceiver PMA block.

8B/10B Encoder

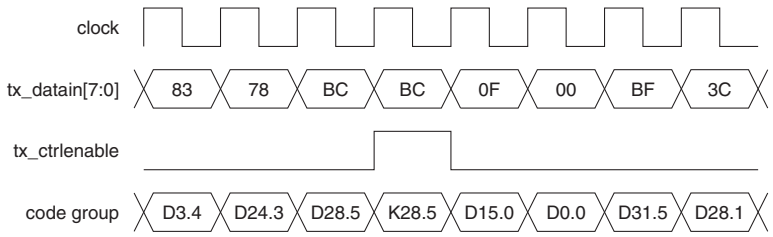
Figure 1–14 shows the inputs and outputs of the 8B/10B encoder.

Figure 1–14. 8B/10B Encoder



The 8B/10B encoder generates 10-bit code groups from the 8-bit data and 1-bit control identifier. If the `tx_ctrlenable` input is high, the 8B/10B encoder translates the 8-bit input data to a 10-bit control word (Kx.y). Otherwise, the 8B/10B encoder translates the 8-bit input data to a 10-bit data word (Dx.y). Figure 1–15 shows an example of how the second 8'hBC data is encoded as a control word, while the reset of the data are encoded as a data word.

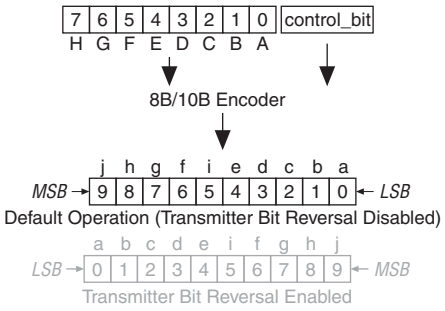
Figure 1–15. Control Word and Data Word Transmission



The IEEE 802.3 8B/10B encoder specification identifies only a set of 8-bit characters for which `tx_ctrlenable` should be asserted. If you assert `tx_ctrlenable` for any other set of characters, the 8B/10B encoder might encode the output 10-bit code as an invalid code (it does not map to a valid Dx.y or Kx.y code), or an unintended valid Dx.y code, depending on the value entered. It is possible for a downstream 8B/10B decoder to decode an invalid control word into a valid Dx.y code without asserting any code error flags. Altera recommends not asserting `tx_ctrlenable` for unsupported 8-bit characters.

Figure 1–16 shows the conversion format. The LSB is transmitted first by default. You can, however, enable the **Transmitter Bit Reversal** option in the ALTGX MegaWizard Plug-In Manager to allow reversing the transmit bit order (MSB first) before it is forwarded to the serializer.

Figure 1–16. 8B/10B Conversion Format

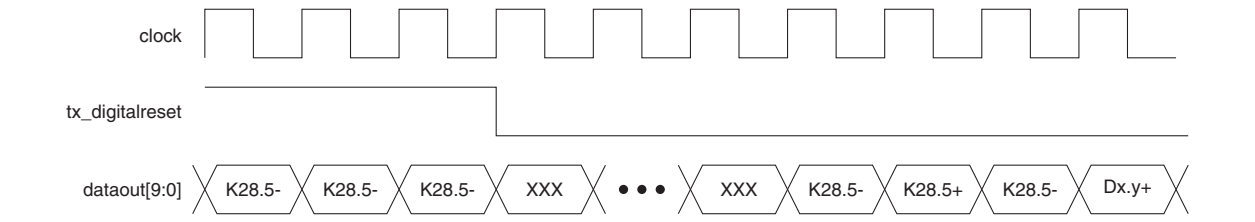


During reset, the running disparity and data registers are cleared. Also, the 8B/10B encoder outputs a K28.5 pattern from the RD- column continuously until `tx_digitalreset` is de-asserted. The input data and control code from the FPGA fabric is ignored during the reset state. After power up or reset, the 8B/10B encoder starts with a negative disparity (RD-) and transmits three K28.5 code groups for synchronization before it starts encoding and transmitting data on its output.

While `tx_digitalreset` is asserted, the downstream 8B/10B decoder that receives the data might observe synchronization or disparity errors.

Figure 1–17 shows the reset behavior of the 8B/10B encoder. 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. Due to some pipelining of the transmitter channel PCS, some "don't cares" (10'hxxx) are sent before the three synchronizing K28.5 code groups. User data follows the third K28.5 code group.

Figure 1–17. 8B/10B Encoder Output during tx_digitalreset Assertion



In Basic functional mode, you can use the `tx_forcedisp` and `tx_dispval` ports to control the running disparity of the output from the 8B/10B encoder. Forcing disparity can either maintain the current running disparity calculations if the forced disparity value (on the `tx_dispval` bit) happens to match the current running disparity, or flip the current running disparity calculations if it does not match. If the forced disparity flips the current running disparity, the downstream 8B/10B decoder might detect a disparity error.

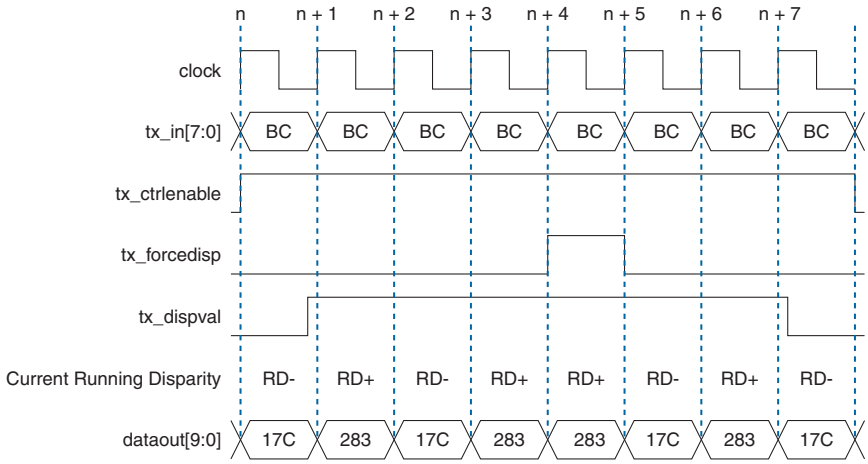
Table 1–5 lists the tx_forcedisp and tx_dispval port values and the effects they have on the data.

Table 1–5. tx_forcedisp and tx_dispval Port Values for Arria II Devices

tx_forcedisp	tx_dispval	Description
0	X	Current running disparity has no change.
1	0	Encoded data has positive disparity.
1	1	Encoded data has negative disparity.

Figure 1–18 shows an example of tx_forcedisp and tx_dispval port use, where data is shown in hexadecimal radix.

Figure 1–18. 8B/10B Encoder Force Running Disparity Operations



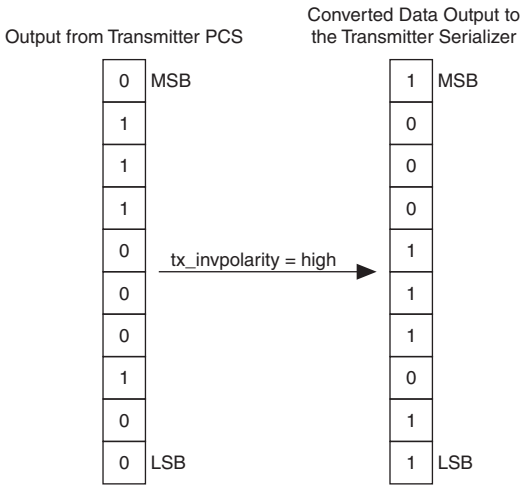
In this example, a series of K28.5 code groups are continuously sent. The stream alternates between a positive running disparity K28.5 (RD+) and a negative running disparity K28.5 (RD-) to maintain a neutral overall disparity. The current running disparity at time n + 3 indicates that the K28.5 in time n + 4 must be encoded with a negative disparity. Because tx_forcedisp is high at time n + 4, and tx_dispval is also high, the K28.5 at time n + 4 is encoded as a positive disparity code group.

The optional tx_invpolarity port is available in all functional modes to dynamically enable the transmitter polarity inversion feature as a workaround to board re-spin or a major update to the FPGA fabric design when the positive and negative signals of a serial differential link are accidentally swapped during board layout.

A high value on the tx_invpolarity port inverts the polarity of every bit of the input data word to the serializer in the transmitter datapath. Correct data is seen by the receiver, because inverting the polarity of each bit has the same effect as swapping the positive and negative signals of the differential link. The tx_invpolarity signal is dynamic and might 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 1–19 shows an example result with the tx_invpolarity feature in a 10-bit wide datapath configuration.

Figure 1–19. Transmitter Polarity Inversion



Transmitter PMA

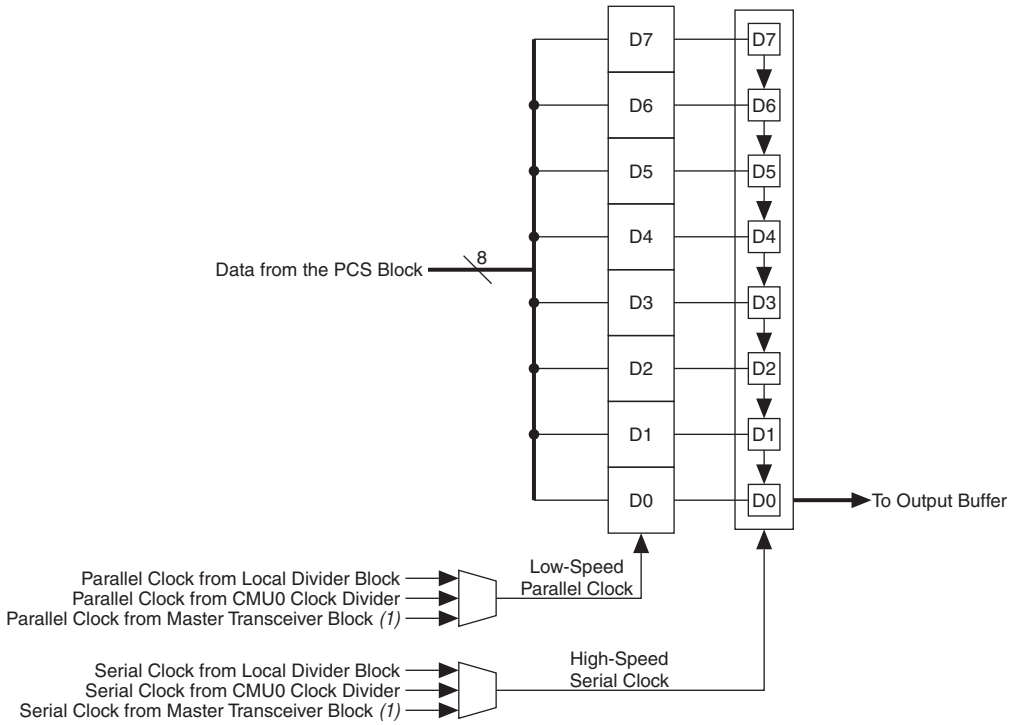
This section describes the transmitter PMA modules that consist of the serializer and the transmitter output buffer.

The tx_analogreset signal resets all modules in the transmitter PMA block. For more information about this signal, refer to the *Reset Control and Power Down in Arria II Devices* chapter.

Serializer

The serializer converts the incoming low-speed parallel signal from the transceiver PCS to the high-speed serial data and sends its LSB first to the transmitter output buffer. Figure 1–20 shows the serializer block diagram in an 8-bit PCS-to-PMA interface.

Figure 1–20. Serializer Block in an 8-Bit PCS-PMA Interface

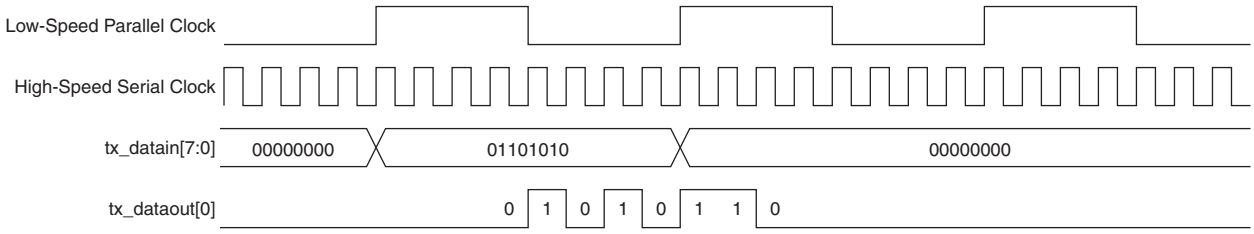


Note to Figure 1–20:

(1) This clock is provided by the CMU0 clock divider of the master transceiver block and is only used in ×8 mode.

Figure 1–21 shows an example of serialized data with a 8'b01101010 value.

Figure 1–21. Serializer Bit Order ***(Note 1)***



Note to Figure 1–21:

(1) The input data to the serializer is 8 bits (channel width = 8 bits with the 8B/10B encoder disabled).

Transmitter Output Buffer

The Arria II GX and GZ transmitter output buffers support the 1.5-V pseudo current mode logic (PCML) I/O standard and can drive 40 inches of FR4 trace (with 50-Ω impedance) across two connectors.

For data rates > 3.75 Gbps, Altera recommends limiting the FR4 trace length to 15 inches.

The transmitter output buffer power supply (V_{CCH}) only provides voltage to the transmitter output buffers in the transceiver channels. This is set to **1.5 V** in the ALTGX MegaWizard Plug-In Manager. The common mode voltage (V_{CM}) for the Arria II GX and GZ transmitter output buffers is 650 mV.

To improve signal integrity, the transmitter output buffer has the following additional circuitry, which you can set in the ALTGX MegaWizard Plug-In Manager:

- Programmable differential output voltage (V_{OD})—This feature allows you to customize the V_{OD} to handle different trace lengths, various backplanes, and various receiver requirements.
- Programmable pre-emphasis—Pre-emphasis boosts high frequencies in the transmit data signal, which might be attenuated in the transmission media because of data-dependent jitter and other intersymbol interference (ISI) effects. It equalizes the frequency response at the receiver so the differences between the low-frequency and high-frequency components are reduced, minimizing the ISI effects from the transmission medium.

  Pre-emphasis requirements increase as data rates through legacy backplanes increase. Using pre-emphasis can maximize the data eye opening at the far-end receiver.
- Programmable differential on-chip termination (OCT)—The Arria II GX transmitter buffer includes a differential OCT of 100 Ω . The Arria II GZ transmitter buffer includes a differential OCT of 85, 100, 120, and 150 Ω . The resistance is adjusted in the calibration block to compensate for temperature, voltage, and process changes (for more information, refer to "Calibration Block" on page 1–47).

  You can set the transmitter termination setting in the ALTGX MegaWizard Plug-In Manager or through the Quartus II Assignment Editor by setting the assignment output termination to **85, 100, 120, or 150** for Arria II GZ devices or **100** for Arria II GX devices on the transmitter output buffer.

  You can disable OCT and use external termination. In this case, the transmitter common mode is tri-stated.

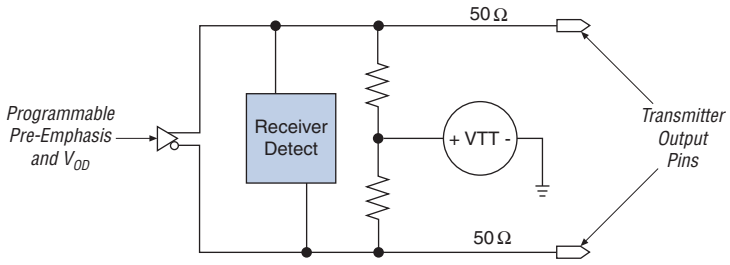
  The Arria II GX and GZ transmitter output buffers in the transceiver block are current-mode drivers. The resulting V_{OD} is a function of the transmitter termination value.

- Receiver-detect capability to support PCIe functional mode—This circuit detects if there is a receiver downstream by sending out a pulse on the common mode of the transmitter and monitoring the reflection. For more information, refer to "PCIe Mode" on page 1–62.
- Tristate-able transmitter buffer to support PCIe electrical idle—This feature is only active in PCIe mode to work hand-in-hand with the receiver-detect capability. For more information, refer to "PCIe Mode" on page 1–62.

For more information about the available settings in each feature, refer to the *Device Datasheet for Arria II Devices.*

Figure 1–22 shows the transmitter output buffer block diagram.

Figure 1–22. Transmitter Output Buffer

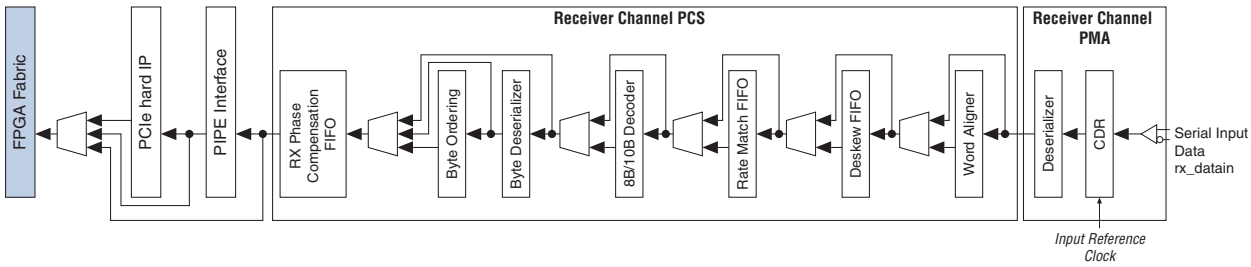


Receiver Channel Datapath

This section describes the Arria II GX and GZ receiver channel datapath architecture. The sub-blocks in the receiver datapath are described in order from the receiver input buffer to the receiver (RX) phase compensation FIFO buffer at the FPGA fabric-to-transceiver interface.

Figure 1–23 shows the receiver channel datapath in Arria II GX and GZ devices.

Figure 1–23. Receiver Channel Datapath



Receiver PMA

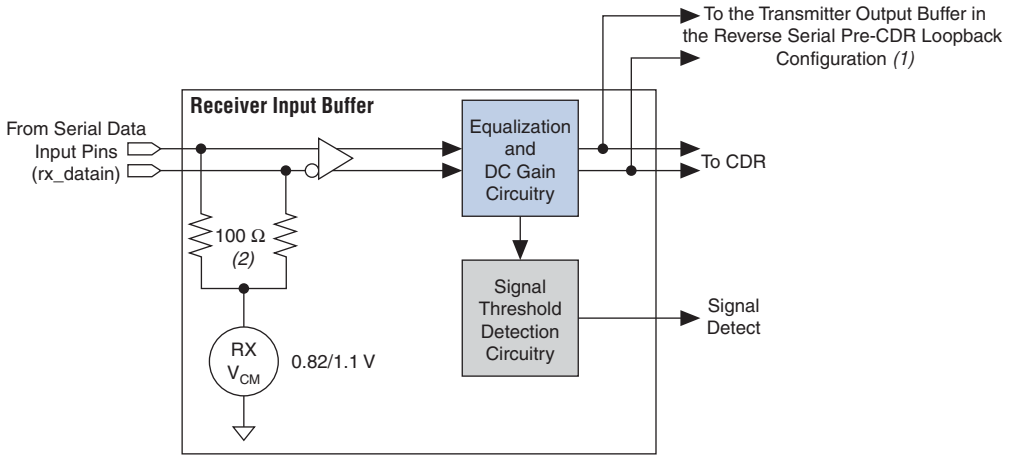
This section describes the receiver PMA modules, which consists of the receiver input buffer, CDR, and deserializer.

The `rx_analogreset` signal resets all modules in the receiver PMA block. For more information about this signal, refer to the *Reset Control and Power Down in Arria II Devices* chapter.

Receiver Input Buffer

The receiver input buffer receives serial data from the rx_datain port and feeds it to the CDR unit. Figure 1–24 shows the receiver input buffer.

Figure 1–24. Receiver Input Buffer for Arria II GX Devices



Notes to Figure 1–24:

(1) For more information about reverse serial pre-CDR loopback mode, refer to "Test Modes" on page 1–85.

(2) For Arria II GZ devices, this is 85, 100, 120, and 150 Ω.

Table 1–6 lists the electrical features supported by the receiver input buffer.

Table 1–6. Electrical Features Supported by the Receiver Input Buffer for Arria II Devices *(Note 1)*

I/O Standard	Programmable Common Mode Voltage (V)	Coupling
1.5 V PCML	0.82	AC, DC
2.5 V PCML	0.82	AC
LVPECL	0.82	AC
LVDS	1.1	AC, DC

Note to Table 1–6:

(1) The differential OCT setting for Arria II GX and GZ transmitters and receivers is 85, 100, 120, and 150 Ω for Arria II GZ devices or 100 Ω for Arria II GX devices.

The following sections describe the features supported in the Arria II GX and GZ receiver input buffers.

Programmable Differential OCT

The Arria II GX transmitter buffer includes a differential OCT of 100 Ω. The Arria II GZ transmitter buffer includes a differential OCT of 85, 100, 120, and 150 Ω. The resistance is adjusted in the calibration block to compensate for temperature, voltage, and process changes (for more information, refer to "Calibration Block" on page 1–47). You can set this option in the Quartus II Assignment Editor by setting the assignment input termination to **OCT 100** Ω for Arria II GX devices and **OCT 85, 100, 120, and 150** Ω for Arria II GZ devices on the receiver input buffer.

Programmable Common Mode Voltage

The Arria II GX and GZ receivers have on-chip biasing circuitry to establish the required common mode voltage at the receiver input that supports two common mode voltage settings of 0.82 V and 1.1 V. You can select the voltage in the ALTGX MegaWizard Plug-In Manager. For the I/O standards supported by each common mode voltage setting, refer to Table 1–6.

This feature is effective only if you use programmable OCT for the receiver input buffers as well. If you use external termination, you must implement off-chip biasing circuitry to establish the common mode voltage at the receiver input buffer.

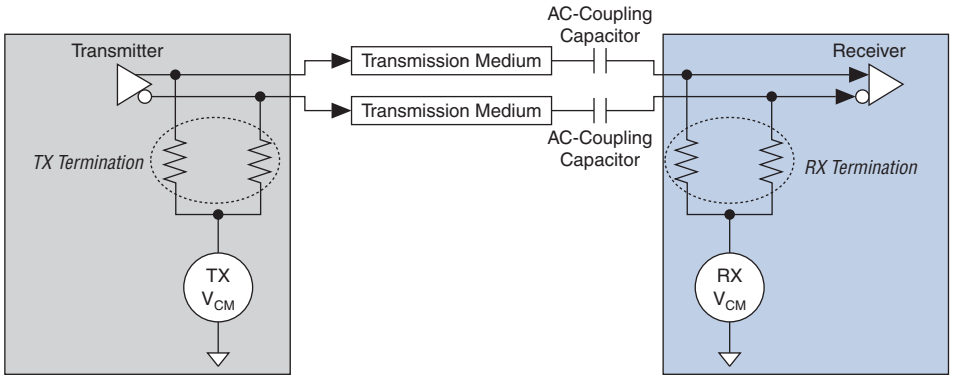
AC and DC Coupling

A high-speed serial link can either be AC-coupled or DC-coupled, depending on the serial protocol implementation. Most of the serial protocols require links to be AC-coupled, protocols similar to SONET optionally allow DC coupling.

In an AC-coupled link, the AC-coupling capacitor blocks the transmitter DC common mode voltage. The on-chip receiver termination and biasing circuitry automatically restores the selected common mode voltage. AC-coupled links are required in GbE, PCIe, SRIO, SDI, and XAUI protocols.

Figure 1–25 shows an AC-coupled link.

Figure 1–25. AC-Coupled Link



In a DC-coupled link, the transmitter DC common mode voltage is seen unblocked at the receiver input buffer. The link common mode voltage depends on the transmitter common mode voltage and the receiver common mode voltage. The on-chip or off-chip receiver termination and biasing circuitry must ensure compatibility between the transmitter and the receiver common mode voltage.

Figure 1–26 shows a DC-coupled link.

Figure 1–26. DC-Coupled Link

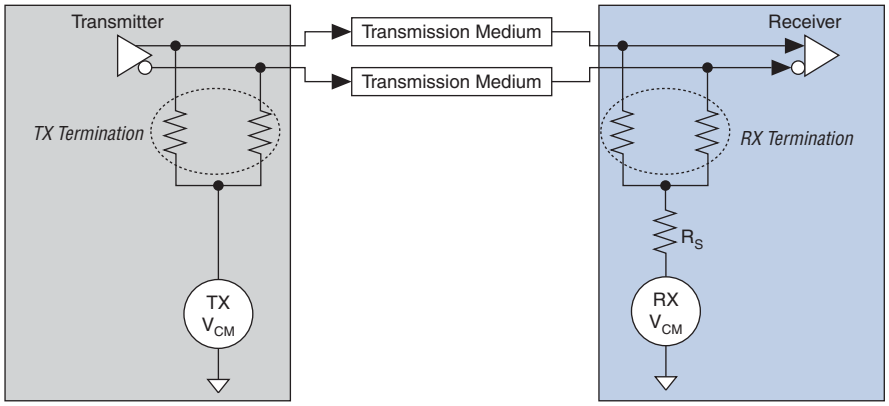


Figure 1–27 shows the DC-coupled link connection from an LVDS transmitter to an Arria II GX and GZ receiver.

Figure 1–27. LVDS Transmitter to Arria II GX and GZ Receiver (PCML) DC-Coupled Link

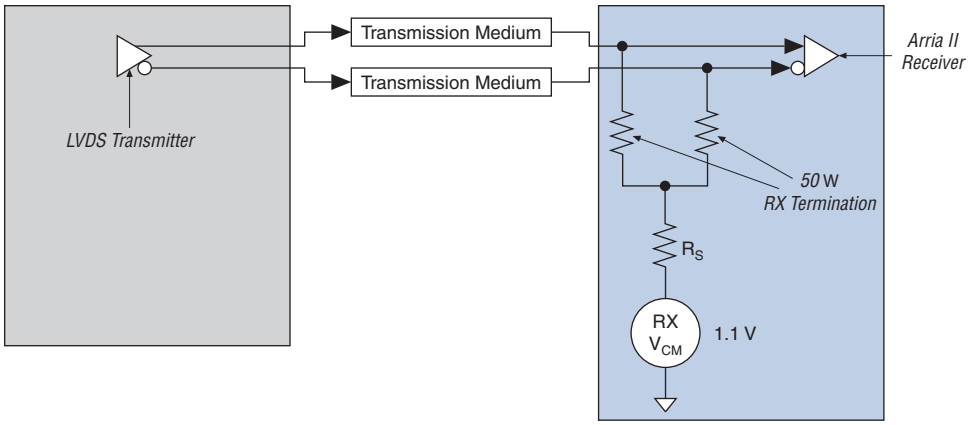


Table 1–7 lists the settings for DC-coupled links between Altera devices. You must comply with the data rates supported by the Arria II GX and GZ receivers.

Table 1–7. DC-Coupled Settings for Arria II Devices *(Note 1)* (Part 1 of 2)

Link	Transceiver Settings	Receiver Settings
	TX VCM (V)	RX VCM (V)
Arria II PCML transmitter to Arria II PCML receiver	0.65	0.82
Stratix II GX PCML transmitter to Arria II PCML receiver	0.6, 0.7	0.82
Arria II PCML transmitter to Stratix II GX PCML receiver	0.65	0.85
Arria II PCML transmitter to Stratix IV GX PCML receiver	0.65	0.82
Stratix IV GX PCML transmitter to Arria II PCML receiver	0.65	0.85

Table 1–7. DC-Coupled Settings for Arria II Devices *(Note 1)* (Part 2 of 2)

Link	Transceiver Settings	Receiver Settings
	TX VCM (V)	RX VCM (V)
LVDS transmitter to Arria II GX and GZ receiver	—	1.1

Note to Table 1–7:

(1) The differential OCT setting for Arria II GX and GZ transmitters and receivers is 85, 100, 120, and 150 Ω for Arria II GZ devices or 100 Ω for Arria II GX devices except for the LVDS transmitter settings, which do not have OCT set on the transmitter (as shown in Figure 1–27).

Programmable Equalization, DC Gain, and Offset Cancellation

Each Arria II GX and GZ receiver input buffer has independently programmable equalization circuitry that boosts the high-frequency gain of the incoming signal, thereby compensating for the low-pass filter effects of the physical medium. The amount of high-frequency gain required depends on the loss characteristics of the physical medium. Arria II GX and GZ equalization circuitry supports equalization settings that provide up to 7 dB (Arria II GX) and 16 dB (Arria II GZ) of high-frequency boost.

The Arria II GX and GZ receiver input buffer also supports programmable DC gain circuitry. Unlike equalization circuitry, DC gain circuitry provides equal boost to the incoming signal across the frequency spectrum.

You can select the proper equalization and DC gain settings in the ALTGX MegaWizard Plug-In Manager. The receiver buffer supports DC gain settings of 0 dB, 3 dB, and 6 dB for Arria II GX devices and up to 12 dB for Arria II GZ devices.

This offset cancellation block cancels offset voltages between the positive and negative differential signals within the equalizer stages in order to reduce the minimum V_{ID} requirement. The receiver input buffer and receiver CDR require offset cancellation.

The offset cancellation for the receiver channels option is automatically enabled in both the ALTGX and ALTGX_RECONFIG MegaWizard Plug-In Managers for **Receiver**, **Transmitter**, and **Receiver only** configurations. When offset cancellation is automatically enabled, you must instantiate the dynamic reconfiguration controller to connect the reconfiguration ports created by the ALTGX MegaWizard Plug-In Manager.

For more information about offset cancellation, refer to *AN 558: Implementing Dynamic Reconfiguration in Arria II Devices*. For the transceiver reset sequence with the offset cancellation feature, refer to the *Reset Control and Power Down in Arria II Devices* chapter.

Signal Threshold Detection Circuitry

Signal threshold detection circuitry has a hysteresis response that filters out any high-frequency ringing caused by ISI effects or high-frequency losses in the transmission medium. If the signal threshold detection circuitry senses the signal level present at the receiver input buffer to be higher than the signal detect threshold, it asserts the `rx_signaldetect` signal high. Otherwise, the `rx_signaldetect` signal is held low.

In PCIe mode, you can enable the optional signal threshold detection circuitry by leaving the **Force signal detection** option unchecked in the ALTGX MegaWizard Plug-In Manager.

The appropriate signal detect threshold level that complies with the PCIe compliance parameter VRX-IDLE-DETDIFFp-p is pending characterization.

