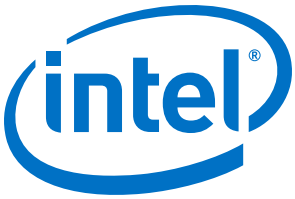


Intel® Cyclone® 10 GX Transceiver PHY User Guide

Updated for Intel® Quartus® Prime Design Suite: **20.1**

Subscribe
Send Feedback

UG-20070 | 2020.05.15
Latest document on the web: PDF | HTML

Contents

1. Intel® Cyclone® 10 GX Transceiver PHY Overview

This user guide provides details about the Intel® Cyclone® 10 GX transceiver physical (PHY) layer architecture, PLLs, clock networks, and transceiver PHY IP core. Intel Quartus® Prime Pro Edition software version 17.1 supports the Intel Cyclone 10 GX transceiver PHY IP core. It also provides protocol specific implementation details and describes features such as transceiver reset and dynamic reconfiguration of transceiver channels and PLLs.

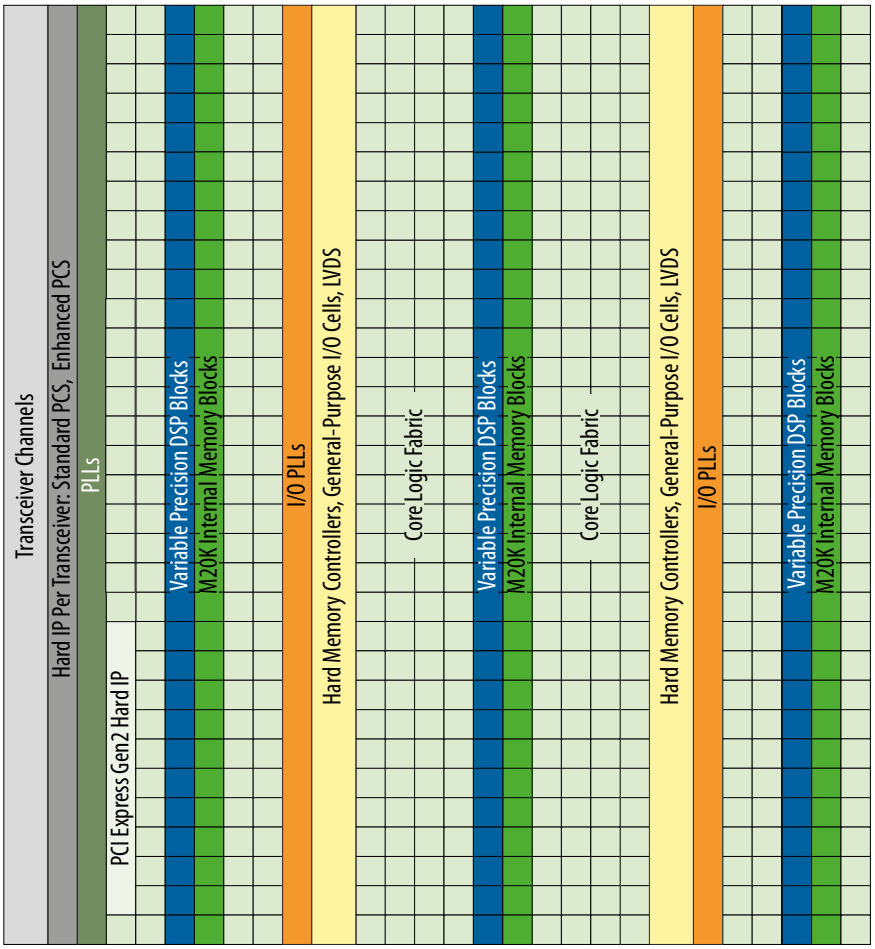
Intel's FPGA Intel Cyclone 10 GX devices offer up to 12 transceiver channels with integrated advanced high speed analog signal conditioning and clock data recovery techniques.

The Intel Cyclone 10 GX devices have transceiver channels that can support data rates up to 12.5 Gbps for chip-to-chip and chip-to-module communication, and up to 6.6 Gbps for backplane communication. You can achieve transmit and receive data rates below 1.0 Gbps with oversampling.

1.1. Device Transceiver Layout

Figure 1. Intel Cyclone 10 GX FPGA Architecture Block Diagram

The transceiver channels are placed on the left side periphery in Intel Cyclone 10 GX devices.



1.1.1. Intel Cyclone 10 GX Device Transceiver Layout

Intel Cyclone 10 GX devices offer 6-, 10-, or 12-transceiver channel counts. Each transceiver bank has up to six transceiver channels. Intel Cyclone 10 GX devices also have one embedded PCI Express Hard IP block.

The figures below illustrate different transceiver bank layouts for Intel Cyclone 10 GX device variants.

Figure 2. Intel Cyclone 10 GX Devices with 12 Transceiver Channels and One PCIe Hard IP Block

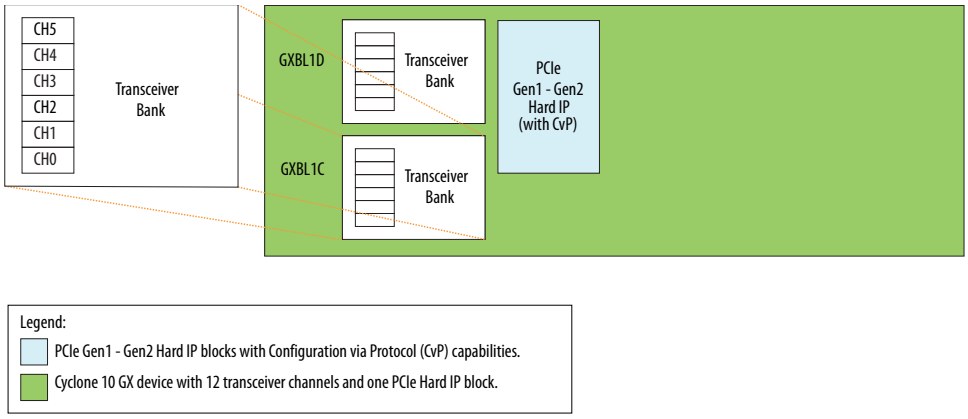


Figure 3. Intel Cyclone 10 GX Devices with 10 Transceiver Channels and One PCIe Hard IP Block

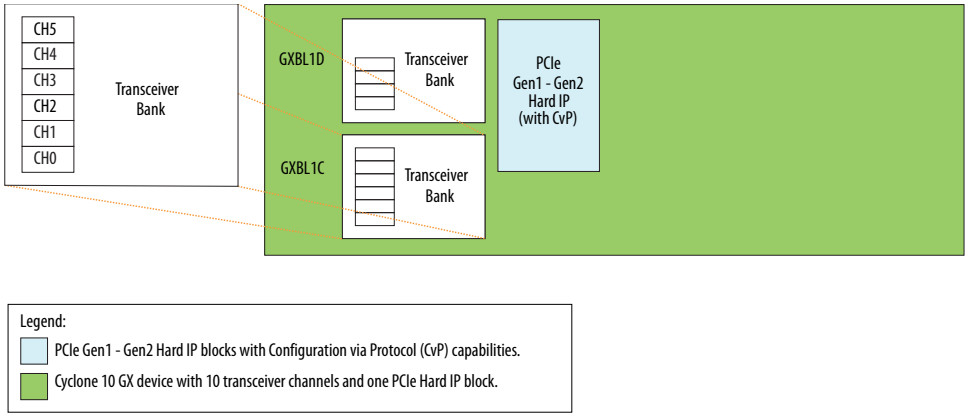
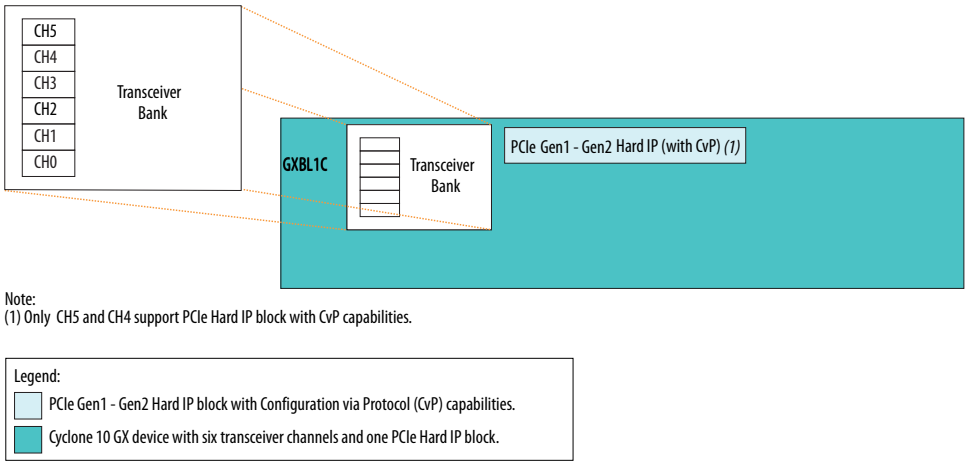


Figure 4. Intel Cyclone 10 GX Devices with 6 Transceiver Channels and One PCIe Hard IP Block



1.1.2. Intel Cyclone 10 GX Device Package Details

The following tables list package sizes, available transceiver channels, and PCI Express Hard IP blocks for Intel Cyclone 10 GX devices.

Table 1. Package Details for Devices with Transceivers and Hard IP Blocks Located on the Left Side Periphery of the Device

- Package U484: 19mm x 19mm package; 484 pins.
- Package F672: 27mm x 27mm package; 672 pins.
- Package F780: 29mm x 29mm package; 780 pins.

Device	U484	F672	F780
	Transceiver Count, PCIe Hard IP Block Count		
10CX085	6, 1	6, 1	N/A
10CX105	6, 1	10, 1	12, 1
10CX150	6, 1	10, 1	12, 1
10CX220	6, 1	10, 1	12, 1

1.2. Transceiver PHY Architecture Overview

A link is defined as a single entity communication port. A link can have one or more transceiver channels. A transceiver channel is synonymous with a transceiver lane.

For example, a 10GBASE-R link has one transceiver channel or lane with a data rate of 10.3125 Gbps. A 40GBASE-R link has four transceiver channels. Each transceiver channel operates at a lane data rate of 10.3125 Gbps. Four transceiver channels give a total collective link bandwidth of 41.25 Gbps (40 Gbps before and after 64B/66B Physical Coding Sublayer (PCS) encoding and decoding).

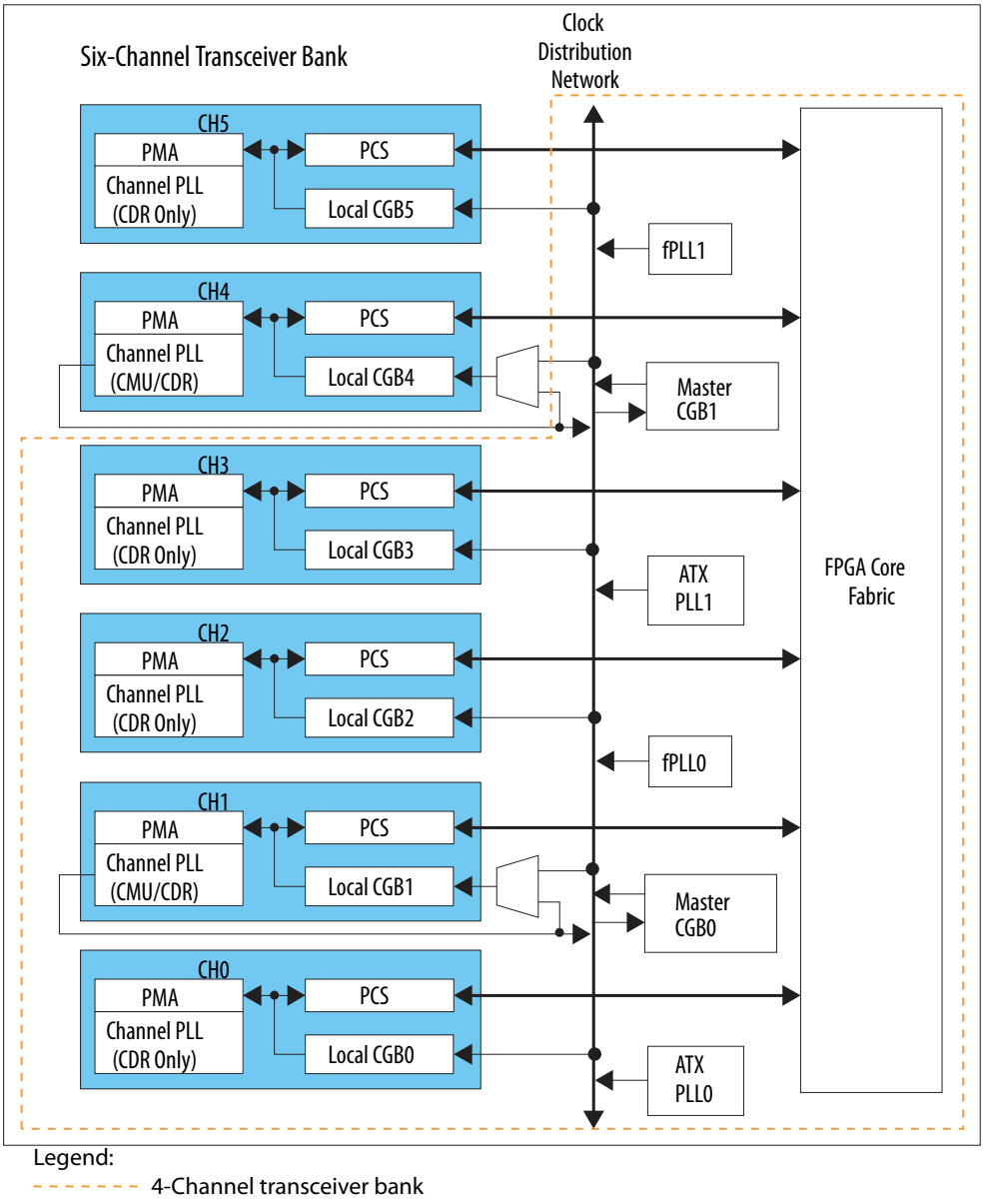
1.2.1. Transceiver Bank Architecture

The transceiver bank is the fundamental unit that contains all the functional blocks related to the device's high speed serial transceivers.

Each transceiver bank includes four or six transceiver channels in all devices.

The figures below show the transceiver bank architecture with the phase locked loop (PLL) and clock generation block (CGB) resources available in each bank.

Figure 5. Transceiver Bank Architecture



Note: This figure is a high level overview of the transceiver bank architecture. For details about the available clock networks refer to the *PLLs and Clock Networks* chapter.

Related Information

PLLs and Clock Networks on page 198

1.2.2. PHY Layer Transceiver Components

Transceivers in Intel Cyclone 10 GX devices support both Physical Medium Attachment (PMA) and Physical Coding Sublayer (PCS) functions at the physical (PHY) layer.

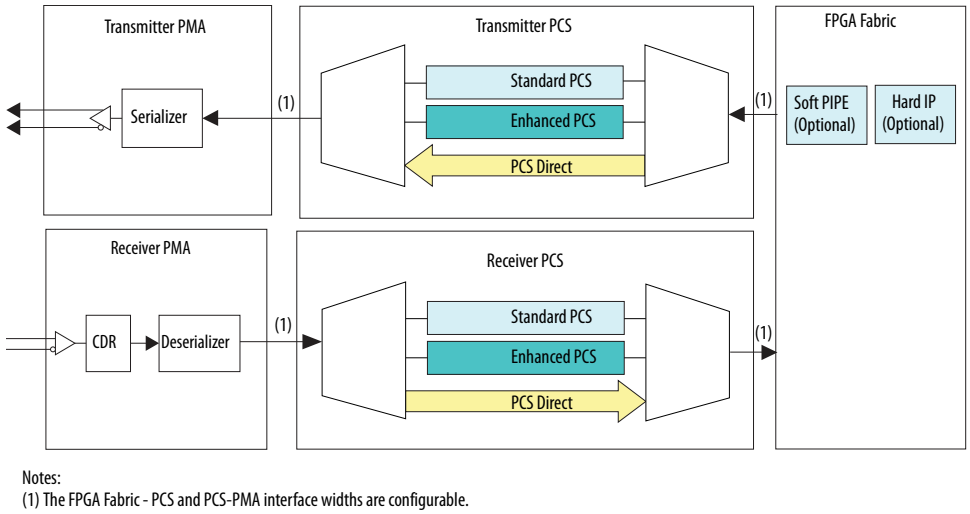
A PMA is the transceiver's electrical interface to the physical medium. The transceiver PMA consists of standard blocks such as:

- serializer/deserializer (SERDES)
- clock and data recovery PLL
- analog front end transmit drivers
- analog front end receive buffers

The PCS can be bypassed with a PCS Direct configuration. Both the PMA and PCS blocks are fed by multiple clock networks driven by high performance PLLs. In PCS Direct configuration, the data flow is through the PCS block, but all the internal PCS blocks are bypassed. In this mode, the PCS functionality is implemented in the FPGA fabric.

1.2.2.1. The Transceiver Channel

Figure 6. Transceiver Channel in Full Duplex Mode



Notes:
(1) The FPGA Fabric - PCS and PCS-PMA interface widths are configurable.

Intel Cyclone 10 GX transceiver channels have three types of PCS blocks that together support continuous data rates between 1.0 Gbps and 10.81344 Gbps.

Table 2. PCS Types Supported by Transceiver Channels

PCS Type	Data Rate
Standard PCS	1.0 Gbps to 10.81344 Gbps
Enhanced PCS	1.0 Gbps to 12.5 Gbps
PCS Direct	1.0 Gbps to 12.5 Gbps

Note: The minimum operational data rate is 1.0 Gbps for both the transmitter and receiver. For transmitter data rates less than 1.0 Gbps, oversampling must be applied at the transmitter. For receiver data rates less than 1.0 Gbps, oversampling must be applied at the receiver.

1.2.3. Transceiver Phase-Locked Loops

Each transceiver channel in Intel Cyclone 10 GX devices has direct access to three types of high performance PLLs:

- Advanced Transmit (ATX) PLL
- Fractional PLL (fPLL)
- Channel PLL / Clock Multiplier Unit (CMU) PLL

These transceiver PLLs along with the Master or Local Clock Generation Blocks (CGB) drive the transceiver channels.

Related Information

PLLs on page 200

1.2.3.1. Advanced Transmit (ATX) PLL

An advanced transmit (ATX) PLL is a high performance PLL that only supports integer frequency synthesis. The ATX PLL is the transceiver channel's primary transmit PLL. It can operate over the full range of supported data rates required for high data rate applications.

Related Information

ATX PLL on page 201

1.2.3.2. Fractional PLL (fPLL)

A fractional PLL (fPLL) is an alternate transmit PLL that generates clock frequencies for up to 12.5 Gbps data rate applications. fPLLs support both integer frequency synthesis and fine resolution fractional frequency synthesis. Unlike the ATX PLL, the fPLL can also be used to synthesize frequencies that can drive the core through the FPGA fabric clock networks.

Related Information

fPLL on page 203

1.2.3.3. Channel PLL (CMU/CDR PLL)

A channel PLL resides locally within each transceiver channel. Its primary function is clock and data recovery in the transceiver channel when the PLL is used in clock data recovery (CDR) mode. The channel PLLs of channel 1 and 4 can be used as transmit PLLs when configured in clock multiplier unit (CMU) mode. The channel PLLs of channel 0, 2, 3, and 5 cannot be configured in CMU mode and therefore cannot be used as transmit PLLs.

Related Information

CMU PLL on page 206

1.2.4. Clock Generation Block (CGB)

In Intel Cyclone 10 GX devices, there are two types of clock generation blocks (CGBs):

- Master CGB
- Local CGB

Transceiver banks with six transceiver channels have two master CGBs. Master CGB1 is located at the top of the transceiver bank and master CGB0 is located at the bottom of the transceiver bank. The master CGB divides and distributes bonded clocks to a bonded channel group. It also distributes non-bonded clocks to non-bonded channels across the x6/xN clock network.

Each transceiver channel has a local CGB. The local CGB is used for dividing and distributing non-bonded clocks to its own PCS and PMA blocks.

Related Information

Clock Generation Block on page 216

1.3. Calibration

Intel Cyclone 10 GX FPGAs contain a dedicated calibration engine to compensate for process variations. The calibration engine calibrates the analog portion of the transceiver to allow both the transmitter and receiver to operate at optimum performance.

The CLKUSR pin clocks the calibration engine. All transceiver reference clocks and the CLKUSR clock must be free running and stable at the start of FPGA configuration to successfully complete the calibration process and for optimal transceiver performance.

Note: For more information about CLKUSR electrical characteristics, refer to *Intel Cyclone 10 GX Device Datasheet*. The CLKUSR can also be used as an FPGA configuration clock. For information about configuration requirements for the CLKUSR pin, refer to the *Configuration, Design Security, and Remote System Upgrades in Intel Cyclone 10 GX Devices* chapter in the *Intel Cyclone 10 GX Core Fabric and General-Purpose I/O Handbook*. For more information about calibration, refer to the *Calibration* chapter. For more information about CLKUSR pin requirements, refer to the *Intel Cyclone 10 GX Device Family Pin Connection Guidelines*.

Related Information

- Calibration on page 373
- Intel Cyclone 10 GX Device Datasheet
- Configuration, Design Security, and Remote System Upgrades in Intel Cyclone 10 GX Devices
- Intel Cyclone 10 GX Device Family Pin Connection Guidelines

1.4. Intel Cyclone 10 GX Transceiver PHY Overview Revision History

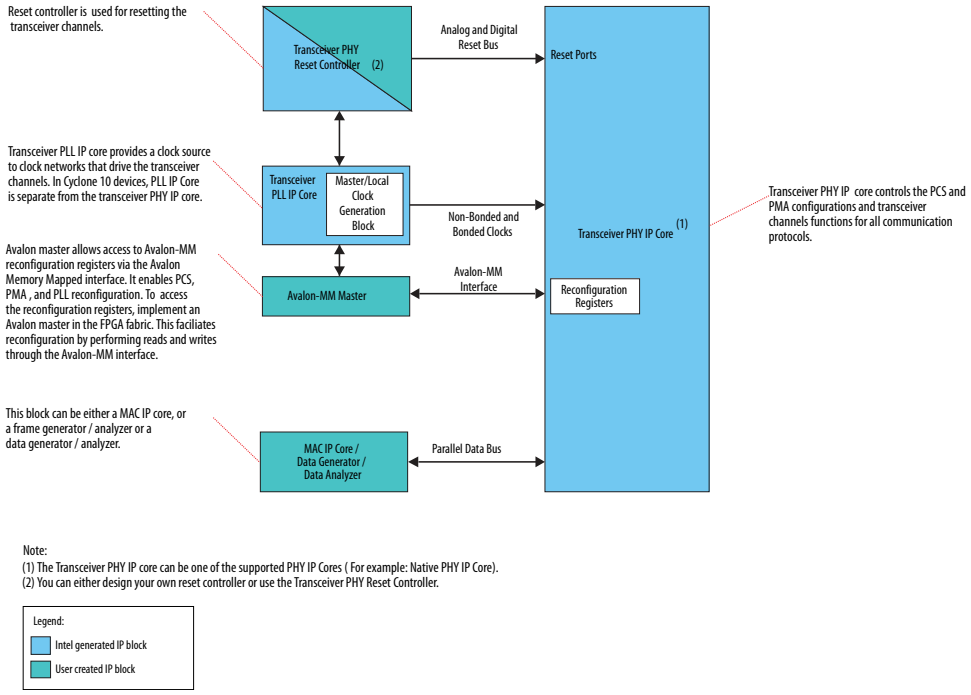
Document Version	Changes
2017.12.28	Made the following changes: • Updated the "Intel Cyclone 10 GX Default Settings Preset" Figure. • Changed the transceiver count back to 6 for 10CX085 package F672 in the "Package Details for Devices with Transceivers and Hard IP Blocks Located on the Left Side Periphery of the Device" table.
2017.11.06	Made the following changes: • Changed the description of the ATX PLL in the "Advanced Transmit (ATX) PLL" section. • Changed the transceiver counts for the F672 package in the "Package Details for Devices with Transceivers and Hard IP Blocks Located on the Left Side Periphery of the Device" table. • Changed the description of the Fractional PLL in the "Fractional PLL (fPLL)" section. • Changed the location of the PCIe Hard IP block in the " Cyclone 10 GX Devices with 12 Transceiver Channels and One PCIe Hard IP Block" figure. • Changed the location of the PCIe Hard IP block in the " Cyclone 10 GX Devices with 10 Transceiver Channels and One PCIe Hard IP Block" figure. • Changed the location of the PCIe Hard IP block in the " Cyclone 10 GX Devices with 6 Transceiver Channels and One PCIe Hard IP Block" figure.
2017.05.08	Initial release.

2. Implementing Protocols in Intel Cyclone 10 GX Transceivers

2.1. Transceiver Design IP Blocks

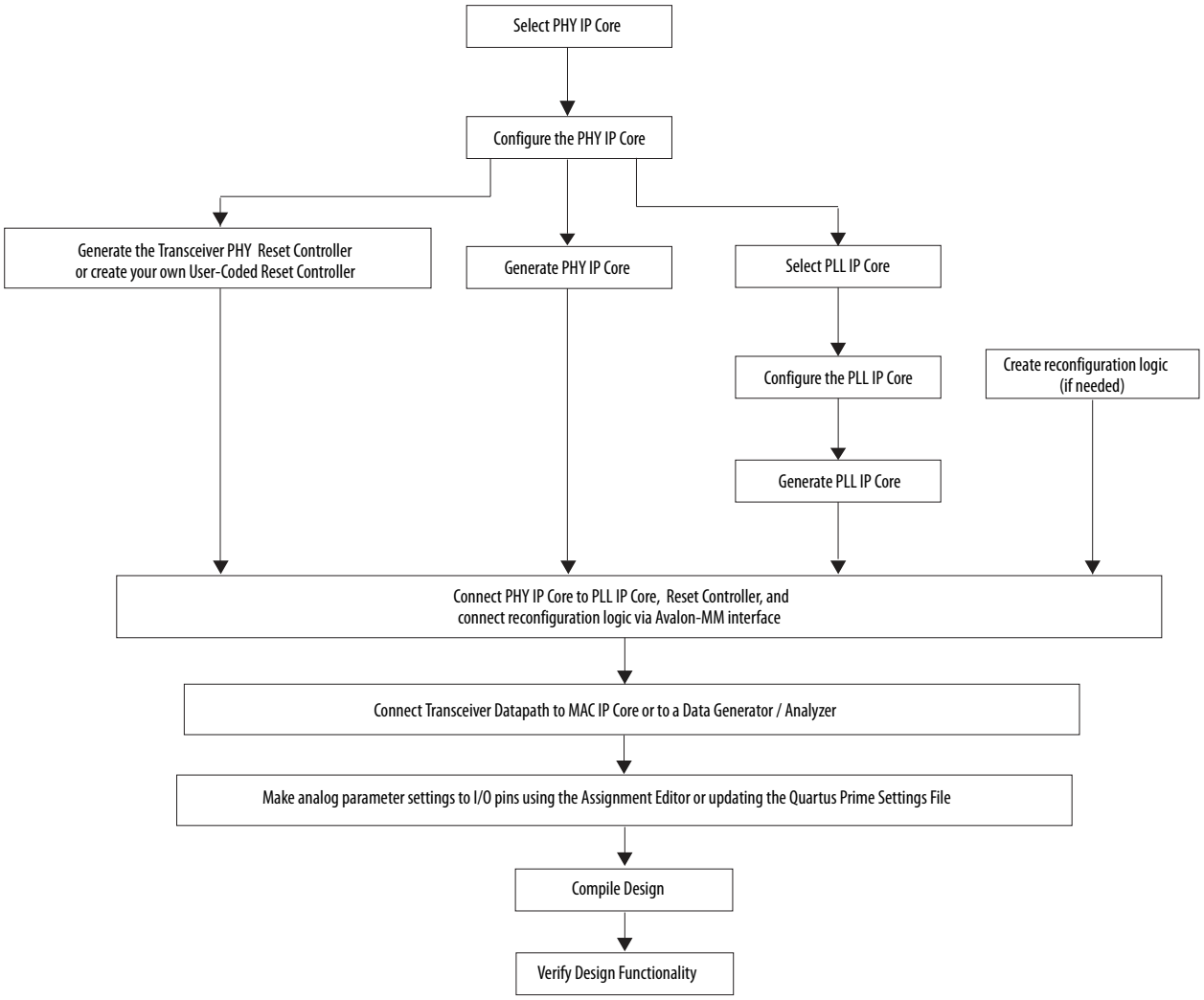
Note: Intel Cyclone 10 GX only supported with Intel Quartus Prime Pro Edition 17.1 and future versions.

Figure 7. Cyclone 10 GX Transceiver Design Fundamental Building Blocks



2.2. Transceiver Design Flow

Figure 8. Transceiver Design Flow



2.2.1. Select and Instantiate the PHY IP Core

Select the appropriate PHY IP core to implement your protocol.

Refer to the *Cyclone 10 GX Transceiver Protocols and PHY IP Support* section to decide which PHY IP to select to implement your protocol.

You can create your Quartus Prime project first, and then instantiate the various IPs required for your design. In this case, specify the location to save your IP HDL files. The current version of the PHY IP does not have the option to set the speed grade. Specify the device family and speed grade when you create the Quartus Prime project.

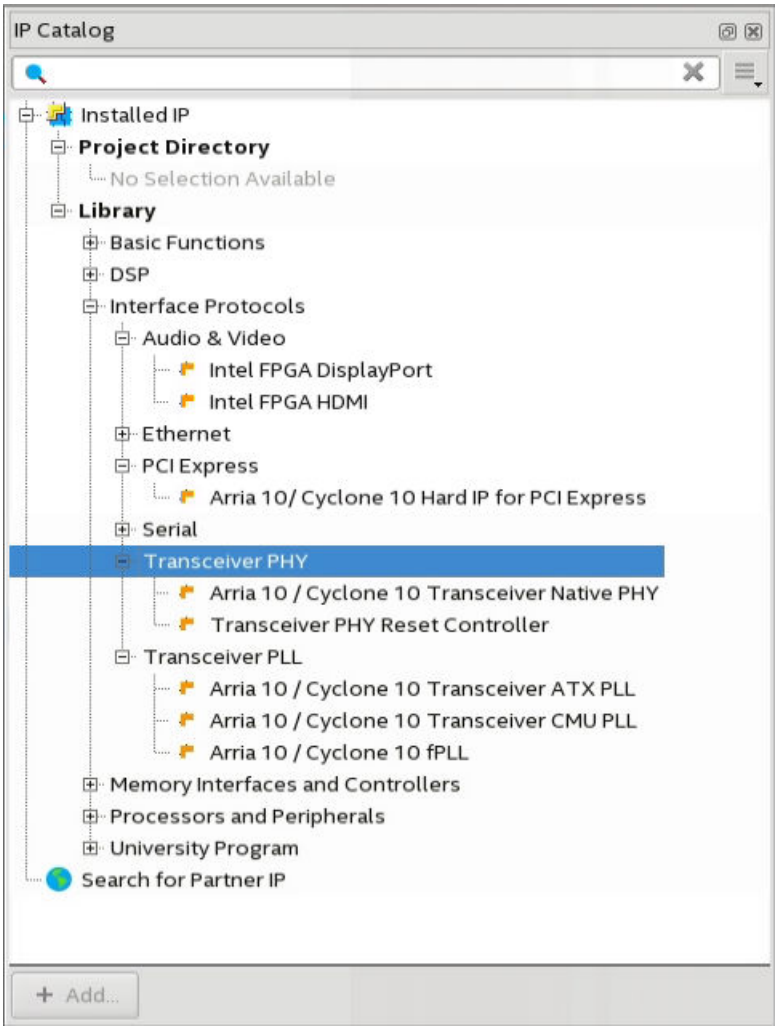
You can also instantiate the PHY IP directly to evaluate the various features.

To instantiate a PHY IP:

1. Open the Quartus Prime software.
2. Click **Tools ➤ IP Catalog**.
3. At the top of the **IP Catalog** window, select **Cyclone 10 GX** device family
4. In **IP Catalog**, under **Library ➤ Interface Protocols**, select the appropriate PHY IP and then click **Add**.
5. In the **New IP Instance Dialog Box**, provide the IP instance name.
6. Select **Cyclone 10 GX** device family.
7. Select the appropriate device and click **OK**.

The PHY IP Parameter Editor window opens.

Figure 9. Cyclone 10 GX Transceiver PHY Types



2.2.2. Configure the PHY IP Core

Configure the PHY IP core by selecting the valid parameters for your design. The valid parameter settings are different for each protocol. Refer to the appropriate protocol's section for selecting valid parameters for each protocol.

2.2.3. Generate the PHY IP Core

After configuring the PHY IP, complete the following steps to generate the PHY IP.

1. Click the **Generate HDL** button in the **Parameter Editor** window. The **Generation** dialog box opens.
2. In **Synthesis** options, under **Create HDL design for synthesis** select `Verilog` or `VHDL`.
3. Select appropriate **Simulation** options depending on the choice of the hardware description language you selected under **Synthesis** options.
4. In **Output Directory**, select **Clear output directories for selected generation targets** if you want to clear any previous IP generation files from the selected output directory.
5. Click **Generate**.

The Quartus Prime software generates a *<phy ip instance name>* folder, *<phy ip instance name>*_sim folder, *<phy ip instance name>*.qip file, *<phy ip instance name>*.qsys file, and *<phy ip instance name>*.v file or *<phy ip instance name>*.vhd file. This *<phy ip instance name>*.v file is the top level design file for the PHY IP and is placed in the *<phy ip instance name>*/synth folder. The other folders contain lower level design files used for simulation and compilation.

Related Information

IP Core File Locations on page 67
For more information about IP core file structure

2.2.4. Select the PLL IP Core

Cyclone 10 GX devices have three types of PLL IP cores:

- Advanced Transmit (ATX) PLL IP core.
- Fractional PLL (fPLL) IP core.
- Channel PLL / Clock Multiplier Unit (CMU) PLL IP core.

Select the appropriate PLL IP for your design. For additional details, refer to the *PLLs and Clock Networks* chapter.

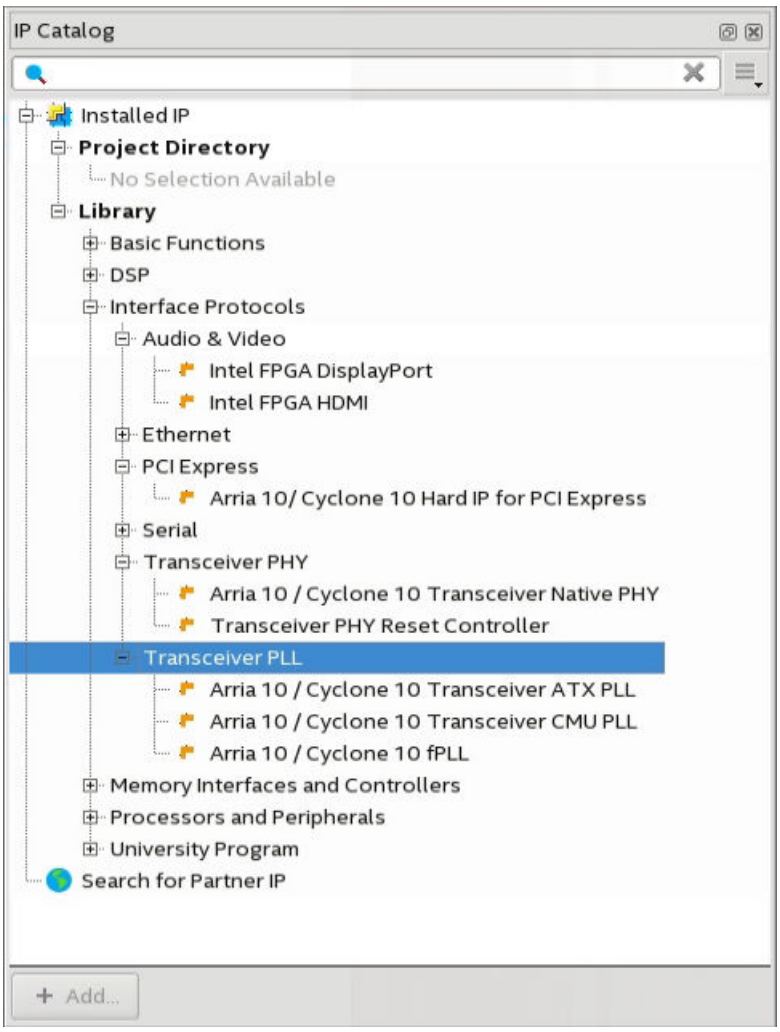
To instantiate a PLL IP:

1. Open the Quartus Prime software.
2. Click **Tools ➤ IP Catalog.**
3. At the top of the **IP Catalog** window, select **Cyclone 10 GX** device family
4. In **IP Catalog**, under **Library ➤ Basic Functions ➤ Clocks, PLLs, and Resets ➤ PLL** choose the PLL IP (**Cyclone 10 GX fPLL**, **Cyclone 10 GX Transceiver ATX PLL**, or **Cyclone 10 GX Transceiver CMU PLL**) you want to include in your design and then click **Add**.

5. In the **New IP Instance Dialog Box**, provide the IP instance name.
6. Select **Cyclone 10 GX** device family.
7. Select the appropriate device and click **OK**.

The PLL IP GUI window opens.

Figure 10. Cyclone 10 GX Transceiver PLL Types



2.2.5. Configure the PLL IP Core

Understand the available PLLs, clock networks, and the supported clocking configurations. Configure the PLL IP to achieve the adequate data rate for your design.

2.2.6. Generate the PLL IP Core

After configuring the PLL IP core, complete the following steps to generate the PLL IP core.

1. Click the **Generate HDL** button in the **Parameter Editor** window. The **Generation** dialog box opens.
2. In **Synthesis** options, under **Create HDL design for synthesis** select `Verilog` or VHDL.
3. Select appropriate **Simulation** options depending on the choice of the hardware description language you selected under **Synthesis** options.
4. In **Output Directory**, select **Clear output directories for selected generation targets** if you want to clear any previous IP generation files from the selected output directory.
5. Click **Generate**.

The Quartus ® Prime software generates a *<pll ip core instance name>* folder, *<pll ip core instance name>*_sim folder, *<pll ip core instance name>*.qip file, *<pll ip core instance name>*.qsys, and *<pll ip core instance name>*.v file or *<pll ip core instance name>*.vhd file. The *<pll ip core instance name>*.v file is the top level design file for the PLL IP core and is placed in the *<pll ip core instance name>*/ synth folder. The other folders contain lower level design files used for simulation and compilation.

Related Information

IP Core File Locations on page 67
For more information about IP core file structure

2.2.7. Reset Controller

There are two methods to reset the transceivers in Cyclone 10 GX devices:

- Use the Transceiver PHY Reset Controller.
- Create your own reset controller that follows the recommended reset sequence.

Related Information

Resetting Transceiver Channels on page 243

2.2.8. Create Reconfiguration Logic

Dynamic reconfiguration is the ability to dynamically modify the transceiver channels and PLL settings during device operation. To support dynamic reconfiguration, your design must include an Avalon® memory-mapped interface.

The Avalon memory-mapped interface master enables PLL and channel reconfiguration. You can dynamically adjust the PMA parameters, such as differential output voltage swing (Vod), and pre-emphasis settings. This adjustment can be done by writing to the Avalon memory-mapped interface reconfiguration registers through the user generated Avalon memory-mapped interface master.

For detailed information on dynamic reconfiguration, refer to *Reconfiguration Interface and Dynamic Reconfiguration* chapter.

2.2.9. Connect the PHY IP to the PLL IP Core and Reset Controller

Connect the PHY IP, PLL IP core, and the reset controller. Write the top level module to connect all the IP blocks.

All of the I/O ports for each IP, can be seen in the <*phy instance name*>.v file or <*phy instance name*>.vhd, and in the <*phy_instance_name*>_bb.v file.

For more information about description of the ports, refer to the ports tables in the *PLLs*, *Using the Transceiver Native PHY IP Core*, and *Resetting Transceiver Channels* chapters.

Related Information

- Resetting Transceiver Channels on page 243
- Using the Cyclone 10 GX Transceiver Native PHY IP Core on page 26
- PLLs and Clock Networks on page 198

2.2.10. Connect Datapath

Connect the transceiver PHY layer design to the Media Access Controller (MAC) IP core or to a data generator / analyzer or a frame generator / analyzer.

2.2.11. Make Analog Parameter Settings

Make analog parameter settings to I/O pins using the **Assignment Editor** or updating the Quartus Prime Settings File.

After verifying your design functionality, make pin assignments and PMA analog parameter settings for the transceiver pins.

1. Assign FPGA pins to all the transceiver and reference clock I/O pins.
2. Set the analog parameters to the transmitter, receiver, and reference clock pins using the **Assignment Editor**.
   All of the pin assignments and analog parameters set using the **Pin Planner** and the **Assignment Editor** are saved in the <top_level_project_name>.qsf file. You can also directly modify the Quartus Settings File (.qsf) to set PMA analog parameters.

2.2.12. Compile the Design

To compile the transceiver design, add the <*phy_instancename*>.qip files for all the IP blocks generated using the IP Catalog to the Quartus Prime project library. You can alternatively add the **.qsys** and **.qip** variants of the IP cores.

Note: If you add both the **.qsys** and the **.qip** file into the Quartus Prime project, the software generates an error.

2.2.13. Verify Design Functionality

Simulate your design to verify the functionality of your design. For more details, refer to *Simulating the Native Transceiver PHY IP Core* section.

Related Information

Intel Quartus Prime Pro Edition Handbook Volume 3: Verification
Information about design simulation and verification.

2.3. Cyclone 10 GX Transceiver Protocols and PHY IP Support

Table 3. Cyclone 10 GX Transceiver Protocols and PHY IP Support

Protocol	Transceiver PHY IP Core	PCS Support	Transceiver Configuration Rule	Protocol Preset
PCIe Gen2 x1, x2, x4	Native PHY IP (PIPE) core/Hard IP for PCI Express (1)	Standard	Gen2 PIPE	PCIe PIPE Gen2 x1 (2)
PCIe Gen1 x1, x2, x4	Native PHY IP (PIPE) core/Hard IP for PCI Express (1)	Standard	Gen1 PIPE	User created(3)
1000BASE-X Gigabit Ethernet	Native PHY IP core	Standard	GbE	GIGE - 1.25 Gbps
1000BASE-X Gigabit Ethernet with 1588	Native PHY IP core	Standard	GbE 1588	GIGE - 1.25 Gbps 1588
10GBASE-R	Native PHY IP core	Enhanced	10GBASE-R	10GBASE-R Low Latency
10GBASE-R 1588	Native PHY IP core	Enhanced	10GBASE-R 1588	10GBASE-R(4)
40GBASE-R	Native PHY IP core	Enhanced	Basic (Enhanced PCS)	Low Latency Enhanced PCS (5)
Interlaken (CEI-6G-SR and CEI-11G-SR) (6)	Native PHY IP core	Enhanced	Interlaken	Interlaken 10x12.5Gbps Interlaken 6x10.3Gbps Interlaken 1x6.25Gbps
OTU-1 (2.7G)	Native PHY IP core	Standard	Basic/Custom (Standard PCS)	User created

continued...

(1) Hard IP for PCI Express is also available as a separate IP core.

(2) For x2 and x4 modes, select PCIe PIPE Gen2 x8. Then change the number of data channels from 8 to 4.

(3) For PCIe Gen1 x1 mode, select PCIe PIPE Gen2 x1 mode. Then change the transceiver configuration rule from Gen 2 PIPE to Gen 1 PIPE.

For PCIe Gen1 x2 and x4 mode, select PCIe PIPE Gen2 x8. Then change the transceiver configuration rule from Gen2 PIPE to Gen1 PIPE and number of data channels from 8 to 2 or 4.

(4) Select the 10GBASE-R preset. Then change the transceiver configuration rule from 10GBASE-R to 10GBASE-R 1588.

(5) To implement 40GBASE-R using the Low Latency Enhanced PCS preset, change the number of data channels to four and select appropriate PCS- FPGA Fabric and PCS-PMA width.

(6) Link training, auto speed negotiation and sequencer functions are not included in the Native PHY IP. The user would have to create soft logic to implement these functions when using Native PHY IP.

A Transmit PCS soft bonding logic required for multi-lane bonding configuration is provided in the design example.

Protocol	Transceiver PHY IP Core	PCS Support	Transceiver Configuration Rule	Protocol Preset
SONET/SDH STS-192/ STM-64 (10G) via SFP +/SFF-8431/CEI-11G	Native PHY IP core	Enhanced	Basic (Enhanced PCS)	User created
SONET/SDH STS-192/ STM-64 (10G) via OIF SFI-5.1s/SxI-5/ SFI-4.2	Native PHY IP core	Enhanced	Basic (Enhanced PCS)	User created
SONET STS-96 (5G) via OIF SFI-5.1s	Native PHY IP core	Enhanced	Basic/Custom (Standard PCS)	SONET/SDH OC-96
SONET/SDH STS-48/ STM-16 (2.5G) via SFP/TFI-5.1	Native PHY IP core	Standard	Basic/Custom (Standard PCS)	SONET/SDH OC-48
SONET/SDH STS-12/ STM-4 (0.622G) via SFP/TFI-5.1	Native PHY IP core(7)	Standard	Basic/Custom (Standard PCS)	SONET/SDH OC-12
SD-SDI/HD-SDI/3G/6G/12G-SDI	Native PHY IP core	Standard	Basic/Custom (Standard PCS)	HD/3G SDI NTSC/PAL SDI multi-rate (up to 12G) RX/TX SDI triple-rate RX
Vx1	Native PHY IP core	Standard	Basic/Custom (Standard PCS)	User created
DisplayPort	Native PHY IP core	Standard	Basic/Custom (Standard PCS)	DisplayPort Duplex 4 SYMBOLS PER CLOCK DisplayPort RX 4 SYMBOLS PER CLOCK DisplayPort TX 4 SYMBOLS PER CLOCK
1.25G/ 2.5G 10G GPON/EPON	Native PHY IP core	Enhanced	Basic (Enhanced PCS)	User created
2.5G/1.25G GPON/ EPON	Native PHY IP core	Standard	Basic/Custom (Standard PCS)	User created
8G/4G/2G/1G Fibre Channel	Native PHY IP core	Standard	Basic/Custom (Standard PCS)	User created
SDR/DDR Infiniband x1, x4, x12	Native PHY IP core	Standard	Basic/Custom (Standard PCS)	User created
SRIO 2.2/1.3	Native PHY IP core	Standard	Basic/Custom with Rate Match(Standard PCS)	Serial Rapid IO 1.25 Gbps
CPRI 4.1/OBSAI RP3 v4.1	Native PHY IP core	Standard	CPRI (Auto)/CPRI (Manual)	User created (8)
SAS 3.0	Native PHY IP core	Enhanced	Basic (Enhanced PCS)	User created

continued...

(7) The minimum operational data rate is 1.0 Gbps for both the transmitter and receiver. If transmitter data rates less than 1.0 Gbps, oversampling must be applied at the transmitter. If receiver data rates less than 1.0 Gbps, oversampling must be applied at the receiver.

(8) Select CPRI 9.8 Gbps Auto/Manual Mode (Intel Arria® 10 only). Then change the datarate from 9830.4 Mbps to 6144 Mbps.

Protocol	Transceiver PHY IP Core	PCS Support	Transceiver Configuration Rule	Protocol Preset
SATA 3.0/2.0/1.0 and SAS 2.0/1.1/1.0	Native PHY IP core	Standard	Basic/Custom (Standard PCS)	SAS Gen2/Gen1.1/ Gen1 SATA Gen3/Gen2/ Gen1
HiGig/HiGig+/HiGig2/ HiGig2+	Native PHY IP core	Standard	Basic/Custom (Standard PCS)	User created
JESD204A / JESD204B	Native PHY IP core	Standard and Enhanced	Basic/Custom (Standard PCS) Basic (Enhanced PCS)	User created
Custom and other protocols	Native PHY IP core	Standard and Enhanced PCS Direct	Basis/Custom (Standard PCS) Basic (Enhanced PCS) Basic/Custom with Rate Match (Standard PCS) PCS Direct	User created

2.4. Using the Cyclone 10 GX Transceiver Native PHY IP Core

This section describes the use of the Intel-provided Cyclone 10 GX Transceiver Native PHY IP core. This Native PHY IP core provides direct access to Cyclone 10 GX transceiver PHY features.

Use the Native PHY IP core to configure the transceiver PHY for your protocol implementation. To instantiate the IP, click **Tools ➤ IP Catalog** to select your IP core variation. Use the **Parameter Editor** to specify the IP parameters and configure the PHY IP for your protocol implementation. To quickly configure the PHY IP, select a preset that matches your protocol configuration as a starting point. Presets are PHY IP configuration settings for various protocols that are stored in the IP **Parameter Editor**. Presets are explained in detail in the *Presets* section below.

You can also configure the PHY IP by selecting an appropriate **Transceiver Configuration Rule**. The transceiver configuration rules check the valid combinations of the PCS and PMA blocks in the transceiver PHY layer, and report errors or warnings for any invalid settings.

Use the Native PHY IP core to instantiate the following PCS options:

- Standard PCS
- Enhanced PCS
- PCS Direct

Based on the Transceiver Configuration Rule that you select, the PHY IP core selects the appropriate PCS. The PHY IP core allows you to select all the PCS blocks if you intend to dynamically reconfigure from one PCS to another. Refer to *General and Datapath Parameters* section for more details on how to enable PCS blocks for dynamic reconfiguration.

After you configure the PHY IP core in the **Parameter Editor**, click **Generate HDL** to generate the IP instance. The top level file generated with the IP instance includes all the available ports for your configuration. Use these ports to connect the PHY IP core to the PLL IP core, the reset controller IP core, and to other IP cores in your design.

Figure 11. Native PHY IP Core Ports and Functional Blocks

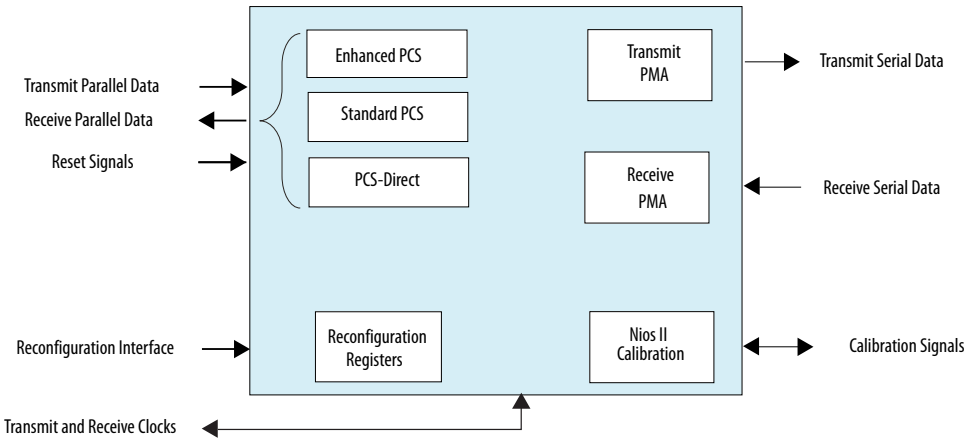
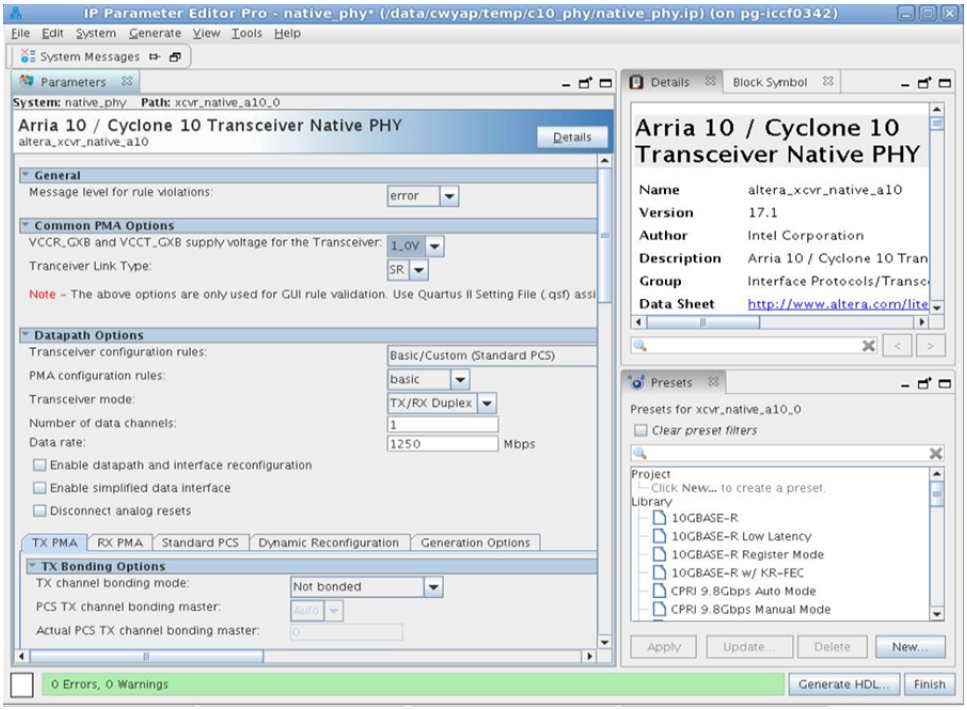


Figure 12. Native PHY IP Core Parameter Editor



Note: Although the Quartus Prime software provides legality checks, refer to the *High-Speed Serial Transceiver-Fabric Interface Performance for Intel Cyclone 10 GX Devices* section of the*Intel Cyclone 10 GX Device Datasheet* for the supported FPGA fabric to PCS interface widths and frequency.

Related Information

- Configure the PHY IP Core on page 19
- Interlaken on page 70
- Gigabit Ethernet (GbE) and GbE with IEEE 1588v2 on page 87

- 10GBASE-R and 10GBASE-R with IEEE 1588v2 Variants on page 98
- PCI Express (PIPE) on page 122
- CPRI on page 149
- Using the "Basic (Enhanced PCS)" Configuration on page 158
- Using the Basic/Custom, Basic/Custom with Rate Match Configurations of Standard PCS on page 166
- PMA Parameters on page 31
- Presets on page 28
- General and Datapath Parameters on page 28
- PMA Ports on page 50
- Enhanced PCS Ports on page 53
- Standard PCS Ports on page 62
- How to Place Channels for PIPE Configurations on page 146
- Intel Cyclone 10 GX Device Datasheet

2.4.1. Presets

You can select preset settings for the Native PHY IP core defined for each protocol. Use presets as a starting point to specify parameters for your specific protocol or application.

To apply a preset to the Native PHY IP core, double-click on the preset name. When you apply a preset, all relevant options and parameters are set in the current instance of the Native PHY IP core. For example, selecting the **Interlaken** preset enables all parameters and ports that the Interlaken protocol requires.

Selecting a preset does not prevent you from changing any parameter to meet the requirements of your design. Any changes that you make are validated by the design rules for the transceiver configuration rules you specified, not the selected preset.

Note: Selecting a preset clears any prior selections user has made so far.

2.4.2. General and Datapath Parameters

You can customize your instance of the Native PHY IP core by specifying parameter values. In the **Parameter Editor**, the parameters are organized in the following sections for each functional block and feature:

- General, Common PMA Options, and Datapath Options
- TX PMA
- RX PMA
- Standard PCS
- Enhanced PCS
- PCS Direct Datapath
- Dynamic Reconfiguration
- Analog PMA Settings (Optional)
- Generation Options

Table 4. General, Common PMA Options, and Datapath Options

Parameter	Value	Description
Message level for rule violations	**error** **warning**	Specifies the messaging level for parameter rule violations. Selecting **error** causes all rule violations to prevent IP generation. Selecting **warning** displays all rule violations as warnings in the message window and allows IP generation despite the violations.
VCCR_GXB and VCCT_GXB supply voltage for the Transceiver	0_9V, 1_0V	Selects the VCCR_GXB and VCCT_GXB supply voltage for the Transceiver. *Note:* This option is only used for GUI rule validation. Use Quartus Prime Setting File (.qsf) assignments to set this parameter in your static design.
Transceiver Link Type	sr, lr	Selects the type of transceiver link. sr-Short Reach (Chip-to-chip communication), lr-Long Reach (Backplane communication). *Note:* This option is only used for GUI rule validation. Use Quartus Prime Setting File (.qsf) assignments to set this parameter in your static design.
Transceiver configuration rules	User Selection	Specifies the valid configuration rules for the transceiver. This parameter specifies the configuration rule against which the **Parameter Editor** checks your PMA and PCS parameter settings for specific protocols. Depending on the transceiver configuration rule selected, the **Parameter Editor** validates the parameters and options selected by you and generates error messages or warnings for all invalid settings. To determine the transceiver configuration rule to be selected for your protocol, refer to Table 3 on page 24 *Transceiver Configuration Rule Parameters* table for more details about each transceiver configuration rule. This parameter is used for rule checking and is not a preset. You need to set all parameters for your protocol implementation.
PMA configuration rules	**Basic** **SATA/SAS** **GPON**	Specifies the configuration rule for PMA. Select Basic for all other protocol modes except for SATA and GPON . SATA (Serial ATA) can be used only if the **Transceiver configuration rule** is set to **Basic/Custom (Standard PCS)**. GPON can be used only if the **Transceiver configuration rule** is set to **Basic (Enhanced PCS)**.
Transceiver mode	**TX/RX Duplex** **TX Simplex** **RX Simplex**	Specifies the operational mode of the transceiver. • **TX/RX Duplex** : Specifies a single channel that supports both transmission and reception. • **TX Simplex** : Specifies a single channel that supports only transmission. • **RX Simplex** : Specifies a single channel that supports only reception. The default is **TX/RX Duplex**.
Number of data channels	**1** – *<n>*	Specifies the number of transceiver channels to be implemented. The maximum number of channels available, (*<n>*), depends on the package you select. The default value is **1**.
Data rate	*< valid Transceiver data rate >*	Specifies the data rate in megabits per second (Mbps).
Enable datapath and interface reconfiguration	**On/Off**	When you turn this option on, you can preconfigure and dynamically switch between the Standard PCS, Enhanced PCS, and PCS direct datapaths.

continued...

Parameter	Value	Description
		The default value is **Off**.
Enable simplified data interface	**On/Off**	By default, all 128-bits are ports for the `tx_parallel_data` and `rx_parallel_data` buses are exposed. You must understand the mapping of data and control signals within the interface. Refer to the *Enhanced PCS TX and RX Control Ports* section for details about mapping of data and control signals. When you turn on this option, the Native PHY IP core presents a simplified data and control interface between the FPGA fabric and transceiver. Only the sub-set of the 128-bits that are active for a particular FPGA fabric width are ports. The default value is **Off**.(9)
Provide separate interface for each channel	**On/Off**	When selected the Native PHY IP core presents separate data, reset and clock interfaces for each channel rather than a wide bus.

Table 5. Transceiver Configuration Rule Parameters

Transceiver Configuration Setting	Description
Basic/Custom (Standard PCS)	Enforces a standard set of rules within the Standard PCS. Select these rules to implement custom protocols requiring blocks within the Standard PCS or protocols not covered by the other configuration rules.
Basic/Custom w /Rate Match (Standard PCS)	Enforces a standard set of rules including rules for the Rate Match FIFO within the Standard PCS. Select these rules to implement custom protocols requiring blocks within the Standard PCS or protocols not covered by the other configuration rules.
CPRI (Auto)	Enforces rules required by the CPRI protocol. The receiver word aligner mode is set to **Auto**. In **Auto** mode, the word aligner is set to deterministic latency.
CPRI (Manual)	Enforces rules required by the CPRI protocol. The receiver word aligner mode is set to **Manual**. In **Manual** mode, logic in the FPGA fabric controls the word aligner.
GbE	Enforces rules that the 1 Gbps Ethernet (1 GbE) protocol requires.
GbE 1588	Enforces rules for the 1 GbE protocol with support for Precision time protocol (PTP) as defined in the *IEEE 1588 Standard*.
Gen1 PIPE	Enforces rules for a Gen1 PCIe ® PIPE interface that you can connect to a soft MAC and Data Link Layer.
Gen2 PIPE	Enforces rules for a Gen2 PCIe PIPE interface that you can connect to a soft MAC and Data Link Layer.
Basic (Enhanced PCS)	Enforces a standard set of rules within the Enhanced PCS. Select these rules to implement protocols requiring blocks within the Enhanced PCS or protocols not covered by the other configuration rules.
Interlaken	Enforces rules required by the Interlaken protocol.
10GBASE-R	Enforces rules required by the 10GBASE-R protocol.
10GBASE-R 1588	Enforces rules required by the 10GBASE-R protocol with 1588 enabled.
PCS Direct	Enforces rules required by the PCS Direct mode. In this configuration the data flows through the PCS channel, but all the internal PCS blocks are bypassed. If required, the PCS functionality can be implemented in the FPGA fabric.

(9) This option cannot be used, if you intend to dynamically reconfigure between PCS datapaths, or reconfigure the interface of the transceiver.

2.4.3. PMA Parameters

You can specify values for the following types of PMA parameters:

TX PMA

- TX Bonding Options
- TX PLL Options
- TX PMA Optional Ports

RX PMA

- RX CDR Options
- Equalization
- RX PMA Optional Ports

Table 6. TX Bonding Options

Parameter	Value	Description
TX channel bonding mode	**Not bonded** **PMA only bonding** **PMA and PCS bonding**	Selects the bonding mode to be used for the channels specified. Bonded channels use a single TX PLL to generate a clock that drives multiple channels, reducing channel-to-channel skew. The following options are available: **Not bonded**: In a non-bonded configuration, only the high speed serial clock is expected to be connected from the TX PLL to the Native PHY IP core. The low speed parallel clock is generated by the local clock generation block (CGB) present in the transceiver channel. For non-bonded configurations, because the channels are not related to each other and the feedback path is local to the PLL, the skew between channels cannot be calculated. **PMA only bonding**: In PMA bonding, the high speed serial clock is routed from the transmitter PLL to the master CGB. The master CGB generates the high speed and low parallel clocks and the local CGB for each channel is bypassed. Refer to the *Channel Bonding* section for more details. **PMA and PCS bonding** : In a PMA and PCS bonded configuration, the local CGB in each channel is bypassed and the parallel clocks generated by the master CGB are used to clock the network. The master CGB generates both the high and low speed clocks. The master channel generates the PCS control signals and distributes to other channels through a control plane block. The default value is **Not bonded**. Refer to *Channel Bonding* section in *PLLs and Clock Networks* chapter for more details.
PCS TX channel bonding master	**Auto, 0 to <number of channels> -1**	Specifies the master PCS channel for PCS bonded configurations. Each Native PHY IP core instance configured with bonding must specify a bonding master. If you select **Auto**, the Native PHY IP core automatically selects a recommended channel. The default value is **Auto**. Refer to the *PLLs and Clock Networks* chapter for more information about the TX channel bonding master.
Actual PCS TX channel bonding master	**0 to <number of channels> -1**	This parameter is automatically populated based on your selection for the **PCS TX channel bonding master** parameter. Indicates the selected master PCS channel for PCS bonded configurations.

Table 7. TX PLL Options

Parameter	Value	Description
TX local clock division factor	**1, 2, 4, 8**	Specifies the value of the divider available in the transceiver channels to divide the TX PLL output clock to generate the correct frequencies for the parallel and serial clocks.
Number of TX PLL clock inputs per channel	**1, 2, 3 , 4**	Specifies the number of TX PLL clock inputs per channel. Use this parameter when you plan to dynamically switch between TX PLL clock sources. Up to four input sources are possible.
Initial TX PLL clock input selection	**0 to <number of TX PLL clock inputs> -1**	Specifies the initially selected TX PLL clock input. This parameter is necessary when you plan to switch between multiple TX PLL clock inputs.

Table 8. TX PMA Optional Ports

Parameter	Value	Description
Enable tx_pma_analog_reset_ack port	**On/Off**	Enables the optional `tx_pma_analog_reset_ack` output port. This port should not be used for register mode data transfers.
Enable tx_pma_clkout port	**On/Off**	Enables the optional `tx_pma_clkout` output clock. This is the low speed parallel clock from the TX PMA. The source of this clock is the serializer. It is driven by the PCS/PMA interface block. (10)
Enable tx_pma_div_clkout port	**On/Off**	Enables the optional `tx_pma_div_clkout` output clock. This clock is generated by the serializer. You can use this to drive core logic, to drive the FPGA - transceivers interface. If you select a **tx_pma_div_clkout division factor** of 1 or 2, this clock output is derived from the PMA parallel clock. If you select a **tx_pma_div_clkout division factor** of 33, 40, or 66, this clock is derived from the PMA high serial clock. This clock is commonly used when the interface to the TX FIFO runs at a different rate than the PMA parallel clock frequency, such as 66:40 applications.
tx_pma_div_clkout division factor	**Disabled**, **1**, **2**, **33**, **40**, **66**	Selects the division factor for the `tx_pma_div_clkout` output clock when enabled. (11)
Enable tx_pma_iqtxrx_clkout port	**On/Off**	Enables the optional `tx_pma_iqtxrx_clkout` output clock. This clock can be used to cascade the TX PMA output clock to the input of a PLL.
Enable tx_pma_elecidle port	**On/Off**	Enables the `tx_pma_elecidle` port. When you assert this port, the transmitter is forced into an electrical idle condition. This port has no effect when the transceiver is configured for PCI Express.
Enable rx_seriallpbken port	**On/Off**	Enables the optional `rx_seriallpbken` control input port. The assertion of this signal enables the TX to RX serial loopback path within the transceiver. This is an asynchronous input signal.

Table 9. RX CDR Options

Parameter	Value	Description
Number of CDR reference clocks	**1 - 5**	Specifies the number of CDR reference clocks. Up to 5 sources are possible. The default value is 1.

continued...

(10) This clock should not be used to clock the FPGA - transceivers interface. This clock may be used as a reference clock to an external clock cleaner.

(11) The default value is **Disabled**.

Parameter	Value	Description
		Use this feature when you want to dynamically re-configure CDR reference clock source.
Selected CDR reference clock	**0 to <number of CDR reference clocks> -1**	Specifies the initial CDR reference clock. This parameter determines the available CDR references used. The default value is 0.
Selected CDR reference clock frequency	*< data rate dependent >*	Specifies the CDR reference clock frequency. This value depends on the data rate specified.
PPM detector threshold	**100** **300** **500** **1000**	Specifies the PPM threshold for the CDR. If the PPM between the incoming serial data and the CDR reference clock, exceeds this threshold value, the CDR loses lock. The default value is 1000.

Table 10. Equalization

Parameters	Value	Description
CTLE adaptation mode	**Manual**	Specifies the Continuous Time Linear Equalization (CTLE) operation mode. For manual mode, set the CTLE options through the Assignment Editor, or modify the Quartus Settings File (.qsf), or write to the reconfiguration registers using the Avalon Memory-Mapped (Avalon-MM) interface. Refer to the *Continuous Time Linear Equalization (CTLE)* section for more details about CTLE architecture. Refer to the *How to Enable CTLE* section for more details on supported adaptation modes.

Table 11. RX PMA Optional Ports

Parameters	Value	Description
Enable rx_analog_reset_ack port	**On/Off**	Enables the optional `rx_analog_reset_ack` output. This port should not be used for register mode data transfers.
Enable rx_pma_clkout port	**On/Off**	Enables the optional `rx_pma_clkout` output clock. This port is the recovered parallel clock from the RX clock data recovery (CDR). (12)
Enable rx_pma_div_clkout port	**On/Off**	Enables the optional `rx_pma_div_clkout` output clock. The deserializer generates this clock. Use this to drive core logic, to drive the RX PCS-to-FPGA fabric interface, or both. If you select a **rx_pma_div_clkout** division factor of 1 or 2, this clock output is derived from the PMA parallel clock. If you select a **rx_pma_div_clkout** division factor of 33, 40, or 66, this clock is derived from the PMA serial clock. This clock is commonly used when the interface to the RX FIFO runs at a different rate than the PMA parallel clock frequency, such as 66:40 applications.
rx_pma_div_clkout division factor	**Disabled**, **1**, **2**, **33**, **40**, **66**	Selects the division factor for the `rx_pma_div_clkout` output clock when enabled. (13)
Enable rx_pma_iqtxrx_clkout port	**On/Off**	Enables the optional `rx_pma_iqtxrx_clkout` output clock. This clock can be used to cascade the RX PMA output clock to the input of a PLL.

continued...

(12) This clock should not be used to clock the FPGA - transceiver interface. This clock may be used as a reference clock to an external clock cleaner.

(13) The default value is **Disabled**.

Parameters	Value	Description
Enable rx_pma_clkslip port	**On/Off**	Enables the optional `rx_pma_clkslip` control input port. A rising edge on this signal causes the RX serializer to slip the serial data by one clock cycle, or 2 unit intervals (UI).
Enable rx_is_lockedtodata port	**On/Off**	Enables the optional `rx_is_lockedtodata` status output port. This signal indicates that the RX CDR is currently in lock to data mode or is attempting to lock to the incoming data stream. This is an asynchronous output signal.
Enable rx_is_lockedtoref port	**On/Off**	Enables the optional `rx_is_lockedtoref` status output port. This signal indicates that the RX CDR is currently locked to the CDR reference clock. This is an asynchronous output signal.
Enable rx_set_lockedtodata port and rx_set_lockedtoref ports	**On/Off**	Enables the optional `rx_set_lockedtodata` and `rx_set_lockedtoref` control input ports. You can use these control ports to manually control the lock mode of the RX CDR. These are asynchronous input signals.
Enable rx_seriallpbken port	**On/Off**	Enables the optional `rx_seriallpbken` control input port. The assertion of this signal enables the TX to RX serial loopback path within the transceiver. This is an asynchronous input signal.
Enable PRBS (Pseudo Random Bit Sequence) verifier control and status port	**On/Off**	Enables the optional `rx_prbs_err`, `rx_prbs_clr`, and `rx_prbs_done` control ports. These ports control and collect status from the internal PRBS verifier.

2.4.4. Enhanced PCS Parameters

This section defines parameters available in the Native PHY IP core GUI to customize the individual blocks in the Enhanced PCS.

The following tables describe the available parameters. Based on the selection of the **Transceiver Configuration Rule** , if the specified settings violate the protocol standard, the Native PHY IP core **Parameter Editor** prints error or warning messages.

Note: For detailed descriptions about the optional ports that you can enable or disable, refer to the *Enhanced PCS Ports* section.

Table 12. Enhanced PCS Parameters

Parameter	Range	Description
Enhanced PCS / PMA interface width	**32**, **40**, **64**	Specifies the interface width between the Enhanced PCS and the PMA.
FPGA fabric /Enhanced PCS interface width	**32**, **40**, **64**, **66**, **67**	Specifies the interface width between the Enhanced PCS and the FPGA fabric. The 66-bit FPGA fabric to PCS interface width uses 64-bits from the TX and RX parallel data. The block synchronizer determines the block boundary of the 66-bit word, with lower 2 bits from the control bus.

continued...

Parameter	Range	Description
		The 67-bit FPGA fabric to PCS interface width uses the 64-bits from the TX and RX parallel data. The block synchronizer determines the block boundary of the 67-bit word with lower 3 bits from the control bus.
Enable Enhanced PCS low latency mode	**On/Off**	Enables the low latency path for the Enhanced PCS. When you turn on this option, the individual functional blocks within the Enhanced PCS are bypassed to provide the lowest latency path from the PMA through the Enhanced PCS.
Enable RX/TX FIFO double width mode	**On/Off**	Enables the double width mode for the RX and TX FIFOs. You can use double width mode to run the FPGA fabric at half the frequency of the PCS.

Table 13. Enhanced PCS TX FIFO Parameters

Parameter	Range	Description
TX FIFO Mode	**Phase-Compensation** **Register** **Interlaken** **Basic** **Fast Register**	Specifies one of the following modes: • **Phase Compensation**: The TX FIFO compensates for the clock phase difference between the read clock `rx_clkout` and the write clocks `tx_coreclkin` or `tx_clkout`. You can tie `tx_enh_data_valid` to 1'b1. • **Register**: The TX FIFO is bypassed. The `tx_parallel_data`, `tx_control` and `tx_enh_data_valid` are registered at the FIFO output. Assert `tx_enh_data_valid` port 1'b1 at all times. The user must connect the write clock `tx_coreclkin` to the read clock `tx_clkout`. • **Interlaken**: The TX FIFO acts as an elastic buffer. In this mode, there are additional signals to control the data flow into the FIFO. Therefore, the FIFO write clock frequency does not have to be the same as the read clock frequency. You can control writes to the FIFO with `tx_enh_data_valid`. By monitoring the FIFO flags, you can avoid the FIFO full and empty conditions. The Interlaken frame generator controls reads. • **Basic**: The TX FIFO acts as an elastic buffer. This mode allows driving write and read side of FIFO with different clock frequencies. `tx_coreclkin` or `rx_coreclkin` must have a minimum frequency of the lane data rate divided by 66. The frequency range for `tx_coreclkin` or `rx_coreclkin` is (data rate/32) - (data rate/66). For best results, Intel recommends that `tx_coreclkin` or `rx_coreclkin` = (data rate/32). Monitor FIFO flag to control write and read operations. For additional details refer to Enhanced PCS FIFO Operation on page 164 section • **Fast Register**: The TX FIFO allows a higher maximum frequency (f_{MAX}) between the FPGA fabric and the TX PCS at the expense of higher latency.
TX FIFO partially full threshold	**10, 11, 12, 13**	Specifies the partially full threshold for the Enhanced PCS TX FIFO. Enter the value at which you want the TX FIFO to flag a partially full status.
TX FIFO partially empty threshold	**2, 3, 4, 5**	Specifies the partially empty threshold for the Enhanced PCS TX FIFO. Enter the value at which you want the TX FIFO to flag a partially empty status.
Enable tx_enh_fifo_full port	**On** / **Off**	Enables the **tx_enh_fifo_full port**. This signal indicates when the TX FIFO is full. This signal is synchronous to `tx_coreclkin`.

continued...

Parameter	Range	Description
Enable tx_enh_fifo_pfull port	**On** / **Off**	Enables the **tx_enh_fifo_pfull port**. This signal indicates when the TX FIFO reaches the specified partially full threshold. This signal is synchronous to `tx_coreclkin`.
Enable tx_enh_fifo_empty port	**On** / **Off**	Enables the **tx_enh_fifo_empty port**. This signal indicates when the TX FIFO is empty. This signal is synchronous to `tx_coreclkin`.
Enable tx_enh_fifo_pempty port	**On** / **Off**	Enables the **tx_enh_fifo_pempty port**. This signal indicates when the TX FIFO reaches the specified partially empty threshold. This signal is synchronous to `tx_coreclkin`.

Table 14. Enhanced PCS RX FIFO Parameters

Parameter	Range	Description
RX FIFO Mode	**Phase-Compensation** **Register** **Interlaken** **10GBASE-R** **Basic**	Specifies one of the following modes for Enhanced PCS RX FIFO: • **Phase Compensation**: This mode compensates for the clock phase difference between the read clocks `rx_coreclkin` or `tx_clkout` and the write clock `rx_clkout`. • **Register** : The RX FIFO is bypassed. The `rx_parallel_data`, `rx_control`, and `rx_enh_data_valid` are registered at the FIFO output. The FIFO's read clock `rx_coreclkin` and write clock `rx_clkout` are tied together. • **Interlaken**: Select this mode for the Interlaken protocol. To implement the deskew process, you must implement an FSM that controls the FIFO operation based on FIFO flags. In this mode the FIFO acts as an elastic buffer. • **10GBASE-R**: In this mode, data passes through the FIFO after block lock is achieved. OS (Ordered Sets) are deleted and Idles are inserted to compensate for the clock difference between the RX PMA clock and the fabric clock of +/- 100 ppm for a maximum packet length of 64000 bytes. • **Basic:** In this mode, the RX FIFO acts as an elastic buffer. This mode allows driving write and read side of FIFO with different clock frequencies. `tx_coreclkin` or `rx_coreclkin` must have a minimum frequency of the lane data rate divided by 66. The frequency range for `tx_coreclkin` or `rx_coreclkin` is (data rate/32) - (data rate/66). The gearbox data valid flag controls the FIFO read enable. You can monitor the `rx_enh_fifo_pfull` and `rx_enh_fifo_empty` flags to determine whether or not to read from the FIFO. For additional details refer to Enhanced PCS FIFO Operation on page 164. *Note:* The flags are for Interlaken and Basic modes only. They should be ignored in all other cases.
RX FIFO partially full threshold	18-29	Specifies the partially full threshold for the Enhanced PCS RX FIFO. The default value is 23.
RX FIFO partially empty threshold	2-10	Specifies the partially empty threshold for the Enhanced PCS RX FIFO. The default value is 2.
Enable RX FIFO alignment word deletion (Interlaken)	**On** / **Off**	When you turn on this option, all alignment words (sync words), including the first sync word, are removed after frame synchronization is achieved. If you enable this option, you must also enable control word deletion.
Enable RX FIFO control word deletion (Interlaken)	**On** / **Off**	When you turn on this option, Interlaken control word removal is enabled. When the Enhanced PCS RX FIFO is configured in **Interlaken** mode, enabling this option, removes all control words after frame synchronization is achieved. Enabling this option requires that you also enable alignment word deletion.

continued...

Parameter	Range	Description
Enable rx_enh_data_valid port	**On** / **Off**	Enables the **rx_enh_data_valid port**. This signal indicates when RX data from RX FIFO is valid. This signal is synchronous to `rx_coreclkin`.
Enable rx_enh_fifo_full port	**On** / **Off**	Enables the **rx_enh_fifo_full port**. This signal indicates when the RX FIFO is full. This is an asynchronous signal.
Enable rx_enh_fifo_pfull port	**On** / **Off**	Enables the **rx_enh_fifo_pfull port**. This signal indicates when the RX FIFO has reached the specified partially full threshold. This is an asynchronous signal.
Enable rx_enh_fifo_empty port	**On** / **Off**	Enables the **rx_enh_fifo_empty port**. This signal indicates when the RX FIFO is empty. This signal is synchronous to `rx_coreclkin`.
Enable rx_enh_fifo_pempty port	**On** / **Off**	Enables the **rx_enh_fifo_pempty port**. This signal indicates when the RX FIFO has reached the specified partially empty threshold. This signal is synchronous to `rx_coreclkin`.
Enable rx_enh_fifo_del port (10GBASE-R)	**On** / **Off**	Enables the optional **rx_enh_fifo_del status output port**. This signal indicates when a word has been deleted from the rate match FIFO. This signal is only used for 10GBASE-R transceiver configuration rule. This is an asynchronous signal.
Enable rx_enh_fifo_insert port (10GBASE-R)	**On** / **Off**	Enables the **rx_enh_fifo_insert port**. This signal indicates when a word has been inserted into the rate match FIFO. This signal is only used for 10GBASE-R transceiver configuration rule. This signal is synchronous to `rx_coreclkin`.
Enable rx_enh_fifo_rd_en port	**On** / **Off**	Enables the **rx_enh_fifo_rd_en input port**. This signal is enabled to read a word from the RX FIFO. This signal is synchronous to `rx_coreclkin`.
Enable rx_enh_fifo_align_val port (Interlaken)	**On** / **Off**	Enables the **rx_enh_fifo_align_val status output port**. Only used for Interlaken transceiver configuration rule. This signal is synchronous to `rx_clkout`.
Enable rx_enh_fifo_align_clr port (Interlaken)	**On** / **Off**	Enables the **rx_enh_fifo_align_clr input port**. Only used for Interlaken. This signal is synchronous to `rx_clkout`.

Table 15. Interlaken Frame Generator Parameters

Parameter	Range	Description
Enable Interlaken frame generator	**On** / **Off**	Enables the frame generator block of the Enhanced PCS.
Frame generator metaframe length	5-8192	Specifies the metaframe length of the frame generator. This metaframe length includes 4 framing control words created by the frame generator.
Enable Frame Generator Burst Control	**On** / **Off**	Enables frame generator burst. This determines whether the frame generator reads data from the TX FIFO based on the input of port `tx_enh_frame_burst_en`.

continued...

Parameter	Range	Description
Enable tx_enh_frame port	**On** / **Off**	Enables the `tx_enh_frame` status output port. When the Interlaken frame generator is enabled, this signal indicates the beginning of a new metaframe. This is an asynchronous signal.
Enable tx_enh_frame_diag_status port	**On** / **Off**	Enables the `tx_enh_frame_diag_status` 2-bit input port. When the Interlaken frame generator is enabled, the value of this signal contains the status message from the framing layer diagnostic word. This signal is synchronous to `tx_clkout`.
Enable tx_enh_frame_burst_en port	**On** / **Off**	Enables the `tx_enh_frame_burst_en` input port. When burst control is enabled for the Interlaken frame generator, this signal is asserted to control the frame generator data reads from the TX FIFO. This signal is synchronous to `tx_clkout`.

Table 16. Interlaken Frame Synchronizer Parameters

Parameter	Range	Description
Enable Interlaken frame synchronizer	**On** / **Off**	When you turn on this option, the Enhanced PCS frame synchronizer is enabled.
Frame synchronizer metaframe length	5-8192	Specifies the metaframe length of the frame synchronizer.
Enable rx_enh_frame port	**On** / **Off**	Enables the **rx_enh_frame status output port**. When the Interlaken frame synchronizer is enabled, this signal indicates the beginning of a new metaframe. This is an asynchronous signal.
Enable rx_enh_frame_lock port	**On** / **Off**	Enables the **rx_enh_frame_lock output port**. When the Interlaken frame synchronizer is enabled, this signal is asserted to indicate that the frame synchronizer has achieved metaframe delineation. This is an asynchronous output signal.
Enable rx_enh_frame_diag_status port	**On** / **Off**	Enables the**rx_enh_frame_diag_status output port**. When the Interlaken frame synchronizer is enabled, this signal contains the value of the framing layer diagnostic word (bits [33:32]). This is a 2 bit per lane output signal. It is latched when a valid diagnostic word is received. This is an asynchronous signal.

Table 17. Interlaken CRC32 Generator and Checker Parameters

Parameter	Range	Description
Enable Interlaken TX CRC-32 Generator	**On** / **Off**	When you turn on this option, the TX Enhanced PCS datapath enables the CRC32 generator function. CRC32 can be used as a diagnostic tool. The CRC contains the entire metaframe including the diagnostic word.
Enable Interlaken TX CRC-32 generator error insertion	**On** / **Off**	When you turn on this option, the error insertion of the interlaken CRC-32 generator is enabled. Error insertion is cycle-accurate. When this feature is enabled, the assertion of `tx_control[8]` or `tx_err_ins` signal causes the CRC calculation during that word is incorrectly inverted, and thus, the CRC created for that metaframe is incorrect.
Enable Interlaken RX CRC-32 checker	**On** / **Off**	Enables the CRC-32 checker function.
Enable rx_enh_crc32_err port	**On** / **Off**	When you turn on this option, the Enhanced PCS enables the **rx_enh_crc32_err port**. This signal is asserted to indicate that the CRC checker has found an error in the current metaframe. This is an asynchronous signal.

Table 18. 10GBASE-R BER Checker Parameters

Parameter	Range	Description
Enable rx_enh_highber port (10GBASE-R)	**On** / **Off**	Enables the **rx_enh_highber port**. For 10GBASE-R transceiver configuration rule, this signal is asserted to indicate a bit error rate higher than 10^{-4} . Per the 10GBASE-R specification, this occurs when there are at least 16 errors within 125 µs. This is an asynchronous signal.
Enable rx_enh_highber_clr_cnt port (10GBASE-R)	**On** / **Off**	Enables the **rx_enh_highber_clr_cnt input port**. For the 10GBASE-R transceiver configuration rule, this signal is asserted to clear the internal counter. This counter indicates the number of times the BER state machine has entered the "BER_BAD_SH" state. This is an asynchronous signal.
Enable rx_enh_clr_errblk_count port (10GBASE-R)	**On** / **Off**	Enables the **rx_enh_clr_errblk_count input port**. For the 10GBASE-R transceiver configuration rule, this signal is asserted to clear the internal counter. This counter indicates the number of the times the RX state machine has entered the RX_E state. This is an asynchronous signal.

Table 19. 64b/66b Encoder and Decoder Parameters

Parameter	Range	Description
Enable TX 64b/66b encoder (10GBASE-R)	**On** / **Off**	When you turn on this option, the Enhanced PCS enables the TX 64b/66b encoder.
Enable RX 64b/66b decoder (10GBASE-R)	**On** / **Off**	When you turn on this option, the Enhanced PCS enables the RX 64b/66b decoder.
Enable TX sync header error insertion	**On** / **Off**	When you turn on this option, the Enhanced PCS supports cycle-accurate error creation to assist in exercising error condition testing on the receiver. When error insertion is enabled and the error flag is set, the encoding sync header for the current word is generated incorrectly. If the correct sync header is 2'b01 (control type), 2'b00 is encoded. If the correct sync header is 2'b10 (data type), 2'b11 is encoded.

Table 20. Scrambler and Descrambler Parameters

Parameter	Range	Description
Enable TX scrambler (10GBASE-R/ Interlaken)	**On** / **Off**	Enables the scrambler function. This option is available for the Basic (Enhanced PCS) mode, Interlaken, and 10GBASE-R protocols. You can enable the scrambler in Basic (Enhanced PCS) mode when the block synchronizer is enabled and with 66:32, 66:40, or 66:64 gear box ratios.
TX scrambler seed (10GBASE-R/ Interlaken)	User-specified 58-bit value	You must provide a non-zero seed for the Interlaken protocol. For a multi-lane Interlaken Transceiver Native PHY IP, the first lane scrambler has this seed. For other lanes' scrambler, this seed is increased by 1 per each lane. The initial seed for 10GBASE-R is 0x03FFFFFFFFFFFFFF. This parameter is required for the 10GBASE-R and Interlaken protocols.
Enable RX descrambler (10GBASE-R/ Interlaken)	**On** / **Off**	Enables the descrambler function. This option is available for Basic (Enhanced PCS) mode, Interlaken, and 10GBASE-R protocols. You can enable the descrambler in Basic (Enhanced PCS) mode with the block synchronizer enabled and with 66:32, 66:40, or 66:64 gear box ratios.

Table 21. Interlaken Disparity Generator and Checker Parameters

Parameter	Range	Description
Enable Interlaken TX disparity generator	**On** / **Off**	When you turn on this option, the Enhanced PCS enables the disparity generator. This option is available for the Interlaken protocol.
Enable Interlaken RX disparity checker	**On** / **Off**	When you turn on this option, the Enhanced PCS enables the disparity checker. This option is available for the Interlaken protocol.
Enable Interlaken TX random disparity bit	**On** / **Off**	Enables the Interlaken random disparity bit. When enabled, a random number is used as disparity bit which saves one cycle of latency.

Table 22. Block Synchronizer Parameters

Parameter	Range	Description
Enable RX block synchronizer	**On** / **Off**	When you turn on this option, the Enhanced PCS enables the RX block synchronizer. This options is available for the Basic (Enhanced PCS) mode, Interlaken, and 10GBASE-R protocols.
Enable rx_enh_blk_lock port	**On** / **Off**	Enables the `rx_enh_blk_lock` port. When you enable the block synchronizer, this signal is asserted to indicate that the block delineation has been achieved.

Table 23. Gearbox Parameters

Parameter	Range	Description
Enable TX data bitslip	**On** / **Off**	When you turn on this option, the TX gearbox operates in bitslip mode. The **tx_enh_bitslip port** controls number of bits which TX parallel data slips before going to the PMA.
Enable TX data polarity inversion	**On** / **Off**	When you turn on this option, the polarity of TX data is inverted. This allows you to correct incorrect placement and routing on the PCB.
Enable RX data bitslip	**On** / **Off**	When you turn on this option, the Enhanced PCS RX block synchronizer operates in bitslip mode. When enabled, the **rx_bitslip port** is asserted on the rising edge to ensure that RX parallel data from the PMA slips by one bit before passing to the PCS.
Enable RX data polarity inversion	**On** / **Off**	When you turn on this option, the polarity of the RX data is inverted. This allows you to correct incorrect placement and routing on the PCB.
Enable tx_enh_bitslip port	**On** / **Off**	Enables the **tx_enh_bitslip port**. When TX bit slip is enabled, this signal controls the number of bits which TX parallel data slips before going to the PMA.
Enable rx_bitslip port	**On** / **Off**	Enables the **rx_bitslip port**. When RX bit slip is enabled, the `rx_bitslip` signal is asserted on the rising edge to ensure that RX parallel data from the PMA slips by one bit before passing to the PCS. This port is shared between Standard PCS and Enhanced PCS.

Related Information

- Enhanced PCS Ports on page 53
- Cyclone 10 GX Enhanced PCS Architecture on page 283
- Interlaken on page 70
- 10GBASE-R and 10GBASE-R with IEEE 1588v2 Variants on page 98

- Using the "Basic (Enhanced PCS)" Configuration on page 158

2.4.5. Standard PCS Parameters

This section provides descriptions of the parameters that you can specify to customize the Standard PCS.

For specific information about configuring the Standard PCS for these protocols, refer to the sections of this user guide that describe support for these protocols.

Table 24. Standard PCS Parameters

Note: For detailed descriptions of the optional ports that you can enable or disable, refer to the *Standard PCS Ports* section.

Parameter	Range	Description
Standard PCS/PMA interface width	**8, 10, 16, 20**	Specifies the data interface width between the Standard PCS and the transceiver PMA.
FPGA fabric/Standard TX PCS interface width	**8, 10, 16, 20, 32, 40**	Shows the FPGA fabric to TX PCS interface width. This value is determined by the current configuration of individual blocks within the Standard TX PCS datapath.
FPGA fabric/Standard RX PCS interface width	**8, 10, 16, 20, 32, 40**	Shows the FPGA fabric to RX PCS interface width. This value is determined by the current configuration of individual blocks within the Standard RX PCS datapath.
Enable Standard PCS low latency mode	**On** / **Off**	Enables the low latency path for the Standard PCS. Some of the functional blocks within the Standard PCS are bypassed to provide the lowest latency. You cannot turn on this parameter while using the **Basic/Custom w/Rate Match (Standard PCS)** specified for **Transceiver configuration rules**.

Table 25. Standard PCS FIFO Parameters

Parameter	Range	Description
TX FIFO mode	**low_latency** **register_fifo** **fast_register**	Specifies the Standard PCS TX FIFO mode. The following modes are available: • **low_latency**: This mode adds 2-3 cycles of latency to the TX datapath. • **register_fifo**: In this mode the FIFO is replaced by registers to reduce the latency through the PCS. Use this mode for protocols that require deterministic latency, such as CPRI. • **fast_register**: This mode allows a higher maximum frequency (f_{MAX}) between the FPGA fabric and the TX PCS at the expense of higher latency.
RX FIFO mode	**low_latency** **register_fifo**	The following modes are available: • **low_latency**: This mode adds 2-3 cycles of latency to the RX datapath. • **register_fifo**: In this mode the FIFO is replaced by registers to reduce the latency through the PCS. Use this mode for protocols that require deterministic latency, such as CPRI or 1588.
Enable tx_std_pcfifo_full port	**On** / **Off**	Enables the `tx_std_pcfifo_full` port. This signal indicates when the standard TX phase compensation FIFO is full. This signal is synchronous with `tx_coreclkin`.

continued...

Parameter	Range	Description
Enable tx_std_pcfifo_empty port	**On** / **Off**	Enables the `tx_std_pcfifo_empty` port. This signal indicates when the standard TX phase compensation FIFO is empty. This signal is synchronous with `tx_coreclkin`.
Enable rx_std_pcfifo_full port	**On** / **Off**	Enables the `rx_std_pcfifo_full` port. This signal indicates when the standard RX phase compensation FIFO is full. This signal is synchronous with `rx_coreclkin`.
Enable rx_std_pcfifo_empty port	**On** / **Off**	Enables the `rx_std_pcfifo_empty` port. This signal indicates when the standard RX phase compensation FIFO is empty. This signal is synchronous with `rx_coreclkin`.

Table 26. Byte Serializer and Deserializer Parameters

Parameter	Range	Description
Enable TX byte serializer	**Disabled** **Serialize x2** **Serialize x4**	Specifies the TX byte serializer mode for the Standard PCS. The transceiver architecture allows the Standard PCS to operate at double or quadruple the data width of the PMA serializer. The byte serializer allows the PCS to run at a lower internal clock frequency to accommodate a wider range of FPGA interface widths. **Serialize x4** is only applicable for PCIe protocol implementation.
Enable RX byte deserializer	**Disabled** **Deserialize x2** **Deserialize x4**	Specifies the mode for the RX byte deserializer in the Standard PCS. The transceiver architecture allows the Standard PCS to operate at double or quadruple the data width of the PMA deserializer. The byte deserializer allows the PCS to run at a lower internal clock frequency to accommodate a wider range of FPGA interface widths. **Deserialize x4** is only applicable for PCIe protocol implementation.

Table 27. 8B/10B Encoder and Decoder Parameters

Parameter	Range	Description
Enable TX 8B/10B encoder	**On** / **Off**	When you turn on this option, the Standard PCS enables the TX 8B/10B encoder.
Enable TX 8B/10B disparity control	**On** / **Off**	When you turn on this option, the Standard PCS includes disparity control for the 8B/10B encoder. You can force the disparity of the 8B/10B encoder using the `tx_forcedisp` control signal.
Enable RX 8B/10B decoder	**On** / **Off**	When you turn on this option, the Standard PCS includes the 8B/10B decoder.

Table 28. Rate Match FIFO Parameters

Parameter	Range	Description
RX rate match FIFO mode	**Disabled** **Basic 10-bit PMA width** **Basic 20-bit PMA width** **GbE** **PIPE** **PIPE 0 ppm**	Specifies the operation of the RX rate match FIFO in the Standard PCS. Rate Match FIFO in Basic (Single Width) Mode on page 173 Rate Match FIFO Basic (Double Width) Mode on page 175 Rate Match FIFO for GbE on page 92 Transceiver Channel Datapath for PIPE on page 123
RX rate match insert/ delete -ve pattern (hex)	User-specified 20 bit pattern	Specifies the -ve (negative) disparity value for the RX rate match FIFO as a hexadecimal string.

continued...

Parameter	Range	Description
RX rate match insert/ delete +ve pattern (hex)	User-specified 20 bit pattern	Specifies the +ve (positive) disparity value for the RX rate match FIFO as a hexadecimal string.
Enable rx_std_rmfifo_full port	**On** / **Off**	Enables the optional `rx_std_rmfifo_full` port.
Enable rx_std_rmfifo_empty port	**On** / **Off**	Enables the `rx_std_rmfifo_empty` port.

Table 29. Word Aligner and Bitslip Parameters

Parameter	Range	Description
Enable TX bitslip	**On** / **Off**	When you turn on this option, the PCS includes the bitslip function. The outgoing TX data can be slipped by the number of bits specified by the `tx_std_bitslipboundarysel` control signal.
Enable tx_std_bitslipboundarysel port	**On** / **Off**	Enables the `tx_std_bitslipboundarysel` control signal.
RX word aligner mode	**bitslip** **manual (PLD controlled)** **synchronous state machine** **deterministic latency**	Specifies the RX word aligner mode for the Standard PCS. The word aligned width depends on the PCS and PMA width, and whether or not 8B/10B is enabled. Refer to "Word Aligner" for more information.
RX word aligner pattern length	**7**, **8**, **10**, **16**, **20**, **32**, **40**	Specifies the length of the pattern the word aligner uses for alignment. Refer to "RX Word Aligner Pattern Length" table in "Word Aligner". It shows the possible values of "Rx Word Aligner Pattern Length" in all available word aligner modes.
RX word aligner pattern (hex)	User-specified	Specifies the word alignment pattern in hex.
Number of word alignment patterns to achieve sync	`0-255`	Specifies the number of valid word alignment patterns that must be received before the word aligner achieves synchronization lock. The default is 3.
Number of invalid words to lose sync	`0-63`	Specifies the number of invalid data codes or disparity errors that must be received before the word aligner loses synchronization. The default is 3.
Number of valid data words to decrement error count	`0-255`	Specifies the number of valid data codes that must be received to decrement the error counter. If the word aligner receives enough valid data codes to decrement the error count to 0, the word aligner returns to synchronization lock.
Enable fast sync status reporting for deterministic Latency SM	**On** / **Off**	When enabled, the `rx_syncstatus` asserts high immediately after the deserializer has completed slipping the bits to achieve word alignment. When it is not selected, `rx_syncstatus` asserts after the cycle slip operation is complete and the word alignment pattern is detected by the PCS (i.e. `rx_patterndetect` is asserted). This parameter is only applicable when the selected protocol is CPRI (Auto).
Enable rx_std_wa_patternalign port	**On** / **Off**	Enables the `rx_std_wa_patternalign` port. When the word aligner is configured in manual mode and when this signal is enabled, the word aligner aligns to next incoming word alignment pattern.

continued...

Parameter	Range	Description
Enable rx_std_wa_a1a2size port	**On** / **Off**	Enables the optional `rx_std_wa_a1a2size` control input port.
Enable rx_std_bitslipboundarysel port	**On** / **Off**	Enables the optional `rx_std_bitslipboundarysel` status output port.
Enable rx_bitslip port	**On** / **Off**	Enables the `rx_bitslip` port. This port is shared between the Standard PCS and Enhanced PCS.

Table 30. Bit Reversal and Polarity Inversion

Parameter	Range	Description
Enable TX bit reversal	**On** / **Off**	When you turn on this option, the 8B/10B Encoder reverses TX parallel data before transmitting it to the PMA for serialization. The transmitted TX data bit order is reversed. The normal order is LSB to MSB. The reverse order is MSB to LSB. During the operation of the circuit, this setting can be changed through dynamic reconfiguration.
Enable TX byte reversal	**On** / **Off**	When you turn on this option, the 8B/10B Encoder reverses the byte order before transmitting data. This function allows you to reverse the order of bytes that were erroneously swapped. The PCS can swap the ordering of either one of the 8- or 10-bit words, when the PCS/PMA interface width is 16 or 20 bits. This option is not valid under certain **Transceiver configuration rules**.
Enable TX polarity inversion	**On** / **Off**	When you turn on this option, the `tx_std_polinv` port controls polarity inversion of TX parallel data to the PMA. When you turn on this parameter, you also need to turn on the **Enable tx_polinv port**.
Enable tx_polinv port	**On** / **Off**	When you turn on this option, the `tx_polinv` input control port is enabled. You can use this control port to swap the positive and negative signals of a serial differential link, if they were erroneously swapped during board layout.
Enable RX bit reversal	**On** / **Off**	When you turn on this option, the word aligner reverses RX parallel data. The received RX data bit order is reversed. The normal order is LSB to MSB. The reverse order is MSB to LSB. This setting can be changed through dynamic reconfiguration. When you enable **Enable RX bit reversal**, you must also enable **Enable rx_std_bitrev_ena port**.
Enable rx_std_bitrev_ena port	**On** / **Off**	When you turn on this option and assert the `rx_std_bitrev_ena` control port, the RX data order is reversed. The normal order is LSB to MSB. The reverse order is MSB to LSB.
Enable RX byte reversal	**On** / **Off**	When you turn on this option, the word aligner reverses the byte order, before storing the data in the RX FIFO. This function allows you to reverse the order of bytes that are erroneously swapped. The PCS can swap the ordering of either one of the 8- or 10-bit words, when the PCS / PMA interface width is 16 or 20 bits. This option is not valid under certain **Transceiver configuration rules**. When you enable **Enable RX byte reversal**, you must also select the **Enable rx_std_byterev_ena port**.
Enable rx_std_byterev_ena port	**On** / **Off**	When you turn on this option and assert the `rx_std_byterev_ena` input control port, the order of the individual 8- or 10-bit words received from the PMA is swapped.

continued...

Parameter	Range	Description
Enable RX polarity inversion	**On** / **Off**	When you turn on this option, the `rx_std_polinv` port inverts the polarity of RX parallel data. When you turn on this parameter, you also need to enable **Enable rx_polinv port**.
Enable rx_polinv port	**On** / **Off**	When you turn on this option, the `rx_polinv` input is enabled. You can use this control port to swap the positive and negative signals of a serial differential link if they were erroneously swapped during board layout.
Enable rx_std_signaldetect port	**On** / **Off**	When you turn on this option, the optional `rx_std_signaldetect` output port is enabled. This signal is required for the PCI Express protocol. If enabled, the signal threshold detection circuitry senses whether the signal level present at the RX input buffer is above the signal detect threshold voltage that you specified. You can specify the signal detect threshold using a Quartus Prime Assignment Editor or by modifying the Quartus Settings File (.qsf)

Table 31. PCIe Ports

Parameter	Range	Description
Enable PCIe dynamic datarate switch ports	**On** / **Off**	When you turn on this option, the `pipe_rate`, `pipe_sw`, and `pipe_sw_done` ports are enabled. You should connect these ports to the PLL IP core instance in multi-lane PCIe Gen2 configurations. The `pipe_sw` and `pipe_sw_done` ports are only available for multi-lane bonded configurations.
Enable PCIe pipe_hclk_in and pipe_hclk_out ports	**On** / **Off**	When you turn on this option, the `pipe_hclk_in`, and `pipe_hclk_out` ports are enabled. These ports must be connected to the PLL IP core instance for the PCI Express configurations.
Enable PCIe electrical idle control and status ports	**On** / **Off**	When you turn on this option, the `pipe_rx_eidleinfersel` and `pipe_rx_elecidle` ports are enabled. These ports are used for PCI Express configurations.
Enable PCIe pipe_rx_polarity port	**On** / **Off**	When you turn on this option, the `pipe_rx_polarity` input control port is enabled. You can use this option to control channel signal polarity for PCI Express configurations. When the Standard PCS is configured for PCIe, the assertion of this signal inverts the RX bit polarity. For other **Transceiver configuration rules** the optional `rx_polinv` port inverts the polarity of the RX bit stream.

2.4.6. PCS Direct

Table 32. PCS Direct Datapath Parameters

Parameter	Range	Description
PCS Direct interface width	**8, 10, 16, 20, 32, 40, 64**	Specifies the data interface width between the PLD and the transceiver PMA.

2.4.7. Dynamic Reconfiguration Parameters

Dynamic reconfiguration allows you to change the behavior of the transceiver channels and PLLs without powering down the device. Each transceiver channel and PLL includes an Avalon-MM slave interface for reconfiguration. This interface provides direct access to the programmable address space of each channel and PLL. Because each channel and PLL includes a dedicated Avalon-MM slave interface, you can

dynamically modify channels either concurrently or sequentially. If your system does not require concurrent reconfiguration, you can parameterize the Transceiver Native PHY IP to share a single reconfiguration interface.

You can use dynamic reconfiguration to change many functions and features of the transceiver channels and PLLs. For example, you can change the reference clock input to the TX PLL. You can also change between the Standard and Enhanced datapaths.

To enable Intel Cyclone 10 GX transceiver toolkit capability in the Native PHY IP core, you must enable the following options:

- **Enable dynamic reconfiguration**
- **Enable Native PHY Debug Master Endpoint**
- **Enable capability registers**
- **Enable control and status registers**
- **Enable PRBS (Pseudo Random Binary Sequence) soft accumulators**

Table 33. Dynamic Reconfiguration

Parameter	Value	Description
Enable dynamic reconfiguration	**On/Off**	When you turn on this option, the dynamic reconfiguration interface is enabled.
Share reconfiguration interface	**On/Off**	When you turn on this option, the Transceiver Native PHY IP presents a single Avalon-MM slave interface for dynamic reconfiguration for all channels. In this configuration, the upper [*n*-*1*:10] address bits of the reconfiguration address bus specify the channel. The channel numbers are binary encoded. Address bits [9:0] provide the register offset address within the reconfiguration space for a channel.
Enable Native PHY Debug Master Endpoint	**On/Off**	When you turn on this option, the Transceiver Native PHY IP includes an embedded Native PHY Debug Master Endpoint (NPDME) that connects internally to the Avalon-MM slave interface for dynamic reconfiguration. The NPDME can access the reconfiguration space of the transceiver. It can perform certain test and debug functions via JTAG using the System Console. This option requires you to enable the **Share reconfiguration interface** option for configurations using more than one channel.
Separate reconfig_waitrequest from the status of AVMM arbitration with PreSICE	**On/Off**	When enabled, the `reconfig_waitrequest` does not indicate the status of AVMM arbitration with PreSICE. The AVMM arbitration status is reflected in a soft status register bit. This feature requires that the "Enable control and status registers" feature under "Optional Reconfiguration Logic" be enabled.

Table 34. Optional Reconfiguration Logic

Parameter	Value	Description
Enable capability registers	**On/Off**	Enables capability registers that provide high level information about the configuration of the transceiver channel.
Set user-defined IP identifier	**User-defined**	Sets a user-defined numeric identifier that can be read from the `user_identifier` offset when the capability registers are enabled.
Enable control and status registers	**On/Off**	Enables soft registers to read status signals and write control signals on the PHY interface through the embedded debug.
Enable PRBS (Pseudo Random Binary Sequence) soft accumulators	**On/Off**	Enables soft logic for performing PRBS bit and error accumulation when the hard PRBS generator and checker are used.

Table 35. Configuration Files

Parameter	Value	Description
Configuration file prefix	*<prefix>*	Here, the file prefix to use for generated configuration files is specified. Each variant of the Transceiver Native PHY IP should use a unique prefix for configuration files.
Generate SystemVerilog package file	**On/Off**	When you turn on this option, the Transceiver Native PHY IP generates a SystemVerilog package file, **reconfig_parameters.sv**. This file contains parameters defined with the attribute values required for reconfiguration.
Generate C header file	**On/Off**	When you turn on this option, the Transceiver Native PHY IP generates a C header file, **reconfig_parameters.h**. This file contains macros defined with the attribute values required for reconfiguration.
Generate MIF (Memory Initialization File)	**On/Off**	When you turn on this option, the Transceiver Native PHY IP generates a MIF, **reconfig_parameters.mif**. This file contains the attribute values required for reconfiguration in a data format.
Include PMA analog settings in configuration files	**On/Off**	When enabled, the IP allows you to configure the PMA analog settings that are selected in the Analog PMA settings (Optional) tab. These settings are included in your generated configuration files. *Note:* You must still specify the analog settings for your current configuration using Quartus Prime Setting File (.qsf) assignments in Quartus. This option does not remove the requirement to specify Quartus Prime Setting File (.qsf) assignments for your analog settings. Refer to the *Analog Parameter Settings* chapter in the *Cyclone 10 GX Transceiver PHY User Guide* for details on using the QSF assignments.

Table 36. Configuration Profiles

Parameter	Value	Description
Enable multiple reconfiguration profiles	**On/Off**	When enabled, you can use the GUI to store multiple configurations. This information is used by Quartus to include the necessary timing arcs for all configurations during timing driven compilation. The Native PHY generates reconfiguration files for all of the stored profiles. The Native PHY also checks your multiple reconfiguration profiles for consistency to ensure you can reconfigure between them. Among other things this checks that you have exposed the same ports for each configuration.(14)
Enable embedded reconfiguration streamer	**On/Off**	Enables the embedded reconfiguration streamer, which automates the dynamic reconfiguration process between multiple predefined configuration profiles. This is optional and increases logic utilization. The PHY includes all of the logic and data necessary to dynamically reconfigure between pre-configured profiles.
Generate reduced reconfiguration files	**On/Off**	When enabled, The Native PHY generates reconfiguration report files containing only the attributes or RAM data that are different between the multiple configured profiles. The reconfiguration time decreases with the use of reduced .mif files.
Number of reconfiguration profiles	1-8	Specifies the number of reconfiguration profiles to support when multiple reconfiguration profiles are enabled.
Selected reconfiguration profile	0-7	Selects which reconfiguration profile to store/load/clear/refresh, when clicking the relevant button for the selected profile.

continued...

(14) For more information on timing closure, refer to the *Reconfiguration Interface and Dynamic Reconfiguration* chapter.

Parameter	Value	Description
Store configuration to selected profile	-	Clicking this button saves or stores the current Native PHY parameter settings to the profile specified by the **Selected reconfiguration profile** parameter.
Load configuration from selected profile	-	Clicking this button loads the current Native PHY with parameter settings from the stored profile specified by the **Selected reconfiguration profile** parameter.
Clear selected profile	-	Clicking this button clears or erases the stored Native PHY parameter settings for the profile specified by the **Selected reconfiguration profile** parameter. An empty profile defaults to the current parameter settings of the Native PHY.
Clear all profiles	-	Clicking this button clears the Native PHY parameter settings for all the profiles.
Refresh selected profile	-	Clicking this button is equivalent to clicking the **Load configuration from selected profile** and **Store configuration to selected profile** buttons in sequence. This operation loads the Native PHY parameter settings from stored profile specified by the **Selected reconfiguration profile** parameter and subsequently stores or saves the parameters back to the profile.

Table 37. Analog PMA Settings (Optional) for Dynamic Reconfiguration

Parameter	Value	Description
TX Analog PMA Settings		
Analog Mode (Load Intel-recommended Default settings)	Cei_11100_lr to xfp_9950	Selects the analog protocol mode to pre-select the TX pin swing settings (VOD, Pre-emphasis, and Slew Rate). After loading the pre-selected values in the GUI, if one or more of the individual TX pin swing settings need to be changed, then enable the option to override the Intel-recommended defaults to individually modify the settings.
Override Intel-recommended Analog Mode Default settings	**On/Off**	Enables the option to override the Intel-recommended settings for the selected TX Analog Mode for one or more TX analog parameters.
Output Swing Level (VOD)	**0-31**	Selects the transmitter programmable output differential voltage swing.
Pre-Emphasis First Pre-Tap Polarity	`Fir_pre_1t_neg` `Fir_pre_1t_pos`	Selects the polarity of the first pre-tap for pre-emphasis.
Pre-Emphasis First Pre-Tap Magnitude	**0-16** (15)	Selects the magnitude of the first pre-tap for pre-emphasis
Pre-Emphasis Second Pre-Tap Polarity	`Fir_pre_2t_neg` `Fir_pre_2t_pos`	Selects the polarity of the second pre-tap for pre-emphasis.
Pre-Emphasis Second Pre-Tap Magnitude	**0-7** (16)	Selects the magnitude of the second pre-tap for pre-emphasis.

continued...

(15) For more information refer to *Available Options* table in the *XCVR_C10_TX_PRE_EMP_SWITCHING_CTRL_PRE_TAP_1T* section of the *Analog Parameter Settings* chapter.

(16) For more information refer to *Available Options* table in the *XCVR_C10_TX_PRE_EMP_SWITCHING_CTRL_PRE_TAP_2T* section of the *Analog Parameter Settings* chapter.

Parameter	Value	Description
Pre-Emphasis First Post-Tap Polarity	Fir_post_1t_neg Fir_post_1t_pos	Selects the polarity of the first post-tap for pre-emphasis
Pre-Emphasis First Post-Tap Magnitude	**0-25** (17)	Selects the magnitude of the first post-tap for pre-emphasis.
Pre-Emphasis Second Post-Tap Polarity	`Fir_post_2t_neg` `Fir_post_2t_pos`	Selects the polarity of the second post-tap for pre-emphasis.
Pre-Emphasis Second Post-Tap Magnitude	**0-12** (18)	Selects the magnitude of the second post-tap for pre-emphasis
Slew Rate Control	`slew_r0` to `slew_r5`	Selects the slew rate of the TX output signal. Valid values span from slowest to the fastest rate.
High-Speed Compensation	Enable/Disable	Enables the power-distribution network (PDN) induced inter-symbol interference (ISI) compensation in the TX driver. When enabled, it reduces the PDN induced ISI jitter, but increases the power consumption.
On-Chip termination	`r_r1` `r_r2`	Selects the on-chip TX differential termination.
RX Analog PMA Settings		
Override Intel-recommended Default settings	On/Off	Enables the option to override the Intel-recommended settings for one or more RX analog parameters
CTLE (Continuous Time Linear Equalizer) mode	`non_s1_mode`	Selects the RX high gain mode `non_s1_mode` for the Continuous Time Linear Equalizer (CTLE).
DC gain control of high gain mode CTLE	`No_dc_gain` to `stg4_gain7`	Selects the DC gain of the Continuous Time Linear Equalizer (CTLE) in high gain mode
AC Gain Control of High Gain Mode CTLE	`radp_ctle_acgain_4s_0` to `radp_ctle_acgain_4s_28`	Selects the AC gain of the Continuous Time Linear Equalizer (CTLE) in high gain mode when CTLE is in manual mode.
Variable Gain Amplifier (VGA) Voltage Swing Select	`radp_vga_sel_0` to `radp_vga_sel_4`	Selects the Variable Gain Amplifier (VGA) output voltage swing.
On-Chip termination	`R_ext0, r_r1, r_r2`	Selects the on-chip RX differential termination.

Table 38. Generation Options

Parameter	Value	Description
Generate parameter documentation file	**On/Off**	When you turn on this option, generation produces a Comma-Separated Value (**.csv**) file with descriptions of the Transceiver Native PHY IP parameters.

(17) For more information refer to *Available Options* table in the *XCVR_C10_TX_PRE_EMP_SWITCHING_CTRL_1ST_POST_TAP* section of the *Analog Parameter Settings* chapter.

(18) For more information refer to *Available Options* table in the *XCVR_C10_TX_PRE_EMP_SWITCHING_CTRL_2ND_POST_TAP* section of the *Analog Parameter Settings* chapter.

2.4.8. PMA Ports

This section describes the PMA and calibration ports for the Cyclone 10 GX Transceiver Native PHY IP core.

The following tables, the variables represent these parameters:

- <n>—The number of lanes
- <d>—The serialization factor
- <s>—The symbol size
- <p>—The number of PLLs

Table 39. TX PMA Ports

Name	Direction	Clock Domain	Description
tx_serial_data[<n>-1:0]	Input	N/A	This is the serial data output of the TX PMA.
tx_serial_clk0	Input	Clock	This is the serial clock from the TX PLL. The frequency of this clock depends on the data rate and clock division factor. This clock is for non bonded channels only. For bonded channels use the tx_bonding_clocks clock TX input.
tx_bonding_clocks[<n><6>-1:0]	Input	Clock	This is a 6-bit bus which carries the low speed parallel clock per channel. These clocks are outputs from the master CGB. Use these clocks for bonded channels only.
Optional Ports			
tx_serial_clk1 tx_serial_clk2 tx_serial_clk3 tx_serial_clk4	Inputs	Clocks	These are the serial clocks from the TX PLL. The frequency of these clocks depends on the data rate and clock division factor. These additional ports are enabled when you specify more than one TX PLL.
tx_analog_reset_ack	Output	Asynchronous	Enables the optional tx_pma_analog_reset_ack output. This port should not be used for register mode data transfers
tx_pma_clkout	Output	Clock	This clock is the low speed parallel clock from the TX PMA. It is available when you turn on **Enable tx_pma_clkout** port in the Transceiver Native PHY IP core **Parameter Editor**. (19)
tx_pma_div_clkout	Output	Clock	If you specify a tx_pma_div_clkout division factor of 1 or 2, this clock output is derived from the PMA parallel clock (low speed parallel clock). If you specify a tx_pma_div_clkout division factor of 33, 40, or 66, this clock is derived from the PMA serial clock. This clock is commonly used when the interface to the TX FIFO runs at a different rate than the PMA parallel clock frequency, such as 66:40 applications.

continued...

(19) This clock is not to be used to clock the FPGA - transceiver interface. This clock may be used as a reference clock to an external clock cleaner.

Name	Direction	Clock Domain	Description
tx_pma_iqtxrx_clkout	Output	Clock	This port is available if you turn on **Enable tx_pma_iqtxrx_clkout** port in the Transceiver Native PHY IP core **Parameter Editor**. This output clock can be used to cascade the TX PMA output clock to the input of a PLL.
tx_pma_elecidle[<n>-1:0]	Input	Asynchronous	When you assert this signal, the transmitter is forced to electrical idle. This port has no effect when you configure the transceiver for the PCI Express protocol.
rx_seriallpbken[<n>-1:0]	Input	Asynchronous	This port is available if you turn on **Enable rx_seriallpbken** port in the Transceiver Native PHY IP core **Parameter Editor**. The assertion of this signal enables the TX to RX serial loopback path within the transceiver. This signal can be enabled in Duplex or Simplex mode. If enabled in Simplex mode, you must drive the signal on both the TX and RX instances from the same source. Otherwise the design fails compilation.

Table 40. RX PMA Ports

Name	Direction	Clock Domain	Description
rx_serial_data[<n>-1:0]	Input	N/A	Specifies serial data input to the RX PMA.
rx_cdr_refclk0	Input	Clock	Specifies reference clock input to the RX clock data recovery (CDR) circuitry.
Optional Ports			
rx_cdr_refclk1–rx_cdr_refclk4	Input	Clock	Specifies reference clock inputs to the RX clock data recovery (CDR) circuitry.
rx_analog_reset_ack	Output	Asynchronous	Enables the optional rx_pma_analog_reset_ack output. This port should not be used for register mode data transfers.
rx_pma_clkout	Output	Clock	This clock is the recovered parallel clock from the RX CDR circuitry.
rx_pma_div_clkout	Output	Clock	The deserializer generates this clock. This is used to drive core logic, PCS-to-FPGA fabric interface, or both. If you specify a **rx_pma_div_clkout** division factor of 1 or 2, this clock output is derived from the PMA parallel clock (low speed parallel clock). If you specify a **rx_pma_div_clkout** division factor of 33, 40, or 66, this clock is derived from the PMA serial clock. This clock is commonly used when the interface to the RX FIFO runs at a different rate than the PMA parallel clock (low speed parallel clock) frequency, such as 66:40 applications.
rx_pma_iqtxrx_clkout	Output	Clock	This port is available if you turn on **Enable rx_pma_iqtxrx_clkout port** in the Transceiver Native PHY IP core **Parameter Editor**. This output clock can be used to cascade the RX PMA output clock to the input of a PLL.
rx_pma_clkslip	Output	Clock	When asserted, indicates that the deserializer has either skipped one serial bit or paused the serial clock for one cycle to achieve word alignment. As a result, the period of the parallel clock could be extended by 1 unit interval (UI) during the clock slip operation.
rx_is_lockedtodata[<n>-1:0]	Output	rx_clkout	When asserted, indicates that the CDR PLL is locked to the incoming data, rx_serial_data.
rx_is_lockedtoref[<n>-1:0]	Output	rx_clkout	When asserted, indicates that the CDR PLL is locked to the input reference clock.

continued...

Name	Direction	Clock Domain	Description
rx_set_locktodata[<n>-1:0]	Input	Asynchronous	This port provides manual control of the RX CDR circuitry.
rx_set_locktoref[<n>-1:0]	Input	Asynchronous	This port provides manual control of the RX CDR circuitry.
rx_seriallpbken[<n>-1:0]	Input	Asynchronous	This port is available if you turn on **Enable rx_ seriallpbken port** in the Transceiver Native PHY IP core **Parameter Editor**. The assertion of this signal enables the TX to RX serial loopback path within the transceiver. This signal is enabled in Duplex or Simplex mode. If enabled in Simplex mode, you must drive the signal on both the TX and RX instances from the same source. Otherwise the design fails compilation.
rx_prbs_done[<n>-1:0]	Output	rx_coreclkin or rx_clkout	When asserted, indicates the verifier has aligned and captured consecutive PRBS patterns and the first pass through a polynomial is complete.
rx_prbs_err[<n>-1:0]	Output	rx_coreclkin or rx_clkout	When asserted, indicates an error only after the rx_prbs_done signal has been asserted. This signal gets asserted for three parallel clock cycles for every error that occurs. Errors can only occur once per word.
rx_prbs_err_clr[<n>-1:0]	Input	rx_coreclkin or rx_clkout	When asserted, clears the PRBS pattern and deasserts the rx_prbs_done signal.

Table 41. Calibration Status Ports

Name	Direction	Clock Domain	Description
tx_cal_busy[<n>-1:0]	Output	Asynchronous	When asserted, indicates that the initial TX calibration is in progress. For both initial and manual recalibration, this signal is asserted during calibration and deasserts after calibration is completed. You must hold the channel in reset until calibration completes.
rx_cal_busy[<n>-1:0]	Output	Asynchronous	When asserted, indicates that the initial RX calibration is in progress. For both initial and manual recalibration, this signal is asserted during calibration and deasserts after calibration is completed.

Table 42. Reset Ports

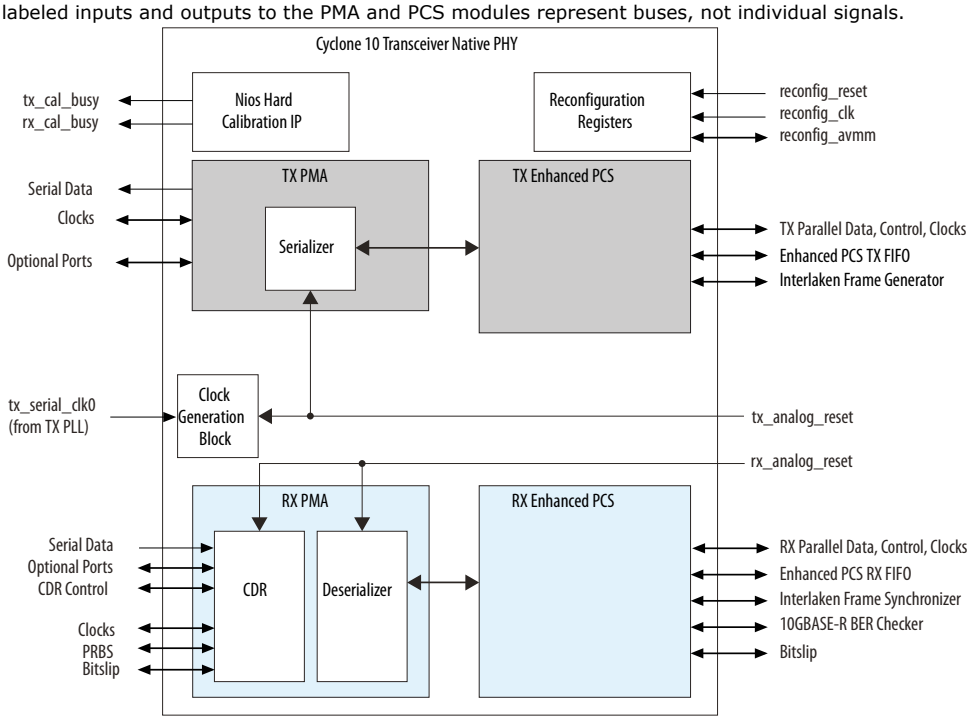
Name	Direction	Clock Domain[20]	Description
tx_analogreset[<n>-1:0]	Input	Asynchronous	Resets the analog TX portion of the transceiver PHY.
tx_digitalreset[<n>-1:0]	Input	Asynchronous	Resets the digital TX portion of the transceiver PHY.
rx_analogreset[<n>-1:0]	Input	Asynchronous	Resets the analog RX portion of the transceiver PHY.
rx_digitalreset[<n>-1:0]	Input	Asynchronous	Resets the digital RX portion of the transceiver PHY.

