7 Series FPGAs GTP Transceivers

User Guide

UG482 (v1.0) January 3, 2012

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
01/03/12	1.0	Initial Xilinx release.

Table of Contents

Chapter 3: Transmitter

Chapter 4: Receiver

About This Guide

Xilinx® 7 series FPGAs include three unified FPGA families that are all designed for lowest power to enable a common design to scale across families for optimal power, performance, and cost. The Artix™-7 family is optimized for lowest cost and absolute power for the highest volume applications. The Virtex®-7 family is optimized for highest system performance and capacity. The Kintex™-7 family is an innovative class of FPGAs optimized for the best price-performance. This guide serves as a technical reference describing the 7 series FPGAs GTP transceivers.

The 7 series FPGAs GTP transceivers user guide, part of an overall set of documentation on the 7 series FPGAs, is available on the Xilinx website at www.xilinx.com/7.

In this document:

- 7 series FPGAs GTP transceiver channel is abbreviated as GTP transceiver.
- GTPE2_CHANNEL is the name of the instantiation primitive that instantiates one GTP transceiver channel.
- GTPE2_COMMON is the name of the primitive that instantiates two ring oscillator PLLs (PLL0 and PLL1).
- A Quad or Q is a cluster or set of four GTP transceiver channels, one GTPE2_COMMON primitive, two differential reference clock pin pairs, and analog supply pins.

Guide Contents

This manual contains:

Additional Resources

To find additional documentation, see the Xilinx website at:

http://www.xilinx.com/support/documentation/index.htm.

To search the Answer Database of silicon, software, and IP questions and answers, or to create a technical support WebCase, see the Xilinx website at:

http://www.xilinx.com/support.

Additional References

These documents provide additional information useful to this document:

1. *High-Speed Serial I/O Made Simple*
   http://www.xilinx.com/publications/archives/books/serialio.pdf

XILINX®

Chapter 1

Transceiver and Tool Overview

Overview and Features

The 7 series FPGAs GTP transceiver is a power-efficient transceiver, supporting line rates between 500 Mb/s and 6.6 Gb/s. The GTP transceiver is highly configurable and tightly integrated with the programmable logic resources of the FPGA. Table 1-1 summarizes the features by functional group that support a wide variety of applications.

Table 1-1: **7 Series FPGAs Transceiver Features**

Group	Feature	GTP	GTX	GTH
PCS	2-byte internal datapath	x	x	x
	4-byte internal datapath		x	x
	8B/10B encoding and decoding	x	x	x
	64B/66B and 64B/67B support	x	x	x
	Comma detection and byte and word alignment	x	x	x
	PRBS generator and checker	x	x	x
	FIFO for clock correction and channel bonding	x	x	x
	Programmable FPGA logic interface	x	x	x
PMA	One shared LC tank PLL per Quad		x	x
	One ring oscillator PLL per channel		x	x
	Two shared ring oscillator PLLs per Quad	x		
	Flexible reference clocking options	x	x	x
	Decision feedback equalization (DFE)		x	x
	Power-efficient adaptive linear equalizer mode called the low-power mode (LPM)	x	x	x
	TX Pre-emphasis	x	x	x
	Beacon signaling for PCI Express® designs	x	x	x
	Out-of-band (OOB) signaling including COM signal support for Serial ATA (SATA) designs	x	x	x
	RX Margin Analysis	x	x	x

The GTP transceiver supports these use modes:

- PCI Express, Revision 1.1/2.0
- Interlaken
- 10 Gb Attachment Unit Interface (XAUI), Reduced Pin eXtended Attachment Unit Interface (RXAUI)
- Common Packet Radio Interface (CPRI™)/Open Base Station Architecture Initiative (OBSAI)
- OC-48
- OTU-1
- Serial RapidIO (SRIO)
- Serial Advanced Technology Attachment (SATA)/Serial Attached SCSI (SAS)
- Serial Digital Interface (SDI)

In comparison to prior generation transceivers in Spartan®-6 FPGAs, the GTP transceiver in the 7 series FPGAs has the following new or enhanced features:

- 2-byte internal datapath
- Two ring oscillator PLLs per Quad
- Power-efficient, adaptive continuous time linear equalizer (CTLE)
- RX margin analysis feature to provide non-destructive, 2-D post-equalization eye scan.

The first-time user is recommended to read *High-Speed Serial I/O Made Simple* [Ref 1], which discusses high-speed serial transceiver technology and its applications. The CORE Generator™ tool includes a wizard to automatically configure GTP transceivers to support configurations for different protocols or perform custom configuration. The GTP transceiver offers a data rate range and features that allow physical layer support for various protocols.

Figure 1-1, page 13 shows the GTP transceiver placement in an example Artix™-7 device (XCA100T). This device has 8 GTP transceivers.

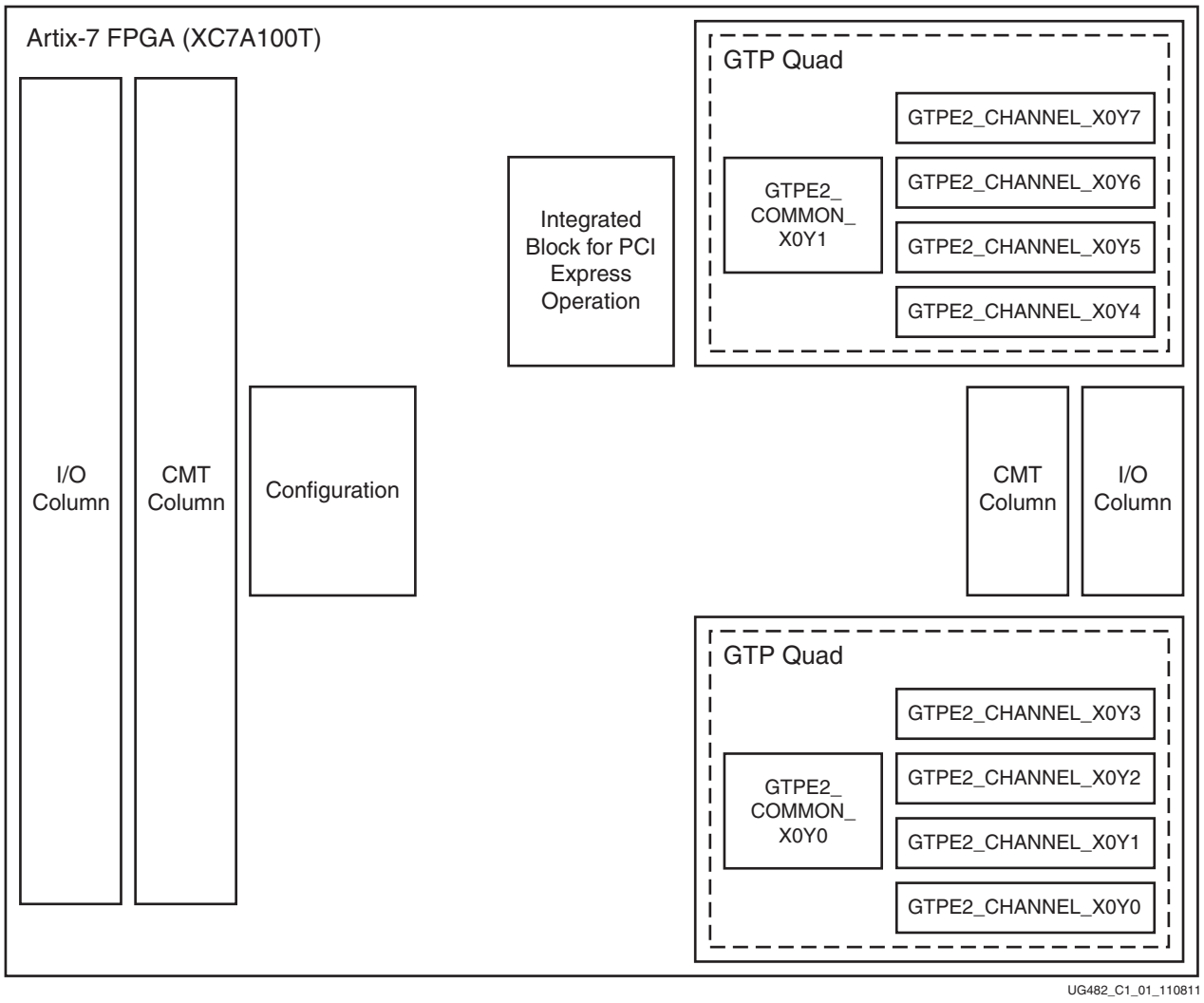


Figure 1-1: **GTP Transceiver Inside Artix-7 XC7A100T FPGA**

Additional information on the functional blocks of 7 series FPGAs is available at:

UG470, *7 Series FPGAs Configuration User Guide* provides more information on the configuration and clocking.

UG471, *7 Series FPGAs SelectIO Resources User Guide* provides more information on the I/O blocks.

UG472, *7 Series FPGAs Clocking Resources User Guide* provides more information on the mixed mode clock manager (MMCM).

Figure 1-2 illustrates the clustering of four GTPE2_CHANNEL primitives and one GTPE2_COMMON primitive to form a Quad.

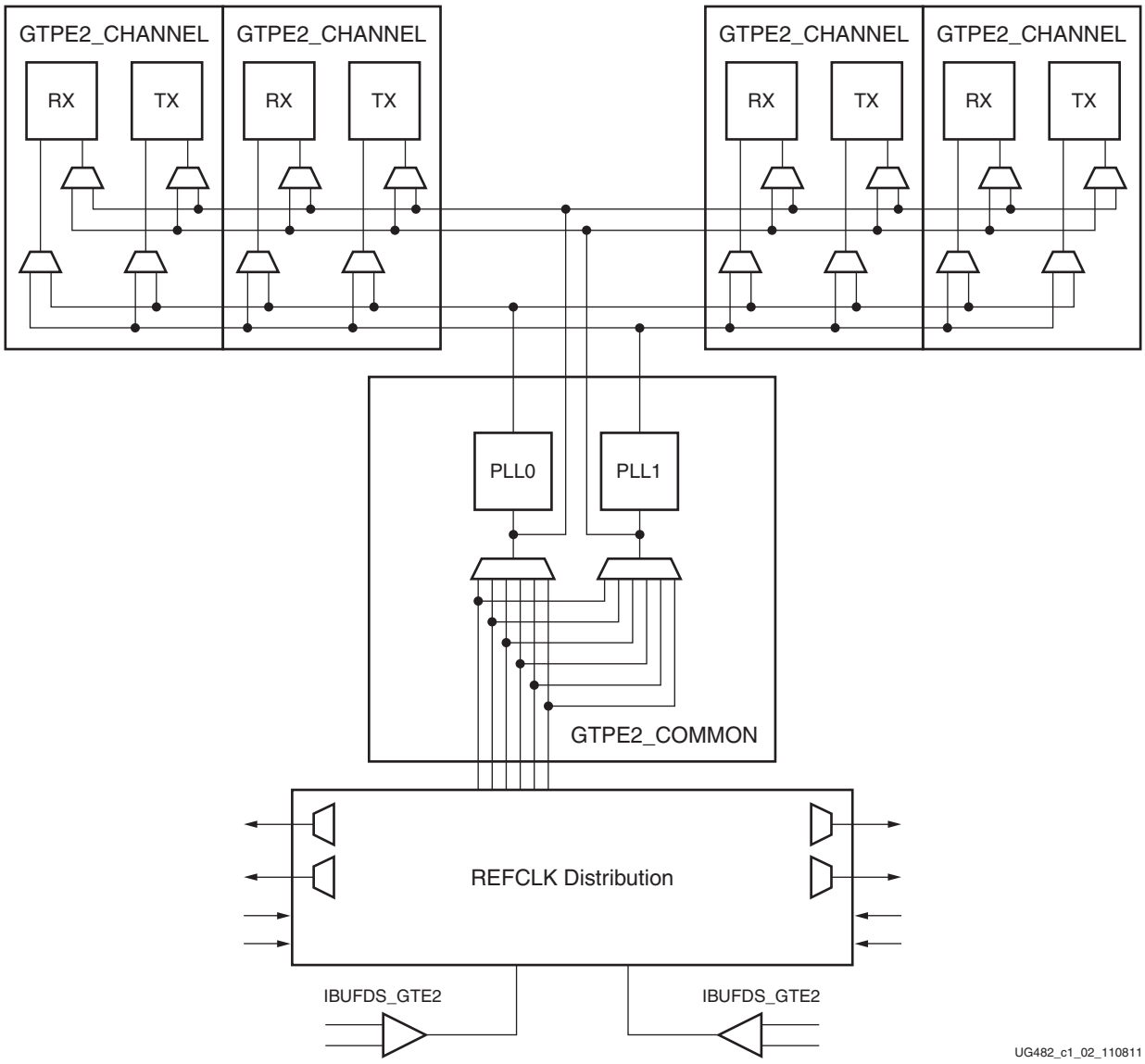


Figure 1-2: **GTP Transceiver Quad Configuration**

Four GTPE2 channels clustered together with one GTPE2_COMMON primitive are called a *Quad* or *Q*.

The GTPE2_COMMON primitive contains two ring oscillator PLLs (PLL0 and PLL1). GTPE2_COMMON must always be instantiated.

Each GTPE2_CHANNEL primitive consists of a transmitter and a receiver.

Figure 1-3 illustrates the topology of a GTPE2_CHANNEL primitive.

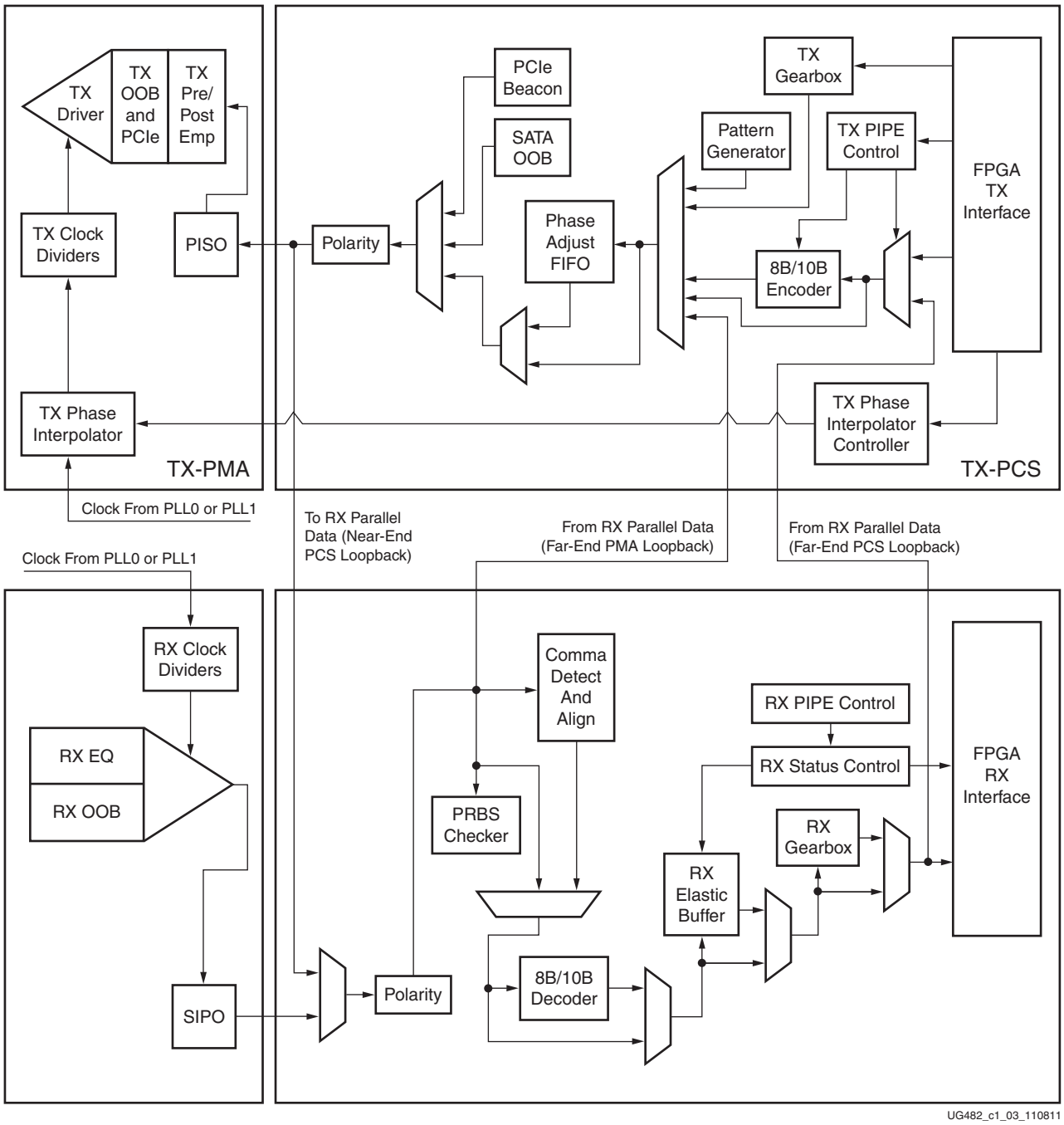


Figure 1-3: **GTPE2_CHANNEL Primitive Topology**

Refer to Figure 2-9, page 35 for the description of the channel clocking architecture, which provides clocks to the RX and TX clock dividers.

7 Series FPGAs Transceivers Wizard

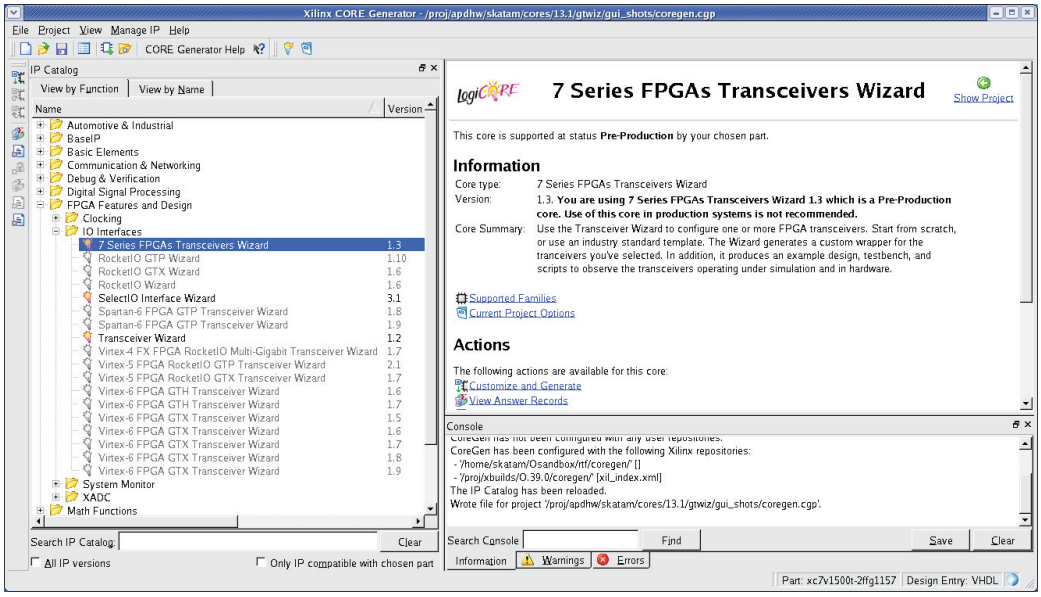
The 7 Series FPGAs Transceivers Wizard (hereinafter called the Wizard) is the preferred tool to generate a wrapper to instantiate GTP transceiver primitives called GTPE2_COMMON and GTPE2_CHANNEL. The Wizard is located in the CORE Generator tool. The user is recommended to download the most up-to-date IP update before using the Wizard. Details on how to use this Wizard can be found in UG769, *LogiCORE IP 7 Series FPGAs Transceivers Wizard User Guide*.

Follow these steps to launch the Wizard:

1. Start the CORE Generator tool.
2. Locate the 7 Series FPGAs Transceivers Wizard in the taxonomy tree under:

   `/FPGA Features & Design/IO Interfaces`

   See Figure 1-4.



UG482_c1_04_110911

Figure 1-4: **7 Series FPGAs Transceivers Wizard**

3. Double-click 7 Series FPGAs Transceivers Wizard to launch the Wizard.

Simulation

Functional Description

Simulations using the GTPE2_CHANNEL and GTPE2_COMMON primitives have specific prerequisites that the simulation environment and the test bench must fulfill. For instructions on how to set up the simulation environment for supported simulators depending on the used hardware description language (HDL), see the latest version of UG626, *Synthesis and Simulation Design Guide*. This design guide can be downloaded from the Xilinx website.

The prerequisites for simulating a design with the GTPE2_CHANNEL and GTPE2_COMMON primitives are:

- A simulator with support for SecureIP models.

  SecureIP models are encrypted versions of the Verilog HDL used for implementation of the modeled block. SecureIP is an IP encryption methodology. To support SecureIP models, a Verilog LRM - IEEE Std 1364-2005 encryption compliant simulator is required.

- A mixed-language simulator for VHDL simulation.

  SecureIP models use a Verilog standard. To use them in a VHDL design, a mixed-language simulator is required. The simulator must be able to simulate VHDL and Verilog simultaneously.

- An installed GTP transceiver SecureIP model.
- The correct setup of the simulator for SecureIP use (initialization file, environment variables).
- The ability to run COMPXLIB, which compiles the simulation libraries (e.g., UNISIM, SIMPRIMS) in the correct order.
- The correct simulator resolution (Verilog).
- The user guide of the simulator and UG626, *Synthesis and Simulation Design Guide* provide a detailed list of settings for SecureIP support.

Ports and Attributes

There are no simulation-only ports on the GTPE2_COMMON and GTPE2_CHANNEL primitives.

GTPE2_COMMON Attributes

The GTPE2_COMMON primitive has attributes intended only for simulation. Table 1-2 lists the simulation-only attributes of the GTPE2_COMMON primitive. The names of these attributes start with *SIM_*.

Table 1-2: **GTPE2_COMMON Simulation-Only Attributes**

Attribute	Type	Description
SIM_PLL0REFCLK_SEL	3-bit Binary	This attribute selects the reference clock source used to drive PLL0 in simulation for designs where PLL0 is always driven by the same reference clock source. SIM_PLL0REFCLK_SEL allows for simulation before and after the port swap changes. This allows for the block to be simulated with the correct clock source both before and after the port swap. SIM_PLL0REFCLK_SEL must be set to the same value as PLL0REFCLK SEL[2:0]. For designs that require the reference clock source to be changed on the fly, the port PLL0REFCLKSEL is used instead to dynamically select the reference clock source.
SIM_PLL1REFCLK_SEL	3-bit Binary	This attribute selects the reference clock source used to drive PLL1 in simulation for designs where PLL1 is always driven by the same reference clock source. SIM_PLL1REFCLK_SEL allows for simulation before and after the port swap changes. This allows for the block to be simulated with the correct clock source both before and after the port swap. SIM_PLL1REFCLK_SEL must be set to the same value as PLL1REFCLK SEL[2:0]. For designs that require the reference clock source to be changed on the fly, the port PLL1REFCLKSEL is used instead to dynamically select the reference clock source.

Table 1-2: **GTPE2_COMMON Simulation-Only Attributes *(Cont'd)***

Attribute	Type	Description
SIM_RESET_SPEEDUP	Boolean	If the SIM_RESET_SPEEDUP attribute is set to TRUE (default), an approximated reset sequence is used to speed up the reset time for simulations, where faster reset times and faster simulation times are desirable. If the SIM_RESET_SPEEDUP attribute is set to FALSE, the model emulates hardware reset behavior in detail.
SIM_VERSION	String	This attribute selects the simulation version to match different steppings of silicon. The default for this attribute is 1.1.

GTPE2_CHANNEL Attributes

The GTPE2_CHANNEL primitive has attributes intended only for simulation. Table 1-3 lists the simulation-only attributes of the GTPE2_CHANNEL primitive. The names of these attributes start with *SIM_*.

Table 1-3: **GTPE2_CHANNEL Simulation-Only Attributes**

Attribute	Type	Description
SIM_RESET_SPEEDUP	Boolean	If the SIM_RESET_SPEEDUP attribute is set to TRUE (default), an approximated reset sequence is used to speed up the reset time for simulations, where faster reset times and faster simulation times are desirable. If the SIM_RESET_SPEEDUP attribute is set to FALSE, the model emulates hardware reset behavior in detail.
SIM_RECEIVER_DETECT_PASS	Boolean	SIM_RECEIVER_DETECT_PASS is a string TRUE/FALSE attribute to determine if a receiver detect operation should indicate a pass or fail in simulation.
SIM_TX_EIDLE_DRIVE_LEVEL	String	SIM_TX_EIDLE_DRIVE_LEVEL can be set to 0, 1, X, or Z to allow for simulation of electrical idle and receiver detect operations using an external pull-up resistor. The default for this attribute is X.
SIM_VERSION	Real	This attribute selects the simulation version to match different steppings of silicon. The default for this attribute is 1.0.

Implementation

Functional Description

This section provides the information needed to map 7 series GTP transceivers instantiated in a design to device resources, including:

- The location of the GTP transceiver Quads on the available device and package combinations.
- The pad numbers of external signals associated with each GTP transceiver Quad.
- How the GTPE2_CHANNEL primitive, the GTPE2_COMMON primitive, and clocking resources instantiated in a design are mapped to available locations with a user constraints file (UCF).

It is a common practice to define the location of GTP transceiver Quads early in the design process to ensure correct usage of clock resources and to facilitate signal integrity analysis during board design. The implementation flow facilitates this practice through the use of location constraints in the UCF.

This section describes how to instantiate GTP transceiver clocking components.

The position of each GTP transceiver channel and common primitive is specified by an XY coordinate system that describes the column number and the relative position within that column.

For a given device/package combination, the transceiver with the coordinates *X0Y0* is always located at the lowest position of the lowest available bank.

There are two ways to create a UCF for designs that utilize the GTP transceiver. The preferred method is to use the 7 Series FPGAs Transceivers Wizard. The Wizard automatically generates UCF templates that configure the transceivers and contain placeholders for GTP transceiver placement information. The UCFs generated by the Wizard can then be edited to customize operating parameters and placement information for the application.

The second approach is to create the UCF by hand. When using this approach, the designer must enter both configuration attributes that control transceiver operation as well as tile location parameters. Care must be taken to ensure that all of the parameters needed to configure the GTP transceiver are correctly entered.

If a design requires the use of any of the GTP channels in a given GTP Quad, a GTPE2_COMMON primitive must be instantiated as shown in Figure 1-5. At a minimum, at least one GTPE2_CHANNEL must also be instantiated. Figure 1-5 shows four GTPE2_CHANNEL primitives instantiated.

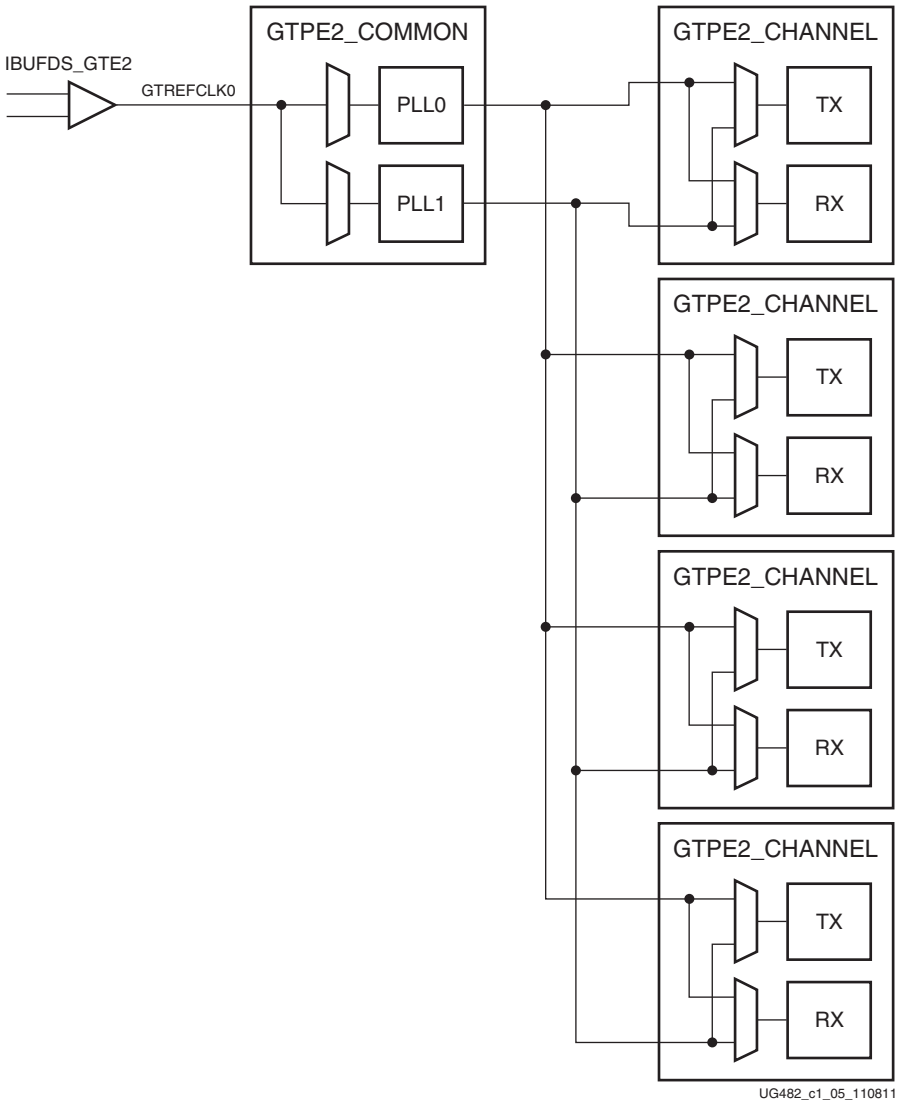


Figure 1-5: **Four Channel Configuration**

Serial Transceiver Channels by Device/Package

See UG475, *7 Series FPGAs Packaging and Pinout Specification*.

Chapter 2

Shared Features

Reference Clock Input Structure

Functional Description

The reference clock input structure is illustrated in Figure 2-1. The input is terminated internally with 50Ω on each leg to 4/5 MGTAVCC. The reference clock is instantiated in software with the IBUFDS_GTE2 software primitive. The ports and attributes controlling the reference clock input are tied to the IBUFDS_GTE2 software primitive.

Figure 2-1 shows the internal structure of the reference clock input buffer.

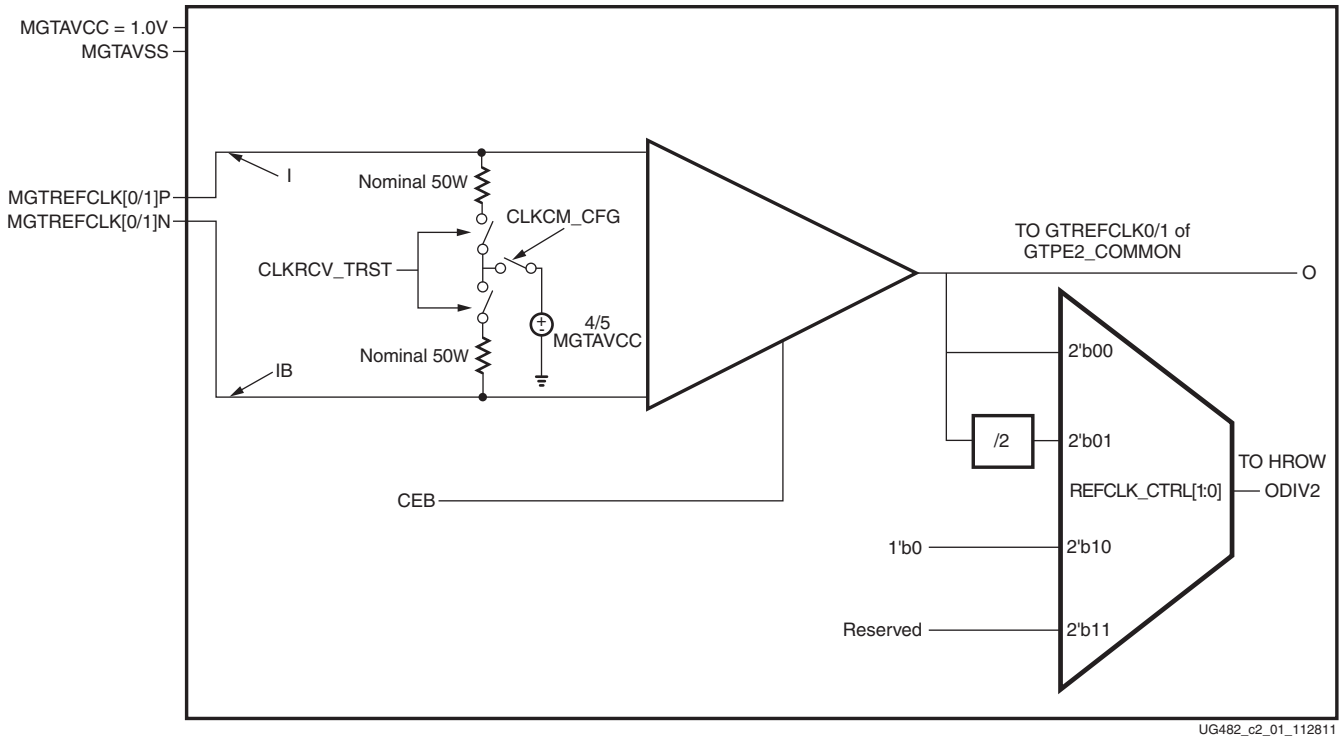


Figure 2-1: **Reference Clock Input Structure**

Ports and Attributes

Table 2-1 defines the reference clock input ports in the IBUFDS_GTE2 software primitive.

Table 2-1: **Reference Clock Input Ports (IBUFDS_GTE2)**

Port	Dir	Clock Domain	Description
I IB	In (pad)	N/A	These are the reference clock input ports that get mapped to GTREFCLK0P/ GTREFCLK0N and GTREFCLK1P/ GTREFCLK1P.
CEB	In	N/A	This is the active-Low asynchronous clock enable signal for the clock buffer. Setting this signal High powers down the clock buffer.
O	Out	N/A	This output drives the GTREFCLK[0/1] signals in the GTPE2_COMMON software primitives. Refer to Reference Clock Selection and Distribution, page 25 for more details.
ODIV2(1)	Out	N/A	This output is a divide-by-2 version of the O signal, which can drive the BUFG* software primitives via Hrow routing. The selection is controlled automatically by the software depending on whether port O or ODIV2 is connected. Refer to Reference Clock Selection and Distribution, page 25 for more details.

Notes:

1. The O and ODIV2 outputs are not phase matched to each other.

Table 2-2 defines the attributes in the IBUFDS_GTE2 software primitive that configure the reference clock input.

Table 2-2: **Reference Clock Input Attributes (IBUFDS_GTE2)**

Attribute	Type	Description
CLK_RCV_TRST	Boolean	Reserved. This attribute switches the 50Ω termination resistors into the signal path. This attribute must always be set to TRUE.
CLKCM_CFG	Boolean	Reserved. This attribute switches in the termination voltage for the 50Ω termination. This attribute must always be set to TRUE.
CLKSWING_CFG	2-bit Binary	Reserved. This attribute controls the internal swing of the clock. This attribute must always be set to `2'b11`.

Use Modes: Reference Clock Termination

The reference clock input is to be externally AC coupled. Table 2-3 shows the port and attribute settings required to achieve this.

Table 2-3: **Port and Attribute Settings**

Input Type	Settings
Ports	CEB = 0
Attributes	CLKRCV_TRST = TRUE CLKCM_CFG = TRUE CLKSWING_CFG = `2'b11`

Reference Clock Selection and Distribution

Functional Description

The GTP transceivers in 7 series FPGAs provide different reference clock input options. Clock selection and availability differs slightly from 7 series GTX transceiver in that reference clock routing is east and west bound rather than north and south bound.

Architecturally, the concept of a Quad (or Q), contains a grouping of four GTPE2_CHANNEL primitives, one GTPE2_COMMON primitive, two dedicated external reference clock pin pairs, and dedicated reference clock routing. The GTPE2_COMMON primitive must always be instantiated, and the GTPE2_CHANNEL primitive must be instantiated for each transceiver. For the larger Artix™-7 devices that contain 16 GTP transceivers, the reference clock supplied to the PLLs in a given Quad can also be sourced from the adjacent Quad in the same half of the device. A Quad located in the top half of the device can share its two local reference clocks with the other Quad located in the top half. Similarly, a Quad located in the bottom half of the device can share its two reference clocks with the other Quad located in the bottom half.

Reference clock features include:

- Clock routing for east and west bound clocks.
- Flexible reference clock inputs available for PLL0 and PLL1.
- Static or dynamic selection of the reference clock for PLL0 and PLL1.

Figure 2-2 shows the reference clock architecture with the GTPE2_COMMON primitive, two dedicated reference clock pin pairs, and dedicated east or west reference clock routing. Each GTPE2_COMMON in a Quad has four clock inputs available:

- Two local reference clock pin pairs, GTREFCLK0 or GTREFCLK1
- Two reference clock pin pairs from the other Quad situated in the same half of the device

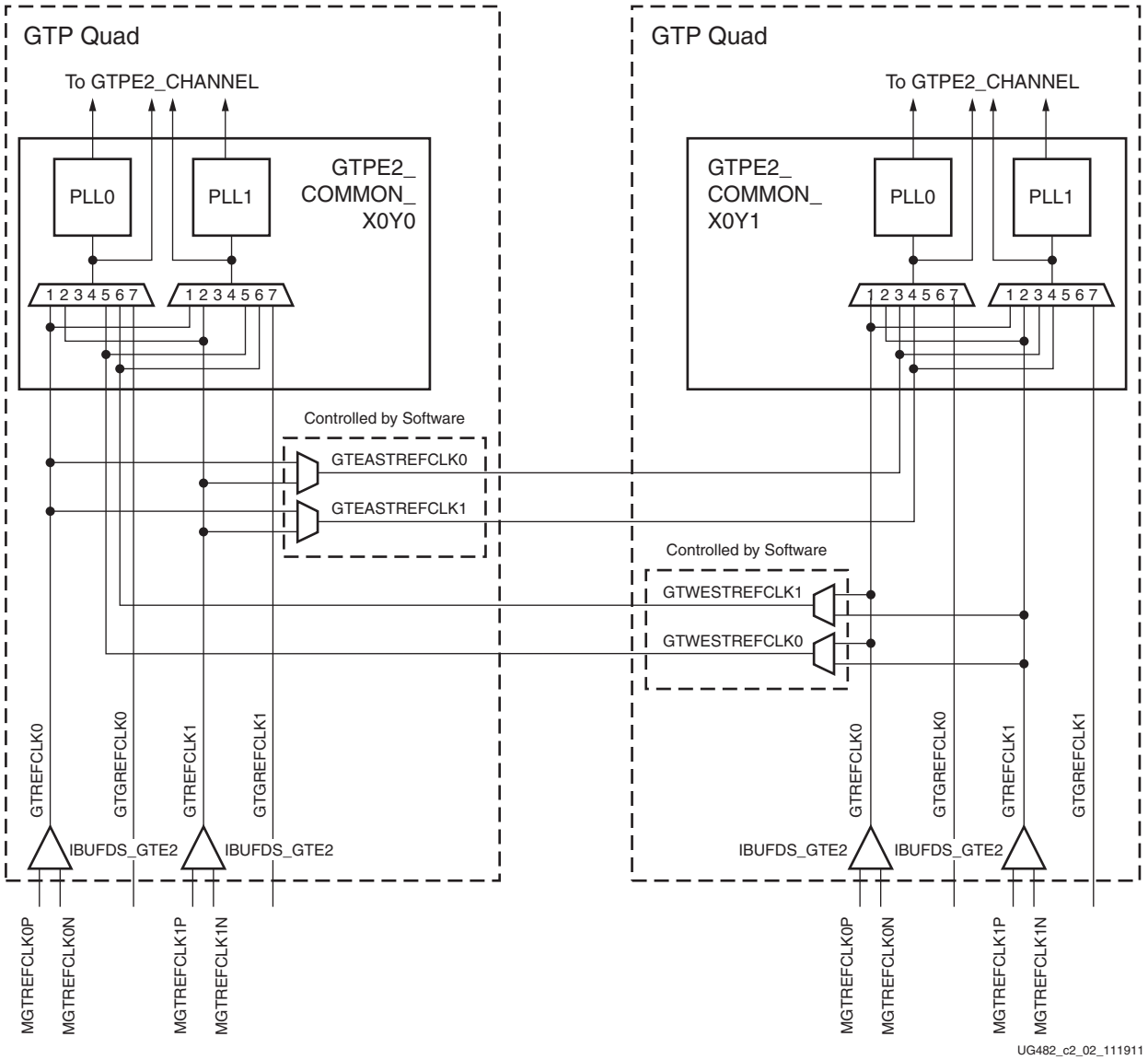


Figure 2-2: **Conceptual View of GTP Transceiver Reference Clocking**

Figure 2-3 shows the shows the detailed view of the reference clock multiplexer structures within a single GTPE2_COMMON primitive. The PLL0REFCLKSEL and PLL1REFCLKSEL ports are required when multiple reference clock sources are connected to the multiplexers. A single reference clock is most commonly used. In this case, the PLL[0/1]REFCLKSEL port can be tied to `3'b001`, and the Xilinx software tools handle the complexity of the multiplexers and associated routing. See External Reference Clock Use Model, page 31 for more information.

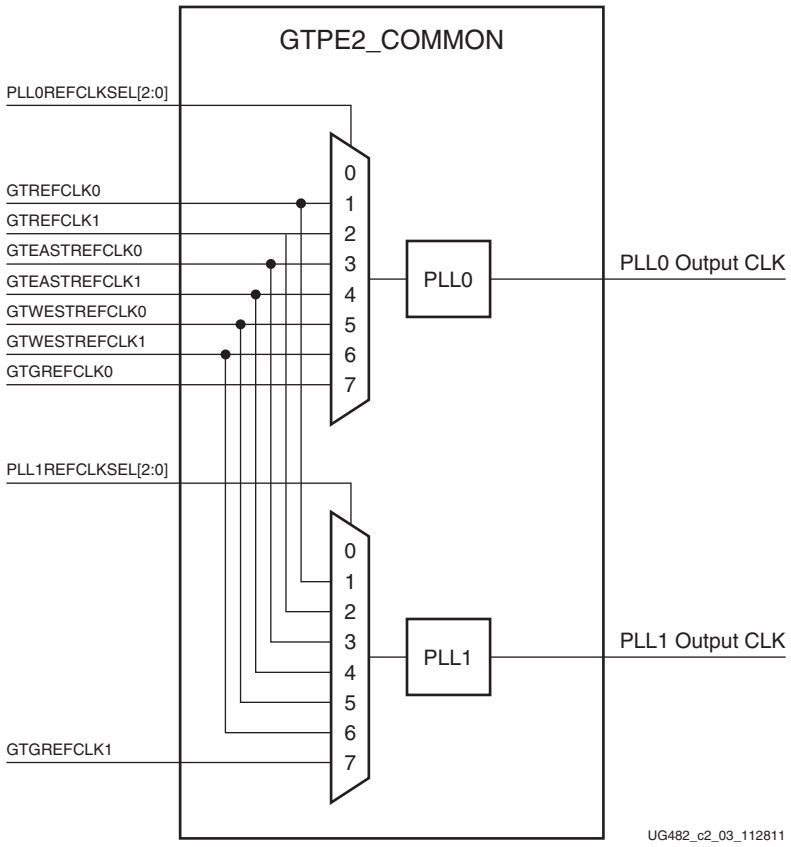


Figure 2-3: **PLL0 and PLL1 Reference Clock Selection Multiplexer**

Ports and Attributes

Table 2-4 and Table 2-5 define the clocking ports and attributes for the GTPE2_COMMON primitive.

Table 2-4: **GTPE2_COMMON Clocking Ports**

Port	Direction	Clock Domain	Description
GTGREFCLK0	In	Clock	Reference clock generated by the internal FPGA logic. This input is reserved for internal testing purposes only.
GTGREFCLK1	In	Clock	Reference clock generated by the internal FPGA logic. This input is reserved for internal testing purposes only.
GTREFCLK0	In	Clock	External clock driven by IBUFDS_GTE2 for PLL0 and/or PLL1.
GTREFCLK1	In	Clock	External clock driven by IBUFDS_GTE2 for PLL0 and/or PLL1.
GTWESTREFCLK0	In	Clock	West-bound clock from the Quad on the right side of the device.

Table 2-4: **GTPE2_COMMON Clocking Ports *(Cont'd)***

Port	Direction	Clock Domain	Description
GTWESTREFCLK1	In	Clock	West-bound clock from the Quad on the right side of the device.
GTEASTREFCLK0	In	Clock	East-bound clock from the Quad on the left side of the device.
GTEASTREFCLK1	In	Clock	East-bound clock from the Quad on the left side of the device.
PLL0OUTCLK	Out	Clock	PLL0 clock output. The user must connect this port to the PLL0CLK port on the GTPE2_CHANNEL primitive.
PLL1OUTCLK	Out	Clock	PLL1 clock output. The user must connect this port to the PLL1CLK port on the GTPE2_CHANNEL primitive.
PLL0OUTREFCLK	Out	Clock	The user must connect this port to the PLL0REFCLK port on the GTPE2_CHANNEL primitive.
PLL1OUTREFCLK	Out	Clock	The user must connect this port to the PLL1REFCLK port on the GTPE2_CHANNEL primitive.

Table 2-4: **GTPE2_COMMON Clocking Ports *(Cont'd)***

Port	Direction	Clock Domain	Description
PLL0REFCLKSEL[2:0]	In	Async	Input to dynamically select the input reference clock to PLL0. This input should be set to `3'b001` when only one clock source is connected to the PLL0 reference clock selection multiplexer. Reset must be applied to PLL0 after changing the reference clock input. `000`: Reserved `001`: GTREFCLK0 selected `010`: GTREFCLK1 selected `011`: GTEASTREFCLK0 selected `100`: GTEASTREFCLK1 selected `101`: GTWESTREFCLK0 selected `110`: GTWESTREFCLK1 selected `111`: GTGREFCLK0 selected
PLL1REFCLKSEL[2:0]	In	Async	Input to dynamically select the input reference clock to PLL1. This input should be set to `3'b001` when only one clock source is connected to the PLL1 reference clock selection multiplexer. Reset must be applied to PLL1 after changing the reference clock input. `000`: Reserved `001`: GTREFCLK0 selected `010`: GTREFCLK1 selected `011`: GTEASTREFCLK0 selected `100`: GTEASTREFCLK1 selected `101`: GTWESTREFCLK0 selected `110`: GTWESTREFCLK1 selected `111`: GTGREFCLK1 selected

Table 2-5: **GTPE2_COMMON Attributes**

Attribute	Type	Description
SIM_PLL0REFCLK_SEL	3-bit Binary	This attribute selects the reference clock source used to drive PLL0 in simulation for designs where PLL0 is always driven by the same reference clock source. SIM_PLL0REFCLK_SEL allows for simulation before and after the port swap changes. This allows for the block to be simulated with the correct clock source both before and after the port swap. SIM_PLL0REFCLK_SEL must be set to the same value as PLL0REFCLK SEL[2:0]. For designs that require the reference clock source to be changed on the fly, the port PLL0REFCLKSEL is used instead to dynamically select the reference clock source.
SIM_PLL1REFCLK_SEL	3-bit Binary	This attribute selects the reference clock source used to drive PLL1 in simulation for designs where PLL1 is always driven by the same reference clock source. SIM_PLL1REFCLK_SEL allows for simulation before and after the port swap changes. This allows for the block to be simulated with the correct clock source both before and after the port swap. SIM_PLL1REFCLK_SEL must be set to the same value as PLL1REFCLK SEL[2:0]. For designs that require the reference clock source to be changed on the fly, the port PLL1REFCLKSEL is used instead to dynamically select the reference clock source.

Table 2-6 defines the clocking ports for the GTPE2_CHANNEL primitive.

Table 2-6: **GTPE2_CHANNEL Clocking Ports**

Port	Direction	Clock Domain	Description
RXSYSCLKSEL	In	Async	Selects the PLL clock source to drive the RX datapath: RXSYSCLKSEL[0] = `1'b0` (PLL0) RXSYSCLKSEL[0] = `1'b1` (PLL1) Selects the reference clock source to drive RXOUTCLK: RXSYSCLKSEL[1] = `1'b0` (reference clock from PLL0) RXSYSCLKSEL[1] = `1'b1` (reference clock from PLL1)
TXSYSCLKSEL	In	Async	Selects the PLL clock source to drive the TX datapath: TXSYSCLKSEL[0] = `1'b0` (PLL0) TXSYSCLKSEL[0] = `1'b1` (PLL1) Selects the reference clock source to drive TXOUTCLK: TXSYSCLKSEL[1] = `1'b0` (reference clock from PLL0) TXSYSCLKSEL[1] = `1'b1` (reference clock form PLL1)
PLL0CLK	In	Clock	PLL0 clock input. The user must connect this port to the PLL0OUTCLK port on the GTPE2_COMMON primitive.
PLL1CLK	In	Clock	PLL1 clock input. The user must connect this port to the PLL1OUTCLK port on the GTPE2_COMMON primitive.
PLL0REFCLK	In	Clock	The user must connect this port to the PLL0OUTREFCLK port on the GTPE2_COMMON primitive.
PLL1REFCLK	In	Clock	The user must connect this port to the PLL1OUTREFCLK port on the GTPE2_COMMON primitive.

External Reference Clock Use Model

Each Quad has two dedicated differential reference clock inputs that can be connected to external reference clock sources. An IBUFDS_GTE2 primitive must be instantiated to use these dedicated reference clock pin pairs. The user design connects the IBUFDS_GTE2 output (O) to the GTREFCLK[0/1], GTEASTREFCLK[0/1] or GTWESTREFCLK[0/1] ports of the GTPE2_COMMON primitive where the reference clock selection multiplexer is located. Depending on the line rate requirement, the user design has the flexibility to use different combinations of PLL0 or PLL1 to drive the TX and/or RX datapath, as shown in Figure 2-4.

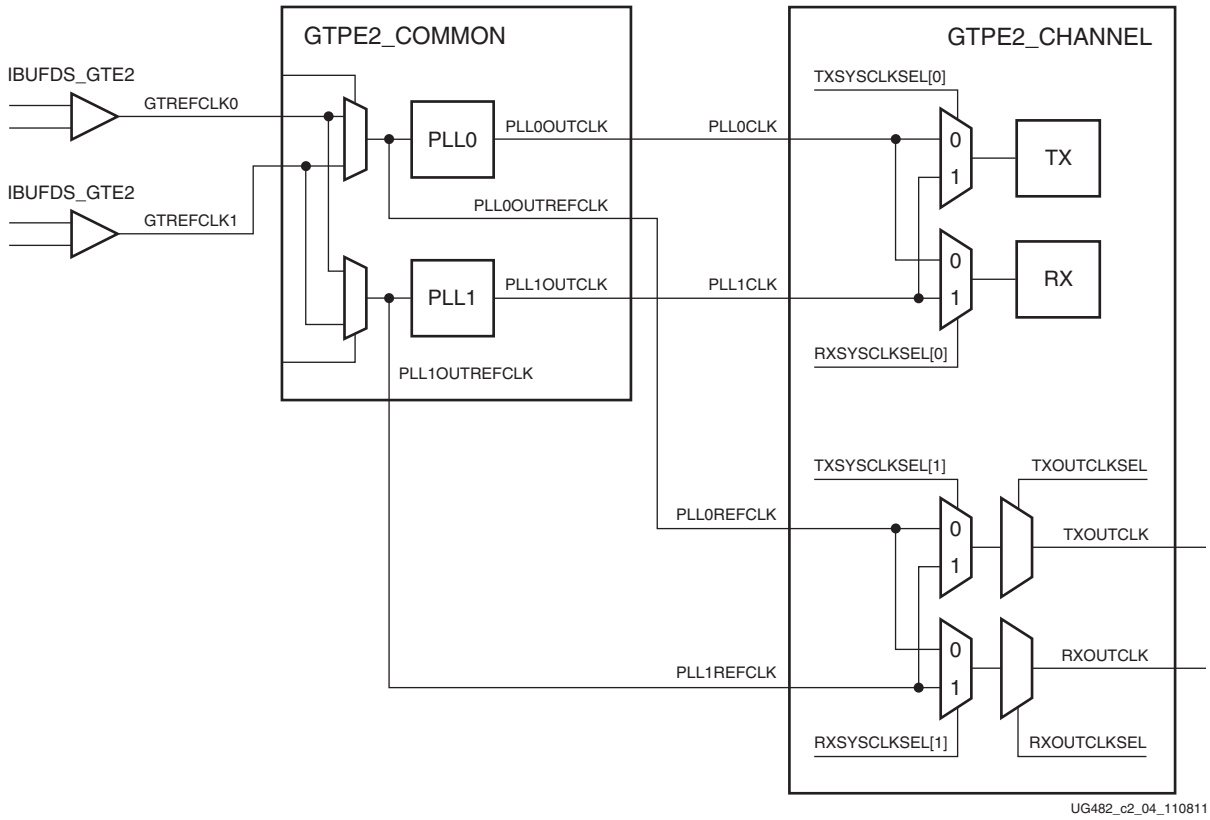


Figure 2-4: **External Reference Clock Use Case**

Single External Reference Clock Use Model

In the single external reference clock use model, the user connects the IBUFDS_GTE2 output (O) to the GTREFCLK0 input port of the GTPE2_COMMON primitive. The user design can leave the other unused reference clock ports floating. The IBUFDS_GTE2 input pins can be constrained in the user constraints file (UCF). Figure 2-5 shows a single GTPE2_COMMON primitive connected to a single IBUFDS_GTE2 primitive.

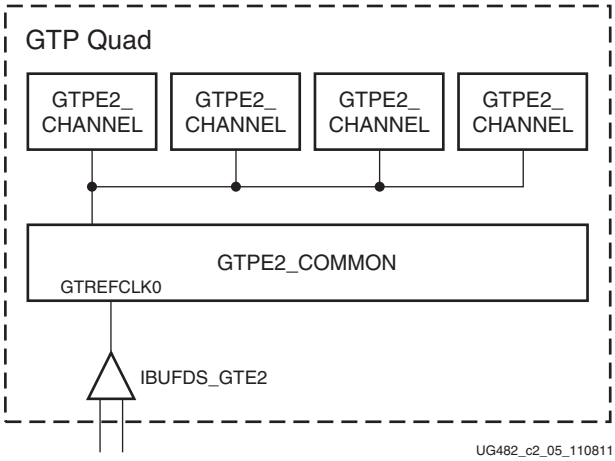


Figure 2-5: **Single GTP Quad with a Single Local Reference Clock**

Figure 2-6 shows a single reference clock connected to two GTP Quads. The user connects the IBUFDS_GTE2 output (O) to the GTREFCLK0 input port of both GTPE2_COMMON primitive instances. Such a scenario is only possible in the larger Artix-7 devices that contain east and west GTP Quads adjacent to each other.