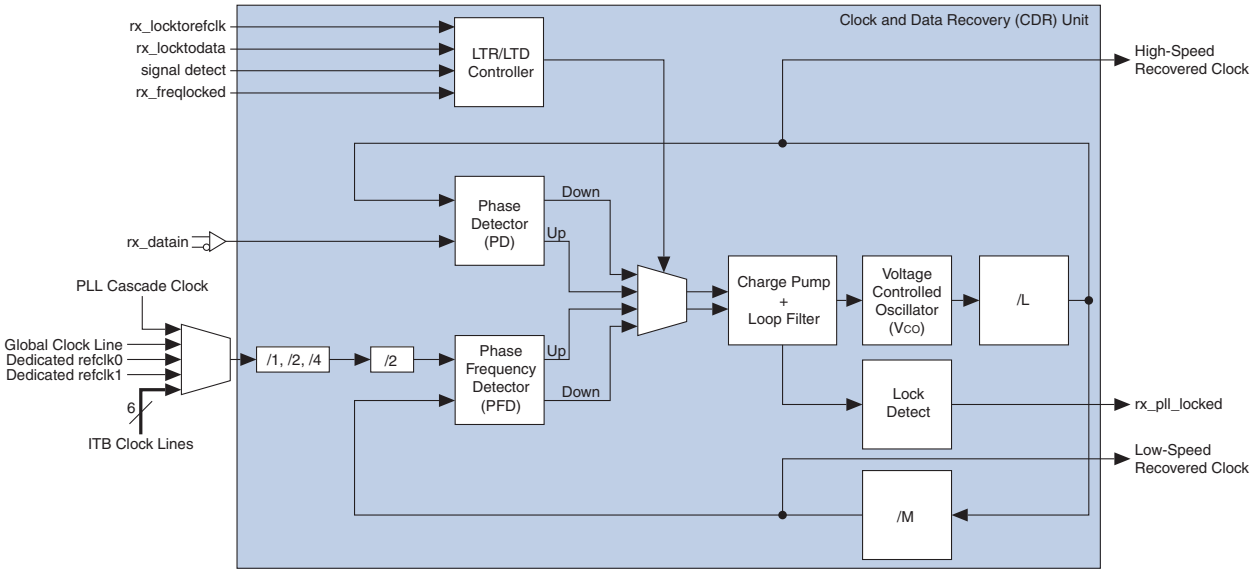
If you enable the **Force signal detection** option in the ALTGX MegaWizard Plug-In Manager, the `rx_signaldetect` signal is always asserted high, irrespective of the signal level on the receiver input buffer. When enabled, this option senses whether the signal level present at the receiver input buffer is above the signal detect threshold voltage that you specified in the **What is the signal detect and signal loss threshold?** option in the ALTGX MegaWizard Plug-In Manager.

The `rx_signaldetect` signal is also used by the LTR/LTD controller in the receiver CDR to switch between LTR and LTD lock modes. When the signal threshold detection circuitry de-asserts the `rx_signaldetect` signal, the LTR/LTD controller switches the receiver CDR from lock-to-data (LTD) to lock-to-reference (LTR) lock mode.

CDR

Each Arria II GX and GZ receiver channel has an independent CDR unit to recover the clock from the incoming serial data stream. High-speed and low-speed recovered clocks are used to clock the receiver PMA and PCS blocks. Figure 1–28 shows the CDR block.

Figure 1–28. CDR Block



The CDR operates in two modes:

- LTR mode—The PFD in the CDR tracks the receiver input reference clock (`rx_cruclk`) and controls the charge pump that tunes the VCO in the CDR. An active high `rx_pll_locked` status signal is asserted to indicate that the CDR has locked to phase and frequency of the receiver input reference clock. In this mode, the phase detector is inactive.

  Depending on the data rate and the selected input reference clock frequency, the Quartus II software automatically selects the appropriate divider values such that the CDR output clock frequency is half the data rate. This includes the pre-divider before the PFD.

- LTD mode—The phase detector in the CDR tracks the incoming serial data at the receiver input buffer to keep the recovered clock phase-matched to the data. Depending on the phase difference between the incoming data and the CDR output clock, the phase detector controls the CDR charge pump that tunes the VCO.

  In this mode, the PFD and the /M divider block are inactive. In addition, the `rx_pll_locked` signal toggles randomly and has no significance in LTD mode.

  The CDR must be in LTD mode to recover the clock from the incoming serial data during normal operation. The actual LTD lock time depends on the transition density of the incoming data and the PPM difference between the receiver input reference clock and the upstream transmitter reference clock. The receiver PCS logic must be held in reset until the CDR asserts the `rx_freqlocked` signal and produces a stable recovered clock.

  For more information about receiver reset recommendations, refer to the *Reset Control and Power Down* chapter.

The CDR must be kept in LTR mode until it locks to the input reference clock after the power-up and reset cycle. When locked to the input reference clock, the CDR output clock is trained to the configured data rate and can switch to LTD mode to recover the clock from the incoming data. You can use the optional input ports (`rx_locktorefclk` and `rx_locktodata`) to control the LTR or LTD mode manually or let the lock happen automatically.

Table 1–8 lists the relationship between the optional input ports and the LTR/LTD controller lock mode.

Table 1–8. Optional Input Ports and LTR/LTD Controller Lock Mode for Arria II Devices *(Note 1)*

rx_locktorefclk	rx_locktodata	LTR/LTD Controller Lock Mode
1	0	Manual – LTR Mode
X	1	Manual – LTD Mode
0	0	Automatic Lock Mode

Note to Table 1–8:

(1) If you do not instantiate the optional `rx_locktorefclk` and `rx_locktodata` signals in the ALTGX megafunction, the Quartus II software automatically configures the LTR/LTD controller in automatic lock mode.

Automatic Lock Mode

In automatic lock mode, the LTR/LTD controller relies on the PPM detector and the phase relationship detector to set the CDR in LTR or LTD mode. Initially, the CDR is set to LTR mode. After the CDR locks to the input reference clock, the LTR/LTD controller automatically sets it to LTD mode and asserts the `rx_freqlocked` signal when the following three conditions are met:

- Signal threshold detection circuitry indicates the presence of valid signal levels at the receiver input buffer
- CDR output clock is within the configured PPM frequency threshold setting with respect to the input reference clock (frequency is locked)
- CDR output clock and input reference clock are phase matched within approximately 0.08 UI (phase is locked)

If the CDR does not stay locked-to-data due to frequency drift or severe amplitude attenuation, the LTR/LTD controller switches the CDR back to LTR mode to lock to the input reference clock. The LTR/LTD controller switches the CDR from LTD to LTR mode and de-asserts the `rx_freqlocked` signal when the following conditions are met:

- Signal threshold detection circuitry indicates the absence of valid signal levels at the receiver input buffer
- CDR output clock is not in the configured PPM frequency threshold setting with respect to the input reference clock

Manual Lock Mode

You may want to use manual lock mode if your application requires faster CDR lock time. In manual lock mode, the LTR/LTD controller sets the CDR in LTR or LTD mode, depending on the logic level on the `rx_locktorefclk` and `rx_locktodata` signals, as shown in Table 1–8 on page 1–27.

When the `rx_locktorefclk` signal is asserted high, the `rx_freqlocked` signal does not have significance and is always driven low, indicating that the CDR is in LTR mode. When the `rx_locktodata` signal is asserted high, the `rx_freqlocked` signal is always driven high, indicating that the CDR is in LTD mode. If both signals are de-asserted, the CDR is in automatic lock mode.

You must comply with the different transceiver reset sequences depending on the CDR lock mode.

Deserializer

The deserializer block latches the serial input data from the receiver input buffer with the high-speed serial recovery clock, deserializes it using the low-speed parallel recovery clock, and drives the deserialized data to the receiver PCS channel.

The deserializer supports 8-, 10-, 16-, and 20-bit deserialization factors. Figure 1–29 shows the deserializer operation with a 10-bit deserialization factor.

Figure 1–29. 10-Bit Deserializer Operation

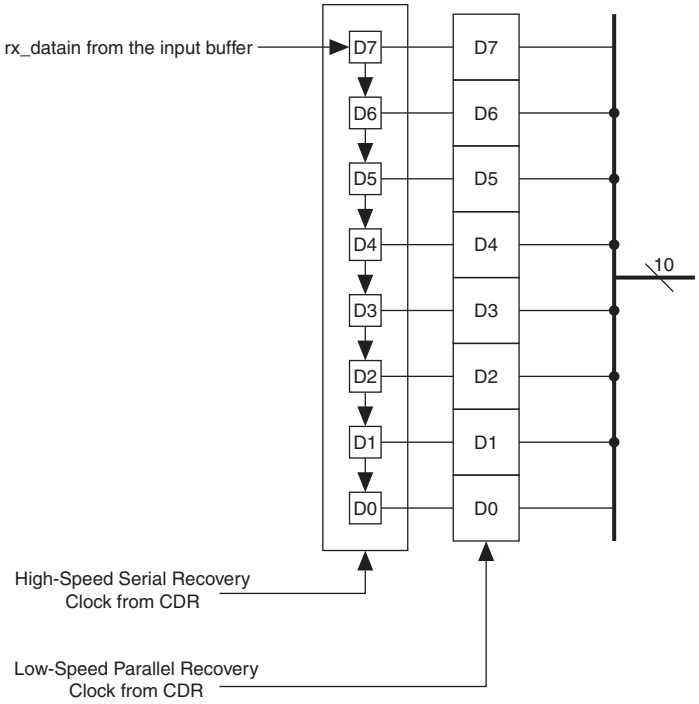
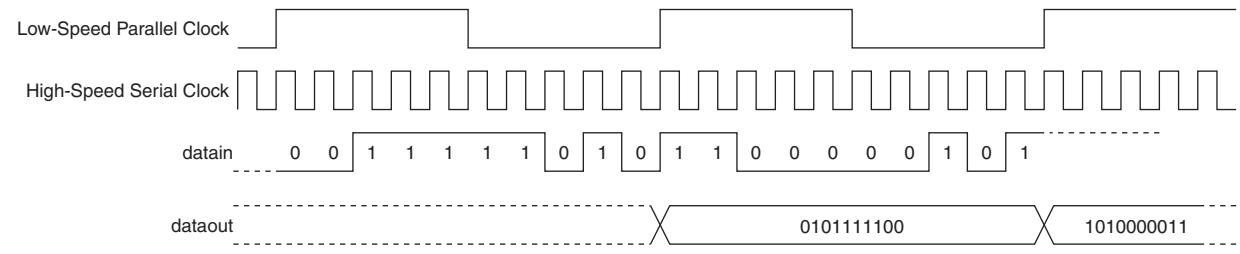


Figure 1–30 shows the serial bit order of the deserializer block input and the parallel data output of the deserializer block with a 10-bit deserialization factor. The serial stream (10'b0101111100) is deserialized to a value 10'h17C. The serial data is assumed to have received the LSB first.

Figure 1–30. 10-Bit Deserializer Bit Order



Receiver PCS

This section describes the receiver PCS modules, which consist of the word aligner, deskew FIFO, rate-match FIFO, 8B/10B decoder, byte deserializer, byte ordering, and RX phase compensation FIFO.

The `rx_digitalreset` signal resets all modules in the receiver PCS block.

For more information about this signal, refer to the *Reset Control and Power Down in Arria II Devices* chapter.

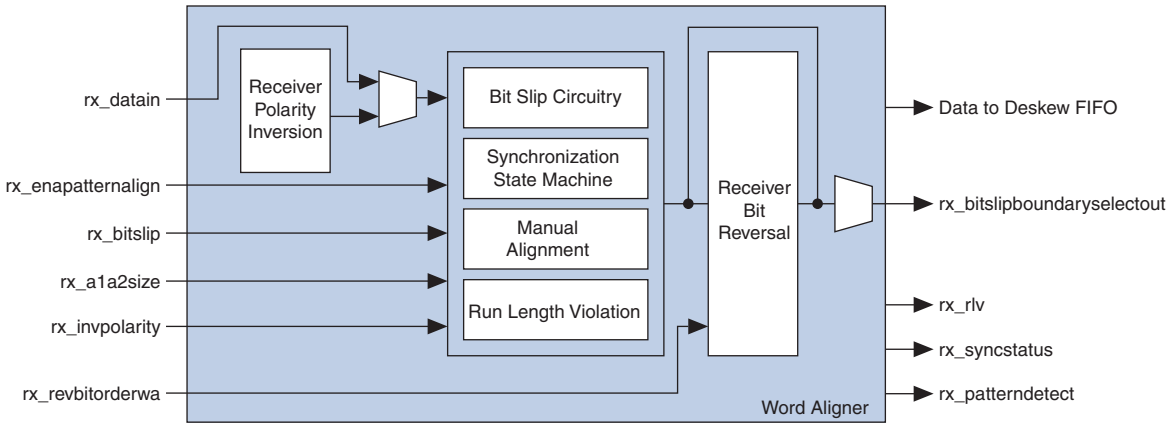
Word Aligner

The word aligner receives parallel data from the deserializer and restores the word boundary based on a pre-defined alignment pattern that must be received during link synchronization.

Serial protocols such as GbE, PCIe, SRIO, SONET/SDH, and XAUI specify a standard word alignment pattern. The Arria II GX and GZ transceiver architecture allows you to select a custom word alignment pattern specific to your implementation if you use proprietary protocols.

Figure 1–31 shows the word aligner block diagram.

Figure 1–31. Word Aligner



In addition to restoring word boundaries, the word aligner also implements the following features:

- Programmable run length violation detection—This feature is available in all functional modes. It detects consecutive 1s or 0s in the data stream. If a preset maximum number of consecutive 1s or 0s is detected, the run length violation status signal (`rx_rlv`) is asserted. This signal has lower latency when compared with the parallel data on the `rx_dataout` port.

  The `rx_rlv` signal in each channel is clocked by its parallel recovered clock and is asserted for a minimum of two recovered clock cycles to ensure that the FPGA fabric clock can latch the `rx_rlv` signal reliably because the FPGA fabric clock might have phase differences, PPM differences (in asynchronous systems), or both, with the recovered clock.

Table 1–9 lists the detection capabilities of the run-length violation circuit.

Table 1–9. Detection Capabilities of the Run-Length Violation Circuit for Arria II Devices

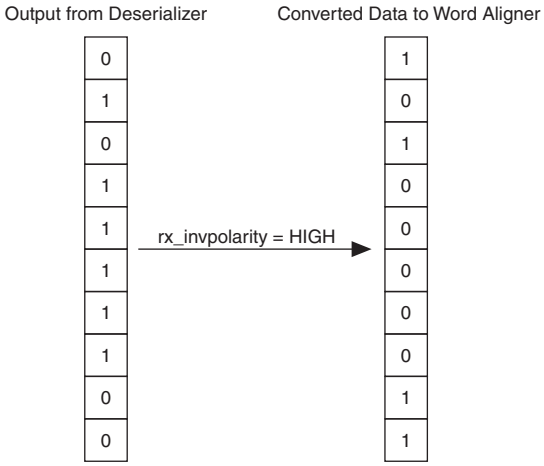
PMA-PCS Interface Width	Run Length Violation Detector Range	
	Minimum	Maximum
8 bit	4	128
10 bit	5	160
16 bit	8	512
20 bit	10	640

- Receiver polarity inversion—This feature is available in all functional modes except PCIe. It offers an optional `rx_invpolarity` port to dynamically enable the receiver polarity inversion feature as a workaround to board re-spin or a major update to the FPGA fabric design when the positive and negative signals of a serial differential link are accidentally swapped during board layout.

  A high value on the `rx_invpolarity` port inverts the polarity of every bit of the input data word to the word aligner in the receiver datapath. Because 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` signal is dynamic and might cause initial disparity errors in an 8B/10B encoded link. The downstream system must be able to tolerate these disparity errors.

  Figure 1–32 shows an example result with the `rx_invpolarity` feature in a 10-bit wide datapath configuration.

Figure 1–32. 10-Bit Receiver Polarity Inversion



This generic receiver polarity inversion feature is different from the PCIe 8B/10B polarity inversion feature because it inverts the polarity of the data bits at the input of the word aligner, whereas the PCIe 8B/10B polarity inversion feature inverts the polarity of the data bits at the input of the 8B/10B decoder.

- Receiver bit reversal—This feature is only available in Basic mode. By default, the Arria II GX and GZ receiver assumes LSB-to-MSB transmission. If the transmission order is MSB-to-LSB, the receiver forwards the bit-flipped version of the parallel data to the FPGA fabric on the `rx_dataout` port. The receiver bit reversal feature is available to correct this situation by flipping the parallel data so that the `rx_dataout` port contains the correct bit-ordered data.

  This feature is available through the `rx_revbitordwa` port in Basic mode only with the word aligner configured in bit-slip mode. When you drive the `rx_revbitordwa` signal high in this configuration, the 8-bit or 10-bit data `D[7:0]` or `D[9:0]` at the output of the word aligner gets rewired to `D[0:7]` or `D[0:9]`, respectively.

  Figure 1–33 shows the receiver bit reversal feature in Basic mode with 10-bit wide datapath configurations.

Figure 1–33. 10-Bit Receiver Bit Reversal in Basic Mode with Word Aligner in Bit-Slip Mode

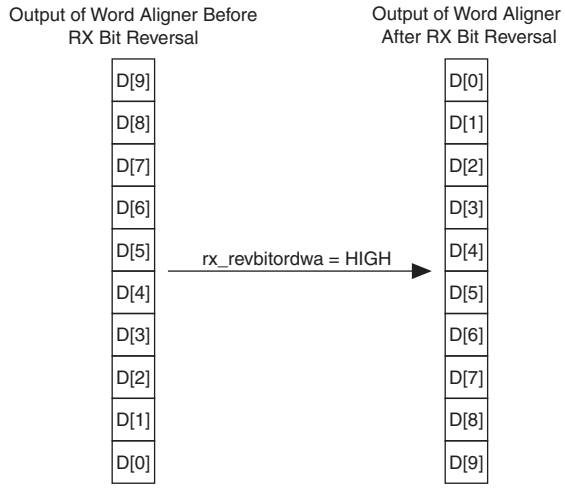


Table 1–10 lists the three modes of the word aligner and their supported data width, functional mode, and allowed alignment pattern length for Arria II devices.

Table 1–10. Word Aligner Modes for Arria II Devices

Word Aligner Mode	Data Width Supported (bits)	Functional Mode Supported	Allowed Word Alignment Pattern Length
Manual Alignment	8	Basic, OC-12, and OC-48	16 bits
	10	Basic and Deterministic Latency	7 or 10 bits
	16	Basic and Deterministic Latency	8, 16, or 32 bits
	20	Basic and Deterministic Latency	7, 10, or 20 bits
Bit-Slip	8	Basic	16 bits
	10	Basic and SDI	7 or 10 bits for Basic N/A for SDI
	16	Basic and Deterministic Latency	8, 16, or 32 bits
	20	Basic and Deterministic Latency	7, 10, or 20 bits
Automatic Synchronization State Machine	10	Basic, GbE, PCIe, Serial RapidIO, and XAUI	10 bits for all functional modes 7 bits or 10 bits for Basic

Manual Alignment Mode

This mode is automatically used in SONET/SDH functional mode. In Basic mode, you can configure the word aligner in manual alignment mode by selecting the **Use manual word alignment mode** option in the word aligner tab of the ALTGX MegaWizard Plug-In Manager.

In manual alignment mode, the input signal (`rx_enapatternalign`) controls the word aligner. The 8-bit word aligner is edge-sensitive to the `rx_enapatternalign` signal; the 10-bit word aligner is level-sensitive to this signal.

If the word alignment pattern is unique and does not appear between word boundaries, you can constantly hold the `rx_enapatternalign` signal high because there is no possibility of false word alignment. If there is a possibility of the word alignment pattern occurring across word boundaries, you must control the `rx_enapatternalign` signal to lock the word boundary after the desired word alignment is achieved to avoid re-alignment to an incorrect word boundary.

With 8-bit width data, a rising edge on the `rx_enapatternalign` signal after de-assertion of the `rx_digitalreset` signal triggers the word aligner to look for the word alignment pattern in the received data stream.

In SONET/SDH OC-12 and OC-48 modes, the word aligner looks for 16'hF628 (A1A2) or 32'hF6F62828 (A1A1A2A2), depending on whether the input signal (`rx_a1a2size`) is driven low or high, respectively. In Basic mode, the word aligner looks for the 16-bit word alignment pattern programmed in the ALTGX MegaWizard Plug-In Manager.

With 10-bit width data, the word aligner looks for the programmed 7-bit or 10-bit word alignment pattern in the received data stream, if the `rx_enapatternalign` signal is held high. It updates the word boundary if it finds the word alignment pattern in a new word boundary. If the `rx_enapatternalign` signal is de-asserted low, the word aligner maintains the current word boundary even when it sees the word alignment pattern in a new word boundary.

The `rx_syncstatus` and `rx_patterndetect` status signals have the same latency as the datapath and are forwarded to the FPGA fabric to indicate word aligner status. On receiving the first word alignment pattern after the assertion of the `rx_enapatternalign` signal, both the `rx_syncstatus` and `rx_patterndetect` signals are driven high for one parallel clock cycle synchronous to the most significant byte (MSByte) of the word alignment pattern. Any word alignment pattern received thereafter in the same word boundary causes only the `rx_patterndetect` signal to go high for one clock cycle.

Any word alignment pattern received thereafter in a different word boundary causes the word aligner to re-align to the new word boundary only if there is a rising edge in the `rx_enapatternalign` signal (in the 8-bit word aligner) or if the `rx_enapatternalign` signal is held high (in 10-bit word aligner). The word aligner asserts the `rx_syncstatus` and `rx_patterndetect` signals for one parallel clock cycle whenever it re-aligns to the new word boundary.

Figure 1–34 shows word aligner behavior in SONET/SDH OC-12 functional mode. The least significant byte (LSByte) (8'hF6) and the MSByte (8'h28) of the 16-bit word alignment pattern are received in parallel clock cycles n and n + 1, respectively. The rx_syncstatus and rx_patterndetect signals are both driven high for one parallel clock cycle synchronous to the MSByte (8'h28) of the word alignment pattern. After the initial word alignment, the 16-bit word alignment pattern (16'h28F6) is again received across the word boundary in clock cycles m, m + 1, and m + 2. The word aligner does not re-align to the new word boundary for lack of a preceding rising edge on the rx_enapatternalign signal. If there is a rising edge on the rx_enapatternalign signal before the word alignment pattern occurs across these clock cycles, the word aligner re-aligns to the new word boundary, causing both the rx_syncstatus and rx_patterndetect signals to go high for one parallel clock cycle.

Figure 1–34. Manual Alignment Mode in 8-Bit PMA-PCS Interface Mode

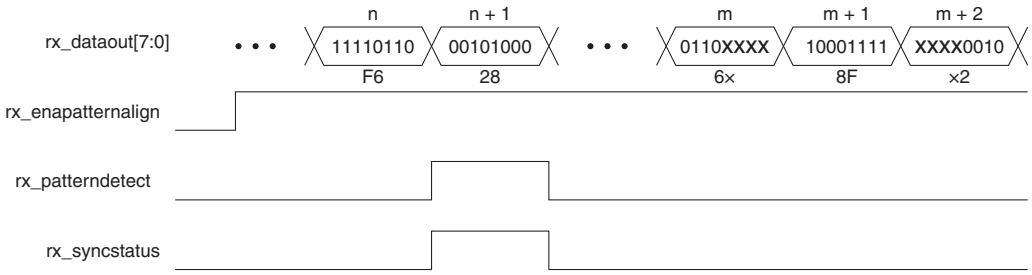
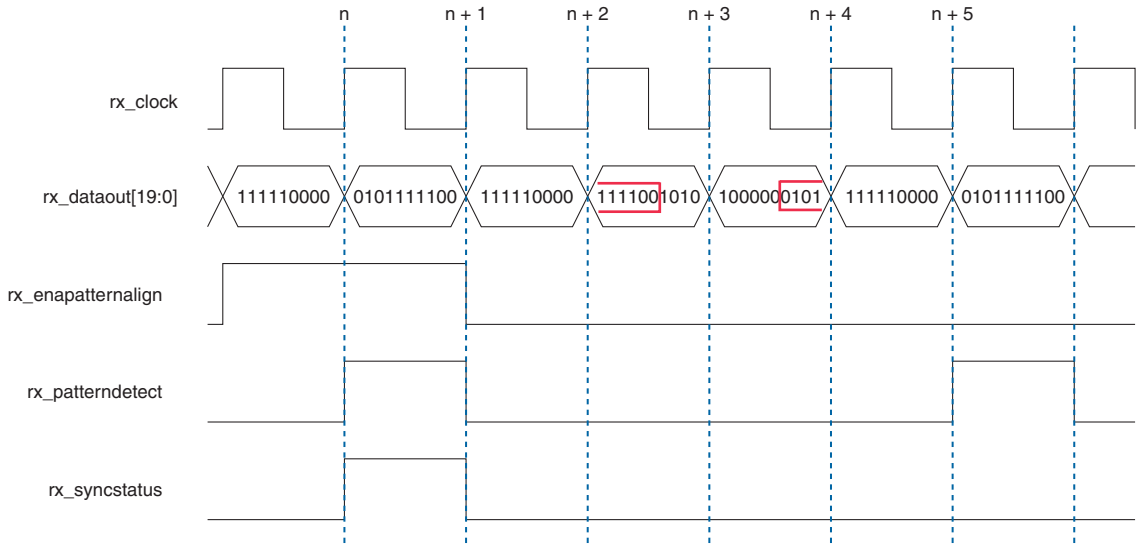


Figure 1–35 shows the manual alignment mode word aligner operation in 10-bit PMA-PCS interface mode. In this example, a /K28.5/ (10'b0101111100) is specified as the word alignment pattern. The word aligner aligns to the /K28.5/ alignment pattern in cycle n because the rx_enapatternalign signal is asserted high. The rx_syncstatus signal goes high for one clock cycle indicating alignment to a new word boundary. The rx_patterndetect signal also goes high for one clock cycle to indicate initial word alignment. At time n + 1, the rx_enapatternalign signal is de-asserted to instruct the word aligner to lock the current word boundary. The alignment pattern is detected again in a new word boundary across cycles n + 2 and n + 3. The word aligner does not align to this new word boundary because the rx_enapatternalign signal is held low. The /K28.5/ word alignment pattern is detected again in the current word boundary during cycle n + 5, causing the rx_patterndetect signal to go high for one parallel clock cycle.

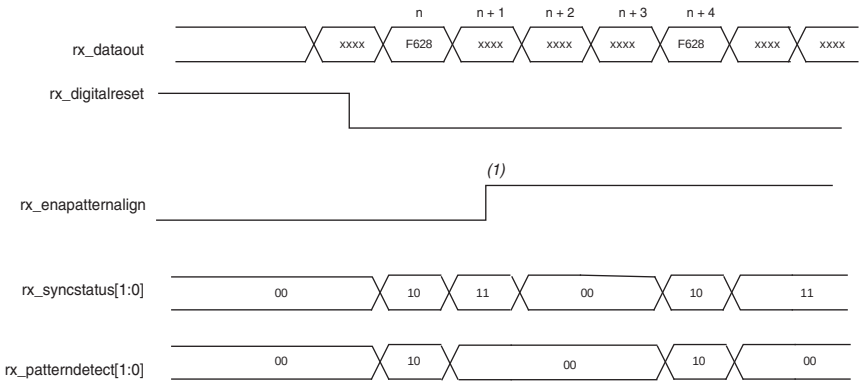
Figure 1–35. Word Aligner in 10-Bit PMA-PCS Manual Alignment Mode



With a 16- or 20-bit width data, the word aligner starts looking for the programmed 8-bit, 16-bit, or 32-bit word alignment pattern in the received data stream as soon as `rx_digitalreset` is de-asserted low. It aligns to the first word alignment pattern received regardless of the logic level driven on the `rx_enapatternalign` signal. Any word alignment pattern received thereafter in a different word boundary does not cause the word aligner to re-align to this new word boundary. After the initial word alignment, following de-assertion of the `rx_digitalreset` signal, if a word re-alignment is required, you must use the `rx_enapatternalign` signal.

Figure 1–36 shows the manual alignment mode word aligner operation in 16-bit PMA-PCS interface mode.

Figure 1–36. Manual Alignment Mode Word Aligner in 16-Bit PMA-PCS Interface Modes



Note to Figure 1–36:

(1) The `rx_syncstatus` de-assertion latency after the assertion of `rx_enapatternalign` depends on your RX PCS implementation.

Bit-Slip Mode

In Basic, deterministic latency, and SDI functional modes, you can configure the word aligner in bit-slip mode by selecting the **Use manual bit slipping mode** option in the ALTGX MegaWizard Plug-In Manager.

Bit slip in the 10-bit wide word aligner allows 7-bit and 10-bit word alignment patterns, whereas bit-slip in the 8-bit wide word aligner allows only a 16-bit word alignment pattern. Other than this, the bit-slip operation is the same between the 8-bit and 10-bit word aligner.

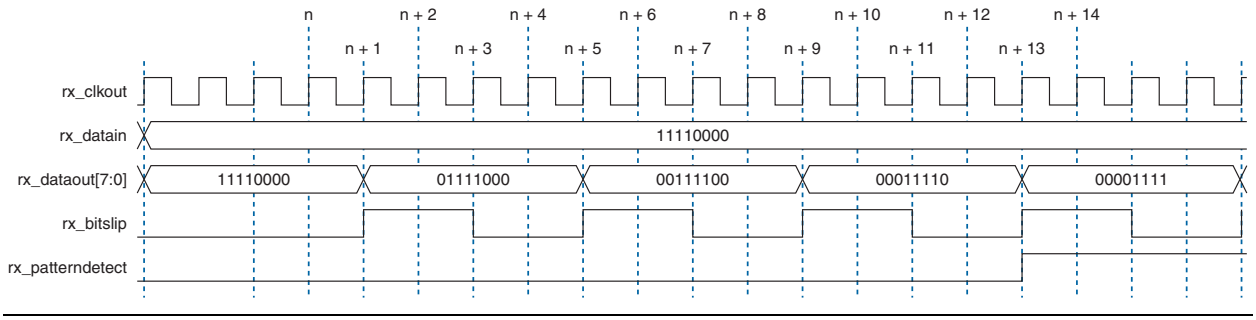
The `rx_bitslip` signal controls the word aligner operation in bit-slip mode. At every rising edge of the `rx_bitslip` signal, the bit-slip circuitry slips one bit into the received data stream, effectively shifting the word boundary by one bit. The `rx_patterndetect` signal is driven high for one parallel clock cycle when the received data after bit-slipping matches the 16-bit word alignment pattern programmed in the ALTGX MegaWizard Plug-In Manager.

You can implement a bit-slip controller in the FPGA fabric that monitors either the `rx_dataout` signal and/or the `rx_patterndetect` signal and controls the `rx_bitslip` signal to achieve word alignment.

Figure 1–37 shows an example of the word aligner configured in bit-slip mode, which has the following events:

- 8'b11110000 is received back-to-back
- 16'b0000111100011110 is specified as the word alignment pattern
- A rising edge on the `rx_bitslip` signal at time n + 1 slips a single bit 0 at the MSB position, forcing the `rx_dataout` to 8'b01111000
- Another rising edge on the `rx_bitslip` signal at time n + 5 forces `rx_dataout` to 8'b00111100
- Another rising edge on the `rx_bitslip` signal at time n + 9 forces `rx_dataout` to 8'b00011110
- Another rising edge on the `rx_bitslip` signal at time n + 13 forces `rx_dataout` to 8'b00001111. At this instance, `rx_dataout` in cycles n + 12 and n + 13 are 8'b00011110 and 8'b00001111, respectively, which matches the specified 16-bit alignment pattern 16'b0000111100011110. This results in the assertion of the `rx_patterndetect` signal.

Figure 1–37. Example of Word Aligner Configured in Bit-Slip Mode



Bit-Slip Mode Word Aligner with 16-Bit PMA-PCS Interface Modes

In some Basic double-width configurations with 16-bit PMA-PCS interface, you can configure the word aligner in bit-slip mode by selecting the **Use manual bit slipping mode** option in the ALTGX MegaWizard Plug-In Manager.

The word aligner operation for Basic double-width with 16-bit PMA-PCS interface is similar to the word aligner operation in Basic single-width mode with 8-bit PMA-PCS interface. The only difference is that the bit-slip word aligner in 16-bit PMA-PCS interface modes allows 8-bit and 16-bit word alignment patterns, whereas the bit-slip word aligner in 8-bit PMA-PCS interface modes allows only a 16-bit word alignment pattern.

Word Aligner in Double-Width Mode with 20-Bit PMA-PCS Interface Modes

A 20-bit PMA-PCS interface is supported only in Basic double-width mode.