(20) Although the reset ports are not synchronous to any clock domain, Intel recommends that you synchronize the reset ports with the system clock.

2.4.9. Enhanced PCS Ports

Figure 13. Enhanced PCS Interfaces

The labeled inputs and outputs to the PMA and PCS modules represent buses, not individual signals.



In the following tables, the variables represent these parameters:

- <*n*>—The number of lanes
- <*d*>—The serialization factor
- <*s*>— The symbol size
- <*p*>—The number of PLLs

Table 43. Enhanced TX PCS: Parallel Data, Control, and Clocks

Name	Direction	Clock Domain	Description
`tx_parallel_data[<n>128-1:0]`	Input	Synchronous to the clock driving the write side of the FIFO (`tx_coreclkin` or `tx_clkout`)	TX parallel data inputs from the FPGA fabric to the TX PCS. If you select **Enable simplified interface** in the Transceiver Native PHY IP **Parameter Editor**, `tx_parallel_data` includes only the bits required for the configuration you specify. You must ground the data pins that are not active. For single width configuration, the following bits are active: • 32-bit FPGA fabric to PCS interface width: tx_parallel_data[31:0]. Ground [127:32]. • 40-bit FPGA fabric to PCS interface width: tx_parallel_data[39:0]. Ground [127:40]. • 64-bit FPGA fabric to PCS interface width: tx_parallel_data[63:0] Ground [127:64].

continued...

Name	Direction	Clock Domain	Description
			For double width configuration, the following bits are active: • 40-bit FPGA fabric to PCS interface width: data[103:64], [39:0]. Ground [127:104], [63:40]. • 64-bit FPGA fabric to PCS interface width: data[127:64], [63:0]. Double-width mode is not supported for 32-bit, 50-bit, and 67-bit FPGA fabric to PCS interface widths.
unused_tx_parallel_data	Input	tx_clkout	Port is enabled, when you enable **Enable simplified data interface**. Connect all of these bits to 0. When **Enable simplified data interface** is disabled, the unused bits are a part of tx_parallel_data. Refer to tx_parallel_data to identify the bits you need to ground.
tx_control[<n><3>-1:0] or tx_control[<n><18>-1:0]	Input	Synchronous to the clock driving the write side of the FIFO (tx_coreclkin or tx_clkout)	tx_control bits have different functionality depending on the transceiver configuration rule selected. When **Simplified data interface** is enabled, the number of bits in this bus changes, as the unused bits are shown as part of the unused_tx_control port. Refer to Enhanced PCS TX and RX Control Ports on page 59 section for more details.
unused_tx_control[<n> <15>-1:0]	Input	Synchronous to the clock driving the write side of the FIFO (tx_coreclkin or tx_clkout)	This port is enabled when you enable **Enable simplified data interface**. Connect all of these bits to 0. When **Enable simplified data interface** is disabled, the unused bits are a part of the tx_control. Refer to tx_control to identify the bits you need to ground.
tx_err_ins	Input	tx_coreclkin	For the Interlaken protocol, you can use this bit to insert the synchronous header and CRC32 errors if you have turned on **Enable simplified data interface**. When asserted, the synchronous header for that cycle word is replaced with a corrupted one. A CRC32 error is also inserted if **Enable Interlaken TX CRC-32 generator error insertion** is turned on. The corrupted sync header is 2'b00 for a control word, and 2'b11 for a data word. For CRC32 error insertion, the word used for CRC calculation for that cycle is incorrectly inverted, causing an incorrect CRC32 in the Diagnostic Word of the Metaframe. Note that a synchronous header error and a CRC32 error cannot be created for the Framing Control Words because the Frame Control Words are created in the frame generator embedded in TX PCS. Both the synchronous header error and the CRC32 errors are inserted if the CRC-32 error insertion feature is enabled in the Transceiver Native PHY IP GUI.
tx_coreclkin	Input	Clock	The FPGA fabric clock. Drives the write side of the TX FIFO. For the Interlaken protocol, the frequency of this clock could be from datarate/67 to datarate/32. Using frequency lower than this range can cause the TX FIFO to underflow and result in data corruption.
tx_clkout	Output	Clock	This is a parallel clock generated by the local CGB for non bonded configurations, and master CGB for bonded configurations. This clocks the blocks of the TX Enhanced PCS. The frequency of this clock is equal to the datarate divided by PCS/PMA interface width.

Table 44. Enhanced RX PCS: Parallel Data, Control, and Clocks

Name	Direction	Clock Domain	Description
`rx_parallel_data[<n>128-1:0]`	Output	Synchronous to the clock driving the read side of the FIFO (`rx_coreclkin` or `rx_clkout`)	RX parallel data from the RX PCS to the FPGA fabric. If you select, **Enable simplified data interface** in the Transceiver Native PHY IP GUI, `rx_parallel_data` includes only the bits required for the configuration you specify. Otherwise, this interface is 128 bits wide. When FPGA fabric to PCS interface width is 64 bits, the following bits are active for interfaces less than 128 bits. You can leave the unused bits floating or not connected. • 32-bit FPGA fabric to PCS width: data[31:0]. • 40-bit FPGA fabric to PCS width: data[39:0]. • 64-bit FPGA fabric to PCS width: data[63:0]. When the FPGA fabric to PCS interface width is 128 bits, the following bits are active: • 40-bit FPGA fabric to PCS width: data[103:64], [39:0]. • 64-bit FPGA fabric to PCS width: data[127:0].
`unused_rx_parallel_data`	Output	`rx_clkout`	This signal specifies the unused data when you turn on **Enable simplified data interface**. When simplified data interface is not set, the unused bits are a part of `rx_parallel_data`. You can leave the unused data outputs floating or not connected.
`rx_control[<n><20>-1:0]`	Output	Synchronous to the clock driving the read side of the FIFO (`rx_coreclkin` or `rx_clkout`)	Indicates whether the `rx_parallel_data` bus is control or data. Refer to the Enhanced PCS TX and RX Control Ports on page 59 section for more details.
`unused_rx_control[<n>10-1:0]`	Output	Synchronous to the clock driving the read side of the FIFO (`rx_coreclkin` or `rx_clkout`)	These signals only exist when you turn on **Enable simplified data interface**. When simplified data interface is not set, the unused bits are a part of `rx_control`. These outputs can be left floating.
`rx_coreclkin`	Input	Clock	The FPGA fabric clock. Drives the read side of the RX FIFO. For Interlaken protocol, the frequency of this clock could be from datarate/67 to datarate/32.
`rx_clkout`	Output	Clock	The low speed parallel clock recovered by the transceiver RX PMA, that clocks the blocks in the RX Enhanced PCS. The frequency of this clock is equal to data rate divided by PCS/PMA interface width.

Table 45. Enhanced PCS TX FIFO

Name	Direction	Clock Domain	Description
`tx_enh_data_valid[<n>-1:0]`	Input	Synchronous to the clock driving the write side of the FIFO (`tx_coreclkin` or `tx_clkout`)	Assertion of this signal indicates that the TX data is valid. Connect this signal to 1'b1 for 10GBASE-R without 1588. For 10GBASE-R with 1588, you must control this signal based on the gearbox ratio. For Basic and Interlaken, you need to control this port based on TX FIFO flags so that the FIFO does not underflow or overflow.

continued...

Name	Direction	Clock Domain	Description
			Refer to Enhanced PCS FIFO Operation on page 164 for more details.
tx_enh_fifo_full[<n>-1:0]	Output	Synchronous to the clock driving the write side of the FIFO (tx_coreclkin or tx_clkout)	Assertion of this signal indicates the TX FIFO is full. Because the depth is always constant, you can ignore this signal for the phase compensation mode. Refer to Enhanced PCS FIFO Operation on page 164 for more details.
tx_enh_fifo_pfull[<n>-1:0]	Output	Synchronous to the clock driving the write side of the FIFO tx_coreclkin or tx_clkout	This signal gets asserted when the TX FIFO reaches its partially full threshold. Because the depth is always constant, you can ignore this signal for the phase compensation mode. Refer to Enhanced PCS FIFO Operation on page 164 for more details.
tx_enh_fifo_empty[<n>-1:0]	Output	Synchronous to the clock driving the write side of the FIFO tx_coreclkin or tx_clkout	When asserted, indicates that the TX FIFO is empty. This signal gets asserted for 2 to 3 clock cycles. Because the depth is always constant, you can ignore this signal for the phase compensation mode. Refer to Enhanced PCS FIFO Operation on page 164 for more details.
tx_enh_fifo_pempty[<n>-1:0]	Output	Synchronous to the clock driving the write side of the FIFO tx_coreclkin or tx_clkout	When asserted, indicates that the TX FIFO has reached its specified partially empty threshold. When you turn this option on, the Enhanced PCS enables the tx_enh_fifo_pempty port, which is asynchronous. This signal gets asserted for 2 to 3 clock cycles. Because the depth is always constant, you can ignore this signal for the phase compensation mode. Refer to Enhanced PCS FIFO Operation on page 164 for more details.

Table 46. Enhanced PCS RX FIFO

Name	Direction	Clock Domain	Description
rx_enh_data_valid[<n>-1:0]	Output	Synchronous to the clock driving the read side of the FIFO rx_coreclkin or rx_clkout	When asserted, indicates that rx_parallel_data is valid. Discard invalid RX parallel data whenrx_enh_data_valid signal is low. This option is available when you select the following parameters: • Enhanced PCS **Transceiver configuration rules** specifies Interlaken • Enhanced PCS **Transceiver configuration rules** specifies Basic, and RX FIFO mode is **Phase compensation** • Enhanced PCS **Transceiver configuration rules** specifies Basic, and RX FIFO mode is **Register** Refer to Enhanced PCS FIFO Operation on page 164 for more details.
rx_enh_fifo_full[<n>-1:0]	Output	Synchronous to the clock driving the read side of the FIFO rx_coreclkin or rx_clkout	When asserted, indicates that the RX FIFO is full. This signal gets asserted for 2 to 3 clock cycles.Because the depth is always constant, you can ignore this signal for the phase compensation mode. Refer to Enhanced PCS FIFO Operation on page 164 for more details.
rx_enh_fifo_pfull[<n>-1:0]	Output	Synchronous to the clock driving the read side of	When asserted, indicates that the RX FIFO has reached its specified partially full threshold. This signal gets asserted for 2 to 3 clock cycles. Because the depth is always constant, you can ignore this signal for the phase compensation mode.

continued...

Name	Direction	Clock Domain	Description
		the FIFO `rx_coreclkin` or `rx_clkout`	Refer to Enhanced PCS FIFO Operation on page 164 for more details.
`rx_enh_fifo_empty[<n>-1:0]`	Output	Synchronous to the clock driving the read side of the FIFO `rx_coreclkin` or `rx_clkout`	When asserted, indicates that the RX FIFO is empty. Because the depth is always constant, you can ignore this signal for the phase compensation mode. Refer to Enhanced PCS FIFO Operation on page 164 for more details.
`rx_enh_fifo_pempty[<n>-1:0]`	Output	Synchronous to the clock driving the read side of the FIFO `rx_coreclkin` or `rx_clkout`	When asserted, indicates that the RX FIFO has reached its specified partially empty threshold. Because the depth is always constant, you can ignore this signal for the phase compensation mode. Refer to Enhanced PCS FIFO Operation on page 164 for more details.
`rx_enh_fifo_del[<n>-1:0]`	Output	Synchronous to the clock driving the read side of the FIFO `rx_coreclkin` or `rx_clkout`	When asserted, indicates that a word has been deleted from the RX FIFO. This signal gets asserted for 2 to 3 clock cycles. This signal is used for the 10GBASE-R protocol.
`rx_enh_fifo_insert[<n>-1:0]`	Output	Synchronous to the clock driving the read side of the FIFO `rx_coreclkin` or `rx_clkout`	When asserted, indicates that a word has been inserted into the RX FIFO. This signal is used for the 10GBASE-R protocol.
`rx_enh_fifo_rd_en[<n>-1:0]`	Output	Synchronous to the clock driving the read side of the FIFO `rx_coreclkin` or `rx_clkout`	For Interlaken only, when this signal is asserted, a word is read form the RX FIFO. You need to control this signal based on RX FIFO flags so that the FIFO does not underflow or overflow.
`rx_enh_fifo_align_val[<n>-1:0]`	Input	Synchronous to the clock driving the read side of the FIFO `rx_coreclkin` or `rx_clkout`	When asserted, indicates that the word alignment pattern has been found. This signal is only valid for the Interlaken protocol.
`rx_enh_fifo_align_clr[<n>-1:0]`	Input	Synchronous to the clock driving the read side of the FIFO `rx_coreclkin` or `rx_clkout`	When asserted, the FIFO resets and begins searching for a new alignment pattern. This signal is only valid for the Interlaken protocol. Assert this signal for at least 4 cycles.

Table 47. Interlaken Frame Generator, Synchronizer, and CRC32

Name	Direction	Clock Domain	Description
`tx_enh_frame[<n>-1:0]`	Output	`tx_clkout`	Asserted for 2 or 3 parallel clock cycles to indicate the beginning of a new metaframe.
`tx_enh_frame_diag_status[<n> 2-1:0]`	Input	`tx_clkout`	Drives the lane status message contained in the framing layer diagnostic word (bits[33:32]). This message is inserted into the next diagnostic word generated by the

continued...

Name	Direction	Clock Domain	Description
			frame generator block. This bus must be held constant for 5 clock cycles before and after the tx_enh_frame pulse. The following encodings are defined: • Bit[1]: When 1, indicates the lane is operational. When 0, indicates the lane is not operational. • Bit[0]: When 1, indicates the link is operational. When 0, indicates the link is not operational.
tx_enh_frame_burst_en[<n>-1:0]	Input	tx_clkout	If **Enable frame burst** is enabled, this port controls frame generator data reads from the TX FIFO to the frame generator. It is latched once at the beginning of each Metaframe. If the value of tx_enh_frame_burst_en is 0, the frame generator does not read data from the TX FIFO for current Metaframe. Instead, the frame generator inserts SKIP words as the payload of Metaframe. When tx_enh_frame_burst_en is 1, the frame generator reads data from the TX FIFO for the current Metaframe. This port must be held constant for 5 clock cycles before and after the tx_enh_frame pulse.
rx_enh_frame[<n>-1:0]	Output	rx_clkout	When asserted, indicates the beginning of a new received Metaframe. This signal is pulse stretched.
rx_enh_frame_lock[<n>-1:0]	Output	rx_clkout	When asserted, indicates the Frame Synchronizer state machine has achieved Metaframe delineation. This signal is pulse stretched.
rx_enh_frame_diag_status[2 <n>-1:0]	Output	rx_clkout	Drives the lane status message contained in the framing layer diagnostic word (bits[33:32]). This signal is latched when a valid diagnostic word is received in the end of the Metaframe while the frame is locked. The following encodings are defined: • Bit[1]: When 1, indicates the lane is operational. When 0, indicates the lane is not operational. • Bit[0]: When 1, indicates the link is operational. When 0, indicates the link is not operational.
rx_enh_crc32_err[<n>-1:0]	Output	rx_clkout	When asserted, indicates a CRC error in the current Metaframe. Asserted at the end of current Metaframe. This signal gets asserted for 2 or 3 cycles.

Table 48. 10GBASE-R BER Checker

Name	Direction	Clock Domain	Description
rx_enh_highber[<n>-1:0]	Output	rx_clkout	When asserted, indicates a bit error rate that is greater than 10^{-4}. For the 10GBASE-R protocol, this BER rate occurs when there are at least 16 errors within 125 µs. This signal gets asserted for 2 to 3 clock cycles.
rx_enh_highber_clr_cnt[<n>-1:0]	Input	rx_clkout	When asserted, clears the internal counter that indicates the number of times the BER state machine has entered the BER_BAD_SH state.
rx_enh_clr_errblk_count[<n>-1:0] (10GBASE-R)	Input	rx_clkout	When asserted the error block counter resets to 0. Assertion of this signal clears the internal counter that counts the number of times the RX state machine has entered the RX_E state.

Table 49. Block Synchronizer

Name	Direction	Clock Domain	Description
rx_enh_blk_lock<n>-1:0]	Output	rx_clkout	When asserted, indicates that block synchronizer has achieved block delineation. This signal is used for 10GBASE-R and Interlaken.

Table 50. Gearbox

Name	Direction	Clock Domain	Description
`rx_bitslip[<n>-1:0]`	Input	`rx_clkout`	The `rx_parallel_data` slips 1 bit for every positive edge of the `rx_bitslip` input. Keep the minimum interval between `rx_bitslip` pulses to at least 20 cycles. The maximum shift is < *pcswidth -1*> bits, so that if the PCS is 64 bits wide, you can shift 0-63 bits.
`tx_enh_bitslip[<n>-1:0]`	Input	`rx_clkout`	The value of this signal controls the number of bits to slip the `tx_parallel_data` before passing to the PMA.

2.4.9.1. Enhanced PCS TX and RX Control Ports

This section describes the `tx_control` and `rx_control` bit encodings for different protocol configurations.

When Enable simplified data interface is ON, all of the unused ports shown in the tables below, appear as a separate port. For example: It appears as `unused_tx_control`/ `unused_rx_control` port.

Enhanced PCS TX Control Port Bit Encodings

Table 51. Bit Encodings for Interlaken

Name	Bit	Functionality	Description
`tx_control`	[1:0]	Synchronous header	The value 2'b01 indicates a data word. The value 2'b10 indicates a control word.
	[2]	Inversion control	A logic low indicates that the built-in disparity generator block in the Enhanced PCS maintains the Interlaken running disparity.
	[7:3]	Unused	
	[8]	Insert synchronous header error or CRC32	You can use this bit to insert synchronous header error or CRC32 errors. The functionality is similar to `tx_err_ins`. Refer to `tx_err_ins` signal description for more details.
	[17:9]	Unused	

Table 52. Bit Encodings for 10GBASE-R

Name	Bit	Functionality
`tx_control`	[0]	XGMII control signal for `parallel_data[7:0]`
	[1]	XGMII control signal for `parallel_data[15:8]`
	[2]	XGMII control signal for `parallel_data[23:16]`
	[3]	XGMII control signal for `parallel_data[31:24]`
	[4]	XGMII control signal for `parallel_data[39:32]`
	[5]	XGMII control signal for `parallel_data[47:40]`
	[6]	XGMII control signal for `parallel_data[55:48]`
	[7]	XGMII control signal for `parallel_data[63:56]`
	[17:8]	Unused

Table 53. Bit Encodings for Basic Single Width Mode

For basic single width mode, the total word length is 66-bit with 64-bit data and 2-bit synchronous header.

Name	Bit	Functionality	Description
tx_control	[1:0]	Synchronous header	The value 2'b01 indicates a data word. The value 2'b10 indicates a control word.
	[17:2]	Unused	

Table 54. Bit Encodings for Basic Double Width Mode

For basic double width mode, the total word length is 66-bit with 128-bit data and 4-bit synchronous header.

Name	Bit	Functionality	Description
tx_control	[1:0]	Synchronous header	The value 2'b01 indicates a data word. The value 2'b10 indicates a control word.
	[8:2]	Unused	
	[10:9]	Synchronous header	The value 2'b01 indicates a data word. The value 2'b10 indicates a control word.
	[17:11]	Unused	

Table 55. Bit Encodings for Basic Mode

In this case, the total word length is 67-bit with 64-bit data and 2-bit synchronous header.

Name	Bit	Functionality	Description
tx_control	[1:0]	Synchronous header	The value 2'b01 indicates a data word. The value 2'b10 indicates a control word.
	[2]	Inversion control	A logic low indicates that built-in disparity generator block in the Enhanced PCS maintains the running disparity.

Enhanced PCS RX Control Port Bit Encodings

Table 56. Bit Encodings for Interlaken

Name	Bit	Functionality	Description
rx_control	[1:0]	Synchronous header	The value 2'b01 indicates a data word. The value 2'b10 indicates a control word.
	[2]	Inversion control	A logic low indicates that the built-in disparity generator block in the Enhanced PCS maintains the Interlaken running disparity. In the current implementation, this bit is always tied logic low (1'b0).
	[3]	Payload word location	A logic high (1'b1) indicates the payload word location in a metaframe.
	[4]	Synchronization word location	A logic high (1'b1) indicates the synchronization word location in a metaframe.
	[5]	Scrambler state word location	A logic high (1'b1) indicates the scrambler word location in a metaframe.

continued...

Name	Bit	Functionality	Description
	[6]	SKIP word location	A logic high (1'b1) indicates the SKIP word location in a metaframe.
	[7]	Diagnostic word location	A logic high (1'b1) indicates the diagnostic word location in a metaframe.
	[8]	Synchronization header error, metaframe error, or CRC32 error status	A logic high (1'b1) indicates synchronization header error, metaframe error, or CRC32 error status.
	[9]	Block lock and frame lock status	A logic high (1'b1) indicates that block lock and frame lock have been achieved.
	[19:10]	Unused	

Table 57. Bit Encodings for 10GBASE-R

Name	Bit	Functionality
rx_control	[0]	XGMII control signal for parallel_data[7:0]
	[1]	XGMII control signal for parallel_data[15:8]
	[2]	XGMII control signal for parallel_data[23:16]
	[3]	XGMII control signal for parallel_data[31:24]
	[4]	XGMII control signal for parallel_data[39:32]
	[5]	XGMII control signal for parallel_data[47:40]
	[6]	XGMII control signal for parallel_data[55:48]
	[7]	XGMII control signal for parallel_data[63:56]
	[19:8]	Unused

Table 58. Bit Encodings for Basic Single Width Mode

For basic single width mode, the total word length is 66-bit with 64-bit data and 2-bit synchronous header.

Name	Bit	Functionality	Description
rx_control	[1:0]	Synchronous header	The value 2'b01 indicates a data word. The value 2'b10 indicates a control word.
	[7:2]	Unused	
	[9:8]	Synchronous header error status	The value 2'b01 indicates a data word. The value 2'b10 indicates a control word.
	[19:10]	Unused	

Table 59. Bit Encodings for Basic Double Width Mode

For basic double width mode, total word length is 66-bit with 128-bit data, and 4-bit synchronous header.

Name	Bit	Functionality	Description
rx_control	[1:0]	Synchronous header	The value 2'b01 indicates a data word. The value 2'b10 indicates a control word.
	[7:2]	Unused	

continued...

Name	Bit	Functionality	Description
	[8]	Synchronous header error status	Active-high status signal that indicates a synchronous header error.
	[9]	Block lock is achieved	Active-high status signal indicating when block lock is achieved.
	[11:10]	Synchronous header	The value 2'b01 indicates a data word. The value 2'b10 indicates a control word.
	[17:12]	Unused	
	[18]	Synchronous header error status	Active-high status signal that indicates a synchronous header error.
	[19]	Block lock is achieved	Active-high status signal indicating when Block Lock is achieved.

Table 60. Bit Encodings for Basic Mode

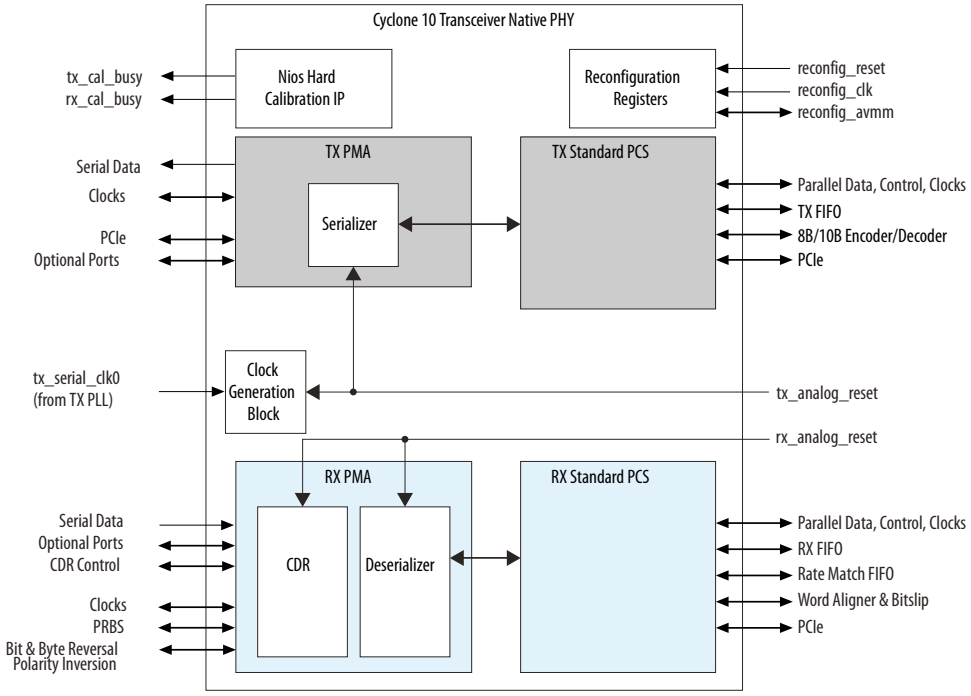
In this case, the total word length is 67-bit with 64-bit data and 2-bit synchronous header.

Name	Bit	Functionality	Description
rx_control	[1:0]	Synchronous header	The value 2'b01 indicates a data word. The value 2'b10 indicates a control word.
	[2]	Inversion control	A logic low indicates that built-in disparity generator block in the Enhanced PCS maintains the running disparity.

2.4.10. Standard PCS Ports

Figure 14. Transceiver Channel using the Standard PCS Ports

Standard PCS ports appears, if either one of the transceiver configuration modes is selected that uses Standard PCS or if **Data Path Reconfiguration** is selected even if the transceiver configuration is not one of those that uses Standard PCS.



In the following tables, the variables represent these parameters:

- <*n*>—The number of lanes
- <*w*>—The width of the interface
- <*d*>—The serialization factor
- <*s*>— The symbol size
- <*p*>—The number of PLLs

Table 61. TX Standard PCS: Data, Control, and Clocks

Name	Direction	Clock Domain	Description
`tx_parallel_data[<n>128-1:0]`	Input	`tx_clkout`	TX parallel data input from the FPGA fabric to the TX PCS.
`unused_tx_parallel_data`	Input	`tx_clkout`	This signal specifies the unused data when you turn on **Enable simplified data interface**. When simplified data interface is not set, the unused bits are a part of `tx_parallel_data`. Connect all these bits to 0. If you do not connect the unused data bits to 0, then TX parallel data may not be serialized correctly by the Native PHY IP core.
`tx_coreclkin`	Input	Clock	The FPGA fabric clock. This clock drives the write port of the TX FIFO.
`tx_clkout`	Output	Clock	This is the parallel clock generated by the local CGB for non bonded configurations, and master CGB for bonded configuration. This clocks the `tx_parallel_data` from the FPGA fabric to the TX PCS.

Table 62. RX Standard PCS: Data, Control, Status, and Clocks

Name	Direction	Clock Domain	Description
`rx_parallel_data[<n>128-1:0]`	Output	Synchronous to the clock driving the read side of the FIFO (`rx_coreclkin` or `rx_clkout`)	RX parallel data from the RX PCS to the FPGA fabric. For each 128-bit word of `rx_parallel_data`, the data bits correspond to `rx_parallel_data[7:0]` when 8B/10B decoder is enabled and `rx_parallel_data[9:0]` when 8B/10B decoder is disabled.
`unused_rx_parallel_data`	Output	Synchronous to the clock driving the read side of the FIFO (`rx_coreclkin` or `rx_clkout`)	This signal specifies the unused data when you turn on **Enable simplified data interface**. When simplified data interface is not set, the unused bits are a part of `rx_parallel_data`. These outputs can be left floating.
`rx_clkout`	Output	Clock	The low speed parallel clock recovered by the transceiver RX PMA, that clocks the blocks in the RX Standard PCS.
`rx_coreclkin`	Input	Clock	RX parallel clock that drives the read side clock of the RX FIFO.

Table 63. Standard PCS FIFO

Name	Direction	Clock Domain	Description
tx_std_pcfifo_full[<n>-1:0]	Output	Synchronous to the clock driving the write side of the FIFO (tx_coreclkin or tx_clkout)	Indicates when the standard TX FIFO is full.
tx_std_pcfifo_empty[<n>-1:0]	Output	Synchronous to the clock driving the write side of the FIFO (tx_coreclkin or tx_clkout)	Indicates when the standard TX FIFO is empty.
rx_std_pcfifo_full[<n>-1:0]	Output	Synchronous to the clock driving the read side of the FIFO (rx_coreclkin or rx_clkout)	Indicates when the standard RX FIFO is full.
rx_std_pcfifo_empty[<n>-1:0]	Output	Synchronous to the clock driving the read side of the FIFO (rx_coreclkin or rx_clkout)	Indicates when the standard RX FIFO is empty.

Table 64. Rate Match FIFO

Name	Direction	Clock Domain	Description
rx_std_rmfifo_full[<n>-1:0]	Output	Asynchronous	Rate match FIFO full flag. When asserted the rate match FIFO is full. You must synchronize this signal. This port is only used for GigE mode.
rx_std_rmfifo_empty[<n>-1:0]	Output	Asynchronous	Rate match FIFO empty flag. When asserted, match FIFO is empty. You must synchronize this signal. This port is only used for GigE mode.
rx_rmfifostatus[<n>-1:0]	Output	Asynchronous	Indicates FIFO status. The following encodings are defined: • 2'b00: Normal operation • 2'b01: Deletion, rx_std_rmfifo_full = 1 • 2'b10: Insertion, rx_std_rmfifo_empty = 1 • 2'b11: Full. rx_rmfifostatus is a part of rx_parallel_data. rx_rmfifostatus corresponds to rx_parallel_data[14:13].

Table 65. 8B/10B Encoder and Decoder

Name	Direction	Clock Domain	Description
tx_datak	Input	tx_clkout	tx_datak is exposed if 8B/10B enabled and simplified data interface is set.When 1, indicates that the 8B/10B encoded word of tx_parallel_data is control. When 0, indicates that the 8B/10B encoded word of tx_parallel_data is data. tx_datak is a part of tx_parallel_data when simplified data interface is not set.
tx_forcedisp[<n>(<w>/<s>-1:0]	Input	Asynchronous	This signal allows you to force the disparity of the 8B/10B encoder. When "1", forces the disparity of the output data to the value driven on tx_dispval. When "0", the current running disparity continues. tx_forcedisp is a part of tx_parallel_data. tx_forcedisp corresponds to tx_parallel_data[9].
tx_dispval[<n>(<w>/<s>-1:0]	Input	Asynchronous	Specifies the disparity of the data. When 0, indicates positive disparity, and when 1, indicates negative disparity. tx_dispval is a part of tx_parallel_data. tx_dispval corresponds to tx_dispval[10].
rx_datak[<n><w>/<s>-1:0]	Output	rx_clkout	rx_datak is exposed if 8B/10B is enabled and simplified data interface is set. When 1, indicates that the 8B/10B decoded word of rx_parallel_data is control. When 0, indicates that the 8B/10B decoded word of rx_parallel_data is data. rx_datak is a part of rx_parallel_data when simplified data interface is not set.
rx_errdetect[<n><w>/<s>-1:0]	Output	Synchronous to the clock driving the read side of the FIFO (rx_coreclkin or rx_clkout)	When asserted, indicates a code group violation detected on the received code group. Used along with rx_disperr signal to differentiate between code group violation and disparity errors. The following encodings are defined for rx_errdetect/rx_disperr: • 2'b00: no error • 2'b10: code group violation • 2'b11: disparity error. rx_errdetect is a part of rx_parallel_data. For each 128-bit word, rx_errdetect corresponds to rx_parallel_data[9].
rx_disperr[<n><w>/<s>-1:0]	Output	Synchronous to the clock driving the read side of the FIFO (rx_coreclkin or rx_clkout)	When asserted, indicates a disparity error on the received code group. rx_disperr is a part of rx_parallel_data. For each 128-bit word, rx_disperr corresponds to rx_parallel_data[11].
rx_runningdisp[<n><w>/<s>-1:0]	Output	Synchronous to the clock driving the read side of the FIFO (rx_coreclkin or rx_clkout)	When high, indicates that rx_parallel_data was received with negative disparity. When low, indicates that rx_parallel_data was received with positive disparity. rx_runningdisp is a part of rx_parallel_data. For each 128 bit word, rx_runningdisp corresponds to rx_parallel_data[15].
rx_patterndetect[<n><w>/<s>-1:0]	Output	Asynchronous	When asserted, indicates that the programmed word alignment pattern has been detected in the current word boundary. rx_patterndetect is a part of

continued...

Name	Direction	Clock Domain	Description
			rx_parallel_data. For each 128-bit word, rx_patterndetect corresponds to rx_parallel_data[12].
rx_syncstatus[<n><w>/<s>-1:0]	Output	Asynchronous	When asserted, indicates that the conditions required for synchronization are being met. rx_syncstatus is a part of rx_parallel_data. For each 128-bit word, rx_syncstatus corresponds to rx_parallel_data[10].

Table 66. Word Aligner and Bitslip

Name	Direction	Clock Domain	Description
tx_std_bitslipboundarysel[5 <n>-1:0]	Input	Asynchronous	Bitslip boundary selection signal. Specifies the number of bits that the TX bit slipper must slip.
rx_std_bitslipboundarysel[5 <n>-1:0]	Output	Asynchronous	This port is used in deterministic latency word aligner mode. This port reports the number of bits that the RX block slipped. This port values should be taken into consideration in either Deterministic Latency Mode or Manual Mode of Word Aligner.
rx_std_wa_patternalign[<n>-1:0]	Input	Synchronous to rx_clkout	Active when you place the word aligner in manual mode. In manual mode, you align words by asserting rx_std_wa_patternalign. When the PCS-PMA Interface width is 10 bits, rx_std_wa_patternalign is level sensitive. For all the other PCS-PMA Interface widths, rx_std_wa_patternalign is positive edge sensitive. You can use this port only when the word aligner is configured in manual or deterministic latency mode. When the word aligner is in manual mode, and the PCS-PMA interface width is 10 bits, this is a level sensitive signal. In this case, the word aligner monitors the input data for the word alignment pattern, and updates the word boundary when it finds the alignment pattern. For all other PCS-PMA interface widths, this signal is edge sensitive.This signal is internally synchronized inside the PCS using the PCS parallel clock and should be asserted for at least 2 clock cycles to allow synchronization.
rx_std_wa_a1a2size[<n>-1:0]	Input	Asynchronous	Used for the SONET protocol. Assert when the A1 and A2 framing bytes must be detected. A1 and A2 are SONET backplane bytes and are only used when the PMA data width is 8 bits.
rx_bitslip[<n>-1:0]	Input	Asynchronous	Used when word aligner mode is bitslip mode. When the Word Aligner is in either Manual (PLD controlled), Synchronous State Machine or Deterministic Latency ,the rx_bitslip signal is not valid and should be tied to 0. For every rising edge of the rx_std_bitslip signal, the word boundary is shifted by 1 bit. Each bitslip removes the earliest received bit from the received data.

Table 67. Bit Reversal and Polarity Inversion

Name	Direction	Clock Domain	Description
rx_std_byterev_ena[<n>-1:0]	Input	Asynchronous	This control signal is available when the PMA width is 16 or 20 bits. When asserted, enables byte reversal on the RX interface. Used if the MSB and LSB of the transmitted data are erroneously swapped.
rx_std_bitrev_ena[<n>-1:0]	Input	Asynchronous	When asserted, enables bit reversal on the RX interface. Bit order may be reversed if external transmission circuitry transmits the most significant bit first. When enabled, the

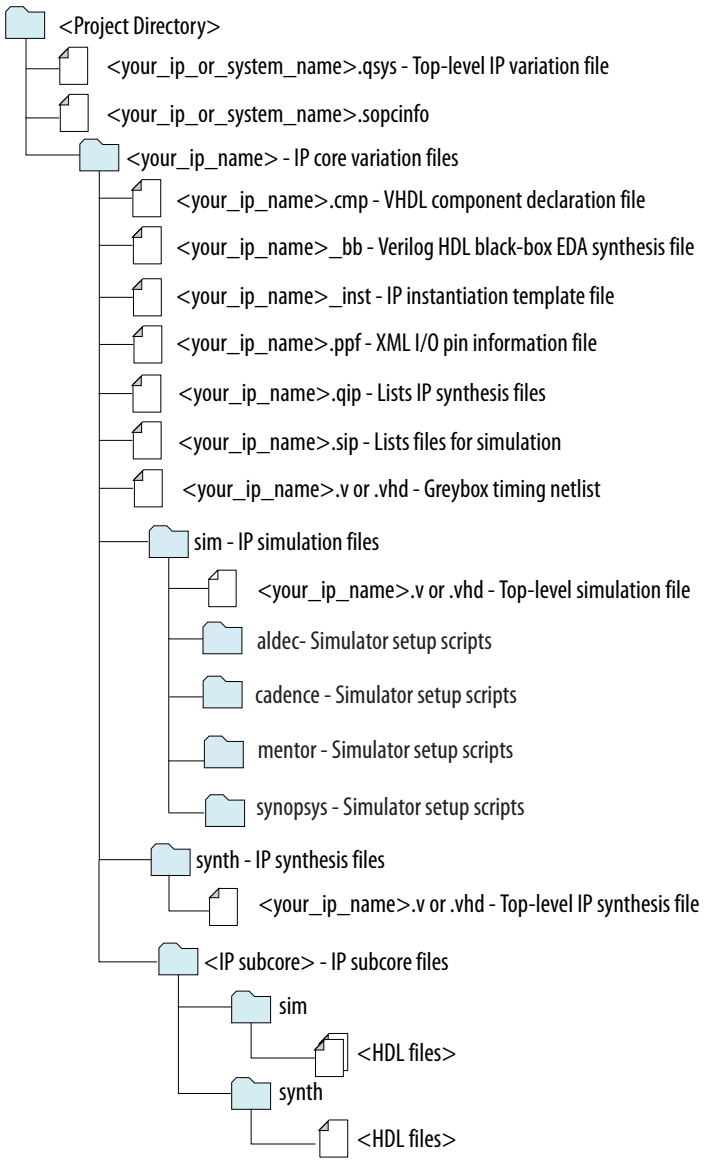
continued...

Name	Direction	Clock Domain	Description
			receive circuitry receives all words in the reverse order. The bit reversal circuitry operates on the output of the word aligner.
`tx_polinv[<n>-1:0]`	Input	Asynchronous	When asserted, the TX polarity bit is inverted. Only active when TX bit polarity inversion is enabled.
`rx_polinv[<n>-1:0]`	Input	Asynchronous	When asserted, the RX polarity bit is inverted. Only active when RX bit polarity inversion is enabled.
`rx_std_signaldetect[<n>-1:0]`	Output	Asynchronous	When enabled, the signal threshold detection circuitry senses whether the signal level present at the RX input buffer is above the signal detect threshold voltage. You can specify the signal detect threshold using a Quartus Prime Settings File (.qsf) assignment. This signal is required for the PCI Express, SATA and SAS protocols.

2.4.11. IP Core File Locations

When you generate your Transceiver Native PHY IP, the Quartus® Prime software generates the HDL files that define your instance of the IP. In addition, the Quartus Prime software generates an example Tcl script to compile and simulate your design in the ModelSim simulator. It also generates simulation scripts for Synopsys VCS, Aldec Active-HDL, Aldec Riviera-Pro, and Cadence Incisive Enterprise.

Figure 15. Directory Structure for Generated Files



The following table describes the directories and the most important files for the parameterized Transceiver Native PHY IP core and the simulation environment. These files are in clear text.

Table 68. Transceiver Native PHY Files and Directories

File Name	Description
<project_dir>	The top-level project directory.
<your_ip_name> **.v** or **.vhd**	The top-level design file.
<your_ip_name> **.qip**	A list of all files necessary for Quartus Prime compilation.

continued...

File Name	Description
<your_ip_name> **.bsf**	A Block Symbol File (.bsf) for your Transceiver Native PHY instance.
<project_dir>/<your_ip_name>/	The directory that stores the HDL files that define the Transceiver Native PHY IP.
<project_dir>/sim	The simulation directory.
*<project_dir>/sim/**aldec***	Simulation files for Riviera-PRO simulation tools.
*<project_dir>/sim/**cadence***	Simulation files for Cadence simulation tools.
*<project_dir>/sim/**mentor***	Simulation files for Mentor simulation tools.
*<project_dir>/sim/**synopsys***	Simulation files for Synopsys simulation tools.
<project_dir>/synth	The directory that stores files used for synthesis.

The Verilog and VHDL Transceiver Native PHY IP cores have been tested with the following simulators:

- ModelSim SE
- Synopsys VCS MX
- Cadence NCSim

If you select VHDL for your transceiver PHY, only the wrapper generated by the Quartus Prime software is in VHDL. All the underlying files are written in Verilog or SystemVerilog. To enable simulation using a VHDL-only ModelSim license, the underlying Verilog and SystemVerilog files for the Transceiver Native PHY IP are encrypted so that they can be used with the top-level VHDL wrapper without using a mixed-language simulator.

For more information about simulating with ModelSim, refer to the *Mentor Graphics ModelSim Support* chapter in volume 3 of the *Quartus Prime Handbook*.

The Transceiver Native PHY IP cores do not support the NativeLink feature in the Quartus Prime software.

Related Information

Mentor Graphics ModelSim Support

2.4.12. Unused Transceiver Channels

To prevent performance degradation of unused transceiver channels over time, the following assignments for RX pins must be added to an Cyclone 10 GX device QSF. You can either use a global assignment or per-pin assignment. For the per-pin assignment, true or complement RX pin can be specified.

```
set_global_assignment -name PRESERVE_UNUSED_XCVR_CHANNEL ON
```
or

```
set_instance_assignment -name PRESERVE_UNUSED_XCVR_CHANNEL ON -to
[pin_name (BB6, for example)]
```

Example of <pin_name> is AF26 (Do not use PIN_AF26)

```
set_instance_assignment -name PRESERVE_UNUSED_XCVR_CHANNEL ON -to
AF26
```

Note: This assignment applies to either an RX or a TX pin. If you assign it to both, the fitter fails.

2.5. Interlaken

Interlaken is a scalable, channelized chip-to-chip interconnect protocol.

The key advantages of Interlaken are scalability and low I/O count compared to earlier protocols such as SPI 4.2. Other key features include flow control, low overhead framing, and extensive integrity checking. Interlaken operates on 64-bit data words and 3 control bits, which are striped round-robin across the lanes. The protocol accepts packets on 256 logical channels and is expandable to accommodate up to 65,536 logical channels. Packets are split into small bursts that can optionally be interleaved. The burst semantics include integrity checking and per logical channel flow control.

The Interlaken interface is supported with 1 to 12 lanes running at data rates up to 12.5 Gbps per lane on Cyclone 10 GX devices. Interlaken is implemented using the Enhanced PCS. The Enhanced PCS has demonstrated interoperability with Interlaken ASSP vendors and third-party IP suppliers.

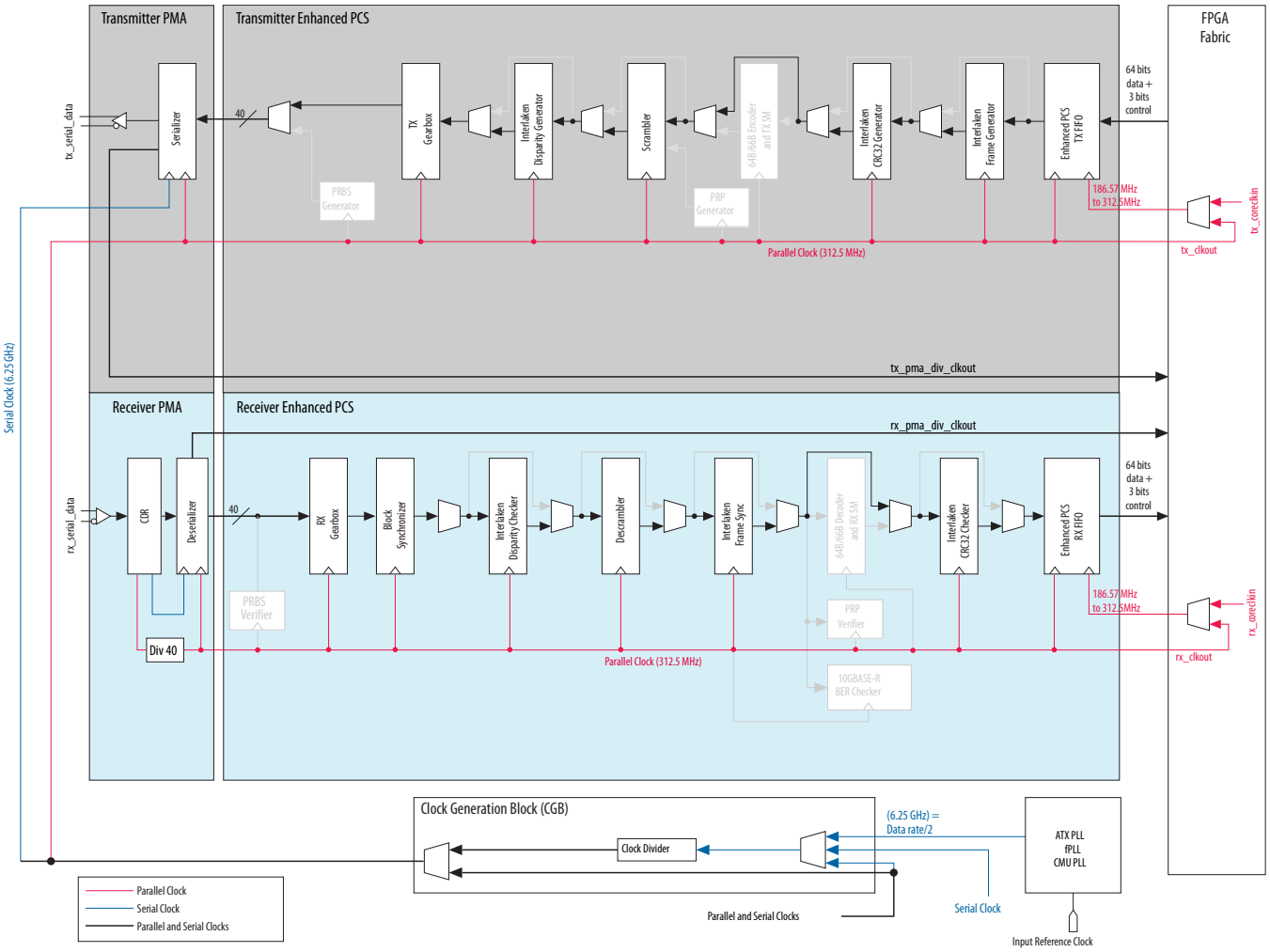
Cyclone 10 GX devices provide three preset variations for Interlaken in the Cyclone 10 GX Transceiver Native PHY IP Parameter Editor:

- Interlaken 10x12.5 Gbps
- Interlaken 1x6.25 Gbps
- Interlaken 6x10.3 Gbps

Depending on the line rate, the enhanced PCS can use a PMA to PCS interface width of 32, 40, or 64 bits.

Figure 16. Transceiver Channel Datapath and Clocking for Interlaken

This figure assumes the serial data rate is 12.5 Gbps and the PMA width is 40 bits.



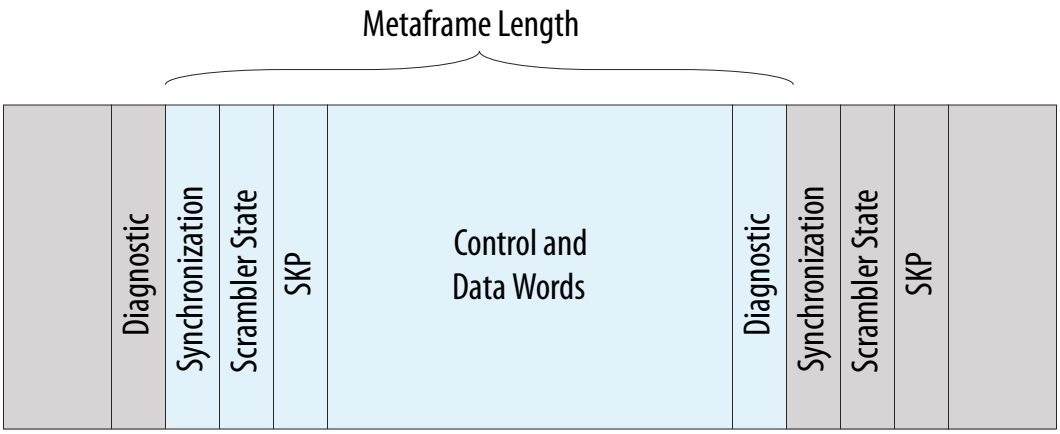
Related Information

- Interlaken Protocol Definition v1.2
- Interlaken Look-Aside Protocol Definition, v1.1

2.5.1. Metaframe Format and Framing Layer Control Word

The Enhanced PCS supports programmable metaframe lengths from 5 to 8192 words. However, for stability and performance, Intel recommends you set the frame length to no less than 128 words. In simulation, use a smaller metaframe length to reduce simulation times. The payload of a metaframe could be pure data payload and a Burst/Idle control word from the MAC layer.

Figure 17. Framing Layer Metaframe Format



The framing control words include:

- Synchronization (SYNC)—for frame delineation and lane alignment (deskew)
- Scrambler State (SCRM)—to synchronize the scrambler
- Skip (SKIP)—for clock compensation in a repeater
- Diagnostic (DIAG)—provides per-lane error check and optional status message

To form a metaframe, the Enhanced PCS frame generator inserts the framing control words and encapsulates the control and data words read from the TX FIFO as the metaframe payload.

Figure 18. Interlaken Synchronization and Scrambler State Words Format

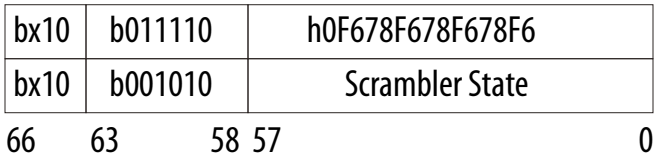
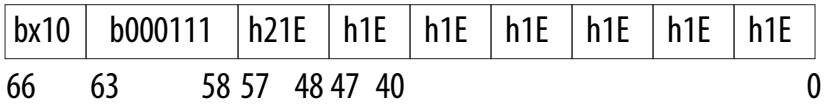


Figure 19. Interlaken Skip Word Format

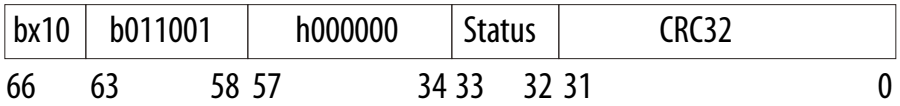


The DIAG word is comprised of a status field and a CRC-32 field. The 2-bit status is defined by the Interlaken specification as:

- Bit 1 (Bit 33): Lane health
  - 1: Lane is healthy
  - 0: Lane is not healthy
- Bit 0 (Bit 32): Link health
  - 1: Link is healthy
  - 0: Link is not healthy

The `tx_enh_frame_diag_status[1:0]` input from the FPGA fabric is inserted into the Status field each time a DIAG word is created by the framing generator.

Figure 20. Interlaken Diagnostic Word



2.5.2. Interlaken Configuration Clocking and Bonding

The Cyclone 10 GX Interlaken PHY layer solution is scalable and has flexible data rates. You can implement a single lane link or bond up to 48 lanes together. You can choose a lane data rate up to 12.5 Gbps. You can also choose between different reference clock frequencies, depending on the PLL used to clock the transceiver.

You can use an ATX PLL or fPLL to provide the clock for the transmit channel. An ATX PLL has better jitter performance compared to an fPLL. You can use the CMU PLL to clock only the non-bonded Interlaken transmit channels. However, if you use the CMU PLL, you lose one RX transceiver channel.

For the multi-lane Interlaken interface, TX channels are usually bonded together to minimize the transmit skew between all bonded channels. Currently, xN bonding and PLL feedback compensation bonding schemes are available to support a multi-lane Interlaken implementation. If the system tolerates higher channel-to-channel skew, you can choose to not bond the TX channels.

To implement bonded multi-channel Interlaken, all channels must be placed contiguously. The channels may all be placed in one bank (if not greater than six lanes) or they may span several banks.

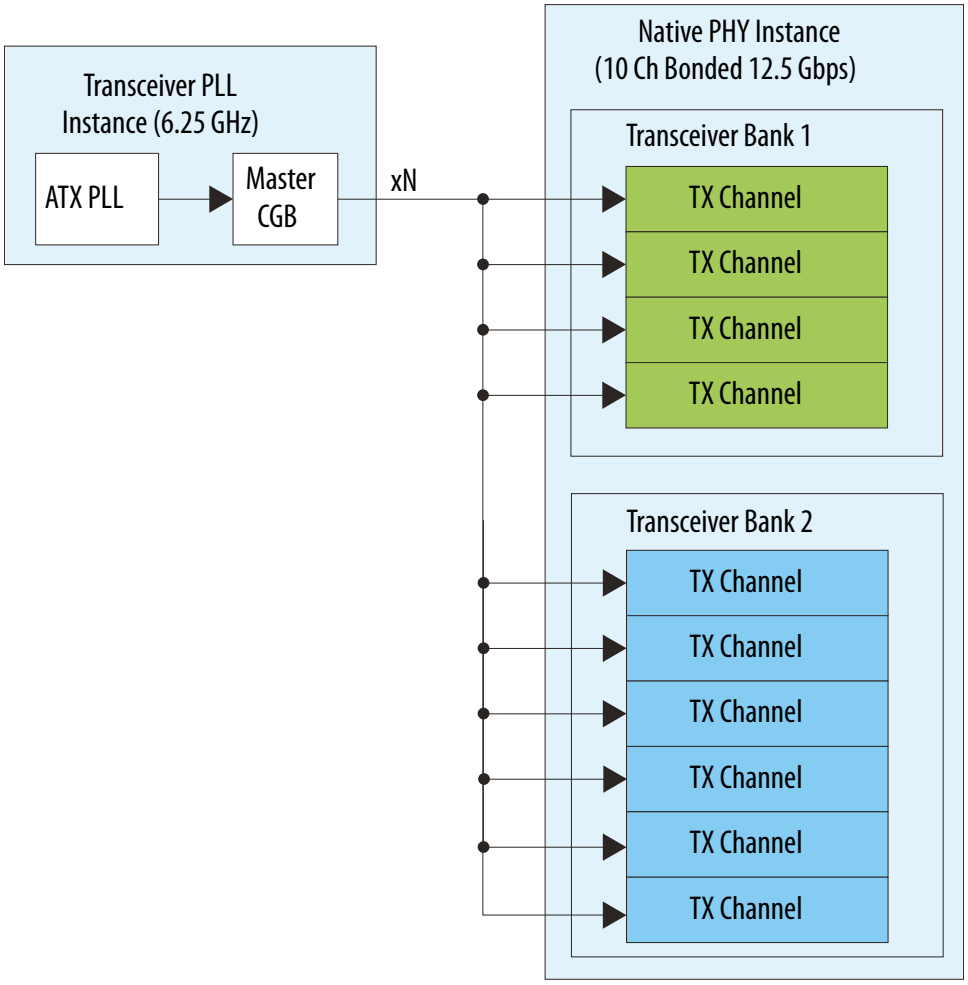
Related Information

Using PLLs and Clock Networks on page 231
For more information about implementing PLLs and clocks

2.5.2.1. xN Clock Bonding Scenario

The following figure shows a xN bonding example supporting 10 lanes. Each lane is running at 12.5 Gbps. The first six TX channels reside in one transceiver bank and the other four TX channels reside in the adjacent transceiver bank. The ATX PLL provides the serial clock to the master CGB. The CGB then provides parallel and serial clocks to all of the TX channels inside the same bank and other banks through the xN clock network.

Figure 21. 10X12.5 Gbps xN Bonding



Note: Intel Cyclone 10 GX devices have transceiver channels that can support data rates up to 12.5 Gbps for chip-to-chip and chip-to-module communication, and up to 6.6 Gbps for backplane communication.

Related Information

- Implementing x6/xN Bonding Mode on page 236
  For detailed information on xN bonding limitations
- Using PLLs and Clock Networks on page 231
  For more information about implementing PLLs and clocks

2.5.2.2. TX Multi-Lane Bonding and RX Multi-Lane Deskew Alignment State Machine

The Interlaken configuration sets the enhanced PCS TX and RX FIFOs in Interlaken elastic buffer mode. In this mode of operation, TX and RX FIFO control and status port signals are provided to the FPGA fabric. Connect these signals to the MAC layer as

required by the protocol. Based on these FIFO status and control signals, you can implement the multi-lane deskew alignment state machine in the FPGA fabric to control the transceiver RX FIFO block.

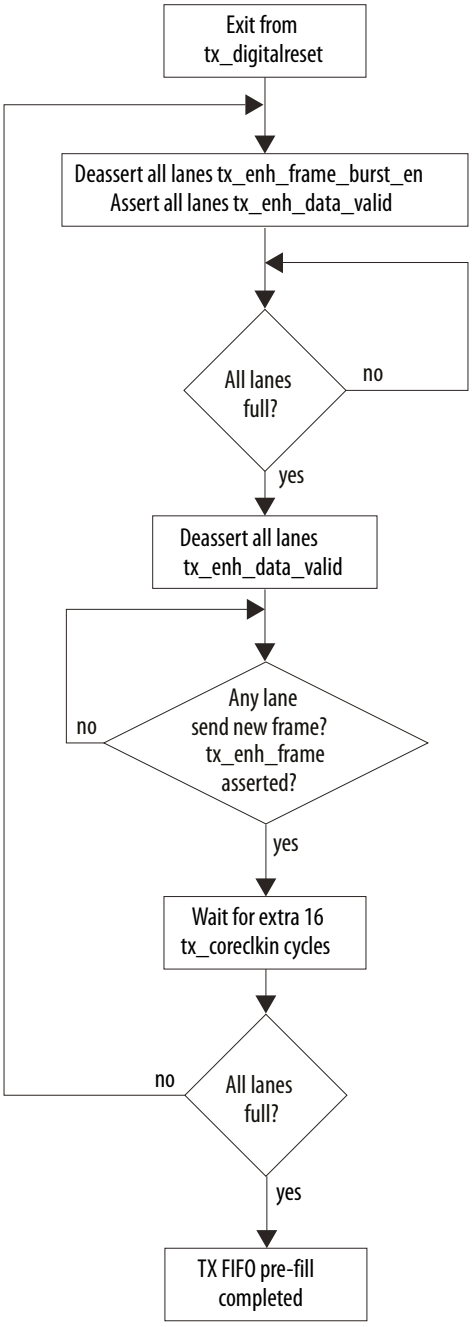
Note: You must also implement the soft bonding logic to control the transceiver TX FIFO block.

2.5.2.2.1. TX FIFO Soft Bonding

The MAC layer logic and TX soft bonding logic control the writing of the Interlaken word to the TX FIFO with `tx_enh_data_valid` (functions as a TX FIFO write enable) by monitoring the TX FIFO flags (`tx_fifo_full`, `tx_fifo_pfull`, `tx_fifo_empty`, `tx_fifo_pempty`, and so forth). On the TX FIFO read side, a read enable is controlled by the frame generator. If `tx_enh_frame_burst_en` is asserted high, the frame generator reads data from the TX FIFO.

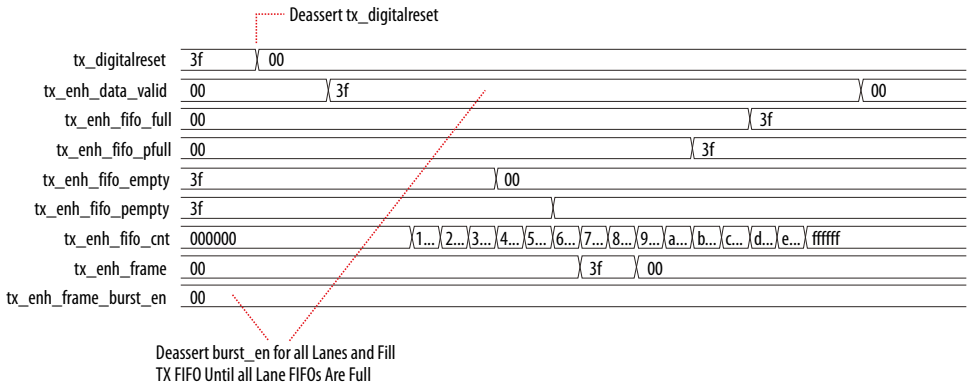
A TX FIFO pre-fill stage must be implemented to perform the TX channel soft bonding. The following figure shows the state of the pre-fill process.

Figure 22. TX Soft Bonding Flow



The following figure shows that after deasserting `tx_digitalreset`, TX soft bonding logic starts filling the TX FIFO until all lanes are full.

Figure 23. TX FIFO Pre-fill (6-lane Interface)

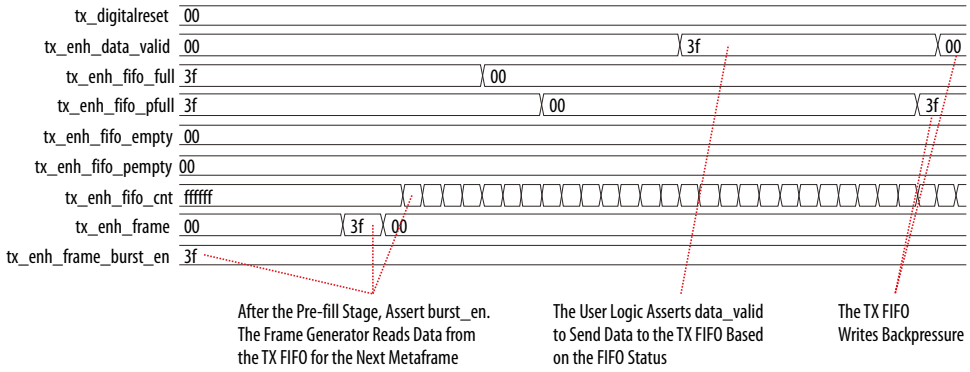


After the TX FIFO pre-fill stage completes, the transmit lanes synchronize and the MAC layer begins to send valid data to the transceiver's TX FIFO. You must never allow the TX FIFO to overflow or underflow. If it does, you must reset the transceiver and repeat the TX FIFO pre-fill stage.

For a single lane Interlaken implementation, TX FIFO soft bonding is not required. You can begin sending an Interlaken word to the TX FIFO after `tx_digitalreset` deasserts.

The following figure shows the MAC layer sending valid data to the Native PHY after the pre-fill stage. `tx_enh_frame_burst_en` is asserted, allowing the frame generator to read data from the TX FIFO. The TX MAC layer can now control `tx_enh_data_valid` and write data to the TX FIFO based on the FIFO status signals.

Figure 24. MAC Sending Valid Data (6-lane Interface)

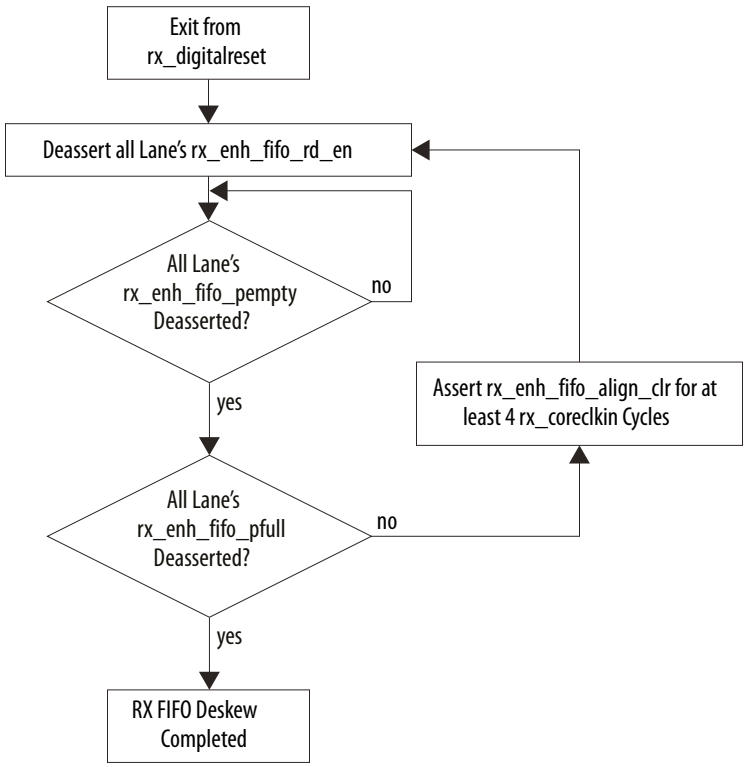


2.5.2.2.2. RX Multi-lane FIFO Deskew State Machine

Add deskew logic at the receiver side to eliminate the lane-to-lane skew created at the transmitter of the link partner, PCB, medium, and local receiver PMA.

Implement a multi-lane alignment deskew state machine to control the RX FIFO operation based on available RX FIFO status flags and control signals.

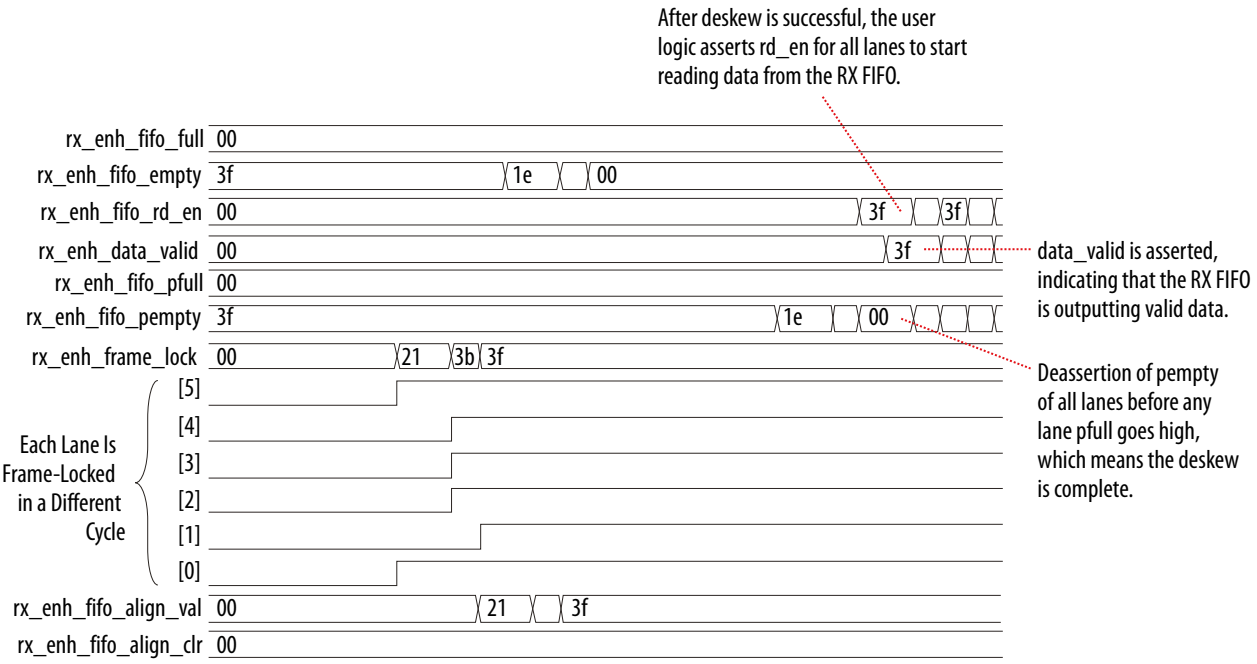
Figure 25. State Flow of the RX FIFO Deskew



Each lane's `rx_enh_fifo_rd_en` should remain deasserted before the RX FIFO deskew is completed. After frame lock is achieved (indicated by the assertion of `rx_enh_frame_lock`; this signal is not shown in the above state flow), data is written into the RX FIFO after the first alignment word (SYNC word) is found on that channel. Accordingly, the RX FIFO partially empty flag (`rx_enh_fifo_pempty`) of that channel is asserted. The state machine monitors the `rx_enh_fifo_pempty` and `rx_enh_fifo_pfull` signals of all channels. If the `rx_enh_fifo_pempty` signals from all channels deassert before any channels `rx_enh_fifo_pfull` assert, which implies the SYNC word has been found on all lanes of the link, the MAC layer can start reading from all the RX FIFO by asserting `rx_enh_fifo_rd_en` simultaneously. Otherwise, if the `rx_enh_fifo_pfull` signal of any channel asserts high before the `rx_enh_fifo_pempty` signals deassertion on all channels, the state machine needs to flush the RX FIFO by asserting `rx_enh_fifo_align_clr` high for 4 cycles and repeating the soft deskew process.

The following figure shows one RX deskew scenario. In this scenario, all of the RX FIFO partially empty lanes are deasserted while the pfull lanes are still deasserted. This indicates the deskew is successful and the FPGA fabric starts reading data from the RX FIFO.

Figure 26. RX FIFO Deskew



2.5.3. How to Implement Interlaken in Cyclone 10 GX Transceivers

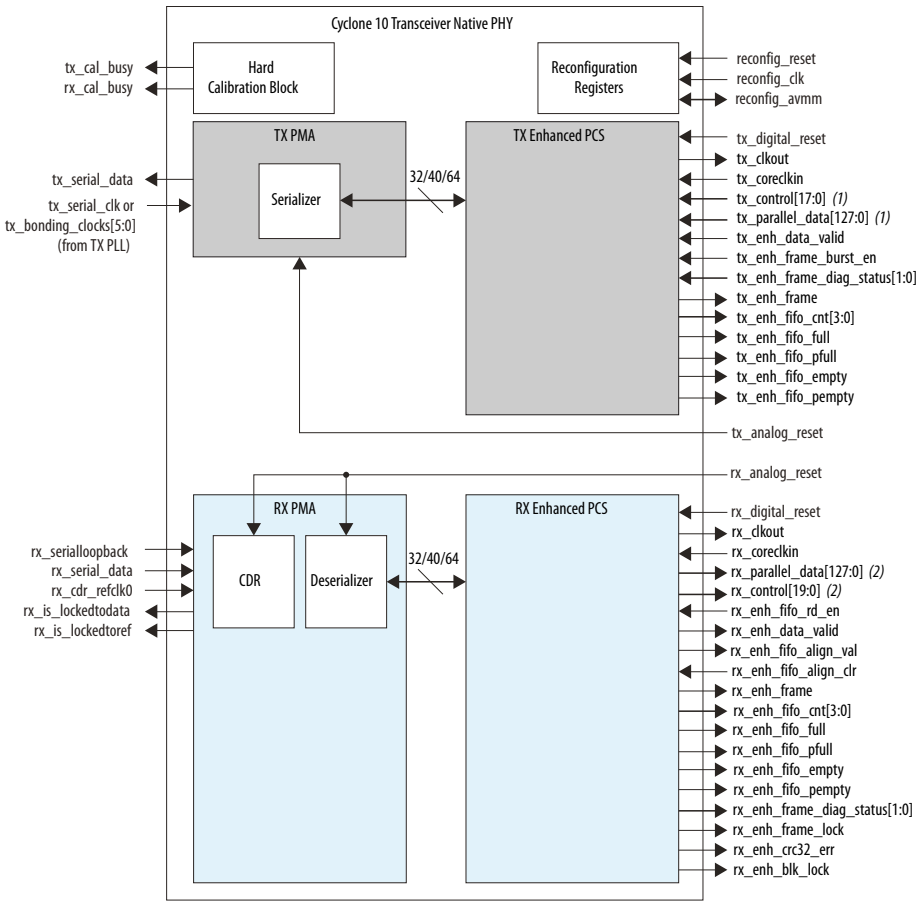
You should be familiar with the Interlaken protocol, Enhanced PCS and PMA architecture, PLL architecture, and the reset controller before implementing the Interlaken protocol PHY layer.

Cyclone 10 GX devices provide three preset variations for Interlaken in the IP Parameter Editor:

- Interlaken 1x6.25 Gbps
- Interlaken 6x10.3 Gbps

1. Instantiate the **Cyclone 10 GX Transceiver Native PHY IP** from the IP Catalog (**Installed IP ➤ Library ➤ Interface Protocols ➤ Transceiver PHY ➤ Cyclone 10 GX Transceiver Native PHY**).
   Refer to Select and Instantiate the PHY IP Core on page 17 for more details.
2. Select **Interlaken** from the **Transceiver configuration rules** list located under **Datapath Options**, depending on which protocol you are implementing.
3. Use the parameter values in the tables in Transceiver Native PHY IP Parameters for Interlaken Transceiver Configuration Rules.... Or you can use the protocol presets described in Transceiver Native PHY Presets. You can then modify the settings to meet your specific requirements.
4. Click **Generate** to generate the Native PHY IP (this is your RTL file).

Figure 27. Signals and Ports of Native PHY IP for Interlaken



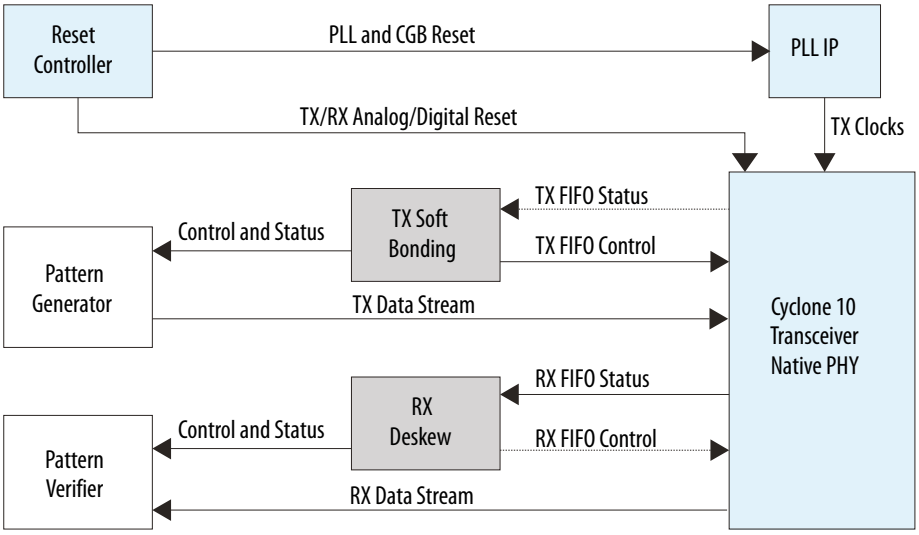
Notes:
(1) The width of tx_parallel_data and tx_control depends on whether the simplified interface is enabled or not. If the simplified interface is enabled, then tx_parallel_data = 64 bits and tx_control = 3 bits. The width shown here is without simplified interface.
(2) The width of rx_parallel_data and rx_control depends on whether the simplified interface is enabled or not. If the simplified interface is enabled, then rx_parallel_data = 64 bits and rx_control = 10 bits. The width shown here is without simplified interface.

5. Configure and instantiate your PLL.
6. Create a transceiver reset controller. You can use your own reset controller or use the Transceiver PHY Reset Controller.
7. Implement a TX soft bonding logic and an RX multi-lane alignment deskew state machine using fabric logic resources for multi-lane Interlaken implementation.
8. Connect the Native PHY IP to the PLL IP and the reset controller.

Figure 28. Connection Guidelines for an Interlaken PHY Design

This figure shows and example connection for an Interlake PHY design.

For the blue blocks, Intel provides an IP core. The gray blocks use the TX soft bonding logic and RX deskew logic. The white blocks are your test logic or MAC layer logic.



9. Simulate your design to verify its functionality.

Figure 29. 12 Lanes Bonded Interlaken Link, TX Direction

To show more details, three different time segments are shown with the same zoom level.

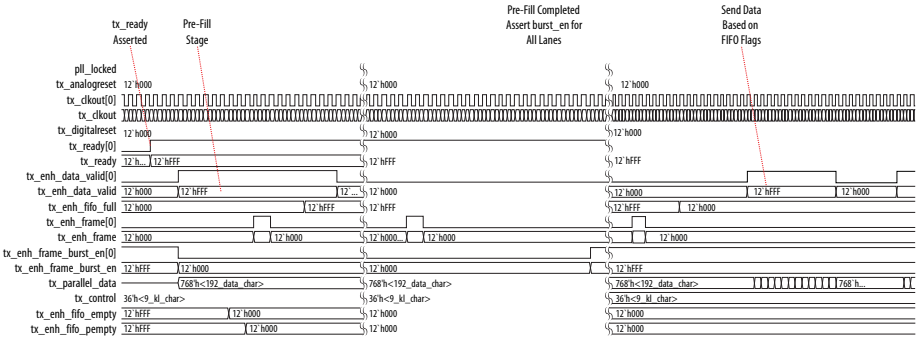
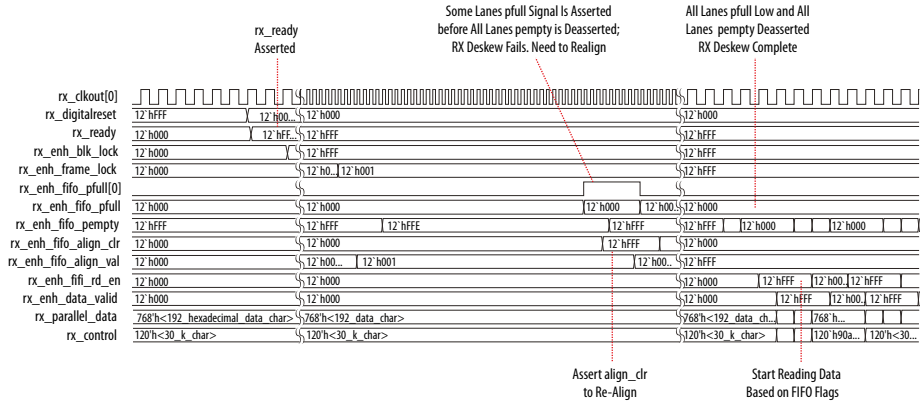


Figure 30. 12 Lanes Bonded Interlaken Link, RX Direction

To show more details, three different time segments are shown with different zoom level.



2.5.4. Native PHY IP Parameter Settings for Interlaken

This section contains the recommended parameter values for this protocol. Refer to *Using the Cyclone 10 GX Transceiver Native PHY IP Core* for the full range of parameter values.

Table 69. General and Datapath Parameters

Parameter	Value
Message level for rule violations	**error** **warning**
Transceiver configuration rules	**Interlaken**
PMA configuration rules	**basic**
Transceiver mode	**TX / RX Duplex** **TX Simplex** **RX Simplex**
Number of data channels	**1** to **12**
Data rate	**Up to 12.5 Gbps for GX devices** (Depending on Enhanced PCS to PMA interface width selection)
Enable datapath and interface reconfiguration	**On** / **Off**
Enable simplified data interface	**On** / **Off**
Provide separate interface for each channel	**On** / **Off**

Table 70. TX PMA Parameters

Parameter	Value
TX channel bonding mode	**Not bonded** **PMA-only bonding** **PMA and PCS bonding**
PCS TX channel bonding master	If **TX channel bonding mode** is set to **PMA and PCS bonding**, then: **Auto**, **0**, **1**, **2**, **3**,...,[Number of data channels – 1]

continued...

Parameter	Value
Actual PCS TX channel bonding master	If **TX channel bonding mode** is set to **PMA and PCS bonding**, then: **0**, **1**, **2**, **3**,...,[Number of data channels – 1]
TX local clock division factor	If **TX channel bonding mode** is not bonded, then: **1**, **2**, **4**, **8**
Number of TX PLL clock inputs per channel	If **TX channel bonding mode** is not bonded, then: **1**, **2**, **3**, **4**
Initial TX PLL clock input selection	**0**
Enable tx_pma_clkout port	**On** / **Off**
Enable tx_pma_div_clkout port	**On** / **Off**
tx_pma_div_clkout division factor	When **Enable tx_pma_div_clkout port** is **On**, then: **Disabled**, **1**, **2**, **33**, **40**, **66**
Enable tx_pma_elecidle port	**On** / **Off**
Enable rx_seriallpbken port	**On** / **Off**

Table 71. RX PMA Parameters

Parameter	Value
Number of CDR reference clocks	**1** to **5**
Selected CDR reference clock	**0** to **4**
Selected CDR reference clock frequency	Select legal range defined by the Quartus Prime software
PPM detector threshold	**100**, **300**, **500**, **1000**
CTLE adaptation mode	**manual**,
Enable rx_pma_clkout port	**On** / **Off**
Enable rx_pma_div_clkout port	**On** / **Off**
rx_pma_div_clkout division factor	When **Enable rx_pma_div_clkout port** is **On**, then: **Disabled**, **1**, **2**, **33**, **40**, **66**
Enable rx_pma_clkslip port	**On** / **Off**
Enable rx_is_lockedtodata port	**On** / **Off**
Enable rx_is_lockedtoref port	**On** / **Off**
Enable rx_set_locktodata and rx_set_locktoref ports	**On** / **Off**
Enable rx_seriallpbken port	**On** / **Off**
Enable PRBS verifier control and status ports	**On** / **Off**

Table 72. Enhanced PCS Parameters

Parameter	Value
Enhanced PCS / PMA interface width	**32**, **40**, **64**
FPGA fabric / Enhanced PCS interface width	**67**
Enable 'Enhanced PCS' low latency mode	Allowed when the PMA interface width is 32 and preset variations for data rate is 10.3125 Gbps or 6.25 Gbps; otherwise **Off**

continued...

Parameter	Value
Enable RX/TX FIFO double-width mode	**Off**
TX FIFO mode	**Interlaken**
TX FIFO partially full threshold	**8** to **15**
TX FIFO partially empty threshold	**1** to **8**
Enable tx_enh_fifo_full port	**On** / **Off**
Enable tx_enh_fifo_pfull port	**On** / **Off**
Enable tx_enh_fifo_empty port	**On** / **Off**
Enable tx_enh_fifo_pempty port	**On** / **Off**
RX FIFO mode	**Interlaken**
RX FIFO partially full threshold	from **10**-**29** (no less than pempty_threshold+8)
RX FIFO partially empty threshold	**2** to **10**
Enable RX FIFO alignment word deletion (Interlaken)	**On** / **Off**
Enable RX FIFO control word deletion (Interlaken)	**On** / **Off**
Enable rx_enh_data_valid port	**On** / **Off**
Enable rx_enh_fifo_full port	**On** / **Off**
Enable rx_enh_fifo_pfull port	**On** / **Off**
Enable rx_enh_fifo_empty port	**On** / **Off**
Enable rx_enh_fifo_pempty port	**On** / **Off**
Enable rx_enh_fifo_del port (10GBASE-R)	**Off**
Enable rx_enh_fifo_insert port (10GBASE-R)	**Off**
Enable rx_enh_fifo_rd_en port	**On**
Enable rx_enh_fifo_align_val port (Interlaken)	**On** / **Off**
Enable rx_enh_fifo_align_clr port (Interlaken)	**On**

Table 73. Interlaken Frame Generator Parameters

Parameter	Value
Enable Interlaken frame generator	**On**
Frame generator metaframe length	**5** to **8192** (Intel recommends a minimum metaframe length of 128)
Enable frame generator burst control	**On**
Enable tx_enh_frame port	**On**
Enable tx_enh_frame_diag_status port	**On**
Enable tx_enh_frame_burst_en port	**On**

Table 74. Interlaken Frame Synchronizer Parameters

Parameter	Value
Enable Interlaken frame synchronizer	**On**
Frame synchronizer metaframe length	**5** to **8192** (Intel recommends a minimum metaframe length of 128)
Enable rx_enh_frame port	**On**
Enable rx_enh_frame_lock port	**On** / **Off**
Enable rx_enh_frame_diag_status port	**On** / **Off**

Table 75. Interlaken CRC-32 Generator and Checker Parameters

Parameter	Value
Enable Interlaken TX CRC-32 generator	**On**
Enable Interlaken TX CRC-32 generator error insertion	**On** / **Off**
Enable Interlaken RX CRC-32 checker	**On**
Enable rx_enh_crc32_err port	**On** / **Off**

Table 76. Scrambler and Descrambler Parameters

Parameter	Value
Enable TX scrambler (10GBASE-R / Interlaken)	**On**
TX scrambler seed (10GBASE-R / Interlaken)	**0x1** to **0x3FFFFFFFFFFFFFF**
Enable RX descrambler (10GBASE-R / Interlaken)	**On**

Table 77. Interlaken Disparity Generator and Checker Parameters

Parameter	Value
Enable Interlaken TX disparity generator	**On**
Enable Interlaken RX disparity checker	**On**
Enable Interlaken TX random disparity bit	**On** / **Off**

Table 78. Block Sync Parameters

Parameter	Value
Enable RX block synchronizer	**On**
Enable rx_enh_blk_lock port	**On** / **Off**

Table 79. Gearbox Parameters

Parameter	Value
Enable TX data bitslip	**Off**
Enable TX data polarity inversion	**On** / **Off**
Enable RX data bitslip	**Off**

continued...

Parameter	Value
Enable RX data polarity inversion	**On** / **Off**
Enable tx_enh_bitslip port	**Off**
Enable rx_bitslip port	**Off**

Table 80. Dynamic Reconfiguration Parameters

Parameter	Value
Enable dynamic reconfiguration	**On** / **Off**
Share reconfiguration interface	**On** / **Off**
Enable Native PHY Debug Master Endpoint	**On** / **Off**
Separate reconfig_waitrequest from the status of AVMM arbitration with PreSICE	**On** / **Off**
Enable capability registers	**On** / **Off**
Set user-defined IP identifier:	**0** to **255**
Enable control and status registers	**On** / **Off**
Enable prbs soft accumulators	**On** / **Off**

Table 81. Configuration Files Parameters

Parameter	Value
Configuration file prefix	—
Generate SystemVerilog package file	**On** / **Off**
Generate C header file	**On** / **Off**
Generate MIF (Memory Initialization File)	**On** / **Off**
Include PMA analog settings in configuration files	**On** / **Off**

Table 82. Configuration Profiles Parameters

Parameter	Value
Enable multiple reconfiguration profiles	**On** / **Off**
Enable embedded reconfiguration streamer	**On** / **Off**
Generate reduced reconfiguration files	**On** / **Off**
Number of reconfiguration profiles	1 to 8
Selected reconfiguration profile	1 to 7

2.6. Ethernet

The Ethernet standard comprises many different PHY standards with variations in signal transmission medium and data rates. The 1G/10GbE and 10GBASE-R PHY IP Core enables Ethernet connectivity at 1 Gbps and 10 Gbps.

Table 83. 1G/10G Data Rates and Transceiver Configuration Rules

Data Rate	Transceiver Configuration Rule/IP
1G	• Gigabit Ethernet • Gigabit Ethernet 1588
10G	• 10GBASE-R • 10GBASE-R 1588
1G/10G	1G/10G Ethernet PHY IP

2.6.1. Gigabit Ethernet (GbE) and GbE with IEEE 1588v2

Gigabit Ethernet (GbE) is a high-speed local area network technology that provides data transfer rates of about 1 Gbps. GbE builds on top of the ethernet protocol, but increases speed tenfold over Fast Ethernet. IEEE 802.3 defines GbE as an intermediate (or transition) layer that interfaces various physical media with the media access control (MAC) in a Gigabit Ethernet system. Gigabit Ethernet PHY shields the MAC layer from the specific nature of the underlying medium and is divided into three sub-layers shown in the following figure.

Figure 31. GbE PHY Connection to IEEE 802.3 MAC and RS

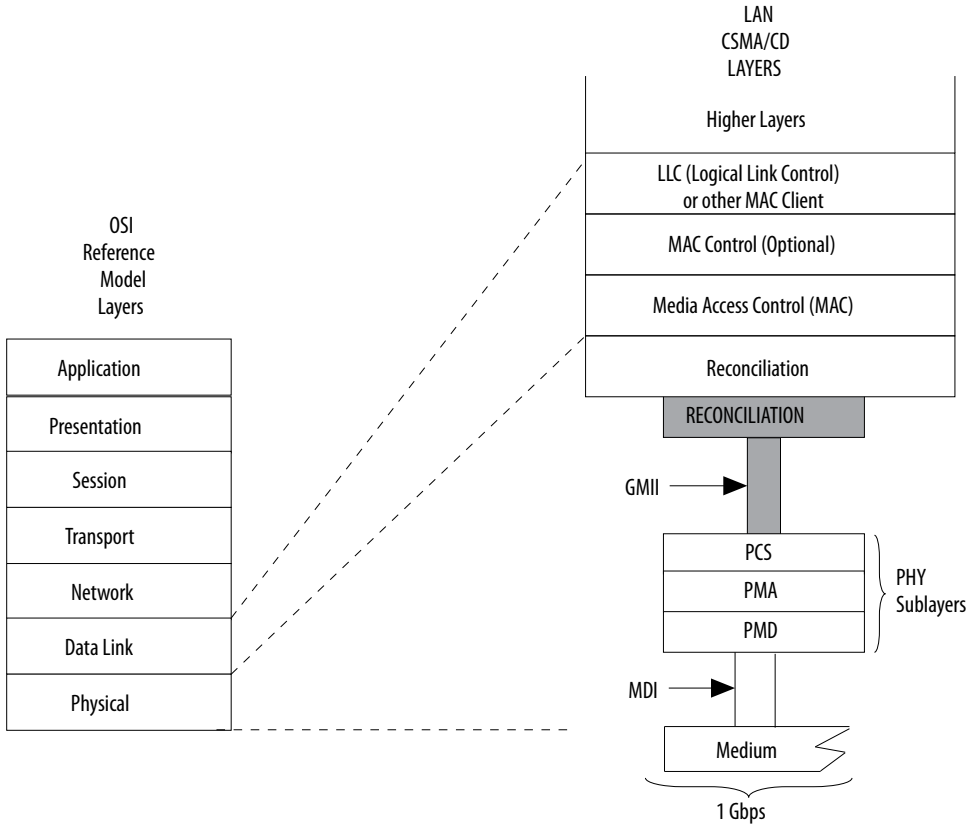
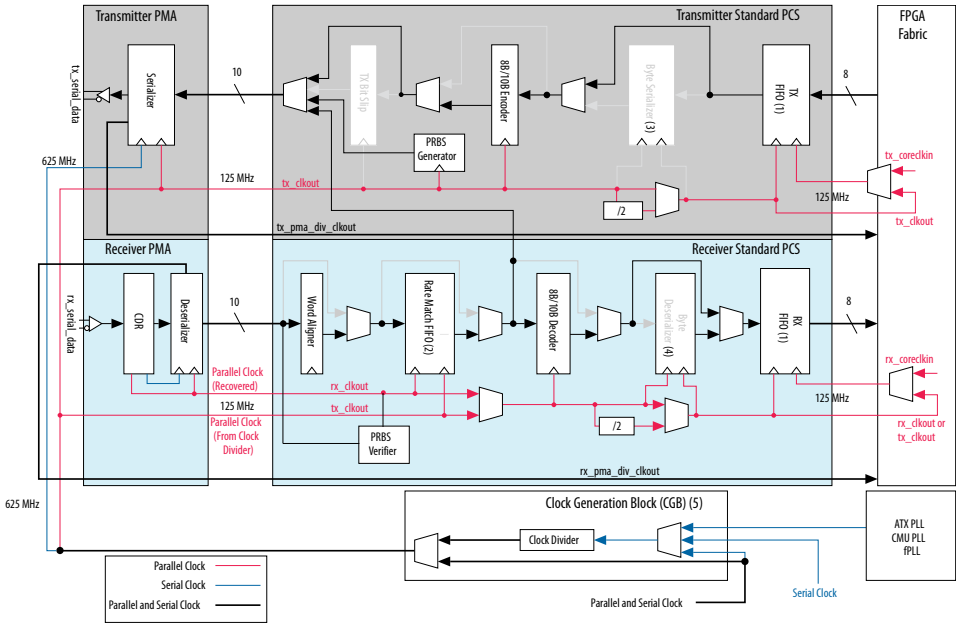


Figure 32. Transceiver Channel Datapath and Clocking at 1250 Mbps for GbE, GbE with IEEE 1588v2



Notes:
1. This block is set in low latency mode for GbE and register_fifo mode for GbE with IEEE 1588v2.
2. The rate match FIFO of the hard PCS is disabled for GbE with IEEE 1588v2 because it is not able to achieve deterministic latency. It is also disabled for Triple-speed Ethernet (TSE) configurations that require an auto-negotiation sequence. The insertion/deletion operation could break the auto-negotiation functionality due to the rate matching of different frequency PPM scenarios.The soft rate match FIFO is constructed in the GbE Serial Gigabit Media Independent Interface (SGMII) IP core.
3. The byte serializer can be enabled or disabled.
4. The byte deserializer can be enabled or disabled.
5. The CGB is in the Native PHY.

Note: The Native PHY only supports basic PCS functions. The Native PHY does not support auto-negotiation state machine, collision-detect, and carrier-sense. If required, you must implement these functions in the FPGA fabric or external circuits.

GbE with IEEE 1588v2

GbE with IEEE 1588v2 provides a standard method to synchronize devices on a network. To improve performance, the protocol synchronizes slave clocks to a master clock so that events and time stamps are synchronized in all devices. The protocol enables heterogeneous systems that include clocks of various inherent precision, resolution, and stability to synchronize to a grandmaster clock.

The TX FIFO and RX FIFO are set to **register_fifo** mode for GbE with IEEE 1588v2.

2.6.1.1. 8B/10B Encoding for GbE, GbE with IEEE 1588v2

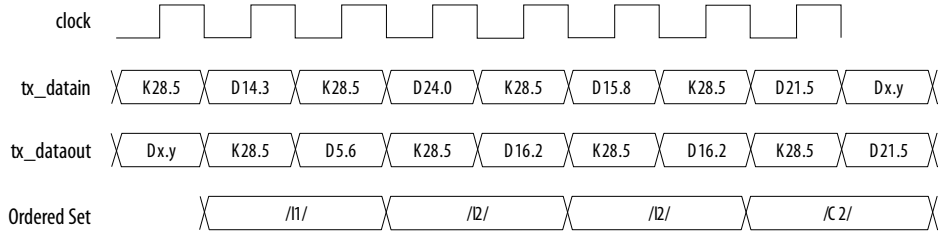
The 8B/10B encoder clocks 8-bit data and 1-bit control identifiers from the transmitter phase compensation FIFO and generates 10-bit encoded data. The 10-bit encoded data is sent to the PMA.

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

For the GbE protocol, the transmitter replaces any /Dx.y/ following a /K28.5/ comma 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, a /I2/ ordered set is generated. The disparity at the end of a /I1/ is the opposite of that at the beginning of the /I1/. The disparity at the end of a /I2/ is the same as the beginning running disparity immediately preceding transmission of the Idle code. This sequence ensures a negative running disparity at the end of an Idle ordered set. A /Kx.y/ following a /K28.5/ does not get replaced.

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

Figure 33. Idle Ordered-Set Generation Example



Related Information

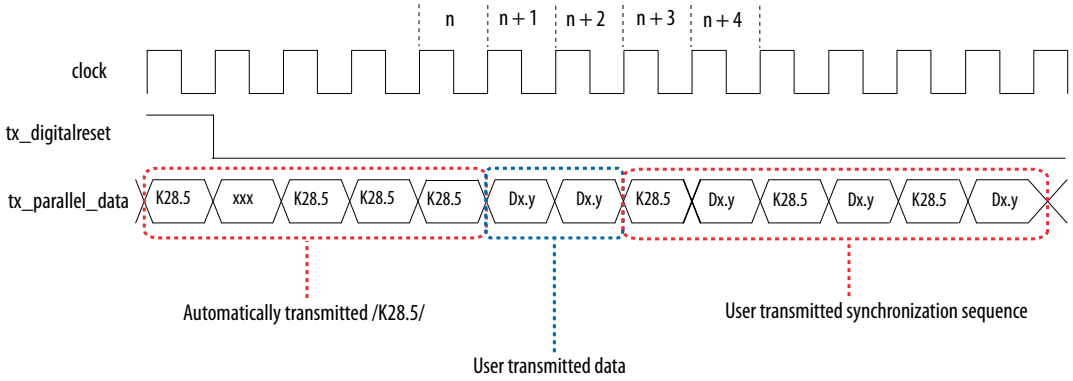
8B/10B Encoder on page 302

2.6.1.1.1. Reset Condition for 8B/10B Encoder in GbE, GbE with IEEE 1588v2

After deassertion of `tx_digitalreset`, the transmitters automatically transmit at least three /K28.5/ comma code groups before transmitting user data on the `tx_parallel_data` port. This transmission 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. The synchronization state machine treats this as an error condition and goes into the loss of synchronization state.

Figure 34. Reset Condition



2.6.1.2. Word Alignment for GbE, GbE with IEEE 1588v2

The word aligner for the GbE and GbE with IEEE 1588v2 protocols is configured in automatic synchronization state machine mode. The Intel Quartus Prime software automatically configures the synchronization state machine to indicate synchronization when the receiver receives 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.

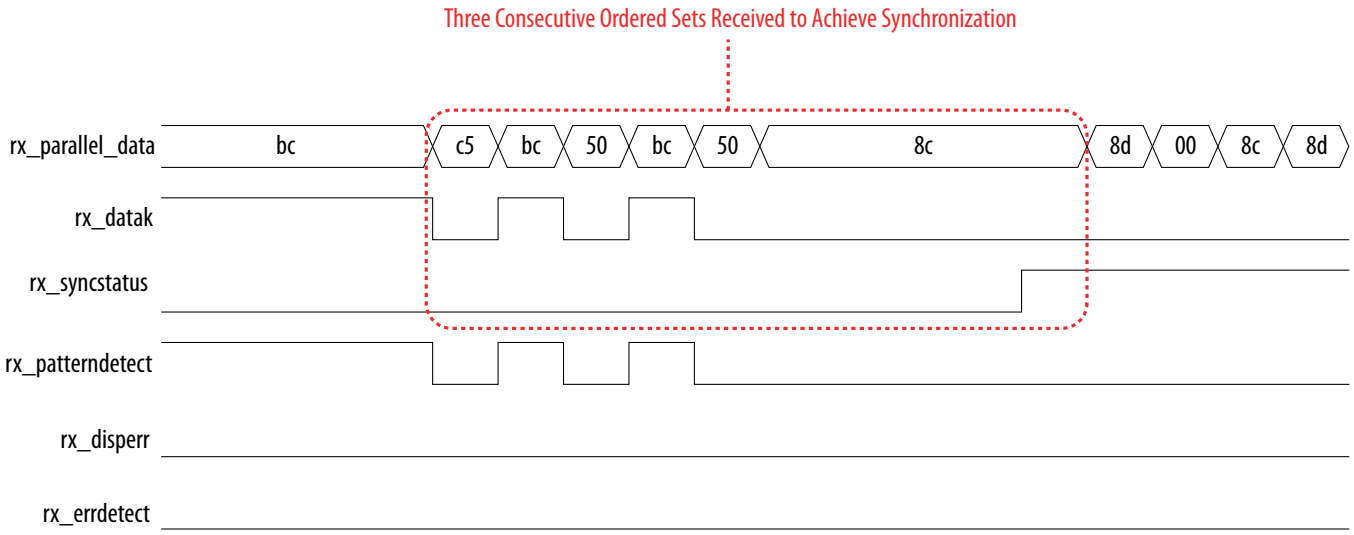
The GbE PHY IP core signals receiver synchronization status on the `rx_syncstatus` port of each channel. A high on the `rx_syncstatus` port indicates that the lane is synchronized; a low on the `rx_syncstatus` port indicates that the lane has fallen out of synchronization. The receiver loses synchronization when it detects three invalid code groups separated by less than three valid code groups or when it is reset.

Table 84. Synchronization State Machine Parameter Settings for GbE

Synchronization State Machine Parameter	Setting
Number of word alignment patterns to achieve sync	3
Number of invalid data words to lose sync	3
Number of valid data words to decrement error count	3

The following figure shows `rx_syncstatus` high when three consecutive ordered sets are sent through `rx_parallel_data`.

Figure 35. rx_syncstatus High



Related Information

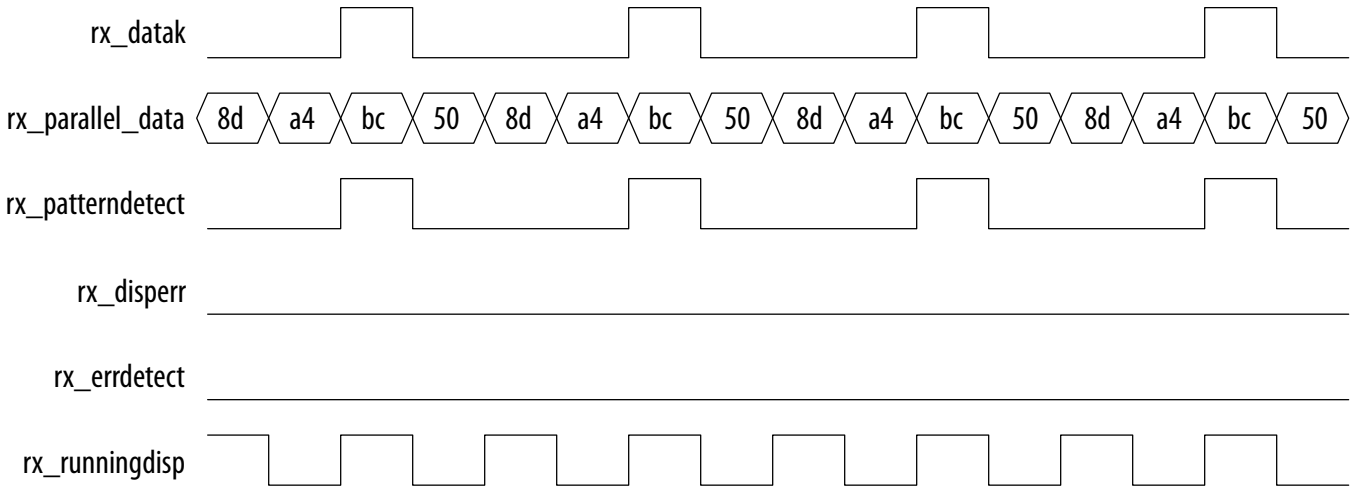
Word Aligner on page 305

2.6.1.3. 8B/10B Decoding for GbE, GbE with IEEE 1588v2

The 8B/10B decoder takes a 10-bit encoded value as input and produces an 8-bit data value and 1-bit control value as output.

Figure 36. Decoding for GbE

Dx.y(0x8d), Dx.y(0xa4), K28.5(0xbc), and Dx.y(0x50) are received at `rx_parallel_data`. /K28.5/ is set as the word alignment pattern. `rx_patterndetect` goes high whenever it detects /K28.5/(0xbc). `rx_datak` is high when bc is received, indicating that the decoded word is a control word. Otherwise, `rx_datak` is low. `rx_runningdisp` is high for 0x8d, indicating that the decoded word has negative disparity and 0xa4 has positive disparity.



Related Information

8B/10B Decoder on page 311

2.6.1.4. Rate Match FIFO for GbE

The rate match FIFO compensates frequency Part-Per-Million (ppm) differences between the upstream transmitter and the local receiver reference clock up to 125 MHz ± 100 ppm difference.

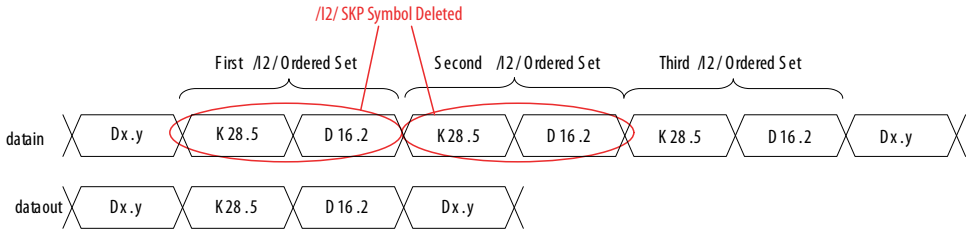
Note: 200 ppm total is only true if calculated as (125 MHz + 100 ppm) - (125 MHz - 100 ppm) = 200 ppm. By contrast, (125 MHz + 0 ppm) - (125 MHz - 200 ppm) supports center-spread clocking, but does not support downstream clocking.

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 (IPG) adhering to the rules listed in the IEEE 802.3-2008 specification.

The rate match operation begins after the synchronization state machine in the word aligner indicates synchronization is acquired by driving the `rx_syncstatus` signal high. The rate matcher deletes or inserts both symbols /K28.5/ and /D16.2/ of the /I2/ ordered sets as a pair in the operation to prevent the rate match FIFO from overflowing or underflowing. The rate match operation can insert or delete as many /I2/ ordered sets as necessary.

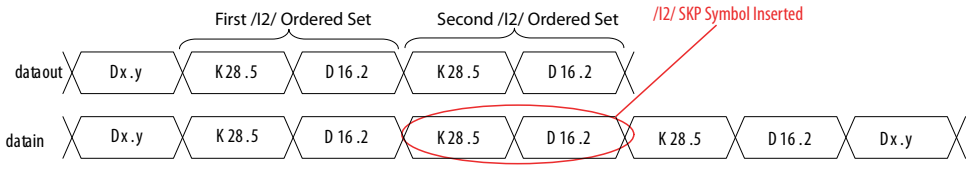
The following figure shows a rate match deletion operation example where three symbols must be deleted. Because the rate match FIFO can only delete /I2/ ordered sets, it deletes two /I2/ ordered sets (four symbols deleted).

Figure 37. Rate Match FIFO Deletion



The following figure 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).

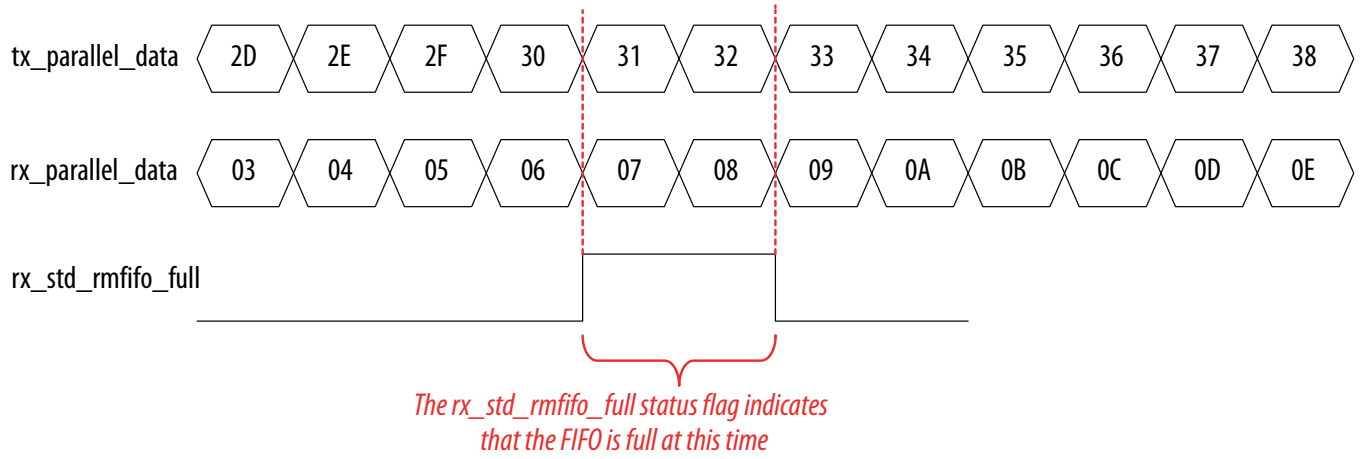
Figure 38. Rate Match FIFO Insertion



`rx_std_rmfifo_full` and `rx_std_rmfifo_empty` are forwarded to the FPGA fabric to indicate rate match FIFO full and empty conditions.

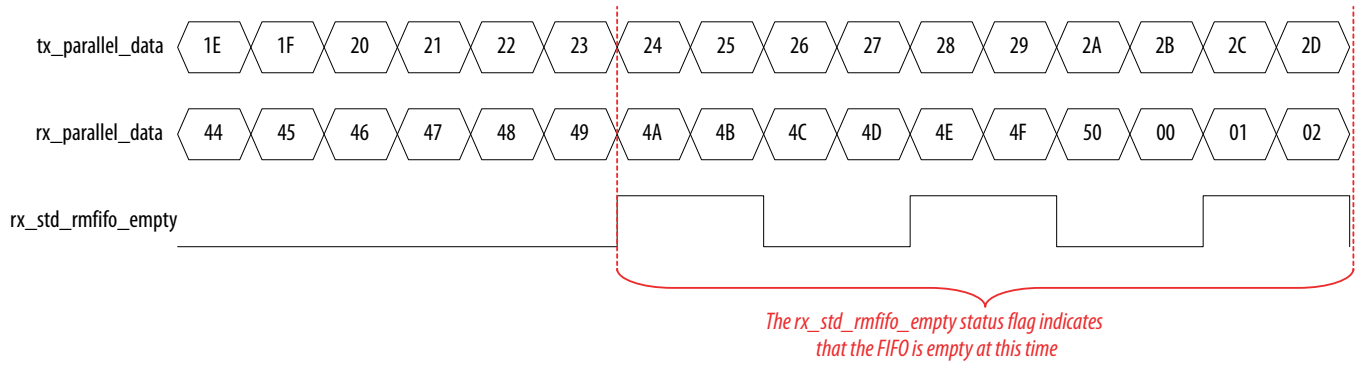
The rate match FIFO does not delete code groups to overcome a FIFO full condition. It asserts the `rx_std_rmfifo_full` flag for at least two recovered clock cycles to indicate rate match FIFO full. The following figure shows the rate match FIFO full condition when the write pointer is faster than the read pointer.

Figure 39. Rate Match FIFO Full Condition



The rate match FIFO does not insert code groups to overcome the FIFO empty condition. It asserts the `rx_std_rmfifo_empty` flag for at least two recovered clock cycles to indicate that the rate match FIFO is empty. The following figure shows the rate match FIFO empty condition when the read pointer is faster than the write pointer.

Figure 40. Rate Match FIFO Empty Condition



In the case of rate match FIFO full and empty conditions, you must assert the `rx_digitalreset` signal to reset the receiver PCS blocks.

Related Information

Rate Match FIFO on page 310

2.6.1.5. How to Implement GbE, GbE with IEEE 1588v2 in Intel Cyclone 10 GX Transceivers

You should be familiar with the Standard PCS and PMA architecture, PLL architecture, and the reset controller before implementing the GbE protocol.

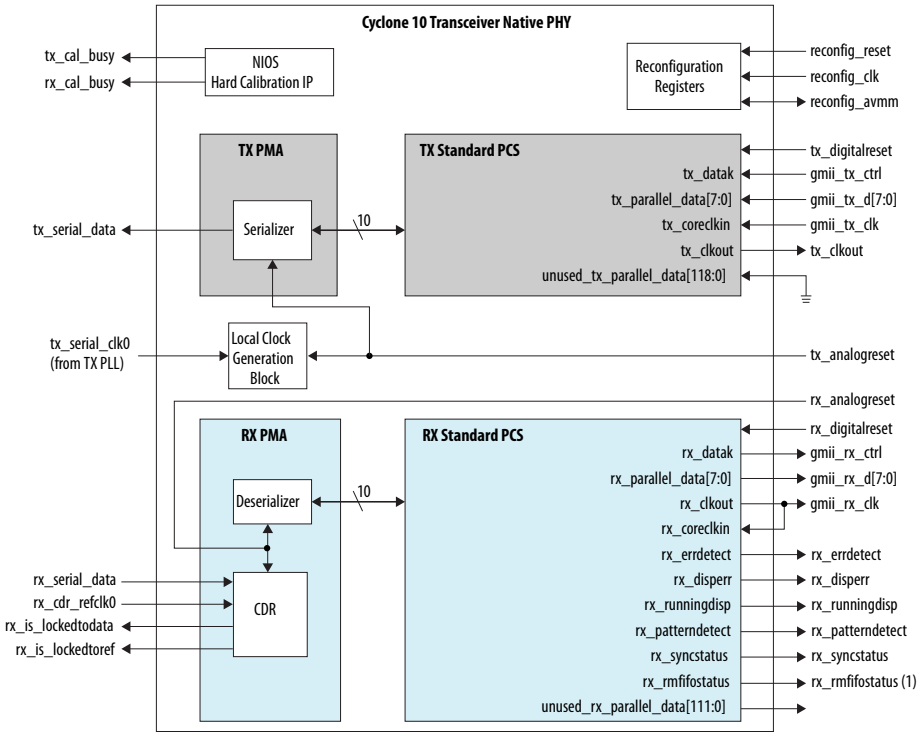
1. Instantiate the **Intel Cyclone 10 GX Transceiver Native PHY IP** from the IP Catalog.

Refer to Select and Instantiate the PHY IP Core on page 17.

2. Select **GbE** or **GbE 1588** from the **Transceiver configuration rules** list located under **Datapath Options**, depending on which protocol you are implementing.
3. Use the parameter values in the tables in Native PHY IP Parameter Settings for GbE and GbE with IEEE 1588v2 on page 95 as a starting point. Or, you can use the protocol presets described in Transceiver Native PHY Presets. Use the **GIGE-1.25 Gbps** preset for GbE, and the **GIGE-1.25 Gbps 1588** preset for GbE 1588. You can then modify the setting to meet your specific requirements.
4. Click **Generate** to generate the Native PHY IP core top-level RTL file.

Figure 41. Signals and Ports for Native PHY IP Configured for GbE or GbE with IEEE 1588v2

Generating the IP core creates signals and ports based on your parameter settings.

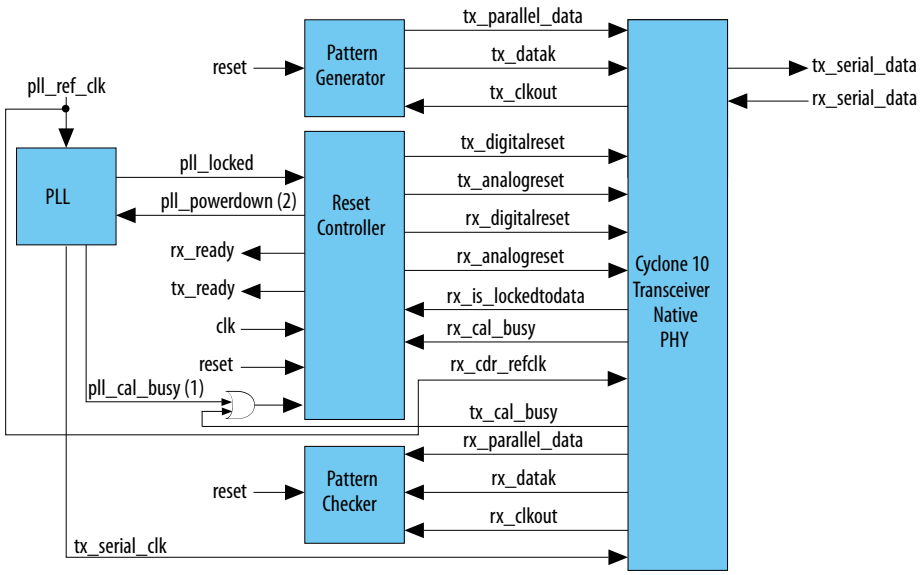


Note:
1. rx_rmfifostatus is not available in the GbE with 1588 configuration.

5. Instantiate and configure your PLL.
6. Instantiate a transceiver reset controller.

   You can use your own reset controller or use the Native PHY Reset Controller IP core.
7. Connect the Native PHY IP to the PLL IP and the reset controller. Use the information in the figure below to connect the ports.

Figure 42. Connection Guidelines for a GbE/GbE with IEEE 1588v2 PHY Design



Note:
1. The pll_cal_busy signal is not available when using the CMU PLL.
2. The pll_powerdown signal is not available separately for user control when using the fPLL. The reset controller handles PLL reset for the fPLL.

8. Simulate your design to verify its functionality.

2.6.1.6. Native PHY IP Parameter Settings for GbE and GbE with IEEE 1588v2

This section contains the recommended parameter values for this protocol. Refer to *Using the Cyclone 10 GX Transceiver Native PHY IP Core* for the full range of parameter values.

Table 85. General and Datapath Options

The first two sections of the Native PHY [IP] parameter editor for the Native PHY IP provide a list of general and datapath options to customize the transceiver.

Parameter	Value
Message level for rule violations	**error** **warning**
Transceiver configuration rules	**GbE** (for GbE) **GbE 1588** (for GbE with IEEE 1588v2)
Transceiver mode	**TX/RX Duplex** **TX Simplex** **RX Simplex**
Number of data channels	**1 to 12**
Data rate	**1250 Mbps**
Enable datapath and interface reconfiguration	**On/Off**
Enable simplified data interface	**On/Off**

Table 86. TX PMA Parameters

Parameter	Value
TX channel bonding mode	**Not bonded**
TX local clock division factor	**1**, **2**, **4**, **8**
Number of TX PLL clock inputs per channel	**1**, **2**, **3**, **4**
Initial TX PLL clock input selection	**0 to 3**
Enable tx_pma_clkout port	**On/Off**
Enable tx_pma_div_clkout port	**On/Off**
tx_pma_div_clkout division factor	**Disabled**, **1**, **2**, **33**, **40**, **66**
Enable tx_pma_elecidle port	**On/Off**
Enable rx_seriallpbken port	**On/Off**

Table 87. RX PMA Parameters

Parameter	Value
Number of CDR reference Clocks	**1** to **5**
Selected CDR reference clock	**0** to **4**
Selected CDR reference clock frequency	Select legal range defined by the Quartus Prime software
PPM detector threshold	**100**, **300**, **500**, **1000**
CTLE adaptation mode	**manual**
Enable rx_pma_clkout port	**On/Off**
Enable rx_pma_div_clkout port	**On/Off**
rx_pma_div_clkout division factor	**Disabled**, **1**, **2**, **33**, **40**, **66**
Enable rx_pma_iqtxrx_clkout port	**On/Off**
Enable rx_pma_clkslip port	**On/Off**
Enable rx_is_lockedtodata port	**On/Off**
Enable rx_is_lockedtoref port	**On/Off**
Enable rx_set_locktodata and rx_set_locktoref ports	**On/Off**
Enable rx_seriallpbken port	**On/Off**
Enable PRBS verifier control and status ports	**On/Off**

Table 88. Standard PCS Parameters

Parameters	Value
Standard PCS / PMA interface width	**10**
FPGA fabric / Standard TX PCS interface width	**8**
FPGA fabric / Standard RX PCS interface width	**8**
Enable Standard PCS low latency mode	**Off**
TX FIFO mode	**low latency** (for GbE) **register_fifo** (for GbE with IEEE 1588v2)

continued...

Parameters	Value
RX FIFO mode	**low latency** (for GbE) **register_fifo** (for GbE with IEEE 1588v2)
Enable tx_std_pcfifo_full port	**On/Off**
Enable tx_std_pcfifo_empty port	**On/Off**
Enable rx_std_pcfifo_full port	**On/Off**
Enable rx_std_pcfifo_empty port	**On/Off**
TX byte serializer mode	**Disabled**
RX byte deserializer mode	**Disabled**
Enable TX 8B/10B encoder	**On**
Enable TX 8B/10B disparity control	**On/Off**
Enable RX 8B/10B decoder	**On**
RX rate match FIFO mode	**gige** (for GbE) **disabled** (for GbE with IEEE 1588v2)
RX rate match insert / delete -ve pattern (hex)	0x000ab683 (/K28.5/D2.2/) (for GbE) 0x00000000 (disabled for GbE with IEEE 1588v2)
RX rate match insert / delete +ve pattern (hex)	0x000a257c (/K28.5/D16.2/) (for GbE) 0x00000000 (disabled for GbE with IEEE 1588v2)
Enable rx_std_rmfifo_full port	**On/Off** (option disabled for GbE with IEEE 1588v2)
Enable rx_std_rmfifo_empty port	**On/Off** (option disabled for GbE with IEEE 1588v2)
Enable TX bit slip	**Off**
Enable tx_std_bitslipboundarysel port	**On/Off**
RX word aligner mode	**Synchronous state machine**
RX word aligner pattern length	**7**
RX word aligner pattern (hex)	0x000000000000007c (Comma) (for 7-bit aligner pattern length), 0x000000000000017c (/K28.5/) (for 10-bit aligner pattern length)
Number of word alignment patterns to achieve sync	**3**
Number of invalid data words to lose sync	**3**
Number of valid data words to decrement error count	**3**
Enable fast sync status reporting for deterministic latency SM	**On/Off**
Enable rx_std_wa_patternalign port	**Off**
Enable rx_std_wa_a1a2size port	**Off**
Enable rx_std_bitslipboundarysel port	**Off**
Enable rx_bitslip port	**Off**
Enable TX bit reversal	**Off**
Enable TX byte reversal	**Off**

continued...