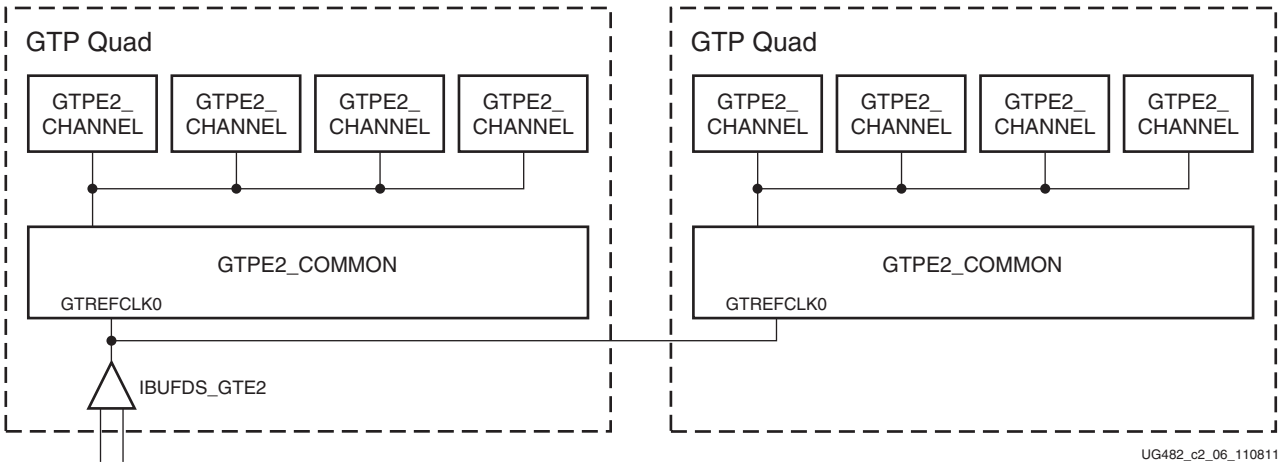


Figure 2-6: **Two GTP Quads with a Single Shared Reference Clock**

When required, as is the case for the design in Figure 2-6, the Xilinx implementation tools make the necessary adjustments to the east/west routing shown in Figure 2-2, page 26, as well as any necessary pin swapping to the GTPE2_COMMON clock inputs to route the reference clocks between two Quads.

Multiple External Reference Clock Use Model

In Figure 2-7 and Figure 2-9, because the reference clock multiplexer structures in the GTPE2_COMMON have more than one reference clock source, the user design is required to connect the output of the IBUFDS_GTE2 to the correct clock input ports on the GTPE2_COMMON primitive. Figure 2-7 shows an example of a single GTP Quad using both of its dedicated differential reference clock inputs. Two IBUFDS_GTE2 primitives and a single GTPE2_COMMON primitive are instantiated.

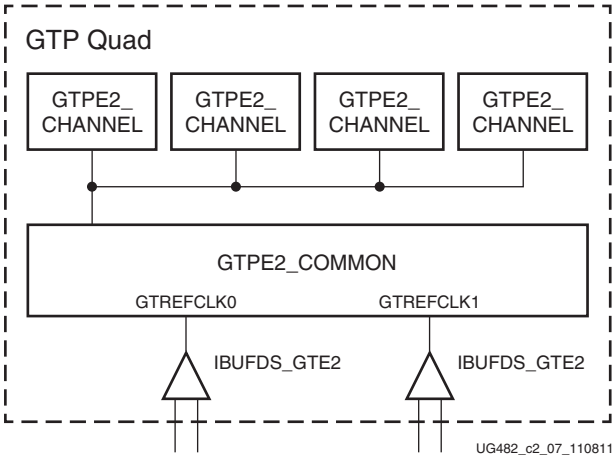


Figure 2-7: **Single GTP Quad using Multiple Local Reference Clocks**

Figure 2-8 shows two GTP Quads, each utilizing their own dedicated differential reference clock inputs as well as the dedicated differential reference clock inputs of their neighboring GTP Quad. Such a scenario is only possible in the larger Artix-7 devices that contain east and west GTP Quads adjacent to each other. The user is responsible for properly connecting the output of the IBUFDS_GTE2 to the appropriate GTREFCLK[0/1], GTWESTREFCLK[0/1], and GTEASTREFCLK[0/1] input ports on the GTPE2_COMMON primitive.

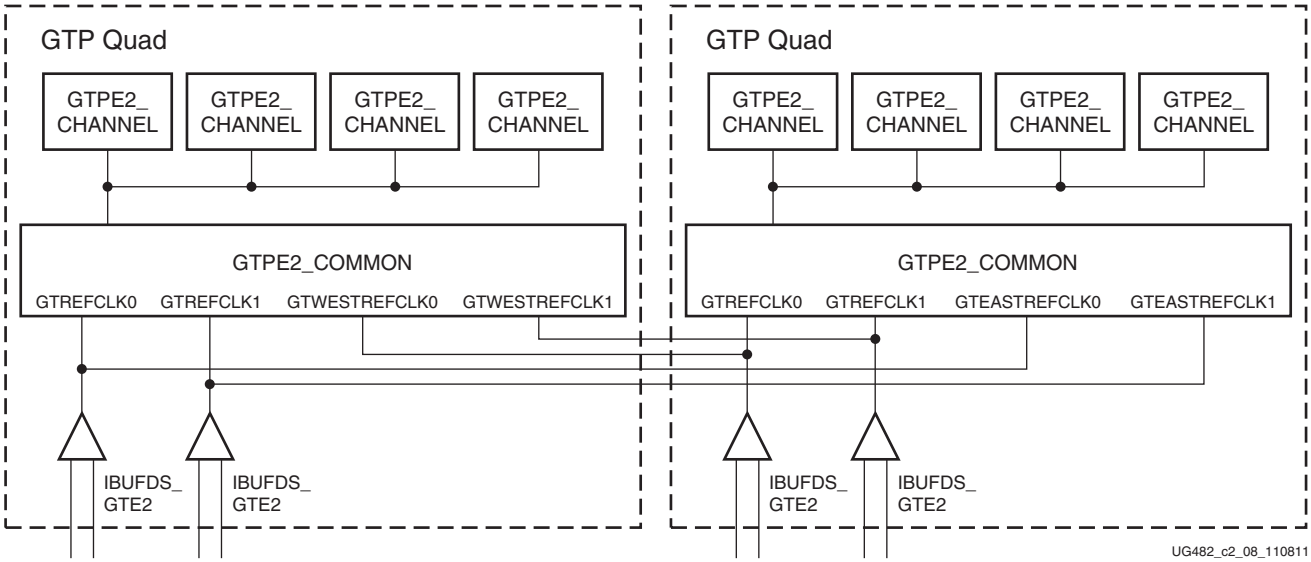


Figure 2-8: **Two GTP Quads using Multiple Reference Clocks from Different Quads**

For multi-rate designs that require the reference clock to be changed in real time, the PLL0REFCLKSEL and PLL1REFCLKSEL ports are used to dynamically select the reference clock source. When the selection has been made, the user design is responsible for resetting the PLL via PLL0RESET or PLL1RESET.

PLL

Functional Description

The GTP Quad contains two ring oscillator PLLs (PLL0 and PLL1). The internal clocking architecture is shown in Figure 2-9. When the TX and RX datapaths operate in the same line rate range, PLL0 or PLL1 can be shared between the TX and RX datapaths. The TX and RX clock dividers can individually select the clock from PLL0 or PLL1 to allow the TX and RX datapaths to operate at asynchronous frequencies using different reference clock inputs.

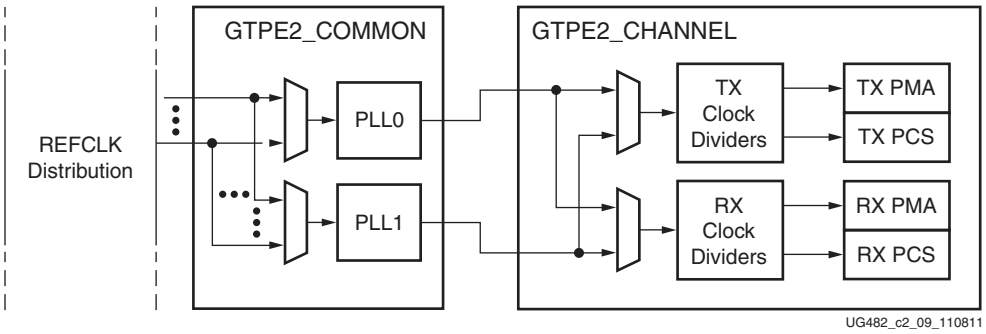


Figure 2-9: **Internal Clocking Architecture**

The PLL input clock selection is described in Reference Clock Selection and Distribution, page 25. The PLL outputs feed the TX and RX clock divider blocks, which control the generation of serial and parallel clocks used by the PMA and PCS blocks.

Figure 2-10 illustrates a conceptual view of the PLL architecture. The input clock can be divided by a factor of M before feeding into the phase frequency detector. The feedback divider N determines the VCO multiplication ratio and the PLL output frequency. A lock indicator block compares the frequencies of the reference clock and the VCO feedback clock to determine if a frequency lock has been achieved.

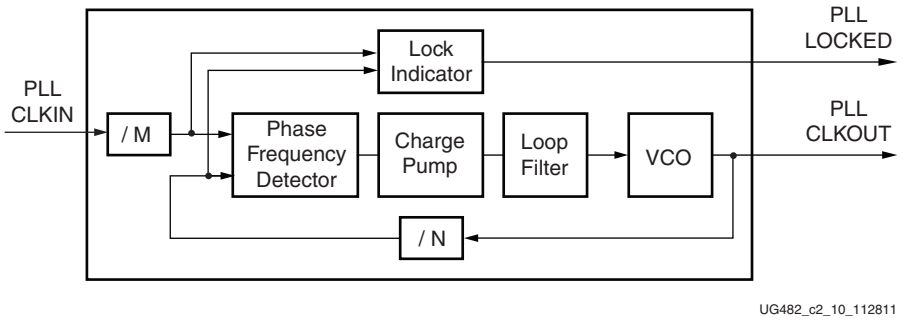


Figure 2-10: **PLL Block Diagram**

The PLL has a nominal operating range between 1.6 GHz to 3.3 GHz. The 7 Series FPGAs Transceivers Wizard chooses the appropriate PLL settings based on application requirements.

Equation 2-1 shows how to determine the PLL output frequency (GHz).

$$f_{PLLClkout} = f_{PLLClkin} \times \frac{N}{M} \qquad \textit{Equation 2-1}$$

Equation 2-2 shows how to determine the line rate (Gb/s). D represents the value of the TX or RX clock divider block in the channel.

$$f_{LineRate} = \frac{f_{PLLClkout} \times 2}{D} \qquad \textit{Equation 2-2}$$

Table 2-7 lists the allowable divider settings.

Table 2-7: **PLL Divider Settings**

Factor	Attribute	Valid Settings
M	PLL0_REFCLK_DIV PLL1_REFCLK_DIV	1, 2
N	PLL0_FBDIV PLL1_FBDIV	1, 2, 3, 4, 5, 6, 8, 10, 12, 16, 20
D	RXOUT_DIV TXOUT_DIV	1, 2, 4, 8

Ports and Attributes

Table 2-8 and Table 2-9 defines the ports and attributes for the PLL.

Table 2-8: **PLL Ports**

Port	Direction	Clock Domain	Description
PLL0LOCKDETCLK PLL1LOCKDETCLK	In	Clock	Stable reference clock for the detection of the feedback and reference clock signals to the PLL. The input reference clock to the PLL or any output clock generated from the PLL (e.g., TXOUTCLK) must not be used to drive this clock. This clock is required only when using the PLL[0/1]FBCLKLOST and PLL[0/1]REFCLKLOST ports. It does not affect the PLL lock detection, reset, and power-down functions.
PLL0LOCKEN PLL1LOCKEN	In	Async	This port enables the PLL lock detector. It must always be tied High.
PLL0PD PLL1PD	In	Async	Active-High signal that powers down the PLL for power savings.

Table 2-8: **PLL Ports *(Cont'd)***

Port	Direction	Clock Domain	Description
PLL0REFCLKSEL PLL1REFCLKSEL	In	Async	Input to dynamically select the input reference clock to the PLL. This input should be set to `3'b001` when only one clock source is connected to the PLL reference clock selection multiplexer. Reset must be applied to the PLL after changing the reference clock input. `000`: Reserved `001`: GTREFCLK0 selected `010`: GTREFCLK1 selected `011`: GTEASTREFCLK0 selected `100`: GTEASTREFCLK1 selected `101`: GTWESTREFCLK0 selected `110`: GTWESTREFCLK1 selected `111`: GTGREFCLK0 (PLL0) or GTGREFCLK1 (PLL1) selected
PLL0RESET PLL1RESET	In	Async	This active-High port resets the dividers inside the PLL as well as the PLL lock indicator and status block.
PLL0FBCLKLOST PLL1FBCLKLOST	Out	PLL0LOCKDETCLK PLL1LOCKDETCLK	A High on this signal indicates the feedback clock from the PLL feedback divider to the phase frequency detector of the PLL is lost.
PLL0LOCK PLL1LOCK	Out	Async	This active-High PLL frequency lock signal indicates that the PLL frequency is within predetermined tolerance. The transceiver and its clock outputs are not reliable until this condition is met.
PLL0REFCLKLOST PLL1REFCLKLOST	Out	PLL0LOCKDETCLK PLL1LOCKDETCLK	A High on this signal indicates the reference clock to the phase frequency detector of the PLL is lost.

Table 2-9: **PLL Attributes**

Attribute	Type	Description
PLL0_CFG PLL1_CFG	27-bit Hex	Reserved. Configuration setting for the PLL. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
PLL0_FBDIV PLL1_FBDIV	Integer	PLL feedback divider setting as shown in Figure 2-10, page 35. Valid settings are 1, 2, 3, 4, 5, 6, 8, 10, 12, 16, and 20.
PLL0_LOCK_CFG PLL1_LOCK_CFG	9-bit Hex	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
PLL0_REFCLK_DIV PLL1_REFCLK_DIV	Integer	PLL reference clock divider M settings as shown in Figure 2-10, page 35. Valid settings are 1 and 2.
PLL0_INIT_CFG PLL1_INIT_CFG	24-bit Hex	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
PLL0_DMON_CFG PLL1_DMON_CFG	1-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

Loopback

Functional Description

Loopback modes are specialized configurations of the transceiver datapath where the traffic stream is folded back to the source. Typically, a specific traffic pattern is transmitted and then compared to check for errors. Figure 2-11 illustrates a loopback test configuration with four different loopback modes.

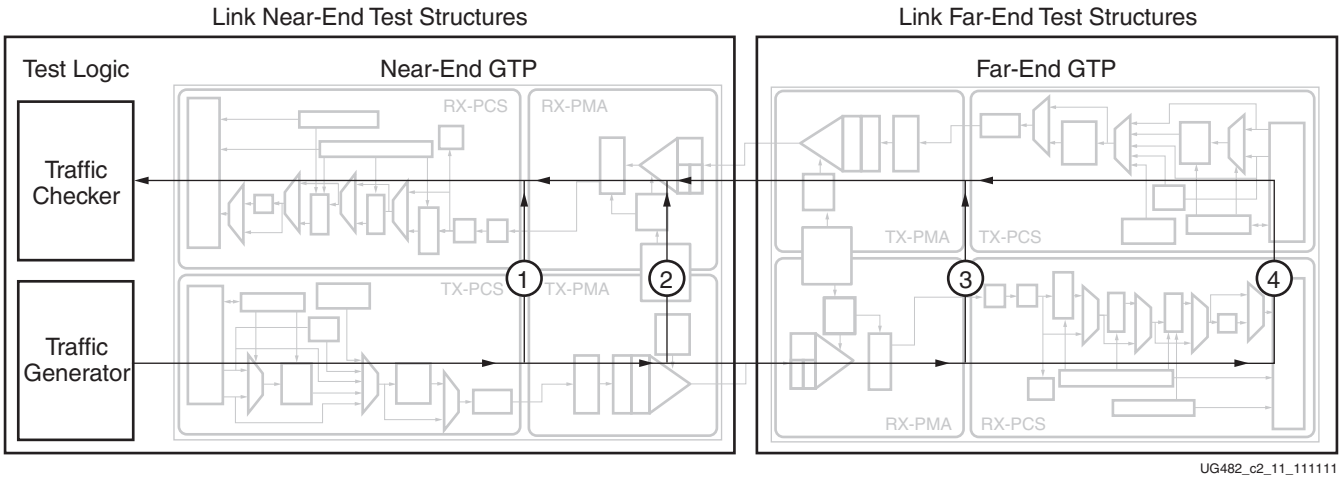


Figure 2-11: **Loopback Testing Overview**

Loopback test modes fall into two broad categories:

- Near-end loopback modes loop transmit data back in the transceiver closest to the traffic generator.
- Far-end loopback modes loop received data back in the transceiver at the far end of the link.

Loopback testing can be used either during development or in deployed equipment for fault isolation. The traffic patterns used can be either application traffic patterns or specialized pseudo-random bit sequences. Each GTP transceiver has a built-in PRBS generator and checker.

Each GTP transceiver features several loopback modes to facilitate testing:

- Near-End PCS Loopback (path 1 in Figure 2-11)
- Near-End PMA Loopback (path 2 in Figure 2-11)
- Far-End PMA Loopback (path 3 in Figure 2-11)

  In Far-End PMA loopback, the transmitter internally uses the RX recovered clock to transmit data.
- Far-End PCS Loopback (path 4 in Figure 2-11)

  If clock correction is not used, a transceiver in Far-end PCS loopback must use the same reference clock used by the transceiver that is the source of the loopback data. Regardless of whether clock correction is used or not, the ports TXUSRCLK and RXUSRCLK must be driven by the same clocking resource (BUFG, BUFH, BUFR).

Ports and Attributes

Table 2-10 and Table 2-11 define the loopback ports and attributes, respectively.

Table 2-10: **Loopback Ports**

Port	Dir	Clock Domain	Description
LOOPBACK[2:0]	In	Async	`000`: Normal operation `001`: Near-End PCS Loopback `010`: Near-End PMA Loopback `011`: Reserved `100`: Far-End PMA Loopback `101`: Reserved `110`: Far-End PCS Loopback

Table 2-11: **Loopback Attributes**

Attribute	Type	Description
LOOPBACK_CFG	1-bit Binary	Reserved.
PMA_LOOPBACK_CFG	1-bit Binary	For use with Far-End PMA loopback. When this attribute is set to `1'b1`, the loopback path includes the continuous time linear equalizer (CTLE). When set to `1'b0`, the loopback path occurs before the CTLE.

Dynamic Reconfiguration Port

Functional Description

The dynamic reconfiguration port (DRP) allows the dynamic change of parameters of the GTPE2_CHANNEL and GTPE2_COMMON primitives. The DRP interface is a processor-friendly synchronous interface with an address bus (DRPADDR) and separated data buses for reading (DRPDO) and writing (DRPDI) configuration data to the primitives. An enable signal (DRPEN), a read/write signal (DRPWE), and a ready/valid signal (DRPRDY) are the control signals that implement read and write operations, indicate operation completion, or indicate the availability of data.

Ports and Attributes

Table 2-12 shows the DRP related ports for GTPE2_CHANNEL.

Table 2-12: **DRP Ports of GTPE2_CHANNEL**

Port	Dir	Clock Domain	Description
DRPADDR[8:0]	In	DRPCLK	DRP address bus.
DRPCLK	In	N/A	DRP interface clock.
DRPEN	In	DRPCLK	DRP enable signal. 0: No read or write operation performed. 1: Enables a read or write operation.
DRPDI[15:0]	In	DRPCLK	Data bus for writing configuration data from the FPGA logic resources to the transceiver.
DRPRDY	Out	DRPCLK	Indicates operation is complete for write operations and data is valid for read operations.
DRPDO[15:0]	Out	DRPCLK	Data bus for reading configuration data from the GTP transceiver to the FPGA logic resources.
DRPWE	In	DRPCLK	DRP write enable. 0: Read operation when DEN is 1. 1: Write operation when DEN is 1.

Table 2-13 shows the DRP related ports for GTPE2_COMMON.

Table 2-13: **DRP Ports of GTPE2_COMMON**

Port	Dir	Clock Domain	Description
DRPADDR[7:0]	In	DRPCLK	DRP address bus.
DRPCLK	In	N/A	DRP interface clock.
DRPEN	In	DRPCLK	DRP enable signal. 0: No read or write operation performed. 1: Enables a read or write operation.
DRPDI[15:0]	In	DRPCLK	Data bus for writing configuration data from the FPGA logic resources to the transceiver.

Table 2-13: **DRP Ports of GTPE2_COMMON *(Cont'd)***

Port	Dir	Clock Domain	Description
DRPRDY	Out	DRPCLK	Indicates operation is complete for write operations and data is valid for read operations.
DRPDO[15:0]	Out	DRPCLK	Data bus for reading configuration data from the GTP transceiver to the FPGA logic resources.
DRPWE	In	DRPCLK	DRP write enable. 0: Read operation when DEN is 1. 1: Write operation when DEN is 1.

Usage Model

Write Operation

Figure 2-12 shows the DRP write operation timing. New DRP operation can be initiated when DRPRDY is asserted.

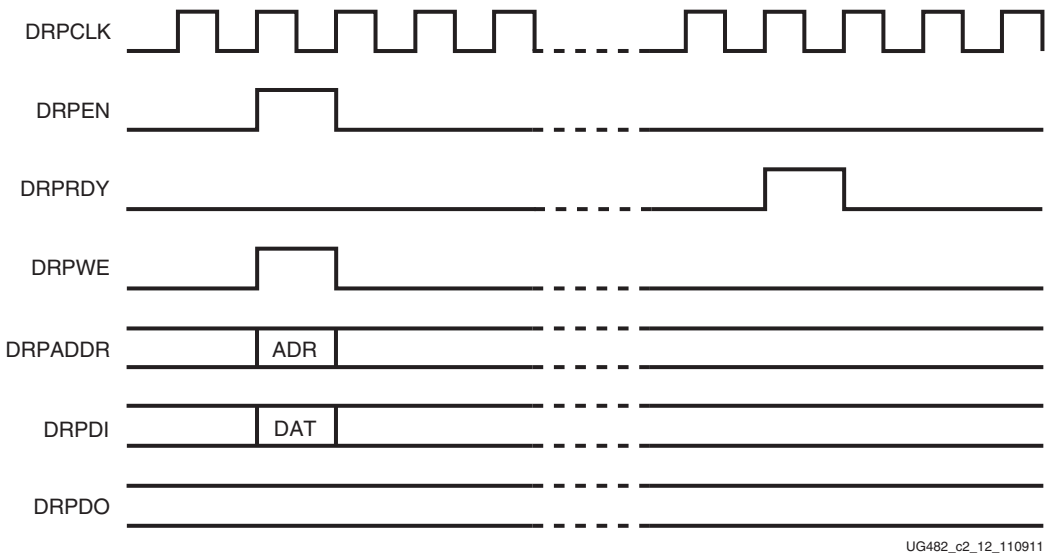


Figure 2-12: **DRP Write Timing**

Read Operation

Figure 2-13 shows the DRP read operation timing. New DRP operation can be initiated when DRPRDY is asserted.

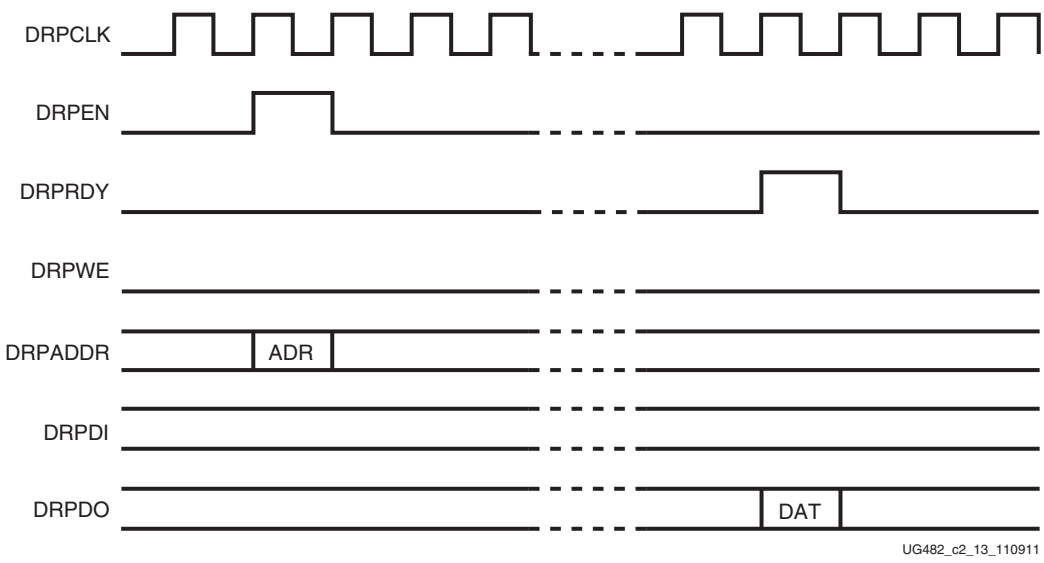


Figure 2-13: **DRP Read Timing**

Chapter 3

Transmitter

TX Overview

Functional Description

This chapter shows how to configure and use each of the functional blocks inside the transmitter (TX). Each transceiver includes an independent transmitter, which consists of a PCS and a PMA. Figure 3-1 shows the functional blocks of the transmitter. Parallel data flows from the FPGA logic into the FPGA TX interface, through the PCS and PMA, and then out the TX driver as high-speed serial data.

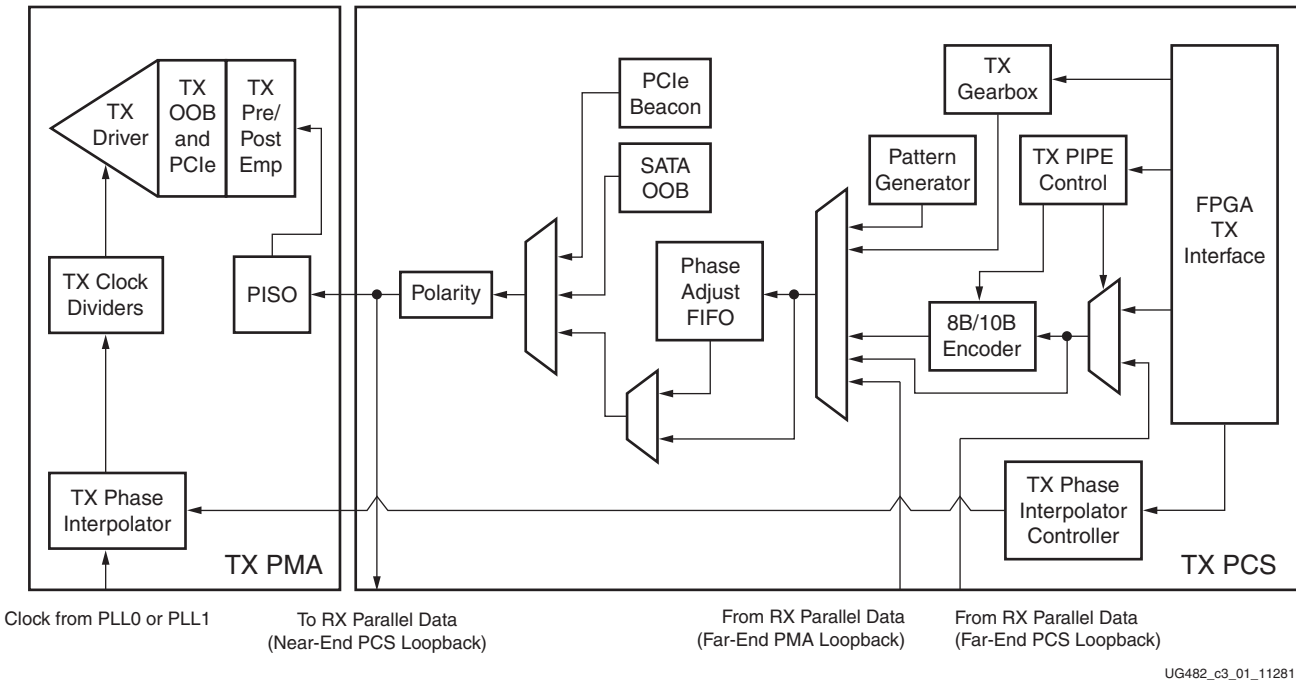


Figure 3-1: **GTP Transceiver TX Block Diagram**

The key elements within the GTP transceiver TX are:

1. FPGA TX Interface, page 44
2. TX 8B/10B Encoder, page 51
3. TX Gearbox, page 54
4. TX Buffer, page 63

5. TX Buffer Bypass, page 65
6. TX Pattern Generator, page 69
7. TX Polarity Control, page 73
8. TX Fabric Clock Output Control, page 74
9. TX Configurable Driver, page 80
10. TX Receiver Detect Support for PCI Express Designs, page 87
11. TX Out-of-Band Signaling, page 90

FPGA TX Interface

Functional Description

The FPGA TX interface is the FPGA's gateway to the TX datapath of the GTP transceiver. Applications transmit data through the GTP transceiver by writing data to the TXDATA port on the positive edge of TXUSRCLK2. The width of the port can be configured to be two or four bytes wide. The actual width of the port depends on the TX_DATA_WIDTH attribute and TX8B10BEN port setting. Port widths can be 16, 20, 32, and 40 bits. The rate of the parallel clock (TXUSRCLK2) at the interface is determined by the TX line rate, the width of the TXDATA port, and whether or not 8B/10B encoding is enabled. A second parallel clock (TXUSRCLK) must be provided for the internal PCS logic in the transmitter. This section shows how to drive the parallel clocks and explains the constraints on those clocks for correct operation.

Interface Width Configuration

The 7 series FPGA GTP transceiver contains a 2-byte internal datapath. The FPGA interface width is configurable by setting the TX_DATA_WIDTH attribute. When the 8B/10B encoder is enabled, the TX_DATA_WIDTH attribute must be configured to 20 bits, 40 bits, or 80 bits, and in this case, the FPGA TX interface only uses the TXDATA ports. For example, TXDATA[15:0] is used when the FPGA interface width is 16. When the 8B/10B encoder is bypassed, the TX_DATA_WIDTH attribute can be configured to any of the available widths: 16, 20, 32, or 40 bits.

Table 3-1 shows how the interface width for the TX datapath is selected. 8B/10B encoding is described in more detail in TX 8B/10B Encoder, page 51.

Table 3-1: **FPGA TX Interface Datapath Configuration**

TX8B10BEN	TX_DATA_WIDTH	FPGA Interface Width	Internal Data Width
1	20	16	20
	40	32	20
0	16	16	16
	20	20	20
	32	32	16
	40	40	20

When the 8B/10B encoder is bypassed and the TX_DATA_WIDTH is 20 or 40, the TXCHARDISPMODE and TXCHARDISPVAL ports are used to extend the TXDATA port

from 16 to 20 bits, or 32 to 40 bits. Table 3-2 shows the data transmitted when the 8B/10B encoder is disabled. When the TX gearbox is used, refer to TX Gearbox, page 54 for data transmission order.

Table 3-2: **TX Data Transmitted when 8B/10B Encoder Bypassed**

	< < < Data Transmission Order is Right to Left (LSB to MSB) < < <																																				
	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data Transmitted	TXCHARDISPMODE[3]	TXCHARDIPSVAL[3]	TXDATA[31:24]								TXCHARDISPMODE[2]	TXCHARDIPSVAL[2]	TXDATA[23:16]								TXCHARDISPMODE[1]	TXCHARDIPSVAL[1]	TXDATA[15:8]								TXCHARDISPMODE[0]	TXCHARDIPSVAL[0]	TXDATA[7:0]					

TXUSRCLK and TXUSRCLK2 Generation

The FPGA TX interface includes two parallel clocks: TXUSRCLK and TXUSRCLK2. TXUSRCLK is the internal clock for the PCS logic in the GTP transceiver transmitter. The required rate for TXUSRCLK depends on the internal datapath width of the GTPE2_CHANNEL primitive and the TX line rate of the GTP transceiver transmitter. Equation 3-1 shows how to calculate the required rate for TXUSRCLK.

$$TXUSRCLK\ Rate = \frac{Line\ Rate}{Internal\ Datapath\ Width} \qquad \textit{Equation 3-1}$$

TXUSRCLK2 is the main synchronization clock for all signals into the TX side of the GTP transceiver. Most signals into the TX side of the GTP transceiver are sampled on the positive edge of TXUSRCLK2. TXUSRCLK2 and TXUSRCLK have a fixed-rate relationship based on the TX_DATA_WIDTH setting. Table 3-3 shows the relationship between TXUSRCLK2 and TXUSRCLK per TX_DATA_WIDTH value.

Table 3-3: **TXUSRCLK2 Frequency Relationship to TXUSRCLK**

FPGA Interface Width	TX_DATA_WIDTH	TXUSRCLK2 Frequency
2-Byte	16, 20	$F_{TXUSRCLK2} = F_{TXUSRCLK}$
4-Byte	32, 40	$F_{TXUSRCLK2} = F_{TXUSRCLK}/2$

These rules about the relationships between clocks must be observed for TXUSRCLK and TXUSRCLK2:

- TXUSRCLK and TXUSRCLK2 must be positive-edge aligned, with as little skew as possible between them. As a result, low-skew clock resources (BUFGs and BUFRs) should be used to drive TXUSRCLK and TXUSRCLK2.
- Even though they might run at different frequencies, TXUSRCLK, TXUSRCLK2, and the transmitter reference clock must have the same oscillator as their source. Thus TXUSRCLK and TXUSRCLK2 must be multiplied or divided versions of the transmitter reference clock.

Ports and Attributes

Table 3-4 defines the FPGA TX Interface ports.

Table 3-4: **FPGA TX Interface Ports**

Port	Dir	Clock Domain	Description
TXCHARDISPMODE[3:0]	In	TXUSRCLK2	When 8B/10B encoding is disabled, TXCHARDISPMODE is used to extend the data bus for 20- and 40-bit TX interfaces.
TXCHARDISPVAL[3:0]	In	TXUSRCLK2	When 8B/10B encoding is disabled, TXCHARDISPVAL is used to extend the data bus for 20- and 40-bit TX interfaces.
TXDATA[31:0]	In	TXUSRCLK2	The bus for transmitting data. The width of this port depends on TX_DATA_WIDTH: TX_DATA_WIDTH = 16, 20: TXDATA[15:0] = 16 bits wide TX_DATA_WIDTH = 32, 40: TXDATA[31:0] = 32 bits wide When a 20-bit or 40-bit bus is required, the TXCHARDISPVAL and TXCHARDISPMODE ports from the 8B/10B encoder is concatenated with the TXDATA port. See Table 3-2, page 45.
TXUSRCLK	In	Clock	This port is used to provide a clock for the internal TX PCS datapath.
TXUSRCLK2	In	Clock	This port is used to synchronize the FPGA logic with the TX interface. This clock must be positive-edge aligned to TXUSRCLK when TXUSRCLK is provided by the user.

Table 3-5 defines the FPGA TX interface attributes.

Table 3-5: **FPGA TX Interface Attributes**

Attribute	Type	Description
TX_DATA_WIDTH	Integer	Sets the bit width of the TXDATA port. When 8B/10B encoding is enabled, TX_DATA_WIDTH must be set to 20 or 40. Valid settings are 16, 20, 32, and 40. See Interface Width Configuration, page 44 for more information.

Using TXOUTCLK to Drive the TX Interface

Depending on the TXUSRCLK and TXUSRCLK2 frequencies, there are different ways FPGA clock resources can be used to drive the parallel clock for the TX interface. Figure 3-2 through Figure 3-5 show different ways FPGA clock resources can be used to drive the parallel clocks for the TX interface. In these examples, the TXOUTCLK is derived from the MGTREFCLK0[P/N] or MGTREFCLK1[P/N] and the TXOUTCLKSEL = 011 to select the TXPLLREFCLK_DIV1 path as indicated in Figure 3-19, page 74.

- Depending on the input reference clock frequency and the required line rate, an MMCM and the appropriate TXOUTCLKSEL port setting is required. The

CORE Generator™ tool creates a sample design based on different design requirements for most cases.

- In use models where TX buffer is bypassed, there are additional restrictions on the clocking resources. Refer to TX Buffer Bypass, page 65 for more information.

TXOUTCLK Driving GTP Transceiver TX in 2-Byte

In Figure 3-2, TXOUTCLK is used to drive TXUSRCLK and TXUSRCLK2 for 2-byte mode (TX_DATA_WIDTH = 16 or 20) in a single-lane configuration. The frequency of TXUSRCLK2 is equal to TXUSRCLK.

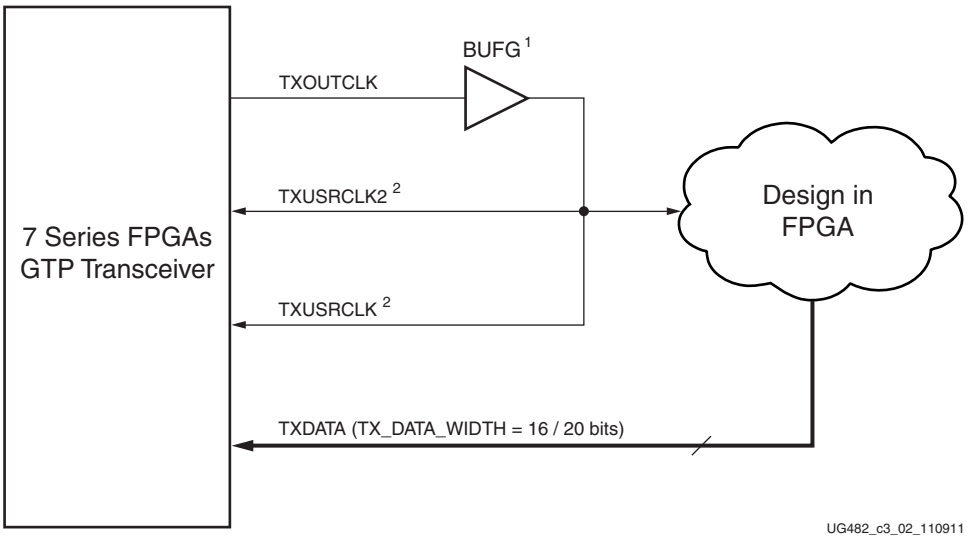


Figure 3-2: **Single Lane—TXOUTCLK Drives TXUSRCLK2 (2-Byte Mode)**

Notes relevant to Figure 3-2:

1. BUFH can be used with certain limitations. In the larger Artix™-7 devices (XC7A200T and XC7A350T), BUFR via BUFMR can be used with certain limitations. For details about placement constraints and restrictions on clocking resources (MMCM, BUFH, BUFMR, BUFR, BUFG, etc.), refer to UG472, *7 Series FPGAs Clocking Resources User Guide*.
2. $F_{TXUSRCLK2} = F_{TXUSRCLK}$.

Similarly, Figure 3-3 shows the shows the same settings in multiple lanes configuration.

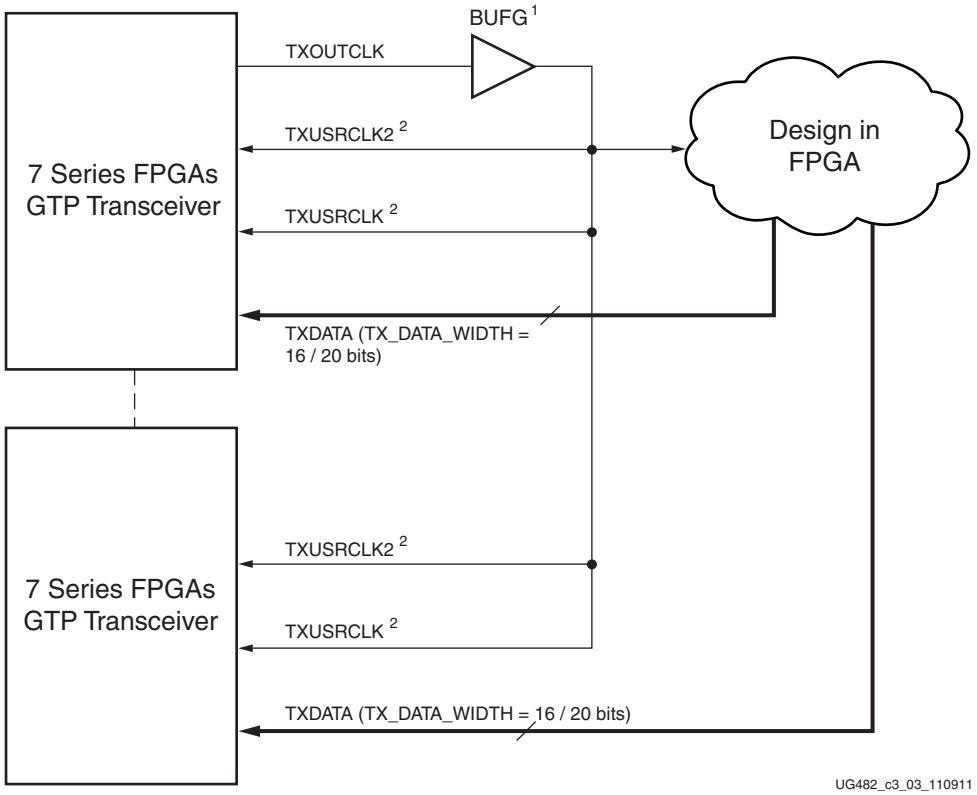


Figure 3-3: **Multiple Lanes—TXOUTCLK Drives TXUSRCLK2 (2-Byte Mode)**

Notes relevant to Figure 3-3:

1. BUFH can be used with certain limitations. In the larger Artix-7 devices (XC7A200T and XC7A350T), BUFR via BUFMR can be used with certain limitations. For details about placement constraints and restrictions on clocking resources (MMCM, BUFH, BUFMR, BUFR, BUFG, etc.), refer to UG472, *7 Series FPGAs Clocking Resources User Guide*.
2. $F_{TXUSRCLK2} = F_{TXUSRCLK}$.

TXOUTCLK Driving GTP Transceiver TX in 4-Byte Mode

In Figure 3-4, TXOUTCLK is used to drive TXUSRCLK2 for 4-byte mode (TX_DATA_WIDTH = 32 or 40). The frequency of TXUSRCLK2 is equal to half of the frequency of TXUSRCLK. MMCMs or PLLs, which are part of the clock management tiles (CMTs) located in the top half of the device, can only drive the BUFGs in the top half of the devices. Similarly, MMCMs or PLLs located in the bottom half can only drive BUFGs in the bottom half.

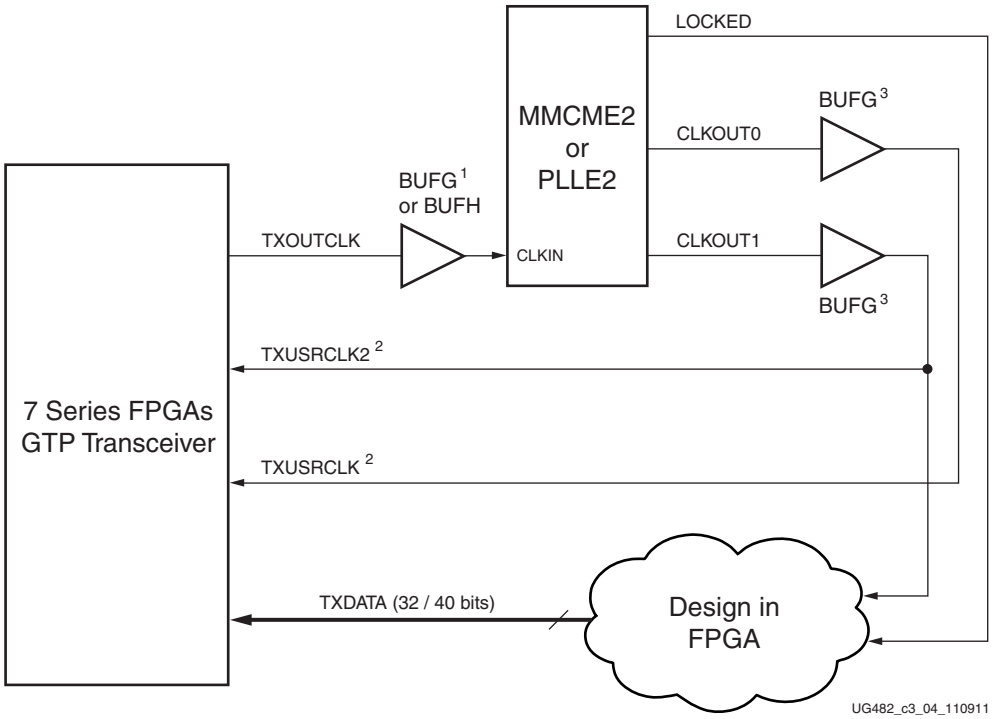


Figure 3-4: **Single Lane—TXOUTCLK Drives TXUSRCLK2 (4-Byte Mode)**

Notes relevant to Figure 3-4:

1. In the larger Artix-7 devices (XC7A200T and XC7A350T), a BUFH or BUFG is not required.
2. $F_{TXUSRCLK2} = F_{TXUSRCLK}/2$
3. In the larger Artix-7 devices (XC7A200T and XC7A350T), BUFR or BUFH can be used with certain limitations. For details about placement constraints and restrictions on clocking resources (MMCM, BUFR, BUFH, BUFG, etc.), refer to UG472, *7 Series FPGAs Clocking Resources User Guide*.

Similarly, Figure 3-5 shows the shows the same settings in multiple lanes configuration.

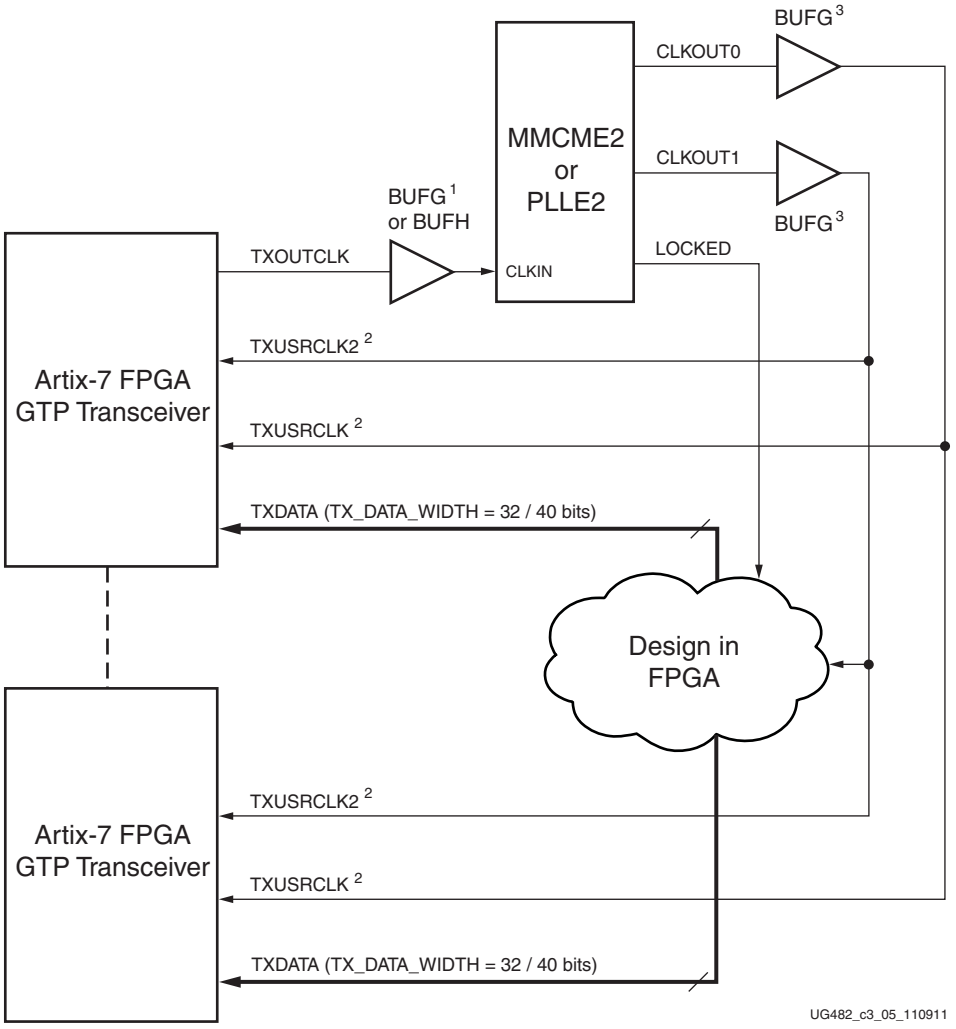


Figure 3-5: **Multiple Lanes—TXOUTCLK Drives TXUSRCLK2 (4-Byte Mode)**

Notes relevant to Figure 3-5:

1. In the larger Artix-7 devices (XC7A200T and XC7A350T), a BUFH or BUFG is not required.
2. $F_{TXUSRCLK2} = F_{TXUSRCLK}/2$.
3. In the larger Artix-7 devices (XC7A200T and XC7A350T), BUFR or BUFH can be used with certain limitations. For details about placement constraints and restrictions on clocking resources (MMCM, BUFR, BUFH, BUFG, etc.), refer to UG472, *7 Series FPGAs Clocking Resources User Guide*.

TX 8B/10B Encoder

Functional Description

Many protocols use 8B/10B encoding on outgoing data. 8B/10B is an industry standard encoding scheme that trades two bits overhead per byte for achieved DC-balance and bounded disparity to allow reasonable clock recovery. The GTP transceiver has a built-in 8B/10B TX path to encode TX data without consuming FPGA resources. Enabling the 8B/10B encoder increases latency through the TX path. The 8B/10B encoder can be disabled or bypassed to minimize latency, if not needed.

8B/10B Bit and Byte Ordering

The order of the bits after the 8B/10B encoder is the opposite of the order shown in Appendix C, 8B/10B Valid Characters, because 8B/10B encoding requires bit a0 to be transmitted first, and the GTP transceiver always transmits the right-most bit first. To match with 8B/10B, the 8B/10B encoder in the GTP transceiver automatically reverses the bit order. Figure 3-6 shows data transmitted by the GTP transceiver when TX_DATA_WIDTH = 20 and 40. The number of bits used by TXDATA and corresponding byte orders are determined by TX_DATA_WIDTH.

- Only use TXDATA[15:0] if TX_DATA_WIDTH = 20
- Use full TXDATA[31:0] if TX_DATA_WIDTH = 40

When the 8B/10B encoder is bypassed and TX_DATA_WIDTH is set to a multiple of 10, 10-bit characters are passed to TX data interface with this format:

- The corresponding TXCHARDISPMODE represents the 9th bit
- The corresponding TXCHARDISPVAL represents the 8th bit
- The corresponding TXDATA byte represents [7:0] bits

K Characters

The 8B/10B table includes special characters (K characters) that are often used for control functions. TXCHARISK ports are used to indicate if data on TXDATA are K characters or regular data. The 8B/10B encoder checks received TXDATA byte to match any K character if corresponding TXCHARISK bit is driven High.

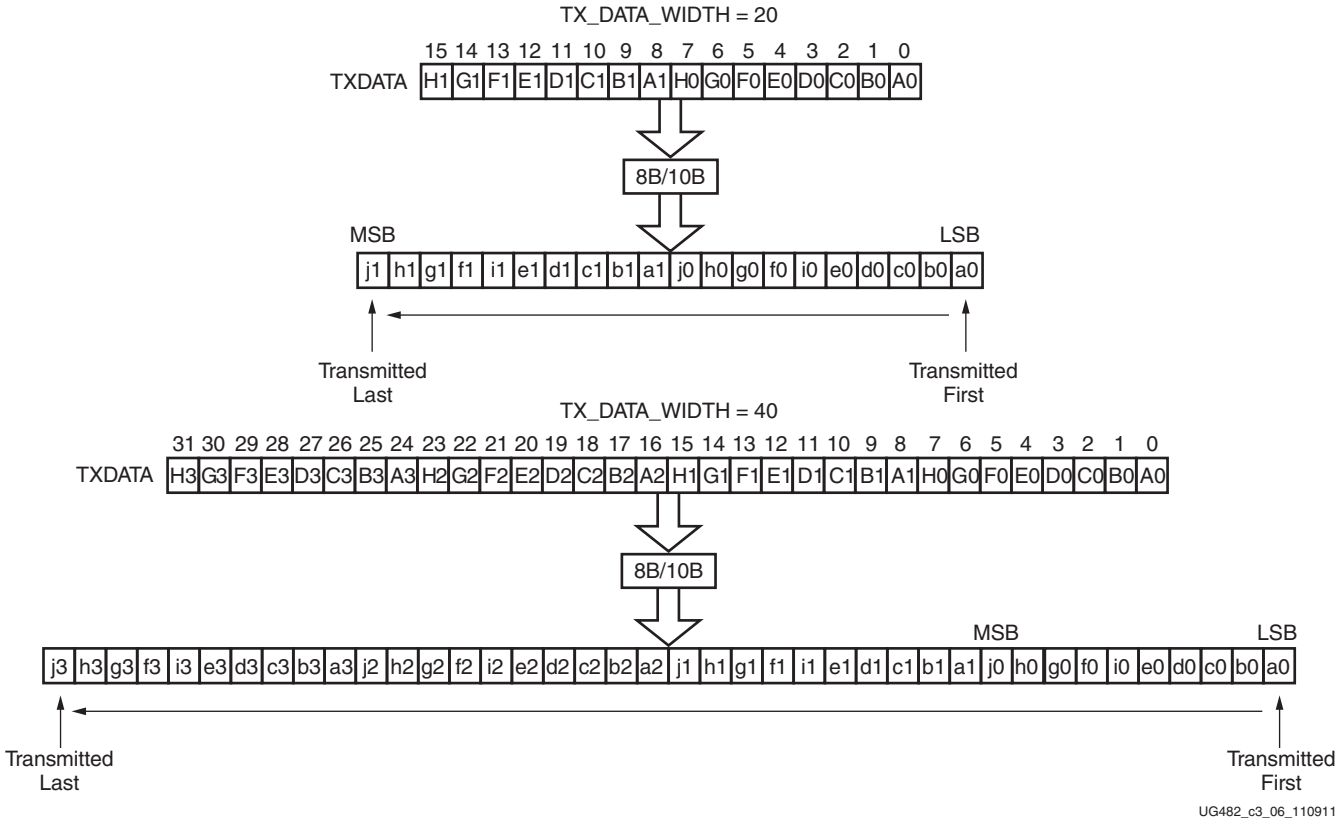


Figure 3-6: **8B/10B Bit and Byte Ordering**

Running Disparity

8B/10B coding is DC-balanced, meaning that the long-term ratio of 1s and 0s transmitted should be exactly 50%. To achieve this, the encoder always calculates the difference between the number of 1s transmitted and the number of 0s transmitted, and at the end of each character transmitted, makes the difference either +1 or –1. This difference is known as the *running disparity*.

To accommodate protocols that use disparity to send control information, the running disparity not only can be generated by the 8B/10B encoder but is also controllable through TXCHARDISPMODE and TXCHARDISPVAL as shown in Table 3-6. For example, an Idle character sent with reversed disparity might be used to trigger clock correction.

Table 3-6: **TXCHARDISPMODE and TXCHARDISPVAL versus Outgoing Disparity**

TXCHARDISPMODE	TXCHARDISPVAL	Outgoing Disparity
0	0	Calculated by the 8B/10B encoder.
0	1	Inverts running disparity when encoding TXDATA.
1	0	Forces running disparity negative when encoding TXDATA.
1	1	Forces running disparity positive when encoding TXDATA.

Ports and Attributes

Table 3-7 lists the ports required by the TX 8B/10B encoder.

Note: There are no TX encoder attributes.