Table 1–11 shows the word aligner configurations allowed in functional modes with a 20-bit PMA-PCS interface.

Table 1–11. Word Aligner in 20-Bit PMA-PCS Interface Modes for Arria II Devices

Functional Mode	Allowed Word Aligner Configurations	Allowed Word Alignment Pattern Length
Basic double-width	Manual alignment, Bit-slip	7 bits, 10 bits, 20 bits

Manual Alignment Mode Word Aligner with 20-Bit PMA-PCS Interface Modes

The word aligner operation in Basic double-width mode with 20-bit PMA-PCS interface is similar to the word aligner operation in Basic double-width mode with a 16-bit PMA-PCS interface. The only difference is that the manual alignment mode word aligner in 20-bit PMA-PCS interface modes allows 7-, 10-, and 20-bit word alignment patterns, whereas the manual alignment mode word aligner in 16-bit PMA-PCS interface modes allows only 8-, 16-, and 32-bit word alignment patterns.

Bit-Slip Mode Word Aligner with 20-Bit PMA-PCS Interface Modes

In some Basic single-width configurations with a 20-bit PMA-PCS interface, you can configure the word aligner in bit-slip mode by selecting the **Use manual bit slipping mode** option in the ALTGX MegaWizard Plug-In Manager.

The word aligner operation for Basic double-width with 20-bit PMA-PCS interface is similar to the word aligner operation in Basic single-width mode with an 8-bit PMA-PCS interface. The difference is that the bit-slip word aligner in 20-bit PMA-PCS interface modes allows only 7-, 10-, and 20-bit word alignment patterns, whereas the bit-slip word aligner in 8-bit PMA-PCS interface modes allows only a 16-bit word alignment pattern.

Table 1–12 summarizes the word aligner options available in Basic single-width and double-width modes.

Table 1–12. Word Aligner Options Available in Basic Single-Width and Double-Width Modes for Arria II Devices *(Note 1)* (Part 1 of 2)

Functional Mode	PMA-PCS Interface Width	Word Alignment Mode	Word Alignment Pattern Length	rx_enapatternalign Sensitivity	rx_syncstatus Behavior	rx_patterndetect Behavior
Basic Single-Width	8-bit	Manual Alignment	16-bit	Rising Edge Sensitive	Asserted high for one parallel clock cycle when the word aligner aligns to a new word boundary.	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.
		Bit-Slip	16-bit	—	—	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.
	10-bit	Manual Alignment	7- and 10-bit	Level Sensitive	Asserted high for one parallel clock cycle when the word aligner aligns to a new word boundary.	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.
		Bit-Slip	7- and 10-bit	—	—	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.
		Automatic Synchronization State Machine	7- and 10-bit	—	Stays high as long as the synchronization conditions are satisfied.	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.

Table 1–12. Word Aligner Options Available in Basic Single-Width and Double-Width Modes for Arria II Devices *(Note 1)* (Part 2 of 2)

Functional Mode	PMA-PCS Interface Width	Word Alignment Mode	Word Alignment Pattern Length	rx_enapatternalign Sensitivity	rx_syncstatus Behavior	rx_patterndetect Behavior
Basic Double-Width	16-bit	Manual Alignment	8-, 16-, and 32-bit	Rising Edge Sensitive	Stays high after the word aligner aligns to the word alignment pattern. Goes low on receiving a rising edge on `rx_enapatternalign` until a new word alignment pattern is received.	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.
		Bit-Slip	8-, 16-, and 32-bit	—	—	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.
	20-bit	Manual Alignment	7-, 10-, and 20-bit	Rising Edge Sensitive	Stays high after the word aligner aligns to the word alignment pattern. Goes low on receiving a rising edge on `rx_enapatternalign` until a new word alignment pattern is received.	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.
		Bit-Slip	7-, 10-, and 20-bit	—	—	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.

Note to Table 1–12:

(1) For more information about word aligner operation, refer to "Word Aligner" on page 1–30.

Automatic Synchronization State Machine Mode

You must use this mode with 8B/10B encoded data if the input data to the word aligner is 10 bits.

Protocols such as PCIe, XAUI, Gigabit Ethernet, and SRIO require the receiver PCS logic to implement a synchronization state machine to provide hysteresis during link synchronization. Each of these protocols defines a specific number of synchronization code groups that the link must receive to acquire synchronization and a specific number of erroneous code groups that it must receive to fall out of synchronization.

The Quartus II software configures the word aligner in automatic synchronization state machine mode for PCIe, XAUI, Gigabit Ethernet, and SRIO functional modes. It automatically selects the word alignment pattern length and the word alignment pattern as specified by each protocol. In each of these functional modes, the protocol-compliant synchronization state machine is implemented in the word aligner.

By using Basic functional mode with the 10-bit PMA-PCS interface, you can configure the word aligner in automatic synchronization state machine mode by selecting the **Use the automatic synchronization state machine** option in the ALTGX MegaWizard Plug-In Manager. Basic mode also allows you to program a custom 7-bit or 10-bit word alignment pattern that the word aligner uses for synchronization.

Table 1–13 lists the synchronization state machine parameters that the Quartus II software allows in supported functional modes. The synchronization state machine parameters are fixed for PCIe, XAUI, GbE, and SRIO modes as specified by the respective protocol. You can program these parameters as suited to your proprietary protocol implementation for Basic mode.

Table 1–13. Synchronization State Machine Parameters for Arria II Devices

Parameter	PCIe	XAUI	GbE	SRIO	Basic Mode
Number of valid synchronization code groups or ordered sets received to achieve synchronization	4	4	3	127	1–256
Number of erroneous code groups received to lose synchronization	17	4	4	3	1–64
Number of continuous good code groups received to reduce the error count by one	16	4	4	255	1–256

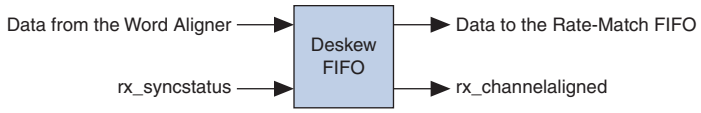
After de-assertion of the `rx_digitalreset` signal in automatic synchronization state machine mode, the word aligner starts looking for the word alignment pattern or synchronization code groups in the received data stream. When the programmed number of valid synchronization code groups or ordered sets is received, the `rx_syncstatus` signal is driven high to indicate that synchronization is acquired. The `rx_syncstatus` signal is constantly driven high until the programmed number of erroneous code groups is received without receiving intermediate good groups; after which the `rx_syncstatus` signal is driven low. The word aligner indicates loss of synchronization (`rx_syncstatus` signal remains low) until the programmed number of valid synchronization code groups are received again.

Deskew FIFO

Use this module, only available in XAUI mode, to align all four channels to meet the XAUI maximum skew of 40 UI (12.8 ns) as seen at the receiver of the four lanes. The deskew operation in XAUI functional mode is compliant to the PCS deskew state machine diagram specified in 8 of the IEEE P802.3ae.

The deskew circuitry consists of a 16-word deep deskew FIFO in each of the four channels and control logic in the `CMU0` channel of the transceiver block that controls the deskew FIFO write and read operations in each channel. Figure 1–38 shows the deskew FIFO block diagram.

Figure 1–38. Deskew FIFO ***(Note 1)***



Note to Figure 1–38:

(1) For more information about the deskew FIFO operation, refer to "XAUI" on page 1–79.

Rate-Match FIFO

In asynchronous systems, you can clock the upstream transmitter and local receiver with independent reference clocks. Frequency differences in the order of a few hundred PPM can corrupt the data when latching from the recovered clock domain (the same clock domain as the upstream transmitter reference clock) to the local receiver reference clock domain.

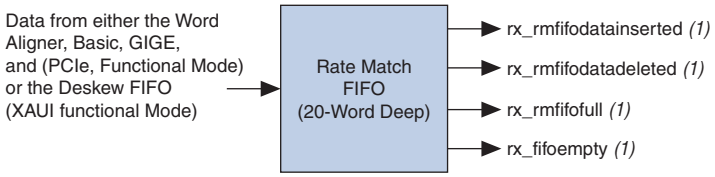
The rate match FIFO compensates for small clock frequency differences between the upstream transmitter and the local receiver clocks by inserting or removing SKP symbols or ordered-sets from the IPG or idle streams. It deletes SKP symbols or ordered-sets when the upstream transmitter reference clock frequency is higher than the local receiver reference clock frequency. It inserts SKP symbols or ordered-sets when the local receiver reference clock frequency is higher than the upstream transmitter reference clock frequency.

The rate match FIFO consists of a 20-word deep FIFO and necessary logic that controls insertion and deletion of a SKP character or ordered-set, depending on the PPM difference. This module is optional in Basic functional mode, but is mandatory and cannot be bypassed in GbE, PCIe, and XAUI functional modes.

For the Gigabit Ethernet protocol, if you enabled rate match FIFO in the auto-negotiation state machine in an FPGA core, refer to "Rate Match FIFO in GbE Mode" on page 1–61.

Figure 1–39 shows the rate-match FIFO block diagram.

Figure 1–39. Rate Match FIFO



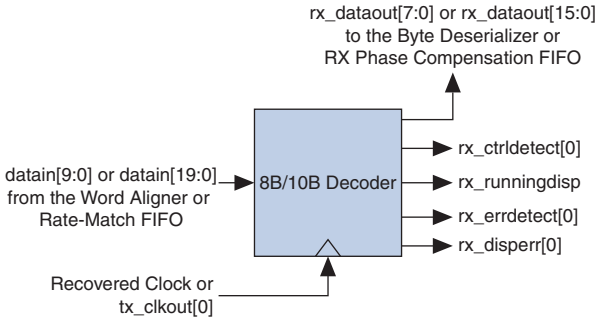
Note to Figure 1–39:

(1) These signals are not available in PCIe functional mode because the rate match FIFO status is encoded in the `pipestatus[2:0]` signal.

8B/10B Decoder

Figure 1–40 shows the block diagram of the 8B/10B decoder.

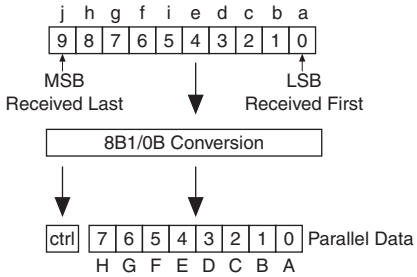
Figure 1–40. 8B/10B Decoder



The 8B/10B decoder is compliant to Clause 36 in the IEEE802.3 specification, decoding the 10-bit data input into an 8-bit data and a 1-bit control identifier. This module is required in GbE, PCIe, SRIO, and XAUI functional modes, but is optional in Basic functional mode.

Figure 1–41 shows a 10-bit code group decoded into an 8-bit data and a 1-bit control identifier by the 8B/10B decoder.

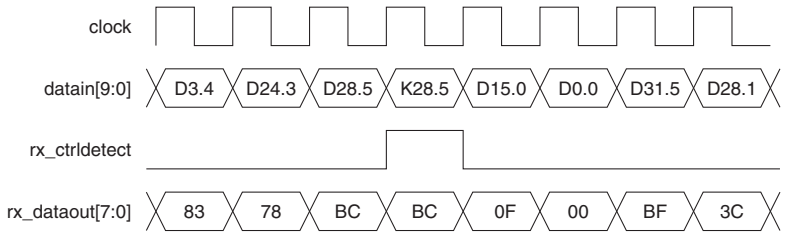
Figure 1–41. 8B/10B Decoder Operation



The 8B/10B decoder indicates whether the decoded 8-bit code group is a data or control code group on the `rx_ctrldetect` port. If the received 10-bit code group is one of the 12 control code groups (/Kx.y/) specified in the IEEE802.3 specification, the `rx_ctrldetect` signal is driven high. If the received 10-bit code group is a data code group (/Dx.y/), the `rx_ctrldetect` signal is driven low.

Figure 1–42 shows the 8B/10B decoder decoding the received 10-bit /K28.5/ control code group into an 8-bit data code group (8'hBC) driven on the `rx_dataout` port. The `rx_ctrldetect` signal is asserted high synchronous with 8'hBC on the `rx_dataout` port, indicating that it is a control code group. The rest of the codes received are data code groups /Dx.y/.

Figure 1–42. 8B/10B Decoder in Control Code Group Detection



Byte Deserializer

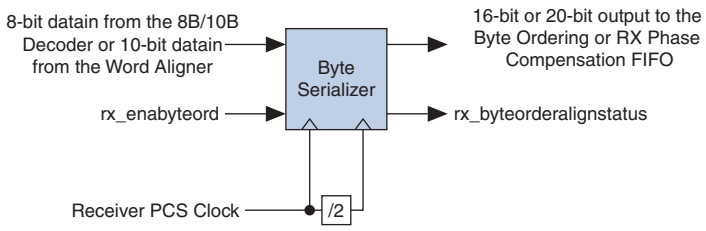
Some serial data rates violate the maximum frequency for the FPGA fabric-to-transceiver interface. In such configurations, the byte deserializer is required to reduce the FPGA fabric-to-transceiver interface frequency to half while doubling the parallel data width. This module is optional in configurations that do not exceed the FPGA fabric-to-transceiver interface clock upper frequency limit.

For example, at a 3.2 Gbps data rate with a deserialization factor of 10, the receiver PCS datapath runs at 320 MHz. The byte serializer converts the 10-bit parallel received data at 320 MHz into 20-bit parallel data at 160 MHz before forwarding it to the FPGA fabric.

In Arria II GX devices, you cannot enable the byte deserializer in double-width mode. However, in Arria II GZ devices, you can enable both double-width mode and the byte deserializer to achieve a 32- or 40-bit PCS-FPGA interface.

Figure 1–43 shows the block diagram of the byte deserializer with 8-bit or 10-bit PMA-to-PCS interface.

Figure 1–43. Byte Deserializer

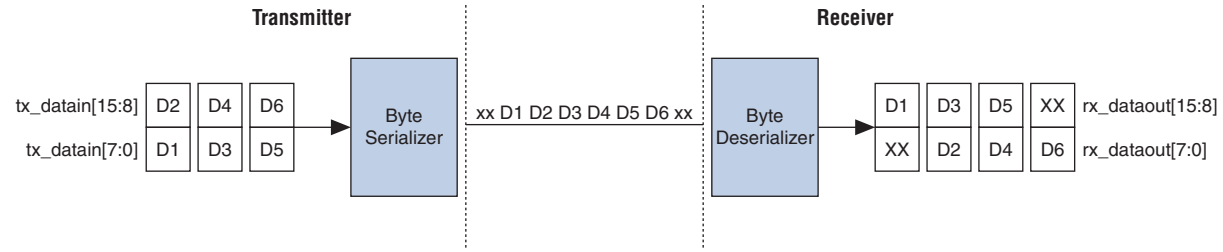


Byte Ordering

Depending on when the receiver PCS logic comes out of reset, byte ordering at the output of the byte deserializer may not match the original byte ordering of the transmitted data. The byte misalignment resulting from byte deserialization is unpredictable because it depends on which byte is being received by the byte deserializer when it comes out of reset.

Figure 1–44 shows a scenario in which the MSByte and LSByte of the two-byte transmitter data appears straddled across two word boundaries after getting byte deserialized at the receiver.

Figure 1–44. MSByte and LSByte of the Two-Byte Transmitter Data Straddled Across Two Word Boundaries



The byte ordering block looks for the user-programmed byte ordering pattern in the byte-deserialized data. You must select a byte ordering pattern that you know appears at the LSByte(s) position of the parallel transmitter data. If the byte ordering block finds the programmed byte ordering pattern in the MSbyte(s) position of the byte-deserialized data, it inserts the appropriate number of user-programmed PAD bytes to push the byte ordering pattern to the LSByte(s) position, thereby restoring proper byte ordering.

The byte ordering block is available in the following functional modes:

- SONET/SDH OC-48—The Quartus II software automatically configures the byte ordering pattern and byte ordering PAD pattern in this mode.
- SONET/SDH OC-96—For Arria II GZ devices only.
- Basic mode with 16-bit FPGA fabric-to-transceiver interface, no 8B/10B decoder (8-bit PMA-PCS interface) and word aligner in manual alignment mode—You can program a custom 8-bit byte ordering pattern and 8-bit byte ordering PAD pattern in the ALTGX MegaWizard Plug-In Manager.
- Basic mode with 16-bit FPGA fabric-to-transceiver interface, 8B/10B decoder, and word aligner in automatic synchronization state machine mode—You can program a custom 9-bit byte ordering pattern and 9-bit byte ordering PAD pattern in the ALTGX MegaWizard Plug-In Manager. If a /Kx.y/ control code group is selected as the byte ordering pattern, the MSB of the 9-bit byte ordering pattern must be 1'b1. If a /Dx.y/ data code group is selected as the byte ordering pattern, the MSB of the 9-bit byte ordering pattern must be 1'b0. The least significant 8 bits must be the 8B/10B decoded version of the code group used for byte ordering.

The byte ordering modules have two different modes of operation when you select the rx_syncstatus signal from the word aligner option in the **What do you want the byte ordering to be based on?** field in the ALTGX MegaWizard Plug-In Manager:

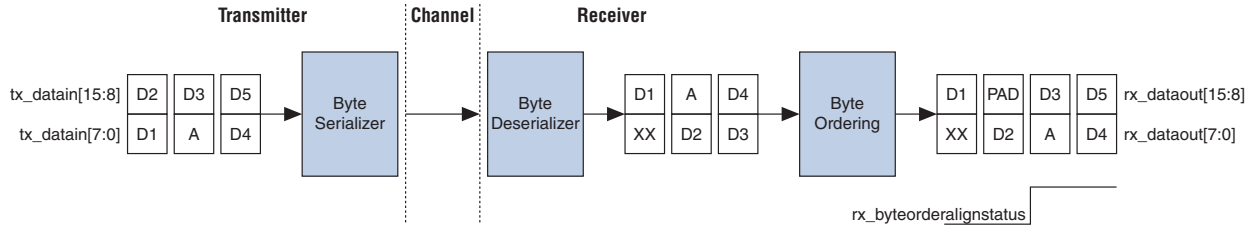
- Word-Alignment-Based Byte Ordering—In this mode, the byte ordering block starts looking for the byte ordering pattern in the byte-deserialized data every time it sees a rising edge on the rx_syncstatus signal.

  If the byte ordering block finds the first data byte that matches the programmed byte ordering pattern in the MSByte position of the byte-deserialized data, it inserts one programmed PAD pattern to push the byte ordering pattern into the LSByte position after a rising edge on the rx_syncstatus signal.

  If the byte ordering blocks finds the first data byte that matches the programmed byte ordering pattern in the LSByte position of the byte-deserialized data, it considers the data to be ordered and does not insert any PAD pattern. In either case, the byte ordering block asserts the rx_byteorderalignstatus signal.

  Figure 1–45 shows an example of the byte ordering operation where A is the programmed byte ordering pattern and PAD is the programmed PAD pattern. The byte deserialized data places the byte ordering pattern A in the MSByte position, resulting in incorrect byte ordering. Assuming that a rising edge on the rx_syncstatus signal had occurred before the byte ordering block sees the byte ordering pattern A in the MSByte position, the byte ordering block inserts a PAD byte and pushes the byte ordering pattern A into the LSByte position. The data at the output of the byte ordering block has correct byte ordering as reflected on the rx_byteorderalignstatus signal.

Figure 1–45. Example of Byte Ordering Operation



If the byte ordering block sees another rising edge on the rx_syncstatus signal from the word aligner, it de-asserts the rx_byteorderalignstatus signal and repeats the byte ordering operation.

- User-Controlled Byte Ordering—Unlike word-alignment-based byte ordering, user-controlled byte ordering provides control to the user logic to restore correct byte ordering at the receiver.

  When enabled, an rx_enabyteord port is available as a trigger to the byte ordering operation. A rising edge on the rx_enabyteord port triggers the byte ordering block. If the byte ordering block finds the first data byte that matches the programmed byte ordering pattern in the MSByte position of the byte-deserialized data, it inserts one programmed PAD pattern to push the byte ordering pattern

into the LSByte position after a rising edge on the rx_enabyteord signal. If the byte ordering blocks find the first data byte that matches the programmed byte ordering pattern in the LSByte position of the byte-deserialized data, it considers the data to be byte ordered and does not insert any PAD byte. In either case, the byte ordering block asserts the rx_byteorderalignstatus signal.

RX Phase Compensation FIFO

The RX phase compensation FIFO in each channel ensures reliable transfer of data and status signals between the receiver channel and the FPGA fabric. The RX phase compensation FIFO compensates for the phase difference between the parallel receiver PCS clock (FIFO write clock) and the FPGA fabric clock (FIFO read clock).

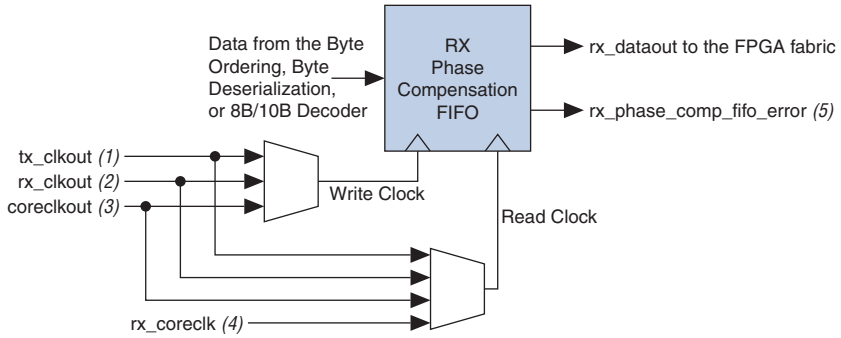
Table 1–14 lists the RX phase compensation FIFO modes. The Quartus II software automatically sets the mode that applies to a particular functional mode.

Table 1–14. RX Phase Compensation FIFO Modes for Arria II Devices

Mode	FIFO Depth	Latency Through FIFO	Applicable Functional Modes
Low Latency	4-words deep	2-to-3 parallel clock cycles	All functional modes except PCIe and Deterministic Latency
High-Latency	8-words deep	4-to-5 parallel clock cycles	PCIe
Register	—	1	Deterministic Latency

Figure 1–46 shows the RX phase compensation FIFO block diagram. The write clock and read clock to the FIFO are at half-rate if the byte serializer is used in the configuration.

Figure 1–46. RX Phase Compensation FIFO Block Diagram



Notes to Figure 1–46:

(1) The tx_clkout clock is from the transmitter channel clock divider and is used in non-bonded configurations with the rate match FIFO.

(2) The rx_clkout clock is from the receiver channel CDR and is used in non-bonded configurations that does not use the rate match FIFO.

(3) The coreclkout clock is from the CMU0 block of the associated transceiver block or the master transceiver block for ×4 bonded or ×8 bonded channel configurations, respectively.

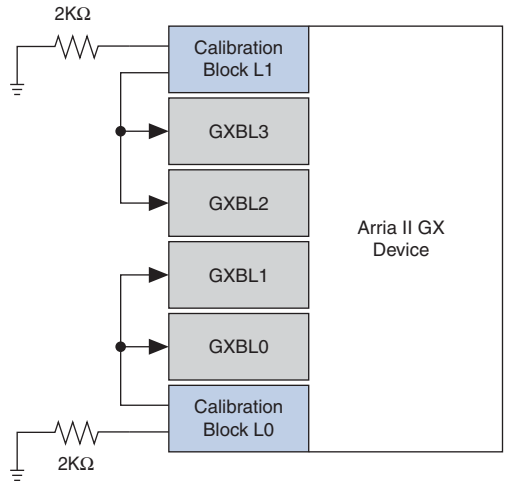
(4) Use this clock for the FIFO read clock if you instantiate the rx_coreclk port in the ALTGX MegaWizard Plug-In Manager, regardless of the channel configurations. Otherwise, use the same clock used for the write clock for the read clock.

(5) The rx_phase_comp_fifo_error is optional and available in all functional modes. This signal is asserted high to indicate an overflow or underflow condition.

Calibration Block

This block calibrates the OCT resistors and the analog portions of the transceiver blocks to ensure that the functionality is independent of process, voltage, or temperature (PVT) variations. Figure 1–47 shows the location of the calibration block and how it is connected to the transceiver blocks in Arria II GX devices.

Figure 1–47. Arria II GX Transceiver Calibration Blocks Location and Connection *(Note 1)*, *(2)*, *(3)*



Notes to Figure 1–47:

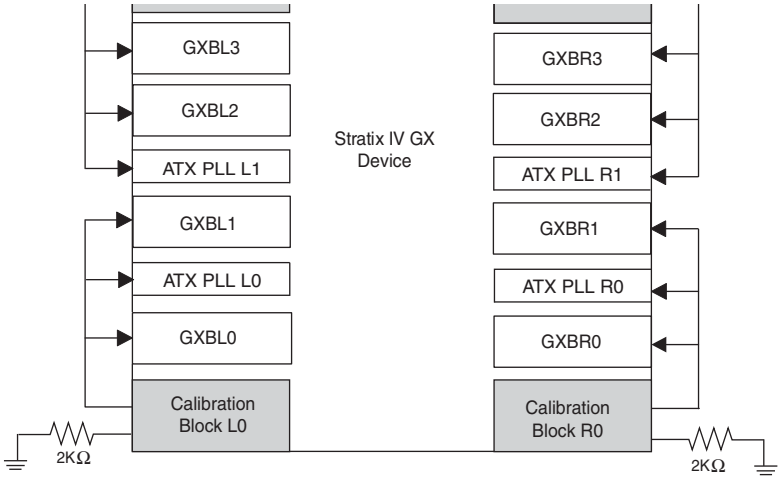
(1) You must connect a separate 2 kΩ (tolerance max ± 1%) external resistor on each RREF pin in the Arria II device to GND. To ensure proper operation of the calibration block, the RREF resistor connection in the board must be free from external noise.

(2) The Quartus II software automatically selects the appropriate calibration block based on the pin assignments of the transceiver tx_dataout and rx_datain pins.

(3) For RREF pin information, refer to the *Device Pin Connection Guidelines*.

Figure 1–48 shows the location of the calibration block and how it is connected to the transceiver blocks in Arria II GZ devices.

Figure 1–48. Arria II GZ Transceiver Calibration Blocks Location and Connection *(Note 1)*, *(2)*, *(3)*



Notes to Figure 1–48:

(1) You must connect a separate 2 kΩ (tolerance max ± 1%) external resistor on each RREF pin in the Arria II device to GND. To ensure proper operation of the calibration block, the RREF resistor connection in the board must be free from external noise.

(2) The Quartus II software automatically selects the appropriate calibration block based on the pin assignments of the transceiver tx_dataout and rx_datain pins.

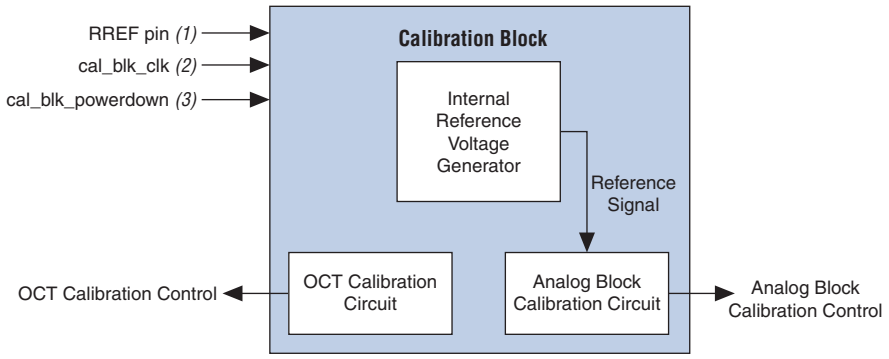
(3) For RREF pin information, refer to the *Device Pin Connection Guidelines*.

The calibration block internally generates a constant internal reference voltage, independent of PVT variations and uses this voltage and the external reference resistor on the RREF pin to generate constant reference currents. These reference currents are used by the analog block calibration circuit to calibrate the transceiver blocks.

The OCT calibration circuit calibrates the OCT resistors present in the transceiver channels. You can enable the OCT resistors in the transceiver channels through the ALTGX MegaWizard Plug-In Manager.

Figure 1–49 shows the calibration block diagram. You can instantiate the `cal_blk_clk` and `cal_blk_powerdown` ports in the ALTGX MegaWizard Plug-In Manager to control the calibration block. Drive the `cal_blk_clk` port of all ALTGX instances that are associated with the same calibration block from the same input pin or logic.

Figure 1–49. Input Signals to the Calibration Blocks



Notes to Figure 1–49:

(1) You must connect a separate 2 kΩ (tolerance max ± 1%) external resistor on each `RREF` pin in the Arria II GX and GZ device to GND. To ensure proper operation of the calibration block, the `RREF` resistor connection in the board must be free from external noise.

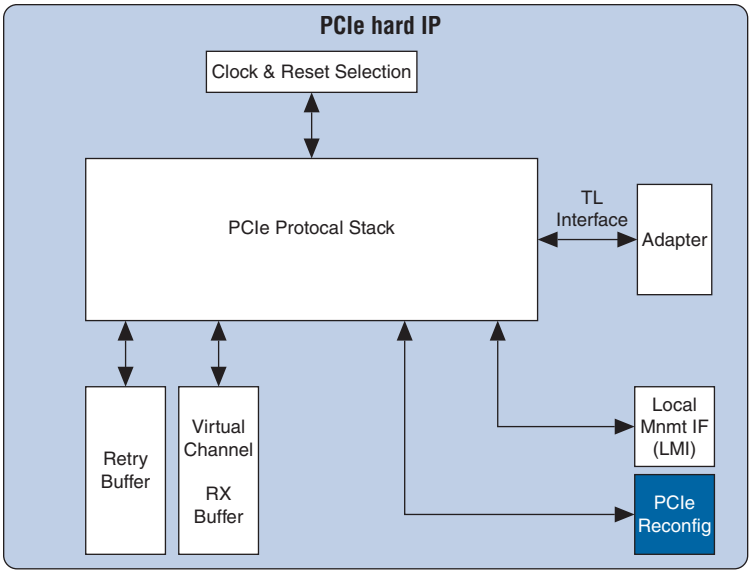
(2) This is the input clock to the calibration block. You can use dedicated clock routes such as the global or regional clock; however, if you do not have a suitable input reference clock or dedicated clock routing resources available, use divide-down logic from the FPGA fabric to generate a slow clock, local clocking routing, or both.

(3) Drive this signal of all ALTGX instances that are associated with the same calibration block from the same input or logic. During calibration, the link may experience higher bit error rates due to changes in the signaling condition. The transceiver on-chip termination calibration process takes approximately 33,000 `cal_blk_clk` cycles from the de-assertion of the `cal_blk_powerdown` signal.

PCIe Hard IP Block

Figure 1–50 shows the block diagram of the PCIe hard IP block used to implement the PHY MAC, Data Link Layer, and Transaction Layer for PCIe interfaces. When you use this block, the PIPE interface is used as the interface between the FPGA fabric and the transceiver.

Figure 1–50. PCIe Hard IP High-Level Block Diagram



This block is enabled only when you use the PCIe MegaCore function. For more information about using this block, refer to the *PCI Express Compiler User Guide*.

You can configure the root port or end point in x1, x4, and x8 modes. You can also include instances of both the soft and hard IP PCIe MegaCore function in a single device.

The PCIe hard IP block ignores autonomous and speed change bits set in TS1 and TS2 ordered sets. Altera is fully compatible to the PCIe Base Specification v2.0.