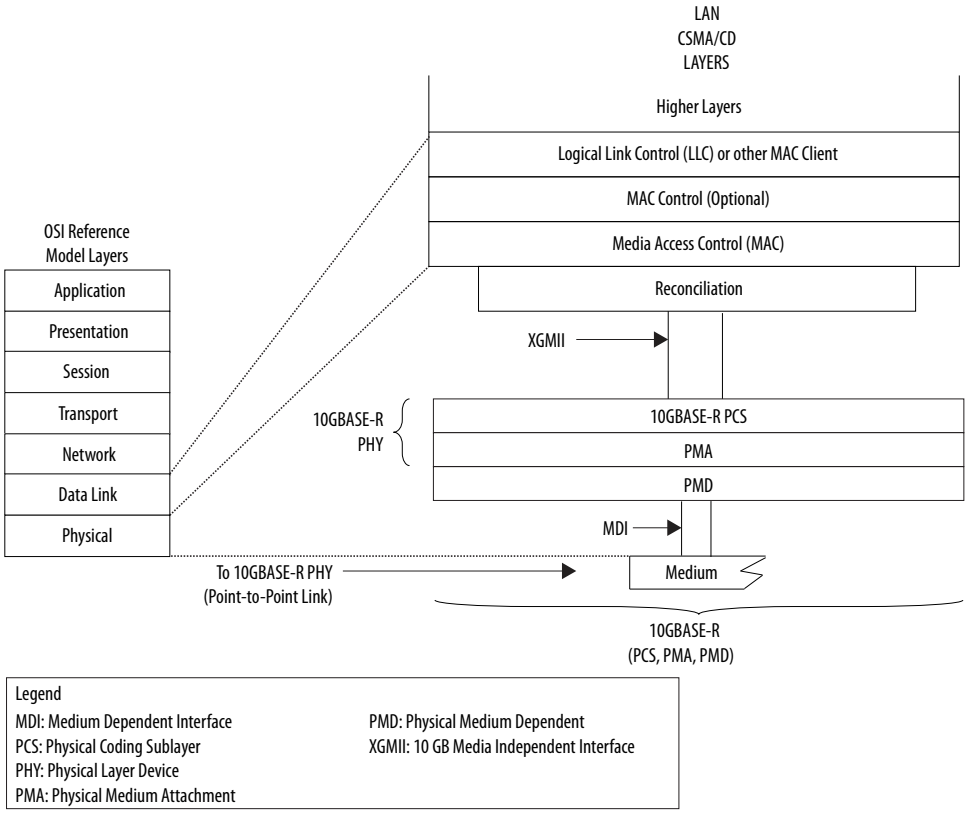
Parameters	Value
Enable TX polarity inversion	**On/Off**
Enable tx_polinv port	**On/Off**
Enable RX bit reversal	**Off**
Enable rx_std_bitrev_ena port	**Off**
Enable RX byte reversal	**Off**
Enable rx_std_byterev_ena port	**Off**
Enable RX polarity inversion	**On/Off**
Enable rx_polinv port	**On/Off**
Enable rx_std_signaldetect port	**On/Off**
All options under **PCIe Ports**	**Off**

2.6.2. 10GBASE-R and 10GBASE-R with IEEE 1588v2 Variants

10GBASE-R PHY is the Ethernet-specific physical layer running at a 10.3125-Gbps data rate as defined in Clause 49 of the IEEE 802.3-2008 specification. Cyclone 10 GX transceivers can implement 10GBASE-R variants like 10GBASE-R with IEEE 1588v2.

The 10GBASE-R parallel data interface is the 10 Gigabit Media Independent Interface (XGMII) that interfaces with the Media Access Control (MAC), which has the optional Reconciliation Sub-layer (RS).

Figure 43. 10GBASE-R PHY as Part of the IEEE802.3-2008 Open System Interconnection (OSI)



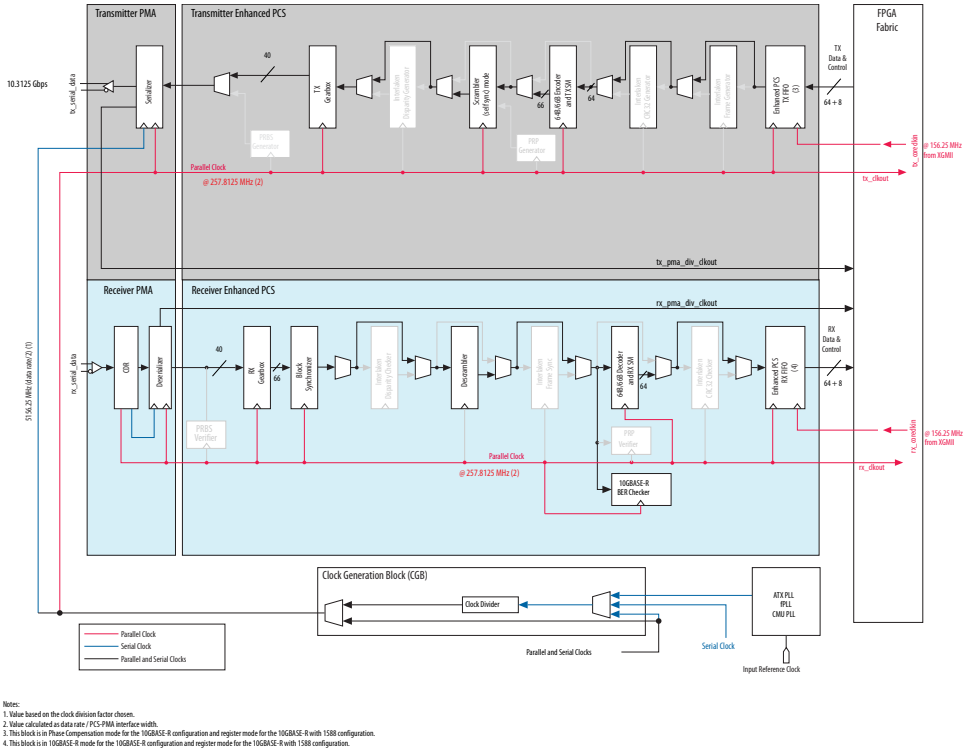
10GBASE-R is a single-channel protocol that runs independently. You can configure the transceivers to implement 10GBASE-R PHY functionality by using the presets of the Native PHY IP. The complete PCS and PHY solutions can be used to interface with a third-party PHY MAC layer as well.

The following 10GBASE-R variants area available from presets:

- 10GBASE-R
- 10GBASE-R Low Latency
- 10GBASE-R Register Mode

Intel recommends that you use the presets for selecting the suitable 10GBASE-R variants directly if you are configuring through the Native PHY IP core.

Figure 44. Transceiver Channel Datapath and Clocking for 10GBASE-R



Notes:
1. Value based on the clock division factor chosen.
2. Value calculated as data rate / PCS-PMA interface width.
3. This block is in Phase Compensation mode for the 10GBASE-R configuration and register mode for the 10GBASE-R with 1588 configuration.
4. This block is in 10GBASE-R mode for the 10GBASE-R configuration and register mode for the 10GBASE-R with 1588 configuration.

10GBASE-R with IEEE 1588v2

When choosing the 10GBASE-R PHY with IEEE 1588v2 mode preset, the hard TX and RX FIFO are set to register mode. The output clock frequency of `tx_clkout` and `rx_clkout` to the FPGA fabric is based on the PCS-PMA interface width. For example, if the PCS-PMA interface is 40-bit, `tx_clkout` and `rx_clkout` run at 10.3125 Gbps/40-bit = 257.8125 MHz.

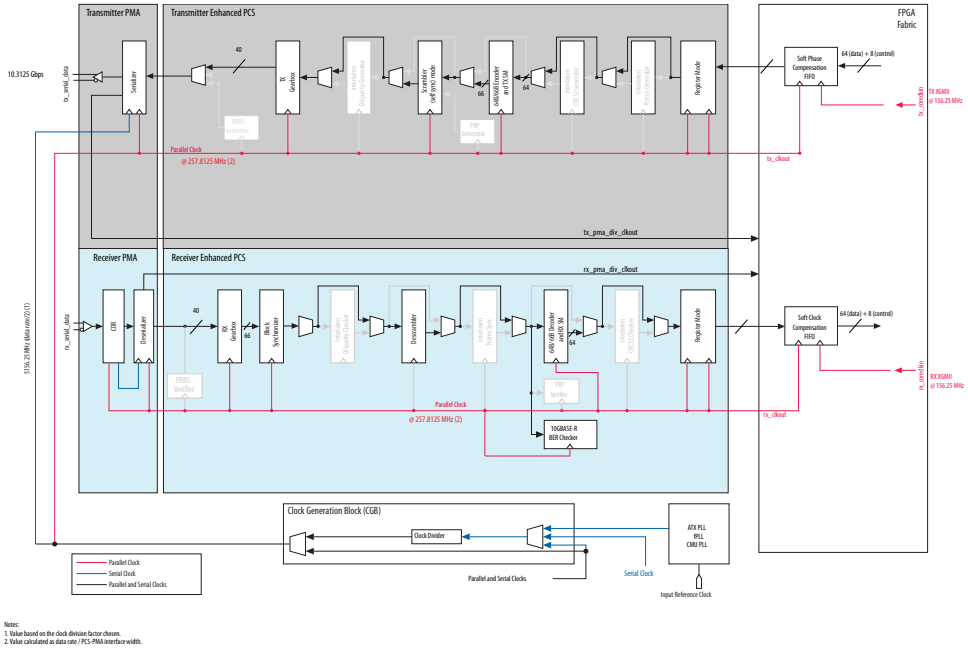
The 10GBASE-R PHY with IEEE 1588v2 creates the soft TX phase compensation FIFO and the RX clock compensation FIFO in the FPGA core so that the effective XGMII data is running at 156.25 MHz interfacing with the MAC layer.

The IEEE 1588 Precision Time Protocol (PTP) is supported by the preset of the Cyclone 10 GX transceiver Native PHY that configures 10GBASE-R PHY IP in IEEE-1588v2 mode. PTP is used for precise synchronization of clocks in applications such as:

- Distributed systems in telecommunications
- Power generation and distribution
- Industrial automation
- Robotics
- Data acquisition
- Test equipment
- Measurement

The protocol is applicable to systems communicating by local area networks including, but not limited to, Ethernet. The protocol enables heterogeneous systems that include clocks of various inherent precision, resolution, and stability to synchronize to a grandmaster clock.

Figure 45. Transceiver Channel Datapath and Clocking for 10GBASE-R with IEEE 1588v2



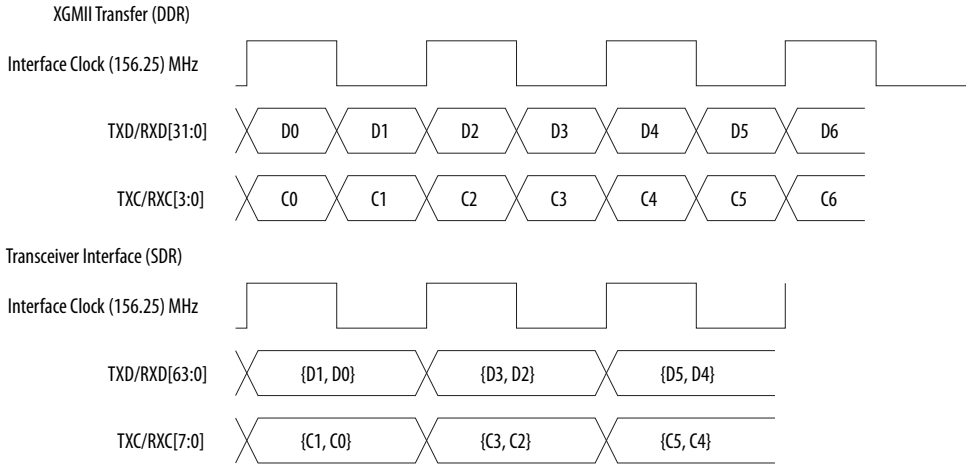
Notes:
1. Value based on the clock division factor chosen.
2. Value calculated as data rate / PCS-PMA interface width.

2.6.2.1. The XGMII Clocking Scheme in 10GBASE-R

The XGMII interface, specified by IEEE 802.3-2008, defines the 32-bit data and 4-bit wide control character. These characters are clocked between the MAC/RS and the PCS at both the positive and negative edge (double data rate – DDR) of the 156.25 MHz interface clock.

The transceivers do not support the XGMII interface to the MAC/RS as defined in the IEEE 802.3-2008 specification. Instead, they support a 64-bit data and 8-bit control single data rate (SDR) interface between the MAC/RS and the PCS.

Figure 46. XGMII Interface (DDR) and Transceiver Interface (SDR) for 10GBASE-R Configurations



Note: Clause 46 of the IEEE 802.3-2008 specification defines the XGMII interface between the 10GBASE-R PCS and the Ethernet MAC/RS.

The dedicated reference clock input to the variants of the 10GBASE-R PHY can be run at either 322.265625 MHz or 644.53125 MHz.

For 10GBASE-R, you must achieve 0 ppm of the frequency between the read clock of TX phase compensation FIFO (PCS data) and the write clock of TX phase compensation FIFO (XGMII data in the FPGA fabric). This can be achieved by using the same reference clock as the transceiver dedicated reference clock input as well as the reference clock input for a core PLL (fPLL, for example) to produce the XGMII clock. The same core PLL can be used to drive the RX XGMII data. This is because the RX clock compensation FIFO is able to handle the frequency PPM difference of ±100 ppm between RX PCS data driven by the RX recovered clock and RX XGMII data.

Note: 10GBASE-R is the single-channel protocol that runs independently. Therefore Intel recommends that you use the presets for selecting the suitable 10GBASE-R variants directly. If it is being configured through the Native PHY IP, the channel bonding option should be disabled. Enabling the channel bonding for multiple channels could degrade the link performance in terms of TX jitter eye and RX jitter tolerance.

2.6.2.1.1. TX FIFO and RX FIFO

In 10GBASE-R configuration, the TX FIFO behaves as a phase compensation FIFO and the RX FIFO behaves as a clock compensation FIFO.

In 10GBASE-R with 1588 configuration, both the TX FIFO and the RX FIFO are used in register mode. The TX phase compensation FIFO and the RX clock compensation FIFO are constructed in the FPGA fabric by the PHY IP automatically.

Related Information

Cyclone 10 GX Enhanced PCS Architecture on page 283
For more information about the Enhanced PCS Architecture

2.6.2.2. How to Implement 10GBASE-R and 10GBASE-R with IEEE 1588v2 in Intel Cyclone 10 GX Transceivers

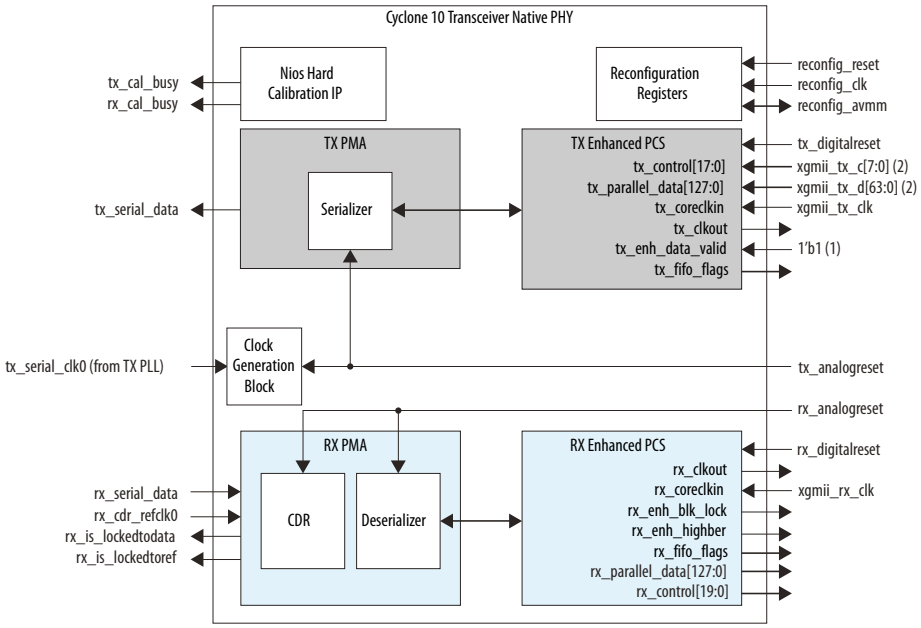
You should be familiar with the 10GBASE-R and PMA architecture, PLL architecture, and the reset controller before implementing the 10GBASE-R or 10GBASE-R with IEEE 1588v2 Transceiver Configuration Rules.

You must design your own MAC and other layers in the FPGA to implement the 10GBASE-R or 10GBASE-R with 1588 Transceiver Configuration Rule using the Native PHY IP.

1. Instantiate the **Intel Cyclone 10 GX Transceiver Native PHY IP** from the IP Catalog.
   Refer to Select and Instantiate the PHY IP Core on page 17 for more details.
2. Select **10GBASE-R** or **10GBASE-R 1588** from the **Transceiver configuration rule** list located under **Datapath Options**, depending on which protocol you are implementing.
3. Use the parameter values in the tables in Transceiver Native PHY Parameters for the 10GBASE-R Protocol as a starting point. Or, you can use the protocol presets described in Transceiver Native PHY Presets. Select **10GBASE-R Register Mode for 10GBASE-R with IEEE 1588v2**. You can then modify the settings to meet your specific requirements.
4. Click **Generate** to generate the Native PHY IP core RTL file.

Figure 47. Signals and Ports of Native PHY IP Core for the 10GBASE-R and 10GBASE-R with IEEE 1588v2

Generating the IP core creates signals and ports based on your parameter settings.

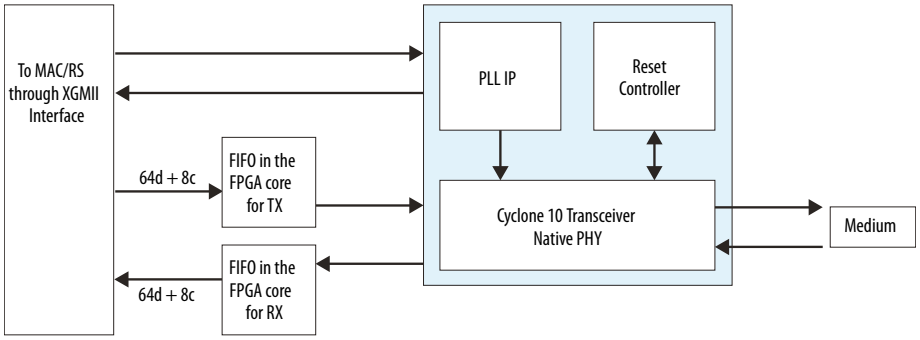


Notes:
1. For 10GBASE-R with 1588 configurations, this signal is user-controlled.
2. For 10GBASE-R with 1588 configurations, this signal is connected from the output of TX FIFO in the FPGA fabric.

5. Instantiate and configure your PLL.

6. Create a transceiver reset controller. You can use your own reset controller or use the Intel Cyclone 10 GX Transceiver Native PHY Reset Controller IP.
7. Connect the Intel Cyclone 10 GX Transceiver Native PHY to the PLL IP and the reset controller.

Figure 48. Connection Guidelines for a 10GBASE-R with IEEE 1588v2 PHY Design



8. Simulate your design to verify its functionality.

2.6.2.3. Native PHY IP Parameter Settings for 10GBASE-R and 10GBASE-R with IEEE 1588v2

This section contains the recommended parameter values for this protocol. Refer to *Using the Cyclone 10 GX Transceiver Native PHY IP Core* for the full range of parameter values.

Table 89. General and Datapath Parameters

The first two sections of the Transceiver Native PHY parameter editor provide a list of general and datapath options to customize the transceiver.

Parameter	Range
Message level for rule violations	**error**, **warning**
Transceiver Configuration Rule	**10GBASE-R** **10GBASE-R 1588**
Transceiver mode	**TX / RX Duplex**, **TX Simplex**, **RX Simplex**
Number of data channels	**1** to **12**
Data rate	**10312.5 Mbps**
Enable datapath and interface reconfiguration	**Off**
Enable simplified data interface	**On** **Off**

Table 90. TX PMA Parameters

Parameter	Range
TX channel bonding mode	**Not bonded**
TX local clock division factor	**1**, **2**, **4**, **8**
Number of TX PLL clock inputs per channel	**1**, **2**, **3**, **4**
Initial TX PLL clock input selection	**0**, **1**, **2**, **3**

Table 91. RX PMA Parameters

Parameter	Range
Number of CDR reference clocks	**1** to **5**
Selected CDR reference clock	**0** to **4**
Selected CDR reference clock frequency	**322.265625** MHz and **644.53125** MHz
PPM detector threshold	**100**, **300**, **500**, **1000**
CTLE adaptation mode	**manual**

Table 92. Enhanced PCS Parameters

Parameter	Range
Enhanced PCS/PMA interface width	**32**, **40**, **64**
FPGA fabric/Enhanced PCS interface width	**66**
Enable Enhanced PCS low latency mode	**On** **Off**
Enable RX/TX FIFO double-width mode	**Off**
TX FIFO mode	• **Phase Compensation** (10GBASE-R) • **Register or Fast register** (10GBASE-R with 1588)
TX FIFO partially full threshold	**11**
TX FIFO partially empty threshold	**2**
RX FIFO mode	• **10GBASE-R** (10GBASE-R) • **Register** (10GBASE-R with 1588)
RX FIFO partially full threshold	**23**
RX FIFO partially empty threshold	**2**

Table 93. 64B/66B Encoder and Decoder Parameters

Parameter	Range
Enable TX 64B/66B encoder	**On**
Enable RX 64B/66B decoder	**On**
Enable TX sync header error insertion	**On** **Off**

Table 94. Scrambler and Descrambler Parameters

Parameter	Range
Enable TX scrambler (10GBASE-R / Interlaken)	**On**
TX scrambler seed (10GBASE-R / Interlaken)	0x03fffffffffffff
Enable RX descrambler (10GBASE-R / Interlaken)	**On**

Table 95. Block Sync Parameters

Parameter	Range
Enable RX block synchronizer	On
Enable rx_enh_blk_lock port	On Off

Table 96. Gearbox Parameters

Parameter	Range
Enable TX data polarity inversion	On Off
Enable RX data polarity inversion	On Off

Table 97. Dynamic Reconfiguration Parameters

Parameter	Range
Enable dynamic reconfiguration	On Off
Share reconfiguration interface	On Off
Enable Native PHY Debug Master Endpoint	On Off
De-couple reconfig_waitrequest from calibration	On Off

Table 98. Configuration Files Parameters

Parameter	Range
Configuration file prefix	—
Generate SystemVerilog package file	On Off
Generate C header file	On Off
Generate MIF (Memory Initialization File)	On Off

Table 99. Generation Options Parameters

Parameter	Range
Generate parameter documentation file	On Off

2.6.2.4. Native PHY IP Ports for 10GBASE-R and 10GBASE-R with IEEE 1588v2 Transceiver Configurations

Figure 49. High BER

This figure shows the `rx_enh_highber` status signal going high when there are errors on the `rx_parallel_data` output.

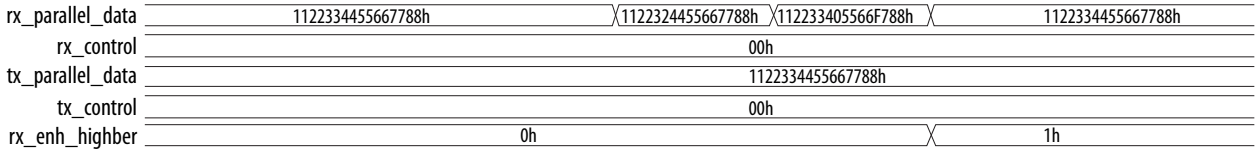
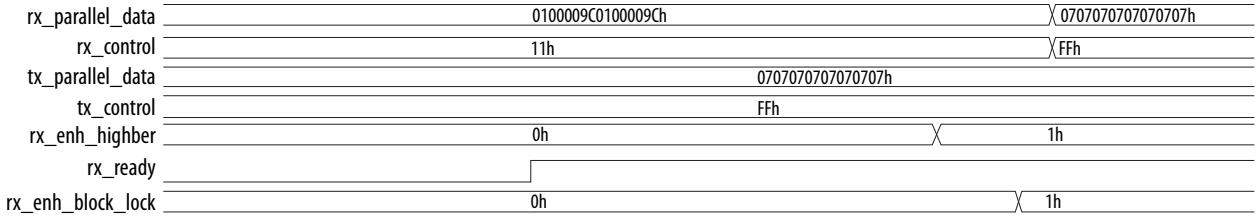


Figure 50. Block Lock Assertion

This figure shows the assertion on `rx_enh_blk_lock` signal when the Receiver detects the block delineation.



The following figures show Idle insertion and deletion.

Figure 51. IDLE Word Insertion

This figure shows the insertion of IDLE words in the receiver data stream.

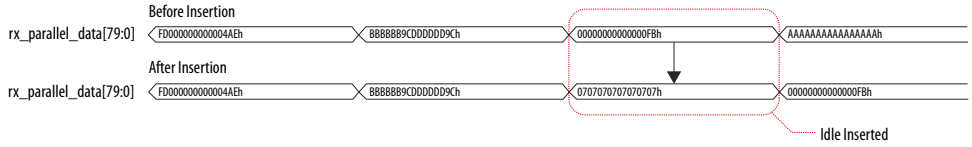


Figure 52. IDLE Word Deletion

This figure shows the deletion of IDLE words from the receiver data stream.

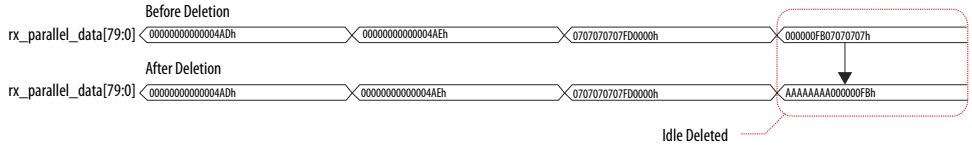
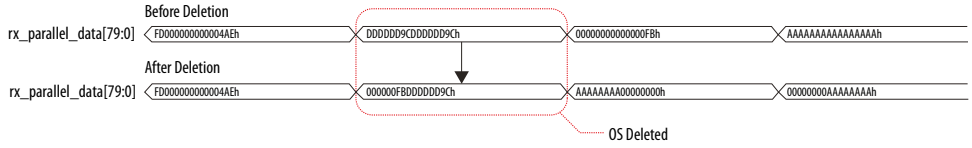


Figure 53. OS Word Deletion

This figure shows the deletion of Ordered set word in the receiver data stream.



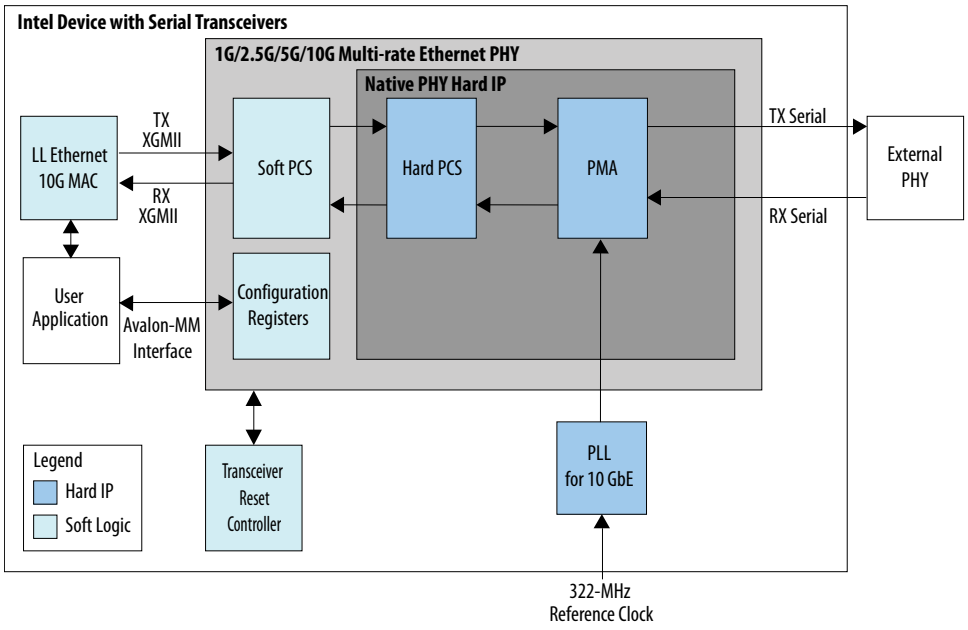
2.6.3. 1G/2.5G/5G/10G Multi-rate Ethernet PHY Intel FPGA IP Core

2.6.3.1. About the 1G/2.5G/5G/10G Multi-rate Ethernet PHY Intel FPGA IP Core

The 1G/2.5G/5G/10G Multi-rate Ethernet PHY Intel FPGA IP core implements the Ethernet protocol as defined in Clause 36 of the *IEEE 802.3 2005 Standard*. The PHY IP core consists of a physical coding sublayer (PCS) function and an embedded physical media attachment (PMA). You can dynamically switch the PHY operating speed.

Note: Intel FPGAs implement and support the required Media Access Control (MAC) and PHY (PCS+PMA) IP to interface in a chip-to-chip or chip-to-module channel with external MGBASE-T and NBASE-T PHY standard devices. You are required to use an external PHY device to drive any copper media.

Figure 54. Block Diagram of the PHY IP Core



Related Information

- Using the Cyclone 10 GX Transceiver Native PHY IP Core on page 26
- Recommended Reset Sequence on page 246
- Low Latency Ethernet 10G MAC Intel FPGA IP User Guide
  Describes the Low Latency Ethernet 10G MAC Intel FPGA IP core.
- Low Latency Ethernet 10G MAC Intel Cyclone 10 GX FPGA IP Design Example User Guide

2.6.3.1.1. Features

Table 100. PHY Features

Feature	Description
Multiple operating speeds	10M, 100M, 1G, 2.5G, 5G, and 10G.
MAC-side interface	32-bit XGMII for 10M/100M/1G/2.5G/5G/10G (USXGMII).
Network-side interface	10.3125 Gbps for 10M/100M/1G/2.5G/5G/10G (USXGMII).
Avalon Memory-Mapped (Avalon-MM) interface	Provides access to the configuration registers of the PHY.
PCS function	USXGMII PCS for 10M/100M/1G/2.5G/5G/10G.
Auto-negotiation	USXGMII Auto-negotiation supported in the 10M/100M/1G/2.5G/5G/10G (USXGMII) configuration.
Sync-E	Provides the clock for Sync-E implementation.

2.6.3.1.2. Release Information

Table 101. PHY Release Information

Item	Description
Version	19.1
Release Date	April 2019
Ordering Codes	IP-10GMRPHY
Product ID	00E4
Vendor ID	6AF7
Open Core Plus	Supported

2.6.3.1.3. Device Family Support

Table 102. Intel FPGA IP Core Device Support Levels

Device Support Level	Definition
Preliminary	Intel verifies the IP core with preliminary timing models for this device family. The IP core meets all functional requirements, but might still be undergoing timing analysis for the device family. This IP core can be used in production designs with caution.
Final	Intel verifies the IP core with final timing models for this device family. The IP core meets all functional and timing requirements for the device family. This IP core is ready to be used in production designs.

Device Family	Operating Mode	Support Level
Intel Cyclone 10 GX	10M/100M/1G/2.5G/5G/10G	Final

2.6.3.1.4. Resource Utilization

The following estimates are obtained by compiling the PHY IP core with the Intel Quartus Prime software.

Table 103. Resource Utilization

Device	Speed	ALMs	ALUTs	Logic Registers	Memory Block (M20K)
Intel Cyclone 10 GX	10M/100M/1G/ 2.5G/5G/10G (USXGMII)	700	950	1750	3

2.6.3.2. Using the IP Core

The Intel FPGA IP Library is installed as part of the Intel Quartus Prime Pro Edition installation process. You can select the 1G/2.5G/5G/10G Multi-rate Ethernet Intel FPGA IP core from the library and parameterize it using the IP parameter editor.

2.6.3.2.1. Parameter Settings

You customize the PHY IP core by specifying the parameters in the parameter editor in the Intel Quartus Prime software. The parameter editor enables only the parameters that are applicable to the selected speed.

Table 104. 1G/2.5G/5G/10G Multi-rate Ethernet PHY Intel FPGA IP Parameters

Name	Value	Description
Speed	10M/100M/1G/ 2.5G/5G/10G	The operating speed of the PHY.
Enable IEEE 1588 Precision Time Protocol	On, Off	Select this option for the PHY to provide latency information to the MAC. The MAC requires this information if it enables the IEEE 1588v2 feature. This option is enabled only for 2.5G and 1G/2.5G. *Note:* This option is not available for Intel Cyclone 10 GX devices.
Connect to MGBASE-T PHY	On, Off	Select this option when the external PHY is MGBASE-T compatible. This parameter is enabled for 2.5G, 1G/2.5G, and 1G/ 2.5G/10G (MGBASE-T) modes. *Note:* This option is not available for Intel Cyclone 10 GX devices.
Connect to NBASE-T PHY	On, Off	Select this option when the external PHY is NBASE-T compatible. This parameter is enabled for 10M/100M/1G/ 2.5G/5G/10G (USXGMII) modes.
PHY ID (32 bit)	32-bit value	An optional 32-bit unique identifier: • Bits 3 to 24 of the Organizationally Unique Identifier (OUI) assigned by the IEEE • 6-bit model number • 4-bit revision number If unused, do not change the default value, which is 0x00000000. *Note:* This option is not available for Intel Cyclone 10 GX devices.
Reference clock frequency for 10 GbE (MHz)	322.265625, 644.53125	Specify the frequency of the reference clock for 10GbE.
Selected TX PMA local clock division factor for 1 GbE	1, 2, 4, 8	This parameter is the local clock division factor in the 1G mode. It is directly mapped to the Native PHY IP Core GUI options.

continued...

Name	Value	Description
		Note: This option is not available for Intel Cyclone 10 GX devices.
Selected TX PMA local clock division factor for 2.5 GbE	1, 2	This parameter is the local clock division factor in the 2.5G mode. It is directly mapped to the Native PHY IP Core GUI options. *Note:* This option is not available for Intel Cyclone 10 GX devices.
Enable Native PHY Debug Master Endpoint (NPDME)	On, Off	Available in Native PHY and TX PLL IP parameter editors. When enabled, the Native PHY Debug Master Endpoint (NPDME) is instantiated and has access to the Avalon memory-mapped interface of the Native PHY. You can access certain test and debug functions using System Console with the NPDME. Refer to the *Embedded Debug Features* section for more details about NPDME.
Enable capability registers	On, Off	Available in Native PHY and TX PLL IP parameter editors. Enables capability registers. These registers provide high-level information about the transceiver channel's/ PLL's configuration.
Set user-defined IP identifier	User-specified	Available in Native PHY and TX PLL IP parameter editors. Sets a user-defined numeric identifier that can be read from the `user_identifier` offset when the capability registers are enabled.
Enable control and status registers	On, Off	Available in Native PHY and TX PLL IP parameter editors. Enables soft registers for reading status signals and writing control signals on the PHY/PLL interface through the NPDME or reconfiguration interface.
Enable PRBS soft accumulators	On, Off	Available in Native PHY IP parameter editor only. Enables soft logic to perform PRBS bit and error accumulation when using the hard PRBS generator and checker.

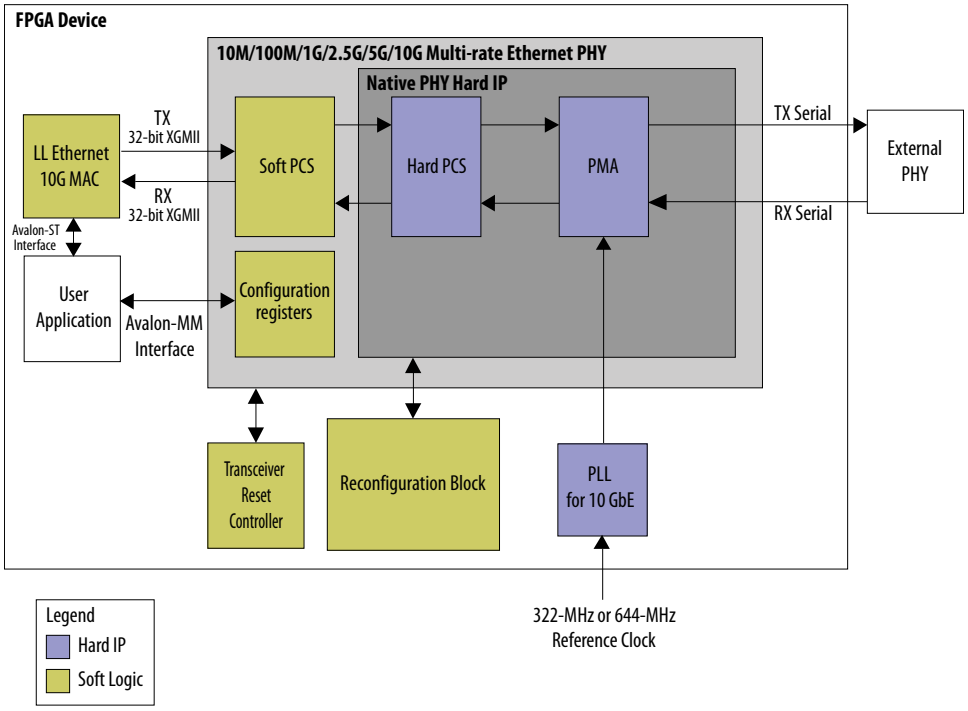
Related Information

Embedded Debug Features on page 353

2.6.3.3. Functional Description

The 1G/2.5G/5G/10G Multi-rate PHY Intel FPGA IP core for Intel Cyclone 10 GX devices implements the 10M to 10Gbps Ethernet PHY in accordance with the IEEE 802.3 Ethernet Standard. This IP core handles the frame encapsulation and flow of data between a client logic and Ethernet network via a 10M to 10GbE PCS and PMA (PHY).

Figure 55. Architecture of 10M/100M/1G/2.5G/5G/10G (USXGMII) Configuration



In the transmit direction, the PHY encodes the Ethernet frame as required for reliable transmission over the media to the remote end. In the receive direction, the PHY passes frames to the MAC.

Note: You can generate the MAC and PHY design example using the Low Latency Ethernet 10G MAC Intel FPGA IP Parameter Editor.

The IP core includes the following interfaces:

- Datapath client-interface:
  - 10M/100M/1G/2.5G/5G/10G (USXGMII)—XGMII, 32 bits
- Management interface—Avalon-MM host slave interface for PHY management.
- Datapath Ethernet interface with the following available options:
  - 10M/100M/1G/2.5G/5G/10G (USXGMII) —Single 10.3125 Gbps serial link
- Transceiver PHY dynamic reconfiguration interface—an Avalon-MM interface to read and write the Intel Cyclone 10 GX Native PHY IP core registers. This interface supports dynamic reconfiguration of the transceiver. It is used to configure the transceiver operating modes to switch to desired Ethernet operating speeds.

The 10M/100M/1G/2.5G/5G/10G (USXGMII) configuration supports the following features:

- USXGMII—10M/100M/1G/2.5G/5G/10G speeds
- Full duplex data transmission
- USXGMII Auto-Negotiation

2.6.3.3.1. Clocking and Reset Sequence

Clocking Requirements:

- For 32-bit XGMII, the 312.5 MHz clock must have zero ppm difference with reference clock of 10G transceiver PLL. Therefore, the 312.5 MHz clock must derived from the transceiver 10G reference clock for 10M/100M/1G/2.5G/5G/10G (USXGMII) variant.

The 1G/2.5G/5G/10G Multi-rate Ethernet PHY Intel FPGA IP core for Intel Cyclone 10 GX devices supports up to ±100 ppm clock frequency difference for a maximum packet length of 16,000 bytes.

2.6.3.3.2. Timing Constraints

Constrain the PHY based on the fastest speed. For 10M/100M/1G/2.5G/5G/10G (USXGMII) operating mode, constraint it based on 10G.

Table 105. Timing Constraints

PHY Configuration	Constrain PHY for
10M/100M/1G/2.5G/5G/10G (USXGMII)	10G datapath

2.6.3.3.3. Switching Operation Speed

Table 106. Supported Operating Speed

PHY Configurations	Features	10M	100M	1G	2.5G	5G	10G
10M/ 100M/1G/ 2.5G/5G/10 G (USXGMII)	Protocol	10GBASE-R 1000x data replication	10GBASE-R 100x data replication	10GBASE-R 10x data replication	10GBASE-R 4x data replication	10GBASE-R 2x data replication	10GBASE-R No data replication
	Transceiver Data Rate(21)	10.3125 Gbps	10.3125 Gbps	10.3125 Gbps	10.3125 Gbps	10.3125 Gbps	10.3125 Gbps
	MAC Interface	32-bit XGMII @ 312.5 MHz	32-bit XGMII @ 312.5 MHz	32-bit XGMII @ 312.5 MHz	32-bit XGMII @ 312.5 MHz	32-bit XGMII @ 312.5 MHz	32-bit XGMII @ 312.5 MHz

2.6.3.4. Configuration Registers

You can access the 32-bit configuration registers via the Avalon memory-mapped interface.

Observe the following guidelines when accessing the registers:

- Do not write to reserved or undefined registers.
- When writing to the registers, perform read-modify-write to ensure that reserved or undefined register bits are not overwritten.

(21) With oversampling for lower data rates.

Table 107. Types of Register Access

Access	Definition
RO	Read only.
RW	Read and write.
RWC	Read, and write and clear. The user application writes 1 to the register bit(s) to invoke a defined instruction. The IP core clears the bit(s) upon executing the instruction.

Table 108. PHY Register Definitions

Addr	Name	Description	Access	HW Reset Value
0x400	usxgmii_control	Control Register	—	—
		Bit [0]: USXGMII_ENA: • 0: 10GBASE-R mode • 1: USXGMII mode	RW	0x0
		Bit [1]: USXGMII_AN_ENA is used when USXGMII_ENA is set to 1: • 0: Disables USXGMII Auto-Negotiation and manually configures the operating speed with the USXGMII_SPEED register. • 1: Enables USXGMII Auto-Negotiation, and automatically configures operating speed with link partner ability advertised during USXGMII Auto-Negotiation.	RW	0x1
		Bit [4:2]: USXGMII_SPEED is the operating speed of the PHY in USXGMII mode and USE_USXGMII_AN is set to 0. • 3'b000: 10M • 3'b001: 100M • 3'b010: 1G • 3'b011: 10G • 3'b100: 2.5G • 3'b101: 5G • 3'b110: Reserved • 3'b111: Reserved	RW	0x0
		Bit [8:5]: Reserved	—	—
		Bit [9]: RESTART_AUTO_NEGOTIATION Write 1 to restart Auto-Negotiation sequence The bit is cleared by hardware when Auto-Negotiation is restarted.	RWC (hardware self-clear)	0x0
		Bit [15:10]: Reserved	—	—
		Bit [30:16]: Reserved	—	—
0x401	usxgmii_status	Status Register	—	—
		Bit [1:0]: Reserved	—	—
		Bit [2]: LINK_STATUS indicates link status for USXGMII all speeds • 1: Link is established • 0: Link synchronization is lost, a 0 is latched	RO	0x0
		Bit [3]: Reserved	—	—

continued...

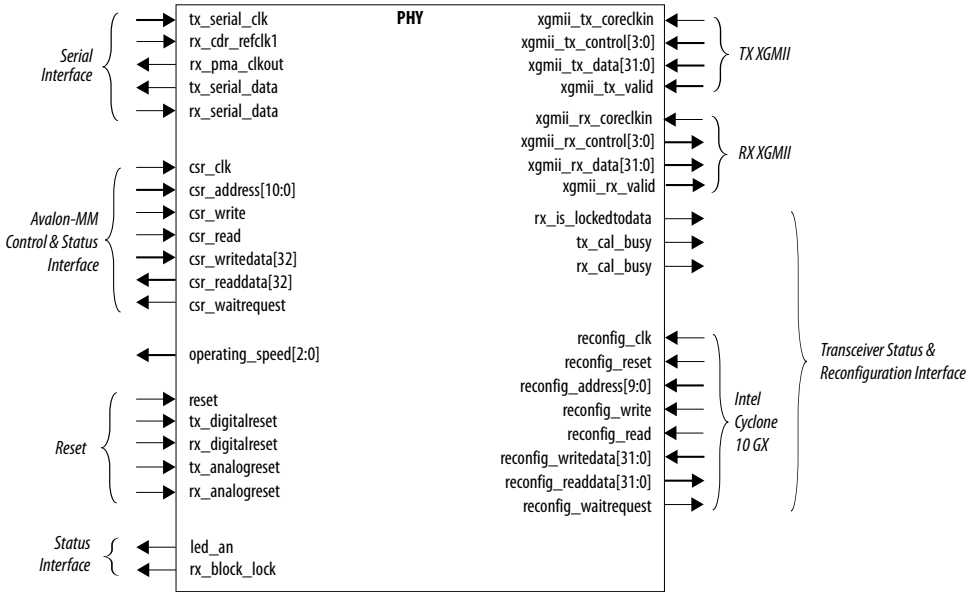
Addr	Name	Description	Access	HW Reset Value
		Bit [4]: Reserved	—	—
		Bit [5]: AUTO_NEGOTIATION_COMPLETE A value of 1 indicates the Auto-Negotiation process is completed.	RO	0x0
		Bit [15:6]: Reserved	—	—
		Bit [31:16]: Reserved	—	—
0x402:0x404	Reserved	—	—	—
0x405	usxgmii_partner_ability	Device abilities advertised to the link partner during Auto-Negotiation	—	—
		Bit [0]: Reserved	—	—
		Bit [6:1]: Reserved	—	—
		Bit [7]: EEE_CLOCK_STOP_CAPABILITY Indicates whether or not energy efficient ethernet (EEE) clock stop is supported. • 0: Not supported • 1: Supported	RO	0x0
		Bit [8]: EEE_CAPABILITY Indicates whether or not EEE is supported. • 0: Not supported • 1: Supported	RO	0x0
		Bit [11:9]: SPEED • 3'b000: 10M • 3'b001: 100M • 3'b010: 1G • 3'b011: 10G • 3'b100: 2.5G • 3'b101: 5G • 3'b110: Reserved • 3'b111: Reserved	RO	0x0
		Bit [12]: DUPLEX Indicates the duplex mode. • 0: Half duplex • 1: Full duplex	RO	0x0
		Bit [13]: Reserved	—	—
		Bit [14]: ACKNOWLEDGE A value of 1 indicates that the device has received three consecutive matching ability values from its link partner.	RO	0x0
		Bit [15]: LINK Indicates the link status. • 0: Link down • 1: Link up	RO	0x0
		Bit [31:16]: Reserved	—	—
0x406:0x411	Reserved	—	—	—

continued...

Addr	Name	Description	Access	HW Reset Value
0x412	usxgmii_link_timer	Auto-Negotiation link timer. Sets the link timer value in bit [19:14] from 0 to 2 ms in approximately 0.05 ms steps. You must program the link timer to ensure that it matches the link timer value of the external NBASE-T PHY IP Core. The reset value sets the link timer to approximately 1.6 ms. Bits [13:0] are reserved and always set to 0.	[19:14]: RW [13:0]: RO	[19:14]: 0x1F [13:0]: 0x0
0x413:0x41F	Reserved	—	—	—
0x461	phy_serial_loopback	Configures the transceiver serial loopback in the PMA from TX to RX.	—	—
		Bit [0] • 0: Disables the PHY serial loopback • 1: Enables the PHY serial loopback	RW	0x0
		Bit [15:1]: Reserved	—	—
		Bit [31:16]: Reserved	—	—

2.6.3.5. Interface Signals

Figure 56. PHY Interface Signals



2.6.3.5.1. Clock and Reset Signals

Table 109. Clock and Reset Signals

Signal Name	Direction	Width	Description
Clock signals			
csr_clk	Input	1	Clock for the control and status Avalon memory-mapped interface. Intel recommends 125 – 156.25 MHz for this clock.
xgmii_tx_coreclkin	Input	1	XGMII TX clock. Provides timing reference and 312.5 MHz for 10M/100M/1G/2.5G/5G/10G (USXGMII) mode. Synchronous to tx_serial_clk with zero ppm.
xgmii_rx_coreclkin	Input	1	XGMII RX clock. Provides timing reference and 312.5 MHz for 10M/100M/1G/2.5G/5G/10G (USXGMII) mode.
tx_serial_clk	Input	1	Serial clock from transceiver PLLs. • 10M/100M/1G/2.5G/5G/10G (USXGMII) mode: Provide 5156.25 MHz to this input port.
rx_cdr_refclk_1	Input	1	RX CDR reference clock for 10GbE. The frequency of this clock can be either 322.265625 MHz or 644.53125 MHz, as specified by the **Reference clock frequency for 10 GbE (MHz)** parameter setting.
rx_pma_clkout	Output	1	Recovered clock from CDR, operates at the following frequency: • 10GbE: 322.265625 MHz
Reset signals			
reset	Input	1	Active-high global reset. Assert this signal to trigger an asynchronous global reset.
tx_analogreset	Input	1	Connect this signal to the Transceiver PHY Reset Controller IP core. When asserted, triggers an asynchronous reset to the analog block on the TX path.
tx_digitalreset	Input	1	Connect this signal to the Transceiver PHY Reset Controller IP core. When asserted, triggers an asynchronous reset to the digital logic on the TX path.
rx_analogreset	Input	1	Connect this signal to the Transceiver PHY Reset Controller IP core. When asserted, triggers an asynchronous reset to the receiver CDR.
rx_digitalreset	Input	1	Connect this signal to the Transceiver PHY Reset Controller IP core. When asserted, triggers an asynchronous reset to the digital logic on the RX path.

2.6.3.5.2. Operating Mode and Speed Signals

Table 110. Transceiver Mode and Operating Speed Signals

Signal Name	Direction	Width	Description
operating_speed	Output	3	Connect this signal to the MAC. This signal provides the current operating speed of the PHY: • 0x0 = 10G • 0x1 = 1G • 0x2 = 100M • 0x3 = 10M • 0x4 = 2.5G • 0x5 = 5G

2.6.3.5.3. XGMII Signals

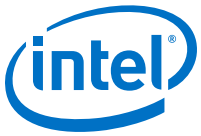
The XGMII supports 10GbE at 312.5 MHz.

Table 111. XGMII Signals

Signal Name	Direction	Width	Description
TX XGMII signals—synchronous to xgmii_tx_coreclkin			
xgmii_tx_data	Input	32	TX data from the MAC. The MAC sends the data in the following order: bits[7:0], bits[15:8], and so forth. The width is: • 32 bits for 10M/100M/1G/2.5G/5G/10G configurations.
xgmii_tx_control	Input	4	TX control from the MAC: • xgmii_tx_control[0] corresponds to xgmii_tx_data[7:0] • xgmii_tx_control[1] corresponds to xgmii_tx_data[15:8] • and so forth. The width is: • 4 bits for 10M/100M/1G/2.5G/5G/10G configurations.
xgmii_tx_valid	Input	1	Indicates valid data on xgmii_tx_control and xgmii_tx_data from the MAC. Your logic/MAC must toggle the valid data as shown below:

Speed	Toggle Rate
10M	Asserted once every 1000 clock cycles
100M	Asserted once every 100 clock cycles
1G	Asserted once every 10 clock cycles

continued...



<table>
<tr><th>Signal Name</th><th>Direction</th><th>Width</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td><table><tr><th>Speed</th><th>Toggle Rate</th></tr><tr><td>2.5G</td><td>Asserted once every 4 clock cycles</td></tr><tr><td>5G</td><td>Asserted once every 2 clock cycles</td></tr><tr><td>10G</td><td>Asserted in every clock cycle</td></tr></table></td></tr>
<tr><td colspan="4">RX XGMII signals—synchronous to <code>xgmii_rx_coreclkin</code></td></tr>
<tr><td><code>xgmii_rx_data</code></td><td>Output</td><td>32</td><td>RX data to the MAC. The PHY sends the data in the following order: bits[7:0], bits[15:8], and so forth.
The width is:
• 32 bits for 10M/100M/1G/2.5G/5G/10G (USXGMII) configurations.</td></tr>
<tr><td><code>xgmii_rx_control</code></td><td>Output</td><td>4</td><td>RX control to the MAC.
• <code>xgmii_rx_control</code>[0] corresponds to <code>xgmii_rx_data</code>[7:0]
• <code>xgmii_rx_control</code>[1] corresponds to <code>xgmii_rx_data</code>[15:8]
• and so forth.
The width is:
• 4 bits for 10M/100M/1G/2.5G/5G/10G (USXGMII) configurations.</td></tr>
<tr><td><code>xgmii_rx_valid</code></td><td>Output</td><td>1</td><td>Indicates valid data on <code>xgmii_rx_control</code> and <code>xgmii_rx_data</code> from the MAC.
The toggle rate from the PHY is shown in the table below.
Note: The toggle rate may vary when the start of a packet is received or when rate match occurs inside the PHY. You should not expect the valid data pattern to be fixed.<table><tr><th>Speed</th><th>Toggle Rate</th></tr><tr><td>10M</td><td>Asserted every 1000 clock cycles</td></tr><tr><td>100M</td><td>Asserted every 100 clock cycles</td></tr><tr><td>1G</td><td>Asserted once every 10 clock cycles</td></tr><tr><td>2.5G</td><td>Asserted once every 4 clock cycles</td></tr><tr><td>5G</td><td>Asserted once every 2 clock cycles</td></tr><tr><td>10G</td><td>Asserted in every clock cycle</td></tr></table></td></tr>
</table>

2.6.3.5.4. Status Signals

Table 112. Status Signals

Signal Name	Direction	Clock Domain	Width	Description
led_an	Output	Synchronous to rx_clkout	1	Asserted when auto-negotiation is completed.
rx_block_lock	Output	Synchronous to rx_clkout	1	Asserted when the link synchronization for 10GbE is successful.

2.6.3.5.5. Serial Interface Signals

The serial interface connects to an external device.

Table 113. Serial Interface Signals

Signal Name	Direction	Width	Description
tx_serial_data	Output	1	TX data.
rx_serial_data	Input	1	RX data.

2.6.3.5.6. Transceiver Status and Reconfiguration Signals

Table 114. Control and Status Signals

Signal Name	Direction	Width	Description
rx_is_lockedtodata	Output	1	Asserted when the CDR is locked to the RX data.
tx_cal_busy	Output	1	Asserted when TX calibration is in progress.
rx_cal_busy	Output	1	Asserted when RX calibration is in progress.
Transceiver reconfiguration signals for Cyclone 10 GX devices			
reconfig_clk	Input	1	Reconfiguration signals connected to the reconfiguration block. The reconfig_clk signal provides the timing reference for this interface.
reconfig_reset	Input	1	
reconfig_address	Input	10	
reconfig_write	Input	1	
reconfig_read	Input	1	
reconfig_writedata	Input	32	
reconfig_readdata	Output	32	
reconfig_waitrequest	Output	1	

2.6.3.5.7. Avalon Memory-Mapped Interface Signals

The Avalon memory-mapped interface is an Avalon memory-mapped interface slave port. This interface uses word addressing and provides access to the 16-bit configuration registers of the PHY.

Table 115. Avalon Memory-Mapped Interface Signals

Signal Name	Direction	Width	Description
csr_address	Input	11	Use this bus to specify the register address to read from or write to. The width is: • 11 bits for 10M/100M/1G/2.5G/5G/10G (USXGMII) configurations.
csr_read	Input	1	Assert this signal to request a read operation.
csr_readdata	Output	32	Data read from the specified register. The data is valid only when the csr_waitrequest signal is deasserted. The width is: • 32 bits for 10M/100M/1G/2.5G/5G/10G (USXGMII) configurations. The upper 16 bits are reserved.
csr_write	Input	1	Assert this signal to request a write operation.
csr_writedata	Input	32	Data to be written to the specified register. The data is written only when the csr_waitrequest signal is deasserted. The width is: • 32 bits for 10M/100M/1G/2.5G/5G/10G (USXGMII) configurations. The upper 16 bits are reserved.
csr_waitrequest	Output	1	When asserted, indicates that the PHY is busy and not ready to accept any read or write requests. • When you have requested for a read or write, keep the control signals to the Avalon memory-mapped interface constant while this signal is asserted. The request is complete when it is deasserted. • This signal can be high or low during idle cycles and reset. Therefore, the user application must not make any assumption of its assertion state during these periods.

2.6.4. Acronyms

Table 116. Ethernet Acronyms

Acronym	Definition
AN	Auto-Negotiation in Ethernet as described in Clause 73 of IEEE 802.3ap-2007.
BER	Bit Error Rate.
DME	Differential Manchester Encoding.
FEC	Forward error correction.
GMII	Gigabit Media Independent Interface.
KR	Short hand notation for Backplane Ethernet with 64b/66b encoding.
LD	Local Device.
LT	Link training in backplane Ethernet Clause 72 for 10GBASE-KR and 40GBASE-KR4.
LP	Link partner, to which the LD is connected.
MAC	Media Access Control.
MII	Media independent interface.
OSI	Open System Interconnection.

continued...

Acronym	Definition
PCS	Physical Coding Sublayer.
PHY	Physical Layer in OSI 7-layer architecture, also in Intel device scope is: PCS + PMA.
PMA	Physical Medium Attachment.
PMD	Physical Medium Dependent.
SGMII	Serial Gigabit Media Independent Interface.
WAN	Wide Area Network.

2.7. PCI Express (PIPE)

You can use Cyclone 10 GX transceivers to implement a complete PCI Express solution for Gen1 and Gen2 at data rates of 2.5 and 5.0 Gbps, respectively.

Configure the transceivers for PCIe functionality using one of the following methods:

- **Cyclone 10 GX Hard IP for PCIe**

  This is a complete PCIe solution that includes the Transaction, Data Link, and PHY/MAC layers. The Hard IP solution contains dedicated hard logic, which connects to the transceiver PHY interface.

- **Native PHY IP Core in PIPE Gen1/Gen2 Transceiver Configuration Rules**

  Use the Native PHY IP Core to configure the transceivers in PCIe mode, giving access to the PIPE interface (commonly called PIPE mode in transceivers). This mode enables you to connect the transceiver to a third-party MAC to create a complete PCIe solution.

  The PIPE specification (version 2.0) provides implementation details for a PCIe-compliant physical layer. The Native PHY IP Core for PIPE Gen1 and Gen2 supports x1, x2 or x4 operation for a total aggregate bandwidth ranging from 2 to 16 Gbps. In a x1 configuration, the PCS and PMA blocks of each channel are clocked and reset independently. The x2 and x4 configurations support channel bonding for two-lane and four-lane links. In these bonded channel configurations, the PCS and PMA blocks of all bonded channels share common clock and reset signals.

Gen1 and Gen2 modes use 8B/10B encoding, which has a 20% overhead to overall link bandwidth. Gen1 and Gen2 modes use the Standard PCS, for its operation.

Table 117. Transceiver Solutions

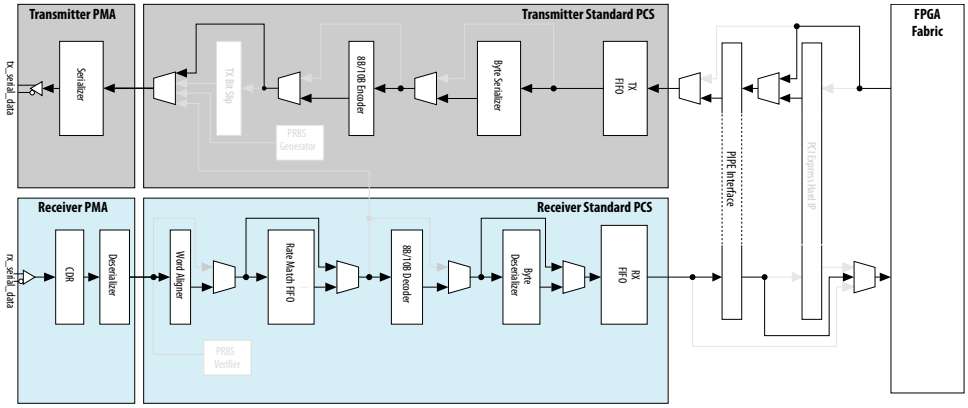
Support	Cyclone 10 GX Hard IP for PCI Express	Native PHY IP Core for PCI Express (PIPE)
Gen1 and Gen2 data rates	Yes	Yes
MAC, data link, and transaction layer	Yes	User implementation in FPGA fabric
Transceiver interface	Hard IP through PIPE 2.0 based interface	• PIPE 2.0 for Gen1 and Gen2

Related Information

PHY Interface For the PCI Express, SATA, and USB 3.1 Architectures

2.7.1. Transceiver Channel Datapath for PIPE

Figure 57. Transceiver Channel Datapath for PIPE Gen1/Gen2 Configurations



2.7.2. Supported PIPE Features

PIPE Gen1 and Gen2 configurations support different features.

Table 118. Supported Features for PIPE Configurations

Protocol Feature	Gen1 (2.5 Gbps)	Gen2 (5 Gbps)
x1, x2, x4 link configurations	Yes	Yes
PCIe-compliant synchronization state machine	Yes	Yes
Total 600 ppm clock rate compensation between transmitter reference clock and receiver reference clock	Yes	Yes
Transmitter driver electrical idle	Yes	Yes
Receiver detection	Yes	Yes
8B/10B encoding/decoding disparity control	Yes	Yes
Power state management	Yes	Yes
Receiver PIPE status encoding `pipe_rxstatus[2:0]`	Yes	Yes
Dynamic switching between 2.5 Gbps and 5 Gbps signaling rate	No	Yes
Dynamic transmitter margining for differential output voltage control	No	Yes
Dynamic transmitter buffer de-emphasis of –3.5 dB and –6 dB	No	Yes
PCS PMA interface width (bits)	10	10
Receiver Electrical Idle Inference (EII)	Your implementation in the FPGA fabric	Your implementation in the FPGA fabric

Related Information

Intel PHY Interface for the PCI Express (PIPE) Architecture

2.7.2.1. Gen1/Gen2 Features

In a PIPE configuration, each channel has a PIPE interface block that transfers data, control, and status signals between the PHY-MAC layer and the transceiver channel PCS and PMA blocks. The PIPE configuration is based on the PIPE 2.0 specification. If you use a PIPE configuration, you must implement the PHY-MAC layer in the FPGA fabric.

2.7.2.1.1. Dynamic Switching Between Gen1 (2.5 Gbps) and Gen2 (5 Gbps)

In a PIPE configuration, Native PHY IP Core provides an input signal `pipe_rate [1:0]` that is functionally equivalent to the RATE signal specified in the PCIe specification. A change in value from 2'b00 to 2'b01 on this input signal `pipe_rate [1:0]` initiates a data rate switch from Gen1 to Gen2. A change in value from 2'b01 to 2'b00 on the input signal initiates a data rate switch from Gen2 to Gen1.

2.7.2.1.2. Transmitter Electrical Idle Generation

The PIPE interface block in Cyclone 10 GX devices puts the transmitter buffer in an electrical idle state when the electrical idle input signal is asserted. During electrical idle, the transmitter buffer differential and common mode output voltage levels are compliant with the PCIe Base Specification 2.0 for both PCIe Gen1 and Gen2 data rates.

The PCIe specification requires the transmitter driver to be in electrical idle in certain power states.

Note: For more information about input signal levels required in different power states, refer to *Power State Management* section.

Related Information

Power State Management on page 124

2.7.2.1.3. Power State Management

Table 119. Power States Defined in the PCIe Specification

To minimize power consumption, the physical layer device must support the following power states.

Power States	Description
P0	Normal operating state during which packet data is transferred on the PCIe link.
P0s, P1, and P2	The PHY-MAC layer directs the physical layer to transition into these low-power states.

The PIPE interface in Cyclone 10 GX transceivers provides a `pipe_powerdown[1:0]` input port for each transceiver channel configured in a PIPE configuration.

The PCIe specification requires the physical layer device to implement power-saving measures when the P0 power state transitions to the low power states. Cyclone 10 GX transceivers do not implement these power-saving measures except for putting the transmitter buffer in electrical idle mode in the lower power states.

2.7.2.1.4. 8B/10B Encoder Usage for Compliance Pattern Transmission Support

The PCIe transmitter transmits a compliance pattern when the Link Training and Status State Machine (LTSSM) enters the Polling.Compliance substate. The Polling.Compliance substate assesses if the transmitter is electrically compliant with the PCIe voltage and timing specifications.

2.7.2.1.5. Receiver Status

The PCIe specification requires the PHY to encode the receiver status on a 3-bit status signal `pipe_rx_status[2:0]` for each channel. This status signal is used by the PHY-MAC layer for its operation. The PIPE interface block receives status signals from the transceiver channel PCS and PMA blocks, and encodes the status on the `pipe_rx_status[2:0]` signal to the FPGA fabric. The encoding of the status signals on the `pipe_rx_status[2:0]` signal conforms to the PCIe specification.

2.7.2.1.6. Receiver Detection

The PIPE interface block in Cyclone 10 GX transceivers provides an input signal `pipe_tx_detectrx_loopback[0:0]` for the receiver detect operation. The PCIe protocol requires this signal to be high during the Detect state of the LTSSM. When the `pipe_tx_detectrx_loopback[0:0]` signal is asserted in the P1 power state, the PIPE interface block sends a command signal to the transmitter driver in that channel to initiate a receiver detect sequence. In the P1 power state, the transmitter buffer must always be in the electrical idle state. After receiving this command signal, the receiver detect circuitry creates a step voltage at the output of the transmitter buffer. The time constant of the step voltage on the trace increases if an active receiver that complies with the PCIe input impedance requirements is present at the far end. The receiver detect circuitry monitors this time constant to determine if a receiver is present.

Note: For the receiver detect circuitry to function reliably, the transceiver on-chip termination must be used. Also, the AC-coupling capacitor on the serial link and the receiver termination values used in your system must be compliant with the PCIe Base Specification 2.0.

The PIPE core provides a 1-bit PHY status signal `pipe_phy_status[0:0]` and a 3-bit receiver status signal `pipe_rx_status[2:0]` to indicate whether a receiver is detected, as per the PIPE 2.0 specifications.

2.7.2.1.7. Gen1 and Gen2 Clock Compensation

In compliance with the PIPE specification, Intel Cyclone 10 GX receiver channels have a rate match FIFO to compensate for small clock frequency differences up to ±600 ppm between the upstream transmitter and the local receiver clocks.

Consider the following guidelines for PIPE clock compensation:

- Insert or delete one SKP symbol in an SKP ordered set.
- Minimum limit is imposed on the number of SKP symbols in SKP ordered set after deletion. An ordered set may have an empty COM case after deletion.
- Maximum limit is imposed on the number of the SKP symbols in the SKP ordered set after insertion. An ordered set may have more than five symbols after insertion.

- For INSERT/DELETE cases: The flag status appears on the COM symbol of the SKP ordered set where insertion or deletion occurs.
- For FULL/EMPTY cases: The flag status appears where the character is inserted or deleted.

  Note: When the PIPE interface is on, it translates the value of the flag to the appropriate `pipe_rx_status[2:0]` signal.
- The PIPE mode also has a "0 ppm" configuration option that you can use in synchronous systems. The Rate Match FIFO Block is not expected to do any clock compensation in this configuration, but latency is minimized.

Figure 58. Rate Match Deletion

This figure shows an example of rate match deletion in the case where two /K28.0/ SKP symbols must be deleted. Only one /K28.0/ SKP symbol is deleted per SKP ordered set received.

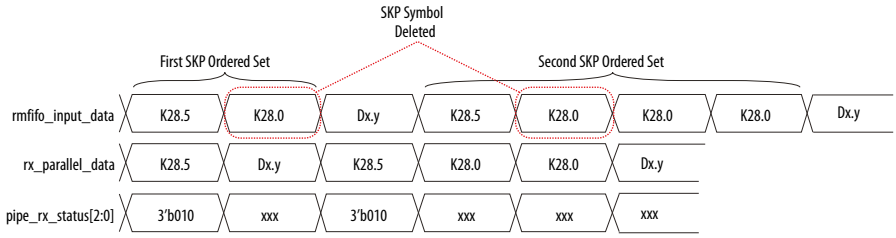


Figure 59. Rate Match Insertion

The figure below shows an example of rate match insertion in the case where two SKP symbols must be inserted. Only one /K28.0/ SKP symbol is inserted per SKP ordered set received.

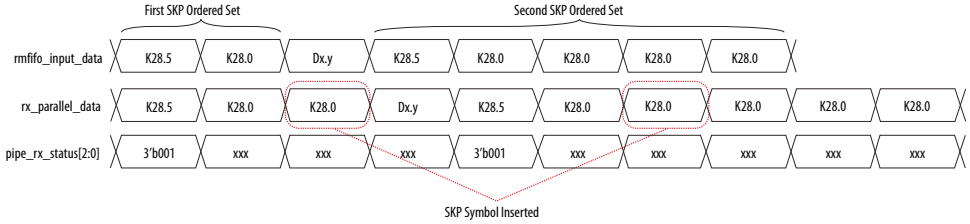


Figure 60. Rate Match FIFO Full

The rate match FIFO in PIPE mode automatically deletes the data byte that causes the FIFO to go full and drives `pipe_rx_status[2:0]` = 3'b101 synchronous to the subsequent data byte. The figure below shows the rate match FIFO full condition in PIPE mode. The rate match FIFO becomes full after receiving data byte D4.

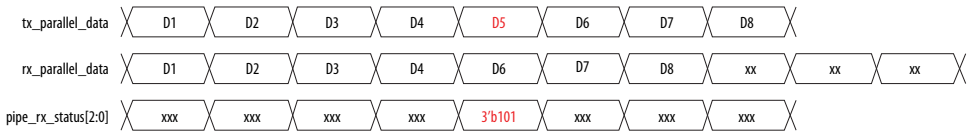
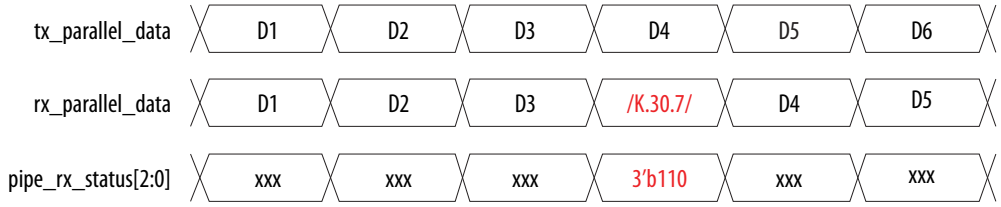


Figure 61. Rate Match FIFO Empty

The rate match FIFO automatically inserts /K30.7/ (9'h1FE) after the data byte that causes the FIFO to become empty and drives `pipe_rx_status[2:0]` = 3'b110 synchronous to the inserted /K30.7/ (9'h1FE). The figure below shows rate match FIFO empty condition in PIPE mode. The rate match FIFO becomes empty after reading out data byte D3.



PIPE 0 ppm

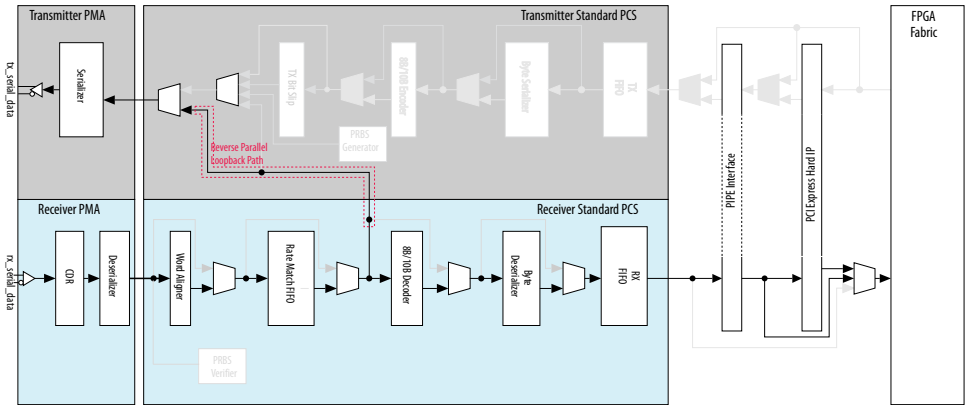
The PIPE mode also has a "0 ppm" configuration option that can be used in synchronous systems. The Rate Match FIFO Block is not expected to do any clock compensation in this configuration, but latency is minimized.

2.7.2.1.8. PCIe Reverse Parallel Loopback

PCIe reverse parallel loopback is only available in a PCIe functional configuration for Gen1 and Gen2 data rates. The received serial data passes through the receiver CDR, deserializer, word aligner, and rate matching FIFO buffer. The data is then looped back to the transmitter serializer and transmitted out through the transmitter buffer. The received data is also available to the FPGA fabric through the `rx_parallel_data` port. This loopback mode is based on PCIe specification 2.0. Cyclone 10 GX devices provide an input signal `pipe_tx_detectrx_loopback[0:0]` to enable this loopback mode.

Note: This is the only loopback option supported in PIPE configurations.

Figure 62. PCIe Reverse Parallel Loopback Mode Datapath

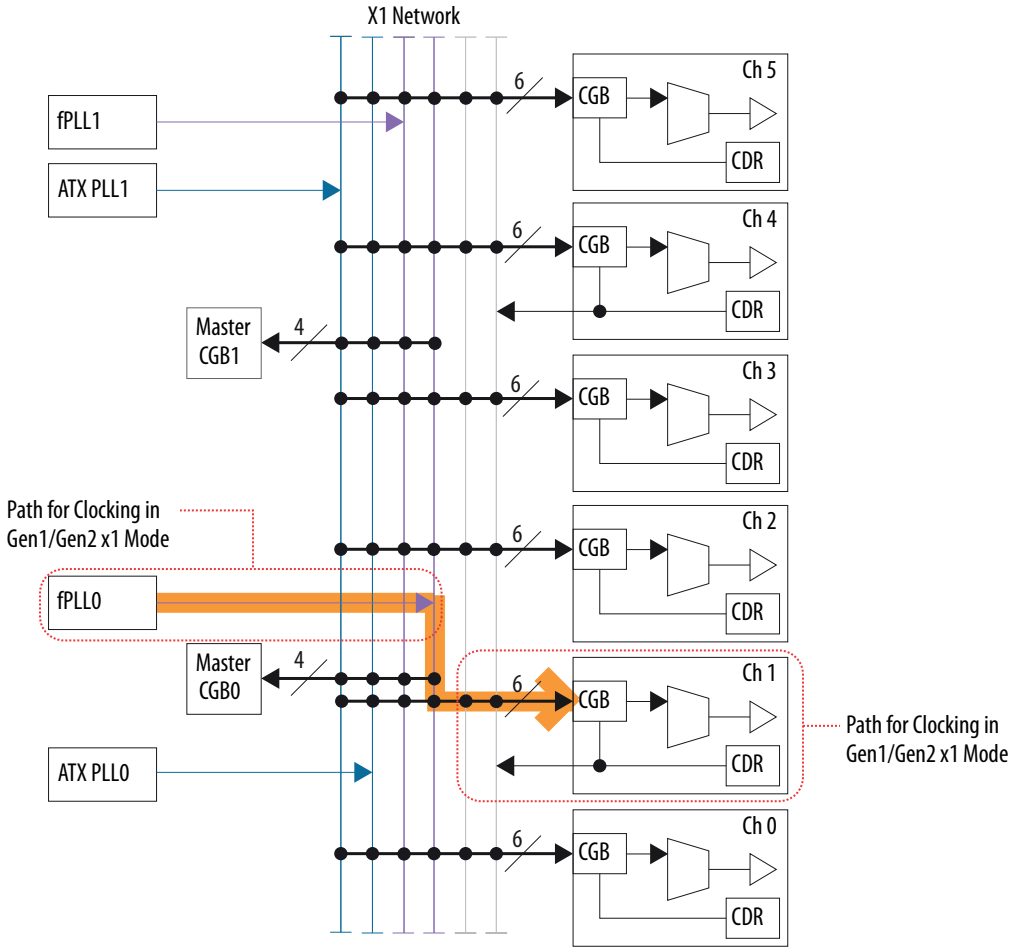


Related Information

- Cyclone 10 GX Standard PCS Architecture on page 299
- Intel PHY Interface for the PCI Express (PIPE) Architecture PCI Express 2.0

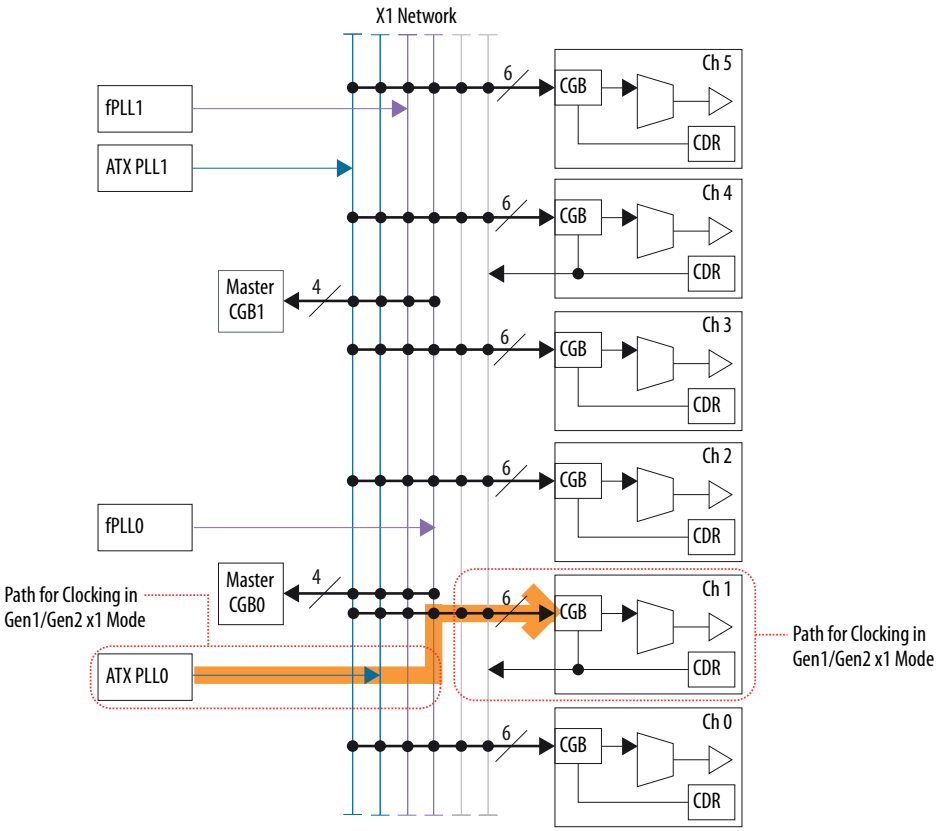
2.7.3. How to Connect TX PLLs for PIPE Gen1 and Gen2 Modes

Figure 63. Use fPLL for Gen1/Gen2 x1 Mode



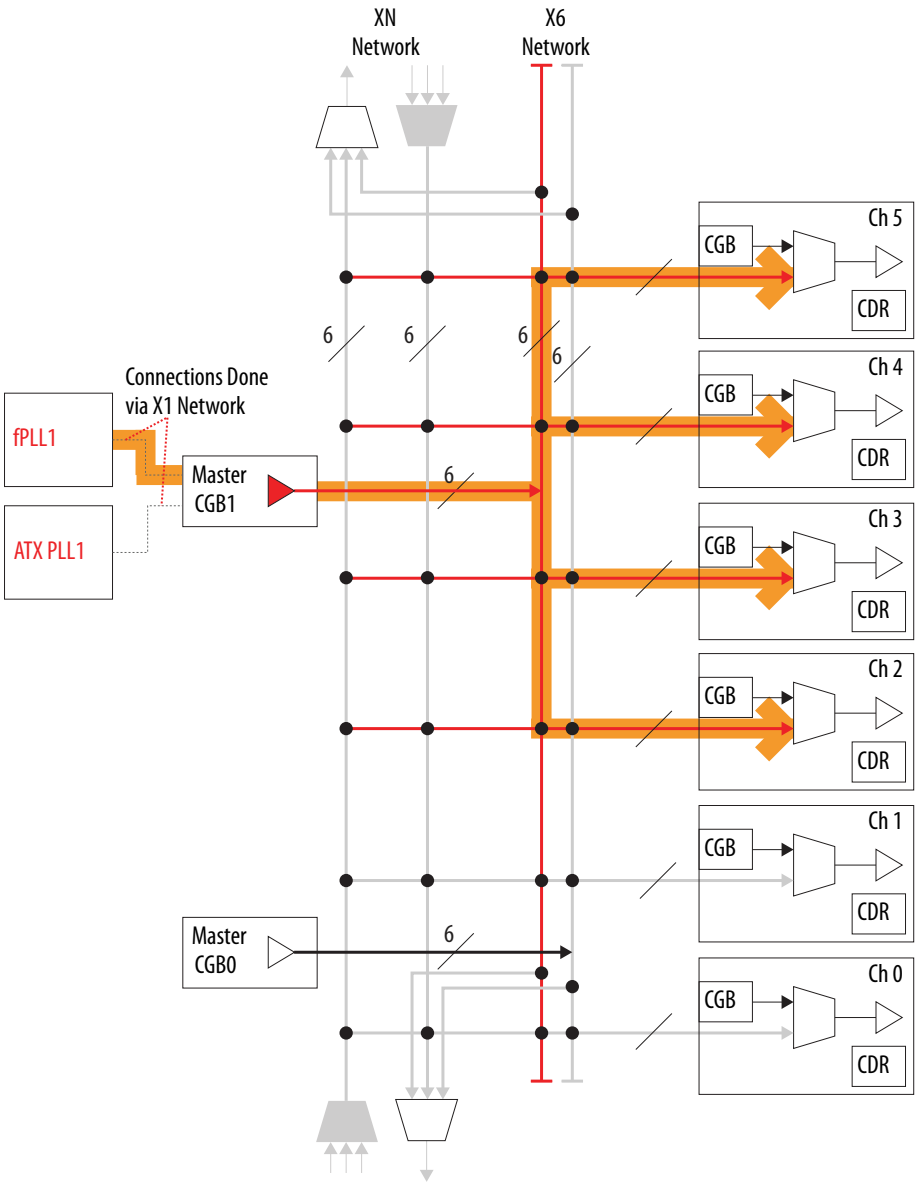
Notes:
1. The figure shown is just one possible combination for the PCIe Gen1/Gen2 x1 mode.
2. Gen1/Gen2 x1 mode uses the ATX PLL or fPLL.
3. Gen1/Gen2 x1 can use any channel from the given bank for which the ATX PLL or fPLL is enabled.
4. Connect pll_pcie_clk from either ATX PLL or fPLL to the pipe_hclk_in port on Native PHY.

Figure 64. Use ATX PLL for Gen1/Gen2 x1 Mode



Notes:
1. The figure shown is just one possible combination for the PCIe Gen1/Gen2 x1 mode.
2. Gen1/Gen2 x1 mode uses the ATX PLL or fPLL.
3. Gen1/Gen2 x1 can use any channel from the given bank for which the ATX PLL or fPLL is enabled.
4. Connect pll_pcie_clk from either ATX PLL or fPLL to the pipe_hclk_in port on Native PHY.

Figure 65. Use ATX PLL or fPLL for Gen1/Gen2 x4 Mode



Notes:

1. The figure shown is just one possible combination for the PCIe Gen1/Gen2 x4 mode.
2. The x6 and xN clock networks are used for channel bonding applications.
3. Each master CGB drives one set of x6 clock lines.
4. Gen1/Gen2 x4 modes use either ATX PLL or fPLL only.
5. Connect pll_pcie_clk from either ATX PLL or fPLL to the pipe_hclk_in port on Native PHY.
6. In this case the Master PCS channel is logical channel 3 (physical channel 4).

Related Information

- Using PLLs and Clock Networks on page 231
  For more information about implementing clock configurations and configuring PLLs.

- PIPE Design Example
  For more information about the PLL configuration for PCIe.
- Transmit PLL recommendation based on Data rates on page 200
  For more information about ATX PLL placement restrictions

2.7.4. How to Implement PCI Express (PIPE) in Cyclone 10 GX Transceivers

You must be familiar with the Standard PCS architecture, PLL architecture, and the reset controller before implementing the PCI Express protocol.

1. Go to the IP Catalog and select the **Cyclone 10 GX Transceiver Native PHY IP Core**. Refer to Select and Instantiate the PHY IP Core on page 17 for more details.
2. Select **Gen1/Gen2 PIPE** from the **Cyclone 10 GX Transceiver configuration rules** list, located under **Datapath Options**.
3. Use the parameter values in the tables in Transceiver Native PHY IP Parameters for PCI Express Transceiver Configurations Rules as a starting point. Alternatively, you can use ***Cyclone 10 GX Transceiver Native PHY Presets*** . You can then modify the settings to meet your specific requirements.
4. Click **Finish** to generate the Native PHY IP (this is your RTL file).
5. Instantiate and configure your PLL.
6. Create a transceiver reset controller. You can use your own reset controller or use the Transceiver PHY Reset Controller.
7. Connect the Native PHY IP to the PLL IP core and the reset controller. Use the information in Transceiver Native PHY IP Ports for PCI Express Transceiver Configuration Rules to connect the ports.
8. Simulate your design to verify its functionality.

2.7.5. Native PHY IP Parameter Settings for PIPE

Table 120. Parameters for Cyclone 10 GX Native PHY IP in PIPE Gen1, Gen2 Modes

This section contains the recommended parameter values for this protocol. Refer to *Using the Cyclone 10 GX Transceiver Native PHY IP Core* for the full range of parameter values.

	Gen1 PIPE	Gen2 PIPE
Parameter		
Message level for rule violations	Error	Error
Common PMA Options		
VCCR_GXB and VCCT_GXB supply voltage for the Transceiver	Gen1: 0_9V	Gen2: 0_9V
Transceiver link type	Gen1: sr	Gen2: sr
Datapath Options		
Transceiver configuration rules	Gen1 PIPE	Gen2 PIPE
PMA configuration rules	Basic	Basic
Transceiver mode	TX / RX Duplex	TX / RX Duplex
Number of data channels	Gen1 x1: 1 channel	Gen2 x1: 1 channel

continued...

	Gen1 PIPE	Gen2 PIPE
	Gen1 x2: 2 channels Gen1 x4: 4 channels	Gen2 x2: 2 channels Gen2 x4: 4 channels
Data rate	2.5 Gbps	5 Gbps
Enable datapath and interface reconfiguration	Optional	Optional
Enable simplified data interface	Optional (22)	Optional (22)
Provide separate interface for each channel	Optional	Optional

Table 121. Parameters for Cyclone 10 GX Native PHY IP in PIPE Gen1, Gen2 Modes - TX PMA

This section contains the recommended parameter values for this protocol. Refer to *Using the Cyclone 10 GX Transceiver Native PHY IP Core* for the full range of parameter values.

	Gen1 PIPE	Gen2 PIPE
TX Bonding Options		
TX channel bonding mode	Nonbonded (x1) PMA & PCS Bonding (x2 and x4)	Nonbonded (x1) PMA & PCS Bonding (x2 and x4)
PCS TX channel bonding master	Auto (23)	Auto (23)
Default PCS TX channel bonding master	Gen1 x1: Channel 0 Gen1 x2: Channel 1 Gen1 x4: Channel 2	Gen1 x1: Channel 0 Gen1 x2: Channel 1 Gen1 x4: Channel 2
TX PLL Options		
TX local clock division factor	1	1
Number of TX PLL clock inputs per channel	1	1
Initial TX PLL clock input selection	0	0
TX PMA Optional Ports		
Enable `tx_analog_reset_ack` port	Optional	Optional
Enable `tx_pma_clkout` port	Optional	Optional
Enable `tx_pma_div_clkout` port	Optional	Optional
`tx_pma_div_clkout` division factor	Optional	Optional
Enable `tx_pma_elecidle` port	Off	Off
Enable `rx_seriallpbken` port	Off	Off

(22) Refer to Table 127 on page 143 for bit settings when simplified data interface is enabled.

(23) Setting this parameter is placement-dependent. In AUTO mode, the Native PHY IP Parameter Editor selects the middle-most channel of the configuration as the default PCS TX channel bonding master. You must ensure that this selected channel is physically placed as Ch1 or Ch4 of the transceiver bank. Else, use the manual selection for the PCS TX channel bonding master to select a channel that can be physically placed as Ch1 or Ch4 of the transceiver bank. Refer to section How to place channels for PIPE configurations for more details.

Table 122. Parameters for Cyclone 10 GX Native PHY IP in PIPE Gen1, Gen2 Modes - RX PMA

This section contains the recommended parameter values for this protocol. Refer to *Using the Cyclone 10 GX Transceiver Native PHY IP Core* for the full range of parameter values.

	Gen1 PIPE	Gen2 PIPE
RX CDR Options		
Number of CDR reference clocks	1	1
Selected CDR reference clock	0	0
Selected CDR reference clock frequency	100, 125 MHz	100, 125 MHz
PPM detector threshold	1000	1000
Equalization		
CTLE adaptation mode	Manual	Manual
DFE adaptation mode	Disabled	Disabled
Number of fixed dfe taps	NA	NA
RX PMA Optional Ports		
Enable `rx_analog_reset_ack` port	Optional	Optional
Enable `rx_pma_clkout` port	Optional	Optional
Enable `rx_pma_div_clkout` port	Optional	Optional
`rx_pma_div_clkout` division factor	Optional	Optional
Enable `rx_pma_clkslip` port	Optional	Optional
Enable `rx_is_lockedtodata` port	Optional	Optional
Enable `rx_is_lockedtoref` port	Optional	Optional
Enable `rx_set_locktodata` and `rx_set_locktoref` ports	Optional	Optional
Enable `rx_seriallpbken` port	Optional	Optional
Enable PRBS Verifier Control and Status ports	Optional	Optional

Table 123. Parameters for Cyclone 10 GX Native PHY IP in PIPE Gen1, Gen2 Modes - Standard PCS

This section contains the recommended parameter values for this protocol. Refer to *Using the Cyclone 10 GX Transceiver Native PHY IP Core* for the full range of parameter values.

Parameter	Gen1 PIPE	Gen2 PIPE
Standard PCS configurations		
Standard PCS / PMA interface width	10	10
FPGA Fabric / Standard TX PCS interface width	8, 16	16
FPGA Fabric / Standard RX PCS interface width	8, 16	16
Enable Standard PCS low latency mode	Off	Off
Standard PCS FIFO		
TX FIFO mode	`low_latency`	`low_latency`

continued...

Parameter	Gen1 PIPE	Gen2 PIPE
RX FIFO mode	low_latency	low_latency
Enable tx_std_pcfifo_full port	Optional	Optional
Enable tx_std_pcfifo_empty port	Optional	Optional
Enable rx_std_pcfifo_full port	Optional	Optional
Enable rx_std_pcfifo_empty port	Optional	Optional
Byte Serializer and Deserializer		
TX byte serializer mode	Disabled, Serialize x2	Serialize x2
RX byte deserializer mode	Disabled, Serialize x2	Serialize x2
8B/10B Encoder and Decoder		
Enable TX 8B/10B encoder	Enabled	Enabled
Enable TX 8B/10B disparity control	Enabled	Enabled
Enable RX 8B/10B decoder	Enabled	Enabled
Rate Match FIFO		
Rate Match FIFO mode	PIPE, PIPE 0ppm	PIPE, PIPE 0ppm
RX rate match insert / delete -ve pattern (hex)	0x0002f17c (K28.5/K28.0/)	0x0002f17c (K28.5/K28.0/)
RX rate match insert / delete +ve pattern (hex)	0x000d0e83 (K28.5/K28.0/)	0x000d0e83 (K28.5/K28.0/)
Enable rx_std_rmfifo_full port	Optional	Optional
Enable rx_std_rmfifo_empty port	Optional	Optional
Word Aligner and Bit Slip		
Enable TX bit slip	Off	Off
Enable tx_std_bitslipboundarysel port	Optional	Optional
RX word aligner mode	Synchronous State Machine	Synchronous State Machine
RX word aligner pattern length	10	10
RX word aligner pattern (hex)	0x0000 00000000017c (/K28.5/)	0x0000 00000000017c (/K28.5/)
Number of word alignment patterns to achieve sync	3	3
Number of invalid data words to lose sync	16	16
Number of valid data words to decrement error count	15	15
Enable rx_std_wa_patternalign port	Optional	Optional
Enable rx_std_wa_a1a2size port	Off	Off
Enable rx_std_bitslipboundarysel port	Optional	Optional
Enable rx_bitslip port	Off	Off
Bit Reversal and Polarity Inversion		
Enable TX bit reversal	Off	Off
Enable TX byte reversal	Off	Off

continued...

Parameter	Gen1 PIPE	Gen2 PIPE
Enable TX polarity inversion	Off	Off
Enable `tx_polinv` port	Off	Off
Enable RX bit reversal	Off	Off
Enable `rx_std_bitrev_ena` port	Off	Off
Enable RX byte reversal	Off	Off
Enable `rx_std_byterev_ena` port	Off	Off
Enable RX polarity inversion	Off	Off
Enable `rx_polinv` port	Off	Off
Enable `rx_std_signaldetect` port	Optional	Optional
PCIe Ports		
Enable PCIe dynamic datarate switch ports	Off	Enabled
Enable PCIe `pipe_hclk_in` and `pipe_hclk_out` ports	Enabled	Enabled
Enable PCIe electrical idle control and status ports	Enabled	Enabled
Enable PCIe `pipe_rx_polarity` port	Enabled	Enabled
Dynamic reconfiguration		
Enable dynamic reconfiguration	Disabled	Disabled

Note: The signals in the left-most column are automatically mapped to a subset of a 128-bit `tx_parallel_data` word when the Simplified Interface is enabled.

Related Information

- Using the Cyclone 10 GX Transceiver Native PHY IP Core on page 26
- Bit Mappings When the Simplified Interface Is Disabled on page 143

2.7.6. fPLL IP Parameter Core Settings for PIPE

Table 124. Parameter Settings for Cyclone 10 GX fPLL IP core in PIPE Gen1, Gen2 modes

This section contains the recommended parameter values for this protocol. Refer to *Using the Cyclone 10 GX Transceiver Native PHY IP Core* for the full range of parameter values.

Parameter	Gen1 PIPE	Gen2 PIPE
PLL		
General		
fPLL mode	Transceiver	Transceiver
Protocol Mode	PCIe Gen 1	PCIe Gen 2
Message level for rule violation	Error	Error
Number of PLL reference clocks	1	1
Selected reference clock source	0	0
Enable fractional mode	Disable	Disable

continued...

Parameter	Gen1 PIPE	Gen2 PIPE
Enable manual counter configuration	Disable	Disable
Enable ATX to fPLL cascade clock input port	Disable	Disable
Settings		
Bandwidth	Low, Medium, High	Low, Medium, High
Feedback		
Operation mode	Direct	Direct
Output frequency		
Transceiver usage		
PLL output frequency	1250MHz	2500MHz
PLL datarate	2500Mbps	5000Mbps
PLL integer reference clock frequency	100 MHz, 125 MHZ	100 MHz, 125 MHZ
Master Clock Generation Block (MCGB)		
Include master clock generation block	Disable for x1 Enable for x2, x4	Disable for x1 Enable for x2, x4
Clock division factor	N/A for x1 1 for x2, x4	N/A for x1 1 for x2, x4
Enable x6/xN non-bonded high-speed clock output port	N/A for x1 Disable for x2, x4	N/A for x1 Disable for x2, x4
Enable PCIe clock switch interface	N/A for x1 Disable for x2, x4	N/A for x1 Enable for x2, x4
Number of auxiliary MCGB clock input ports	N/A for x1 0 for x2, x4	N/A for x1 0 for x2, x4
MCGB input clock frequency	1250MHz	2500MHz
MCGB output data rate	2500Mbps	5000Mbps
Bonding		
Enable bonding clock output ports	N/A for x1 design Enable for x2, x4	N/A for x1 design Enable for x2, x4
Enable feedback compensation bonding	N/A for x1 design Disable for x2, x4	N/A for x1 design Disable for x2, x4
PMA interface width	N/A for x1 design 10 for x2, x4	N/A for x1 design 10 for x2, x4
Dynamic Reconfiguration		
Enable dynamic reconfiguration	Disable	Disable
Enable Native PHY Debug Master Endpoint	Disable	Disable
Separate avmm_busy from reconfig_waitrequest	N/A	N/A
Optional Reconfiguration Logic		
Enable capability registers	N/A	N/A
Set user-defined IP identifier	N/A	N/A

continued...

Parameter	Gen1 PIPE	Gen2 PIPE
Enable control and status registers	N/A	N/A
Configuration Files		
Configuration file prefix	N/A	N/A
Generate SystemVerilog package file	N/A	N/A
Generate C Header file	N/A	N/A
Generate MIF (Memory Initialize file)	N/A	N/A
Generation Options		
Generate parameter documentation file	Enable	Enable

Related Information

Using the Cyclone 10 GX Transceiver Native PHY IP Core on page 26

2.7.7. ATX PLL IP Parameter Core Settings for PIPE

Table 125. Parameters for Cyclone 10 GX ATX PLL IP core in PIPE Gen1, Gen2 modes

This section contains the recommended parameter values for this protocol. Refer to *Using the Cyclone 10 GX Transceiver Native PHY IP Core* for the full range of parameter values.

Parameter	Gen1 PIPE	Gen2 PIPE
PLL		
General		
Message level for rule violations	Error	Error
Protocol Mode	PCIe Gen 1	PCIe Gen 2
Bandwidth	Low, medium, high	Low, medium, high
Number of PLL reference clocks	1	1
Selected reference clock source	0	0
Ports		
Primary PLL clock output buffer	GX clock output buffer	GX clock output buffer
Enable PLL GX clock output port	Enable	Enable
Enable PCIe clock output port **pll_pcie_clk**	Enable	Enable
Enable ATX to fPLL cascade clock output port	Disable	Disable
Output Frequency		
PLL output frequency	1250MHz	2500MHz
PLL output datarate	2500Mbps	5000Mbps
Enable fractional mode	Disable	Disable
PLL integer reference clock frequency	100MHz, 125MHZ	100MHz, 125MHZ
Configure counters manually	Disable	Disable
Multiple factor (M counter)	N/A	N/A
Divide factor (N counter)	N/A	N/A

continued...

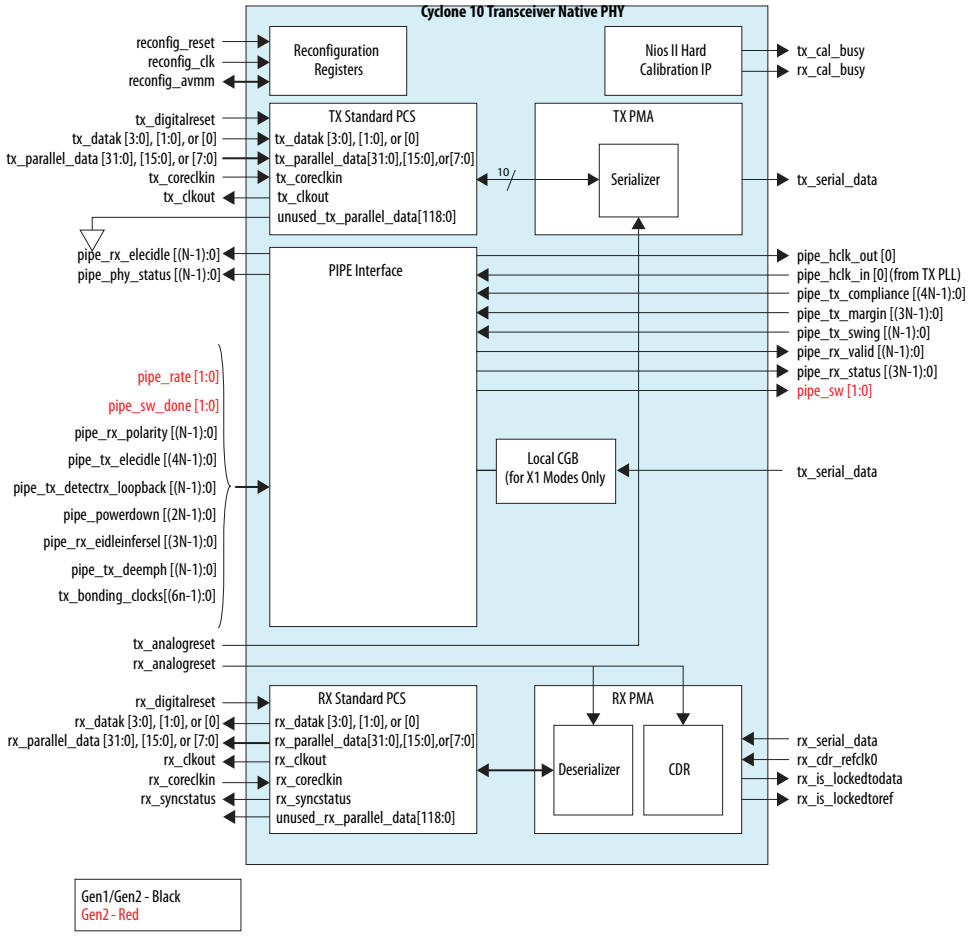
Parameter	Gen1 PIPE	Gen2 PIPE
Divide factor (L counter)	N/A	N/A
Master Clock Generation Block		
MCGB		
Include master clock generation block	Disable for x1 Enable for x2, x4	Disable for x1 Enable for x2, x4
Clock division factor	N/A for x1 1 for x2, x4	N/A for x1 1 for x2, x4
Enable x6/xN non-bnded high speed clock output port	N/A for x1 Disable for x2, x4	N/A for x1 Disable for x2, x4
Enable PCIe clock switch interface	N/A for x1 Disable for x2, x4	N/A for x1 Enable for x2, x4
Number of auxiliary MCGB clock input ports	N/A for x1 0 for x2, x4	N/A for x1 0 for x2, x4
MCGB input clock frequency	1250 MHz	2500 MHz
MCGB output data rate	2500 Mbps	5000 Mbps
Bonding		
Enable bonding clock output ports	N/A for x1 Enable for x2, x4	N/A for x1 Enable for x2, x4
Enable feedback compensation bonding	N/A for x1 design Disable for x2, x4	N/A for x1 design Disable for x2, x4
PMA interface width	N/A for x1 design 10 for x2, x4	N/A for x1 design 10 for x2, x4
Dynamic Reconfiguration		
Enable dynamic reconfiguration	Disable	Disable
Enable Native PHY Debug Master Endpoint	Disable	Disable
Separate avmm_busy from reconfig_waitrequest	N/A	N/A
Optional Reconfiguration Logic		
Enable capability registers	N/A	N/A
Set user-defined IP identifier	N/A	N/A
Enable control and status registers	N/A	N/A
Configuration Files		
Configuration file prefix	N/A	N/A
Generate SystemVerilog package file	N/A	N/A
Generate C Header file	N/A	N/A
Generate MIF (Memory Initialize file)	N/A	N/A
Generation Options		
Generate parameter documentation file	Enable	Enable

Related Information

Using the Cyclone 10 GX Transceiver Native PHY IP Core on page 26

2.7.8. Native PHY IP Ports for PIPE

Figure 66. Signals and Ports of Native PHY IP for PIPE



Note: N is the number of PCIe channels

Table 126. Ports for Cyclone 10 GX Transceiver Native PHY in PIPE Mode

This section contains the recommended settings for this protocol. Refer to *Using the Cyclone 10 GX Transceiver Native PHY IP Core* for the full range of parameter settings.

Port	Direction	Clock Domain	Description
Clocks			
rx_cdr_refclk0	In	N/A	The 100/125 MHz input reference clock source for the PHY's TX PLL and RX CDR.
tx_serial_clk0	In	N/A	The high speed serial clock generated by the PLL.
pipe_hclk_in[0]	In	N/A	The 500 MHz clock used for the Auto-Speed Negotiation (ASN) block. This clock is generated by the PLL, configured for Gen1/ Gen2.

continued...

Port	Direction	Clock Domain	Description
pipe_hclk_out[0]	Out	N/A	The 500 MHz clock output provided to the PHY - MAC interface.
PIPE Input from PHY - MAC Layer			
tx_parallel_data[15:0] or [7:0]	In	tx_coreclkin	The TX parallel data driven from the MAC. For Gen1 this can be 8 or 16 bits. For Gen2 this is 16 bits. Note: unused_tx_parallel_data should be tied to '0'. Active High. Refer to table *Bit Mappings when the Simplified Interface is Disabled* for additional details.
tx_datak[1:0] or [0]	In	tx_coreclkin	The data and control indicator for the transmitted data. For Gen1 or Gen2, when 0, indicates that tx_parallel_data is data, when 1, indicates that tx_parallel_data is control. Active High. Refer to table *Bit Mappings when the Simplified Interface is Disabled* for additional details.
pipe_tx_elecidle[(4N-1):0]	In	Asynchronous	Forces the transmit output to electrical idle. Refer to the *Intel PHY Interface for PCI Express (PIPE)* for timing diagrams. Gen1 - Width of signal is 1 bit/lane. Gen2 - Width of signal is 2 bits/lane. For example, if the MAC connected to PIPE Gen2x4 has 1bit/lane, then you can use the following mapping to connect to PIPE: {pipe_tx_elecidle[7:0] = {{2{tx_elecidle_ch3}}, {2{tx_elecidle_ch2}},{2{tx_elecidle_ch1}}, {2{tx_elecidle_ch0}}} where tx_elecidle_* is the output signal from MAC. Active High
pipe_tx_detectrx_loopback [(N-1):0]	In	tx_coreclkin	Instructs the PHY to start a receive detection operation. After power-up, asserting this signal starts a loopback operation. Refer to section 6.4 of the *Intel PHY Interface for PCI Express (PIPE)* for a timing diagram. Active High
pipe_tx_compliance[(4N-1):0]	In	tx_coreclkin	Asserted for one cycle to set the running disparity to negative. Used when transmitting the compliance pattern. Refer to section 6.11 of the *Intel PHY Interface for PCI Express (PIPE) Architecture* for more information. Gen1 - Width of signal is 1 bit/lane. Gen2 - Width of signal is 2 bits/lane. For example, if the MAC connected to PIPE Gen2x4 has 1bit/lane, then you can use the following mapping to connect to PIPE: {pipe_tx_compliance[7:0] = {{2{tx_compliance_ch3}}, {2{tx_compliance _ch2}}, {2{tx_compliance_ch1}}, {2{tx_compliance _ch0}}}. Where tx_compliance_* is the output signal from MAC. Active High

continued...

Port	Direction	Clock Domain	Description
pipe_rx_polarity[(N-1):0]	In	Asynchronous	When 1'b1, instructs the PHY layer to invert the polarity on the received data. Active High
pipe_powerdown[(2N-1):0]	In	tx_coreclkin	Requests the PHY to change its power state to the specified state. The Power States are encoded as follows: 2'b00: P0 - Normal operation. 2'b01: P0s - Low recovery time, power saving state. 2'b10: P1 - Longer recovery time, lower power state . 2'b11: P2 - Lowest power state.
pipe_tx_margin[(3N-1):0]	In	tx_coreclkin	Transmit V_{OD} margin selection. The PHY-MAC sets the value for this signal based on the value from the Link Control 2 Register. The following encodings are defined: 3'b000: Normal operating range 3'b001: Full swing: 800 - 1200 mV; Half swing: 400 - 700 mV. 3'b010:-3'b011: Reserved. 3'b100-3'b111: Full swing: 200 - 400mV; Half swing: 100 - 200 mV else reserved.
pipe_tx_swing[(N-1):0]	In	tx_coreclkin	Indicates whether the transceiver is using Full swing or Half swing voltage as defined by the pipe_tx_margin. 1'b0-Full swing. 1'b1-Half swing.
pipe_tx_deemph[(N-1):0]	In	Asynchronous	Transmit de-emphasis selection. In PCI Express Gen2 (5 Gbps) mode it selects the transmitter de-emphasis: 1'b0: –6 dB. 1'b1: –3.5 dB.
pipe_rx_eidleinfersel[(3N-1):0]	In	Asynchronous	When asserted high, the electrical idle state is inferred instead of being identified using analog circuitry to detect a device at the other end of the link. The following encodings are defined: 3'b0xx: Electrical Idle Inference not required in current LTSSM state. 3'b100: Absence of COM/SKP OS in 128 ms. 3'b101: Absence of TS1/TS2 OS in 1280 UI interval for Gen1 or Gen2. 3'b110: Absence of Electrical Idle Exit in 2000 UI interval for Gen1 and 16000 UI interval for Gen2. 3'b111: Absence of Electrical Idle exit in 128 ms window for Gen1. *Note:* Recommended to implement Receiver Electrical Idle Inference (EII) in FPGA fabric.
pipe_rate[1:0]	In	Asynchronous	The 2-bit encodings defined in the following list: 2'b00: Gen1 rate (2.5 Gbps) 2'b01: Gen2 rate (5.0 Gbps)

continued...

Port	Direction	Clock Domain	Description
pipe_sw_done[1:0]	In	N/A	Signal from the Master clock generation buffer, indicating that the rate switch has completed. Use this signal for bonding mode only (x2 and x4). For non-bonded applications (x1), this signal is internally connected to the local CGB.
PIPE Output to PHY - MAC Layer			
rx_parallel_data[15:0] or [7:0]	Out	rx_coreclkin	The RX parallel data driven to the MAC. For Gen1 this can be 8 or 16 bits. For Gen2 this is 16 bits only. Refer to *Bit Mappings When the Simplified Interface is Disabled* for more details.
rx_datak[1:0] or [0]	Out	rx_coreclkin	The data and control indicator. For Gen1 or Gen2, when 0, indicates that rx_parallel_data is data, when 1, indicates that rx_parallel_data is control.
pipe_rx_valid[(N-1):0]	Out	rx_coreclkin	Asserted when RX data and control are valid.
pipe_phy_status[(N-1):0]	Out	rx_coreclkin	Signal used to communicate completion of several PHY requests. Active High
pipe_rx_elecidle[(N-1):0]	Out	Asynchronous	When asserted, the receiver has detected an electrical idle. Active High
pipe_rx_status[(3N-1):0]	Out	rx_coreclkin	Signal encodes receive status and error codes for the receive data stream and receiver detection. The following encodings are defined: 3'b000 - Receive data OK 3'b001 - 1 SKP added 3'b010 - 1 SKP removed 3'b011 - Receiver detected 3'b100 - Either 8B/10B or 128b/130b decode error and (optionally) RX disparity error 3'b101 - Elastic buffer overflow 3'b110 - Elastic buffer underflow 3'b111 - Receive disparity error, not used if disparity error is reported using 3'b100.
pipe_sw[1:0]	Out	N/A	Signal to clock generation buffer indicating the rate switch request. Use this signal for bonding mode only (x2 and x4). For non-bonded applications (x1), this signal is internally connected to the local CGB. Active High. Refer to Table 127 on page 143 *Bit Mappings When the Simplified Interface is Disabled* for more details.

Table 127. Bit Mappings When the Simplified Interface Is Disabled

This section contains the recommended settings for this protocol. Refer to *Using the Cyclone 10 GX Transceiver Native PHY IP Core* for the full range of parameter values.

Signal Name	Gen1 (TX Byte Serializer and RX Byte Deserializer disabled)	Gen1 (TX Byte Serializer and RX Byte Deserializer in X2 mode), Gen2 (TX Byte Serializer and RX Byte Deserializer in X2 mode)
tx_parallel_data	tx_parallel_data[7:0]	tx_parallel_data[29:22,7:0]
tx_datak	tx_parallel_data[8]	tx_parallel_data[30,8]
pipe_tx_compliance	tx_parallel_data[9]	tx_parallel_data[31,9]
pipe_tx_elecidle	tx_parallel_data[10]	tx_parallel_data[32,10]
pipe_tx_detectrx_loopbacK	tx_parallel_data[46]	tx_parallel_data[46]
pipe_powerdown	tx_parallel_data[48:47]	tx_parallel_data[48:47]
pipe_tx_margin	tx_parallel_data[51:49]	tx_parallel_data[51:49]
pipe_tx_swing	tx_parallel_data[53]	tx_parallel_data[53]
rx_parallel_data	rx_parallel_data[7:0]	rx_parallel_data[39:32,7:0]
rx_datak	rx_parallel_data[8]	rx_parallel_data[40,8]
rx_syncstatus	rx_parallel_data[10]	rx_parallel_data[42,10]
pipe_phy_status	rx_parallel_data[65]	rx_parallel_data[65]
pipe_rx_valid	rx_parallel_data[66]	rx_parallel_data[66]
pipe_rx_status	rx_parallel_data[69:67]	rx_parallel_data[69:67]
pipe_tx_deemph	N/A	tx_parallel_data[52]

Refer to section 6.6 of *Intel PHY Interface for PCI Express (PIPE) Architecture* for more information.

Related Information

- Intel PHY Interface for PCI Express (PIPE) Architecture
- Bit Mappings When the Simplified Interface is Disabled on page 143
- Using the Cyclone 10 GX Transceiver Native PHY IP Core on page 26

2.7.9. fPLL Ports for PIPE

Table 128. fPLL Ports for PIPE

This section contains the recommended settings for this protocol. Refer to *Using the Cyclone 10 GX Transceiver Native PHY IP Core* for the full range of parameter settings.

Port	Direction	Clock Domain	Description
Pll_powerdown	Input	Asynchronous	Resets the PLL when asserted high. Needs to be connected to the Transceiver PHY Reset Controller pll_powerdown output.
Pll_reflck0	Input	N/A	Reference clock input port 0. There are five reference clock input ports. The number of reference clock ports available depends on the Number of PLL reference clocks parameter.

continued...

Port	Direction	Clock Domain	Description
tx_serial_clk	Output	N/A	High speed serial clock output port for GX channels. Represents the x1 clock network. For Gen1x1, Gen2x1, connect the output from this port to the tx_serial_clk[5:0] input of the native PHY IP. For Gen1x2, x4 use the tx_bonding_clocks[5:0] output port to connect to the Native PHY IP. For Gen2x2, x4 use the tx_bonding_clocks output port to connect to the Native PHY IP.
pll_locked	Output	Asynchronous	Active high status signal which indicates if PLL is locked.
pll_pcie_clk	Output	N/A	This is the hclk required for PIPE interface. For Gen1x1, x2, x4 use this port to drive the pipe_hclk_infor the PIPE interface. For Gen2x1, x2, x4 use this port to drive the pipe_hclk_in for the PIPE interface.
Pll_cal_busy	Output	Asynchronous	Status signal which is asserted high when PLL calibration is in progress. If this port is not enabled in Transceiver PHY Reset Controller, then perform logical OR with this signal and the tx_cal_busy output signal from Native PHY to input the tx_cal_busy on the reset controller IP.
Mcgb_rst	Input	Asynchronous	Master CGB reset control.
tx_bonding_clocks[5:0]]	Output	N/A	Optional 6-bit bus which carries the low speed parallel clock outputs from the Master CGB. It is used for channel bonding, and represents the x6/xN clock network. For Gen1x1, this port is disabled. For Gen1x2, x4 connect the output from this port to the tx_bonding_clocks input on Native PHY. For Gen2x1, this port is disabled. For Gen2x2, x4 connect the output from this port to the tx_bonding_clocks input on Native PHY.
pcie_sw[1:0]	Input	Asynchronous	2-bit rate switch control input used for PCIe protocol implementation. For Gen1, this port is N/A For Gen 2x2, x4 connect the pipe_sw output from Native PHY to this port.
pcie_sw_done[1:0]	Output	Asynchronous	2-bit rate switch status output used for PCIe protocol implementation. For Gen1, this port is N/A. For Gen 2x2, x4 connect the pcie_sw_done[1:0] output from fPLL to the pipe_sw_done input of Native PHY .

Related Information

Using the Cyclone 10 GX Transceiver Native PHY IP Core on page 26

2.7.10. ATX PLL Ports for PIPE

Table 129. ATX PLL Ports for PIPE

This section contains the recommended settings for this protocol. Refer to *Using the Cyclone 10 GX Transceiver Native PHY IP Core* for the full range of parameter settings.

Port	Direction	Clock Domain	Description
`Pll_powerdown`	Input	Asynchronous	Resets the PLL when asserted high. Needs to be connected to the Transceiver PHY Reset Controller `pll_powerdown` output.
`Pll_reflck0`	Input	N/A	Reference clock input port 0. There are five reference clock input ports. The number of reference clock ports available depends on the Number of PLL reference clocks parameter.
`tx_serial_clk`	Output	N/A	High speed serial clock output port for GX channels. Represents the x1 clock network. For Gen1x1, Gen2x1, connect the output from this port to the tx_serial_clk input of the native PHY IP. For Gen1x2, x4 use the `tx_bonding_clocks[5:0]` output port to connect to the Native PHY. For Gen2x2, x4 use the `tx_bonding_clocks[5:0]` output port to connect to the Native PHY.
`pll_locked`	Output	Asynchronous	Active high status signal which indicates if PLL is locked.
`pll_pcie_clk`	Output	N/A	This is the hclk required for PIPE interface. For Gen1x1,x2,x4 use this port to drive the `pipe_hclk_in` on the PIPE interface. For Gen2x1,x2,x4 use this port to drive the `pipe_hclk_in` on the PIPE interface.
`Pll_cal_busy`	Output	Asynchronous	Status signal which is asserted high when PLL calibration is in progress. If this port is not enabled in the Transceiver PHY Reset Controller, then perform logical OR with this signal and the `tx_cal_busy` output signal from Native PHY to input the `tx_cal_busy` on the reset controller IP.
`Mcgb_rst`	Input	Asynchronous	Master CGB reset control.
`tx_bonding_clocks[5:0]`	Output	N/A	Optional 6-bit bus which carries the low speed parallel clock outputs from the Master CGB. Used for channel bonding, and represents the x6/xN clock network. For Gen1x1, this port is disabled. For Gen1x2,x4 connect the output from this port to the `tx_bonding_clocks[5:0]` input on Native PHY. For Gen2x1, this port is disabled For Gen2x2,x4 connect the output from this port to `tx_bonding_clocks[5:0]` input on Native PHY.
`pcie_sw[1:0]`	Input	Asynchronous	2-bit rate switch control input used for PCIe protocol implementation. For Gen1, this port is N/A. For Gen 2x2,x4 connect the `pipe_sw[1:0]` output from Native PHY to this port.
`pcie_sw_done[1:0]`	Output	Asynchronous	2-bit rate switch status output used for PCIe protocol implementation. For Gen1, this port is N/A. For Gen2x2, x4 connect the `pcie_sw_done[1:0]` output from ATX PLL to `pipe_sw_done` input of Native PHY .

Related Information

Using the Cyclone 10 GX Transceiver Native PHY IP Core on page 26

2.7.11. How to Place Channels for PIPE Configurations

Instead of the fitter or software model, the hardware dictates all the placement restrictions. The restrictions are listed below:

- The channels must be contiguous for bonded designs.
- The master CGB is the only way to access x6 lines and must be used in bonded designs. The local CGB cannot be used to route clock signals to slave channels because the local CGB does not have access to x6 lines.

For ATX PLL placement restrictions, refer to the section "Transmit PLL Recommendations Based on Data Rates" of *PLLs and Clock Networks* chapter.

Related Information

PLLs and Clock Networks on page 198

2.7.11.1. Master Channel in Bonded Configurations

For PCIe, both the PMA and PCS must be bonded. There is no need to specify the PMA Master Channel because of the separate Master CGB in the hardware. However, you must specify the PCS Master Channel through the Native PHY. You can choose any one of the data channels (part of the bonded group) as the logical PCS Master Channel.

Note: Whichever channel you pick as the PCS master, the fitter selects physical CH1 or CH4 of a transceiver bank as the master channel. This is because the Auto-Speed Negotiation (ASN) block and Master CGB connectivity only exists in the hardware of these two channels of the transceiver bank.

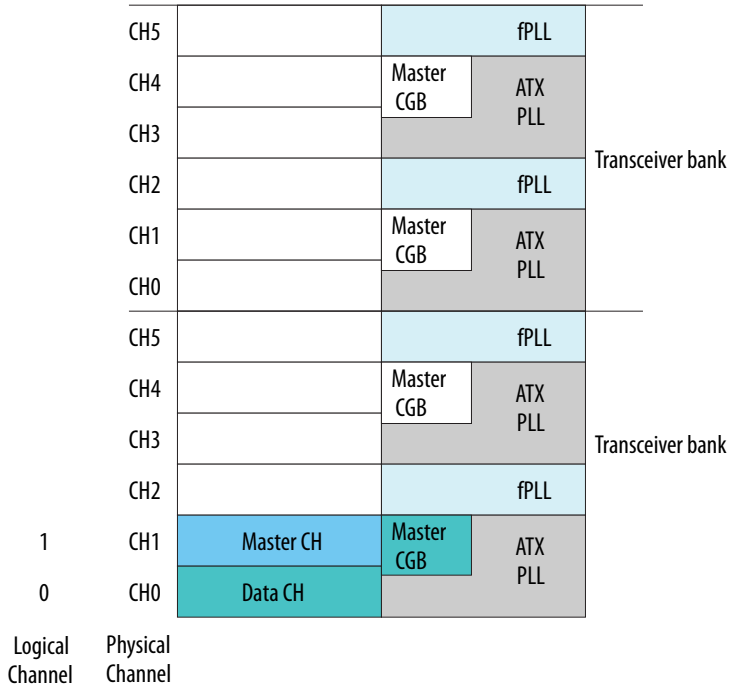
Table 130. Logical PCS Master Channel for PIPE Configuration

PIPE Configuration	Logical PCS Master Channel # (default)
x1	0[24]
x2	1[24]
x4	2[24]

[24] Ensure that the Logical PCS Master Channel aligns with Physical Channel 1 or 4 in a given transceiver bank.

The following figures show the default configurations:

Figure 67. x2 Configuration



Note: The physical channel 0 aligns with logical channel 0. The logical PCS Master Channel 1 is specified as Physical Channel 1.

Figure 68. x4 Configuration

The figure below shows an alternate way of placing 4 bonded channels. In this case, the logical PCS Master Channel number 2 must be specified as Physical channel 4.