Table 3-7: **TX 8B/10B Encoder Ports**

Port	Dir	Clock Domain	Description
TX8B10BBYPASS[3:0]	In	TXUSRCLK2	This active-High port allows byte-interleaved data to bypass 8B/10B on a per-byte basis. TX8B10BEN must be High to use this per-byte bypass mode. TX8B10BBYPASS [3] corresponds to TXDATA[31:24] TX8B10BBYPASS [2] corresponds to TXDATA[23:16] TX8B10BBYPASS [1] corresponds to TXDATA[15:8] TX8B10BBYPASS [0] corresponds to TXDATA[7:0] TX8B10BBYPASS[x] = 1, encoder for byte x is bypassed. TX8B10BBYPASS[x] = 0, encoder for byte x is used.
TX8B10BEN	In	TXUSRCLK2	TX8B10BEN is set High to enable the 8B/10B encoder. TX_DATA_WIDTH must be set to 20 or 40 when the 8B/10B encoder is enabled. 0: 8B/10B encoder bypassed. This option reduces latency. 1: 8B/10B encoder enabled.
TXCHARDISPMODE[3:0]	In	TXUSRCLK2	Set High to work with TXCHARDISPVAL to force running disparity negative or positive when encoding TXDATA. Set Low to use normal running disparity. Refer to Table 3-6 for a detailed definition. TXCHARDISPMODE[3] corresponds to TXDATA[31:24] TXCHARDISPMODE[2] corresponds to TXDATA[23:16] TXCHARDISPMODE[1] corresponds to TXDATA[15:8] TXCHARDISPMODE[0] corresponds to TXDATA[7:0]
TXCHARDISPVAL[3:0]	In	TXUSRCLK2	Work with TXCHARDISPMODE to provide running disparity control. Refer to Table 3-6 for detailed information. TXCHARDISPVAL[3] corresponds to TXDATA[31:24] TXCHARDISPVAL[2] corresponds to TXDATA[23:16] TXCHARDISPVAL[1] corresponds to TXDATA[15:8] TXCHARDISPVAL[0] corresponds to TXDATA[7:0]
TXCHARISK[3:0]	In	TXUSRCLK2	When High, indicates the corresponding data byte on TXDATA is a valid K character. TXCHARISK[3] corresponds to TXDATA[31:24] TXCHARISK[2] corresponds to TXDATA[23:16] TXCHARISK[1] corresponds to TXDATA[15:8] TXCHARISK[0] corresponds to TXDATA[7:0] A TXCHARISK bit should be driven Low when the corresponding data byte from TXDATA is set to bypass the 8B/10B encoder.

Enabling and Disabling 8B/10B Encoding

To enable the 8B/10B encoder, TX8B10BEN must be driven High. The TX 8B/10B encoder allows byte interleaved data to bypass the encoder on a per-byte basis. When TX8B10BEN is driven Low, all encoders are turned off and no data from TXDATA can be encoded. When TX8B10BEN is High, driving a bit from TX8B10BBYPASS High can make the corresponding byte channel from TXDATA bypass 8B/10B encoding. When the encoder is turned off, the operation of the TXDATA port is as described in the FPGA TX interface.

TX Gearbox

Functional Description

Some high-speed data rate protocols use 64B/66B encoding to reduce the overhead of 8B/10B encoding while retaining the benefits of an encoding scheme. The TX gearbox provides support for 64B/66B and 64B/67B header and payload combining. The Interlaken interface protocol specification uses the 64B/67B encoding scheme. Refer to the Interlaken specification for further information. The Interlaken specification can be downloaded from: http://www.interlakenalliance.com/.

The TX gearbox supports 2-byte and 4-byte interfaces. Scrambling of the data is done in the FPGA logic.

Ports and Attributes

Table 3-8 defines the TX gearbox ports.

Table 3-8: **TX Gearbox Ports**

Port Name	Dir	Clock Domain	Description
TXGEARBOXREADY	Out	TXUSRCLK2	This output indicates if data can be applied to the 64B/66B or 64B/67B gearbox when GEARBOX_MODE is set to use the gearbox. 0: No data can be applied 1: Data must be applied
TXHEADER[2:0]	In	TXUSRCLK2	These ports are the header inputs. [1:0] are used for the 64B/66B gearbox, and [2:0] are used for the 64B/67B gearbox.
TXSEQUENCE[6:0]	In	TXUSRCLK2	These inputs are used for the fabric sequence counter when the TX gearbox is used. [5:0] are used for the 64B/66B gearbox, and [6:0] are used for the 64B/67B gearbox.
TXSTARTSEQ	In	TXUSRCLK2	This input indicates the first word to be applied after reset for the 64B/66B or 64B/67B gearbox. The internal sequencer counter must be enabled by the GEARBOX_MODE attribute.

Table 3-9 defines the TX gearbox attributes.

Table 3-9: **TX Gearbox Attributes**

Attribute	Type	Description
GEARBOX_MODE	3-bit Binary	This attribute indicates the TX and RX gearbox modes: • Bit 2: Set to 0. Unused. • Bit 1: 0: Use the external sequence counter and apply inputs to TXSEQUENCE. 1: Use the internal sequence counter, gate the input header and data with the TXGEARBOXREADY output. • Bit 0: 0: 64B/67B gearbox mode for Interlaken. 1: 64B/66B gearbox.
TXGEARBOX_EN	Boolean	When TRUE, this attribute enables the TX gearbox.

Enabling the TX Gearbox

To enable the TX gearbox for the GTP transceiver, set the attribute TXGEARBOX_EN to TRUE. The GEARBOX_MODE attribute controls the GTP transceiver's TX and RX gearbox use modes.

TX Gearbox Bit and Byte Ordering

Figure 3-7 shows an example of the first four cycles of data entering and exiting the TX gearbox for 64B/66B encoding when using a 2-byte logic interface (TX_DATA_WIDTH = 16 (2-byte)). The input consists of a 2-bit header and 16 bits of data. On the first cycle, the header and 14 bits of data exit the TX gearbox. On the second cycle, the remaining two data bits from the previous cycle's TXDATA input along with 14 data bits from the current TXDATA input exit the TX gearbox. This continues for the third and fourth cycle. On the fifth cycle, the output of the TX gearbox contains two remaining data bits from the first 66-bit block, the header of the second 66-bit block, and 28 data bits from the second 66-bit block.

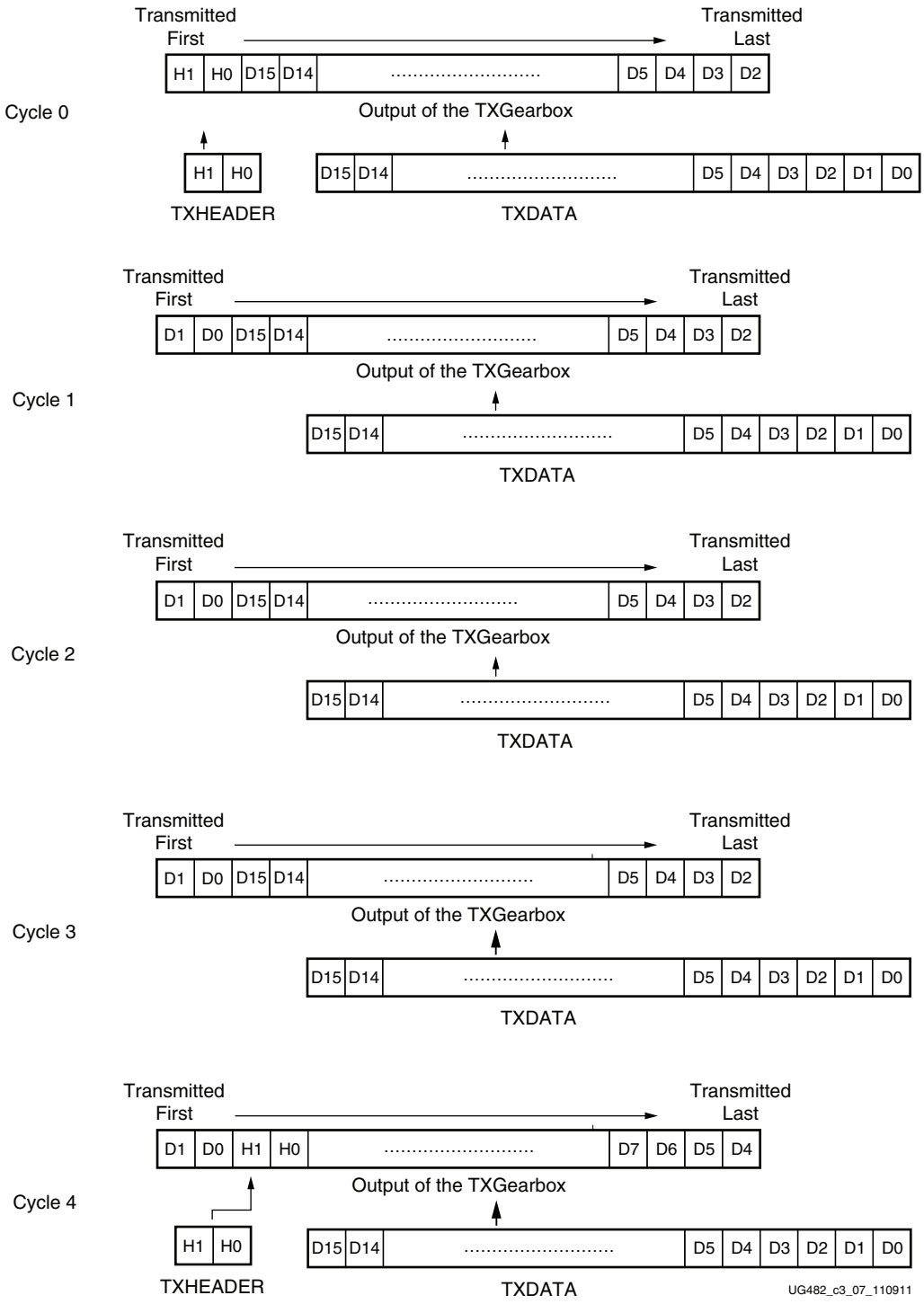


Figure 3-7: **TX Gearbox Bit Ordering**

Note relevant to Figure 3-7:

1. Per IEEE802.3ae nomenclature, H1 corresponds to TxB<0>, H0 to TxB<1>, etc.

TX Gearbox Operating Modes

The TX gearbox has two operating modes. The external sequence counter operating mode must be implemented in user logic. The second mode uses an internal sequence counter. The TX gearbox supports 2-byte and 4-byte interfaces to the FPGA logic.

External Sequence Counter Operating Mode

As shown in Figure 3-8, the external sequence counter operating mode uses the TXSEQUENCE [6:0], TXDATA[31:0], and TXHEADER[2:0] inputs. A binary counter must exist in the user logic to drive the TXSEQUENCE port. For 64B/66B encoding, the counter increments from 0 to 32 and repeats from 0. For 64B/67B encoding, the counter increments from 0 to 66 and repeats from 0. When using 64B/66B encoding, tie TXSQUENCE [6] to logic 0 and tie the unused TXHEADER [2] to logic 0. The sequence counter increment ranges ({0 to 32}, {0 to 66}) are identical for 2-byte and 4-byte interfaces. However, the counter must increment once every two TXUSRCLK2 cycles when using a 2-byte interface and every TXUSRCLK2 cycle when using a 4-byte interface.

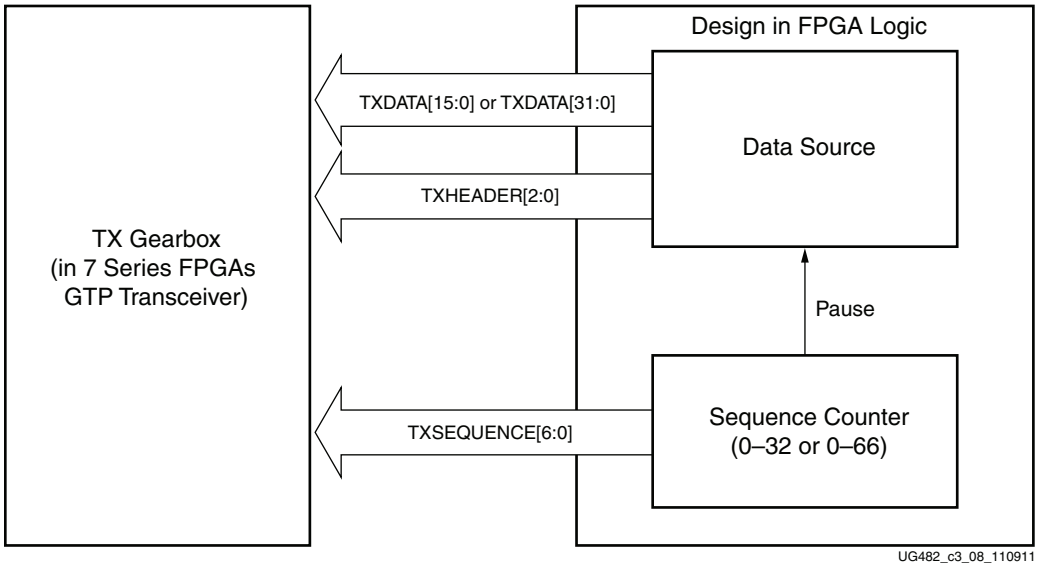


Figure 3-8: **TX Gearbox in External Sequence Counter Mode**

Due to the nature of the 64B/66B and 64B/67B encoding schemes, user data is held (paused) during various sequence counter values. Data is paused for two TXUSRCLK2 cycles in 2-byte mode and for one TXUSRCLK2 cycle in 4-byte mode. Valid data transfer is resumed on the next TXUSRCLK2 cycle. The data pause only applies to TXDATA and not to TXHEADER. The TXSEQUENCE pause locations for various modes are described in Table 3-10 and Table 3-11.

Table 3-10: **64B/66B Encoding Frequency of TXSEQUENCE and Pause Locations**

TX_DATA_WIDTH	Frequency of TXSEQUENCE	TXSEQUENCE PAUSE
32 (4-byte)	1 X TXUSRCLK2	31
16 (2-byte)	2 X TXUSRCLK2	31

Table 3-11: **64B/67B Encoding Frequency of TXSEQUENCE and Pause Locations**

TX_DATA_WIDTH	Frequency of TXSEQUENCE	TXSEQUENCE PAUSE
32 (4-byte)	1 X TXUSRCLK2	21, 44, 65
16 (2-byte)	2 X TXUSRCLK2	21, 44, 65

Figure 3-9 shows how a pause occurs at counter value 31 when using an 4-byte fabric interface in external sequence counter mode with 64B/66B encoding.

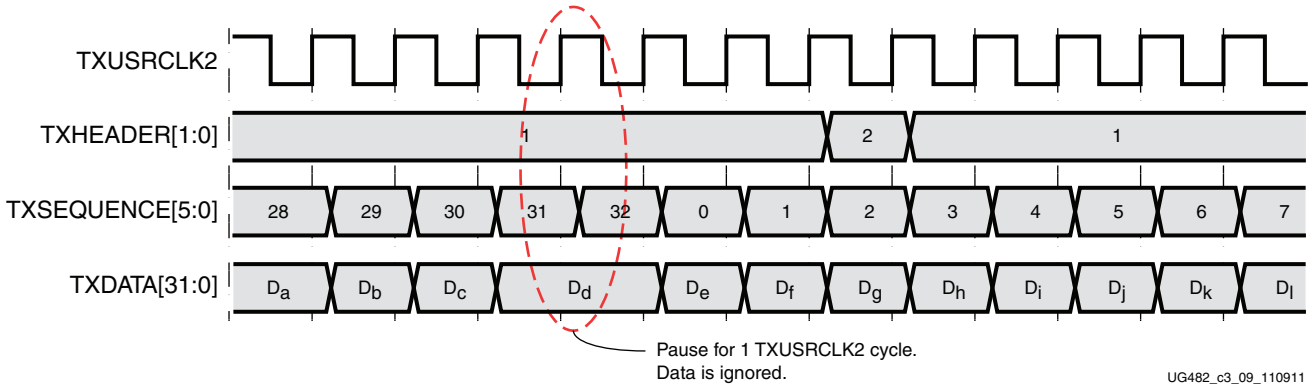


Figure 3-9: **Pause at Sequence Counter Value 31**

Figure 3-10 shows how a pause occurs at counter value 44 when using a 2-byte fabric interface in external sequence counter mode with 64B/67B encoding.

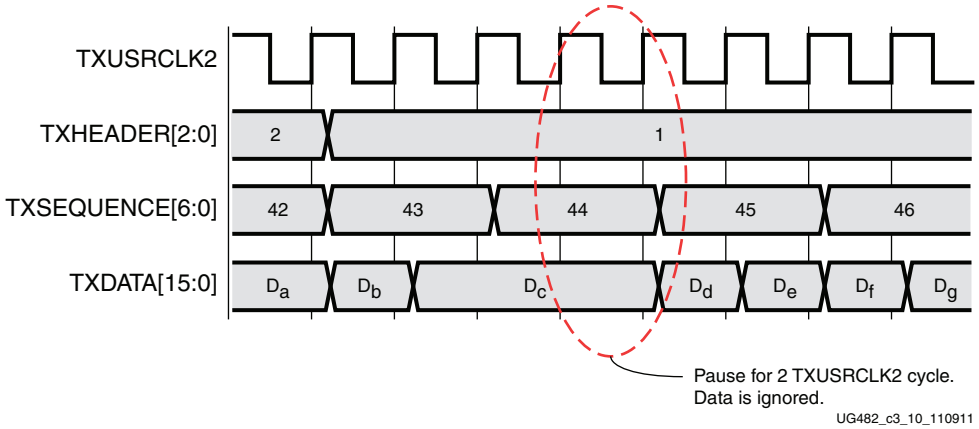


Figure 3-10: **Pause at Sequence Counter Value 44**

The sequence of transmitting 64/67 data for the external sequence counter mode is:

1. Apply GTTXRESET and wait until the reset cycle is completed.
2. During reset, apply `7'h00` to TXSEQUENCE, header information to TXHEADER, and initial data to TXDATA. This state can be held indefinitely until data transmission is ready.
3. On count 0, apply data to TXDATA and header information to TXHEADER. For a 2-byte interface (TX_DATA_WIDTH = 16), drive the second 2 bytes to TXDATA while still on count 0.
4. The sequence counter increments to 1 while data is driven on TXDATA.
5. After applying 4 bytes of data, the counter increments to 2. Apply data on TXDATA and header information on TXHEADER.
6. On count 21, stop data pipeline.
7. On count 22, drive data on TXDATA.
8. On count 44, stop data pipeline.

9. On count 45, drive data on TXDATA.
10. On count 65, stop data pipeline.
11. On count 66, drive data on TXDATA.

The sequence of transmitting 64/66 data for the external sequence counter mode is as follows:

1. Apply GTTXRESET and wait until the reset cycle is completed.
2. During reset, apply `6'h00` to TXSEQUENCE, the appropriate header data to TXHEADER, and initial data to TXDATA. This state can be held indefinitely until data transmission is ready.
3. On count 0, apply data to TXDATA and header information to TXHEADER. For a 2-byte interface (TX_DATA_WIDTH = 16), drive the second 2 bytes to TXDATA while still on count 0.
4. The sequence counter increments to 1 while data is driven on TXDATA.
5. After applying 4 bytes of data, the counter increments to 2. Drive data on TXDATA and header information on TXHEADER.
6. On count 31, stop data pipeline.
7. On count 32, drive data on TXDATA.

Internal Sequence Counter Operating Mode

As shown in Figure 3-11, the internal sequence counter operating mode uses the TXSTARTSEQ input and the TXGEARBOXREADY output, in addition to the TXDATA data inputs and the TXHEADER header inputs. In this use model, the TXSEQUENCE inputs are not used. The use model is similar to the previous use model, except that the TXGEARBOXREADY output is used.

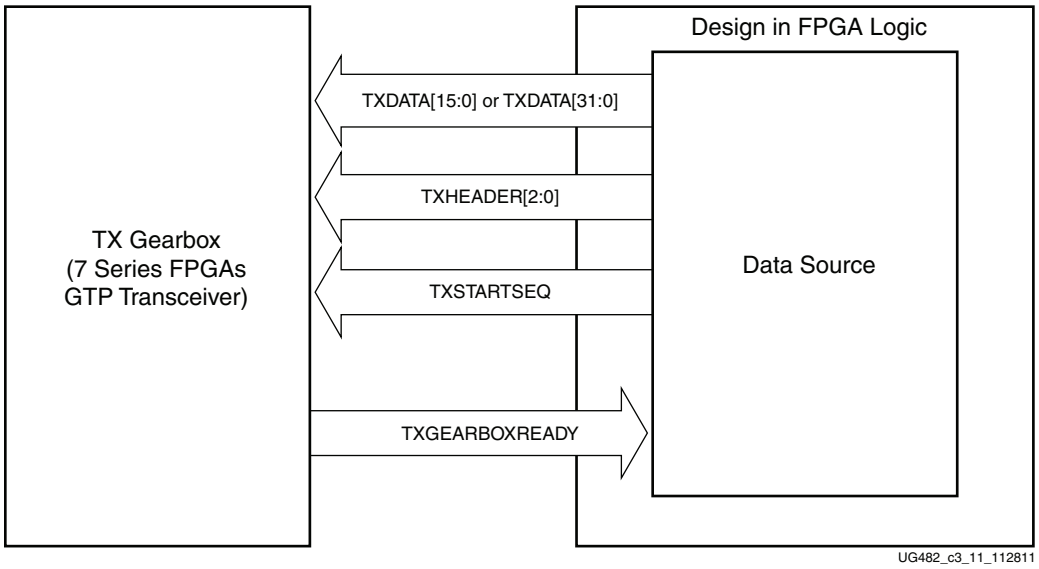


Figure 3-11: **TX Gearbox in Internal Sequence Counter Mode**

The TXSTARTSEQ input indicates to the TX gearbox when the first byte of data after a reset is valid. TXSTARTSEQ is asserted High when the first byte of valid data is applied after a reset condition. The TXDATA and TXHEADER inputs must be held stable after reset, and TXSTARTSEQ must be held Low until data can be applied continuously. There are no

requirements on how long a user can wait before starting to transmit data. TXSTARTSEQ is asserted High along with the first 2 or 4 bytes of valid data and not before. After the first bytes of data, TXSTARTSEQ can be held at any value that is convenient.

After data is driven, TXGEARBOXREADY is deasserted Low for two TXUSRCLK2 cycles (4-byte mode) or three TXUSRCLK2 cycles (2-byte mode). Figure 3-12 and Figure 3-13 show the behavior of TXGEARBOXREADY for a 4-byte fabric interface and a 2-byte fabric interface, respectively. When TXGEARBOXREADY is deasserted Low, only one TXUSRCLK2 cycle remains before the data pipe must be stopped. The 1-cycle latency is fixed and cannot be changed. After one cycle of latency, data must be held through until TXGEARBOXREADY transitions High, where new data must be driven. For this mode of operation, the number of hold points is identical to when using the external sequence counter mode for 64B/67B and 64B/66B.

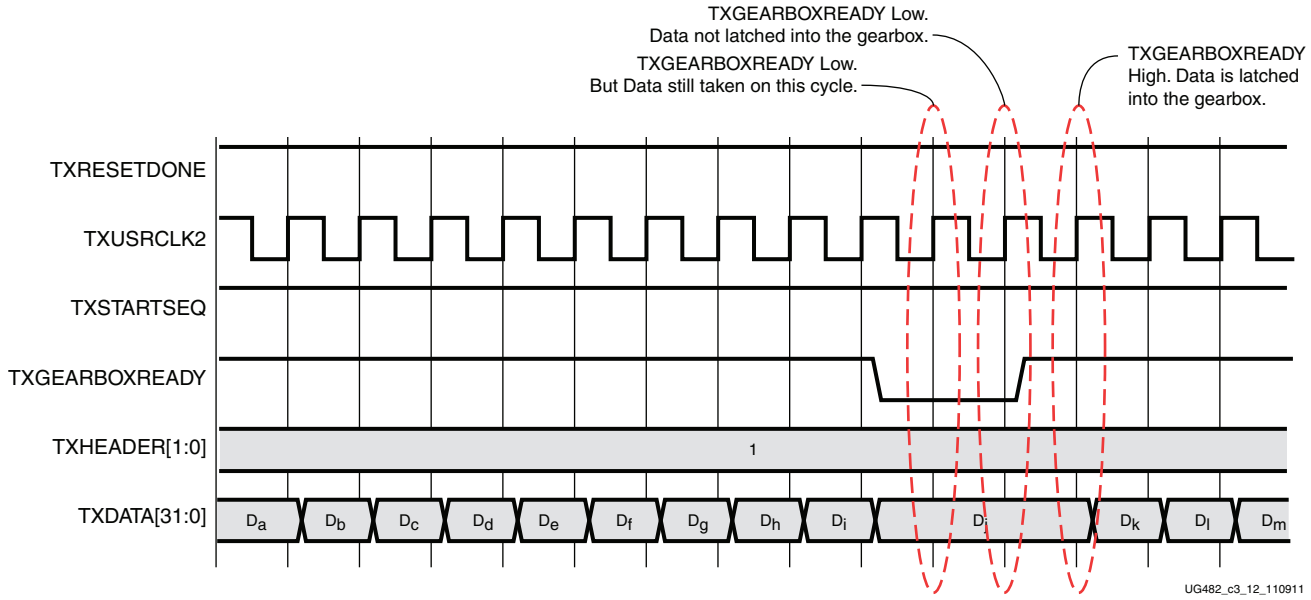


Figure 3-12: **TX Gearbox Internal Sequence Mode, TX_DATA_WIDTH = 32 (4-Byte), 64B/66B**

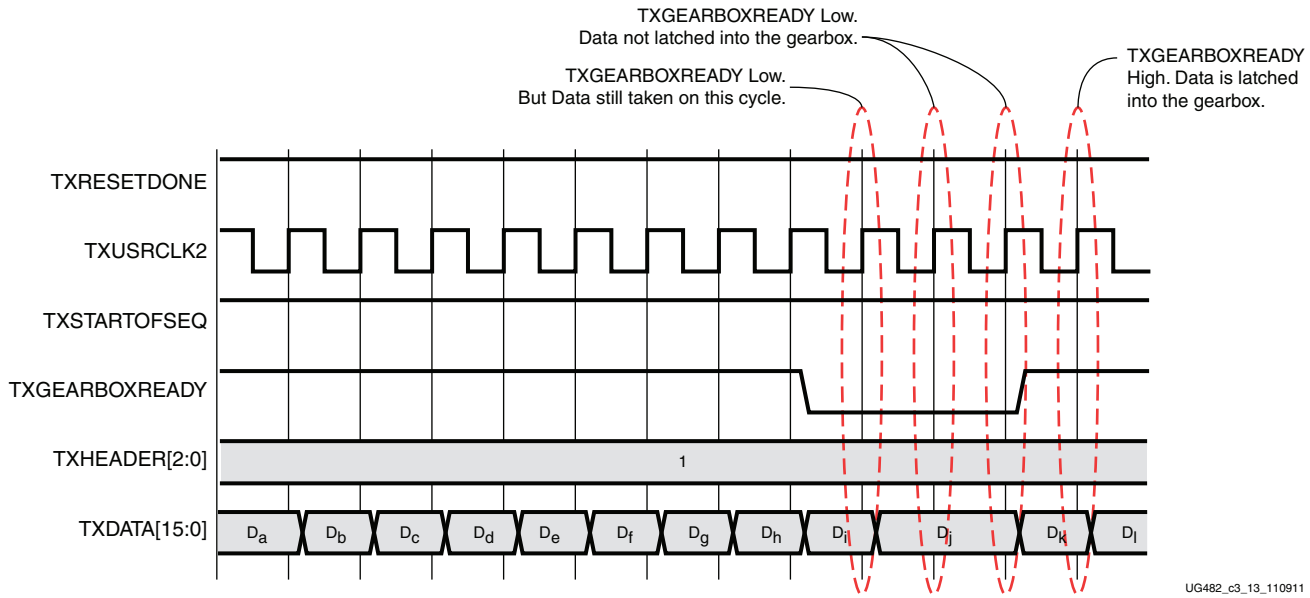


Figure 3-13: **TX Gearbox Internal Sequence Mode, TX_DATA_WIDTH = 16 (2-Byte), 64B/67B**

The sequence of transmitting data for the internal sequence counter mode is:

1. Hold TXSTARTSEQ Low.
2. Assert GTTXRESET and wait until the reset cycle is completed.
3. TXGEARBOXREADY goes High.
4. During reset, place the appropriate header data on TXHEADER and the initial data on TXDATA. This state can be held indefinitely in readiness for data transmission.
5. Drive TXSTARTSEQ High and place the first valid header information on TXHEADER and data on TXDATA.
6. Continue to drive header information and data until TXGEARBOXREADY goes Low.
7. When TXGEARBOXREADY goes Low, drive the last 2 or 4 bytes of data and the header information.
8. Hold the data pipeline for one TXUSRCLK2 cycle for a 4-byte input or two TXUSRCLK2 cycles for a 2-byte input.
9. On the next TXUSRCLK2 cycle, drive data on the TXDATA inputs. TXGEARBOXREADY is asserted High on the previous TXUSRCLK2 cycle.

TX Buffer

Functional Description

The GTP transceiver TX datapath has two internal parallel clock domains used in the PCS: the PMA parallel clock domain (XCLK) and the TXUSRCLK domain. To transmit data, the XCLK rate must match the TXUSRCLK rate, and all phase differences between the two domains must be resolved. Figure 3-14 shows the XCLK and TXUSRCLK domains.

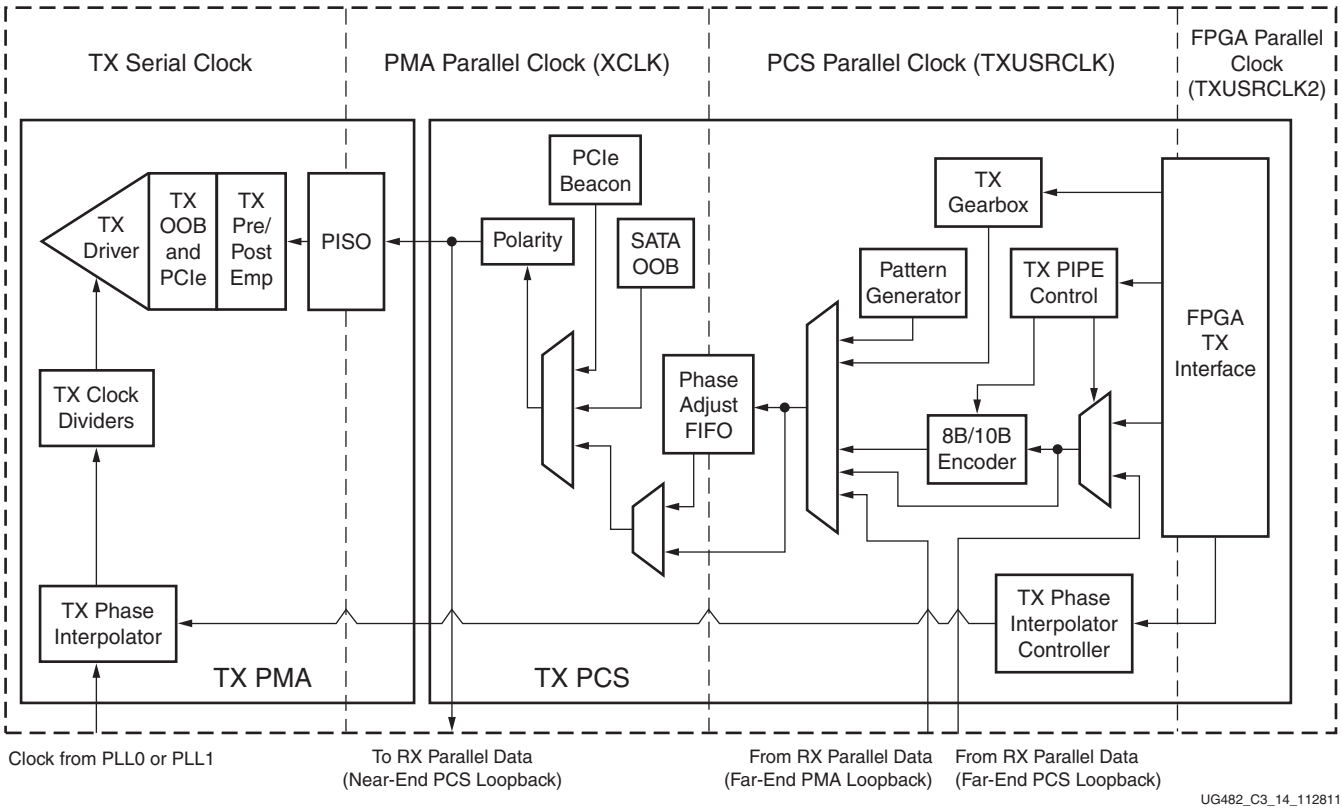


Figure 3-14: **TX Clock Domains**

The GTP transceiver transmitter includes a TX buffer and a TX phase alignment circuit to resolve phase differences between the XCLK and TXUSRCLK domains. The TX phase alignment circuit is used when TX buffer is bypassed (see TX Buffer Bypass, page 65). All TX datapaths must use either the TX buffer or the TX phase alignment circuit. Table 3-12 shows trade-offs between buffering and phase alignment.

Table 3-12: **TX Buffering versus Phase Alignment**

	TX Buffer	TX Phase Alignment
Ease of Use	The TX buffer is the recommended default to use when possible. It is robust and easier to operate.	Phase alignment is an advanced feature that requires extra logic and additional constraints on clock sources. TXOUTCLKSEL must select the GTP transceiver reference clock as the source of TXOUTCLK to drive TXUSRCLK.
Latency	If low latency is critical, the TX buffer must be bypassed.	Phase alignment uses fewer register in the TX datapath to achieve lower and deterministic latency.
TX Lane-to-Lane Deskew		The TX phase alignment circuit can be used to reduce the lane skew between separate GTP transceivers. All GTP transceivers involved must use the same line rate.

Ports and Attributes

Table 3-13 defines the TX buffer ports.

Table 3-13: **TX Buffer Ports**

Port	Dir	Clock Domain	Description
TXBUFSTATUS[1:0]	Out	TXUSRCLK2	TX buffer status. TXBUFSTATUS[1]: TX buffer overflow or underflow status. When TXBUFSTATUS[1] is set High, it remains High until the TX buffer is reset. 1: TX FIFO has overflow or underflow. 0: No TX FIFO overflow or underflow error. TXBUFSTATUS[0]: TX buffer fullness. 1: TX FIFO is at least half full. 0: TX FIFO is less than half full.

Table 3-14 defines the TX buffer attributes.

Table 3-14: **TX Buffer Attributes**

Attribute	Type	Description
TXBUF_EN	Boolean	Use or bypass the TX buffer. TRUE: Uses the TX buffer (default). FALSE: Bypasses the TX buffer (advanced feature).

Table 3-14: **TX Buffer Attributes *(Cont'd)***

Attribute	Type	Description
TX_XCLK_SEL	String	Selects the clock source used to drive the PMA parallel clock domain (XCLK). TXOUT: Selects TXOUTCLK as source of XCLK. Use when using the TX buffer. TXUSR: Selects TXUSRCLK as source of XCLK. Used when bypassing the TX buffer.
TXBUF_RESET_ON_RATE_CHANGE	Boolean	GTP transceiver internally generated TX buffer reset on rate change. TRUE: Enables auto TX buffer reset on rate change. FALSE: Disables auto TX buffer reset on rate change.

Using the TX Buffer

The TX buffer should be reset whenever TXBUFSTATUS indicates an overflow or underflow condition. The TX buffer can be reset by using GTTXRESET, TXPCSRESET, or the GTP transceiver internally generated TX buffer reset on rate change when TXBUF_RESET_ON_RATE_CHANGE = TRUE. Assertion of GTTXRESET triggers a sequence that resets the entire transmitter of the GTP transceiver. These settings are use to enable the TX buffer to resolve phase differences between the XCLK and TXUSRCLK domains:

- TXBUF_EN = TRUE
- TX_XCLK_SEL = TXOUT

TX Buffer Bypass

Functional Description

Bypassing the TX buffer is an advanced feature of the 7 series GTP transceiver. The TX phase alignment circuit is used to adjust the phase difference between the PMA parallel clock domain (XCLK) and the TXUSRCLK domain when the TX buffer is bypassed. It also performs the TX delay alignment by adjusting the TXUSRCLK to compensate for the temperature and voltage variations. The combined TX phase and delay alignments can be automatically performed by the GTP transceiver or manually controlled by the user. Refer to Figure 3-14, page 63 for the XCLK and TXUSRCLK domains and Table 3-12, page 64 for trade-offs between buffering and phase alignment.

Ports and Attributes

Table 3-15 defines the TX buffer bypass ports.

Table 3-15: **TX Buffer Bypass Ports**

Port	Dir	Clock Domain	Description
TXPHDLYRESET	In	Async	TX phase alignment hard reset to force TXUSRCLK to the center of the delay alignment tap. The delay alignment tap has a full range of ±4 ns and a half range of ±2 ns. This hard reset can be used to initiate the GTP transceiver to perform the TX phase and delay alignment automatically when all other TX buffer bypass input ports are set Low.
TXPHALIGN	In	Async	Sets the TX phase alignment. Tied Low when using the auto alignment mode.
TXPHALIGNEN	In	Async	Enables the TX phase alignment in manual mode. Tied Low when using the auto mode.
TXPHDLYPD	In	Async	TX phase and delay alignment circuit power down. Tied High when: • TX buffer bypass is not in use. • TXPD is asserted. • TXOUTCLKSEL is set to `3'b011` but the reference clock is not connected. TXPHDLYPD is Tied Low during normal operation of TX buffer bypass mode. 0: Power-up the TX phase and delay alignment circuit. 1: Power-down the TX phase and delay alignment circuit.
TXPHINIT	In	Async	TX phase alignment initialization. Reserved. Tied Low when using the auto alignment mode.
TXPHOVRDEN	In	Async	TX phase alignment counter override enable. Tied Low when not in use. 0: Normal operation. 1: Enables TX phase alignment counter override with the value from TXPH_CFG[10:0].

Table 3-15: **TX Buffer Bypass Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
TXDLYSRESET	In	Async	TX delay alignment soft reset to gradually shift TXUSRCLK to the center of the delay alignment tap. The delay alignment tap has a full range of ±4 ns and a half range of ±2 ns. This soft reset can be used to initiate the GTP transceiver to perform the TX phase and delay alignment automatically when all other TX buffer bypass input ports are set Low.
TXDLYBYPASS	In	Async	TX delay alignment bypass. 0: Uses the TX delay alignment circuit. 1: Bypasses the TX delay alignment circuit.
TXDLYEN	In	Async	Enables the TX delay alignment in manual mode. Tied Low when using the auto mode.
TXDLYOVRDEN	In	Async	TX delay alignment counter override enable. Tied Low when not in use. 0: Normal operation. 1: Enables TX delay alignment counter override with the value from TXDLY_CFG[14:6].
TXPHDLYTSTCLK	In	Async	TX phase and delay alignment test clock. Used with TXDLYHOLD and TXDLYUPDOWN.
TXDLYHOLD	In	TXPHDLYTSTCLK	TX delay alignment hold. Used as a hold override when TXPHDLY_CFG[1] = 1 to bypass the TX phase and delay alignment voter. Tied Low when not in use.
TXDLYUPDOWN	In	TXPHDLYTSTCLK	TX delay alignment up or down. Used as an up or down override when TXPHDLY_CFG[1] = 1 to bypass the TX phase and delay alignment voter. Tied Low when not in use.
TXPHALIGNDONE	Out	Async	TX phase alignment done. When the auto TX phase and delay alignment is used, the second rising edge of TXPHALIGNDONE detected after TXDLYSRESETDONE assertion indicates TX phase and delay alignment are done.
TXPHINITDONE	Out	Async	Indicates that TX phase alignment initialization is done.

Table 3-15: **TX Buffer Bypass Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
TXDLYSRESETDONE	Out	Async	Indicates that TX delay alignment soft reset is done.
TXSYNCMODE	In	Async	Selects between TX buffer bypass slave and master lanes. 0: TX buffer bypass slave lane 1: TX buffer bypass master lane
TXSYNCIN	In	Async	For use in TX buffer bypass multilane applications. This input is connected to the TXSYNCOUT from the TX buffer bypass master lane.
TXSYNCALLIN	In	Async	• Single-lane mode: This input is connected to its own TXPHALIGNDONE. • Multi-lane mode: This input is connected to the ANDed signal of TXPHALIGNDONE of the master and all slave lanes.
TXSYNCOUT	Out	Async	For use only by the TX buffer bypass master lane in a multilane application. Connects to the TXSYNCIN port of all lanes that are part of a multi-lane application.
TXSYNCDONE	Out	Async	Indicates TX buffer bypass alignment procedure completion. Only valid for TX buffer bypass master lane.

Table 3-16: **TX Buffer Attributes**

Attribute	Type	Description
TXBUF_EN	Boolean	Use or bypass the TX buffer. TRUE: Uses the TX buffer (default). FALSE: Bypasses the TX buffer (advanced feature).
TX_XCLK_SEL	String	Selects the clock source used to drive the PMA parallel clock domain (XCLK). TXOUT: Selects TXOUTCLKPMA as the source of XCLK. Used when using the TX buffer. TXUSR: Selects TXUSRCLK as the source of XCLK. Used when bypassing the TX buffer.
TXPH_CFG	16-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

Table 3-16: **TX Buffer Attributes *(Cont'd)***

Attribute	Type	Description
TXPH_MONITOR_SEL	5-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
TXPHDLY_CFG	24-bit Binary	TX phase and delay alignment configuration. TXPHDLY_CFG[19] = 1 is used to set the TX delay alignment tap to the full range of ±4 ns. TXPHDLY_CFG[19] = 0 is used to set the TX delay alignment tap to the half range of ±2 ns. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
TXDLY_CFG	16-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
TXDLY_LCFG	9-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
TXDLY_TAP_CFG	16-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
TXSYNC_OVRD	1-bit Binary	Manual mode override. 0: TX buffer bypass auto mode is enabled. 1: TX buffer bypass manual mode is used. TX buffer bypass control must be implemented in fabric logic.
TXSYNC_SKIP_DA	1-bit Binary	Control to skip the delay alignment procedure. Only valid on the buffer bypass master lane. 0: TX delay alignment procedure occurs. 1: TX delay alignment procedure is skipped.
TXSYNC_MULTILANE	1-bit Binary	Indicates whether the lane is used as part of a multilane interface. Only valid on the TX buffer bypass master lane. 0: This lane is used in single-lane mode. 1: This lane is used in multilane mode.

TX Pattern Generator

Functional Description

Pseudo-random bit sequences (PRBS) are commonly used to test the signal integrity of high-speed links. These sequences appear random but have specific properties that can be used to measure the quality of a link. The GTP transceiver pattern generator block can generate several industry-standard PRBS patterns listed in Table 3-17.

Table 3-17: **Supported PRBS Patterns**

Name	Polynomial	Length of Sequence	Description
PRBS-7	$1 + X^6 + X^7$	2^7 - 1 bits	Used to test channels with 8B/10B.
PRBS-15	$1 + X^{14} + X^{15}$	2^{15} - 1 bits	ITU-T Recommendation O.150, Section 5.3. PRBS-15 is often used for jitter measurement because it is the longest pattern the Agilent DCA-J sampling scope can handle.
PRBS-23	$1 + X^{18} + X^{23}$	2^{23} - 1 bits	ITU-T Recommendation O.150, Section 5.6. PRBS-23 is often used for non-8B/10B encoding schemes. It is one of the recommended test patterns in the SONET specification.
PRBS-31	$1 + X^{28} + X^{31}$	2^{31} - 1 bits	ITU-T Recommendation O.150, Section 5.8. PRBS-31 is often used for non-8B/10B encoding schemes. It is a recommended PRBS test pattern for 10 Gigabit Ethernet. See IEEE 802.3ae-2002.

In addition to PRBS patterns, the GTP transceiver supports 16-UI or 20-UI square wave test patterns, depending on data width as well as a 2-UI square wave test pattern and PCI Express compliance pattern generation (see Table 3-18 and Figure 3-15). Clocking patterns are usually used to check PLL random jitter often done with a spectrum analyzer.

Table 3-18: **PCI Express Compliance Pattern**

Symbol	K28.5	D21.5	K28.5	D10.2
Disparity	0	1	1	0
Pattern	0011111010	1010101010	1100000101	0101010101

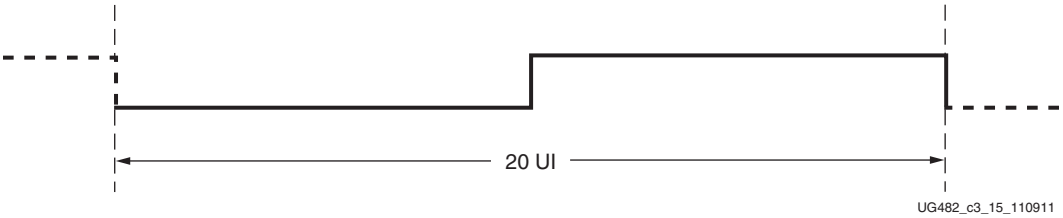


Figure 3-15: **20-UI Square Wave**

The error insertion function is supported to verify link connection and also for jitter tolerance tests. When an inverted PRBS pattern is necessary, the TXPOLARITY signal is used to control polarity. Figure 3-16 shows the TX pattern generator block.

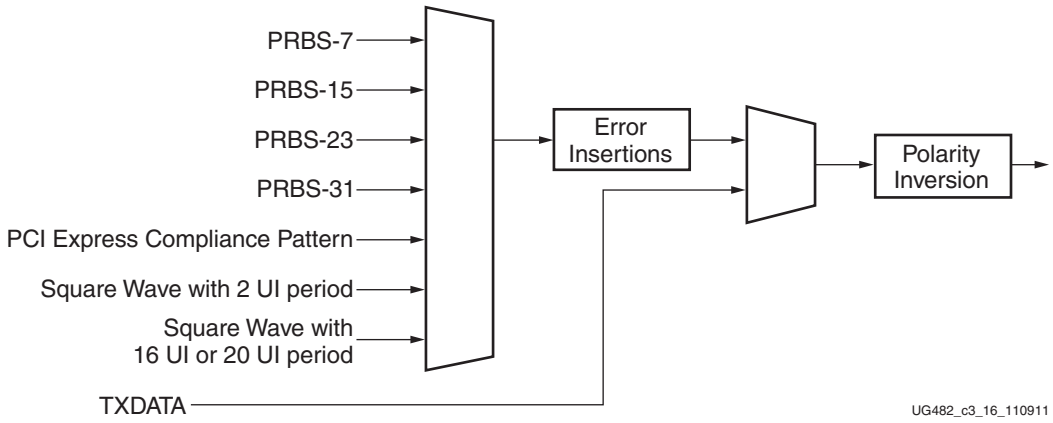


Figure 3-16: **TX Pattern Generator Block**

Ports and Attributes

Table 3-19 defines the pattern generator ports.

Table 3-19: **Pattern Generator Ports**

Port Name	Dir	Clock Domain	Description
TXPRBSSEL[2:0]	In	TXUSRCLK2	Transmitter PRBS generator test pattern control. 000: Standard operation mode (test pattern generation is off) 001: PRBS-7 010: PRBS-15 011: PRBS-23 100: PRBS-31 101: PCI Express compliance pattern. Only works with 20-bit and 40-bit modes 110: Square wave with 2 UI (alternating 0s/1s) 111: Square wave with 16 UI or 20 UI period (based on data width)
TXPRBSFORCEERR	In	TXUSRCLK2	When this port is driven High, errors are forced in the PRBS transmitter. While this port is asserted, the output data pattern contains errors. When TXPRBSSEL is set to 000, this port does not affect TXDATA.

Table 3-20 defines the pattern generator attribute.

Table 3-20: **Pattern Generator Attribute**

Attribute	Type	Description
RXPRBS_ERR_LOOPBACK	1-bit Binary	When set to 1, causes RXPRBSERR bit to be internally looped back to TXPRBSFORCEERR of the same GTP transceiver. This allows synchronous and asynchronous jitter tolerance testing without worrying about data clock domain crossing. When set to 0, TXPRBSFORCEERR forces onto the TX PRBS.

Use Models

The pattern generation and check function are usually used for verifying link quality tests and also for jitter tolerance tests. For link quality testing, the test pattern is chosen by setting TXPRBSSEL and RXPRBSSEL to a non-000 value, and RXPRBS_ERR_LOOPBACK is set to 0 (Figure 3-17). Only the PRBS pattern is recognized by the RX pattern checker.

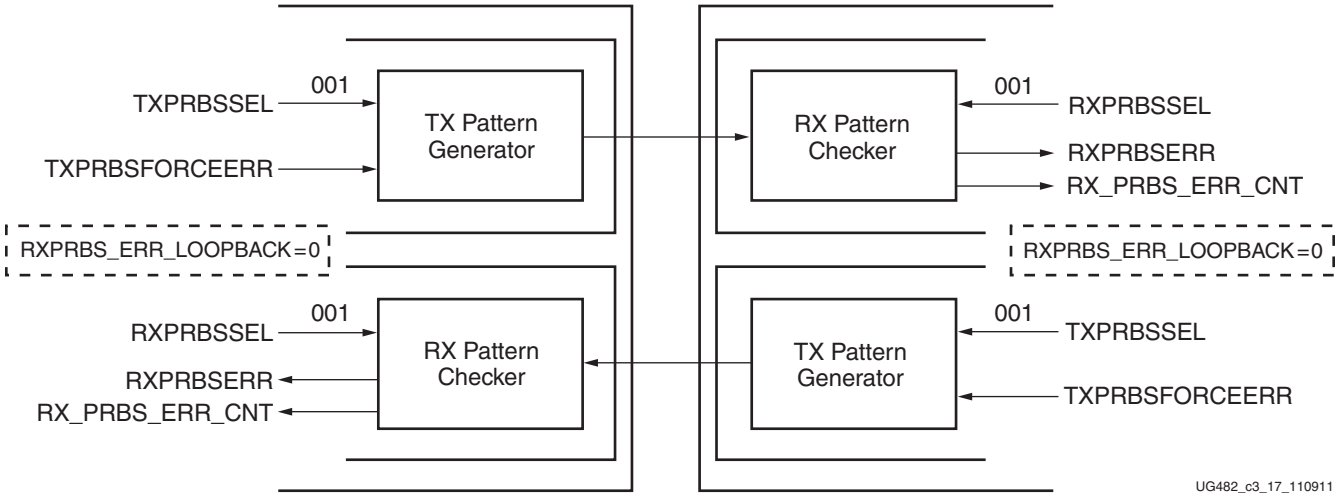


Figure 3-17: **Link Test Mode with a PRBS-7 Pattern**

To calculate accurately the receiver's bit error rate (BER), an external jitter tolerance tester should be used. For the test, the GTP transceiver should loop the received error status back through the transmitter by setting RXPRBS_ERR_LOOPBACK to 1 (Figure 3-18). The same setting should be applied to RXPRBSSEL and TXPRBSSEL.

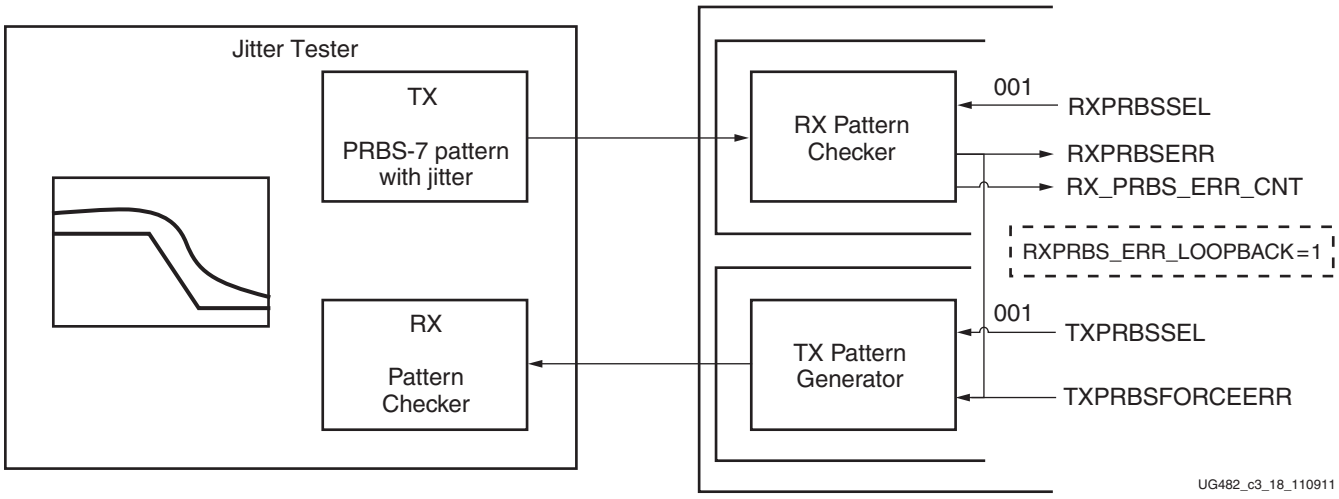


Figure 3-18: **Jitter Tolerance Test Mode with a PRBS-7 Pattern**

TX Polarity Control

Functional Description

If TXP and TXN differential traces are accidentally swapped on the PCB, the differential data transmitted by the GTP transceiver TX is reversed. One solution is to invert the parallel data before serialization and transmission to offset the reversed polarity on the differential pair. The TX polarity control can be accessed through the TXPOLARITY input from the fabric user interface. It is driven High to invert the polarity of outgoing data.

Ports and Attributes

Table 3-21 defines the ports required for TX polarity control.

Table 3-21: **TX Polarity Control Ports**

Port	Dir	Clock Domain	Description
TXPOLARITY	In	TXUSRCLK2	The TXPOLARITY port is used to invert the polarity of outgoing data. 0: Not inverted. TXP is positive, and TXN is negative. 1: Inverted. TXP is negative, and TXN is positive.

Using TX Polarity Control

TXPOLARITY can be tied High if the polarity of TXP and TXN needs to be reversed.

TX Fabric Clock Output Control

Functional Description

The TX Clock Divider Control block has two main components: serial clock divider control and parallel clock divider and selector control. The clock divider and selector details are illustrated in Figure 3-19.