Functional Modes

You can configure Arria II GX and GZ transceivers in one of the following functional modes with the ALTGX MegaWizard Plug-In Manager:

- Basic at 600 Mbps to 6.375 Gbps
- Deterministic latency, used for CPRI and OBSAI protocols
- GbE (1.25 Gbps)
- PCIe (Gen1 at 2.5 Gbps) (Gen2 at 5.0 Gbps for Arria II GZ devices only)
- SDI (HD at 1.485 and 1.4835 Gbps, 3G at 2.97 and 2.967 Gbps)
- SRIO(1.25 Gbps, 2.5 Gbps, and 3.125 Gbps)
- SONET/SDH (OC-12 and OC-48) (OC-96 for Arria II GZ devices only)

- XAUI (3.125 Gbps up to HiGig and HiGig+ at 3.75 Gbps)

The following sections describe the functional modes available in the ALTGX MegaWizard Plug-In Manager that you can set through the **Which protocol will you be using?** option.

Table 1–15 lists the PCS latency for the different functional modes of Arria II devices.

Table 1–15. Functional Modes PCS Latency for Arria II Devices *(Note 1)*

Functional Mode	PCS Latency (FPGA Fabric-Transceiver Interface Clock Cycles)					
	TX PCS			RX PCS		
Basic	*(2)*			*(2)*		
Deterministic latency	4			6 (byte SERDES enabled)	8 (byte SERDES disabled)	
GbE	5–6			20–24		
PCIe	4–5.5 (byte SERDES enabled)	5–6 (byte SERDES disabled)		11.5–14.5 (byte SERDES enabled)	20–24 (byte SERDES disabled)	
SDI	4–5.5 (byte SERDES enabled)	5–6 (byte SERDES disabled)		6–8 (byte SERDES enabled)	9–11 (byte SERDES disabled)	
SRIO	4–5			6–7 (Rate Match FIFO disable)	19–20 (Rate Match FIFO enable)	
SONET/SDH	5–6 (OC-12)	4–5.5 (OC-48)	4–5.5 (OC-96)	11–13 (OC-12)	7–9 (OC-48)	6.5–8.5 (OC-96)
XAUI	4.5–6			14.5–18		

Notes to Table 1–15:

(1) Not all modes are support by both Arria II GX and Arria II GZ devices. Refer to the respective functional mode sections for the supported modes in each device family.

(2) For basic mode latency values, refer to Figure 1–51, Figure 1–52, Figure 1–53, and Figure 1–54.

Basic Mode

The Arria II GX and GZ transceiver datapath is highly flexible in Basic functional mode. It allows 8-bit and 10-bit PMA-to-PCS interface, which is determined by whether you bypass or use the 8B/10B encoder/decoder.

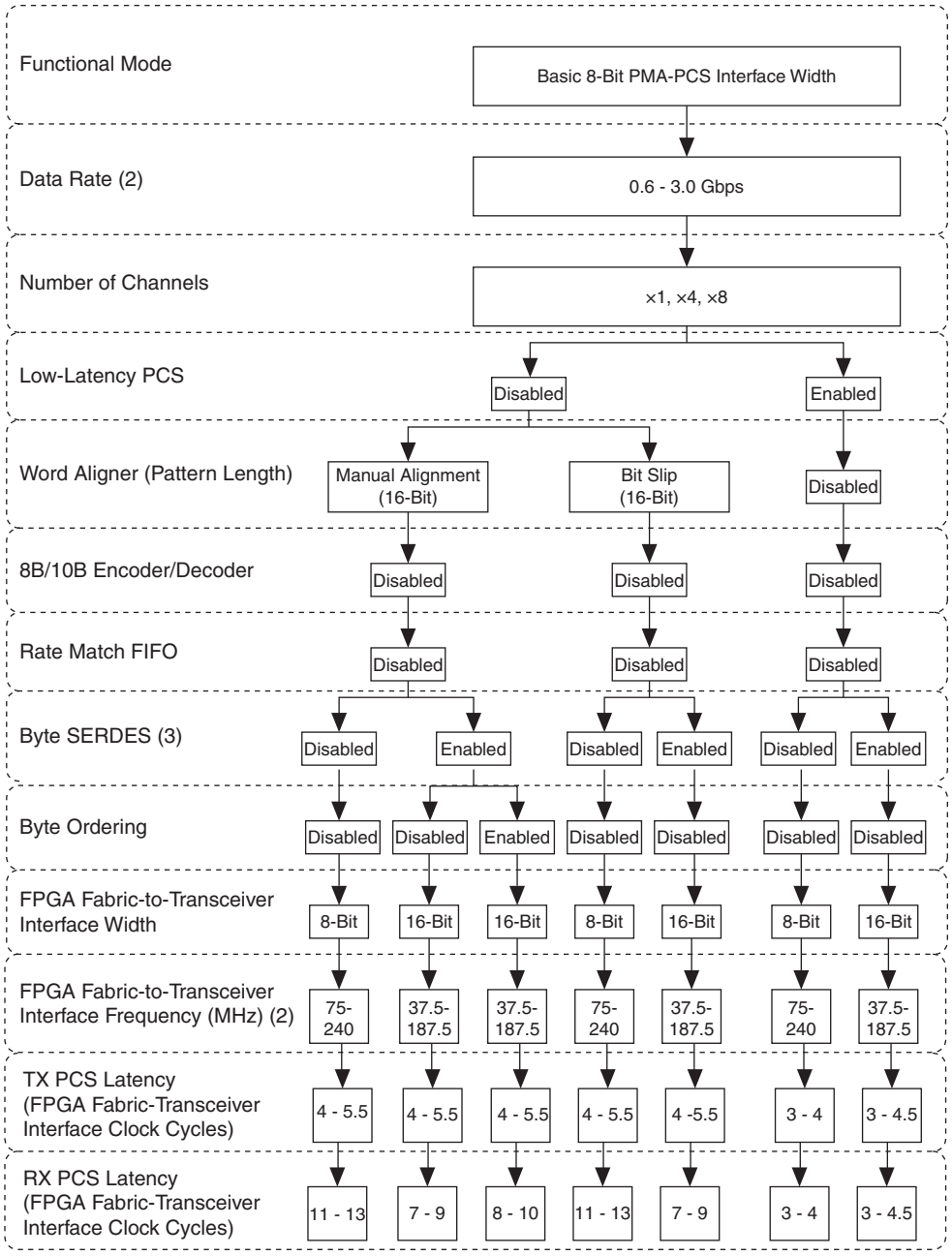
Depending on the targeted data rate, you can optionally bypass the byte serializer and deserializer blocks in Basic mode but the transmitter and RX phase compensation FIFOs are always enabled. The word aligner is always enabled in regular Basic mode, but bypassed in low latency PCS mode, which can be enabled through the **Enable low latency PCS mode** option in the ALTGX MegaWizard Plug-In Manager.

The low latency PCS mode creates a Basic functional mode configuration that bypasses the following transmitter and receiver channel PCS blocks to form a low latency PCS datapath:

- 8B/10B encoder and decoder
- Word aligner
- Deskew FIFO
- Rate match (clock rate compensation) FIFO
- Byte ordering

Figure 1–51 shows the Arria II GX and GZ transceiver configurations allowed in Basic functional mode with an 8-bit wide PMA-PCS interface.

Figure 1–51. Transceiver Configurations in Basic Mode with an 8-Bit Wide PMA-to-PCS Interface *(Note 1)*



Notes to Figure 1–51:

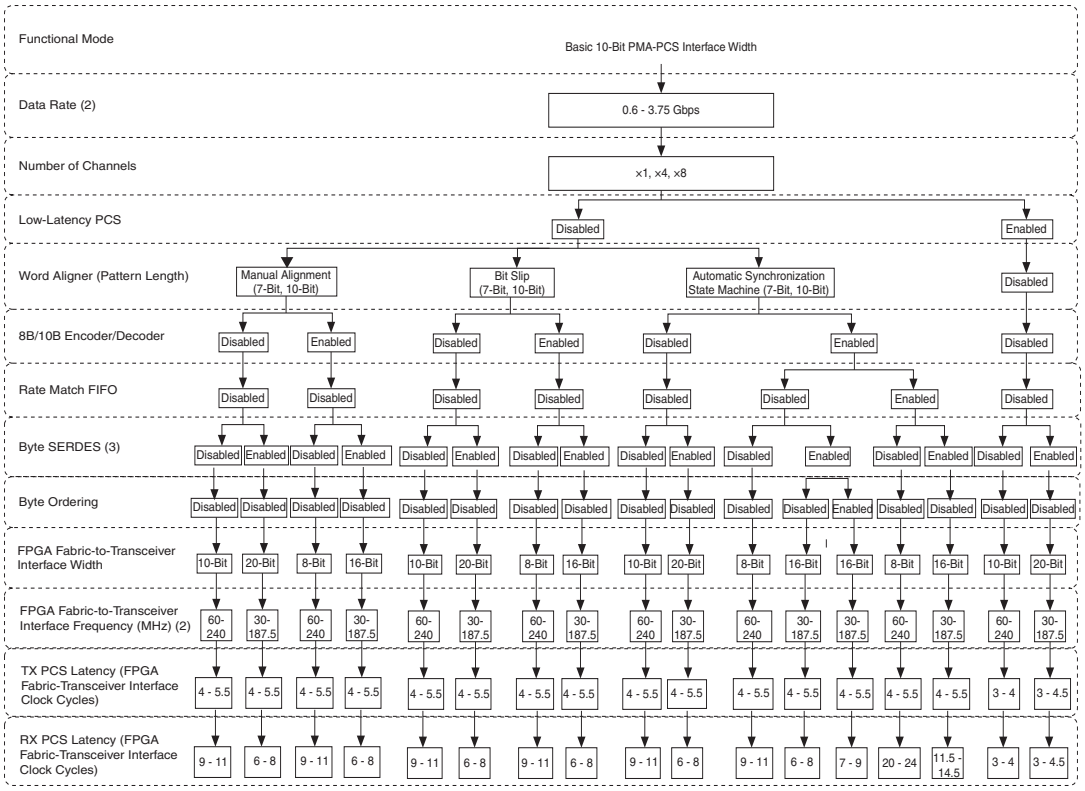
(1) The 10-bit configuration is listed in Figure 1–52.

(2) The maximum data rate specification shown in Figure 1–51 is valid only for the -3 speed grade devices. For data rate specifications for other speed grades offered, refer to the *Device Datasheet for Arria II Devices* chapter.

(3) When you enable byte SERDES, the maximum data is 3G; otherwise, it is 1.92G.

Figure 1–52 shows Arria II GX and GZ transceiver configurations allowed in Basic functional mode with a 10-bit wide PMA-to-PCS interface.

Figure 1–52. Transceiver Configurations in Basic Mode with a 10-Bit Wide PMA-to-PCS Interface ***(Note 1)***



Notes to Figure 1–52:

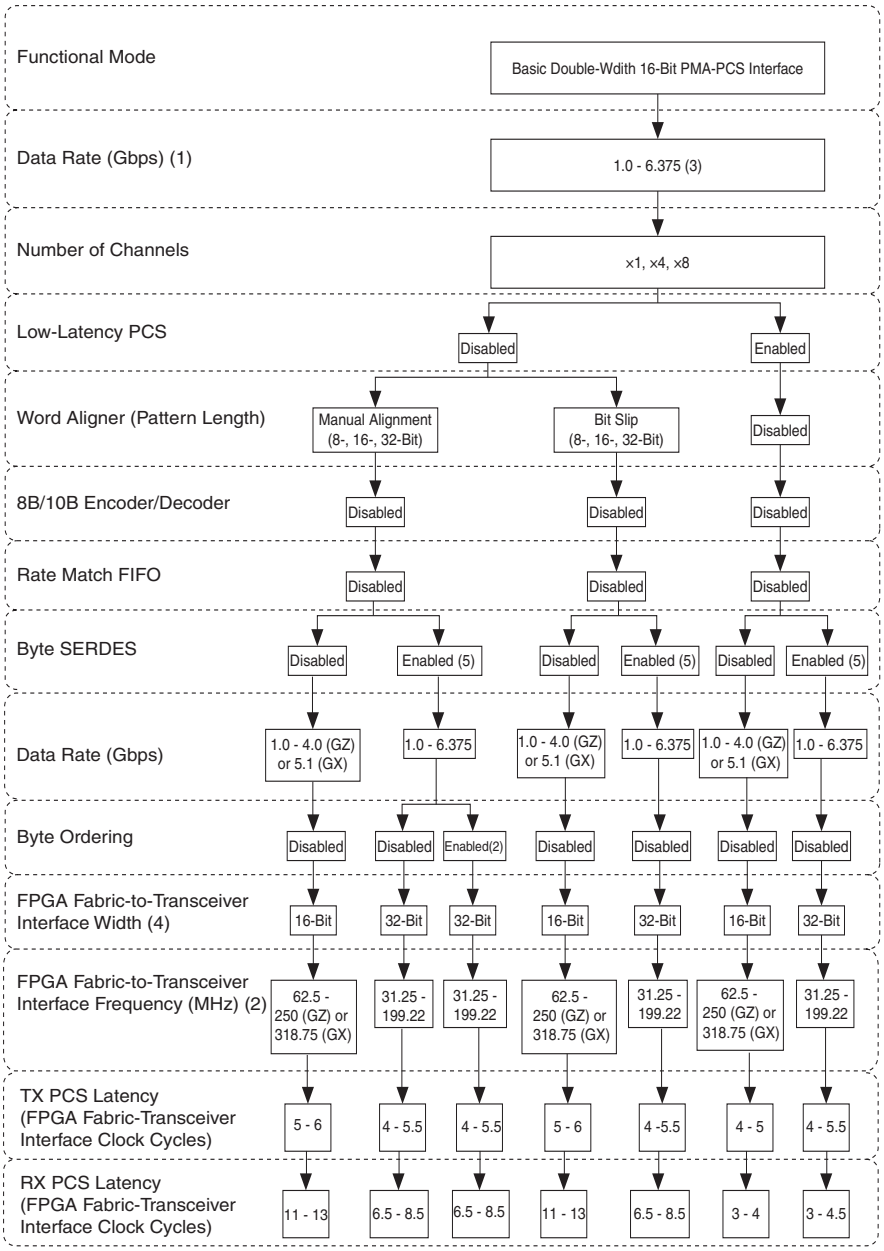
(1) The 8-bit configuration is listed in Figure 1–51.

(2) The maximum data rate specification shown in Figure 1–52 is valid only for the -3 speed grade devices with byte SERDES enabled. For data rate specifications for other speed grades offered, refer to the *Device Datasheet for Arria II Devices* chapter.

(3) When you enable byte SERDES, the maximum data is 3G; otherwise, it is 1.92G.

Figure 1–53 shows Arria II transceiver configurations allowed in Basic double-width functional mode with a 16-bit PMA-PCS interface.

Figure 1–53. Transceiver Configurations in Basic Double-Width Mode with a 16-Bit PMA-PCS Interface for Arria II Devices

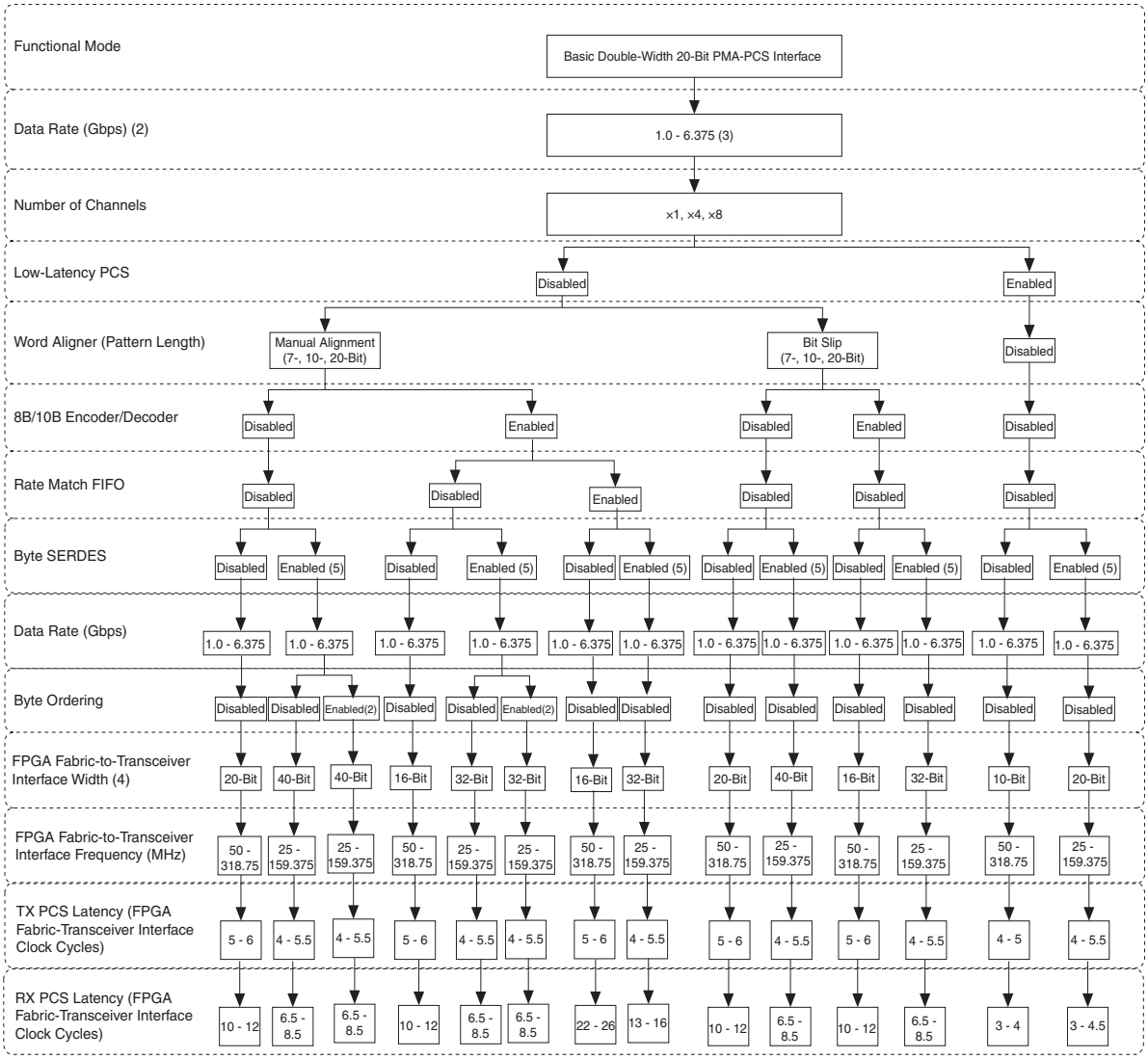


Notes to Figure 1–53:

(1) The maximum data rate specification shown in Figure 1–53 is valid only for the -3 (fastest) speed grade devices.
(2) The byte ordering block is available only if you select the word alignment pattern length of 16 or 32 bits.
(3) Arria II GX I3 devices can support up to 6.375 Gbps. For more information, refer to the *Device Datasheet for Arria II Devices*.
(4) The FPGA fabric-to-transceiver interface width of 32-bits applies to Arria II GZ devices only.
(5) Byte SERDES is only supported for Arria II GZ devices in double-wide mode.

Figure 1–54 shows Arria II transceiver configurations allowed in Basic double-width functional mode with a 20-bit PMA-PCS interface.

Figure 1–54. Transceiver Configurations in Basic Double-Width Mode with a 20-Bit PMA-PCS Interface for Arria II Devices



Notes to Figure 1–54:

(1) The maximum data rate specification shown in Figure 1–54 is valid only for the -3 (fastest) speed grade devices.

(2) The byte ordering block is available only if you select the word alignment pattern length of 20 bits.

(3) Arria II GX I3 devices can support up to 6.375 Gbps. For more information, refer to the *Device Datasheet for Arria II Devices*.

(4) The FPGA fabric-to-transceiver interface width of 40-bits applies to Arria II GZ devices only.

(5) Byte SERDES is only supported for Arria II GZ devices in double-wide mode.

Deterministic Latency

This mode is typically used to create a CPRI or Open Base Station Architecture Initiative Reference Point 3 (OBSAI RP3) interface to connect radio frequency (RF) processing remote radio heads located at the top of cell phone towers with the base band processing equipment typically found at the bottom of cell phone towers.

CPRI and OBSAI protocols have a requirement for the accuracy of the round trip delay measurement for single-hop and multi-hop connections to be within ± 16.276 ns. For single hops, the round trip delay may only vary within ± 16.276 ns. For multi-hop connections, the round trip variation is equal to ± 16.276 ns divided by the number of hops.

Deterministic latency is the only functional mode that allows 16-bit and 20-bit data on the PCS-to-PMA interface. This is to allow data rates of 2457.6, 3072, 4915.2, and 6144 Mbps for the CPRI protocol and to allow 3072 and 6144 Mbps data rates for the OBSAI protocol.

When you choose the deterministic latency protocol in the ALTGX MegaWizard Plug-In Manager, the bit-slip circuitry in the transmitter channel is automatically enabled and the RX phase compensation FIFO is automatically set to register mode. In addition, two extra ports are created—the `rx_bitslipboundaryselectout` output port from the receiver's word aligner and the `tx_bitslipboundaryselect` input port for the transceiver bit-slip circuitry. You can also set the TX phase compensation FIFO in register mode.

In register mode, the phase compensation FIFO acts as a register and removes the uncertainty in latency. To ensure that the phase relationship between the low-speed parallel clock and the CMU PLL input reference clock is deterministic, you can enable the CMU PLL feedback path, which is only available in this mode. When the feedback path is enabled, you must provide an input reference clock to the CMU PLL that has the same frequency as the low-speed parallel clock.

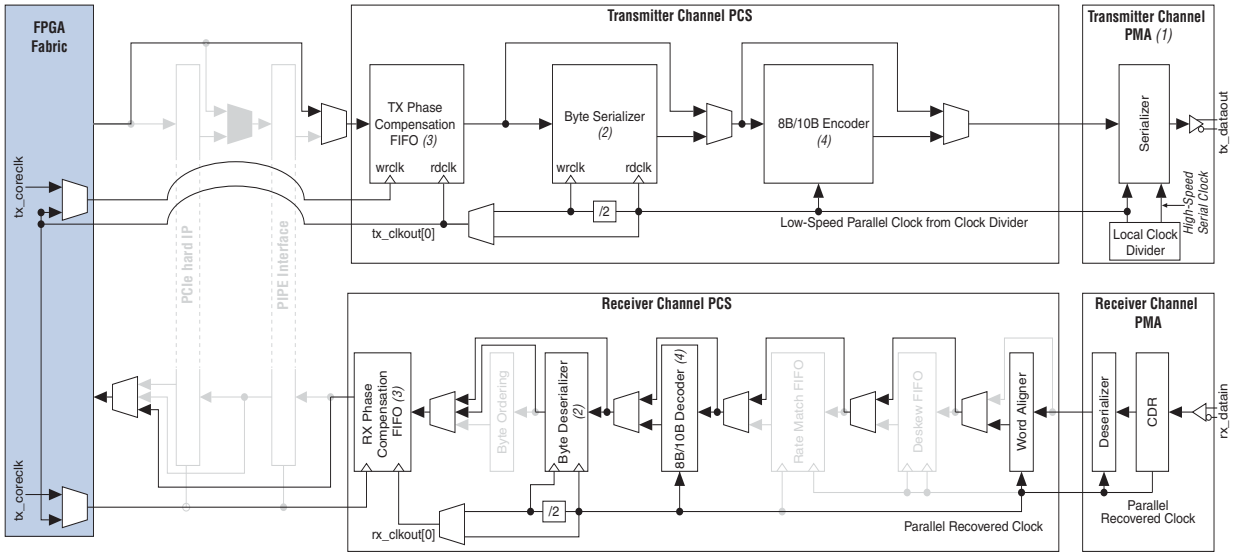
The information on the `rx_bitslipboundaryselectout[4:0]` output port helps calculate the latency through the receiver datapath. Connect `rx_bitslipboundaryselectout[4:0]` to `tx_bitslipboundaryselect[4:0]` to cancel out the latency uncertainty.

The number of bits slipped in the receiver's word aligner is shown on the `rx_bitslipboundaryselectout[4:0]` output port. In 8- or 10-bit channel width, the number of bits slipped in the receiver path is given out sequentially on this output. For example, if zero bits are slipped, the output on `rx_bitslipboundaryselectout[4:0]` shows a value of 0(5'b00000); if two bits are slipped, the output on `rx_bitslipboundaryselectout[4:0]` shows a value of 2 (5'b00010). In 16- or 20-bit channel width, the output is 19 minus the number of bits slipped. For example, if zero bits are slipped, the output on `rx_bitslipboundaryselectout[4:0]` shows a value of 19 (5'b10011); if two bits are slipped, the output on `rx_bitslipboundaryselectout[4:0]` shows a value of 17 (5'b10001).

You can slip zero to nine bits with 8- or 10-bit channel width and you can slip zero to 19 bits with 16- or 20-bit channel width.

Figure 1–55 shows the block diagram for deterministic latency.

Figure 1–55. Deterministic Latency Functional Mode



Notes to Figure 1–55:

(1) The transmitter is in bit-slip mode.

(2) This block is optional in this mode.

(3) The RX phase compensation FIFO is automatically set in register mode. However, you have the option to set the TX phase compensation FIFO in register mode, which is not set by default.

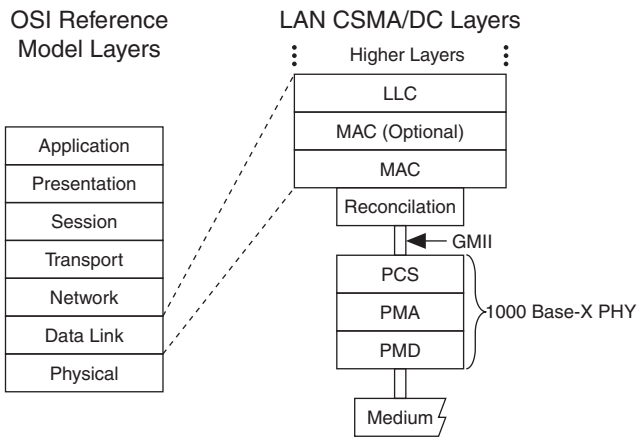
(4) Typically, the 8B/10B encoder and decoder are used when you use deterministic latency to implement CPRI or OBSAI protocols. However, you have the option to disable this module.

GbE

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 GbE system. It shields the MAC layer from the specific nature of the underlying medium. The 1000 Base-X PHY, which has a physical interface data rate of 1.25 Gbps, is divided into three sublayers—the physical coding sublayer (PCS), physical media attachment (PMA), and physical medium dependent (PMD). These sublayers interface with the MAC through the gigabit medium independent interface (GMII).

Figure 1–56 shows the 1000 Base-X PHY position in a GbE OSI reference model.

Figure 1–56. 1000 Base-X PHY in a GbE OSI Reference Model



In GbE functional mode, Arria II GX and GZ devices have built-in circuitry to support the following PCS and PMA functions defined in the IEEE 802.3 specification:

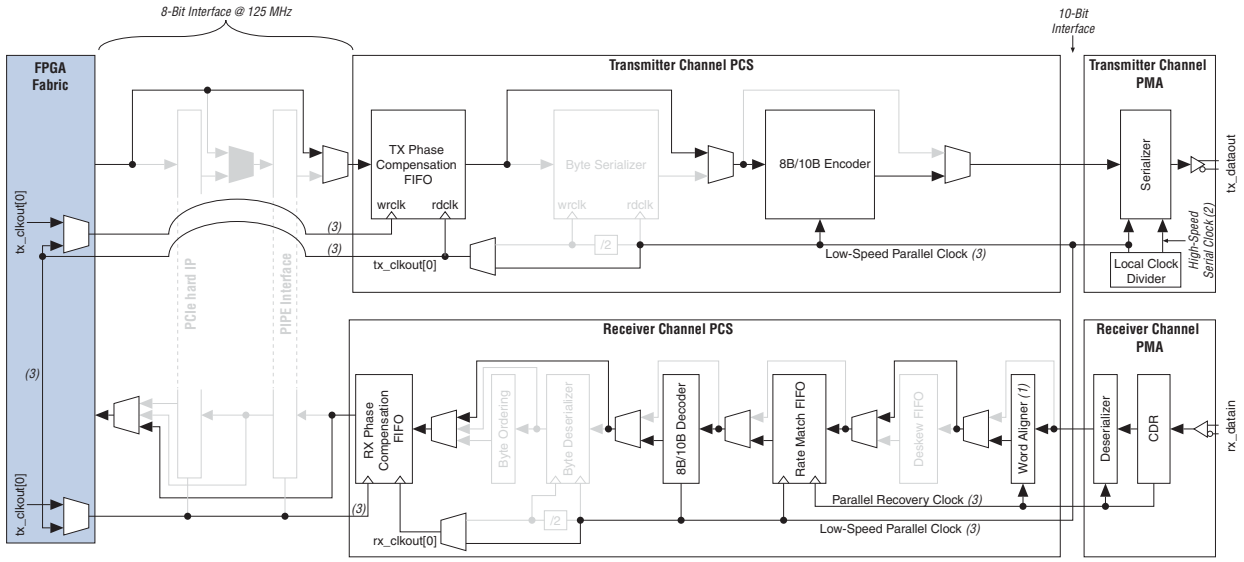
- 8B/10B encoding and decoding
- Synchronization
- Clock recovery from the encoded data forwarded by the receiver PMD
- Optional `rx_recovclkout` port enables the recovered clock at the pin level (use with the voltage-controlled crystal oscillator [VCXO])
- Serialization and deserialization

If you enabled the auto-negotiation state machine in the FPGA core with the rate match FIFO, refer to the "Rate Match FIFO in GbE Mode" on page 1–61.

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

Figure 1–57 shows the transceiver datapath when configured in GbE functional mode.

Figure 1–57. GbE Functional Mode



Notes to Figure 1–57:

(1) The word aligner uses automatic synchronization state machine mode (7-bit comma, 10-Bit /K28.5/).
(2) High-speed serial clock is running at 625 MHz.
(3) These clocks are running at 125 MHz.

Idle Ordered-Set Generation in GbE Mode

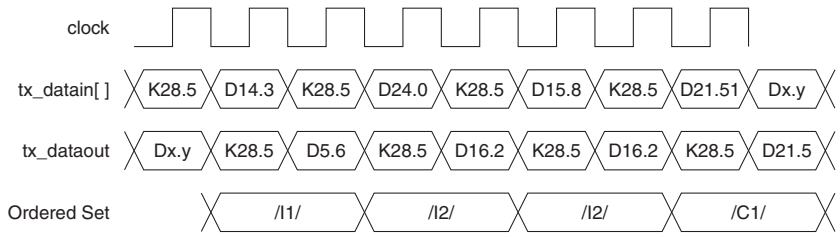
The IEEE 802.3 specification requires the GbE PHY to transmit idle ordered sets (/I/) continuously and repetitively whenever the GMII is idle. This ensures that the receiver maintains bit and word synchronization whenever there is no active data to be transmitted.

In GbE 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, an /I1/ ordered set is generated. If the running disparity is negative, an /I2/ ordered set is generated. The disparity at the end of an /I1/ is the opposite of that at the beginning of the /I1/. The disparity at the end of an /I2/ is the same as the beginning running disparity (right before the idle code). This ensures a negative running disparity at the end of an idle ordered set. A /Kx.y/ following a /K28.5/ is not replaced.

/D14.3/, /D24.0/, and /D15.8/ are replaced by /D5.6/ or /D16.2/ (for /I1/ and /I2/ ordered sets). /D21.5/ (part of the /C1/ order set) is not replaced.

Figure 1–58 shows the automatic idle ordered set generation.