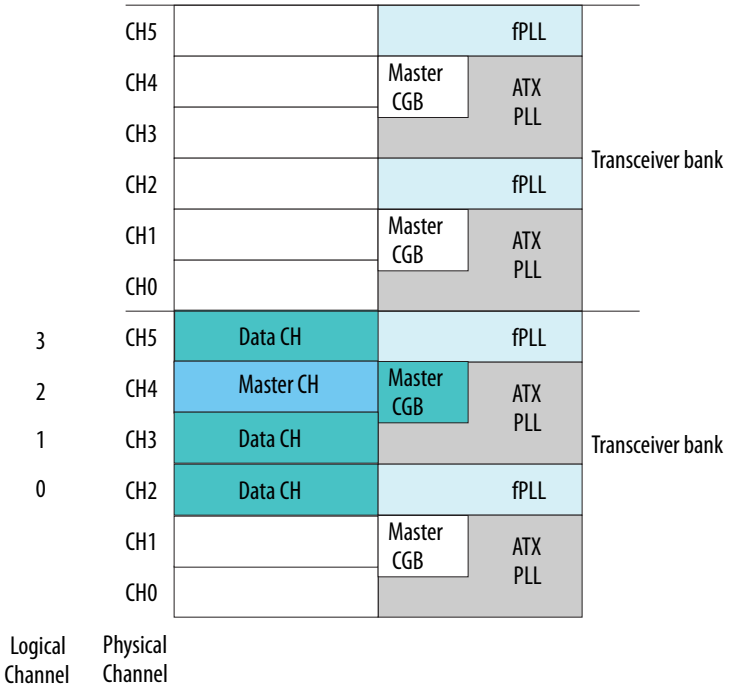
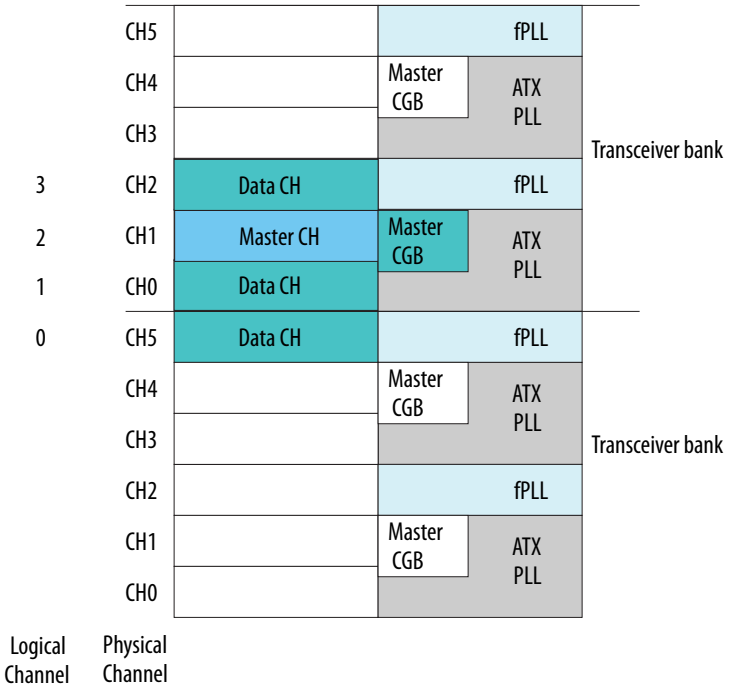


Figure 69. x4 Alternate Configuration

The figure below shows an alternate way of placing 4 bonded channels. In this case, the logical PCS Master Channel number 2 must be specified as Physical channel 1.



As indicated in the figures above, the fitter picks either physical CH1 or CH4 as the PCS master in bonded configurations for PIPE.

2.8. CPRI

The common public radio interface (CPRI) is a high-speed serial interface developed for wireless network radio equipment controller (REC) to uplink and downlink data from available remote radio equipment (RE).

The CPRI protocol defines the interface of radio base stations between the REC and the RE. The physical layer supports both the electrical interfaces (for example, traditional radio base stations) and the optical interface (for example, radio base stations with a remote radio head). The scope of the CPRI specification is restricted to the link interface only, which is a point-to-point interface. The link has all the features necessary to enable a simple and robust usage of any given REC and RE network topology, including a direct interconnection of multiport REs.

2.8.1. Transceiver Channel Datapath and Clocking for CPRI

Figure 70. Transceiver Channel Datapath and Clocking for CPRI

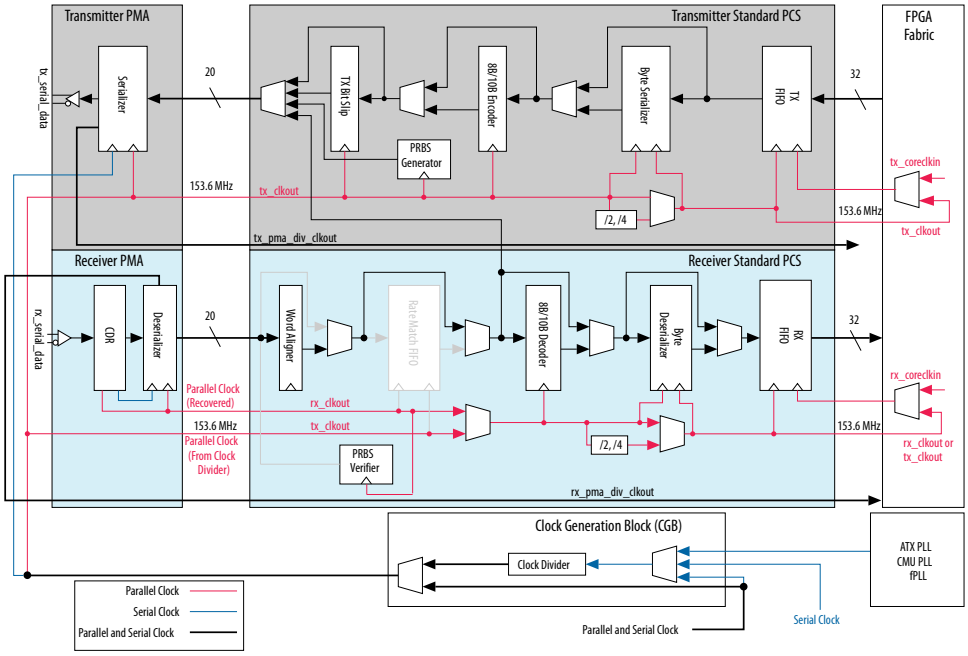


Table 131. Channel Width Options for Supported Serial Data Rates

Serial Data Rate (Mbps)	Channel Width (FPGA-PCS Fabric)			
	8/10 Bit Width		16/20 Bit Width	
	8-Bit	16-Bit	16-Bit	32-Bit
614.4[25]	Yes	Yes	N/A	N/A
1228.8	Yes	Yes	Yes	Yes
2457.6	Yes	Yes	Yes	Yes
3072	Yes	Yes	Yes	Yes
4915.2	N/A	N/A	Yes	Yes
6144	N/A	N/A	Yes	Yes

2.8.1.1. TX PLL Selection for CPRI

Choose a transmitter PLL that fits your required data rate.

Table 132. TX PLL Supported Data Rates

ATX and fPLL support the clock bonding feature.

TX PLLs	Supported Data Rate (Mbps)
ATX	614.4, 1228.8, 2457.6, 3072, 4915.2, 6144
fPLL	614.4, 1228.8, 2457.6, 3072, 4915.2, 6144
CMU	614.4, 1228.8, 2457.6, 3072, 4915.2, 6144

Note:

- Channels that use the CMU PLL cannot be bonded. The CMU PLL that provides the clock can only drive channels in the transceiver bank where it resides.
- Over-sampling is required to implement 614.4 Mbps.

2.8.1.2. Auto-Negotiation

When auto-negotiation is required, the channels initialize at the highest supported frequency and switch to successively lower data rates if frame synchronization is not achieved. If your design requires auto-negotiation, choose a base data rate that minimizes the number of PLLs required to generate the clocks required for data transmission.

By selecting an appropriate base data rate, you can change data rates by changing the local clock generation block (CGB) divider. If a single base data rate is not possible, you can use an additional PLL to generate the required data rates.

Table 133. Recommended Base Data Rates and Clock Generation Blocks for Available Data Rates

Data Rate (Mbps)	Base Data Rate (Mbps)	Local CGB Divider
1228.8	4915.2	4
2457.6	4915.2	2

continued...

[25] Over-sampling is required to implement 614.4Mbps

Data Rate (Mbps)	Base Data Rate (Mbps)	Local CGB Divider
3072.0	6144.0	2
4915.2	4915.2	1
6144.0	6144.0	1

2.8.2. Supported Features for CPRI

The CPRI protocol places stringent requirements on the amount of latency variation that is permissible through a link that implements these protocols.

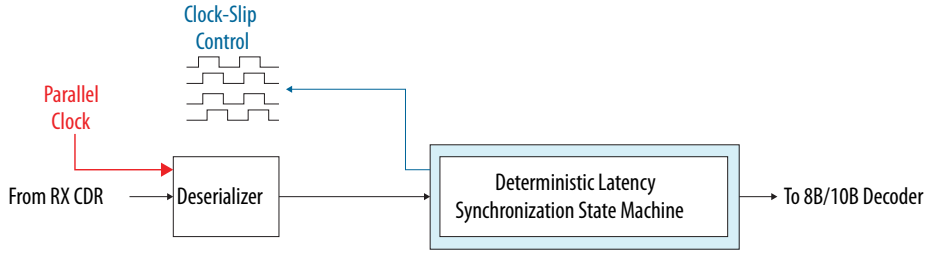
CPRI (Auto) and CPRI (Manual) transceiver configuration rules are both available for CPRI designs. Both modes use the same functional blocks, but the configuration mode of the word aligner is different between the Auto and Manual modes. In CPRI (Auto) mode, the word aligner works in deterministic mode. In CPRI (Manual) mode, the word aligner works in manual mode.

To avoid transmission interference in time division multiplexed systems, every radio in a cell network requires accurate delay estimates with minimal delay uncertainty. Lower delay uncertainty is always desired for increased spectrum efficiency and bandwidth. The Cyclone 10 GX devices are designed with features to minimize the delay uncertainty for both RECs and REs.

2.8.2.1. Word Aligner in Deterministic Latency Mode for CPRI

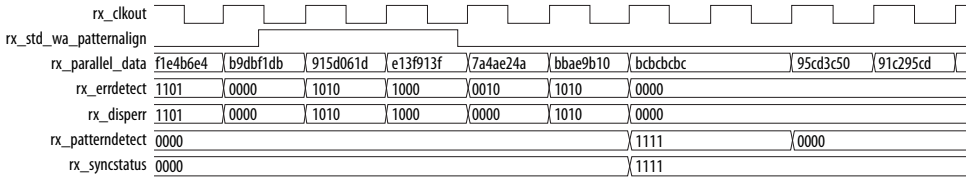
The deterministic latency state machine in the word aligner reduces the known delay variation from the word alignment process. It automatically synchronizes and aligns the word boundary by slipping one half of a serial clock cycle (1UI) in the deserializer. Incoming data to the word aligner is aligned to the boundary of the word alignment pattern (K28.5).

Figure 71. Deterministic Latency State Machine in the Word Aligner



When using deterministic latency state machine mode, assert `rx_std_wa_patternalign` to initiate the pattern alignment after the reset sequence is complete. This is an edge-triggered signal in all cases except one: when the word aligner is in manual mode and the PMA width is 10, in which case `rx_std_wa_patternalign` is level sensitive.

Figure 72. Word Aligner in Deterministic Mode Waveform



Related Information

Word Aligner on page 305

2.8.2.1.1. Transmitter and Receiver Latency

The latency variation from the link synchronization function (in the word aligner block) is deterministic with the `rx_bitslipboundaryselectout` port. Additionally, you can use the `tx_bitslipboundaryselect` port to fix the round trip transceiver latency for port implementation in the remote radio head to compensate for latency variation in the word aligner block. The `tx_bitslipboundaryselect` port is available to control the number of bits to be slipped in the transmitter serial data stream. You can optionally use the `tx_bitslipboundaryselect` port to round the round-trip latency to a whole number of cycles.

When using the byte deserializer, additional logic is required in the FPGA fabric to determine if the comma byte is received in the lower or upper byte of the word. The delay is dependent on the word in which the comma byte appears.

Note: Latency numbers are pending device characterization.

2.8.3. Word Aligner in Manual Mode for CPRI

When configuring the word aligner in CPRI (Manual), the word aligner parses the incoming data stream for a specific alignment character. After `rx_digitalreset` deasserts, asserting the `rx_std_wa_patternalign` triggers the word aligner to look for the predefined word alignment pattern or its complement in the received data stream. It is important to note that the behavior of the word aligner in Manual mode operates in different ways depending on the PCS-PMA interface width.

Table 134. Word Aligner Signal Status Behaviors in Manual Mode

PCS-PMA Interface Width	`rx_std_wa_patternalign` Behavior	`rx_syncstatus` Behavior	`rx_patterndetect` Behavior
10	Level sensitive	One parallel clock cycle (When three control patterns are detected)	One parallel clock cycle
20	Edge sensitive	Remains asserted until next rising edge of `rx_std_wa_patternalign`	One parallel clock cycle

PCS-PMA Width = 10

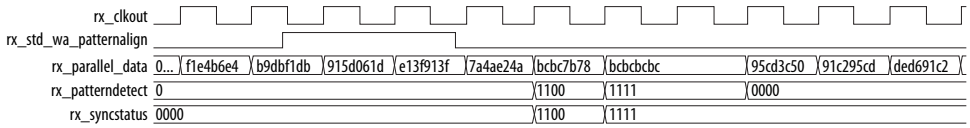
When the PCS-PMA interface width is 10, 3 consecutive word alignment patterns found after the initial word alignment in a different word boundary causes the word aligner to resynchronize to this new word boundary if the `rx_std_wa_patternalign` remains asserted; `rx_std_wa_patternalign` is level sensitive. If you deassert

rx_std_wa_patternalign, the word aligner maintains the current word boundary even when it finds the alignment pattern in a new word boundary. When the word aligner is synchronized to the new word boundary, rx_patterndetect and rx_syncstatus are asserted for one parallel clock cycle.

PCS-PMA Width =20

When the PMA-PCS width is 20, any alignment pattern found after the initial alignment in a different word boundary causes the word aligner to resynchronize to this new word boundary on the rising edge of rx_std_wa_patternalign; rx_std_wa_patternalign is edge sensitive. The word aligner maintains the current word boundary until the next rising edge of rx_std_wa_patternalign. When the word aligner is synchronized to the new word boundary, rx_patterndetect asserts for one parallel clock cycle, and rx_syncstatus remains asserted until the next rising edge of rx_std_wa_patternalign.

Figure 73. Word Aligner in Manual Alignment Mode Waveform



Related Information

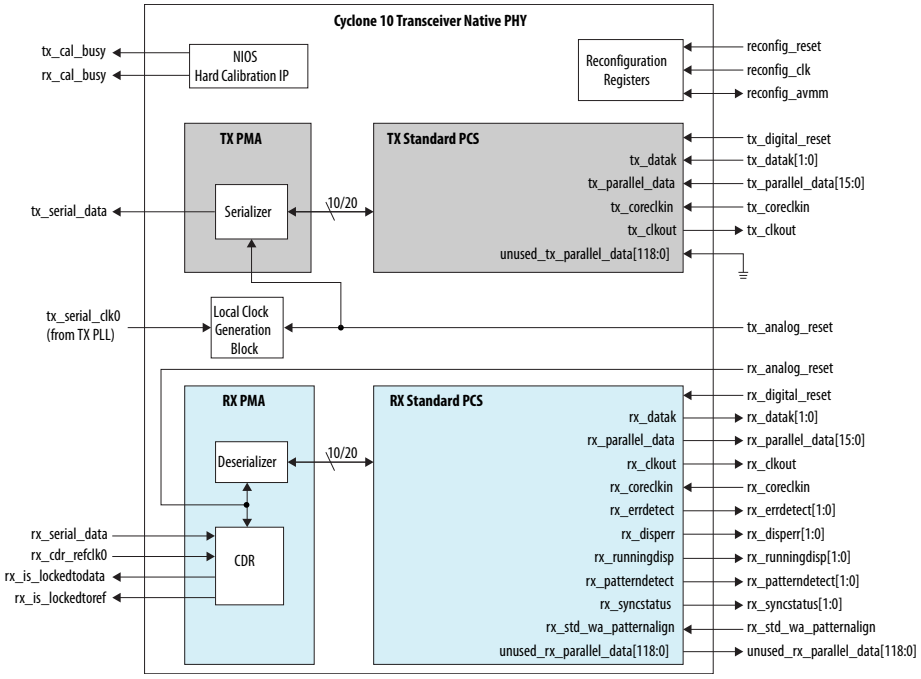
Word Aligner on page 305

2.8.4. How to Implement CPRI in Cyclone 10 GX Transceivers

You should be familiar with the Standard PCS and PMA architecture, PLL architecture, and the reset controller before implementing your CPRI protocol.

1. Instantiate the **Cyclone 10 Transceiver Native PHY IP** from the IP Catalog. Refer to Select and Instantiate the PHY IP Core on page 17 for more details.
2. Select **CPRI (Auto)** or **CPRI (Manual)** from the **Transceiver configuration rules** list located under **Datapath Options**, depending on which protocol you are implementing.
3. Use the parameter values in the tables in Native PHY IP Parameter Settings for CPRI on page 155 as a starting point. Or, you can use the protocol presets described in Preset Configuration Options. You can then modify the setting to meet your specific requirements.
4. Click **Generate** to generate the Native PHY IP (this is your RTL file).

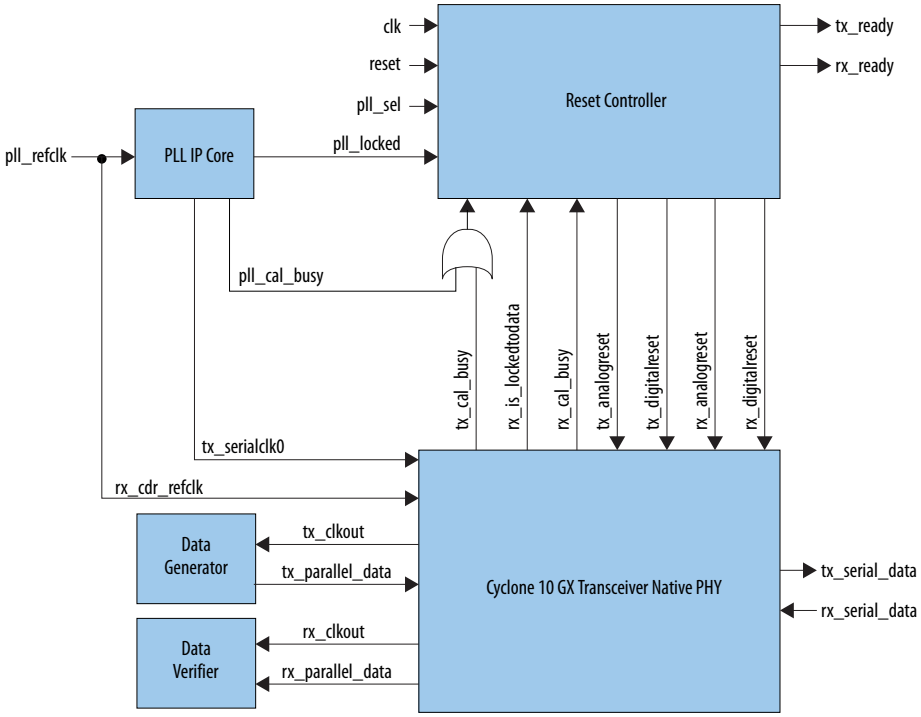
Figure 74. Signals and Ports of Native PHY IP for CPRI



5. Instantiate and configure your PLL.
6. Create a transceiver reset controller.

   You can use your own reset controller or use the Native PHY Reset Controller IP.
7. Connect the Native PHY IP to the PLL IP core and the reset controller. Use the information in the following figure to connect the ports.

Figure 75. Connection Guidelines for a CPRI PHY Design



8. Simulate your design to verify its functionality.

2.8.5. Native PHY IP Parameter Settings for CPRI

Table 135. General and Datapath Options

The first two sections of the Parameter Editor for the Native PHY IP provide a list of general and datapath options to customize the transceiver.

Parameter	Value
Message level for rule violations	**error** **warning**
Transceiver configuration rules	**CPRI (Auto)** **CPRI (Manual)**
PMA configuration rules	**basic**
Transceiver mode	**TX/RX Duplex**
Number of data channels	**1-12**
Data rate	1228.8 Mbps 2457.6 Mbps 3072 Mbps 4915.2 Mbps 6144 Mbps
Enable datapath and interface reconfiguration	**Off**
Enable simplified data interface	**On**

Table 136. TX PMA Parameters

Parameter	Value
TX channel bonding mode	Not Bonded / PMA Bonding Only / PMA and PCS Bonding
TX local clock division factor	1
Number of TX PLL clock inputs per channel	1
Initial TX PLL clock input selection	0
Enable tx_pma_clkout port	Off
Enable tx_pma_div_clkout port	On
tx_pma_div_clkout division factor	2
Enable tx_pma_elecidle port	Off
Enable rx_seriallpbken port	Off

Table 137. RX PMA Parameters

Parameter	Value
Number of CDR reference clocks	1
Selected CDR reference clock	0
Selected CDR reference clock frequency	Select legal range defined by the Quartus Prime software
PPM detector threshold	1000
CTLE adaptation mode	manual
Enable rx_pma_clkout port	Off
Enable rx_pma_div_clkout port	On
rx_pma_div_clkout division factor	2
Enable rx_pma_clkslip port	Off
Enable rx_is_lockedtodata port	On
Enable rx_is_lockedtoref port	On
Enable rx_set_locktodata and rx_set_locktoref ports	Off
Enable rx_seriallpbken port	Off
Enable PRBS verifier control and status ports	Off

Table 138. Standard PCS Parameters

Parameters	Value
Standard PCS / PMA interface width	20
FPGA fabric / Standard TX PCS interface width	32
FPGA fabric / Standard RX PCS interface width	32
Enable 'Standard PCS' low latency mode	Off
TX FIFO mode	register_fifo
RX FIFO mode	register_fifo

continued...

Parameters	Value
Enable tx_std_pcfifo_full port	**Off**
Enable tx_std_pcfifo_empty port	**Off**
Enable rx_std_pcfifo_full port	**Off**
Enable rx_std_pcfifo_empty port	**Off**
TX byte serializer mode	**Serialize x2**
RX byte deserializer mode	**Deserialize x2**
Enable TX 8B/10B encoder	**On**
Enable TX 8B/10B disparity control	**Off**
Enable RX 8B/10B decoder	**On**
RX rate match FIFO mode	**Disabled**
RX rate match insert / delete -ve pattern (hex)	0x00000000
RX rate match insert / delete +ve pattern (hex)	0x00000000
Enable rx_std_rmfifo_full port	**Off**
Enable rx_std_rmfifo_empty port	**Off**
Enable TX bit slip	**Off** (CPRI Auto configuration) **On** (CPRI Manual configuration)
Enable tx_std_bitslipboundarysel port	**Off** (CPRI Auto configuration) **On** (CPRI Manual configuration)
RX word aligner mode	**deterministic latency** (CPRI Auto configuration) **manual (FPGA fabric controlled)** (CPRI Manual configuration)
RX word aligner pattern length	**10**
RX word aligner pattern (hex)	0x000000000000017c
Number of word alignment patterns to achieve sync	**3** (26)
Number of invalid data words to lose sync	**3** (26)
Number of valid data words to decrement error count	**3** (26)
Enable fast sync status reporting for deterministic latency SM	**On / Off**
Enable rx_std_wa_patternalign port	**On / Off**
Enable rx_std_wa_a1a2size port	**Off**
Enable rx_std_bitslipboundarysel port	**Off** (CPRI Auto configuration) **On** (CPRI Manual configuration)
Enable rx_bitslip port	**Off** (CPRI Auto configuration) **On** (CPRI Manual configuration)
All options under **Bit Reversal and Polarity Inversion**	**Off**
All options under **PCIe Ports**	**Off**

(26) These are unused when the transceiver PHY is in CPRI mode.

Table 139. Dynamic Reconfiguration

Parameter	Value
Enable dynamic reconfiguration	**Off**
Share reconfiguration interface	**Off**
Enable Native PHY Debug Master Endpoint	**Off**
Enable embedded debug	**Off**
Enable capability registers	**Off**
Set user-defined IP identifier	**0**
Enable control and status registers	**Off**
Enable prbs soft accumulators	**Off**
Configuration file prefix	`altera_xcvr_native_c10`
Generate SystemVerilog package file	**Off**
Generate C header file	**Off**
Generate MIF (Memory Initialization File)	**Off**

Table 140. Generation Options

Parameter	Value
Generate parameter documentation file	**On**

2.9. Other Protocols

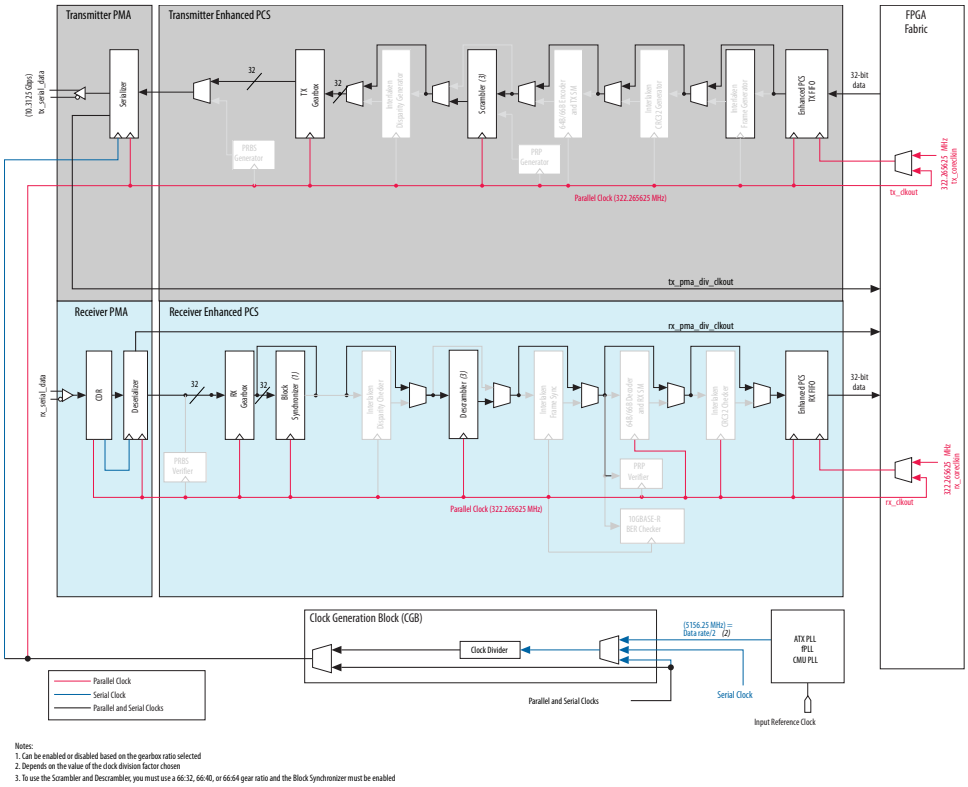
2.9.1. Using the "Basic (Enhanced PCS)" Configuration

You can use Cyclone 10 GX transceivers to configure the Enhanced PCS to support other 10G or 10G-like protocols. The Basic (Enhanced PCS) transceiver configuration rule allows access to the Enhanced PCS with full user control over the transceiver interfaces, parameters, and ports.

You can configure the transceivers for Basic functionality using the **Native PHY IP Basic (Enhanced PCS)** transceiver configuration rule.

Note: This configuration supports the FIFO in phase compensation and register modes. You can implement all other required logic for your specific application, such as standard or proprietary protocol multi-channel alignment in the FPGA fabric in soft IP.

Figure 76. Transceiver Channel Datapath and Clocking for Basic (Enhanced PCS) Configuration

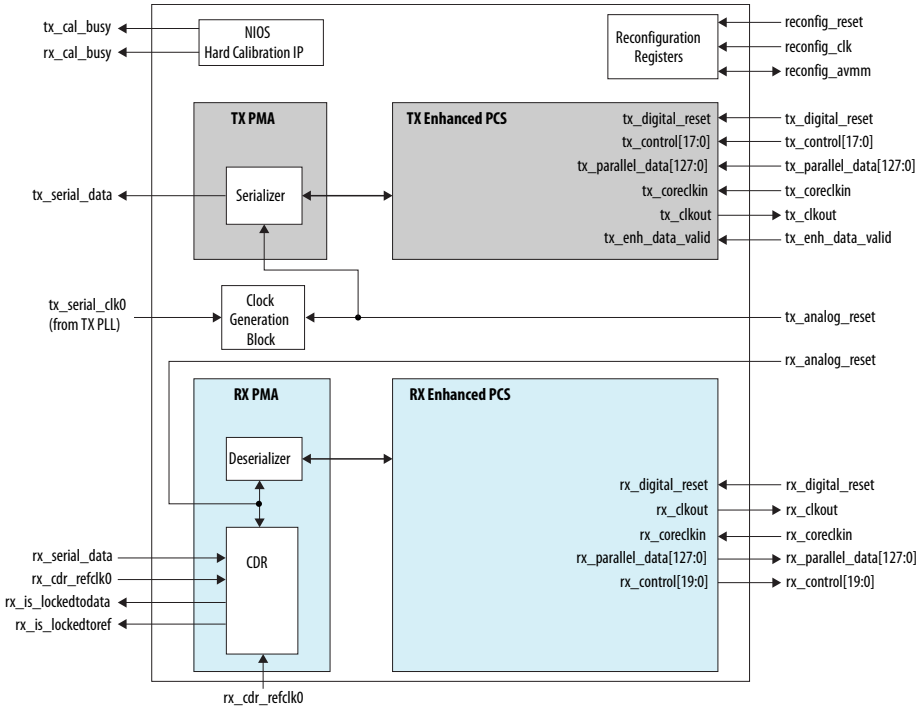


2.9.1.1. How to Implement the Basic (Enhanced PCS) Transceiver Configuration Rules in Cyclone 10 GX Transceivers

You should be familiar with the Basic (Enhanced PCS) and PMA architecture, PLL architecture, and the reset controller before implementing the Basic (Enhanced PCS) Transceiver Configuration Rule.

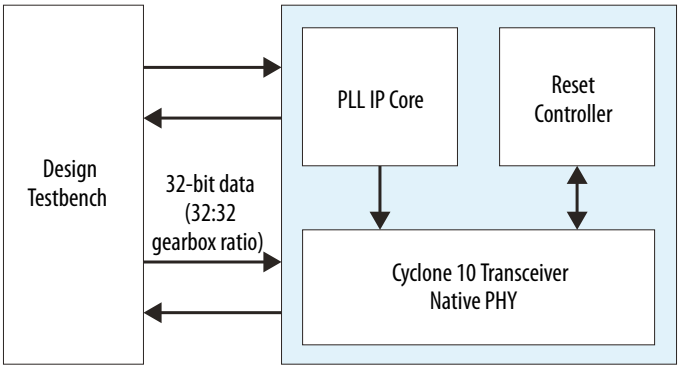
1. Open the IP Catalog and select the **Cyclone 10 GX Transceiver Native PHY IP**. Refer to Select and Instantiate the PHY IP Core on page 17 for more details.
2. Select **Basic (Enhanced PCS)** from the **Transceiver Configuration Rules** list located under **Datapath Options**.
3. Use the parameter values in the tables in Transceiver Native PHY IP Parameters for Basic (Enhanced PCS) Transceiver Configuration Rules as a starting point. Or, you can use the protocol presets described in Transceiver Native PHY Presets. You can then modify the settings to meet your specific requirements.
4. Click **Finish** to generate the Native PHY IP (this is your RTL file).

Figure 77. Signals and Ports of Native PHY IP for Basic (Enhanced PCS) Configurations



5. Configure and instantiate the PLL.
6. Create a transceiver reset controller. You can use your own reset controller or use the Transceiver PHY Reset Controller.
7. Connect the Native PHY IP core to the PLL IP core and the reset controller.

Figure 78. Connection Guidelines for a Basic (Enhanced PCS) Transceiver Design



8. Simulate your design to verify its functionality.

2.9.1.2. Native PHY IP Parameter Settings for Basic (Enhanced PCS)

This section contains the recommended parameter values for this protocol. Refer to *Using the Cyclone 10 GX Transceiver Native PHY IP Core* for the full range of parameter values.

Table 141. General and Datapath Parameters

The first two sections of the Parameter Editor for the Transceiver Native PHY provide a list of general and datapath options to customize the transceiver.

Parameter	Range
Message level for rule violations	**error**, **warning**
Transceiver configuration rules	**Basic (Enhanced PCS)**
PMA configuration rules	**Basic**, **GPON**
Transceiver mode	**TX / RX Duplex**, **TX Simplex**, **RX Simplex**
Number of data channels	**1** to **12**
Data rate	GX transceiver channel: **1 Gbps** to **12.5 Gbps**
Enable datapath and interface reconfiguration	**On** / **Off**
Enable simplified data interface	**On** / **Off**

Table 142. TX PMA Parameters

Parameter	Range
TX channel bonding mode	**Not bonded**, **PMA only bonding**, **PMA and PCS bonding**
PCS TX channel bonding master	**Auto**, **0** to **n-1**, **n** (where n = the number of data channels)
Actual PCS TX channel bonding master	**n-1** (where n = the number of data channels)
TX local clock division factor	**1**, **2**, **4**, **8**
Number of TX PLL clock inputs per channel	**1**, **2**, **3**, **4**
Initial TX PLL clock input selection	**0**
Enable tx_pma_clkout port	**On** / **Off**
Enable tx_pma_div_clkout port	**On** / **Off**
tx_pma_div_clkout division factor	**Disabled**, **1**, **2**, **33**, **40**, **66**
Enable tx_pma_elecidle port	**On** / **Off**
Enable rx_serialpbken port	**On** / **Off**

Table 143. RX PMA Parameters

Parameter	Range
Number of CDR reference clocks	**1** to **5**
Selected CDR reference clock	**0** to **4**
Selected CDR reference clock frequency	For Basic (Enhanced PCS): Depends on the data rate parameter
PPM detector threshold	**100**, **300**, **500**, **1000**
CTLE adaptation mode	**manual**
Enable rx_pma_clkout port	**On** / **Off**
Enable rx_pma_div_clkout port	**On** / **Off**
rx_pma_div_clkout division factor	**Disabled**, **1**, **2**, **33**, **40**, **66**
Enable rx_pma_clkslip port	**On** / **Off**

continued...

Parameter	Range
Enable rx_is_lockedtodata port	On / Off
Enable rx_is_lockedtoref port	On / Off
Enable rx_set_locktodata and rx_set_locktoref ports	On / Off
Enable rx_serialpbken port	On / Off
Enable PRBS verifier control and status ports	On / Off

Table 144. Enhanced PCS Parameters

Parameter	Range
Enhanced PCS/PMA interface width	32, 40, 64
FPGA fabric/Enhanced PCS interface width	32, 40, 50, 64, 66, 67
Enable Enhanced PCS low latency mode	On / Off
Enable RX/TX FIFO double width mode	On / Off
TX FIFO mode	Phase compensation, Register, Interlaken, Basic, Fast register *Note:* Only Basic Enhanced is valid.
TX FIFO partially full threshold	10, 11, 12, 13, 14, 15
TX FIFO partially empty threshold	1, 2, 3, 4, 5
Enable tx_enh_fifo_full port	On / Off
Enable tx_enh_fifo_pfull port	On / Off
Enable tx_enh_fifo_empty port	On / Off
Enable tx_enh_fifo_pempty port	On / Off
RX FIFO mode	Phase Compensation, Register, Basic
RX FIFO partially full threshold	0 to 31
RX FIFO partially empty threshold	0 to 31
Enable RX FIFO alignment word deletion (Interlaken)	On / Off
Enable RX FIFO control word deletion (Interlaken)	On / Off
Enable rx_enh_data_valid port	On / Off
Enable rx_enh_fifo_full port	On / Off
Enable rx_enh_fifo_pfull port	On / Off
Enable rx_enh_fifo_empty port	On / Off
Enable rx_enh_fifo_pempty port	On / Off
Enable rx_enh_fifo_del port (10GBASE-R)	On / Off
Enable rx_enh_fifo_insert port (10GBASE-R)	On / Off
Enable rx_enh_fifo_rd_en port	On / Off
Enable rx_enh_fifo_align_val port (Interlaken)	On / Off
Enable rx_enh_fifo_align_cir port (Interlaken)	On / Off
Enable TX 64b/66b encoder	On / Off

continued...

Parameter	Range
Enable RX 64b/66b decoder	On / Off
Enable TX sync header error insertion	On / Off
Enable RX block synchronizer	On / Off
Enable rx_enh_blk_lock port	On / Off
Enable TX data bitslip	On / Off
Enable TX data polarity inversion	On / Off
Enable RX data bitslip	On / Off
Enable RX data polarity inversion	On / Off
Enable tx_enh_bitslip port	On / Off
Enable rx_bitslip port	On / Off
Enable tx_enh_frame port	On / Off
Enable rx_enh_frame port	On / Off
Enable rx_enh_frame_dian_status port	On / Off

Table 145. Dynamic Reconfiguration Parameters

Parameter	Range
Enable dynamic reconfiguration	On / Off
Share reconfiguration interface	On / Off
Enable Native PHY Debug Master Endpoint	On / Off
Enable embedded debug	On / Off
Enable capability registers	On / Off
Set user-defined IP identifier	number
Enable control and status registers	On / Off
Enable prbs soft accumulators	On / Off
Configuration file prefix	text string
Generate SystemVerilog package file	On / Off
Generate C header file	On / Off

Table 146. Generate Options Parameters

Parameter	Range
Generate parameter documentation file	On / Off

Related Information

Using the Cyclone 10 GX Transceiver Native PHY IP Core on page 26

2.9.1.3. How to Enable Low Latency in Basic Enhanced PCS

In the Parameter Editor, use the following settings to enable low latency:

1. Select the **Enable 'Enhanced PCS' low latency mode** option.
2. Select one of the following gear ratios:

   Single-width mode: **32:32**, **40:40**, **64:64**, **66:40**, **66:64**, or **64:32**

   Double-width mode: **40:40**, **64:64**, or **66:64**
3. Select **Phase_compensation** in the TX and RX FIFO mode list.
4. If you need the Scrambler and Descrambler features, enable **Block Synchronize** and use the **66:32**, **66:40**, or **66:64** gear ratio.

2.9.1.4. Enhanced PCS FIFO Operation

Phase Compensation Mode

Phase compensation mode ensures correct data transfer between the core clock and parallel clock domains. The read and write sides of the TX Core or RX Core FIFO must be driven by the same clock frequency. The depth of the TX or RX FIFO is constant in this mode. Therefore, the TX Core or RX Core FIFO flag status can be ignored. You can tie `tx_fifo_wr_en` or `rx_data_valid` to 1.

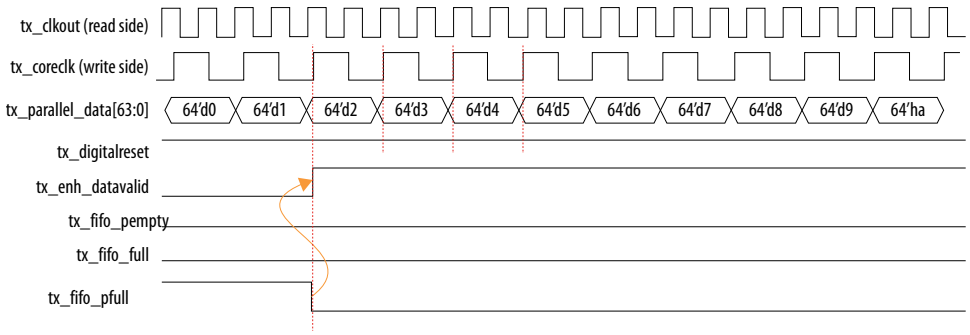
Basic Mode

Basic mode allows you to drive the write and read side of a FIFO with different clock frequencies. `tx_coreclkin` or `rx_coreclkin` must have a minimum frequency of the lane data rate divided by 66. The frequency range for `tx_coreclkin` or `rx_coreclkin` is (data rate/32) to (data rate/66). For best results, Intel recommends that `tx_coreclkin` or `rx_coreclkin` be set to (data rate/32). Monitor the FIFO flag to control write and read operations.

For TX FIFO, assert `tx_enh_data_valid` with the `tx_fifo_pfull` signal going low. This can be done with the following example assignment:

```
assign tx_enh_data_valid = ~tx_fifo_pfull;
```

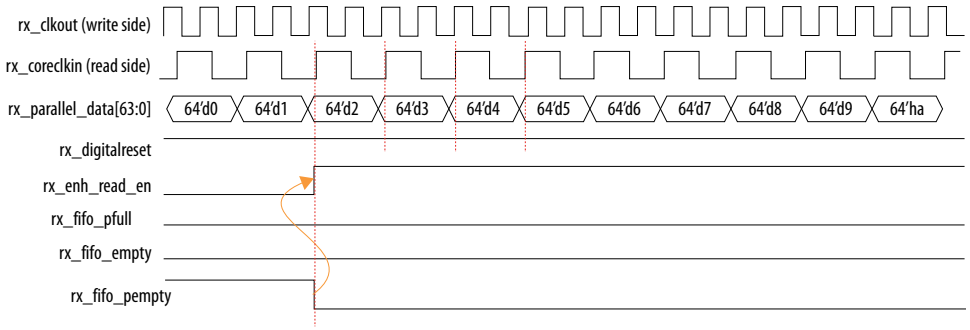
Figure 79. TX FIFO Basic Mode Operation



For RX FIFO, assert `rx_enh_read_en` with the `rx_fifo_pempty` signal going low. This can be done with the following example assignment:

```
assign rx_enh_read_en = ~rx_fifo_pempty;
```

Figure 80. RX FIFO Basic Mode Operation



If you are using even gear ratios, the `rx_enh_data_valid` signal is always high. For uneven gear ratios, `rx_enh_data_valid` toggles. RX parallel data is valid when `rx_enh_data_valid` is high. Discard invalid RX parallel data when the `rx_enh_datavalid` signal is low.

Register and Fast Register Mode

This FIFO mode is used for protocols, which require deterministic latency. You can tie `tx_fifo_wr_en` to 1.

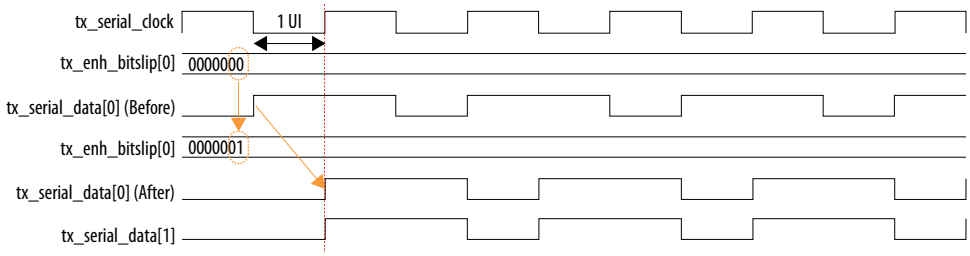
2.9.1.5. TX Data Bitslip

The bit slip feature in the TX gearbox allows you to slip the transmitter bits before they are sent to the serializer.

The value specified on the TX bit slip bus indicates the number of bit slips. The minimum slip is one UI. The maximum number of bits slipped is equal to the FPGA fabric-to-transceiver interface width minus 1. For example, if the FPGA fabric-to-transceiver interface width is 64 bits, the bit slip logic can slip a maximum of 63 bits. Each channel has 6 bits to determine the number of bits to slip. The TX bit slip bus is a level-sensitive port, so the TX serial data is bit slipped statically by TX bit slip port assignments. Each TX channel has its own TX bit slip assignment and the bit slip amount is relative to the other TX channels. You can improve lane-to-lane skew by assigning TX bit slip ports with proper values.

The following figure shows the effect of slipping `tx_serial_data[0]` by one UI to reduce the skew with `tx_serial_data[1]`. After the bit slip, `tx_serial_data[0]` and `tx_serial_data[1]` are aligned.

Figure 81. TX Bit Slip



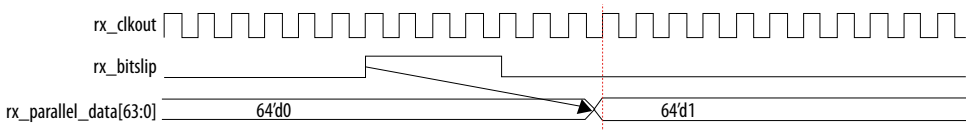
2.9.1.6. TX Data Polarity Inversion

Use the TX data polarity inversion feature to swap the positive and negative signals of a serial differential link if they were erroneously swapped during board layout. To enable TX data polarity inversion, select the **Enable TX data polarity inversion** option in the **Gearbox** section of Platform Designer. It can also be dynamically controlled with dynamic reconfiguration. The Enhanced PCS only supports the static polarity inversion feature.

2.9.1.7. RX Data Bitslip

The RX data bit slip in the RX gearbox allows you to slip the recovered data. An asynchronous active high edge on the `rx_bitslip` port changes the word boundary, shifting `rx_parallel_data` one bit at a time. Use the `rx_bitslip` port with its own word aligning logic. Assert the `rx_bitslip` signal for at least two parallel clock cycles to allow synchronization. You can verify the word alignment by monitoring `rx_parallel_data`. Using the RX data bit slip feature is optional.

Figure 82. RX Bit Slip



2.9.1.8. RX Data Polarity Inversion

Use the RX data polarity inversion feature to swap the positive and negative signals of a serial differential link if they were erroneously swapped during board layout. To enable RX data polarity inversion, select the **Enable RX data polarity inversion** option in the Gearbox section of Platform Designer. It can also be dynamically controlled with dynamic reconfiguration. The Enhanced PCS only supports the static polarity inversion feature.

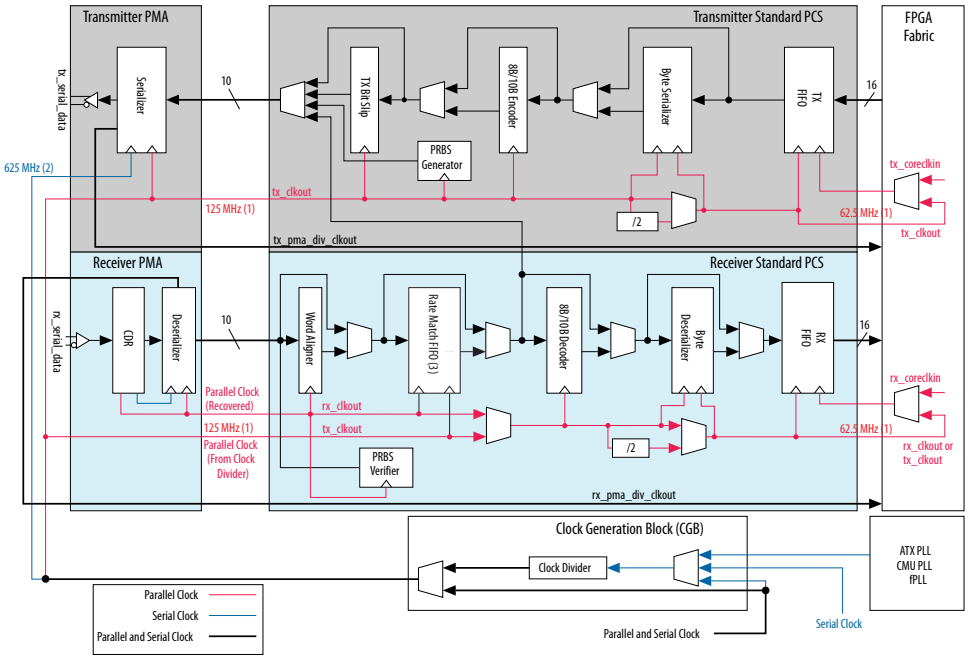
2.9.2. Using the Basic/Custom, Basic/Custom with Rate Match Configurations of Standard PCS

Use one of the following transceiver configuration rules to implement protocols such as SONET/SDH, SDI/HD, SATA, or your own custom protocol:

- Basic protocol
- Basic protocol with low latency enabled
- Basic with rate match protocol

Figure 83. Transceiver Channel Datapath and Clocking for the Basic and Basic with Rate Match Configurations

The clocking calculations in this figure are for an example when the data rate is 1250 Mbps and the PMA width is 10 bits.

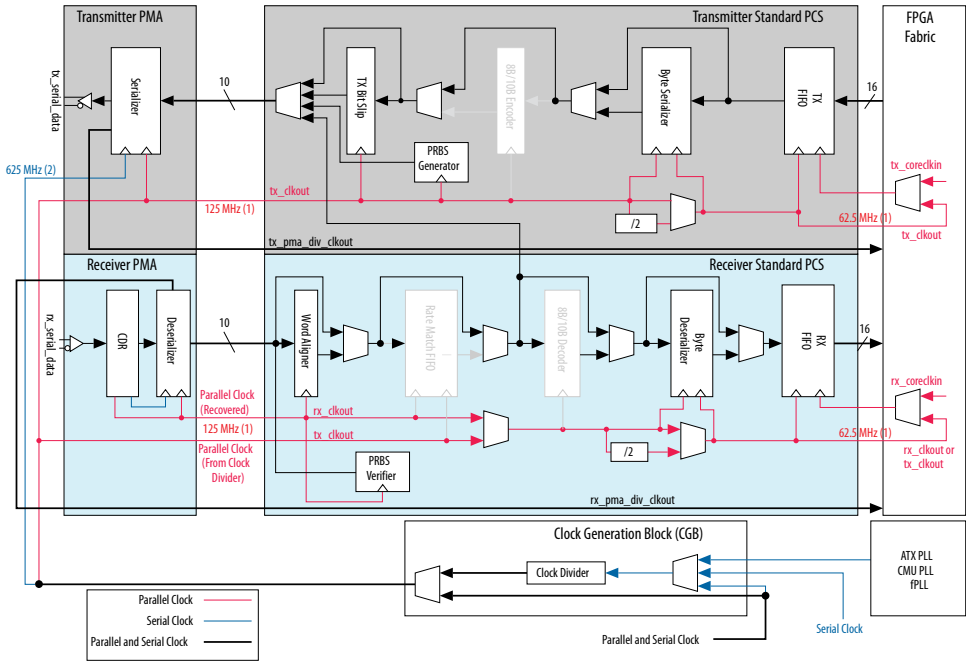


Notes:
1. The parallel clock (tx_clkout or rx_clkout) is calculated as data rate/PCS-PMA interface width = 1250/10 = 125 MHz. When the Byte Serializer is set to Serialize x2 mode, tx_clkout and rx_clkout become 1250/20 = 62.5 MHz.
2. The serial clock is calculated as data rate/2. The PMA runs on a dual data rate clock.
3. This block is only enabled when using the Basic with Rate Match transceiver configuration rule.

In low latency modes, the transmitter and receiver FIFOs are always enabled. Depending on the targeted data rate, you can optionally bypass the byte serializer and deserializer blocks.

Figure 84. Transceiver Channel Datapath and Clocking for Basic Configuration with Low Latency Enabled

The clocking calculations in this figure are for an example when the data rate is 1250 Mbps and the PMA width is 10 bits.



Notes:
1. The parallel clock (tx_clkout or rx_clkout) is calculated as data rate/PCS-PMA interface width = 1250/10 = 125 MHz. When the Byte Serializer is set to Serialize x2 mode, tx_clkout and rx_clkout become 1250/20 = 62.5 MHz.
2. The serial clock is calculated as data rate/2. The PMA runs on a dual data rate clock.

In low latency modes, the transmitter and receiver FIFOs are always enabled. Depending on the targeted data rate, you can optionally bypass the byte serializer and deserializer blocks.

Related Information

Cyclone 10 GX Standard PCS Architecture on page 299

2.9.2.1. Word Aligner Manual Mode

To use this mode:

1. Set the **RX word aligner mode** to **Manual (FPGA Fabric controlled)**.
2. Set the **RX word aligner pattern length** option according to the PCS-PMA interface width.
3. Enter a hexadecimal value in the **RX word aligner pattern (hex)** field.

This mode adds `rx_patterndetect` and `rx_syncstatus`. You can select the **Enable rx_std_wa_patternalign port** option to enable `rx_std_wa_patternalign`. An active high on `rx_std_wa_patternalign` re-aligns the word aligner one time.

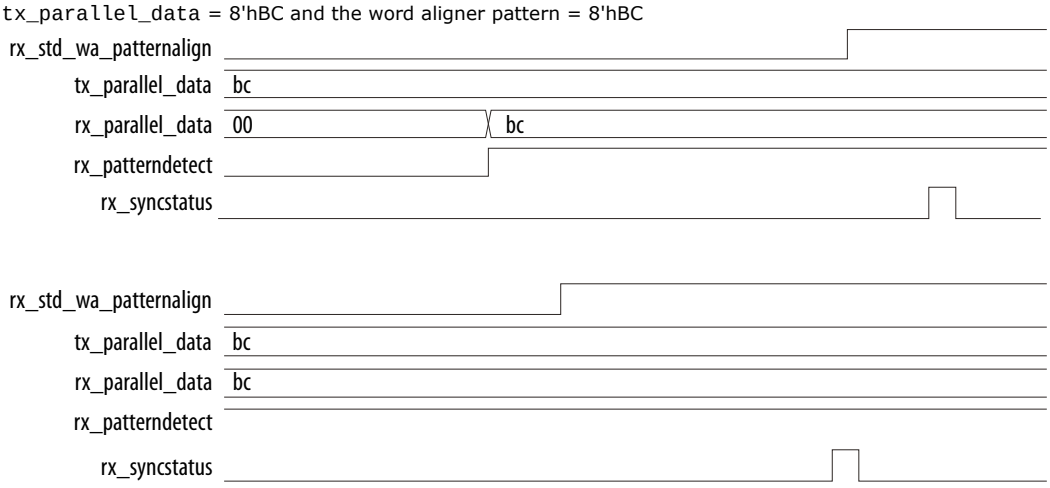
Note:

- `rx_patterndetect` is asserted whenever there is a pattern match.
- `rx_syncstatus` is asserted after the word aligner achieves synchronization.
- `rx_std_wa_patternalign` is asserted to re-align and resynchronize.
- If there is more than one channel in the design, `rx_patterndetect`, `rx_syncstatus` and `rx_std_wa_patternalign` become buses in which each bit corresponds to one channel.

You can verify this feature by monitoring `rx_parallel_data`.

The following timing diagrams demonstrate how to use the ports and show the relationship between the various control and status signals. In the top waveform, `rx_parallel_data` is initially misaligned. After asserting the `rx_std_wa_patternalign` signal, it becomes aligned. The bottom waveform shows the behavior of the `rx_syncstatus` signal when `rx_parallel_data` is already aligned.

Figure 85. Manual Mode when the PCS-PMA Interface Width is 8 Bits



In manual alignment mode, the word alignment operation is manually controlled with the `rx_std_wa_patternalign` input signal or the `rx_enapatternalign` register. The word aligner operation is level-sensitive to `rx_enapatternalign`. The word aligner asserts the `rx_syncstatus` signal for one parallel clock cycle whenever it re-aligns to the new word boundary.

Figure 86. Manual Mode when the PCS-PMA Interface Width is 10 Bits

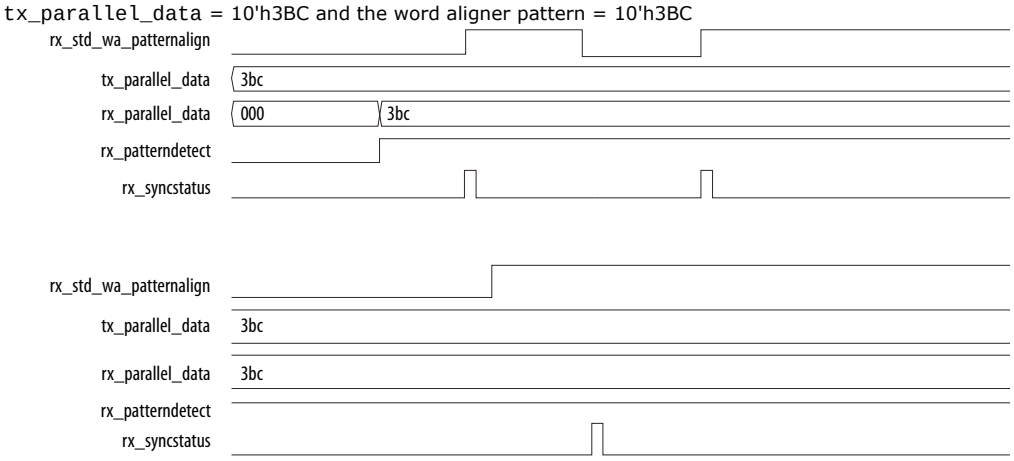


Figure 87. Manual Mode when the PCS-PMA Interface Width is 16 Bits

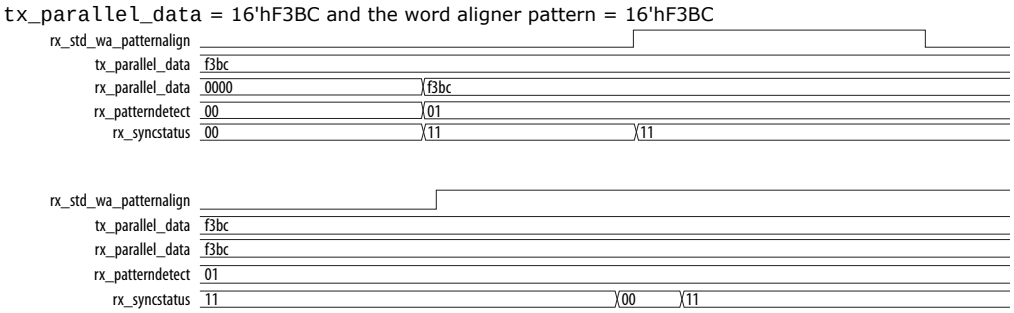
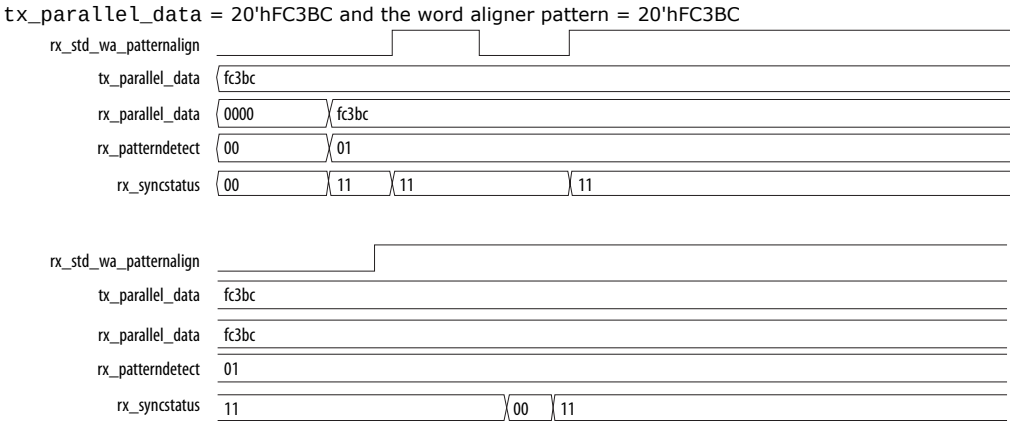


Figure 88. Manual Mode when the PCS-PMA Interface Width is 20 Bits



2.9.2.2. Word Aligner Synchronous State Machine Mode

To use this mode:

- Select the **Enable TX 8B/10B encoder** option.
- Select the **Enable RX 8B/10B decoder** option.

The 8B/10B encoder and decoder add the following additional ports:

- `tx_datak`
- `rx_datak`
- `rx_errdetect`
- `rx_disperr`
- `rx_runningdisp`

1. Set the **RX word aligner mode** to **synchronous state machine**.
2. Set the **RX word aligner pattern length** option according to the PCS-PMA interface width.
3. Enter a hexadecimal value in the **RX word aligner pattern (hex)** field.

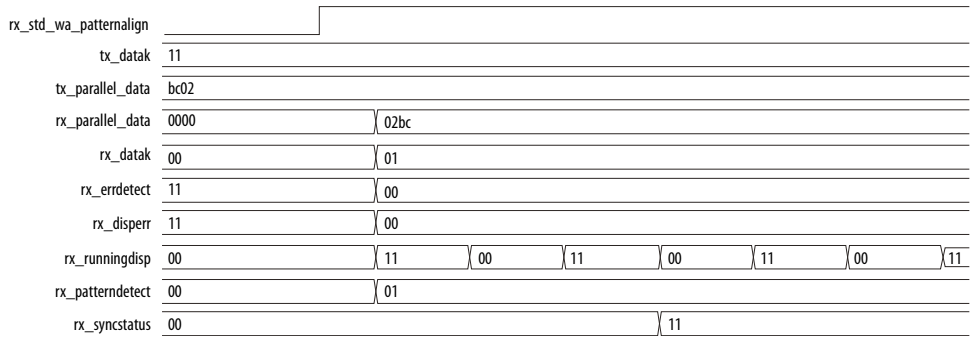
The RX word aligner pattern is the 8B/10B encoded version of the data pattern. You can also specify the number of word alignment patterns to achieve synchronization, the number of invalid data words to lose synchronization, and the number of valid data words to decrement error count. This mode adds two additional ports: `rx_patterndetect` and `rx_syncstatus`.

Note:

- `rx_patterndetect` is asserted whenever there is a pattern match.
- `rx_syncstatus` is asserted after the word aligner achieves synchronization.
- `rx_std_wa_patternalign` is asserted to re-align and re-synchronize.
- If there is more than one channel in the design, `tx_datak`, `rx_datak`, `rx_errdetect`, `rx_disperr`, `rx_runningdisp`, `rx_patterndetect`, and `rx_syncstatus` become buses in which each bit corresponds to one channel.

You can verify this feature by monitoring `rx_parallel_data`.

Figure 89. Synchronization State Machine Mode when the PCS-PMA Interface Width is 20 Bits



2.9.2.3. RX Bit Slip

To use the RX bit slip, select **Enable rx_bitslip port** and set the word aligner mode to **bit slip**. This adds `rx_bitslip` as an input control port. An active high edge on `rx_bitslip` slips one bit at a time. When `rx_bitslip` is toggled, the word aligner slips one bit at a time on every active high edge. Assert the `rx_bitslip` signal for at least two parallel clock cycles to allow synchronization. You can verify this feature by monitoring `rx_parallel_data`.

The RX bit slip feature is optional and may or may not be enabled.

Figure 90. RX Bit Slip in 8-bit Mode

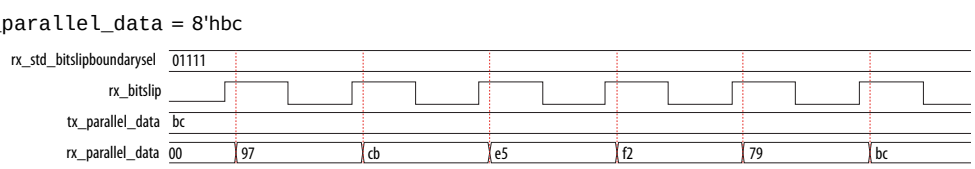


Figure 91. RX Bit Slip in 10-bit Mode

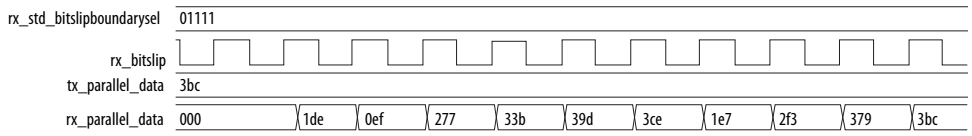


Figure 92. RX Bit Slip in 16-bit Mode

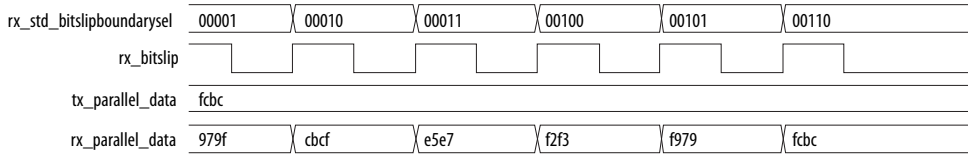
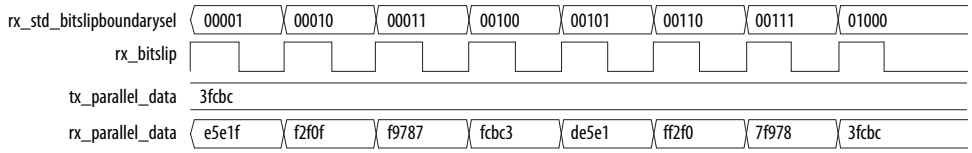


Figure 93. RX Bit Slip in 20-bit Mode



2.9.2.4. RX Polarity Inversion

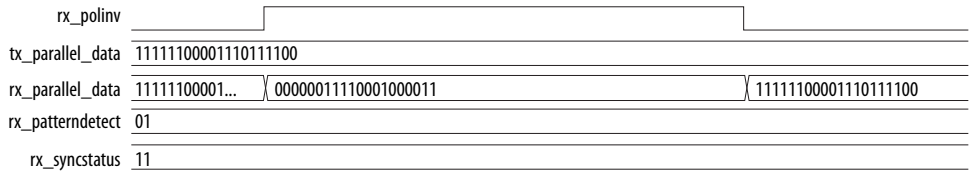
Receiver polarity inversion can be enabled in low latency, basic, and basic rate match modes. The Standard PCS supports both the static and dynamic polarity inversion features.

To enable the RX polarity inversion feature, select the **Enable RX polarity inversion** and **Enable rx_polinv port** options.

This mode adds `rx_polinv`. If there is more than one channel in the design, `rx_polinv` is a bus in which each bit corresponds to a channel. As long as `rx_polinv` is asserted, the RX data received has a reverse polarity.

You can verify this feature by monitoring `rx_parallel_data`.

Figure 94. RX Polarity Inversion



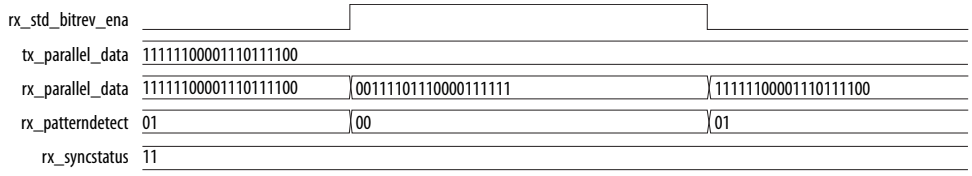
2.9.2.5. RX Bit Reversal

The RX bit reversal feature can be enabled in low latency, basic, and basic rate match mode. The word aligner is available in any mode, bit slip, manual, or synchronous state machine.

To enable this feature, select the **Enable RX bit reversal** and **Enable rx_std_bitrev_ena port** options. This adds `rx_std_bitrev_ena`. If there is more than one channel in the design, `rx_std_bitrev_ena` becomes a bus in which each bit corresponds to a channel. As long as `rx_std_bitrev_ena` is asserted, the RX data received by the core shows bit reversal.

You can verify this feature by monitoring `rx_parallel_data`.

Figure 95. RX Bit Reversal



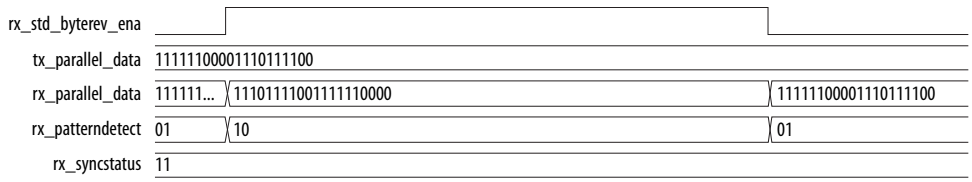
2.9.2.6. RX Byte Reversal

The RX byte reversal feature can be enabled in low latency, basic, and basic rate match mode. The word aligner is available in any mode.

To enable this feature, select the **Enable RX byte reversal** and **Enable rx_std_byterev_ena port** options. This adds `rx_std_byterev_ena`. If there is more than one channel in the design, `rx_std_byterev_ena` becomes a bus in which each bit corresponds to a channel. As long as `rx_std_byterev_ena` is asserted, the RX data received by the core shows byte reversal.

You can verify this feature by monitoring `rx_parallel_data`.

Figure 96. RX Byte Reversal



2.9.2.7. Rate Match FIFO in Basic (Single Width) Mode

Only the rate match FIFO operation is covered in these steps.

1. Select **basic (single width)** in the **RX rate match FIFO mode** list.
2. Enter values for the following parameters.

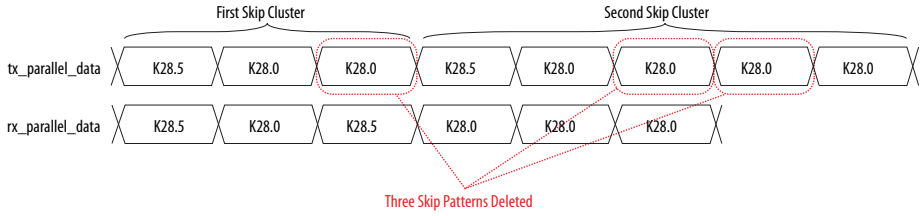
Parameter	Value	Description
RX rate match insert/delete +ve pattern (hex)	20 bits of data specified as a hexadecimal string	The first 10 bits correspond to the skip pattern and the last 10 bits correspond to the control pattern. The skip pattern must have neutral disparity.
RX rate match insert/delete –ve pattern (hex)	20 bits of data specified as a hexadecimal string	The first 10 bits correspond to the skip pattern and the last 10 bits correspond to the control pattern. The skip pattern must have neutral disparity.

ve (volt encodes) are NRZ_L conditions where +ve encodes 0 and –ve encodes 1. ve is a running disparity (+/–RD) specifically used with the rate matcher. Depending on the ppm difference (which is defined by protocol) between the recovered clock and the local clock, the rate matcher adds or deletes a maximum of four skip patterns (neutral disparity). The net neutrality is conserved even after the skip word insertion or deletion because the control words alternate between positive and negative disparity.

In the following figure, the first skip cluster has a /K28.5/ control pattern followed by two /K28.0/ skip patterns. The second skip cluster has a /K28.5/ control pattern followed by four /K28.0/ skip patterns. The rate match FIFO deletes only one /K28.0/ skip pattern from the first skip cluster to maintain at least one skip pattern in the cluster after deletion. Two /K28.0/ skip patterns are deleted from the second cluster for a total of three skip patterns deletion requirement.

The rate match FIFO can insert a maximum of four skip patterns in a cluster, if there are no more than five skip patterns in the cluster after insertion.

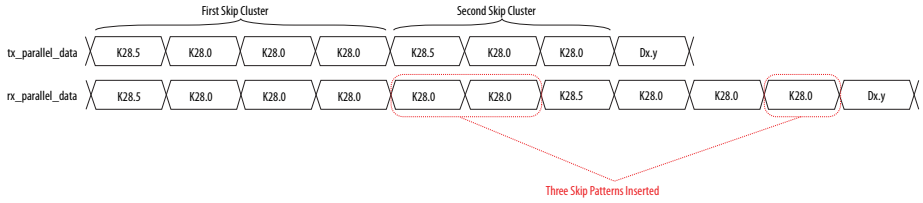
Figure 97. Rate Match FIFO Deletion with Three Skip Patterns Required for Deletion



Note: /K28.5/ is the control pattern and /K28.0/ is the skip pattern

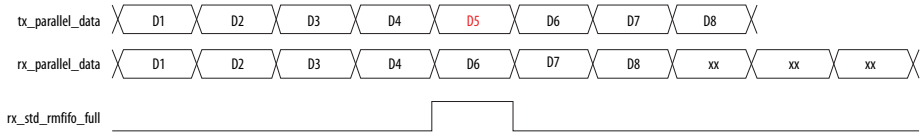
In the following figure, /K28.5/ is the control pattern and neutral disparity /K28.0/ is the skip pattern. The first skip cluster has a /K28.5/ control pattern followed by three /K28.0/ skip patterns. The second skip cluster has a /K28.5/ control pattern followed by two /K28.0/ skip patterns. The rate match FIFO inserts only two /K28.0/ skip patterns into the first skip cluster to maintain a maximum of five skip patterns in the cluster after insertion. One /K28.0/ skip pattern is inserted into the second cluster for a total of three skip patterns to meet the insertion requirement.

Figure 98. Rate Match FIFO Insertion with Three Skip Patterns Required for Insertion



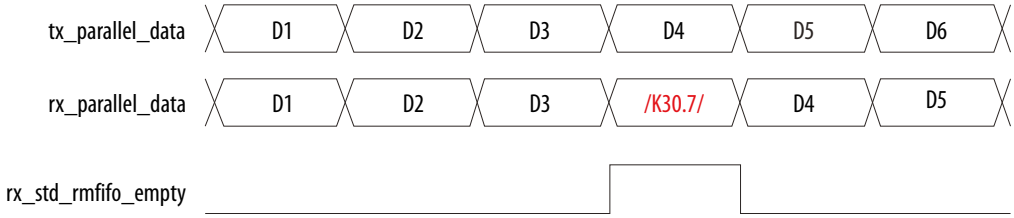
The following figure shows the deletion of **D5** when the upstream transmitter reference clock frequency is greater than the local receiver reference clock frequency. It asserts `rx_std_rmfifo_full` for one parallel clock cycle while the deletion takes place.

Figure 99. Rate Match FIFO Becoming Full After Receiving D5



The following figure shows the insertion of skip symbols when the local receiver reference clock frequency is greater than the upstream transmitter reference clock frequency. It asserts `rx_std_rmfifo_empty` for one parallel clock cycle while the insertion takes place.

Figure 100. Rate Match FIFO Becoming Empty After Receiving D3



2.9.2.8. Rate Match FIFO Basic (Double Width) Mode

1. Select **basic (double width)** in the **RX rate match FIFO mode** list.
2. Enter values for the following parameters.

Parameter	Value	Description
RX rate match insert/delete +ve pattern (hex)	20 bits of data specified as a hexadecimal string	The first 10 bits correspond to the skip pattern and the last 10 bits correspond to the control pattern. The skip pattern must have neutral disparity.
RX rate match insert/delete -ve pattern (hex)	20 bits of data specified as a hexadecimal string	The first 10 bits correspond to the skip pattern and the last 10 bits correspond to the control pattern. The skip pattern must have neutral disparity.

The rate match FIFO can delete as many pairs of skip patterns from a cluster as necessary to avoid the rate match FIFO from overflowing. The rate match FIFO can delete a pair of skip patterns only if the two 10-bit skip patterns appear in the same clock cycle on the LSByte and MSByte of the 20-bit word. If the two skip patterns appear straddled on the MSByte of a clock cycle and the LSByte of the next clock cycle, the rate match FIFO cannot delete the pair of skip patterns.

In the following figure, the first skip cluster has a /K28.5/ control pattern in the LSByte and /K28.0/ skip pattern in the MSByte of a clock cycle followed by one /K28.0/ skip pattern in the LSByte of the next clock cycle. The rate match FIFO cannot delete the two skip patterns in this skip cluster because they do not appear in the same clock cycle. The second skip cluster has a /K28.5/ control pattern in the MSByte of a clock cycle followed by two pairs of /K28.0/ skip patterns in the next two cycles. The rate match FIFO deletes both pairs of /K28.0/ skip patterns (for a total of four skip patterns deleted) from the second skip cluster to meet the three skip pattern deletion requirement.

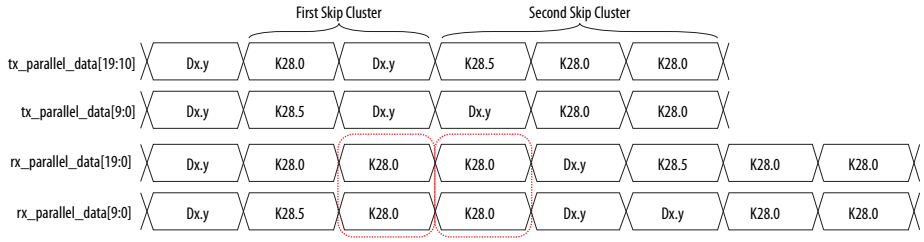
The rate match FIFO can insert as many pairs of skip patterns into a cluster necessary to avoid the rate match FIFO from under running. The 10-bit skip pattern can appear on the MSByte, the LSByte, or both, of the 20-bit word.

Figure 101. Rate Match FIFO Deletion with Four Skip Patterns Required for Deletion

/K28.5/ is the control pattern and neutral disparity /K28.0/ is the skip pattern.

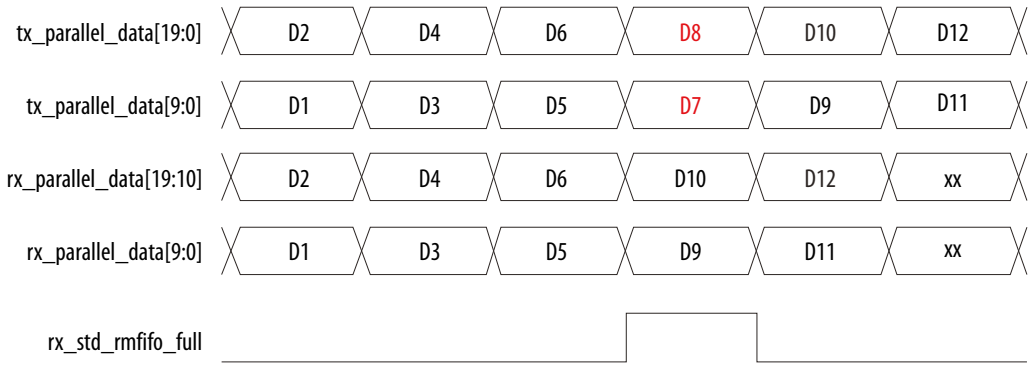
In the following figure, /K28.5/ is the control pattern and neutral disparity /K28.0/ is the skip pattern. The first skip cluster has a /K28.5/ control pattern in the LSByte and /K28.0/ skip pattern in the MSByte of a clock cycle. The rate match FIFO inserts pairs of skip patterns in this skip cluster to meet the three skip pattern insertion requirement.

Figure 102. Rate Match FIFO Insertion with Four Skip Patterns Required for Insertion



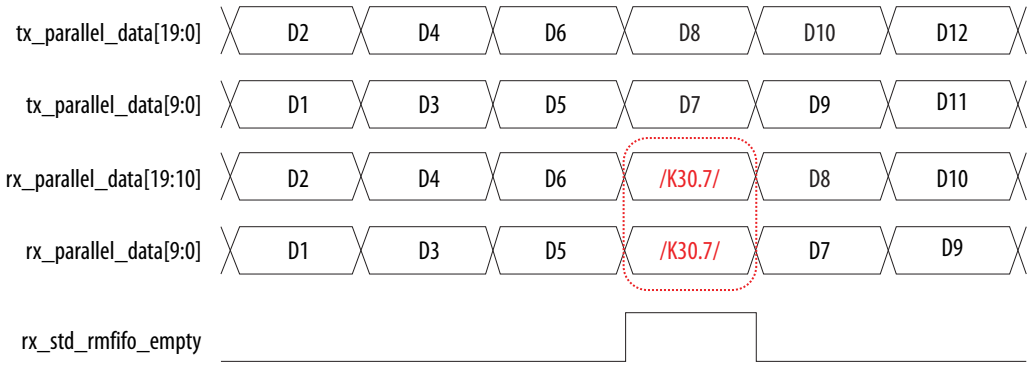
The following figure shows the deletion of the 20-bit word D7D8.

Figure 103. Rate Match FIFO Becoming Full After Receiving the 20-Bit Word D5D6



The following figure shows the insertion of two skip symbols.

Figure 104. Rate Match FIFO Becoming Empty After Reading out the 20-Bit Word D5D6



2.9.2.9. 8B/10B Encoder and Decoder

To enable the 8B/10B Encoder and the 8B/10B Decoder, select the **Enable TX 8B/10B Encoder** and **Enable RX 8B/10B Decoder** options on the **Standard PCS** tab in the IP Editor. Platform Designer allows implementing the 8B/10B decoder in **RX-only** mode.

The following ports are added:

- `tx_datak`
- `rx_datak`
- `rx_runningdisp`
- `rx_disperr`
- `rx_errdetect`

`rx_datak` and `tx_datak` indicate whether the parallel data is a control word or a data word. The incoming 8-bit data (`tx_parallel_data`) and the control identifier (`tx_datak`) are converted into a 10-bit data. After a power on reset, the 8B/10B encoder takes the 10-bit data from the RD- column. Next, the encoder chooses the 10-bit data from the RD+ column to maintain neutral disparity. The running disparity is shown by `rx_runningdisp`.

2.9.2.10. 8B/10B TX Disparity Control

The Disparity Control feature controls the running disparity of the output from the 8B/10B Decoder.

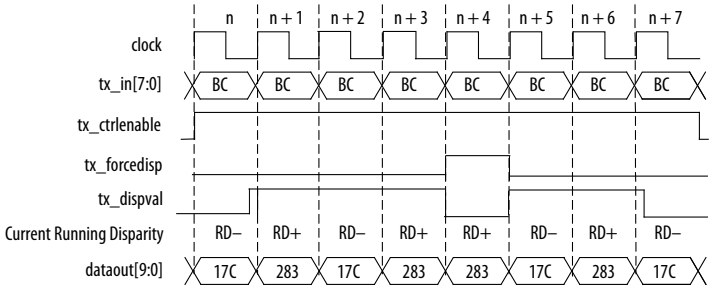
To enable TX Disparity Control, select the **Enable TX 8B/10B Disparity Control** option. The following ports are added:

- `tx_forcedisp`—a control signal that indicates whether a disparity value has to be forced or not
- `tx_dispval`—a signal that indicates the value of the running disparity that is being forced

When the number of data channels is more than 1, `tx_forcedisp` and `tx_dispval` are shown as buses in which each bit corresponds to one channel.

The following figure shows the current running disparity being altered in Basic single-width mode by forcing a positive disparity /K28.5/ when it was supposed to be a negative disparity /K28.5/. In this example, a series of /K28.5/ code groups are continuously being sent. The stream alternates between a positive running disparity (RD+) /K28.5/ and a negative running disparity (RD-) /K28.5/ to maintain a neutral overall disparity. The current running disparity at time n + 3 indicates that the /K28.5/ in time n + 4 should be encoded with a negative disparity. Because `tx_forcedisp` is high at time n + 4, and `tx_dispval` is low, the /K28.5/ at time n + 4 is encoded as a positive disparity code group.

Figure 105. 8B/10B TX Disparity Control



2.9.2.11. How to Enable Low Latency in Basic

In the Cyclone 10 GX Transceiver Native PHY IP Parameter Editor, use the following settings to enable low latency:

1. Select the **Enable 'Standard PCS' low latency mode** option.
2. Select either **low_latency** or **register FIFO** in the **TX FIFO mode** list.
3. Select either **low_latency** or **register FIFO** in the **RX FIFO mode** list.
4. Select either **Disabled** or **Serialize x2** in the **TX byte serializer mode** list.
5. Select either **Disabled** or **Serialize x2** in the **RX byte deserializer mode** list.
6. Ensure that **RX rate match FIFO mode** is **disabled**.
7. Set the **RX word aligner mode** to **bitslip**.
8. Set the **RX word aligner pattern length** to **7** or **16**.

   Note: TX bitslip, RX bitslip, bit reversal, and polarity inversion modes are supported.

2.9.2.12. TX Bit Slip

To use the TX bit slip, select the **Enable TX bitslip** and **Enable tx_std_bitslipboundarysel port** options. This adds the `tx_std_bitslipboundarysel` input port. The TX PCS automatically slips the number of bits specified by `tx_std_bitslipboundarysel`. There is no port for TX bit slip. If there is more than one channel in the design, `tx_std_bitslipboundarysel` ports are multiplied by the number of channels. You can verify this feature by monitoring the `tx_parallel_data` port.

Enabling the TX bit slip feature is optional.

Note: The `rx_parallel_data` values in the following figures are based on the TX and RX bit reversal features being disabled.

Figure 106. TX Bit Slip in 8-bit Mode

`tx_parallel_data` = 8'hbc. `tx_std_bitslipboundarysel` = 5'b00001 (bit slip by 1 bit).

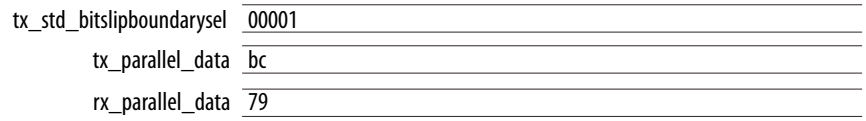


Figure 107. TX Bit Slip in 10-bit Mode

`tx_parallel_data` = 10'h3bc. `tx_std_bitslipboundarysel` = 5'b00011 (bit slip by 3 bits).

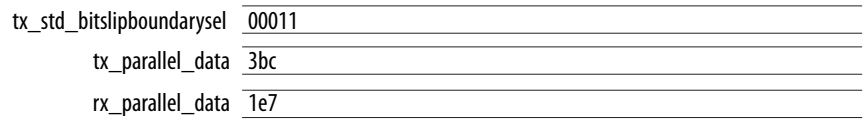


Figure 108. TX Bit Slip in 16-bit Mode

`tx_parallel_data` = 16'hfcbc. `tx_std_bitslipboundarysel` =5'b00011 (bit slip by 3 bits).

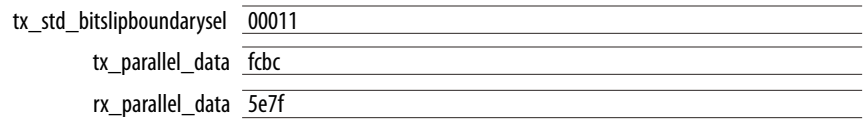
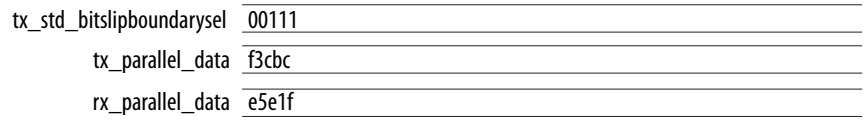


Figure 109. TX Bit Slip in 20-bit Mode

`tx_parallel_data` = 20'hF3CBC. `tx_std_bitslipboundarysel` = 5'b00111 (bit slip by 7 bits).



2.9.2.13. TX Polarity Inversion

The positive and negative signals of a serial differential link might accidentally be swapped during board layout. Solutions such as a board respin or major updates to the PLD logic can be expensive. The transmitter polarity inversion feature is provided to correct this situation. The Standard PCS supports both the static and dynamic polarity inversion features.

Transmitter polarity inversion can be enabled in low latency, basic, and basic rate match modes.

To enable TX polarity inversion, select the **Enable TX polarity inversion** and **Enable tx_polinv port** options in Platform Designer. It can also be dynamically controlled with dynamic reconfiguration.

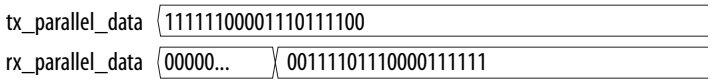
This mode adds `tx_polinv`. If there is more than one channel in the design, `tx_polinv` is a bus with each bit corresponding to a channel. As long as `tx_polinv` is asserted, the TX data transmitted has a reverse polarity.

2.9.2.14. TX Bit Reversal

The TX bit reversal feature can be enabled in low latency, basic, and basic rate match mode. The word aligner is available in any mode. This feature is parameter-based, and creates no additional ports. If there is more than one channel in the design, all channels have TX bit reversal.

To enable TX bit reversal, select the **Enable TX bit reversal** option in Platform Designer. It can also be dynamically controlled with dynamic reconfiguration.

Figure 110. TX Bit Reversal

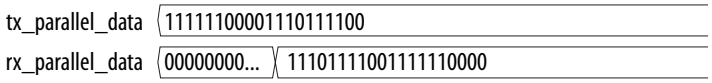


2.9.2.15. TX Byte Reversal

The TX byte reversal feature can be enabled in low latency, basic, and basic rate match mode. The word aligner is available in any mode. This feature is parameter-based, and creates no additional ports. If there is more than one channel in the design, all channels have TX byte reversal.

To enable TX byte reversal, select the **Enable TX byte reversal** option in Platform Designer. It can also be dynamically controlled with dynamic reconfiguration.

Figure 111. TX Byte Reversal

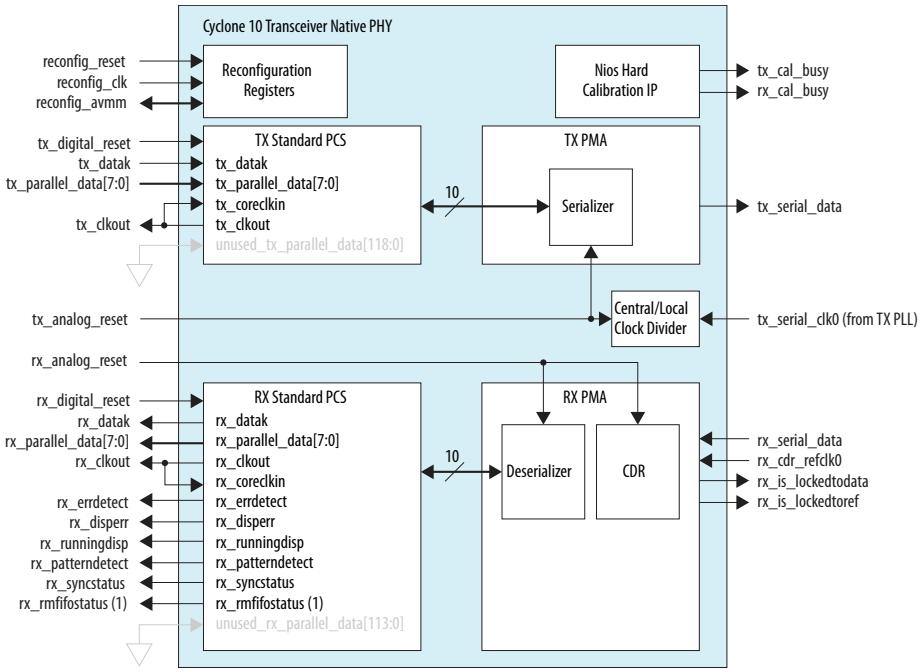


2.9.2.16. How to Implement the Basic, Basic with Rate Match Transceiver Configuration Rules in Cyclone 10 GX Transceivers

You should be familiar with the Standard PCS and PMA architecture, PLL architecture, and the reset controller before implementing your Basic protocol IP.

1. Open the IP Catalog and select the Native PHY IP.
   Refer to Select and Instantiate the PHY IP Core on page 17.
2. Select **Basic/Custom (Standard PCS)** or **Basic/Custom w/Rate Match (Standard PCS)** from the **Transceiver configuration rules** list located under **Datapath Options** depending on which configuration you want to use.
3. Use the parameter values in the tables in Transceiver Native PHY IP Parameter Settings for the Basic Protocol as a starting point. Or, you can use the protocol presets described in Transceiver Native PHY Presets. You can then modify the setting to meet your specific requirements.
4. Click **Finish** to generate the Native PHY IP (this is your RTL file).

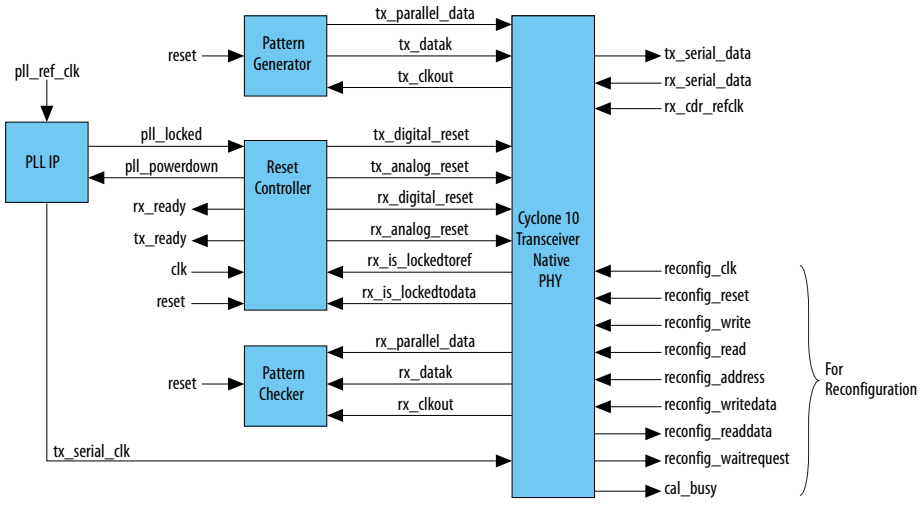
Figure 112. Signals and Ports of Native PHY IP for Basic, Basic with Rate Match Configurations



Note:
1. Only applies when using the Basic with Rate Match transceiver configuration rule.

5. Instantiate and configure your PLL.
6. Create a transceiver reset controller.
7. Connect the Native PHY IP to the PLL IP and the reset controller. Use the information in Transceiver Native PHY Ports for the Protocol to connect the ports.

Figure 113. Connection Guidelines for a Basic/Custom Design



8. Simulate your design to verify its functionality.

2.9.2.17. Native PHY IP Parameter Settings for Basic, Basic with Rate Match Configurations

This section contains the recommended parameter values for this protocol. Refer to *Using the Cyclone 10 GX Transceiver Native PHY IP Core* for the full range of parameter values.

Table 147. General and Datapath Options Parameters

Parameter	Range
Message level for rule violations	**error** **warning**
Transceiver configuration rules	**Basic/Custom (Standard PCS)** **Basic/Custom w/Rate Match (Standard PCS)**
PMA configuration rules	**basic**
Transceiver mode	**TX/RX Duplex** **TX Simplex** **RX Simplex**
Number of data channels	1 to 12
Data rate	611 Mbps to 10.81344 Gbps
Enable datapath and interface reconfiguration	**On/Off**
Enable simplified data interface	**On/Off**

Table 148. TX PMA Parameters

Parameter	Range
TX channel bonding mode	**Not bonded** **PMA-only bonding** **PMA and PCS bonding**
PCS TX channel bonding master	**Auto**, **n-1** (where n = the number of data channels)
Actual PCS TX channel bonding master	**n-1** (where n = the number of data channels)
TX local clock division factor	**1**, **2**, **4**, **8**
Number of TX PLL clock inputs per channel	**1**, **2**, **3**, **4**
Initial TX PLL clock input selection	**0** (Depends on the **Number of TX PLL clock inputs per channel** value)
Enable tx_pma_clkout port	**On/Off**
Enable tx_pma_div_clkout port	**On/Off**
tx_pma_div_clkout division factor	**Disabled**, **1**, **2**, **33**, **40**, **66**
Enable tx_pma_elecidle port	**On/Off**
Enable rx_seriallpbken port	**On/Off**

Table 149. RX PMA Parameters

Parameter	Range
Number of CDR reference clocks	**1**, **2**, **3**, **4**, **5**
Selected CDR reference clock	**0**, **1**, **2**, **3**, **4**
Selected CDR reference clock frequency	Legal range defined by Quartus Prime software
PPM detector threshold	**100**, **300**, **500**, **1000**
CTLE adaptation mode	**manual**
Enable rx_pma_clkout port	**On/Off**
Enable rx_pma_div_clkout port	**On/Off**
rx_pma_div_clkout division factor	**Disabled**, **1**, **2**, **33**, **40**, **50**, **66**
Enable rx_pma_clkslip port	**On/Off**
Enable rx_is_lockedtodata port	**On/Off**
Enable rx_is_lockedtoref port	**On/Off**
Enable rx_set_locktodata and rx_set_locktoref ports	**On/Off**
Enable rx_seriallpbken port	**On/Off**
Enable PRBS verifier control and status ports	**On/Off**

Table 150. Standard PCS Parameters

Parameter	Range
Standard PCS / PMA interface width	**8**, **10**, **16**, **20**
FPGA fabric / Standard TX PCS interface width	**8**, **10**, **16**, **20**, **32**, **40**
FPGA fabric / Standard RX PCS interface width	**8**, **10**, **16**, **20**, **32**, **40**
Enable 'Standard PCS' low latency mode	**On/Off** **Off** (for Basic with Rate Match)
TX FIFO mode	**low_latency** **register_fifo** **fast_register**
RX FIFO Mode	**low_latency** **register_fifo**
Enable tx_std_pcfifo_full port	**On/Off**
Enable tx_std_pcfifo_empty port	**On/Off**
Enable rx_std_pcfifo_full port	**On/Off**
Enable rx_std_pcfifo_empty port	**On/Off**
TX byte serializer mode	**Disabled** **Serialize x2** **Serialize x4**
RX byte deserializer mode	**Disabled** **Deserialize x2** **Deserialize x4**
Enable TX 8B/10B encoder	**On/Off**

continued...

Parameter	Range
Enable TX 8B/10B disparity control	**On/Off**
Enable RX 8B/10B decoder	**On/Off**
RX rate match FIFO mode	**Disabled** **Basic 10-bit PMA** (for Basic with Rate Match) **Basic 20-bit PMA** (for Basic with Rate Match)
RX rate match insert/delete -ve pattern (hex)	User-defined value
RX rate match insert/delete +ve pattern (hex)	User-defined value
Enable rx_std_rmfifo_full port	**On/Off**
Enable rx_std_rmfifo_empty port	**On/Off**
Enable TX bit slip	**On/Off**
Enable tx_std_bitslipboundarysel port	**On/Off**
RX word aligner mode	**bitslip** **manual (PLD controlled)** **synchronous state machine**
RX word aligner pattern length	**7**, **8**, **10**, **16**, **20**, **32**, **40**
RX word aligner pattern (hex)	User-defined value
Number of word alignment patterns to achieve sync	0-255
Number of invalid data words to lose sync	0-63
Number of valid data words to decrement error count	0-255
Enable fast sync status reporting for deterministic latency SM	**On/Off**
Enable rx_std_wa_patternalign port	**On/Off**
Enable rx_std_wa_a1a2size port	**On/Off**
Enable rx_std_bitslipboundarysel port	**On/Off**
Enable rx_bitslip port	**On/Off**
Enable TX bit reversal	**On/Off**
Enable TX byte reversal	**On/Off**
Enable TX polarity inversion	**On/Off**
Enable tx_polinv port	**On/Off**
Enable RX bit reversal	**On/Off**
Enable rx_std_bitrev_ena port	**On/Off**
Enable RX byte reversal	**On/Off**
Enable rx_std_byterev_ena port	**On/Off**
Enable RX polarity inversion	**On/Off**
Enable rx_polinv port	**On/Off**
Enable rx_std_signaldetect port	**On/Off**
Enable PCIe dynamic datarate switch ports	**Off**

continued...

Parameter	Range
Enable PCIe pipe_hclk_in and pipe_hclk_out ports	Off
Enable PCIe electrical idle control and status ports	Off
Enable PCIe pipe_rx_polarity port	Off

Table 151. Dynamic Reconfiguration Parameters

Parameter	Range
Enable dynamic reconfiguration	On/Off
Share reconfiguration interface	On/Off
Enable Native PHY Debug Master Endpoint	On/Off

Table 152. Generation Options Parameters

Parameter	Range
Generate parameter documentation file	On/Off

Related Information

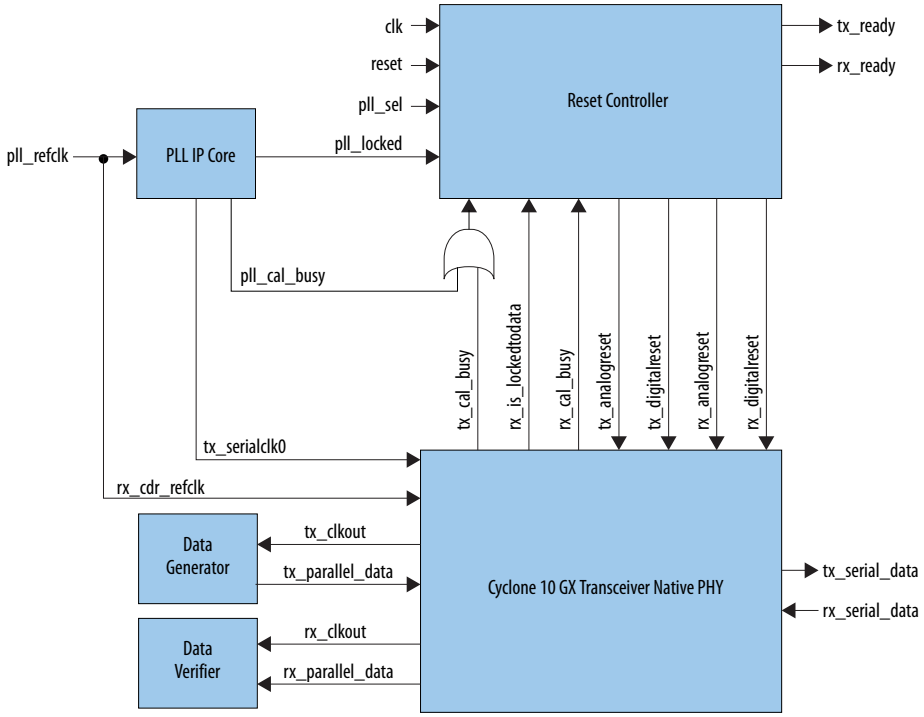
Using the Cyclone 10 GX Transceiver Native PHY IP Core on page 26

2.9.3. How to Implement PCS Direct Transceiver Configuration Rule

You should be familiar with PCS Direct architecture, PMA architecture, PLL architecture, and the reset controller before implementing PCS Direct Transceiver Configuration Rule.

1. Open the IP Catalog and select **Cyclone 10 GX Transceiver Native PHY IP**.
2. Select **PCS Direct** from the **Transceiver configuration rules** list located under **Datapath Options**.
3. Configure your Native PHY IP.
4. Click **Generate** to generate the Native PHY IP (this is your RTL file).
5. Instantiate and configure your PLL.
6. Create a transceiver reset controller. You can use your own controller or use the Transceiver PHY Reset Controller.
7. Connect the Native PHY IP to the PLL IP and the reset controller.

Figure 114. Connection Guidelines for a PCS Direct PHY Design



8. Simulate your design to verify its functionality.

2.10. Simulating the Transceiver Native PHY IP Core

Use simulation to verify the Native PHY transceiver functionality. The Quartus Prime software supports register transfer level (RTL) and gate-level simulation in both ModelSim® - Intel FPGA Edition and third-party simulators. You run simulations using your Quartus Prime project files.

The following simulation flows are available:

- NativeLink—This flow simplifies simulation by allowing you to start a simulation from the Quartus Prime software. This flow automatically creates a simulation script and compiles design files, IP simulation model files, and Intel simulation library models.

  Note: The Quartus Prime Pro Edition software does not support NativeLink RTL simulation

- Scripting IP Simulation—In this flow you perform the following actions:
  1. Run the ip-setup-simulation utility to generate a single simulation script that compiles simulation files for all the underlying IPs in your design. This script needs to be regenerated whenever you upgrade or modify IPs in the design.
  2. You create a top-level simulation script for compiling your testbench files and simulating the testbench. It sources the script generated in the first action. You do not have to modify this script even if you upgrade or modify the IPs in your design.
- Custom Flow—This flow allows you to customize simulation for more complex requirements. You can use this flow to compile design files, IP simulation model files, and Intel simulation library models manually.

You can simulate the following netlist:

- The RTL functional netlist—This netlist provides cycle-accurate simulation using Verilog HDL, SystemVerilog, and VHDL design source code. Intel and third-party EDA vendors provide the simulation models.

Prerequisites to Simulation

Before you can simulate your design, you must have successfully passed Quartus Prime Analysis and Synthesis.

Related Information

Simulating Altera Designs

2.10.1. NativeLink Simulation Flow

The NativeLink settings available in the Quartus Prime software allow you to specify your simulation environment, simulation scripts, and testbenches. The Quartus Prime software saves these settings in your project. After you specify the NativeLink settings, you can start simulations easily from the Quartus Prime software.

2.10.1.1. How to Use NativeLink to Specify a ModelSim Simulation

Complete the following steps to specify the directory path and testbench settings for your simulator:

1. On the **Tools** menu, click **Options**, and then click **EDA Tool Options**.
2. Browse to the directory for your simulator. The following table lists the directories for supported simulators:

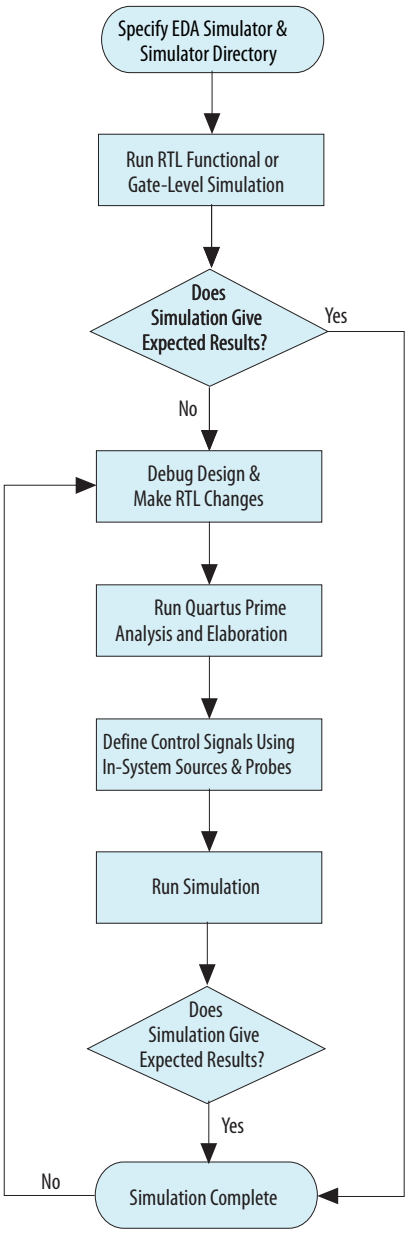
Simulator	Directory
Mentor Graphics ModelSim - Intel FPGA Edition	*<drive>*:*<simulator install path>***win32aloem** (Windows) /*<simulator install path>*/**bin** (Linux)

3. On the **Assignments** menu, click **Settings**.
4. In the **Category** list, under **EDA Tool Settings** select **Simulation**.
5. In the **Tool name** list, select your simulator.

   Note: ModelSim refers to ModelSim SE and PE. These simulators use the same commands as QuestaSim.ModelSim - Intel FPGA Edition refers to ModelSim - Intel FPGA Edition Starter Edition and ModelSim - Intel FPGA Edition Subscription Edition.
6. In the **Output directory**, browse to the directory for your output files.
7. To map illegal HDL characters, turn on **Map illegal HDL characters**.
8. To filter netlist glitches , turn on **Enable glitch filtering**.
9. Complete the following steps to specify additional options for NativeLink automation:
   a. Turn on **Compile test bench**.
   b. Click **Test Benches**.
      The **Test Benches** dialog box appears.
   c. Click **New**.
   d. Under **Create new test bench settings**, for **Test bench name** type the test bench name. For Top level module in the test bench, type the top-level module name. These names should match the actual test bench module names.
   e. Select **Use test bench to perform VHDL timing simulation** and specify the name of your design instance under **Design instance name in test bench**.
   f. Under the **Simulation period**, turn on **Run simulation until all vector stimuli are used**.
   g. Under **Test bench and simulation files**, select your test bench file from your folder. Click **Add**.
   h. Click **OK**.

2.10.1.2. How to Use NativeLink to Run a ModelSim RTL Simulation

Figure 115. NativeLink Simulation Flow Diagram



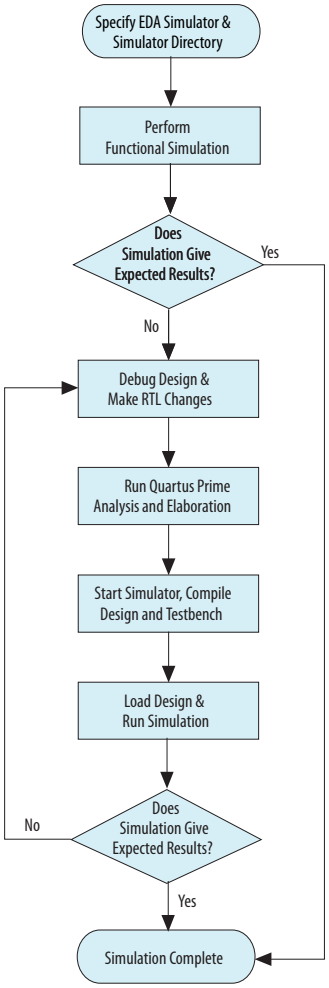
Complete the following steps to run an RTL functional simulation:

1. Open your Quartus Prime project.
2. On the Tools menu, select **Run Simulation Tool**, then select **RTL Simulation** or **Gate Level Simulation**.
3. Run Quartus Prime Analysis and Elaboration and re-instantiate control signals that you defined using the In-System Sources and Probe Editor. The In-System Sources and Probe Editor can only access the pins of the device. Consequently, you must route any signal that you want to observe to the top-level of your design.
4. To monitor additional signals, highlight the desired instances or nodes in **Instance**, and right-click **Add wave**.
5. Select **Simulate** and then **Run**.
6. Specify the simulation duration.
7. Complete the following steps to restart the simulation:
   a. On the Simulate menu, select **restart**, then click **ok**.
      This action clears the existing waves.
   b. Highlight **run** and select the appropriate options to run the simulation.

2.10.1.3. How to Use NativeLink to Specify Third-Party RTL Simulators

The following figure illustrates the high-level steps for using the NativeLink with Third-Party EDA RTL simulator.

Figure 116. Using NativeLink with Third-Party Simulators



Complete the following steps to specify the directory path and testbench settings for your simulator:

1. On the **Tools** menu, click **Options**, and then click **EDA Tool Options**.
2. Browse to the directory for your simulator. The following table lists the directories for supported third-party simulators:

Table 153. Simulator Path

Simulator	Path
Mentor Graphics ModelSim Mentor Graphics QuestaSim	*<drive>*:*<simulator install path>***win32** (Windows) /*<simulator install path>*/**bin** (Linux)
Synopsys VCS/VCS MX	/*<simulator install path>*/**bin** (Linux)
Cadence Incisive Enterprise	/*<simulator install path>*/**tools**/**bin** (Linux)
Aldec Active-HDL Aldec Riviera-Pro	*<drive>*:*<simulator install path>***bin** (Windows) /*<simulator install path>*/**bin** (Linux)

3. On Assignments menu, click **Settings**.

4. In the **Category** list, under **EDA Tool Settings**, select **Simulation**.
5. In the **Tool name** list, select your simulator.
6. To enable your simulator, on the **Tools** menu, click **Options** and then click **License Setup** . Make necessary changes for EDA tool licenses.
7. Compile your design and testbench files.
8. Load the design and run the simulation in the EDA tool.

To learn more about third-party simulators, click on the appropriate link below.

Related Information

- Mentor Graphics ModelSim and QuestaSim Support
- Synopsys VCS and VCS MX Support
- Cadence Incisive Enterprise Simulator Support
- Aldec Active-HDL and Riviera-Pro Support

2.10.2. Scripting IP Simulation

The Intel Quartus Prime software supports the use of scripts to automate simulation processing in your preferred simulation environment. You can use your preferred scripting methodology to control simulation.

Intel recommends the use of a version-independent top-level simulation script to control design, testbench, and IP core simulation. Because Quartus Prime-generated simulation file names may change.

You can use the ip-setup simulation utility to generate or regenerate underlying setup scripts after any software or IP version upgrade or regeneration. Use of a top-level script and ip-setup-simulation eliminates the requirement to manually update simulation scripts.

2.10.2.1. Generating a Combined Simulator Setup Script

Platform Designer system generation creates the interconnect between components. It also generates files for synthesis and simulation, including the **.spd** files necessary for the ip-setup-simulation utility.

The Intel Quartus Prime software provides utilities to help you generate and update IP simulation scripts. You can use the ip-setup-simulation utility to generate a combined simulator setup script, for all Intel FPGA IP in your design, for each supported simulator. You can subsequently rerun ip-setup-simulation to automatically update the combined script. Each simulator's combined script file contains a rudimentary template that you can adapt for integration of the setup script into a top-level simulation script.

Related Information

Quartus II Handbook Volume 3: Verification
Provides more information about the steps to generate top-level simulation script.

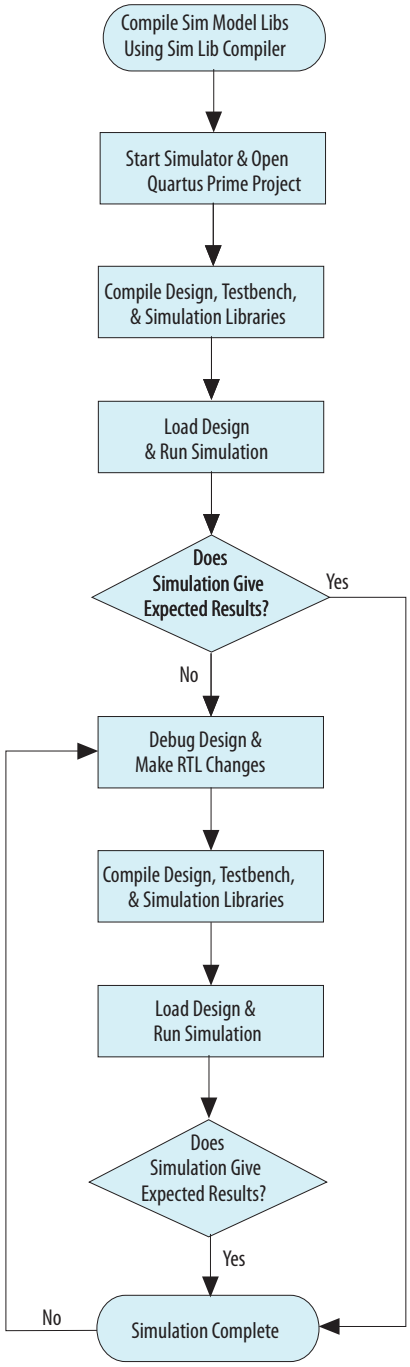
2.10.3. Custom Simulation Flow

The custom simulation flow allows you to customize the simulation process for more complex simulation requirements. This flow allows you to control the following aspects of your design:

- Component binding
- Compilation order
- Run commands
- IP cores
- Simulation library model files

The following figure illustrates the steps for custom flow simulation. If you use a simulation script, you can automate some of the steps.

Figure 117. Custom flow Simulation



2.10.3.1. How to Use the Simulation Library Compiler

The Simulation Library Compiler compiles Intel simulation libraries for supported simulation tools, and saves the simulation files in the output directory you specify.

Note: Because the ModelSim - Intel FPGA Edition software provides precompiled simulation libraries, you do not have to compile simulation libraries if you are using the software.

Complete the following steps to compile the simulation model libraries using the Simulation Library Compiler:

1. On the Tools menu, click **Launch Simulation Library Compiler**.
2. Under **EDA simulation tool**, for the **Tool name**, select your simulation tool.
3. Under **Executable location**, browse to the location of the simulation tool you specified. You must specify this location before you can run the EDA Simulation Library Compiler.
4. Under **Library families**, select one or more family names and move them to the **Selected families** list.
5. Under **Library language**, select **Verilog**, **VHDL**, or both.
6. In the **Output directory** field, specify a location to store the compiled libraries.
7. Click **Start Compilation**.

Complete the following steps to add the simulation files to your project:

1. On the Assignments menu, click **Settings**.
2. In the **Category** list, select **Files**.
3. Click **Browse** to open the **Select File** dialog box and select one or more files in the **Files** list to add to your project.
4. Click **Open**, and then **Add** to add the selected file(s) to your project.
5. Click **OK** to close the **Settings** dialog box.

Related Information

- Preparing for EDA Simulation
- Altera Simulation Models

2.10.3.2. Custom Simulation Scripts

You can automate simulations by creating customized scripts. You can generate scripts manually. In addition, you can use NativeLink to generate a simulation script as a template and then make the necessary changes. The following table shows a list of script directories NativeLink generates.

Table 154. Custom Simulation Scripts for Third Party RTL Simulation

Simulator	Simulation File	Use
Mentor Graphics ModelSim or QuestaSim	**/simulation/ modelsim/ modelsim_setup.do** Or **mentor/msim_setup.tcl**	Source directly with your simulator. Run `do msim_setup.tcl`, followed by `ld_debug`. If you have more than one IP, each IP has a dedicated `msim_setup.tcl` file. Make sure that you combine all the files included in the `msim_setup.tcl` files into one common `msim_setup.tcl` file.
Aldec Riviera Pro	**/simulation/ aldec/ rivierapro_setup.tcl**	Source directly with your simulator.

continued...

Simulator	Simulation File	Use
Synopsys VCS	**/simulation/synopsys/vcs/vcs_setup.sh**	Add your testbench file name to this file to pass the testbench file to VCS using the `–file` option. If you specify a testbench file for NativeLink and do not choose to simulate, NativeLink generates a script that runs VCS.
Synopsys VCS MX	**/simulation/synopsys/vcsmx/vcsmx_setup.sh**	Run this script at the command line using `quartus_sh–t <script>`. Any testbench you specify with NativeLink is included in this script.
Cadence Incisive (NCSim)	**/simulation/cadence/ncsim_setup.sh**	Run this script at the command line using `quartus_sh –t <script>`. Any testbench you specify with NativeLink is included in this script.

2.11. Implementing Protocols in Intel Cyclone 10 GX Transceivers Revision History

Document Version	Changes
2019.12.13	Made the following changes to the "1G/2.5G/5G/10G Multi-rate Ethernet PHY Intel FPGA IP Core" section: • Updated the Clocking and Reset Sequence topic to state that the 1G/ 2.5G/5G/10G Multi-rate Ethernet PHY Intel FPGA IP core for Intel Cyclone 10 GX devices supports up to ±100 ppm clock frequency difference for a maximum packet length of 16,000 bytes.
2019.06.12	Made the following change: • Clarified that the Enhanced PCS only supports the static polarity inversion feature, but the Standard PCS supports both the static and dynamic polarity inversion features.
2019.05.13	Made the following change: • Renamed Altera Debug Master Endpoint (ADME) to Native PHY DebugMaster Endpoint (NPDME). • Updated Table: *Ethernet Acronyms*.
2018.09.24	Made the following changes to the "Using the Cyclone 10 GX Transceiver Native PHY IP Core" section: • Added details about how to enable the transceiver toolkit capability in the "Dynamic Reconfiguration Parameters" section. Made the following changes to the "1G/2.5G/5G/10G Multi-rate Ethernet PHY Intel FPGA IP Core" section: • Added this section.
2017.12.04	Made the following changes: • Removed the `tx_pma_txdetectrx[<n>-1:0]` port from the "TX PMA Ports" table.
2017.11.30	Made the following changes in the "PCI Express" section: • Removed the following parameters from the "Parameters for Intel Cyclone 10 GX Native PHY IP in PIPE Gen1, Gen2 Modes - TX PMA" table: — **Enable tx_pma_qpipullup port (QPI)** — **Enable tx_pma_qpipulldn port (QPI)** — **Enable tx_pma_txdetectrx port (QPI)** — **Enable tx_pma_rxfound port (QPI)** • Removed the **Enable rx_pma_qpipulldn port (QPI)** parameter from the "Parameters for Intel Cyclone 10 GX Native PHY IP in PIPE Gen1, Gen2 Modes - RX PMA" table. Made the following changes to the "Using the Transceiver Native PHY IP Core" section: ***continued...***

Document Version	Changes
	• Removed the QPI protocol mode from the **PMA configuration rules** parameter in the "General, Common PMA Options, and Datapath Options" table. • Removed the following parameters from the "TX PMA Parameters" table: — **Enable tx_pma_qpipullup port (QPI)** — **Enable tx_pma_qpipulldn port (QPI)** — **Enable tx_pma_txdetectrx port (QPI)** — **Enable tx_pma_rxfound port (QPI)**
2017.11.06	Made the following changes to the "Other Protocols" section: • Changed Native PHY IP data rate to 12.5 from 10.3125 Made the following changes to the "PCI Express" section: • Added note "Connect `pll_pcie_clk` from either ATX PLL or fPLL to the `pipe_hclk_in` port on Native PHY" in figure "Use fPLL for Gen1/Gen2 x1 Mode". • Changed the clock frequency differential in the "Gen1 and Gen2 Clock Compensation" section introduction paragraph. Made the following changes to the "Transceiver Design Flow Overview" section: • Added a note "Link training, auto speed negotiation and sequencer functions are not included in the Native PHY IP. The user would have to create soft logic to implement these functions when using Native PHY IP" in the "Transceiver Protocols and PHY IP Support" section. • Added a row for "CPRI 4.1/OBSAI RP3 v4.1" protocol in the "Transceiver Protocols and PHY IP Support" section. • Added footnote "For x2 and x4 modes, select PCIe PIPE Gen2 x8. Then change the number of data channels from 8 to 4. " for "PCIe Gen2 x1, x2, x4" protocol in the "Transceiver Protocols and PHY IP Support" section. • Added footnotes in the "Transceiver Protocols and PHY IP Support" section. • Updated protocol presets for "SD-SDI/HD-SDI/3G/6G/12G-SDI ", "DisplayPort" and "CPRI 4.1/OBSAI RP3 v4.1" protocols in the "Transceiver Protocols and PHY IP Support" section. • Added a note that Intel Cyclone 10 GX is only supported with Intel Quartus Prime Pro Edition 17.1 and future versions in the "Transceiver Design IP Blocks" section. • Added a note "Intel Cyclone 10 GX only supported with Intel Quartus Prime Pro Edition 17.1 and future versions." • Added a note "Link training, auto speed negotiation and sequencer functions are not included in the Native PHY IP. The user would have to create soft logic to implement these functions when using Native PHY IP. " in the "Protocols and PHY IP Support" table. • Added CPRI 4.1/OBSAI RP3 v4.1 protocol in "Protocols and PHY IP Support" table. Made the following changes in the "10GBASE-R" section: • Added the "Native PHY IP Parameter Settings for 10GBASE-R and 10GBASE-R with IEEE 1588v2" section. • Changed the range of values for the **Number of data channels** parameter in the "General and Datapath Parameters" table. • Changed the range of values for the **Initial TX PLL clock input selection** parameter in the "TX PMA Parameters" table. Made the following changes to the "Gigabit Ethernet (GbE) and GbE with 1588" section: • Changed the note in the "Rate Match FIFO for GbE" section. • Added the "Native PHY IP Parameter Settings for GbE and GbE with IEEE 1588v2" section. Made the following changes to the "Using the Intel Cyclone 10 GX Transceiver Native PHY IP Core" section: • Added this chapter.
2017.05.08	Initial release.