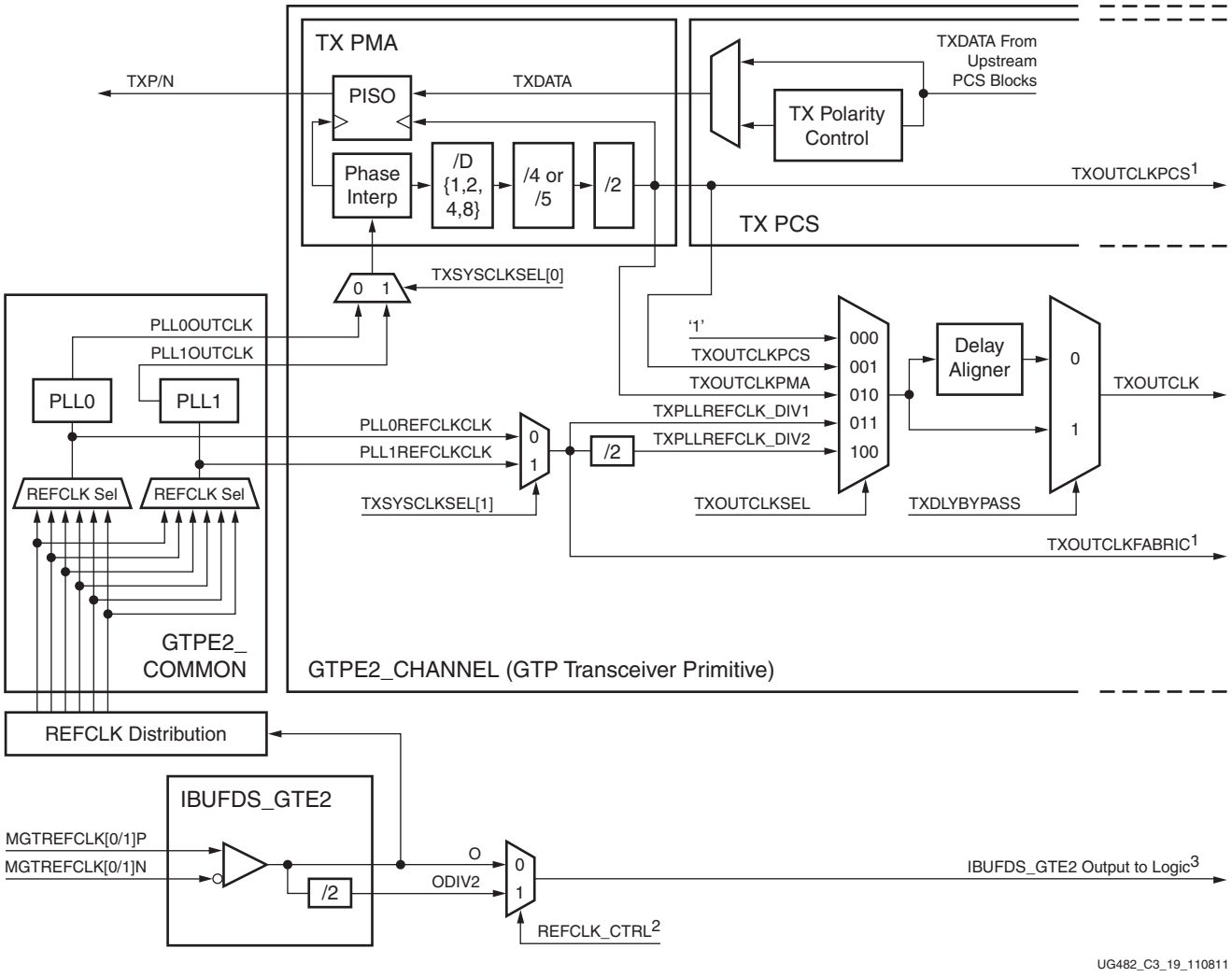


Figure 3-19: **TX Serial and Parallel Clock Divider**

Notes relevant to Figure 3-19:

1. TXOUTCLKPCS and TXOUTCLKFABRIC are redundant outputs. Use TXOUTCLK for new designs.
2. The REFCLK_CTRL option is controlled automatically by software and is not user selectable. The user can only route one of the IBUFDS_GTE2's O or ODIV2 outputs to the FPGA logic.
3. IBUFDS_GTE2 is a redundant output for additional clocking scheme flexibility.

4. The selection of the /4 or /5 divider block is controlled by the TX_DATA_WIDTH attribute from the GTPE2_CHANNEL primitive. /4 is selected when TX_DATA_WIDTH = 16 or 32. /5 is selected when TX_DATA_WIDTH = 20 or 40.
5. For details about placement constraints and restrictions on clocking resources (MMCME2, PLLE2, BUFGCTRL, IBUFDS_GTE2, BUFG, etc.), refer to the UG472, *7 Series FPGAs Clocking Resources User Guide*.

Serial Clock Divider

Each transmitter PMA module has a D divider that divides down the clock from the PLL for lower line rate support. This serial clock divider, D, can be set statically for applications with a fixed line rate or it can be changed dynamically for protocols with multiple line rates.

To use the D divider in fixed line rate applications, the TXOUT_DIV attribute must be set to the appropriate value, and the TXRATE port needs to be tied to `3'b000`. Refer to the Static Setting via Attribute column in Table 3-22 for details.

To use the D divider in multiple line rate applications, the TXRATE port is used to dynamically select the D divider value. The TXOUT_DIV attribute and the TXRATE port must select the same D divider value upon device configuration. After device configuration, the TXRATE is used to dynamically change the D divider value. Refer to the Dynamic Control via Ports column in Table 3-22 for details.

The control for the serial divider is shown in Table 3-22. For details about the line rate range per speed grade, refer to the 7 series FPGAs documentation page for the appropriate data sheet.

Table 3-22: **TX PLL Output Divider Setting**

D Divider Value	Static Setting via Attribute	Dynamic Control via Ports
1	TXOUT_DIV = 1 TXRATE = `3'b000`	TXOUT_DIV = Ignored TXRATE = `3'b001`
2	TXOUT_DIV = 2 TXRATE = `3'b000`	TXOUT_DIV = Ignored TXRATE = `3'b010`
4	TXOUT_DIV = 4 TXRATE = `3'b000`	TXOUT_DIV = Ignored TXRATE = `3'b011`
8	TXOUT_DIV = 8 TXRATE = `3'b000`	TXOUT_DIV = Ignored TXRATE = `3'b100`

Parallel Clock Divider and Selector

The parallel clock outputs from the TX clock divider control block can be used as a fabric logic clock, depending on the line rate requirement.

The recommended clock for the fabric is the TXOUTCLK from one of the GTP transceivers. It is also possible to bring the MGTREFCLK directly to the FPGA logic and use as the fabric clock. TXOUTCLK is preferred for general applications as it has an output delay control used for applications that bypass the TX buffer for output lane deskewing or constant datapath delay. Refer to TX Buffer Bypass, page 65 for more details.

The TXOUTCLKSEL port controls the input selector and allows these clocks to be output via the TXOUTCLK port:

- TXOUTCLKSEL = `3'b001`: The TXOUTCLKPCS path is not recommended for use because it incurs extra delay from the PCS block.
- TXOUTCLKSEL = `3'b010`: TXOUTCLKPMA is the divided down PLL clock after the TX phase interpolator and is used by the TX PCS block. This clock is interrupted when the PLL is reset by one of the related reset signals.
- TXOUTCLKSEL = `3'b011` or `3'b100`: TXPLLREFCLK_DIV1 or TXPLLREFCLK_DIV2 is the input reference clock to PLL0 or PLL1, depending on the TXSYSCLKSEL[1] setting. TXPLLREFCLK is the recommended clock for general usage and is required for the TX buffer bypass mode.

Ports and Attributes

Table 3-23 defines the ports required for TX fabric clock output control.

Table 3-23: **TX Fabric Clock Output Control Ports**

Port	Dir	Clock Domain	Description
TXOUTCLKSEL[2:0]	In	Async	This port controls the multiplexer select signal in Figure 3-19. `3'b000`: Static 1 `3'b001`: TXOUTCLKPCS path `3'b010`: TXOUTCLKPMA path `3'b011`: TXPLLREFCLK_DIV1 path `3'b100`: TXPLLREFCLK_DIV2 path Others: Reserved.
TXRATE[2:0]	In	TXUSRCLK2 (TXRATEMODE makes this port asynchronous)	This port dynamically controls the setting for the TX serial clock divider D (see Table 3-22), and it is used with the TXOUT_DIV attribute. `3'b000`: Use the TXOUT_DIV divider value `3'b001`: Set the D divider to 1 `3'b010`: Set the D divider to 2 `3'b011`: Set the D divider to 4 `3'b100`: Set the D divider to 8
TXOUTCLKFABRIC	Out	Clock	TXOUTCLKFABRIC is a redundant output reserved for testing. TXOUTCLK with TXOUTCLKSEL = `3'b001` should be used instead.
TXOUTCLK	Out	Clock	TXOUTCLK is the recommended clock output to the FPGA logic. The TXOUTCLKSEL port is the input selector for TXOUTCLK and allows the PLL input reference clock to the FPGA logic.
TXOUTCLKPCS	Out	Clock	TXOUTCLKPCS is a redundant output. TXOUTCLK with TXOUTCLKSEL = `3'b011` should be used instead.

Table 3-23: **TX Fabric Clock Output Control Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
TXRATEDONE	Out	TXUSRCLK2	The TXRATEDONE port is asserted High for one TXUSRCLK2 cycle in response to a change on the TXRATE port. The TRANS_TIME_RATE attribute defines the period of time between a change on the TXRATE port and the assertion of TXRATEDONE.
TXDLYBYPASS	In	Async	TX delay alignment bypass: 0: Uses the TX delay alignment circuit. Set to `1'b0` when the TX buffer is bypassed. 1: Bypasses the TX delay alignment circuit. Set to `1'b1` when the TX buffer is used.
TXRATEMODE	In	Async	Determines if TXRATE should be treated as synchronous or asynchronous.

Table 3-24 defines the attributes required for TX fabric clock output control.

Table 3-24: **TX Fabric Clock Output Control Attributes**

Attribute	Type	Description
TRANS_TIME_RATE	8-bit Hex	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. This attribute determines when PHYSTATUS and TXRATEDONE are asserted after a rate change.
TX_EN_RATE_RESET_BUF	Boolean	When set to TRUE, this attribute enables an automatic TX buffer reset during a rate change event initiated by a change in TXRATE.
TXOUT_DIV	Integer	This attribute controls the setting for the TX serial clock divider. This attribute is only valid when TXRATE = `3'b000`. Otherwise the D divider value is controlled by TXRATE. Valid settings are 1, 2, 4, and 8.

TX Phase Interpolator PPM Controller

Functional Description

The TX Phase Interpolator Parts Per Million (TXPIPPM) Controller module provides support for dynamically controlling the TX phase interpolator (TX PI). Located in the TX PCS, its inputs come from the FPGA TX Interface and it outputs to the TX PMA. Applications exist that require fine-tune control of the data in the TX PMA. Control of the output clock from the PLL is achieved through a TX PI, which in turn can be controlled by the TX phase interpolator PPM controller module. The FPGA logic can control the TX PI in

the TX PMA through the use of the TX phase interpolator PPM controller module in the PCS.

Ports and Attributes

Table 3-25 defines the ports required for the TX phase interpolator PPM controller.

Table 3-25: **TX Phase Interpolator PPM Controller Ports**

Port	Dir	Clock Domain	Description
TXPIPPMEN	In	Async	• `1'b0`: Disables the TX Phase Interpolator PPM Controller block. • `1'b1`: Enables the TX Phase Interpolator PPM Controller block.
TXPIPPMOVRDEN	In	TXUSRCLK2	• `1'b0`: Normal operation. • `1'b1`: Enables direct control of the PI code output to the TX PI in the TX PMA. Use with TXPPMOVRD_VALUE[6:0] to program the value of PI code.
TXPIPPMSEL	In	Async	Reserved. This should always be tied to `1'b1`.
TXPIPPMPD	In	Async	• `1'b0`: Does not power down the TX phase interpolator PPM controller module. • `1'b1`: Powers down the TX phase interpolator PPM controller module.
TXPIPPMSTEPSIZE[4:0]	In	Async	TXPIPPMSTEPSIZE[4]: • `1'b1`: Increments PI code • `1'b0`: Decrements PI code TXPIPPMSTEPSIZE[3:0] is the amount to increment or decrement PI code. Its values range from 0 to 15.

Table 3-26 defines the attributes required for the TX phase interpolator PPM controller.

Table 3-26: **TX Phase Interpolator PPM Controller Attributes**

Attribute	Type	Description
TXPI_SYNFREQ_PPM[2:0]	3-bit Binary	When the TX phase interpolator PPM controller module is enabled, this attribute specifies how often PI code to the TX PI is updated. It is updated every TXPI_SYNFREQ_PPM[2:0] cycles. The GT Wizard's default value should be used for this attribute.
TXPI_PPM_CFG[7:0]	8-bit Binary	When TXPIPPMOVRDEN = `1'b1`, the lower 7 bits of this attribute should be programmed to one of the 128 values output to the TX PI. The most significant bit needs to be pulsed (asserted High and then Low) for the TX PI to register the new 7-bit value of TXPI_PPM_CFG[6:0].
TXPI_INVSTROBE_SEL	1-bit Binary	Reserved. Tied to `1'b0`.
TXPI_GREY_SEL	1-bit Binary	Reserved. Tied to `1'b1`.
TXPI_PPMCLK_SEL	String	Reserved. The GT Wizard's default value should be used for this attribute.

TX Phase Interpolator PPM Controller Use Mode

The following describes a sample use case:

- A frequency counter in the fabric determines the lead/lag relationship between the two clocks of interest and increments or decrements (TXPIPPMSTEPSIZE[4]) PI code by a certain step size (TXPIPPMSTEPSIZE[3:0]).
- A sampler and lock detect circuit in the fabric determines when the two clocks are phase aligned. When not phase aligned, the user asserts a signal to the TX phase interpolator PPM controller with the desirable PI code.

This continual phase shifting (fine-tuning) occurs when the lock detect circuit deems the two clocks out of phase and enables the TX phase interpolator PPM controller.

TX Configurable Driver

Functional Description

The GTP transceiver TX driver is a high-speed current-mode differential output buffer. To maximize signal integrity, it includes these features:

- Differential voltage control
- Pre-cursor and post-cursor transmit pre-emphasis
- Calibrated termination resistors

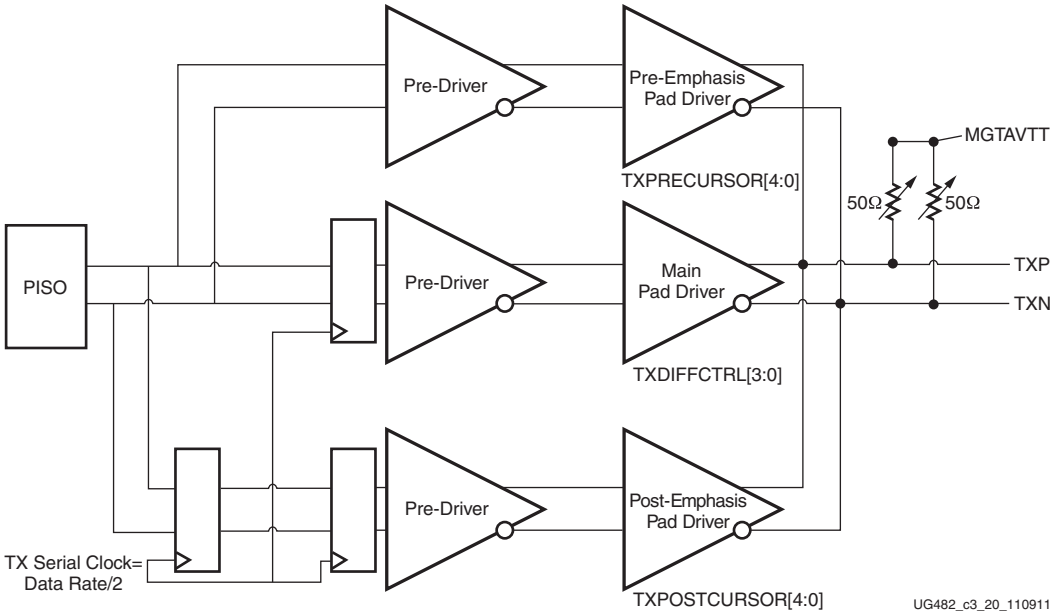


Figure 3-20: **TX Configurable Driver Block Diagram**

Ports and Attributes

Table 3-27 defines the TX configurable driver ports.

Table 3-27: **TX Configurable Driver Ports**

Port	Dir	Clock Domain	Description
TXBUFDIFFCTRL[2:0]	In	Async	Pre-driver Swing Control. The default is `3'b100` (nominal value). Do *not* modify this value.
TXDEEMPH	In	TXUSRCLK2	TX de-emphasis control for PCI Express PIPE 2.0 interface. This signal is mapped internally to TXPOSTCURSOR via attributes. 0: 6.0 dB de-emphasis (TX_DEEMPH_0[4:0] attribute) 1: 3.5 dB de-emphasis (TX_DEEMPH_1[4:0] attribute)

Table 3-27: **TX Configurable Driver Ports *(Cont'd)***

<table>
<tr><th>Port</th><th>Dir</th><th>Clock Domain</th><th>Description</th></tr>
<tr><td>TXDIFFCTRL[3:0]</td><td>In</td><td>Async</td><td>Driver Swing Control. The default is user specified. All listed values are in mV_{PPD}.
<table>
<tr><th>[3:0]</th><th>mV_{PPD}</th></tr>
<tr><td>4'b0000</td><td>250</td></tr>
<tr><td>4'b0001</td><td>300</td></tr>
<tr><td>4'b0010</td><td>350</td></tr>
<tr><td>4'b0011</td><td>400</td></tr>
<tr><td>4'b0100</td><td>450</td></tr>
<tr><td>4'b0101</td><td>500</td></tr>
<tr><td>4'b0110</td><td>550</td></tr>
<tr><td>4'b0111</td><td>600</td></tr>
<tr><td>4'b1000</td><td>650</td></tr>
<tr><td>4'b1001</td><td>700</td></tr>
<tr><td>4'b1010</td><td>750</td></tr>
<tr><td>4'b1011</td><td>800</td></tr>
<tr><td>4'b1100</td><td>850</td></tr>
<tr><td>4'b1101</td><td>900</td></tr>
<tr><td>4'b1110</td><td>950</td></tr>
<tr><td>4'b1111</td><td>1000</td></tr>
</table>
Note: These are preliminary values.</td></tr>
<tr><td>TXELECIDLE</td><td>In</td><td>TXUSRCLK2</td><td>When High, this signal forces GTPTXP and GTPTXN both to Common mode, creating an electrical idle signal.</td></tr>
<tr><td>TXINHIBIT</td><td>In</td><td>TXUSRCLK2</td><td>When High, this signal blocks transmission of TXDATA and forces GTPTXP to 0 and GTPTXN to 1.</td></tr>
<tr><td>TXMAINCURSOR[6:0]</td><td>In</td><td>Async</td><td>Allows the main cursor coefficients to be directly set if the TX_MAINCURSOR_SEL attribute is set to 1'b1.
51 – TXPOSTCURSOR coefficient units – TXPRECURSOR coefficient units $\leq$ TXMAINCURSOR coefficient units $\leq$ 80 –TXPOSTCURSOR coefficient units – TXPRECURSOR coefficient units.</td></tr>
</table>

Table 3-27: **TX Configurable Driver Ports *(Cont'd)***

<table>
<tr><th>Port</th><th>Dir</th><th>Clock Domain</th><th>Description</th></tr>
<tr><td>TXMARGIN[2:0]</td><td>In</td><td>Async</td><td>TX Margin control for PCI Express PIPE 2.0 Interface. These signals are mapped internally to TXDIFFCTRL/TXBUFDIFFCTRL via attributes.
<table>
<tr><th>[2:0]</th><th>Full Range</th><th>Half Range</th><th>Full Range Attribute</th><th>Half Range Attribute</th></tr>
<tr><td>000</td><td>800-1200</td><td>400-1200</td><td>TX_MARGIN_FULL_0</td><td>TX_MARGIN_LOW_0</td></tr>
<tr><td>001</td><td>800-1200</td><td>400-700</td><td>TX_MARGIN_FULL_1</td><td>TX_MARGIN_LOW_1</td></tr>
<tr><td>010</td><td>800-1200</td><td>400-700</td><td>TX_MARGIN_FULL_2</td><td>TX_MARGIN_LOW_2</td></tr>
<tr><td>011</td><td>200-400</td><td>100-200</td><td>TX_MARGIN_FULL_3</td><td>TX_MARGIN_LOW_3</td></tr>
<tr><td>100</td><td>100-200</td><td>100-200</td><td>TX_MARGIN_FULL_4</td><td>TX_MARGIN_LOW_4</td></tr>
<tr><td>101</td><td colspan="4" rowspan="3">default to “DIRECT” mode</td></tr>
<tr><td>110</td></tr>
<tr><td>111</td></tr>
</table></td></tr>
<tr><td>PMARSVDIN1</td><td>In</td><td>Async</td><td>Reserved.</td></tr>
<tr><td>PMARSVDIN0</td><td>In</td><td>Async</td><td>Reserved.</td></tr>
</table>

Table 3-27: **TX Configurable Driver Ports *(Cont'd)***

<table>
<tr><th>Port</th><th>Dir</th><th>Clock Domain</th><th>Description</th></tr>
<tr><td>TXPOSTCURSOR[4:0]</td><td>In</td><td>Async</td><td>Transmitter post-cursor TX pre-emphasis control. The default is user specified. All listed values (dB) are typical.
<table>
<tr><th>[4:0]</th><th>Emphasis (dB)</th><th>Coefficient Units</th></tr>
<tr><td>5'b00000</td><td>0.00</td><td>0</td></tr>
<tr><td>5'b00001</td><td>0.22</td><td>1</td></tr>
<tr><td>5'b00010</td><td>0.45</td><td>2</td></tr>
<tr><td>5'b00011</td><td>0.68</td><td>3</td></tr>
<tr><td>5'b00100</td><td>0.92</td><td>4</td></tr>
<tr><td>5'b00101</td><td>1.16</td><td>5</td></tr>
<tr><td>5'b00110</td><td>1.41</td><td>6</td></tr>
<tr><td>5'b00111</td><td>1.67</td><td>7</td></tr>
<tr><td>5'b01000</td><td>1.94</td><td>8</td></tr>
<tr><td>5'b01001</td><td>2.21</td><td>9</td></tr>
<tr><td>5'b01010</td><td>2.50</td><td>10</td></tr>
<tr><td>5'b01011</td><td>2.79</td><td>11</td></tr>
<tr><td>5'b01100</td><td>3.10</td><td>12</td></tr>
<tr><td>5'b01101</td><td>3.41</td><td>13</td></tr>
<tr><td>5'b01110</td><td>3.74</td><td>14</td></tr>
<tr><td>5'b01111</td><td>4.08</td><td>15</td></tr>
<tr><td>5'b10000</td><td>4.44</td><td>16</td></tr>
<tr><td>5'b10001</td><td>4.81</td><td>17</td></tr>
<tr><td>5'b10010</td><td>5.19</td><td>18</td></tr>
<tr><td>5'b10011</td><td>5.60</td><td>19</td></tr>
<tr><td>5'b10100</td><td>6.02</td><td>20</td></tr>
<tr><td>5'b10101</td><td>6.47</td><td>21</td></tr>
<tr><td>5'b10110</td><td>6.94</td><td>22</td></tr>
<tr><td>5'b10111</td><td>7.43</td><td>23</td></tr>
<tr><td>5'b11000</td><td>7.96</td><td>24</td></tr>
<tr><td>5'b11001</td><td>8.52</td><td>25</td></tr>
<tr><td>5'b11010</td><td>9.12</td><td>26</td></tr>
<tr><td>5'b11011</td><td>9.76</td><td>27</td></tr>
<tr><td>5'b11100</td><td>10.46</td><td>28</td></tr>
<tr><td>5'b11101</td><td>11.21</td><td>29</td></tr>
<tr><td>5'b11110</td><td>12.04</td><td>30</td></tr>
<tr><td>5'b11111</td><td>12.96</td><td>31</td></tr>
</table>
Note: These are preliminary values.</td></tr>
<tr><td>TXPOSTCURSORINV</td><td>In</td><td>Async</td><td>When set to 1'b1, inverts the polarity of the TXPOSTCURSOR coefficient. The default is 1'b0.</td></tr>
</table>

Table 3-27: **TX Configurable Driver Ports *(Cont'd)***

<table>
<tr><th>Port</th><th>Dir</th><th>Clock Domain</th><th>Description</th></tr>
<tr><td>TXPRECURSOR[4:0]</td><td>In</td><td>Async</td><td>Transmitter pre-cursor TX pre-emphasis control. The default is user specified. All listed values (dB) are typical.
<table>
<tr><th>[4:0]</th><th>Emphasis (dB)</th><th>Coefficient Units</th></tr>
<tr><td>5'b00000</td><td>0.00</td><td>0</td></tr>
<tr><td>5'b00001</td><td>0.22</td><td>1</td></tr>
<tr><td>5'b00010</td><td>0.45</td><td>2</td></tr>
<tr><td>5'b00011</td><td>0.68</td><td>3</td></tr>
<tr><td>5'b00100</td><td>0.92</td><td>4</td></tr>
<tr><td>5'b00101</td><td>1.16</td><td>5</td></tr>
<tr><td>5'b00110</td><td>1.41</td><td>6</td></tr>
<tr><td>5'b00111</td><td>1.67</td><td>7</td></tr>
<tr><td>5'b01000</td><td>1.94</td><td>8</td></tr>
<tr><td>5'b01001</td><td>2.21</td><td>9</td></tr>
<tr><td>5'b01010</td><td>2.50</td><td>10</td></tr>
<tr><td>5'b01011</td><td>2.79</td><td>11</td></tr>
<tr><td>5'b01100</td><td>3.10</td><td>12</td></tr>
<tr><td>5'b01101</td><td>3.41</td><td>13</td></tr>
<tr><td>5'b01110</td><td>3.74</td><td>14</td></tr>
<tr><td>5'b01111</td><td>4.08</td><td>15</td></tr>
<tr><td>5'b10000</td><td>4.44</td><td>16</td></tr>
<tr><td>5'b10001</td><td>4.81</td><td>17</td></tr>
<tr><td>5'b10010</td><td>5.19</td><td>18</td></tr>
<tr><td>5'b10011</td><td>5.60</td><td>19</td></tr>
<tr><td>5'b10100</td><td>6.02</td><td>20</td></tr>
<tr><td>5'b10101</td><td>6.02</td><td>20</td></tr>
<tr><td>5'b10110</td><td>6.02</td><td>20</td></tr>
<tr><td>5'b10111</td><td>6.02</td><td>20</td></tr>
<tr><td>5'b11000</td><td>6.02</td><td>20</td></tr>
<tr><td>5'b11001</td><td>6.02</td><td>20</td></tr>
<tr><td>5'b11010</td><td>6.02</td><td>20</td></tr>
<tr><td>5'b11011</td><td>6.02</td><td>20</td></tr>
<tr><td>5'b11100</td><td>6.02</td><td>20</td></tr>
<tr><td>5'b11101</td><td>6.02</td><td>20</td></tr>
<tr><td>5'b11110</td><td>6.02</td><td>20</td></tr>
<tr><td>5'b11111</td><td>6.02</td><td>20</td></tr>
</table>
Note: These are preliminary values.</td></tr>
<tr><td>TXPRECURSORINV</td><td>In</td><td>Async</td><td>When set to 1'b1, inverts the polarity of the TXPRECURSOR coefficient. The default is 1'b0.</td></tr>
</table>

Table 3-27: **TX Configurable Driver Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
GTPTXP GTPTXN	Out (Pad)	TX Serial Clock	GTPTXP and GTPTXN are differential complements of one another forming a differential transmit output pair. These ports represent the pads. The locations of these ports must be constrained (see Implementation, page 20) and brought to the top level of the design.
TXSWING	In	Async	TX swing control for PCI Express PIPE 2.0 Interface. This signal is mapped internally to TXDIFFCTRL/TXBUFDIFFCTRL. 0: Full swing 1: Low swing

Table 3-28 defines the TX configurable driver attributes.

Table 3-28: **TX Configurable Driver Attributes**

Attribute	Type	Description
TX_DEEMPH0[4:0]	5-bit Binary	This attribute has the value of TXPOSTCURSOR[4:0] that has to be mapped when TXDEEMPH = 0. TX_DEEMPH_0[4:0] = TXPOSTCURSOR[4:0]. The default is `5'b10100`. **Note:** These are preliminary values. Do not modify this value.
TX_DEEMPH1[4:0]	5-bit Binary	This attribute has the value of TXPOSTCURSOR[4:0] that has to be mapped when TXDEEMPH = 1. TX_DEEMPH_1[4:0] = TXPOSTCURSOR[4:0]. The default is `5'b01101`. **Note:** These are preliminary values. Do not modify this value.
TX_DRIVE_MODE	String	This attribute selects whether PCI Express PIPE 2.0 ports or TX Drive Control ports control the TX driver. The default is "DIRECT." DIRECT: TXBUFDIFFCRL, TXDIFFCTRL, TXPOSTCURSOR, TXPRECURSOR and TXMAINCURSOR (If TX_MAINCURSOR_SEL = 1'b1) control the TX driver settings. PIPE: TXDEEMPH, TXMARGIN, TXSWING, TXPRECURSOR and TXMAINCURSOR (If TX_MAINCURSOR_SEL = 1'b1) control the TX driver settings.
TX_MAINCURSOR_SEL	1-bit Binary	Allows independent control of the main cursor. `1'b0`: The TXMAINCURSOR coefficient is automatically determined by the equation: 80 – TXPOSTCURSOR coefficient – TXPRECURSOR coefficient `1'b1`: TXMAINCURSOR coefficient can be independently set by the TXMAINCURSOR port within the range specified in the pin description.
TX_MARGIN_FULL_0[6:0]	7-bit Binary	This attribute has the value of TXBUFDIFFCTRL[2:0] and TXDIFFCTRL[3:0] that has to be mapped when TXMARGIN = `000` and TXSWING = 0. TX_MARGIN_FULL_0 = TXBUFDIFFCTRL[2:0], TXDIFFCTRL[3:0]. The default is `7'b1001111` (1000 mV_{PPD} typical). **Note:** These are preliminary values. Do not modify this value.

Table 3-28: **TX Configurable Driver Attributes *(Cont'd)***

Attribute	Type	Description
TX_MARGIN_FULL_1[6:0]	7-bit Binary	This attribute has the value of TXBUFDIFFCTRL[2:0] and TXDIFFCTRL[3:0] that has to be mapped when TXMARGIN = 001 and TXSWING = 0. TX_MARGIN_FULL_0 = TXBUFDIFFCTRL[2:0], TXDIFFCTRL[3:0]. The default is `7'b1001111` (1000 mV_{PPD} typical). **Note:** These are preliminary values. Do not modify this value.
TX_MARGIN_FULL_2[6:0]	7-bit Binary	This attribute has the value of TXBUFDIFFCTRL[2:0] and TXDIFFCTRL[3:0] that has to be mapped when TXMARGIN = 010 and TXSWING = 0. TX_MARGIN_FULL_0 = TXBUFDIFFCTRL[2:0], TXDIFFCTRL[3:0]. The default is `7'b1001111` (1000 mV_{PPD} typical). **Note:** These are preliminary values. Do not modify this value.
TX_MARGIN_FULL_3[6:0]	7-bit Binary	This attribute has the value of TXBUFDIFFCTRL[2:0] and TXDIFFCTRL[3:0] that has to be mapped when TXMARGIN = 0011 and TXSWING = 0. TX_MARGIN_FULL_0 = TXBUFDIFFCTRL[2:0], TXDIFFCTRL[3:0]. The default is `7'b1000001` (300 mV_{PPD} typical). **Note:** These are preliminary values. Do not modify this value.
TX_MARGIN_FULL_4[6:0]	7-bit Binary	This attribute has the value of TXBUFDIFFCTRL[2:0] and TXDIFFCTRL[3:0] that has to be mapped when TXMARGIN = 100 and TXSWING = 0. TX_MARGIN_FULL_0 = TXBUFDIFFCTRL[2:0], TXDIFFCTRL[3:0]. The default is `7'b1000000` (250 mV_{PPD} typical). **Note:** These are preliminary values. Do not modify this value.
TX_MARGIN_LOW_0[6:0]	7-bit Binary	This attribute has the value of TXBUFDIFFCTRL[2:0] and TXDIFFCTRL[3:0] that has to be mapped when TXMARGIN = `000` and TXSWING = 1. TX_MARGIN_FULL_0 = TXBUFDIFFCTRL[2:0], TXDIFFCTRL[3:0]. The default is `7'b1000111` (600 mV_{PPD} typical). **Note:** These are preliminary values. Do not modify this value.
TX_MARGIN_LOW_1[6:0]	7-bit Binary	This attribute has the value of TXBUFDIFFCTRL[2:0] and TXDIFFCTRL[3:0] that has to be mapped when TXMARGIN = `001` and TXSWING = 1. TX_MARGIN_FULL_0 = TXBUFDIFFCTRL[2:0], TXDIFFCTRL[3:0]. The default is `7'b1000110` (550 mV_{PPD} typical). **Note:** These are preliminary values. Do not modify this value.

Table 3-28: **TX Configurable Driver Attributes *(Cont'd)***

Attribute	Type	Description
TX_MARGIN_LOW_2[6:0]	7-bit Binary	This attribute has the value of TXBUFDIFFCTRL[2:0] and TXDIFFCTRL[3:0] that has to be mapped when TXMARGIN = 010 and TXSWING = 1. TX_MARGIN_FULL_0 = TXBUFDIFFCTRL[2:0], TXDIFFCTRL[3:0]. The default is `7'b1000100` (450 mV_{PPD} typical). **Note:** These are preliminary values. Do not modify this value.
TX_MARGIN_LOW_3[6:0]	7-bit Binary	This attribute has the value of TXBUFDIFFCTRL[2:0] and TXDIFFCTRL[3:0] that has to be mapped when TXMARGIN = 0011 and TXSWING = 1. TX_MARGIN_FULL_0 = TXBUFDIFFCTRL[2:0], TXDIFFCTRL[3:0]. The default is `7'b1000000` (250 mV_{PPD} typical). **Note:** These are preliminary values. Do not modify this value.
TX_MARGIN_LOW_4[6:0]	7-bit Binary	This attribute has the value of TXBUFDIFFCTRL[2:0] and TXDIFFCTRL[3:0] that has to be mapped when TXMARGIN = `100` and TXSWING = 1. TX_MARGIN_FULL_0 = TXBUFDIFFCTRL[2:0], TXDIFFCTRL[3:0]. The default is `7'b1000000` (250 mV_{PPD} typical). **Note:** These are preliminary values. Do not modify this value.
TX_PREDRIVER_MODE	1-bit Binary	This is a restricted attribute. Always set this to `1'b0`. Do not modify this attribute.
PMA_RSV5	1-bit Binary	Reserved.

TX Receiver Detect Support for PCI Express Designs

Functional Description

The PCI Express specification includes a feature that allows the transmitter on a given link to detect if a receiver is present. The decision if a receiver is present is based on the rise time of TXP/TXN. Figure 3-21 shows the circuit model used for receive detection. The GTP transceiver must be in the P1 power down state to perform receiver detection. Receiver detection requires a 75 nF to 200 nF external coupling capacitor between the transmitter and receiver, and the receiver must be terminated to GND. The receiver detection sequence starts with the assertion of TXDETECTRX. In response, the receiver detection logic drives TXN and TXP to (V_{DD} - V_{SWING}/2) and then releases them. After a programmable interval, the levels of TXN and TXP are compared with a threshold voltage. At the end of the sequence, the receiver detection status is presented on RXSTATUS when PHYSTATUS is asserted High for one cycle.

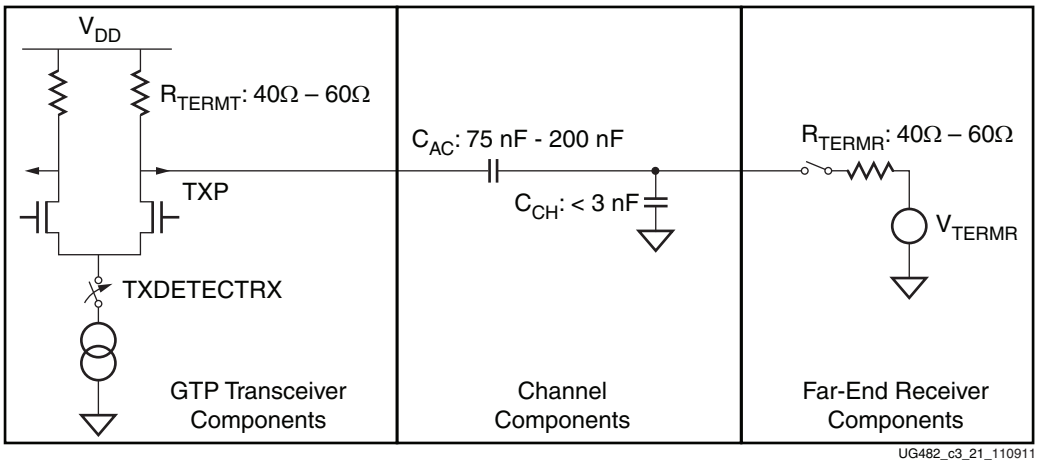


Figure 3-21: **Receiver Detection Circuit Model**

Ports and Attributes

Table 3-29 describes the TX receiver detection ports.

Table 3-29: **TX Receiver Detection Ports**

Port	Dir	Clock Domain	Description
TXDETECTRX	In	TXUSRCLK2	Used to tell the GTP transceiver to begin a receiver detection operation. 0: Normal operation. 1: Receiver detection.
TXPD[1:0]	In	TXUSRCLK2	Power up or down the TX and RX of the GTP transceiver. In PCI Express mode, TXPD and RXPD should be tied to the same source. To perform receiver detection, set these signals to the P1 power saving state. 00: P0 power state for normal operation. 01: P0s power saving state with low recovery time latency. 10: P1 power saving state with longer recovery time latency. 11: P2 power saving state with lowest power.
RXPD[1:0]	In	RXUSRCLK2	
PHYSTATUS	Out	RXUSRCLK2	In PCI Express mode, this signal is used to communicate completion of several GTP transceiver functions, including power management state transitions, rate change, and receiver detection. During receiver detection, this signal is asserted High to indicate receiver detection completion.

Table 3-29: **TX Receiver Detection Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
RXSTATUS[2:0]	Out	RXUSRCLK2	During receiver detection, this signal is read when PHYSTATUS is asserted High. Only these encodings are valid during receiver detection: 000: Receiver not present. 011: Receiver present.

Table 3-30: **TX Receiver Detection Attributes**

Attribute	Type	Description
TX_RXDETECT_CFG	14-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
TX_RXDETECT_REF	3-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

Using the TX Receiver Detection for PCI Express

While in the P1 power state, the GTP transceiver can be instructed to perform a receiver detection operation to determine if there is a receiver at the other end of the link. Figure 3-22 shows an example use mode on how to perform receiver detection in PCI Express mode.

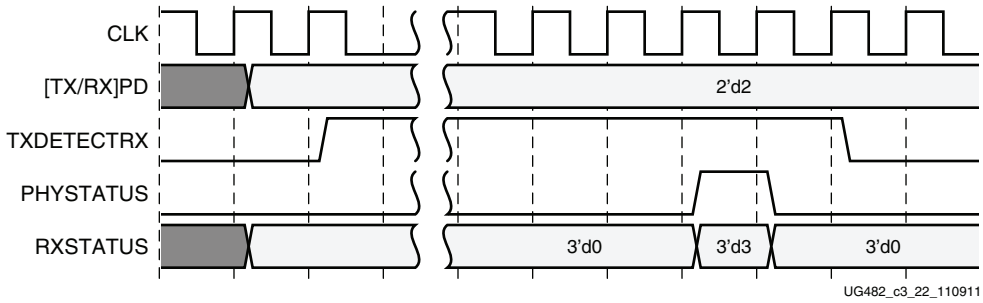


Figure 3-22: **PCI Express Receiver Detection**

Note: Figure 3-22 shows the sequence of events for the receiver present case and is not drawn to scale.

Notes relevant to Figure 3-22:

1. Ensure that the GTP transceiver has successfully entered the P1 power state with [TX/RX]PD = 2'd2 before receiver detection is performed by asserting TXDETECTRX.
2. Wait for PHYSTATUS = 1'd1 to read RXSTATUS on the same PCLK cycle. In PCI Express mode, PCLK is [TX/RX]USRCLK. If RXSTATUS = 3'd3, then the receiver is present. If RXSTATUS = 3'd0, then the receiver is not present. Deassert TXDETECTRX to exit receiver detection.

TX Out-of-Band Signaling

Functional Description

Each GTP transceiver provides support for generating the out-of-band (OOB) sequences described in the Serial ATA (SATA), Serial Attach SCSI (SAS) specification, and beaconing described in the PCI Express specification.

Ports and Attributes

Table 3-31 shows the OOB signaling related ports.

Table 3-31: **TX OOB Signaling Ports**

Port	Dir	Clock Domain	Description
TXCOMFINISH	Out	TXUSRCLK2	Indicates completion of transmission of the last SAS or SATA COM beacon.
TXCOMINIT	In	TXUSRCLK2	Initiates transmission of the COMINIT sequence for SATA/SAS.
TXCOMSAS	In	TXUSRCLK2	Initiates transmission of the COMSAS sequence for SAS.
TXCOMWAKE	In	TXUSRCLK2	Initiates transmission of the COMWAKE sequence for SATA/SAS.
TXPDELECIDLEMODE	In	TXUSRCLK2	Determines if TXELECIDLE and TXPD should be treated as synchronous or asynchronous signals. Enables compliance during cold and warm PCI Express resets. 1: Asynchronous 0: Synchronous
TXPD[1:0]	In	TXUSRCLK2	Powers down the TX lane according to the PCI Express encoding. 00: P0 normal operation 01: P0s low recovery time power down 10: P1 longer recovery time, RecDet still on 11: P2 lowest power state. Attributes can control the transition times between these power down mode (PD_TRANS_TIME_FROM_P2, PD_TRANS_TIME_NONE_P2, PD_TRANS_TIME_TO_P2).

Table 3-32 shows the OOB signaling attributes.

Table 3-32: **TX OOB Signaling Attributes**

Attribute	Type	Description
SATA_CPLL_CFG	2-bit Binary	Configuration bits for the PLL setting related to SAS/SATA.
SATA_BURST_SEQ_LEN	4-bit Binary	Number of bursts in a COM sequence for SAS/SATA.
TXOOB_CFG	1-bit Binary	TX OOB configuration.

Chapter 4

Receiver

RX Overview

Functional Description

This section shows how to configure and use each of the functional blocks inside the receiver (RX). Each GTP transceiver includes an independent receiver, made up of a PCS and a PMA. Figure 4-1 shows the blocks of the GTP transceiver RX. High-speed serial data flows from traces on the board into the PMA of the GTP transceiver RX, into the PCS, and finally into the FPGA logic. Refer to Figure 2-9, page 35 for the description of the channel clocking architecture, which provides clocks to the RX and TX clock dividers.

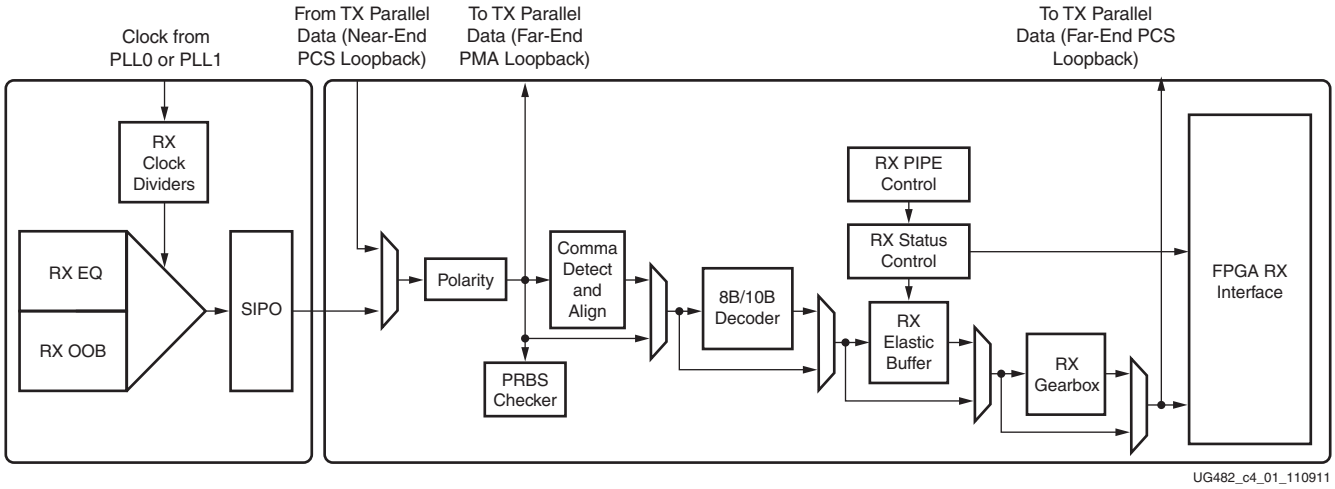


Figure 4-1: **GTP Transceiver RX Block Diagram**

The key elements within the GTP transceiver RX are:

1. RX Analog Front End, page 94
2. RX Out-of-Band Signaling, page 97
3. RX Equalizer, page 99
4. RX CDR, page 101
5. RX Fabric Clock Output Control, page 105
6. RX Margin Analysis, page 108
7. RX Polarity Control, page 116
8. RX Pattern Checker, page 117

RX Analog Front End

Functional Description

The RX analog front end (AFE) is a high-speed current-mode input differential buffer (see Figure 4-1). It has these features:

- Configurable RX termination voltage
- Calibrated termination resistors

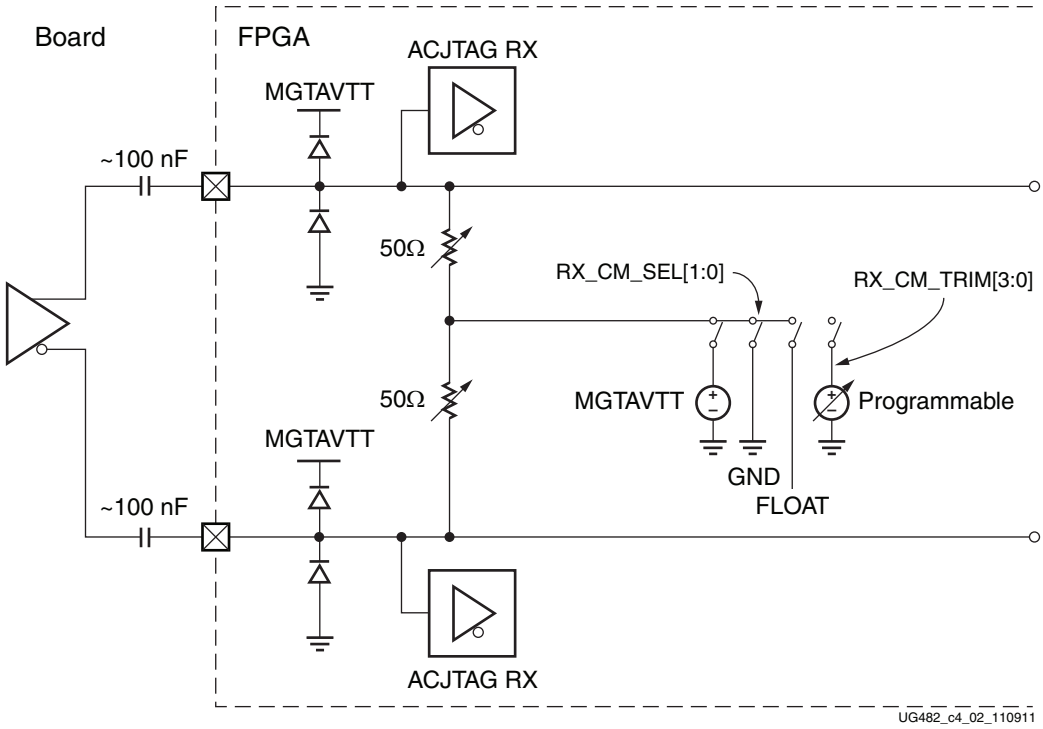


Figure 4-2: **RX Analog Front End**

Ports and Attributes

Table 4-1 defines the RX AFE ports.

Table 4-1: **RX AFE Ports**

Port	Dir	Clock Domain	Description
GTPRXN, GTPRXP	In (Pad)	RX Serial Clock	GTPRXN and GTPRXP are differential complements of one another forming a differential receiver input pair. These ports represent pads. The location of these ports must be constrained (see Implementation, page 20) and brought to the top level of the design.
PMARSVDOUT1	Out	Async	Reserved.
PMARSVDOUT0	Out	Async	Reserved.
PMARSVDIN2	In	Async	Reserved.

Table 4-2 defines the RX AFE attributes.

Table 4-2: **RX AFE Attributes**

Attribute	Type	Description
RX_CM_SEL [1:0]	2-bit Binary	Controls the mode for the RX termination voltage. `2'b00` - AVTT `2'b01` - GND `2'b10` - Floating `2'b11` - Programmable
RX_CM_TRIM [3:0]	4-bit Binary	Controls the Common mode in Programmable mode. `4'b0000` – 100 mV `4'b0001` – 200 mV `4'b0010` – 250 mV `4'b0011` – 300 mV `4'b0100` – 350 mV `4'b0101` – 400 mV `4'b0110` – 500 mV `4'b0111` – 550 mV `4'b1000` – 600 mV `4'b1001` – 700 mV `4'b1010` – 800 mV `4'b1011` – 850 mV `4'b1100` – 900 mV `4'b1101` – 950 mV `4'b1110` – 1000 mV `4'b1111` – 1100 mV

Table 4-2: **RX AFE Attributes *(Cont'd)***

Attribute	Type	Description
TERM_RCAL_CFG[14:0]	15-bit Binary	Controls the internal termination calibration circuit. This feature is intended for internal use only.
TERM_RCAL_OVRD	3-bit Binary	Selects whether the external 100Ω precision resistor connected to the MGTRREF pin or an override value is used, as defined by TERM_RCAL_CFG [14:0]. This feature is intended for internal use only.

Use Modes—RX Termination

Table 4-3: **Use Mode 1—RX Termination**

Use Mode	External AC Coupling	Term Voltage	Max Swing mV_{DPP}	Suggested Protocols and Usage Notes
1	On	Gnd	1200	Protocol: • PCIe® Attribute Settings: • RX_CM_SEL[1:0] = `2'b01`

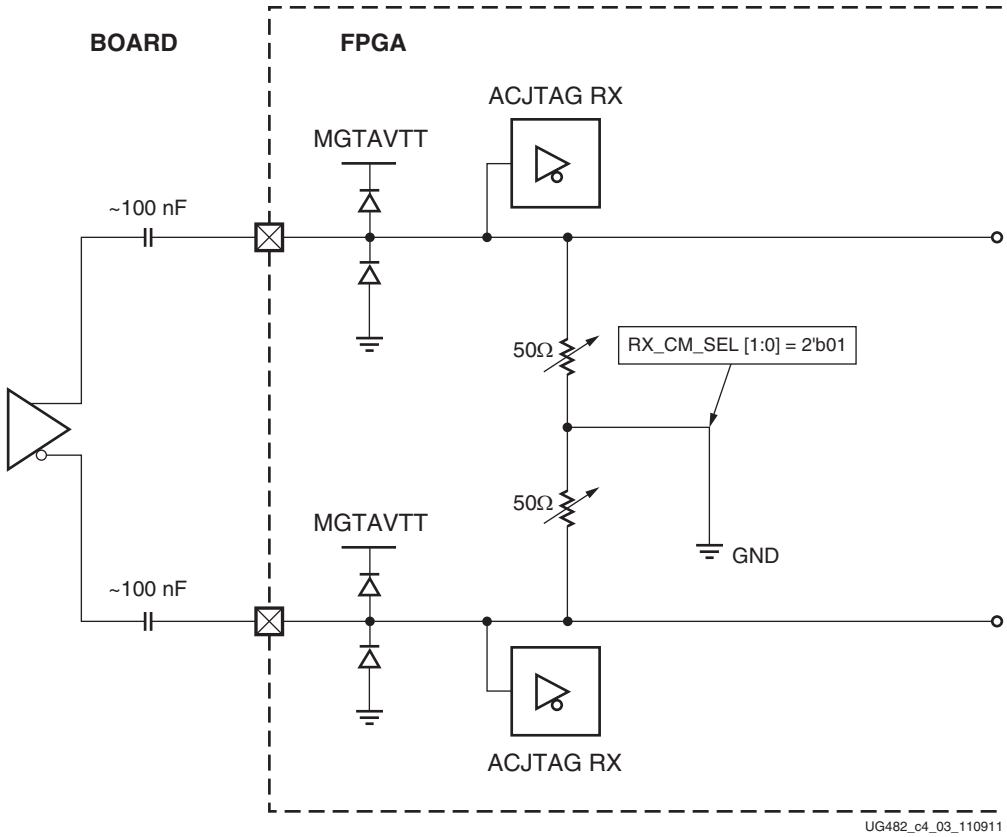


Figure 4-3: **Use Mode 1**

Table 4-4: **Use Mode 2—RX Termination**

Use Mode	External AC Coupling	Term Voltage	Max Swing mV_{DPP}	Suggested Protocols and Usage Notes
2	On	AVTT	1200	Protocol: • Backplane • CEI-6 (1200 mV_{DPP}) • Wireless • Serial RapidIO • DisplayPort (0.4/0.6/0.8V option) Attribute Settings: • RX_CM_SEL[1:0] = `2'b00`

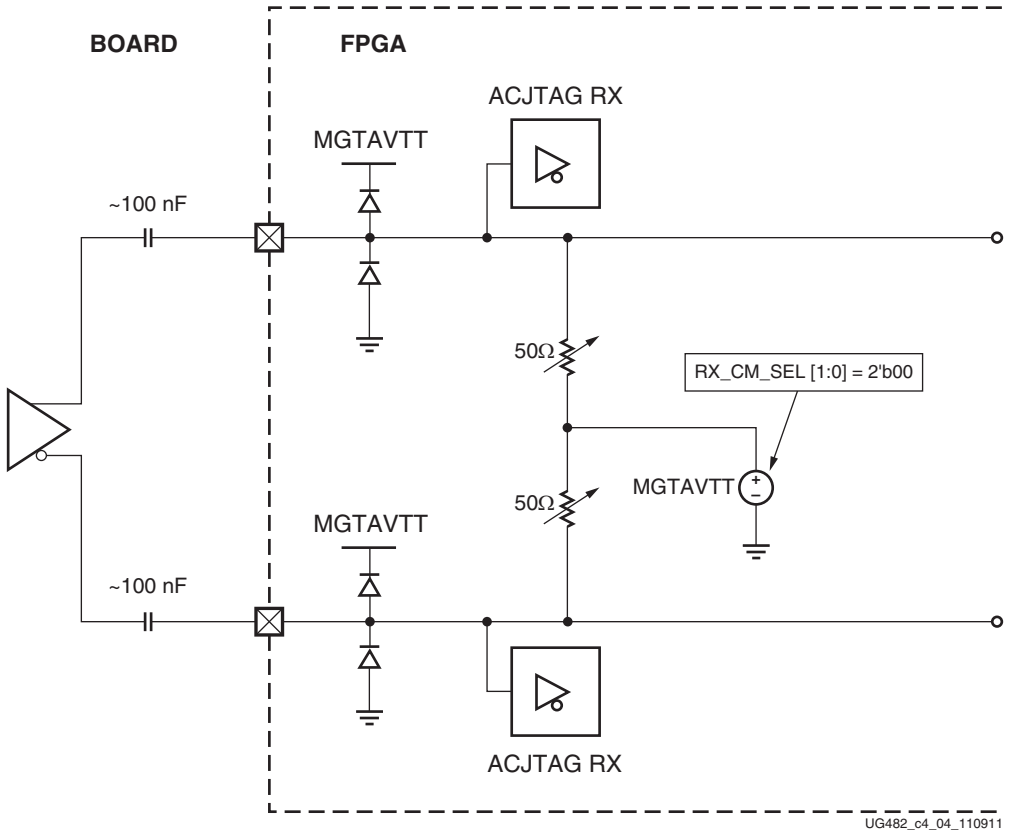


Figure 4-4: **Use Mode 2**

RX Out-of-Band Signaling

Functional Description

The GTP transceiver receiver provides support for decoding the out-of-band (OOB) sequences described in the Serial ATA (SATA) and Serial Attach SCSI (SAS) specifications and supports beaconing described in the PCI Express specification. GTP transceiver receiver support for SATA/SAS OOB signaling consists of the analog circuitry required to

decode the OOB signal state and state machines to decode bursts of OOB signals for SATA/SAS COM sequences.

The GTP transceiver receiver also supports beacons that are PCI Express compliant by using interface signals defined in the *PHY Interface for the PCI Express (PIPE) Specification*. The FPGA logic decodes the beacon sequence.

Ports and Attributes

Table 4-5 defines the OOB signaling related ports.