Figure 1–58. Example of Automatic Ordered Set Generation

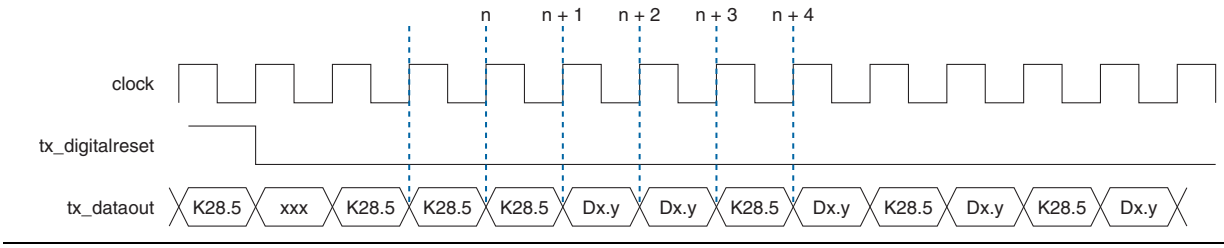


GbE Mode Reset Condition

After de-assertion of the `tx_digitalreset` signal, the GbE transmitter automatically transmits three /K28.5/ comma code groups before transmitting user data on the `tx_datain` port. This could affect the synchronization state machine behavior at the receiver. Depending on when you start transmitting the synchronization sequence, there could be an even or odd number of /Dx.y/ code groups transmitted between the last of the three automatically sent /K28.5/ code groups and the first /K28.5/ code group of the synchronization sequence. If there is an even number of /Dx.y/ code groups received between these two /K28.5/ code groups, the first /K28.5/ code group of the synchronization sequence begins at an odd code group boundary. An IEEE802.3-compliant GbE synchronization state machine treats this as an error condition and goes into the Loss-of-Sync state.

Figure 1–59 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 /K28.5/ code group received at an odd code group boundary in cycle n + 3 takes the receiver synchronization state machine in Loss-of-Sync state. The first synchronization ordered-set /K28.5/Dx.y/ in cycles n + 3 and n + 4 are discounted and three additional ordered sets are required for successful synchronization.

Figure 1–59. Example of Reset Condition in GbE Mode



Word Aligner in GbE Mode

The word aligner in GbE functional mode is configured in automatic synchronization state machine mode, which complies with the IEEE P802.3ae standard. The Quartus II software automatically configures the synchronization state machine to indicate synchronization when the receiver acquires three consecutive synchronization ordered sets. A synchronization ordered set is a /K28.5/ code group followed by an odd number of valid /Dx.y/ code groups. The fastest way for the receiver to achieve synchronization is to receive three continuous {/K28.5/, /Dx.y/} ordered sets.

Receiver synchronization is indicated on the rx_syncstatus port of each channel. A high on the rx_syncstatus port indicates that the lane is synchronized; a low on the rx_syncstatus port indicates that the lane has fallen out of synchronization. Each invalid code group increases the error count. The error count can be reduced by 1 if the state machine sees four continuous valid code groups. The receiver loses synchronization when it detects four invalid code groups separated by less than three valid code groups, or when it is reset.

Rate Match FIFO in GbE Mode

In GbE mode, the rate match FIFO is capable of compensating up to ±100 PPM (200 PPM total) difference between the upstream transmitter and the local receiver reference clock. The GbE protocol requires the transmitter to send idle ordered sets /I1/ (/K28.5/D5.6/) and /I2/ (/K28.5/D16.2/) during inter-packet gaps, adhering to the rules listed in the IEEE P802.3ae specification.

The rate match operation begins after the synchronization state machine in the word aligner indicates synchronization has been acquired by driving the rx_syncstatus signal high. The rate match FIFO deletes or inserts both symbols of the /I2/ ordered sets (which consist of /K28.5/ and /D16.2/) to prevent the rate match FIFO from overflowing or underflowing. It can insert or delete as many /I2/ ordered sets as necessary to perform the rate match operation.

If you have the auto-negotiation state machine in the FPGA, note that the rate match FIFO is capable of inserting or deleting the first two bytes (/K28.5//D2.2/) of /C2/ ordered sets during auto-negotiation. However, the insertion or deletion of the first two bytes of /C2/ ordered sets can cause the auto-negotiation link to fail. For more information, refer to the Altera Knowledge Base Support Solution.

The status flags rx_rmfifodatadeleted and rx_rmfifodatainserted, indicating rate match FIFO deletion and insertion events, respectively, are forwarded to the FPGA fabric. These two flags are asserted for two clock cycles for each deleted and inserted /I2/ ordered set, respectively.

Figure 1–60 shows an example of rate match FIFO deletion where three symbols must be deleted. Because the rate match FIFO can only delete /I2/ ordered sets, it deletes two /I2/ ordered sets (four symbols deleted).

Figure 1–60. Example of Rate Match Deletion in GbE Mode

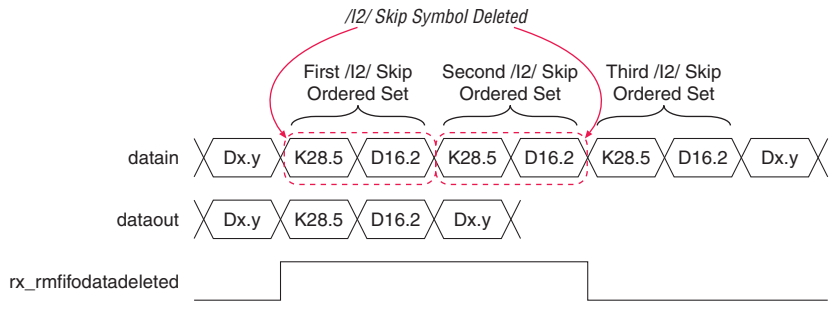
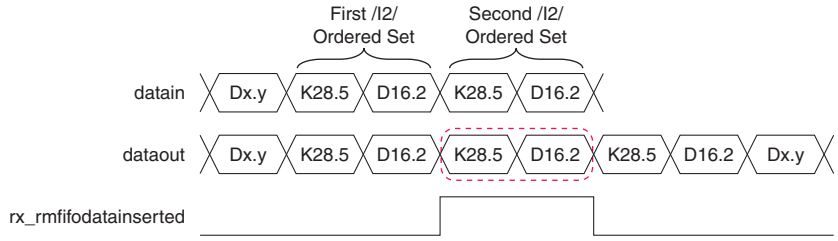


Figure 1–61 shows an example of rate match FIFO insertion in the case where one symbol must be inserted. Because the rate match FIFO can only insert /I2/ ordered sets, it inserts one /I2/ ordered set (two symbols inserted).

Figure 1–61. Example of Rate Match Insertion in GbE Mode



The rate match FIFO does not insert or delete code groups automatically to overcome FIFO empty or full conditions. In this case, the rate match FIFO asserts the `rx_rmfifofull` and `rx_rmfifoempty` flags for at least two recovered clock cycles to indicate rate match FIFO full and empty conditions, respectively. You must then assert the `rx_digitalreset` signal to reset the receiver PCS blocks.

PCIe Mode

Intel Corporation has developed a PHY interface for the PCIe architecture specification to enable implementation of a PCIe-compliant physical layer device. This specification also defines a standard interface between the physical layer device and the media access control layer (MAC). Version 2.00 of the specification provides implementation details for a PCIe-compliant physical layer device at both Gen1 (2.5 Gbps) and Gen2 (5 Gbps) signaling rates.

Arria II GX and GZ transceivers support ×1, ×4, and ×8 lane configurations in PCIe functional mode at Gen1 (2.5 Gbps) data rates. Arria II GZ devices also support ×1 and ×4 lane configurations in PCIe functional mode at Gen2 (5.0 Gbps). In PCIe ×1 configuration, the PCS and PMA blocks of each channel are clocked and reset independently. PCIe ×4 and ×8 configurations support channel bonding for four-lane and eight-lane PCIe links, where the PCS and PMA blocks of all bonded channels share common clock and reset signals.

You can configure Arria II GX and GZ transceivers to implement a Version 2.00 PCIe-compliant PHY using one of the following methods:

- PCIe Compiler—This method allows you to use the Arria II GX and GZ devices built-in PCIe hard IP blocks to implement the PHY-MAC layer, Data Link layer, and Transaction layer of the PCIe protocol stack. In this mode, each Arria II GX and GZ transceiver channel uses a PIPE interface block that transfers data, control, and status signals between the PHY-MAC layer and the transceiver channel PCS and PMA blocks. The PIPE interface block is used only in this mode and cannot be bypassed.
- ALTGX MegaWizard Plug-In Manager—This method requires implementing the PHY-MAC layer, Data Link layer, and Transaction layer in the FPGA fabric using a soft IP. Use this method if you do not use the PCIe hard IP block. (You cannot access the PCIe hard IP block if you use this method.)

For descriptions of PCIe hard IP architecture and PCIe mode configurations allowed when using the PCIe hard IP block, refer to the *PCI Express Compiler User Guide*.

For more information about transceiver datapath clocking in different PCIe configurations, refer to the *Transceiver Clocking in Arria II Devices* chapter.

The transmitter datapath in PCIe mode consists of the:

- PIPE interface
- TX phase compensation FIFO
- Optional byte serializer (enabled for 16-bit and disabled for 8-bit FPGA fabric-transceiver interface)
- 8B/10B encoder
- 10:1 serializer
- Transmitter buffer with receiver detect circuitry

The receiver datapath in PCIe mode consists of the:

- Receiver input buffer with signal detect circuitry
- 1:10 deserializer
- Word aligner that implements PCIe-compliant synchronization state machine
- Optional rate match FIFO (clock rate compensation) that can tolerate up to 600 PPM frequency difference
- 8B/10B decoder
- Optional byte deserializer (enabled for 16-bit and disabled for 8-bit FPGA fabric-transceiver interface)
- RX phase compensation FIFO
- PIPE interface

Table 1–16 lists features supported in PCIe functional mode for Gen1 (2.5 Gbps) and Gen2 (5.0 Gbps) data rate configurations.

Table 1–16. Supported Features in PCIe Mode for Arria II Devices (Part 1 of 2)

Feature	2.5 Gbps (Gen1)	5.0 Gbps (Gen2) *(1)*
×1, ×4, ×8 link configurations	✓	Only ×1 and ×4 are supported
PCIe-compliant synchronization state machine	✓	✓
±300 PPM (total 600 PPM) clock rate compensation	✓	✓
8-bit FPGA fabric-transceiver interface	✓	—
16-bit FPGA fabric-transceiver interface	✓	✓
Transmitter buffer electrical idle	✓	✓
Receiver detection	✓	✓
8B/10B encoder disparity control when transmitting compliance pattern	✓	✓
Power state management	✓	✓
Receiver status encoding	✓	✓

Table 1–16. Supported Features in PCIe Mode for Arria II Devices (Part 2 of 2)

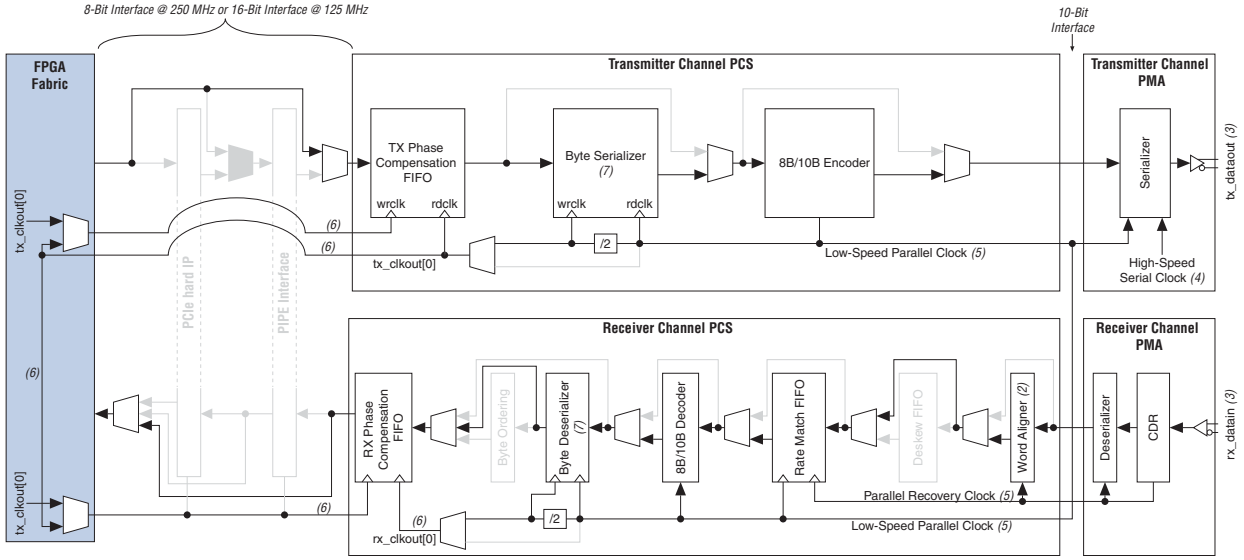
Feature	2.5 Gbps (Gen1)	5.0 Gbps (Gen2) *(1)*
Dynamic switch between 2.5 Gbps and 5 Gbps signaling rate	—	✓
Dynamically selectable transmitter margining for differential output voltage control	—	✓
Dynamically selectable transmitter buffer de-emphasis of -3.5 db and -6 db	—	✓

Note to Table 1–16:

(1) For Arria II GZ devices only.

Figure 1–62 shows the Arria II GX and GZ transceiver datapath when configured in PCIe functional mode.

Figure 1–62. Arria II GX and GZ Transceiver Datapath in PCIe Mode *(Note 1)*



Notes to Figure 1–62:

(1) The transceiver datapath clock varies between non-bonded (×1) and bonded (×4 and ×8) configurations in PCIe mode, described in the *Transceiver Clocking for Arria II Devices* chapter.

(2) The word aligner uses automatic synchronization state machine mode (10-Bit /K28.5/).

(3) This can be ×1, ×4, or ×8 at 2.5 Gbps or ×1 or ×4 at 5.0 Gbps (for Arria II GZ devices only).

(4) The high-speed serial clock is running at 1.25 GHz.

(5) The parallel clocks are running at 250 MHz.

(6) This clock is running at 125 MHz if the byte serializer and deserializer are used. Otherwise, this clock is running at 250 MHz.

(7) If you use the PCIe hard IP, you can enable the byte serializer and deserializer with an 8-bit FPGA fabric-to-transceiver interface running at 250 MHz or disabled with a 16-bit FPGA fabric-to-transceiver interface running at 125 MHz. Otherwise, these blocks are always disabled and your 16-bit FPGA fabric-to-transceiver interface is running at 125 MHz.

Besides transferring data, control, and status signals between the PHY-MAC layer and the transceiver, the PIPE interface block implements the following functions required in a PIPE-compliant physical layer device:

- Manages the PIPE power states
- Forces the transmitter buffer in the electrical idle state
- Initiates the receiver detect sequence
- Controls 8B/10B encoder disparity when transmitting compliance pattern
- Indicates completion of various PHY functions, such as receiver detection and power state transitions on the `pipephydonestatus` signal
- Encodes receiver status and error conditions on the `pipestatus[2:0]` signal as specified in the PIPE specification

The following subsections describe each Arria II GX and GZ transceiver function.

Power State Management

The PCIe specification defines four power states that the physical layer device must support to minimize power consumption:

- P0 is the normal operation state during which packet data is transferred on the PCIe link.
- P0s, P1, and P2 are low-power states into which the physical layer must transition as directed by the PHY-MAC layer to minimize power consumption.

The PCIe specification provides the mapping of these power states to the long-term sample storage module (LTSSM) states specified in the PCIe Base Specification 2.0. The PHY-MAC layer is responsible for implementing the mapping logic between the LTSSM states and the four power states in the PCIe-compliant PHY.

The PIPE interface in Arria II GX and GZ transceivers provide an input port (`powerdn[1:0]`) to set the transceivers in one of the four power states, as shown in Table 1–17.

Table 1–17. powerdn [1:0] Port Power State Functions and Descriptions for Arria II Devices

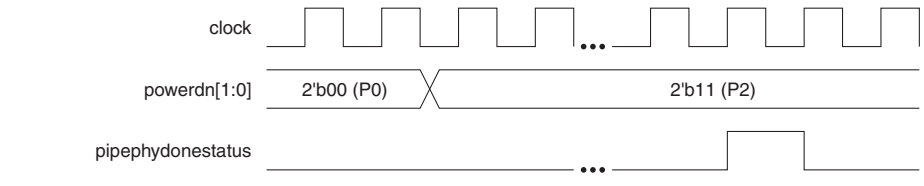
powerdn [1:0] Values	Power State	Description	Function
2'b00	P0	Normal operation mode	Transmits normal data, transmits electrical idle, or enters into loopback mode
2'b01	P0s	Low recovery time saving state	Only transmits electrical idle
2'b10	P1	High recovery time power saving state	Transmitter buffer is powered down and can perform a receiver detect while in this state
2'b11	P2	Lowest power saving state	Transmits electrical idle or a beacon to wake up the downstream receiver

When the device transitions from the P0 power state to lower power states (P0s, P1, and P2), the PCIe specification requires the physical layer device to implement power saving measures. Arria II GX and GZ transceivers do not implement these power saving measures except when putting the transmitter buffer in electrical idle in the lower power states.

The PIPE interface block indicates a successful power state transition by asserting the pipephydonestatus signal for one parallel clock cycle as specified in the PCIe specification. The PHY-MAC layer must not request any further power state transition until the pipephydonestatus signal has indicated the completion of the current power state transition request.

Figure 1–63 shows an example waveform for a transition from the P0 to the P2 power state.

Figure 1–63. Example of Power State Transition from P0 to P2



The PCIe specification allows the PIPE interface to perform protocol functions such as receiver detect, loopback, and beacon transmission in specified power states only. This requires the PHY-MAC layer to drive the tx_detectrxloop and tx_forceelecidle signals appropriately in each power state to perform these functions. Table 1–18 lists the logic levels that the PHY-MAC layer must drive on the tx_detectrxloop and tx_forceelecidle signals in each power state.

Table 1–18. Logic Levels for the PHY-MAC Layer for Arria II Devices

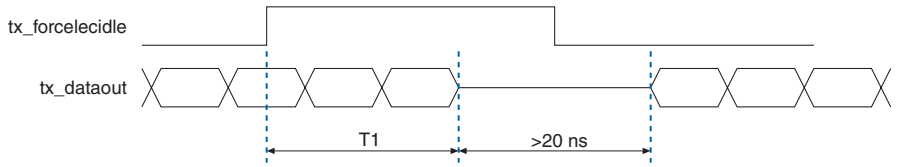
Power State	tx_detectrxloop Value	tx_forceelecidle Value
P0	0: normal mode 1: loopback mode	0: must be de-asserted 1: illegal mode
P0s	Don't care	0: illegal mode 1: must be asserted in this state
P1	0: electrical state 1: receiver detect	0: illegal mode 1: must be asserted in this state
P2	Don't care	De-asserted in this state for sending beacon. Otherwise asserted.

Transmitter Buffer Electrical Idle

The PCIe specification requires the transmitter buffer to be in electrical idle in the P1 power state, as shown in Table 1–18. During electrical idle, the transmitter buffer differential and common mode output voltage levels are compliant to the PCIe Base Specification 2.0 for both PCIe Gen1 and Gen2 data rates.

In Arria II GX and GZ transceivers, asserting the input signal tx_forceelecidle puts the transmitter buffer in that channel in the electrical idle state. Figure 1–64 shows the relationship between asserting the tx_forceelecidle signal and the transmitter buffer output on the tx_dataout port. Time T1 taken from the assertion of the tx_forceelecidle signal to the transmitter buffer reaching electrical idle voltage levels is a minimum of 8 ns. When in the electrical idle state, the PCIe protocol requires the transmitter buffer to stay in electrical idle for a minimum of 20 ns for both Gen1 and Gen2 data rates.

Figure 1–64. Transmitter Buffer Electrical Idle State



Receiver Detection

During the detect substate of the LTSSM state machine, the PCIe protocol requires the transmitter channel to perform a receiver detect sequence to detect if a receiver is present at the far end of each lane. The PCIe specification requires that a receiver detect operation be performed during the P1 power state where the transmitter output buffer is in electrical idle (tri-stated).

This feature requires the transmitter output buffer to be tri-stated (in electrical idle mode), have OCT utilization, and run at 125 MHz on the `fixedclk` signal. You can enable this feature in PCIe functional mode by setting the `tx_forceelecidle` and `tx_detectrxloop` ports to 1'b1.

When the `tx_detectrxloop` signal is asserted high in the P1 power state, the PIPE interface block sends a command signal to the transmitter buffer in that channel to initiate a receiver detect sequence. On receiving this command signal, the receiver detect circuitry creates a step voltage at the transmitter output buffer common mode voltage. If an active receiver (that complies with the PCIe input impedance requirements) is present at the far end, the time constant of the step voltage on the trace is higher compared to when the receiver is not present. The receiver-detect circuitry monitors the time constant of the step signal seen on the trace to decide if a receiver was detected or not.

If a far-end receiver is successfully detected, the PIPE interface block asserts the `pipephydonestatus` signal for one clock cycle and synchronously drives the `pipestatus[2:0]` signal to 3'b011. If a far-end receiver is not detected, the PIPE interface block asserts the `pipephydonestatus` signal for one clock cycle and synchronously drives the `pipestatus[2:0]` signal to 3'b000.

There is some latency after asserting the `tx_detectrxloop` signal before receiver detection is indicated on the `pipephydonestatus` port. In addition, the `tx_forceelecidle` port must be asserted at least 10 parallel clock cycles prior to the `tx_detectrxloop` port to ensure that the transmitter buffer is tri-stated.

For the receiver detect circuitry to function reliably, the AC-coupling capacitor on the serial link and the receiver termination values used in your system must be compliant to the PCIe Base Specification 2.0. Receiver detect circuitry communicates the status of the receiver detect operation to the PIPE interface block.

Figure 1–65 and Figure 1–66 show the receiver detect operation where a receiver was successfully detected and where a receiver was not detected, respectively.

Figure 1–65. Receiver Detect Successful Operation

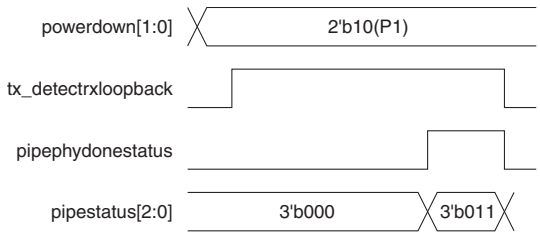
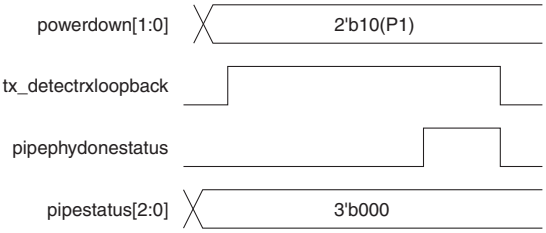


Figure 1–66. Receiver Detect Unsuccessful Operation



Compliance Pattern Transmission Support

The LTSSM state machine can enter the polling.compliance substate where the transmitter must transmit a compliance pattern as specified in the PCIe Base Specification 2.0. The polling.compliance substate is intended to assess if the transmitter is electrically compliant with the PCIe voltage and timing specifications.

The compliance pattern is a repeating sequence of the following four code groups:

- /K28.5/
- /D21.5/
- /K28.5/
- /D10.2/

The PCIe protocol requires the first /K28.5/ code group of the compliance pattern to be encoded with negative current disparity. To satisfy this requirement, the PIPE interface block provides an input signal (`tx_forcedispcompliance`). A high level on the `tx_forcedispcompliance` signal forces the associated parallel transmitter data on the `tx_datain` port to transmit with a negative current running disparity.

For 8-bit transceiver channel width configurations, you must drive the `tx_forcedispcompliance` signal high in the same parallel clock cycle as the first /K28.5/ of the compliance pattern on the `tx_datain` port. For 16-bit transceiver channel width configurations, you must drive only the LSB of the `tx_forcedispcompliance[1:0]` signal high in the same parallel clock cycle as /K28.5/D21.5/ of the compliance pattern on the `tx_datain` port.

Figure 1–67 and Figure 1–68 show the required level on the `tx_forcedispcompliance` signal while transmitting the compliance pattern in 8-bit and 16-bit channel width configurations, respectively.

Figure 1–67. Compliance Pattern Transmission Support, 8-Bit Wide Channel Configuration

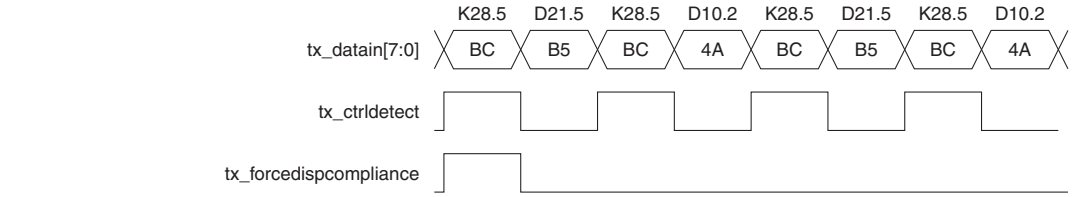
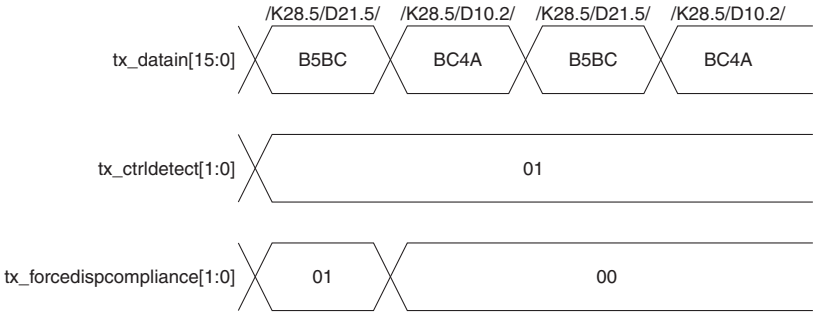


Figure 1–68. Compliance Pattern Transmission Support, 16-Bit Wide Channel Configuration



Receiver Status

The PCIe specification requires the PHY to encode the receiver status on a 3-bit `RxStatus[2:0]` signal. The PIPE interface block receives status signals from the transceiver channel PCS and PMA blocks and drives the status on the 3-bit output signal (`pipestatus[2:0]`) to the FPGA fabric. The encoding of the status signals on the `pipestatus[2:0]` port is compliant with the PCIe specification and is listed in Table 1–19.

Table 1–19. Encoding of the Status Signals on the pipestatus[2:0] Port for Arria II Devices

pipestatus[2:0]	Description	Error Condition Priority *(1)*
3'b000	Received data OK	N/A
3'b001	One SKP symbol added	5
3'b010	One SKP symbol deleted	6
3'b011	Receiver detected	N/A
3'b100	8B/10B decode error	1
3'b101	Elastic buffer (rate match FIFO) overflow	2
3'b110	Elastic buffer (rate match FIFO) underflow	3
3'b111	Received disparity error	4

Note to Table 1–19:

(1) The PIPE interface follows the priority listed in Table 1–19 while encoding the receiver status on the `pipestatus[2:0]` port. Two or more error conditions; for example, 8B/10B decode error (code group violation), rate match FIFO overflow or underflow, or receiver disparity error, can occur simultaneously. When this happens, PIPE drives 3'b100 on the `pipestatus[2:0]` signal.

Fast Recovery Mode

The PCIe Base Specification fast training sequences (FTS) are used for bit and byte synchronization to transition from L0s to L0 (PIPE P0s to P0) power states. The PCIe specification requires the physical layer device to acquire bit and byte synchronization after you transition from L0s to L0 state within 16 ns to 4 µs.

If the Arria II GX and GZ receiver CDR is configured in Automatic Lock mode, the receiver cannot meet the PCIe specification of acquiring bit and byte synchronization within 4 µs due to the signal detect and PPM detector time. To meet this specification, each Arria II GX and GZ transceiver has built-in fast recovery circuitry that you can optionally enable in the ALTGX MegaWizard Plug-In Manager with the **Enable fast recovery mode** option.

Fast recovery circuitry controls the receiver CDR `rx_locktorefclk` and `rx_locktodata` signals to force the receiver CDR in LTR or LTD modes, by relying on the Electrical Idle Ordered Sets (EIOS), NFTS sequences received in L0 power state, and the signal detect signal from the receiver input buffer to control the receiver CDR lock mode. It is self-operational and does not require user inputs.

When you enable fast recovery mode, the `rx_locktorefclk` and `rx_locktodata` ports are not available in the ALTGX MegaWizard Plug-In Manager.

Electrical Idle Inference

The PCIe protocol allows inferring the electrical idle condition at the receiver instead of detecting the electrical idle condition using analog circuitry. PCIe Base Specification 2.0, section .2.4.3, specifies conditions to infer electrical idle at the receiver in various sub-states of the LTSSM state machine.

In all PCIe modes (×1, ×4, and ×8), each receiver channel PCS has an optional electrical idle inference module designed to implement the electrical idle inference conditions specified in the PCIe Base Specification 2.0.

You can enable the electrical idle inference module by selecting the **Enable electrical idle inference functionality** option in the ALTGX MegaWizard Plug-In Manager. This feature infers electrical idle depending on the logic level driven on the `rx_elecidleinfersel[2:0]` input signal. The electrical idle inference module drives the `pipeelecidle` signal high in each receiver channel when an electrical idle condition is inferred. For the electrical idle inference module to correctly infer an electrical idle condition in each LTSSM substate, you must drive the `rx_elecidleinfersel[2:0]` signal appropriately, as shown in Table 1–20.

Table 1–20. Electrical Idle Inference Conditions for Arria II Devices (Part 1 of 2)

rx_elecidleinfersel[2:0]	LTSSM State	Description
3'b100	L0	Absence of update FC or alternatively skip ordered set in 128 µs window
3'b101	Recovery.RcvrCfg	Absence of TS1 or TS2 ordered set in 1280 UI interval
3'b101	Recovery.Speed when successful speed negotiation = 1'b1	Absence of TS1 or TS2 ordered set in 1280 UI interval

Table 1–20. Electrical Idle Inference Conditions for Arria II Devices (Part 2 of 2)

rx_elecidleinfersel[2:0]	LTSSM State	Description
3'b110	Recovery.Speed when successful speed negotiation = 1'b0	Absence of an exit from electrical idle in 2000 UI interval
3'b111	Loopback.Active (as slave)	Absence of an exit from electrical idle in 128 µs window

The electrical idle inference module cannot detect an electrical idle exit condition based on the reception of the electrical idle exit ordered set (EIEOS), as specified in the PCIe Base Specification.

If you select the **Enable Electrical Idle Inference Functionality** option in the ALTGX MegaWizard Plug-In Manager and drive `rx_elecidleinfersel[2:0]` = 3'b0xx, the electrical idle inference block uses EIOS detection from the fast recovery circuitry to drive the `pipeelecidle` signal. Otherwise, the electrical idle inference module is disabled. In this case, the `rx_signaldetect` signal from the signal detect circuitry in the receiver input buffer is inverted and driven as the `pipeelecidle` signal.