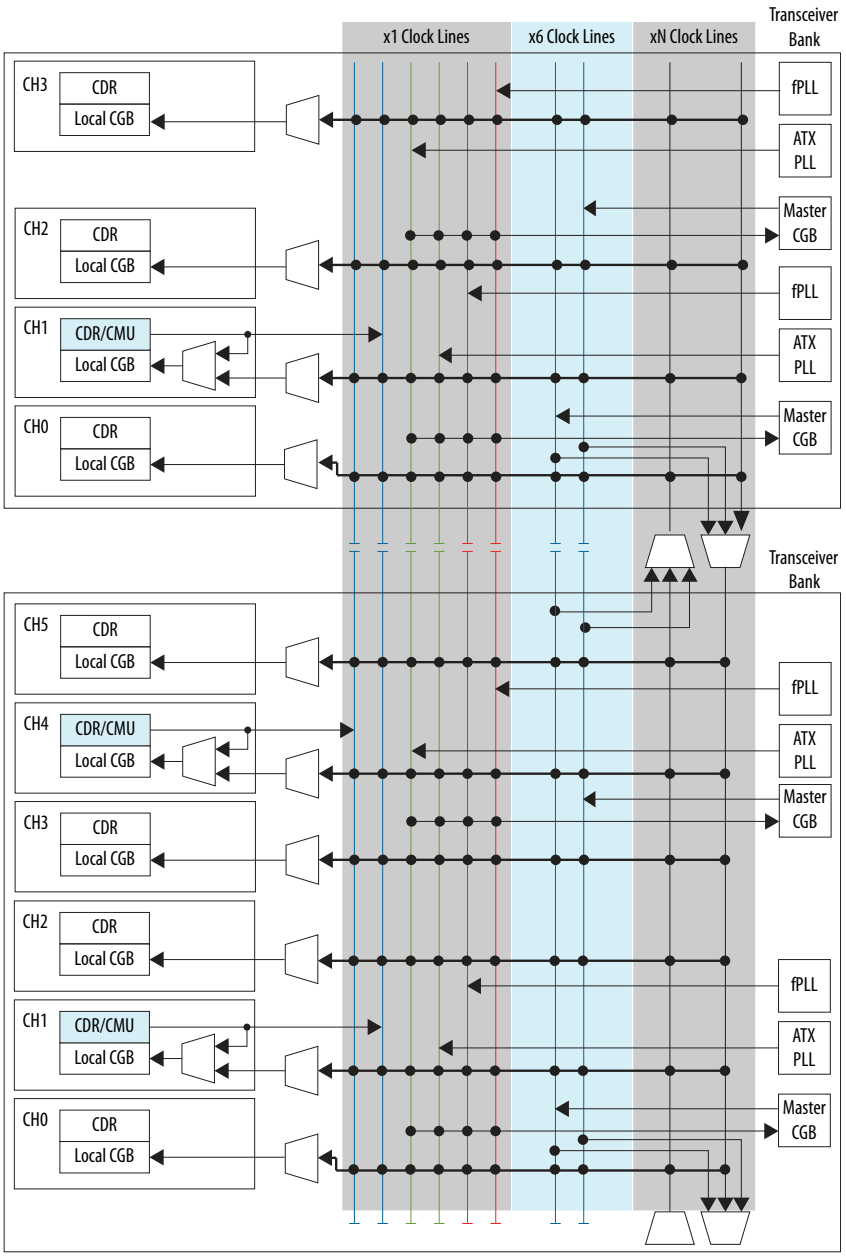
3. PLLs and Clock Networks

This chapter describes the transceiver phase locked loops (PLLs), internal clocking architecture, and the clocking options for the transceiver and the FPGA fabric interface.

As shown in the following figure, transceiver banks can have either four or six transceiver channels. For every three channels, you get one advanced transmit (ATX) PLL, one fractional PLL (fPLL), and one Master clock generation block (CGB). Refer to the *Device Transceiver Layout* section to identify which devices have three channel transceiver banks.

The Cyclone 10 GX transceiver clocking architecture supports both bonded and non-bonded transceiver channel configurations. Channel bonding is used to minimize the clock skew between multiple transceiver channels. For Cyclone 10 GX transceivers, the term bonding can refer to PMA bonding as well as PMA and PCS bonding. Refer to the *Channel Bonding* section for more details.

Figure 118. Cyclone 10 GX PLLs and Clock Networks



Related Information

- Channel Bonding on page 222
- Device Transceiver Layout on page 8
- Using PLLs and Clock Networks on page 231
  Information on how to use PLL IP to implement bonded and non-bonded transceiver designs.

3.1. PLLs

Table 155. Transmit PLLs in Cyclone 10 GX Devices

PLL Type	Characteristics
Advanced Transmit (ATX) PLL	• Best jitter performance • LC tank based voltage controlled oscillator (VCO) • Used for both bonded and non-bonded channel configurations
Fractional PLL (fPLL)	• Ring oscillator based VCO • Supports fractional synthesis mode • Used for both bonded and non-bonded channel configurations
Clock Multiplier Unit (CMU) PLL or Channel PLL (27)	• Ring oscillator based VCO • Used as an additional clock source for non-bonded applications

Related Information

Refer to *Using PLL and Clock Networks* section for guidelines and usage on page 231

3.1.1. Transmit PLLs Spacing Guidelines when using ATX PLLs and fPLLs

ATX PLL-to-ATX PLL Spacing Guidelines

ATX PLLs' VCO frequency offset must be 100 MHz apart. If this requirement cannot be met, use fPLL as transmit PLL to avoid more than one ATX PLL usage. For applications that requires multi-data rate support, use TX PLL switching or TX local clock dividers to achieve the desire data rate reconfiguration.

Note: You are not allowed to recalibrate an ATX PLL if there are TX channels driven by another ATX PLL in transmitting mode.

ATX PLL-to-fPLL Spacing Guidelines

If you are using both ATX PLL and fPLL, and you meet the below two conditions in your applications:

- When ATX PLL VCO frequency and fPLL VCO frequency is within 50 MHz.
- ATX PLL is used to drive 6G or 12G SDI protocol.

The ATX PLL and fPLL must be separated at least by one ATX PLL in between.

If you are using both ATX PLL and fPLL, and you meet the following two conditions in your applications:

- fPLL user re-calibration process is triggered.
- ATX PLL is used to drive 6G or 12G SDI protocol.

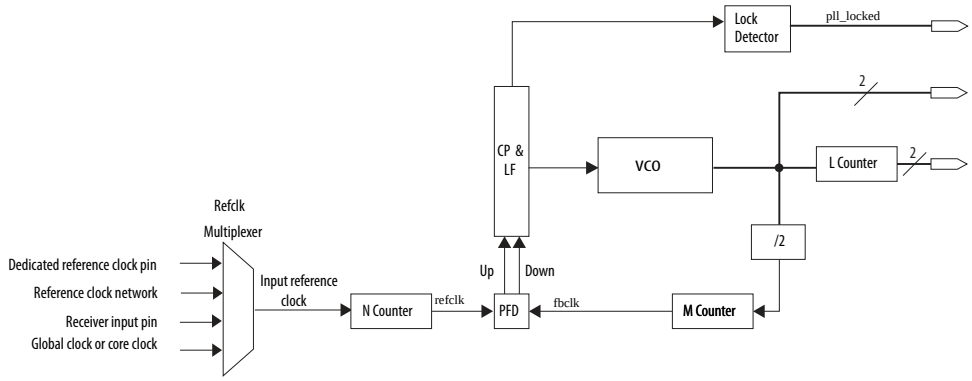
then the ATX PLL and fPLL must be separated at least by one ATX PLL in between (regardless of the ATX PLL and fPLL VCO frequency offset).

(27) The CMU PLL or Channel PLL of channel 1 and channel 4 can be used as a transmit PLL or as a clock data recovery (CDR) block. The channel PLL of all other channels (0, 2, 3, and 5) can only be used as a CDR.

3.1.2. ATX PLL

The ATX PLL contains LC tank-based voltage controlled oscillators (VCOs). These LC VCOs have different frequency ranges to support a continuous range of operation. When driving the transceiver directly, the ATX PLL only supports the integer mode.

Figure 119. ATX PLL Block Diagram



Input Reference Clock

This is the dedicated input reference clock source for the PLL.

The input reference clock can be sourced from one of the following:

- Dedicated reference clock pin
- Reference clock network
- Receiver input pin
- Global clock or the core clock network

The input reference clock to the dedicated reference clock pin is a differential signal. Intel recommends using the dedicated reference clock pin as the input reference clock source for the best jitter performance. The input reference clock must be stable and free-running at device power-up for proper PLL operation and PLL calibration. If the reference clock is not available at device power-up, then you must recalibrate the PLL when the reference clock is available.

Note: The ATX PLL calibration is clocked by the CLKUSR clock which must be stable and available for calibration to proceed. Refer to the *Calibration* section for more details about the CLKUSR clock.

Reference Clock Multiplexer

The reference clock (`refclk`) multiplexer selects the reference clock to the PLL from the various reference clock sources available.

N Counter

The N counter divides the `refclk` mux's output. The division factors supported are 1, 2, 4, and 8.

Phase Frequency Detector (PFD)

The reference clock(`refclk`) signal at the output of the N counter block and the feedback clock (`fbclk`) signal at the output of the M counter block are supplied as inputs to the PFD. The output of the PFD is proportional to the phase difference between the `refclk` and `fbclk` inputs. It is used to align the `refclk` signal at the output of the N counter to the feedback clock (`fbclk`) signal. The PFD generates an "Up" signal when the reference clock's falling edge occurs before the feedback clock's falling edge. Conversely, the PFD generates a "Down" signal when the feedback clock's falling edge occurs before the reference clock's falling edge.

Charge Pump and Loop Filter

The PFD output is used by the charge pump and loop filter (CP and LF) to generate a control voltage for the VCO. The charge pump translates the "Up" or "Down" pulses from the PFD into current pulses. The current pulses are filtered through a low pass filter into a control voltage that drives the VCO frequency. The charge pump, loop filter, and VCO settings determine the bandwidth of the ATX PLL.

Lock Detector

The lock detector block indicates when the reference clock and the feedback clock are phase aligned. The lock detector generates an active high `pll_locked` signal to indicate that the PLL is locked to its input reference clock.

Voltage Controlled Oscillator

The voltage controlled oscillator (VCO) used in the ATX PLL is LC tank based. The output of charge pump and loop filter serves as an input to the VCO. The output frequency of the VCO depends on the input control voltage. The output frequency is adjusted based on the output voltage of the charge pump and loop filter.

L Counter

The L counter divides the differential clocks generated by the ATX PLL. The L counter is not in the feedback path of the PLL.

M Counter

The M counter's output is the same frequency as the N counter's output. The VCO frequency is governed by the equation:

VCO freq = 2 * M * input reference clock/N

An additional divider divides the high speed serial clock output of the VCO by 2 before it reaches the M counter.

The M counter supports division factors in a continuous range from 8 to 127 in integer frequency synthesis mode.

Multiple Reconfiguration Profiles

Under the ATX PLL IP Parameter Editor Dynamic Reconfiguration tab, in the Configuration Profiles section, multiple reconfiguration profiles can be enabled. This allows to create, store, and analyze the parameter settings for multiple configurations or profiles of the ATX PLL IP.

The ATX PLL IP GUI can generate configuration files (SystemVerilog, C header or MIF) for a given configuration. With the multi reconfiguration profile options enabled, the ATX PLL IP Parameter Editor can produce configuration files for all of the profiles simultaneously. In addition, by enabling the reduced reconfiguration files generation, the IP Parameter Editor produces a reduced configuration file by internally comparing the corresponding parameter settings of all the profiles and identifying the differences.

Embedded Reconfiguration Streamer

This option enables a push-button flow to reconfigure between multiple configurations or profiles. Here are the steps to follow:

1. Multiple reconfiguration profiles creation
   - In the ATX PLL IP GUI, create configurations for each profiles using the multi-profile feature.
2. Reconfiguration report files
   - The IP GUI generates the reconfiguration report files that contain parameter and register settings for all the selected profiles. If the reduced reconfiguration files option is selected, the IP parameter editor compares the settings between the profiles and generate reduced report files which only contain the differences.
3. Select **Enable embedded reconfiguration streamer logic** in the GUI to generate the following:
   - Necessary HDL files to perform streaming
   - The individual report files for each profile, an SystemVerilog package file with configuration data for all the profiles concatenated together which is used to initialize the configuration ROM
4. Generate the ATX PLL IP and control the reconfiguration streamer using the AVMM master.
5. If you reconfigure PLL for data rate change, you must recalibrate the PLL.

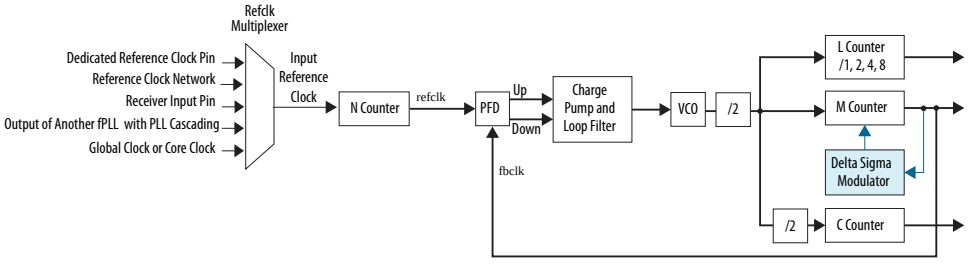
Related Information

Calibration on page 373

3.1.3. fPLL

There are two fPLLs in each transceiver bank with six channels (one located at the top and the other at the bottom of the bank). Transceiver banks with three channels have only one fPLL.

Figure 120. fPLL Block Diagram



When in core mode, for the fPLL to generate output clocks with a fixed frequency and phase relation to an input reference clock, the **Enable phase alignment** option must be selected. In the fractional frequency mode, the fPLL supports data rates from 1 Gbps to 12.5 Gbps.

Input Reference Clock

This is the dedicated input reference clock source for the PLL.

The input reference clock can be sourced from one of the following:

- Dedicated reference clock pin
- Reference clock network
- Receiver input pin
- Output of another fPLL with PLL cascading
- Global clock or the core clock network

The input reference clock is a differential signal. Intel recommends using the dedicated reference clock pin as the input reference clock source for best jitter performance. For protocol jitter compliance at datarate > 10 Gbps, Intel recommends using the dedicated reference clock pin in the same triplet with the fPLL as the input reference clock source. The input reference clock must be stable and free-running at device power-up for proper PLL operation. If the reference clock is not available at device power-up, then you must recalibrate the PLL when the reference clock is available.

Note: The fPLL calibration is clocked by the CLKUSR clock, which must be stable and available for the calibration to proceed. Refer to the Calibration on page 373 section for details about PLL calibration and CLKUSR clock.

Reference Clock Multiplexer

The `refclk` mux selects the reference clock to the PLL from the various available reference clock sources.

N Counter

The N counter divides the reference clock (`refclk`) mux's output. The N counter division helps lower the loop bandwidth or reduce the frequency within the phase frequency detector's (PFD) operating range. The N counter supports division factors from 1 to 32.

Phase Frequency Detector

The reference clock (`refclk`) signal at the output of the N counter block and the feedback clock (`fbclk`) signal at the output of the M counter block are supplied as an inputs to the PFD. The output of the PFD is proportional to the phase difference between the `refclk` and `fbclk` inputs. The PFD aligns the `fbclk` to the `refclk`. The PFD generates an "Up" signal when the reference clock's falling edge occurs before the feedback clock's falling edge. Conversely, the PFD generates a "Down" signal when the feedback clock's falling edge occurs before the reference clock's falling edge.

Charge Pump and Loop Filter (CP + LF)

The PFD output is used by the charge pump and loop filter to generate a control voltage for the VCO. The charge pump translates the "Up"/"Down" pulses from the PFD into current pulses. The current pulses are filtered through a low pass filter into a control voltage that drives the VCO frequency.

Voltage Controlled Oscillator

The fPLL has a ring oscillator based VCO. The VCO transforms the input control voltage into an adjustable frequency clock.

VCO freq = 2 * M * Input reference clock/N. (N and M are the N counter and M counter division factors.)

L Counter

The L counter divides the VCO's clock output. When the fPLL acts as a transmit PLL, the output of the L counter drives the clock generation block (CGB) and the TX PMA via the X1 clock lines.

M Counter

The M counter divides the VCO's clock output. The M counter can select any VCO phase. The outputs of the M counter and N counter have same frequency. M counter range is 8 to 127 in integer mode and 11 to 123 in fractional mode.

Delta Sigma Modulator

The delta sigma modulator is used in fractional mode. It modulates the M counter divide value over time so that the PLL can perform fractional frequency synthesis.

In fractional mode, the M value is as follows:

M (integer) + K/2^32, where K is the fractional multiply factor (K) in the fPLL IP Parameter Editor. The legal values of K are greater than 1% and less than 99% of the full range of 2^32 and can only be manually entered in the fPLL IP Parameter Editor in the Quartus Prime software.

The output frequencies can be exact when the fPLL is configured in fractional mode. Due to the K value 32-bit resolution, translating to 1.63 Hz step for a 7 GHz VCO frequency, not all desired fractional values can be achieved exactly. The lock signal is not available, when configured in fractional mode in the K-precision mode (K < 0.1 or K > 0.9).

C Counter

The fPLL C counter division factors range from 1 to 512.

Dynamic Phase Shift

The dynamic phase shift block allows you to adjust the phase of the C counters in user mode. In fractional mode, dynamic phase shift is only available for the C counters.

Latency

The C counters can be configured to select any VCO phase and a delay of up to 128 clock cycles. The selected VCO phase can be changed dynamically.

Related Information

Calibration on page 373

3.1.4. CMU PLL

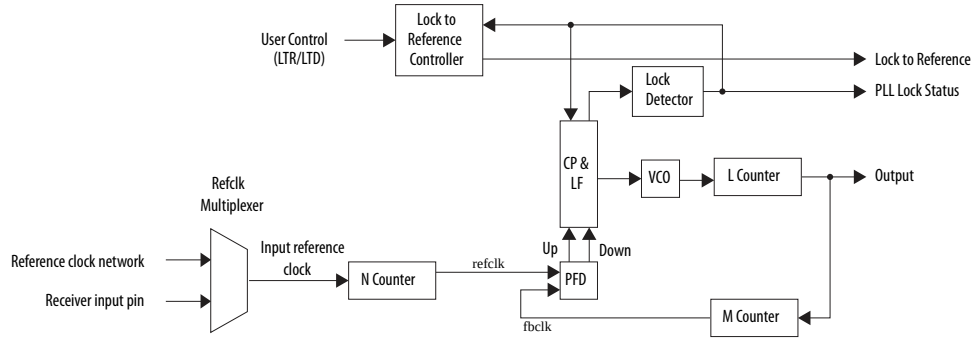
The clock multiplier unit (CMU) PLL resides locally within each transceiver channel. The channel PLL's primary function is to recover the receiver clock and data in the transceiver channel. In this case the PLL is used in clock and data recovery (CDR) mode.

When the channel PLL of channels 1 or 4 is configured in the CMU mode, the channel PLL can drive the local clock generation block (CGB) of its own channel, then the channel cannot be used as a receiver.

The CMU PLL from transceiver channel 1 and channel 4 can also be used to drive other transceiver channels within the same transceiver bank. The CDR of channels 0, 2, 3, and 5 cannot be configured as a CMU PLL.

For datarates lower than 6 Gbps, the local CGB divider has to be engaged (TX local division factor in transceiver PHY IP under the TX PMA tab).

Figure 121. CMU PLL Block Diagram



Input Reference Clock

The input reference clock for a CMU PLL can be sourced from either the reference clock network or a receiver input pin. The input reference clock is a differential signal. For protocol jitter compliance at a datarate > 10 Gbps, Intel recommends using the dedicated reference clock pin in the same triplet with the CMU PLL as the input reference clock source. The input reference clock must be stable and free-running at device power-up for proper PLL operation. If the reference clock is not available at device power-up, then you must recalibrate the PLL when the reference clock is available. Refer to the *Calibration* section for details about PLL calibration and the CLKUSR clock requirement.

Note: The CMU PLL calibration is clocked by the CLKUSR clock which must be stable and available for calibration to proceed. Refer to the *Calibration* section for more details about the CLKUSR clock.

Reference Clock Multiplexer (Refclk Mux)

The refclk mux selects the input reference clock to the PLL from the various reference clock sources available.

N Counter

The N counter divides the refclk mux's output. The N counter division helps lower the loop bandwidth or reduce the frequency to within the phase frequency detector's (PFD) operating range. Possible divide ratios are 1 (bypass), 2, 4, and 8.

Phase Frequency Detector (PFD)

The reference clock (`refclk`) signal at the output of the N counter block and the feedback clock (`fbclk`) signal at the output of the M counter block is supplied as an input to the PFD. The PFD output is proportional to the phase difference between the two inputs. It aligns the input reference clock (`refclk`) to the feedback clock (`fbclk`). The PFD generates an "Up" signal when the reference clock's falling edge occurs before the feedback clock's falling edge. Conversely, the PFD generates a "Down" signal when feedback clock's falling edge occurs before the reference clock's falling edge.

Charge Pump and Loop Filter (CP + LF)

The PFD output is used by the charge pump and loop filter to generate a control voltage for the VCO. The charge pump translates the "Up"/"Down" pulses from the PFD into current pulses. The current pulses are filtered through a low pass filter into a control voltage which drives the VCO frequency.

Voltage Controlled Oscillator (VCO)

The CMU PLL has a ring oscillator based VCO. For VCO frequency range, refer to the datasheet.

L Counter

The L counter divides the differential clocks generated by the CMU PLL.

M Counter

The M counter is used in the PFD's feedback path. The output of the L counter is connected to the M counter. The combined division ratios of the L counter and the M counter determine the overall division factor in the PFD's feedback path.

Lock Detector (LD)

The lock detector indicates when the CMU PLL is locked to the desired output's phase and frequency. The lock detector XORs the "Up"/"Down" pulses and indicates when the M counter's output and N counter's output are phase-aligned.

The reference clock (`refclk`) and feedback clock (`fbclk`) are sent to the PCS's ppm detector block. There is a pre-divider to lower the frequency in case the frequency is too high.

Related Information

- Calibration on page 373
- Cyclone 10 GX Device Datasheet

3.2. Input Reference Clock Sources

The transmitter PLL and the clock data recovery (CDR) block need an input reference clock source to generate the clocks required for transceiver operation. The input reference clock must be stable and free-running at device power-up for proper PLL calibrations.

Cyclone 10 GX transceiver PLLs have five possible input reference clock sources, depending on jitter requirements:

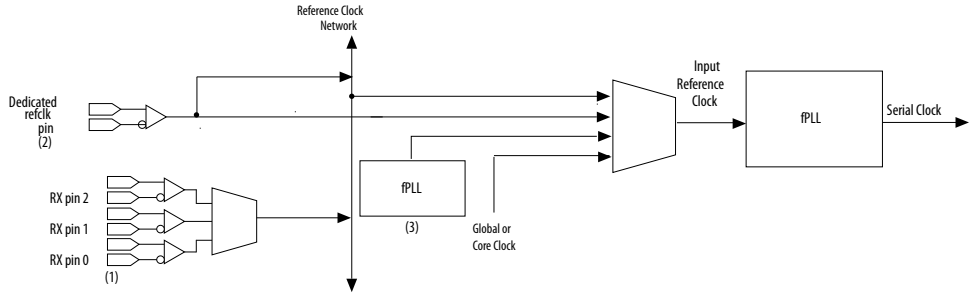
- Dedicated reference clock pins
- Reference clock network
- The output of another fPLL with PLL cascading
- Receiver input pins
- Global clock or core clock [28]

For the best jitter performance, Intel recommends placing the reference clock as close as possible to the transmit PLL. For protocol jitter compliance at a data rate > 10 Gbps, place the reference clock pin in the same triplet as the transmit PLL. The following protocols require the reference clock to be placed in same bank as the transmit PLL:

- OC-192 and 10 GPON

Note: For optimum performance, the reference clock of transmit PLL is recommended to be from a dedicated reference clock pin in the same bank.

Figure 122. Input Reference Clock Sources



Note : (1) You can choose only one of the three RX pins to be used as an input reference clock source. Any RX pin on the same side of the device can be used as an input reference clock.

(2) Dedicated refclk pin can be used as an input reference clock source only for ATX or fPLL or to the reference clock network. Reference clock network can then drive the CMU PLL.

(3) The output of another PLL can be used as an input reference clock source during PLL cascading. Cyclone 10 GX transceivers support fPLL to fPLL cascading.

Note: To successfully complete the calibration process, the reference clocks driving the PLLs (ATX PLL, fPLL, CDR/CMU PLL) must be stable and free running at start of FPGA configuration. Otherwise, recalibration is necessary.

[28] Not available for CMU.

Related Information

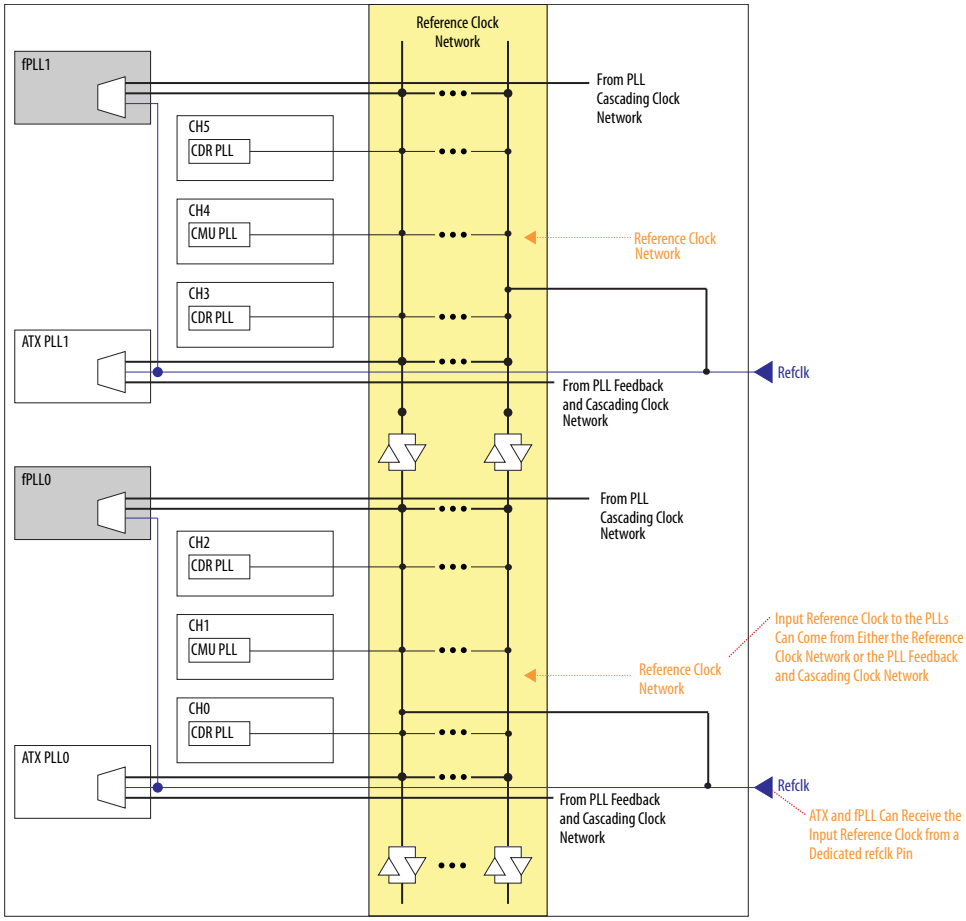
Calibration on page 373
For more information about the calibration process

3.2.1. Dedicated Reference Clock Pins

To minimize the jitter, the advanced transmit (ATX) PLL and the fractional PLL (fPLL) can source the input reference clock directly from the reference clock buffer without passing through the reference clock network. The input reference clock is also fed into the reference clock network.

Figure 123. Dedicated Reference Clock Pins

There are two dedicated reference clock (`refclk`) pins available in each transceiver bank. The bottom `refclk` pin feeds the bottom ATX PLL and fPLL. The top `refclk` pin feeds the top ATX PLL and fPLL. The dedicated reference clock pins including the 4 channels bank can also drive the reference clock network.



3.2.2. Receiver Input Pins

Receiver input pins can be used as an input reference clock source to transceiver PLLs. However, they cannot be used to drive core fabric.

The receiver input pin drives the feedback and cascading clock network, which can then feed any number of transmitter PLLs on the same side of the device. When a receiver input pin is used as an input reference clock source, the clock data recovery (CDR) block of that channel is not available. As indicated in Figure 122 on page 208, only one RX differential pin pair per three channels can be used as an input reference clock source at any given time.

3.2.3. PLL Cascading as an Input Reference Clock Source

In this mode, the output of one PLL drives the reference clock input of another PLL. PLL cascading can generate frequency outputs not normally possible with a single PLL solution. In PLL cascading, PLL outputs are connected to the feedback and cascading clock network. The transceiver in Cyclone 10 GX devices support fPLL to fPLL cascading, with only maximum two fPLLs allowed in the cascading chain.

Note:

- To successfully complete the calibration process, the reference clocks driving the PLLs (ATX PLL, fPLL, CDR/CMU PLL) must be stable and free running at start of FPGA configuration. Otherwise, recalibration is necessary.
- When the fPLL is used as a cascaded fPLL (downstream fPLL), a user recalibration on the fPLL is required. Refer to "User Recalibration" section in "Calibration" chapter for more information.

Related Information

Calibration on page 373
For more information about the calibration process

3.2.4. Reference Clock Network

The reference clock network distributes a reference clock source to either the entire left or right side of the FPGA where the transceivers reside. This allows any reference clock pin to drive any transmitter PLL on the same side of the device. Designs using multiple transmitter PLLs which require the same reference clock frequency and are located along the same side of the device, can share the same dedicated reference clock (`refclk`) pin.

3.2.5. Global Clock or Core Clock as an Input Reference Clock

The global clock or the core clock can be used as an input reference clock for ATX PLL and fPLL.

The global or core clock network routes the clock directly to the PLL. In this case the PLL reference clock network is not used. For best performance, use the dedicated reference clock pins or the reference clock network.

3.3. Transmitter Clock Network

The transmitter clock network routes the clock from the transmitter PLL to the transmitter channel. It provides two types of clocks to the transmitter channel:

- High Speed Serial clock—high-speed clock for the serializer.
- Low Speed Parallel clock—low-speed clock for the serializer and the PCS.

In a bonded channel configuration, both the serial clock and the parallel clock are routed from the transmitter PLL to the transmitter channel. In a non-bonded channel configuration, only the serial clock is routed to the transmitter channel, and the parallel clock is generated locally within the channel. To support various bonded and non-bonded clocking configurations, four types of transmitter clock network lines are available:

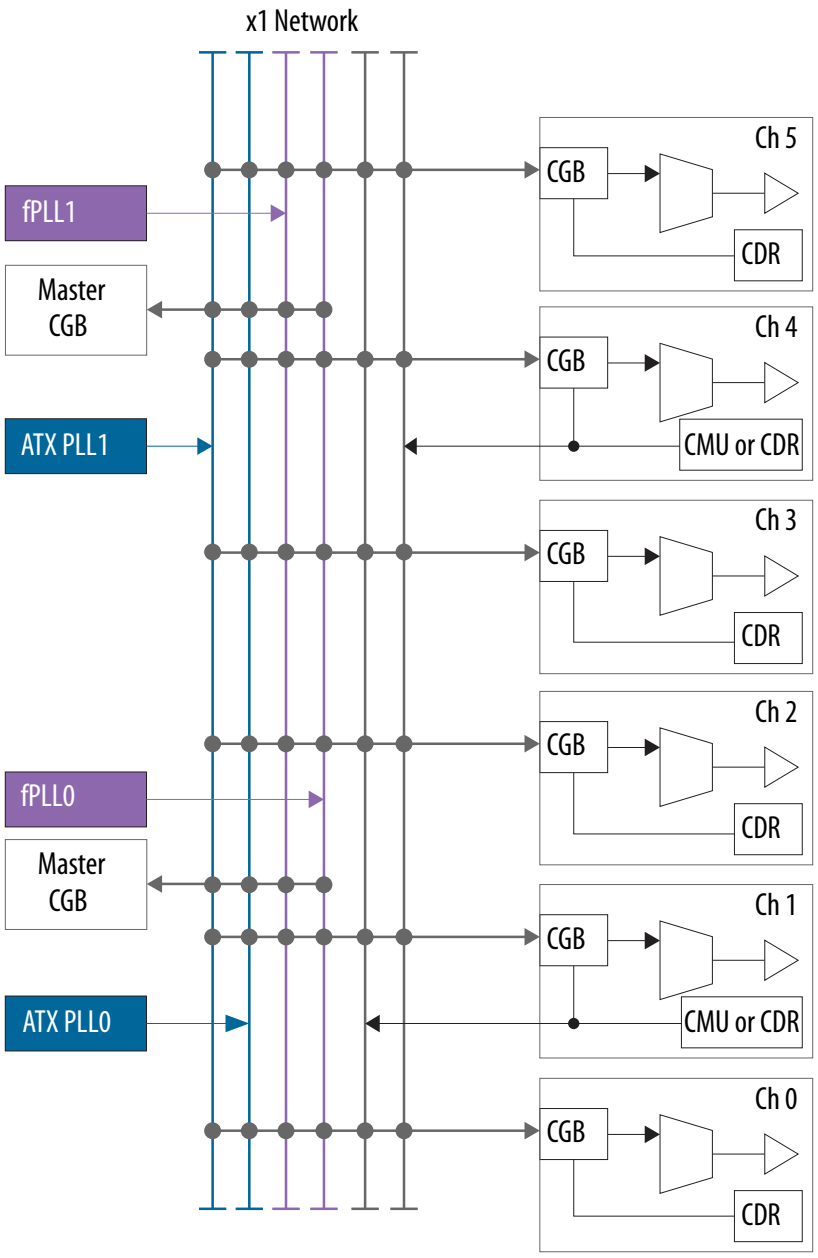
- x1 clock lines
- x6 clock lines
- xN clock lines

3.3.1. x1 Clock Lines

The x1 clock lines route the high speed serial clock output of a PLL to any channel within a transceiver bank. The low speed parallel clock is then generated by that particular channel's local clock generation block (CGB). Non-bonded channel configurations use the x1 clock network.

The x1 clock lines can be driven by the ATX PLL, fPLL, or by either one of the two channel PLLs (channel 1 and 4 when used as a CMU PLL) within a transceiver bank.

Figure 124. x1 Clock Lines



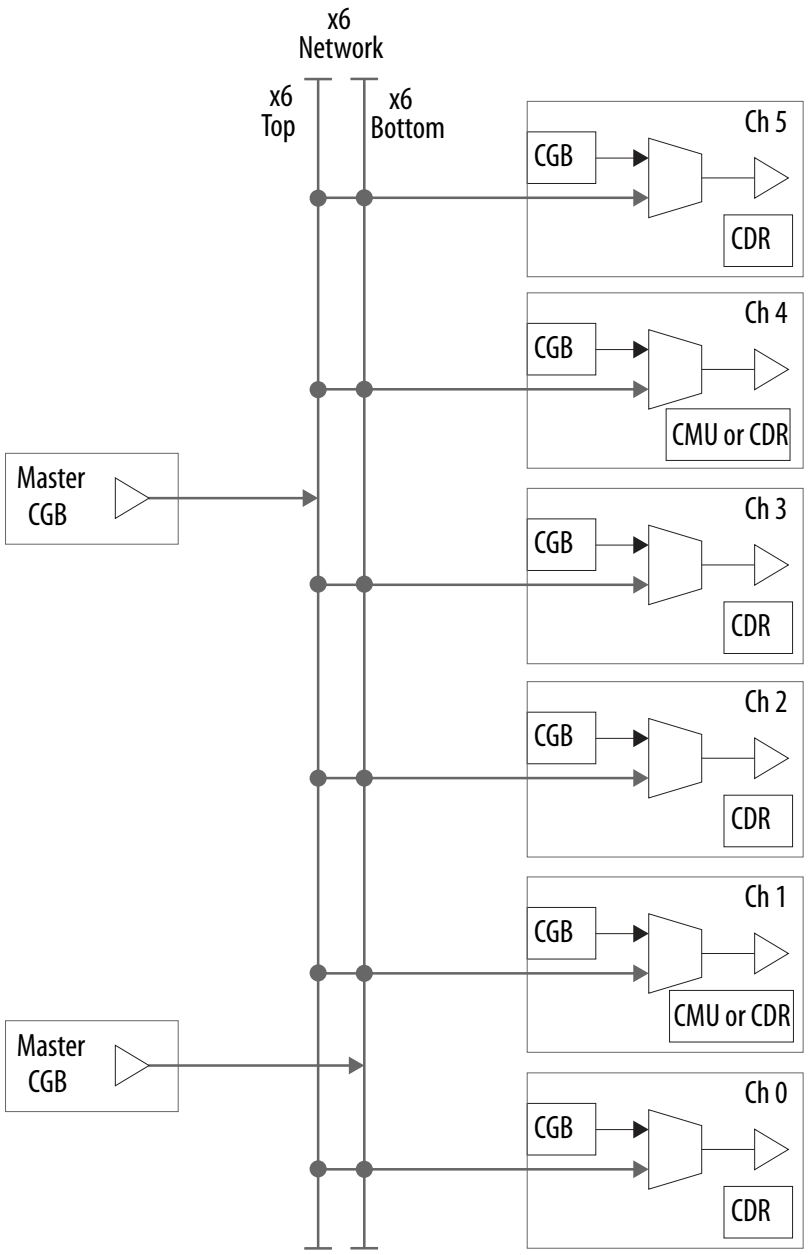
3.3.2. x6 Clock Lines

The x6 clock lines route the clock within a transceiver bank. The x6 clock lines are driven by the master CGB. The master CGB can only be driven by the ATX PLL or the fPLL. Because the CMU PLLs cannot drive the master CGB, the CMU PLLs cannot be used for bonding purposes. There are two x6 clock lines per transceiver bank, one for each master CGB. Any channel within a transceiver bank can be driven by the x6 clock lines.

For bonded configuration mode, the low speed parallel clock output of the master CGB is used and the local CGB within each channel is bypassed. For non-bonded configurations, the master CGB also provides a high speed serial clock output to each channel without bypassing the local CGB within each channel.

The x6 clock lines also drive the xN clock lines which route the clocks to the neighboring transceiver banks.

Figure 125. x6 Clock Lines

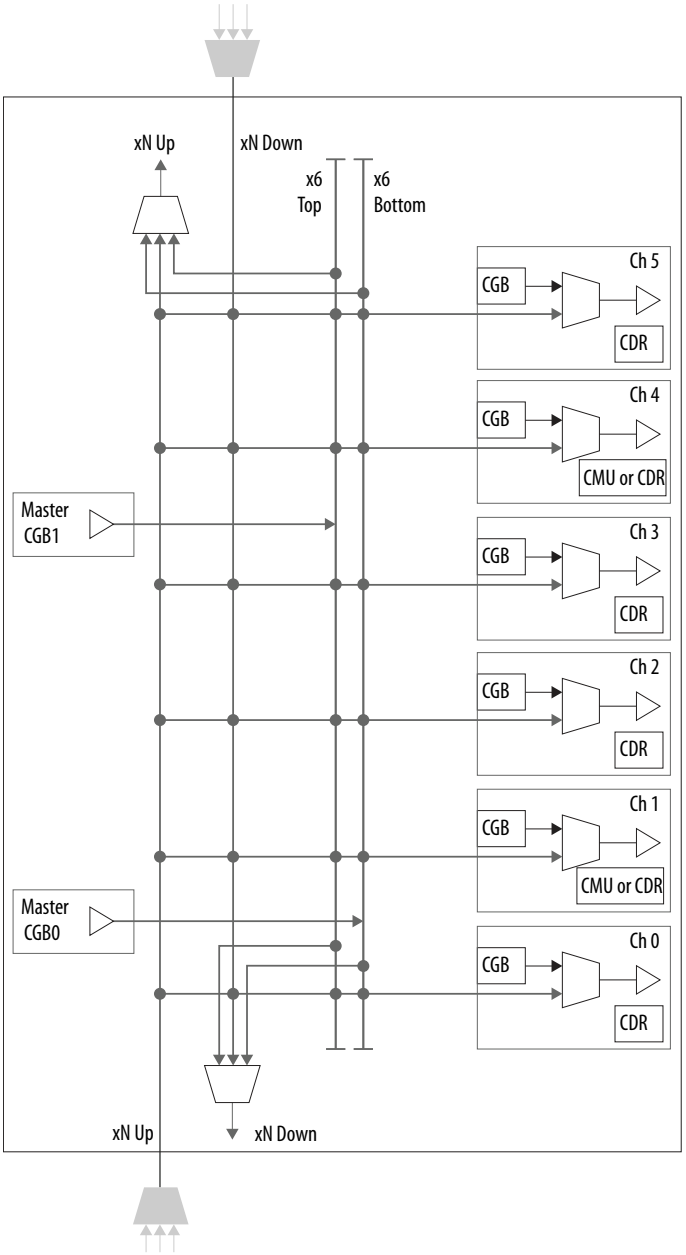


3.3.3. xN Clock Lines

The xN clock lines route the transceiver clocks across multiple transceiver banks.

The master CGB drives the x6 clock lines and the x6 clock lines drive the xN clock lines. There are two xN clock lines: xN Up and xN Down. xN Up clock lines route the clocks to transceiver banks located above the master CGB and xN Down clock lines route the clocks to transceiver banks located below the master CGB. The xN clock lines can be used in both bonded and non-bonded configurations. For bonded configurations, the low speed parallel clock output of the master CGB is used, and the local CGB within each channel is bypassed. For non-bonded configurations, the master CGB provides a high speed serial clock output to each channel.

Figure 126. xN Clock Network



Related Information

- Implementing x6/xN Bonding Mode on page 236
- x6/xN Bonding on page 222
- Cyclone 10 GX Data Sheet.

3.4. Clock Generation Block

In Cyclone 10 GX devices, there are two types of clock generation blocks (CGBs)

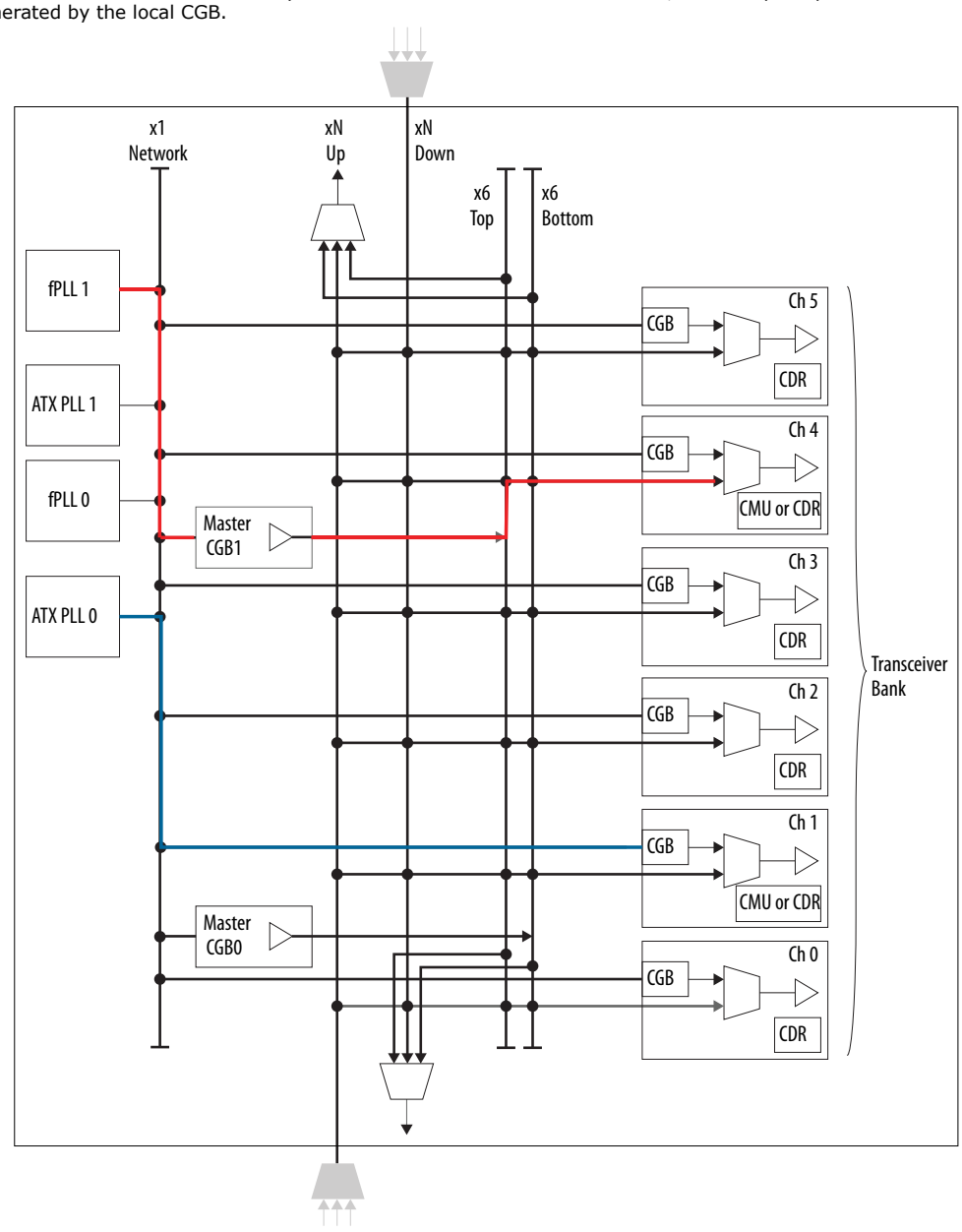
- Local clock generation block (local CGB)
- Master clock generation block (master CGB)

Each transmitter channel has a local clock generation block (CGB). For non-bonded channel configurations, the serial clock generated by the transmit PLL drives the local CGB of each channel. The local CGB generates the parallel clock used by the serializer and the PCS.

There are two standalone master CGBs within each transceiver bank. The master CGB provides the same functionality as the local CGB within each transceiver channel. The output of the master CGB can be routed to other channels within a transceiver bank using the x6 clock lines. The output of the master CGB can also be routed to channels in other transceiver banks using the xN clock lines. Each transmitter channel has a multiplexer to select its clock source from either the local CGB or the master CGB.

Figure 127. Clock Generation Block and Clock Network

The local clock for each transceiver channel can be sourced from either the local CGB via the x1 network, or the master CGB via the x6/xN network. For example, as shown by the red highlighted path, the fPLL 1 drives the x1 network which in turn drives the master CGB. The master CGB then drives the x6 clock network which routes the clocks to the local channels. As shown by the blue highlighted path, the ATX PLL 0 can also drive the x1 clock network which can directly feed a channel's local CGB. In this case, the low speed parallel clock is generated by the local CGB.



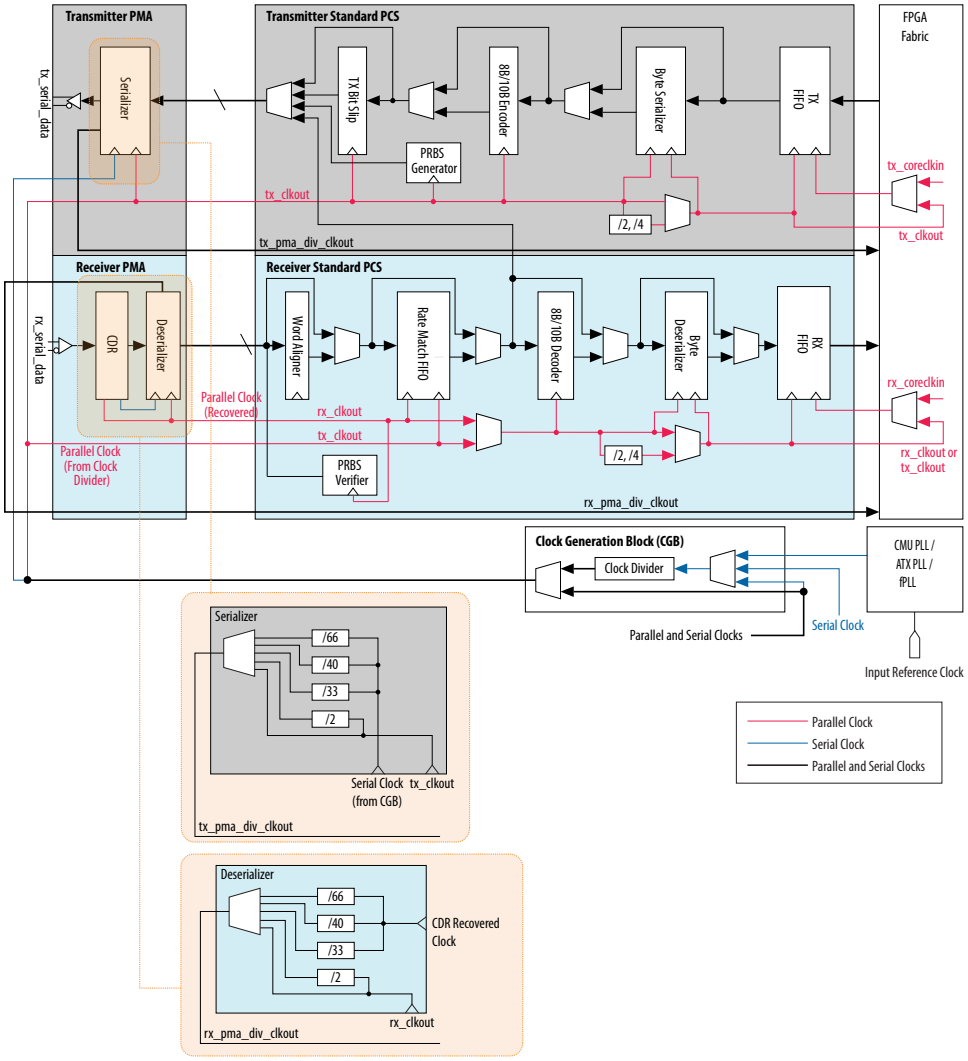
3.5. FPGA Fabric-Transceiver Interface Clocking

The FPGA fabric-transceiver interface consists of clock signals from the FPGA fabric into the transceiver and clock signals from the transceiver into the FPGA fabric. These clock signals use the global (GCLK), regional (RCLK), and periphery (PCLK) clock

networks in the FPGA core. If the Global Signal is set to Off, it does not choose any of the previously mentioned clock networks. Instead, it chooses directly from the local routing between transceiver and FPGA fabric.

The transmitter channel forwards a parallel output clock `tx_clkout` to the FPGA fabric to clock the transmitter data and control signals. The receiver channel forwards a parallel output clock `rx_clkout` to the FPGA fabric to clock the data and status signals from the receiver into the FPGA fabric. Based on the receiver channel configuration, the parallel output clock is recovered from either the receiver serial data or the `rx_clkout` clock (in configurations without the rate matcher) or the `tx_clkout` clock (in configurations with the rate matcher).

Figure 128. FPGA Fabric - Transceiver Interface Clocking



The divided versions of the `tx_clkout` and `rx_clkout` are available as `tx_pma_div_clkout` and `rx_pma_div_clkout`, respectively.

The output frequency of `tx_pma_div_clkout` and `rx_pma_div_clkout` can be one of the following:

- A divided down version of the `tx_clkout` or `rx_clkout` respectively, where divide by 1 and divide by 2 ratios are available.
- A divided down version of the serializer clock where divide by 33, 40, and 66 ratios are available.

These clocks can be used to meet core timing by operating the TX and RX FIFO in double-width mode, as this halves the required clock frequency at the PCS to/from FPGA interface. These clocks can also be used to clock the core side of the TX and RX FIFOs when the Enhanced PCS Gearbox is used.

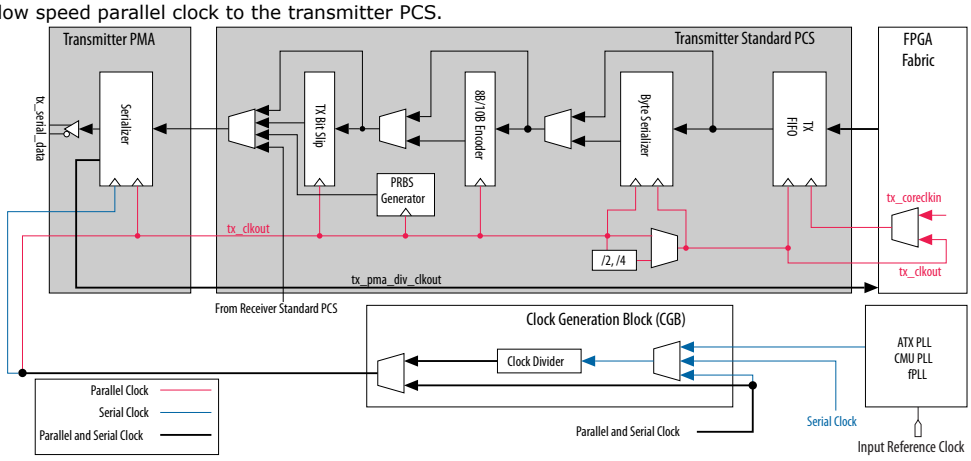
For example, if you use the Enhanced PCS Gearbox with a 66:40 ratio, then you can use `tx_pma_div_clkout` with a divide-by-33 ratio to clock the write side of the TX FIFO, instead of using a PLL to generate the required clock frequency, or using an external clock source.

3.6. Transmitter Data Path Interface Clocking

The clocks generated by the PLLs are used to clock the channel PMA and PCS blocks. The clocking architecture is different for the standard PCS and the enhanced PCS.

Figure 129. Transmitter Standard PCS and PMA Clocking

The master or the local CGB provides the high speed serial clock to the serializer of the transmitter PMA, and the low speed parallel clock to the transmitter PCS.



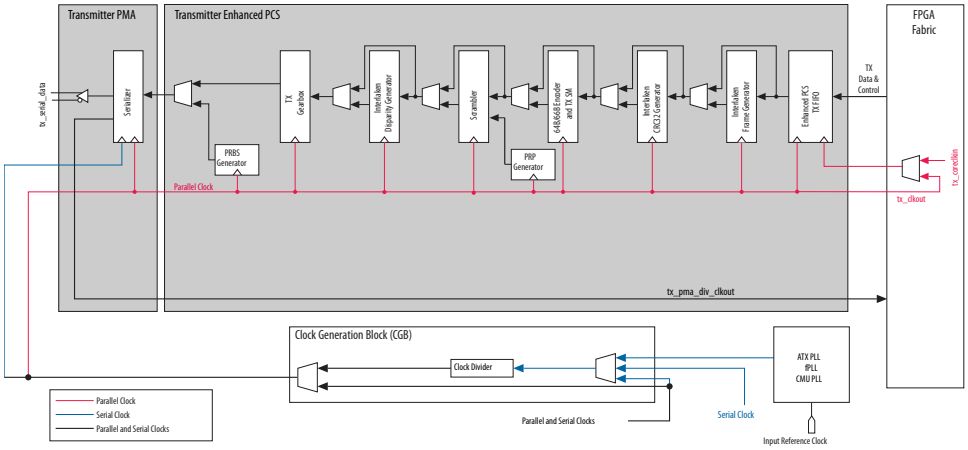
In the Standard PCS, for configurations that do not use the byte serializer, the parallel clock is used by all the blocks up to the read side of the TX phase compensation FIFO. For configurations that use the byte serializer block, the clock divided by 2 or 4 is used by the byte serializer and the read side of the TX phase compensation FIFO. The clock used to clock the read side of the TX phase compensation FIFO is also forwarded to the FPGA fabric to provide an interface between the FPGA fabric and the transceiver.

If the `tx_clkout` that is forwarded to the FPGA fabric is used to clock the write side of the phase compensation FIFO, then both sides of the FIFO have 0 ppm frequency difference because it is the same clock that is used.

If you use a different clock than the `tx_clkout` to clock the write side of the phase compensation FIFO, then you must ensure that the clock provided has a 0 ppm frequency difference with respect to the `tx_clkout`.

Figure 130. Transmitter Enhanced PCS and PMA Clocking

The master or local CGB provides the serial clock to the serializer of the transmitter PMA, and the parallel clock to the transmitter PCS.



In the Enhanced PCS, the parallel clock is used by all the blocks up to the read side of the TX phase compensation FIFO. The clocks of all channels in bonded configuration are forwarded. You can pick `tx_clkout[0]` as the source for clocking their TX logic in core.

For the enhanced PCS, the transmitter PCS forwards the following clocks to the FPGA fabric:

`tx_clkout` for each transmitter channel in non-bonded and bonded configuration. In bonded configuration, any `tx_clkout` can be used depending on your core timing requirements.

You can clock the transmitter datapath interface using one of the following methods:

- Quartus Prime selected transmitter datapath interface clock
- User-selected transmitter datapath interface clock

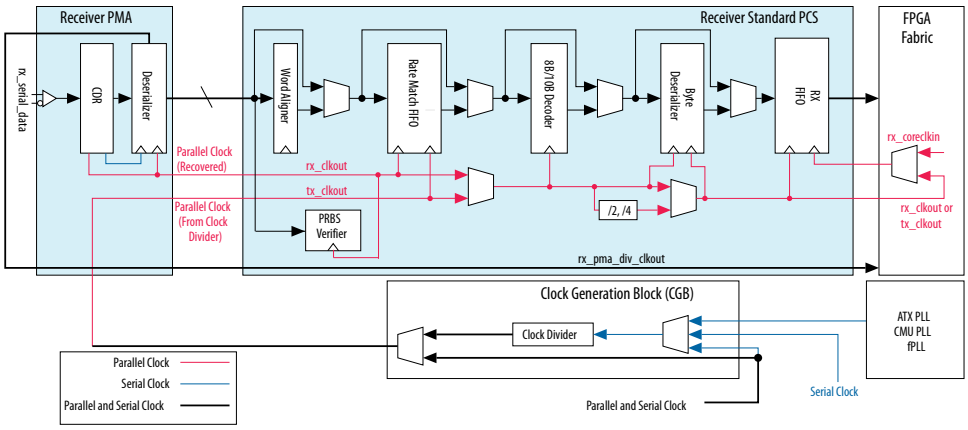
3.7. Receiver Data Path Interface Clocking

The CDR block present in the PMA of each channel recovers the serial clock from the incoming data. The CDR block also divides the recovered serial clock to generate the recovered parallel clock. Both the recovered serial and the recovered parallel clocks are used by the deserializer. The receiver PCS can use the following clocks based on the configuration of the receiver channel:

- Recovered parallel clock from the CDR in the PMA
- Parallel clock from the clock divider used by the transmitter PCS (if enabled) for that channel

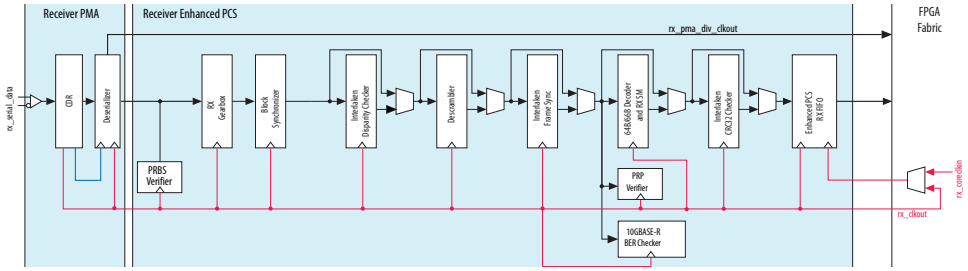
For configurations that use the byte deserializer block, the clock divided by 2 or 4 is used by the byte deserializer and the write side of the RX phase compensation FIFO.

Figure 131. Receiver Standard PCS and PMA Clocking



All configurations that use the standard PCS channel must have a 0 ppm phase difference between the receiver datapath interface clock and the read side clock of the RX phase compensation FIFO.

Figure 132. Receiver Enhanced PCS and PMA Clocking



The receiver PCS forwards the following clocks to the FPGA fabric:

- `rx_clkout` — for each receiver channel when the rate matcher is not used.
- `tx_clkout` — for each receiver channel when the rate matcher is used.

You can clock the receiver datapath interface using one of the following methods:

- Quartus Prime selected receiver datapath interface clock
- User-selected receiver datapath interface clock

Related Information

Unused or Idle Clock Line Requirements on page 221
For more information about unused or idle transceiver clock lines in design.

3.8. Unused/Idle Clock Line Requirements

Unused or idle transceiver clock lines can degrade if the devices are powered up to normal operating conditions and the devices are not configured. This issue also affects designs that configure transceiver channels to use the idle clock lines at a later date by using dynamic reconfiguration or a new device programming file. Clock lines affected are unused idle receiver (RX) serial clock lines. Active RX serial clock lines and non-transceiver circuits are not impacted by this issue.

In order to prevent the performance degradation, configure the devices as long as it is powered up to normal operations. For idle transceiver RX channels, compile designs with the assignment described in the link shown below. The CLKUSR pin must be assigned a 100-125 MHz clock. For used transceiver TX and RX channels, do not assert the analog reset signals indefinitely.

Related Information

Unused Transceiver channels Settings on page 402
For more information about unused or idle transceiver clock lines in the design. It describes the unused or idle transceiver clock lines assignments in the qsf file.

3.9. Channel Bonding

For Cyclone 10 GX devices, two types of bonding modes are available:

- PMA bonding
- PMA and PCS bonding

Related Information

Resetting Transceiver Channels on page 243

Refer to the *Timing Constraints for Bonded PCS and PMA Channels* section in the *Resetting Transceiver Channels* chapter for additional details.

3.9.1. PMA Bonding

PMA bonding reduces skew between PMA channels. In PMA bonding, only the PMA portion of the transceiver datapath is skew compensated and the PCS is not skew compensated.

In Cyclone 10 GX devices, there are two PMA bonding schemes:

- x6/xN bonding
- PLL feedback compensation bonding

In either case, the channels in the bonded group need not be placed contiguously.

3.9.1.1. x6/xN Bonding

In x6/xN bonding mode, a single transmit PLL is used to drive multiple channels.

The steps below explain the x6/xN bonding process:

1. The ATX PLL or the fPLL generates a high speed serial clock.
2. The PLL drives the high speed serial clock to the master CGB via the x1 clock network.
3. The master CGB drives the high speed serial and the low speed parallel clock into the x6 clock network.
4. The x6 clock network feeds the TX clock multiplexer for the transceiver channels within the same transceiver bank. The local CGB in each transceiver channel is bypassed.
5. To drive the channels in adjacent transceiver banks, the x6 clock network drives the xN clock network. The xN clock network feeds the TX clock mutiplexer for the transceiver channels in these adjacent transceiver banks.

Related Information

- xN Clock Lines on page 214
- Cyclone 10 GX Device Datasheet

3.9.1.2. PLL Feedback Compensation Bonding

In PLL feedback compensation bonding, channels are divided into bonded groups based on physical location with a four-channel or six-channel transceiver bank. All channels within the same six-channel transceiver bank are assigned to the same bonded group.

In PLL feedback compensation bonding, each bonded group is driven by its own set of high-speed serial and low-speed parallel clocks. Each bonded group has its own PLL and master CGB. To maintain the same phase relationship, the PLL and master CGB for different groups share the same reference clocks.

The steps below explain the PLL feedback compensation bonding process:

1. The same input reference clock drives the local PLL in each three-channel or six-channel transceiver bank.
2. The local PLL for the bonding group drives the master CGB.
3. The master CGB feeds the x6 clock lines. The master CGB drives the transceiver channels in the bonding group via the x6 clock network.
4. The parallel output of the master CGB is the feedback input to the PLL.
5. In this mode, all channels are phase aligned to the same input reference clock.

PLL Feedback Compensation Bonding Advantages over x6/xN Bonding Mode

- There is no data rate restriction. The x6 clock network used for PLL feedback compensation bonding can run up to the maximum data rate of the device used.

PLL Feedback Compensation Bonding Disadvantages over x6/xN Bonding Mode

- It uses more resources compared to x6/xN bonding. One PLL and one master CGB are used per transceiver bank. This causes higher power consumption compared to x6/xN bonding.
- The skew is higher compared to x6/xN bonding. The reference clock skew between each transceiver bank is higher than the skew contributed by the xN clock network in x6/xN bonding.
- Because the feedback clock for the PLL comes from the master CGB and not from the PLL, the PLL feedback compensation bonding mode has a reference clock limitation. The PLL's N-counter (reference clock divider) is bypassed resulting in only one valid reference clock frequency for a given data rate.
- Feedback compensation bonding only supports integer mode.

Note: In order to minimize the reference clock skew for PLL feedback compensation bonding, use a reference clock input near the center of the bonded group.

x6/xN Bonding Advantages over PLL Feedback Compensation Bonding

- x6/xN uses less resources compared to PLL feedback compensation bonding. Only one PLL and one master CGB are required to drive all channels in the bonded group.
- x6/xN has lower skew compared to PLL feedback compensation bonding.

Related Information

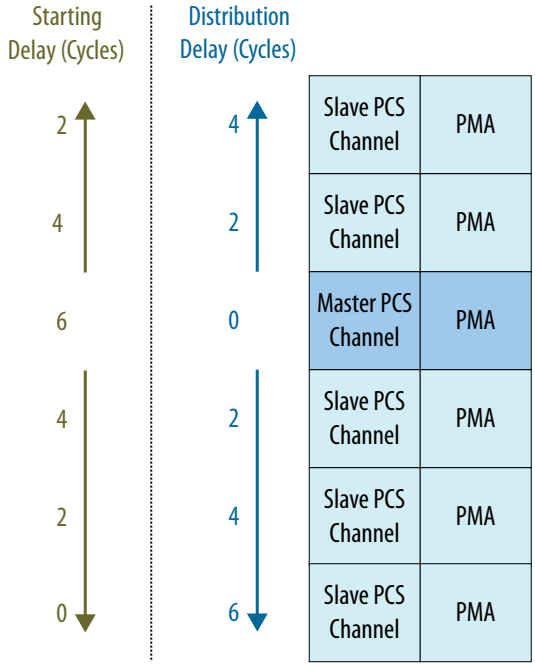
Implementing PLL Feedback Compensation Bonding Mode on page 237

3.9.2. PMA and PCS Bonding

PMA and PCS bonding reduces skew between both the PMA and PCS outputs within a group of channels.

For PMA bonding, either x6/xN or PLL feedback compensation bonding is used. For PCS bonding, some of the PCS control signals within the bonded group are skew aligned using dedicated hardware inside the PCS.

Figure 133. PMA and PCS Bonding



For PMA and PCS bonding, the concept of master and slave channels is used. One PCS channel in the bonded group is selected as the master channel and all others are slave channels. To ensure that all channels start transmitting data at the same time and in the same state, the master channel generates a start condition. This condition is transmitted to all slave channels. The signal distribution of this start condition incurs a two parallel clock cycle delay. Because this signal travels sequentially through each PCS channel, this delay is added per channel. The start condition used by each slave channel is delay compensated based on the slave channel's distance from the master channel. This results in all channels starting on the same clock cycle.

The transceiver PHY IP automatically selects the center channel to be the master PCS channel. This minimizes the total starting delay for the bonded group. For PLL feedback compensation bonding up to all channels on one side can be bonded if the master PCS channel is placed in the center of the bonded group.

Note: Because the PMA and PCS bonding signals travel through each PCS block, the PMA and PCS bonded groups must be contiguously placed. The channel order needs to be maintained when doing the pin assignments to the dedicated RX serial inputs and TX serial outputs (for example: PIN_BC7 and PIN_BC8 for GXBR4D_TX_CH0p and GXBR4D_TX_CH0n TX serial outputs). Channels need to be placed in an ascending order from bottom to top. Swapping of channels, when doing pin assignments, leads to errors.

3.9.3. Selecting Channel Bonding Schemes

In Cyclone 10 GX devices, select PMA and PCS bonding for bonded protocols that are explicitly supported by the hard PCS blocks. For example, PCI-Express and SFI-S.

Select PMA-only bonding when a bonded protocol is not explicitly supported by the hard PCS blocks. For example, for Interlaken protocol, PMA-only bonding is used and a soft PCS bonding IP is implemented in the FPGA fabric.

3.9.4. Skew Calculations

To calculate the maximum skew between the channels, the following parameters are used:

- PMA to PCS datapath interface width (S)
- Maximum difference in number of parallel clock cycles between deassertion of each channel's FIFO reset (N)

To calculate the channel skew, the following scenarios are considered:

- Non-bonded

  In this case, both the PMA and PCS are non-bonded. Skew ranges from 0 UI to [(S-1) + N*S] UI.
- PMA bonding using x6 / xN clock network

  In this case, the PCS is non-bonded. Skew ranges from [0 to (N*S)] UI + x6/xN clock skew.
- PMA bonding using the PLL feedback compensation clock network

  In this case, the PCS is non-bonded. Skew ranges from [0 to (N*S)] UI + (reference clock skew) + (x6 clock skew).
- PMA and PCS bonding using the x6 / xN clock network

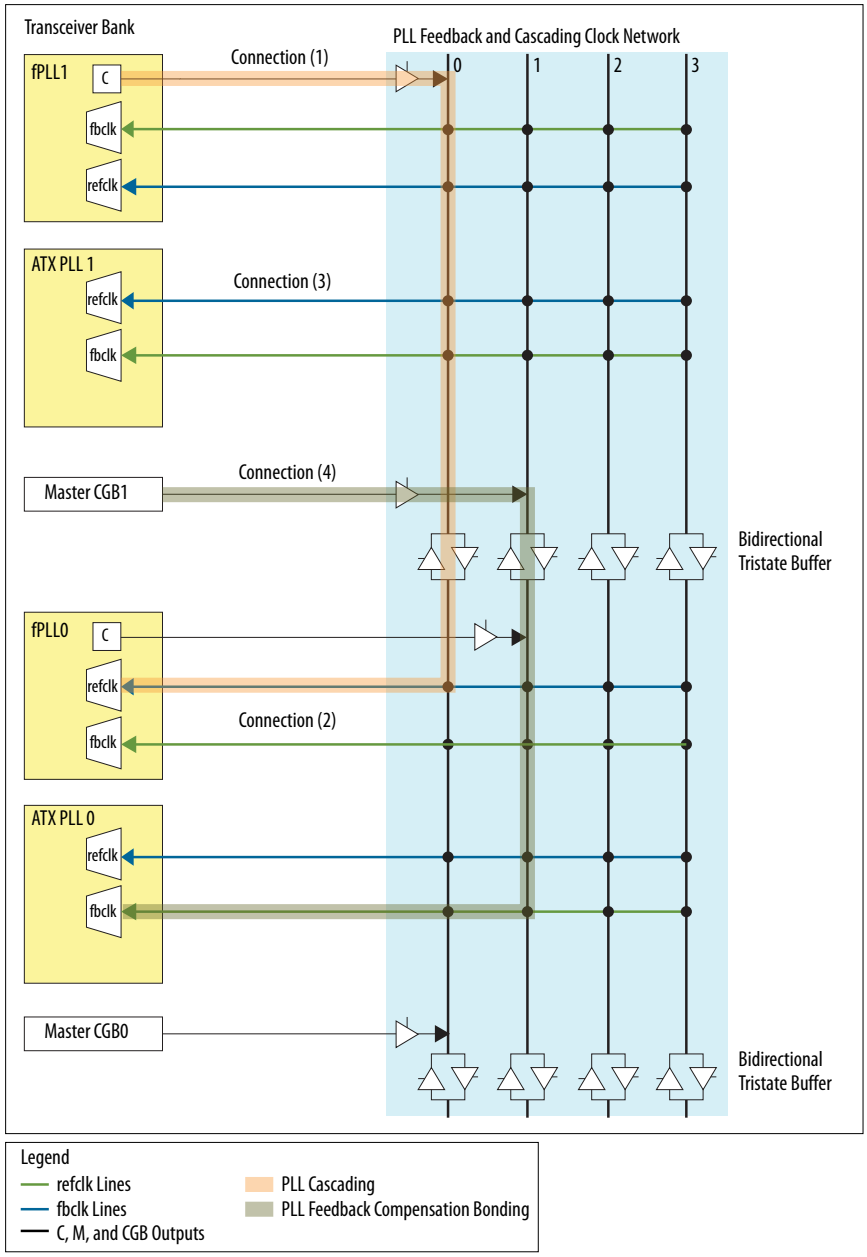
  Skew = x6 / xN clock skew.
- PMA and PCS bonding using PLL feedback compensation clock network

  Skew = (reference clock skew) + (x6 clock skew).

3.10. PLL Feedback and Cascading Clock Network

The PLL feedback and cascading clock network spans the entire side of the device, and is used for PLL feedback compensation bonding and PLL cascading.

Figure 134. PLL Feedback and Cascading Clock Network



To support PLL feedback compensation bonding and PLL cascading, the following connections are present:

1. The C counter output of the fPLL drives the **feedback and cascading clock** network.
2. The **feedback and cascading clock** network drives the **feedback clock** input of all PLLs.
3. The **feedback and cascading clock** network drives the **reference clock** input of all PLLs.
4. The **master CGB's parallel clock output** drives the **feedback and cascading clock** network.

For PLL cascading, connections (1) and (3) are used to connect the output of one PLL to the reference clock input of another PLL.

The transceivers in Cyclone 10 GX devices support fPLL to fPLL. Only a maximum of two PLLs are allowed in the cascading chain.

Note: When the fPLL is used as a cascaded fPLL (downstream fPLL), a user recalibration on the fPLL is required. Refer to "User Recalibration" section in "Calibration" chapter for more information.

For PLL feedback compensation bonding, connections (2) and (4) are used to connect the master CGB's parallel clock output to the PLL feedback clock input port.

PLL feedback compensation bonding can be used instead of xN bonding. The primary difference between PLL feedback compensation and xN bonding configurations, is for PLL feedback compensation, the bonded interface is broken down into smaller groups of 6 bonded channels within a transceiver bank. A PLL within each transceiver bank (ATX PLL or fPLL) is used as a transmit PLL. All the transmit PLLs share the same input reference clock.

In xN bonding configurations, one PLL is used for each bonded group. In PLL feedback compensation bonding, one PLL is used for each transceiver bank that the bonded group spans. There are no data rate limitations in PLL feedback compensation bonding, other than the natural data rate limitations of the transceiver channel and the PLL.

For feedback compensation bonding, the low-speed parallel clock must be the same frequency as the reference clock for the PLL.

fPLL Driving the Core

The fPLL can be used to drive the FPGA fabric. To ensure phase alignment between the input reference clock and the fPLL output clock, the fPLL needs to be configured in integer mode. Refer to the following figures when doing dynamic reconfiguration.

Figure 135. Fractional and not Phase Aligned

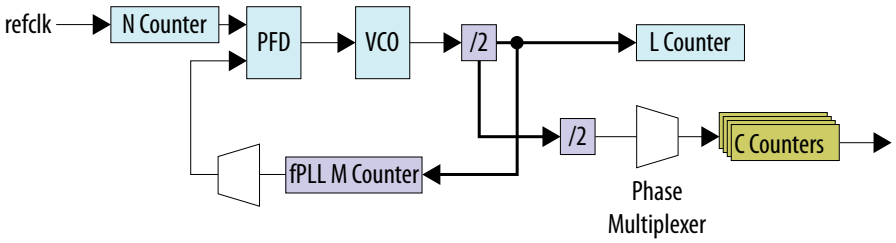


Figure 136. Integer and Phase Aligned

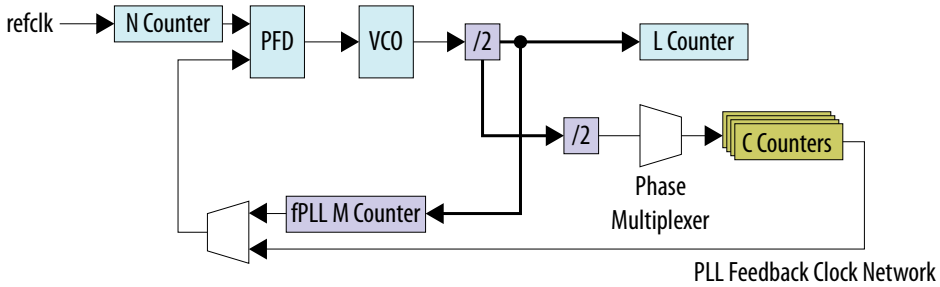
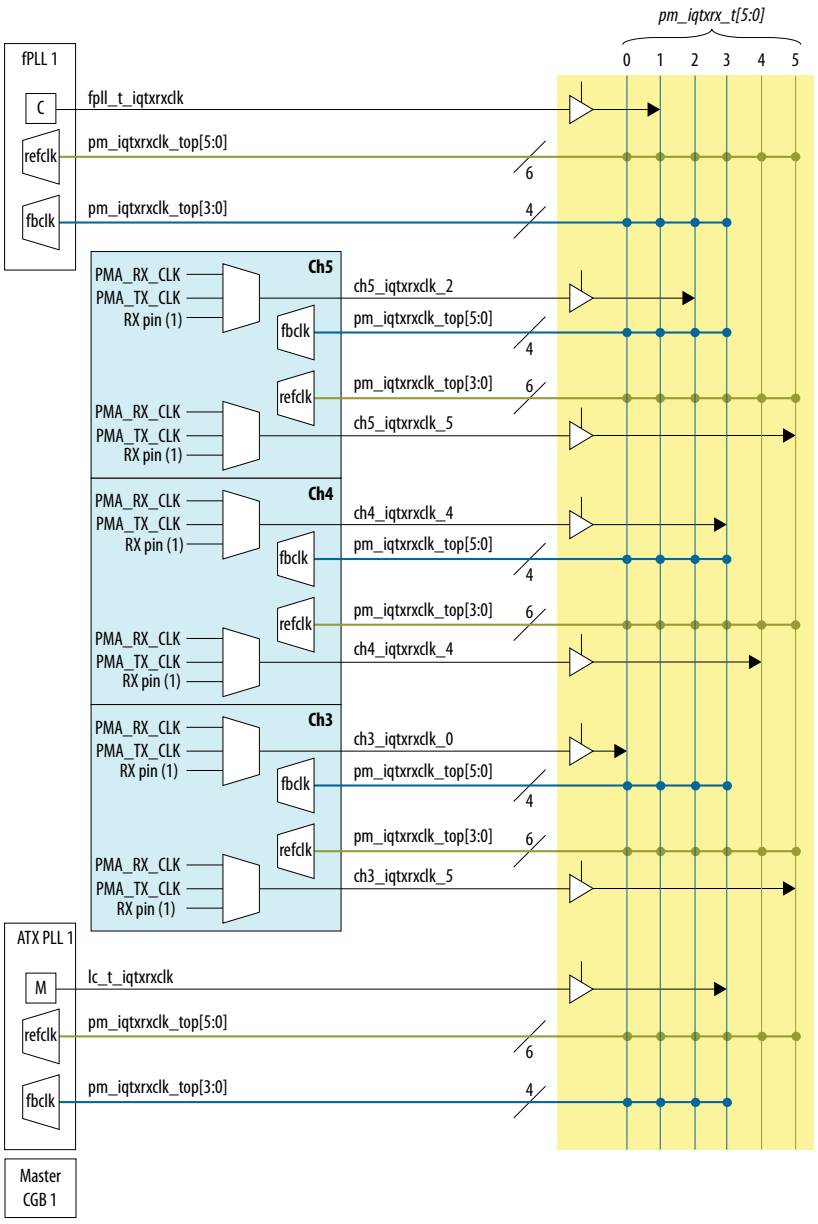


Figure 137. Integer Mode phase aligned and external feedback



Note: (1) RX pin used as reference clock

Related Information

- User Recalibration on page 383
- Implementing PLL Cascading on page 240

3.11. Using PLLs and Clock Networks

In Cyclone 10 GX devices, PLLs are not integrated in the Native PHY IP core. You must instantiate the PLL IP cores separately. Unlike in previous device families, PLL merging is no longer performed by the Quartus Prime software. This gives you more control, transparency, and flexibility in the design process. You can specify the channel configuration and PLL usage.

Related Information

PLLs and Clock Networks on page 198

3.11.1. Non-bonded Configurations

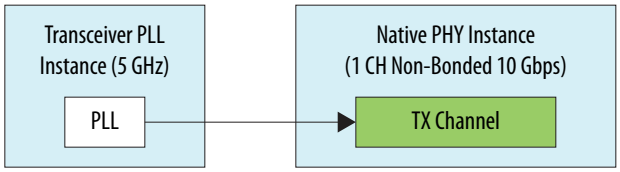
In a non-bonded configuration, only the high speed serial clock is routed from the transmitter PLL to the transmitter channel. The low speed parallel clock is generated by the local clock generation block (CGB) present in the transceiver channel. For non-bonded configurations, because the channels are not related to each other and the feedback path is local to the PLL, the skew between channels cannot be calculated. Also, the skew introduced by the clock network is not compensated.

3.11.1.1. Implementing Single Channel x1 Non-Bonded Configuration

In x1 non-bonded configuration, the PLL source is local to the transceiver bank and the x1 clock network is used to distribute the clock from the PLL to the transmitter channel.

For a single channel design, a PLL is used to provide the clock to a transceiver channel.

Figure 138. PHY IP Core and PLL IP Core Connection for Single Channel x1 Non-Bonded Configuration Example



To implement this configuration, instantiate a PLL IP core and a PHY IP core and connect them together as shown in the above figure.

Steps to implement a Single Channel x1 Non-Bonded Configuration

1. Instantiate the PLL IP core (ATX PLL, fPLL, or CMU PLL) you want to use in your design.
2. Configure the PLL IP core using the **IP Parameter Editor**.
   - For ATX PLL IP core, do not include the Master CGB.
   - For fPLL IP core, set the PLL feedback operation mode to **direct.**
   - For CMU PLL IP core, specify the reference clock and the data rate. No special configuration rule is required.
3. Configure the Native PHY IP core using the **IP Parameter Editor** .

- Set the **Native PHY IP Core TX Channel bonding mode** to **Non Bonded** .

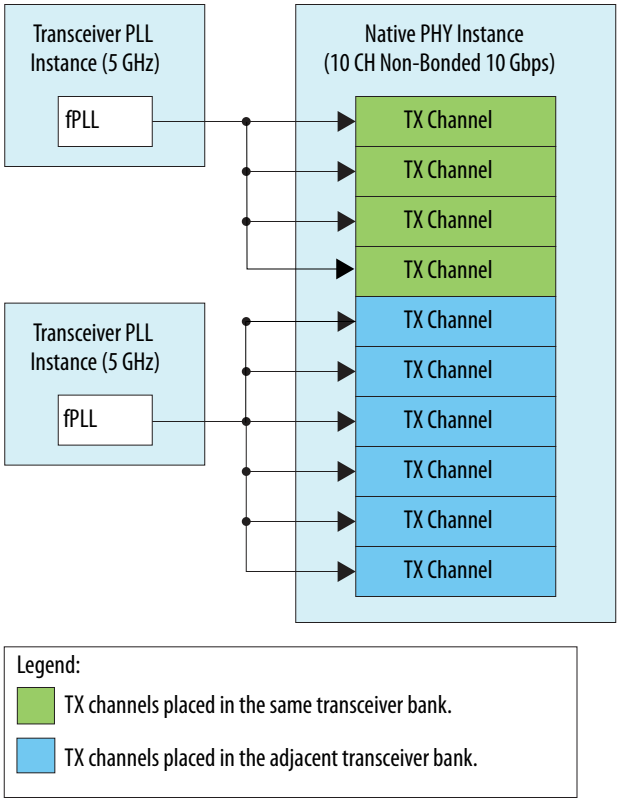
4. Connect the PLL IP core to the Native PHY IP core. Connect the `tx_serial_clk` output port of the PLL to IP to the corresponding `tx_serial_clk0` input port of the Native PHY IP core. This port represents the input to the local CGB of the channel. The `tx_serial_clk` for the PLL represents the high speed serial clock generated by the PLL.

3.11.1.2. Implementing Multi-Channel x1 Non-Bonded Configuration

This configuration is an extension of the x1 non-bonded case. In the following example, 10 channels are connected to two instances of the PLL IP core. Two PLL instances are required because PLLs using the x1 clock network can only span the 6 channels within the same transceiver bank. A second PLL instance is required to provide the clock to the remaining 4 channels.

Because 10 channels are not bonded and are unrelated, you can use a different PLL type for the second PLL instance. It is also possible to use more than two PLL IP cores and have different PLLs driving different channels. If some channels are running at different data rates, then you need different PLLs driving different channels.

Figure 139. PHY IP Core and PLL IP Core Connection for Multi-Channel x1 Non-Bonded Configuration



Steps to implement a Multi-Channel x1 Non-Bonded Configuration

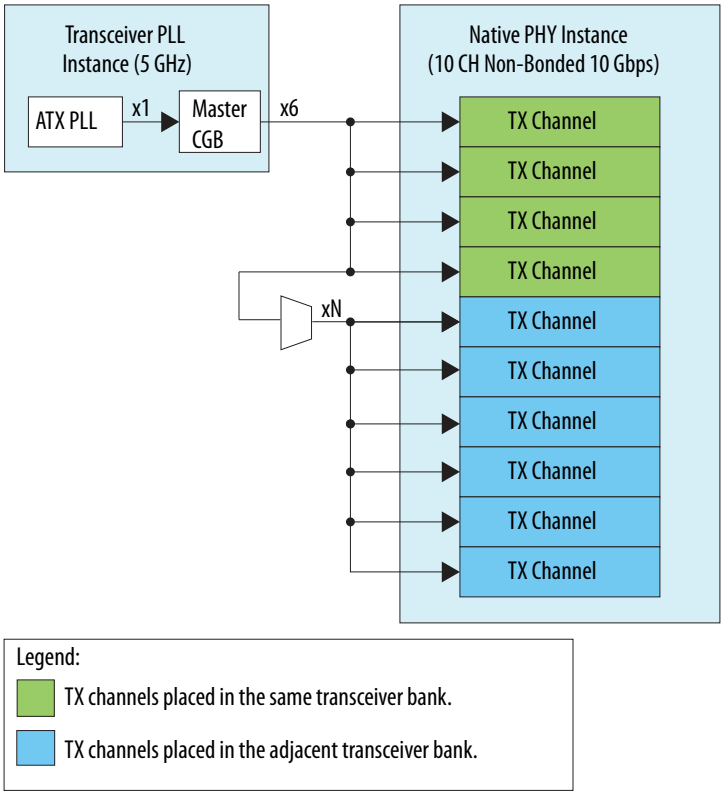
1. Choose the PLL IP core (ATX PLL, fPLL, or CMU PLL) you want to instantiate in your design and instantiate the PLL IP core.
2. Configure the PLL IP core using the **IP Parameter Editor**
   - For the ATX PLL IP core do not include the Master CGB. If your design uses the ATX PLL IP core and more than 6 channels, the x1 Non-Bonded Configuration is not a suitable option. Multi-channel xN Non-Bonded or Multi-Channel x1/xN Non-Bonded are the required configurations when using the ATX PLL IP core and more than 6 channels in the Native PHY IP core.
   - Refer to Figure 140 on page 234 Implementing Multi-Channel xN Non-Bonded Configuration section or the Figure 141 on page 235 Multi-Channel x1/xN Non-Bonded Example.
   - For the fPLL IP core, set the PLL feedback operation mode to **direct.**
   - For the CMU PLL IP core, specify the reference clock and the data rate. No special configuration rule is required.
3. Configure the Native PHY IP core using the **IP Parameter Editor**
   - Set the **Native PHY IP core TX Channel bonding mode** to **Non-Bonded**.
   - Set the number of channels as per your design requirement. In this example, the number of channels is set to 10.
4. Create a top level wrapper to connect the PLL IP core to the Native PHY IP core.
   - The `tx_serial_clk output` port of the PLL IP core represents the high speed serial clock.
   - The Native PHY IP core has 10 (for this example) `tx_serial_clk input` ports. Each port corresponds to the input of the local CGB of the transceiver channel.
   - As shown in the figure above, connect the first 6 `tx_serial_clk input` to the first transceiver PLL instance.
   - Connect the remaining 4 `tx_serial_clk input` to the second transceiver PLL instance.

3.11.1.3. Implementing Multi-Channel xN Non-Bonded Configuration

Using the xN non-bonded configuration reduces the number of PLL resources and the reference clock sources used.

Figure 140. PHY IP Core and PLL IP Core Connection for Multi-Channel xN Non-Bonded Configuration

In this example, the same PLL is used to drive 10 channels across two transceiver banks.



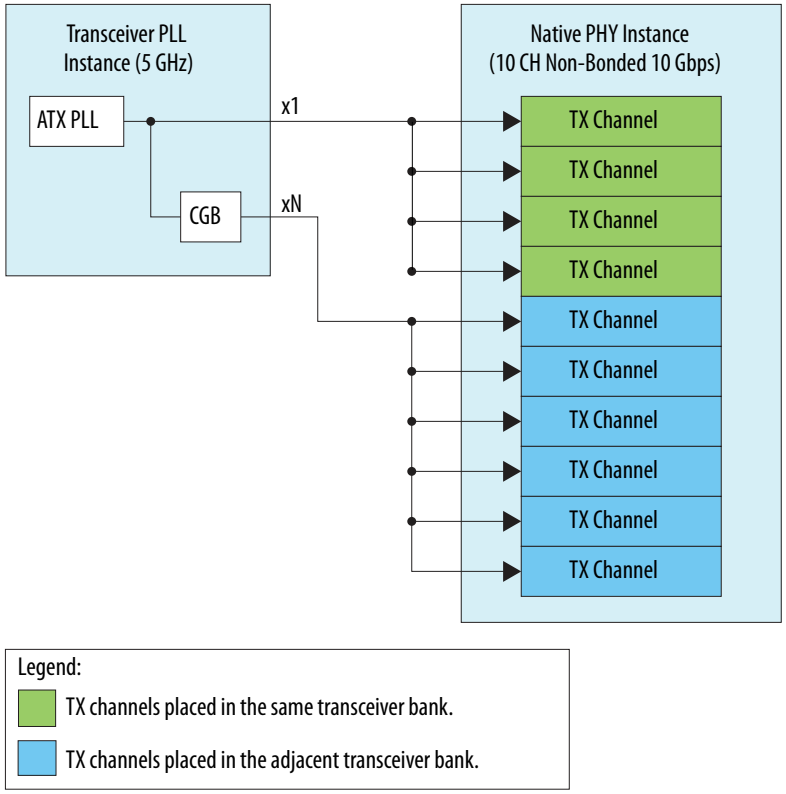
Steps to implement a multi-channel xN non-bonded configuration

1. You can use either the ATX PLL or fPLL for multi-channel xN non-bonded configuration.
   - Because the CMU PLL cannot drive the master CGB, only the ATX PLL or fPLL can be used for this example.
2. Configure the PLL IP core using the **IP Parameter Editor**. Enable **Include Master Clock Generation Block** .
3. Configure the Native PHY IP core using the **IP Parameter Editor**
   - Set the **Native PHY IP core TX Channel bonding mode** to **Non-Bonded** .
   - Set the number of channels as per your design requirement. In this example, the number of channels is set to 10.
4. Create a top level wrapper to connect the PLL IP core to the Native PHY IP core.

- In this case, the PLL IP core has `mcgb_serial_clk` output port. This represents the xN clock line.
- The Native PHY IP core has 10 (for this example) `tx_serial_clk input` ports. Each port corresponds to the input of the local CGB of the transceiver channel.
- As shown in the figure above, connect the `mcgb_serial_clk` output port of the PLL IP core to the 10 `tx_serial_clk input` ports of the Native PHY IP core.

Figure 141. Multi-Channel x1/xN Non-Bonded Example

The ATX PLL IP core has a `tx_serial_clk` output port. This port can optionally be used to clock the six channels within the same transceiver bank as the PLL. These channels are clocked by the x1 network. The remaining four channels outside the transceiver bank are clocked by the xN clock network.



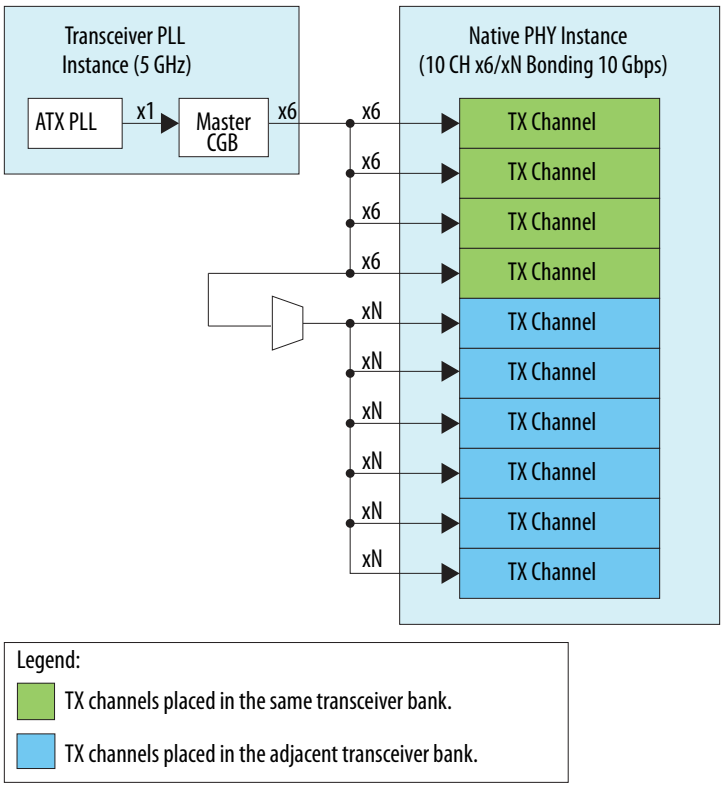
3.11.2. Bonded Configurations

In a bonded configuration, both the high speed serial and low speed parallel clocks are routed from the transmitter PLL to the transmitter channel. In this case, the local CGB in each channel is bypassed and the parallel clocks generated by the master CGB are used to clock the network.

In bonded configurations, the transceiver clock skew between the channels is minimized. Use bonded configurations for channel bonding to implement protocols such as PCIe*.

3.11.2.1. Implementing x6/xN Bonding Mode

Figure 142. PHY IP Core and PLL IP Core Connection for x6/xN Bonding Mode



Steps to implement a x6/xN bonded configuration

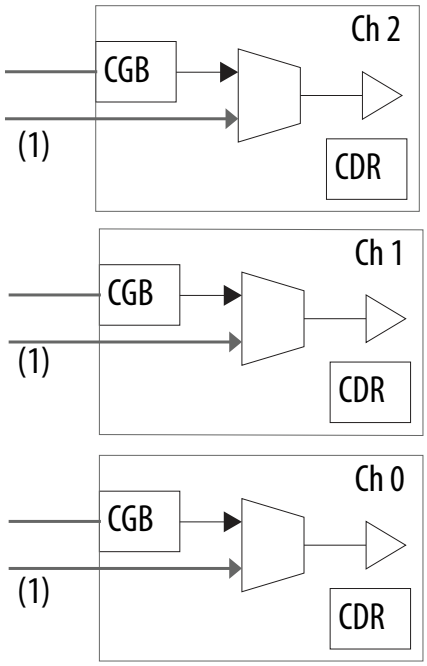
1. You can instantiate either the ATX PLL or the fPLL for x6/xN bonded configuration.
   - The CMU PLL cannot drive the Master CGB, only the ATX PLL or fPLL can be used for bonded configurations.
2. Configure the PLL IP core using the **IP Parameter Editor**. Enable **Include Master Clock Generation Block** and **Enable bonding** clock output ports.
3. Configure the Native PHY IP core using the **IP Parameter Editor** .
   - Set the **Native PHY IP core TX Channel bonding mode** to either **PMA bonding** or **PMA/PCS bonding** .
   - Set the number of channels required by your design. In this example, the number of channels is set to 10.
4. Create a top level wrapper to connect the PLL IP core to Native PHY IP core.

- In this case, the PLL IP core has `tx_bonding_clocks` output bus with width [5:0].
- The Native PHY IP core has `tx_bonding_clocks` input bus with width [5:0] multiplied by the number of transceiver channels (10 in this case). For 10 channels, the bus width is [59:0].

  Note: While connecting `tx_bonding_clocks`, leave `pll_ref_clk` open to avoid any Quartus Prime software fitter errors.
- Connect the PLL IP core to the PHY IP core by duplicating the output of the PLL[5:0] for the number of channels. For 10 channels, the Verilog syntax for the input port connection is `.tx_bonding_clocks ({10{tx_bonding_clocks_output}})` .

Note: Although the above diagram looks similar to the 10-channel non-bonded configuration example, the clock input ports on the transceiver channels bypass the local CGB in x6/xN bonding configuration. This internal connection is taken care of when the **Native PHY channel bonding mode** is set to **Bonded**.

Figure 143. x6/xN Bonding Mode —Internal Channel Connections



Note: (1) The local CGB is bypassed by the clock input ports in bonded mode.

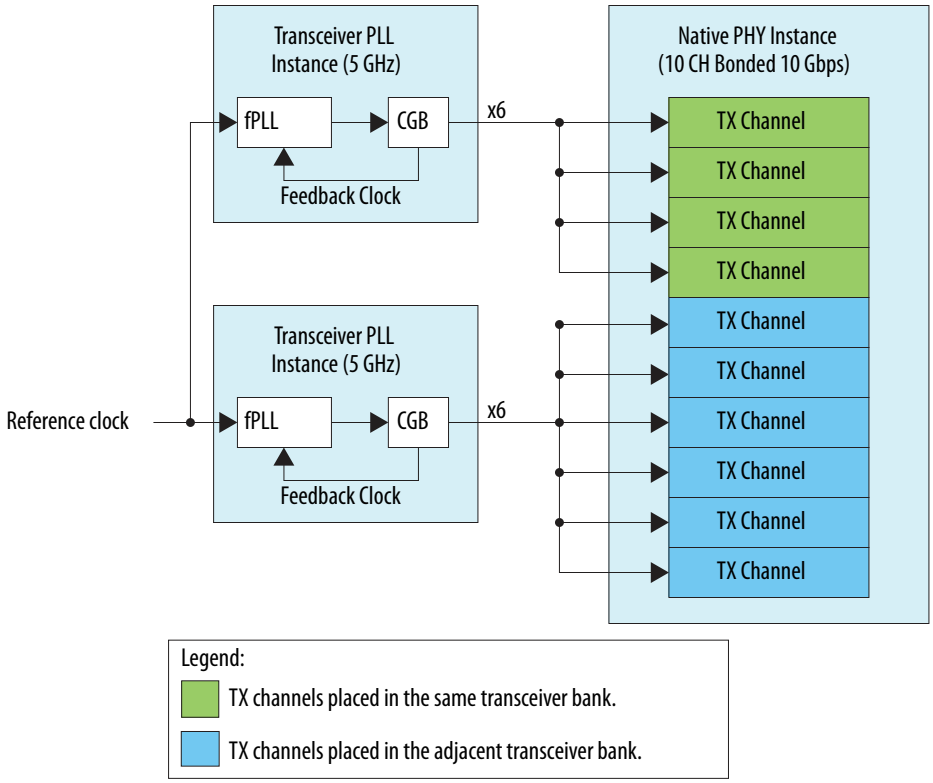
Related Information

xN Clock Lines on page 214
Information on xN Clock Network Span.

3.11.2.2. Implementing PLL Feedback Compensation Bonding Mode

In this bonding mode, the channel span limitations of xN bonding mode are removed. This is achieved by dividing all channels into multiple bonding groups.

Figure 144. PHY IP Core and PLL IP Core Connection for PLL Feedback Compensation Bonding



The data rate is limited by the x6 network speed limit. A disadvantage of using PLL feedback compensation bonding is that it consumes more PLL resources. Each transceiver bank consumes one PLL and one master CGB.

In PLL feedback compensation bonding mode, the N counter (reference clock divider) is bypassed in order to ensure that the reference clock skew is minimized between the PLLs in the bonded group. Because the N counter is bypassed, the PLL reference clock has a fixed value for any given data rate.

The PLL **IP Core Parameter Editor** window displays the required data rate in the **PLL reference clock frequency** drop down menu.

Steps to implement a PLL Feedback Compensation Bonding Configuration

1. Instantiate the PLL IP core (ATX PLL or fPLL) you want to use in your design. The CMU PLL cannot drive the master CGB, only the ATX PLL or fPLL can be used for feedback compensation bonding.
2. Configure the PLL IP core using the **IP Parameter Editor**.

- If you use the ATX PLL, set the following configuration settings:
  - Under the **Master Clock Generation Block Tab**
    - Enable **Include Master Clock Generation Block**.
    - Turn ON **Enable Bonding Clock output ports**.
    - Turn ON **Enable feedback compensation bonding**.
  - Under the **Dynamic Reconfiguration Tab**
    - Turn ON **Enable Dynamic Reconfiguration**.
- If you use the fPLL, set the following configuration settings:
  - Under the **PLL Tab**
    - Set the **PLL Feedback type** to **feedback compensation bonding**.
  - Under the **Master Clock Generation Block Tab**
    - Turn ON **Enable Bonding Clock output ports**.
  - Under the **Dynamic Reconfiguration Tab**
    - Turn ON **Enable Dynamic Reconfiguration**.

3. Configure the Native PHY IP core using the **IP Parameter Editor**
   - Set the **Native PHY IP core TX Channel bonding mode** to either **PMA bonding** or **PMA/PCS bonding**.
   - Turn ON **Enable Dynamic Reconfiguration**.
4. Create a top level wrapper to connect the PLL IP cores to Native PHY IP core.
   - In this case, the PLL IP core has `tx_bonding_clocks` output bus with width [5:0].
   - The Native PHY IP core has `tx_bonding_clocks` input bus with width [5:0] multiplied by the number of channels in a transceiver bank. (six channels in the transceiver bank).
   - Unlike the x6/xN bonding mode, for this mode, the PLL should be instantiated multiple times. (One PLL is required for each transceiver bank that is a part of the bonded group.) Instantiate a PLL for each transceiver bank used.
   - Connect the `tx_bonding_clocks` output from each PLL to (up to) six channels in the same transceiver bank.
   - Connect the PLL IP core to the PHY IP core by duplicating the output of the PLL[5:0] for the number of transceiver channels used in the bonding group.

Steps to recalibrate the PLL after power up calibration

1. Dynamic reconfigure the PLL to change the feedback from the master CGB to feedback from PLL.
   - For ATX PLL, Read-Modify-Write 0x1 to offset address 0x110[2] of the ATX PLL.
   - For fPLL, Read-Modify-Write 0x1 to offset address 0x126[0] of the fPLL.
2. Recalibrate the PLL.
3. After recalibration completes, ensure the PLL achieves lock. Dynamic reconfigure the PLL to change the feedback to master CGB.

- For ATX PLL, Read-Modify-Write 0x0 to offset address 0x110[2] of the ATX PLL.
- For fPLL, Read-Modify-Write 0x0 to offset address 0x126[0] of the fPLL.

4. Recalibrate TX PMA of all the bonded channels driven by the ATX PLL or the fPLL.

Note: For this 10-channel example, two ATX PLLs are instantiated. Six channels of the `tx_bonding_clocks` on the Native PHY IP core are connected to the first ATX PLL and the remaining four channels are connected to the second ATX PLL's `tx_bonding_clock` outputs.

Related Information

- ATX PLL Recalibration on page 385
- Fractional PLL Recalibration on page 385
- PMA Recalibration on page 386

3.11.3. Implementing PLL Cascading

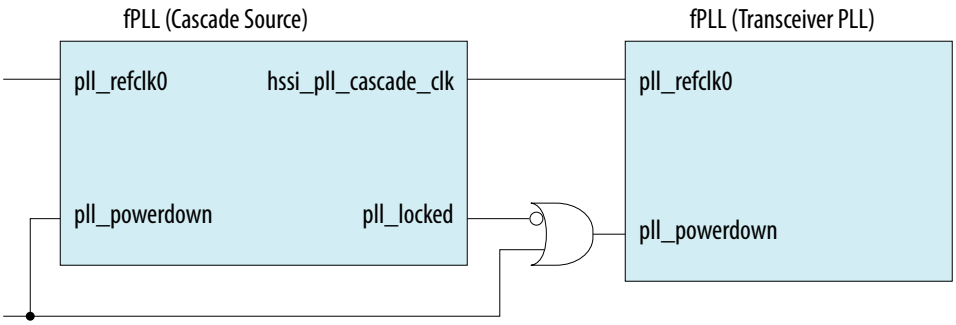
In PLL cascading, the output of the first PLL feeds the input reference clock to the second PLL.

For example, if the input reference clock has a fixed frequency, and the desired data rate was not an integer multiple of the input reference clock, the first PLL can be used to generate the correct reference clock frequency. This output is fed as the input reference clock to the second PLL. The second PLL generates the clock frequency required for the desired data rate.

The transceivers in Cyclone 10 GX devices support fPLL to fPLL cascading. Only maximum two PLLs are allowed in the cascading chain.

Note: When the fPLL is used as a cascaded fPLL (downstream fPLL), a user recalibration on the fPLL is required. Refer to the "User Recalibration" section for more information.

Figure 145. PLL Cascading



Steps to implement fPLL to fPLL cascading:

1. Instantiate the fPLL IP core.
2. Set the following configuration settings for the fPLL IP core in the **Parameter Editor**:

- Set the **fPLL Mode** to **Cascade Source**.
- Set the **Desired output clock frequency**.

3. Instantiate the fPLL IP core (the second PLL in PLL cascading configuration).
4. Configure the second fPLL IP core for the desired data rate and the reference clock frequency. Set reference clock frequency for the second fPLL same as the output frequency of the first fPLL.
5. Connect the fPLL IP core (cascade source) to fPLL IP core (transceiver PLL) as shown in the above figure. Ensure the following connections:
   - The fPLL has an output port `hssi_pll_cascade_clk`. Connect this port to the second fPLL's `pll_refclk0` port.
6. Set the source (upstream) fPLL bandwidth to Low setting and the destination (downstream) fPLL bandwidth to High setting.
7. If the input reference clock is available at device power-up, the first PLL is calibrated during the power-up calibration. The second PLL need to be recalibrated. Refer to the *User Recalibration* section. If the input reference clock is not available at device power-up, then re-run the calibration for the first PLL. After the first PLL has been calibrated, re-calibrate the second PLL.

Note: No special configuration is required for the Native PHY instance.

Related Information

User Recalibration on page 383

3.11.4. Timing Closure Recommendations

Register mode is harder to close timing in Cyclone 10 GX devices. Intel recommends using negative edge capture on the RX side for periphery to core transfers greater than 240 MHz. To be specific, capture on a negative edge clock in the core and then immediately transfer to a positive edge clock.

- Use PCLK clock network for frequencies up to 250 MHz.
- Local routing is recommended for higher frequencies.

For core to periphery transfers on TX targeting higher frequencies (beyond 250 MHz), Intel recommends using TX Fast Register mode as the PCS FIFO mode. This is the mode with PCLK that should be used by default for most 10GbE 1588 modes.

- You can use local routing to get up to 320 MHz in Register mode for the highest speed grade.

3.12. PLLs and Clock Networks Revision History

Document Version	Changes
2019.05.13	Updated the *Bonded Configurations* topic.
2017.11.06	Made the following changes:

continued...

Document Version	Changes
	• Added step 5 "If you reconfigure PLL for data rate change you must recalibrate the PLL" in Embedded Reconfiguration Streamer block of ATX PLL. • Added "The CLKUSR pin must be assigned a 100-125 MHz clock. For used transceiver TX and RX channels, do not assert the analog reset signals indefinitely" in "Unused/Idle Clock Line Requirements" section. • Added a sentence in fPLL/CMU PLL "For protocol jitter compliance at datarate > 10 Gbps, Intel recommends using the dedicated reference clock pin in the same triplet with the fPLL/CMU PLL as the input reference clock source."
2017.05.08	Initial release.

4. Resetting Transceiver Channels

To ensure that transceiver channels are ready to transmit and receive data, you must properly reset the transceiver PHY. Intel recommends a reset sequence that ensures the physical coding sublayer (PCS) and physical medium attachment (PMA) in each transceiver channel initialize and function correctly. You can either use the Transceiver PHY Reset Controller or create your own reset controller.

4.1. When Is Reset Required?

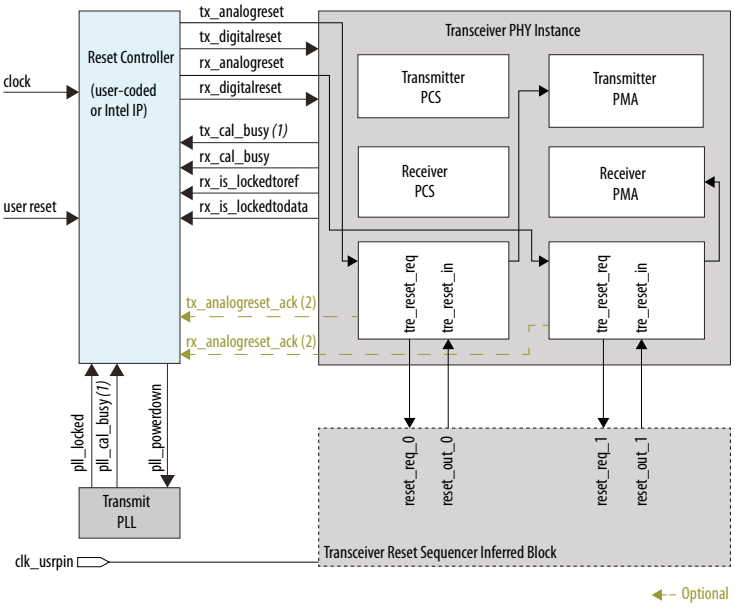
You can reset the transmitter (TX) and receiver (RX) data paths independently or together. The recommended reset sequence requires reset and initialization of the PLL driving the TX or RX channels, as well as the TX and RX datapaths. A reset is required after any of the following events:

Table 156. Reset Conditions

Event	Reset Requirement
Device power up and configuration	Requires reset to the transceiver PHY and the associated PLLs to a known initialize state.
PLL reconfiguration	Requires reset to ensure that the PLL acquires lock and also to reset the PHY. Both the PLL and the transmitter channel must be held in reset before performing PLL reconfiguration.
PLL reference clock frequency change	Requires reset to the PLL to ensure PLL lock. You must also reset the PHY.
PLL recalibration	Requires reset to the PLL to ensure PLL lock. You must also reset the PHY. Both the PLL and the transmitter channel must be held in reset before performing PLL recalibration.
PLL lock loss or recovery	Requires reset after a PLL acquired lock from a momentary loss of lock. You must also reset the PHY.
Channel dynamic reconfiguration	Requires holding the channel in reset before performing a dynamic reconfiguration that causes rate change. A PLL reset is not required during channel reconfiguration.
Optical module connection	Requires reset of RX to ensure lock of incoming data.
RX CDR lock mode change	Requires reset of the RX channel any time the RX clock and data recovery (CDR) block switches from lock-to-reference to lock-to-data RX channel.

4.2. Transceiver PHY Implementation

Figure 146. Typical Transceiver PHY Implementation



Notes:
(1) You can logical OR the pll_cal_busy and tx_cal_busy signals.
(2) tx_analogreset_ack and rx_analogreset_ack are status signals from the Transceiver PHY IP core when these ports are enabled for manual user implementation of Model 2.

Transceiver Reset Endpoints—The Transceiver PHY IP core contains Transceiver Reset Endpoints (TREs)[29].

Transceiver Reset Sequencer—The Quartus Prime software detects the presence of TREs and automatically inserts only one Transceiver Reset Sequencer (TRS)[29]. The `tx_analogreset` and `rx_analogreset` requests from the reset controller (User coded or Transceiver PHY Reset Controller) is received by the TREs. The TRE sends the reset request to the TRS for scheduling. TRS schedules all the requested PMA resets and sends them back to TREs. You can use either Transceiver PHY Reset Controller or your own reset controller. However, for the TRS to work correctly, the required timing duration must be followed. See Figure 147 on page 246 for required timing duration.

Note: The TRS IP is an inferred block and is not visible in the RTL. You have no control over this block.

CLKUSR connection—The clock to the TRS must be stable and free-running (100-125 MHz). By default, the Quartus Prime software automatically connects the TRS clock input to the CLKUSR pin on the device. If you are using the CLKUSR pin for your own logic (feeding it to the core), you must instantiate `altera_a10_xcvr_clock_module`

```
altera_a10_xcvr_clock_module reset_clock (.clk_in(mgmt_clk));
```

For more information about the CLKUSR pin, refer to the *Cyclone 10 GX Pin Connection Guidelines*.

[29] There is only one centralized TRS instantiated for one or more Native PHY.

Note: To successfully complete the calibration process, the reference clocks driving the PLLs (ATX PLL, fPLL, CDR/CMU PLL) must be stable and free running at start of FPGA configuration. Otherwise, recalibration is necessary.

Related Information

- Calibration on page 373
- Cyclone 10 GX Device Family Pin Connection Guidelines

4.3. How Do I Reset?

You reset a transceiver PHY or PLL by integrating a reset controller in your system design to initialize the PCS and PMA blocks. You can save time by using the Intel-provided Transceiver PHY Reset Controller IP core, or you can implement your own reset controller that follows the recommended reset sequence. You can design your own reset controller if you require individual control of each signal for reset or need additional control or status signals as part of the reset functionality.

You can choose from two models for resetting the transceivers, based on your applications:

- Model 1—Default model (Minimum Assertion Time Requirement)

  Choose the **Cyclone 10 GX Default Settings** preset for the Transceiver PHY Reset Controller IP.
- Model 2—Acknowledgment model

  This model uses an event-driven mechanism. The model is used for applications with strict timing requirements.

4.3.1. Model 1: Default Model

4.3.1.1. Recommended Reset Sequence

How to Enable Model 1

Choose the Intel Cyclone 10 GX Default Settings for the Transceiver PHY Reset Controller IP. This populates the reset duration fields with the correct values required by the transceiver reset sequencer (TRS).

Figure 147. Cyclone 10 GX Default Settings Preset

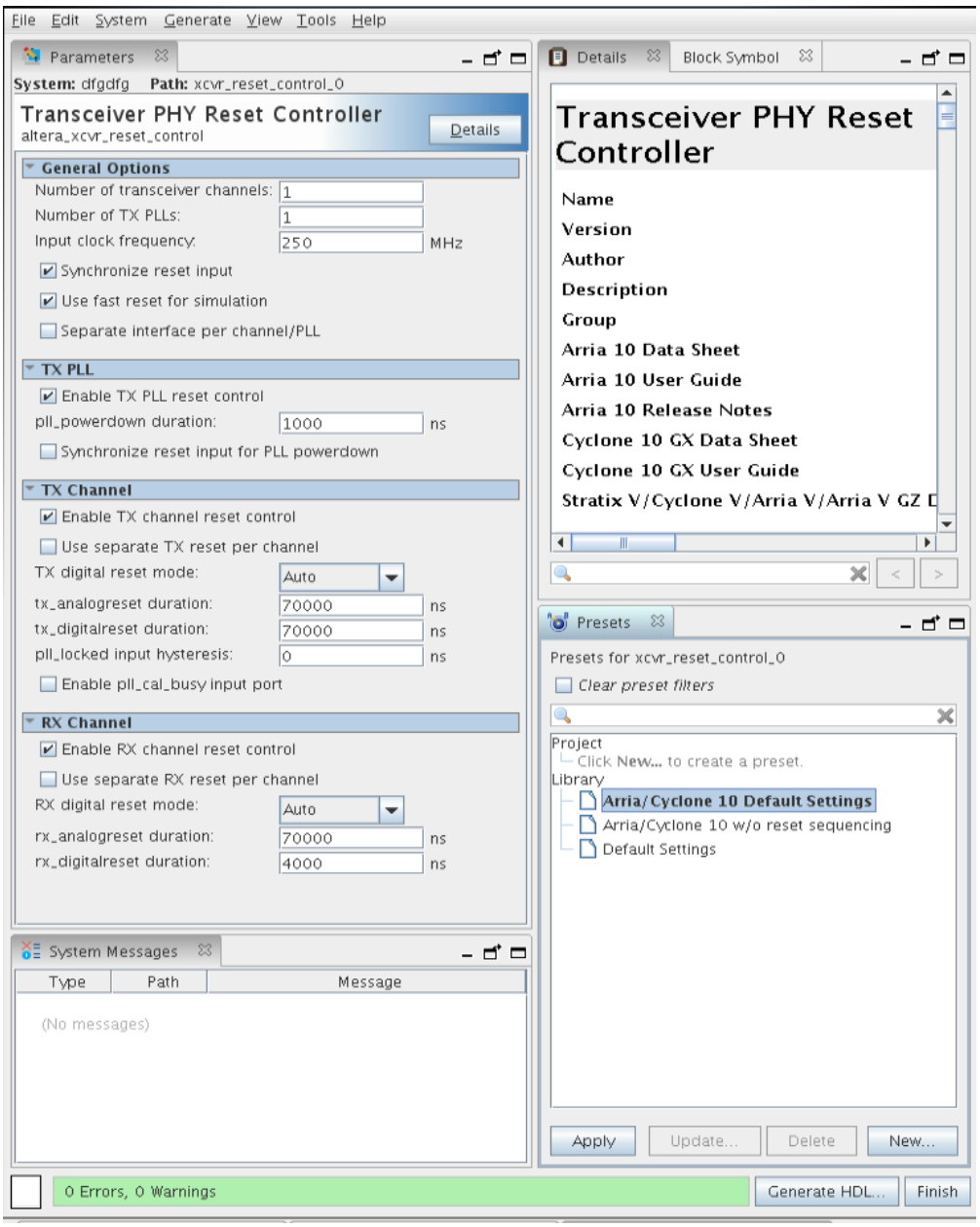
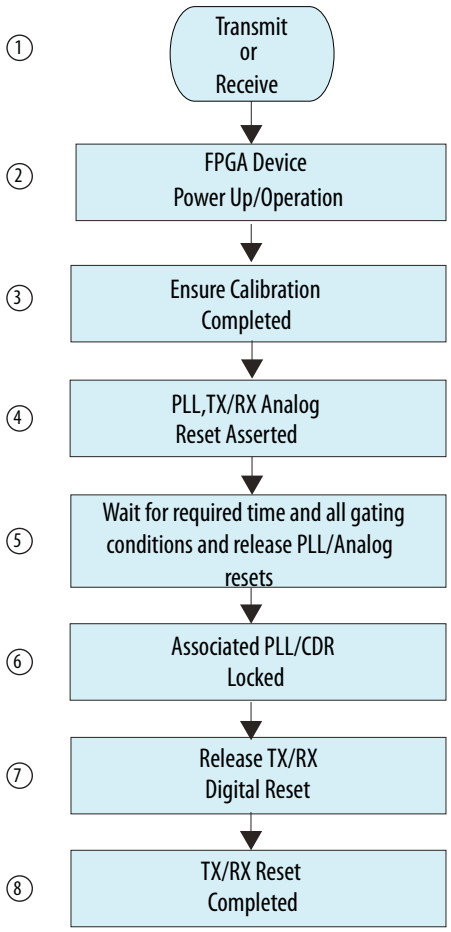


Figure 148. Transmitter and Receiver Reset Sequence



4.3.1.2. Resetting the Transmitter During Device Operation

Follow this reset sequence to reset the PLL or the analog or digital blocks of the transmitter at any point during the device operation. Use this reset sequence to reestablish a link or after dynamic reconfiguration. The following steps detail the transmitter reset sequence during device operation. The step numbers correspond to the numbers in the following figure.

1. Perform the following steps:

a. Assert `tx_analogreset`, `pll_powerdown`, and `tx_digitalreset` while `pll_cal_busy` and `tx_cal_busy` are low.

b. Deassert `pll_powerdown` after a minimum duration of 70 µs.

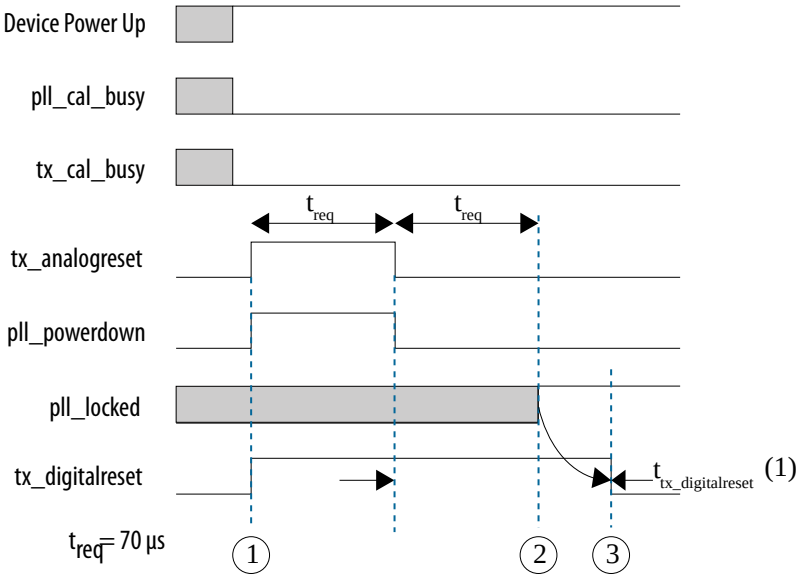
c. Deassert `tx_analogreset`. This step can be done at the same time or after you deassert `pll_powerdown`.

2. The `pll_locked` signal goes high after the TX PLL acquires lock. Wait for a minimum of 70 µs after deasserting `tx_analogreset` to monitor the `pll_locked` signal.

3. Deassert `tx_digitalreset`, after `pll_locked` goes high. The `tx_digitalreset` signal must stay asserted for a minimum $t_{tx\_digitalreset}$ duration after `tx_analogreset` is deasserted.