Table 4-5: **RX OOB Signaling Ports**

Port	Dir	Clock Domain	Description
RXOOBRESET	In	Async	Reserved. Tie to GND.
RXCOMINITDET	Out	RXUSRCLK2	Indicates reception of the COMINIT sequence for SATA/SAS.
RXCOMSASDET	Out	RXUSRCLK2	Indicates reception of the COMSAS sequence for SAS.
RXCOMWAKEDET	Out	RXUSRCLK2	Indicates reception of the COMWAKE sequence for SATA/SAS.
RXELECIDLE	Out	RXUSRCLK2	Status output that indicates OOB signal detection for PCI Express and SATA. 0 = Activity is seen on the receiver 1 = No activity is seen
RXELECIDLEMODE[1:0]	In	Async	Input signal to set the mode of the RXELECIDLE. `2'b00` = Default setting. RXELECIDLE behaves the same as described. `2'b11` = RXELECIDLE is gated with output equal to `1'b0`.

Table 4-6 defines the OOB signaling attributes.

Table 4-6: **RX OOB Signaling Attributes**

Attribute	Type	Description
RXOOB_CFG	7-bit Binary	OOB block configuration.
RXOOB_CLK_CFG	String	Selects which clock to use for OOB.
SATA_BURST_VAL	3-bit Binary	Number of bursts to declare a COM match for SAS/SATA.
SATA_EIDLE_VAL	3-bit Binary	Number of idles to declare a COM match for SAS/SATA.
SAS_MIN_COM	Integer	1-63. Lower bound on activity burst for COM FSM for SAS/SATA.
SATA_MIN_INIT	Integer	1-63. Lower bound on idle count during COMSAS for SAS.

Table 4-6: **RX OOB Signaling Attributes *(Cont'd)***

Attribute	Type	Description
SATA_MIN_WAKE	Integer	1-63. Lower bound on idle count during COMINIT/COMRESET for SAS/SATA.
SATA_MAX_BURST	Integer	1-63. Upper bound on activity burst for COM FSM for SAS/SATA.
SAS_MAX_COM	Integer	1-127. Upper bound on idle count during COMSAS for SAS.
SATA_MAX_INIT	Integer	1-63. Upper bound on idle count during COMINIT/COMRESET for SAS/SATA.
SATA_MAX_WAKE	Integer	1-63. Upper bound on idle count during COMWAKE for SAS/SATA.

RX Equalizer

Functional Description

The GTP transceiver receiver has a power-efficient adaptive continuous time linear equalizer (CTLE) to compensate for signal distortion due to high-frequency attenuation in the physical channel. To maintain parity with the 7 series FPGAs GTX and GTH transceivers, the CTLE is referred to as the low-power mode (LPM).

The LPM mode (see Figure 4-5) has adaptive low and high frequency boosts.

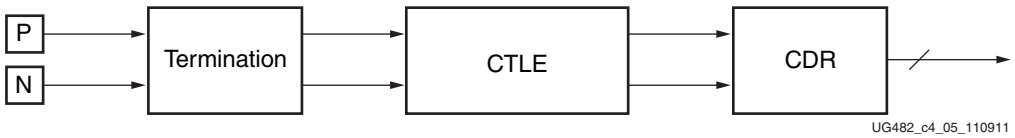


Figure 4-5: **RX Equalization Block Diagram**

Ports and Attributes

Table 4-7 defines the RX equalizer ports.

Table 4-7: **RX Equalizer Ports**

Port	Dir	Clock Domain	Description
RXLPMRESET	In	Async	Resets the LPM circuitry.
RXLPMHFHOLD	In	Async	When set to `1'b1`, the current value of the high-frequency boost is held. When set to `1'b0`, the high-frequency boost is adapted.

Table 4-7: **RX Equalizer Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
RXLPMHFOVRDEN	In	Async	When set to `1'b1`, the high-frequency boost is controlled by the RXLPM_HF_CFG attribute. When set to `1'b0`, the high-frequency boost is controlled by the RXLPMHFHOLD signal.
RXLPMLFHOLD	In	Async	When set to `1'b1`, the current value of the low-frequency boost is held. When set to `1'b0`, the low-frequency boost is adapted.
RXLPMLFKLOVRDEN	In	Async	When set to `1'b1`, the low-frequency boost is controlled by the RXLPM_LF_CFG attribute. When set to `1'b0`, the low-frequency boost is controlled by the RXLPMLFHOLD signal.

Table 4-8 defines the RX equalizer attributes.

Table 4-8: **RX Equalizer Attributes**

Attribute	Type	Description
ADAPT_CFG0	20-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RX_LPM_HOLD_DURING_EIDLE	1-bit Binary	When set to `1'b1`, all adapted values for the LPM and offset cancellation are held when the GTP transceiver RX is in electrical idle and restored after electrical idle is exited.
RXLPMRESET_TIME	7-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RXLPM_CFG	1-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RXLPM_CFG1	4-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

Table 4-8: **RX Equalizer Attributes *(Cont'd)***

Attribute	Type	Description
RXLPM_HF_CFG2	4-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RXLPM_HF_CFG	14-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RXLPM_BIAS_STARTUP_DISABLE	1-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RXLPM_LF_CFG	18-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

RX CDR

Functional Description

The RX clock data recovery (CDR) circuit in each GTPE2_CHANNEL transceiver extracts the recovered clock and data from an incoming data stream. Figure 4-6 illustrates the architecture of the CDR block. Clock paths are shown with dotted lines for clarity.

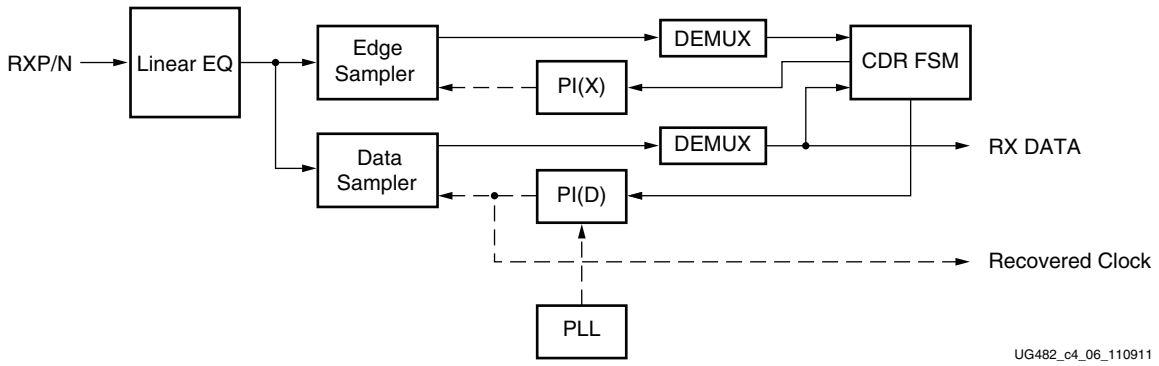


Figure 4-6: **CDR Detail**

The GTPE2_CHANNEL transceiver employs phase rotator CDR architecture. Incoming data first goes through receiver equalization stages. The equalized data is captured by an edge and a data sampler. The data captured by the data sampler is fed to the CDR state machine and the downstream transceiver blocks.

The CDR state machine uses the data from both the edge and data samplers to determine the phase of the incoming data stream and to control the phase interpolators (PIs). The phase for the edge sampler is locked to the transition region of the data stream while the phase of the data sampler is positioned in the middle of the data eye.

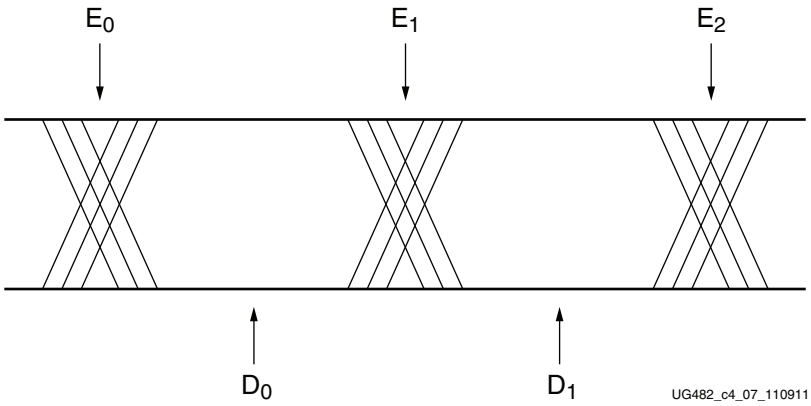


Figure 4-7: **CDR Sampler Positions**

The PLL0 or PLL1 provides a base clock to the phase interpolator. The phase interpolator in turn produces fine, evenly spaced sampling phases to allow the CDR state machine to have fine phase control. The CDR state machine can track incoming data streams that can have a frequency offset from the local PLL reference clock.

The GTPE2_CHANNEL transceiver has a CDR lock indicator. The detection of the CDR lock is based on the PI's phase gap in a certain period of cycles. When the PI's phase gap becomes within a specified value in a certain period of time, the CDR is considered to be locked and the CDR lock indicator is asserted. This is a coarse indication of CDR lock. To ensure the locked status, it is recommended to verify incoming data as well.

One method for detecting CDR lock is finding known data in the incoming data stream (for example, commas or A1/A2 framing characters). In general, several consecutive known data patterns must be received without error to indicate a CDR lock.

Ports and Attributes

Table 4-9 defines the CDR ports.

Table 4-9: **CDR Ports**

Port	Dir	Clock Domain	Description
RXCDRFREQRESET	In	Async	CDR frequency detector reset.
RXCDRHOLD	In	Async	Hold the CDR control loop frozen.
RXCDROVRDEN	In	Async	Reserved.
RXCDRRESET	In	Async	CDR phase detector reset.
RXCDRRESETRSV	In	Async	Reserved.

Table 4-9: **CDR Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
RXRATE[2:0]	In	RXUSRCLK2 (RXRATEMODE makes this port asynchronous)	This port dynamically controls the setting for the RX serial clock divider D (see Table 4-11) and it is used with RXOUT_DIV attribute. `3'b000`: Use RXOUT_DIV divider value `3'b001`: Set D divider to 1 `3'b010`: Set D divider to 2 `3'b011`: Set D divider to 4 `3'b100`: Set D divider to 8 RXBUF_RESET_ON_RATE_CHANGE attribute enable optional automatic reset.
RXCDRLOCK	Out	Async	Indicator for CDR locking.
RXOSHOLD	In	Async	When set to `1'b1`, the current value of the offset cancellation is held. When set to `1'b0`, the offset cancellation is adapted.
RXOSOVRDEN	In	Async	When set to `1'b1`, the Offset Cancellation is controlled by the RX_OS_CFG attribute. When set to `1'b0`, the AGC is controlled by the RXOSHOLD signal.
RXOSCALRESET	In	Async	Reserved. Tied Low.
RXOSINTPD	In	Async	Reserved. Tied Low.
RXOSINTCFG[3:0]	In	Async	Reserved. Tied Low.
RXOSINTD0[3:0]	In	Async	Reserved. Tied Low.
RXOSINTOVRDEN	In	Async	Reserved. Tied Low.
RXOSINTSTROBE	In	Async	Reserved. Tied Low.
RXOSINTHOLD	In	Async	Reserved. Tied Low.
RXOSINTTESTOVRDEN	In	Async	Reserved. Tied Low.
RXOSINTSTARTED	Out	Async	Reserved.
RXOSINTSTROBESTARTED	Out	Async	Reserved.
RXOSINTDONE	Out	Async	Reserved.

Table 4-10 defines the CDR related attributes.

Table 4-10: **CDR Attributes**

Attribute	Type	Description
CFOK_CFG	43-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
CFOK_CFG2	7-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
CFOK_CFG3	7-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RXCDR_CFG	72-bit Hex	CDR configuration. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RXCDR_LOCK_CFG	6-bit Binary	CDR Lock loop configuration. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RXCDR_HOLD_DURING_EIDLE	Binary	Enables the CDR to hold its internal states during an optional PCI Express reset sequence during electrical idle.
RXCDR_FR_RESET_ON_EIDLE	Binary	Enables automatic reset of the CDR frequency during the optional PCI Express reset sequence during electrical idle.
RXCDR_PH_RESET_ON_EIDLE	Binary	Enables automatic reset of the CDR phase during the optional PCI Express reset sequence during electrical idle.
RX_OS_CFG[12:0]	13-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

RX Fabric Clock Output Control

Functional Description

The RX clock divider control block has two main components: serial clock divider control and parallel clock divider and selector control. The clock divider and selector details are illustrated in Figure 4-8.

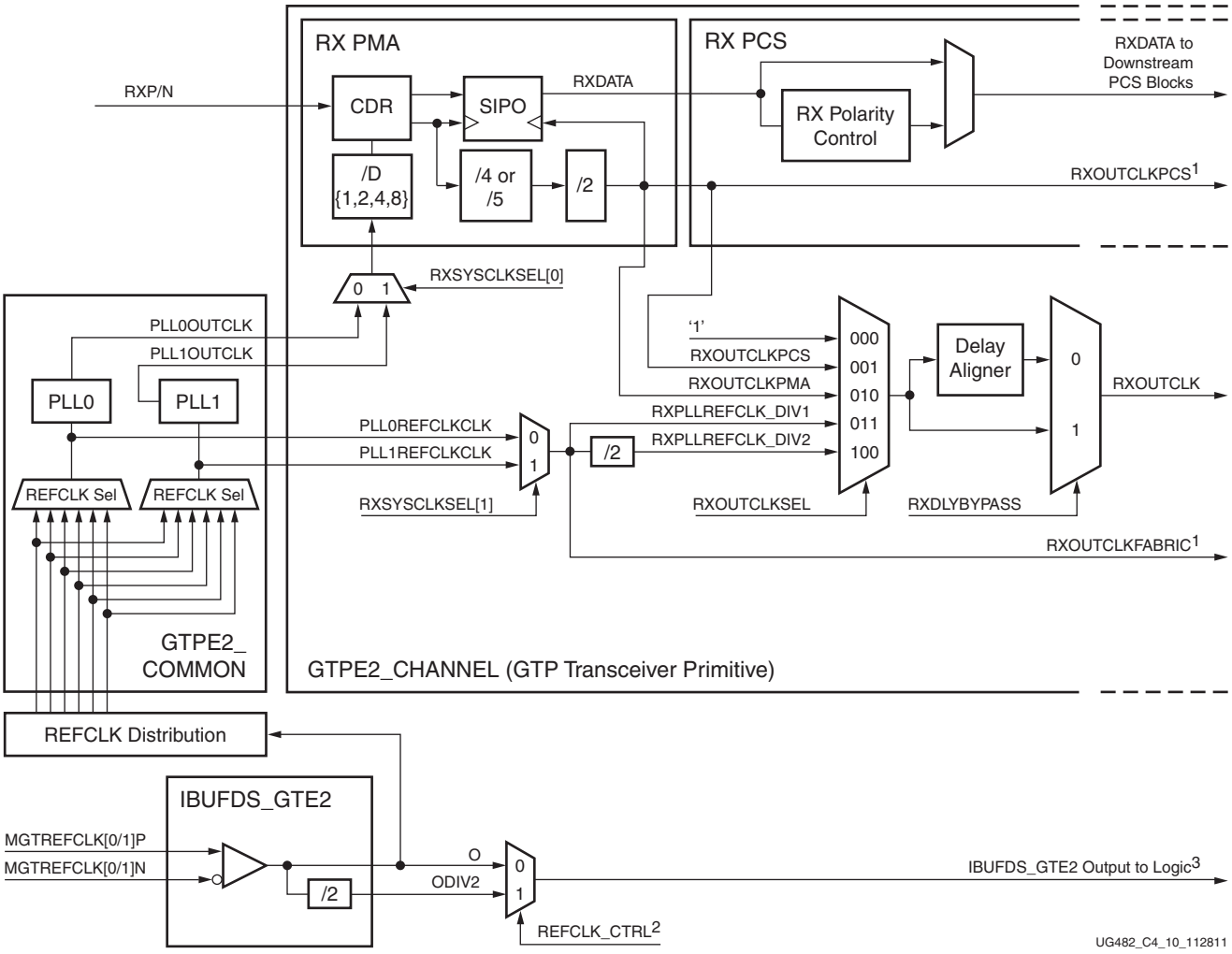


Figure 4-8: **RX Serial and Parallel Clock Divider**

Note relevant to Figure 4-8:

1. RXOUTCLKPCS and RXOUTCLKFABRIC are redundant outputs. RXOUTCLK should be used for new designs.
2. The REFCLK_CTRL option is controlled automatically by software and is not user selectable. The user can only route one of IBUFDS_GTE2's O or ODIV2 outputs to the FPGA logic.
3. IBUFDS_GTE2 is a redundant output for additional clocking scheme flexibility.

4. The selection of the /4 or /5 divider block is controlled by the RX_DATA_WIDTH attribute from the GTPE2_CHANNEL primitive. /4 is selected when RX_DATA_WIDTH = 16 or 32. /5 is selected when RX_DATA_WIDTH = 20 or 40.
5. For details about placement constraints and restrictions on clocking resources (MMCME2, PLLE2, IBUFDS_GTE2, BUFG, etc.), refer to UG472, *7 Series FPGAs Clocking Resources User Guide*.

Serial Clock Divider

Each transmitter PMA module has a D divider that divides down the clock from the PLL for lower line rate support. This serial clock divider, D, can be set statically for applications with a fixed line rate or it can be changed dynamically for protocols with multiple line rates. The control for the serial divider is described in Table 4-11. For details about the line rate range per speed grade, refer to the 7 series FPGAs documentation page for the appropriate data sheet.

To use the D divider in fixed line rate applications, the RXOUT_DIV attribute must be set to the appropriate value, and the RXRATE port needs to be tied to `3'b000`. Refer to the Static Setting via Attribute column in Table 4-11 for details.

To use the D divider in multiple line rate applications, the RXRATE port is used to dynamically select the D divider value. The RXOUT_DIV attribute and the RXRATE port must select the same D divider value upon device configuration. After device configuration, the RXRATE is used to dynamically change the D divider value. Refer to the Dynamic Control via Ports column in Table 4-11 for details.

Table 4-11: **RX PLL Output Divider Setting**

D Divider Value	Static Setting via Attribute	Dynamic Control via Ports
1	RXOUT_DIV = 1 RXRATE = `3'b000`	RXOUT_DIV = Ignored RXRATE = `3'b001`
2	RXOUT_DIV = 2 RXRATE = `3'b000`	RXOUT_DIV = Ignored RXRATE = `3'b010`
4	RXOUT_DIV = 4 RXRATE = `3'b000`	RXOUT_DIV = Ignored RXRATE = `3'b011`
8	RXOUT_DIV = 8 RXRATE = `3'b000`	RXOUT_DIV = Ignored RXRATE = `3'b100`

Parallel Clock Divider and Selector

The parallel clock outputs from the RX clock divider control block can be used as a fabric logic clock depending on the line rate and protocol requirements.

The recommended clock for the FPGA logic is the RXOUTCLK from one of the GTP transceivers. It is also possible to bring the MGTREFCLK directly to the fabric and use as the fabric clock. RXOUTCLK is preferred for general applications because it has an output delay control used for applications that bypass the RX buffer for constant datapath delay. Refer to RX Buffer Bypass, page 134 for more details.

The RXOUTCLKSEL port controls the input selector and allows these clocks to be output via RXOUTCLK port:

- RXOUTCLKSEL = `3'b001`: RXOUTCLKPCS path is not recommended to be used as it incurs extra delay from the PCS block.

- RXOUTCLKSEL = `3'b010`: RXOUTCLKPMA is the recovered clock that can be brought out to the FPGA logic. The recovered clock is used by protocols that do not have a clock compensation mechanism and require to use a clock synchronous to the data (the recovered clock), to clock the downstream fabric logic. It is also used by the RX PCS block. This clock is interrupted when the PLL or CDR is reset by one of the related reset signals.
- RXOUTCLKSEL = `3'b011` or `3'b100`: RXPLLREFCLK_DIV1 or RXPLLREFCLK_DIV2 is the input reference clock to the PLL0 or PLL1 depending on the RXSYSCLKSEL[1] setting. For usages that do not require outputting a recovered clock to the fabric, RXPLLREFCLK_DIV1 or RXPLLREFCLK_DIV2 can be used as the system clock. However, TXOUTCLK is usually used as system clock.

Ports and Attributes

Table 4-12 defines the ports required for RX fabric clock output control.

Table 4-12: **RX Fabric Clock Output Control Ports**

Port	Dir	Clock Domain	Description
RXOUTCLKSEL[2:0]	In	Async	This port controls the multiplexer select signal in Figure 4-8. `3'b000`: Static 1 `3'b001`: RXOUTCLKPCS path `3'b010`: RXOUTCLKPMA path `3'b011`: RXPLLREFCLK_DIV1 path `3'b100`: RXPLLREFCLK_DIV2 path Others: Reserved.
RXRATE[2:0]	In	RXUSRCLK2 (RXRATEMODE makes this port asynchronous)	This port dynamically controls the setting for the RX serial clock divider D (see Table 4-11) and it is used with RXOUT_DIV attribute. `3'b000`: Use RXOUT_DIV divider value `3'b001`: Set D divider to 1 `3'b010`: Set D divider to 2 `3'b011`: Set D divider to 4 `3'b100`: Set D divider to 8
RXOUTCLKFABRIC	Out	Clock	RXOUTCLKFABRIC is a redundant output reserved for testing. RXOUTCLK with RXOUTCLKSEL = `3'b001` should be used instead.
RXOUTCLK	Out	Clock	RXOUTCLK is the recommended clock output to the FPGA logic. The RXOUTCLKSEL port is the input selector for RXOUTCLK and allows the PLL input reference clock to the FPGA logic.

Table 4-12: **RX Fabric Clock Output Control Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
RXOUTCLKPCS	Out	Clock	RXOUTCLKPCS is a redundant output. RXOUTCLK with RXOUTCLKSEL = `3'b011` should be used instead.
RXRATEDONE	Out	RXUSRCLK2	The RXRATEDONE port is asserted High for one RXUSRCLK2 cycle in response to a change on the RXRATE port. The TRANS_TIME_RATE attribute defines the period of time between a change on the RXRATE port and the assertion of RXRATEDONE.
RXRATEMODE	In	Async	Determines if RXRATE should be treated as synchronous or asynchronous.
RXDLYBYPASS	In	Async	RX delay alignment bypass: 0: Uses the RX delay alignment circuit. Set to `1'b0` when the RX buffer is bypassed. 1: Bypasses the RX delay alignment circuit. Set to `1` when the RX buffer is used.

Table 4-13 defines the attributes required for RX fabric clock output control.

Table 4-13: **RX Fabric Clock Output Control Attributes**

Attribute	Type	Description
TRANS_TIME_RATE	8-bit Hex	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used. This attribute determines when PHYSTATUS and RXRATEDONE are asserted after a rate change.
RX_EN_RATE_RESET_BUF	Boolean	When set to TRUE, this attribute enables automatic RX buffer reset during a rate change event initiated by a change in RXRATE.
RXOUT_DIV	Integer	This attribute controls the setting for the RX serial clock divider. This attribute is only valid when RXRATE = `3'b000`. Otherwise the D divider value is controlled by RXRATE. Valid settings are 1, 2, 4, and 8.

RX Margin Analysis

Functional Description

As line rates and channel attenuation increase, the receiver equalizers are more often enabled to overcome channel attenuation. This poses a challenge to system bring-up

because the quality of the link cannot be determined by measuring the far-end eye opening at the receiver pins. At high line rates, the received eye measured on the printed circuit board can appear to be completely closed even though the internal eye after the receiver equalizer is open.

The 7 series FPGAs GTP transceivers RX eye scan provides a mechanism to measure and visualize the receiver eye margin after the equalizer. Additional use modes enable several other methods to determine and diagnose the effects of equalization settings.

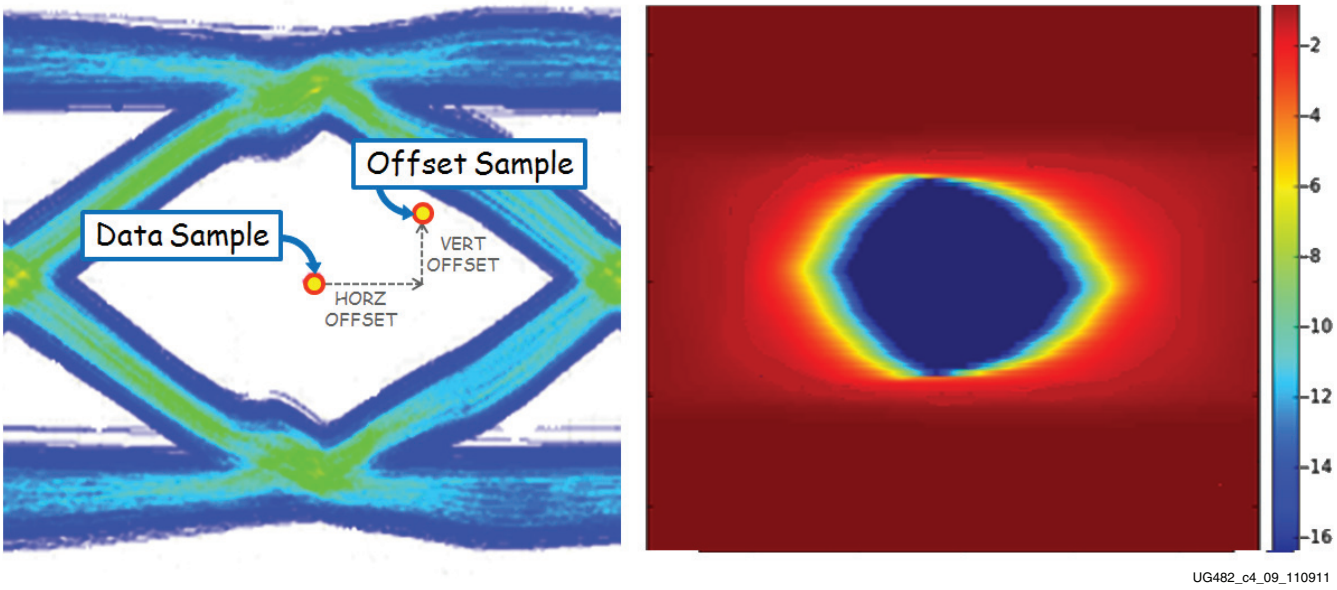


Figure 4-9: **Offset Sample and Data Sample to Calculate BER as a Function of Offset—the Statistical Eye**

Eye Scan Theory

RXDATA is recovered from the equalized differential waveform by sampling after the RX equalizer. The horizontal sampling position is determined by the CDR function and the vertical position is differential zero. This is indicated as data sample in Figure 4-9.

To enable eye scan functionality, an additional sampler is provided with programmable (horizontal and vertical) offsets from the data sample point. This is indicated as offset sample in Figure 4-9.

A single eye scan measurement consists of accumulating the number of data samples (sample count) and the number of times that the offset sample disagreed with the data sample (error count). The bit error ratio (BER) at the programmed vertical and horizontal offset is the ratio of the error count to the sample count. The sample count can range from tens of thousands to greater than 10^{14}.

Repeating such BER measurements for the full array of horizontal and vertical offsets (or a subsampled set of offsets) produces a BER map as shown in Figure 4-9, commonly referred to as a *statistical eye*, where the color map represents $\log_{10}$(BER). In this view, the eye is apparently smaller than a traditional oscilloscope view (as in Figure 4-9) because it has been closed by very low probability jitter and noise that does not show up in the much lower number of samples of an oscilloscope.

Because this functionality puts no restrictions on the data patterns being received nor requires any changes in the RX settings, it can be performed while application data is being

received without error. Furthermore, no FPGA logic is required—only the ability to read and write attributes.

Eye Scan Architecture

The blocks with shaded gray in Figure 4-10 describe the portion of the PMA architecture that supports eye scan. The horizontal offset (HORZ_OFFSET) advances or delays the sampling time of the offset samples relative to the data samples. The vertical offset (VERT_OFFSET) raises or lowers the differential voltage threshold to which the equalized waveform is compared. The data samples are deserialized into the Rdata bus, and the offset samples are deserialized into the Sdata bus.

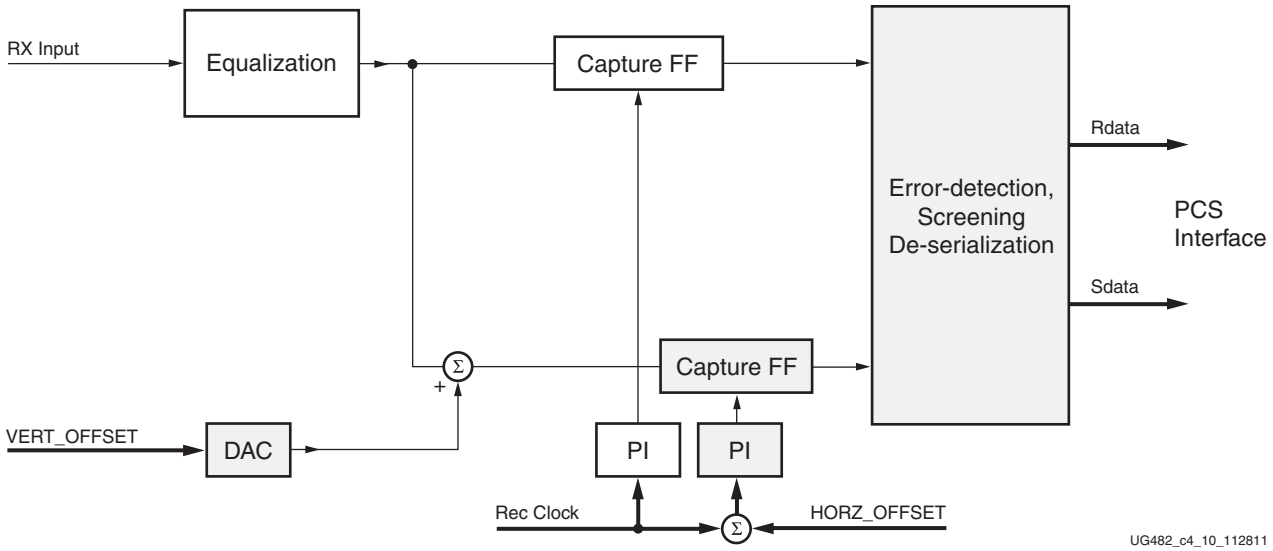


Figure 4-10: **PMA Architecture to Support Eye Scan**

Figure 4-11 describes the portion of the PCS architecture that supports eye scan. The 40-bit Rdata bus contains the data samples, and each bit of the 40-bit Sdata bus is one if and only if the corresponding data sample and offset sample are not equal. (See ES_ERRDET_EN in Table 4-15, page 113.)

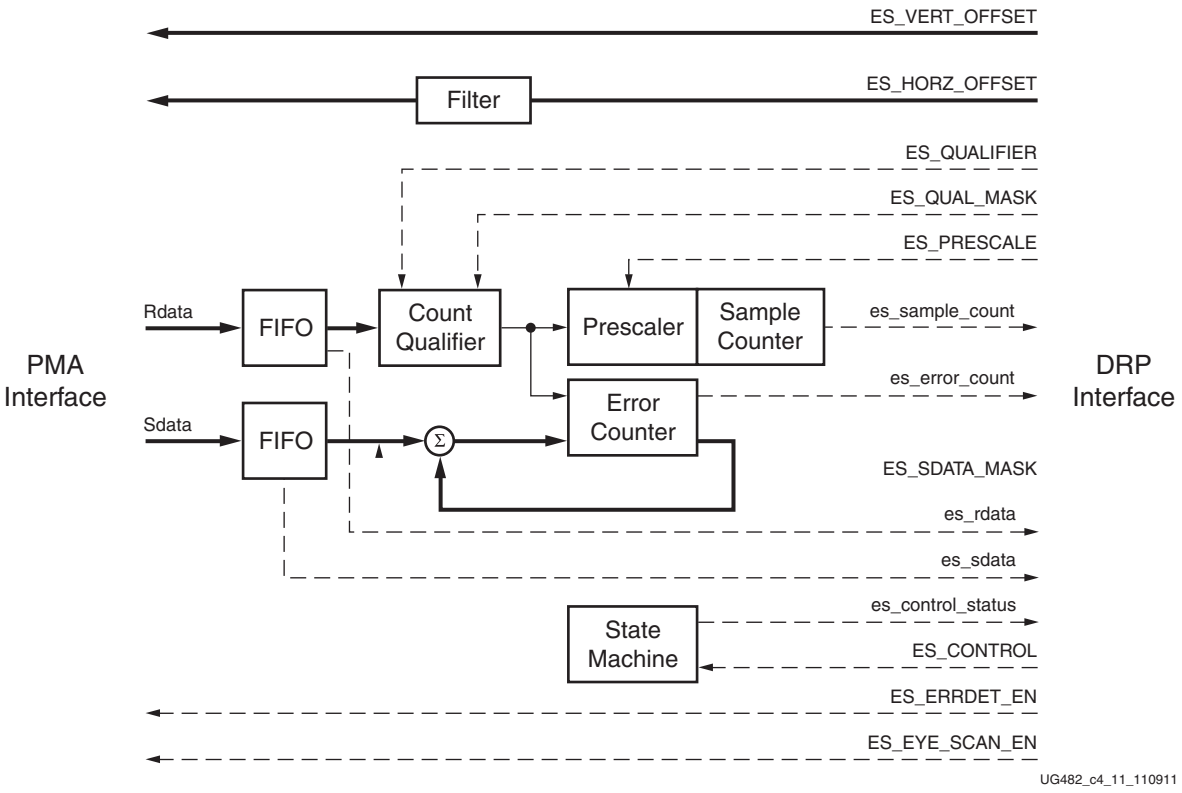


Figure 4-11: **PCS Architecture to Support Eye Scan**

Two consecutive cycles of Sdata are masked by ES_SDATA_MASK[79:0] (i.e., bit-by-bit Sdata[i] AND NOT mask[i]). The algebraic sum of bits [39:0] of this result is the number of errors to be added in the error counter.

Two consecutive cycles of Rdata are compared with the pattern in ES_QUALIFIER[79:0], and that result is masked by (i.e., bit-by-bit ORed with) ES_QUAL_MASK[79:0]. The logical AND of this result determines whether the prescaler/sample counter is incremented and the errors added to the error counter. For a statistical eye, ES_QUAL_MASK is `80'b1`, so the sample counter and error counter accumulate on every cycle. ES_SDATA_MASK unmasks only the current data (bit 39 and below; see the description of RX_DATA_WIDTH) to avoid double counting errors because they appear first in the lower 40 bits and then in the upper 40 bits on the next cycle.

Alternate use modes produce scope-like displays by unmasking a sequence of Rdata bits (up to 20), causing error and sample accumulation only if Rdata matches ES_QUALIFIER in that range of bits. In these use modes, only one Sdata bit per measurement is unmasked. In diagnostic use modes, Rdata and Sdata are *frozen* and can be read out through the DRP interface when:

- An error occurs,
- A count qualifier occurs,
- A fabric port causes a trigger, or

- A trigger is forced via an attribute write.

The diagnostic use modes could be used, for example, to examine the pattern of burst errors due to equalization behavior.

Figure 4-12 documents the state transitions in the eye scan state machine.

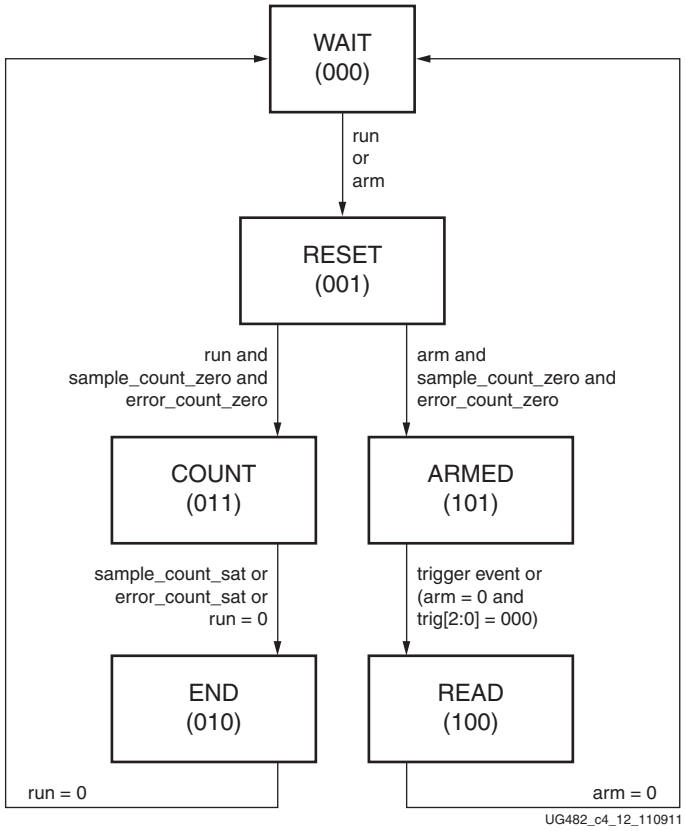


Figure 4-12: **Eye Scan State Machine**

ES_CONTROL[1:0] are the signals arm and run, respectively. From the WAIT state, run initiates the BER measurement loop (left) and arm starts the diagnostic loop (right).

The RESET state zeros the error and sample counters, then enters the COUNT state or the ARMED state (depending on whether run or arm is active).

In the COUNT state, samples and errors are accumulated in the counters. When either counter is saturated, both counters stop and transition to the END state. This transition to the END state is detected by polling es_control_status[3:0]. Bit 0 (done) is set active only in the END, READ, and WAIT states. Bits [3:1] display the current state of the state machine.

The END state transitions to the WAIT state when run is set back to zero. The es_sample_count[15:0] and es_error_count[15:0] can be read either in the END or WAIT state.

In the ARMED state, the FIFOs (successive cycles of Rdata and of Sdata) are stopped when a trigger event occurs. The trigger event is either the count qualifier pulse, the logical OR of all bits into the error counter, or a manual trigger provided from a DRP data input or from a port. One of these four options is selected by trig[3:0] = ES_CONTROL[5:2].

In the READ state, the last two cycles of Rdata can be read from the COE status register, es_rdata[79:0], and the last two cycles of Sdata can be read from the COE status register, es_sdata[79:0].

Ports and Attributes

Table 4-14 defines ports related to the RX eye scan function.

Table 4-14: **RX Margin Analysis Ports**

Port	Dir	Domain	Description
EYESCANDATAERROR	Out	async	Asserts high for one REC_CLK cycle when an (unmasked) error occurs while in the COUNT or ARMED state.
EYESCANTRIGGER	In	RXUSRCLK2	Causes a trigger event. See ES_CONTROL[4] below.
RXRATE	In	RXUSRCLK2	This port dynamically controls the setting for the RX serial clock divider D (see Table 4-11) and it is used with RXOUT_DIV attribute. `3'b000`: Use RXOUT_DIV divider value `3'b001`: Set D divider to 1 `3'b010`: Set D divider to 2 `3'b011`: Set D divider to 4 `3'b100`: Set D divider to 8

Table 4-15 defines RX eye scan attributes. Lower case attribute names indicate R/O.

Table 4-15: **RX Margin Analysis Attributes**

Attribute	Type	Description
ES_VERT_OFFSET	9-bit Binary	Controls the vertical (differential voltage) offset of the scan sample: [6:0]: Offset magnitude. [7]: Offset sign (1 is negative, 0 is positive).

ES_HORZ_OFFSET	12-bit Hex	Controls the horizontal (phase) offset of the scan sample.
		[10:0]: Phase offset (two's complement). The center of data eye (0 UI) corresponds to a count of `11'd0` for all data rates. The table below lists the minimum count (representing -0.5 UI) and maximum count (representing +0.5 UI) for each data rate.
		`Rate  min count [dec(bin)]  eye center [dec(bin)]  max count [dec(bin)]`
		`Full  -32 (11'b11111100000)  +0(11'b00000000000)  +32(11'b00000100000)`
		`Half  -64 (11'b11111000000)  +0(11'b00000000000)  +64(11'b00001000000)`
		`Qrtr  -128 (11'b11110000000)  +0(11'b00000000000)  +128(11'b00010000000)`
		`Octal  -256 (11'b11100000000)  +0(11'b00000000000)  +256(11'b00100000000)`
		[11]: Phase unification. Must be set to 0 for all positive counts (including zero) and to 1 for all negative counts.

ES_PRESCALE	5-bit Binary	Controls the pre-scaling of the sample count to keep both sample count and error count in reasonable precision within the 16-bit register range. Prescale = $2^{(1 + \text{register value})}$, so minimum prescale is $2^{(1+0)}$ = 2 and maximum prescale is $2^{(1+31)}$ = 4,284,967,296.

Table 4-15: **RX Margin Analysis Attributes *(Cont'd)***

Attribute	Type	Description
ES_SDATA_MASK	80-bit Hex	This attribute masks up to two cycles of the 40-bit Sdata bus. Binary 1 causes the corresponding bus bit to be masked and binary 0 leaves it unmasked. To support the statistical eye view, the error counter accumulates the total number of unmasked 1s on the most recent cycle of the Sdata bus (masked by ES_SDATA_MASK[39:0]). To support the scope and waveform views, the error counter increments by only one for any non-zero number of unmasked 1s on the previous cycle of the Sdata bus (masked by ES_SDATA_MASK[79:40]). This attribute and ES_QUAL_MASK must also mask out unused bits. For the statistical eye view, this attribute would assume the following values as a function of bus width: 20-bit width: ES_SDATA_MASK = (`40'b1, 20'b0, 20'b1`) 16-bit width: ES_SDATA_MASK = (`40'b1, 16'b0, 24'b1`) Scope and waveform views require a sequence of measurements, unmasking only a single bit per measurement.
ES_QUALIFIER	80-bit Hex	Eye scan can qualify BER measurements based on patterns up to 20 contiguous bits long in any position in the input data. Because the data, and therefore the qualifier pattern, is not aligned, the position of the pattern must be discovered by a barrel-shifting search. For example, looking for the pattern `10'b0011111010` (K28.5 in 8B/10B code) with a 20-bit data width would require a sequence of measurements such as the following, searching for a non-zero sample count at the correct alignment: ES_QUALIFIER = (`50'b?, 10'b0011111010, 20'b?`) ES_QUALIFIER = (`49'b?, 10'b0011111010, 21'b?`) ES_QUALIFIER = (`48'b?, 10'b0011111010, 22'b?`) …etc… (where ? represents a DON'T CARE bit that will be masked) The qualifier pattern is shifted only over the valid bits for the bus width (40, 32, 20, or 16).
ES_QUAL_MASK	80-bit Hex	This attribute masks those bits not included in the qualifier pattern. For example, the corresponding values for the K28.5 example above would be: ES_QUAL_MASK = (`50'b1, 10'b0, 20'b1`) ES_QUAL_MASK = (`49'b1, 10'b0, 21'b1`) ES_QUAL_MASK = (`48'b1, 10'b0, 22'b1`) …etc…
ES_EYE_SCAN_EN	1-bit Binary	This bit should always be 1 when using eye scan. Setting this bit to 0 powers down the eye scan circuitry in the PMA and forces the eye scan state to WAIT. Re-enabling eye scan functionality requires reasserting this bit and asserting/deasserting PMA reset.
ES_ERRDET_EN	1-bit Binary	1: Each bit of the Sdata bus is 1 if and only if the corresponding offset data sample does not agree with the recovered data sample. This is used for the statistical eye view. 0: Each bit of the Sdata bus is the recovered data sample. Therefore, if no errors occurred, the Sdata bus would be identical to the Rdata bus. This is used for the scope and waveform views.

Table 4-15: **RX Margin Analysis Attributes *(Cont'd)***

Attribute	Type	Description
ES_CONTROL	6-bit Binary	[0]: RUN. Asserting this bit causes a state transition from the WAIT state to the RESET state, initiating a BER measurement sequence. [1]: ARM Asserting this bit causes a state transition from the WAIT state to the RESET state, initiating a diagnostic sequence. In the ARMED state, deasserting this bit causes a state transition to the READ state if one of the states of bits [5:2] below is not met. [5:2]: `0001` In the ARMED state, causes a trigger event (transition to the READ state) when an error is detected (i.e., an unmasked 1 on the Sdata bus). `0010` In the ARMED state, causes a trigger event (transition to the READ state) when the qualifier pattern is detected in Rdata. `0100` In the ARMED state, causes a trigger event (transition to the READ state) when the eye_scan_trigger port asserts High. `1000` In the ARMED state, causes a trigger event (transition to the READ state) immediately.

es_control_status	4-bit Binary	[0]: DONE. Asserted High only in the WAIT, END, or READ states.
		[3:1]: Current state of the state machine:
		`WAIT 000`
		`RESET 001`
		`COUN 011`
		`END 010`
		`ARMED 101`
		`READ 100`

es_rdata	80-bit Binary	When a trigger event occurs in the ARMED state, es_rdata[39:0] is the present state of the Rdata bus and es_rdata[79:40] is the previous state of the Rdata bus.
es_sdata	80-bit Binary	When a trigger event occurs in the ARMED state, es_sdata[39:0] is the present state of the Sdata bus and es_sdata[79:40] is the previous state of the Sdata bus.
es_error_count	16-bit Hex	In END and WAIT states, contains the final error count for the preceding BER measurement.
es_sample_count	16-bit Hex	In END and WAIT states, contains the final sample count for the preceding BER measurement.

RX_DATA_WIDTH	Integer	Sets the bit width of the RXDATA port. When 8B/10B encoding is enabled, RX_DATA_WIDTH must be set to 20 or 40. Valid settings are 16, 20, 32, or 40.
		See Interface Width Configuration, page 175 for more details.
		Width of valid data on Rdata and Sdata buses is the width of the internal datapath (16-bit or 20-bit).
		For the different possible bus widths, the previous and current valid Rdata and Sdata bits correspond to the following indices in ES_SDATA_MASK, ES_QUALIFIER, ES_QUAL_MASK, es_rdata, and es_sdata:
		`valid Rdata and Sdata width  previous data  current data`
		`16  [79:64]  [39:24]`
		`20  [79:60]  [39:20]`

Table 4-15: **RX Margin Analysis Attributes *(Cont'd)***

Attribute	Type	Description
RXOUT_DIV	Integer	PLL0 or PLL1 output clock divider D for the RX datapath as shown in Figure 2-9, page 35. Valid settings are 1, 2, 4, or 8. This attribute sets the divider only if the RXRATE port is set to `3'b000`.

Table 4-16: **DRP Address Map for Eye Scan Read-Only (R) Registers**

DRP Address Hex	DRP Bits	R/W	Name	Attribute Bit
151	15:0	R	es_error_count	15:0
152	15:0	R	es_sample_count	15:0
153	3:0	R	es_control_status	3:0
154	15:0	R	es_rdata	79:64
155	15:0	R	es_rdata	63:48
156	15:0	R	es_rdata	47:32
157	15:0	R	es_rdata	31:16
158	15:0	R	es_rdata	15:0
159	15:0	R	es_sdata	79:64
15A	15:0	R	es_sdata	63:48
15B	15:0	R	es_sdata	47:32
15C	15:0	R	es_sdata	31:16
15D	15:0	R	es_sdata	15:0

RX Polarity Control

Functional Description

If RXP and RXN differential traces are accidentally swapped on the PCB, the differential data received by the GTP transceiver RX are reversed. The GTP transceiver RX allows inversion to be done on parallel bytes in the PCS after the SIPO to offset reversed polarity on differential pair. Polarity control function uses the RXPOLARITY input, which is driven High from the fabric user interface to invert the polarity.

Ports and Attributes

Table 4-17 defines the ports required by the RX polarity control function.

Table 4-17: **RX Polarity Control Ports**

Port	Dir	Clock Domain	Description
RXPOLARITY	In	RXUSRCLK2	The RXPOLARITY port can invert the polarity of incoming data: 0: Not inverted. RXP is positive and RXN is negative. 1: Inverted. RXP is negative and RXN is positive.

Using RX Polarity Control

RXPOLARITY can be tied High if the polarity of RXP and RXN needs to be reversed.

RX Pattern Checker

Functional Description

The GTP transceiver receiver includes a built-in PRBS checker (see Figure 4-13). This checker can be set to check for one of four industry-standard PRBS patterns. The checker is self-synchronizing and works on the incoming data before comma alignment or decoding. This function can be used to test the signal integrity of the channel.

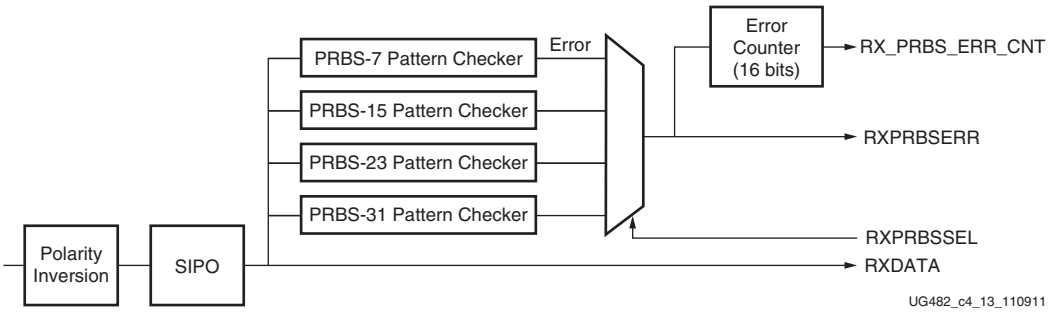


Figure 4-13: **RX Pattern Checker Block**

Ports and Attributes

Table 4-18 defines the pattern checker ports.

Table 4-18: **Pattern Checker Ports**

Port	Dir	Clock Domain	Description
RXPRBSCNTRESET	In	RXUSRCLK2	Resets the PRBS error counter.
RXPRBSSEL[2:0]	In	RXUSRCLK2	Receiver PRBS checker test pattern control. Only these settings are valid: `000`: Standard operation mode. (PRBS check is off) `001`: PRBS-7 `010`: PRBS-15 `011`: PRBS- 23 `100`: PRBS-31 No checking is done for non-PRBS patterns. Single bit errors cause bursts of PRBS errors because the PRBS checker uses data from the current cycle to generate next cycle's expected data.
RXPRBSERR	Out	RXUSRCLK2	This non-sticky status output indicates that PRBS errors have occurred.

Table 4-19 defines the pattern checker attributes.

Table 4-19: **Pattern Checker Attributes**

Attribute	Type	Description
RX_PRBS_ERR_CNT	16-bit Binary	PRBS error counter. This counter can be reset by asserting RXPRBSCNTRESET. When an error(s) in incoming parallel data occurs, this counter increments by 1 and counts up to `0xFFFF`. This error counter can only be accessed via the DRP. The address for this counter is `0x15E`.
RXPRBS_ERR_LOOPBACK	1-bit Binary	When this attribute is set to 1, the RXPRBSERR bit is internally looped back to TXPRBSFORCEERR of the same GTP transceiver. This allows synchronous and asynchronous jitter tolerance testing without worrying about data clock domain crossing. When this attribute is set to 0, TXPRBSFORCEERR is forced onto the TX PRBS.

Use Models

To use the built-in PRBS checker, RXPRBSSEL must be set to match the PRBS pattern being sent to the receiver. The RXPRBSSEL entry in Table 4-18 shows the available settings. When the PRBS checker is running, it attempts to find the selected PRBS pattern in the incoming data. If the incoming data is inverted by the transmitter or reversed RXP/RXN, the received data should also inverted by controlling RXPOLARITY. Otherwise, the PRBS checker does not lock. When it finds the pattern, it can detect PRBS errors by comparing

the incoming pattern with the expected pattern. The expected pattern is generated from the previous incoming data. The checker counts the number of word (20 bits per word) errors and increments the word error counter by 1 when an error(s) is found in the incoming parallel data. Thus the word error counter might not match the actual number of bit errors if the incoming parallel data contains two or more bit errors. The error counter stops counting when reaching `0xFFFF`.

When the error occurs, RXPRBSERR is asserted. When no error is found in the following incoming data, RXPRBSERR is cleared. Asserting RXPRBSCNTRESET clears the error counter. GTRXRESET and RXPCSRESET also reset the count.

Refer to TX Pattern Generator, page 69 for more information about use models.

RX Byte and Word Alignment

Functional Description

Serial data must be aligned to symbol boundaries before it can be used as parallel data. To make alignment possible, transmitters send a recognizable sequence, usually called a comma. The receiver searches for the comma in the incoming data. When it finds a comma, it moves the comma to a byte boundary so the received parallel words match the transmitted parallel words.

Figure 4-14 shows the alignment to a 10-bit comma. The RX receiving unaligned bits are on the right side. The serial data with the comma is highlighted in the middle. Byte aligned RX parallel data is on the left.

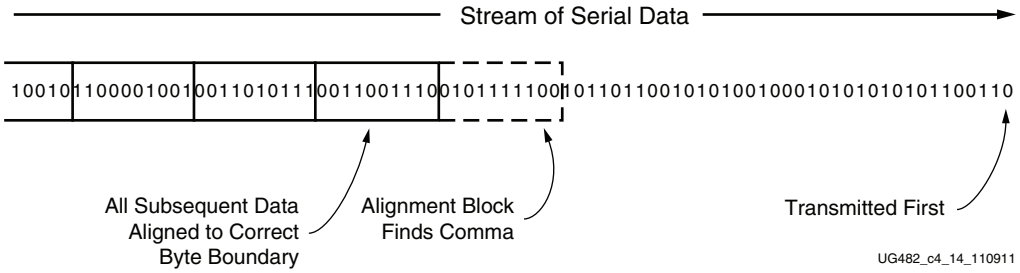


Figure 4-14: **Conceptual View of Comma Alignment (Aligning to a 10-Bit Comma)**

Figure 4-15 shows TX parallel data on the left side, and RX receiving recognizable parallel data after comma alignment on the right side.

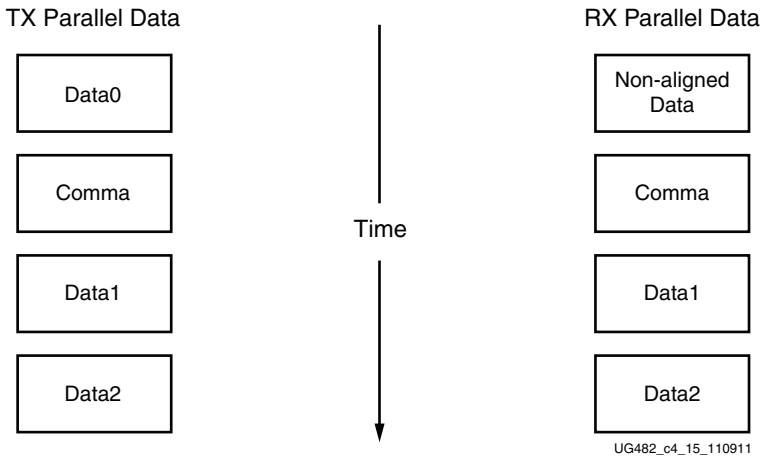


Figure 4-15: **Parallel Data View of Comma Alignment**

Enabling Comma Alignment

To enable the comma alignment block, the RXCOMMADETEN port is driven High. RXCOMMADETEN is driven Low to bypass the block completely for minimum latency.

Configuring Comma Patterns

To set the comma pattern that the block searches for in the incoming data stream, the ALIGN_MCOMMA_VALUE, ALIGN_PCOMMA_VALUE, and ALIGN_COMMA_ENABLE attributes are used. The comma lengths depend on RX_DATA_WIDTH (see Table 4-42, page 177). Figure 4-16 shows how the ALIGN_COMMA_ENABLE masks each of the comma values to allow partial pattern matching.