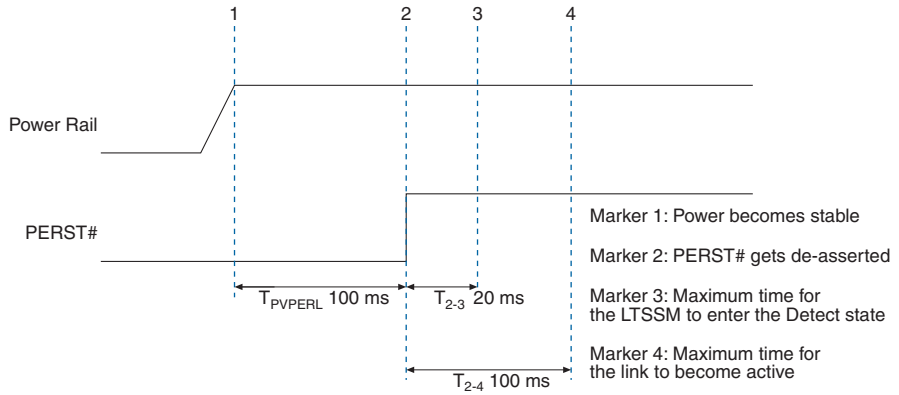
PCIe Cold Reset Requirements

The PCIe Base Specification 2.0 defines the following three types of conventional resets to the PCIe system components:

- Cold reset—fundamental reset after power up
- Warm reset—fundamental reset without removal and re-application of power
- Hot reset—in-band conventional reset initiated by higher layer by setting the hot reset bit in the TS1 or TS2 training sequences

Figure 1–69 shows the PCIe cold reset timing requirements.

Figure 1–69. PCIe Cold Reset Requirements



The following is the time taken by a PCIe port, implemented in an Arria II GX or GZ device, to go from the power-up to the link-active state:

- Power-on reset—begins after power rails become stable, which typically takes 12 ms
- FPGA configuration and programming—begins after power on reset. Configuration time depends on the FPGA density
- Time taken from de-assertion of PERST# to link active—typically takes 40 ms

To meet the PCIe specification of 200 ms from the power-on to the link-active state, the Arria II GX and GZ device configuration time must be less than 148 ms (200 ms to 12 ms for power on reset, 40 ms for the link to become active after PERST# de-assertion).

For the typical Arria II GX and GZ configuration times using the Fast Passive Parallel (FPP) configuration scheme at 125 MHz, refer to the *Device Datasheet for Arria II Devices*.

For more information about the FPP configuration scheme, refer to the *Configuration, Design Security, Remote System Upgrades in Arria II Devices* chapter.

Most flash memories available in the market can run up to 100 MHz. Altera recommends using a MAX II device to convert the 16-bit flash memory output at 62.5 MHz to 8-bit configuration data input to the Arria II GX and GZ devices at 125 MHz.

SDI

The Society of Motion Picture and Television Engineers (SMPTE) defines various SDI standards for the transmission of uncompressed video.

The following three SMPTE standards are popular in video broadcasting applications:

- SMPTE 259M standard, more popularly known as the standard-definition (SD) SDI—is defined to carry video data at 270 Mbps.
- SMPTE 292M standard, more popularly known as the high-definition (HD) SDI—is defined to carry video data at 1485 Mbps or 1483.5 Mbps.
- SMPTE 424M standard, more popularly known as the third-generation (3G) SDI—is defined to carry video data at 2970 Mbps or 2967 Mbps.

Table 1–21 lists the data rates, refclk frequencies, and interface widths supported by Arria II GX and GZ transceivers in SDI mode.

Table 1–21. SDI Mode Data Rates, refclk Frequencies, and Interface Widths in Arria II Devices (Part 1 of 2)

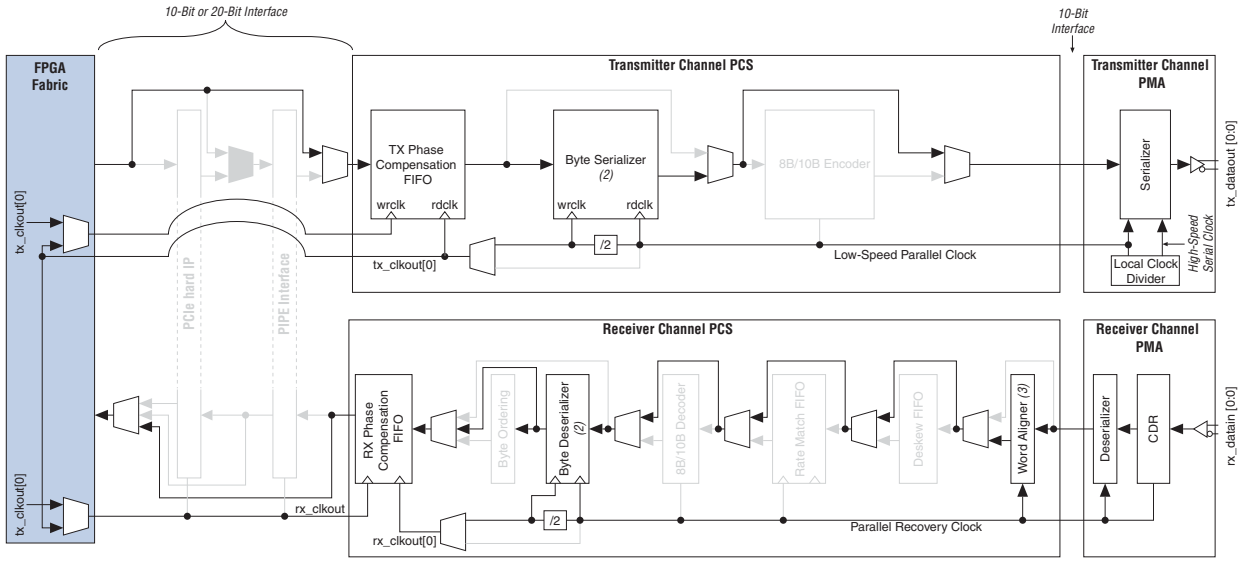
Configuration	Data Rate (Mbps)	Support refclk Frequencies (MHz)	FPGA Fabric-to-Transceiver Width	Byte Serializer/Deserializer Usage
HD	1483.5	74.175	20-bit	Used
		148.35	10-bit	Not used
	1485	74.25	20-bit	Used
		148.5	10-bit	Not used

Table 1–21. SDI Mode Data Rates, refclk Frequencies, and Interface Widths in Arria II Devices (Part 2 of 2)

Configuration	Data Rate (Mbps)	Support refclk Frequencies (MHz)	FPGA Fabric-to-Transceiver Width	Byte Serializer/Deserializer Usage
3G	2967	148.35	20-bit	Used
		296.7		
	2970	148.5		
		297		

Figure 1–70 shows the transceiver datapath when configured in SDI mode.

Figure 1–70. SDI Mode Datapath ***(Note 1)***



Notes to Figure 1–70:

(1) For the frequency, data rate, and interface width supported, refer to Table 1–21 on page 1–72.

(2) This allows the fabric-to-transceiver interface to run below the maximum interface frequency. For more information, refer to Table 1–21 on page 1–72.

(3) The word aligner uses bit-slip mode. However, this block is not useful because word alignment and framing occurs after de-scrambling. Altera recommends driving the ALTGXB megafunction `rx_bitslip` signal low to prevent the word aligner from inserting bits in the received data stream.

In HD-SDI mode, the transmitter is purely a parallel-to-serial converter. SDI transmitter functions, such as scrambling and cyclical redundancy check (CRC) code generation, must be implemented in the FPGA logic array. Similarly, SDI receiver functions, such as de-scrambling, framing, and CRC checker, must be implemented in the FPGA logic array.

SRIO

The RapidIO Trade Association defines a high-performance, packet-switched interconnect standard to pass data and control information between microprocessors, digital signal, communications and network processors, system memories, and peripheral devices.

The SRIO physical layer specification defines three line rates—1.25 Gbps, 2.5 Gbps, and 3.125 Gbps. It also defines two link widths—single-lane (×1) and bonded four-lane (×4) at each line rate. Arria II GX and GZ transceivers support only single-lane (×1) configuration at all three line rates. You can instantiate four ×1 channels configured in SRIO mode to achieve one non-bonded ×4 SRIO link. The four receiver channels in this ×4 SRIO link do not have lane alignment or deskew capability.

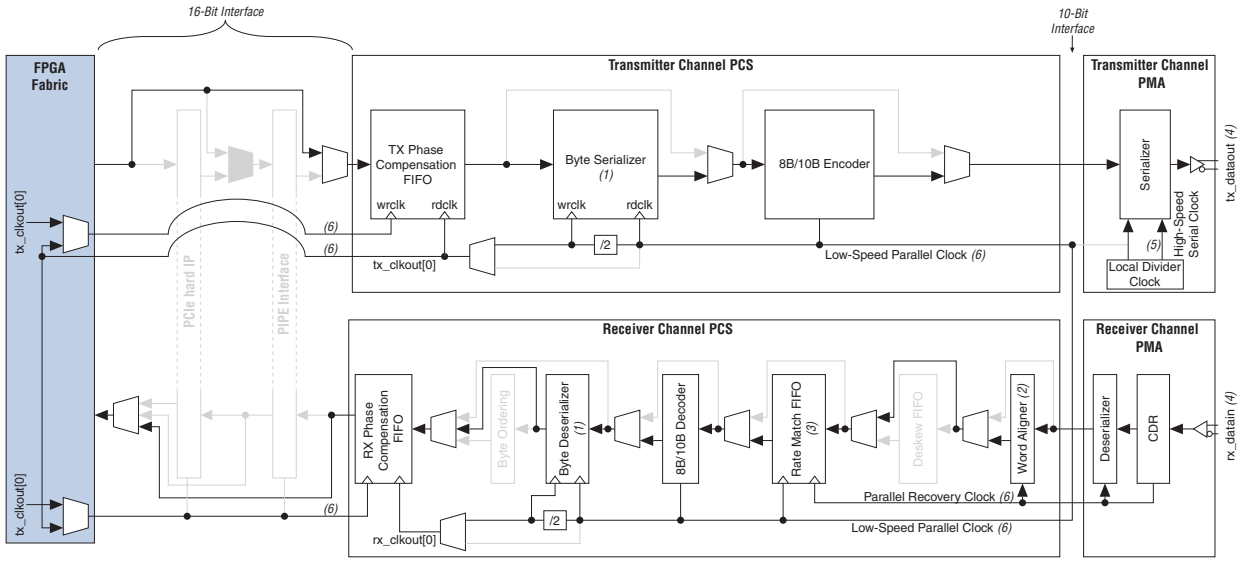
Arria II GX and GZ transceivers, when configured in SRIO functional mode, provide the following PCS and PMA functions:

- 8B/10B encoding and decoding
- Word alignment
- Lane synchronization state machine
- Clock recovery from the encoded data
- Serialization and deserialization

Arria II GX and GZ transceivers do not have built-in support for some PCS functions, such as pseudo-random 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.

Figure 1–71 shows the ALTGX transceiver datapath when configured in SRIO mode.

Figure 1–71. SRIO Mode Datapath



Notes to Figure 1–71:

(1) This allows the fabric-to-transceiver interface to run below the maximum interface frequency and is always enabled for SRIO functional mode.
(2) The word aligner uses the automatic synchronization state machine (10 bit /K28.5/) and is compliant with the SRIO protocol.
(3) This module is optional.
(4) This can run at 1.25, 2.5, or 3.125 Gbps.
(5) This is running at half the rate of the data rate.
(6) This is running at 62.5 MHz for 1.25 Gbps data rate, 125 MHz for 2.5 Gbps data rate, or 156.25 MHz for 3.125 Gbps data rate.

In SRIO mode, the ALTGX MegaWizard Plug-In Manager automatically defaults the synchronization state machine to indicate synchronization (a high logic level on the `rx_syncstatus` port) when the receiver acquires 127 K28.5 (10'b0101111100 or 10'b1010000011) synchronization code groups without receiving an intermediate invalid code group. When synchronized, the state machine indicates loss of synchronization (a low logic level on the `rx_syncstatus` port) when it detects three invalid code groups separated by less than 255 valid code groups, or when it is reset.

SRIO only allows one insertion or deletion of the skip character /R/from the /K/, /R/, /R/, /R/ clock compensation sequence. However, the Arria II GX and GZ rate match FIFO may perform multiple insertions or deletions if the PPM difference is more than the ±200 PPM range. Ensure that the PPM difference in your system is less than ±200 ppm.

SONET/ SDH

SONET/SDH is one of the most common serial-interconnect protocols used in backplanes deployed in communications and telecom applications. SONET/SDH defines various optical carrier (OC) subprotocols for carrying signals of different capacities through a synchronous optical hierarchy.

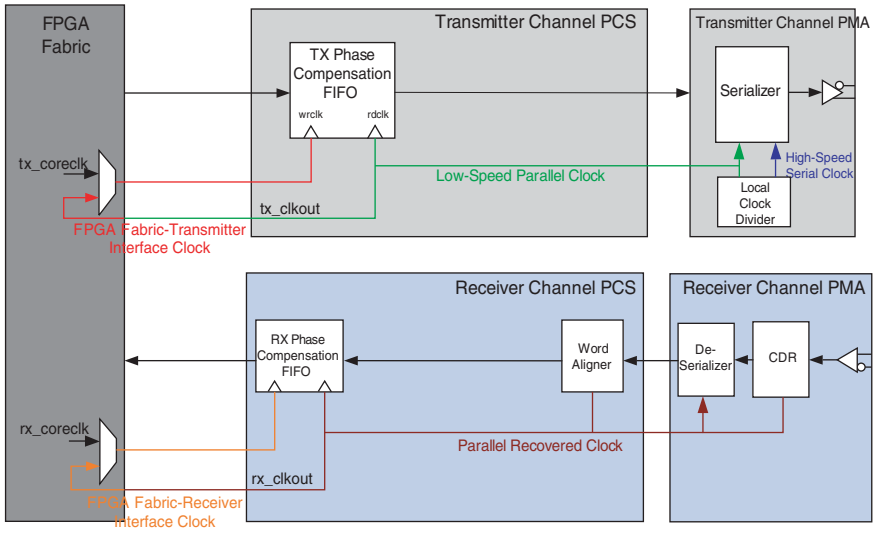
You can use Arria II GX and GZ transceivers as physical layer devices in a SONET/SDH system. These transceivers provide support for SONET/SDH protocol-specific functions and electrical features; for example, alignment to an A1A2 or A1A1A2A2 pattern.

In SONET/SDH systems, A1 is defined as 8'hF6 and A2 is defined as 8'h28. Transport overhead bytes A1 and A2 are used for restoring frame boundary from the serial data stream. Frame sizes are fixed, so the A1 and A2 bytes appear within the serial data stream every 125 µs. In an OC-12 system, 12 A1 bytes are followed by 12 A2 bytes. Similarly, in an OC-48 system, 48 A1 bytes are followed by 48 A2 bytes.
OC-96 systems are for Arria II GZ devices only and have 96 A 1 bytes followed by 96 A2 bytes.

Arria II GX transceivers are designed to support the protocols OC-12 at 622 Mbps with 8-bit channel width and OC-48 at 2488.32 Mbps with 16-bit channel width. Arria II GZ transceivers are designed to support the protocol OC-96 at 4,967 Mbps.

Figure 1–72 shows the transceiver datapath when configured in SONET/SDH OC-12 mode.

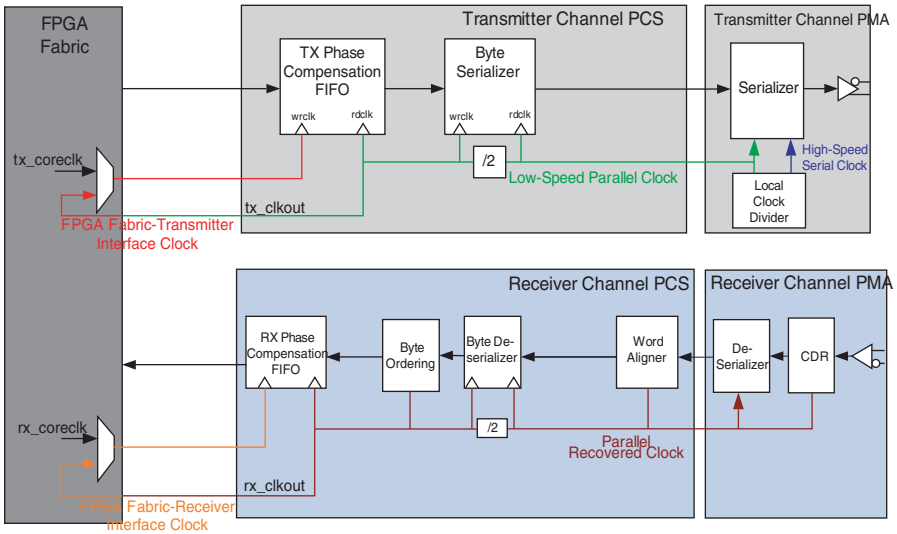
Figure 1–72. SONET/SDH OC-12 Datapath



SONET/SDH OC-48 Datapath

Figure 1–73 shows the transceiver datapath when configured in SONET/SDH OC-48 mode.

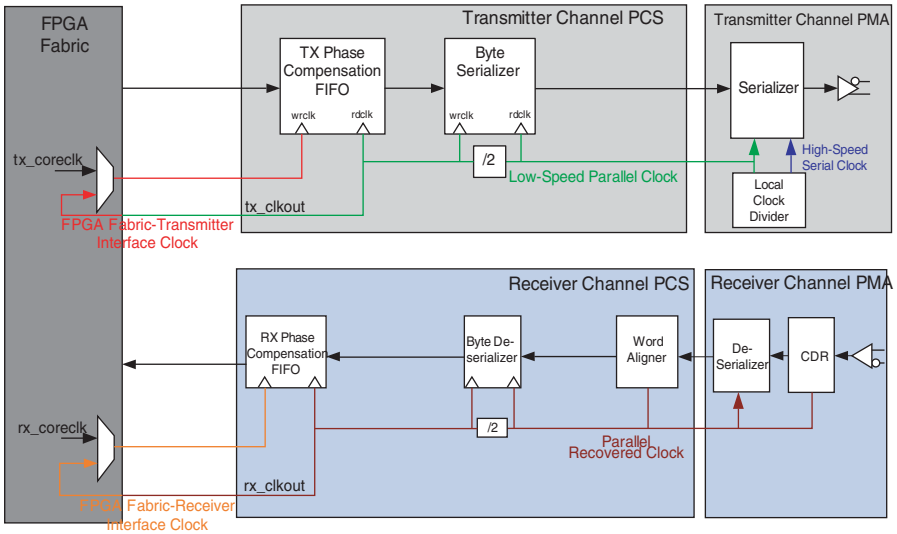
Figure 1–73. SONET/SDH OC-48 Datapath



SONET/SDH OC-96 Datapath

Figure 1–74 shows the transceiver datapath when configured in SONET/SDH OC-96 mode.

Figure 1–74. SONET/SDH OC-96 Datapath



Unlike Ethernet, where the LSB of the parallel data byte is transferred first, SONET/SDH requires the MSB to be transferred first. To facilitate the MSB-to-LSB transfer, you must enable the **Flip Transmitter input data bits** and **Flip Receiver output data bits** options in the ALTGX MegaWizard Plug-In Manager.

Depending on whether data bytes are transferred MSB-to-LSB or LSB-to-MSB, you must select the appropriate word aligner settings in the ALTGX MegaWizard Plug-In Manager.

Word Aligner in SONET/SDH Mode

The word aligner in SONET/SDH functional mode is configured in manual alignment mode. You can configure the word aligner to either align to a 16-bit A1A2 pattern or a 32-bit A1A1A2A2 pattern, controlled by the `rx_a1a2size` input port to the transceiver.

A low level on the `rx_a1a2size` port configures the word aligner to align to a 16-bit A1A2 pattern; a high level configures the word aligner to align to a 32-bit A1A1A2A2 pattern.

In OC-96 configurations, the word aligner is only allowed to align to an A1A1A2A2 pattern, so the input port `rx_a1a2size` is unavailable. Barring this difference, the OC-96 word alignment operation is similar to that of the OC-12 and OC-48 configurations.

You can also configure the word aligner to flip the word alignment pattern bits programmed in the ALTGX MegaWizard Plug-In Manager and compare them with the incoming data for alignment. This feature offers flexibility to the SONET backplane system for either a MSB-to-LSB or LSB-to-MSB data transfer. Table 1–22 lists word alignment patterns that you must program in the ALTGX MegaWizard Plug-In Manager based on the bit-transmission order and the word aligner bit-flip option.

Table 1–22. Word Aligner Settings for Arria II Devices

Serial Bit Transmission Order	Flip the Word Alignment Pattern Bits	Word Alignment Pattern
MSB-to-LSB	On	16'hF628
MSB-to-LSB	Off	16'h146F
LSB-to-MSB	Off	16'h28F6

OC-48 and OC-96 Byte Serializer and Deserializer

The OC-48 and OC-96 transceiver datapath includes the byte serializer and deserializer to allow the PLD interface to run at a lower speed. The OC-12 configuration does not use the byte serializer and deserializer blocks. The byte serializer and deserializer blocks are explained in "Byte Serializer" on page 1–14 and "Byte Deserializer" on page 1–43, respectively.

The OC-48 byte serializer converts 16-bit data words from the FPGA fabric and translates the 16-bit data words into two 8-bit data bytes at twice the rate. The OC-48 byte deserializer takes in two consecutive 8-bit data bytes and translates them into a 16-bit data word to the FPGA fabric at half the rate.

The OC-96 byte serializer converts 32-bit data words from the FPGA fabric and translates them into two 16-bit data words at twice the rate. The OC-96 byte deserializer takes in two consecutive 16-bit data words and translates them into a 32-bit data word to the FPGA fabric at half the rate.

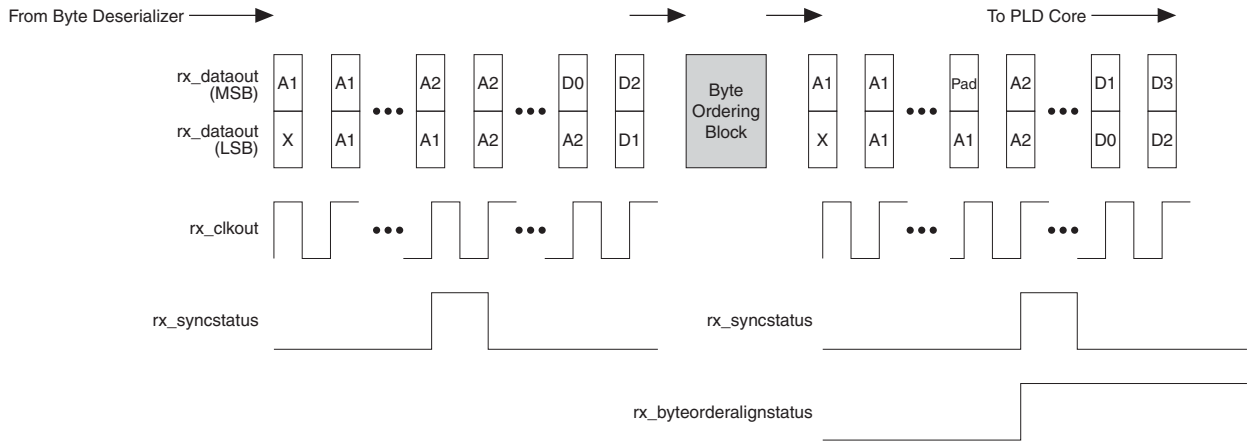
Byte Ordering in SONET/SDH OC-48 Mode

Because of byte deserialization, the MSByte of a word might appear at the `rx_dataout` port along with the LSByte of the next word. In a SONET/SDH OC-48 configuration, you can use the byte ordering block that is built into the datapath to perform byte ordering. Byte ordering in a SONET/SDH OC-48 configuration is in word alignment-based mode, where the byte ordering block is triggered by the rising edge of the `rx_syncstatus` signal.

At the rising edge of the `rx_syncstatus` signal, the byte ordering block compares the LSByte coming out of the byte deserializer with the A2 byte of the A1A2 alignment pattern. If the LSByte coming out of the byte deserializer does not match the A2 byte set in the ALTGX MegaWizard Plug-In Manager, the byte ordering block inserts a PAD character, as shown in Figure 1–75. Inserting this PAD character enables the byte ordering block to restore the correct byte order.

The PAD character is defaulted to the A1 byte of the A1A2 alignment pattern.

Figure 1–75. SONET/SDH OC-48 Byte Ordering Example



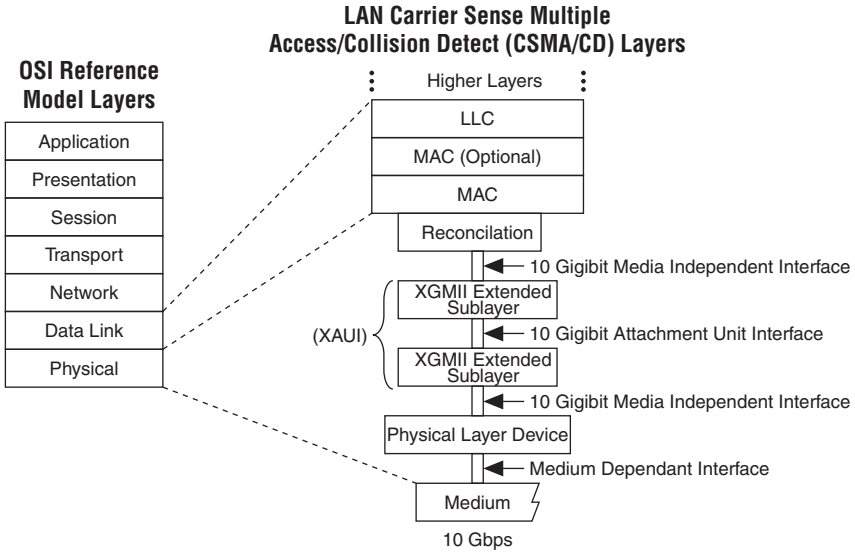
XAUI

Use this functional mode for XAUI, HiGig, or HiGig+ protocols.

The XAUI is an optional, self-managed interface that you can insert between the reconciliation sublayer and the PHY layer to transparently extend the physical reach of the XGMII.

Figure 1–76 shows the relationships between the XGMII and XAUI layers.

Figure 1–76. XAUI and XGMII Layers



The XGMII interface consists of four 8-bit lanes. At the transmit side of the XAUI interface, the data and control characters are converted within the XGMII extender sublayer (XGXS) into an 8B/10B encoded data stream. Each data stream is then transmitted across a single differential pair running at 3.125 Gbps (3.75 Gbps for HiGig/HiGig+). At the XAUI receiver, the incoming data is decoded and mapped back to the 32-bit XGMII format. This provides a transparent extension of the physical reach of the XGMII and also reduces the interface pin count.

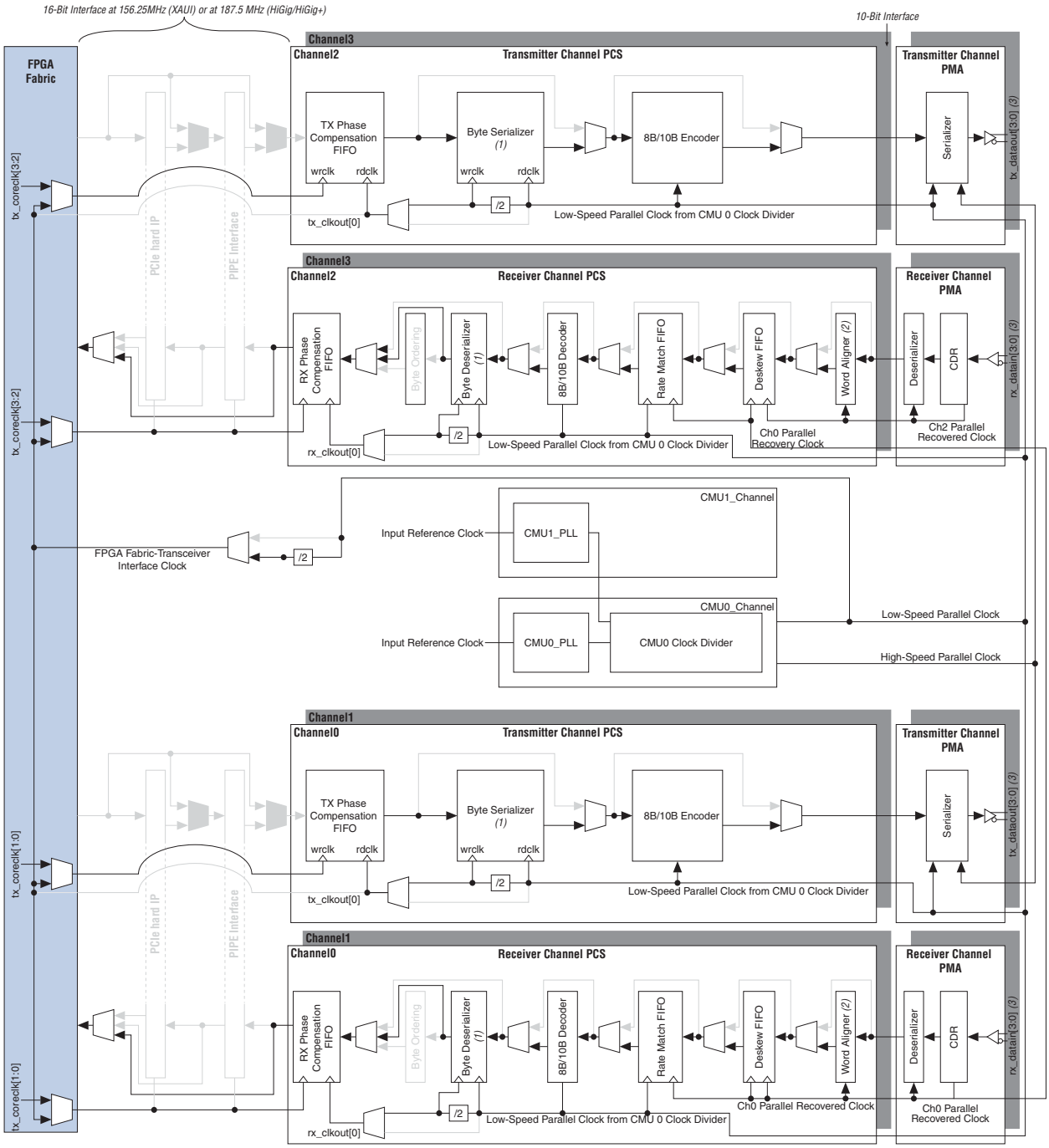
XAUI functions as a self-managed interface because code group synchronization, channel deskew, and clock domain decoupling are handled with no upper layer support requirements. This functionality is based on the PCS code groups that are used during the IPG time and idle periods.

Arria II GX and GZ transceivers configured in XAUI mode provide the following protocol features:

- XGMII-to-PCS code conversion at the transmitter—The 8B/10B encoder in the Arria II GX and GZ transmitter datapath is controlled by a transmitter state machine that maps various 8-bit XGMII codes to 10-bit PCS code groups. This state machine complies with the IEEE P802.3ae PCS transmit source state diagram.
- PCS-to-XGMII code conversion at the receiver—The 8B/10B decoder in the Arria II GX and GZ receiver datapath is controlled by a XAUI receiver state machine that converts received PCS code groups into specific 8-bit XGMII codes.
- 8B/10B encoding and decoding
- IEEE P802.3ae-compliant synchronization state machine
- ±100 PPM clock rate compensation
- Channel deskew of four lanes of the XAUI link

Figure 1–77 shows the ALTGX megafunction transceiver datapath when configured in XAUI mode.

Figure 1–77. Transceiver Datapath in XAUI Mode



Notes to Figure 1–77:

(1) This allows the fabric-to-transceiver interface to run below the maximum interface frequency.
(2) The word aligner uses the automatic synchronization state machine (10-bit /K28.5/).
(3) This is running at half the rate of the data rate.