Note: You must reset the PCS blocks by asserting `tx_digitalreset`, every time you assert `pll_powerdown` and `tx_analogreset`.

Figure 149. Transmitter Reset Sequence During Device Operation



Note:
(1) The Cyclone 10 GX Default setting presets tx_digitalreset to 70 µs.
(2) Area in gray is don't care logic state.

Related Information

Cyclone 10 GX Device Datasheet

4.3.1.3. Resetting the Receiver During Device Operation

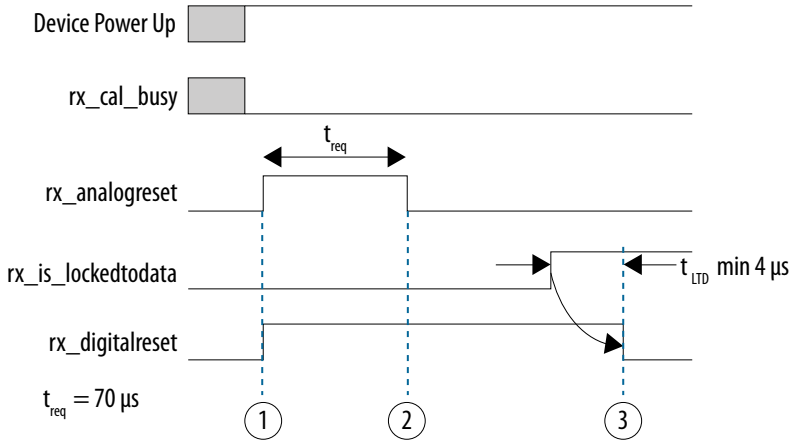
Follow this reset sequence to reset the analog or digital blocks of the receiver at any point during the device operation. Use this reset to re-establish a link or after dynamic reconfiguration.

4.3.1.3.1. Clock Data Recovery in Auto Lock Mode

The step numbers correspond to the numbers in the following figure:

1. Assert `rx_analogreset` and `rx_digitalreset`. Ensure that `rx_cal_busy` is low. You must reset the PCS by asserting `rx_digitalreset` every time you assert `rx_analogreset`.
2. Deassert `rx_analogreset` after a minimum duration of 70 µs.
3. Ensure `rx_is_lockedtodata` is asserted for t_{LTD} (minimum of 4 µs) before deasserting `rx_digitalreset`.

Figure 150. Resetting the Receiver During Device Operation (Auto Mode)



Note: `rx_is_lockedtodata` toggles when there is no data at the receiver input. `rx_is_lockedtoref` is a "do not care" when `rx_is_lockedtodata` is asserted.

4.3.1.3.2. Clock Data Recovery in Manual Lock Mode

Use the clock data recovery (CDR) manual lock mode to override the default CDR automatic lock mode depending on your design requirements.

Related Information

"*Transceiver PHY Reset Controller IP Core*" chapter of the *Altera Transceiver PHY IP Core User Guide*.

Refer to the description of the `rx_digitalreset` signal in the "Top-Level Signals" table for information about using the manual lock mode.

4.3.1.3.3. Control Settings for CDR Manual Lock Mode

Use the following control settings to set the CDR lock mode:

Table 157. Control Settings for the CDR in Manual Lock Mode

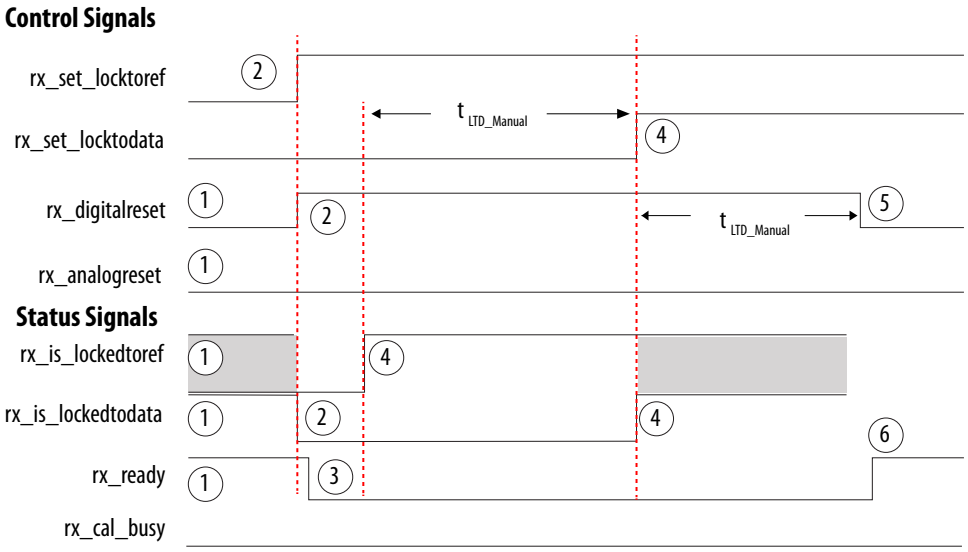
rx_set_locktoref	rx_set_locktodata	CDR Lock Mode
0	0	Automatic
1	0	Manual-RX CDR LTR
X	1	Manual-RX CDR LTD

4.3.1.3.4. Resetting the Transceiver in CDR Manual Lock Mode

The numbers in this list correspond to the numbers in the following figure, which guides you through the steps to put the CDR in manual lock mode.

1. Make sure that the calibration is complete (`rx_cal_busy` is low) and the transceiver goes through the initial reset sequence. The `rx_digitalreset` and `rx_analogreset` signals should be low. The `rx_is_lockedtoref` is a don't care and can be either high or low. The `rx_is_lockedtodata` and `rx_ready` signals should be high, indicating that the transceiver is out of reset. Alternatively, you can start directly with the CDR in manual lock mode after the calibration is complete.
2. Assert the `rx_set_locktoref` signal high to switch the CDR to the lock-to-reference mode. The `rx_is_lockedtodata` status signal is deasserted. Assert the `rx_digitalreset` signal high at the same time or after `rx_set_lockedtoref` is asserted if you use the user-coded reset. When the Transceiver PHY reset controller is used, the `rx_digitalreset` is automatically asserted.
3. After the `rx_digitalreset` signal gets asserted, the `rx_ready` status signal is deasserted.
4. Assert the `rx_set_locktodata` signal high, $t_{LTR\_LTD\_Manual}$ (minimum 15 µs) after the CDR is locked to reference. `rx_is_locktoref` should be high and stable for a minimum $t_{LTR\_LTD\_Manual}$ (15 µs), before asserting `rx_set_locktodata`. This is required to filter spurious glitches on `rx_is_lockedtoref`. The `rx_is_lockedtodata` status signal gets asserted, which indicates that the CDR is now set to LTD mode.
   The `rx_is_lockedtoref` status signal can be a high or low and can be ignored after asserting `rx_set_locktodata` high after the CDR is locked to reference.
5. Deassert the `rx_digitalreset` signal after a minimum of t_{LTD_Manual} (4 µs).
6. If you are using the Transceiver PHY Reset Controller, the `rx_ready` status signal gets asserted after the `rx_digitalreset` signal is deasserted. This indicates that the receiver is now ready to receive data with the CDR in manual mode.

Figure 151. Reset Sequence Timing Diagram for Receiver when CDR is in Manual Lock Mode

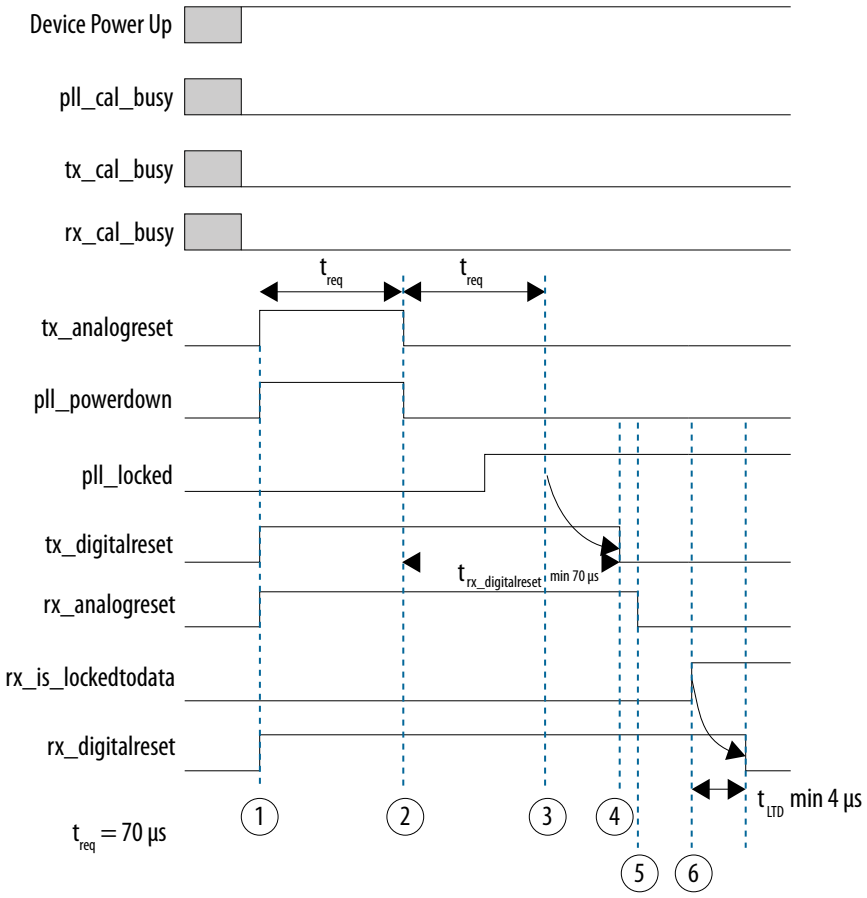


4.3.1.4. Resetting the Transceiver Channel During Device Operation

The numbers in this list correspond to the numbers in the following figure.

1. Assert `tx_analogreset`, `pll_powerdown`, `tx_digitalreset`, `rx_analogreset`, and `rx_digitalreset`. Ensure that `pll_cal_busy`, `tx_cal_busy`, and `rx_cal_busy` are low.
2. Deassert `pll_powerdown` and `tx_analogreset` at the same time, after a minimum duration of 70 µs.
3. The `pll_locked` signal goes high after the TX PLL acquires lock. Wait for a minimum 70 µs after deasserting `tx_analogreset` to monitor the `pll_locked` signal.
4. Deassert `tx_digitalreset` after `pll_locked` goes high. The `tx_digitalreset` signal must stay asserted for a minimum $t_{tx\_digitalreset}$ (minimum of 70 µs) duration after `tx_analogreset` is deasserted.
5. Deassert `rx_analogreset` after deasserting `tx_analogreset`.
6. Ensure `rx_is_lockedtodata` is asserted for t_{LTD} (minimum of 4 µs) before deasserting `rx_digitalreset`.

Figure 152. Resetting the Transceiver Channel During Device Operation



4.3.1.5. Dynamic Reconfiguration of Channel Using the Default Model

TX Channel

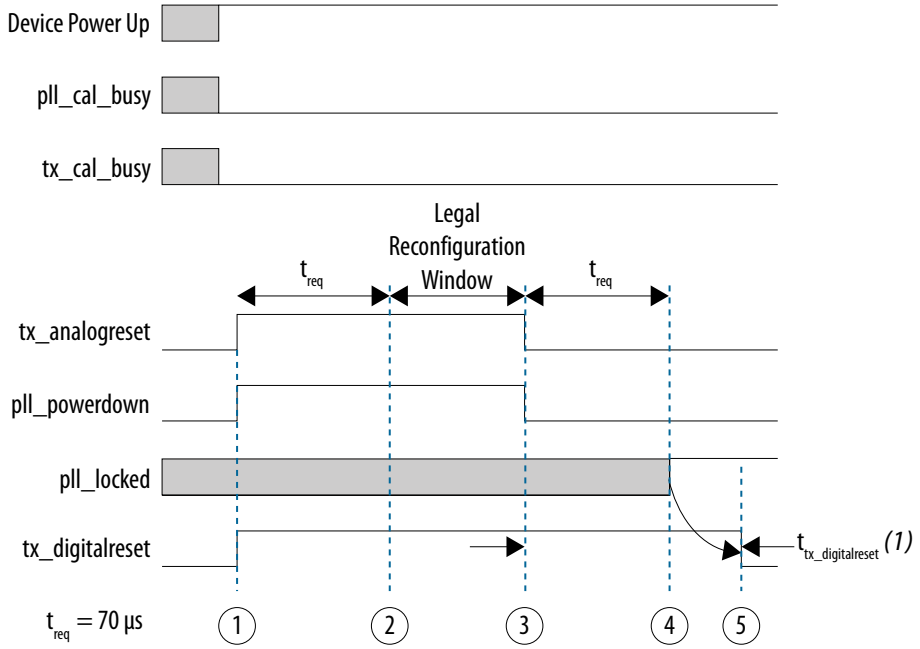
The numbers in this list correspond to the numbers in the following figure.

1. Assert `tx_analogreset`, `pll_powerdown`, and `tx_digitalreset`, while `pll_cal_busy` and `tx_cal_busy` are low.
2. Perform dynamic reconfiguration after minimum 70 µs of asserting `tx_analogreset`.
3. Deassert `pll_powerdown` after performing a dynamic reconfiguration.

Deassert `tx_analogreset`. This step can be done at the same time or after you deassert `pll_powerdown`.

4. The `pll_locked` signal goes high after the TX PLL acquires lock. Wait for minimum 70 µs after deasserting `tx_analogreset` to monitor the `pll_locked` signal.
5. Deassert `tx_digitalreset` after `pll_locked` goes high. The `tx_digitalreset` signal must stay asserted for a minimum $t_{tx\_digitalreset}$ duration after `tx_analogreset` is deasserted.

Figure 153. Dynamic Reconfiguration of Transmitter Channel During Device Operation



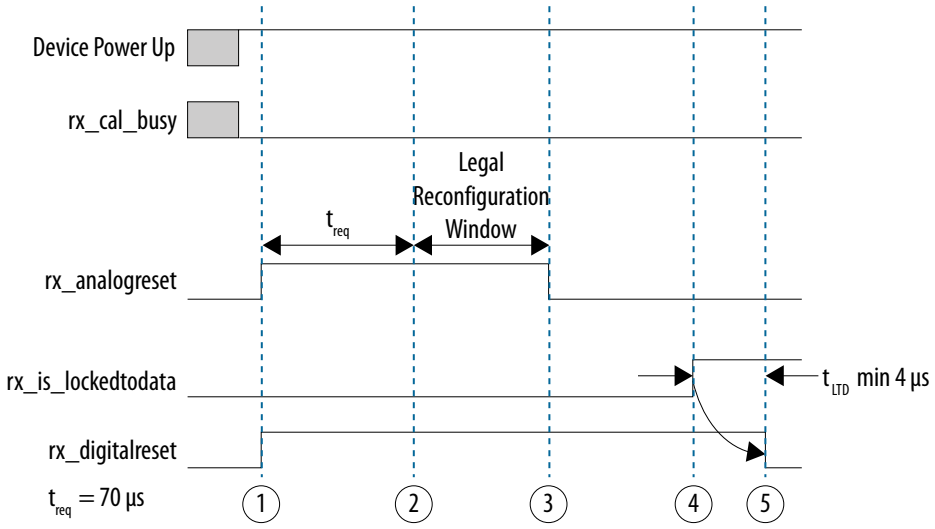
Note:
(1) The Cyclone 10 GX Default setting presets ttx_digitalreset µs.70
(2) Area in gray is don't care zone.

RX Channel

The numbers in this list correspond to the numbers in the following figure.

1. Assert `rx_analogreset` and `rx_digitalreset`. Ensure that `rx_cal_busy` is low. You must reset the PCS by asserting `rx_digitalreset` every time you assert `rx_analogreset`.
2. Perform dynamic reconfiguration after minimum 70 µs of asserting `rx_analogreset`.
3. Deassert `rx_analogreset` after performing dynamic reconfiguration.
4. The `rx_is_lockedtodata` signal goes high after the CDR acquires lock.
5. Ensure `rx_is_lockedtodata` is asserted for t_{LTD} (minimum of 4 µs) before deasserting `rx_digitalreset`.

Figure 154. Dynamic Reconfiguration of Receiver Channel During Device Operation



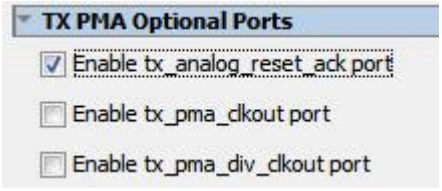
4.3.2. Model 2: Acknowledgment Model

The acknowledgment model uses an event-driven mechanism. It is used for applications with strict timing requirements. Instead of waiting for a minimum assertion time of 70 µs for `tx_analogreset` and `rx_analogreset`, you must wait to receive the acknowledgment from the Transceiver Native PHY IP core to ensure successful assertion and deassertion of the analog resets.

To enable the acknowledgment model, enable the following ports in the Transceiver Native PHY IP core:

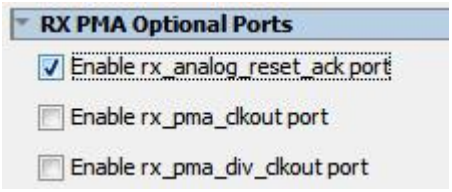
- Enable the `tx_analog_reset_ack` port in the TX PMA

Figure 155. Enabling the tx_analog_reset_ack Port



- Enable the `rx_analog_reset_ack` port in the RX PMA

Figure 156. Enabling the rx_analog_reset_ack Port



Note: `tx_analog_reset_ack` and `rx_analog_reset_ack` must be treated as asynchronous signals. You must pass them through a synchronizer before sending them to control logic.

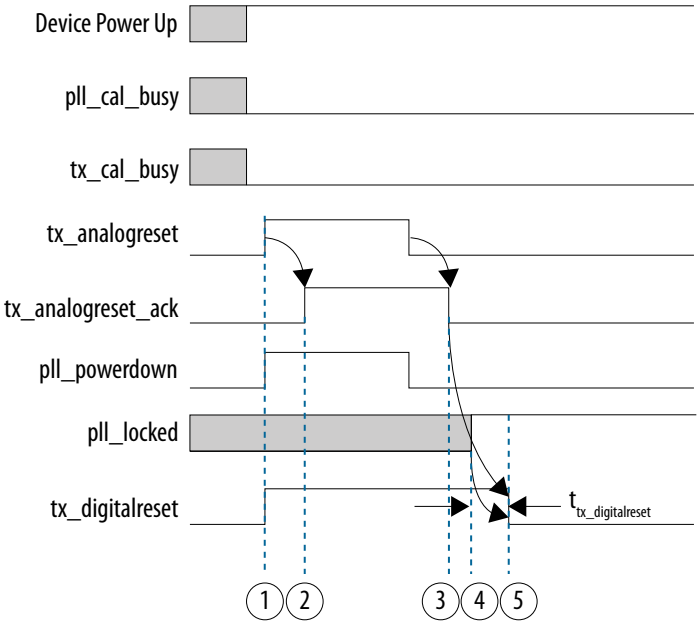
4.3.2.1. Recommended Reset Sequence

4.3.2.1.1. Resetting the Transmitter During Device Operation

The numbers in this list correspond to the numbers in the following figure.

1. Assert `tx_analogreset`, `pll_powerdown`, and `tx_digitalreset`, while `pll_cal_busy` and `tx_cal_busy` are low.
2. Wait for `tx_analogreset_ack` to go high, to ensure successful assertion of `tx_analogreset`. `tx_analogreset_ack` goes high when TRS has successfully completed the reset request for assertion.
   a. Deassert `pll_powerdown` after $t_{pll\_powerdown}$.
   b. Deassert `tx_analogreset`. This step can be done at the same time or after you deassert `pll_powerdown`.
3. Wait for `tx_analogreset_ack` to go low, to ensure successful deassertion of `tx_analogreset`. `tx_analogreset_ack` goes low when TRS has successfully completed the reset request for deassertion.
4. The `pll_locked` signal goes high after the TX PLL acquires lock. Wait for `tx_analogreset_ack` to go low before monitoring the `pll_locked` signal.
5. Deassert `tx_digitalreset` a minimum $t_{tx\_digitalreset}$ time after `pll_locked` goes high.

Figure 157. Transmitter Reset Sequence During Device Operation



Note:

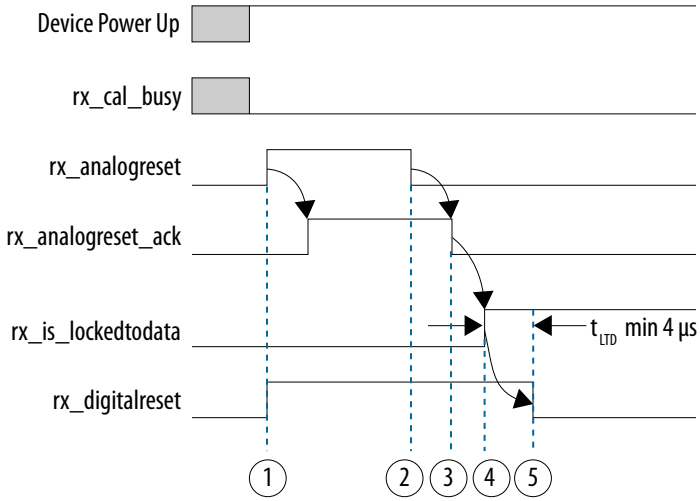
(1) Area in gray is don't care logic state.

4.3.2.1.2. Resetting the Receiver During Device Operation

The numbers in this list correspond to the numbers in the following figure.

1. Assert `rx_analogreset` and `rx_digitalreset` while `rx_cal_busy` is low.
2. Wait for `rx_analogreset_ack` to go high, to ensure successful assertion of `rx_analogreset`. `rx_analogreset_ack` goes high when TRS has successfully completed the reset request for assertion.
   a. Deassert `rx_analogreset`.
3. Wait for `rx_analogreset_ack` to go low, to ensure successful deassertion of `rx_analogreset`. `rx_analogreset_ack` goes low when TRS has successfully completed the reset request for deassertion.
4. The `rx_is_lockedtodata` signal goes high after the CDR acquires lock.
5. Ensure `rx_is_lockedtodata` is asserted for t_{LTD} (minimum of 4 µs) before deasserting `rx_digitalreset`.

Figure 158. Receiver Reset Sequence During Device Operation



4.3.2.1.3. Dynamic Reconfiguration of Transmitter Channel Using the Acknowledgment Model

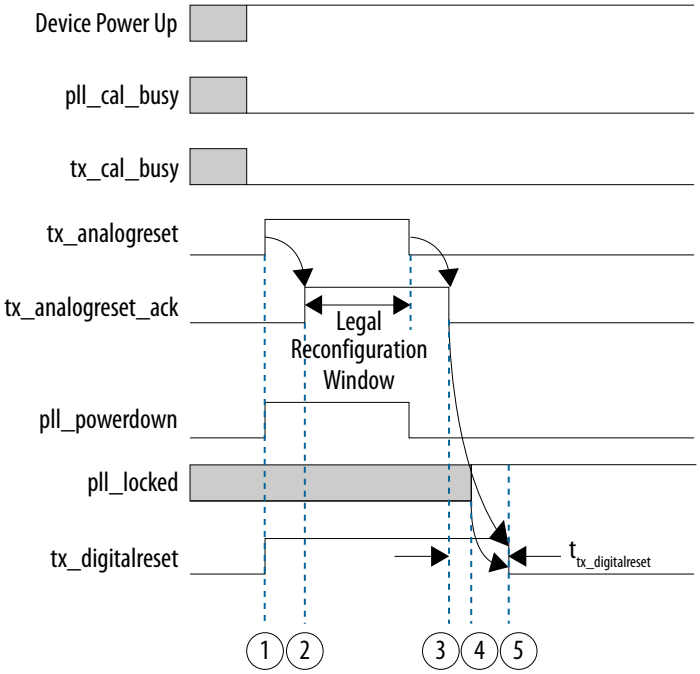
The numbers in this list correspond to the numbers in the following figure.

1. Assert `tx_analogreset`, `pll_powerdown`, and `tx_digitalreset`, while `pll_cal_busy` and `tx_cal_busy` are low.
2. Wait for `tx_analogreset_ack` to go high, to ensure successful assertion of `tx_analogreset`. `tx_analogreset_ack` goes high when TRS has successfully completed the reset request for assertion.

a. Perform dynamic reconfiguration.
b. Deassert `pll_powerdown` after $t_{pll\_powerdown}$.
c. Deassert `tx_analogreset`. This step can be done at the same time or after you deassert `pll_powerdown`.

3. Wait for `tx_analogreset_ack` to go low, to ensure successful deassertion of `tx_analogreset`. `tx_analogreset_ack` goes low when TRS has successfully completed the reset request for deassertion.
4. The `pll_locked` signal goes high after the TX PLL acquires lock. Wait for `tx_analogreset_ack` to go low before monitoring the `pll_locked` signal.
5. Deassert `tx_digitalreset` a minimum $t_{tx\_digitalreset}$ time after `pll_locked` goes high.

Figure 159. Dynamic Reconfiguration of Transmitter Channel During Device Operation



Note:

(1) Area in gray is don't care zone.

4.3.2.1.4. Dynamic Reconfiguration of Receiver Channel Using the Acknowledgment Model

The numbers in this list correspond to the numbers in the following figure.

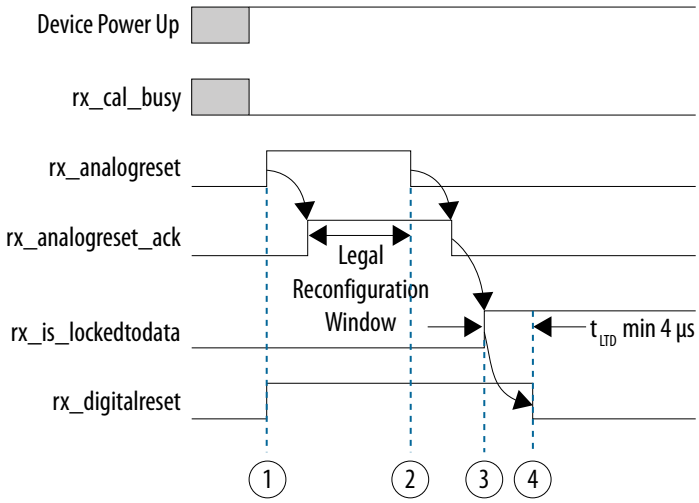
1. Assert `rx_analogreset` and `rx_digitalreset` while `rx_cal_busy` is low.

a. Wait for `rx_analogreset_ack` to go high, to ensure successful assertion of `rx_analogreset`. `rx_analogreset_ack` goes high when TRS has successfully completed the reset request for assertion.

b. Deassert `rx_analogreset`.

2. Wait for `rx_analogreset_ack` to go low, to ensure successful deassertion of `rx_analogreset`. `rx_analogreset_ack` goes low when TRS has successfully completed the reset request for deassertion.
3. Ensure `rx_is_lockedtodata` signal goes high after the CDR is locked to data. Wait for `rx_analogreset_ack` to go low before monitoring `rx_is_lockedtodata` signal.
4. Deassert `rx_digitalreset` after a minimum of t_{LTD} (minimum of 4 µs) after `rx_is_lockedtodata` goes high.

Figure 160. Dynamic Reconfiguration of Receiver Channel During Device Operation



4.3.3. Transceiver Blocks Affected by Reset and Powerdown Signals

You must reset the digital PCS each time you reset the analog PMA or PLL. However, you can reset the digital PCS block alone.

Table 158. Transceiver Blocks Affected by Specified Reset and Powerdown Signals

Transceiver Block	pll_powerdown	tx_analogreset	tx_digitalreset	rx_analogreset	rx_digitalreset
CMU PLL	Yes				
ATX PLL	Yes				
fPLL	Yes				
CDR				Yes	
Receiver Standard PCS					Yes
Receiver Enhanced PCS					Yes

continued...

Transceiver Block	pll_powerdown	tx_analogreset	tx_digitalreset	rx_analogreset	rx_digitalreset
Receiver PMA				Yes	
Transmitter Standard PCS			Yes		
Transmitter Enhanced PCS			Yes		
Transmitter PMA		Yes			

Related Information

User Recalibration on page 383

4.4. Using the Transceiver PHY Reset Controller

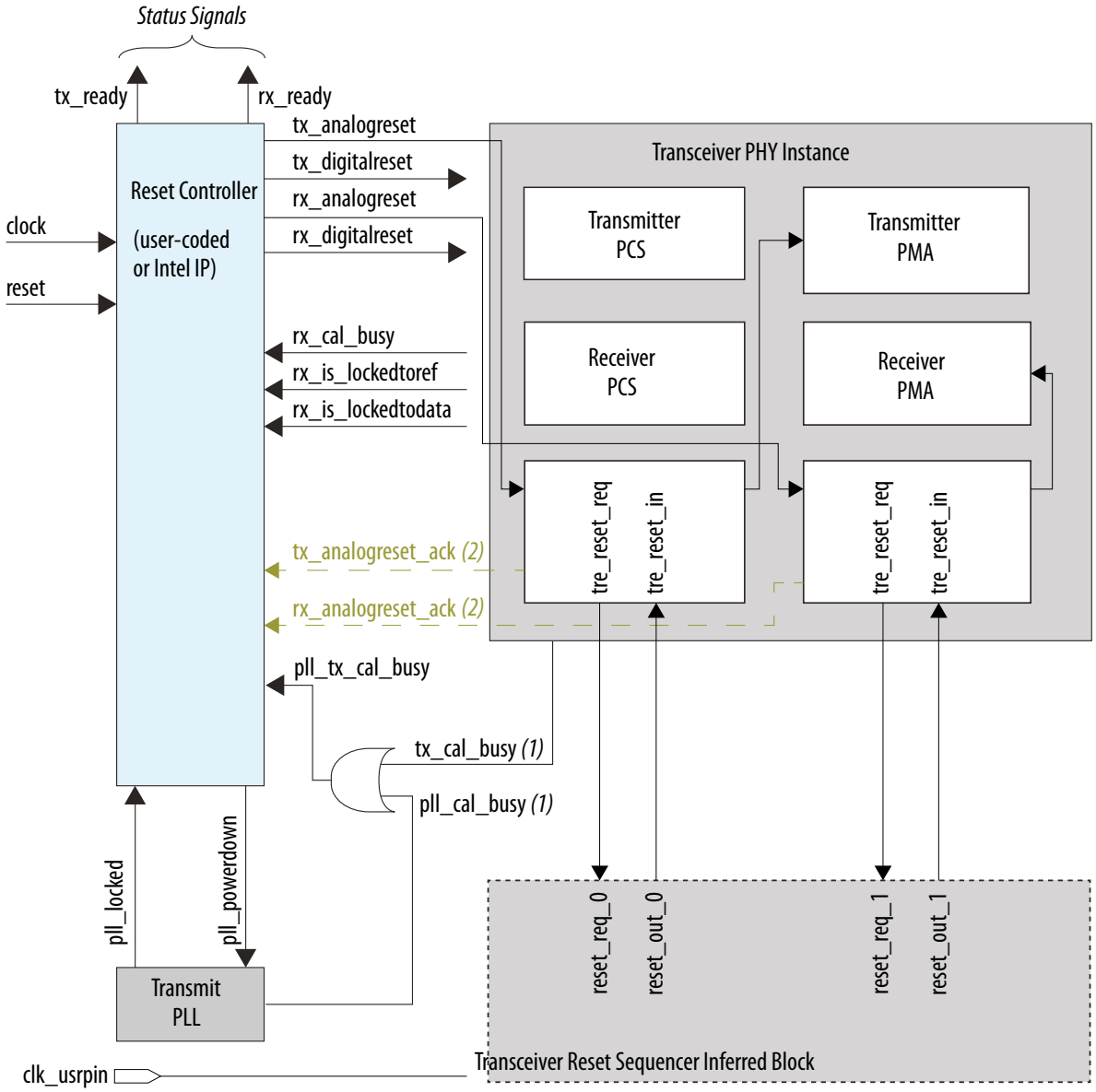
Transceiver PHY Reset Controller is a configurable IP core that resets transceivers mainly in response to PLL lock activity. You can use this IP core rather than creating your own user-coded reset controller. You can define a custom reset sequence for the IP core. You can also modify the IP cores's generated clear text Verilog HDL file to implement custom reset logic.

The Transceiver PHY Reset Controller handles all transceiver reset sequencing and supports the following options:

- Separate or shared reset controls per channel in response to PLL lock activity
- Separate controls for the TX and RX channels and PLLs
- Synchronization of the reset inputs
- Hysteresis for PLL locked status inputs
- Configurable reset timing
- Automatic or manual reset recovery mode in response to loss of PLL lock

You should create your own reset controller if the Transceiver PHY Reset Controller IP does not meet your requirements, especially when you require independent transceiver channel reset. The following figure illustrates the typical use of the Transceiver PHY Reset Controller in a design that includes a transceiver PHY instance and the transmit PLL.

Figure 161. Transceiver PHY Reset Controller System Diagram



Note:

(1) You can logical OR the pll_cal_busy and tx_cal_busy signals.
pll_tx_cal_busy connects to the controller's tx_cal_busy input port.

(2) The ports are inputs for user logic that implement Model 2. The ports can be used as status monitoring for Model 1 implementation.

The Transceiver PHY Reset Controller IP core connects to the Transceiver PHY and the Transmit PLL. The Transceiver PHY Reset Controller IP core receives status from the Transceiver PHY and the Transmit PLL. Based on the status signals or the reset input, it generates TX and RX reset signals to the Transceiver PHY and TX PLL.

The `tx_ready` signal indicates whether the TX PMA exits the reset state, and if the TX PCS is ready to transmit data. The `rx_ready` signal indicates whether the RX PMA exits the reset state, and if the RX PCS is ready to receive data. You must monitor these signals to determine when the transmitter and receiver are out of the reset sequence.

4.4.1. Parameterizing the Transceiver PHY Reset Controller IP

This section lists steps to configure the Transceiver PHY Reset Controller IP Core in the IP Catalog. You can customize the following Transceiver PHY Reset Controller parameters for different modes of operation by clicking **Tools ➤ IP Catalog**.

To parameterize and instantiate the Transceiver PHY Reset Controller IP core:

1. For **Device Family**, select your target device from the list.
2. Click **Installed IP ➤ Library ➤ Interface Protocols ➤ Transceiver PHY ➤ Transceiver PHY Reset Controller**.
3. Select the options required for your design. For a description of these options, refer to the **Transceiver PHY Reset Controller Parameters**.
4. Click **Finish**. The wizard generates files representing your parameterized IP variation for synthesis and simulation.

4.4.2. Transceiver PHY Reset Controller Parameters

The Quartus Prime software provides a GUI to define and instantiate a Transceiver PHY Reset Controller to reset transceiver PHY and external PLL.

Table 159. General Options

Name	Range	Description
Number of transceiver channels	`1-N`	Specifies the number of channels that connect to the Transceiver PHY Reset Controller IP core. The maximum N limit of the range is determined by your FPGA architecture.
Number of TX PLLs	`1-N`	Specifies the number of TX PLLs that connect to the Transceiver PHY Reset Controller IP core.
Input clock frequency	`1-500 MHz`	Input clock to the Transceiver PHY Reset Controller IP core. The frequency of the input clock in MHz. The upper limit on the input clock frequency is the frequency achieved in timing closure.
Synchronize reset input	**On** /**Off**	When **On**, the Transceiver PHY Reset Controller synchronizes the reset to the Transceiver PHY Reset Controller input clock before driving it to the internal reset logic. When **Off**, the reset input is not synchronized.
Use fast reset for simulation	**On** /**Off**	When **On**, the Transceiver PHY Reset Controller uses reduced reset counters for simulation.

continued...

Name	Range	Description
Separate interface per channel/PLL	**On /Off**	When **On**, the Transceiver PHY Reset Controller provides a separate reset interface for each channel and PLL.
TX PLL		
Enable TX PLL reset control	**On /Off**	When **On**, the Transceiver PHY Reset Controller IP core enables the reset control of the TX PLL. When **Off**, the TX PLL reset control is disabled.
pll_powerdown duration	`1-999999999`	Specifies the duration of the PLL powerdown period in ns. The value is rounded up to the nearest clock cycle. The default value is 1000 ns.
Synchronize reset input for PLL powerdown	**On /Off**	When **On**, the Transceiver PHY Reset Controller synchronizes the PLL powerdown reset with the Transceiver PHY Reset Controller input clock. When **Off**, the PLL powerdown reset is not synchronized.
TX Channel		
Enable TX channel reset control	**On /Off**	When **On**, the Transceiver PHY Reset Controller enables the control logic and associated status signals for TX reset. When **Off**, disables TX reset control and status signals.
Use separate TX reset per channel	**On /Off**	When **On**, each TX channel has a separate reset. When **Off**, the Transceiver PHY Reset Controller uses a shared TX reset controller for all channels.
TX digital reset mode	**Auto**, **Manual**, **Expose Port**	Specifies the Transceiver PHY Reset Controller behavior when the `pll_locked` signal is deasserted. The following modes are available: • **Auto**—The associated `tx_digitalreset` controller automatically resets whenever the `pll_locked` signal is deasserted. Intel recommends this mode. • **Manual**—The associated `tx_digitalreset` controller is not reset when the `pll_locked` signal is deasserted, allowing you to choose corrective action. • **Expose Port**—The `tx_manual` signal is a top-level signal of the IP core. You can dynamically change this port to Auto or Manual. (1= Manual , 0 = Auto)
tx_analogreset duration	`1-999999999`	Specifies the time in ns to continue to assert `tx_analoglreset` after the reset input and all other gating conditions are removed. The value is rounded up to the nearest clock cycle. *Note:* Model 1 requires this to be set to 70 µs. Select the **Cyclone 10 GX Default Settings** preset.
tx_digitalreset duration	`1-999999999`	Specifies the time in ns to continue to assert the `tx_digitalreset` after the reset input and all other gating conditions are removed. The value is rounded up to the nearest clock cycle. *Note:* Model 1 requires this to be set to 70 µs. Select the **Cyclone 10 GX Default Settings** preset. The default value for Model 2 is 20 ns.
pll_locked input hysteresis	`0-999999999`	Specifies the amount of hysteresis in ns to add to the `pll_locked` status input to filter spurious unreliable assertions of the `pll_locked` signal. A

continued...

Name	Range	Description
		value of 0 adds no hysteresis. A higher value filters glitches on the `pll_locked` signal. Intel recommends that the amount of hysteresis be longer than `tpll_lock_max_time`.
RX Channel		
Enable RX channel reset control	**On /Off**	When **On**, each RX channel has a separate reset input. When **Off**, each RX channel uses a shared RX reset input for all channels. This implies that if one of the RX channels is not locked, all the other RX channels is held in reset until all RX channels are locked. Digital reset stays asserted until all RX channels have acquired lock.
Use separate RX reset per channel	**On /Off**	When **On**, each RX channel has a separate reset input. When **Off**, uses a shared RX reset controller for all channels.
RX digital reset mode	**Auto**, **Manual**, **Expose Port**	Specifies the Transceiver PHY Reset Controller behavior when the PLL lock signal is deasserted. The following modes are available: • **Auto**—The associated `rx_digitalreset` controller automatically resets whenever the `rx_is_lockedtodata` signal is deasserted. • **Manual**—The associated `rx_digitalreset` controller is not reset when the `rx_is_lockedtodata` signal is deasserted, allowing you to choose corrective action. • **Expose Port**—The `rx_manual` signal is a top-level signal of the IP core. If the core includes separate reset control for each RX channel, each RX channel uses its respective `rx_is_lockedtodata` signal for automatic reset control; otherwise, the inputs are ANDed to provide internal status for the shared reset controller.
rx_analogreset duration	`1-999999999`	Specifies the time in ns to continue to assert the `rx_analogreset` after the reset input and all other gating conditions are removed. The value is rounded up to the nearest clock cycle. The default value is 40 ns. *Note:* Model 1 requires this to be set to 70 µs. Select the **Cyclone 10 GX Default Settings** preset.
rx_digitalreset duration	`1-999999999`	Specifies the time in ns to continue to assert the `rx_digitalreset` after the reset input and all other gating conditions are removed. The value is rounded up to the nearest clock cycle. The default value is 4000 ns.

4.4.3. Transceiver PHY Reset Controller Interfaces

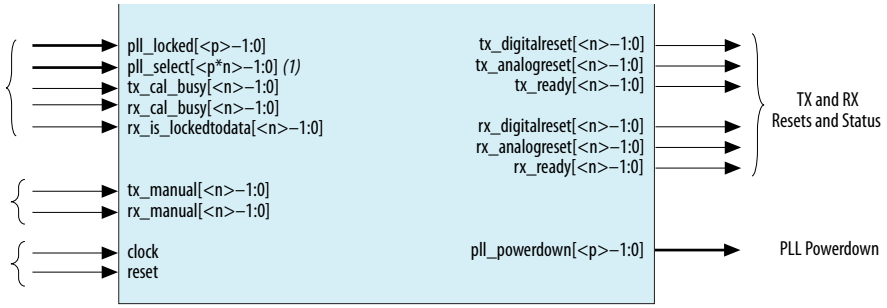
This section describes the top-level signals for the Transceiver PHY Reset Controller IP core.

The following figure illustrates the top-level signals of the Transceiver PHY Reset Controller IP core. Many of the signals in the figure become buses if you choose separate reset controls. The variables in the figure represent the following parameters:

- <*n*>—The number of lanes
- <*p*>—The number of PLLs

Figure 162. Transceiver PHY Reset Controller IP Core Top-Level Signals

Generating the IP core creates signals and ports based on your parameter settings.



pll_select signal width when a single TX reset sequence is used for all channels.

Note: PLL control is available when you enable the **Expose Port** parameter.

Table 160. Top-Level Signals

This table describes the signals in the above figure in the order that they are shown in the figure.

Signal Name	Direction	Clock Domain	Description
`pll_locked[<p>-1:0]`	Input	Asynchronous	Provides the PLL locked status input from each PLL. When asserted, indicates that the TX PLL is locked. When deasserted, the PLL is not locked. There is one signal per PLL.
`pll_select[<p*n>-1:0]`	Input	Synchronous to the Transceiver PHY Reset Controller input clock. Set to zero when not using multiple PLLs.	When you select **Use separate TX reset per channel**, this bus provides enough inputs to specify an index for each `pll_locked` signal to listen to for each channel. When **Use separate TX reset per channel** is disabled, the `pll_select` signal is used for all channels. n=1 when a single TX reset sequence is used for all channels.
`tx_cal_busy[<n> -1:0]`	Input	Asynchronous	This is the calibration status signal that results from the logical OR of `pll_cal_busy` and `tx_cal_busy` signals. The signal goes high when either the TX PLL or Transceiver PHY initial calibration is active. It is not asserted if you manually re-trigger the calibration IP. The signal goes low when calibration is completed. This signal gates the TX reset sequence. The width of this signals depends on the number of TX channels.

continued...

Signal Name	Direction	Clock Domain	Description
rx_cal_busy[<n> -1:0]	Input	Asynchronous	This is calibration status signal from the Transceiver PHY IP core. When asserted, the initial calibration is active. When deasserted, calibration has completed. It is not asserted if you manually re-trigger the calibration IP. This signal gates the RX reset sequence. The width of this signals depends on the number of RX channels.
rx_is_lockedtodata[<*n*>-1:0]	Input	Synchronous to CDR	Provides the rx_is_lockedtodata status from each RX CDR. When asserted, indicates that a particular RX CDR is ready to receive input data. If you do not choose separate controls for the RX channels, these inputs are ANDed together internally to provide a single status signal.
tx_manual[<*n*>-1:0]	Input	Asynchronous	This optional signal places tx_digitalreset controller under automatic or manual control. When asserted, the associated tx_digitalreset controller logic does not automatically respond to deassertion of the pll_locked signal. However, the initial tx_digitalreset sequence still requires a one-time rising edge on pll_locked before proceeding. When deasserted, the associated tx_digitalreset controller automatically begins its reset sequence whenever the selected pll_locked signal is deasserted.
rx_manual[<*n*> -1:0]	Input	Asynchronous	This optional signal places rx_digitalreset logic controller under automatic or manual control. In manual mode, the rx_digitalreset controller does not respond to the assertion or deassertion of the rx_is_lockedtodata signal. The rx_digitalreset controller asserts rx_ready when the rx_is_lockedtodata signal is asserted.
clock	Input	N/A	A free running system clock input to the Transceiver PHY Reset Controller from which all internal logic is driven. If a free running clock is not available, hold reset until the system clock is stable.
reset	Input	Asynchronous	Asynchronous reset input to the Transceiver PHY Reset Controller. When asserted, all configured reset outputs are asserted. Holding the reset input signal asserted holds all other reset outputs asserted. An option is available to synchronize with the system clock. In synchronous mode, the reset signal needs to stay asserted for at least (2) clock cycles by default.
tx_digitalreset[<*n*>-1:0]	Output	Synchronous to the Transceiver PHY Reset Controller input clock.	Digital reset for TX channels. The width of this signal depends on the number of TX channels. This signal is asserted when any of the following conditions is true: • reset is asserted • pll_powerdown is asserted • pll_cal_busy is asserted • tx_cal_busy is asserted • PLL has not reached the initial lock (pll_locked deasserted) • pll_locked is deasserted and tx_manual is deasserted When all of these conditions are false, the reset counter begins its countdown for deassertion of tx_digitalreset.

continued...

Signal Name	Direction	Clock Domain	Description
tx_analogreset[<n>-1:0]	Output	Synchronous to the Transceiver PHY Reset Controller input clock.	Analog reset for TX channels. The width of this signal depends on the number of TX channels. This signal is asserted when reset is asserted. This signal follows pll_powerdown, which is deasserted after pll_locked goes high.
tx_ready[<n>-1:0]	Output	Synchronous to the Transceiver PHY Reset Controller input clock.	Status signal to indicate when the TX reset sequence is complete. This signal is deasserted while the TX reset is active. It is asserted a few clock cycles after the deassertion of tx_digitalreset. Some protocol implementations may require you to monitor this signal prior to sending data. The width of this signal depends on the number of TX channels.
rx_digitalreset[<n>-1:0]	Output	Synchronous to the Transceiver PHY Reset Controller input clock.	Digital reset for RX. The width of this signal depends on the number of channels. This signal is asserted when any of the following conditions is true: • reset is asserted • rx_analogreset is asserted • rx_cal_busy is asserted • rx_is_lockedtodata is deasserted and rx_manual is deasserted When all of these conditions are false, the reset counter begins its countdown for deassertion of rx_digitalreset.
rx_analogreset[<n>-1:0]	Output	Synchronous to the Transceiver PHY Reset Controller input clock.	Analog reset for RX. When asserted, resets the RX CDR and the RX PMA blocks of the transceiver PHY. This signal is asserted when any of the following conditions is true: • reset is asserted • rx_cal_busy is asserted The width of this signal depends on the number of channels.
rx_ready[<n>-1:0]	Output	Synchronous to the Transceiver PHY Reset Controller input clock.	Status signal to indicate when the RX reset sequence is complete. This signal is deasserted while the RX reset is active. It is asserted a few clock cycles after the deassertion of rx_digitalreset. Some protocol implementations may require you to monitor this signal prior to sending data. The width of this signal depends on the number of RX channels.
pll_powerdown[<p>-1:0]	Output	Synchronous to the Transceiver PHY Reset Controller input clock.	Asserted to power down a transceiver PLL circuit. When asserted, the selected TX PLL is reset.

Usage Examples for pll_select

- If a single channel can switch between three TX PLLs, the pll_select signal indicates which one of the selected three TX PLL's pll_locked signal is used to communicate the PLL lock status to the TX reset sequence. In this case, to select the 3-bits wide pll_locked port, the pll_select port is 2-bits wide.
- If three channels are instantiated with three TX PLLs and with a separate TX reset sequence per channel, the pll_select field is 6-bits wide (2-bits per channel). In this case, pll_select [1:0] represents channel 0, pll_select[3:2] represents channel 1, and pll_select[5:4] represents channel 2. For each channel, a separate pll_locked signal indicates the PLL lock status.
- If three channels are instantiated with three TX PLLs and with a single TX reset sequence for all three channels, then pll_select field is 2-bits wide. In this case, the same pll_locked signal indicates the PLL lock status for all three channels.
- If one channel is instantiated with one TX PLL, pll_select field is 1-bit wide. Connect pll_select to logic 0.
- If three channels are instantiated with only one TX PLL and with a separate TX reset sequence per channel, the pll_select field is 3-bits wide. In this case, pll_select should be set to 0 since there is only one TX PLL available.

4.4.4. Transceiver PHY Reset Controller Resource Utilization

This section describes the estimated device resource utilization for two configurations of the transceiver PHY reset controller. The exact resource count varies by Quartus Prime version number, as well as by optimization options.

Table 161. Reset Controller Resource Utilization

Configuration	Combination ALUTs	Logic Registers
Single transceiver channel	approximately 50	approximately 50
Four transceiver channels, shared TX reset, separate RX resets	approximately 100	approximately 150

4.5. Using a User-Coded Reset Controller

You can design your own user-coded reset controller instead of using Transceiver PHY Reset Controller. Your user-coded reset controller must provide the following functionality for the recommended reset sequence:

- A clock signal input for your reset logic
- Holds the transceiver channels in reset by asserting the appropriate reset control signals
- Checks the PLL status (for example, checks the status of pll_locked and pll_cal_busy)

Note: You must ensure a stable reference clock is present at the PLL transmitter before releasing pll_powerdown.

4.5.1. User-Coded Reset Controller Signals

Refer to the signals in the following figure and table for implementation of a user-coded reset controller.

Figure 163. User-Coded Reset Controller, Transceiver PHY, and TX PLL Interaction

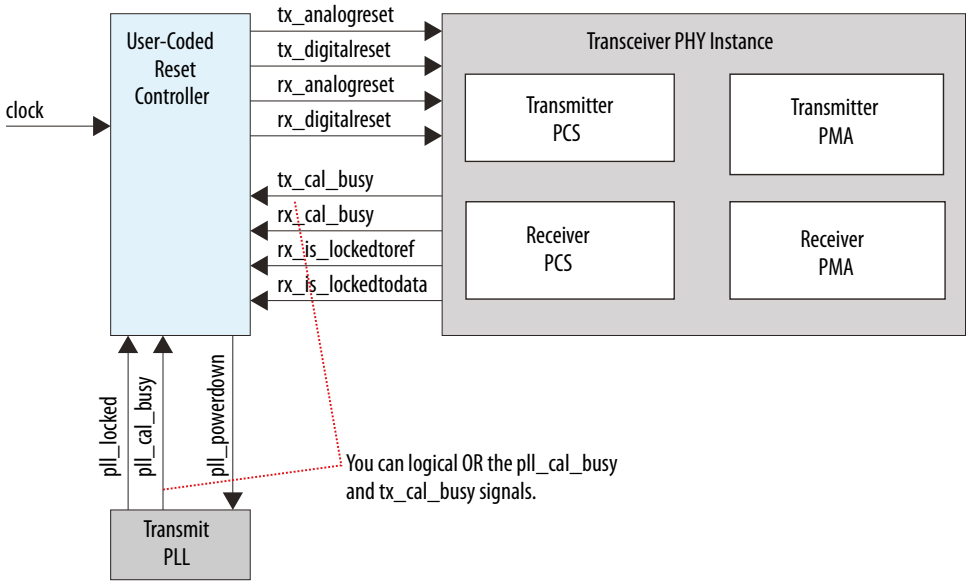


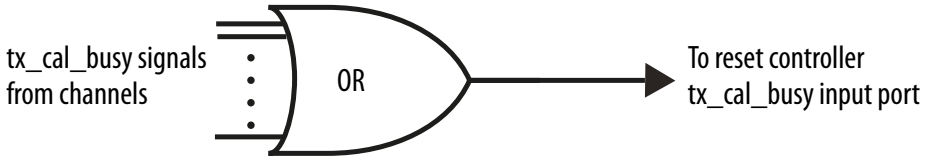
Table 162. User-coded Reset Controller, Transceiver PHY, and TX PLL Signals

Signal Name	Direction	Description
`pll_powerdown`	Output	Resets the TX PLL when asserted high.
`tx_analogreset`	Output	Resets the TX PMA when asserted high.
`tx_digitalreset`	Output	Resets the TX PCS when asserted high.
`rx_analogreset`	Output	Resets the RX PMA when asserted high.
`rx_digitalreset`	Output	Resets the RX PCS when asserted high.
`clock`	Input	Clock signal for the user-coded reset controller. You can use the system clock without synchronizing it to the PHY parallel clock. The upper limit on the input clock frequency is the frequency achieved in timing closure.
`pll_cal_busy`	Input	A high on this signal indicates the PLL is being calibrated.
`pll_locked`	Input	A high on this signal indicates that the TX PLL is locked to the ref clock.
`tx_cal_busy`	Input	A high on this signal indicates that TX calibration is active. If you have multiple PLLs, you can OR their `pll_cal_busy` signals together.
`rx_is_lockedtodata`	Input	A high on this signal indicates that the RX CDR is in the lock-to-data (LTD) mode.
`rx_cal_busy`	Input	A high on this signal indicates that RX calibration is active.
`rx_is_lockedtoref`	Input	A high on this signal indicates that the RX CDR is in the lock-to-reference (LTR) mode. This signal may toggle or be deasserted when the CDR is in LTD mode.

4.6. Combining Status or PLL Lock Signals

You can combine multiple PHY status signals before feeding into the reset controller as shown below.

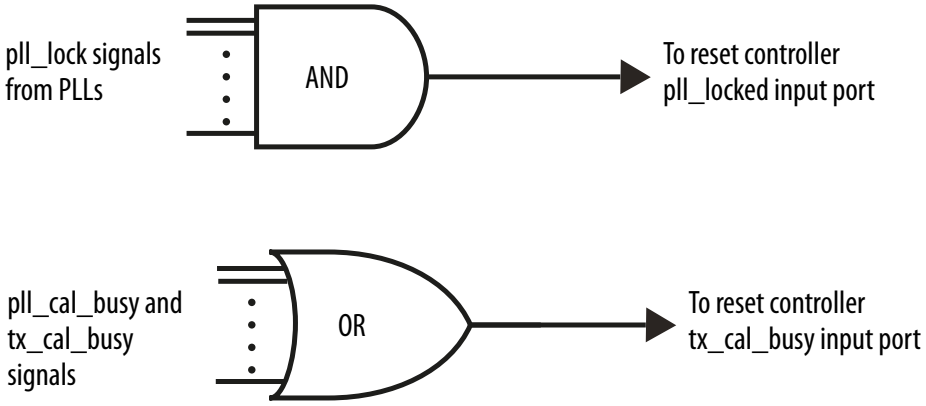
Figure 164. Combining Multiple PHY Status Signals



Note: This configuration also applies to the `rx_cal_busy` signals.

When using multiple PLLs, you can logical AND the `pll_locked` signals feeding the reset controller. Similarly, you can logical OR the `pll_cal_busy` signals to the reset controller `tx_cal_busy` port as shown below.

Figure 165. Multiple PLL Configuration



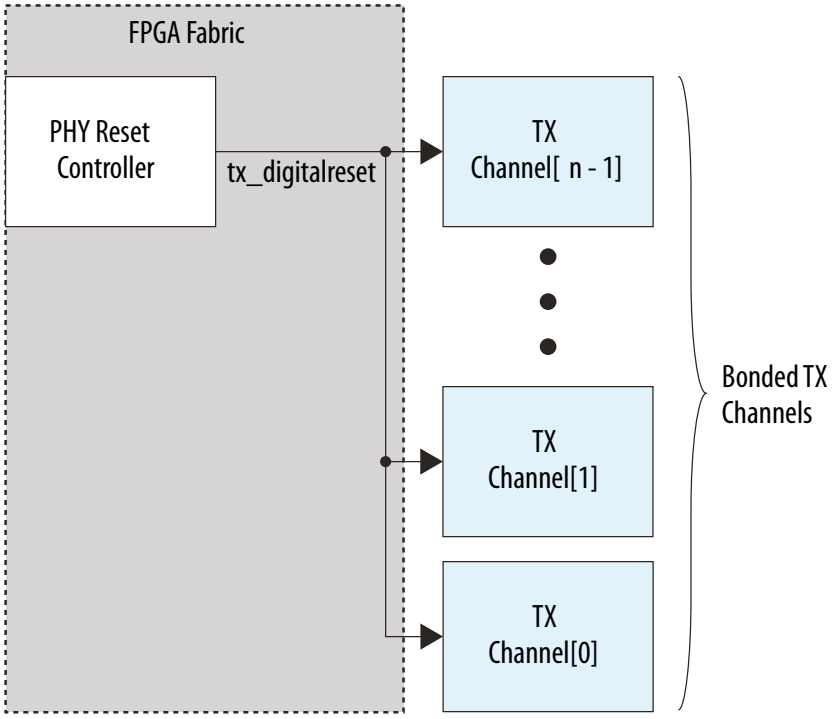
Resetting different channels separately requires multiple reset controllers. For example, a group of channels configured for Interlaken requires a separate reset controller from another group of channels that are configured for optical communication.

4.7. Timing Constraints for Bonded PCS and PMA Channels

For designs that use TX **PMA and PCS Bonding**, the digital reset signal (`tx_digitalreset`) to all TX channels within a bonded group must meet a maximum skew tolerance imposed by physical routing. This skew tolerance is one-half the TX parallel clock cycle (`tx_clkout`). This requirement is not necessary for TX **PMA Bonding** or for RX PCS channels.

Note: If the design is not able to meet the maximum skew tolerance requirement with a positive margin, Intel recommends reassigning the channels' locations that are not adjacent to the PCIe Hard IP block.

Figure 166. Physical Routing Delay Skew in Bonded Channels



You must provide a Synopsys Design Constraint (SDC) for the reset signals to guarantee that your design meets timing requirements. The Quartus Prime software generates an `.sdc` file when you generate the Transceiver Native PHY IP core.

This `.sdc` contains basic false paths for most asynchronous signals, including resets. In the case of bonded designs, this file contains examples for maximum skew on bonded designs. This `.sdc` file contains an example `false_path` and an example `max_skew` constraint for the `tx_digitalreset` signals.

All modified IP constraints from a generated `.sdc` file must be moved to the project's main `.sdc` file, because changes are lost if the IP is regenerated.

This skew is present whether you tie all `tx_digitalresets` together, or you control them separately. If your design includes the Transceiver PHY Reset Controller IP core, you can substitute your instance and interface names for the generic names shown in the example.

Example 1. SDC Constraint for TX Digital Reset When Bonded Clocks Are Used

```
set_max_skew -from *<IP_INSTANCE_NAME> *tx_digitalreset*r_reset
-to *pld_pcs_interface* <1/2 coreclk period in ps>
```

In the above example, you must make the following substitutions:

- *<IP_INSTANCE_NAME>*—substitute the name of your reset controller IP instance or PHY IP instance
- *<½ coreclk period in ps>*—substitute half of the clock period of your design in picoseconds

If your design has custom reset logic, replace the *<IP_INSTANCE_NAME>*tx_digitalreset*r_reset with the source register for the TX PCS reset signal, tx_digitalreset.

For more information about the set_max_skew constraint, refer to the *SDC and Timing Analyzer API Reference Manual*.

Related Information

SDC and TimeQuest API Reference Manual

4.8. Resetting Transceiver Channels Revision History

Document Version	Changes
2017.11.06	Made the following changes: • Added a note "If the design is not able to meet the maximum skew tolerance requirement with a positive margin, Intel recommends reassigning the channels locations that are not adjacent to the PCIe Hard IP block."
2017.05.08	Initial release.

5. Cyclone 10 GX Transceiver PHY Architecture

5.1. Cyclone 10 GX PMA Architecture

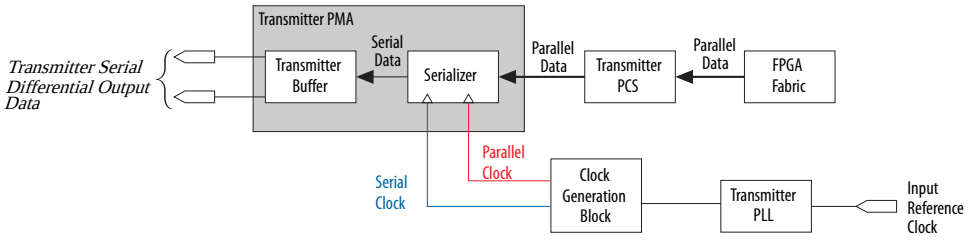
The Physical Medium Attachment (PMA) acts as the analog front end for the Cyclone 10 GX transceivers.

The PMA receives and transmits high-speed serial data depending on the transceiver channel configuration. All serial data transmitted and received passes through the PMA.

5.1.1. Transmitter

The transmitter takes the parallel data and serializes it to create a high-speed serial data stream. The transmitter portion of the PMA is composed of the transmitter serializer and the transmitter buffer. The serializer clock is provided from the transmitter PLL.

Figure 167. Transmitter PMA Block Diagram



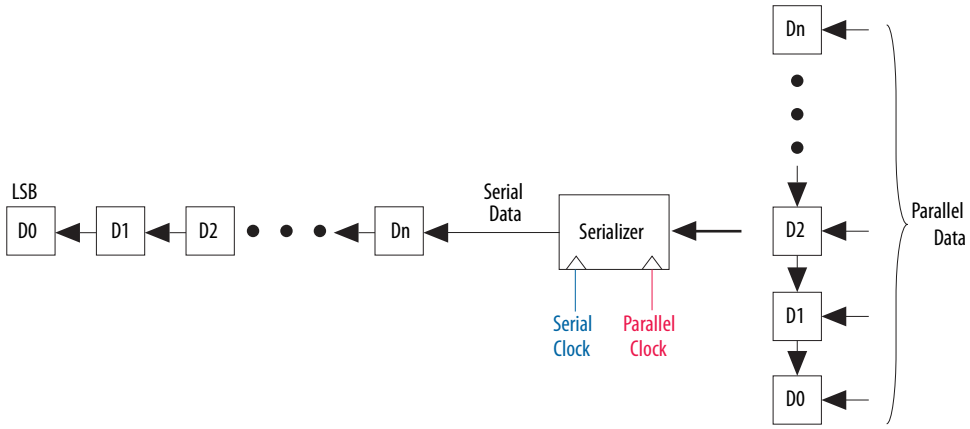
5.1.2. Serializer

The serializer converts the incoming low-speed parallel data from the transceiver PCS or FPGA fabric to high-speed serial data and sends the data to the transmitter buffer.

The channel serializer supports the following serialization factors: 8, 10, 16, 20, 32, 40, and 64.

Figure 168. Serializer Block

The serializer block sends out the least significant bit (LSB) of the input data first.

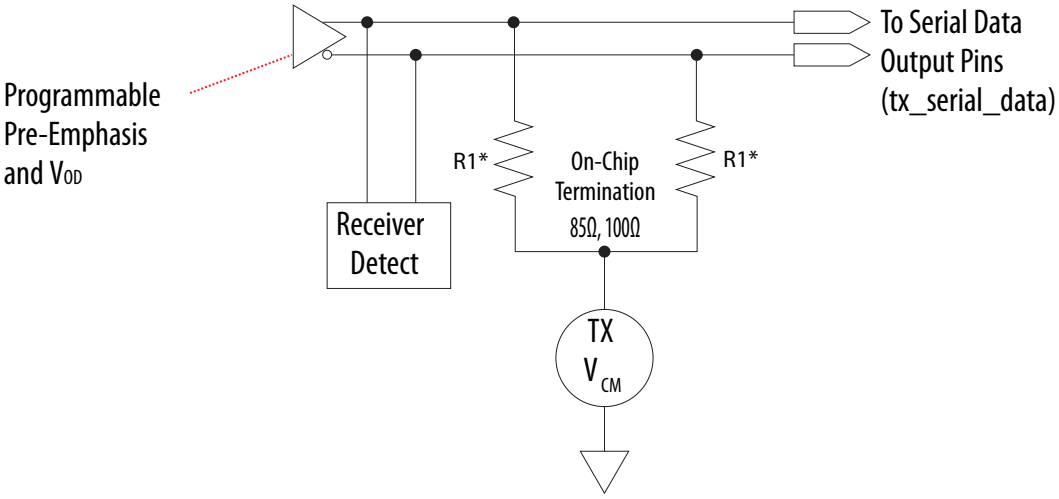


5.1.3. Transmitter Buffer

The transmitter buffer includes the following circuitry:

- High Speed Differential I/O
- Programmable differential output voltage (V_{OD})
  - Main tap
- Programmable four-tap pre-emphasis circuitry
  - Two pre-cursor taps
  - Two post-cursor taps
- Power distribution network (PDN) induced inter-symbol interference (ISI) compensation
- Internal termination circuitry
- Receiver detect capability to support the PCI Express configuration

Figure 169. Transmitter Buffer



R1* - Half of the actual on-chip termination selected.

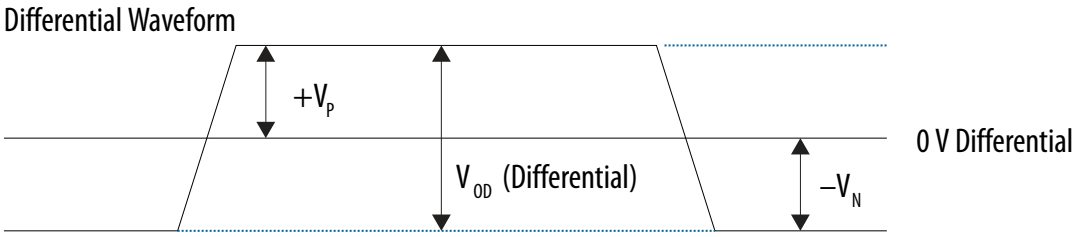
5.1.3.1. High Speed Differential I/O

To improve performance, the Cyclone 10 GX transmitter uses a new architecture in the output buffer—High Speed Differential I/O. You should select "High Speed Differential I/O" for I/O standard of Cyclone 10 GX transmitter pin in Quartus Prime Assignment Editor or QSF file.

5.1.3.2. Programmable Output Differential Voltage

You can program the differential output voltage (output swing) to handle different channel losses and receiver requirements. There are 31 differential V_{OD} settings up to VCCT power supply level. The step size is 1/30 of the VCCT power supply level.

Figure 170. V_{OD} (Differential) Signal Level



$$V_{OD}\ (Differential) = V_P - V_N$$

Related Information

For more information, refer to Cyclone 10 GX Pre-Emphasis and Output Swing Settings

5.1.3.3. Programmable Pre-Emphasis

Pre-emphasis can maximize the eye at the far-end receiver. The programmable pre-emphasis module in each transmit buffer amplifies high frequencies in the transmit data signal, to compensate for attenuation in the transmission media.

The pre-tap pre-emphasizes the bit before the transition and de-emphasizes the remaining bits. A different polarity on pre-tap does the opposite.

Table 163. Pre-Emphasis Taps

All four pre-emphasis taps provide inversion control, shown by negative values.

Pre-Emphasis Tap	Number of Settings	Channel Loss Compensation (dB)
Second pre-tap	15	2.31
First pre-tap	33	6.62
First post-tap	51	15.56
Second post-tap	25	4.44

You can set pre-emphasis taps through the Quartus Assignment Editor, the Avalon-MM registers, and the QSF settings.

Related Information

For more information, refer to Cyclone 10 GX Pre-Emphasis and Output Swing Settings

5.1.3.4. Power Distribution Network (PDN) induced Inter-Symbol Interference (ISI) compensation

The Cyclone 10 GX Transmitter driver includes a compensation circuitry to reduce PDN induced ISI jitter. You can enable this compensation circuitry to reduce jitter through QSF setting, Quartus Assignment Editor or Avalon-MM interface. The power consumption increases when you enable the compensation.

5.1.3.5. Programmable Transmitter On-Chip Termination (OCT)

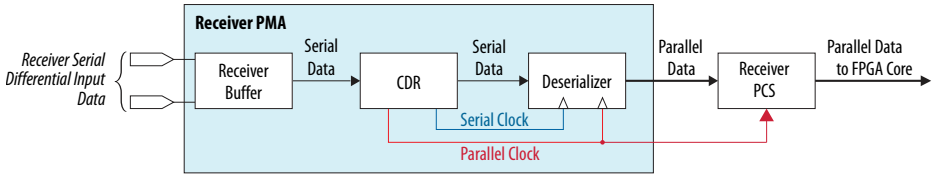
Transmitter buffers include programmable on-chip differential termination of 85Ω or 100Ω. You can set the OCT value through the Quartus Assignment Editor and the Avalon-MM registers.

5.1.4. Receiver

The receiver deserializes the high-speed serial data, creates a parallel data stream for either the receiver PCS or the FPGA fabric, and recovers the clock information from the received data.

The receiver portion of the PMA is comprised of the receiver buffer, the clock data recovery (CDR) unit, and the deserializer.

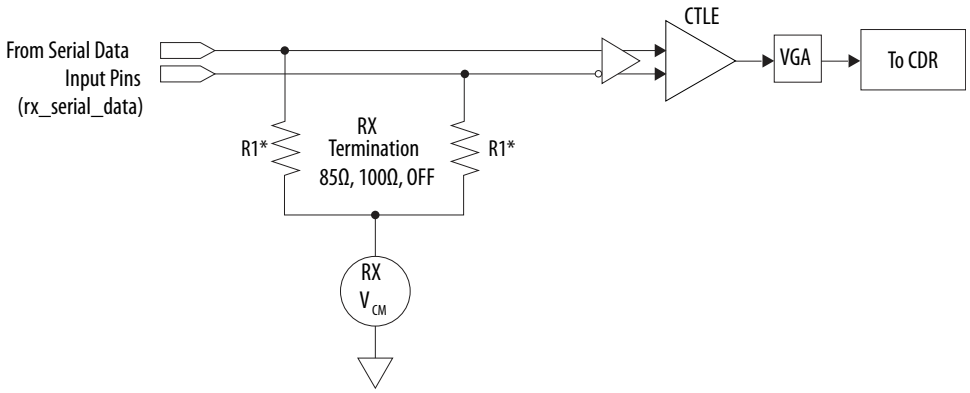
Figure 171. Receiver PMA Block Diagram



5.1.5. Receiver Buffer

The receiver input buffer receives serial data from `rx_serial_data` and feeds the serial data to the clock data recovery (CDR) unit and deserializer.

Figure 172. Receiver Buffer



R1* - Half of the actual on-chip termination selected

The receiver buffer supports the following features:

- Programmable common mode voltage (V_{CM})
- Programmable differential On-Chip Termination (OCT)
- Signal Detector
- Continuous Time Linear Equalization (CTLE)
- Variable Gain Amplifiers (VGA)

5.1.5.1. Programmable Common Mode Voltage (V_{CM})

The receiver buffer has on-chip biasing circuitry to establish the required V_{CM} at the receiver input.

The Quartus Prime software automatically chooses the optimal setting for RX V_{CM}.

Note: On-chip biasing circuitry is available only if you select OCT. If you select external termination, you must implement off-chip biasing circuitry to establish the V_{CM} at the receiver input buffer.

Manual V_{CM} adjustment is not supported and is only adjusted by calibrations.

5.1.5.2. Programmable Differential On-Chip Termination (OCT)

Receiver buffers include programmable on-chip differential termination of 85Ω, 100Ω, or OFF.

You can disable OCT and use external termination. If you select external termination, the receiver common mode is tri-stated. Common mode is based on the external termination connection. You also need to implement off-chip biasing circuitry to establish the V_{CM} at the receiver buffer.

5.1.5.3. Signal Detector

You can enable the optional signal threshold detection circuitry. If 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 assignment editor.

5.1.5.4. Continuous Time Linear Equalization (CTLE)

The CTLE boosts the signal that is attenuated due to channel characteristics. Each receiver buffer has independently programmable equalization circuits. These equalization circuits amplify the high-frequency component of the incoming signal by compensating for the low-pass characteristics of the physical medium. The CTLE can support both DC and AC gain.

DC gain circuitry provides an equal amplification to the incoming signal across the frequency spectrum. AC gain circuitry provides amplification to the high-frequency spectrum gain of the incoming signal. Refer to Figure 173 on page 278.

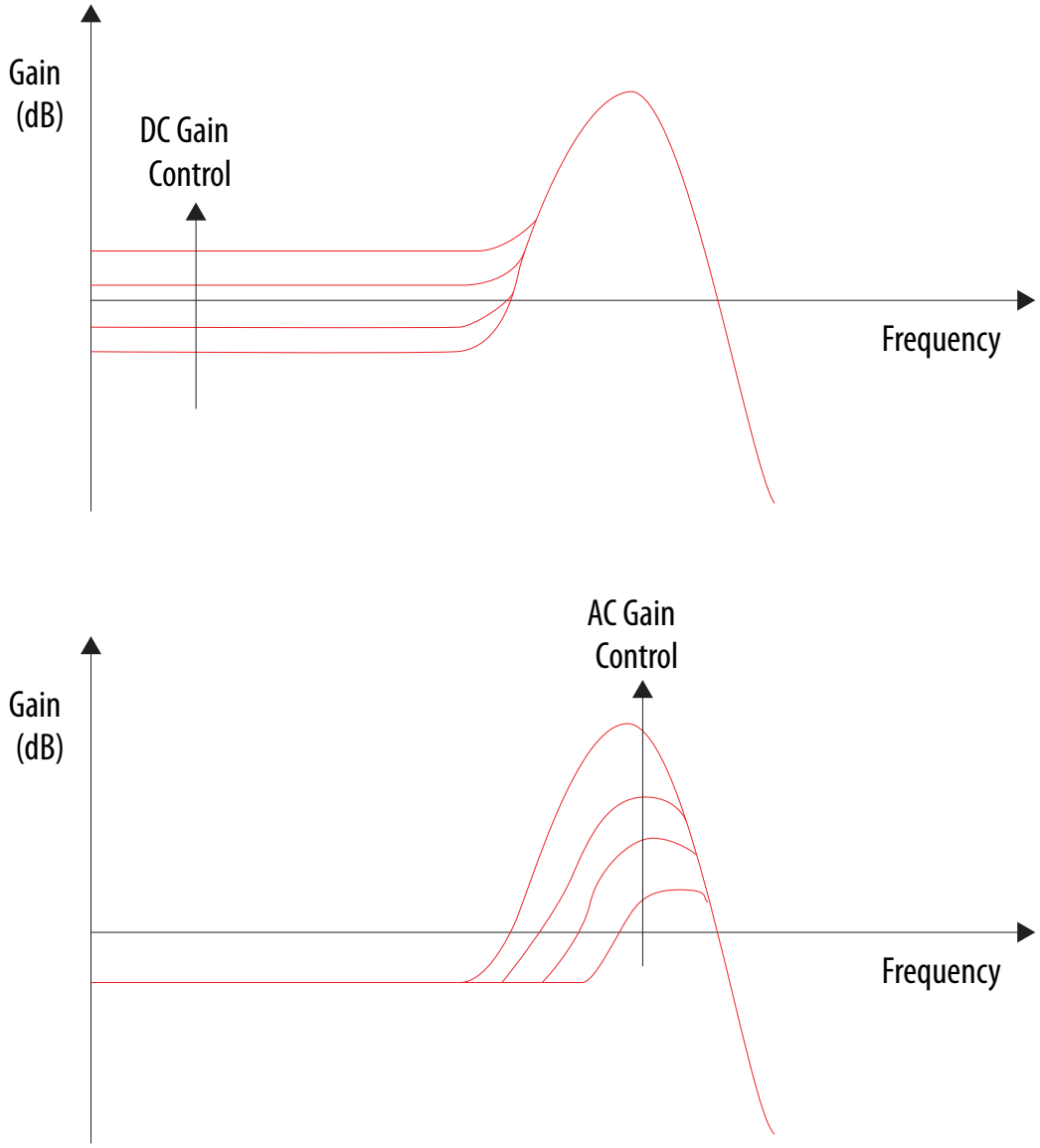
Cyclone 10 GX transceivers support high gain mode CTLE.

5.1.5.4.1. High Gain Mode

This mode provides both AC and DC gain. There are two bandwidth settings available for this mode.

- Full Bandwidth—This mode has a peaking frequency of 6.25 GHz offering AC gain around 15 dB.
- Medium Bandwidth—This mode has a peaking frequency of 3.125 GHz offering AC gain around 20 dB.

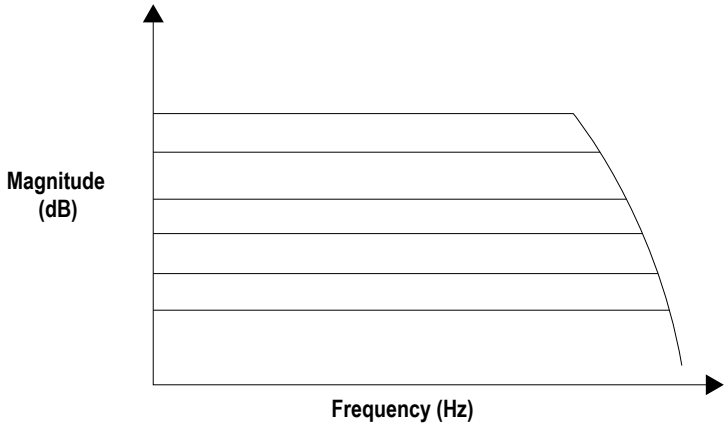
Figure 173. CTLE Frequency Response for DC Gain and AC Gain



5.1.5.5. Variable Gain Amplifier (VGA)

Cyclone 10 GX channels have a variable gain amplifier to optimize the signal amplitude prior to the CDR sampling. VGA can only be operated in manual mode. VGA gain can be selected through Quartus Assignment Editor or Quartus Setting File (qsf) or the Avalon MM register. You must set VGA manually for all combinations of CTLE modes.

Figure 174. VGA Frequency Response for Different Gain Settings



Related Information

How to Enable CTLE on page 279

5.1.5.6. How to Enable CTLE

Table 164. Summary of Receiver Equalization Modes

Receiver Equalization	Modes
CTLE adaptation mode	Manual

5.1.5.6.1. Configuration Methods

Configure the modes using one of the following methods:

Method 1 - Using Cyclone 10 GX Transceiver Native PHY IP Core

1. Select the CTLE mode in the RX PMA tab of the PHY IP Core
2. Compile the design
3. If the CTLE is in Manual mode, set the CTLE gain value using the assignment editor/Quartus Settings File (.qsf). Then, recompile the design to make these values effective.

   Refer to *Analog Parameter Settings* for more details about *Receiver Equalization Settings*.

Method 2 - Using AVMM Interface

1. Any changes you make using AVMM interface take precedence over what was configured in Native PHY IP GUI or Assignment Editor.
   a. For CTLE in Manual mode, set the CTLE gain value using the reconfiguration interface. The values are written dynamically and do not require design re-compilation.

      Refer to *Intel Cyclone 10 GX Register Map* for details on the specific registers that set the CTLE gain values.

   Note: You must set VGA manually for all combinations of CTLE mode.

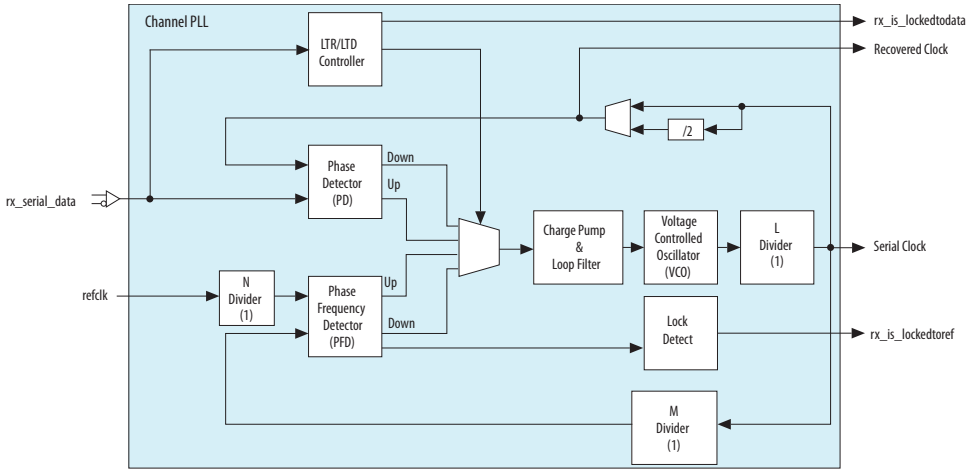
Related Information

- Analog Parameter Settings on page 388
- Intel Cyclone 10 GX Register Map

5.1.6. Clock Data Recovery (CDR) Unit

The PMA of each channel includes a channel PLL that you can configure as a receiver clock data recovery (CDR) for the receiver. You can also configure the channel PLL of channels 1 and 4 as a clock multiplier unit (CMU) PLL for the transmitter in the same bank.

Figure 175. Channel PLL Configured as CDR



Note:
1. The Quartus® Prime Pro Edition software automatically chooses the optimal values.

5.1.6.1. Lock-to-Reference Mode

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

Note: The phase detector (PD) is inactive in LTR mode.

5.1.6.2. Lock-to-Data Mode

During normal operation, the CDR must be in LTD mode to recover the clock from the incoming serial data. In LTD mode, the PD in the CDR tracks the incoming serial data at the receiver input. Depending on the phase difference between the incoming data and the CDR output clock, the PD controls the CDR charge pump that tunes the VCO.

Note: The PFD is inactive in LTD mode. The `rx_is_lockedtoref` status signal toggles randomly and is not significant in LTD mode.

After switching to LTD mode, the `rx_is_lockedtodata` status signal is asserted. The actual lock time depends on the transition density of the incoming data and the parts per million (ppm) difference between the receiver input reference clock and the

upstream transmitter reference clock. The `rx_is_lockedtodata` signal toggles until the CDR sees valid data; therefore, you should hold receiver PCS logic in reset (`rx_digitalreset`) for a minimum of 4 µs after `rx_is_lockedtodata` remains continuously asserted.

5.1.6.3. CDR Lock Modes

You can configure the CDR in either automatic lock mode or manual lock mode. By default, the Quartus Prime software configures the CDR in automatic lock mode.

5.1.6.3.1. Automatic Lock Mode

In automatic lock mode, the CDR initially locks to the input reference clock (LTR mode). After the CDR locks to the input reference clock, the CDR locks to the incoming serial data (LTD mode) when the following conditions are met:

- The signal threshold detection circuitry indicates the presence of valid signal levels at the receiver input buffer when `rx_std_signaldetect` is enabled.
- The CDR output clock is within the configured ppm frequency threshold setting with respect to the input reference clock (frequency locked).
- The CDR output clock and the input reference clock are phase matched within approximately 0.08 unit interval (UI) (phase locked).

If the CDR does not stay locked to data because of frequency drift or severe amplitude attenuation, the CDR switches back to LTR mode.

5.1.6.3.2. Manual Lock Mode

The PPM detector and phase relationship detector reaction times can be too long for some applications that require faster CDR lock time. You can manually control the CDR to reduce its lock time using two optional input ports (`rx_set_locktoref` and `rx_set_locktodata`).

Table 165. Relationship Between Optional Input Ports and the CDR Lock Mode

rx_set_locktoref	rx_set_locktodata	CDR Lock Mode
0	0	Automatic
1	0	Manual-RX CDR LTR
X	1	Manual-RX CDR LTD

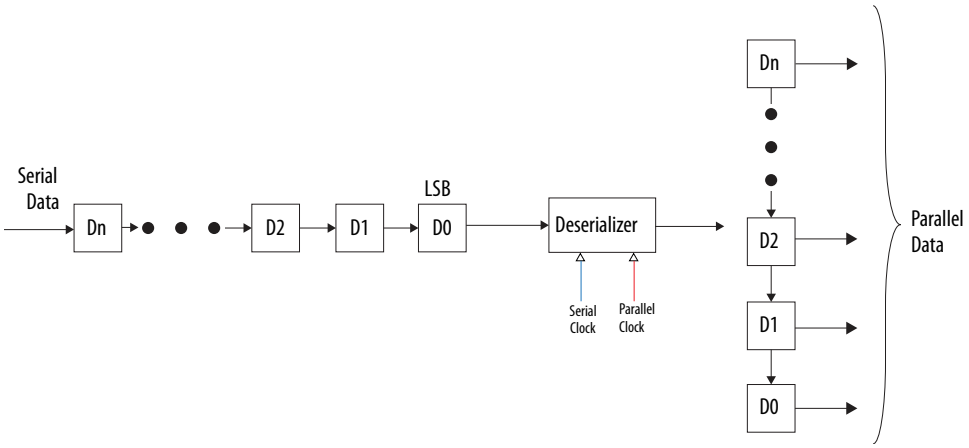
5.1.7. Deserializer

The deserializer block clocks in serial input data from the receiver buffer using the high-speed serial recovered clock and deserializes the data using the low-speed parallel recovered clock. The deserializer forwards the deserialized data to the receiver PCS or FPGA fabric.

The deserializer supports the following deserialization factors: 8, 10, 16, 20, 32, 40, and 64.

Figure 176. Deserializer Block Diagram

The deserializer block sends out the LSB of the input data first.



5.1.8. Loopback

The PMA supports serial, diagnostic, and reverse loopback paths.

Figure 177. Serial Loopback Path

The serial loopback path sets the CDR to recover the data from serializer while data from receiver serial input pin is ignored by CDR. The transmitter buffer sends data normally.

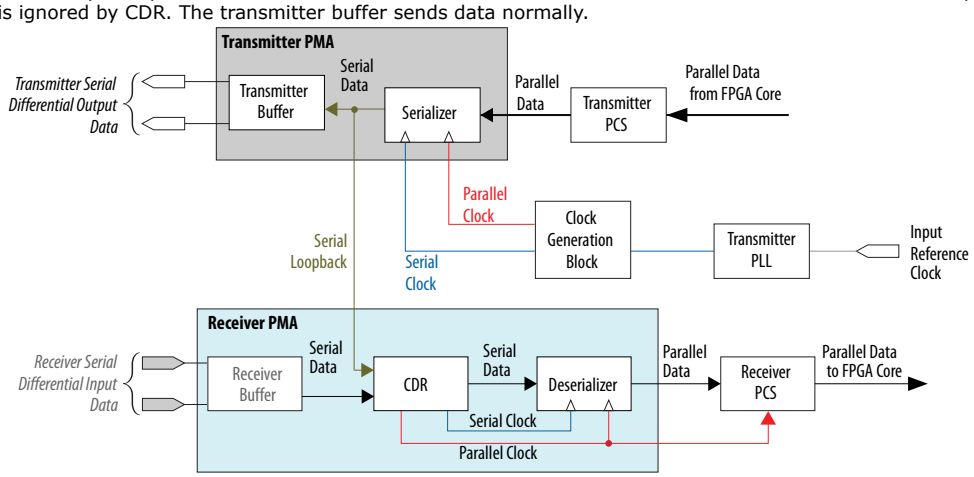
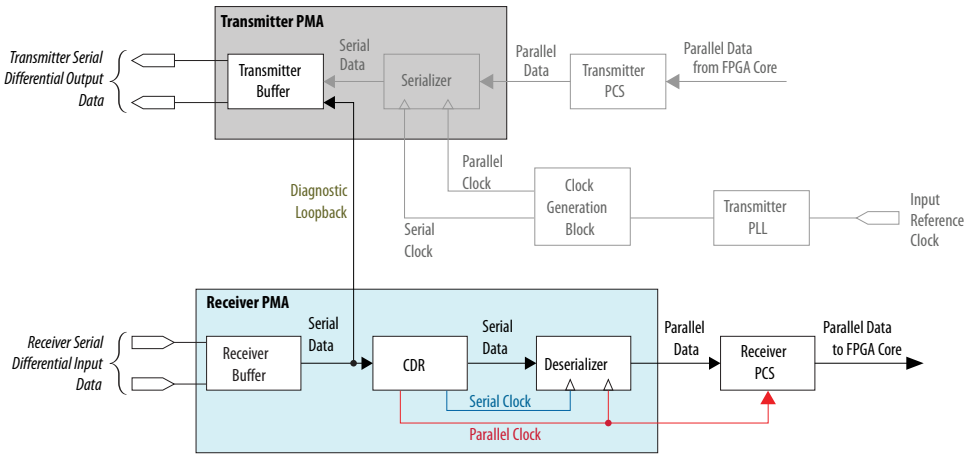


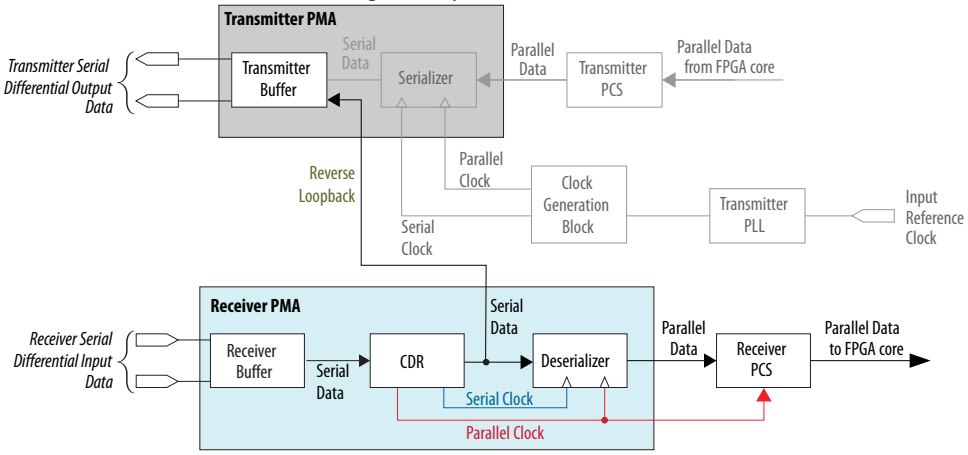
Figure 178. Reverse Serial Loopback Path/Pre CDR



Note: TX pre-emp is not supported in pre-CDR loopback. TX pre-emp is recommended to set to zero for all taps.

Figure 179. Reverse Serial Loopback Path/Post CDR

The reverse serial loopback path sets the transmitter buffer to transmit data fed directly from the CDR recovered data. Data from the serializer is ignored by the transmitter buffer.



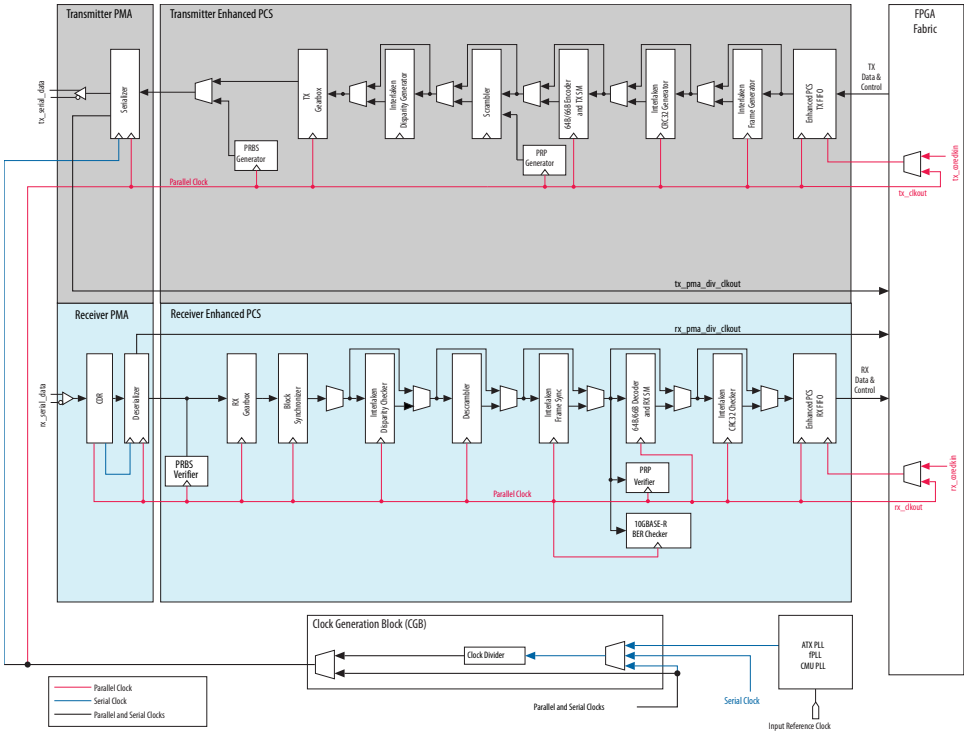
5.2. Cyclone 10 GX Enhanced PCS Architecture

You can use the Enhanced PCS to implement multiple protocols that operate at around 10 Gbps or higher line rates.

The Enhanced PCS provides the following functions:

- Performs functions common to most serial data industry standards, such as word alignment, block synchronization, encoding/decoding, and framing, before data is sent or received off-chip through the PMA
- Handles data transfer to and from the FPGA fabric
- Internally handles data transfer to and from the PMA
- Provides frequency compensation
- Performs channel bonding for multi-channel low skew applications

Figure 180. Enhanced PCS Datapath Diagram



Related Information

Implementing Protocols in Intel Cyclone 10 GX Transceivers on page 16

5.2.1. Transmitter Datapath

5.2.1.1. Enhanced PCS TX FIFO

The Enhanced PCS TX FIFO provides an interface between the transmitter channel PCS and the FPGA fabric. The TX FIFO can operate for phase compensation between the channel PCS and FPGA fabric. You can also use the TX FIFO as an elastic buffer to control the input data flow, using `tx_enh_data_valid`. The TX FIFO also allows channel bonding. The TX FIFO has a width of 73 bits and a depth of 16 words.

You can set the TX FIFO partially full and empty thresholds through the Transceiver and PLL Address Map. Refer to the *Reconfiguration Interface and Dynamic Reconfiguration* chapter for more details.

The TX FIFO supports the following operating modes:

- Phase Compensation mode
- Register mode
- Interlaken mode
- Basic mode

Related Information

Reconfiguration Interface and Dynamic Reconfiguration on page 315

5.2.1.1.1. Phase Compensation Mode

In Phase Compensation mode, the TX Core FIFO decouples phase variations between `tx_coreclkin` and `PCS_clkout_x2(tx)`. In this mode, read and write of the TX Core FIFO can be driven by clocks from asynchronous clock sources but must be same frequency. You can use `tx_coreclkin` (FPGA fabric clock) or `tx_clkout1` (TX parallel clock) to clock the write side of the TX Core FIFO.

Note: Phase Compensation mode, TX parallel data is valid for every low speed clock cycle, and `tx_enh_data_valid` signal should be tied with `1'b1`.

Note: Phase Compensation can also be used in double rate transfer mode, where the FPGA fabric data width is doubled to allow the FPGA fabric clock to run at half rate. The double rate transfer mode is set in the Native PHY IP Parameter Editor. Refer to the "Transmitter Data Path Interface Clocking" and "Receiver Data Path Interface Clocking" sections in the *PLLs and Clock Networks* chapter for details about the clock frequencies, when using FIFO single and double rate transfer mode.

Related Information

- Transmitter Data Path Interface Clocking on page 219
- Receiver Data Path Interface Clocking on page 220

5.2.1.1.2. Register Mode

The Register Mode bypasses the FIFO functionality to eliminate the FIFO latency uncertainty for applications with stringent latency requirements. This is accomplished by tying the read clock of the FIFO with its write clock.

In Register mode, `tx_parallel_data` (data), `tx_control` (indicates whether `tx_parallel_data` is a data or control word), and `tx_enh_data_valid` (data valid) are registered at the FIFO output.

Note: Intel recommends that you implement a soft FIFO in the FPGA fabric with a minimum of 32 words under the following conditions:

- When the Enhanced PCS TX FIFO is set to register mode.
- When using the recovered clock to drive the core logic.
- When there is no soft FIFO being generated along with the IP Catalog.

5.2.1.1.3. Interlaken Mode

In Interlaken mode, the TX Core FIFO operates as an elastic buffer. In this mode, you have additional signals to control the data flow into the FIFO. Therefore, the FIFO write clock frequency does not have to be the same as the read clock frequency. You control the writing to the TX Core FIFO with `tx_fifo_wr_en` by monitoring the FIFO flags. The goal is to prevent the FIFO from becoming full or empty. On the read side, read enable is controlled by the Interlaken frame generator.

5.2.1.1.4. Basic Mode

In Basic mode, the TX Core FIFO operates as an elastic buffer, where buffer depths can vary. This mode allows driving write and read side of TX Core FIFO with different clock frequencies. Monitor the FIFO flag to control write and read operations. For TX Core FIFO, assert `tx_fifo_wr_en` with `tx_fifo_pempty` signal going low.

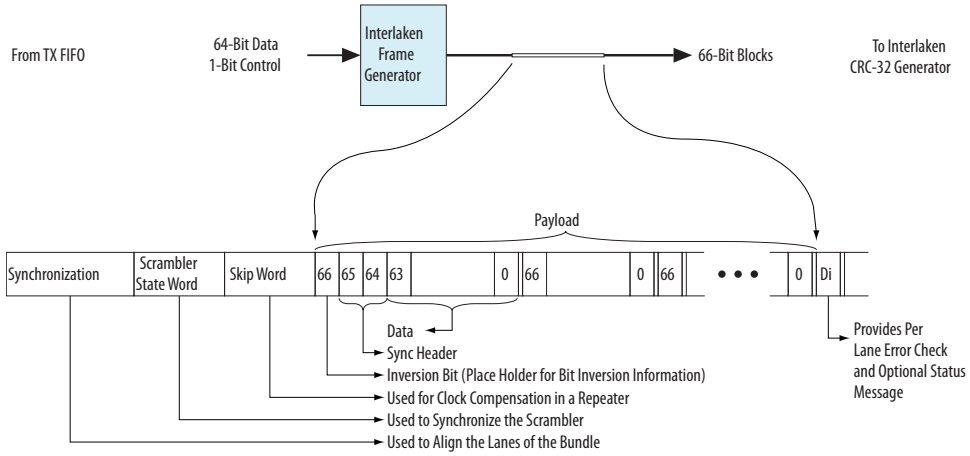
5.2.1.2. Interlaken Frame Generator

The Interlaken frame generator block takes the data from the TX FIFO and encapsulates the payload and burst/idle control words from the FPGA fabric with the framing layer's control words (synchronization word, scrambler state word, skip word, and diagnostic word) to form a metaframe. The Native PHY IP Parameter Editor allows you to set the metaframe length from five 8-byte words to a maximum value of 8192 (64Kbyte words).

Use the same value on frame generator metaframe length for the transmitter and receiver.

Figure 181. Interlaken Frame Generator

The Interlaken frame generator implements the Interlaken protocol.



5.2.1.3. Interlaken CRC-32 Generator

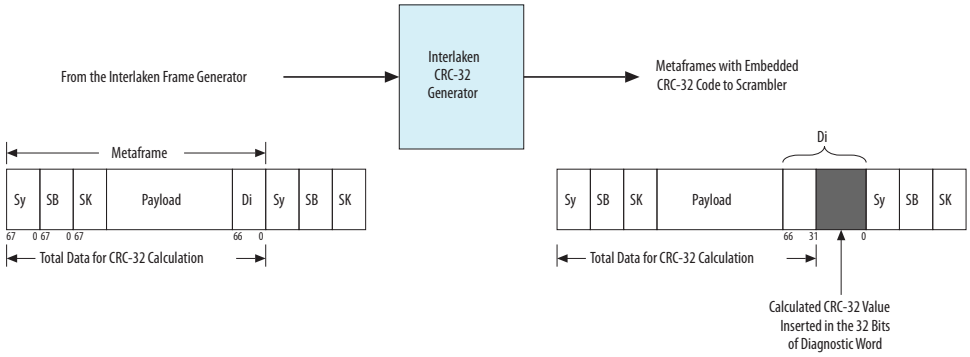
The Interlaken CRC-32 generator block receives data from the Interlaken frame generator and calculates the cyclic redundancy check (CRC) code for each block of data. This CRC code value is stored in the CRC32 field of the diagnostic word. CRC-32 provides a diagnostic tool for each lane. This helps to trace the errors on the interface back to an individual lane.

The CRC-32 calculation covers most of the metaframe, including the diagnostic word, except the following:

- Bits [66:64] of each word
- 58-bit scrambler state within the scrambler state word
- 32-bit CRC-32 field within the diagnostic word

Figure 182. Interlaken CRC-32 Generator

The Interlaken CRC-32 generator implements the Interlaken protocol.



5.2.1.4. 64B/66B Encoder and Transmitter State Machine (TX SM)

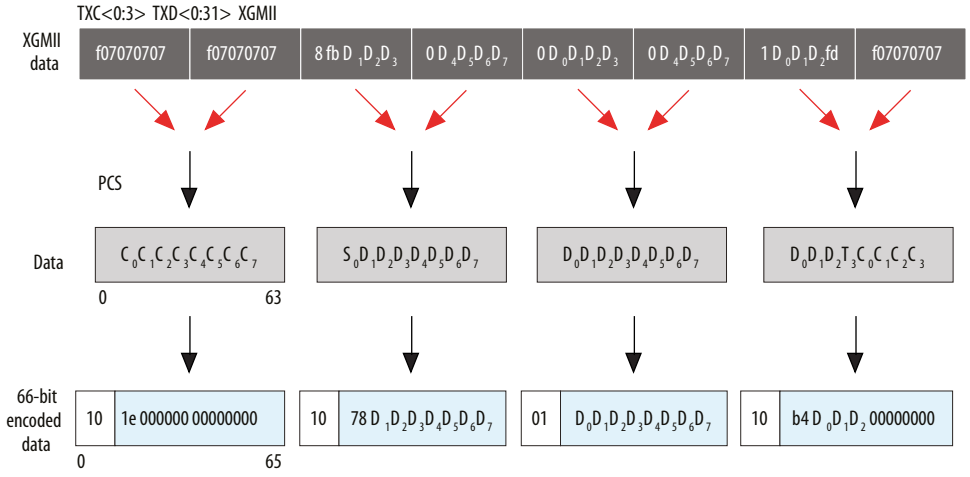
The 64B/66B encoder is used to achieve DC-balance and sufficient data transitions for clock recovery. It encodes 64-bit XGMII data and 8-bit XGMII control into 10GBASE-R 66-bit control or data blocks in accordance with Clause 49 of the IEEE802.3-2008 specification.

The 66-bit encoded data contains two overhead sync header bits that the receiver PCS uses for block synchronization and bit-error rate (BER) monitoring. The sync header is 01 for data blocks and 10 for control blocks. Sync headers are not scrambled and are used for block synchronization. (The sync headers 00 and 11 are not used, and generate an error if seen.) The remainder of the block contains the payload. The payload is scrambled and the sync header bypasses the scrambler.

The encoder block also has a state machine (TX SM) designed in accordance with the IEEE802.3-2008 specification. The TX SM ensures valid packet construction on data sent from the MAC layer. It also performs functions such as transmitting local faults under reset, as well as transmitting error codes when the 10GBASE-R PCS rules are violated.

Note: The 64B/66B encoder is available to implement the 10GBASE-R protocol.

Figure 183. Example Data Pattern for 64B/66B Encoding



64B/66B Encoder Reset Condition

The `tx_digitalreset` signal resets the 64B/66B encoder. During the reset condition, the 64B/66B encoder does not output any signal in contrast with the 8B/10B encoder.

5.2.1.5. Pattern Generators

The Cyclone 10 GX transceivers contain hardened generators and checkers to provide a simple and easy way to verify and characterize high speed links. Hardening the pattern generators and checkers save FPGA core logic resources. The following pattern generator blocks are supported in Cyclone 10 GX:

- Pseudo Random Binary Sequence (PRBS)
- Pseudo Random Pattern (PRP)

Note: The pattern generators and checkers are supported for non-bonded channels only.

The pattern generators or checkers are enabled by writing to the respective register bits of the Transceiver. Refer to the *Reconfiguration Interface and Dynamic Reconfiguration* chapter for configuration details.

Related Information

Reconfiguration Interface and Dynamic Reconfiguration on page 315

5.2.1.5.1. PRBS Pattern Generator (Shared between Enhanced PCS and Standard PCS)

You can use Cyclone 10 GX pseudo-random bit sequence PRBS generator to simulate traffic without developing or fully implementing any upper layer of a protocol stack. The PRBS generator in Cyclone 10 GX devices is a shared hardened block between the Standard and Enhanced datapaths through the PCS instead of being two unique instances: one for Standard PCS and one for the Enhanced PCS. There is only one set of control signals and registers for using this feature. The data lines from the various PCSes and shared PRBS, are muxed before they are sent to the PMA. When the PRBS generator is enabled, the data on the PRBS data lines is selected to be sent to the PMA. At any instant, either the data from the PCS or the data generated from the PRBS generator, is sent to the PMA.

The PRBS generator can be configured for two widths of the PCS-PMA interface: 10 bits and 64 bits. PRBS9 is available in 10-bit and 64-bit PCS-PMA widths. All other PRBS patterns are available in 64-bit PCS-PMA width only. The PRBS generator patterns can only be used when PCS-PMA interface width is configured to 10 bits or 64 bits.

Table 166. Supported PRBS Patterns

PRBS Pattern	10 bit PCS-PMA width	64 bit PCS-PMA width
PRBS7: $x^7 + x^6 + 1$		Yes
PRBS9: $x^9 + x^5 + 1$	Yes	Yes
PRBS15: $x^{15} + x^{14} + 1$		Yes
PRBS23: $x^{23} + x^{18} + 1$		Yes
PRBS31: $x^{31} + x^{28} + 1$		Yes

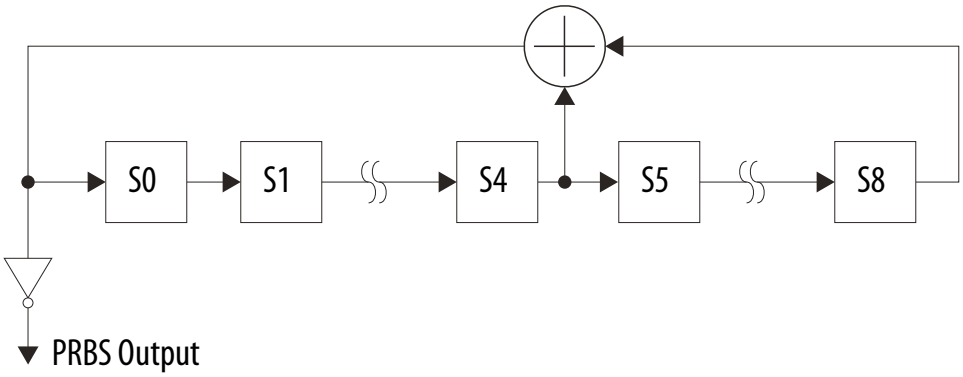
PRBS test patterns may be considered equivalent to "noise". Use these patterns to test the transceiver link with a noisy signal by placing the transceiver in loopback mode.

Use PRBS7 and PRBS9 to test transceiver links with linear impairments, and with 8B/10B.

Use PRBS15 for jitter evaluation.

Use PRBS23 or PRBS31 for jitter evaluation (data-dependent jitter) of non-8B/10B links, such as SDH/SONET/OTN jitter testers. Most 40G, 100G, and 10G applications use PRBS31 for link evaluation.

Figure 184. PRBS Generator for Serial Implementation of PRBS9 Pattern



Note: All supported PRBS generators are similar to the PRBS9 generator.

Refer to the *Reconfiguration Interface and Dynamic Reconfiguration* chapter for configuration details.

Related Information

Reconfiguration Interface and Dynamic Reconfiguration on page 315

5.2.1.5.2. Pseudo-Random Pattern Generator

The pseudo-random pattern (PRP) generator is specifically designed for the 10GBASE-R and 1588 protocols. The PRP generator block operates in conjunction with the scrambler to generate pseudo-random patterns for the TX and RX tests in the 10G Ethernet mode. You can use the Cyclone 10 GX Pseudo Random Pattern (PRP) generator in the scrambler to generate random data pattern and seed that the scrambler can use. The PRP mode is a test mode of the scrambler. Two seeds are available to seed the scrambler: all 0s or two local fault-ordered sets. The seed is used in the scrambler to produce the pattern. PRP is only available when the scrambler is enabled.

Refer to the *Reconfiguration Interface and Dynamic Reconfiguration* chapter for configuration details.

Related Information

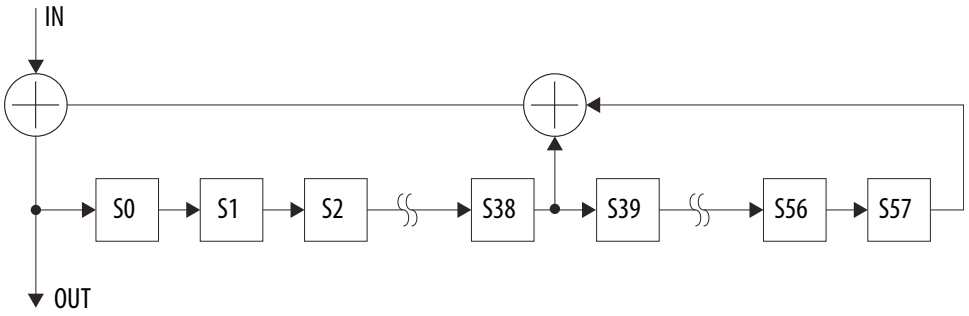
Reconfiguration Interface and Dynamic Reconfiguration on page 315

5.2.1.6. Scrambler

The scrambler randomizes data to create transitions to DC-balance the signal and help CDR circuits. The scrambler uses a $x^{58} + x^{39} + 1$ polynomial and supports both synchronous scrambling used for Interlaken and asynchronous (also called self-synchronized) scrambling used for the 10GBASE-R protocol.

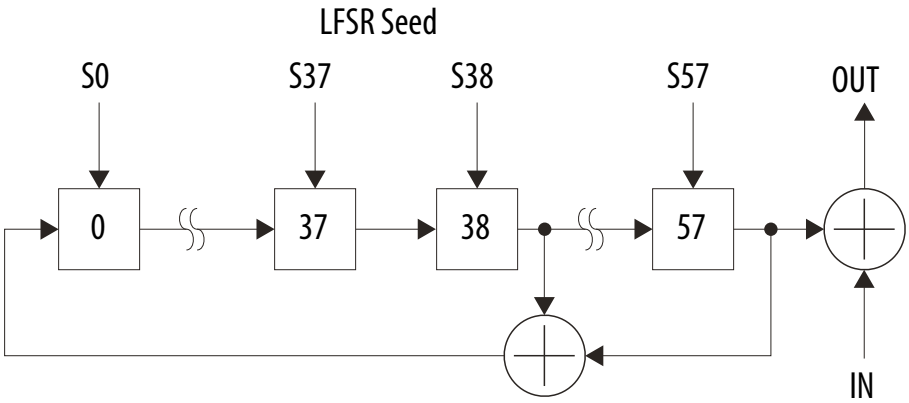
The asynchronous (self-synchronizing) mode does not require an initialization seed. Except for the two sync header bits in each 66-bit data block, the entire 64-bit payload is scrambled by feeding it into a linear feedback shift register (LFSR) continuously to generate scrambled data while the sync-header bits bypass the scrambler. The initial seed is set to all 1s. You can change the seed for the 10GBASE-R protocol using the Native PHY IP Parameter Editor.

Figure 185. Asynchronous Scrambler in Serial Implementation



In synchronous mode, the scrambler is initially reset to different programmable seeds on each lane. The scrambler then runs by itself. Its current state is XOR'd with the data to generate scrambled data. A data checker in the scrambler monitors the data to determine if it should be scrambled or not. If a synchronization word is found, it is transmitted without scrambling. If a Scrambler State Word is detected, the current scramble state is written into the 58-bit scramble state field in the Scrambler State Word and sent over the link. The receiver uses this scramble state to synchronize the descrambler. The seed is automatically set for Interlaken protocol.

Figure 186. Synchronous Scrambler Showing Different Programmable Seeds



5.2.1.7. Interlaken Disparity Generator

The Interlaken disparity generator block is in accordance with the Interlaken protocol specification and provides a DC-balanced data output.

The Interlaken protocol solves the unbounded baseline wander, or DC imbalance, of the 64B/66B coding scheme used in 10Gb Ethernet by inverting the transmitted data. The disparity generator monitors the transmitted data and makes sure that the running disparity always stays within a ±96-bit bound. It adds the 67th bit (bit 66) to signal the receiver whether the data is inverted or not.

Table 167. Inversion Bit Definition

Bit 66	Interpretation
0	Bits [63:0] are not inverted; the receiver processes this word without modification
1	Bits [63:0] are inverted; the receiver inverts the bits before processing this word

Note: The Interlaken disparity generator is available to implement the Interlaken protocol.

5.2.1.8. TX Gearbox, TX Bitslip and Polarity Inversion

The TX gearbox adapts the PCS data width to the smaller bus width of the PCS-PMA interface (Gearbox Reduction). It supports different ratios (FPGA fabric-PCS Interface Width: PCS-PMA Interface Width) such as 66:32, 66:40, 64:32, 40:40, 32:32, 64:64, 67:64, and 66:64. The gearbox mux selects a group of consecutive bits from the input data bus depending on the gearbox ratio and the data valid control signals.

The TX gearbox also has a bit slipping feature to adjust the data skew between channels. The TX parallel data is slipped on the rising edge of `tx_enh_bitslip` before it is passed to the PMA. The maximum number of the supported bitslips is PCS data width-1 and the slip direction is from MSB to LSB and from current to previous word.

Figure 187. TX Bitslip

`tx_enh_bitslip` = 2 and PCS width of gearbox is 67

	Current Word							Previous Word						
Bit Index	66	65	64	...	2	1	0	66	65	64	...	2	1	0

You can use transmitter data polarity inversion to invert the polarity of every bit of the input data word to the serializer in the transmitter path. The inversion has the same effect as swapping the positive and negative signals of the differential TX buffer. This is useful if these signals are reversed on the board or backplane layout. Enable polarity inversion through the Native PHY IP Parameter Editor.

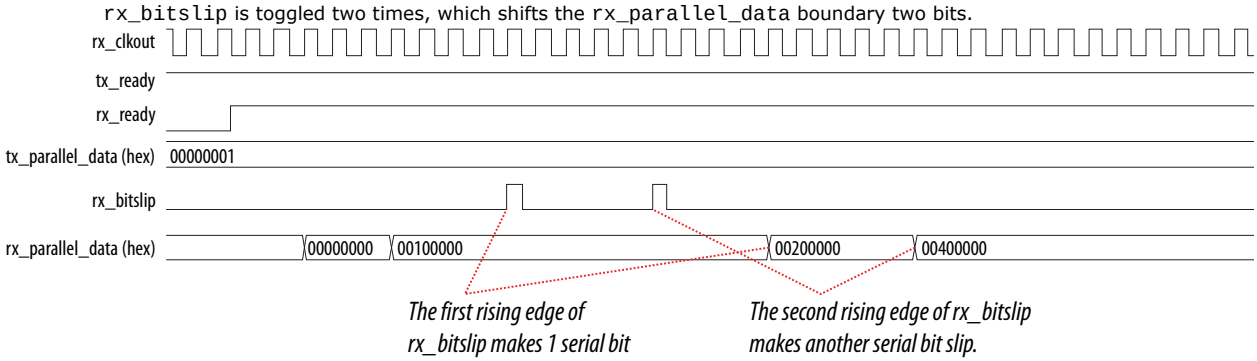
5.2.2. Receiver Datapath

5.2.2.1. RX Gearbox, RX Bitslip, and Polarity Inversion

The RX gearbox adapts the PMA data width to the larger bus width of the PCS channel (Gearbox Expansion). It supports different ratios (PCS-PMA interface width : FPGA fabric–PCS interface width) such as 32:66, 40:66, 32:67, 32:64, 40:40, 32:32, 64:64, 67:64, and 66:64 and a bit slipping feature.

RX bitslip is engaged when the RX block synchronizer or `rx_bitslip` is enabled to shift the word boundary. On the rising edge of the bitslip signal of the RX block synchronizer or `rx_bitslip` from the FPGA fabric, the word boundary is shifted by one serial bit or 1UI. Each bit slip removes the earliest received bit from the received data.

Figure 188. RX Bitslip



The receiver gearbox can invert the polarity of the incoming data. This is useful if the receiver signals are reversed on the board or backplane layout. Enable polarity inversion through the Native PHY IP Parameter Editor.

Data valid generation logic is essential for gearbox operation. Each block of data is accompanied by `rx_enh_data_valid` data valid signal which "qualifies" the block as valid or not. The data valid toggling pattern is dependent on the data width conversion ratio. For example, if the ratio is 66:40, the data valid signal is high in 20 out of 33 cycles or approximately 2 out of 3 cycles and the pattern repeats every 33 `rx_clkout` RX low-speed parallel clock cycles.

5.2.2.2. Block Synchronizer

The block synchronizer determines the block boundary of a 66-bit word in the case of the 10GBASE-R protocol or a 67-bit word in the case of the Interlaken protocol. The incoming data stream is slipped one bit at a time until a valid synchronization header (bits 65 and 66) is detected in the received data stream. After the predefined number of synchronization headers (as required by the protocol specification) is detected, the block synchronizer asserts `rx_enh_blk_lock` (block lock status signal) to other receiver PCS blocks down the receiver datapath and to the FPGA fabric.

Note: The block synchronizer is designed in accordance with Interlaken Protocol specification (as described in Figure 13 of Interlaken Protocol Definition v1.2) and 10GBASE-R protocol specification (as described in IEEE 802.3-2008 clause-49).

5.2.2.3. Interlaken Disparity Checker

The Interlaken disparity checker examines the received inversion bit inserted by the far end disparity generator, to determine whether to reverse the inversion process of the Interlaken disparity generation.

Note: The Interlaken disparity checker is available to implement the Interlaken protocol.

5.2.2.4. Descrambler

The descrambler block descrambles received data to regenerate unscrambled data using the $x^{58} + x^{39} + 1$ polynomial. Like the scrambler, it operates in asynchronous mode or synchronous mode.

Related Information

Scrambler on page 290

5.2.2.5. Interlaken Frame Synchronizer

The Interlaken frame synchronizer delineates the metaframe boundaries and searches for each of the framing layer control words: Synchronization, Scrambler State, Skip, and Diagnostic. When four consecutive synchronization words have been identified, the frame synchronizer achieves the frame locked state. Subsequent metaframes are then checked for valid synchronization and scrambler state words. If four consecutive invalid synchronization words or three consecutive mismatched scrambler state words are received, the frame synchronizer loses frame lock. In addition, the frame synchronizer provides `rx_enh_frame_lock` (receiver metaframe lock status) to the FPGA fabric.

Note: The Interlaken frame synchronizer is available to implement the Interlaken protocol.

5.2.2.6. 64B/66B Decoder and Receiver State Machine (RX SM)

The 64B/66B decoder reverses the 64B/66B encoding process. The decoder block also contains a state machine (RX SM) designed in accordance with the IEEE802.3-2008 specification. The RX SM checks for a valid packet structure in the data sent from the remote side. It also performs functions such as sending local faults to the Media Access Control (MAC)/Reconciliation Sublayer (RS) under reset and substituting error codes when the 10GBASE-R and 10GBASE-KR PCS rules are violated.

Note: The 64B/66B decoder is available to implement the 10GBASE-R protocol.

5.2.2.6.1. PRBS Checker

You can use Cyclone 10 GX pseudo-random bit stream (PRBS) checker to easily characterize high-speed links without developing or fully implementing any upper layer of a protocol stack. The PRBS checker in Cyclone 10 GX devices is a shared hardened block between the Standard and Enhanced datapaths. Hence, there is only one set of control signals and registers for this feature.

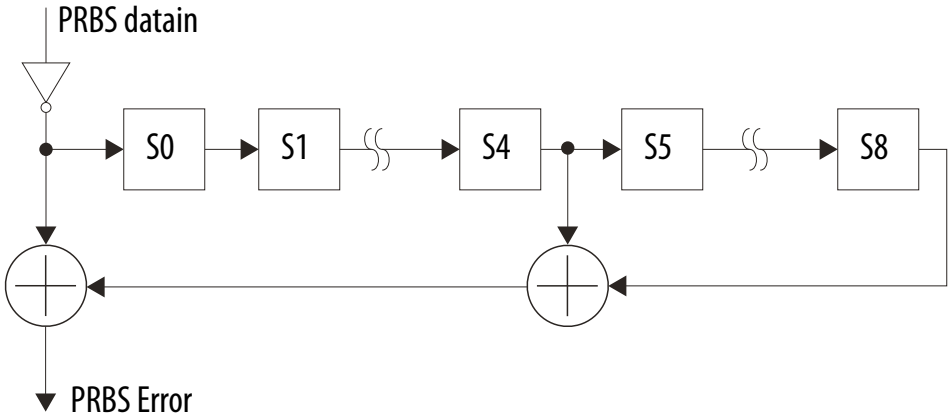
You can use the PRBS checker block to verify the pattern generated by the PRBS generator. The PRBS checker can be configured for two widths of the PCS-PMA interface: 10 bits and 64 bits. PRBS9 is available in both 10-bit and 64-bit PCS-PMA widths. All other PRBS patterns are available in 64-bit PCS-PMA width only. The PRBS checker patterns can only be used when the PCS-PMA interface width is configured to 10 bits or 64 bits.

The pseudo-random bit stream (PRBS) block verifies the pattern generated by the PRBS generator. The verifier supports the 64-bit PCS-PMA interface. PRBS7 supports 64-bit width only. PRBS9 supports 10-bit PMA data width to allow testing at a lower data rate.

Table 168. Supported PRBS Patterns

PRBS Pattern	10 bit PCS-PMA width	64 bit PCS-PMA width
PRBS7: $x^7 + x^6 + 1$		Yes
PRBS9: $x^9 + x^5 + 1$	Yes	Yes
PRBS15: $x^{15} + x^{14} + 1$		Yes
PRBS23: $x^{23} + x^{18} + 1$		Yes
PRBS31: $x^{31} + x^{28} + 1$		Yes

Figure 189. PRBS9 Verify Serial Implementation



The PRBS checker has the following control and status signals available to the FPGA fabric:

- `rx_prbs_done`—Indicates the PRBS sequence has completed one full cycle. It stays high until you reset it with `rx_prbs_err_clr`.
- `rx_prbs_err`—Goes high if an error occurs. This signal is pulse-extended to allow you to capture it in the RX FPGA CLK domain.
- `rx_prbs_err_clr`—Used to reset the `rx_prbs_err` signal.