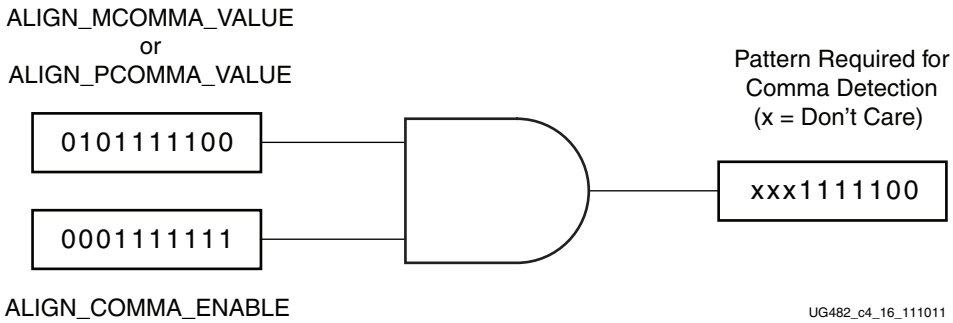


Figure 4-16: **Comma Pattern Masking**

Figure 4-17 shows how the commas are combined when ALIGN_COMMA_DOUBLE is TRUE.

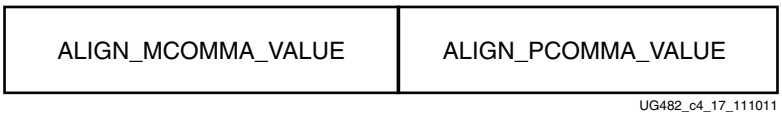


Figure 4-17: **Extended Comma Pattern Definition**

Figure 4-18 shows how a comma is combined with ALIGN_COMMA_ENABLE to make a wild-carded comma for a 20-bit internal comma. If ALIGN_COMMA_DOUBLE is TRUE, the MCOMMA and PCOMMA patterns are combined so that the block searches for two commas in a row. The number of bits in the comma depends on RX_DATA_WIDTH. Either a 16-bit or a 20-bit comma alignment mode is possible. A double comma is only detected when the received data has a PCOMMA defined by ALIGN_PCOMMA_VALUE followed by an MCOMMA defined by ALIGN_MCOMMA_VALUE with no extra bits in between.

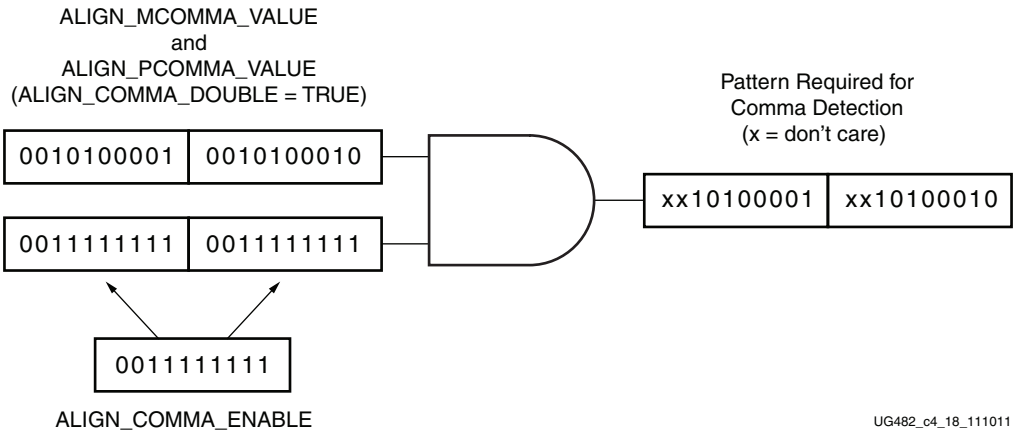


Figure 4-18: **Extended Comma Pattern Masking**

Activating Comma Alignment

Commas are aligned to the closest boundary providing they are found while comma alignment is active. RXMCOMMAALIGNEN is driven High to align on the MCOMMA pattern. RXPCOMMAALIGNEN is driven High to activate alignment on the PCOMMA pattern. Both enable ports are driven to align to either pattern. When ALIGN_COMMA_DOUBLE is TRUE, both enable ports must always be driven to the same value.

Alignment Status Signals

While MCOMMA or PCOMMA alignment is active, any matching comma pattern causes the block to realign to the closest boundary. After successful alignment, the block holds RXBYTEISALIGNED High. At this time, RXMCOMMAALIGNEN and RXPCOMMAALIGNEN can be driven Low to turn off alignment and keep the current alignment position. RXPCOMMAALIGNEN must be TRUE for PCOMMAs to cause RXBYTEISALIGNED to go High. Similarly, RXMCOMMAALIGNEN must be TRUE for MCOMMAs to cause RXBYTEISALIGNED to go High. Commas can arrive while RXBYTEISALIGNED is High. If the commas arrive aligned to boundaries, there is no change. If the commas arrive out of position, the block deasserts RXBYTEISALIGNED

until the commas are aligned again. If alignment is still activated for the comma that arrives, the block automatically aligns the new comma to the closest boundary and drives RXBYTEREALIGN High for one RXUSRCLK2 cycle.

Alignment Boundaries

The allowed boundaries for alignment are defined by ALIGN_COMMA_WORD. The spacing of the possible boundaries is determined by RX_DATA_WIDTH, and the number of boundary positions is determined by the number of bytes in the RXDATA interface (refer to Table 4-38, page 175 for RX_DATA_WIDTH settings). Figure 4-19 shows the boundaries that can be selected.

RX_DATA_WIDTH	ALIGN_COMMA_WORD	Possible RX Alignments (Grey = Comma Can Appear on Byte)
16/20 (2-byte)	1	Byte1 Byte0
16/20 (2-byte)	2	Byte1 Byte0
32/40 (4-byte)	1	Byte3 Byte2 Byte1 Byte0
32/40 (4-byte)	2	Byte3 Byte2 Byte1 Byte0

UG482_c4_19_112811

Figure 4-19: **Comma Alignment Boundaries**

Manual Alignment

RXSLIDE can be used to override the automatic comma alignment and to shift the parallel data. RXSLIDE is driven High for one RXUSRCLK2 cycle to shift the parallel data by one bit. RXSLIDE must be Low for at least 32 RXUSRCLK2 cycles before it can be used again.

Figure 4-20 shows the waveforms for manual alignment using RXSLIDE in RXSLIDE_MODE = PCS, before and after the data shift. When RXSLIDE_MODE = PCS is used, the number of bit shift positions when consecutive RXSLIDE pulses are issued is also determined by the comma alignment boundary set by ALIGN_COMMA_WORD and RX_DATA_WIDTH. For example, if the RX_DATA_WIDTH is 20 bits and ALIGN_COMMA_WORD is 1, after the 9th slide operation, the slide position returns back to 0. For the same RX_DATA_WIDTH setting, for an ALIGN_COMMA_WORD setting of 2, the slide position returns to 0 after the 19th slide operation.

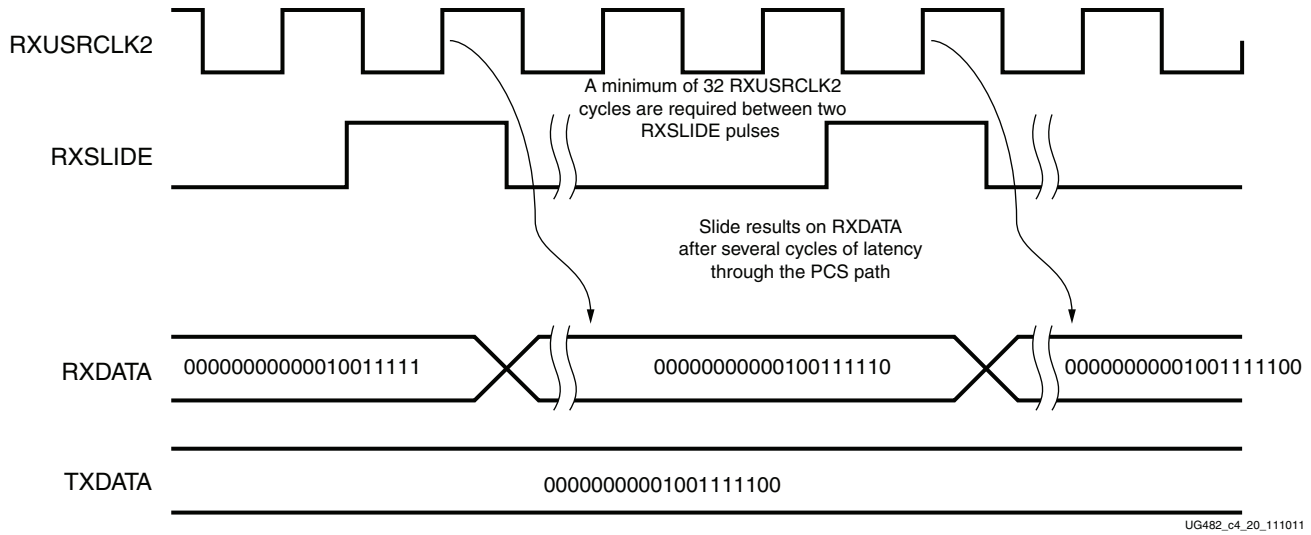


Figure 4-20: **Manual Data Alignment Using RXSLIDE for RX_DATA_WIDTH = 20 Bits and RXSLIDE_MODE = PCS**

Note relevant to Figure 4-20:

1. Latency between the slide and the slide result at RXDATA depends on the number of active RX PCS blocks in the datapath.

Figure 4-21 shows the waveforms for manual alignment using RXSLIDE in RXSLIDE_MODE = PMA before and after the data shift. In this mode, the data is shifted right by one bit for every RXSLIDE pulse issued, but there is some intermediate data with the bits shifted left before the final data appears on the bus. When RXSLIDE_MODE = PMA is used, the RX recovered clock phase is shifted by 2 UI for every alternate RXSLIDE pulse.

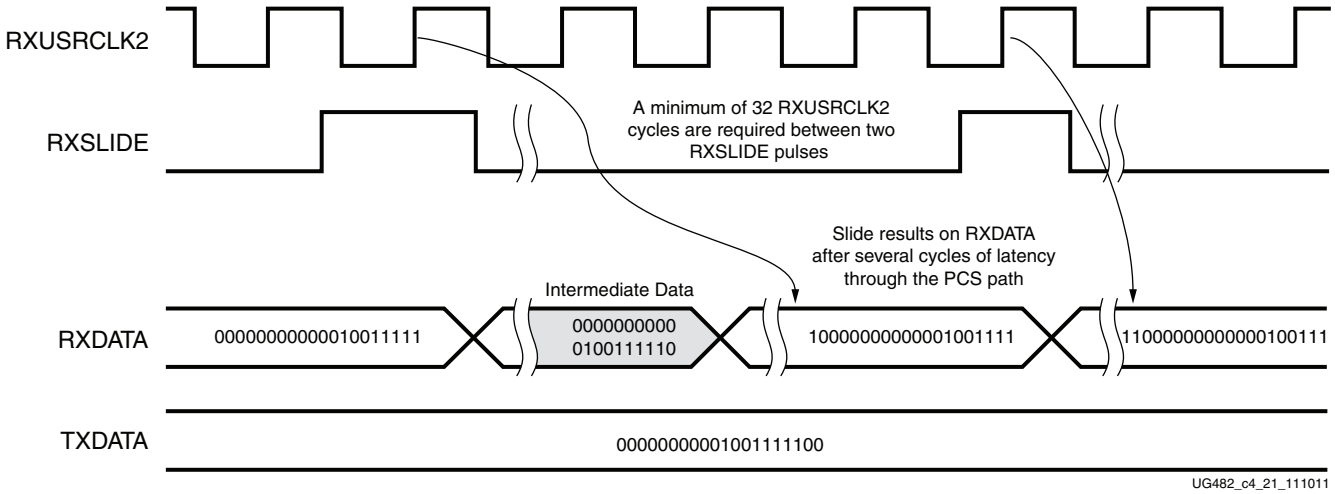


Figure 4-21: **Manual Data Alignment Using RXSLIDE for RX_DATA_WIDTH = 20 Bits and RXSLIDE_MODE = PMA**

Note relevant to Figure 4-21:

1. Latency between the slide and the slide result at RXDATA depends on the number of active RX PCS blocks in the datapath.

Ports and Attributes

Table 4-20 defines the RX byte and word alignment ports.

Table 4-20: **RX Byte and Word Alignment Ports**

Port Name	Dir	Clock Domain	Description
RXBYTEISALIGNED	Out	RXUSRCLK2	This signal from the comma detection and realignment circuit is High to indicate that the parallel data stream is properly aligned on byte boundaries according to comma detection. 0: Parallel data stream not aligned to byte boundaries 1: Parallel data stream aligned to byte boundaries There are several cycles after RXBYTEISALIGNED is asserted before aligned data is available at the FPGA RX interface. RXBYTEISALIGNED responds to plus comma alignment when RXPCOMMAALIGNEN is TRUE. RXBYTEISALIGNED responds to minus comma alignment when RXMCOMMAALIGNEN is TRUE.
RXBYTEREALIGN	Out	RXUSRCLK2	This signal from the comma detection and realignment circuit indicates that the byte alignment within the serial data stream has changed due to comma detection. 0: Byte alignment has not changed 1: Byte alignment has changed Data can be lost or repeated when alignment occurs, which can cause data errors (and disparity errors when the 8B/10B decoder is used).
RXCOMMADET	Out	RXUSRCLK2	This signal is asserted when the comma alignment block detects a comma. The assertion occurs several cycles before the comma is available at the FPGA RX interface. 0: Comma not detected 1: Comma detected

Table 4-20: **RX Byte and Word Alignment Ports *(Cont'd)***

Port Name	Dir	Clock Domain	Description
RXCOMMADETEN	In	RXUSRCLK2	RXCOMMADETEN activates the comma detection and alignment circuit. 0: Bypass the circuit 1: Use the comma detection and alignment circuit Bypassing the comma and alignment circuit reduces RX datapath latency.
RXPCOMMAALIGNEN	In	RXUSRCLK2	Aligns the byte boundary when comma plus is detected. 0: Disabled 1: Enabled.
RXMCOMMAALIGNEN	In	RXUSRCLK2	Aligns the byte boundary when comma minus is detected. 0: Disabled 1: Enabled.
RXSLIDE	In	RXUSRCLK2	RXSLIDE implements a comma alignment bump control. When RXSLIDE is asserted, the byte alignment is adjusted by one bit, which permits determination and control of byte alignment by the FPGA logic. Each assertion of RXSLIDE causes just one adjustment. RXSLIDE must be deasserted for more than 32 RXUSRCLK2 cycles before it can be reasserted to cause another adjustment. When asserted, RXSLIDE takes precedence over normal comma alignment. For proper operation, the user should set these values: RXPCOMMAALIGNEN = 0; RXMCOMMAALIGNEN = 0; RXCOMMADETEN = 1; SHOW_REALIGN_COMMA = FALSE

Table 4-21 defines the RX byte and word alignment attributes.

Table 4-21: **RX Byte and Word Alignment Attributes**

Attribute	Type	Description
ALIGN_COMMA_WORD	Integer	This attribute controls the alignment of detected commas within a multi-byte datapath. 1: Align comma to either of the 2 bytes for a 2-byte interface, any of the 4 bytes for a 4-byte interface. The comma can be aligned to either the even bytes or the odd bytes of RXDATA output. 2: Align comma to the even bytes only. The aligned comma is guaranteed to be aligned to even bytes RXDATA[9:0] for a 2-byte interface, RXDATA[9:0]/RXDATA[29:20] for a 4-byte interface. Refer to Figure 4-19, page 122 for comma alignment boundaries allowed for the different ALIGN_COMMA_WORD and RX_DATA_WIDTH settings. Protocols that send commas in even and odd positions must set ALIGN_COMMA_WORD to 1.
ALIGN_COMMA_ENABLE	10-bit Binary	Sets which bits in MCOMMA/PCOMMA must be matched to incoming data and which bits can be of any value. This attribute is a 10-bit mask with a default value of `1111111111`. Any bit in the mask that is reset to 0 effectively turns the corresponding bit in MCOMMA or PCOMMA to a don't care bit.
ALIGN_COMMA_DOUBLE	Boolean	Specifies whether a comma match consists of either a comma plus or a comma minus alone, or if both are required in the sequence. FALSE: The plus comma (PCOMMA) and minus comma (MCOMMA) are handled separately. An individual match for either can lead to comma detection and alignment. TRUE: A comma match consists of a comma plus followed immediately by a comma minus. The match pattern is 20 or 16 bits (as determined by RX_DATA_WIDTH). When ALIGN_COMMA_DOUBLE is TRUE, ALIGN_PCOMMA_DET must be the same as ALIGN_MCOMMA_DET, and RXPCOMMAALIGNEN must be the same as RXMCOMMAALIGNEN.

Table 4-21: **RX Byte and Word Alignment Attributes *(Cont'd)***

Attribute	Type	Description
ALIGN_MCOMMA_VALUE	10-bit Binary	Defines comma minus to raise RXCOMMADET and align the parallel data. The reception order is right to left. (ALIGN_MCOMMA_VALUE [0] is received first.) The default value is `10'b1010000011` (K28.5). This definition does not affect 8B/10B encoding or decoding.
ALIGN_MCOMMA_DET	Boolean	Controls the raising of RXCOMMADET on comma minus. FALSE: Do not raise RXCOMMADET when comma minus is detected. TRUE: Raise RXCOMMADET when comma minus is detected. (This setting does not affect comma alignment.)
ALIGN_PCOMMA_VALUE	10-bit Binary	Defines comma plus to raise RXCOMMADET and align parallel data. The reception order is right to left. (ALIGN_PCOMMA_VALUE [0] is received first.) The default value is `10'b0101111100` (K28.5). This definition does not affect 8B/10B encoding or decoding.
ALIGN_PCOMMA_DET	Boolean	Controls the raising of RXCOMMADET on comma plus. FALSE: Do not raise RXCOMMADET when comma plus is detected. TRUE: Raise RXCOMMADET when comma plus is detected. (This setting does not affect comma alignment.)
SHOW_REALIGN_COMMA	Boolean	Defines if a comma that caused realignment is brought out to the FPGA RX. FALSE: Do not bring the comma that causes realignment to the FPGA RX. This setting reduces RX datapath latency TRUE: Bring the realignment comma to the FPGA RX. SHOW_REALIGN_COMMA = TRUE should not be used when ALIGN_COMMA_DOUBLE = TRUE or when manual alignment is used.

Table 4-21: **RX Byte and Word Alignment Attributes *(Cont'd)***

Attribute	Type	Description
RXSLIDE_MODE	String	Defines the RXSLIDE mode. OFF: Default setting. The RXSLIDE feature is not used. PCS: PCS is used to perform the bit-slipping function. RXSLIDE is driven High for one RXUSRCLK2 cycle to shift the parallel data (RXDATA) to the left by one bit within the comma alignment boundary determined by the ALIGN_COMMA_WORD and RX_DATA_WIDTH settings. In this mode, even if RXOUTCLK is sourcing from the RX PMA, the clock phase remains the same. This option requires SHOW_REALIGN_COMMA to be FALSE. PMA: PMA is used to perform the bit-slipping function. RXSLIDE is driven High for one RXUSRCLK2 cycle to shift the parallel data (RXDATA) to the right by one bit. If RXOUTCLK is sourcing from the RX PMA, its phase might be changed. This mode provides minimum variation of latency compared to PCS mode. This option requires SHOW_REALIGN_COMMA to be FALSE. AUTO: This is an automated PMA mode without using the FPGA logic to monitor the RXDATA and issue RXSLIDE pulses. In this mode, RXSLIDE is ignored. In PCIe applications, this setting is used for FTS lane deskew. This option requires SHOW_ALIGN_COMMA to be FALSE.
RXSLIDE_AUTO_WAIT	Integer	Defines how long the PCS (in terms of RXUSRCLK clock cycle) waits for the PMA to auto slide before checking the alignment again. Valid settings are from 0 to 15. The default value is 7. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RX_SIG_VALID_DLY	Integer	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

RX 8B/10B Decoder

Functional Description

If RX received data is 8B/10B encoded, it must be decoded. The GTP transceiver has a built-in 8B/10B encoder in the GTP transceiver TX and an 8B/10B decoder in the GTP

transceiver RX, which includes two one-byte 8B/10B decoder modules on the datapath to decode data without consuming FPGA resources. The RX 8B/10B decoder has these features:

1. Supports 2-byte and 4-byte datapath operation
2. Provides daisy-chained hookup of running disparity for proper disparity
3. Generates K characters and status outputs
4. Can be bypassed if incoming data is not 8B/10B encoded
5. Pipes out 10-bit literal encoded values when encountering a not-in-table error

8B/10B Bit and Byte Ordering

The order of the bits into the 8B/10B decoder is the opposite of the order shown in Appendix C, 8B/10B Valid Characters. 8B/10B decoding requires bit a0 to be received first, but the GTP transceiver always receives the right-most bit first. Consequently, the 8B/10B decoder automatically reverses the bit order of received data before decoding it. Decoded data is available on RXDATA ports. Figure 4-22 shows data received by the GTP transceiver RX when RX_DATA_WIDTH = 20 or 40. Data is reconstructed into bytes and sent to the RXDATA interface after the 8B/10B decoder. The number of bits used by RXDATA and corresponding byte orders are determined by RX_DATA_WIDTH.

- Only use RXDATA[15:0] if RX_DATA_WIDTH = 20
- Use full RXDATA[31:0] if RX_DATA_WIDTH = 40

When the 8B/10B decoder is bypassed but RX_DATA_WIDTH is set to multiple of 10, 10-bit characters are passed to the RX data interface with this format:

- The corresponding RXDISPERR represents the 9th bit
- The corresponding RXCHARISK represents the 8th bit
- The corresponding RXDATA byte represents the [7:0] bits

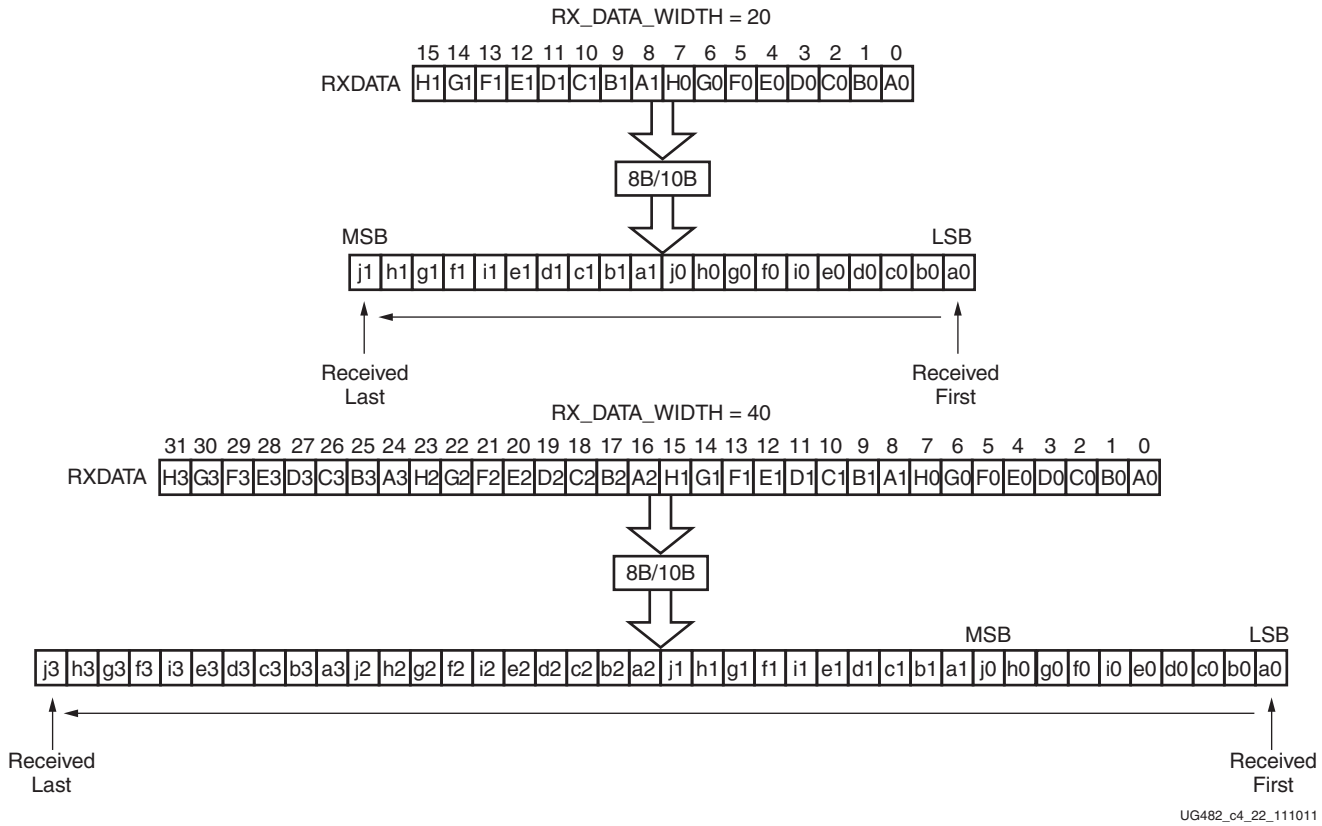


Figure 4-22: **8B/10B Decoder Bit and Byte Order**

RX Running Disparity

Disparity check is performed and the decoder drives the corresponding RXDISPERR High when the data byte on RXDATA arrives with the wrong disparity. In addition to disparity errors, the 8B/10B decoder detects 20-bit out-of-table error codes. The decoder drives the RXNOTINTABLE port High when decoder is enabled but a received 10-bit character cannot be mapped into a valid 8B/10B character listed in Appendix C, 8B/10B Valid Characters. The non-decoded 10-bit character is piped out of the decoder through the RX data interface with this format:

- The corresponding RXDISPERR represents the 9th bit
- The corresponding RXCHARISK represents the 8th bit
- The corresponding RXDATA byte represents the [7:0] bits

Figure 4-23 shows a waveform at the RX data interface when the decoder receives good data (A), data with disparity error (B), an out-of-table character (C), and an out-of-table character with disparity error (D).

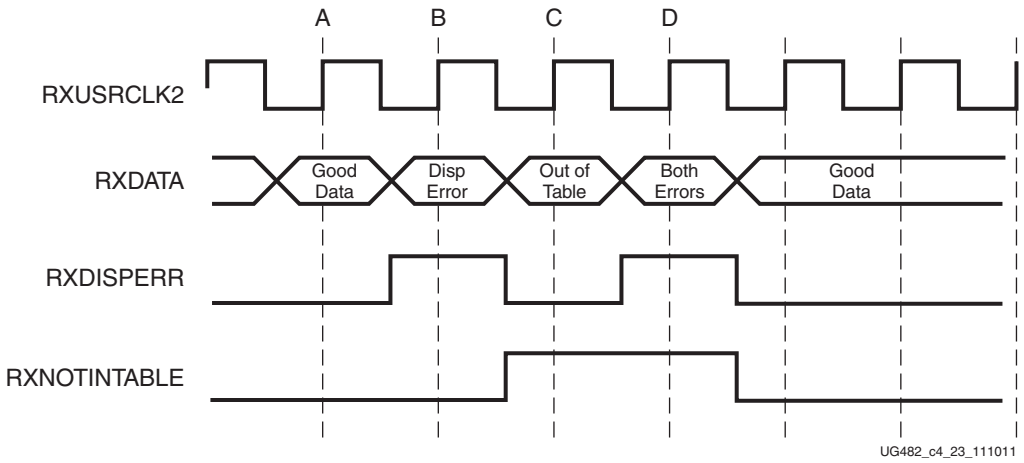


Figure 4-23: **RX Data with 8B/10B Errors**

Special Characters

8B/10B decoding includes special characters (K characters) that are often used for control functions. When RXDATA is a K character, the decoder drives RXCHARISK High.

If DEC_PCOMMA_DETECT is set to TRUE, the decoder drives the corresponding RXCHARISCOMMA High whenever RXDATA is a positive 8B/10B comma. If DEC_MCOMMA_DETECT is TRUE, the decoder drives the corresponding RXCHARISCOMMA bit High whenever RXDATA is a negative 8B/10B comma.

Ports and Attributes

Table 4-22 defines the ports required by RX 8B/10B decoder.

Table 4-22: **RX 8B/10B Decoder Ports**

Port	Dir	Clock Domain	Description
RX8B10BEN	In	RXUSRCLK2	RX8B10BEN selects the use of the 8B/10B decoder in the RX datapath, just after the comma detection/realignment block. If this input is Low, the literal 10-bit data comes out as {RXDISPERR, RXCHARISK, RXDATA<8 bits>}. 1: 8B/10B decoder enabled 0: 8B/10B decoder bypassed (reduces latency)
RXCHARISCOMMA[3:0]	Out	RXUSRCLK2	Active High indicates the corresponding byte shown on RXDATA is a comma character. RXCHARISCOMMA[3] corresponds to RXDATA[31:24] RXCHARISCOMMA[2] corresponds to RXDATA[23:16] RXCHARISCOMMA[1] corresponds to RXDATA[15:8] RXCHARISCOMMA[0] corresponds to RXDATA[7:0]

Table 4-22: **RX 8B/10B Decoder Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
RXCHARISK[3:0]	In	RXUSRCLK2	Active High indicates the corresponding byte shown on RXDATA is a K character when 8B/10B decoding is enabled. RXCHARISK[3] corresponds to RXDATA[31:24] RXCHARISK[2] corresponds to RXDATA[23:16] RXCHARISK[1] corresponds to RXDATA[15:8] RXCHARISK[0] corresponds to RXDATA[7:0] This is bit 8 of non-decoded data if the 8B/10B decoder is bypassed or the corresponding bit of RXNOTINTABLE is High. Refer to FPGA RX Interface, page 174.
RXDISPERR[3:0]	Out	RXUSRCLK2	Active High indicates the corresponding byte shown on RXDATA has a disparity error. RXDISPERR[3] corresponds to RXDATA[31:24] RXDISPERR[2] corresponds to RXDATA[23:16] RXDISPERR[1] corresponds to RXDATA[15:8] RXDISPERR[0] corresponds to RXDATA[7:0] This is bit 9 of non-decoded data if the 8B/10B decoder is bypassed or the corresponding bit of RXNOTINTABLE is High. Refer to FPGA RX Interface, page 174.
RXNOTINTABLE[3:0]	Out	RXUSRCLK2	Active High indicates the corresponding byte shown on RXDATA was not a valid character in the 8B/10B table. RXNOTINTABLE[3] corresponds to RXDATA[31:24] RXNOTINTABLE[2] corresponds to RXDATA[23:16] RXNOTINTABLE[1] corresponds to RXDATA[15:8] RXNOTINTABLE[0] corresponds to RXDATA[7:0]

Table 4-23: **RX 8B/10B Decoder Attributes**

Attribute	Type	Description
RX_DISPERR_SEQ_MATCH	Boolean	Specifies whether the disparity error status of a decoded byte must match the indicator in the channel bonding and clock correction sequence. When TRUE, indicates the disparity error status must be matched. When FALSE, ignores the disparity error status.
DEC_MCOMMA_DETECT	Boolean	When set to TRUE, drives the per byte flag RXCHARISCOMMA High when an MCOMMA is detected. When set to FALSE, RXCHARISCOMMA is Low when a negative comma detected.

Table 4-23: **RX 8B/10B Decoder Attributes *(Cont'd)***

Attribute	Type	Description
DEC_PCOMMA_DETECT	Boolean	When set to TRUE, drives the per byte flag RXCHARISCOMMA High when a PCOMMA is detected. When set to FALSE, RXCHARISCOMMA is Low when a positive comma detected.
DEC_VALID_COMMA_ONLY	Boolean	When set to TRUE, drives the per byte flag RXCHARISCOMMA High when only IEEE 802.3 valid commas K28.1, K28.5, and K28.7 are detected. When set to FALSE, RXCHARISCOMMA is for positive or negative 8B/10B commas, depending how the user sets DEC_PCOMMA_DETECT and DEC_MCOMMA_DETECT.
RX_DATA_WIDTH	3-bit Binary	The PCS data width is set at the Fabric user interface with values of 16 or 32 (if 8B/10B decoding is not used) or 20 or 40 (if 8B/10B decoding is used).

Enabling and Disabling 8B/10B Decoding

To enable the 8B/10B decoder, RX8B10BEN must be driven High. RX_DATA_WIDTH must be set to a multiple of 8 (8, 16, 32) when the 8B/10B decoder enabled.

To disable the 8B/10B decoder on the GTP transceiver receiver path, RX8B10BEN must be driven Low. When the encoder is disabled, RX_DATA_WIDTH can be set to a multiple of 10 (10, 20, 40). The operation of the RXDATA port with 8B/10B decoding bypassed is described in FPGA RX Interface, page 174.

RX Buffer Bypass

Functional Description

Bypassing the RX elastic buffer is an advanced feature of the 7 series GTP transceiver. The RX phase alignment circuit is used to adjust the phase difference between the PMA parallel clock domain (XCLK) and the RXUSRCLK domain when the RX elastic buffer is bypassed. It also performs the RX delay alignment by adjusting the RXUSRCLK to compensate for the temperature and voltage variations. The combined RX phase and delay alignments can be automatically performed by the GTP transceiver or manually controlled by the user. Figure 4-25 shows the XCLK and RXUSRCLK domains, and Table 4-26 shows trade-offs between buffering and phase alignment.

The RX elastic buffer can be bypassed to reduce latency when the RX recovered clock is used to source RXUSRCLK and RXUSRCLK2. When the RX elastic buffer is bypassed, latency through the RX datapath is low and deterministic, but clock correction and channel bonding are not available.

Figure 4-24 shows how RX phase alignment allows the RX elastic buffer to be bypassed. Before RX phase alignment, there is no guaranteed phase relationship between the PMA parallel clock domain (XCLK) and the RXUSRCLK domain. RX phase alignment selects a

phase shifted version of the RX recovered clock from the CDR (XCLK) so that there is no significant phase difference between XCLK and RXUSRCLK.

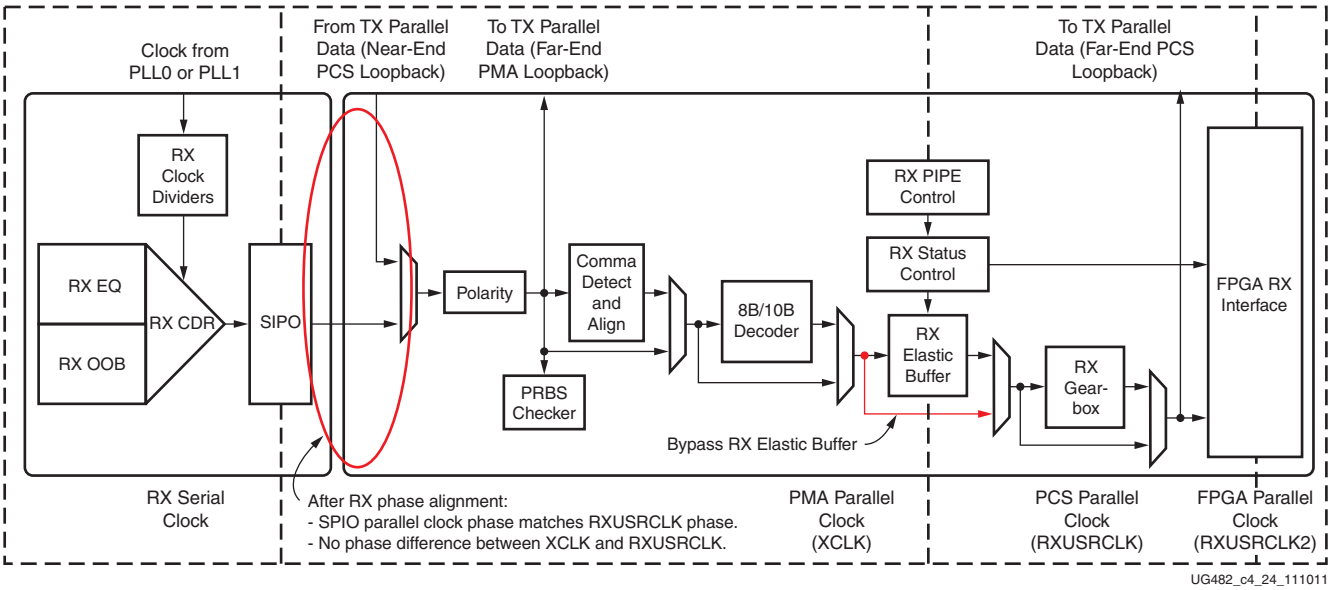


Figure 4-24: **Using RX Phase Alignment**

Ports and Attributes

Table 4-24 defines the RX buffer bypass ports.

Table 4-24: **RX Buffer Bypass Ports**

Port	Dir	Clock Domain	Description
RXPHDLYRESET	In	Async	RX phase alignment hard reset to force RXUSRCLK to the center of the delay alignment tap. The delay alignment tap has a full range of ±4 ns and a half range of ±2 ns. This hard reset can be used to initiate the GTP transceiver to perform the RX phase and delay alignment automatically when all other RX buffer bypass input ports are set Low.
RXPHALIGN	In	Async	Sets the RX phase alignment. Tied Low when using the auto alignment mode.
RXPHALIGNEN	In	Async	RX phase alignment enable. Tied Low when using the auto alignment mode.

Table 4-24: **RX Buffer Bypass Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
RXPHDLYPD	In	Async	RX phase and delay alignment circuit power down. Tied Low in RX buffer bypass mode. 0: Power-up the RX phase and delay alignment circuit. 1: Power-down the RX phase and delay alignment circuit.
RXPHOVRDEN	In	Async	RX phase alignment counter override enable. Tied Low when not in use. 0: Normal operation. 1: Enables the RX phase alignment counter override with the RXPH_CFG[10:0] value.
RXDLYSRESET	In	Async	RX delay alignment soft reset to gradually shift RXUSRCLK to the center of the delay alignment tap. The delay alignment tap has a full range of ±4 ns and a half range of ±2 ns. This soft reset can be used to initiate the GTP transceiver to perform the RX phase and delay alignment automatically when all other RX bypass buffer input ports are Low.
RXDLYBYPASS	In	Async	RX delay alignment bypass. 0: Uses the RX delay alignment circuit. 1: Bypasses the RX delay alignment circuit.
RXDLYEN	In	Async	RX delay alignment enable. Tied Low when not in use.
RXDLYOVRDEN	In	Async	RX delay alignment counter override enable. Tied Low when not in use. 0: Normal operation. 1: Enables the RX delay alignment counter override with the RXDLY_CFG[14:6] value.
RXDDIEN	In	Async	RX data delay insertion enable in the deserializer. Set High in RX buffer bypass mode.
RXPHALIGNDONE	Out	Async	RX phase alignment done. When the auto RX phase and delay alignment are used, the second rising edge of RXPHALIGNDONE detected after RXDLYSRESETDONE assertion indicates RX phase and delay alignment are done.

Table 4-24: **RX Buffer Bypass Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
RXPHMONITOR	Out	Async	RX phase alignment monitor.
RXPHSLIPMONITOR	Out	Async	RX phase alignment slip monitor.
RXDLYSRESETDONE	Out	Async	RX delay alignment soft reset done.
RXSYNCMODE	In	Async	0: RX buffer bypass slave lane 1: RX buffer bypass master lane
RXSYNCIN	In	Async	For use in RX buffer bypass multilane applications. This input is connected to the RXSYNCOUT from the RX buffer bypass master lane.
RXSYNCALLIN	In	Async	Single-lane mode: This input is connected to its own RXPHALIGNDONE. Multilane mode: This input is connected to the ANDed signal of RXPHALIGNDONE of the master and all slave lanes.
RXSYNCOUT	Out	Async	For use only by the RX buffer bypass master lane in a multilane application. Connects to RXSYNCIN port of all lanes that are part of a multi-lane application.
RXSYNCDONE	Out	Async	Indicates RX buffer bypass alignment procedure completion. Only valid for RX buffer bypass master lane.

Table 4-25 defines the RX buffer attributes.

Table 4-25: **RX Buffer Attributes**

Attribute	Type	Description
RXBUF_EN	Boolean	Use or bypass the RX elastic buffer. TRUE: Uses the RX elastic buffer (default). FALSE: Bypasses the RX elastic buffer (advanced feature).
RX_XCLK_SEL	String	Selects the clock source used to drive the RX parallel clock domain (XCLK). RXREC: Selects the RX recovered clock as source of XCLK. Used when using the RX elastic buffer. RXUSR: Selects RXUSRCLK as the source of XCLK. Used when bypassing the RX elastic buffer.
RXPH_CFG	16-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

Table 4-25: **RX Buffer Attributes *(Cont'd)***

Attribute	Type	Description
RXPH_MONITOR_SEL	5-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RXPHDLY_CFG	24-bit Binary	RX phase and delay alignment configuration. RXPHDLY_CFG[19] = 1 is used to set the RX delay alignment tap to the full range of ±4 ns. RXPHDLY_CFG[19] = 0 is used to set the RX delay alignment tap to the half range of ±2 ns. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RXDLY_CFG	16-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RXDLY_LCFG	9-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RXDLY_TAP_CFG	16-bit Binary	Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RX_DDI_SEL	6-bit Binary	RX data delay insertion select. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RXSYNC_OVRD	1-bit Binary	Manual mode override. 0: RX buffer bypass auto mode is enabled. 1: RX buffer bypass manual mode is used. TX buffer bypass control must be implemented in FPGA logic.
RXSYNC_SKIP_DA	1-bit Binary	Control to skip Delay Alignment procedure. Only valid on buffer bypass master lane. 0: RX delay alignment procedure occurs. 1: RX delay alignment procedure is skipped.
RXSYNC_MULTILANE	1-bit Binary	Indicates whether the lane is used as part of a multi-lane interface. Only valid on RX buffer bypass master lane. 0: This lane is used in single-lane mode. 1: This lane is used in multi-lane mode.

RX Elastic Buffer

Functional Description

The GTP transceiver RX datapath has two internal parallel clock domains used in the PCS: The PMA parallel clock domain (XCLK) and the RXUSRCLK domain. To receive data, the PMA parallel rate must be sufficiently close to the RXUSRCLK rate, and all phase differences between the two domains must be resolved. Figure 4-25 shows the two parallel clock domains: XCLK and RXUSRCLK.

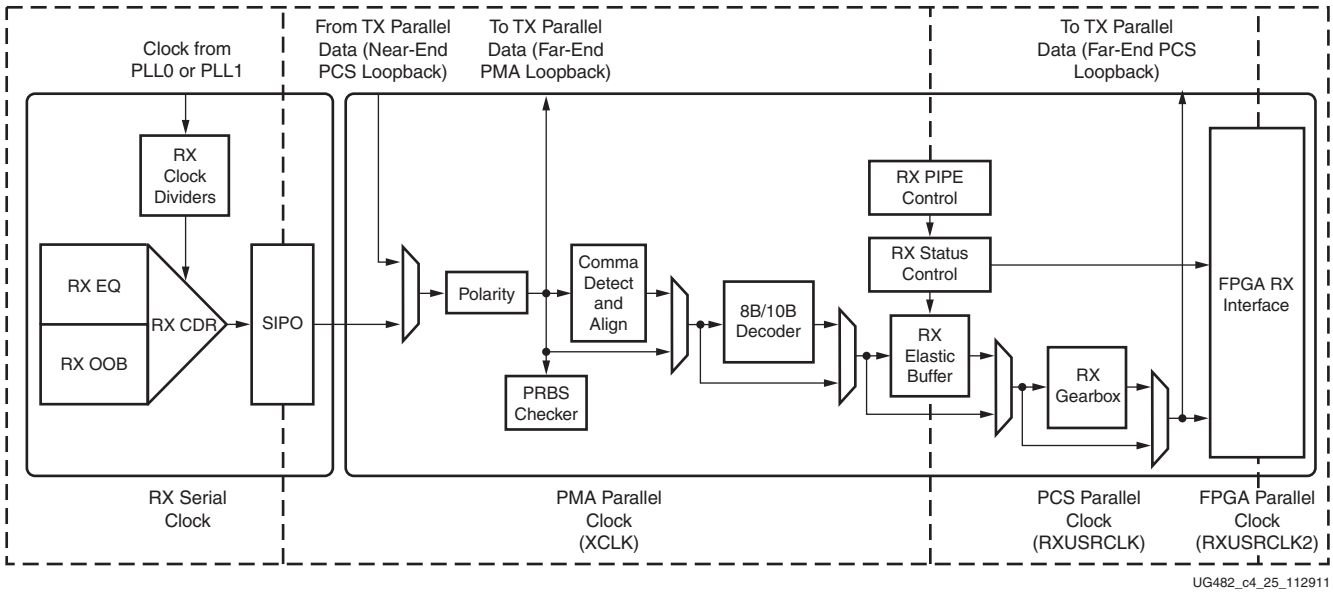


Figure 4-25: **RX Clock Domains**

The GTP transceiver includes an RX elastic buffer to resolve differences between the XCLK and RXUSRCLK domains. The phase of the two domains can also be matched by using the RX recovered clock from the transceiver to drive RXUSRCLK and adjusting its phase to match XCLK when the RX buffer is bypassed (see RX Buffer Bypass, page 134). All RX datapaths must use one of these approaches. The costs and benefits of each approach are shown in Table 4-26.

Table 4-26: **RX Buffering versus Phase Alignment**

	RX Elastic Buffer	RX Phase Alignment
Ease of Use	The RX buffer is the recommended default to use when possible. It is robust and easier to operate.	Phase alignment is an advanced feature that requires extra logic and additional constraints on clock sources. RXOUTCLKSEL must select the RX recovered clock as the source of RXOUTCLK to drive RXUSRCLK.
Clocking Options	Can use RX recovered clock or local clock (with clock correction).	Must use the RX recovered clock.
Initialization	Works immediately.	Must wait for all clocks to stabilize before performing the RX phase and delay alignment procedure.

Table 4-26: **RX Buffering versus Phase Alignment *(Cont'd)***

	RX Elastic Buffer	RX Phase Alignment
Latency	Buffer latency depends on features use, such as clock correction and channel bonding.	Lower deterministic latency.
Clock Correction and Channel Bonding	Required for clock correction and channel bonding.	Not performed inside the transceiver. Required to be implemented in user logic.

Ports and Attributes

Table 4-27 defines the RX buffer ports.

Table 4-27: **RX Buffer Ports**

Port	Dir	Clock Domain	Description
RXBUFRESET	In	Async	Resets and reinitializes the RX elastic buffer.
RXBUFSTATUS[2:0]	Out	RXUSRCLK2	RX buffer status. 000b: Nominal condition. 001b: Number of bytes in the buffer are less than CLK_COR_MIN_LAT 010b: Number of bytes in the buffer are greater than CLK_COR_MAX_LAT 101b: RX elastic buffer underflow 110b: RX elastic buffer overflow

Table 4-28 defines the RX buffer attributes.

Table 4-28: **RX Buffer Attributes**

Attribute	Type	Description
RXBUF_EN	Boolean	Use or bypass the RX elastic buffer. TRUE: Uses the RX elastic buffer (default). FALSE: Bypasses the RX elastic buffer (advanced feature).
RX_XCLK_SEL	String	Selects the clock source used to drive the RX parallel clock domain (XCLK). RXREC: Selects the RX recovered clock as the source of XCLK. Used when using the RX elastic buffer. RXUSR: Selects RXUSRCLK as the source of XCLK. Used when bypassing the RX elastic buffer.

Table 4-28: **RX Buffer Attributes *(Cont'd)***

Attribute	Type	Description
RX_BUFFER_CFG	6-bit Binary	RX elastic buffer configuration. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RX_DEFER_RESET_BUF_EN	Boolean	Defer RX elastic buffer reset on comma realignment. The time deferred is controlled by RXBUF_EIDLE_HI_CNT. TRUE: Enables deferral of RX elastic buffer reset on comma realignment. FALSE: Disables deferral of RX elastic buffer reset on comma realignment.
RXBUF_ADDR_MODE	String	RX elastic buffer address mode. FULL: Enables the RX elastic buffer for clock correction and channel bonding support. The maximum frequency is limited. FAST: Enables the RX elastic buffer for phase compensation without clock correction and channel bonding support. This mode is recommended for high line rates.
RXBUF_EIDLE_HI_CNT	4-bit Binary	Controls the timing of asserting the GTP transceiver internally generated RX elastic buffer reset on electrical idle when valid data is not present on the RXP/RXN serial lines. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RXBUF_EIDLE_LO_CNT	4-bit Binary	Controls the timing of deasserting the GTP transceiver internally generated RX elastic buffer reset on electrical idle when valid data is present on the RXP/RXN serial lines. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

Table 4-28: **RX Buffer Attributes *(Cont'd)***

Attribute	Type	Description
RXBUF_RESET_ON_CB_CHANGE	Boolean	GTP transceiver internally generated RX elastic buffer reset on channel bonding change. TRUE: Enables auto RX elastic buffer reset on channel bonding change. FALSE: Disables auto RX elastic buffer reset on channel bonding change.
RXBUF_RESET_ON_COMMAALIGN	Boolean	GTP transceiver internally generated RX elastic buffer reset on comma realignment. TRUE: Enables auto RX elastic buffer reset on comma alignment. FALSE: Disables auto RX elastic buffer reset on comma alignment.
RXBUF_RESET_ON_EIDLE	Boolean	GTP transceiver internally generated RX elastic buffer reset on electrical idle. TRUE: Enables auto RX elastic buffer reset on electrical idle. FALSE: Disables auto RX elastic buffer reset on electrical idle.
RXBUF_RESET_ON_RATE_CHANGE	Boolean	GTP transceiver internally generated RX elastic buffer reset on rate change. TRUE: Enables auto RX elastic buffer reset on rate change. FALSE: Disables auto RX elastic buffer reset on rate change.
RXBUF_THRESH_OVRD	Boolean	RX elastic buffer threshold override. TRUE: Use the RXBUF_THRESH_OVFLW and RXBUF_THRESH_UNDFLW attributes to set the RX elastic buffer overflow and underflow thresholds, respectively. FALSE: Automatically calculates the RX elastic buffer overflow and underflow thresholds. This is the recommended default setting.

Table 4-28: **RX Buffer Attributes *(Cont'd)***

Attribute	Type	Description
RXBUF_THRESH_OVFLW	Integer	RX elastic buffer overflow threshold specified as the number of bytes. If the data latency through the RX elastic buffer is at or above this threshold, the buffer is considered to be in an overflow condition. Used when RXBUF_THRESH_OVRD = TRUE. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RXBUF_THRESH_UNDFLW	Integer	RX elastic buffer underflow threshold specified as number of bytes. If the data latency through the RX elastic buffer is at or below this threshold, the buffer is consider to be in underflow condition. Used when RXBUF_THRESH_OVRD = TRUE. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.
RXBUFRESET_TIME	5-bit Binary	RX elastic buffer reset time. Reserved. The recommended value from the 7 Series FPGAs Transceivers Wizard should be used.

Using the RX Elastic Buffer

These settings are used to enable the RX elastic buffer to resolve phase differences between the XCLK and RXUSRCLK domains:

- RXBUF_EN = TRUE
- RX_XCLK_SEL = RXREC

The content of the RX elastic buffer becomes invalid if an RX elastic buffer overflow or underflow condition occurs. When any of these conditions occur, the RX elastic buffer should be reset and reinitialized by using GTRXRESET, RXPCSRESET, RXBUFRESET, or the GTP transceiver internally generated RX elastic buffer reset. The internally generated RX elastic buffer reset can occur on channel bonding change, comma realignment, electrical idle, or rate change conditions.

The RX elastic buffer is also used for clock correction (see RX Clock Correction) and channel bonding (see RX Channel Bonding, page 154). Clock correction is used in cases where XCLK and RXUSRCLK are not frequency matched. Table 4-29 lists common clock configurations and shows whether they require clock correction.

Table 4-29: **Common Clock Configurations**

Types of Clocking	Require Clock Correction?
Synchronous system where both sides uses the reference clock from the same physical oscillator.	No
Asynchronous system when separate reference clocks are used and the GTP transceiver receiver uses an RX recovered clock.	No
Asynchronous system when separate reference clocks are used and the GTP transceiver receiver uses a local clock.	Yes

RX Clock Correction

Functional Description

The RX elastic buffer is designed to bridge between two different clock domains, RXUSRCLK and XCLK, which is the recovered clock from CDR. Even if RXUSRCLK and XCLK are running at same clock frequency, there is always a small frequency difference. Because XCLK and RXUSRCLK are not exactly the same, the difference can be accumulated to cause the RX elastic buffer to eventually overflow or underflow unless it is corrected. To allow correction, each GTP transceiver TX periodically transmits one or more special characters that the GTP transceiver RX is allowed to remove or replicate in the RX elastic buffer as necessary. By removing characters when the RX elastic buffer is too full and replicating characters when the RX elastic buffer is too empty, the receiver can prevent overflow or underflow.

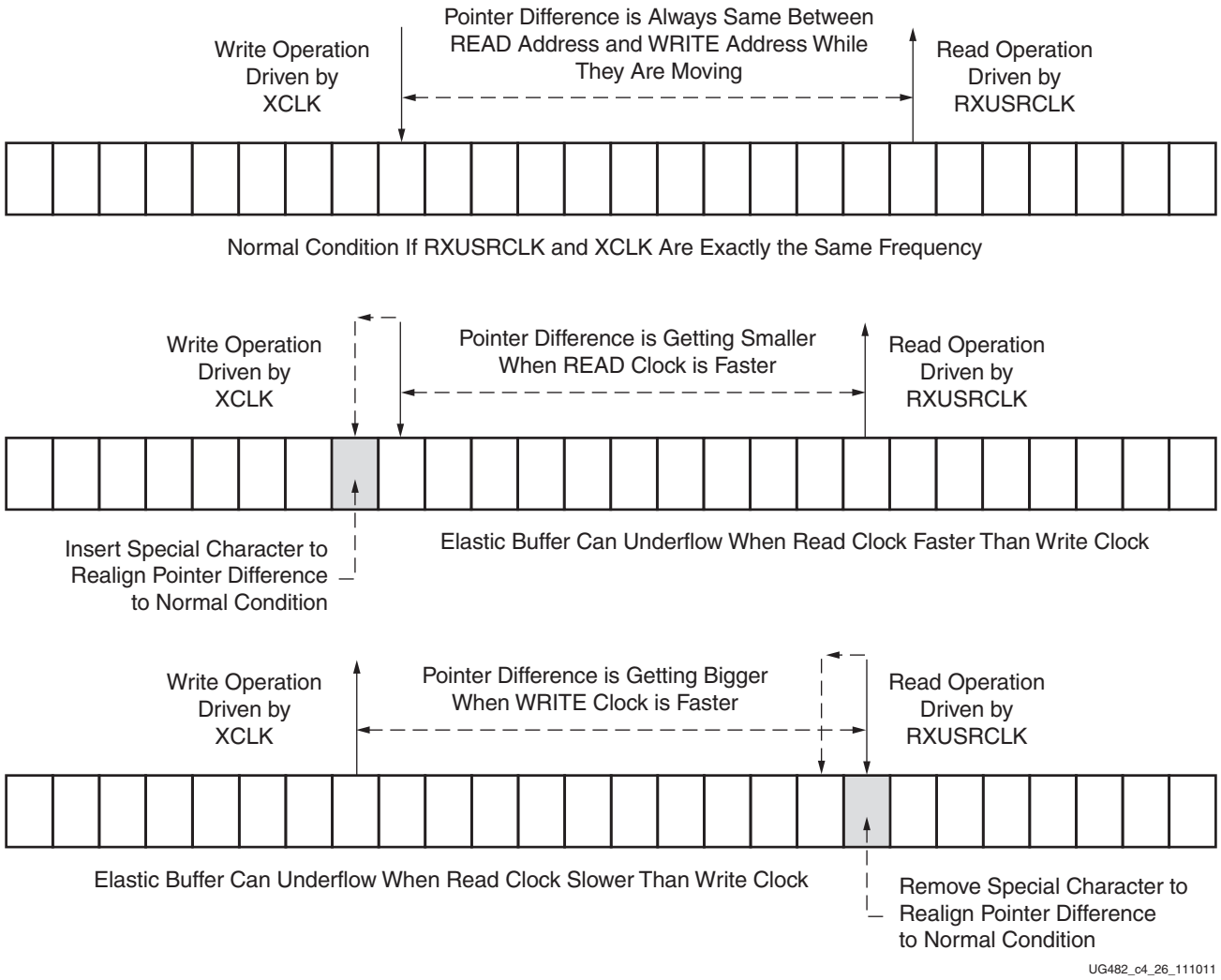


Figure 4-26: **Clock Correction Conceptual View**

Ports and Attributes

Table 4-30 defines the ports required by RX clock correction functions.