Word Aligner in XAUI Mode

The word aligner in XAUI functional mode is configured in automatic synchronization state machine mode that is compliant to the PCS synchronization state diagram specified in section 8 of the IEEE P802.3ae specification. The Quartus II software automatically configures the synchronization state machine to indicate synchronization when the receiver acquires four /K28.5/ comma code groups without intermediate invalid code groups.

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. The receiver loses synchronization when it detects four invalid code groups separated by less than four valid code groups, or when it is reset.

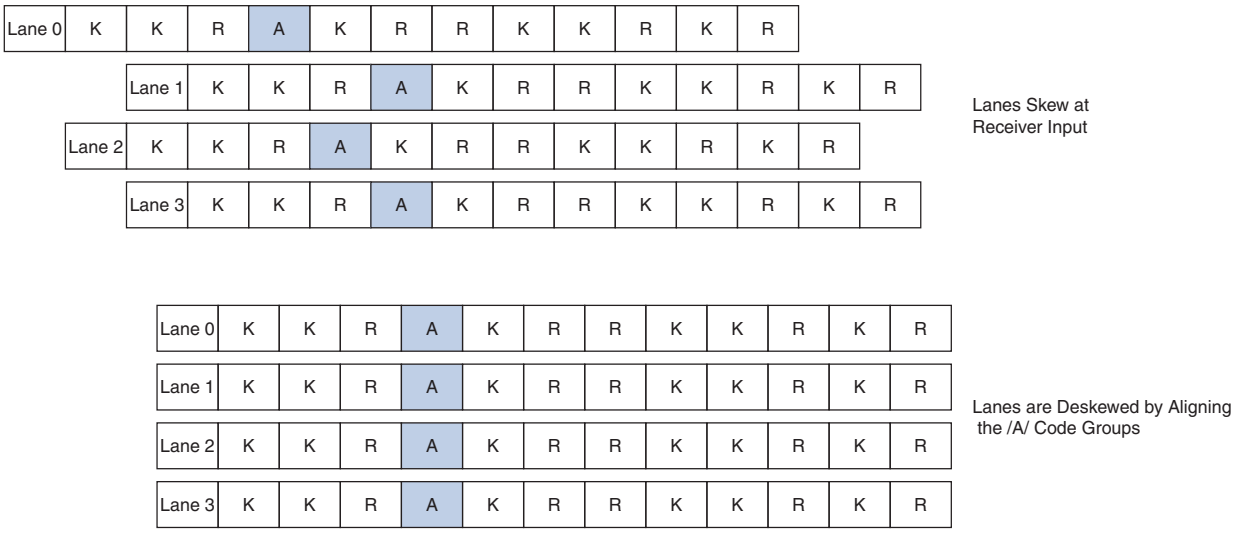
Deskew FIFO in XAUI Mode

The XAUI protocol requires the physical layer device to implement deskew circuitry to align all four channels. The skew introduced in the physical medium and the receiver channels can cause the /A/ code groups to be received misaligned with respect to each other. To enable the deskew circuitry at the receiver to align the four channels, the transmitter sends an /A/ (/K28.3/) code group simultaneously on all four channels during an inter-packet gap (IPG). The deskew operation begins only after link synchronization is achieved on all four channels as indicated by a high level on the `rx_syncstatus` signal from the word aligner in each channel. Until the first /A/ code group is received, the deskew FIFO read and write pointers in each channel are not incremented. After the first /A/ code group is received, the write pointer starts incrementing for each word received but the read pointer is frozen. If the /A/ code group is received on each of the four channels in 10 recovered clock cycles of each other, the read pointer of all four deskew FIFOs is released simultaneously, aligning all four channels.

The deskew FIFO operation in XAUI functional mode is compliant to the PCS deskew state machine diagram specified in 8 of the IEEE P802.3ae specification.

Figure 1–78 shows lane skew at the receiver input and how the deskew FIFO uses the /A/ code group to align the channels.

Figure 1–78. Deskew FIFO–Lane Skew at the Receiver Input



After alignment of the first ||A|| column, if three additional aligned ||A|| columns are observed at the output of the deskew FIFOs of the four channels, the `rx_channelaligned` signal is asserted high, indicating channel alignment is acquired. After acquiring channel alignment, if four misaligned ||A|| columns are seen at the output of the deskew FIFOs in all four channels with no aligned ||A|| columns in between, the `rx_channelaligned` signal is de-asserted low, indicating loss of channel alignment.

Rate Match FIFO in XAUI Mode

In XAUI mode, the rate match FIFO is capable of compensating up to ±100 PPM (200 PPM total) difference between the upstream transmitter and the local receiver reference clock. The XAUI protocol requires the transmitter to send /R/ (/K28.0/) code groups simultaneously on all four lanes (denoted as the ||R|| column) during inter-packet gaps, adhering to rules listed in the IEEE P802.3ae specification.

The rate match operation begins after `rx_syncstatus` and `rx_channelaligned` are asserted. The `rx_syncstatus` signal is from the word aligner, indicating that synchronization is acquired on all four channels; the `rx_channelaligned` signal is from the deskew FIFO, indicating channel alignment.

The rate match FIFO looks for the ||R|| column (simultaneous /R/ code groups on all four channels) and deletes or inserts ||R|| columns to prevent the rate match FIFO from overflowing or underrunning. The rate match FIFO can insert or delete as many ||R|| columns as necessary to perform the rate match operation.

The `rx_rmfifodatadeleted` and `rx_rmfifodatainserted` flags indicate rate match FIFO deletion and insertion events, respectively, and are forwarded to the FPGA fabric. If an ||R|| column is deleted, the `rx_rmfifodeleted` flag from each of the four channels goes high for one clock cycle per deleted ||R|| column. If an ||R|| column is inserted, the `rx_rmfifoinserted` flag from each of the four channels goes high for one clock cycle per inserted ||R|| column.

Figure 1–79 shows an example of rate match deletion in the case where three ||R|| columns must be deleted.

Figure 1–79. Example of Rate Match Deletion in XAUI Mode

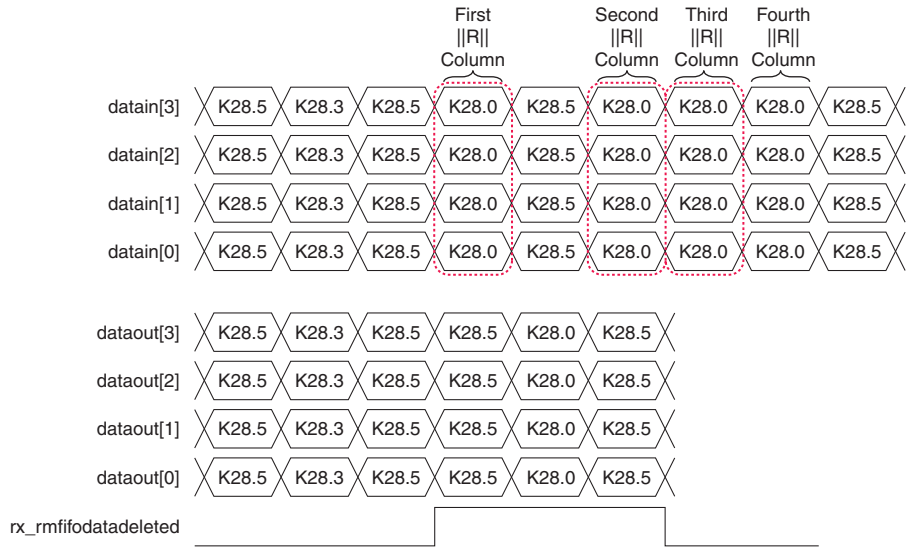
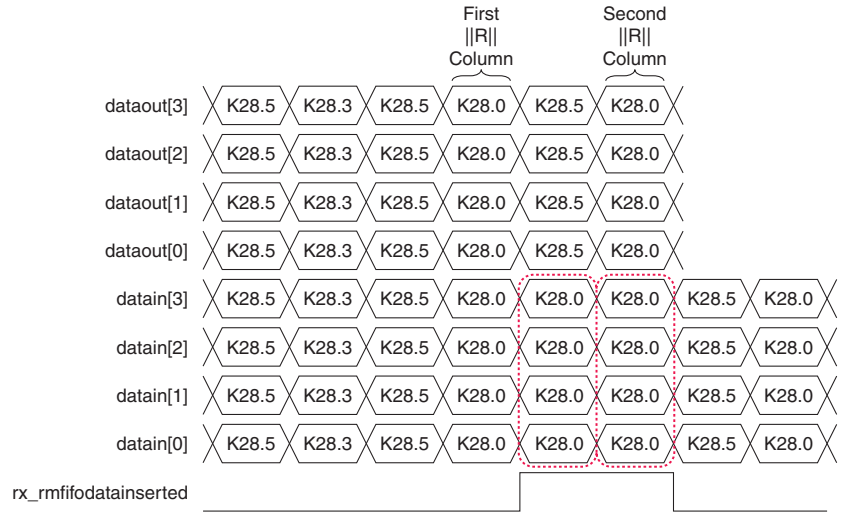


Figure 1–80 shows an example of rate match insertion in the case where two ||R|| columns must be inserted.

Figure 1–80. Example of Rate Match Insertion in XAUI Mode



CAUTION

The rate match FIFO does not insert or delete code groups automatically to overcome FIFO empty or full conditions. In this case, the rate match FIFO asserts the `rx_rmfifofull` and `rx_rmfifoempty` flags for at least three recovered clock cycles to indicate rate match FIFO full and empty conditions, respectively. You must then assert the `rx_digitalreset` signal to reset the receiver PCS blocks.

Test Modes

Arria II GX and GZ devices provide various loopback options, pattern generators, and verifiers that allow you to ensure the working of different functional blocks in the transceiver channel. These modes include:

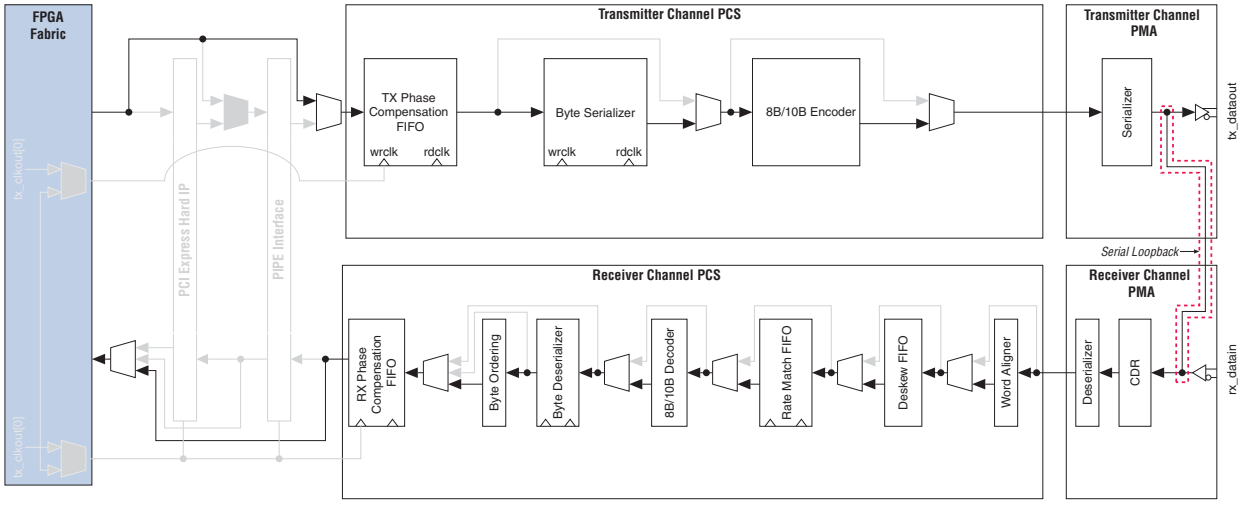
- Serial loopback
- Reverse serial loopback
- Reverse serial pre-CDR loopback
- PCIe reverse parallel loopback
- BIST and PRBS Modes

If you generate a **Transmitter-only** or **Receiver-only** configuration and enable loopback mode, you will have an extra port that you must connect to the transceiver's counterpart port. These ports are described in Table 1–31 on page 1–98 (the PMA port list).

Serial Loopback

This option is available for all functional modes except PCIe mode. Figure 1–81 shows the datapath for serial loopback.

Figure 1–81. Serial Loopback Datapath



The data from the FPGA fabric passes through the transmitter channel and loops back to the receiver channel, bypassing the receiver input buffer. The received data is available to the FPGA logic for verification. Using this option, you can check the operation for all enabled PCS and PMA functional blocks in the transmitter and receiver channels.

When you enable the serial loopback option, the ALTGX MegaWizard Plug-In Manager provides the `rx_seriallpbken` port to dynamically enable serial loopback on a channel-by-channel basis when the signal is asserted high.

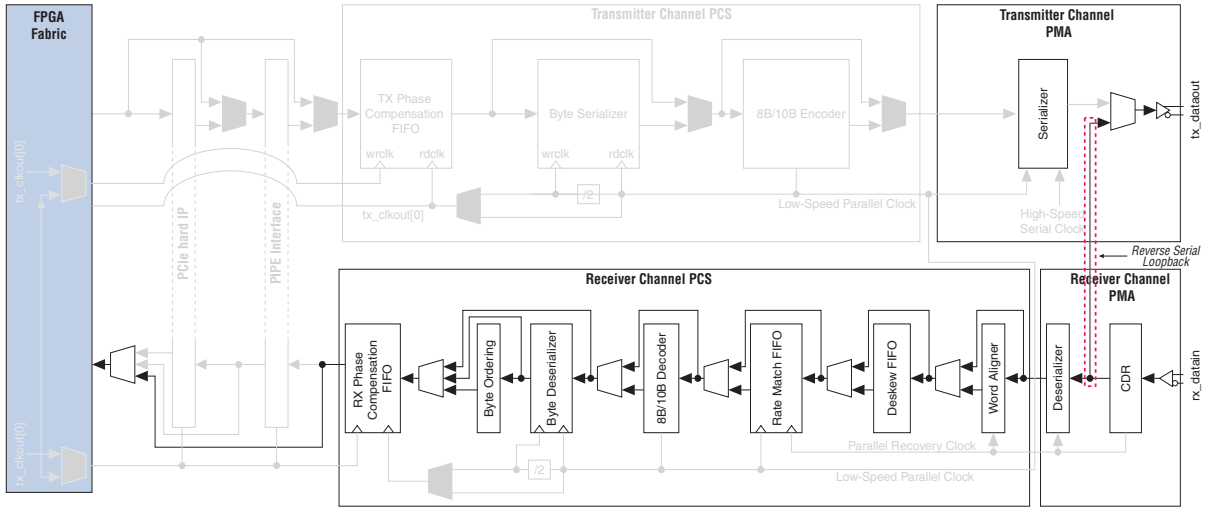
When you enable serial loopback, the transmitter channel sends the data to both the tx_dataout output port and the receiver channel. The differential output voltage on the tx_dataout ports is based on the selected V_{OD} settings. The looped back data is received by the receiver CDR and then timed again through different clock domains. You must provide an alignment pattern for the word aligner to enable the receiver channel to retrieve the byte boundary.

Reverse Serial Loopback

Reverse serial loopback is available in Basic functional mode only and is often implemented when using a bit error rate tester (BERT) on the upstream transmitter. In this mode, the data is received through the rx_datain port, timed again through the receiver CDR, and sent out to the tx_dataout port. The received data is also available to the FPGA logic. You can enable the reverse serial loopback option using the ALTGX MegaWizard Plug-In Manager. Unlike other loopback modes, there is no dynamic pin control to enable or disable reverse serial loopback.

Figure 1–82 shows the transceiver channel datapath for reverse serial loopback mode.

Figure 1–82. Reverse Serial Loopback Datapath ***(Note 1)***



Note to Figure 1–82:

(1) The only active block of the transmitter channel is the transmitter buffer.

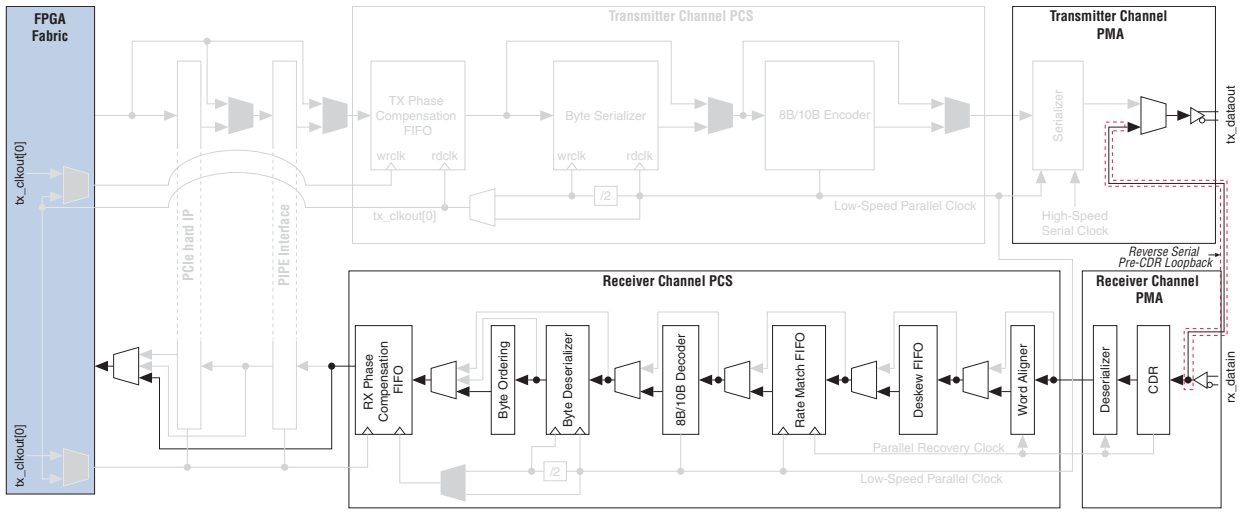
You can change the output differential voltage on the transmitter buffer through the ALTGX MegaWizard Plug-In Manager. However, you cannot alter the pre-emphasis settings for the transmitter buffer.

Reverse Serial Pre-CDR Loopback

Reverse serial pre-CDR loopback is available in Basic functional mode only. In this mode, the data received through the `rx_datain` port is looped back to the `tx_dataout` port before the receiver CDR. The received data is also available to the FPGA logic. You can enable the reverse serial pre-CDR loopback option using the ALTGX MegaWizard Plug-In Manager. Unlike other loopback modes, there is no dynamic pin control to enable or disable reverse serial pre-CDR loopback.

Figure 1–83 shows the transceiver channel datapath for reverse serial pre-CDR loopback mode.

Figure 1–83. Reverse Serial Pre-CDR Loopback Datapath ***(Note 1)***



Note to Figure 1–83:

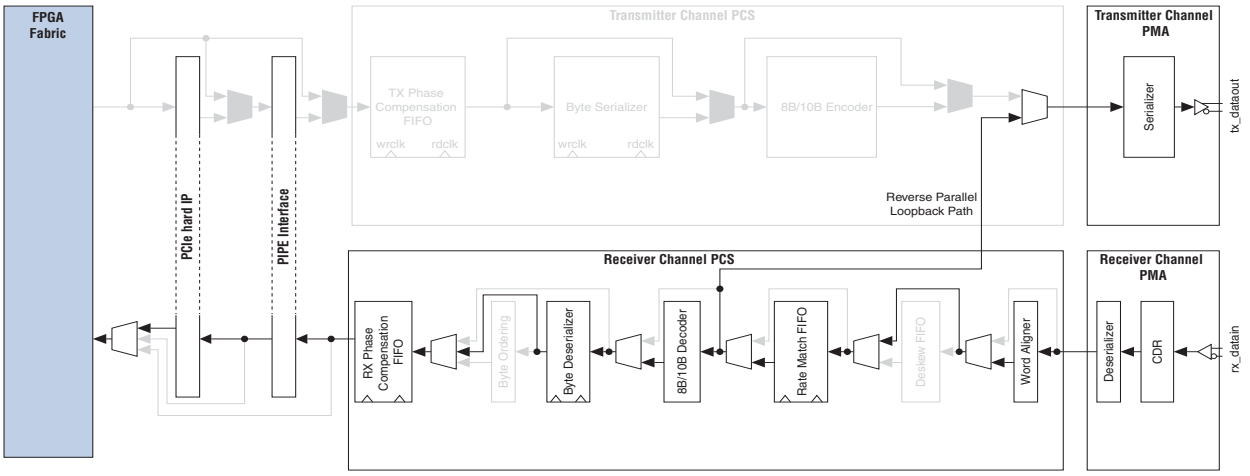
(1) The only active block of the transmitter channel is the transmitter buffer.

You can change the V_{OD} on the transmitter buffer through the ALTGX MegaWizard Plug-In Manager. However, you cannot change the pre-emphasis settings for the transmitter buffer.

PCIe (Reverse Parallel Loopback)

PCIe reverse parallel loopback is only available in PCIe functional mode for the Gen1 and Gen2 data rates. As shown in Figure 1–84, the received serial data passes through the receiver CDR, deserializer, word aligner, and rate match FIFO buffer. The data is then looped back to the transmitter serializer and transmitted out through the `tx_dataout` port. The received data is also available to the FPGA fabric through the `rx_dataout` port. This loopback mode is compliant with the PCIe Base Specification 2.0. To enable PCIe reverse parallel loopback mode, assert the `tx_detectrxloop` port.

Figure 1–84. PCIe Reverse Parallel Loopback Mode Datapath

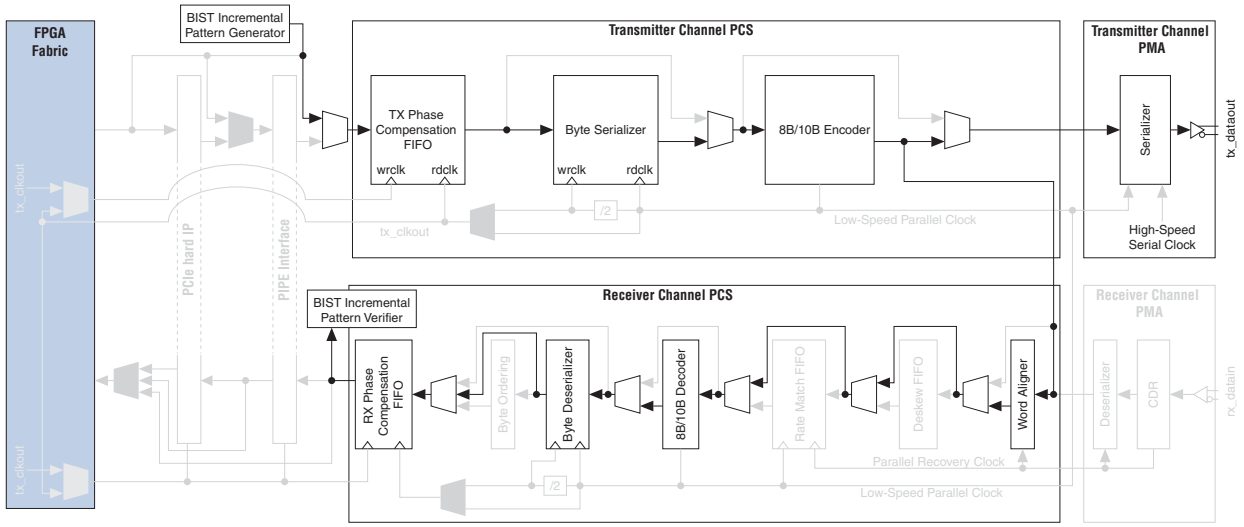


Built-In Self Test (BIST) and Pseudo Random Binary Sequence (PRBS)

Each transceiver channel in Arria II GX and GZ devices contain a pattern generator and a pattern verifier circuit. Using these patterns, you can verify the functionality of the functional blocks in the transceiver channel without requiring user logic. The functionality is provided as an optional mechanism for debugging transceiver channels. To use the Arria II GX and GZ pattern generator and verifier, use the pattern BIST and PRBS sub-protocols under Basic functional mode.

Figure 1–85 shows the datapath for the BIST mode.

Figure 1–85. Enabled PCS Functional Blocks in Parallel Loopback

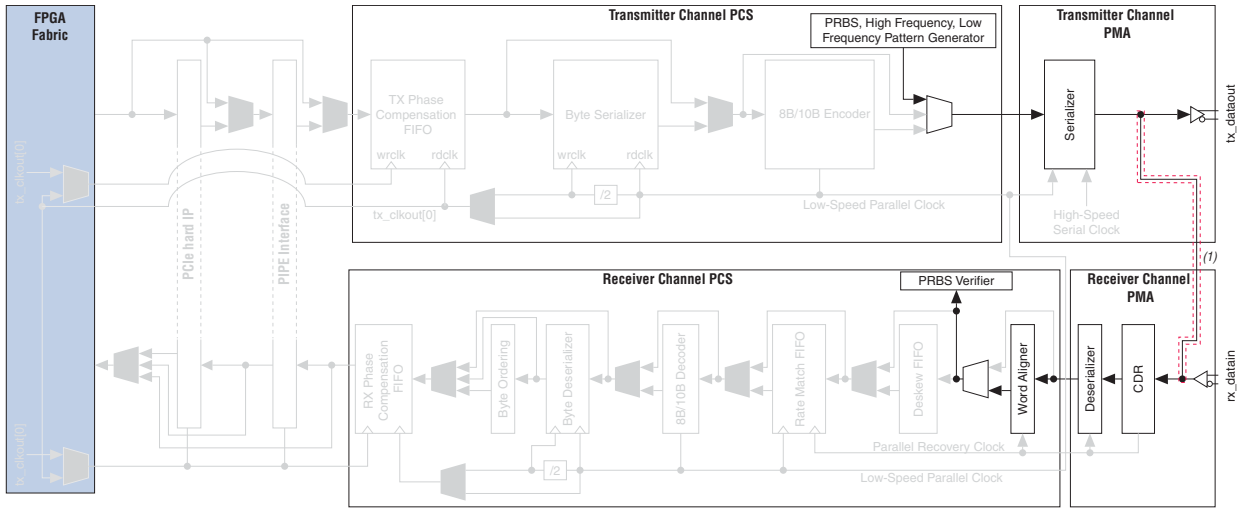


BIST mode allows you to verify the complete PCS blocks for both the transmitter and receiver channel. This mode is available only with a built-in 16-bit incremental pattern generator and verifier; therefore, you must set the channel width to **16 bits** in this mode. The incremental pattern 00-FF is looped back to the receiver channel at the PCS functional block boundary before the PMA and is sent out to the `tx_dataout` port.

The received data is verified by the verifier, but is not available in the FPGA fabric. The V_{OD} of the transmitted serial data on the `tx_dataout` port is based on the selected V_{OD} settings.

Figure 1–86 shows the datapath for the PRBS patterns. The generated pattern is sent to the serializer. The verifier checks the data from the word aligner.

Figure 1–86. Datapath for the PRBS Mode



Note to Figure 1–86:

(1) Serial loopback can be dynamically enabled through the `rx_seriallpbken` port.

PRBS mode has two pattern generator options, selectable in the **BIST** tab of the MegaWizard Plug-In Manager when you choose PRBS as a sub protocol under Basic functional mode.

- PRBS7, PRBS8, PRBS10, and PRBS23 generator and verifier—This is the generator and verifier interface with the serializer and deserializer in the PMA blocks. The advantage of using a PRBS data stream is that the randomness yields an environment that stresses the transmission medium. In the data stream, you can observe both random jitter and deterministic jitter using a time interval analyzer, bit error rate tester, or oscilloscope.

  The PRBS repeats after completing an iteration. The number of bits the PRBSx pattern sends before repeating the pattern is (2^×-1) bits. This mode is available as a sub protocol under Basic functional mode.

- High-frequency and low-frequency pattern generator—The high-frequency patterns generate alternate ones and zeros and the low-frequency patterns generate five ones and five zeroes in single width mode, and ten ones and ten zeroes in double width mode. These patterns do not have a corresponding verifier.

Table 1–23 lists various PRBS patterns and corresponding word alignment patterns.

Table 1–23. Patterns in PRBS Mode for Arria II Devices (Part 1 of 2)

Patterns	Polynomial	Channel Width of 8-Bit *(1)*	Word Alignment Pattern	Maximum Data Rate
PRBS 7	$X^7 + X^6 + 1$	8 bit	16'h3040	2.5
PRBS 8	$X^8 + X^7 + 1$	8 bit	16'hFF5A	2.5
PRBS 10	$X^{10} + X^7 + 1$	10 bit	10'h3FF	3.125
PRBS 23	$X^{23} + X^{18} + 1$	8 bit	16'hFFFF	2.5

Table 1–23. Patterns in PRBS Mode for Arria II Devices (Part 2 of 2)

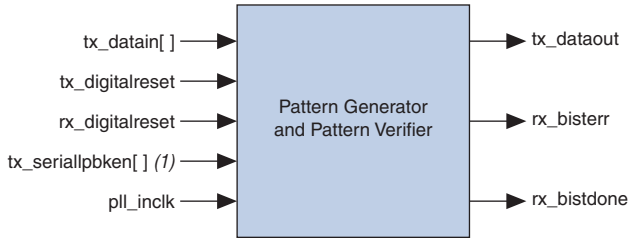
Patterns	Polynomial	Channel Width of 8-Bit *(1)*	Word Alignment Pattern	Maximum Data Rate
High frequency *(1)*	1010101010	8 or 10 bit	NA	2.5 for 8-bit pattern and 3.125 for 10-bit pattern
Low frequency *(1)*	0000011111	10 bit	NA	3.125

Note to Table 1–23:

(1) A verifier is not available for the specified patterns.

Figure 1–87 shows the enabled input and output ports of the pattern generator and pattern verifier.

Figure 1–87. Input and Output Ports for the BIST and PRBS Modes



Note to Figure 1–87:

(1) `rx_serialpbken` is optional.

You can reset the PRBS pattern generator and verifier by asserting the `tx_digitalreset` and `rx_digitalreset` signals, respectively.

`rx_digitalreset` does not reset the BIST output signals when the following conditions are true:

- `pll_powerdown` is high in BIST mode
- `pll_powerdown` or `rx_analogreset` is high in PRBS mode

Dynamic Reconfiguration

Dynamic reconfiguration allows you to reconfigure the transceiver block without reconfiguring the FPGA. For hot-plug or open-standard systems, this feature allows you to support multiple data rates or standards without reconfiguring the system. For all systems, it allows you to make changes to the bit error rate to compensate in-system for the effects of process and temperature.