Enable the PRBS checker control and status ports through the Native PHY IP Parameter Editor in the Quartus Prime software.

5.2.2.7. Pseudo Random Pattern Verifier

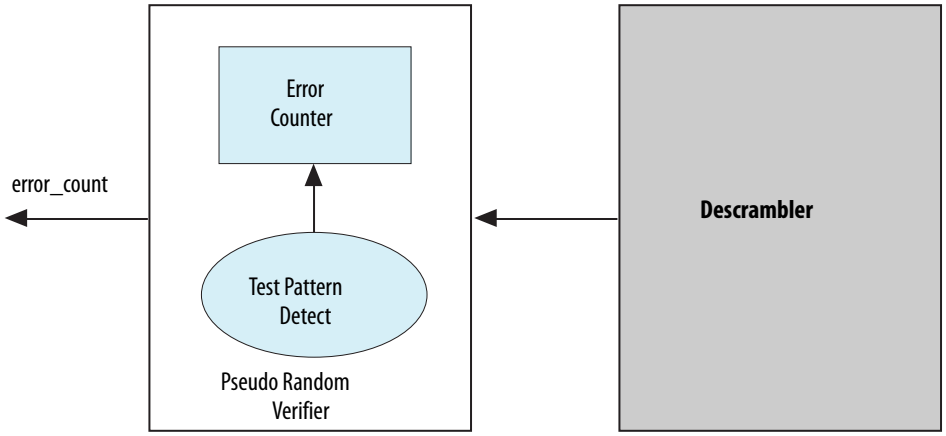
The Pseudo Random Pattern (PRP) verifier is available for 10GBASE-R and 10GBASE-R 1588 protocol modes. The PRP verifier block operates in conjunction with the descrambler. The PRP verifier monitors the output of the descrambler when block synchronization is achieved.

The `rx_prbs_err` error signal is shared between the PRBS checker and the PRP verifier.

The PRP verifier:

- Searches for a test pattern (two local faults, or all 0s) or its inverse
- Tracks the number of mismatches with a 16-bit error counter

Figure 190. PRP Verifier



Refer to the *Reconfiguration Interface and Dynamic Reconfiguration* chapter for configuration details.

Related Information

Reconfiguration Interface and Dynamic Reconfiguration on page 315

5.2.2.8. 10GBASE-R Bit-Error Rate (BER) Checker

The 10GBASE-R BER checker block is designed in accordance with the 10GBASE-R protocol specification as described in IEEE 802.3-2008 clause-49. After block lock synchronization is achieved, the BER checker starts to count the number of invalid synchronization headers within a 125-µs period. If more than 16 invalid synchronization headers are observed in a 125-µs period, the BER checker provides the status signal `rx_enh_highber` to the FPGA fabric, indicating a high bit error rate condition.

When the optional control input `rx_enh_highber_clr_cnt` is asserted, the internal counter for the number of times the BER state machine has entered the "BER_BAD_SH" state is cleared.

When the optional control input `rx_enh_clr_errblk_count` is asserted, the internal counter for the number of times the RX state machine has entered the "RX_E" state for the 10GBASE-R protocol is cleared.

Note: The 10GBASE-R BER checker is available to implement the 10GBASE-R protocol.

5.2.2.9. Interlaken CRC-32 Checker

The Interlaken CRC-32 checker verifies that the data transmitted has not been corrupted between the transmit PCS and the receive PCS. The CRC-32 checker calculates the 32-bit CRC for the received data and compares it against the CRC value that is transmitted within the diagnostic word. `rx_enh_crc32_err` (CRC error signal) is sent to the FPGA fabric.

5.2.2.10. Enhanced PCS RX FIFO

The Enhanced PCS RX FIFO is designed to compensate for the phase diference, clock difference, or both between the receiver channel PCS and the FPGA fabric. It can operate as a phase-compensation, clock-compensation, elastic buffer, or a deskew FIFO in Interlaken mode. The RX FIFO has a width of 74 bits and a depth of 32 words for all protocols.

The RX FIFO supports the following modes:

- Phase Compensation mode
- Register mode
- Interlaken mode (deskew FIFO)
- 10GBASE-R mode (clock compensation FIFO)
- Basic mode (elastic buffer FIFO)

5.2.2.10.1. Phase Compensation Mode

The RX FIFO compensates for the phase difference between the read clock and write clocks. `rx_clkout` (RX parallel low-speed clock) clocks the write side of the RX FIFO. `rx_coreclkin` (FPGA fabric clock) or `rx_clkout` clocks the read side of the RX FIFO.

When phase compensation is used in double-width mode, the FPGA data width is doubled to allow the FPGA fabric clock to run at half rate, similar to the TX FIFO phase compensation in double-width mode.

Depth of RX FIFO is constant in this mode, therefore RX FIFO flag status can be ignored. You can tie `tx_enh_data_valid` with one.

5.2.2.10.2. Register Mode

The Register Mode bypasses the FIFO functionality to eliminate the FIFO latency uncertainty for applications with stringent latency requirements. This is accomplished by tying the read clock of the FIFO with its write clock.

In Register mode, `rx_parallel_data` (data), `rx_control` indicates whether `rx_parallel_data` is a data or control word, and `rx_enh_data_valid` (data valid) are registered at the FIFO output. The RX FIFO in register mode has one register stage or one parallel clock latency.

Note: Intel recommends that you implement a soft FIFO in the FPGA fabric with minimum of 32 words under the following conditions:

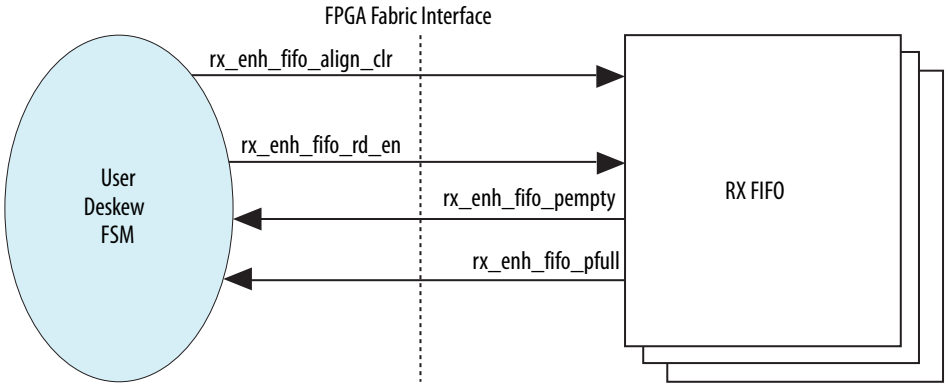
- When the Enhanced PCS RX FIFO is set to register mode.
- When using the recovered clock to drive the core logics.
- When there is no soft FIFO being generated along with the IP Catalog.

5.2.2.10.3. Interlaken Mode

In Interlaken mode, the RX FIFO operates as an Interlaken deskew FIFO. To implement the deskew process, implement an FSM that controls the FIFO operation based on available FPGA input and output flags.

For example, after frame lock is achieved, data is written after the first alignment word (SYNC word) is found on that channel. As a result, `rx_enh_fifo_pempty` (FIFO partially empty flag) of that channel goes low. You must monitor the `rx_enh_fifo_pempty` and `rx_enh_fifo_pfull` flags of all channels. If `rx_enh_fifo_pempty` flags from all channels deassert before any `rx_enh_fifo_pfull` flag asserts, which implies alignment word has been found on all lanes of the link, you start reading from all the FIFOs by asserting `rx_enh_fifo_rd_en`. Otherwise, if a `rx_enh_fifo_pfull` flag from any channel goes high before a `rx_enh_fifo_pempty` flag deassertion on all channels, you must reset the FIFO by toggling the `rx_enh_fifo_align_clr` signal and repeating the process.

Figure 191. RX FIFO as Interlaken Deskew FIFO



5.2.2.10.4. 10GBASE-R Mode

In 10GBASE-R mode, the RX FIFO operates as a clock compensation FIFO. When the block synchronizer achieves block lock, data is sent through the FIFO. Idle ordered sets (OS) are deleted and Idles are inserted to compensate for the clock difference between the RX low speed parallel clock and the FPGA fabric clock (±100 ppm for a maximum packet length of 64,000 bytes).

5.2.2.10.5. Idle OS Deletion

Deletion of Idles occurs in groups of four OS (when there are two consecutive OS) until the `rx_enh_fifo_pfull` flag deasserts. Every word—consisting of a lower word (LW) and an upper word (UW)—is checked for whether it can be deleted by looking at both the current and previous words.

For example, the current LW can be deleted if it is Idle and the previous UW is not a Terminate.

Table 169. Conditions Under Which a Word Can be Deleted

In this table X=don't care, T=Terminate, I=Idle, and OS=order set.

Deletable	Case	Word	Previous	Current	Output	
Lower Word	1	UW	!T	X	!T	X
		LW	X	I	X	X
	2	UW	OS	X	OS	X

continued...

Deletable	Case	Word	Previous	Current	Output	
		LW	X	OS	X	X
Upper Word	1	UW	X	I	X	X
		LW	X	!T	X	!T
	2	UW	X	OS	X	X
		LW	X	OS	X	OS

If only one word is deleted, data shifting is necessary because the datapath is two words wide. After two words have been deleted, the FIFO stops writing for one cycle and a synchronous flag (`rx_control[8]`) appears on the next block of 8-byte data. There is also an asynchronous status signal `rx_enh_fifo_del`, which does not go through the FIFO.

Figure 192. IDLE Word Deletion

This figure shows the deletion of IDLE words from the receiver data stream.

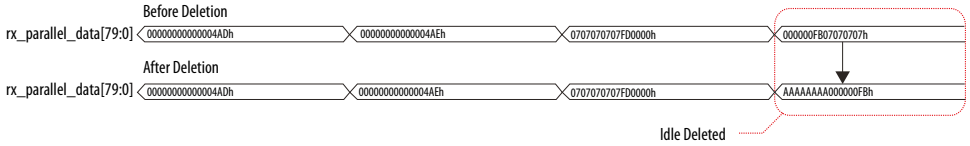
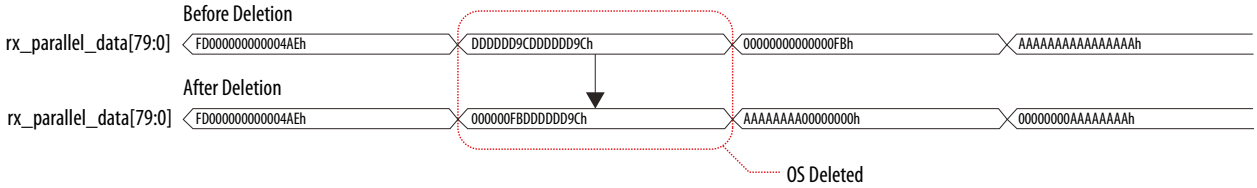


Figure 193. OS Word Deletion

This figure shows the deletion of Ordered set words in the receiver data stream.



5.2.2.10.6. Idle Insertion

Idle insertion occurs in groups of 8 Idles when the `rx_enh_fifo_pempty` flag is deasserted. Idles can be inserted following Idles or OS. Idles are inserted in groups of 8 bytes. Data shifting is not necessary. There is a synchronous status `rx_enh_fifo_insert` signal that is attached to the 8-byte Idles being inserted.

Table 170. Cases Where Two Idle Words are Inserted

In this table X=don't care, S=start, OS=order set, I-DS=idle in data stream, and I-In=idle inserted. In cases 3 and 4, the Idles are inserted between the LW and UW.

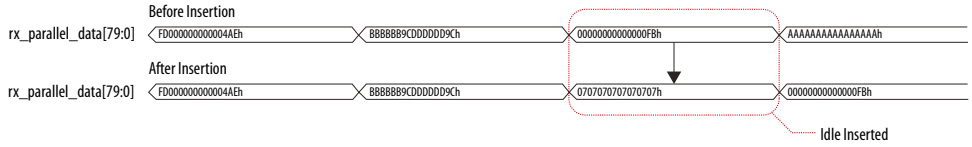
Case	Word	Input	Output	
1	UW	I-DS	I-DS	I-In
	LW	X	X	I-In
2	UW	OS	OS	I-In
	LW	X	X	I-In
3	UW	S	I-In	S

continued...

Case	Word	Input	Output	
	LW	I-DS	I-DS	I-In
4	UW	S	I-In	S
	LW	OS	OS	I-In

Figure 194. IDLE Word Insertion

This figure shows the insertion of IDLE words in the receiver data stream.



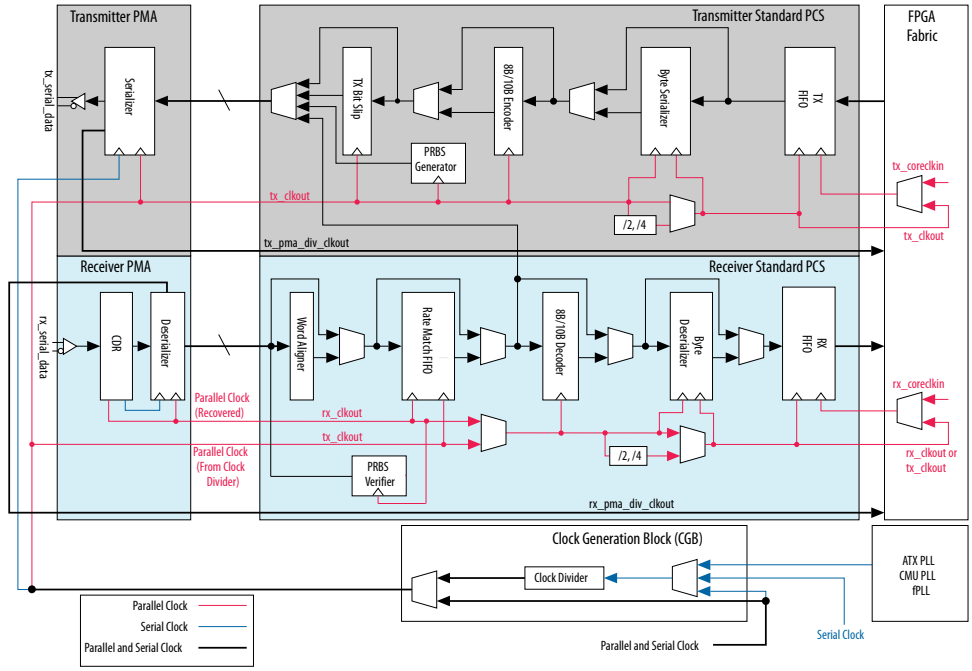
5.2.2.10.7. Basic Mode

In Basic mode, the RX FIFO operates as an elastic buffer, where buffer depths can vary. This mode allows driving write and read side of RX FIFO with different clock frequencies. Monitor the FIFO flag to control write and read operations. For RX FIFO, assert `rx_enh_read_en` signal with `rx_fifo_pfull` signal going low.

5.3. Cyclone 10 GX Standard PCS Architecture

The standard PCS can operate at a data rate of up to 10.8 Gbps. Protocols such as PCI-Express, CPRI 4.1, GigE, IEEE 1588 are supported in Hard PCS while the other protocols can be implemented using Basic/Custom (Standard PCS) transceiver configuration rules.

Figure 195. Standard PCS Datapath Diagram

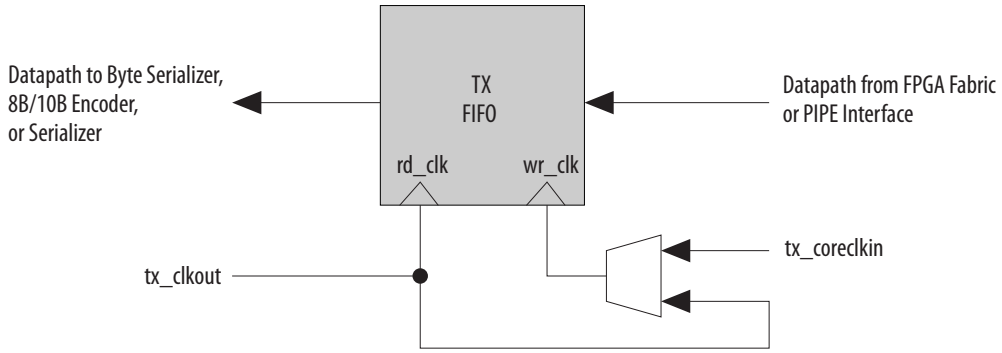


5.3.1. Transmitter Datapath

5.3.1.1. TX FIFO (Shared with Enhanced PCS and PCIe Gen2 PCS)

The TX FIFO interfaces between the transmitter PCS and the FPGA fabric and ensures reliable transfer of data and status signals. It compensates for the phase difference between the FPGA fabric clock and `tx_clkout` (the low-speed parallel clock). The TX FIFO has a depth of 8 and operates in low latency mode, register mode, and fast register mode.

Figure 196. TX FIFO Block Diagram



You can control the write port using `tx_clkout` or `tx_coreclkin`. Use the `tx_clkout` signal for a single channel and `tx_coreclkin` when using multiple channels. The TX FIFO is shared with PCIe Gen2 and Enhanced PCS data paths.

5.3.1.1.1. TX FIFO Low Latency Mode

The low latency mode incurs two to three cycles of latency (latency uncertainty) when connecting it with the FPGA fabric. The FIFO empty and the FIFO full threshold values are made closer so that the depth of the FIFO decreases, which in turn decreases the latency.

5.3.1.1.2. TX FIFO Register Mode

The register mode bypasses the FIFO functionality to eliminate the FIFO latency uncertainty for applications with stringent latency requirements. This is accomplished by tying the read clock of the FIFO with its write clock. The register mode incurs only one clock cycle of latency when interfacing to the FPGA fabric.

5.3.1.1.3. TX FIFO Fast Register Mode

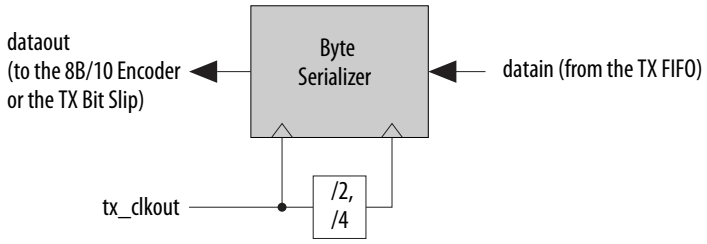
This mode allows a higher maximum frequency (f_{MAX}) between the FPGA fabric and the TX PCS by enabling the optional fast register interface with additional latency.

5.3.1.2. Byte Serializer

In certain applications, the FPGA fabric cannot operate at the same clock rate as the transmitter channel (PCS) because the transmitter channel is capable of operating at higher clock rates compared to the FPGA fabric. The byte serializer allows the transmitter channel to operate at higher data rates while keeping the FPGA fabric interface clock rate below its maximum limit. This is accomplished by increasing the

channel width two or four times (FPGA fabric-to-PCS interface width) and dividing the clock (tx_clkout) rate by 2 or 4. The byte serializer can be disabled, or operate in Serialize x2 or Serialize x4 modes.

Figure 197. Byte Serializer Block Diagram



Related Information

- Resetting Transceiver Channels on page 243
- Implementing Protocols in Intel Cyclone 10 GX Transceivers on page 16

5.3.1.2.1. Bonded Byte Serializer

The bonded byte serializer is available in Cyclone 10 GX devices, and is used in applications such as PIPE, CPRI, and custom applications where multiple channels are grouped together. The bonded byte serializer is implemented by bonding all the control signals to prevent skew induction between channels during byte serialization. In this configuration, one of the channels acts as master and the remaining channels act as slaves.

5.3.1.2.2. Byte Serializer Disabled Mode

In disabled mode, the byte serializer is bypassed. The data from the TX FIFO is directly transmitted to the 8B/10B encoder, TX Bitslip, or Serializer, depending on whether or not the 8B/10B encoder and TX Bitslip are enabled. Disabled mode is used in low speed applications such as GigE, where the FPGA fabric and the TX standard PCS can operate at the same clock rate.

5.3.1.2.3. Byte Serializer Serialize x2 Mode

The serialize x2 mode is used in high-speed applications such as the PCIe Gen1 or Gen2 protocol implementation, where the FPGA fabric cannot operate as fast as the TX PCS.

In serialize x2 mode, the byte serializer serializes 16-bit, 20-bit (when 8B/10B encoder is not enabled), 32-bit, and 40-bit (when 8B/10B encoder is not enabled) input data into 8-bit, 10-bit, 16-bit, and 20-bit data, respectively. As the parallel data width from the TX FIFO is halved, the clock rate is doubled.

After byte serialization, the byte serializer forwards the least significant word first followed by the most significant word. For example, if the FPGA fabric-to-PCS Interface width is 32, the byte serializer forwards `tx_parallel_data[15:0]` first, followed by `tx_parallel_data[31:16]`.

Related Information

PCI Express (PIPE) on page 122
For more information about using the Serialize x2 mode in the PCIe protocol.

5.3.1.2.4. Byte Serializer Serialize x4 Mode

The serialize x4 mode is used in high-speed applications such as the PCIe Gen2 protocol mode, where the FPGA fabric cannot operate as fast as the TX PCS.

In serialize x4 mode, the byte serializer serializes 32-bit data into 8-bit data. As the parallel data width from the TX FIFO is divided four times, the clock rate is quadrupled.

After byte serialization, the byte serializer forwards the least significant word first followed by the most significant word. For example, if the FPGA fabric-to-PCS Interface width is 32, the byte serializer forwards `tx_parallel_data[7:0]` first, followed by `tx_parallel_data[15:8]`, `tx_parallel_data[23:16]` and `tx_parallel_data[31:24]`.

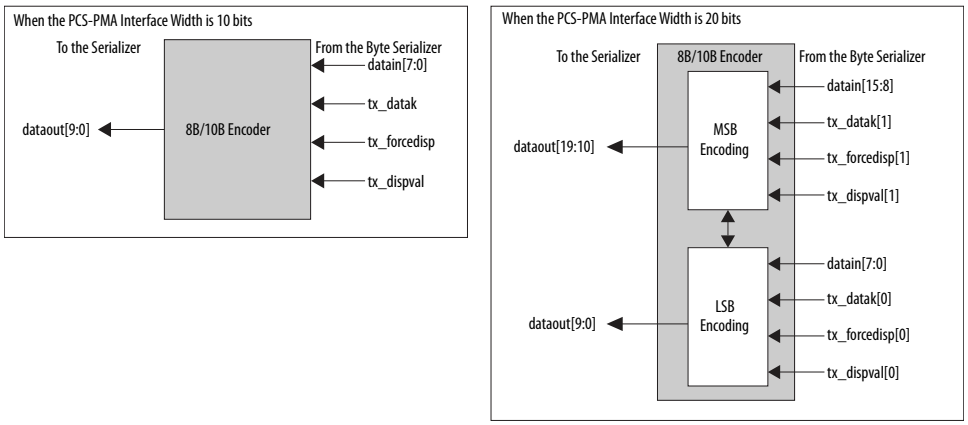
Related Information

PCI Express (PIPE) on page 122
For more information about using the Serialize x4 mode in the PCIe protocol.

5.3.1.3. 8B/10B Encoder

The 8B/10B encoder takes in 8-bit data and 1-bit control as input and converts them into a 10-bit output. The 8B/10B encoder automatically performs running disparity check for the 10-bit output. Additionally, the 8B/10B encoder can control the running disparity manually using the `tx_forcedisp` and `tx_dispval` ports.

Figure 198. 8B/10B Encoder Block Diagrams

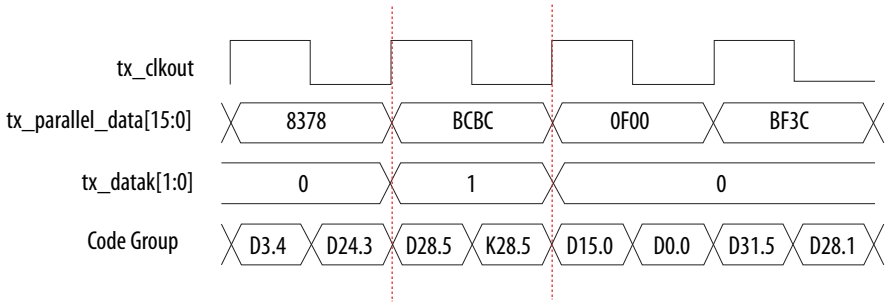


When the PCS-PMA interface width is 10 bits, one 8B/10B encoder is used to convert the 8-bit data into a 10-bit output. When the PCS-PMA interface width is 20 bits, two cascaded 8B/10B encoders are used to convert the 16-bit data into a 20-bit output. The first eight bits (LSByte) is encoded by the first 8B/10B encoder and the next eight bits (MSByte) is encoded by the second 8B/10B encoder. The running disparity of the LSByte is calculated first and passed on to the second encoder to calculate the running disparity of the MSByte.

Note: You cannot enable the 8B/10B encoder when the PCS-PMA interface width is 8 bits or 16 bits.

5.3.1.3.1. 8B/10B Encoder Control Code Encoding

Figure 199. Control Code Encoding Diagram



The `tx_datak` signal indicates whether the 8-bit data being sent at the `tx_parallel_data` port should be a control word or a data word. When `tx_datak` is high, the 8-bit data is encoded as a control word (Kx.y). When `tx_datak` is low, the 8-bit data is encoded as a data word (Dx.y). Depending upon the PCS-PMA interface width, the width of `tx_datak` is either 1 bit or 2 bits. When the PCS-PMA interface width is 10 bits, `tx_datak` is a 1-bit word. When the PCS-PMA interface width is 20 bits, `tx_datak` is a 2-bit word. The LSB of `tx_datak` corresponds to the LSByte of the input data sent to the 8B/10B encoder and the MSB corresponds to the MSByte of the input data sent to the 8B/10B encoder.

Related Information

Refer to *Specifications & Additional Information* for more information about 8B/10B encoder codes.

5.3.1.3.2. 8B/10B Encoder Reset Condition

The `tx_digitalreset` signal resets the 8B/10B encoder. During the reset condition, the 8B/10B encoder outputs K28.5 continuously until `tx_digitalreset` goes low.

5.3.1.3.3. 8B/10B Encoder Idle Character Replacement Feature

The idle character replacement feature is used in protocols such as Gigabit Ethernet, which requires the running disparity to be maintained during idle sequences. During these idle sequences, the running disparity has to be maintained such that the first byte of the next packet always starts when the running disparity of the current packet is negative.

When an ordered set, which consists of two code-groups, is received by the 8B/10B encoder, the second code group is converted into /I1/ or /I2 so that the final running disparity of the data code-group is negative. The first code group is /K28.5/ and the second code group is a data code-group other than /D21.5/ or /D2.2/. The ordered set /I1/ (/K28.5/D5.6/) is used to flip the running disparity and /I2/ (/K28.5/D16.2/) is used to preserve the running disparity.

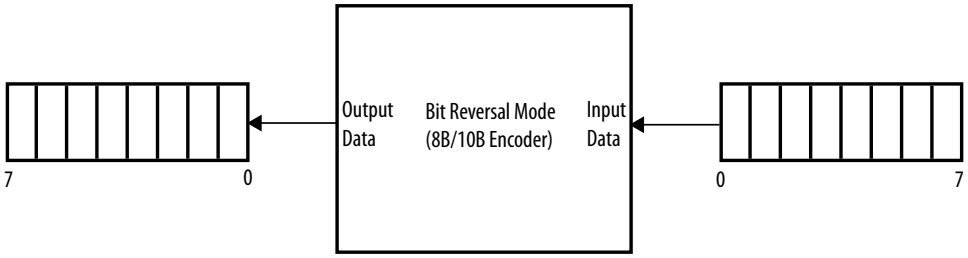
5.3.1.3.4. 8B/10B Encoder Current Running Disparity Control Feature

The 8B/10B encoder performs a running disparity check on the 10-bit output data. The running disparity can also be controlled using `tx_forcedisp` and `tx_dispval`. When the PCS-PMA interface width is 10 bits, `tx_forcedisp` and `tx_dispval` are one bit each. When the PCS-PMA interface width is 20 bits, `tx_forcedisp` and

`tx_dispval` are two bits each. The LSB of `tx_forcedisp` and `tx_dispval` corresponds to the LSByte of the input data and the MSB corresponds to the MSByte of the input data.

5.3.1.3.5. 8B/10B Encoder Bit Reversal Feature

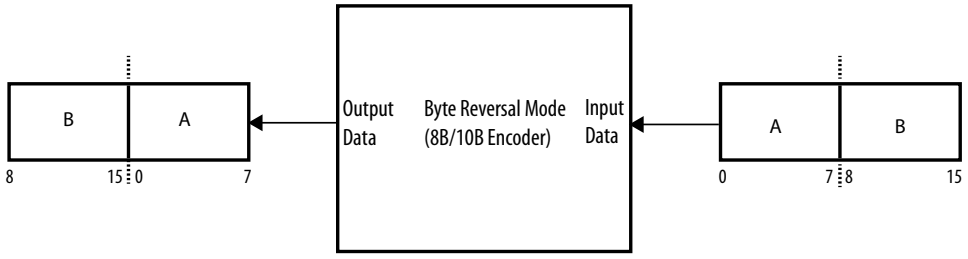
Figure 200. 8B/10B Encoder Bit Reversal Feature



The bit reversal feature reverses the order of the bits of the input data. Bit reversal is performed at the output of the 8B/10B Encoder and is available even when the 8B/10B Encoder is disabled. For example, if the input data is 20-bits wide, bit reversal switches bit [0] with bit [19], bit [1] with bit [18] and so on.

5.3.1.3.6. 8B/10B Encoder Byte Reversal Feature

Figure 201. 8B/10B Encoder Byte Reversal Feature



The byte reversal feature is available only when the PCS-PMA interface width is 16 bits or 20 bits. Byte reversal is performed at the output of the 8B/10B Encoder and is available even when the 8B/10B Encoder is disabled. This feature swaps the LSByte with the MSByte. For example, when the PCS-PMA interface width is 16-bits, [7:0] bits (LSByte) gets swapped with [15:8] bits (MSByte).

5.3.1.4. Polarity Inversion Feature

The polarity inversion feature is used in situations where the positive and the negative signals of a serial differential link are erroneously swapped during board layout. You can control this feature through `tx_polinv port`, by enabling the **Enable TX Polarity Inversion** option under the Standard PCS tab of the Native PHY IP Core. The polarity inversion feature inverts the value of each bit of the input data. For example, if the input data is 00101001, then the data gets changed to 11010110 after polarity inversion.

5.3.1.5. Pseudo-Random Binary Sequence (PRBS) Generator

Note: Refer to the PRBS Generator section in the Enhanced PCS Architecture chapter.

Related Information

Cyclone 10 GX Enhanced PCS Architecture on page 283

5.3.1.6. TX Bit Slip

The TX bit slip allows the word boundary to be controlled by `tx_std_bitslipboundarysel`. The TX bit slip feature is used in applications, such as CPRI, which has a data rate greater than 6 Gbps. The maximum number of the supported bit slips is PCS data width-1 and the slip direction is from MSB to LSB and from current to previous word.

5.3.2. Receiver Datapath

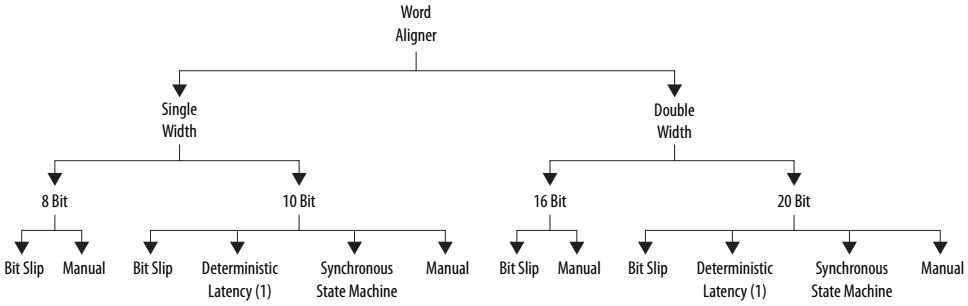
5.3.2.1. Word Aligner

The word aligner receives the serial data from the PMA and realigns the serial data to have the correct word boundary according to the word alignment pattern configured. This word alignment pattern can be 7, 8, 10, 16, 20, 32 and 40 bits in length.

Depending on your PCS-PMA interface width, the word aligner can be configured in one of the following modes:

- Bit slip
- Manual alignment
- Synchronous state machine
- Deterministic latency

Figure 202. Word Aligner Conditions and Modes



Note:
1. This option is available in CPRI mode.

5.3.2.1.1. Word Aligner Bit Slip Mode

In bit slip mode, the word aligner operation is controlled by `rx_bitslip`, which has to be held for two parallel clock cycles. At every rising edge of `rx_bitslip`, the bit slip circuitry slips one bit into the received data stream, effectively shifting the word boundary by one bit. Pattern detection is not used in bit slipping mode; therefore, `rx_syncstatus` is not valid in this mode.

5.3.2.1.2. Word Aligner Manual Mode

In manual alignment mode, the word aligner operation is controlled by `rx_std_wa_patternalign`. The word aligner operation is edge-sensitive or level-sensitive to `rx_std_wa_patternalign`, depending upon the PCS-PMA interface width selected.

Table 171. Word Aligner `rx_std_wa_patternalign` Behavior

PCS-PMA Interface Width	`rx_std_wa_patternalign` Behavior
8	Rising edge sensitive
10	Level sensitive
16	Rising edge sensitive
20	Rising edge sensitive

If `rx_std_wa_patternalign` is asserted, the word aligner looks for the programmed word alignment pattern in the received data stream. It updates the word boundary if it finds the word alignment pattern in a new word boundary. If `rx_std_wa_patternalign` is deasserted, 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` signals, with the same latency as the datapath, are forwarded to the FPGA fabric to indicate the word aligner status.

After receiving the first word alignment pattern after `rx_std_wa_patternalign` is asserted, both `rx_syncstatus` and `rx_patterndetect` are driven high for one parallel clock cycle. Any word alignment pattern received thereafter in the same word boundary causes only `rx_patterndetect` 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 `rx_std_wa_patternalign` is asserted. The word aligner asserts `rx_syncstatus` for one parallel clock cycle whenever it re-aligns to the new word boundary.

5.3.2.1.3. Word Aligner Synchronous State Machine Mode

In synchronous state machine mode, when the programmed number of valid synchronization code groups or ordered sets is received, `rx_syncstatus` 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 `rx_syncstatus` is driven low.

The word aligner indicates loss of synchronization (`rx_syncstatus` remains low) until the programmed number of valid synchronization code groups are received again.

5.3.2.1.4. Word Aligner Deterministic Latency Mode

In deterministic latency mode, the state machine removes the bit level latency uncertainty. The deserializer of the PMA creates the bit level latency uncertainty as it comes out of reset.

The PCS performs pattern detection on the incoming data from the PMA. The PCS aligns the data, after it indicates to the PMA the number of serial bits to clock slip the boundary.

If the incoming data has to be realigned, `rx_std_wa_patternalign` must be reasserted to initiate another pattern alignment. Asserting `rx_std_wa_patternalign` can cause the word align to lose synchronization if already achieved. This may cause `rx_syncstatus` to go low.

Table 172. PCS-PMA Interface Widths and Protocol Implementations

PCS-PMA Interface Width	Protocol Implementations
8	Basic
10	• Basic • Basic rate match • CPRI • PCIe Gen1 and Gen2 • GigE
16	Basic
20	• CPRI • Basic • Basic rate match

5.3.2.1.5. Word Aligner Pattern Length for Various Word Aligner Modes

Table 173. Word Aligner Pattern Length for Various Word Aligner Modes

PCS-PMA Interface Width	Supported Word Aligner Modes	Supported Word Aligner Pattern Lengths	rx_std_wa_patternalign behavior	rx_syncstatus behavior	rx_patterndetect behavior
8	Bit slip	8	`rx_std_wa_patternalign` has no effect on word alignment. The single width word aligner updates the word boundary, only when the FPGA fabric-asserted BITSLIP signal toggles.	N/A	N/A
	Manual	8, 16	Word alignment is controlled by `rx_std_wa_patternalign` and is edge-sensitive to this signal.	Asserted high for one parallel clock cycle when the word aligner aligns to a new boundary.	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.
10	Bit slip	7	`rx_std_wa_patternalign` has no effect on word alignment. The single width word aligner updates the word boundary, only when the	N/A	N/A

continued...

PCS-PMA Interface Width	Supported Word Aligner Modes	Supported Word Aligner Pattern Lengths	rx_std_wa_patternalign behavior	rx_syncstatus behavior	rx_patterndetect behavior
			FPGA fabric-asserted BITSLIP signal toggles.		
	Manual	7, 10	Word alignment is controlled by rx_std_wa_patternalign and is level-sensitive to this signal.	Asserted high for one parallel clock cycle when the word aligner aligns to a new boundary.	Asserted high for one parallel clock cycle when the word alignment pattern appears in the current word boundary.
	Deterministic latency (CPRI mode only)	10	Word alignment is controlled by rx_std_wa_patternalign (edge-sensitive to this signal) and the state machine works in conjunction with PMA to achieve deterministic latency on the RX path for CPRI and OBSAI applications.	—	—
	Synchronous State Machine	7, 10	rx_std_wa_patternalign has no effect on word alignment.	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.
16	Bit slip	16	rx_std_wa_patternalign has no effect on word alignment. The double width word aligner updates the word boundary, only when the FPGA fabric-asserted BITSLIP signal toggles.	N/A	N/A
	Manual	8, 16, 32	Word alignment is controlled by rising-edge of rx_std_wa_patternalign.	Stays high after the word aligner aligns to the word alignment pattern. Goes low on receiving a rising edge on rx_std_wa_patternalign 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.
20	Bit slip	7	rx_std_wa_patternalign has no effect on word alignment. The	N/A	N/A

continued...

PCS-PMA Interface Width	Supported Word Aligner Modes	Supported Word Aligner Pattern Lengths	rx_std_wa_patternalign behavior	rx_syncstatus behavior	rx_patterndetect behavior
			double width word aligner updates the word boundary, only when the FPGA fabric-asserted BITSLIP signal toggles.		
	Manual	7, 10, 20, 40	Word alignment is controlled by rising edge of rx_std_wa_patternalign.	Stays high after the word aligner aligns to the word alignment pattern. Goes low on receiving a rising edge on rx_std_wa_patternalign 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.
	Deterministic latency (CPRI mode only)	10	Word alignment is controlled by rx_std_wa_patternalign (edge-sensitive to this signal) and the deterministic latency state machine which controls the PMA to achieve deterministic latency on the RX path for CPRI and OBSAI applications.	—	—
	Synchronous State Machine	7, 10, 20	FPGA fabric-driven rx_std_wa_patternalign signal has no effect on word alignment.	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.

5.3.2.1.6. Word Aligner RX Bit Reversal Feature

The RX bit reversal feature reverses the order of the data received from the PMA. It is performed at the output of the Word Aligner and is available even when the Word Aligner is disabled. If the data received from the PMA is a 10-bit data width, the bit reversal feature switches bit [0] with bit [9], bit [1] with bit [8], and so on. For example, if the 10-bit data is 1000010011, the bit reversal feature, when enabled, changes the data to 1100100001.

5.3.2.1.7. Word Aligner RX Byte Reversal Feature

The RX byte reversal feature is available only when the PCS-PMA interface width is 16 bits or 20 bits. This feature reverses the order of the data received from the PMA. RX byte reversal reverses the LSByte of the received data with its MSByte and vice versa. If the data received is 20-bits, bits[0..9] are swapped with bits[10..20] so that the

resulting 20-bit data is [[10..20],[0..9]]. For example, if the 20-bit data is 11001100001000011111, the byte reversal feature changes the data to 10000111111100110000.

5.3.2.2. RX Polarity Inversion Feature

The RX polarity inversion feature inverts each bit of the data received from the PMA. If the data received is a 10-bit data. Bit[0] content is inverted to its complement, ~bit[0], bit[1] is inverted to its complement, ~bit[1], bit[2] is inverted to its complement, ~bit[2], and so on. For example, if the 10-bit data is 1111100000, the polarity inversion feature inverts it to 0000011111.

5.3.2.3. Rate Match FIFO

The rate match FIFO compensates for the frequency differences between the local clock and the recovered clock up to ± 300 ppm by inserting and deleting skip/idle characters in the data stream. The rate match FIFO has several different protocol specific modes of operation. All of the protocol specific modes depend upon the following parameters:

- Rate match deletion—occurs when the distance between the write and read pointers exceeds a certain value due to write clock having a higher frequency than the read clock.
- Rate match insertion—occurs when the distance between the write and the read pointers becomes less than a certain value due to the read clock having a higher frequency than the write clock.
- Rate match full—occurs when the write pointer wraps around and catches up to the slower-advancing read pointer.
- Rate match empty—occurs when the read pointer catches up to the slower-advancing write pointer.

Rate match FIFO operates in six modes:

- Basic single width
- Basic double width
- GigE
- PIPE
- PIPE 0 ppm
- PCIe

Related Information

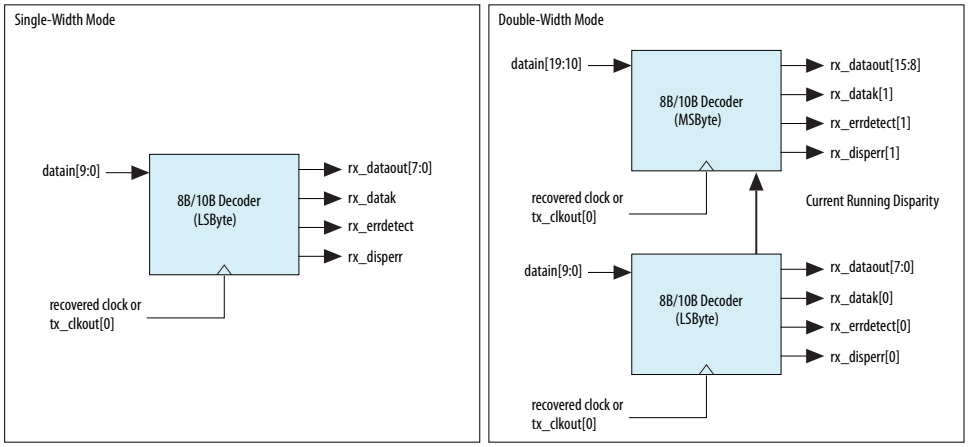
- How to Implement GbE, GbE with IEEE 1588v2 in Intel Cyclone 10 GX Transceivers on page 93
  For more information about implementing rate match FIFO in GigE mode.
- PCI Express (PIPE) on page 122
  For more information about implementing rate match FIFO in PCIe mode.
- Using the Basic/Custom, Basic/Custom with Rate Match Configurations of Standard PCS on page 166
- How to Implement the Basic Rate Match Protocol Using the Cyclone 10 GX Transceiver Native PHY IP Core on page 166
  For more information about implementing rate match FIFO for each mode.

5.3.2.4. 8B/10B Decoder

The general functionality for the 8B/10B decoder is to take a 10-bit encoded value as input and produce an 8-bit data value and a 1-bit control value as output. In configurations with the rate match FIFO enabled, the 8B/10B decoder receives data from the rate match FIFO. In configurations with the rate match FIFO disabled, the 8B/10B decoder receives data from the word aligner. The 8B/10B decoder operates in two conditions:

- When the PCS-PMA interface width is 10 bits and FPGA fabric-PCS interface width is 8 bits
- When the PCS-PMA interface width is 20 bits and FPGA fabric-PCS interface width is 16 bits

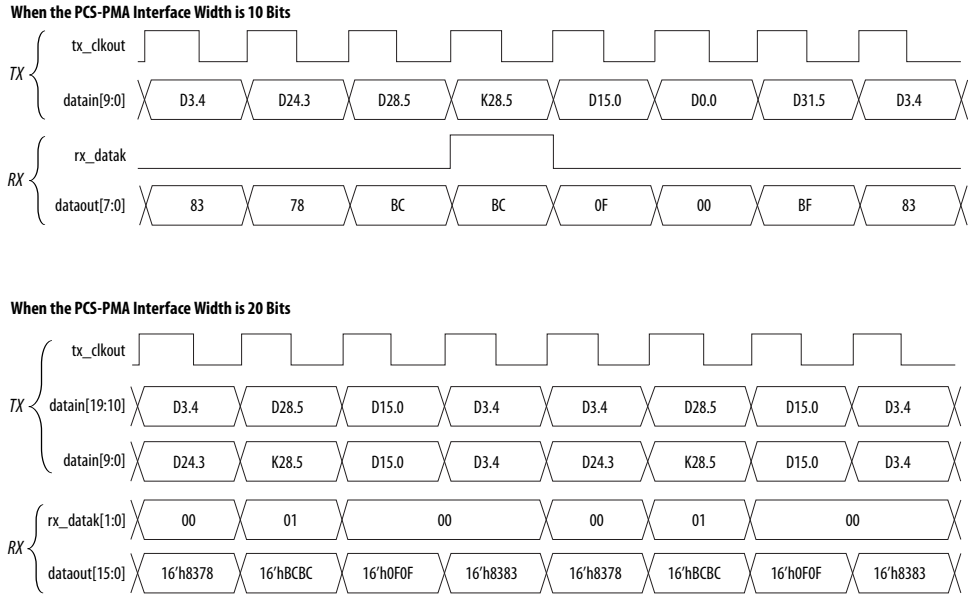
Figure 203. 8B/10B Decoder in Single-Width and Double-Width Mode



When the PCS-PMA interface width is 10 bits, only one 8B/10B decoder is used to perform the conversion. When the PCS-PMA interface width is 20 bits, two cascaded 8B/10B decoders are used. The 10-bit LSByte of the received 20-bit encoded data is decoded first and the ending running disparity is forwarded to the 8B/10B decoder responsible for decoding the 10-bit MSByte. The cascaded 8B/10B decoder decodes the 20- bit encoded data into 16-bit data + 2-bit control identifier. The MSB and LSB of the 2-bit control identifier correspond to the MSByte and LSByte of the 16-bit decoded data code group. The decoded data is fed to the byte deserializer or the RX FIFO.

5.3.2.4.1. 8B/10B Decoder Control Code Encoding

Figure 204. 8B/10B Decoder in Control Code Group Detection



The 8B/10B decoder indicates whether the decoded 8-bit code group is a data or control code group on `rx_datak` . If the received 10-bit code group is one of the 12 control code groups (/Kx.y/) specified in the IEEE 802.3 specification, `rx_datak` is driven high. If the received 10-bit code group is a data code group (/Dx.y/), `rx_datak` is driven low.

5.3.2.4.2. 8B/10B Decoder Running Disparity Checker Feature

Running disparity checker resides in 8B/10B decoder module. This checker checks the current running disparity value and error based on the rate match output. `rx_runningdisp` and `rx_disperr` indicate positive or negative disparity and disparity errors, respectively.

5.3.2.5. Pseudo-Random Binary Sequence (PRBS) Checker

Note: Refer to the PRBS Checker section in the Enhanced PCS Architecture chapter.

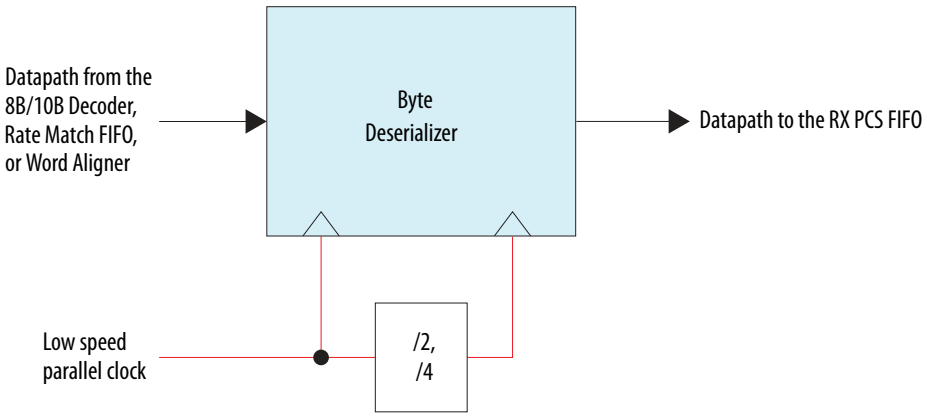
Related Information

Cyclone 10 GX Enhanced PCS Architecture on page 283

5.3.2.6. Byte Deserializer

The byte deserializer allows the transceiver to operate at data rates higher than those supported by the FPGA fabric. It deserializes the recovered data by multiplying the data width two or four times, depending upon the deserialization mode selected. The byte deserializer is optional in designs that do not exceed the FPGA fabric interface frequency upper limit. You can bypass the byte deserializer by disabling it in the Quartus Prime Transceiver Native PHY. The byte deserializer operates in disabled, deserialize x2, or deserialize x4 modes.

Figure 205. Byte Deserializer Block Diagram



5.3.2.6.1. Byte Deserializer Disabled Mode

In disabled mode, the byte deserializer is bypassed. The data from the 8B/10B decoder, rate match FIFO, or word aligner is directly transmitted to the RX FIFO, depending on whether or not the 8B/10B decoder and rate match FIFO are enabled. Disabled mode is used in low speed applications such as GigE, where the FPGA fabric and the PCS can operate at the same clock rate.

5.3.2.6.2. Byte Deserializer Deserialize x2 Mode

The deserialize x2 mode is used in high-speed applications such as the PCIe Gen1 or Gen2 protocol implementation, where the FPGA fabric cannot operate as fast as the TX PCS.

In deserialize x2 mode, the byte deserializer deserializes 8-bit, 10-bit (when the 8B/10B encoder is not enabled), 16-bit, and 20-bit (when the 8B/10B encoder is not enabled) input data into 16-bit, 20-bit, 32-bit, and 40-bit data, respectively. As the parallel data width from the word aligner is doubled, the clock rate is halved.

5.3.2.6.3. Byte Deserializer Deserialize x4 Mode

The deserialize x4 mode is used in high-speed applications where the FPGA fabric cannot operate as fast as the TX PCS.

In deserialize x4 mode, the byte deserializer deserializes 8-bit data into 32-bit data. As the parallel data width from the word aligner is quadrupled, the clock rate is divided four times.

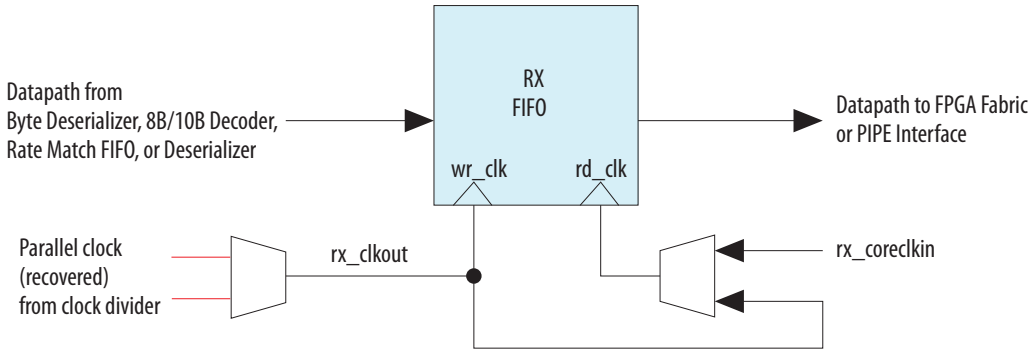
5.3.2.6.4. Bonded Byte Deserializer

The bonded byte deserializer is also available for channel-bundled applications such as PIPE. In this configuration, the control signals of the byte deserializers of all the channels are bonded together. A master channel controls all the other channels to prevent skew between the channels.

5.3.2.7. RX FIFO (Shared with Enhanced PCS and PCIe Gen2 PCS)

The RX FIFO interfaces between the PCS on the receiver side and the FPGA fabric and ensures reliable transfer of data and status signals. It compensates for the phase difference between the FPGA fabric and the PCS on the receiver side. The RX FIFO has a depth of 8. It operates in register FIFO and low latency modes.

Figure 206. RX FIFO Block Diagram



5.3.2.7.1. RX FIFO Low Latency Mode

The low latency mode incurs two to three cycles of latency when connecting it with the FPGA fabric. The FIFO empty and the FIFO full threshold values are made closer so that the depth of the FIFO decreases, which in turn decreases the latency.

5.3.2.7.2. RX FIFO Register Mode

The register mode bypasses the FIFO functionality to eliminate the FIFO latency uncertainty for applications with stringent latency requirements. This is accomplished by tying the read clock of the FIFO with its write clock. The register mode incurs only one clock cycle of latency when interfacing to the FPGA fabric.

5.4. Intel Cyclone 10 GX Transceiver PHY Architecture Revision History

Document Version	Changes
2020.05.15	Made the following change: • Removed OFF as an option in the "Transmitter Buffer" figure and *Programmable Transmitter On-Chip Termination (OCT)*.
2017.11.30	Made the following changes: • Removed the QPI configuration from the "Transmitter Buffer" section.
2017.11.06	Made the following changes: • Added a link to the Intel Cyclone 10 GX Register Map in the "Configuration Methods" section.
2017.05.08	Initial release.

6. Reconfiguration Interface and Dynamic Reconfiguration

Dynamic reconfiguration is the process of dynamically modifying transceiver channels and PLLs to meet changing requirements during device operation. Cyclone 10 GX transceiver channels and PLLs are fully customizable, allowing a system to adapt to its operating environment. You can customize channels and PLLs by dynamically triggering reconfiguration during device operation or following power-up. Dynamic reconfiguration is available for Cyclone 10 GX Transceiver Native PHY, fPLL, ATX PLL, and CMU PLL IP cores.

Use the reconfiguration interface to dynamically change the transceiver channel or PLL settings for the following applications:

- Fine tuning signal integrity by adjusting TX and RX analog settings
- Enabling or disabling transceiver channel blocks, such as the PRBS generator and the checker
- Changing data rates to perform auto negotiation in CPRI, SATA, or SAS applications
- Changing data rates in Ethernet (1G/10G) applications by switching between standard and enhanced PCS datapaths
- Changing TX PLL settings for multi-data rate support protocols such as CPRI
- Changing RX CDR settings from one data rate to another
- Switching between multiple TX PLLs for multi-data rate support

The Native PHY and Transmit PLL IP cores provide the following features that allow dynamic reconfiguration:

- Reconfiguration interface
- Configuration files
- Feature to add PMA analog settings (optional) to the Configuration files (Native PHY only)
- Multiple reconfiguration profiles (Native PHY and ATX PLL)
- Embedded reconfiguration streamer (Native PHY and ATX PLL)
- Native PHY Debug Master Endpoint (NPDME)
- Optional reconfiguration logic

6.1. Reconfiguring Channel and PLL Blocks

The following table lists some of the available dynamic reconfiguration features in Cyclone 10 GX devices.

Table 174. Cyclone 10 GX Dynamic Reconfiguration Feature Support

Reconfiguration	Features
Channel Reconfiguration	PMA analog features • V_{OD} • Pre-emphasis • Continuous Time Linear Equalizer (CTLE)
	TX PLL • TX local clock dividers • TX PLL switching
	RX CDR • RX CDR settings • RX CDR reference clock switching
	Reconfiguration of PCS blocks within the datapath
	Datapath switching • Standard, Enhanced, PCS Direct
PLL Reconfiguration	PLL settings • Counters
	PLL reference clock switching

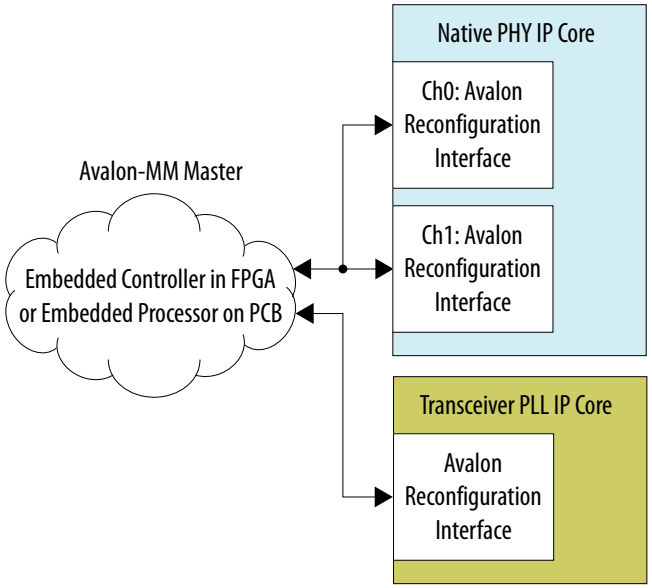
Related Information

Unsupported Features on page 371

6.2. Interacting with the Reconfiguration Interface

Each transceiver channel and PLL contains an Avalon Memory-Mapped (Avalon-MM) reconfiguration interface. The reconfiguration interface provides direct access to the programmable space of each channel and PLL. Communication with the channel and PLL reconfiguration interface requires an Avalon-MM master. Because each channel and PLL has its own dedicated Avalon-MM interface, you can dynamically modify channels either concurrently or sequentially, depending on how the Avalon-MM master is connected to the Avalon-MM reconfiguration interface.

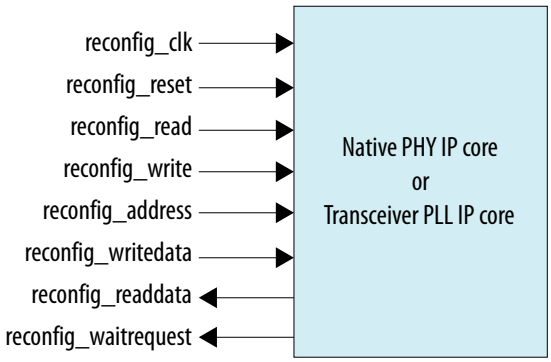
Figure 207. Reconfiguration Interface in Cyclone 10 GX Transceiver IP Cores



A transmit PLL instance has a maximum of one reconfiguration interface. Unlike PLL instances, a Native PHY IP core instance can specify multiple channels. You can use a dedicated reconfiguration interface for each channel or share a single reconfiguration interface across all channels to perform dynamic reconfiguration.

Avalon-MM masters interact with the reconfiguration interface by performing Avalon read and write operations to initiate dynamic reconfiguration of specific transceiver parameters. All read and write operations must comply with Avalon-MM specifications.

Figure 208. Top-Level Signals of the Reconfiguration Interface



The user-accessible Avalon-MM reconfiguration interface and PreSICE Avalon-MM interface share a single internal configuration bus. This bus is arbitrated to get access to the Avalon-MM interface of the channel or PLL. Refer to the *Arbitration* section for more details about requesting access to and returning control of the internal configuration bus from PreSICE.

Related Information

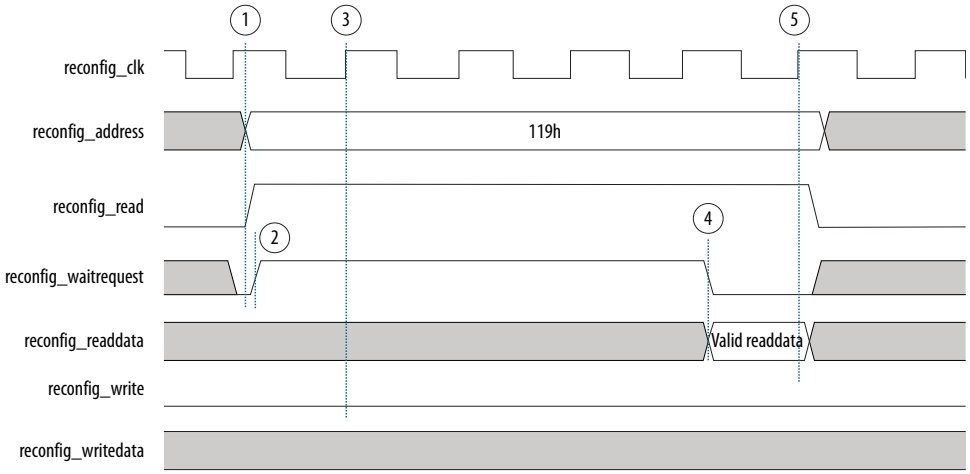
- Arbitration on page 325

- Reconfiguration Interface and Arbitration with PreSICE Calibration Engine on page 373
- *Avalon Interface Specifications*

6.2.1. Reading from the Reconfiguration Interface

Reading from the reconfiguration interface of the Transceiver Native PHY IP core or Transceiver PLL IP core retrieves the current value at a specific address. The dynamic reconfiguration interface is compliant to the AVMM specifications.

Figure 209. Reading from the Reconfiguration Interface



1. The master asserts reconfig_address and reconfig_read after the rising edge of reconfig_clk.
2. The slave asserts reconfig_waitrequest, stalling the transfer.
3. The master samples reconfig_waitrequest. Because reconfig_waitrequest is asserted, the cycle becomes a wait state and reconfig_address, reconfig_read, and reconfig_write remain constant.
4. The slave presents valid reconfig_readdata and deasserts reconfig_waitrequest.
5. The master samples reconfig_waitrequest and reconfig_readdata, completing the transfer.

After the `reconfig_read` signal is asserted, the `reconfig_waitrequest` signal asserts for a few `reconfig_clock` cycles, then deasserts. This deassertion indicates the `reconfig_readdata` bus contains valid data.

Note: You must check for the internal configuration bus arbitration before performing reconfiguration. Refer to the *Arbitration* section for more details about requesting access to and returning control of the internal configuration bus from PreSICE.

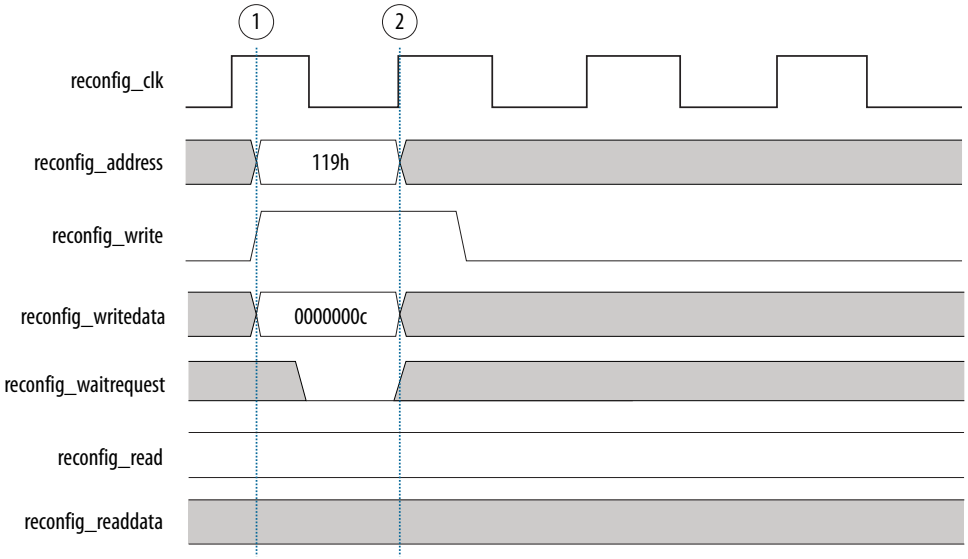
Related Information

Arbitration on page 325

6.2.2. Writing to the Reconfiguration Interface

Writing to the reconfiguration interface of the Transceiver Native PHY IP core or TX PLL IP core changes the data value at a specific address. All writes to the reconfiguration interface must be read-modify-writes, because two or more features may share the same reconfiguration address. When two or more features share the same reconfiguration address, one feature's data bits are interleaved with another feature's data bits.

Figure 210. Writing to the Reconfiguration Interface



1. The master asserts the reconfig_address, reconfig_write, and reconfig_writedata signals.
2. The slave (channel or PLL) captures reconfig_writedata, ending the transfer.

Note: You must check for the internal configuration bus arbitration before performing reconfiguration. Refer to the *Arbitration* section for more details about requesting access to and returning control of the internal configuration bus from PreSICE.

Related Information

- Arbitration on page 325
- Ports and Parameters on page 346

6.3. Configuration Files

The Cyclone 10 GX Transceiver Native PHY and Transmit PLL IP cores optionally allow you to save the parameters you specify for the IP instances as configuration files. The configuration file stores addresses and data values for that specific IP instance.

The configuration files are generated during IP generation. They are located in the <IP instance name>/reconfig/ subfolder of the IP instance. The configuration data is available in the following formats:

- **SystemVerilog packages**: <name>**.sv**
- **C Header files**: <name>**.h**
- **Memory Initialization File (MIF)**: <name>**.mif**

Select one or more of the configuration file formats on the **Dynamic Reconfiguration** tab of the Transceiver Native PHY or Transmit PLL parameter editor to store the configuration data. All configuration files generated for a particular IP instance contain the same address and data values. The contents of the configuration files can be used to reconfigure from one transceiver /PLL configuration to another.

You can optionally allow the Native PHY IP core to include PMA Analog settings in the configuration files by enabling the feature **Include PMA Analog settings in configuration files** in the **Dynamic Reconfirmation** tab of the Transceiver Native PHY IP Parameter Editor. This feature is disabled by default. Enabling this feature adds the PMA analog settings specified in the **Analog PMA settings (Optional)** tab of the Native PHY IP Parameter Editor to the configuration files. Even with this option enabled in the Native PHY IP Parameter Editor, you must still specify Quartus Settings File (QSF) assignments for your analog settings when compiling your static design. The analog settings selected in the Native PHY IP Parameter Editor are used only to include these settings and their dependent settings in the selected configuration files. Refer to the *Analog Parameter Settings* chapter for details about QSF assignments for the analog settings.

Example 2. SystemVerilog Configuration File

```
    26'h008FF04,
// [25:16]-DPRIO address=0x008;
// [15:8]-bit mask=0xFF;
// [7:7]- hssi_tx_pcs_pma_interface_pldif_datawidth_mode=pldif_data_10bit(1'h0);
// [6:5]-hssi_tx_pcs_pma_interface_tx_pma_data_sel=ten_g_pcs(2'h0);
// [4:4]-hssi_tx_pcs_pma_interface_prbs_gen_pat=prbs_gen_dis(1'h0);
// [3:0]-hssi_tx_pcs_pma_interface_sq_wave_num=sq_wave_default(4'h4);
…

localparam HSSI_TX_PCS_PMA_INTERFACE_PLDIF_DATAWIDTH_MODE_VALUE = "pldif_data_10bit";
localparam HSSI_TX_PCS_PMA_INTERFACE_PLDIF_DATAWIDTH_MODE_ADDR_OFST = 8;
localparam HSSI_TX_PCS_PMA_INTERFACE_PLDIF_DATAWIDTH_MODE_ADDR_FIELD_OFST = 7;
localparam HSSI_TX_PCS_PMA_INTERFACE_PLDIF_DATAWIDTH_MODE_ADDR_FIELD_HIGH = 7;
localparam HSSI_TX_PCS_PMA_INTERFACE_PLDIF_DATAWIDTH_MODE_ADDR_FIELD_SIZE = 1;
localparam HSSI_TX_PCS_PMA_INTERFACE_PLDIF_DATAWIDTH_MODE_ADDR_FIELD_BITMASK =
    32'h00000080;
localparam HSSI_TX_PCS_PMA_INTERFACE_PLDIF_DATAWIDTH_MODE_ADDR_FIELD_VALMASK =
    32'h00000000;
localparam HSSI_TX_PCS_PMA_INTERFACE_PLDIF_DATAWIDTH_MODE_ADDR_FIELD_VALUE = 1'h0;
```

The SystemVerilog configuration files contain two parts. The first part consists of a data array of 26-bit hexadecimal values. The second part consists of parameter values. For the data array, each 26-bit hexadecimal value is associated with a comment that describes the various bit positions.

Table 175. Mapping of SystemVerilog Configuration File Line

Bit Position	Description
[25:16]	The channel or PLL address.
[15:8]	The channel or PLL bit mask. The bit mask exposes the bits that are configured in either the Transceiver Native PHY or the transmit PLL IP cores.
[7:0]	Feature bit values.

For example, a value of 26'h008FF04 represents an address of 0x008 and a bit mask of 0xFF. The four features that reside at address 0x008 are:

- `hssi_tx_pcs_pma_interface_pldif_datawidth_mode` with a value of 1'h0
- `hssi_tx_pcs_pma_interface_tx_pma_data_sel` with a value of 2'h0
- `hssi_tx_pcs_pma_interface_prbs_gen_pat` with a value of 1'h0
- `hssi_tx_pcs_pma_interface_sq_wave_num` with a value of 4'h4

Writing to bit 7 of address 0x008 changes the `hssi_tx_pcs_pma_interface_pldif_datawidth_mode` feature.

The MIF file and C header file are set up similarly to the SystemVerilog package file. Multiple transceiver features may reside at the same address. Also, a single transceiver feature may span across multiple addresses.

Dynamic reconfiguration requires at least two configurations of the Transceiver Native PHY IP core or PLL IP core. One configuration defines the base transceiver or PLL configuration and the other configurations define the modified or target configurations. Use the IP Parameter Editor to create base and modified configurations of the Transceiver Native PHY or PLL IP core, according to the following table.

Note: You can generate the base and modified configuration files in the same or different folders. If you use the same folder, each configuration name must be unique.

Intel recommends following the flow described in the *Steps to Perform Dynamic Reconfiguration* section when performing dynamic reconfiguration of either the Native PHY IP core or transmit PLL IP core.

Related Information

- Steps to Perform Dynamic Reconfiguration on page 328
- Analog Parameter Settings on page 388

6.4. Multiple Reconfiguration Profiles

You can optionally enable multiple configurations or profiles in the same Native PHY IP, ATX PLL IP, or both core Parameter Editors for performing dynamic reconfiguration. This allows the IP Parameter Editor to create, store, and analyze the parameter settings for multiple configurations or profiles.

When you enable multiple reconfiguration profiles feature, the Native PHY and ATX PLL IP cores can generate configuration files for all the profiles in the format desired (SystemVerilog package, MIF, or C header file). The configuration files are located in the `<IP instance name>/reconfig/` subfolder of the IP instance with the configuration profile index added to the filename. For example, the configuration file for Profile 0 is stored as `<filename_CFG0.sv>`. The Quartus Prime Timing Analyzer includes the necessary timing paths for all the configurations based on initial and target profiles. You can also generate reduced configuration files that contain only the attributes that differ between the multiple configured profiles. You can create up to eight reconfiguration profiles (Profile 0 to Profile 7) at a time for each instance of the Native PHY/ATX PLL IP core.

You can optionally allow the Native PHY IP core to include PMA Analog settings in the configuration files by enabling the feature **Include PMA Analog settings in configuration files** in the **Dynamic Reconfiguration** tab of the Transceiver Native PHY IP Parameter Editor. This feature is disabled by default. Enabling this feature adds the PMA analog settings specified in the **Analog PMA settings (Optional)** tab of the Native PHY IP Parameter Editor to the configuration files. Even with this option enabled in the Native PHY IP Parameter Editor, you must still specify QSF assignments for your analog settings when compiling your static design. The analog settings selected in the Native PHY IP Parameter Editor are used only to include these settings and their dependent settings in the selected configuration files. Refer to the *Analog Parameter Settings* chapter for details about QSF assignments for the analog settings.

Refer to *Steps to Perform Dynamic Reconfiguration* for a complete list of steps to perform dynamic reconfiguration using the IP guided reconfiguration flow with multiple reconfiguration profiles enabled.

The Quartus Prime Timing Analyzer Timing Analyzer only includes the necessary PCS timing paths for all the profiles. To perform a PMA reconfiguration such as TX PLL switching, CGB divider switching, or reference clock switching, you must use the flow described in *Steps to Perform Dynamic Reconfiguration*. Refer to *Timing Closure Recommendations* for more details about enabling multiple profiles and running timing analyses.

You can use the multiple reconfiguration profiles feature without using the embedded reconfiguration streamer feature. When using the multiple reconfiguration profiles feature by itself, you must write the user logic to reconfigure all the entries that are different between the profiles while moving from one profile to another.

Note: You must ensure that none of the profiles in the Native PHY IP and ATX PLL IP Parameter Editor gives error messages, or the IP generation fails. The Native PHY IP core and ATX PLL IP only validates the current active profile dynamically. For example, if you store a profile with error messages in the Native PHY IP or ATX PLL IP Parameter Editor and load another profile without any error messages, the error messages disappear in the IP. You are then allowed to generate the IP, but the generation fails.

Related Information

- Steps to Perform Dynamic Reconfiguration on page 328
- Timing Closure Recommendations on page 369
- Analog Parameter Settings on page 388

6.5. Embedded Reconfiguration Streamer

You can optionally enable the embedded reconfiguration streamer in the Native PHY IP, ATX PLL IP, or both cores to automate the reconfiguration operation. The embedded reconfiguration streamer is a feature block that can perform Avalon-MM transactions to access channel and ATX PLL configuration registers in the transceiver. When you enable the embedded streamer, the Native PHY and ATX PLL IP cores embed HDL code for reconfiguration profile storage and reconfiguration control logic in the IP files.

For the ATX PLL IP, you can control the embedded streamer block through the reconfiguration interface. Control and status signals of the streamer block are memory mapped in the PLL's soft control and status registers.

Table 176. Control and Status Register Memory Map for Embedded Reconfiguration Streamer in ATX PLL IP

Reconfiguration Address (hex)	Reconfiguration Bit	Attribute Name	Attribute Description	Bit Encoding	Transceiver Block	Description
340	7	cfg_load	Start streaming	1'b1	Embedded Reconfiguration Streamer	Set to 1'b1 to initiate streaming, self-clearing bit
	[2:0]	cfg_sel	Configuration profile select	Direct mapped	Embedded Reconfiguration Streamer	Binary encoding of the configuration Profile to stream
341	0	rcfg_busy	Busy Status bit	1'b1	Embedded Reconfiguration Streamer	Bit is set to: • **1'b1**—streaming is in progress • **1'b0**—streaming is complete

Note: The soft control and status registers at x340 and x341 are enabled when you enable the embedded reconfiguration streamer in the ATX PLL IP core.

Refer to *Steps to Perform Dynamic Reconfiguration* for a complete list of steps to perform dynamic reconfiguration using the IP guided reconfiguration flow with embedded streamer enabled. To perform a reference clock switching, use the reconfiguration flow for special cases described in *Steps to Perform Dynamic Reconfiguration*.

For the Native PHY IP, you can control the embedded streamer block through the reconfiguration interface. Control and status signals of the streamer block are memory mapped in the PHY's soft control and status registers. These embedded reconfiguration control and status registers are replicated for each channel. You cannot merge reconfiguration interfaces across multiple IP cores when the embedded reconfiguration streamer is enabled because the embedded reconfiguration streamer makes use of soft logic for control and status registers.

You can optionally allow the Native PHY IP core to include PMA Analog settings in the configuration files by enabling the feature **Include PMA Analog settings in configuration files** in the **Dynamic Reconfirmation** tab of the Transceiver Native PHY IP Parameter Editor. This feature is disabled by default. Enabling this feature adds the PMA analog settings specified in the **Analog PMA settings (Optional)** tab of the Native PHY IP Parameter Editor to the configuration files. Even with this option enabled in the Native PHY IP Parameter Editor, you must still specify QSF assignments for your analog settings when compiling your static design. The analog settings selected in the Native PHY IP Parameter Editor are used only to include these settings and their dependent settings in the selected configuration files. For details about QSF assignments for the analog settings, refer to the *Analog Parameter Settings* chapter.

For example, if the Native PHY IP core has four channels—logical channel 0 to logical channel 3—and you want to reconfigure logical channel 3 using the embedded reconfiguration streamer, you must write to the control register of logical channel 3 using the reconfiguration interface with the appropriate bit settings.

Note: The soft control and status registers at x340 and x341 are enabled when you enable the embedded reconfiguration streamer in the Native PHY IP core.

Refer to *Steps to Perform Dynamic Reconfiguration* for a complete list of steps to perform dynamic reconfiguration using the IP guided reconfiguration flow with embedded streamer enabled. To perform a PMA reconfiguration such as TX PLL switching, CGB divider switching, or reference clock switching, use the reconfiguration flow for special cases described in *Steps to Perform Dynamic Reconfiguration*.

Table 177. Control and Status Register Memory Map for Embedded Reconfiguration Streamer in Native PHY IP

Reconfiguration Address (hex)	Reconfiguration Bit	Attribute Name	Attribute Description	Bit Encoding	Transceiver Block	Description
340	7	cfg_load	Start streaming	1'b1	Embedded Reconfiguration Streamer	Set to 1'b1 to initiate streaming, self-clearing bit
	6	bcast_en	Broadcast enable	1'b1	Embedded Reconfiguration Streamer	Set to 1'b1 to broadcast the same profile to all the channels
	[2:0]	cfg_sel	Configuration profile select	Direct mapped	Embedded Reconfiguration Streamer	Binary encoding of the configuration Profile to stream
341	0	rcfg_busy	Busy Status bit	1'b1	Embedded Reconfiguration Streamer	Bit is set to: • **1'b1**—streaming is in progress • **1'b0**—streaming is complete

You can write to the address 0x340 at the same time to start the streaming, enable broadcasting, and select the profile to be streamed. Different requests to multiple channels can be made simultaneously by writing to the addresses of the desired channels through the user reconfiguration interface (shared or independent). The `reconfig_waitrequest` signal remains asserted after the reconfiguration streaming is complete.

Related Information

- Steps to Perform Dynamic Reconfiguration on page 328
- Analog Parameter Settings on page 388

6.6. Arbitration

Figure 211. Cyclone 10 GX ATX PLL with Embedded Streamer

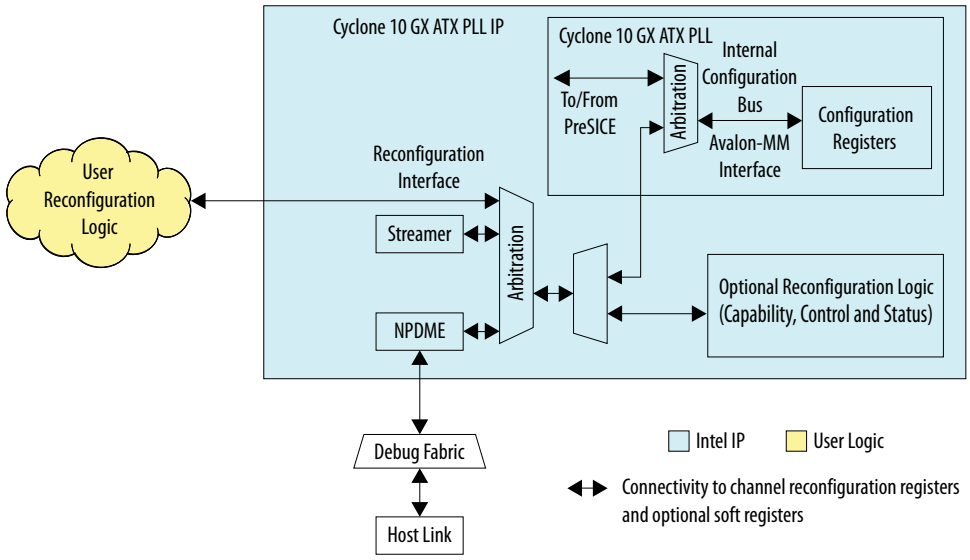
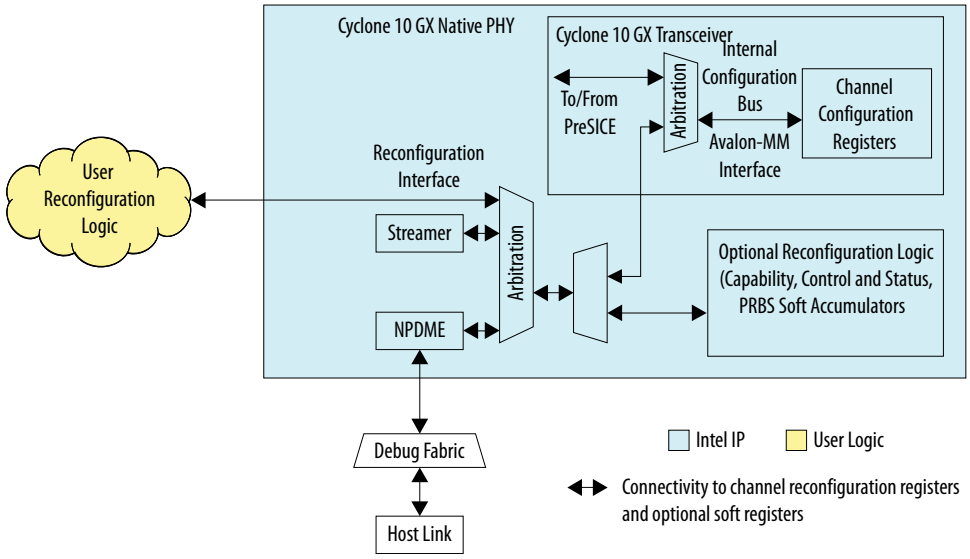


Figure 212. Cyclone 10 GX Native PHY with Embedded Streamer



In Cyclone 10 GX devices, there are two levels of arbitration:

- Reconfiguration interface arbitration with the PreSICE calibration engine

  When you have control over the internal configuration bus, refer to the second level of arbitration: Arbitration between multiple masters within the Native PHY/PLL IPs.

  For more details about arbitration between the reconfiguration interface and PreSICE, refer to the *Calibration* chapter.

- Arbitration between multiple masters within the Native PHY/PLL IPs

  Below are the feature blocks that can access the programmable registers:

  — Embedded reconfiguration streamer (Available in the Native PHY and ATX PLL IPs only)

  — NPDME

  — User reconfiguration logic connected to the reconfiguration interface

  When the internal configuration bus is not owned by the PreSICE, which feature block has access depends on which of them are enabled.

  These feature blocks arbitrate for control over the programmable space of each transceiver channel/PLL. Each of these feature blocks can request access to the programmable registers of a channel/PLL by performing a read or write operation to that channel/PLL. For any of these feature blocks to be used, you must first have control over the internal configuration bus. You must ensure that these feature blocks have completed all the read/write operations before you return the bus access to PreSICE.

  The embedded reconfiguration streamer has the highest priority, followed by the reconfiguration interface, followed by the NPDME. When two feature blocks are trying to access the same transceiver channel on the same clock cycle, the feature block with the highest priority is given access. The only exception is when a lower-priority feature block is in the middle of a read/write operation and a higher-priority feature block tries to access the same channel. In this case, the higher-priority feature block must wait until the lower-priority feature block finishes the read/write operation.

  Note: When you enable NPDME in your design, you must

  — connect an Avalon-MM master to the reconfiguration interface

  — OR connect the `reconfig_clock`,`reconfig_reset` signals and ground the `reconfig_write`, `reconfig_read`, `reconfig_address` and `reconfig_writedata` signals of the reconfiguration interface. If the reconfiguration interface signals are not connected appropriately, there are no clock or reset for the NPDME, and NPDME does not function as expected.

Related Information

- Steps to Perform Dynamic Reconfiguration on page 328
- Calibration on page 373

6.7. Recommendations for Dynamic Reconfiguration

Recommendations for TX PLLs

Intel recommends you control `pll_powerdown` for the fPLL through the soft registers in the following cases:

- Reconfiguring fPLL from integer mode to fractional mode
- Reconfiguring fPLL within fractional mode from one rate to another

For all other reconfiguration scenarios, do not hold the PLL in reset before and during reconfiguration.

When reconfiguring across data rates or protocol modes, Intel recommends that you hold the channel transmitter (analog and digital) associated with the PLL in reset during reconfiguration and recalibration of the PLL.You can use the `tx_digitalreset`, `rx_digitalreset`, `tx_analogreset`, and `rx_analogreset` ports or use the channel soft register for digital and analog resets. For details about placing the channel in analog reset, refer to the "Model 1: Default Model" and "Model 2: Acknowledgment Model" sections of the *Resetting Transceiver Channels* chapter.

Note: If you need to reconfigure the ATX PLL, use TX PLL switching mode or use local clock divider to achieve new data rate to avoid recalibrating the ATX PLL. Refer to "Transmit PLLs Spacing Guidelines when using ATX PLLs and fPLLs" in the "PLLs and Clock Networks" chapter for more details.

Recommendations for Channels

- When reconfiguring across data rates or protocol modes, Intel recommends that you hold the channel transmitter (analog and digital) in reset during reconfiguration and recalibration of the channel transmitter. You can use the `tx_digitalreset`, `rx_digitalreset`, `tx_analogreset`, and `rx_analogreset` ports or use the channel soft register for digital and analog resets. For details about placing the channel in analog reset, refer to the "Model 1: Default Model" and "Model 2: Acknowledgment Model" sections of the *Resetting Transceiver Channels* chapter.
- When reconfiguring across data rates or protocol modes, Intel recommends that you hold the channel receiver (analog and digital) in reset during reconfiguration and recalibration of the channel receiver. You can use the `tx_digitalreset`, `rx_digitalreset`, `tx_analogreset`, and `rx_analogreset` ports or use the channel soft register for digital and analog resets. For details about placing the channel in analog reset, refer to the "Model 1: Default Model" and "Model 2: Acknowledgment Model" sections of the *Resetting Transceiver Channels* chapter.
- When performing reconfiguration on channels not involving data rate or protocol mode change, Intel recommends that you hold the channel transmitter (digital only) in reset during reconfiguration.
- When performing reconfiguration on channels not involving data rate or protocol mode change, Intel recommends that you hold the channel receiver (digital only) in reset during reconfiguration.

Related Information

- Model 1: Default Model on page 245
- Model 2: Acknowledgment Model on page 254

- Transmit PLLs Spacing Guidelines when using ATX PLLs and fPLLs on page 200

6.8. Steps to Perform Dynamic Reconfiguration

You can dynamically reconfigure blocks in the transceiver channel or PLL through the reconfiguration interface.

The following procedure shows the steps required to reconfigure the channel and PLL blocks.

1. Enable dynamic reconfiguration in the IP.
2. Enable the desired configuration file formats in the IP.
3. Enable the desired dynamic reconfiguration features (such as multiple reconfiguration profiles, including PMA analog settings in configuration files) or feature blocks (such as embedded reconfiguration streamer and NPDME).
4. If you are using:
   - Direct reconfiguration flow—Refer to the register map for feature address and valid value of write data for the feature.
   - IP guided reconfiguration flow—Note the settings of the base configuration and generate the corresponding configuration files. Note the settings of the modified configuration and generate the corresponding configuration files. Find out the differences in settings between the base and modified configurations.
   - IP guided reconfiguration flow using multiple profiles—Create and store the parameter settings between the various configurations or profiles using configuration files. Find out the differences in settings between the various configurations or profiles using configuration files.
   - IP guided reconfiguration flow using the embedded streamer—Refer to the control and status register map of the embedded reconfiguration streamer to stream the desired profile settings.
   - Reconfiguration flow for special cases—Refer to the lookup registers to be accessed for each special case, such as TX PLL switching, TX PLL reference clock switching, and RX CDR reference clock switching.
5. Place the channels in digital reset either simultaneously or one after another. For details about placing the channel in reset, refer to "Model 1: Default Model" and "Model 2: Acknowledgment Model" in the *Resetting Transceiver Channels* chapter.

   If you are reconfiguring:
   - PLLs—Place the channel transmitter associated with the PLL in reset (digital).
   - TX simplex channels—Place the TX channels being reconfigured in reset (digital).
   - RX simplex channels—Place the RX channels being reconfigured in reset (digital).
   - Duplex channels—Place the channel TX and RX being reconfigured in reset (digital).
6. If you are reconfiguring across data rates or protocol modes or enabling/disabling PRBS, place the channels in analog reset. For details about placing the channel in analog reset, refer to "Model 1: Default Model" and "Model 2: Acknowledgment Model" in the *Resetting Transceiver Channels* chapter.

If you are reconfiguring:

- PLLs—Place the channel transmitter associated with the PLL in reset (analog).
- TX simplex channels—Place the TX channels being reconfigured in reset (analog).
- RX simplex channels—Place the RX channels being reconfigured in reset (analog).
- Duplex channels—Place the channel TX and RX being reconfigured in reset (analog).

7. Check for internal configuration bus arbitration. If PreSICE has control, request bus arbitration, otherwise go to the next step. For more details, refer to the "Arbitration" section.
8. Perform the necessary reconfiguration using the flow described in the following sections:
   - *Direct Reconfiguration Flow*
   - *Native PHY or PLL IP Guided Reconfiguration Flow*
   - *Reconfiguration Flow for Special Cases*
9. Perform all necessary reconfiguration. If reconfiguration involved data rate or protocol mode changes, then you may have to reconfigure the PMA analog parameters of the channels. Refer to the *Changing PMA Analog Parameters* section for more details.
10. If reconfiguration involved data rate or protocol mode change, then request recalibration and wait for the calibration to complete. Calibration is complete when `*_cal_busy` is deasserted. For more details about calibration registers and the steps to perform recalibration, refer to the *Calibration* chapter.

    If you reconfigured:

    - PLL for data rate change—you must recalibrate the PLL and the channel TX.
    - TX simplex channel for data rate change—you must recalibrate the channel TX.
    - RX simplex channel for data rate change—you must recalibrate the channel RX.
    - Duplex channel for data rate change—you must recalibrate the channel TX and RX.
11. Release the channel analog resets. For details about placing the channel in reset, refer to "Model 1: Default Model" and "Model 2: Acknowledgment Model" in the *Resetting Transceiver Channels* chapter.

If you reconfigured:

- PLLs—Release the reset (analog) of the channel transmitters associated with the PLL reconfigured.
- TX simplex channels—Release the reset (analog) of the TX channels reconfigured.
- RX simplex channels—Release the reset (analog) of the RX channels reconfigured.
- Duplex channels—Release the reset (analog) of the TX and RX channels reconfigured.

12. Release the channel digital resets either simultaneously or one after another. For details about releasing the channel resets, refer to "Model 1: Default Model" and "Model 2: Acknowledgment Model" in the *Resetting Transceiver Channels* chapter. (The figures in these sections are for analog resets, but they also contain timing information about digital resets.)

    If you reconfigured:

    - PLLs—Release the reset (digital) of the channel transmitters associated with the PLL reconfigured.
    - TX simplex channels—Release the reset (digital) of the TX channels reconfigured.
    - RX simplex channels—Release the reset (digital) of the RX channels reconfigured.
    - Duplex channels—Release the reset (digital) of the TX and RX channels reconfigured.

Note: You cannot merge multiple reconfiguration interfaces across multiple IP blocks (merging independent instances of simplex TX/RX into the same physical location or merging separate CMU PLL and TX channel into the same physical location) when you use the optional reconfiguration logic soft control registers.

Related Information

- Resetting Transceiver Channels on page 243
- Model 1: Default Model on page 245
- Model 2: Acknowledgment Model on page 254
- Direct Reconfiguration Flow on page 330
- Native PHY IP or PLL IP Core Guided Reconfiguration Flow on page 331
- Reconfiguration Flow for Special Cases on page 333
- Changing PMA Analog Parameters on page 338
- Calibration on page 373
- Arbitration on page 325

6.9. Direct Reconfiguration Flow

Use this flow to perform dynamic reconfiguration when you know exactly which parameter and value to change for the transceiver channel or PLL. You can use this flow to change the PMA analog settings, enable/disable PRBS generator, and checker hard blocks of the transceiver channel.

To perform dynamic reconfiguration using direct reconfiguration flow:

1. Perform the necessary steps from steps 1 to 7 in *Steps to Perform Dynamic Reconfiguration*.
2. Read from the desired feature address.
3. Perform a read-modify-write to feature address with a valid value.
4. Perform the necessary steps from steps 9 to 12 in *Steps to Perform Dynamic Reconfiguration*.

Related Information

- Steps to Perform Dynamic Reconfiguration on page 328
- Changing PMA Analog Parameters on page 338
- Using Data Pattern Generators and Checkers on page 359
- Resetting Transceiver Channels on page 243
- Calibration on page 373

6.10. Native PHY IP or PLL IP Core Guided Reconfiguration Flow

Use the Native PHY IP core or PLL IP core guided reconfiguration flow to perform dynamic reconfiguration when you need to change multiple parameters or parameters in multiple addresses for the transceiver channel or PLL. You can use this flow to change data rates, change clock divider values, or switch from one PCS datapath to another. You must generate the required configuration files for the base and modified Transceiver Native PHY IP core or PLL IP core configurations.

The configuration files contain addresses and bit values of the corresponding configuration. Compare the differences between the base and modified configuration files. The differences between these files indicate the addresses and bit values that must change to switch from one configuration to another. Perform read-modify-writes for the bit values that are different from the base configuration to obtain the modified configuration.

To perform dynamic reconfiguration using the IP Guided Reconfiguration Flow:

1. Perform the necessary steps from steps 1 to 7 in *Steps to Perform Dynamic Reconfiguration*.
2. Perform a read-modify-write to all addresses and bit values that are different from the base configuration.
3. Perform the necessary steps from steps 9 to 12 in *Steps to Perform Dynamic Reconfiguration*.

Note: If reconfiguration involved data rate or protocol mode changes, you may need to reconfigure the PMA analog parameters of the channels. Refer to the *Changing PMA Analog Parameters* section for more details.

The bit values that must be changed to obtain the new configuration may span across multiple addresses, such as when switching between Standard, Enhanced, and PCS Direct data paths. It is difficult to manually compare these values for the base and modified configurations and then build logic to stream the different values in the modified configuration. You can use the multiple profiles feature of the Native PHY/ATX PLL IP cores to store the parameter settings (MIF configuration file) to memory. With the configuration content saved, you can read from the memory and write the content

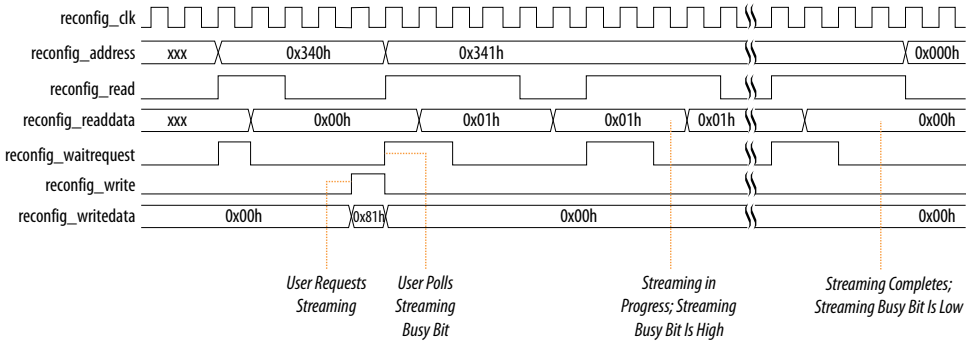
to the target channel for reconfiguration. Optionally, you can also use the embedded reconfiguration streamer feature of the Native PHY/ATX PLL IP cores, which includes the logic to store the individual profile information and logic to perform streaming. Using the embedded reconfiguration streamer, you can reduce the number of read-modify-write operations to obtain the modified configuration.

To perform dynamic reconfiguration using the Embedded Reconfiguration Streamer:

1. Perform the necessary steps from steps 1 to 7 in *Steps to Perform Dynamic Reconfiguration*.
2. Perform a read-modify-write to address x340 with the desired profile select, broadcast bit (applicable for Native PHY only), and configuration load bit set accordingly. For example, to stream profile 1 to a channel, perform a read-modify-write to bits x340[2:0] with 3'b001, bit x340[6] with 1'b0 to disable broadcasting, and bit x340[7] with 1'b1 to initiate streaming.
3. Poll the streamer busy bit at address x341 (x341[0]) at regular intervals. When the busy bit is 1'b0, the reconfiguration is complete.
4. Perform the necessary steps from steps 9 to 12 in *Steps to Perform Dynamic Reconfiguration*.

Note: If reconfiguration involved data rate or protocol mode changes, you may need to reconfigure the PMA analog parameters of the channels. Refer to the *Changing PMA Analog Parameters* section for more details.

Figure 213. Timing Diagram for Embedded Streamer Reconfiguration



Related Information

- Arbitration on page 325
- Changing PMA Analog Parameters on page 338
- Steps to Perform Dynamic Reconfiguration on page 328
- Resetting Transceiver Channels on page 243
- Calibration on page 373

6.11. Reconfiguration Flow for Special Cases

Dynamic reconfiguration can be performed on logical operations such as switching between multiple transmit PLLs or multiple reference clocks. In these cases, configuration files alone cannot be used. Configuration files are generated during IP generation and do not contain information on the placement of PLLs or reference clocks.

To perform dynamic reconfiguration on logical operations, you must use lookup registers that contain information about logical index to physical index mapping. Lookup registers are read-only registers. Use these lookup registers to perform a read-modify-write to the selection MUXes to switch between PLLs or reference clocks.

To perform dynamic reconfiguration using reconfiguration flow for special cases:

1. Perform the necessary steps from steps 1 to 7 in *Steps to Perform Dynamic Reconfiguration*.
2. Read from the desired lookup register. Refer to the *Switching Transmitter PLL* and *Switching Reference Clocks* sections for information about lookup registers.
3. Perform Logical Encoding (only required for Transmitter PLL switching).
4. Perform read-modify-write to the required feature address with the desired/encoded value.
5. Perform the necessary steps from steps 9 to 12 in *Steps to Perform Dynamic Reconfiguration*.

Related Information

- Steps to Perform Dynamic Reconfiguration on page 328
- Switching Transmitter PLL on page 333
- Switching Reference Clocks on page 335
- Resetting Transceiver Channels on page 243
- Calibration on page 373

6.11.1. Switching Transmitter PLL

Dynamically switching data rates increases system flexibility to support multiple protocols. You can change the transceiver channel data rate by switching from one transmit PLL to another. To switch between transmit PLLs, you must reconfigure the local CGB MUX select lines of the channel by performing a channel reconfiguration. You can clock transceiver channels with up to four different transmitter PLLs. You can use the reconfiguration interface on the Native PHY IP core to specify which PLL drives the transceiver channel. The PLL switching method is the same, regardless of the number of transmitter PLLs involved.

Before initiating the PLL switch procedure, ensure that your Transceiver Native PHY instance defines more than one transmitter PLL input. Specify the **Number of TX PLL clock inputs per channel** parameter on the **TX PMA** tab during Transceiver Native PHY parameterization.

The following table shows the addresses and bits for transmitter PLL switching. The number of exposed `tx_serial_clk` bits varies according to the number of transmitter PLLs you specify. Use the Native PHY reconfiguration interface for this operation.

Table 178. Register Map for Switching Transmitter PLLs

Transceiver Native PHY Port	Description	Address	Bits
tx_serial_clk0	Represents logical PLL0. Lookup register x117[3:0] stores the mapping from logical PLL0 to the physical PLL.	0x117 (Lookup Register)	[3:0]
tx_serial_clk1	Represents logical PLL1. Lookup register x117[7:4] stores the mapping from logical PLL1 to the physical PLL.	0x117 (Lookup Register)	[7:4]
tx_serial_clk2	Represents logical PLL2. Lookup register x118[3:0] stores the mapping from logical PLL2 to the physical PLL.	0x118 (Lookup Register)	[3:0]
tx_serial_clk3	Represents logical PLL3. Lookup register x118[7:4] stores the mapping from logical PLL3 to the physical PLL.	0x118 (Lookup Register)	[7:4]
N/A	PLL selection MUX	0x111	[7:0]

When performing a PLL switch, you must specify the lookup register address and bit values you want to switch to. The following procedure describes selection of a specific transmitter PLL when more than one PLL is connected to a channel. To change the data rate of the CDR, follow the detailed steps for reconfiguring channel and PLL blocks. After determining the logical PLL to switch to, follow this procedure to switch to the desired transmitter PLL:

1. Perform the necessary steps from steps 1 to 7 in *Steps to Perform Dynamic Reconfiguration*.
2. Read from the appropriate lookup register address (refer to Table 178 on page 334) and save the required 4-bit pattern. For example, switching to logical PLL1 requires saving bits [7:4] of address 0x117.
3. Encode the 4-bit value read in the previous step into an 8-bit value according to the following table:

Table 179. Logical PLL Encoding

4-bit Logical PLL Bits	8-bit Mapping to Address 0x111
[3..0]	*{~logical_PLL_offset_readdata[3], logical_PLL_offset_readdata[1:0],logical_PLL_offset_readdata[3], logical_PLL_offset_readdata[3:0] }*
[7..4]	*{~logical_PLL_offset_readdata[7], logical_PLL_offset_readdata[5:4],logical_PLL_offset_readdata[7], logical_PLL_offset_readdata[7:4] }*

Note: For example, if reconfiguring to logical PLL1 then bits [7:4] are encoded to an 8-bit value {~bit[7], bit[5:4], bit[7], bit[7:4]}.

4. Perform a read-modify-write to bits[7:0] of address 0x111 using the encoded 8-bit value.
5. Perform the necessary steps from steps 9 to 12 in *Steps to Perform Dynamic Reconfiguration*.

Related Information

Steps to Perform Dynamic Reconfiguration on page 328

6.11.2. Switching Reference Clocks

You can dynamically switch the input clock source for the ATX PLL, the fPLL, the CDR, and the CMU.

6.11.2.1. ATX Reference Clock Switching

You can use the reconfiguration interface on the ATX PLL instance to specify which reference clock source drives the ATX PLL. The ATX PLL supports clocking up to five different reference clock sources. The flow to select between the different reference clock sources is independent of the number of transmitter PLLs specified in the Parameter Editor.

Before initiating a reference clock switch, ensure that your ATX PLL instance defines more than one reference clock source. Specify the **Number of PLL reference clocks** parameter on the **PLL** tab during ATX PLL parameterization.

The following table shows the addresses and bits for switching between ATX PLL reference clock inputs. The number of exposed `pll_refclk` ports varies according to the number of reference clocks you specify. Use the ATX PLL reconfiguration interface for this operation.

Table 180. Register Map for Switching ATX PLL Reference Clock Inputs

Transceiver ATX PLL Port	Description	Address	Bits
`pll_refclk0`	Represents logical `refclk0`. Lookup register `x113[7:0]` stores the mapping from logical `refclk0` to the physical refclk.	0x113 (Lookup Register)	[7:0]
`pll_refclk1`	Represents logical `refclk1`. Lookup register `x114[7:0]` stores the mapping from logical `refclk1` to the physical refclk.	0x114 (Lookup Register)	[7:0]
`pll_refclk2`	Represents logical `refclk2`. Lookup register `x115[7:0]` stores the mapping from logical `refclk2` to the physical refclk.	0x115 (Lookup Register)	[7:0]
`pll_refclk3`	Represents logical `refclk3`. Lookup register `x116[7:0]` stores the mapping from logical `refclk3` to the physical refclk.	0x116 (Lookup Register)	[7:0]
`pll_refclk4`	Represents logical `refclk4`. Lookup register `x117[7:0]` stores the mapping from logical `refclk4` to the physical refclk.	0x117 (Lookup Register)	[7:0]
N/A	ATX refclk selection MUX.	0x112	[7:0]

When performing a reference clock switch, you must specify the lookup register address and respective bits of the replacement clock. After determining the ATX PLL, follow this procedure to switch to the selected reference clock:

1. Perform the necessary steps from steps 1 to 7 in *Steps to Perform Dynamic Reconfiguration*.
2. Read from the lookup register address and save the required 8-bit pattern. For example, switching to logical `refclk2` requires use of bits`[7:0]` at address `0x115`.
3. Perform a read-modify-write to bits `[7:0]` at address `0x112` using the 8-bit value obtained from the lookup register.
4. Perform the necessary steps from steps 9 to 12 in *Steps to Perform Dynamic Reconfiguration*.

Related Information

Steps to Perform Dynamic Reconfiguration on page 328

6.11.2.2. fPLL Reference Clock Switching

You can use the reconfiguration interface on the fPLL instance to specify which reference clock source drives the fPLL. The fPLL supports clocking by up to five different reference clock sources. The flow to select between the different reference clock sources is independent of the number of transmitter PLLs specified in the reconfiguration interface.

Before initiating a reference clock switch, ensure that your fPLL instance defines more than one reference clock source. Specify the **Number of PLL reference clocks** parameter on the **PLL** tab during fPLL parameterization.

The following table shows the addresses and bits for switching between fPLL reference clock inputs. The number of exposed `pll_refclk` ports varies according to the number of reference clocks you specify. Use the fPLL reconfiguration interface for this operation.

Table 181. Register Map for Switching fPLL Reference Clock Inputs

Transceiver fPLL Port	Description	Address	Bits
`pll_refclk0`	Represents logical `refclk0` for `MUX_0`. Lookup register `x117[4:0]` stores the mapping from logical `refclk0` to the physical refclk for `MUX_0`.	0x117 (Lookup Register)	[7:0]
`pll_refclk1`	Represents logical `refclk1` for `MUX_0`. Lookup register `x118[4:0]` stores the mapping from logical `refclk1` to the physical refclk for `MUX_0`.	0x118 (Lookup Register)	[7:0]
`pll_refclk2`	Represents logical `refclk2` for `MUX_0`. Lookup register `x119[4:0]` stores the mapping from logical `refclk2` to the physical refclk for `MUX_0`.	0x119 (Lookup Register)	[7:0]
`pll_refclk3`	Represents logical `refclk3` for `MUX_0`. Lookup register `x11A[4:0]` stores the mapping from logical `refclk3` to the physical refclk for `MUX_0`.	0x11A (Lookup Register)	[7:0]
`pll_refclk4`	Represents logical `refclk4` for `MUX_0`. Lookup register `x11B[4:0]` stores the mapping from logical `refclk4` to the physical refclk for `MUX_0`.	0x11B (Lookup Register)	[7:0]

continued...

Transceiver fPLL Port	Description	Address	Bits
N/A	fPLL refclk selection MUX_0.	0x114	[7:0]
pll_refclk0	Represents logical refclk0 for MUX_1. Lookup register x11D[4:0] stores the mapping from logical refclk0 to the physical refclk for MUX_1.	0x11D (Lookup Register)	[7:0]
pll_refclk1	Represents logical refclk1 for MUX_1. Lookup register x11E[4:0] stores the mapping from logical refclk1 to the physical refclk for MUX_1.	0x11E (Lookup Register)	[7:0]
pll_refclk2	Represents logical refclk2 for MUX_1. Lookup register x11F[4:0] stores the mapping from logical refclk2 to the physical refclk for MUX_1.	0x11F (Lookup Register)	[7:0]
pll_refclk3	Represents logical refclk3 for MUX_1. Lookup register x120[4:0] stores the mapping from logical refclk3 to the physical refclk for MUX_1.	0x120 (Lookup Register)	[7:0]
pll_refclk4	Represents logical refclk4 for MUX_1. Lookup register x121[4:0] stores the mapping from logical refclk4 to the physical refclk for MUX_1.	0x121 (Lookup Register)	[7:0]
N/A	fPLL refclk selection MUX_1.	0x11C	[7:0]

Specify the logical reference clock and respective address and bits of the replacement clock when performing a reference clock switch. Follow this procedure to switch to the selected reference clock:

1. Perform the necessary steps from steps 1 to 7 in *Steps to Perform Dynamic Reconfiguration*.
2. Read from the lookup register for MUX 0 and save the required 5-bit pattern. For example, switching to logical refclk3 requires use of bits[4:0] at address 0x11A.
3. Perform a read-modify-write to bits [4:0] at address 0x114 using the 5-bit value obtained from the lookup register.
4. Read from the lookup register for MUX 1 and save the required 5-bit pattern. For example, switching to logical refclk3 requires use of bits[4:0] at address 0x120.
5. Perform a read-modify-write to bits [4:0] at address 0x11C using the 5-bit value obtained from the lookup register.
6. Perform the necessary steps from steps 9 to 12 in *Steps to Perform Dynamic Reconfiguration*.

Related Information

Steps to Perform Dynamic Reconfiguration on page 328

6.11.2.3. CDR and CMU Reference Clock Switching

You can use the reconfiguration interface to specify which reference clock source drives the CDR and CMU PLL. The CDR and CMU support clocking by up to five different reference clock sources.

Before initiating a reference clock switch, ensure that your CDR and CMU defines more than one reference clock source. For the CDR, specify the parameter on the **RX PMA** tab during the Native PHY IP parameterization. For the CMU, specify the **Number of PLL reference clocks** under the **PLL** tab when parameterizing the CMU PLL.

The following table describes the addresses and bits for switching CDR and CMU reference clock inputs. The number of exposed `rx_cdr_refclk` (CDR) or `pll_refclk` (CMU) varies according to the number of reference clocks you specify. Use the CMU reconfiguration interface for switching the CMU reference clock.

Table 182. Register Map for Switching CDR Reference Clock Inputs

Native PHY Port	Description	Address	Bits
`cdr_refclk0`	Represents logical `refclk0`. Lookup register `x16A[7:0]` stores the mapping from logical `refclk0` to the physical refclk.	0x16A (Lookup Register)	[7:0]
`cdr_refclk1`	Represents logical `refclk1`. Lookup register `x16B[7:0]` stores the mapping from logical `refclk1` to the physical refclk.	0x16B (Lookup Register)	[7:0]
`cdr_refclk2`	Represents logical `refclk2`. Lookup register `x16C[7:0]` stores the mapping from logical `refclk2` to the physical refclk.	0x16C (Lookup Register)	[7:0]
`cdr_refclk3`	Represents logical `refclk3`. Lookup register `x16D[7:0]` stores the mapping from logical `refclk3` to the physical refclk.	0x16D (Lookup Register)	[7:0]
`cdr_refclk4`	Represents logical `refclk4`. Lookup register `x16E[7:0]` stores the mapping from logical `refclk4` to the physical refclk.	0x16E (Lookup Register)	[7:0]
N/A	CDR refclk selection MUX.	0x141	[7:0]

When performing a reference clock switch, note the logical reference clock to switch to and the respective address and bits. After determining the logical reference clock, follow this procedure to switch to the selected CDR reference clock:

1. Perform the necessary steps from steps 1 to 7 in *Steps to Perform Dynamic Reconfiguration*.
2. Read from the lookup register and save the required 8-bit pattern. For example, switching to logical `refclk3` requires saving bits`[7:0]` at address `0x16D`.
3. Perform a read-modify-write to bits `[7:0]` at address `0x141` using the 8-bit value obtained from the lookup register.
4. Perform the necessary steps from steps 9 to 12 in *Steps to Perform Dynamic Reconfiguration*.

Related Information

Steps to Perform Dynamic Reconfiguration on page 328

6.12. Changing PMA Analog Parameters

You can use the reconfiguration interface on the Transceiver Native PHY IP core to change the value of PMA analog features.

The PMA analog settings can be broadly divided into the following groups:

- PMA analog settings that are channel or system dependent:
  - These settings may vary from channel to channel based on channel loss or other factors.
  - You can set these PMA analog settings based on IBIS-AMI or Advanced Link Analyzer simulations.
  - You can set these PMA analog settings using QSF assignments or by performing RMWs to the respective registers.
  - These PMA analog settings are not included in the configuration files by default. To include these PMA analog settings in the configuration files, you must enable the **Include PMA Analog settings in configuration files** option in the **Dynamic Reconfirmation** tab of the Transceiver Native PHY IP Parameter Editor. Enabling this feature adds the PMA analog settings specified in the **Analog PMA settings (Optional)** tab of the Native PHY IP Parameter Editor to the configuration files. Even with this option enabled in the Native PHY IP Parameter Editor, you must still specify QSF assignments for your analog settings when compiling your static design. The analog settings selected in the Native PHY IP Parameter Editor are used only to include these settings and their dependent settings in the selected configuration files. For details about optional analog settings, refer to the "Analog PMA Settings (Optional) for Dynamic Reconfiguration" table in the *Ports and Parameters* section. For details about QSF assignments for the analog settings, refer to the *Analog Parameter Settings* chapter.
  - If you do not enable the **Include PMA Analog settings in configuration files** option, then you can change these analog settings by performing RMWs using direct reconfiguration flow.

Table 183. PMA Analog Settings that are Channel or System Dependent

PMA Analog Feature	Fitter Report Name	Cyclone 10 GX Transceiver Register Map Attribute Name
VOD	vod_output_swing_ctrl	vod_output_swing_ctrl
Pre-emphasis	pre_emp_sign_1st_post_tap	pre_emp_sign_1st_post_tap
	pre_emp_sign_2nd_post_tap	pre_emp_sign_2nd_post_tap
	pre_emp_sign_pre_tap_1t	pre_emp_sign_pre_tap_1t
	pre_emp_sign_pre_tap_2t	pre_emp_sign_pre_tap_2t
	pre_emp_switching_ctrl_1st_post_tap	pre_emp_switching_ctrl_1st_post_tap
	pre_emp_switching_ctrl_2nd_post_tap	pre_emp_switching_ctrl_2nd_post_tap
	pre_emp_switching_ctrl_pre_tap_1t	pre_emp_switching_ctrl_pre_tap_1t
	pre_emp_switching_ctrl_pre_tap_2t	pre_emp_switching_ctrl_pre_tap_2t
CTLE	eq_dc_gain_trim	eq_dc_gain_trim
	eq_bw_sel	eq_bw_sel
	adp_ctle_acgain_4s	adp_ctle_acgain_4s
VGA	adp_vga_sel	adp_vga_sel

- PMA analog settings that are device dependent
  - These settings may vary for each transceiver protocol-type and data rate in your design.
  - These settings are not included in the configuration files by default. To include these analog settings in the configuration files, you must enable the feature **Include PMA Analog settings in configuration files** under the **Dynamic Reconfiguration** tab of the Transceiver Native PHY IP Parameter Editor. Enabling this feature adds the PMA analog settings specified in the **Analog PMA settings (Optional)** tab of Native PHY IP Parameter Editor to the configuration files. Even with this option enabled in the Native PHY IP Parameter Editor, you must still specify QSF assignments for your analog settings when compiling your static design. The analog settings selected in the Native PHY IP Parameter Editor are used only to include these settings and their dependent settings in the selected configuration files. For details about optional analog settings, refer to the "Analog PMA Settings (Optional) for Dynamic Reconfiguration" table in the *Ports and Parameters* section. For details about QSF assignments for the analog settings, refer to the *Analog Parameter Settings* chapter.
  - If the **Include PMA analog settings in configuration files** option is disabled, then you must set these PMA analog settings. In addition to streaming the configuration files generated by the Native PHY IP Parameter Editor, you must perform RMWs using Direct Reconfiguration Flow to change these PMA analog settings through the Avalon-MM reconfiguration interface.
  - The values of all these PMA analog settings that change when changing protocol-type or data rates must be obtained from the respective Fitter reports by performing full compilation for each of the base and target configurations.
  - For example, when changing the data rate from A to B, you must first perform a full compile with the data rate configured to A and note the PMA analog settings from the fitter report. Next, you must perform a full compile with data rate configured to B and note the PMA analog settings from the fitter report. If any of these PMA analog settings changed values between the two compiles, you must perform RMWs with the target values to the respective registers after streaming the configuration files.
  - Examples: Slew rate, Equalizer Bandwidth, Compensation Enable.

Table 184. PMA Analog Settings that are Device Dependent

PMA Analog Feature	Fitter Report Name	Cyclone 10 GX Transceiver Register Map Attribute Name
Slew Rate (TX Buffer)	Slew_rate_ctrl	Slew_rate_ctrl
Equalizer Bandwidth (RX Buffer)	Eq_bw_sel	Eq_bw_sel
Compensation Enable (TX Buffer)	Compensation_en	Compensation_en

Related Information

- Ports and Parameters on page 346
- Analog Parameter Settings on page 388

6.12.1. Changing VOD, Pre-emphasis Using Direct Reconfiguration Flow

1. Perform the necessary steps from steps 1 to 7 in *Steps to Perform Dynamic Reconfiguration*.
2. Read from the PMA analog feature address of the channel you want to change. For example, to change pre-emphasis 1st post-tap, read and store the value of address 0x105.
3. Select a valid value for the feature according to the Cyclone 10 GX register map. For example, a valid setting for pre-emphasis 1st post-tap has a bit encoding of 5'b00001.
4. Perform a read-modify-write to the address of the PMA analog feature using the valid value. For example, to change the pre-emphasis 1st post-tap, write 5'b00001 to address 0x105.
5. Perform the necessary steps from steps 9 to 12 in *Steps to Perform Dynamic Reconfiguration*.

Table 185. Register Map for PMA Analog Feature

PMA Analog Feature	Address	Bit	Values
Pre-emphasis 1st post-tap	0x105	[4:0]	5'b00000 - 5'b11001
Pre-emphasis 1st post-tap polarity	0x105	[6]	1'b0 = positive 1'b1 = negative
Pre-emphasis 2nd post-tap	0x106	[3:0]	4'b0000 - 4'b1100
Pre-emphasis 2nd post-tap polarity	0x106	[5]	1'b0 = positive 1'b1 = negative
Pre-emphasis 1st pre-tap	0x107	[4:0]	5'b00000 - 5'b10000
Pre-emphasis 1st pre-tap polarity	0x107	[5]	1'b0 = positive 1'b1 = negative
Pre-emphasis 2nd pre-tap	0x108	[2:0]	3'b000 - 3'b111
Pre-emphasis 2nd pre-tap polarity	0x108	[4]	1'b0 = positive 1'b1 = negative
Differential output voltage (V_{OD})	0x109	[4:0]	5'b00000 - 5'b11111

The PMA analog settings are governed by a set of rules. Not all combinations of V_{OD} and pre-emphasis are valid. The following table lists the maximum pre-emphasis settings for the corresponding V_{OD} settings.

Table 186. Valid Maximum (Absolute) Pre-Emphasis Settings

V_{OD}	Maximum (Absolute) Pre-Emphasis Settings			
	1st Post-Tap	1st Pre-Tap	2nd Post-Tap	2nd Pre-Tap
31	25	16	12	7
30	25	16	11	6
29	25	16	10	5
28	25	16	9	4
27	25	16	8	3

continued...

V_{OD}	Maximum (Absolute) Pre-Emphasis Settings			
	1st Post-Tap	1st Pre-Tap	2nd Post-Tap	2nd Pre-Tap
26	25	16	7	2
25	25	16	6	1
24	25	16	5	0
23	24	16	4	0
22	23	16	3	0
21	22	16	2	0
20	21	16	1	0
19	20	16	0	0
18	19	15	0	0
17	18	14	0	0
16	17	13	0	0
15	16	12	0	0
14	15	11	0	0
13	14	10	0	0
12	13	9	0	0

Related Information

- Steps to Perform Dynamic Reconfiguration on page 328
- Cyclone 10 GX PMA Architecture on page 272

6.12.2. Changing CTLE Settings in Manual Mode Using Direct Reconfiguration Flow

You can use the reconfiguration interface on the Transceiver Native PHY IP core to change the CTLE settings in manual mode.

1. Perform the necessary steps from steps 1 to 7 in *Steps to Perform Dynamic Reconfiguration*.
2. Read from the CTLE feature address of the channel you want to change. For example, to change CTLE AC gain in high gain mode, read and store the value of address 0x167[5:1].
3. Select a valid value for the feature according to the Cyclone 10 GX register map. For example, a valid setting for CTLE AC Gain has a bit encoding of 5'b00000.
4. Perform a read-modify-write to the address of the CTLE feature using the valid value. For example, to change the CTLE AC gain in high gain mode, write 5'b00000 to address 0x167[5:1].
5. Perform the necessary steps from steps 9 to 12 in *Steps to Perform Dynamic Reconfiguration*.

Table 187. Register Map for CTLE Settings

CTLE Feature	Address	Bits	Values	Description
DC Gain	0x11C, 0x11A	[3:0], [7:0]	12'b000000000000 12'b111000000000 12'b111111000000 12'b111111111000 12'b111111111111	Sets the DC gain values. This register can only be controlled when in Four stage mode.
CTLE AC Gain Four Stage	0x167	[5:1]	5'b00000 – 5'b11100	Sets the AC gain value when four stage mode (High gain mode) is selected.
VGA SEL	0x160	[3:1]	3'b000 – 3'b100	Sets the VGA Gain value

Related Information

- Steps to Perform Dynamic Reconfiguration on page 328
- Cyclone 10 GX PMA Architecture on page 272

6.12.3. Enabling and Disabling Loopback Modes Using Direct Reconfiguration Flow

Cyclone 10 GX devices have three loopback modes:

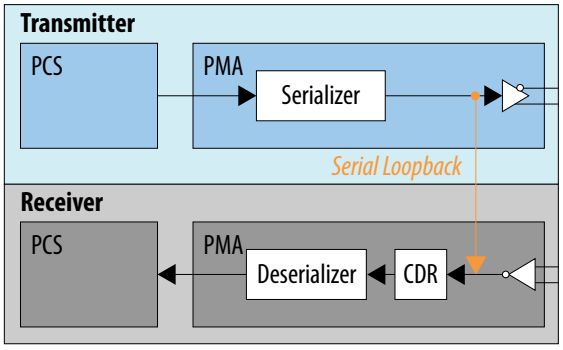
- Serial Loopback
- Reverse Serial Loopback (Pre-CDR)
- Reverse Serial Loopback (Post-CDR)

The loopback mode can be dynamically reconfigured by accessing the register space.

Serial Loopback Mode

In serial loopback mode, a path exists between the serializer of the transmitter and the CDR of the receiver, so that the data from the CDR is recovered from the serializer while the data from the receiver serial input pin is ignored. You can enable or disable this mode.

Figure 214. Serial Loopback Mode



To enable serial loopback mode:

1. Perform the necessary steps from steps 1 to 7 in *Steps to Perform Dynamic Reconfiguration*.
2. Perform a read-modify-write to address 0x2E1 to set bit 0 to 1'b1
3. Perform the necessary steps from steps 9 to 12 in *Steps to Perform Dynamic Reconfiguration*.

To disable serial loopback mode:

1. Perform the necessary steps from steps 1 to 7 in *Steps to Perform Dynamic Reconfiguration*.
2. Perform a read-modify-write to address 0x2E1 to set bit 0 to 1'b0
3. Perform the necessary steps from steps 9 to 12 in *Steps to Perform Dynamic Reconfiguration*.

You can also enable the serial loopback mode by turning on **Enable rx_seriallpbken port** in the Native PHY IP Parameter Editor and driving the port to 1'b1.

Reverse Serial Loopback Mode (Pre-CDR)

In the pre-CDR mode, data received through the RX input buffer is looped back to the TX output buffer. You can enable the reverse serial loopback mode by performing read-modify-write to the following registers.