Table 4-30: **RX Clock Correction Ports**

Port	Dir	Clock Domain	Description
RXBUFRESET	In	Async	Resets the RX elastic buffer and related logic.
RXBUFSTATUS[2:0]	Out	RXUSRCLK2	Indicates the status of the RX elastic buffer: `000`: In nominal operating range where the buffer occupancy is within the CLK_COR_MIN_LAT and CLK_COR_MAX_LAT range `001`: RX elastic buffer occupancy is less than CLK_COR_MIN_LAT `010`: RX elastic buffer occupancy is greater than CLK_COR_MAX_LAT `101`: RX elastic buffer underflow `110`: RX elastic buffer overflow
RXCLKCORCNT[1:0]	Out	RXUSRCLK2	Reports the clock correction status of the RX elastic buffer when the first byte of a clock correction sequence is shown in RXDATA. `00`: No clock correction `01`: One sequence skipped `10`: Two sequences skipped `11`: One sequence added
RX8B10BEN	In	RXUSRCLK2	Active High to enable the 8B/10B decoder in the GTP transceiver RX. If 8B/10B decoding is enabled, RX_DATA_WIDTH must be a multiple of 10 (20, 40). If 8B/10B decoding is not enabled, RX_DATA_WIDTH must be a multiple of 8 (16, 32).

Table 4-31 defines the attributes required by RX channel bonding.

Table 4-31: **RX Clock Correction Attributes**

Attribute	Type	Description
CBCC_DATA_SOURCE_SEL	String	This attribute is used together with RX8B10BEN to select the data source for clock correction and channel bonding. When RX8B10BEN is High, CBCC_DATA_SOURCE_SEL = DECODED, the clock correction sequence matches the data decoded after the 8B/10B decoder. CBCC_DATA_SOURCE_SEL = ENCODED, the clock correction sequence matches the raw data from the comma detection and realignment block before the 8B/10B decoder. When RX8B10BEN is Low, CBCC_DATA_SOURCE_SEL = DECODED is not supported. CBCC_DATA_SOURCE_SEL = ENCODED, the clock correction sequence matches the raw data from the comma detection and realignment block.
CLK_CORRECT_USE	Boolean	Set TRUE to enable the clock correction function. Set FALSE to disable the clock correction function. These attributes need to be set while clock correction disabled: CLK_COR_SEQ_1_1 = `10'b0100000000` CLK_COR_SEQ_2_1 = `10'b0100000000` CLK_COR_SEQ_1_ENABLE = `4'b1111` CLK_COR_SEQ_2_ENABLE = `4'b1111`
CLK_COR_KEEP_IDLE	Boolean	Set TRUE to keep at least one clock correction sequence in the data stream for every continuous stream of clock correction sequences received. Set FALSE to remove all clock correction sequences from the byte stream if needed to recenter the RX elastic buffer range.
CLK_COR_MAX_LAT	Integer	Specifies the maximum RX elastic buffer latency. If the RX elastic buffer exceeds CLK_COR_MAX_LAT, the clock correction circuit removes incoming clock correction sequences to prevent overflow. Valid values for this attribute range from 3 to 48. The 7 Series FPGAs Transceivers Wizard chooses an optimal CLK_COR_MAX_LAT value based on application requirements.

Table 4-31: **RX Clock Correction Attributes *(Cont'd)***

Attribute	Type	Description
CLK_COR_MIN_LAT	Integer	Specifies the minimum RX elastic buffer latency. If the RX elastic buffer drops below CLK_COR_MIN_LAT, the clock correction circuit replicates incoming clock correction sequences to prevent underflow. When the RX elastic buffer is reset, its pointers are set so that there are CLK_COR_MIN_LAT unread (and uninitialized) data bytes in the buffer. Valid values for this attribute range from 3 to 48. Refer to Table 4-32 for the restriction. The 7 Series FPGAs Transceivers Wizard chooses a CLK_COR_MIN_LAT value based on application requirements.
CLK_COR_PRECEDENT	Boolean	Determines whether clock correction or channel bonding takes precedence when both operations are triggered at the same time. TRUE: Clock correction takes precedence over channel bonding if there is opportunity for both FALSE: Channel bonding takes precedence over clock correction if there is opportunity for both
CLK_COR_REPEAT_WAIT	Integer	This attribute specifies the minimum number of RXUSRCLK cycles between two successive clock corrections being placed. If this attribute is 0, no limit is placed on how frequently the clock correction character can be placed. Valid values for this attribute range from 0 to 31.
CLK_COR_SEQ_LEN	Integer	Defines the length of the sequence in bytes that has to match to detect opportunities for clock correction. This attribute also defines the size of the adjustment (number of bytes repeated or skipped) in a clock correction. Valid lengths are 1, 2, and 4 bytes.

Table 4-31: **RX Clock Correction Attributes *(Cont'd)***

Attribute	Type	Description
CLK_COR_SEQ_1_ENABLE	4-bit Binary	Mask enable bit for the first clock correction sequence. CLK_FOR_SEQ_1_ENABLE[0] is the mask bit for CLK_COR_SEQ_1_1. CLK_FOR_SEQ_1_ENABLE[1] is the mask bit for CLK_COR_SEQ_1_2. CLK_FOR_SEQ_1_ENABLE[2] is the mask bit for CLK_COR_SEQ_1_3. CLK_FOR_SEQ_1_ENABLE[3] is the mask bit for CLK_COR_SEQ_1_4. When CLK_FOR_SEQ_1_ENABLE[*] is 0, the corresponding CLK_COR_SEQ_1_* is either considered as a don't care or is matched automatically without a comparison. When CLK_FOR_SEQ_1_ENABLE[*] is 1, the corresponding CLK_COR_SEQ_1_* is compared for a match.
CLK_COR_SEQ_1_1	10-bit Binary	First clock correction sequence 1 to be compared when CLK_FOR_SEQ_1_ENABLE[0] = 1.
CLK_COR_SEQ_1_2	10-bit Binary	First clock correction sequence 2 to be compared when CLK_FOR_SEQ_1_ENABLE[1] = 1.
CLK_COR_SEQ_1_3	10-bit Binary	First clock correction sequence 3 to be compared when CLK_FOR_SEQ_1_ENABLE[2] = 1.
CLK_COR_SEQ_1_4	10-bit Binary	First clock correction sequence 4 to be compared when CLK_FOR_SEQ_1_ENABLE[3] = 1.
CLK_COR_SEQ_2_USE	Boolean	Set to TRUE if the second clock correction sequence (CLK_COR_SEQ_2_*) is used in addition to the CLK_COR_SEQ_1_* that is always used.

Table 4-31: **RX Clock Correction Attributes *(Cont'd)***

Attribute	Type	Description
CLK_COR_SEQ_2_ENABLE	4-bit Binary	Mask enable bit for the second clock correction sequence. CLK_FOR_SEQ_2_ENABLE[0] is the mask bit for CLK_COR_SEQ_2_1. CLK_FOR_SEQ_2_ENABLE[1] is the mask bit for CLK_COR_SEQ_2_2. CLK_FOR_SEQ_2_ENABLE[2] is the mask bit for CLK_COR_SEQ_2_3. CLK_FOR_SEQ_2_ENABLE[3] is the mask bit for CLK_COR_SEQ_2_4. When CLK_FOR_SEQ_2_ENABLE[*] is 0, the corresponding CLK_COR_SEQ_2_* is either considered as a don't care or is matched automatically without a comparison. When CLK_FOR_SEQ_2_ENABLE[*] is 1, the corresponding CLK_COR_SEQ_2_* is compared for a match.
CLK_COR_SEQ_2_1	10-bit Binary	Second clock correction sequence 1 to be compared when CLK_FOR_SEQ_2_ENABLE[0] = 1
CLK_COR_SEQ_2_2	10-bit Binary	Second clock correction sequence 2 to be compared when CLK_FOR_SEQ_2_ENABLE[1] = 1
CLK_COR_SEQ_2_3	10-bit Binary	Second clock correction sequence 3 to be compared when CLK_FOR_SEQ_2_ENABLE[2] = 1
CLK_COR_SEQ_2_4	10-bit Binary	Second clock correction sequence 4 to be compared when CLK_FOR_SEQ_2_ENABLE[3] = 1
RX_DATA_WIDTH	Integer	Sets the bit width of the RXDATA port. When 8B/10B encoding is enabled, RX_DATA_WIDTH must be set to 20 or 40. Valid settings are 16, 20, 32, and 40. See Interface Width Configuration, page 175 for more details.

Table 4-31: **RX Clock Correction Attributes *(Cont'd)***

Attribute	Type	Description
RX_DISPERR_SEQ_MATCH	Boolean	Specifies whether the disparity error status of a decoded byte must match the indicator in the channel bonding and clock correction sequence. TRUE: The disparity error status must be matched. FALSE: The disparity error status is ignored.
ALIGN_COMMA_WORD	Integer	This attribute controls the alignment of detected commas within a multi-byte datapath. 1: Align the comma to either of the 2 bytes for a 2-byte interface and any of the 4 bytes for a 4-byte interface. The comma can be aligned to either the even bytes or the odd bytes of the RXDATA output. 2: Align the comma to the even bytes only. The aligned comma is guaranteed to be aligned to even bytes RXDATA[9:0] for a 2-byte interface and RXDATA[9:0]/RXDATA[29:20] for a 4-byte interface. Refer to Figure 4-19 for comma alignment boundaries that are allowed for the different ALIGN_COMMA_WORD and RX_DATA_WIDTH settings. Protocols that send commas in even and odd positions must set ALIGN_COMMA_WORD to 1.

Using RX Clock Correction

The user must follow the steps described in this section to use the receiver's clock correction feature.

Enabling Clock Correction

Each GTP transceiver includes a clock correction circuit that performs clock correction by controlling the pointers of the RX elastic buffer. To use clock correction, RXBUF_EN is set to TRUE to turn on the RX elastic buffer, and CLK_CORRECT_USE is set to TRUE to turn on the clock correction circuit.

Clock correction is triggered when the RX elastic buffer latency is too high or too low, and the clock correction circuit detects a match sequence. To use clock correction, the clock correction circuit must be configured to set these items:

- RX elastic buffer limits
- Clock correction sequence

Setting RX Elastic Buffer Limits

The RX elastic buffer limits are set using CLK_COR_MIN_LAT (minimum latency) and CLK_COR_MAX_LAT (maximum latency). When the number of bytes in the RX elastic buffer drops below CLK_COR_MIN_LAT, the clock correction circuit writes an additional CLK_COR_SEQ_LEN byte from the first clock correction sequence it matches to prevent buffer underflow. Similarly, when the number of bytes in the RX elastic buffer exceeds CLK_COR_MAX_LAT, the clock correction circuit deletes CLK_COR_SEQ_LEN bytes from the first clock correction sequence it matches, starting with the first byte of the sequence. The 7 Series FPGAs Transceivers Wizard chooses an optimal setting for CLK_COR_MIN_LAT and CLK_COR_MAX_LAT based on application requirements.

Because CLK_COR_MIN_LAT is used to set the initial RX elastic buffer latency, it must be divisible by the ALIGN_COMMA_WORD setting to preserve the comma alignment through the elastic buffer. The value for CLK_COR_MIN_LAT must comply with ALIGN_COMMA_WORD as shown in Table 4-32.

The CLK_COR_MAX_LAT setting has no impact on the RX elastic buffer latency that is established, so it can be set any value from 3 to 48.

Table 4-32: **CLK_COR_MIN_LAT Setting Restriction**

ALIGN_COMMA_WORD	CLK_COR_MIN_LAT
1	No restriction, any value from 3 to 48.
2	Must be divisible by 2.

Setting Clock Correction Sequences

The clock correction sequences are programmed using the CLK_COR_SEQ_1_* attributes and CLK_COR_SEQ_LEN. Each CLK_COR_SEQ_1_* attribute corresponds to one subsequence in clock correction sequence 1. CLK_COR_SEQ_LEN is used to set the number of subsequences to be matched. If the 40-bit or 20-bit internal datapaths are used, the clock correction circuit matches all 10 bits of each subsequence. If the 16-bit or 32-bit internal datapaths are used, only the right-most eight bits of each subsequence are used.

A second, alternate clock correction sequence can be activated by setting CLK_COR_SEQ_2_USE to TRUE. The first and second sequences share length settings, but use different subsequence values for matching. Set the CLK_COR_SEQ_2_* attributes to define the subsequence values for the second sequence.

When using 8B/10B decoding (RX8B10BEN is High), CBCC_DATA_SOURCE_SEL is set to DECODED to search the output of the 8B/10B decoder for sequence matches instead of non-decoded data. This allows the circuit to look for 8-bit values with either positive or negative disparity, and to distinguish K characters from regular characters (see TX 8B/10B Encoder, page 51 and RX 8B/10B Decoder, page 129 for details). Figure 4-27 shows how to set a clock correction sequence byte when RX8B10BEN is High and CBCC_DATA_SOURCE_SEL is set to DECODED.

When CBCC_DATA_SOURCE_SEL is set to ENCODED, the sequence must exactly match incoming raw data. When RX_DISPERR_SEQ_MATCH is set to FALSE, CLK_COR_SEQ_x_y[9] is not used for matching.

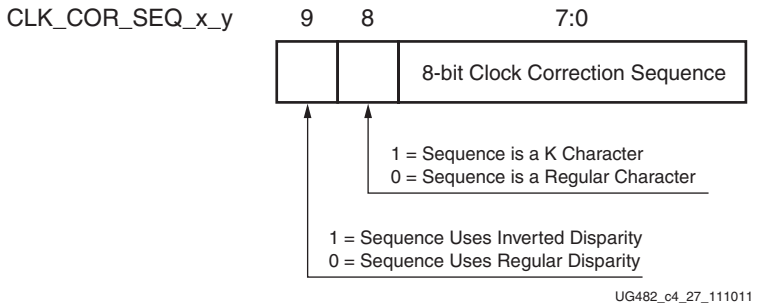


Figure 4-27: **Clock Correction Subsequence Settings with CBCC_DATA_SOURCE_SEL = DECODED**

Some protocols use clock correction sequences with don't care subsequences. The clock correction circuit can be programmed to recognize these sequences using CLK_COR_SEQ_1_ENABLE and CLK_COR_SEQ_2_ENABLE. When the enable bit for a sequence is Low, that byte is considered matched no matter what the value is. Figure 4-28 shows the mapping between the clock correction sequences and the clock correction sequence enable bits.

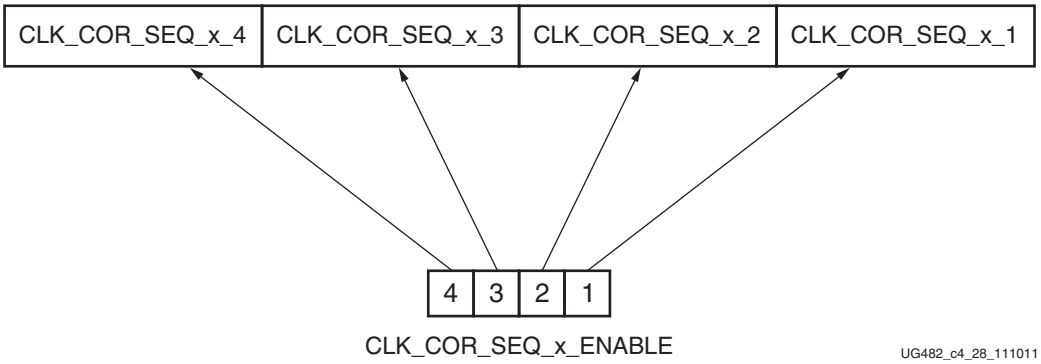


Figure 4-28: **Clock Correction Sequence Mapping**

To preserve comma alignment through the elastic buffer, CLK_COR_SEQ_LEN and ALIGN_COMMA_WORD must be selected such that they comply with Table 4-33.

Table 4-33: **Valid ALIGN_COMMA_WORD/CLK_COR_SEQ_LEN Combinations**

ALIGN_COMMA_WORD	CLK_COR_SEQ_LEN
1	1, 2, 4
2	2, 4

Clock Correction Options

CLK_COR_REPEAT_WAIT is used to control the clock correction frequency. This value is set to the minimum number of RXUSRCLK cycles required between clock correction events. This attribute is set to 0 to allow clock correction to at occur any time. Some protocols allow clock correction to occur at any time, but require that if the clock correction circuit removes sequences, at least one sequence stays in the stream. For protocols with this requirement, CLK_COR_KEEP_IDLE is set to TRUE.

Monitoring Clock Correction

The clock correction circuit can be monitored using the RXCLKCORCNT and RXBUFSTATUS ports. The RXCLKCORCNT entry in Table 4-30 shows how to decode the values of RXCLKCORCNT to determine the status of the clock correction circuit. The RXBUFSTATUS entry in Table 4-30 shows how to decode the values of RXBUFSTATUS to determine how full the RX elastic buffer is.

RX Channel Bonding

Functional Description

Protocols such as XAUI and PCI Express combine multiple serial transceiver connections to create a single higher throughput channel. Each serial transceiver connection is called one lane. Unless each of the serial connections is exactly the same length, skew between the lanes can cause data to be transmitted at the same time but arrive at different times. Channel bonding cancels out the skew between GTP transceiver lanes by using the RX elastic buffer as a variable latency block. Channel bonding is also called channel deskew or lane-to-lane deskew. GTP transceiver transmitters used for a bonded channel all transmit a channel bonding character (or a sequence of characters) simultaneously. When the sequence is received, the GTP transceiver receiver can determine the skew between each lane and adjust the latency of RX elastic buffers, so that data is presented without skew at the RX fabric user interface.

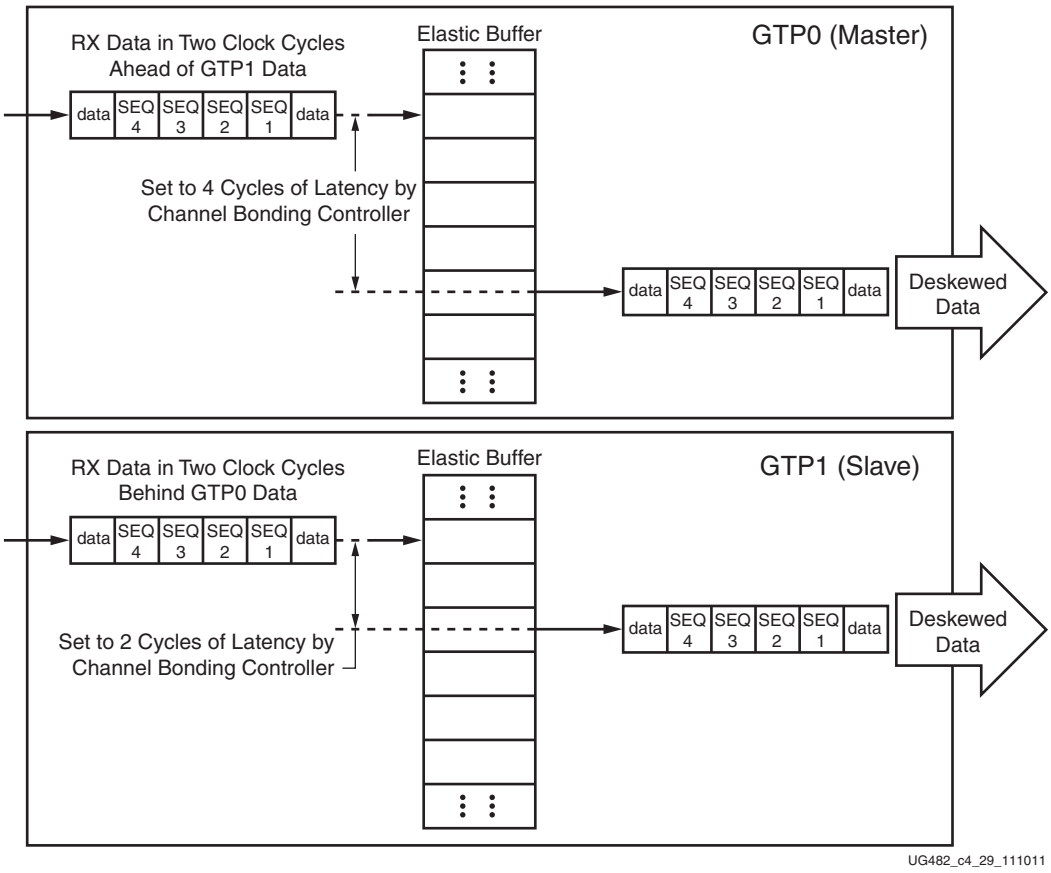


Figure 4-29: **Channel Bonding Conceptual View**

RX channel bonding supports 8B/10B encoded data but does not support these encoded data types:

- 64B/66B
- 64B/67B
- 128B/130B
- Scrambled data

Ports and Attributes

Table 4-34 defines the ports required by RX channel bonding functions.

Table 4-34: **RX Channel Bonding Ports**

Port	Dir	Clock Domain	Description
RXCHANBONDSEQ	Out	RXUSRCLK2	This port goes High when RXDATA contains the start of a channel bonding sequence.
RXCHANISALIGNED	Out	RXUSRCLK2	This signal from the RX elastic buffer goes High to indicate that the channel is properly aligned with the master transceiver according to observed channel bonding sequences in the data stream. This signal goes Low if an unaligned channel bonding sequence is detected, indicating that channel alignment was lost.
RXCHANREALIGN	Out	RXUSRCLK2	This signal from the RX elastic buffer is held High for at least one cycle when the receiver has changed the alignment between this transceiver and the master.
RXCHBONDI[3:0]	In	RXUSRCLK	Channel bonding control ports used by slaves only. These ports are used to receive channel bonding and clock correction control information from master GTP transceiver RXCHBONDO ports or from daisy-chained slave GTP transceiver RXCHBONDO ports, which are concatenated from the master GTP transceiver.
RXCHBONDO[3:0]	Out	RXUSRCLK	Channel bonding control ports used to propagate channel bonding and clock correction information to the slave GTP transceiver from the master or a daisy-chained slave concatenated from the master. The master RXCHBONDO can be tied to one or multiple slave RXCHBONDI ports. The slave RXCHBONDO should be tied to the next level slave RXCHBONDI to form a daisy chain and pass information from the master to each slave.

Table 4-34: **RX Channel Bonding Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
RXCHBONDLEVEL[2:0]	In	RXUSRCLK2	Indicates the amount of internal pipelining used for the RX elastic buffer control signals. A higher value permits more daisy chaining of RXCHBONDO and RXCHBONDI to ease placement and routing constraints. To minimize required latency through the RX elastic buffer, CHAN_BOND_LEVEL in the master is set to the smallest value possible for the required amount of daisy-chaining.
RXCHBONDMASTER	In	RXUSRCLK2	Indicates that the transceiver is the master for channel bonding. Its RXCHBONDO port directly drives the RXCHBONDI ports on one or more slave transceivers. This port cannot be driven High at the same time as RXCHBONDSLAVE.
RXCHBONDSLAVE	In	RXUSRCLK2	Indicates that this transceiver is a slave for channel bonding. Its RXCHBONDI port is directly driven by the RXCHBONDO port of another slave or master transceiver. If its RXCHBONDLEVEL[2:0] setting is greater than 0, its RXCHBONDO port can directly drive the RXCHBONDI ports on one or more other slave transceivers. This port cannot be driven High at the same time as RXCHBONDMASTER.
RXCHBONDEN	In	RXUSRCLK2	This port enables channel bonding (from the FPGA logic to both the master and slaves).

Table 4-35 defines the attributes required by RX channel bonding.

Table 4-35: **RX Channel Bonding Attributes**

Attribute	Type	Description
CHAN_BOND_MAX_SKEW	Integer	This attribute controls the number of USRCLK cycles that the master waits before ordering the slaves to execute channel bonding. This attribute determines the maximum skew that can be handled by channel bonding. It must always be less than one-half the minimum distance (in bytes or 10-bit codes) between channel bonding sequences. Valid values range from 1 to 14.
CHAN_BOND_KEEP_ALIGN	Boolean	Allows preservation of ALIGN characters during channel bonding for PCI Express.
CHAN_BOND_SEQ_1_1 CHAN_BOND_SEQ_1_2 CHAN_BOND_SEQ_1_3 CHAN_BOND_SEQ_1_4	10-bit Binary	The CHAN_BOND_SEQ_1 attributes are used in conjunction with CHAN_BOND_SEQ_1_ENABLE to define channel bonding sequence 1. Each subsequence is 10 bits long. The rules for setting the subsequences depend on RX_DATA_WIDTH and CBCC_DATA_SOURCE_SEL. Not all subsequences need to be used. CHAN_BOND_SEQ_LEN determines how much of the sequence is used for a match. If CHAN_BOND_SEQ_LEN = 1, only CHAN_BOND_SEQ_1_1 is used. CHAN_BOND_SEQ_1_ENABLE can be used to make parts of the sequence don't care. If CHAN_BOND_SEQ_1_ENABLE[k] is 0, CHAN_BOND_SEQ_1_k is a don't-care subsequence and is always considered to be a match.
CHAN_BOND_SEQ_1_ENABLE	4-bit Binary	

Table 4-35: **RX Channel Bonding Attributes *(Cont'd)***

Attribute	Type	Description
CHAN_BOND_SEQ_2_1 CHAN_BOND_SEQ_2_2 CHAN_BOND_SEQ_2_3 CHAN_BOND_SEQ_2_4	10-bit Binary	The CHAN_BOND_SEQ_2 attributes are used in conjunction with CHAN_BOND_SEQ_2_ENABLE to define the second channel bonding sequence. When CHAN_BOND_SEQ_2_USE is TRUE, the second sequence is used as an alternate sequence to trigger channel bonding. Each subsequence is 10 bits long. The rules for setting the subsequence depend on RX_DATA_WIDTH and CBCC_DATA_SOURCE_SEL. Not all subsequences need to be used. CHAN_BOND_SEQ_LEN determines how many of the subsequences are used for a match. If CHAN_BOND_SEQ_LEN = 1, only CHAN_BOND_SEQ_2_1 is used. CHAN_BOND_SEQ_2_ENABLE can be used to make parts of the sequence don't care. If CHAN_BOND_SEQ_2_ENABLE[k] is 0, CHAN_BOND_SEQ_2_k is a don't-care subsequence and is always considered to be a match.
CHAN_BOND_SEQ_2_ENABLE	4-bit Binary	
CHAN_BOND_SEQ_2_USE	Boolean	Determines if the two-channel bonding sequence is to be used. TRUE: Channel bonding can be triggered by channel bonding sequence 1 or 2. FALSE: Channel bonding is only triggered by sequence 1.
CHAN_BOND_SEQ_LEN	Integer	Defines the length in bytes of the channel bonding sequence that the GTP transceiver has to match to find skew. Valid lengths are 1, 2, and 4 bytes.
CBCC_DATA_SOURCE_SEL	String	This attribute is used to select the data source for clock correction and channel bonding. When set to DECODED, selects data from the 8B/10B decoder when RX8B10BEN is High. When set to ENCODED, selects data from the comma detection and realignment block.

Table 4-35: **RX Channel Bonding Attributes *(Cont'd)***

Attribute	Type	Description
FTS_DESKEW_SEQ_ENABLE	4-bit Binary	Enable mask for FTS_LANE_DESKEW_CFG. FTS_DESKEW_SEQ_ENABLE[0] is for FTS_LANE_DESKEW_CFG[0] FTS_DESKEW_SEQ_ENABLE[1] is for FTS_LANE_DESKEW_CFG[1] FTS_DESKEW_SEQ_ENABLE[2] is for FTS_LANE_DESKEW_CFG[2] FTS_DESKEW_SEQ_ENABLE[3] is for FTS_LANE_DESKEW_CFG[3] The default value is `1111`.
FTS_LANE_DESKEW_CFG	4-bit Binary	• Bit 3: This bit is set to `1'b1` on a slave to freeze the alignment to prevent spurious misalignments or modified alignments that can occur following slip-4, snap-4, or clock correction when good channel alignment is still maintained. This bit is set to `1'b0` on a slave to unfreeze the alignment. • Bit 2: Specifies whether a "master" channel doing FTS lane deskew that just reached the end of an FTS OS in its lookahead control logic inhibits its own generation of clock correction commands for a brief time. The purpose is to prevent clock correction commands from interfering with operation of the slave's slip-4 and snap-4 logic. The logic guarantees that clock correction can still occur if a full SKP OS is present. • Bit 1: Specifies whether a "slave" channel doing FTS lane deskew is permitted (`1'b1`) or inhibited (`1'b0`) from performing an immediate backward alignment adjustment by four bytes (slip-4) if the slave is found to have reached the SKP OS following FTS before the master has. • Bit 0: Specifies whether a "slave" channel doing FTS lane deskew is permitted (`1'b1`) or inhibited (`1'b0`) from performing an immediate forward alignment adjustment by four bytes (snap-4) if the master is found to have reached the SKP OS following FTS before the slave has.

Table 4-35: **RX Channel Bonding Attributes *(Cont'd)***

Attribute	Type	Description
FTS_LANE_DESKEW_EN	Boolean	This attribute is set to TRUE to enable channel bonding logic for FTS lane deskew. FTS lane deskew is separate from the standard algorithm using channel bonding sequences 1 and 2, and it operates in parallel with the standard algorithm. FTS lane deskew operates only in two-byte mode.
PCS_PCIE_EN	Boolean	This attribute is set to TRUE when the GTP transceiver is used for PCI Express and set to FALSE for all other protocols. The channel bonding function requires this attribute together with TXCHARDISPMODE and TXCHARDISPVAL to support PIPE encode and FTS lane deskew. It also works together with TXELECIDLE to match a shorter sequence from reusing prior channel bonding information after the GTP transceiver returns from electrical idle.
RX_DATA_WIDTH	Integer	Sets the bit width of the RXDATA port. When 8B/10B encoding is enabled, RX_DATA_WIDTH must be set to 20 or 40. Valid settings are 16, 20, 32, and 40. See Interface Width Configuration, page 175 for more details.
RX_DISPERR_SEQ_MATCH	Boolean	Specifies whether the disparity error status of a decoded byte must match the indicator in the channel bonding and clock correction sequence. TRUE: The disparity error must be matched. FALSE: The disparity error status is ignored.

Using RX Channel Bonding

The user must follow the steps described below to use the receiver's channel bonding feature.

Enabling Channel Bonding

Each GTP transceiver includes a circuit that performs channel bonding by controlling the pointers of the RX elastic buffer. Because channel bonding requires the use of the RX buffer, the RXBUF_EN attribute must be set to TRUE.

Each GTP transceiver has a channel bonding circuit. Configuring a GTP transceiver for channel bonding requires these steps:

1. Set the channel bonding mode for each GTP transceiver.
2. Tie the RXCHBONDMASTER of the master transceiver High.
3. Tie the RXCHBONDSLAVE of the slave transceiver(s) High.
4. Connect the channel bonding port from the master to each slave, either directly or by daisy chaining.
5. Set the channel bonding sequence and detection parameters.

Channel Bonding Mode

The channel bonding mode for each GTP transceiver determines whether channel bonding is active and whether the GTP transceiver is the master or a slave. Each set of channel bonded GTP transceivers must have one master and any number of slaves. To turn on channel bonding for a group of GTP transceivers, one transceiver is set to master. The remaining GTP transceivers in the group are set to slaves.

Connecting Channel Bonding Ports

The channel bonding operation requires connecting the master GTP transceiver RXCHBONDO port to the RXCHBONDI port of all slaves in the group. Only GTP transceivers belonging to the same column can be channel bonded together. A direct connection is required for adjacent GTP transceivers. To directly connect a master to a slave:

1. Connect the RXCHBONDO port of the master to the RXCHBONDI port of the slave.
2. Tie the RXCHBONDMASTER of the master transceiver High.
3. Tie the RXCHBONDSLAVE of each slave transceiver High.

When GTP transceivers are directly connected, meeting the timing constraints becomes difficult as the transceivers get further apart. The solution to this problem is to connect the transceivers in a daisy chain. Daisy chaining is performed using the RXCHBONDLEVEL[2:0] ports to allow additional pipeline stages between the master and the slave. The RXCHBONDO port of each slave is used as a pipeline stage in the RXCHBONDO path from the master. Figure 4-30 and Figure 4-31 show two daisy-chain examples.

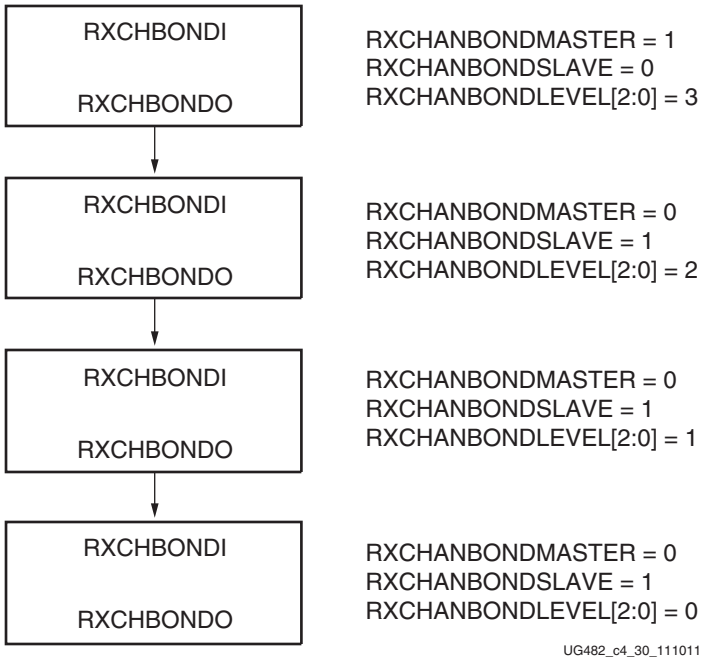


Figure 4-30: **Channel Bonding Daisy Chain Example 1**

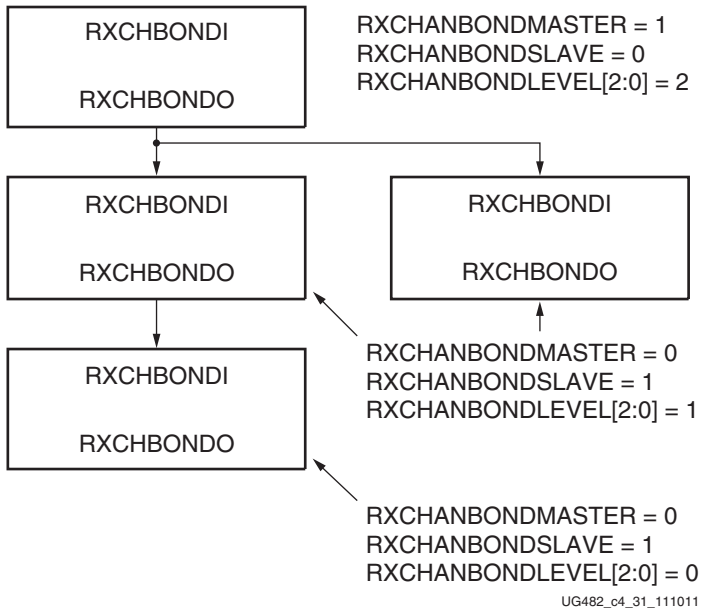


Figure 4-31: **Channel Bonding Daisy Chain Example 2**

To set up a daisy chain, the GTP transceivers are first connected using RXCHBONDO and RXCHBONDI to create a path from the RXCHBONDI port of each slave to the RXCHBONDO port of the master. The following steps describe how to set the RXCHANBONDLEVEL for the GTP transceivers in the chain:

1. Set the RXCHANBONDLEVEL of the master to 7.
2. Set the RXCHANBONDLEVEL of each slave to the RXCHANBONDLEVEL of the GTP transceiver driving the slave's RXCHBONDI port minus 1.
3. Find the slave with the lowest level. Subtract this level from the RXCHANBONDLEVEL of all GTP transceivers so that the lowest slave has level 0 and the master has the minimum level required to service all the slaves.

When the connections between channel bonding ports among GTP transceivers are being decided, the designer must remember that RXCHBONDI and RXCHBONDO belong to the RXUSRCLK clock domain. Meeting the timing constraint of RXUSRCLK becomes increasingly difficult as RXUSRCLK increases in frequency and as directly connected transceivers get further apart.

GTP transceivers in the same half of the device can be bonded with each other. A GTP transceiver located in the top half of the device can be bonded with other GTP transceivers located transceivers in the top half. A GTP transceiver located in the bottom half of the device cannot be bonded with a GTP transceiver located in the top half of the device.

As long as timing constraints are met, there is no limit to the number of GTP transceivers that can be on a particular RXCHANBONDLEVEL.

Setting Channel Bonding Sequences

The channel bonding sequence is programmed in the same way as the clock correction sequence. CHAN_BOND_SEQ_LEN sets the length of the sequence, and CHAN_BOND_SEQ_1_* sets the values of the sequence. If CHAN_BOND_SEQ_2_USE is TRUE, CHAN_BOND_SEQ_2_* sets the values for the alternate second sequence. The number of active bits in each subsequence depends on RX_DATA_WIDTH and CBCC_DATA_SOURCE_SEL (see RX Clock Correction, page 144). When RX_DISPERR_SEQ_MATCH is set to FALSE, CHAN_BOND_SEQ_x_y[9] is not used for matching.

Figure 4-32 shows how the subsequence bits are mapped.

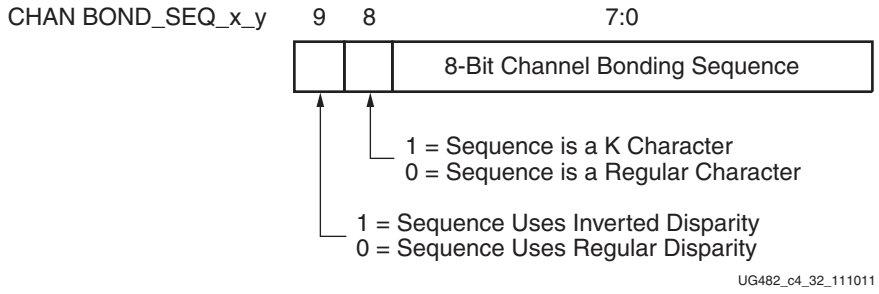


Figure 4-32: **Channel Bonding Sequence Settings**

As with clock correction sequences, channel bonding sequences can have don't care subsequences. CHAN_BOND_SEQ_1_ENABLE and CHAN_BOND_SEQ_2_ENABLE set these bytes. Figure 4-33 shows the mapping of the enable attributes for the channel bonding subsequences.

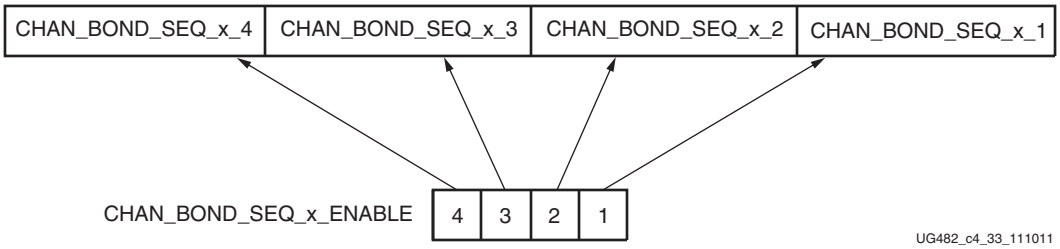


Figure 4-33: **Channel Bonding Sequence Mapping**

Setting the Maximum Skew

When the master receives a channel bonding sequence, it does not trigger channel bonding immediately. Several more bytes must arrive if the slaves have more latency. This wait time effectively becomes the maximum skew that the RX elastic buffer can handle. If the skew is greater than this wait time, the slaves might not receive the sequence by the time the master triggers channel bonding.

Figure 4-34 shows two FIFOs, one for the master and one for the slave. If the slave is behind the master, the master must wait several cycles before triggering channel bonding, otherwise the slow slave does not have the channel bonding sequence in its buffer.

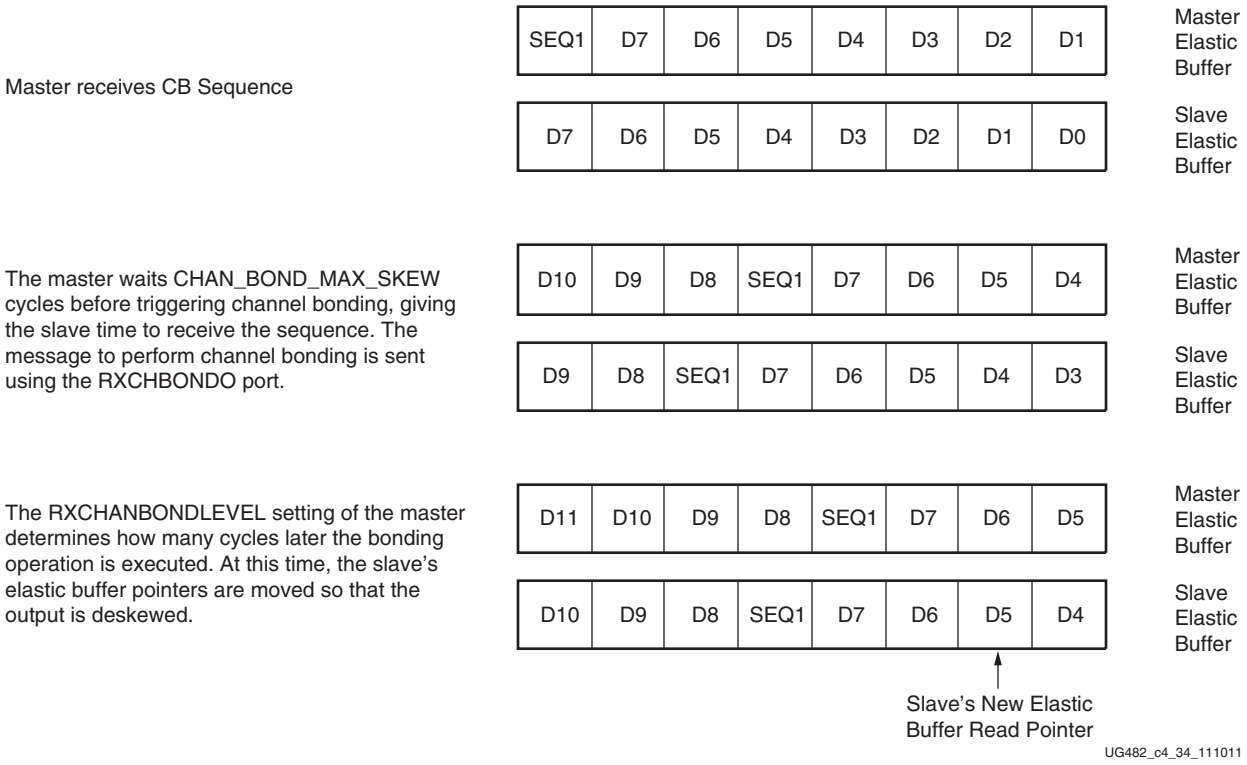


Figure 4-34: **Channel Bonding Example (CHAN_BOND_MAX_SKEW = 2 and Master RXCHANBONDLEVEL[2:0] = 1)**

CHAN_BOND_MAX_SKEW is used to set the maximum skew allowed for channel bonding sequences 1 and 2. The maximum skew range is 1 to 14. This range must always

be less than one-half the minimum distance (in bytes or 10-bit codes) between channel bonding sequences. This minimum distance is determined by the protocol being used.

Precedence between Channel Bonding and Clock Correction

The clock correction (see RX Clock Correction, page 144) and channel bonding circuits both perform operations on the pointers of the RX elastic buffer. Normally, the two circuits work together without conflict, except when clock correction events and channel bonding events occur simultaneously. In this case, one of the two circuits must take precedence. To make clock correction a higher priority than channel bonding, CLK_COR_PRECEDENCE must be set to TRUE. To make channel bonding a higher priority, CLK_COR_PRECEDENCE must be set to FALSE.

RX Gearbox

Functional Description

The RX gearbox provides support for 64B/66B and 64B/67B header and payload separation. The gearbox uses output ports RXDATA[31:0] and RXHEADER[2:0] for the payload and header of the received data. Similar to TX Gearbox, page 54, the RX gearbox operates with the PMA using a single clock. Because of this, occasionally, the output data is invalid. Output ports RXHEADERVALID and RXDATAVALID determine if the appropriate header and data are valid. The RX gearbox supports 2-byte and 4-byte interfaces.

The data out of the RX gearbox is not necessarily aligned. Alignment is done in the FPGA logic. The RXGEARBOXSLIP port can be used to slip the data from the gearbox cycle-by-cycle until correct alignment is reached. It takes a specific number of cycles before the bitslip operation is processed and the output data is stable. Descrambling of the data and block synchronization is done in the FPGA logic.

Ports and Attributes

Table 4-36 defines the RX gearbox ports.

Table 4-36: **RX Gearbox Ports**

Port Name	Dir	Clock Domain	Description
RXDATAVALID[1:0]	Out	RXUSRCLK2	• Bit 0: Status output when Gearbox 64B/66B or 64B/67B is used, which indicates that the data appearing on RXDATA is valid. For example, during 64B/66B encoding, this signal is deasserted every 32 cycles for the 4-byte interface and every 64 cycles for the 2-byte interface. • Bit 1: Reserved.
RXGEARBOXSLIP	In	RXUSRCLK2	When High, this port causes the gearbox contents to slip to the next possible alignment. This port is used to achieve alignment with the FPGA logic. Asserting this port for one RXUSRCLK2 cycle changes the data alignment coming out of the gearbox. RXGEARBOXSLIP must be deasserted for at least one cycle and then reasserted to cause a new realignment of the data. If multiple realignments occur in rapid succession, it is possible to pass the proper alignment point without recognizing the correct alignment point in the FPGA logic.
RXHEADER[2:0]	Out	RXUSRCLK2	Header outputs for 64B/66B (1:0) and 64B/67B (2:0).
RXHEADERVALID	Out	RXUSRCLK2	Indicates that the RXHEADER is valid when using the gearbox.
RXSTARTOFSEQ	Out	RXUSRCLK2	When the gearbox 64B/66B or 64B/67B is enabled, this output indicates when the sequence counter is 0 for the present RXDATA outputs.

Table 4-37 defines the RX gearbox attributes.

Table 4-37: **RX Gearbox Attributes**

Attribute	Type	Description
GEARBOX_MODE	3-bit Binary	This attribute indicates the TX and RX gearbox modes: • Bit 2: Set to 0. Unused. • Bit 1: 0: Use the external sequence counter and apply inputs to TXSEQUENCE in the TX gearbox. 1: Use the internal sequence counter and gate the input header and data with the TXGEARBOXREADY output in the TX gearbox. • Bit 0: 0: 64B/67B gearbox mode for Interlaken 1: 64B/66B gearbox
RXGEARBOX_EN	Boolean	When TRUE, this attribute enables the RX gearbox.

Enabling the RX Gearbox

To enable the RX gearbox for the GTP transceiver, set the attribute RXGEARBOX_EN to TRUE. The GEARBOX_MODE attribute controls the GTP transceiver's TX and RX gearbox use modes.

RX Gearbox Operating Modes

The RX gearbox operates the same in either external sequence counter mode or internal sequence counter mode. The RX gearbox supports 2-byte and 4-byte logic interfaces to the FPGA logic.

As shown in Figure 4-35, either mode uses the RXDATA, RXHEADER, RXDATAOUTVALID, and RXHEADEROUTVALID outputs in addition to the RXGEARBOXSLIP input.

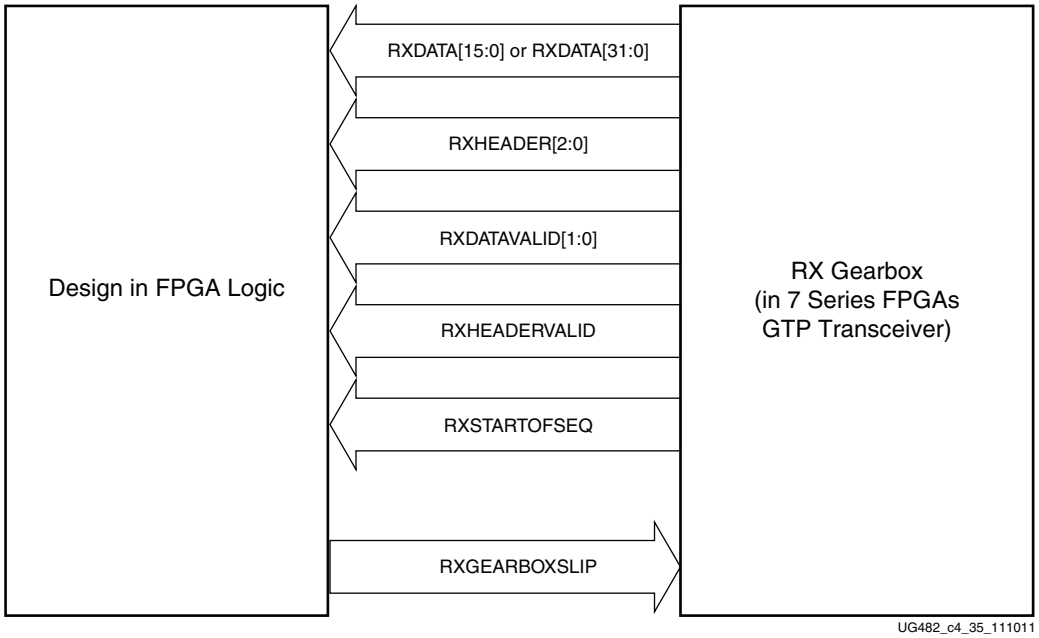


Figure 4-35: **Gearbox in Either Internal or External Sequence Mode**

Figure 4-36 shows an example of five cycles of data entering and exiting the RX gearbox for 64B/66B encoding when using a 2-byte logic interface (RX_DATA_WIDTH = 16 (2-byte)).

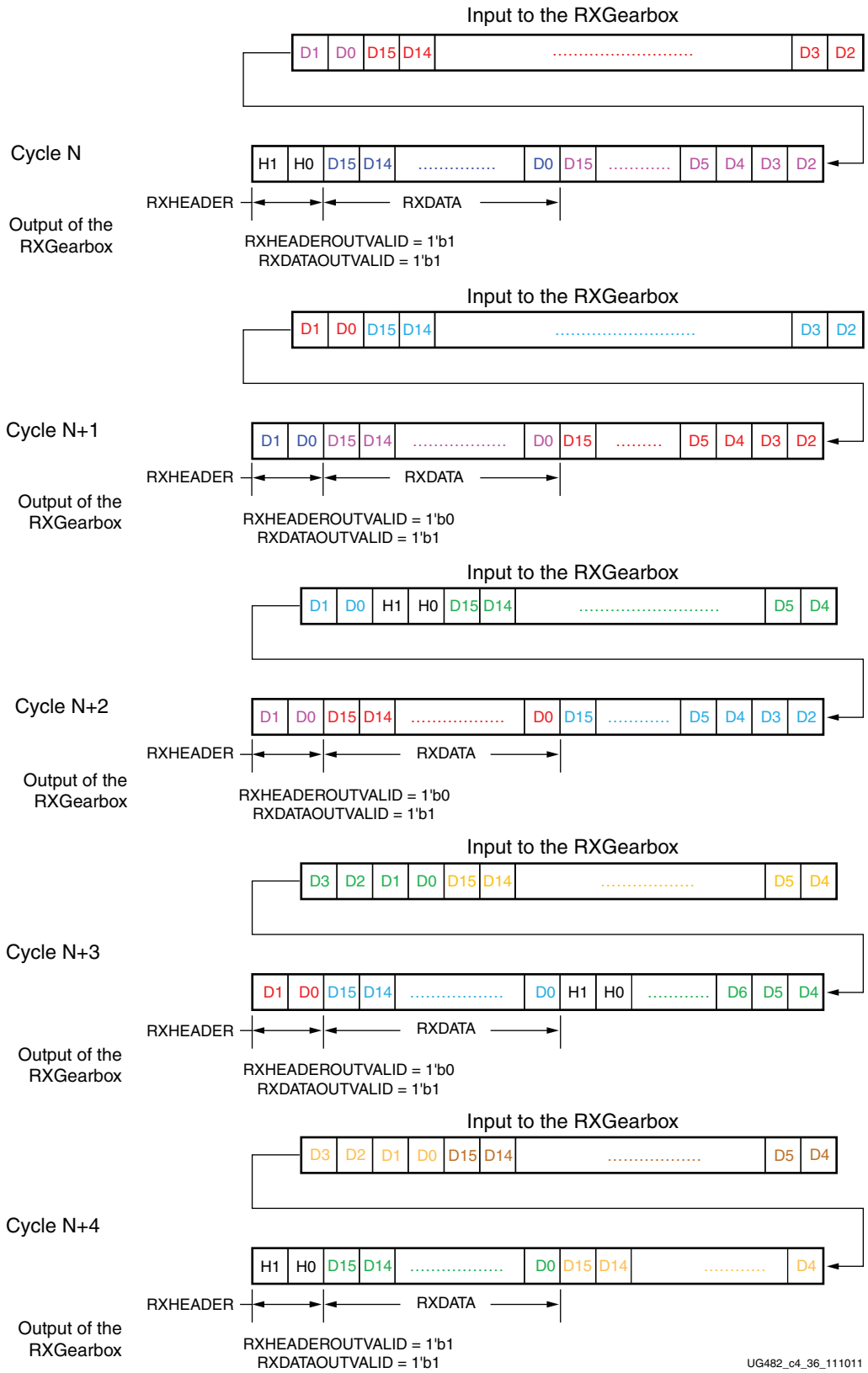


Figure 4-36: **RX Gearbox Operation**

Note relevant to Figure 4-36:

1. As per IEEE Std 802.3ae-2002 nomenclature, H1 corresponds to RxB<0>, H0 to RxB<1>, etc.

The RX gearbox internally manages all sequencing, which differs from the TX gearbox option of either internal or external sequencing. Depending on whether a 2-byte or 4-byte interface is used, RXDATAOUTVALID and RXHEADEROUTVALID assert and deassert for different periods of length. The RX gearbox encounters similar data and header pauses found in the TX gearbox. Figure 4-37 shows such a pause in addition to RXHEADERVALID and RXDATAVALID being deasserted for one cycle. Figure 4-38 shows the operation for 64B/67B encoding when RX_DATA_WIDTH = 16 (2-byte).