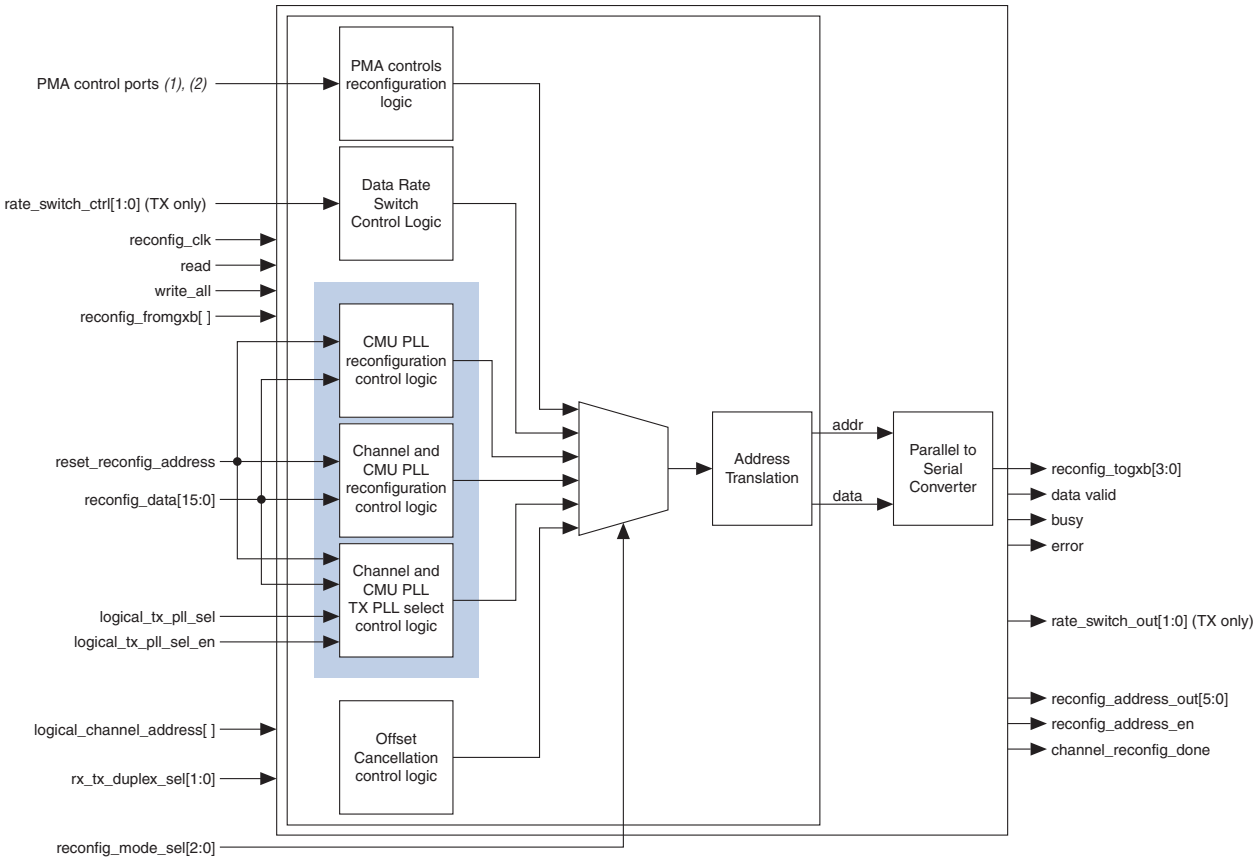
For more information, refer to *AN 558: Implementing Dynamic Reconfiguration in Arria II Devices*.

Each transceiver channel has multiple physical medium attachment controls that you can program to achieve the desired bit error ratio (BER) for your system. When you enable the dynamic reconfiguration feature, you can reconfigure the following portions of each transceiver channel dynamically (one channel at a time) without powering down the other transceiver channels or the FPGA fabric of the device:

- Transmit and receive analog settings
- Transmit data rate in multiples of 1, 2, and 4
- Channel and clock multiplier unit PLL
- CMU PLL only

The dynamic reconfiguration controller is a soft IP that uses FPGA-fabric resources. You can use only one dynamic reconfiguration controller per transceiver block. You cannot use the dynamic reconfiguration controller to control multiple Arria II GX and GZ devices or off-chip interfaces. Figure 1–88 shows the conceptual view of the dynamic reconfiguration controller architecture.

Figure 1–88. Block Diagram of the Dynamic Reconfiguration Controller



Notes to Figure 1–88:

(1) The PMA control ports consist of the V_{OD} controls, pre-emphasis controls, DC gain controls, and manual equalization controls.

(2) Only PMA reconfiguration mode supports manual equalization controls.

The dynamic reconfiguration controller requires input from one of the following:

- Its input ports through user logic where they are translated to the address and data bus inside the controller. The address and data bus are then converted into serial data and forwarded to the selected transceiver channel
- A Memory Initialization File (**.mif**) where the controller receives 16-bit words from the **.mif** that you generate and sends this information to the transceiver channel selected

The different modes of dynamic reconfiguration are:

- PMA settings reconfiguration, available for the following PMA settings:
  - Pre-emphasis settings
  - Equalization settings (channel reconfiguration mode does not support equalization settings)
  - DC gain settings
  - V_{OD} settings
- Receiver offset cancellation

  Process variations create offsets in analog circuit voltages, pushing them outside the expected range. The Arria II GX and GZ devices provide an offset cancellation circuit per receiver channel to counter the offset variations due to process.

  Calibration of the offset cancellation circuit is done at power-up. The receiver input buffer and receiver CDR require offset calibration. Offset cancellation is automatically executed whenever the device is powered on. The control logic for offset cancellation is integrated into the dynamic reconfiguration controller.

  The offset cancellation for the receiver channels option is automatically enabled in both the ALTGX and ALTGX_RECONFIG MegaWizard Plug-In Managers for **Receiver**, **Transmitter**, and **Receiver only** configurations. It is not available for **Transmitter only** configurations.

  
  When offset cancellation is automatically enabled, you must instantiate the dynamic reconfiguration controller to connect the reconfiguration ports created by the ALTGX MegaWizard Plug-In Manager.

  
  For more information about implementing the ALTGX_RECONFIG MegaWizard Plug-In Manager, refer to *AN 558: Implementing Dynamic Reconfiguration in Arria II Devices*.

  You must always connect the ALTGX_RECONFIG instance to the ALTGX (with receiver channels) instance in your design. Connect the `reconfig_fromgxb`, `reconfig_togxb`, and necessary clock signals to both the ALTGX_RECONFIG and ALTGX (with receiver channels) instances. For transmitter-only configuration, the ALTGX_RECONFIG must also be connected to the ALTGX either by selecting a dynamic reconfiguration mode or by instantiating a dummy receiver-only ALTGX instance in each side of the device.

  
  The offset cancellation process changes the transceiver reset sequence. For more information, refer to the *Reset Control and Power Down* chapter.

- Transceiver channel reconfiguration—for transceiver channels, dynamic reconfiguration involves the reconfiguration of the following:
  - Data Rate Reconfiguration—achievable by switching between two TX PLLs set to different data rates or reconfiguring the RX PLLS or reconfiguring the local dividers in the transmit side

    Ensure that the functional mode of the transceiver channel supports the reconfigured data rate.

  - CMU PLL Reconfiguration
  - Functional Mode Reconfiguration

    Ensure that the various clocks involved support the transition.

Transceiver Port List

You instantiate the Arria II GX and GZ transceivers with the ALTGX megafunction instance in the Quartus II MegaWizard Plug-In Manager. The ALTGX megafunction 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.

These signals are available if you enable the block associated with them.

Table 1–24 lists the CMU port names and descriptions for the ALTGX megafunction.

Table 1–24. ALTGX Megafunction CMU Ports for Arria II Devices

Port Name	Input/Output	Description
pll_inclk	Input	Input reference clock for the CMU PLL.
pll_powerdown	Input	Asynchronous active-high signal to power down both CMU PLLs. The minimum pulse width for this signal is specified in the *Device Datasheet for Arria II Devices* chapter. Note: Asserting the pll_powerdown signal does not power down the refclk buffers. Note: While each CMU PLL has its own pll_powerdown port, the ALTGX MegaWizard Plug-In Manager instantiation provides only one port per transceiver block. This port power downs one or both CMU PLLs (if used).
coreclkout	Output	A low-speed parallel clock generated by the CMU0 clock divider for bonded channel configurations. This signal is generated by the CMU0 clock divider in the master transceiver block in ×8 bonded channel configurations and is not available in non-bonded channel configurations. If the byte serializer block is enabled in bonded channel modes, the coreclkout clock output is half the frequency of the low-speed parallel clock. Otherwise, the coreclkout clock output is the same frequency as the low-speed parallel clock. You can also use this clock on the write and read clock ports of the TX phase compensation FIFOs in all bonded channels if tx_coreclk is not enabled in the ALTGX MegaWizard Plug-In Manager.
pll_locked	Output	Asynchronous active-high signal to indicate whether the CMU PLL is locked.

Table 1–25 lists the word aligner port names and descriptions for the ALTGX megafunction.

Table 1–25. ALTGX Megafunction Word Aligner Ports for Arria II Devices (Part 1 of 2)

Port Name	Input/Output	Description
rx_ala2size	Input	Available only in SONET OC-12 and OC-48 modes to select between one of the following two word alignment options: Logic Level — Word Alignment Pattern 0 — 16-bit A1A2 1 — 32'bit A1A1A2A2
rx_bitslip	Input	Asynchronous bit-slip control when the word aligner is configured in bit-slip mode. At every rising edge of this signal, the word aligner slips one bit into the received data stream, effectively shifting the word boundary by one bit. The minimum pulse-width is two recovered clock cycles.
rx_enapatternalign	Input	Asynchronous manual word alignment enable control. This signal is edge-sensitive with 8-bit width data and level sensitive with 10-bit width data. The minimum pulse-width is two recovered clock cycles.
rx_invpolarity	Input	Asynchronous receiver polarity inversion control. When asserted high, the polarity of every bit of the 8-bit or 10-bit input data word to the word aligner is inverted.
rx_revbitorderwa	Input	Asynchronous receiver bit reversal control. Available only in Basic mode with the word aligner configured in bit-slip mode. When asserted high in Basic mode, the 8-bit or 10-bit data D[7:0] or D[9:0] at the output of the word aligner is rewired to D[0:7] or D[0:9], respectively.
rx_bitslipboundary selectout	Output	Asynchronous signal indicating the number of bits slipped in the word aligner when the word aligner is configured in manual mode.
rx_patterndetect	Output	Word alignment pattern detect indicator. A high level indicates that the word alignment pattern is found on the current word boundary. The width of this signal depends on the channel width shown below: Channel Width — rx_patterndetect width 8/10 — 1 16/20 — 2

Table 1–25. ALTGX Megafunction Word Aligner Ports for Arria II Devices (Part 2 of 2)

Port Name	Input/Output	Description
rx_rlv	Output	Asynchronous run-length violation indicator. A high pulse is driven when the number of consecutive 1s or 0s in the received data stream exceeds the programmed run length violation threshold. This signal is driven for a minimum of two recovered clock cycles in configurations without byte serializer and a minimum of three recovered clock cycles in configurations with byte serializer.
rx_syncstatus	Output	Word alignment synchronization status indicator. For word aligner in automatic synchronization state machine mode, this signal is driven high if the conditions required to remain in synchronization are met. For word aligner in manual alignment mode, this signal is driven high for one parallel clock cycle synchronous to the MSByte of the word alignment pattern. This signal is not available for word aligner in bit-slip mode. The width of this signal depends on the channel width shown below: Channel Width — rx_syncstatus width 8/10 — 1 16/20 — 2

Table 1–26 lists the deskew FIFO port name and description for the ALTGX megafunction.

Table 1–26. ALTGX Megafunction Deskew FIFO Port for Arria II Devices

Port Name	Input/Output	Description
rx_channelaligned	Output	Indicates whether all the channels are aligned. This signal is only available in XAUI mode. A high level indicates that the XAUI deskew state machine is either in a ALIGN_ACQUIRED_1, ALIGN_ACQUIRED_2, ALIGN_ACQUIRED_3, or ALIGN_ACQUIRED_4 state, as specified in the PCS deskew state diagram in IEEE P802.3ae specification. A low level indicates that the XAUI deskew state machine is either in a LOSS_OF_ALIGNMENT, ALIGN_DETECT_1, ALIGN_DETECT_2, or ALIGN_DETECT_3 state, as specified in the PCS deskew state diagram in IEEE P802.3ae specification.

Table 1–27 lists the rate match (clock rate compensation) FIFO port names and descriptions for the ALTGX megafunction.

Table 1–27. ALTGX Megafunction Rate Match (Clock Rate Compensation) FIFO Ports for Arria II Devices (Part 1 of 2)

Port Name	Input/Output	Description
rx_rmfifodatadeleted	Output	Rate match FIFO deletion status indicator. A high level indicates that the rate match pattern byte was deleted to compensate for the PPM difference in reference clock frequencies between the upstream transmitter and the local receiver.
rx_rmfifodatainserted	Output	Rate match FIFO insertion status indicator. A high level indicates that the rate match pattern byte was inserted to compensate for the PPM difference in reference clock frequencies between the upstream transmitter and the local receiver.

Table 1–27. ALTGX Megafunction Rate Match (Clock Rate Compensation) FIFO Ports for Arria II Devices (Part 2 of 2)

Port Name	Input/Output	Description
rx_rmfifoempty	Output	Asynchronous rate match FIFO empty status indicator. A high level indicates that the rate match FIFO is empty. This signal is driven for a minimum of two recovered clock cycles in configurations without byte serializer and a minimum of three recovered clock cycles in configurations with byte serializer. You must then assert the rx_digitalreset signal to reset this signal.
rx_rmfifofull	Output	Asynchronous rate match FIFO full status indicator. A high level indicates that the rate match FIFO is full. This signal is driven for a minimum of two recovered clock cycles in configurations without byte serializer and a minimum of three recovered clock cycles in configurations with byte serializer. You must then assert the rx_digitalreset signal to reset this signal.

Table 1–28 lists the 8B/10B decoder port names and descriptions for the ALTGX megafunction. These ports are 1-bit wide with 8-bit channel width and 2-bit wide with 16-bit channel width.

Table 1–28. ALTGX Megafunction 8B/10B Decoder Ports for Arria II Devices

Port Name	Input/Output	Description
rx_ctrldetect	Output	Receiver control code indicator. A high level indicates that the associated received code group is a control (/Kx.y/) code group. A low level indicates that the associated received code group is a data (/Dx.y/) code group.
rx_disperr	Output	8B/10B disparity error indicator port. A high level indicates that a disparity error was detected on the associated received code group.
rx_errdetect	Output	8B/10B code group violation or disparity error indicator. A high level indicates that a code group violation or disparity error was detected on the associated received code group. Use with the rx_disperr signal to differentiate between a code group violation and/or a disparity error as follows: [rx_errdetect: rx_disperr] 2'b00—no error 2'b10—code group violation 2'b11—disparity error or both
rx_runningdisp	Output	8B/10B running disparity indicator. A high level indicates that data on the rx_dataout port was received with a negative running disparity. A low level indicates that data on the rx_dataout port was received with a positive running disparity.

Table 1–29 lists the byte ordering block port names and descriptions for the ALTGX megafunction.

Table 1–29. ALTGX Megafunction Byte Ordering Block Ports for Arria II GX and GZ Devices

Port Name	Input/Output	Description
rx_enabyteord	Input	Asynchronous enable byte ordering control. The byte ordering block is rising-edge sensitive to this signal. A low-to-high transition triggers the byte ordering block to restart the byte ordering operation.
rx_byteorderalignstatus	Output	Byte ordering status indicator. A high level indicates that the byte ordering block has detected the programmed byte ordering pattern in the LSByte of the received data from the byte deserializer.

Table 1–30 lists the RX phase compensation FIFO port names and descriptions for the ALTGX megafunction.

Table 1–30. ALTGX Megafunction RX Phase Compensation FIFO Ports for Arria II Devices

Port Name	Input/Output	Description
coreclkout	Input	Clock from the CMU0 block of the associated transceiver block or the master transceiver block for ×4 bonded or ×8 bonded channel configurations, respectively. This is the default read and write clocks for those configurations.
rx_coreclk	Input	Optional read clock port for the RX phase compensation FIFO. If not enabled, the Quartus II software automatically selects rx_clkout/tx_clkout/coreclkout as the read clock for the RX phase compensation FIFO. If selected, you must drive this port with a clock that has 0 PPM difference with respect to the FIFO write clock.
rx_clkout	Input	Recovered clock from the receiver channel. This is the default read and write clocks for the RX phase compensation FIFO in non-bonded configurations without the rate-match FIFO.
tx_clkout	Input	Clock from the transmitter channel local clock divider. This is the default read and write clocks for the RX phase compensation FIFO in non-bonded configurations with the rate-match FIFO.
rx_dataout	Output	Parallel data output from the receiver to the FPGA fabric. The bus width depends on the channel width multiplied by the number of channels per instance.
rx_phase_comp_fifo_error	Output	RX phase compensation FIFO full or empty indicator. A high level indicates that the RX phase compensation FIFO is either full or empty.

Table 1–31 lists the receiver physical medium attachment (PMA) port names and descriptions for the ALTGX megafunction.

Table 1–31. ALTGX Megafunction Receiver PMA Ports for Arria II Devices (Part 1 of 2)

Port Name	Input/Output	Description
rx_cruclk	Input	Input reference clock for the receiver CDR.
rx_datain	Input	Receiver serial data input port.
rx_locktodata	Input	Asynchronous receiver CDR LTD mode control signal. When asserted high, the receiver CDR is forced to LTD mode. When de-asserted low, the receiver CDR lock mode depends on the rx_locktorefclk signal level.

Table 1–31. ALTGX Megafunction Receiver PMA Ports for Arria II Devices (Part 2 of 2)

Port Name	Input/Output	Description
rx_locktorefclk	Input	Asynchronous receiver CDR LTR mode control signal. The rx_locktorefclk and rx_locktodata signals control whether the receiver CDR is in LTR or LTD mode, as follows: rx_locktodata/rx_locktorefclk 0/0—receiver CDR is in automatic mode 0/1—receiver CDR is in LTR mode 1/x—receiver CDR is in LTD mode
rx_seriallpbken	Input	Active-high serial loopback control port.
rx_seriallpbkin	Input	Input on a **Receiver-only** configuration when you enable serial loopback. You must connect this port to the tx_seriallpbkout port.
tx_revseriallpbkin	Input	Input on a **Transmitter-only** configuration when you enable reverse serial loopback. You must connect this port to the rx_revseriallpbkout port.
rx_freqlocked	Output	Asynchronous receiver CDR lock mode indicator. A high level indicates that the receiver CDR is in LTD mode. A low level indicates that the receiver CDR is in LTR mode.
rx_pll_locked	Output	Asynchronous active high receiver CDR LTR indicator. The receiver CDR is locked to the input reference clock.
rx_revseriallpbkout	Output	Output on a **Receiver-only** configuration when you enable reverse serial loopback. You must connect this port to the tx_revseriallpbkin port.
tx_seriallpbkout	Output	Output on a **Transmitter-only** configuration when you enable serial loopback. You must connect this port to the tx_seriallpbkout port.
rx_signaldetect	Output	Asynchronous signal threshold detect indicator for PCIe mode only. A high level indicates that the signal present at the receiver input buffer is above the programmed signal detection threshold value. If the electrical idle inference block is disabled in PCIe mode, the rx_signaldetect signal is inverted and driven on the pipeelecidle port.

Table 1–32 lists the TX phase compensation FIFO port names and descriptions for the ALTGX megafunction.

Table 1–32. ALTGX Megafunction TX Phase Compensation FIFO Ports for Arria II Devices

Port Name	Input/Output	Description
tx_coreclk	Input	Optional write clock port for the TX phase compensation FIFO. If enabled, you must drive this port with a clock that is frequency locked to tx_clkout/coreclkout (with 0 PPM frequency difference).
tx_datain	Input	Parallel data input from the FPGA fabric to the transmitter. The bus width depends on the channel width multiplied by the number of channels per instance.
tx_clkout	Input	FPGA fabric-transceiver interface clock. Each channel has a tx_clkout signal in non-bonded channel configurations.
tx_phase_comp_fifo_error	Output	TX phase compensation FIFO full or empty indicator. A high level indicates that the TX phase compensation FIFO is either full or empty.
coreclkout	Input	Clock from the CMU0 block of the associated transceiver block or of the master transceiver block for ×4 bonded or ×8 bonded channel configurations, respectively. This is the default read and write clocks for those configurations.

Table 1–33 lists the 8B/10B encoder port names and descriptions for the ALTGX megafunction.

Table 1–33. ALTGX Megafunction 8B/10B Encoder Ports for Arria II Devices

Port Name	Input/Output	Description
tx_bitslipboundaryselect	Input	Indicates the number of bits to slip at the transmitter for word alignment at the receiver.
tx_ctrlenable	Input	8B/10B encoder /Kx.y/ or /Dx.y/ control. When asserted high, the 8B/10B encoder encodes the data on the tx_datain port as a /Kx.y/ control code group. When de-asserted low, it encodes the data on the tx_datain port as a /Dx.y/ data code group. The width of this signal depends on the channel width shown below: Channel Width — tx_ctrlenable 8 — 1 16 — 2
tx_dispval	Input	8B/10B encoder force disparity value. A high level on the tx_dispval signal when the tx_forcedisp signal is asserted high forces the 8B/10B encoder to encode the data on the tx_datain port with a negative starting running disparity. A low level on the tx_dispval signal when the tx_forcedisp signal is asserted high forces the 8B/10B encoder to encode the data on the tx_datain port with a positive starting running disparity. The width of this signal depends on the channel width shown below: Channel Width — tx_dispval 8 — 1 16 — 2
tx_forcedisp	Input	8B/10B encoder force disparity control. When asserted high, it forces the 8B/10B encoder to encode the data on the tx_datain port with a positive or negative disparity, depending on the tx_dispval signal level. When de-asserted low, the 8B/10B encoder encodes the data on the tx_datain port according to the 8B/10B running disparity rules. The width of this signal depends on the channel width shown below: Channel Width — tx_forcedisp 8 — 1 16 — 2
tx_invpolarity	Input	Asynchronous transmitter polarity inversion control. Useful feature for correcting situations where the positive and negative signals of the differential serial link are accidentally swapped during board layout. When asserted high the polarity of every bit of the 8-bit or 10-bit input data to the serializer is inverted.

Table 1–34 lists the transmitter PMA port names and descriptions for the ALTGX megafunction.

Table 1–34. ALTGX Megafunction Transmitter PMA Ports for Arria II Devices

Port Name	Input/Output	Description
fixedclk	Input	125-MHz clock for receiver detect and offset cancellation in PCIe mode.
tx_dataout	Output	Transmitter serial data output port.

Table 1–35 lists the reconfiguration block port names and descriptions for the ALTGX megafunction.

For more information about these ports, refer to *AN 558: Implementing Dynamic Reconfiguration in Arria II Devices*.

Table 1–35. ALTGX Megafunction Reconfiguration Block Ports for Arria II Devices

Port Name	Input/Output	Description
reconfig_clk	Input	Dynamic reconfiguration clock. This clock is also used for offset cancellation in all modes except PCIe mode.
reconfig_fromgxb	Input	The width of this signal is determined by the value you set in the **What is the number of channels controlled by the reconfig controller?** option in the **Reconfiguration settings** screen.
reconfig_togxb[3:0]	Output	The width of this signal is fixed to four bits. It is independent of the value you set in the **What is the number of channels controlled by the reconfig controller?** option in the **Reconfiguration settings** screen.

Table 1–36 lists the PIPE interface port names and descriptions for the ALTGX megafunction.

Table 1–36. ALTGX Megafunction PIPE Interface Ports for Arria II Devices (Available only in PCIe functional mode) (Part 1 of 2)

Port Name	Input/Output	Description
pipe8b10binvpolarity	Input	PCIe polarity inversion control. Functionally equivalent to the RxPolarity signal defined in PIPE specification revision 2.00. Available only in PCIe mode. When asserted high, the polarity of every bit of the 10-bit input data to the 8B/10B decoder is inverted.
powerdn	Input	PCIe power state control. Functionally equivalent to the PowerDown[1:0] signal defined in PIPE specification revision 2.00. The width of this signal is 2 bits and is encoded as follows: ■ 2'b00: P0—Normal Operation ■ 2'b01: P0s—Low Recovery Time Latency, Low Power State ■ 2'b10: P1—Longer Recovery Time Latency, Lower Power State ■ 2'b11: P2—Lowest Power State
tx_detectrxloop	Input	Receiver detect or PCIe loopback control. Functionally equivalent to the TxDetectRx/Loopback signal defined in PIPE specification revision 2.00. When asserted high in the P1 power state with the tx_forceelecidle signal asserted, the transmitter buffer begins the receiver detection operation. When the receiver detect completion is indicated on the pipephydonestatus port, this signal must be de-asserted. When asserted high in the P0 power state with the tx_forceelecidle signal de-asserted, the transceiver datapath is dynamically configured to support parallel loopback, as described in "PCIe (Reverse Parallel Loopback)" on page 1–88.
tx_forcedispcompliance	Input	Forces the 8B/10B encoder to encode with a negative running disparity. Functionally equivalent to the TxCompliance signal defined in PIPE specification revision 2.00. Must be asserted high only when transmitting the first byte of the PCI Express Compliance Pattern to force the 8B/10B encode with a negative running disparity, as required by the PCIe protocol.

Table 1–36. ALTGX Megafunction PIPE Interface Ports for Arria II Devices (Available only in PCIe functional mode) (Part 2 of 2)

Port Name	Input/Output	Description
tx_forceelecidle	Input	Force transmitter buffer to PCIe electrical idle signal levels. Functionally equivalent to the TxElecIdle signal defined in PIPE specification revision 2.00.
pipeelecidle	Output	Asynchronous signal to indicate whether electrical idle is detected or inferred at the receiver. Functionally equivalent to the RxElecIdle signal defined in PIPE specification revision 2.00. If you enable the electrical idle inference block, it drives this signal high when it infers an electrical idle condition, as described in "Electrical Idle Inference" on page 1–70. Otherwise, it drives this signal low. If the electrical idle inference block is disabled, the rx_signaldetect signal from the signal detect circuitry in the receiver input buffer is inverted and driven on this port.
pipephydonestatus	Output	PHY function completion indicator. Functionally equivalent to the PhyStatus signal defined in PIPE specification revision 2.00. Asserted high for one parallel clock cycle to communicate completion of several PHY functions, such as power state transition and receiver detection.
pipestatus	Output	PCIe receiver status port. Functionally equivalent to the RxStatus[2:0] signal defined in PIPE specification revision 2.00. The width of this signal is 3 bits per channel. The encoding of receiver status on the pipestatus port is as follows: ■ 000—Received data OK ■ 001—1 skip added ■ 010—1 skip removed ■ 011—Receiver detected ■ 100—8B/10B decoder error ■ 101—Elastic buffer overflow ■ 110—Elastic buffer underflow ■ 111—Received disparity error
rx_pipedatavalid	Output	Valid data and control on the rx_dataout and rx_ctrldetect ports indicator. Functionally equivalent to the RxValid signal defined in PIPE specification revision 2.00.

Table 1–37 lists the reset and power down port names and descriptions for the ALTGX megafunction.

For more information, refer to the *Reset Control and Power Down in Arria II Devices* chapter.

Table 1–37. ALTGX Megafunction Reset and Power Down Ports for Arria II Devices

Port Name	Input/Output	Description
gxb_powerdown	Input	Asynchronous transceiver block power down signal. When asserted high, all digital and analog circuitry in the PCS, PMA, CMU of the transceiver block is powered down, except for the refclk buffers.
rx_analogreset	Input	Active-high receiver PMA reset.

Table 1–37. ALTGX Megafunction Reset and Power Down Ports for Arria II Devices

Port Name	Input/Output	Description
rx_digitalreset	Input	Active-high receiver PCS reset.
tx_digitalreset	Input	Active-high transmitter PCS reset.

Table 1–38 lists the calibration block port names and descriptions for the ALTGX megafunction.

Table 1–38. ALTGX Megafunction Calibration Block Ports for Arria II Devices

Port Name	Input/Output	Description
cal_blk_clk	Input	Clock for transceiver calibration blocks.
cal_blk_powerdown	Input	Calibration block power down control.

Table 1–39 lists the verifier port names and descriptions for the ALTGX megafunction.

Table 1–39. ALTGX Megafunction Verifier Ports for Arria II Devices

Port Name	Input/Output	Description
rx_bisterr	output	For BIST mode, the rx_bisterr signal asserts and stays high when the verifier detects an error. For PRBS mode, the rx_bisterr signal assets and stays high for a minimum of three rx_clkout clock cycles when the verifier detects an error and de-asserts if the following PRBS sequence has no error.
rx_bistdone	output	For BIST mode, the rx_bistdone port asserts and stays high when the verifier either receives one full cycle of incremental pattern or detects an error in the receiver data. For PRBS mode, the rx_bistdone port asserts high and stays high when the verifier a full cycle of PRBS pattern.

Document Revision History

Table 1–40 lists the revision history for this chapter.

Table 1–40. Document Revision History (Part 1 of 2)

Date	Version	Changes
February 2015	4.6	■ Added a note to the "Rate-Match FIFO" section. ■ Updated the "Rate Match FIFO in GbE Mode" section.
September 2014	4.5	■ Global: changed tx_detectrxloopback to tx_detectrxloop.
October 2013	4.4	■ Updated the "Dynamic Reconfiguration" section.
July 2012	4.3	■ Removed Fiber Channel in Table 1–2. ■ Finalized information in Table 1–4, Table 1–14, and Table 1–15. ■ Removed OC-3 (155 Mbps) from Table 1–1 and Table 1–2.
December 2011	4.2	■ Updated Table 1–1. ■ Added Table 1–15. ■ Updated Figure 1–55. ■ Minor text edits.

Table 1–40. Document Revision History (Part 2 of 2)

Date	Version	Changes
June 2011	4.1	■ Added Table 1–38. ■ Updated Figure 1–1, Figure 1–2, Figure 1–3, Figure 1–24, Figure 1–35, Figure 1–36, Figure 1–49, Figure 1–53, Figure 1–54, Figure 1–87 and Figure 1–88. ■ Updated Table 1–2, Table 1–6, and Table 1–7. ■ Updated the “Transmitter Output Buffer”, “Programmable Differential OCT”, “Dynamic Reconfiguration”, “PCIe Hard IP Block”, and “GIGE” sections. ■ Minor text edits.
December 2010	4.0	■ Updated to add Arria II GZ information. ■ Updated Figure 1–7. ■ Updated Table 1–20. ■ Updated the “Programmable Equalization, DC Gain, and Offset Cancellation” section. ■ Minor text edits.
July 2010	3.0	■ Updated Figure 1–1, Figure 1–4, Figure 1–45, Figure 1–74, Figure 1–76, Figure 1–48, Figure 1–49, and Figure 1–50. ■ Updated the “Transceiver Block Overview”, “Programmable Equalization, DC gain, and Offset Cancellation”, “TX Phase Compensation FIFO”, “Transmitter Output Buffer”, “Functional Modes”, “PCIe Mode” “Reverse Serial Loopback”, “Reverse Serial Pre-CDR Loopback”, and “Dynamic Reconfiguration” sections. ■ Moved Table 1-17 to the Arria II Device Family Datasheet. ■ Converted protocol information to Table 1–1. ■ Minor text edits. For example, change “PCI Express (PIPE)” to “PCIe” and “8B10B” to “8B/10B”.
November 2009	2.1	■ Updated figures. ■ Updated Base Specification references to 2.0. ■ Removed table 1-4 from 2.0 version and referenced Arria II GX Device Data Sheet
June 2009	2.0	■ Reorganized. ■ Added “Deterministic Latency” on page 1–44 and “Built-In Self Test (BIST) and Pseudo Random Binary Sequence (PRBS)” on page 1–77. ■ Updated all figures. ■ Port list tables were updated.
March 2009	1.1	Updated: ■ Dynamic Reconfiguration Controller Architecture ■ All AN 558: Implementing dynamic Reconfiguration in Arria II GX Devices ■ Transceiver Channel Reconfiguration ■ Table 1.2 and Table 1.4 ■ Figure 1.69 and Figure 1.70 Added: ■ Offset Cancellation in the Receiver Buffer and Receiver CDR ■ Basic Double-Width Mode Configurations
February 2009	1.0	Initial release.