Figure 215. Reverse Serial Loopback Mode (Pre-CDR)

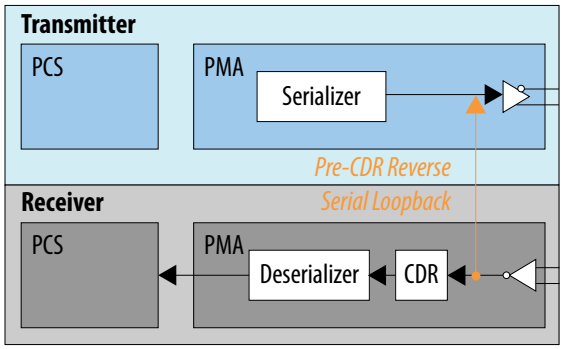


Table 188. Bit Values to Be Set

Address	Bit Values
0x137[7]	1'b1
0x13C[7]	1'b0
0x132[5:4]	2'b00
0x142[4]	1'b1
0x11D[0]	1'b1

Note: No specific order to access these registers.

Reverse Serial Loopback Mode (Post-CDR)

In the post-CDR mode, received data passes through the RX CDR and then loops back to the TX output buffer. Perform read-modify-write to the following registers to enable this mode.

Figure 216. Reverse Serial Loopback Mode (Post-CDR)

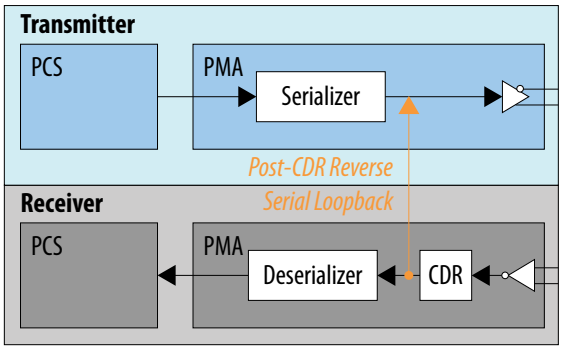


Table 189. Bit Values to Be Set

Address	Bit Values
0x137[7]	1'b0
0x13C[7]	1'b1
0x132[5:4]	2'b01
0x142[4]	1'b0
0x11D[0]	1'b0

Note: No specific order to access these registers.

Disabling Reverse Serial Loopback Mode (Pre-CDR and Post-CDR)

To disable reverse-serial loopback mode, set the address bits to the following values, by performing read-modify-write.

Table 190. Bit Values to Be Set

Address	Bit Values
0x137[7]	1'b0
0x13C[7]	1'b0
0x132[5:4]	2'b00
0x142[4]	1'b0
0x11D[0]	1'b0

Note: No specific order to access these registers.

Related Information

Steps to Perform Dynamic Reconfiguration on page 328

6.13. Ports and Parameters

The reconfiguration interface is integrated in the Native PHY instance and the TX PLL instances. Instantiate the Native PHY and the TX PLL IP cores in Qsys by clicking **Tools ➤ IP Catalog**. You can define parameters for IP cores by using the IP core-specific parameter editor. To expose the reconfiguration interface ports, select the **Enable dynamic reconfiguration** option when parameterizing the IP core.

You can share the reconfiguration interface among all the channels by turning on **Share reconfiguration interface** when parameterizing the IP core. When this option is enabled, the IP core presents a single reconfiguration interface for dynamic reconfiguration of all channels. Address bits [9:0] provide the register address in the reconfiguration space of the selected channel. The remaining address bits of the reconfiguration address specify the selected logical channel. For example, if there are four channels in the Native PHY IP instance, `reconfig_address[9:0]` specifies the address and `reconfig_address[11:10]` are binary encoded to specify the four channels. For example, 2'b01 in `reconfig_address[11:10]` specifies logical channel 1.

The following figure shows the signals available when the Native PHY IP core is configured for four channels and the **Share reconfiguration interface** option is enabled.

Figure 217. Signals Available with Shared Native PHY Reconfiguration Interface

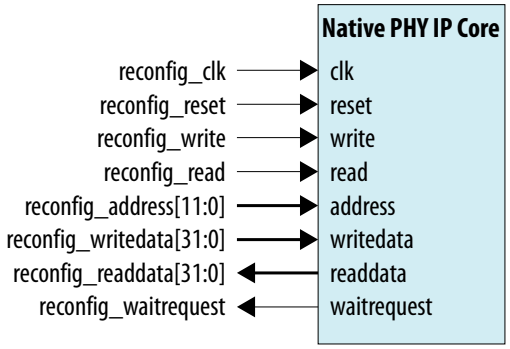


Table 191. Reconfiguration Interface Ports with Shared Native PHY Reconfiguration Interface

The reconfiguration interface ports when **Share reconfiguration interface** is enabled. <*N*> represents the number of channels.

Port Name	Direction	Clock Domain	Description
`reconfig_clk`	Input	N/A	Avalon clock. The clock frequency is 100-125 MHz.
`reconfig_reset`	Input	`reconfig_clk`	Resets the Avalon interface. Asynchronous to assertion and synchronous to deassertion.
`reconfig_write`	Input	`reconfig_clk`	Write enable signal. Signal is active high.
`reconfig_read`	Input	`reconfig_clk`	Read enable signal. Signal is active high.
`reconfig_address[log2<N>+9:0]`	Input	`reconfig_clk`	Address bus. The lower 10 bits specify address and the upper bits specify the channel.

continued...

Port Name	Direction	Clock Domain	Description
reconfig_writedata[31:0]	Input	reconfig_clk	A 32-bit data write bus. Data to be written into the address indicated by reconfig_address.
reconfig_readdata[31:0]	Output	reconfig_clk	A 32-bit data read bus. Valid data is placed on this bus after a read operation. Signal is valid after reconfig_waitrequest goes high and then low.
reconfig_waitrequest	Output	reconfig_clk	A one-bit signal that indicates the Avalon interface is busy. Keep the Avalon command asserted until the interface is ready to proceed with the read/write transfer. The behavior of this signal depends on whether the feature **Separate reconfig_waitrequest from the status of AVMM arbitration with PreSICE** is enabled or not. For more details, refer to the *Arbitration* section.

When **Share reconfiguration interface** is off, the Native PHY IP core provides an independent reconfiguration interface for each channel. For example, when a reconfiguration interface is not shared for a four-channel Native PHY IP instance, reconfig_address[9:0] corresponds to the reconfiguration address bus of logical channel 0, reconfig_address[19:10] correspond to the reconfiguration address bus of logical channel 1, reconfig_address[29:20] corresponds to the reconfiguration address bus of logical channel 2, and reconfig_address[39:30] correspond to the reconfiguration address bus of logical channel 3.

Table 192. Reconfiguration Interface Ports with Independent Native PHY Reconfiguration Interfaces

The reconfiguration interface ports when **Share reconfiguration interface** is disabled. <*N*> represents the number of channels.

Port Name	Direction	Clock Domain	Description
reconfig_clk[N-1:0]	Input	N/A	Avalon clock for each channel. The clock frequency is 100-125 MHz.
reconfig_reset[N-1:0]	Input	reconfig_clk	Resets the Avalon interface for each channel. Asynchronous to assertion and synchronous to deassertion.
reconfig_write[N-1:0]	Input	reconfig_clk	Write enable signal for each channel. Signal is active high.
reconfig_read[N-1:0]	Input	reconfig_clk	Read enable signal for each channel. Signal is active high.
reconfig_address[N*10-1:0]	Input	reconfig_clk	A 10-bit address bus for each channel.
reconfig_writedata[N*32-1:0]	Input	reconfig_clk	A 32-bit data write bus for each channel. Data to be written into the address indicated by the corresponding address field in reconfig_address.
reconfig_readdata[N*32-1:0]	Output	reconfig_clk	A 32-bit data read bus for each channel. Valid data is placed on this bus after a read operation. Signal is valid after waitrequest goes high and then low.
reconfig_waitrequest[N-1:0]	Output	reconfig_clk	A one-bit signal for each channel that indicates the Avalon interface is busy. Keep the Avalon command asserted until the interface is ready to proceed with the read/

continued...

Port Name	Direction	Clock Domain	Description
			write transfer. The behavior of this signal depends on whether the feature **Separate reconfig_waitrequest from the status of AVMM arbitration with PreSICE** is enabled or not. For more details, refer to the *Arbitration* section.

Table 193. Avalon Interface Parameters

The following parameters are available in the **Dynamic Reconfiguration** tab of the Transceiver Native PHY and TX PLL parameter editors.

Note: The Native PHY and the PLL IP Parameter Editors give an error or warning message if any of the parameter selections violate the legality checks.

Parameter	Value	Description
Enable dynamic reconfiguration	On / Off	Available in Native PHY and TX PLL IP parameter editors. Enables the reconfiguration interface. Off by default. The reconfiguration interface is exposed when this option is enabled.
Share reconfiguration interface	On / Off	Available in Native PHY IP parameter editor only. Enables you to use a single reconfiguration interface to control all channels. Off by default. If enabled, the uppermost bits of `reconfig_address` identifies the active channel. The lower 10 bits specify the reconfiguration address. Binary encoding is used to identify the active channel (available only for Transceiver Native PHY). Enable this option if the Native PHY is configured with more than one channel.
Enable Native PHY Debug Master Endpoint	On / Off	Available in Native PHY and TX PLL IP parameter editors. When enabled, the Native PHY Debug Master Endpoint (NPDME) is instantiated and has access to the Avalon-MM interface of the Native PHY. You can access certain test and debug functions using System Console with the NPDME. Refer to the *Embedded Debug Features* section for more details about NPDME.
Separate reconfig_waitrequest from the status of AVMM arbitration with PreSICE	On / Off	When enabled, `reconfig_waitrequest` does not indicate the status of AVMM arbitration with PreSICE. The AVMM arbitration status is reflected in a soft status register bit. This feature requires that the **Enable control and status registers** feature under **Optional Reconfiguration Logic** be enabled. Refer to *Arbitration* for more details on this feature. Refer to the *Calibration* chapter for more details about calibration.
Enable capability registers	On / Off	Available in Native PHY and TX PLL IP parameter editors. Enables capability registers. These registers provide high-level information about the transceiver channel's /PLL's configuration.
Set user-defined IP identifier	User-specified	Available in Native PHY and TX PLL IP parameter editors. Sets a user-defined numeric identifier that can be read from the `user_identifier` offset when the capability registers are enabled.
Enable control and status registers	On / Off	Available in Native PHY and TX PLL IP parameter editors. Enables soft registers for reading status signals and writing control signals on the PHY /PLL interface through the NPDME or reconfiguration interface.
Enable PRBS soft accumulators	On / Off	Available in Native PHY IP parameter editor only. Enables soft logic to perform PRBS bit and error accumulation when using the hard PRBS generator and checker.
Configuration file prefix	User-specified	Available in Native PHY and TX PLL IP parameter editors. Specifies the file prefix used for generating configuration files. Use a unique prefix for configuration files for each variant of the Native PHY and PLL.

continued...

Parameter	Value	Description
Generate SystemVerilog package file	On / Off	Available in Native PHY and TX PLL IP parameter editors. Creates a SystemVerilog package file that contains the current configuration data values for all reconfiguration addresses. Disabled by default.
Generate C header file	On / Off	Available in Native PHY and TX PLL IP parameter editors. Creates a C header file that contains the current configuration data values for all reconfiguration addresses. Disabled by default.
Generate MIF (Memory Initialize File)	On / Off	Available in Native PHY and TX PLL IP parameter editors. Creates a MIF file that contains the current configuration data values for all reconfiguration addresses. Disabled by default.
Include PMA analog settings in the configuration files	On / Off	Available in Native PHY IP parameter editor only. When enabled, the IP allows you to configure the analog settings for the PMA. These settings are included in your generated configuration files. *Note:* Even with this option enabled in the Native PHY IP Parameter Editor, you must still specify QSF assignments for your analog settings when compiling your static design. The analog settings selected in the Native PHY IP Parameter Editor are used only to include these settings and their dependent settings in the selected configuration files. For details about QSF assignments for the analog settings, refer to the *Analog Parameter Settings* chapter.
Enable multiple reconfiguration profiles	On / Off	Available in Native PHY and ATX PLL IP parameter editors only. Use the Parameter Editor to store multiple configurations. The parameter settings for each profile are tabulated in the Parameter Editor.
Enable embedded reconfiguration streamer	On / Off	Available in Native PHY and ATX PLL IP parameter editors only. Embeds the reconfiguration streamer into the Native PHY/ATX PLL IP cores and automates the dynamic reconfiguration process between multiple predefined configuration profiles.
Generate reduced reconfiguration files	On / Off	Available in Native PHY and ATX PLL IP parameter editors only. Enables the Native PHY and ATX PLL IP cores to generate reconfiguration files that contain only the attributes that differ between multiple profiles.
Number of reconfiguration profiles	1 to 8	Available in Native PHY and ATX PLL IP parameter editors only. Specifies the number of reconfiguration profiles to support when multiple reconfiguration profiles are enabled.
Selected reconfiguration profile	0 to 7	Available in Native PHY and ATX PLL IP parameter editors only. Selects which reconfiguration profile to store when you click **Store profile**.
Store configuration to selected profile	N/A	Available in Native PHY and ATX PLL IP parameter editors only. Stores the current Native PHY and ATX PLL parameter settings to the profile specified by the **Selected reconfiguration profile** parameter.
Load configuration from selected profile	N/A	Available in Native PHY and ATX PLL IP parameter editors only. Loads the current Native PHY/ATX PLL IP with parameter settings from the stored profile specified by the **Selected reconfiguration profile** parameter.
Clear selected profile	N/A	Available in Native PHY and ATX PLL IP parameter editors only. Clears the stored Native PHY/ATX PLL IP parameter settings for the profile specified by the **Selected reconfiguration profile** parameter. An empty profile defaults to the current parameter

continued...

Parameter	Value	Description
		settings of the Native PHY/ATX PLL. In other words, an empty profile reflects the Native PHY/ATX PLL current parameter settings.
Clear all profiles	N/A	Available in Native PHY and ATX PLL IP parameter editors only. Clears the Native PHY/ATX PLL IP parameter settings for all the profiles.
Refresh selected_profile	N/A	Available in Native PHY and ATX PLL IP parameter editors only. Equivalent to clicking the **Load configuration from selected profile** and **Store configuration to selected profile** buttons in sequence. This operation loads the parameter settings from stored profile specified by the **Selected reconfiguration profile** parameter and then stores the parameters back to the profile.

Table 194. Analog PMA Settings (Optional) for Dynamic Reconfiguration

The following parameters are available in the **Analog PMA Settings (Optional)** tab of the Transceiver Native PHY parameter editor. Refer to *Changing PMA Analog Parameters* for more details. Refer to the *Analog Parameter Settings* chapter for details about using the QSF assignments.

Parameter	Value	Description
TX Analog PMA Settings		
Analog Mode (Load Intel-recommended Default settings)	**cei_11100_lr** to **xfp_9950**	Selects the analog protocol mode to pre-select the TX pin swing settings (VOD, Pre-emphasis, and Slew Rate). After loading the pre-selected values in the Parameter Editor, if one or more of the individual TX pin swing settings need to be changed, then enable the option to override the Intel-recommended defaults to individually modify the settings. For details about QSF assignments for the analog settings, refer to the *Analog Parameter Settings* chapter.
Override Intel-recommended Analog Mode Default settings	On / Off	Enables the option to override the Intel-recommended settings for the selected TX Analog Mode for one or more TX analog parameters.
Output Swing Level (VOD)	0-31	Selects the transmitter programmable output differential voltage swing.
Pre-Emphasis First Pre-Tap Polarity	Fir_pre_1t_neg, Fir_pre_1t_pos	Selects the polarity of the first pre-tap for pre-emphasis.
Pre-Emphasis First Pre-Tap Magnitude	0-16	Selects the magnitude of the first pre-tap for pre-emphasis.
Pre-Emphasis Second Pre-Tap Polarity	Fir_pre_2t_neg, Fir_pre_2t_pos	Selects the polarity of the second pre-tap for pre-emphasis.
Pre-Emphasis Second Pre-Tap Magnitude	0-7	Selects the magnitude of the second pre-tap for pre-emphasis.
Pre-Emphasis First Post-Tap Polarity	Fir_post_1t_neg, Fir_post_1t_pos	Selects the polarity of the first post-tap for pre-emphasis.
Pre-Emphasis First Post-Tap Magnitude	0-25	Selects the magnitude of the first post-tap for pre-emphasis.
Pre-Emphasis Second Post-Tap Polarity	Fir_post_2t_neg, Fir_post_2t_pos	Selects the polarity of the second post-tap for pre-emphasis.
Pre-Emphasis Second Post-Tap Magnitude	0-12	Selects the magnitude of the second post-tap for pre-emphasis.
Slew Rate Control	slew_r0 to slew_r5	Selects the slew rate of the TX output signal. Valid values span from slowest to the fastest rate.

continued...

Parameter	Value	Description
High-Speed Compensation	Enable / Disable	Enables the power-distribution network (PDN) induced inter-symbol interference (ISI) compensation in the TX driver. When enabled, it reduces the PDN induced ISI jitter, but increases the power consumption.
On-Chip termination	r_r1, r_r2	Selects the on-chip TX differential termination.
RX Analog PMA settings		
Override Intel-recommended Default settings	On / Off	Enables the option to override the Intel-recommended settings for one or more RX analog parameters. For details about QSF assignments for the analog settings, refer to the *Analog Parameter Settings* chapter.
CTLE (Continuous Time Linear Equalizer) mode	non_s1_mode	Selects RX high gain mode (non_s1_mode) for the Continuous Time Linear Equalizer (CTLE).
DC gain control of high gain mode CTLE	no_dc_gain to stg4_gain7	Selects the DC gain of the Continuous Time Linear Equalizer (CTLE) in high gain mode
AC Gain Control of High Gain Mode CTLE	radp_ctle_acgain_4s_0 to radp_ctle_acgain_4s_28	Selects the AC gain of the Continuous Time Linear Equalizer (CTLE) in high gain mode when CTLE is in manual mode
Variable Gain Amplifier (VGA) Voltage Swing Select	radp_vga_sel_0 to radp_vga_sel_4	Selects the Variable Gain Amplifier (VGA) output voltage swing when the CTLE block is in manual mode.

Related Information

- Analog Parameter Settings on page 388
- Embedded Debug Features on page 353
- Arbitration on page 325
- Changing PMA Analog Parameters on page 338
- Calibration on page 373

6.14. Dynamic Reconfiguration Interface Merging Across Multiple IP Blocks

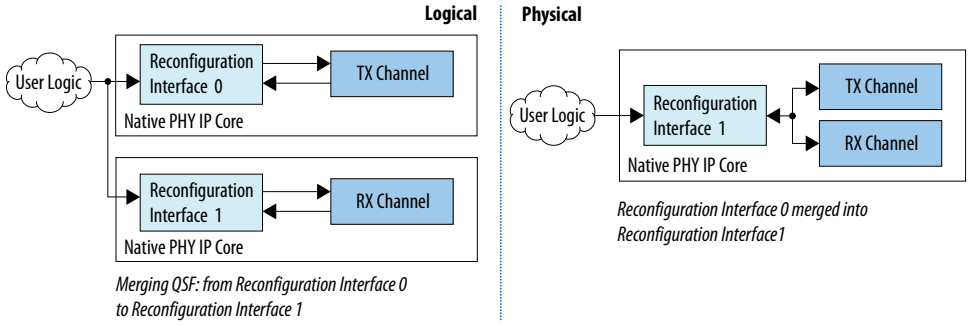
Dynamic reconfiguration interfaces may need to be shared between multiple IP blocks to maximize transceiver channel utilization. The Native PHY provides the ability to create channels that are either simplex or duplex instances. However, each physical transceiver channel in Cyclone 10 GX devices is fully duplex.

You can share the reconfiguration interfaces across different IP blocks by manually making a QSF assignment. There are two cases where a dynamic reconfiguration interface might need to be shared between multiple IP blocks:

- Independent instances of simplex receivers and transmitters in the same physical location
- Separate CMU PLL and TX channel in the same physical location

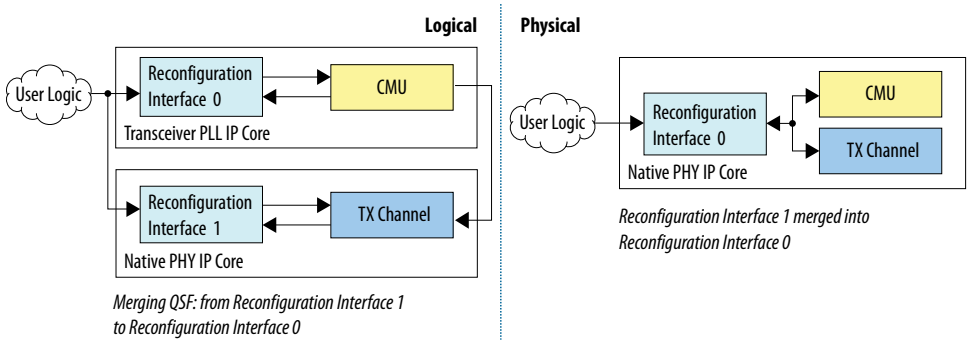
The following example shows one Native PHY IP instance of a TX-only channel and another instance of an RX-only channel.

Figure 218. Independent Instances of Simplex TX/RX in the Same Physical Location



The following example shows one Native PHY IP instance of a TX-only channel and an instance of a CMU PLL.

Figure 219. Separate CMU PLL and TX Channel in the Same Physical Location



Rules for Merging Reconfiguration Interfaces Across Multiple IP Cores

To merge reconfiguration interfaces across multiple IP blocks, you must follow these rules:

1. The control signals for the reconfiguration interfaces of the IP blocks must be driven by the same source. The `reconfig_clk`, `reconfig_reset`, `reconfig_write`, `reconfig_read`, `reconfig_address`, and `reconfig_writedata` ports of the two interfaces to be merged must be driven from the same source.
2. You must make a QSF assignment to manually specify which two reconfiguration interfaces are to be merged.
   a. Use the **XCVR_RECONFIG_GROUP** assignment.
   b. Set the **To** field of the assignment to either the reconfiguration interfaces of the instances to be merged or to the pin names. The reconfiguration interface has the string **twentynm_hssi_avmm_if_inst**.
   c. Assign the two instances to be merged to the same reconfiguration group.

You cannot merge multiple reconfiguration interfaces when NPDME, optional reconfiguration logic, or embedded reconfiguration streamer are enabled in the Native PHY IP core. (30)

You cannot merge the TX and RX channels when the **Shared reconfiguration interface** parameter is enabled in the Native PHY IP core Parameter Editor. You can merge channels only if the reconfiguration interfaces are independent.

Refer to the following two examples to merge reconfiguration interfaces.

Example 3. Using reconfiguration interface names

This example shows how to merge a transmit-only Native PHY instance with a receive-only instance using the reconfiguration interface names. These instances are assigned to reconfiguration group 0.

For Native PHY 0—transmit-only instance:

```
set_instance_assignment -name XCVR_RECONFIG_GROUP 0 -to
topdesign:topdesign_inst|<TX only instance name>*twentynm_hssi_avmm_if_inst*
```

For Native PHY 1—receive-only instance to be merged with Native PHY 0:

```
set_instance_assignment -name XCVR_RECONFIG_GROUP 0 -to
topdesign:topdesign_inst|<RX only instance name>*twentynm_hssi_avmm_if_inst*
```

Example 4. Using pin names

This example shows how to merge a transmit-only Native PHY instance with a receive-only instance using pin names. These instances are assigned to reconfiguration group 1.

For Native PHY 0—transmit-only instance:

```
set_instance_assignment -name XCVR_RECONFIG_GROUP 1 -to tx[0]
```

For Native PHY 1—receive-only instance to be merged with Native PHY 0:

```
set_instance_assignment -name XCVR_RECONFIG_GROUP 1 -to rx[0]
```

6.15. Embedded Debug Features

Note: For details on TTK usage refer to *Debugging Transceiver Links* in *Intel Quartus Prime Pro Edition User Guide: Debug Tools*.

The Cyclone 10 GX Transceiver Native PHY, ATX PLL, fPLL, and CMU PLL IP cores provide the following optional debug features to facilitate embedded test and debug capability:

- Native PHY Debug Master Endpoint (NPDME)
- Optional Reconfiguration Logic

Related Information

Intel Quartus Prime Pro Edition User Guide: Debug Tools

(30) The capability register is not available when merging a simplex Tx and a simplex Rx. Hence, the user cannot check the calibration status through capability register. Please refer to "Calibration" chapter on how to check the calibration status when merging a simplex Tx and simplex Rx.

6.15.1. Native PHY Debug Master Endpoint

The NPDME is a JTAG-based Avalon Memory-Mapped (Avalon-MM) master that provides access to the transceiver and PLL registers through the system console. You can enable NPDME using the **Enable Native PHY Debug Master Endpoint** option available under the **Dynamic Reconfiguration** tab in the Native PHY and PLL IP cores. When using NPDME, the Quartus Prime software inserts the debug interconnect fabric to connect with USB, JTAG, or other net hosts. Select the **Share Reconfiguration Interface** parameter when the Native PHY IP instance has more than one channel.

When you enable NPDME in your design, you must

- connect an Avalon-MM master to the reconfiguration interface.
- OR connect the, `reconfig_reset` signals and ground the `reconfig_write`, `reconfig_read`, `reconfig_address` and `reconfig_write` data signals of the reconfiguration interface. If the reconfiguration interface signals are not connected appropriately, there is no clock or reset for the NPDME, and NPDME does not function as expected.

6.15.2. Optional Reconfiguration Logic

The Cyclone 10 GX Transceiver Native PHY, ATX PLL, fPLL, and CMU PLL IP cores contain soft logic for debug purposes known as the Optional Reconfiguration Logic. This soft logic provides a set of registers that enable you to determine the state of the Native PHY and PLL IP cores.

You can enable the following optional reconfiguration logic options in the transceiver Native PHY and PLL IP cores:

- Capability registers
- Control and status registers
- PRBS soft accumulators (Native PHY IP core only)

6.15.2.1. Capability Registers

The capability registers provide high level information about the transceiver channel and PLL configuration.

The capability registers capture a set of chosen capabilities of the PHY that cannot be reconfigured. The following capability registers are available for the Native PHY IP core.

Table 195. Capability Registers for the Native PHY IP Core

Address	Type	Name	Description
0x200[7:0]	RO	IP Identifier	Unique identifier for the Native PHY IP instance.
0x204[0]	RO	Status Register Enabled	Indicates whether the status registers have been enabled. 1'b1 indicates that the status registers are enabled.
0x205[0]	RO	Control Register Enabled	Indicates whether the control registers have been enabled. 1'b1 indicates that the control registers are enabled.

continued...

Address	Type	Name	Description
0x210[7:0]	RO	Number of Channels	Shows the number of channels specified for the Native PHY IP instance.
0x211[7:0]	RO	Channel Number	Shows the unique channel number.
0x212[7:0]	RO	Duplex	Shows the transceiver mode: • 2'b00 = Unused • 2'b01 = TX • 2'b10 = RX • 2'b11 = Duplex
0x213[0]	RO	PRBS Soft Enabled	Indicates whether the PRBS soft accumulators are enabled. 1'b1 indicates the accumulators are enabled.

The following capability registers are available for the PLL IP cores.

Table 196. Capability Registers for the PLL IP Cores

Address	Type	Name	Description
0x200[7:0]	RO	IP Identifier	Unique identifier for the PLL IP instance.
0x204[0]	RO	Status Register Enabled	Indicates if the status registers have been enabled or not. 1'b1 indicates that the status registers have been enabled.
0x205[0]	RO	Control Register Enabled	Indicates if the control registers have been enabled or not. 1'b1 indicates that the control registers have been enabled.
0x210[7:0]	RO	Master CGB Enabled	Indicates if the Master Clock Generation Block has been enabled. 1'b1 indicates the master CGB is enabled.

6.15.2.2. Control and Status Registers

Control and status registers are optional registers that memory-map some of the status outputs from and control inputs to the Native PHY and PLL.

The following control and status registers are available for the Native PHY IP core.

Table 197. Control Registers for the Native PHY IP Core

Address	Type	Register	Description
0x2E0[0]	RW	`set_rx_locktodata`	Asserts the `set_rx_locktodata` signal to the receiver. 1'b1 sets the NPDME `set_rx_locktodata` register. See `override_set_rx_locktodata`.
0x2E0[1]	RW	`set_rx_locktoref`	Asserts the `set_rx_locktoref` signal to the receiver. 1'b1 sets the NPDME `set_rx_locktoref` register. See `override_set_rx_locktoref` row below.
0x2E0[2]	RW	`override_set_rx_locktodata`	Selects whether the receiver listens to the NPDME `set_rx_locktodata` register or the `rx_set_locktodata` port. 1'b1 indicates that the receiver listens to the NPDME `set_rx_locktodata` register.

continued...

Address	Type	Register	Description
0x2E0[3]	RW	override_set_rx_locktoref	Selects whether the receiver is listens to the NPDME set_rx_locktoref register or the rx_set_locktoref port. 1'b1 indicates that the receiver listens to the NPDME set_rx_locktoref register.
0x2E1[0]	RW	rx_seriallpbken	Enables the rx_seriallopbken feature in the transceiver. 1'b1 enables reverse serial loopback.
0x2E2[0]	RW	rx_analogreset	Drives rx_analogreset when the override is set.
0x2E2[1]	RW	rx_digitalreset	Drives rx_digitalreset when the override is set.
0x2E2[2]	RW	tx_analogreset	Drives tx_analogreset when the override is set.
0x2E2[3]	RW	tx_digitalreset	Drives tx_digitalreset when the override is set.
0x2E2[4]	RW	override_rx_analogreset	Selects whether the receiver listens to the NPDME rx_analogreset register or the rx_analogreset port. 1'b1 indicates the receiver listens to the NPDME rx_analogreset register.
0x2E2[5]	RW	override_rx_digitalreset	Selects whether the receiver listens to the NPDME rx_digitalreset register or the rx_digitalreset port. 1'b1 indicates the receiver listens to the NPDME rx_digitalreset register.
0x2E2[6]	RW	override_tx_analogreset	Selects whether the receiver listens to the NPDME tx_analogreset register or the tx_analogreset port. 1'b1 indicates the receiver listens to the NPDME tx_analogreset register.
0x2E2[7]	RW	override_tx_digitalreset	Selects whether the receiver listens to the NPDME tx_digitalreset register or the tx_digitalreset port. 1'b1 indicates the receiver listens to the NPDME tx_digitalreset register.

Table 198. Status Registers for the Native PHY IP Core

Address	Type	Register	Description
0x280[0]	RO	rx_is_lockedtodata	Shows the status of the current channel's rx_is_lockedtodata signal. 1'b1 indicates the receiver is locked to the incoming data.
0x280[1]	RO	rx_is_lockedtoref	Shows the status of the current channel's rx_is_lockedtoref signal. 1'b1 indicates the receiver is locked to the reference clock.
0x281[0]	RO	tx_cal_busy	Shows the status of the transmitter calibration status. 1'b1 indicates the transmitter calibration is in progress.
0x281[1]	RO	rx_cal_busy	Shows the status of the receiver calibration status. 1'b1 indicates the receiver calibration is in progress.
0x281[2]	RO	avmm_busy	Shows the status of the internal configuration bus arbitration. 1'b1 indicates PreSICE has control of the internal configuration bus. 1'b0 indicates the user has control of the internal configuration bus. Refer to the *Arbitration* section for more details. For more details about calibration registers and performing user recalibration, refer to the *Calibration* chapter.

The following control and status registers are available for the PLL IP cores.

Table 199. Control Registers for the PLL IP Cores

Address	Type	Register	Description
0x2E0[0]	RW	pll_powerdown	Drives the PLL powerdown when the Override is set.
0x2E0[1]	RW	override_pll_powerdown	Selects whether the receiver listens to the NPDME pll_powerdown register or the pll_powerdown port. 1'b1 indicates the receiver islistens to the NPDME pll_powerdown.

Table 200. Status Registers for the PLL IP Cores

Address	Type	Register	Description
0x280[0]	RO	pll_locked	Indicates if the PLL is locked. 1'b1 indicates the PLL is locked.
0x280[1]	RO	pll_cal_busy	Indicates the calibration status. 1'b1 indicates the PLL is currently being calibrated.
0x280[2]	RO	avmm_busy	Shows the status of the internal configuration bus arbitration. 1'b1 indicates PreSICE has control of the internal configuration bus. 1'b0 indicates the user has control of the internal configuration bus. Refer to the *Arbitration* section for more details.

Related Information

- Calibration on page 373
- Arbitration on page 325

6.15.2.3. PRBS Soft Accumulators

The Pseudo Random Binary Sequence (PRBS) soft accumulators are used in conjunction with the hard PRBS blocks in the transceiver channel. This section describes the soft logic that can be added to the Native PHY IP core. To enable this option, turn on the **Enable PRBS Soft Accumulators** option in the Native PHY IP Parameter Editor.

The PRBS soft accumulator has three control bits (Enable, Reset, and Snapshot) and one status bit (PRBS Done).

- **Enable** bit—used to turn on the accumulation logic. This bit is also used for selective error accumulation and to pause the sequence.
- **Reset** bit—resets the PRBS polynomial and the bit and error accumulators. It also resets the snapshot registers if independent channel snapshots are used.
- **Snapshot** bit—captures the current value of the accumulated bits and the errors simultaneously. This neutralizes the impact of the added read time when the Avalon-MM interface is used. Capturing a snapshot provides an accurate error count with respect to the bit count at a specific time.
- **PRBS Done** bit—indicates the PRBS checker has had sufficient time to lock to the incoming pattern.

For example, to capture the accumulated errors at any instance of time and read them back, you can perform the following operations.

1. Perform the necessary steps from steps 1 to 7 in *Steps to Perform Dynamic Reconfiguration*.
2. Perform read-modify-write to address 0x300 and set bit 0 to 1'b1. This action enables the error and bit counters.
3. To capture the errors accumulated at a particular instant, perform read-modify-write to address 0x300 and set bit 2 to 1'b1. This takes a snapshot of the error counters and stores the value to the error count registers.
4. To read the number of errors accumulated when the snapshot was captured, perform a read from the corresponding error registers 0x301 to 0x307.
5. To reset the bit and error accumulators, perform a read-modify-write to address 0x300 bit 1.
6. Perform the necessary steps from steps 9 to 12 in *Steps to Perform Dynamic Reconfiguration*.

Note: You can enable the error and bit counters (0x300[0]) and capture the accumulated bits and errors at different times. The error count registers and bit count registers are updated with the latest counter values as long as the counter enable bit is set.

Use the PRBS soft accumulators to count the number of accumulated bits and errors when the hard PRBS blocks are used. PRBS soft accumulators are word-based counter. The value read out from the PRBS soft accumulators represent the number of words counted. Hence, in order to obtain the total accumulated bit, user needs to multiply the value read out from the Accumulated bit pass through count [49:0] registers with the width of PCS-PMA interface. For Accumulated error count [49:0] registers, it counts one as long as there are bit errors in a word (be it one bit error in a word or all the bits in a word are erroneous). Hence, the Accumulated error count [49:0] registers do not give absolute bit errors counted. For each count, the absolute bit errors could range from one to the width of PCS-PMA interface.

For more information about using the hard PRBS blocks, refer to the "Using Data Pattern Generators and Checkers" section.

Table 201. PRBS Accumulator Registers

Address	Type	Name	Description
0x300[0]	RW	Counter enable (enables both error and bit counters)	Counter enable (enables both error and bit counters)
0x300[1]	RW	Reset	Reset the error accumulators
0x300[2]	RW	Error Count Snapshot	Snapshot captures the current value of accumulated bits and the errors at that time instance
0x300[3]	RO	PRBS Done	PRBS Done when asserted indicates the verifier has captured consecutive PRBS patterns and first pass of polynomial is complete
0x301[7:0]	RO	Accumulated error count [7:0]	Accumulated error count [7:0]
0x302[7:0]	RO	Accumulated error count [15:8]	Accumulated error count [15:8]
0x303[7:0]	RO	Accumulated error count [23:16]	Accumulated error count [23:16]
0x304[7:0]	RO	Accumulated error count [31:24]	Accumulated error count [31:24]

continued...

Address	Type	Name	Description
0x305[7:0]	RO	Accumulated error count [39:32]	Accumulated error count [39:32]
0x306[7:0]	RO	Accumulated error count [47:40]	Accumulated error count [47:40]
0x307[1:0]	RO	Accumulated error count [49:48]	Accumulated error count [49:48]
0x30D[7:0]	RO	Accumulated bit pass through count[7:0]	Accumulated bit pass through count[7:0]
0x30E[7:0]	RO	Accumulated bit pass through count[15:8]	Accumulated bit pass through count[15:8]
0x30F[7:0]	RO	Accumulated bit pass through count[23:16]	Accumulated bit pass through count[23:16]
0x310[7:0]	RO	Accumulated bit pass through count[31:24]	Accumulated bit pass through count[31:24]
0x311[7:0]	RO	Accumulated bit pass through count[39:32]	Accumulated bit pass through count[39:32]
0x312[7:0]	RO	Accumulated bit pass through count[47:40]	Accumulated bit pass through count[47:40]
0x313[1:0]	RO	Accumulated bit pass through count[49:48]	Accumulated bit pass through count[49:48]

Note: Intel recommends that you disable the byte serializer and deserializer blocks when using the soft PRBS accumulators. When the byte serializer and deserializer blocks are enabled, the number of bits counted are halved because the clock is running at half the rate.

Related Information

- Steps to Perform Dynamic Reconfiguration on page 328
- Arbitration on page 325
- Using Data Pattern Generators and Checkers on page 359

6.16. Using Data Pattern Generators and Checkers

The Cyclone 10 GX transceivers contain hardened data generators and checkers to provide a simple and easy way to verify and characterize high speed links.

Hardening the data generators and verifiers saves FPGA fabric logic resources. The pattern generator block supports the following patterns:

- Pseudo Random Binary Sequence (PRBS)
- Pseudo Random Pattern (PRP)

The pattern generators and checkers are supported only for non-bonded channels.

6.16.1. Using PRBS Data Pattern Generator and Checker

Use the Cyclone 10 GX PRBS generator and checker to simulate traffic and easily characterize high-speed links without fully implementing any upper protocol stack layer. The PRBS generator generates a self-aligning pattern and covers a known number of unique sequences. Because the PRBS pattern is generated by a Linear Feedback Shift Register (LFSR), the next pattern can be determined from the previous

pattern. When the PRBS checker receives a portion of the received pattern, it can generate the next sequence of bits to verify whether the next data sequence received is correct.

The PRBS generator and checker are shared between the Standard and Enhanced datapaths through the PCS. Therefore, they have only one set of control signals and registers. The data lines from the various PCSs and shared PRBS generator are MUXed before they are sent to the PMA. When the PRBS generator is enabled, the data on the PRBS data lines is selected to be sent to the PMA. Either the data from the PCS or the data generated from the PRBS generator can be sent to the PMA at any time.

The PRBS generator and checker can be configured for two widths of the PCS-PMA interface: 10 bits and 64 bits. PRBS9 is available in both 10-bit and 64-bit PCS-PMA widths. All other PRBS patterns are available in 64-bit PCS-PMA width only. The PRBS generator and checker patterns can only be used when the PCS-PMA interface width is configured to 10 bits or 64 bits. For any other PCS-PMA width, to ensure the correct clocks are provided to the PRBS blocks you must first reconfigure the width to either 10 or 64 bits before using the PRBS generator and checker. For example, when the transceiver is configured to a 20-bit PCS/PMA interface, you must first reconfigure the PCS-PMA width to 10 bits before setting up the PRBS generator and checker. The PRBS setup does not automatically change the PCS/PMA width.

The 10-bit PCS-PMA width for PRBS9 is available for lower frequency testing. You can configure PRBS9 in either 10-bit or 64-bit width, based on the data rate. The FPGA fabric-PCS interface must run in the recommended speed range of the FPGA core. Therefore, you must configure PRBS9 in one of the two bit width modes, so that the FPGA fabric-PCS interface parallel clock runs in this operating range.

Examples:

- If you want to use PRBS9 and the data rate is 2.5 Gbps, you can use the PRBS9 in 10-bit mode (PCS-PMA width = 10). In this case, the parallel clock frequency = Data rate / PCS-PMA width = 2500 Mbps/10 = 250 MHz.
- If you want to use PRBS9 and the data rate is 6.4 Gbps, you can use the PRBS9 in 64-bit mode (PCS-PMA width = 64). In this case, the parallel clock frequency = Data rate / PCS-PMA width = 6400 Mbps/64 = 100 MHz.
- If you want to use PRBS9 and the data rate is 12.5 Gbps, you can use the PRBS9 in 64 bit mode (PCS-PMA width = 64). In this case, the parallel clock frequency = Data rate / PCS-PMA width = 12500 Mbps/64 = 195.3125 MHz.

Table 202. PRBS Supported Polynomials and Data Widths

Use the 10-bit mode of PRBS9 when the data rate is lower than 3 Gbps.

Pattern	Polynomial	64-Bit	10-Bit
PRBS7	$G(x) = 1+ x^6 + x^7$	X	
PRBS9	$G(x) = 1+ x^5 + x^9$	X	X
PRBS15	$G(x) = 1+ x^{14} + x^{15}$	X	
PRBS23	$G(x) = 1+ x^{18} + x^{23}$	X	
PRBS31	$G(x) = 1+ x^{28} + x^{31}$	X	

The PRBS checker has the following control and status signals available to the FPGA fabric:

- `rx_prbs_done`—Indicates the PRBS sequence has completed one full cycle. It stays high until you reset it with `rx_prbs_err_clr`.
- `rx_prbs_err`—Goes high if an error occurs. This signal is pulse-extended to allow you to capture it in the RX FPGA CLK domain.
- `rx_prbs_err_clr`—Used to reset the `rx_prbs_err signal`.

Enable the PRBS checker control and status ports through the Native PHY IP Parameter Editor in the Quartus Prime software.

Use the PRBS soft accumulators to count the number of accumulated bits and errors when the hard PRBS blocks are used. For more information about using the accumulators and reading the error values, refer to the *PRBS Soft Accumulators* section.

Table 203. Register Map for PRBS Generators for bonded and non bonded designs

Description	Reconfiguration Address	Reconfiguration Bits	Value	Attribute Encoding	Attribute Name
Select PRBS generator block	0x006	[2:0]	3'b100	prbs_pat	tx_pma_data_sel
	0x008	[6:5]	2'b00		
Enable PRBS 9 in 10-bit mode	0x006	[3]	1'b1	prbs9_10b	prbs9_dwidth
Enable PRBS 9 in 64-bit mode			1'b0	prbs9_64b	
Enable PRBS generator clock	0x006	[6]	1'b1	prbs_clk_en	prbs_clken
Disable PRBS generator clock			1'b0	prbs_clk_dis	
Enable PRBS 7 pattern	0x007	[7:4]	4'b0001	prbs_7	prbs_gen_pat
	0x008	[4]	1'b0		
Enable PRBS 9 pattern	0x007	[7:4]	4'b0010	prbs_9	
	0x008	[4]	1'b0		
Enable PRBS 15 pattern	0x007	[7:4]	4'b0100	prbs_15	
	0x008	[4]	1'b0		
Enable PRBS 23 pattern	0x007	[7:4]	4'b1000	prbs_23	
	0x008	[4]	1'b0		
enable PRBS 31 pattern	0x007	[7:4]	4'b0000	prbs_31	
	0x008	[4]	1'b1		
Serializer 64-bit width mode	0x110	[2:0]	3'b011	sixty_four_bit	ser_mode
Serializer 10-bit width mode			3'b100	ten_bit	
Enable xN non bonding	0x111	[4:0]	5'b11000	xN_non_bonding(31)	x1_clock_source_sel
	0x119	[0]	1'b0		

Table 204. Register Map for PRBS Checker for bonded and non bonded designs

Description	Reconfiguration Address	Reconfiguration Bits	Value	Attribute Encoding	Attribute Name
Enable PRBS checker clock	0x00A	[7]	1'b1	prbs_clk_en	prbs_clken
Disable PRBS checker clock			1'b0	prbs_clk_dis	
Mask out the initial errors (from error counter threshold to 1023) seen by PRBS checker	0x00B	[3:2]	2'b11	prbsmask1024	rx_prbs_mask
Mask out the initial errors (from error counter threshold to 127) seen by PRBS checker			2'b00	prbsmask128	
Mask out the initial errors (from error counter threshold to 255) seen by PRBS checker			2'b01	prbsmask256	
Mask out the initial errors (from error counter threshold to 511) seen by PRBS checker			2'b10	prbsmask512	
Enable PRBS 7 pattern	0x00B	[7:4]	4'b0001	prbs_7	prbs_ver
	0x00C	[0]	1'b0		
Enable PRBS 9 pattern	0x00B	[7:4]	4'b0010	prbs_9	
	0x00C	[0]	1'b0		
Enable PRBS 15 pattern	0x00B	[7:4]	4'b0100	prbs_15	
	0x00C	[0]	1'b0		
Enable PRBS 23 pattern	0x00B	[7:4]	4'b1000	prbs_23	
	0x00C	[0]	1'b0		
Enable PRBS 31 pattern	0x00B	[7:4]	4'b0000	prbs_31	
	0x00C	[0]	1'b1		
PRBS 9 10-bit	0x00C	[3]	1'b1	prbs9_10b	prbs9_dwidth
PRBS 9 64-bit			1'b0	prbs9_64b	
Deserialization factor 10-bit mode	0x13F	[3:0]	4'b0001	10-bit mode	deser_factor
Deserialization factor 64-bit mode			4'b1110	64-bit mode	

(31) The pattern generators and checkers are supported only for non-bonded channels. If original design is bonded and you would like to use the PRBS generator/checker, you must read and save the value in registers 0x119[0], 0x111[4:0] before changing the `x1_clock_source_sel` settings to `xN_non_bonding` (by writing 5'b11000 to 0x119[0], 0x111[4:0]). This changes the design from bonded to non-bonded.

To disable the PRBS generator and restore the design back to original bonded design, you need to restore the original value that was previously saved from registers 0x119[0], 0x111[4:0].

Related Information

PRBS Soft Accumulators on page 357

6.16.1.1. Enabling the PRBS Data Generator in Non Bonded designs

You must perform a sequence of read-modify-writes to addresses 0x006, 0x007, 0x008, and 0x110 to enable either the PRBS data generator. To enable either the PRBS data generator, follow these steps:

1. Perform the necessary steps from steps 1 to 7 in *Steps to Perform Dynamic Reconfiguration*.
2. Perform a read-modify-write to address 0x006 according to Register Map for PRBS Generators for bonded and non bonded designs.
3. Perform a read-modify-write to address 0x007 according to Register Map for PRBS Generators for bonded and non bonded designs.
4. Perform a read-modify-write to address 0x008 according to Register Map for PRBS Generators for bonded and non bonded designs.
5. Perform a read-modify-write to address 0x110 with the specified width. This data width is either 64-bit or 10-bit.
6. Perform the necessary steps from steps 9 to 12 in *Steps to Perform Dynamic Reconfiguration*.
   To disable the PRBS generator, write the original values back into the read-modify-write addresses in Register Map for PRBS Generators for bonded and non bonded designs.

Related Information

- Steps to Perform Dynamic Reconfiguration on page 328
- Arbitration on page 325

6.16.1.1.1. Examples of Enabling the PRBS9 and PRBS31 Pattern Generators in Non Bonded designs

Example 5. Enabling the PRBS9 pattern generator in 10-bit mode

1. Perform the necessary steps from steps 1 to 7 in *Steps to Perform Dynamic Reconfiguration*.
2. Perform a read-modify-write to address x006[7:0] with the following bits: 8'b01- - 1100
3. Perform a read-modify-write to address x007[7:0] with the following bits: 8'b0010 - - - -
4. Perform a read-modify-write to address x008[7:0] with the following bits: 8'b - 000 - - - -
5. Perform a read-modify-write to address x110[7:0] with the following bits: 8'b - - - - - - 100
6. Perform the necessary steps from steps 9 to 12 in *Steps to Perform Dynamic Reconfiguration*.

Note: A dash (-) indicates that the corresponding bit value should not be modified during read-modify-write.

Example 6. Enabling the PRBS31 pattern generator in 64-bit mode

1. Perform the necessary steps from steps 1 to 7 in *Steps to Perform Dynamic Reconfiguration*.
2. Perform a read-modify-write to address x006[7:0] with the following bits: 8'b01- - 0100
3. Perform a read-modify-write to address x007[7:0] with the following bits: 8'b0000 - - - -
4. Perform a read-modify-write to address x008[7:0] with the following bits: 8'b - 001 - - - -
5. Perform a read-modify-write to address x110[7:0] with the following bits: 8'b - - - - - - 011
6. Perform the necessary steps from steps 9 to 12 in *Steps to Perform Dynamic Reconfiguration*.

Note: A dash (-) indicates that the corresponding bit value should not be modified during read-modify-write.

Related Information

Steps to Perform Dynamic Reconfiguration on page 328

6.16.1.2. Enabling the PRBS Data Generator in Bonded Designs

You must perform a sequence of read-modify-writes to addresses 0x006, 0x007, 0x008, and 0x110, 0x111and to enable either the PRBS data generator in bonded designs. To enable either the PRBS data generator, follow these steps:

1. Perform the necessary steps from steps 1 to 7 in *Steps to Perform Dynamic Reconfiguration*.
2. Perform a read-modify-write to address 0x006 according to Register Map for PRBS Generators for bonded and non bonded designs.
3. Perform a read-modify-write to address 0x007 according to Register Map for PRBS Generators for bonded and non bonded designs.
4. Perform a read-modify-write to address 0x008 according to Register Map for PRBS Generators for bonded and non bonded designs.
5. Perform a read-modify-write to address 0x110 with the specified width. This data width is either 64-bit or 10-bit.
6. Perform a read-modify-write to address 0x111 according to Register Map for PRBS Generators for bonded and non bonded designs. You must read and save the value in the register 0x111[5:0] before changing the `x1_clock_source_sel` setting to xN non bonding.
7. Perform the necessary steps from steps 9 to 12 in *Steps to Perform Dynamic Reconfiguration*.
   To disable the PRBS generator, write the original values back into the read-modify-write addresses in Register Map for PRBS Generators for bonded and non bonded designs.

6.16.1.2.1. Examples of Enabling the PRBS9 and PRBS31 Pattern Generators in Bonded Designs

Example 7. Enabling the PRBS9 pattern generator in 10-bit mode

1. Perform the necessary steps from steps 1 to 7 in *Steps to Perform Dynamic Reconfiguration*.
2. Perform a read-modify-write to address x006[7:0] with the following bits: 8'b01- - 1100
3. Perform a read-modify-write to address x007[7:0] with the following bits: 8'b0010 - - - -
4. Perform a read-modify-write to address x008[7:0] with the following bits: 8'b - 000 - - - -
5. Perform a read-modify-write to address x110[7:0] with the following bits: 8'b - - - - - 100
6. Perform a read-modify-write to address x111[5:0] with the following bits: 8'b---11000. You must read and save the value in the register 0x111[5:0] before changing the `x1_clock_source_sel` setting to xN non bonding.
7. Perform the necessary steps from steps 9 to 12 in *Steps to Perform Dynamic Reconfiguration*.

Note: A dash (-) indicates that the corresponding bit value should not be modified during read-modify-write.

Example 8. Enabling the PRBS31 pattern generator in 64-bit mode

1. Perform the necessary steps from steps 1 to 7 in *Steps to Perform Dynamic Reconfiguration*.
2. Perform a read-modify-write to address x006[7:0] with the following bits: 8'b01- - 0100
3. Perform a read-modify-write to address x007[7:0] with the following bits: 8'b0000 - - - -
4. Perform a read-modify-write to address x008[7:0] with the following bits: 8'b - 001 - - - -
5. Perform a read-modify-write to address x110[7:0] with the following bits: 8'b - - - - - 011
6. Perform a read-modify-write to address x111[5:0] with the following bits: 8'b---11000. You must read and save the value in the register 0x111[5:0] before changing the `x1_clock_source_sel` setting to xN non bonding.
7. Perform the necessary steps from steps 9 to 12 in *Steps to Perform Dynamic Reconfiguration*.

Note: A dash (-) indicates that the corresponding bit value should not be modified during read-modify-write.

6.16.1.3. Enabling the PRBS Data Checker in Non Bonded design

You must perform a sequence of read-modify-writes to the Transceiver Native PHY reconfiguration interface to enable the PRBS checker. You must perform read-modify-writes to addresses 0x00A, 0x00B, 0x00C, and 0x13F. To enable the PRBS checker, follow these steps:

1. Perform the necessary steps from steps 1 to 7 in *Steps to Perform Dynamic Reconfiguration*.
2. Perform a read-modify-write to address 0x00A with a value of 1'b1 to bit[7].
3. Perform a read-modify-write to address 0x00B according to Register Map for PRBS Checker for bonded and non bonded designs.
4. Perform a read-modify-write to address 0x00C according to Register Map for PRBS Checker for bonded and non bonded designs.
5. Perform a read-modify-write to address 0x13F according to Register Map for PRBS Checker for bonded and non bonded designs.
6. Perform the necessary steps from steps 9 to 12 in *Steps to Perform Dynamic Reconfiguration*.
   To disable the PRBS verifier write the original values back into the read-modify-write addresses listed above.

Related Information

Steps to Perform Dynamic Reconfiguration on page 328

6.16.1.3.1. Examples of Enabling the PRBS Data Checker

Example 9. Enabling the PRBS9 pattern checker in 10-bit mode

1. Perform the necessary steps from steps 1 to 7 in *Steps to Perform Dynamic Reconfiguration*.
2. Perform a read-modify-write to address x00A[7:0] with the following bits: 8'b1- - - - - - -
3. Perform a read-modify-write to address x00B[7:0] with the following bits: 8'b0010 00 - -
4. Perform a read-modify-write to address x00C[7:0] with the following bits: 8'b - - - - 1 - - 0
5. Perform a read-modify-write to address x13F[7:0] with the following bits: 8'b - - - - 0001
6. Perform the necessary steps from steps 9 to 12 in *Steps to Perform Dynamic Reconfiguration*.

Note: A dash (-) indicates that the corresponding bit value should not be modified

Example 10. Enabling the PRBS31 pattern checker in 64-bit mode

1. Perform the necessary steps from steps 1 to 7 in *Steps to Perform Dynamic Reconfiguration*.
2. Perform a read-modify-write to address x00A[7:0] with the following bits: 8'b1- - - - - - -
3. Perform a read-modify-write to address x00B[7:0] with the following bits: 8'b0000 11 - -

4. Perform a read-modify-write to address x00C[7:0] with the following bits: 8'b - - - - - - - 1
5. Perform a read-modify-write to address x13F[7:0] with the following bits: 8'b - - - - 1110
6. Perform the necessary steps from steps 9 to 12 in *Steps to Perform Dynamic Reconfiguration*.

Note: A dash (-) indicates that the corresponding bit value should not be modified

Use the PRBS soft accumulators to count the number of accumulated bits and errors when the hard PRBS blocks are used. For more details about using the accumulators and reading the error values, refer to the "PRBS Soft Accumulators" section.

Related Information

- PRBS Soft Accumulators on page 357
- Steps to Perform Dynamic Reconfiguration on page 328

6.16.1.4. Enabling the PRBS Checker in bonded designs

You must perform a sequence of read-modify-writes to addresses 0x00A, 0x00B, 0x00C, 0x13F, 0x111 and to enable the PRBS data checker in bonded designs. To enable the PRBS checker, follow these steps:

1. Perform the necessary steps from steps 1 to 7 in *Steps to Perform Dynamic Reconfiguration*.
2. Perform a read-modify-write to address 0x00A with a value of 1'b1 to bit[7].
3. Perform a read-modify-write to address 0x00B according to Register Map for PRBS Checker for bonded and non bonded designs.
4. Perform a read-modify-write to address 0x00C according to Register Map for PRBS Checker for bonded and non bonded designs.
5. Perform a read-modify-write to address 0x13F according to Register Map for PRBS Checker for bonded and non bonded designs.
6. Perform a read-modify-write to address 0x111 according to Register Map for PRBS Checker for bonded and non bonded designs.
7. Perform the necessary steps from steps 9 to 12 in *Steps to Perform Dynamic Reconfiguration*.
   To disable the PRBS verifier write the original values back into the read-modify-write addresses listed above.

6.16.1.5. Disabling/Enabling PRBS Pattern Inversion

The default PRBS pattern is inverted for both PRBS generator and checker. You can disable the pattern inversion for PRBS data leaving or entering the PRBS data pattern generator and checker, respectively. Table 205 on page 368 shows the addresses and bits to control the inversion of the PRBS generator or checker. To disable the PRBS pattern inversion for the PRBS generator or checker, follow these steps:

1. Perform the necessary steps from steps 1 to 7 in *Steps to Perform Dynamic Reconfiguration*.
2. To disable the inverted PRBS pattern leaving the PRBS generator, perform a read-modify-write to bit[2] with a value of 1'b1 to address 0x7.
3. To disable the inverted PRBS pattern entering the PRBS checker, perform a read-modify-write to bit[4] with a value of 1'b1 to address 0xA.
4. Perform the necessary steps from steps 9 to 12 in *Steps to Perform Dynamic Reconfiguration*.

Table 205. Register Map for PRBS Pattern Inversion

Reconfiguration Address (HEX)	Reconfiguration Bit	Attribute Name	Bit Encoding	Description
0x007	[2]	tx_static_polarity_inversion	1'b1	Disables PRBS inversion
			1'b0	Enables PRBS inversion (default)
0x00A	[4]	rx_static_polarity_inversion	1'b1	Disables static polarity inversion
			1'b0	Enables PRBS inversion (default)

Related Information

Steps to Perform Dynamic Reconfiguration on page 328

6.16.2. Using Pseudo Random Pattern Mode

You can use the Cyclone 10 GX Pseudo Random Pattern (PRP) generator and verifier in the scrambler and descrambler to generate random data pattern and seed that the scrambler can use. PRP mode is a test mode of the scrambler. Two seeds are available to seed the scrambler: all 0s or two local fault-ordered sets. The seed is used in the scrambler to produce the pattern. The `r_tx_data_pat_sel` is the data pattern that the scrambler scrambles. PRP is only available when the scrambler is enabled. The PRP verifier shares the `rx_prbs_err` error signal with PRBS. The error count can be read out from the corresponding registers.

6.16.2.1. Enabling Pseudo Random Pattern Mode

You must perform a sequence of read-modify-writes to the reconfiguration interface to enable the Pseudo Random Pattern. The read-modify-writes are required to addresses 0x082, 0x097, and 0x0AC. To enable the Pseudo Random Pattern, complete the following steps:

1. Perform the necessary steps from steps 1 to 7 in *Steps to Perform Dynamic Reconfiguration*.
2. Perform a read-modify-write to address 0x082.
3. Perform a read-modify-write to address 0x097.
4. Perform a read-modify-write to address 0x0AC.
5. Perform the necessary steps from steps 9 to 12 in *Steps to Perform Dynamic Reconfiguration*.

To disable the PRP verifier, write the original values back to the read-modify-write addresses listed above.

Related Information

Steps to Perform Dynamic Reconfiguration on page 328

6.17. Timing Closure Recommendations

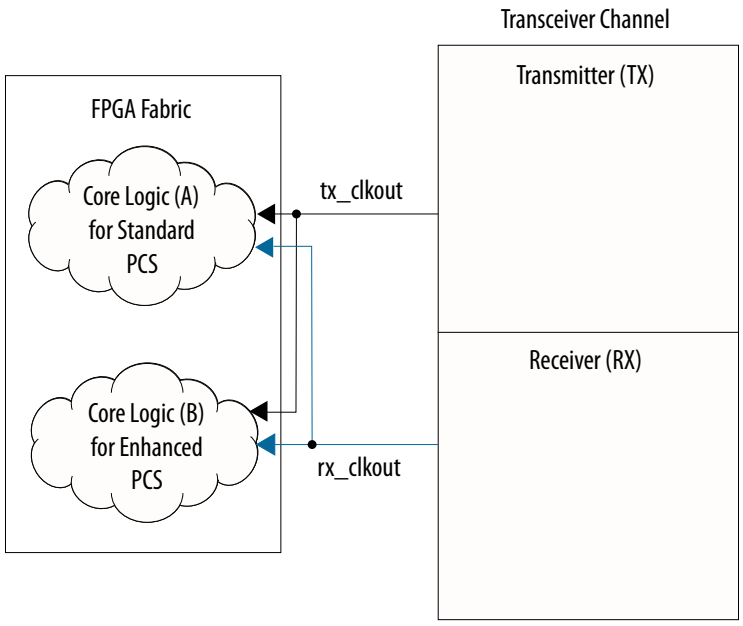
Intel recommends that you enable the multiple reconfiguration profiles feature in the Native PHY IP core if any of the modified or target configurations involve changes to PCS settings. Using multiple reconfiguration profiles is optional if the reconfiguration involves changes to only PMA settings such as PLL switching, CGB divider switching, and refclk switching. When you enable multiple reconfiguration profiles, the Quartus Prime TimeQuest Timing Analyzer includes the necessary PCS timing arcs for all profiles (initial profile and target profiles) during timing driven compilation. These timing arcs make the timing more accurate.

When performing a dynamic reconfiguration, you must:

- Include constraints to create the extra clocks for all modified or target configurations at the PCS-FPGA fabric interface. Clocks for the base configuration are created by the Quartus Prime software. These clocks enable the Quartus Prime software to perform static timing analysis for all the transceiver configurations and their corresponding FPGA fabric core logic blocks.
- Include the necessary false paths between the PCS – FPGA fabric interface and the core logic.

For example, you can perform dynamic reconfiguration to switch the datapath from Standard PCS to Enhanced PCS using the multiple reconfiguration profiles feature. In the following example, the base configuration uses the Standard PCS (data rate = 1.25 Gbps, PCS-PMA width = 10) and drives core logic A in the FPGA fabric. The target or modified configuration is configured to use the Enhanced PCS (data rate = 10.3125 Gbps, PCS-PMA width = 64) and drives core logic B in the FPGA fabric.

Figure 220. Using Multiple Reconfiguration Profiles



To enable the Quartus Prime software to close timing more accurately in this example, the following constraints must be created:

- `create_clock -name tx_clkout_enh -period 5.12 [get_pins {native_inst|xcvr_native_c10_0| g_xcvr_native_insts[0].twentynm_xcvr_native_inst| twentynm_xcvr_native_inst|inst_twentynm_pcs| gen_twentynm_hssi_tx_pld_pcs_interface.inst_twentynm_hssi_tx_pld_pcs_interface|pld_pcs_tx_clk_out}] -add`

  This constraint creates the `tx_clkout` clock that is used to clock the core logic B in the FPGA fabric.

- `create_clock -name rx_clkout_enh –period 5.12 [get_pins {native_inst|xcvr_native_c10_0| g_xcvr_native_insts[0].twentynm_xcvr_native_inst| twentynm_xcvr_native_inst|inst_twentynm_pcs| gen_twentynm_hssi_rx_pld_pcs_interface.inst_twentynm_hssi_rx_pld_pcs_interface|pld_pcs_rx_clk_out}] -add`

  This constraint creates the `rx_clkout` clock that is used to clock the core logic B in the FPGA fabric.

- `set_false_path -from [get_clocks {tx_clkout_enh}] -to [get_registers <Core Logic A>]`

  Based on how the clocks are connected in the design, you might have to include additional constraints to set false paths from the registers in the core logic to the clocks.

- `set_false_path -from [get_clocks {rx_clkout_enh}] -to [get_registers <Core Logic A>]`

  Based on how the clocks are connected in the design, you may have to include additional constraints to set false paths from the registers in the core logic to the clocks.

- `set_false_path -from [get_clocks {tx_clkout}] -to [get_registers <Core Logic B>]`

  Based on how the clocks are connected in the design, you may have to include additional constraints to set false paths from the registers in the core logic to the clocks.

- `set_false_path -from [get_clocks {rx_clkout}] -to [get_registers <Core Logic B>]`

  Based on how the clocks are connected in the design, you may have to include additional constraints to set false paths from the registers in the core logic to the clocks.

  Note: If any of the profile or configuration switch involves switching from FIFO to the register mode, then the false paths should be set between the PCS-PMA interface register and the core logic because the common clock point is within the PCS-PMA interface.

For example, if the base configuration of the above case is configured for the TX and RX FIFOs in the Register Mode, the following constraint needs to be created:

- ```
  set_false_path -from [get_registers {native:native_inst|
  native_altera_xcvr_native_c10_150_lzjn6xi:xcvr_native_c10_0|
  twentynm_xcvr_native:g_xcvr_native_insts[0].twentynm_xcvr_nati
  ve_inst|
  twentynm_xcvr_native_rev_20nm5es:twentynm_xcvr_native_inst|
  twentynm_pcs_rev_20nm5es:inst_twentynm_pcs|
  gen_twentynm_hssi_tx_pld_pcs_interface.inst_twentynm_hssi_tx_p
  ld_pcs_interface~pma_tx_pma_clk_reg.reg}] -to [get_registers
  <Core Logic B>]
  ```
- ```
  set_false_path -from [get_registers {native:native_inst|
  native_altera_xcvr_native_c10_150_lzjn6xi:xcvr_native_c10_0|
  twentynm_xcvr_native:g_xcvr_native_insts[0].twentynm_xcvr_nati
  ve_inst|
  twentynm_xcvr_native_rev_20nm5es:twentynm_xcvr_native_inst|
  twentynm_pcs_rev_20nm5es:inst_twentynm_pcs|
  gen_twentynm_hssi_rx_pld_pcs_interface.inst_twentynm_hssi_rx_p
  ld_pcs_interface~pma_rx_pma_clk_reg.reg}] -to [get_registers
  <Core Logic B>]
  ```

6.18. Unsupported Features

The following features are not supported by either the Transceiver Native PHY IP core or the PLL IP reconfiguration interface:

- Reconfiguration from a bonded configuration to a non-bonded configuration, or vice versa
- Reconfiguration from a bonded protocol to another bonded protocol
- Reconfiguration from PCIe (with Hard IP) to PCIe (without Hard IP) or non-PCIe bonded protocol switching
- Switching between bonding schemes, such as xN to feedback compensation
- Master CGB reconfiguration
- Switching between two master CGBs
- Serialization factor changes on bonded channels
- TX PLL switching on bonded channels

Note: Transceiver Native PHY IP non-bonded configuration to another Transceiver Native PHY IP non-bonded configuration is supported.

6.19. Cyclone 10 GX Transceiver Register Map

The transceiver register map provides a list of available PCS, PMA, and PLL addresses that are used in the reconfiguration process.

Use the register map in conjunction with a transceiver configuration file generated by the Cyclone 10 GX Native PHY IP core. This configuration file includes details about the registers that are set for a specific transceiver configuration. Do not use the register map to locate and modify specific registers in the transceiver. Doing so may result in an illegal configuration. Refer to a valid transceiver configuration file for legal register values and combinations.

The register map is provided as an Excel spreadsheet for easy search and filtering.

Related Information

Cyclone 10 GX Transceiver Register Map

6.20. Reconfiguration Interface and Dynamic Reconfiguration Revision History

Document Version	Changes
2019.05.13	Made the following change: • Renamed Altera Debug Master Endpoint (ADME) to Native PHY DebugMaster Endpoint (NPDME).
2017.11.06	Made the following changes: • Added a note to the "Using PRBS Data Pattern Generator and Checker" chapter. • Updated "Variable Gain Amplifier (VGA) Voltage Swing Select" from "radp_vga_sel_0 to radp_vga_sel_7" to "radp_vga_sel_0 to radp_vga_sel_4". • Added sentence "The dynamic reconfiguration interface is compliant to the AVMM specifications". • Changed VGA select in Register Map for CTLE Setting from 3'b00-3'b111 to 3'b00-3'b100.
2017.05.08	Initial release.

7. Calibration

Transceivers include both analog and digital blocks that require calibration to compensate for process, voltage, and temperature (PVT) variations. Cyclone 10 GX transceiver uses hardened Precision Signal Integrity Calibration Engine (PreSICE) to perform calibration routines.

Power-up Calibration and User Recalibration are the main types of calibration.

- Power-up calibration occurs automatically at device power-up. It runs during device configuration.
- If you perform dynamic reconfiguration, then you must perform User Recalibration. In this case, you are responsible for enabling the required calibration sequence.

Note:

- If you need to reconfigure the ATX PLL, use TX PLL switching mode or use local clock divider to achieve the new data rate and avoid recalibrating the ATX PLL.
- If you are recalibrating your fPLL, follow the fPLL-to-ATX PLL spacing guideline as stated in the "Transmit PLLs Spacing Guideline when using ATX PLLs and fPLLs" section in PLLs and Clock Networks chapter.

Cyclone 10 GX devices use CLKUSR for clocking the transceiver calibration. To successfully complete the calibration process, the CLKUSR clock must be within specification, stable and free running at the start of FPGA configuration. Also, all reference clocks driving transceiver PLLs (ATX PLL, fPLL, CDR/CMU PLL) must be stable and free running at start of FPGA configuration. For more information about CLKUSR pin requirements, refer to the *Cyclone 10 GX GX, GT, and SX Device Family Pin Connection Guidelines*

Related Information

- Transmit PLLs Spacing Guidelines when using ATX PLLs and fPLLs on page 200
- Cyclone 10 GX Device Family Pin Connection Guidelines

7.1. Reconfiguration Interface and Arbitration with PreSICE Calibration Engine

In Cyclone 10 GX devices, calibration is performed using the Precision Signal Integrity Calibration Engine (PreSICE). The PreSICE includes an Avalon-MM interface to access the transceiver channel and PLL programmable registers. This Avalon-MM interface includes a communication mechanism that enables you to request specific calibration sequences from the calibration controller.

The PreSICE Avalon-MM interface and user Avalon-MM reconfiguration interface both share an internal configuration bus. This bus is arbitrated to gain access to the transceiver channel and PLL programmable registers, and the calibration registers.

There are two ways to check who has the access to the internal configuration bus:

- Use `reconfig_waitrequest`
- Use capability registers

The Native PHY IP core and PLL default setting is to use `reconfig_waitrequest`. When PreSICE controls the internal configuration bus, the `reconfig_waitrequest` from the internal configuration bus is high. When user access is granted, the `reconfig_waitrequest` from the internal configuration bus goes low. At the Avalon-MM reconfiguration interface, the `reconfig_waitrequest` can come from a few places inside Native PHY IP core. For example, it can come from the internal configuration bus, streamer, and so on. They are bundled together and become single `reconfig_waitrequest` at the Avalon-MM reconfiguration interface. The `reconfig_address` determines which `reconfig_waitrequest` to show at the Avalon-MM reconfiguration interface. After you return the internal configuration bus to PreSICE, the `reconfig_waitrequest` from the internal configuration bus is high. If you set the `reconfig_address` to the streamer offset address at the Avalon-MM reconfiguration interface during calibration, the `reconfig_waitrequest` can be low before calibration is finished. If you keep the `reconfig_address` the same as the internal configuration bus offset address during calibration, the `reconfig_waitrequest` at the Avalon-MM reconfiguration interface is high until PreSICE returns the internal configuration bus to you. It is important to keep reconfig_address static during calibration.

To use capability registers to check bus arbitration, you can do the following when generating the IP:

1. Select **Enable dynamic reconfiguration** from the **Dynamic Reconfiguration** tab.
2. Select both the **Separate reconfig_waitrequest from the status of AVMM arbitration with PreSICE** and **Enable control and status registers** options.

You can read the capability register 0x281[2] to check who is controlling the channel access, and read the capability register 0x280[2] to check who is controlling the PLL access. When **Separate reconfig_waitrequest from the status of AVMM arbitration with PreSICE** and **Enable control and status registers** are enabled, the `reconfig_waitrequest` is not asserted high when PreSICE controls the internal configuration bus.

To return the internal configuration bus to PreSICE:

- **Write 0x1 to offset address 0x0[7:0] if any calibration bit is enabled from offset address 0x100.**
- **Write 0x3 to offset address 0x0[7:0] if no calibration bit has been enabled from offset address 0x100.**

To check if the calibration process is running, do one of the following:

- Monitor the `pll_cal_busy`, `tx_cal_busy`, and `rx_cal_busy` signals.
- Read the `*_cal_busy` signal status from the capability registers.

The `*_cal_busy` signals remain asserted as long as the calibration process is running. To check whether or not calibration is done, you can read the capability registers or check the `*_cal_busy` signals. The `reconfig_waitrequest` from the Avalon-MM reconfiguration interface is not a reliable indicator to check whether or not

calibration is done. If you write 0x2 to 0x0 during calibration, PreSICE can stop the calibration process and return the internal configuration bus back to you; therefore, calibration is not done while the `reconfig_waitrequest` is low. The PMA `tx_cal_busy` and `rx_cal_busy` are from the same internal node which cannot be separated from the hardware. Configure the capability register 0x281[5:4] to enable or disable `tx_cal_busy` or `rx_cal_busy` individually through the Avalon-MM reconfiguration interface.

Related Information

- Arbitration on page 325
- *Avalon Interface Specifications*
- Reconfiguration Interface and Dynamic Reconfiguration Chapter on page 315

7.2. Calibration Registers

The Cyclone 10 GX transceiver PMA and PLLs include the following types of registers for calibration:

- Avalon-MM interface arbitration registers
- Calibration enable registers
- Capability registers
- Rate switch flag registers

The Avalon-MM interface arbitration registers enable you to request internal configuration bus access.

The PMA and PLL calibration enable registers for user recalibration are mapped to offset address 0x100. All calibration enable registers are self-cleared after the calibration process is completed.

The `tx_cal_busy`, `rx_cal_busy`, ATX PLL `pll_cal_busy`, and fPLL `pll_cal_busy` signals are available from the capability registers.

The rate switch flag registers are only used for CDR rate change.

7.2.1. Avalon Memory-Mapped Interface Arbitration Registers

Table 206. Avalon Memory-Mapped Interface Arbitration Registers

Bit	Offset Address	Description
[0]	0x0[32]	This bit arbitrates the control of Avalon memory-mapped interface. • Set this bit to 0 to request control of the internal configuration bus by user. • Set this bit to 1 to pass the internal configuration bus control to PreSICE.
[1]	0x0	This bit indicates whether or not calibration is done. This is the inverted `cal_busy` signal. You can write to this bit; however, if you accidentally write 0x0 without enabling any calibration bit in
		continued...

[32] The transceiver channel, ATX PLL, and fPLL use the same offset address.

Bit	Offset Address	Description
		0x100, PreSICE may not set this bit to 0x1, and `cal_busy` remains high. Channel reset is triggered if `cal_busy` is connected to the reset controller. When Read: • 1'b1: calibration done • 1'b0: calibration not done When Write: • 1'b1: If user doesn't want to trigger calibration (with bit-0 1'b0 or 1'b1) • 1'b0: To trigger calibration (by also writing 1'b1 to bit-0) The `cal_busy` signal is activated two clock cycles after you write 0x0 to this bit.

Note: During calibration when Nios® (PreSICE) is controlling the internal configuration bus, you can not read offset address 0x0. However, you can write 0x0 to offset address 0x0[0] to request bus access.

7.2.2. Transceiver Channel Calibration Registers

Table 207. Transceiver Channel PMA Calibration Registers

Bit	PMA Calibration Enable Register Offset Address 0x100
0	Reserved
1	PMA RX calibration enable. Set 1, to enable calibration.(33)
2	Reserved
3	Reserved
4	Reserved
5	PMA TX calibration enable. Set 1, to enable calibration.
6	Adaptation mode. Set 0, to disable adaptation mode.
7	Reserved

7.2.3. Fractional PLL Calibration Registers

Table 208. Fractional PLL Calibration Registers

Bit	fPLL Calibration Enable Register Offset Address 0x100
0	Reserved
1	fPLL calibration enable. Set 1 to enable calibration.

(33) CDR/CMU PLL calibration is part of PMA RX calibration.

7.2.4. ATX PLL Calibration Registers

Table 209. ATX PLL Calibration Registers

Bit	ATX PLL Calibration Enable Register Offset Address 0x100
0	ATX PLL calibration enable. Set 1 to enable calibration.
1	Reserved

During calibration when `reconfig_waitrequest` is high, you cannot read or write calibration enable registers.

To enable calibration, you must perform a read-modify-write on offset address 0x100. The following steps are an example of how to enable the ATXPLL calibration enable bit:

1. Read the offset address 0x100.
2. Keep the value from MSB[7:1] and set LSB[0] to 1.
3. Write new value to offset address 0x100.

7.2.5. Capability Registers

Capability registers allow you to read calibration status through the Avalon-MM reconfiguration interface. They are soft logic and reside in the FPGA fabric.

Reading capability registers does not require bus arbitration. You can read them during the calibration process.

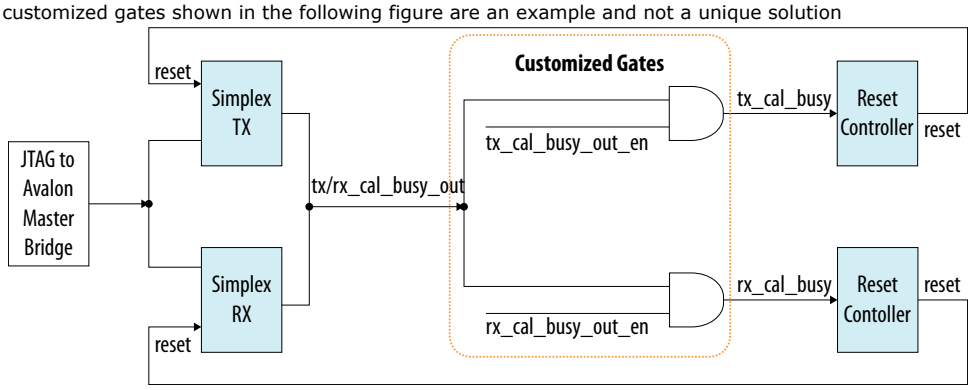
To use capability registers to check calibration status, you must enable the capability registers when generating the Native PHY or PLL IP cores. To enable the capability registers, select the **Enable capability registers** option in the **Dynamic Reconfiguration** tab.

The `tx_cal_busy` and `rx_cal_busy` signals from the hard PHY are from the same hardware and change state (high/low) concurrently during calibration. The register bits 0x281[5:4] are defined to solve this issue. This prevents a TX channel being affected by RX calibration, or an RX channel being affected by TX calibration. This feature cannot be enabled, when a Simplex TX and Simplex RX channel merging is involved. To merge a Simplex TX and a Simplex RX channel into one physical channel, refer to Dynamic Reconfiguration Interface Merging Across Multiple IP Blocks on page 351.

7.2.5.1. Rules to Build Customized Gating Logic to Separate tx_cal_busy and rx_cal_busy signals

Figure 221. An Example of an AND Gate used as Customized Logic

The customized gates shown in the following figure are an example and not a unique solution



The capability register is not available when merging a Simplex TX and a Simplex RX signal into the same physical channel. The `tx_cal_busy_out` and `rx_cal_busy_out` signals share the same port. So, you should build customized gating logic to separate them.

- The `tx_cal_busy_out_en` signal enables the `tx_cal_busy` output.
- The `rx_cal_busy_out_en` signal enables the `rx_cal_busy` output.
- At power up, `tx_cal_busy_out_en` and `rx_cal_busy_out_en` should be set to "1".
- At normal operation:
  - When the RX is calibrating, setting `tx_cal_busy_out_en` to "0" and `rx_cal_busy_out_en` to "1" disables tx_cal_busy, so the TX does not reset while RX is calibrating.
  - When the TX is calibrating, setting `rx_cal_busy_out_en` to "0" and `tx_cal_busy_out_en` to "1" disables `rx_cal_busy`, so the RX does not reset while TX is calibrating.

7.2.5.2. PMA Capability Registers for Calibration Status

Bit	Description
0x281[5]	PMA channel `rx_cal_busy` output enable. The power up default value is 0x1. 0x1: The `rx_cal_busy` output and 0x281[1] are asserted high whenever PMA TX or RX calibration is running. 0x0: The `rx_cal_busy` output or 0x281[1] is never asserted high.
0x281[4]	PMA channel `tx_cal_busy` output enable. The power up default value is 0x1. 0x1: The `tx_cal_busy` output and 0x281[0] are asserted high whenever PMA TX or RX calibration is running. 0x0: The `tx_cal_busy` output or 0x281[0] is never asserted high.
0x281[2]	PreSICE Avalon-MM interface control. This register is available to check who controls the bus, no matter if, separate `reconfig_waitrequest` from the status of AVMM arbitration with PreSICE is enabled or not. 0x1: PreSICE is controlling the internal configuration bus.

continued...

Bit	Description
	0x0: The user has control of the internal configuration bus.
0x281[1]	PMA channel `rx_cal_busy` active high 0x1: PMA RX calibration is running 0x0: PMA RX calibration is done
0x281[0]	PMA channel `tx_cal_busy` active high 0x1: PMA TX calibration is running 0x0: PMA TX calibration is done

The PMA 0x281[5:4] is used to isolate the TX and RX calibration busy status. If you want `rx_cal_busy` unchanged during the TX calibration, you must set 0x281[5] to 0x0 before returning the bus to PreSICE. The channel RX is not reset due to the TX calibration. If you want `tx_cal_busy` unchanged during the RX calibration, you must set 0x281[4] to 0x0 before returning the bus to PreSICE. The channel TX is not reset due to the RX calibration. If you accidentally write 0x00 to 0x281[5:4], `tx_cal_busy` and `rx_cal_busy` is never activated to high in the user interface. Neither of the 0x281[1:0] registers go high either.

Table 210. ATX PLL Capability Registers for Calibration Status

Bit	Description
0x280[2]	PreSICE Avalon-MM interface control. This register is available to check who controls the bus, no matter if, separate `reconfig_waitrequest` from the status of AVMM arbitration with PreSICE is enabled or not. 0x1: PreSICE is controlling the internal configuration bus. 0x0: The user has control of the internal configuration bus.
0x280[1]	ATX PLL `pll_cal_busy` 0x1: ATX PLL calibration is running 0x0: ATX PLL calibration is done

Table 211. fPLL Capability Registers for Calibration Status

Bit	Description
0x280[2]	PreSICE Avalon-MM interface control 0x1: PreSICE is controlling the internal configuration bus. This register is available to check who controls the bus, no matter if, separate `reconfig_waitrequest` from the status of AVMM arbitration with PreSICE is enabled or not. 0x0: The user has control of the internal configuration bus.
0x280[1]	fPLL `pll_cal_busy` 0x1: fPLL calibration is running 0x0: fPLL calibration is done

7.2.6. Rate Switch Flag Register

The rate switch flag is for CDR charge pump calibration. Each SOF has CDR default charge pump settings. After power up, these settings are loaded into the PreSICE memory space. If you change the line rate, it may require new charge pump settings, which is stored into the Avalon-MM reconfiguration register space. During RX PMA calibration (including CDR), PreSICE needs to know which set of CDR charge pump setting to use.

If you set 0x166[7] = 0x1, PreSICE assumes the setting in its memory space is still valid. If after a rate change you set 0x166[7]=0x0, PreSICE uses the setting from the Avalon-MM reconfiguration register uploaded from the dynamic reconfiguration interface or MIF streamed in. After calibration, 0x166[7] = 0x1 is set automatically and PreSICE uses the settings in its memory space. The rate switch flag only tells PreSICE where to obtain the CDR charge pump settings for CDR calibration. The rate switch flag should be used only when there is a rate change.

Multiple MIF files are required for rate change and reconfiguration. When the MIF, which you want to stream in, has CDR charge pump setting 0x139[7] and 0x133[7:5] that is different from the previous MIF, you need to recalibrate with 0x166[7]=0x0. If you stream in the whole MIF, the 0x166[7] is set to correct value inside the MIF. If you stream in reduced MIF, you need to check if CDR charge pump setting 0x139[7] and 0x133[7:5] are inside the reduced MIF or not. If the reduced MIF has CDR charge pump setting 0x139[7] and 0x133[7:5] updated, you need to set 0x166[7]=0x0, if the reduced MIF does not include 0x139[7] and 0x133[7:5], you need to set 0x166[7]=0x1.

Table 212. Rate Switch Flag Register for CDR Calibration

Bit	Description
0x166[7]	Rate switch flag register. Power up default value is 0x1. 0x1, PreSICE uses the default CDR charge pump bandwidth from the default memory space. 0x0, PreSICE uses the CDR charge pump bandwidth setting from the dynamically reconfigurable (DPRIO) register space.

If you use the Avalon-MM reconfiguration interface to perform a rate change, you must write 0x0 to 0x166[7] before returning the bus to PreSICE.

7.3. Power-up Calibration

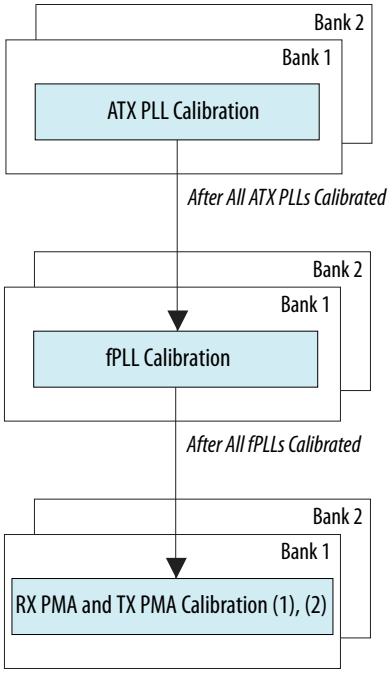
After the device is powered up and programmed, PreSICE automatically initiates the calibration process. The calibration process may continue during device programming. The time required after device power-up to complete the calibration process can vary by device. The total time taken can extend into the user-mode. The `cal_busy` signals deassert to indicate the completion of the calibration process. You must ensure that the transceiver reset sequence in your design waits for the calibration to complete before performing the required reset sequence of the transceiver PLL and the transceiver channel.

The PreSICE may still control the internal configuration bus even after power-up calibration is complete. You can request access when needed. If a system has an fPLL, an ATX PLL, and channels, the fPLL `cal_busy` signal goes low first. The ATX PLL `cal_busy` signal goes low after the channels' `tx_cal_busy` and `rx_cal_busy` signals. Intel recommends that you wait until all `*_cal_busy` signals are low before requesting any access.

All power-up calibration starts from voltage regulator (Vreg) calibration for all banks and channels.

Figure 222. Power-up Calibration Sequence for Non-PCIe Hard IP (HIP) Channels

For applications not using PCIe Hard IP, the power-up calibration starts from Vreg calibration for all banks and channels. Then, PreSICE calibration is done in the sequence as shown in the following figure.

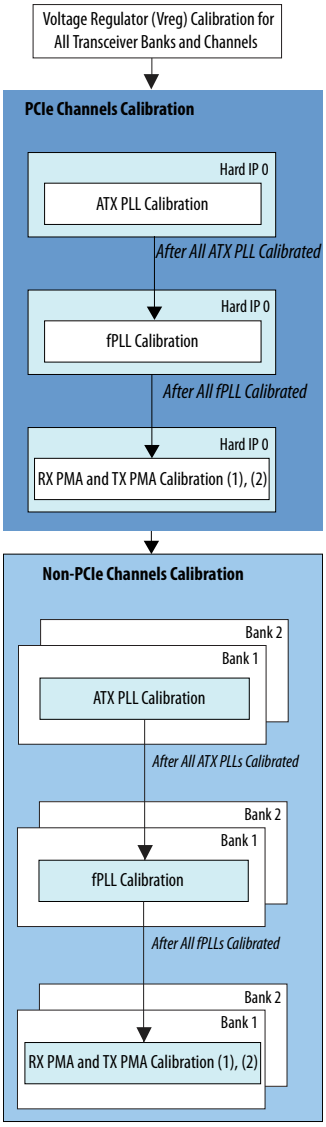


Notes:
(1) CDR/CMU PLL calibration is part of RX PMA calibration.
(2) For power-up calibration, RX PMA calibration happens before TX PMA calibration.

For applications using both PCIe Hard IP and non-PCIe channels, the power-up calibration sequence is:

1. Vreg calibration for all banks and channels.
2. PCIe Hard IP 0 calibration (if used).
3. Calibration of all non-PCIe Hard IP channels in calibration sequence.

Figure 223. Power-up Calibration Sequence for PCIe Hard IP and non-PCIe Channels



Notes:
(1) CDR/CMU PLL calibration is part of RX PMA calibration.
(2) For power-up calibration, RX PMA calibration happens before TX PMA calibration.

7.4. User Recalibration

7.4.1. Conditions That Require User Recalibration

Transceiver Reference Clock Availability and Stability After Device Power-Up

- During device power up, CLKUSR is asserted and running, but the transceiver reference clock remains deasserted until after the power up process is complete.
- During device power up, CLKUSR and the transceiver reference clock are asserted and running. When the device power up process is complete, the transceiver reference clock changes frequency. Either the transceiver reference clock could become unstable, or your application requires a different transceiver reference clock during normal operation, which could cause a data rate change.

After a Dynamic Reconfiguration Process That Triggers a Data Rate Change

After device power up in normal operation, you reconfigure the transceiver data rate by changing the channel configurations or the PLLs, recalibrate the:

- ATX PLL if ATX PLL has new VCO frequency to support new data rate.
- fPLL if the fPLL has new VCO frequency to support new data rate.

  Note: fPLL recalibration is not needed if the dynamic reconfiguration method used to achieve new data rate (new VCO frequency) is done using the fPLL L counter /1,2,4,8 division factor.
- CDR/CMU as TX PLL. You must recalibrate the RX PMA followed by a TX PMA recalibration of the channel which uses the CMU as TX PLL.
- RX PMA and TX PMA channel if the transceiver configuration changes to support new data rates.

Other Conditions That Require User Recalibration

- Recalibrate the fPLL if the fPLL is connected as a second PLL (downstream cascaded PLL). The downstream fPLL received the reference clock from the upstream PLL (could be from fPLL/ CDR). Recalibrating the second fPLL is important especially if the upstream PLL output clock (which is the downstream fPLL's reference clock) is not present or stable during power-up calibration.
- For ATX PLL or fPLL used to drive PLL feedback compensation bonding, recalibrate the PLL after power up calibration.

Note: You should avoid recalibrating the ATX PLL if another TX channel is in transmitting mode (clocked by another ATX PLL in the device). You need to do this to prevent a potential BER on neighboring RX channel placed next to a TX channel clocked by the ATX PLL. You can recalibrate the ATX PLL only if:

- The other TX channel that is in transmitting mode is clocked by fPLL or
- The other TX channel (clocked by another ATX PLL) must be placed under reset condition

If you are recalibrating your fPLL, follow the fPLL-to-ATX PLL spacing guideline as stated in the "Transmit PLLs Spacing Guideline when using ATX PLLs and fPLLs" section under PLLs and Clock Networks chapter.

You can initiate the recalibration process by writing to the specific recalibration registers. You must also reset the transceivers after performing user recalibration. For example, if you perform data rate auto-negotiation that involves PLL reconfiguration, and PLL and channel interface switching, then you must reset the transceivers.

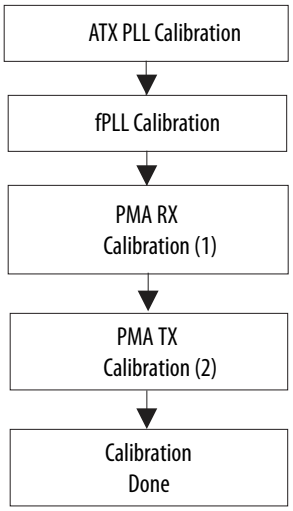
The proper reset sequence is required after calibration. Intel recommends you use the Transceiver PHY Reset Controller IP which has `tx_cal_busy` and `rx_cal_busy` inputs and follow Intel's recommended reset sequence. You need to connect `tx_cal_busy` and `rx_cal_busy` from the Native PHY IP core outputs to the reset controller inputs in your design. Reset upon calibration is automatically processed when you perform user recalibration.

Related Information

- Implementing PLL Cascading on page 240
- Implementing PLL Feedback Compensation Bonding Mode on page 237
- Recommended Reset Sequence on page 246
- Transmit PLLs Spacing Guidelines when using ATX PLLs and fPLLs on page 200

7.4.2. User Recalibration Sequence

Figure 224. User Recalibration Sequence for ATX PLL, fPLL and Native PHY IP (RX PMA / TX PMA)



Note:
(1) If you are using the CDR/CMU PLL, you need to trigger RX PMA recalibration.
(2) If you are using the CMU PLL as TX PLL, you need to trigger RX PMA recalibration followed by a TX PMA recalibration.

User recalibration requires access to the internal configuration bus and calibration registers through the Avalon-MM reconfiguration interface. Follow the recalibration example steps detailed in *Calibration Example* to perform a user recalibration process for each ATX PLL IP, fPLL IP and Native PHY IP (RX PMA / TX PMA).

Related Information

Calibration Example on page 385

7.5. Calibration Example

7.5.1. ATX PLL Recalibration

When you use the ATX PLL in your application, and it requires a line rate or clock frequency change, you must recalibrate the ATX PLL after you have made the changes.

Follow these steps to recalibrate the ATX PLL:

1. Request user access to the internal configuration bus by writing 0x2 to offset address 0x0[7:0].
2. Wait for `reconfig_waitrequest` to be deasserted (logic low) or wait until capability register of PreSICE Avalon-MM interface control 0x280[2]=0x0.
3. To calibrate the ATX PLL, perform a Read-Modify-Write of 0x1 to bit[0] of address 0x100 of the ATX PLL.
4. Release the internal configuration bus to PreSICE to perform recalibration by writing 0x1 to offset address 0x0[7:0].
5. Periodically check the `*cal_busy` output signals or read the capability registers 0x280[1] to check `*cal_busy` status until calibration is complete.

Note: You should avoid recalibrating the ATX PLL if another TX channel is in transmitting mode (clocked by another ATX PLL in the device). You need to do this to prevent a potential BER on neighboring RX channel placed next to a TX channel clocked by the ATX PLL. You can recalibrate the ATX PLL only if:

1. The other TX channel that is in transmitting mode is clocked by fPLL or
2. The other TX channel (clocked by another ATX PLL) must be place under reset condition.

7.5.2. Fractional PLL Recalibration

Follow these steps to recalibrate the fPLL:

1. Request user access to the internal configuration bus by writing 0x2 to offset address 0x0[7:0].
2. Wait for `reconfig_waitrequest` to be deasserted (logic low) or wait until capability register of PreSICE Avalon-MM interface control 0x280[2]=0x0.
3. To recalibrate the fPLL, Read-Modify-Write 0x1 to bit[1] of address 0x100 of the fPLL.
4. Release the internal configuration bus to PreSICE to perform recalibration by writing 0x1 to offset address 0x0[7:0].
5. Periodically check the `*cal_busy` output signals or read the capability registers 0x280[1] to check `*cal_busy` status until calibration is complete.

Note: If you are recalibrating your fPLL and have ATX PLL used on the same side of the device, follow the fPLL-to-ATX PLL spacing guideline as stated in the "Transmit PLLs Spacing Guideline when using ATX PLLs and fPLLs" section under PLLs and Clock Networks chapter.

Related Information

Transmit PLLs Spacing Guidelines when using ATX PLLs and fPLLs on page 200

7.5.3. CDR/CMU PLL Recalibration

CDR/CMU PLL user recalibration is integrated into the PMA RX user recalibration. Enabling 0x100[1] PMA RX calibration recalibrates the CDR/CMU PLL.

7.5.4. PMA Recalibration

PMA calibration includes:

- PMA TX calibration
- PMA RX calibration

The PMA RX calibration includes CDR/CMU PLL calibration, offset cancellation calibration, and V_{CM} calibration. The TX PMA calibration includes TX termination, Vod, and DCD calibration.

Follow these steps to recalibrate the PMA:

1. Request access to the internal configuration bus by writing 0x2 to offset address 0x0[7:0].
2. Wait for `reconfig_waitrequest` to be deasserted (logic low), or wait until capability register of PreSICE Avalon-MM interface control 0x281[2]=0x0.
3. Configure the PMA calibration enable register 0x100.
   - Read-Modify-Write 0x1 to 0x100[1] to start PMA RX calibration.
   - Read-Modify-Write 0x1 to 0x100[5] to start PMA TX calibration.
   - Read-Modify-Write 0x0 to 0x100[6] to disable adaptation mode.
4. If there is a data rate change in the CDR, set the rate switch flag register 0x166[7] for PMA RX calibration.
   - Read-Modify-Write 0x1 to offset address 0x166[7] if no rate switch.
   - Read-Modify-Write 0x0 to offset address 0x166[7] if switched rate with different CDR bandwidth setting.

   Note: Refer to *Rate Switch Flag Register*for more information.
5. Do Read-Modify-Write the proper value to capability register 0x281[5:4] for PMA calibration to enable/disable `tx_cal_busy` or `rx_cal_busy` output.
   - To enable `rx_cal_busy`, Read-Modify-Write 0x1 to 0x281[5].
   - To disable `rx_cal_busy`, Read-Modify-Write 0x0 to 0x281[5].
   - To enable `tx_cal_busy`, Read-Modify-Write 0x1 to 0x281[4].
   - To disable `tx_cal_busy`, Read-Modify-Write 0x0 to 0x281[4].

   Note: Refer to *PMA Capability Registers for Calibration Status* for more information.
6. Release the internal configuration bus to PreSICE to perform recalibration by writing 0x1 to offset address 0x0[7:0].
7. Periodically check the `*cal_busy` output signals or read the capability registers 0x281[1:0] to check `*cal_busy` status until calibration is complete.

Related Information

- Transceiver Channel Calibration Registers on page 376
  For more details about PMA recalibration
- Rate Switch Flag Register on page 379
- PMA Capability Registers for Calibration Status on page 378

7.6. Calibration Revision History

Document Version	Changes
2017.11.06	Made the following change: • Added a note "CDR/CMU PLL calibration is part of PMA RX calibration". • Updated the flow of "User Recalibration" topic. • For the Transceiver Channel PMA Calibration Registers table: Changed bit[6] from Reserved to Adaptation mode. Set 0, to disable adaptation mode. • Clarified the User Recalibration section. • Clarified the PMA Recalibration section.
2017.05.08	Initial release.

8. Analog Parameter Settings

Transceiver analog parameter settings are used to tune the analog functions in the physical medium attachment (PMA) and the PLL blocks while designing high-speed serial protocol solutions. You can use this feature to compensate for signal losses for high data rate communication.

Most transceiver parameters can be set using the **Parameter Editor** before generating the transceiver PHY IP. The parameters that depend on place and route decisions, device constraints, and tunable analog settings that cannot be set before IP generation are controlled in the following ways:

- Making analog parameter settings to I/O pins using the Assignment Editor.
- Updating the Quartus Prime Settings File (**.qsf**) with the known assignment.
- Using the Reconfiguration Interface to dynamically change the analog settings.

8.1. Making Analog Parameter Settings using the Assignment Editor

To make assignments using the **Assignment Editor**, complete the following steps:

1. On the **Assignments** menu, select **Assignment Editor**. The **Assignment Editor** appears.
2. Click inside the **Assignment Name** column and select the appropriate assignment. Refer to the *PMA Analog Settings* section for the list of available assignments for PMA analog settings.
3. Click inside the **Value** column and select the appropriate value for your assignment.
   The Quartus Prime software adds these instance assignments to the **.qsf** file for your project.

Related Information

Analog Parameter Settings List on page 389

8.2. Updating Quartus Settings File with the Known Assignment

The Quartus Prime Settings File (**.qsf**) contains all the entity-level assignments and settings for the current revision of the project. The Quartus Settings File is based on Tcl script syntax.

When you create assignments and settings using the **Parameter Editor** wizards and dialog boxes or Tcl commands, the Quartus Prime software automatically places the assignments at the end of the Quartus Prime Settings File. To control the analog parameters, you can directly add or modify the appropriate assignment in the Quartus Prime Settings File. The assignments you create are recognized, regardless of where you place them in the file.

Related Information

Quartus Prime Settings File (.qsf)
Describes the commands and options available to modify the assignments in the qsf file.

8.3. Analog Parameter Settings List

The following table lists the analog parameter settings for the transmitter and receiver. The details of each of these settings are in the sections following this table.

Note: When changing from one device to another, you must manually edit the settings in the Assignment Editor to obtain accurate **.qsf** files.

Table 213. Receiver Analog Parameter Settings

Analog Parameter Setting	Pin Planner or Assignment Editor Name	Assignment Destination	Usage Guideline
XCVR_C10_RX_TERM_SEL	Receiver On-Chip-Termination	RX serial data	On chip termination
XCVR_C10_RX_ONE_STAGE_ENABLE	Receiver High Data Rate Mode Equalizer	RX serial data	Continuous time-linear equalization (CTLE)
XCVR_C10_RX_EQ_DC_GAIN_TRIM	Receiver High Gain Mode Equalizer DC Gain Control	RX serial data	CTLE
XCVR_C10_RX_ADP_CTLE_ACGAIN_4S	Receiver High Gain Mode Equalizer AC Gain Control	RX serial data	CTLE
XCVR_C10_RX_ADP_CTLE_EQZ_1S_SEL	Receiver High Data Rate Mode Equalizer AC Gain Control	RX serial data	CTLE
XCVR_C10_RX_ADP_VGA_SEL	Receiver Variable Gain Amplifier Voltage Swing Select	RX serial data	VGA

Table 214. Transmitter Analog Parameter Settings

Analog Parameter Setting	Pin Planner or Assignment Editor Name	Assignment Destination	Usage Guideline
XCVR_C10_TX_TERM_SEL	Transmitter On-Chip Termination	TX serial data	On chip termination
XCVR_C10_TX_COMPENSATION_EN	Transmitter High-Speed Compensation	TX serial data	PDN ISI compensation
XCVR_C10_TX_SLEW_RATE_CTRL	Transmitter Slew Rate Control	TX serial data	Slew Rate
XCVR_C10_TX_PRE_EMP_SIGN_PRE_TAP_1T	Transmitter Pre-Emphasis First Pre-Tap Polarity	TX serial data	Pre-emphasis
XCVR_C10_TX_PRE_EMP_SIGN_PRE_TAP_2T	Transmitter Pre-Emphasis Second Pre-Tap Polarity	TX serial data	Pre-emphasis
XCVR_C10_TX_PRE_EMP_SIGN_1ST_POST_TAP	Transmitter Pre-Emphasis First Post-Tap Polarity	TX serial data	Pre-emphasis
XCVR_C10_TX_PRE_EMP_SIGN_2ND_POST_TAP	Transmitter Pre-Emphasis Second Post-Tap Polarity	TX serial data	Pre-emphasis
XCVR_C10_TX_PRE_EMP_SWITCHING_CTRL_PRE_TAP_1T	Transmitter Pre-Emphasis First Pre-Tap Magnitude	TX serial data	Pre-emphasis
XCVR_C10_TX_PRE_EMP_SWITCHING_CTRL_PRE_TAP_2T	Transmitter Pre-Emphasis Second Pre-Tap Magnitude	TX serial data	Pre-emphasis

continued...

Analog Parameter Setting	Pin Planner or Assignment Editor Name	Assignment Destination	Usage Guideline
XCVR_C10_TX_PRE_EMP_SWITCHING_CTRL_1ST_POST_TAP	Transmitter Pre-Emphasis First Post-Tap Magnitude	TX serial data	Pre-emphasis
XCVR_C10_TX_PRE_EMP_SWITCHING_CTRL_2ND_POST_TAP	Transmitter Pre-Emphasis Second Post-Tap Magnitude	TX serial data	Pre-emphasis
XCVR_C10_TX_VOD_OUTPUT_SWING_CTRL	Transmitter Output Swing Level	TX serial data	Differential output voltage

Table 215. Reference Clock Analog Parameter Settings

Analog Parameter Setting	Pin Planner or Assignment Editor Name	Assignment Destination	Usage Guideline
XCVR_C10_REFCLK_TERM_TRISTATE	Dedicated Reference Clock Pin Termination	Reference clock pin	On-chip termination

Note: You must set all the required analog settings according to your protocol configuration. If you do not set the appropriate settings, then the Quartus Prime software selects the default values which may not be appropriate for your protocol implementation.

8.4. Receiver General Analog Settings

8.4.1. XCVR_C10_RX_TERM_SEL

Pin planner or Assignment Editor Name

Receiver On-Chip Termination

Description

Controls the on-chip RX differential termination.

Note: You can either set this setting to R_R1 or R_R2.

Table 216. Available Options

Value	Description
R_EXT0	Tristate
R_R1	100 Ohm
R_R2	85 Ohm

Assign To

RX serial data pin.

Syntax

```
set_instance_assignment -name XCVR_C10_RX_TERM_SEL <value> -to
<rx_serial_data pin name>
```

8.5. Receiver Analog Equalization Settings

8.5.1. CTLE Settings

8.5.1.1. XCVR_C10_RX_ONE_STAGE_ENABLE

Pin planner or Assignment Editor Name

Receiver High Data Rate Mode Equalizer

Description

Selects between the CTLE high gain mode or CTLE high data rate mode for the equalizer. If no assignment is made, **S1_MODE** is chosen by default.

Table 217. Available Options

Value in QSF	Value in Assignment Editor	Description
NON_S1_MODE	**Off**	Selects high gain mode.
S1_MODE	**On**	Selects high data rate mode.

Assign To

RX serial data pin.

Syntax

```
set_instance_assignment -name XCVR_C10_RX_ONE_STAGE_ENABLE
<value> -to <rx_serial_data pin name>
```

8.5.1.2. XCVR_C10_RX_EQ_DC_GAIN_TRIM

Pin planner or Assignment Editor Name

Receiver High Gain Mode Equalizer DC Gain Control

Description

Controls the DC gain of the continuous time linear equalizer (CTLE) in high gain mode. A higher gain setting results in a larger DC gain.

For RX_LINK=SR, the default value is **STG2_GAIN7**.

For PCIe, default value is **NO_DC_GAIN**.

Table 218. Available Options

Value in TTK	Value in Assignment Editor / qsf	Description
DC Gain 0	**NO_DC_GAIN**	No DC gain
DC Gain 1	**STG1_GAIN7**	Equalizer DC gain setting 6
DC Gain 2	**STG2_GAIN7**	Equalizer DC gain setting 13
DC Gain 3	**STG3_GAIN7**	Equalizer DC gain setting 20
DC Gain 4	**STG4_GAIN7**	Equalizer DC gain setting 27

Assign To

RX serial data pin.

Syntax

```
set_instance_assignment -name XCVR_C10_RX_EQ_DC_GAIN_TRIM <value>
-to <rx_serial_data pin name>
```

8.5.1.3. XCVR_C10_RX_ADP_CTLE_ACGAIN_4S

Pin planner or Assignment Editor Name

Receiver High Gain Mode Equalizer AC Gain Control

Description

Controls the AC gain of the continuous time linear equalizer (CTLE) in high gain mode.

The default value is **RADP_CTLE_ACGAIN_4S_1**.

Table 219. Available Options

Value	Description
RADP_CTLE_ACGAIN_4S_<0 to 28>	CTLE AC gain setting <0 to 28>

Assign To

RX serial data pin.

Syntax

```
set_instance_assignment -name XCVR_C10_RX_ADP_CTLE_ACGAIN_4S
<value> -to <rx_serial_data pin name>
```

8.5.1.4. XCVR_C10_RX_ADP_CTLE_EQZ_1S_SEL

Pin planner or Assignment Editor Name

Receiver High Data Rate Mode Equalizer AC Gain Control

Description

Controls the AC gain of the continuous time linear equalizer (CTLE) in high data rate mode.

In high data rate mode, there is only one CTLE stage and 16 possible AC gain settings. Higher gain setting results in larger AC gain. The default value is set to **RADP_CTLE_EQZ_1S_SEL_3** i.e. CTLE AC Gain Setting 3. This QSF assignment only takes effect when one stage CTLE is enabled. If configured in four stage mode, it has no effect on CTLE gain value.

Table 220. Available Options

Value	Description
RADP_CTLE_EQZ_1S_SEL_<0 to 15>	CTLE AC Gain Setting < 0 to 15>

Assign To

RX serial data pin.

Syntax

```
set_instance_assignment -name XCVR_C10_RX_ADP_CTLE_EQZ_1S_SEL
<value> -to <rx_serial_data pin name>
```

8.5.2. VGA Settings

8.5.2.1. XCVR_C10_RX_ADP_VGA_SEL

Pin planner or Assignment Editor Name

Receiver Variable Gain Amplifier Voltage Swing Select

Description

The variable gain amplifier (VGA) amplifies the signal amplitude and ensures a constant voltage swing before the data is fed to the CDR for sampling. This assignment controls the VGA output voltage swing when adaptation mode of VGA is set to manual. The default value is set to **RADP_VGA_SEL_4** for VGA Output Voltage Swing Setting 4.

Default values for VGA if VGA are not set exclusively.

For PCIe Gen1 & Gen2: default value is **RADP_VGA_SEL_2**.

CTLE mode = NON_S1_MODE and datarate <= 4.5Gbps: default value is **RADP_VGA_SEL_2**.

CTLE mode = NON_S1_MODE and datarate > 4.5Gbps: default value is **RADP_VGA_SEL_4**.

CTLE mode = S1_MODE: default value is **RADP_VGA_SEL_2**.

Table 221. Available Options

Value	Description
RADP_VGA_SEL_<0 to 4>	VGA Output Voltage Swing Setting <0 to 4>

Assign To

RX serial data pin.

Syntax

```
set_instance_assignment -name XCVR_C10_RX_ADP_VGA_SEL <value> -to
<rx_serial_data pin name>
```

8.6. Transmitter General Analog Settings

8.6.1. XCVR_C10_TX_TERM_SEL

Pin planner or Assignment Editor Name

Transmitter On-Chip Termination

Description

Controls the on-chip TX differential termination for different protocols.

Table 222. Available Options

Value	Description
R_R1	100 Ohm
R_R2	85 Ohm

Note: When the data rate is less than or equal to 12.5 Gbps, the default value is R_R1.

8.6.2. XCVR_C10_TX_COMPENSATION_EN

Pin planner or Assignment Editor Name

Transmitter High-Speed Compensation

Description

Specifies if the power distribution network (PDN) induced inter-symbol interference (ISI) compensation is enabled or disabled in the TX driver. When enabled, it reduces the PDN induced ISI jitter, but increases the power consumption. Use this feature for high speed applications. It defaults to **ENABLE** for non-PCIe modes.

Table 223. Available Options

Value	Description
ENABLE	Compensation ON
DISABLE	Compensation OFF

Table 224. Rules

Data Rate	Value of XCVR_C10_TX_COMPENSATION_EN
PCIe Gen1, Gen2	DISABLE
Others	ENABLE/DISABLE

Assign To

TX serial data pin.

Syntax

```
set_instance_assignment -name XCVR_C10_TX_COMPENSATION_EN <value>
-to <tx_serial_data pin name>
```

Related Information

EPE (Early Power Estimator)

8.6.3. XCVR_C10_TX_SLEW_RATE_CTRL

Pin planner or Assignment Editor Name

Transmitter Slew Rate Control

Description

Specifies the slew rate of the output signal. The valid values span from the slowest rate to the fastest rate.

Note: Slew_R6 and Slew_R7 are not used.

Table 225. Available Options

Value	Description
SLEW_R0 to **SLEW_R5**	R0 is the slowest rate. R5 is the fastest rate.

If QSF is not specified, the following table lists the default values.

Table 226. Default Values

PCIe	
Value	**Default Slew Rate**
PCIe Gen1	SLEW_R4
PCIe Gen2	SLEW_R5

Non-PCIe				
Value (VCCT/VCCR)	**Default Slew Rate**			
	Data rate < 1G	**Data rate ≥ 1G to < 3G**	**Data rate ≥ 3G to < 6G**	**Data rate ≥ 6G**
0.95 V	SLEW_R0	SLEW_R2	SLEW_R3	SLEW_R5

Assign To

TX serial data pin.

Syntax

```
set_instance_assignment -name XCVR_C10_TX_SLEW_RATE_CTRL <value>
-to <tx_serial_data pin name>
```

Note: Currently, the Quartus Prime software doesn't choose the default values for Non-PCIe mode. You must specify the setting in the **.qsf** file.

8.7. Transmitter Pre-Emphasis Analog Settings

The programmable pre-emphasis block in the transmit buffer amplifies the high frequencies in the transmit data to compensate for attenuation in the transmission media.

Note: Ignore all the pre-emphasis QSF assignments for PCIe Gen1 and Gen2 if your transceiver link is within the PCI Express spec requirements. Quartus Prime assigns default values for these parameters.

8.7.1. XCVR_C10_TX_PRE_EMP_SIGN_PRE_TAP_1T

Pin planner or Assignment Editor Name

Transmitter Pre-Emphasis First Pre-Tap Polarity

Description

Controls the polarity of the first pre-tap for pre-emphasis. The default value is **FIR_PRE_1T_NEG**.

Table 227. Available Options

Value	Description
FIR_PRE_1T_POS	Positive pre-tap 1
FIR_PRE_1T_NEG	Negative pre-tap 1

Assign To

TX serial data pin.

Syntax

```
set_instance_assignment -name XCVR_C10_TX_PRE_EMP_SIGN_PRE_TAP_1T
<value> -to <tx_serial_data pin name>
```

8.7.2. XCVR_C10_TX_PRE_EMP_SIGN_PRE_TAP_2T

Pin Planner or Assignment Editor Name

Transmitter Pre-Emphasis Second Pre-Tap Polarity

Description

Controls the polarity of the second pre-tap for pre-emphasis. The default value is **FIR_PRE_2T_NEG**.

Table 228. Available Options

Value	Description
FIR_PRE_2T_POS	Positive pre-tap 2
FIR_PRE_2T_NEG	Negative pre-tap 2

Assign To

TX serial data pin.

Syntax

```
set_instance_assignment -name XCVR_C10_TX_PRE_EMP_SIGN_PRE_TAP_2T
<value> -to <tx_serial_data pin name>
```

8.7.3. XCVR_C10_TX_PRE_EMP_SIGN_1ST_POST_TAP

Pin planner or Assignment Editor Name

Transmitter Pre-Emphasis First Post-Tap Polarity

Description

Controls the polarity of the first post-tap for pre-emphasis. The default value is **FIR_POST_1T_NEG**.

Table 229. Available Options

Value	Description
FIR_POST_1T_POS	Positive post-tap 1
FIR_POST_1T_NEG	Negative post-tap1

Assign To

TX serial data pin.

Syntax

```
set_instance_assignment -name
XCVR_C10_TX_PRE_EMP_SIGN_1ST_POST_TAP <value> -to <tx_serial_data
pin name>
```

8.7.4. XCVR_C10_TX_PRE_EMP_SIGN_2ND_POST_TAP

Pin planner or Assignment Editor Name

Transmitter Pre-Emphasis Second Post-Tap Polarity

Description

Controls the polarity of the second post-tap for pre-emphasis. The default value is **FIR_POST_2T_NEG**.

Table 230. Available Options

Value	Description
FIR_POST_2T_POS	Positive post-tap 2
FIR_POST_2T_NEG	Negative post-tap 2

Assign To

TX serial data pin.

Syntax

```
set_instance_assignment -name
XCVR_C10_TX_PRE_EMP_SIGN_2ND_POST_TAP <value> -to <tx_serial_data
pin name>
```

8.7.5. XCVR_C10_TX_PRE_EMP_SWITCHING_CTRL_PRE_TAP_1T

Pin planner or Assignment Editor Name

Transmitter Pre-Emphasis First Pre-Tap Magnitude

Description

Controls the magnitude of the first pre-tap for pre-emphasis. The default value is **0**.

Table 231. Available Options

Value	Description
0 – 16	Magnitude 0 – 16

Note: Refer to *Cyclone 10 GX Pre-Emphasis and Output Swing Settings* spreadsheet for selecting legal pre-emphasis and differential output voltage settings.

Assign To

TX serial data pin.

Syntax

```
set_instance_assignment -name
XCVR_C10_TX_PRE_EMP_SWITCHING_CTRL_PRE_TAP_1T <value> -to
<tx_serial_data pin name>
```

Related Information

Cyclone 10 GX Pre-Emphasis and Output Swing Settings

8.7.6. XCVR_C10_TX_PRE_EMP_SWITCHING_CTRL_PRE_TAP_2T

Pin planner or Assignment Editor Name

Transmitter Pre-Emphasis Second Pre-Tap Magnitude

Description

Controls the magnitude of the second pre-tap for pre-emphasis. The default value is **0**.

Table 232. Available Options

Value	Description
0 – 7	Magnitude 0 – 7

Note: Refer to *Cyclone 10 GX Pre-Emphasis and Output Swing Settings* spreadsheet for selecting legal pre-emphasis and differential output voltage settings.

Assign To

TX serial data pin.

Syntax

```
set_instance_assignment -name
XCVR_C10_TX_PRE_EMP_SWITCHING_CTRL_PRE_TAP_2T <value> -to
<tx_serial_data pin name>
```

Related Information

Cyclone 10 GX Pre-Emphasis and Output Swing Settings

8.7.7. XCVR_C10_TX_PRE_EMP_SWITCHING_CTRL_1ST_POST_TAP

Pin planner or Assignment Editor Name

Transmitter Pre-Emphasis First Post-Tap Magnitude

Description

Controls the magnitude of the first post-tap for pre-emphasis. The default value is **0**.

Table 233. Available Options

Value	Description
0 –25	Magnitude 0 –25

Note: Refer to *Cyclone 10 GX Pre-Emphasis and Output Swing Settings* spreadsheet for selecting legal pre-emphasis and differential output voltage settings.

Assign To

TX serial data pin.

Syntax

```
set_instance_assignment -name
XCVR_C10_TX_PRE_EMP_SWITCHING_CTRL_1ST_POST_TAP <value> -to
<tx_serial_data pin name>
```

Related Information

Cyclone 10 GX Pre-Emphasis and Output Swing Settings

8.7.8. XCVR_C10_TX_PRE_EMP_SWITCHING_CTRL_2ND_POST_TAP

Pin planner or Assignment Editor Name

Transmitter Pre-Emphasis Second Post-Tap Magnitude

Description

Controls the magnitude of the second post-tap for pre-emphasis. The default value is **0**.

Table 234. Available Options

Value	Description
0 – 12	Magnitude 0 – 12

Note: Refer to *Cyclone 10 GX Pre-Emphasis and Output Swing Settings* spreadsheet for selecting legal pre-emphasis and differential output voltage settings.

Assign To

TX serial data.

Syntax

```
set_instance_assignment -name
XCVR_C10_TX_PRE_EMP_SWITCHING_CTRL_2ND_POST_TAP <value> -to
<tx_serial_data pin name>
```

Related Information

Cyclone 10 GX Pre-Emphasis and Output Swing Settings

8.8. Transmitter VOD Settings

8.8.1. XCVR_C10_TX_VOD_OUTPUT_SWING_CTRL

Pin planner or Assignment Editor Name

Transmitter Output Swing Level

Description

Controls the transmitter programmable output differential voltage swing. The default value is **31**.

Note: Ignore this assignment for PCIe Gen1 and Gen2 if your transceiver link is within the PCI Express spec requirements. The Quartus Prime software assigns default value for this parameter.

Table 235. Available Options

Value	Description
0 – 31	Magnitude 0 – 31

Note: Refer to *Cyclone 10 GX Pre-Emphasis and Output Swing Settings* spreadsheet for selecting legal pre-emphasis and differential output voltage settings.

Assign To

TX serial data pin.

Syntax

```
set_instance_assignment -name XCVR_C10_TX_VOD_OUTPUT_SWING_CTRL
<value> -to <tx_serial_data pin name>
```

Related Information

Cyclone 10 GX Pre-Emphasis and Output Swing Settings

8.9. Dedicated Reference Clock Settings

8.9.1. XCVR_C10_REFCLK_TERM_TRISTATE

Pin planner or Assignment Editor Name

Dedicated Reference Clock Pin Termination

Description

Specifies if the termination for dedicated reference clock pin is tri-stated. It defaults to **TRISTATE_OFF** for non-HCSL cases.

Table 236. Available Options

Value	Description
TRISTATE_OFF	Internal termination enabled and on-chip biasing circuitry enabled
TRISTATE_ON	Internal termination tri-stated. Off-chip termination and biasing circuitry must be implemented

Table 237. Rules

I/O Standard	Value
HCSL	TRISTATE_ON
CML	TRISTATE_OFF
LVPECL	TRISTATE_OFF
LVDS	TRISTATE_OFF

Assign To

Reference clock pin.

Syntax

```
set_instance_assignment -name XCVR_C10_REFCLK_TERM_TRISTATE
<value> -to <dedicated refclk pin name>
```

8.9.2. XCVR_C10_TX_XTX_PATH_ANALOG_MODE

Pin planner or Assignment Editor Name

Transmitter Analog Presets

Description

Specifies the TX pin swing settings such as vod output swing control, pre-emphasis and slew rate settings for various protocols. The presets are set by selecting an analog_mode setting. The selection is done as a **.QSF** setting on the TX pin. If individual **.QSF** is specified for the vod_output_swing_ctrl, pre-emphasis settings, the individual **.QSF** overrides the preset settings. The parameter defaults to USER_CUSTOM, which requires the individual **.QSF** to be set. Available TX preset values are shown in the Assignment Editor, Transmitter Analog Preset assignment section.

8.10. Unused Transceiver Channels Settings

Add the following Intel Cyclone 10 GX QSF assignments for RX pins to prevent performance degradation of unused receiver serial clocks over time. You can either use a global assignment or per-pin assignments. For per-pin assignments, you can specify a true or a complement RX pin.

Syntax

```
set_global_assignment -name PRESERVE_UNUSED_XCVR_CHANNEL ON
```

```
set_instance_assignment -name PRESERVE_UNUSED_XCVR_CHANNEL ON -to
<pin name (AF26, for example)>
```

8.11. Analog Parameter Settings Revision History

Document Version	Changes
2017.11.30	Made the following changes: • Removed information about QPI from the note in the "XCVR_C10_TX_TERM_SEL" section.
2017.11.06	Made the following changes: • Added new settings.
2017.05.08	Initial release.