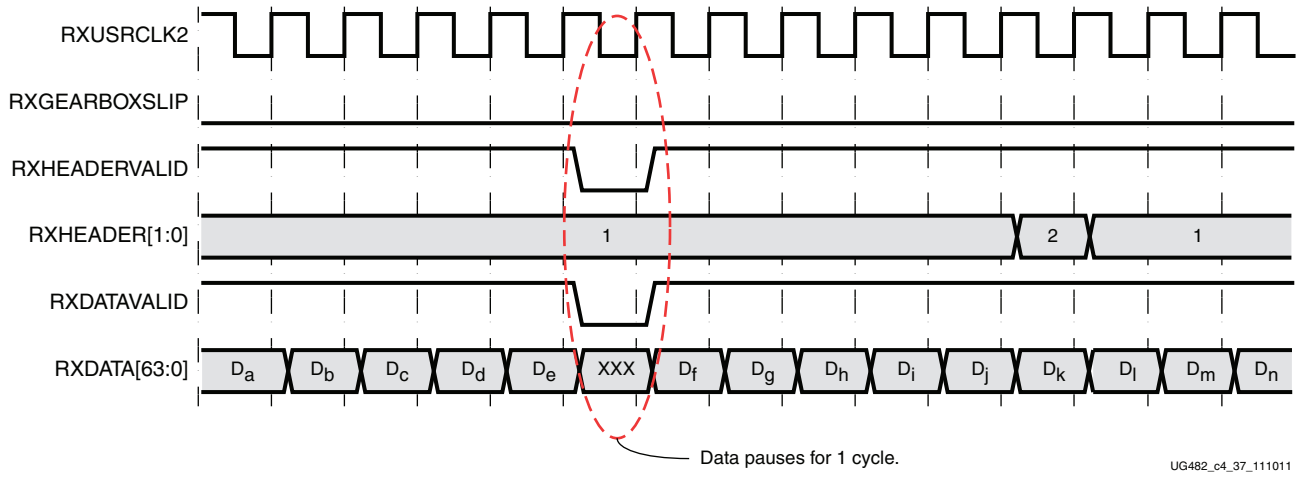


Figure 4-37: **RX Gearbox When Using 64B/66B Encoding and RX_DATA_WIDTH = 32 (4-Byte)**

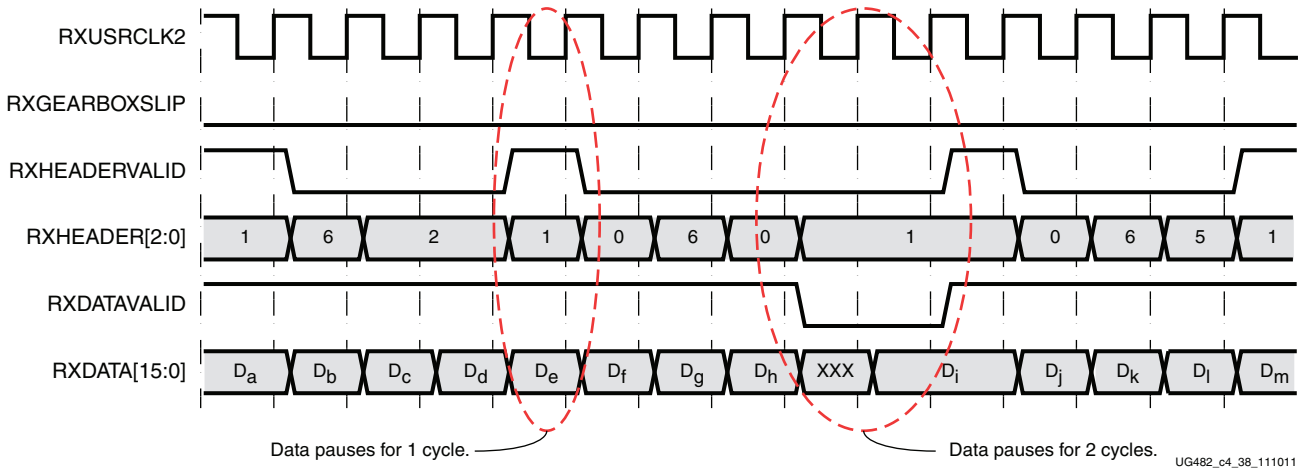


Figure 4-38: **RX Gearbox When Using 64B/67B Encoding and RX_DATA_WIDTH = 16 (2-Byte)**

RX Gearbox Block Synchronization

The 64B/66B and 64B/67B protocols depend on block synchronization to determine their block boundaries. Block synchronization is required because all incoming data is unaligned before block lock is achieved. The goal is to search for the valid synchronization header by changing the data alignment. The RXGEARBOXSLIP input port is used to

change the gearbox data alignment so that all possible alignments can be checked. The RXGEARBOXSLIP signal feeds back from the block synchronization state machine to the RX gearbox and tells it to slip the data alignment. This process of slipping and testing the synchronization header repeats until block lock is achieved. When using the RX gearbox, a block synchronization state machine is required in the FPGA logic. Figure 4-39 shows the operation of a block synchronization state machine. The 7 Series FPGAs Transceivers Wizard has example code for this type of module.

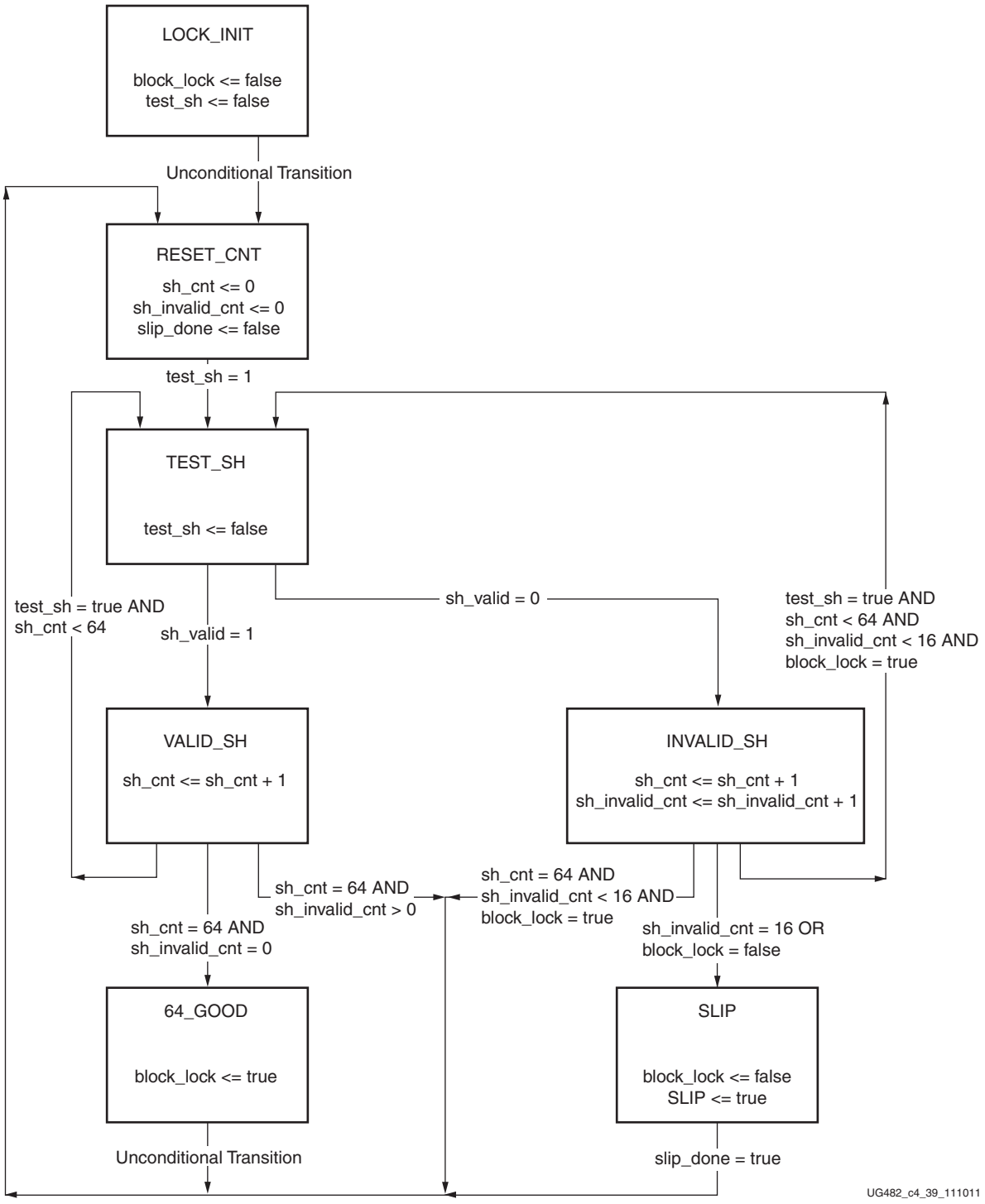


Figure 4-39: **Block Synchronization State Machine**

The state machine works by keeping track of valid and invalid synchronization headers. Upon reset, block lock is deasserted, and the state is LOCK_INIT. The next state is RESET_CNT where all counters are zeroed out. The synchronization header is analyzed in the TEST_SH state. If the header is valid, sh_cnt is incremented in the VALID_SH state, otherwise sh_count and sh_invalid_count are incremented in the INVALID_SH state.

For the block synchronization state machine shown in Figure 4-39, sh_cnt_max and sh_invalid_cnt_max are both constants that are set to 64 and 16, respectively. From the VALID_SH state, if sh_cnt is less than the value sh_cnt_max and test_sh is High, the next state is TEST_SH. If sh_cnt is equal to sh_cnt_max and sh_invalid_cnt equals 0, the next state is GOOD_64 and from there block_lock is asserted. Then the process repeats again and the counters are cleared to zeros. To achieve block lock, the state machine must receive sh_cnt_max number of valid synchronization headers in a row without getting an invalid synchronization header. However, when block lock is achieved sh_invalid_cnt_max – 1, the number of invalid synchronization headers can be received within sh_cnt_max number of valid synchronization headers. Thus, once locked, it is harder to break lock.

Figure 4-40 shows a waveform of the block synchronization state machine asserting RXGEARBOXSLIP numerous times because of invalid synchronization headers before achieving data alignment. After the RXGEARBOXSLIP is issued, the state machine waits 32 RXUSRCLK2 cycles before checking for valid synchronization headers.

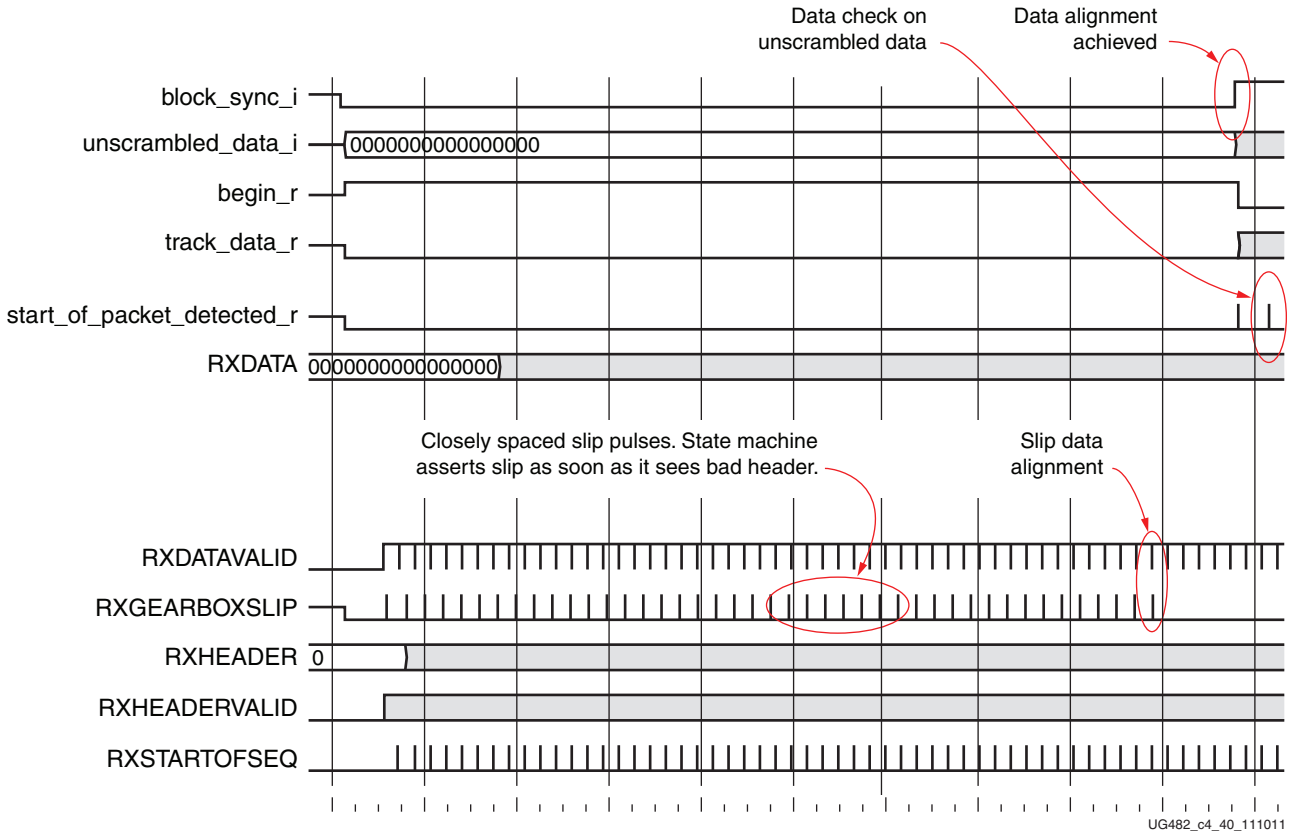


Figure 4-40: **RX Gearbox with Block Synchronization**

FPGA RX Interface

Functional Description

The FPGA RX interface is the FPGA's gateway to the RX datapath of the GTP transceiver. Applications receive data through the GTP transceiver by reading data from the RXDATA port on the positive edge of RXUSRCLK2. The width of the port can be configured to be two or four bytes wide. The actual width of the port depends on the RX_DATA_WIDTH

attribute and RX8B10BEN port setting. Port widths can be 16, 20, 32, and 40 bits. The rate of the parallel clock (RXUSRCLK2) at the interface is determined by the RX line rate, the width of the RXDATA port, and whether or not 8B/10B decoding is enabled. In some operating modes, a second parallel clock (RXUSRCLK) must be provided for the internal PCS logic in the transmitter. This section shows how to drive the parallel clocks and explains the constraints on those clocks for correct operation.

Interface Width Configuration

The 7 series GTP transceiver contains a 2-byte internal datapath. The FPGA interface width is configurable by setting the RX_DATA_WIDTH attribute. When the 8B/10B decoder is enabled, RX_DATA_WIDTH must be configured to 20 bits or 40 bits, and in this case, the FPGA RX interface only uses the RXDATA port. For example, RXDATA[15:0] is used when the FPGA interface width is 16. When the 8B/10B decoder is bypassed, RX_DATA_WIDTH can be configured to any of the available widths: 16, 20, 32, or 40 bits.

Table 4-38 shows how the interface width for the RX datapath is selected. 8B/10B decoding is described in more detail in RX 8B/10B Decoder, page 129.

Table 4-38: **FPGA RX Interface Datapath Configuration**

RX8B10BEN	RX_DATA_WIDTH	FPGA Interface Width	Internal Data Width
1	20	16	20
	40	32	20
0	16	16	16
	20	20	20
	32	32	16
	40	40	20

When the 8B/10B decoder is bypassed and RX_DATA_WIDTH is 20 or 40, the RXDISPERR and RXCHARISK ports are used to extend the RXDATA port from 16 to 20 bits, or 32 to 40 bits. Table 4-39 shows the data received when the 8B/10B decoder is disabled. When the RX gearbox is used, refer to RX Gearbox, page 166 for data transmission order.

Table 4-39: **RX Data Received When the 8B/10B Decoder is Bypassed**

	< < < Data Reception is Right to Left (LSB to MSB) < < <																																	
	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data Received	RXDISPERR[3]	RXCHARISK[3]	RXDATA[31:24]								RXDISPERR[2]	RXCHARISK[2]	RXDATA[23:16]								RXDISPERR[1]	RXCHARISK[1]	RXDATA[15:8]								RXDISPERR[0]	RXCHARISK[0]	RXDATA[7:0]					

RXUSRCLK and RXUSRCLK2 Generation

The FPGA RX interface includes two parallel clocks: RXUSRCLK and RXUSRCLK2. RXUSRCLK is the internal clock for the PCS logic in the GTP transceiver transmitter. The required rate for RXUSRCLK depends on the internal datapath width of the GTPE2_CHANNEL primitive and the RX line rate of the GTP transceiver transmitter. Equation 4-1 shows how to calculate the required rate for RXUSRCLK.

$$RXUSRCLK\ Rate = \frac{Line\ Rate}{Internal\ Datapath\ Width} \qquad \text{Equation 4-1}$$

RXUSRCLK2 is the main synchronization clock for all signals into the RX side of the GTP transceiver. Most signals into the RX side of the GTP transceiver are sampled on the positive edge of RXUSRCLK2. RXUSRCLK2 and RXUSRCLK have a fixed-rate relationship based on the RX_DATA_WIDTH setting. Table 4-40 shows the relationship between RXUSRCLK2 and RXUSRCLK per RX_DATA_WIDTH value.

Table 4-40: **RXUSRCLK2 Frequency Relationship to RXUSRCLK**

FPGA Interface Width	RX_DATA_WIDTH	RXUSRCLK2 Frequency
2-Byte	16, 20	$F_{RXUSRCLK2} = F_{RXUSRCLK}$
4-Byte	32, 40	$F_{RXUSRCLK2} = F_{RXUSRCLK} / 2$

These rules about the relationships between clocks must be observed for RXUSRCLK and RXUSRCLK2:

- RXUSRCLK and RXUSRCLK2 must be positive-edge aligned, with as little skew as possible between them. As a result, low-skew clock resources (BUFGs and BUFRs) should be used to drive RXUSRCLK and RXUSRCLK2.
- If the channel is configured so the same oscillator drives the reference clock for the transmitter and the receiver, TXOUTCLK can be used to drive RXUSRCLK and RXUSRCLK2 in the same way that they are used to drive TXUSRCLK and TXUSRCLK2. When clock correction is turned off or the RX buffer is bypassed, RX phase alignment must be used to align the serial clock and the parallel clocks.
- If separate oscillators are driving the reference clocks for the transmitter and receiver on the channel, and clock correction is not used, RXUSRCLK and RXUSRCLK2 must be driven by RXOUTCLK (RXOUTCLKSEL = `3'b010` for RXOUTCLKPMA), and the phase-alignment circuit must be used.
- If clock correction is used, RXUSRCLK and RXUSRCLK2 can be sourced by RXOUTCLK or TXOUTCLK.

Ports and Attributes

Table 4-41 defines the FPGA RX interface ports.

Table 4-41: **FPGA RX Interface Ports**

Port	Dir	Clock Domain	Description
RXDISPERR[3:0]	Out	RXUSRCLK2	When 8B/10B decoding is disabled, RXDISPERR is used to extend the data bus for 20-bit and 40-bit RX interfaces.
RXCHARISK[3:0]	Out	RXUSRCLK2	When 8B/10B decoding is disabled, RXCHARISK is used to extend the data bus for 20-bit and 40-bit RX interfaces.

Table 4-41: **FPGA RX Interface Ports *(Cont'd)***

Port	Dir	Clock Domain	Description
RXDATA[31:0]	Out	RXUSRCLK2	The bus for transmitting data. The width of this port depends on RX_DATA_WIDTH: RX_DATA_WIDTH = 16, 20: RXDATA[15:0] = 16 bits wide RX_DATA_WIDTH = 32, 40: RXDATA[31:0] = 32 bits wide When a 20-bit or 40-bit bus is required, the RXCHARISK and RXDISPERR ports from the 8B/10B encoder are concatenated with the RXDATA port. See Table 4-39, page 175.
RXUSRCLK	In	Clock	This port is used to provide a clock for the internal RX PCS datapath.
RXUSRCLK2	In	Clock	This port is used to synchronize the FPGA logic with the RX interface. This clock must be positive-edge aligned to RXUSRCLK when RXUSRCLK is provided by the user.

Table 4-42 defines the FPGA RX interface attributes.

Table 4-42: **FPGA RX Interface Attributes**

Attribute	Type	Description
RX_DATA_WIDTH	Integer	Sets the bit width of the RXDATA port. When 8B/10B encoding is enabled, RX_DATA_WIDTH must be set to 20 or 40. Valid settings are 16, 20, 32, or 40. See Interface Width Configuration, page 175 for more details.

XILINX®

Appendix A

Placement Information by Package

This appendix provides the Quad position information for available device and package combinations along with the pad numbers for the external signals associated with each serial transceiver channel and the associated primitive.

FGG676 Package Placement Diagram

Figure A-1 and Figure A-2 show the placement diagram for the FGG676 package.

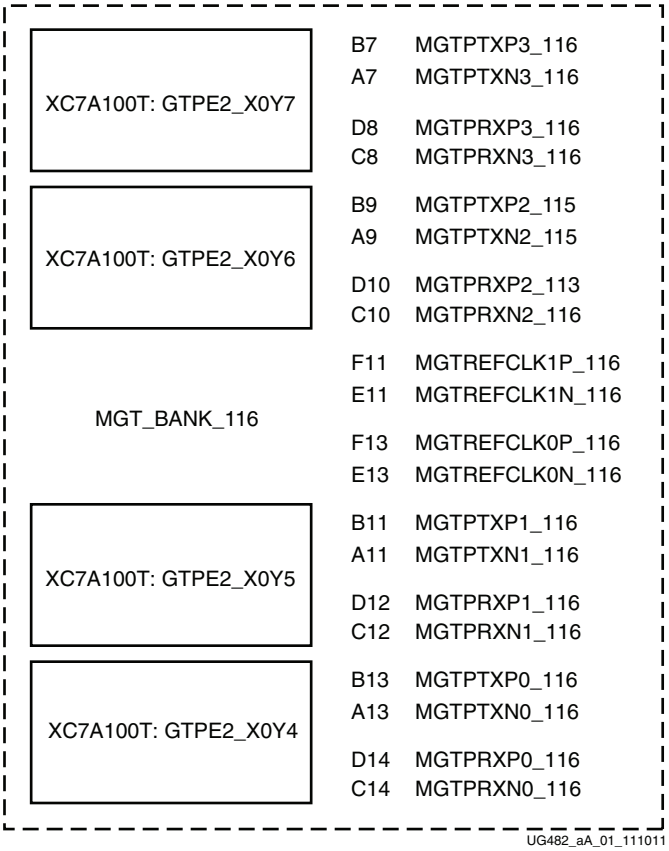


Figure A-1: **Placement Diagram for the FGG676 Package (1 of 2)**

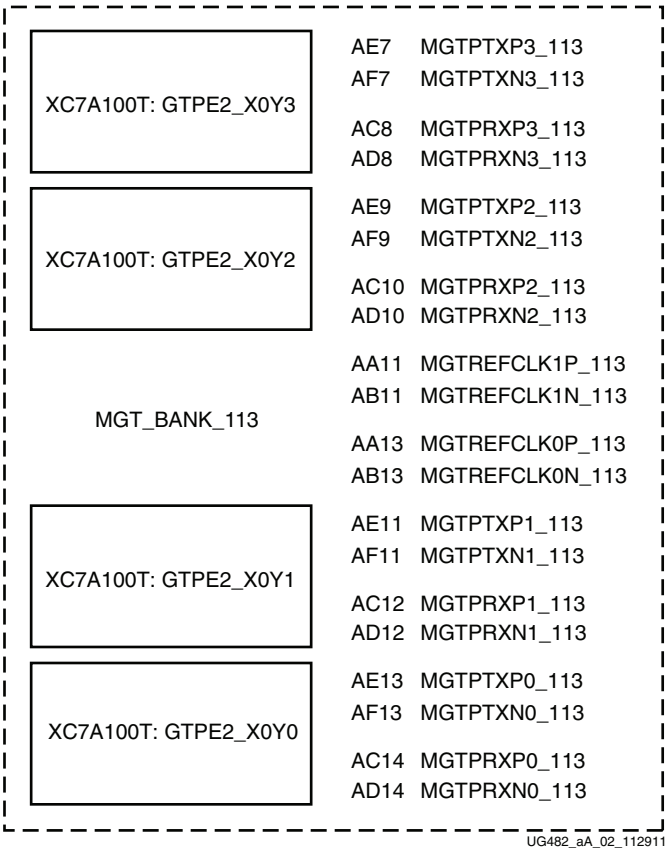


Figure A-2: **Placement Diagram for the FGG676 Package (2 of 2)**

XILINX®

Appendix B

Placement Information by Device

Table B-1 defines the Artix™-7 FPGA device-package combinations and the available GTP transceiver banks. For transceiver location, refer to Appendix A, Placement Information by Package.

Table B-1: **Artix-7 FPGA Device-Package Combinations and GTP Transceiver Banks**

Package	CSG225	FGG484	FGG676	FBG484	FBG676	FFG1156
XC7A30T	MGT_BANK_116	MGT_BANK_116				
XC7A50T	MGT_BANK_116	MGT_BANK_116				
XC7A100T		MGT_BANK_116	MGT_BANK_113, MGT_BANK_116			
XC7A200T				MGT_BANK_216	MGT_BANK_212, MGT_BANK_216	MGT_BANK_112, MGT_BANK_116, MGT_BANK_212, MGT_BANK_216
XC7A350T				MGT_BANK_217	MGT_BANK_212, MGT_BANK_217	MGT_BANK_112, MGT_BANK_117, MGT_BANK_212, MGT_BANK_217

XILINX®

Appendix C

8B/10B Valid Characters

8B/10B encoding includes a set of Data characters and K characters. Eight-bit values are coded into 10-bit values, keeping the serial line DC balanced. K characters are special Data characters designated with a CHARISK. K characters are used for specific informative designations. Table C-1 shows the valid Data characters. Table C-2, page 193 shows the valid K characters.

Table C-1: **Valid Data Characters**

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D0.0	000 00000	100111 0100	011000 1011
D1.0	000 00001	011101 0100	100010 1011
D2.0	000 00010	101101 0100	010010 1011
D3.0	000 00011	110001 1011	110001 0100
D4.0	000 00100	110101 0100	001010 1011
D5.0	000 00101	101001 1011	101001 0100
D6.0	000 00110	011001 1011	011001 0100
D7.0	000 00111	111000 1011	000111 0100
D8.0	000 01000	111001 0100	000110 1011
D9.0	000 01001	100101 1011	100101 0100
D10.0	000 01010	010101 1011	010101 0100
D11.0	000 01011	110100 1011	110100 0100
D12.0	000 01100	001101 1011	001101 0100
D13.0	000 01101	101100 1011	101100 0100
D14.0	000 01110	011100 1011	011100 0100
D15.0	000 01111	010111 0100	101000 1011
D16.0	000 10000	011011 0100	100100 1011
D17.0	000 10001	100011 1011	100011 0100
D18.0	000 10010	010011 1011	010011 0100
D19.0	000 10011	110010 1011	110010 0100
D20.0	000 10100	001011 1011	001011 0100

Table C-1: **Valid Data Characters *(Cont'd)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D21.0	000 10101	101010 1011	101010 0100
D22.0	000 10110	011010 1011	011010 0100
D23.0	000 10111	111010 0100	000101 1011
D24.0	000 11000	110011 0100	001100 1011
D25.0	000 11001	100110 1011	100110 0100
D26.0	000 11010	010110 1011	010110 0100
D27.0	000 11011	110110 0100	001001 1011
D28.0	000 11100	001110 1011	001110 0100
D29.0	000 11101	101110 0100	010001 1011
D30.0	000 11110	011110 0100	100001 1011
D31.0	000 11111	101011 0100	010100 1011
D0.1	001 00000	100111 1001	011000 1001
D1.1	001 00001	011101 1001	100010 1001
D2.1	001 00010	101101 1001	010010 1001
D3.1	001 00011	110001 1001	110001 1001
D4.1	001 00100	110101 1001	001010 1001
D5.1	001 00101	101001 1001	101001 1001
D6.1	001 00110	011001 1001	011001 1001
D7.1	001 00111	111000 1001	000111 1001
D8.1	001 01000	111001 1001	000110 1001
D9.1	001 01001	100101 1001	100101 1001
D10.1	001 01010	010101 1001	010101 1001
D11.1	001 01011	110100 1001	110100 1001
D12.1	001 01100	001101 1001	001101 1001
D13.1	001 01101	101100 1001	101100 1001
D14.1	001 01110	011100 1001	011100 1001
D15.1	001 01111	010111 1001	101000 1001
D16.1	001 10000	011011 1001	100100 1001
D17.1	001 10001	100011 1001	100011 1001
D18.1	001 10010	010011 1001	010011 1001
D19.1	001 10011	110010 1001	110010 1001
D20.1	001 10100	001011 1001	001011 1001

Table C-1: **Valid Data Characters *(Cont'd)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D21.1	001 10101	101010 1001	101010 1001
D22.1	001 10110	011010 1001	011010 1001
D23.1	001 10111	111010 1001	000101 1001
D24.1	001 11000	110011 1001	001100 1001
D25.1	001 11001	100110 1001	100110 1001
D26.1	001 11010	010110 1001	010110 1001
D27.1	001 11011	110110 1001	001001 1001
D28.1	001 11100	001110 1001	001110 1001
D29.1	001 11101	101110 1001	010001 1001
D30.1	001 11110	011110 1001	100001 1001
D31.1	001 11111	101011 1001	010100 1001
D0.2	010 00000	100111 0101	011000 0101
D1.2	010 00001	011101 0101	100010 0101
D2.2	010 00010	101101 0101	010010 0101
D3.2	010 00011	110001 0101	110001 0101
D4.2	010 00100	110101 0101	001010 0101
D5.2	010 00101	101001 0101	101001 0101
D6.2	010 00110	011001 0101	011001 0101
D7.2	010 00111	111000 0101	000111 0101
D8.2	010 01000	111001 0101	000110 0101
D9.2	010 01001	100101 0101	100101 0101
D10.2	010 01010	010101 0101	010101 0101
D11.2	010 01011	110100 0101	110100 0101
D12.2	010 01100	001101 0101	001101 0101
D13.2	010 01101	101100 0101	101100 0101
D14.2	010 01110	011100 0101	011100 0101
D15.2	010 01111	010111 0101	101000 0101
D16.2	010 10000	011011 0101	100100 0101
D17.2	010 10001	100011 0101	100011 0101
D18.2	010 10010	010011 0101	010011 0101
D19.2	010 10011	110010 0101	110010 0101
D20.2	010 10100	001011 0101	001011 0101

Table C-1: **Valid Data Characters *(Cont'd)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D21.2	010 10101	101010 0101	101010 0101
D22.2	010 10110	011010 0101	011010 0101
D23.2	010 10111	111010 0101	000101 0101
D24.2	010 11000	110011 0101	001100 0101
D25.2	010 11001	100110 0101	100110 0101
D26.2	010 11010	010110 0101	010110 0101
D27.2	010 11011	110110 0101	001001 0101
D28.2	010 11100	001110 0101	001110 0101
D29.2	010 11101	101110 0101	010001 0101
D30.2	010 11110	011110 0101	100001 0101
D31.2	010 11111	101011 0101	010100 0101
D0.3	011 00000	100111 0011	011000 1100
D1.3	011 00001	011101 0011	100010 1100
D2.3	011 00010	101101 0011	010010 1100
D3.3	011 00011	110001 1100	110001 0011
D4.3	011 00100	110101 0011	001010 1100
D5.3	011 00101	101001 1100	101001 0011
D6.3	011 00110	011001 1100	011001 0011
D7.3	011 00111	111000 1100	000111 0011
D8.3	011 01000	111001 0011	000110 1100
D9.3	011 01001	100101 1100	100101 0011
D10.3	011 01010	010101 1100	010101 0011
D11.3	011 01011	110100 1100	110100 0011
D12.3	011 01100	001101 1100	001101 0011
D13.3	011 01101	101100 1100	101100 0011
D14.3	011 01110	011100 1100	011100 0011
D15.3	011 01111	010111 0011	101000 1100
D16.3	011 10000	011011 0011	100100 1100
D17.3	011 10001	100011 1100	100011 0011
D18.3	011 10010	010011 1100	010011 0011
D19.3	011 10011	110010 1100	110010 0011
D20.3	011 10100	001011 1100	001011 0011

Table C-1: **Valid Data Characters *(Cont'd)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D21.3	011 10101	101010 1100	101010 0011
D22.3	011 10110	011010 1100	011010 0011
D23.3	011 10111	111010 0011	000101 1100
D24.3	011 11000	110011 0011	001100 1100
D25.3	011 11001	100110 1100	100110 0011
D26.3	011 11010	010110 1100	010110 0011
D27.3	011 11011	110110 0011	001001 1100
D28.3	011 11100	001110 1100	001110 0011
D29.3	011 11101	101110 0011	010001 1100
D30.3	011 11110	011110 0011	100001 1100
D31.3	011 11111	101011 0011	010100 1100
D0.4	100 00000	100111 0010	011000 1101
D1.4	100 00001	011101 0010	100010 1101
D2.4	100 00010	101101 0010	010010 1101
D3.4	100 00011	110001 1101	110001 0010
D4.4	100 00100	110101 0010	001010 1101
D5.4	100 00101	101001 1101	101001 0010
D6.4	100 00110	011001 1101	011001 0010
D7.4	100 00111	111000 1101	000111 0010
D8.4	100 01000	111001 0010	000110 1101
D9.4	100 01001	100101 1101	100101 0010
D10.4	100 01010	010101 1101	010101 0010
D11.4	100 01011	110100 1101	110100 0010
D12.4	100 01100	001101 1101	001101 0010
D13.4	100 01101	101100 1101	101100 0010
D14.4	100 01110	011100 1101	011100 0010
D15.4	100 01111	010111 0010	101000 1101
D16.4	100 10000	011011 0010	100100 1101
D17.4	100 10001	100011 1101	100011 0010
D18.4	100 10010	010011 1101	010011 0010
D19.4	100 10011	110010 1101	110010 0010
D20.4	100 10100	001011 1101	001011 0010

Table C-1: **Valid Data Characters *(Cont'd)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D21.4	100 10101	101010 1101	101010 0010
D22.4	100 10110	011010 1101	011010 0010
D23.4	100 10111	111010 0010	000101 1101
D24.4	100 11000	110011 0010	001100 1101
D25.4	100 11001	100110 1101	100110 0010
D26.4	100 11010	010110 1101	010110 0010
D27.4	100 11011	110110 0010	001001 1101
D28.4	100 11100	001110 1101	001110 0010
D29.4	100 11101	101110 0010	010001 1101
D30.4	100 11110	011110 0010	100001 1101
D31.4	100 11111	101011 0010	010100 1101
D0.5	101 00000	100111 1010	011000 1010
D1.5	101 00001	011101 1010	100010 1010
D2.5	101 00010	101101 1010	010010 1010
D3.5	101 00011	110001 1010	110001 1010
D4.5	101 00100	110101 1010	001010 1010
D5.5	101 00101	101001 1010	101001 1010
D6.5	101 00110	011001 1010	011001 1010
D7.5	101 00111	111000 1010	000111 1010
D8.5	101 01000	111001 1010	000110 1010
D9.5	101 01001	100101 1010	100101 1010
D10.5	101 01010	010101 1010	010101 1010
D11.5	101 01011	110100 1010	110100 1010
D12.5	101 01100	001101 1010	001101 1010
D13.5	101 01101	101100 1010	101100 1010
D14.5	101 01110	011100 1010	011100 1010
D15.5	101 01111	010111 1010	101000 1010
D16.5	101 10000	011011 1010	100100 1010
D17.5	101 10001	100011 1010	100011 1010
D18.5	101 10010	010011 1010	010011 1010
D19.5	101 10011	110010 1010	110010 1010
D20.5	101 10100	001011 1010	001011 1010

Table C-1: **Valid Data Characters *(Cont'd)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D21.5	101 10101	101010 1010	101010 1010
D22.5	101 10110	011010 1010	011010 1010
D23.5	101 10111	111010 1010	000101 1010
D24.5	101 11000	110011 1010	001100 1010
D25.5	101 11001	100110 1010	100110 1010
D26.5	101 11010	010110 1010	010110 1010
D27.5	101 11011	110110 1010	001001 1010
D28.5	101 11100	001110 1010	001110 1010
D29.5	101 11101	101110 1010	010001 1010
D30.5	101 11110	011110 1010	100001 1010
D31.5	101 11111	101011 1010	010100 1010
D0.6	110 00000	100111 0110	011000 0110
D1.6	110 00001	011101 0110	100010 0110
D2.6	110 00010	101101 0110	010010 0110
D3.6	110 00011	110001 0110	110001 0110
D4.6	110 00100	110101 0110	001010 0110
D5.6	110 00101	101001 0110	101001 0110
D6.6	110 00110	011001 0110	011001 0110
D7.6	110 00111	111000 0110	000111 0110
D8.6	110 01000	111001 0110	000110 0110
D9.6	110 01001	100101 0110	100101 0110
D10.6	110 01010	010101 0110	010101 0110
D11.6	110 01011	110100 0110	110100 0110
D12.6	110 01100	001101 0110	001101 0110
D13.6	110 01101	101100 0110	101100 0110
D14.6	110 01110	011100 0110	011100 0110
D15.6	110 01111	010111 0110	101000 0110
D16.6	110 10000	011011 0110	100100 0110
D17.6	110 10001	100011 0110	100011 0110
D18.6	110 10010	010011 0110	010011 0110
D19.6	110 10011	110010 0110	110010 0110
D20.6	110 10100	001011 0110	001011 0110

Table C-1: **Valid Data Characters *(Cont'd)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D21.6	110 10101	101010 0110	101010 0110
D22.6	110 10110	011010 0110	011010 0110
D23.6	110 10111	111010 0110	000101 0110
D24.6	110 11000	110011 0110	001100 0110
D25.6	110 11001	100110 0110	100110 0110
D26.6	110 11010	010110 0110	010110 0110
D27.6	110 11011	110110 0110	001001 0110
D28.6	110 11100	001110 0110	001110 0110
D29.6	110 11101	101110 0110	010001 0110
D30.6	110 11110	011110 0110	100001 0110
D31.6	110 11111	101011 0110	010100 0110
D0.7	111 00000	100111 0001	011000 1110
D1.7	111 00001	011101 0001	100010 1110
D2.7	111 00010	101101 0001	010010 1110
D3.7	111 00011	110001 1110	110001 0001
D4.7	111 00100	110101 0001	001010 1110
D5.7	111 00101	101001 1110	101001 0001
D6.7	111 00110	011001 1110	011001 0001
D7.7	111 00111	111000 1110	000111 0001
D8.7	111 01000	111001 0001	000110 1110
D9.7	111 01001	100101 1110	100101 0001
D10.7	111 01010	010101 1110	010101 0001
D11.7	111 01011	110100 1110	110100 1000
D12.7	111 01100	001101 1110	001101 0001
D13.7	111 01101	101100 1110	101100 1000
D14.7	111 01110	011100 1110	011100 1000
D15.7	111 01111	010111 0001	101000 1110
D16.7	111 10000	011011 0001	100100 1110
D17.7	111 10001	100011 0111	100011 0001
D18.7	111 10010	010011 0111	010011 0001
D19.7	111 10011	110010 1110	110010 0001
D20.7	111 10100	001011 0111	001011 0001

Table C-1: **Valid Data Characters *(Cont'd)***

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D21.7	111 10101	101010 1110	101010 0001
D22.7	111 10110	011010 1110	011010 0001
D23.7	111 10111	111010 0001	000101 1110
D24.7	111 11000	110011 0001	001100 1110
D25.7	111 11001	100110 1110	100110 0001
D26.7	111 11010	010110 1110	010110 0001
D27.7	111 11011	110110 0001	001001 1110
D28.7	111 11100	001110 1110	001110 0001
D29.7	111 11101	101110 0001	010001 1110
D30.7	111 11110	011110 0001	100001 1110
D31.7	111 11111	101011 0001	010100 1110

Table C-2: **Valid Control K Characters**

Special Code Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
K28.0	000 11100	001111 0100	110000 1011
K28.1	001 11100	001111 1001	110000 0110
K28.2	010 11100	001111 0101	110000 1010
K28.3	011 11100	001111 0011	110000 1100
K28.4	100 11100	001111 0010	110000 1101
K28.5	101 11100	001111 1010	110000 0101
K28.6	110 11100	001111 0110	110000 1001
K28.7(1)	111 11100	001111 1000	110000 0111
K23.7	111 10111	111010 1000	000101 0111
K27.7	111 11011	110110 1000	001001 0111
K29.7	111 11101	101110 1000	010001 0111
K30.7	111 11110	011110 1000	100001 0111

Notes:

1. Used for testing and characterization only.

XILINX®

Appendix D

DRP Address Map of the GTP Transceiver

Table D-1 lists the DRP map of the GTPE2_COMMON primitive sorted by address.

Note: The reserved bits should NOT be modified. Attributes that are not described explicitly are set automatically by the 7 Series FPGAs Transceivers Wizard. These attributes must be left at their defaults, except for use cases that explicitly request different values.

Table D-1: **DRP Map of GTPE2_COMMON Primitive**

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Binary Encoding
04	15:0	R/W	BIAS_CFG	15:0	0–65535	0–65535
05	15:0	R/W	BIAS_CFG	31:16	0–65535	0–65535
06	15:0	R/W	BIAS_CFG	47:32	0–65535	0–65535
07	15:0	R/W	BIAS_CFG	63:48	0–65535	0–65535
09	15:0	R/W	COMMON_CFG	15:0	0–65535	0–65535
0A	15:0	R/W	COMMON_CFG	31:16	0–65535	0–65535
0D	15:0	R/W	RSVD_ATTR0	15:0	0–65535	0–65535
0E	15:0	R/W	RSVD_ATTR1	15:0	0–65535	0–65535
10	14	R/W	PLL0_DMON_CFG	0	0–1	0–1
10	13:8	R/W	PLL0_FBDIV	5:0	1	16
					2	0
					3	1
					4	2
					5	3
					6	5
					8	6
					10	7
					12	13
					16	14
					20	15

Table D-1: **DRP Map of GTPE2_COMMON Primitive *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Binary Encoding
11	8:0	R/W	PLL0_LOCK_CFG	8:0	0–511	0–511
12	4:0	R/W	PLL0_REFCLK_DIV	4:0	1	16
13	15:0	R/W	PLL0_INIT_CFG	15:0	0–65535	0–65535
14	7:0	R/W	PLL0_INIT_CFG	23:16	0–255	0–255
16	14	R/W	PLL1_DMON_CFG	0	0–1	0–1
16	13:8	R/W	PLL1_FBDIV	5:0	1	16
					2	0
					3	1
					4	2
					5	3
					6	5
					8	6
					10	7
					12	13
					16	14
					20	15
17	8:0	R/W	PLL1_LOCK_CFG	8:0	0–511	0–511
18	4:0	R/W	PLL1_REFCLK_DIV	4:0	1	16
					2	0
					3	1
					4	2
					5	3
					6	5
					8	6
					10	7
					12	13
					16	14
					20	15
19	15:0	R/W	PLL1_INIT_CFG	15:0	0–65535	0–65535
1A	7:0	R/W	PLL1_INIT_CFG	23:16	0–255	0–255
1C	15:0	R/W	PLL0_CFG	15:0	0–65535	0–65535
1D	10:0	R/W	PLL0_CFG	26:16	0–2047	0–2047

Table D-1: **DRP Map of GTPE2_COMMON Primitive *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Binary Encoding
1E	15:0	R/W	PLL1_CFG	15:0	0–65535	0–65535
1F	10:0	R/W	PLL1_CFG	26:16	0–2047	0–2047
21	15:8	R/W	QPLL_CLKOUT_CFG	7:0	0–255	0–255

Table D-2 lists the DRP map of the GTPE2_CHANNEL primitive sorted by address.

Note: The reserved bits should NOT be modified. Attributes that are not described explicitly are set automatically by the 7 Series FPGAs Transceivers Wizard. These attributes must be left at their defaults, except for use cases that explicitly request different values.

Table D-2: **DRP Map of GTPE2_CHANNEL Primitive**

DRP Address	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
0000	3	R/W	UCODEER_CLR	0	0–1	0–1
0009	9	R/W	A_RXOSCALRESET	0	0–1	0–1
000C	14:0	R/W	TERM_RCAL_CFG	14:0	0–32767	0–32767
000D	15:13	R/W	TERM_RCAL_OVRD	2:0	0–7	0–7
000D	11:6	R/W	SATA_MIN_WAKE	5:0	1–63	1–63
000D	5:0	R/W	SATA_MIN_INIT	5:0	1–63	1–63
000E	11:6	R/W	SAS_MIN_COM	5:0	1–63	1–63
000E	5:0	R/W	SATA_MIN_BURST	5:0	1–61	1–61
000F	11:9	R/W	SATA_EIDLE_VAL	2:0	0–7	0–7
000F	8:5	R/W	SATA_BURST_SEQ_LEN	3:0	0–15	0–15
000F	4:3	R/W	OUTREFCLK_SEL_INV	1:0	0–3	0–3
000F	2:0	R/W	SATA_BURST_VAL	2:0	0–7	0–7
0010	13	R/W	RXCDR_PH_RESET_ON_EIDLE	0	0–1	0–1
0010	12	R/W	RXCDR_FR_RESET_ON_EIDLE	0	0–1	0–1
0010	11	R/W	RXBUF_RESET_ON_EIDLE	0	FALSE	0
					TRUE	1
0010	10:7	R/W	RXBUF_EIDLE_LO_CNT	3:0	0–15	0–15
0010	6:3	R/W	RXBUF_EIDLE_HI_CNT	3:0	0–15	0–15
0010	2	R/W	RXBUF_ADDR_MODE	0	FULL	0
0010	2	R/W	RXBUF_ADDR_MODE	0	FAST	1
0010	1	R/W	RX_DISPERR_SEQ_MATCH	0	FALSE	0
0010	1	R/W	RX_DISPERR_SEQ_MATCH	0	TRUE	1
0011	15:10	R/W	RXBUF_THRESH_UNDFLW	5:0	0–63	0–63

Table D-2: **DRP Map of GTPE2_CHANNEL Primitive *(Cont'd)***

DRP Address	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
0011	9	R/W	RXBUF_RESET_ON_CB_CHANGE	0	FALSE	0
					TRUE	1
0011	8	R/W	RXBUF_RESET_ON_RATE_CHANGE	0	FALSE	0
					TRUE	1
0011	7	R/W	RXBUF_RESET_ON_COMMAALIGN	0	FALSE	0
					TRUE	1
0011	6	R/W	RXBUF_THRESH_OVRD	0	FALSE	0
					TRUE	1
0011	5:0	R/W	RXBUF_THRESH_OVFLW	5:0	0–63	0–63
0012	15	R/W	DEC_PCOMMA_DETECT	0	FALSE	0
					TRUE	1
0012	12:7	R/W	RX_BUFFER_CFG	5:0	0–63	0–63
0012	5:0	R/W	RX_DDI_SEL	5:0	0–63	0–63
0013	15	R/W	DEC_MCOMMA_DETECT	0	FALSE	0
					TRUE	1
0013	14	R/W	DEC_VALID_COMMA_ONLY	0	FALSE	0
					TRUE	1
0013	13	R/W	SHOW_REALIGN_COMMA	0	FALSE	0
					TRUE	1
0013	12	R/W	ALIGN_COMMA_DOUBLE	0	FALSE	0
					TRUE	1
0013	11:10	R/W	ALIGN_COMMA_WORD	1:0	1	1
0013	11:10	R/W	ALIGN_COMMA_WORD	1:0	2	2
0013	9:0	R/W	ALIGN_COMMA_ENABLE	9:0	0–1023	0–1023
0014	12:6	R/W	SAS_MAX_COM	6:0	1–127	1–127
0014	5:0	R/W	SATA_MAX_BURST	5:0	1–63	1–63
0015	11:6	R/W	SATA_MAX_WAKE	5:0	1–63	1–63
0015	5:0	R/W	SATA_MAX_INIT	5:0	1–63	1–63
0016	15:0	R/W	DMONITOR_CFG	15:0	0–65535	0–65535

Table D-2: **DRP Map of GTPE2_CHANNEL Primitive *(Cont'd)***

DRP Address	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
0017	15:11	R/W	RX_CLK25_DIV	4:0	1	0
					2	1
					3	2
					4	3
					5	4
					6	5
					7	6
					8	7
					9	8
					10	9
					11	10
					12	11
					13	12
					14	13
					15	14
					16	15
					17	16
					18	17
					19	18
					20	19
					21	20
					22	21
					23	22
					24	23
					25	24
					26	25
					27	26
					28	27
					29	28
					30	29
					31	30
					32	31

Table D-2: **DRP Map of GTPE2_CHANNEL Primitive *(Cont'd)***

DRP Address	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
0017	7:0	R/W	DMONITOR_CFG	23:16	0–255	0–255
0018	12:8	R/W	ES_PRESCALE	4:0	0–31	0–31
0018	7	R/W	ES_EYE_SCAN_EN	0	FALSE	0
					TRUE	1
0018	6	R/W	ES_ERRDET_EN	0	FALSE	0
					TRUE	1
0018	5:0	R/W	ES_CONTROL	5:0	0–63	0–63
0019	11:0	R/W	ES_HORZ_OFFSET	11:0	0–4095	0–4095
001A	15:0	R/W	ES_QUALIFIER	15:0	0–65535	0–65535
001B	15:0	R/W	ES_QUALIFIER	31:16	0–65535	0–65535
001C	15:0	R/W	ES_QUALIFIER	47:32	0–65535	0–65535
001D	15:0	R/W	ES_QUALIFIER	63:48	0–65535	0–65535
001E	15:0	R/W	ES_QUALIFIER	79:64	0–65535	0–65535
001F	15:0	R/W	ES_QUAL_MASK	15:0	0–65535	0–65535
0020	15:0	R/W	ES_QUAL_MASK	31:16	0–65535	0–65535
0021	15:0	R/W	ES_QUAL_MASK	47:32	0–65535	0–65535
0022	15:0	R/W	ES_QUAL_MASK	63:48	0–65535	0–65535
0023	15:0	R/W	ES_QUAL_MASK	79:64	0–65535	0–65535
0024	15:0	R/W	ES_SDATA_MASK	15:0	0–65535	0–65535
0025	15:0	R/W	ES_SDATA_MASK	31:16	0–65535	0–65535
0026	15:0	R/W	ES_SDATA_MASK	47:32	0–65535	0–65535
0027	15:0	R/W	ES_SDATA_MASK	63:48	0–65535	0–65535
0028	15:0	R/W	ES_SDATA_MASK	79:64	0–65535	0–65535
0029	8:0	R/W	ES_VERT_OFFSET	8:0	0–511	0–511
002A	13:11	R/W	TXPI_SYNFREQ_PPM	2:0	0–7	0–7
002A	10	R/W	TXPI_PPMCLK_SEL	0	0–1	0–1
002A	9:2	R/W	TXPI_PPM_CFG	7:0	0–255	0–255
002A	1	R/W	TXPI_INVSTROBE_SEL	0	0–1	0–1
002A	0	R/W	TXPI_GREY_SEL	0	0–1	0–1
0032	15:0	R/W	TXPHDLY_CFG	15:0	0–65535	0–65535
0033	15	R/W	TX_CLKMUX_PD	0	0–1	0–1
0033	14	R/W	RX_CLKMUX_PD	0	0–1	0–1

Table D-2: **DRP Map of GTPE2_CHANNEL Primitive *(Cont'd)***

DRP Address	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
0033	13	R/W	TX_PREDRIVER_MODE	0	0–1	0–1
0033	12:8	R/W	TXPH_MONITOR_SEL	4:0	0–31	0–31
0033	7:0	R/W	TXPHDLY_CFG	23:16	0–255	0–255
0034	15:0	R/W	TXDLY_CFG	15:0	0–65535	0–65535
0035	15:0	R/W	TXDLY_TAP_CFG	15:0	0–65535	0–65535
0036	15	R/W	PMA_RSV5	0	0–1	0–1
0036	12	R/W	TX_LOOPBACK_DRIVE_HIZ	0	FALSE	0
					TRUE	1
0036	8:0	R/W	TXDLY_LCFG	8:0	0–511	0–511
0037	15:0	R/W	TXPH_CFG	15:0	0–65535	0–65535
0038	3:0	R/W	PMA_RSV4	3:0	0–15	0–15
0039	7:5	R/W	TXPI_CFG2	2:0	0–7	0–7
0039	4:2	R/W	TXPI_CFG1	2:0	0–7	0–7
0039	1:0	R/W	TXPI_CFG0	1:0	0–3	0–3
004A	3:0	R/W	RXLPM_CFG	3:0	0–15	0–15
004B	8:6	R/W	RXPI_CFG3	2:0	0–7	0–7
004B	5	R/W	RXPI_CFG2	0	0–1	0–1
004B	4:2	R/W	RXPI_CFG1	2:0	0–7	0–7
004B	1:0	R/W	RXPI_CFG0	1:0	0–3	0–3
004C	6:0	R/W	RXOOB_CFG	6:0	0–127	0–127
004D	13:0	R/W	RX_DEBUG_CFG	13:0	0–16383	0–16383
004E	15:0	R/W	RXDLY_TAP_CFG	15:0	0–65535	0–65535
004F	15:0	R/W	RXPH_CFG	15:0	0–65535	0–65535
0050	15:12	R/W	RX_CM_TRIM	3:0	0–15	0–15
0050	11:10	R/W	RX_CM_SEL	1:0	0–3	0–3
0050	9	R/W	RXLPM_HOLD_DURING_EIDLE	0	0–1	0–1
0050	8	R/W	RXCDR_HOLD_DURING_EIDLE	0	0–1	0–1
0050	7:0	R/W	RXPH_CFG	23:16	0–255	0–255
0051	13:9	R/W	RXPH_MONITOR_SEL	4:0	0–31	0–31
0051	8:0	R/W	RXDLY_LCFG	8:0	0–511	0–511
0052	15:0	R/W	RXLPM_LF_CFG	15:0	0–65535	0–65535
0053	15:2	R/W	RXLPM_HF_CFG	13:0	0–16383	0–16383

Table D-2: **DRP Map of GTPE2_CHANNEL Primitive *(Cont'd)***

DRP Address	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
0053	1:0	R/W	RXLPM_LF_CFG	17:16	0–3	0–3
0054	15:0	R/W	RXPHDLY_CFG	15:0	0–65535	0–65535
0055	7:0	R/W	RXPHDLY_CFG	23:16	0–255	0–255
0056	15:0	R/W	RXDLY_CFG	15:0	0–65535	0–65535
0059	14:10	R/W	RXCDRPHRESET_TIME	4:0	0–31	0–31
0059	9:5	R/W	RXCDRFREQRESET_TIME	4:0	0–31	0–31
0059	4:0	R/W	RXBUFRESET_TIME	4:0	0–31	0–31
005A	11:7	R/W	RXPMARESET_TIME	4:0	0–31	0–31
005A	6:0	R/W	RXLPMRESET_TIME	6:0	0–127	0–127
005B	4:0	R/W	RXPCSRESET_TIME	4:0	0–31	0–31
005C	14:10	R/W	TXPCSRESET_TIME	4:0	0–31	0–31
005C	9:5	R/W	TXPMARESET_TIME	4:0	0–31	0–31
005C	4:0	R/W	RXISCANRESET_TIME	4:0	0–31	0–31
005D	9:5	R/W	RXOSCALRESET_TIMEOUT	4:0	0–31	0–31
005D	4:0	R/W	RXOSCALRESET_TIME	4:0	0–31	0–31
005F	15:14	R/W	CHAN_BOND_SEQ_LEN	1:0	1	0
					2	1
					3	2
					4	3
005F	13:10	R/W	CHAN_BOND_SEQ_1_ENABLE	3:0	0–15	0–15
005F	9:0	R/W	CHAN_BOND_SEQ_1_1	9:0	0–1023	0–1023
0060	14	R/W	CHAN_BOND_KEEP_ALIGN	0	FALSE	0
					TRUE	1
0060	13:10	R/W	CHAN_BOND_MAX_SKEW	3:0	1–14	1–14
0060	9:0	R/W	CHAN_BOND_SEQ_1_2	9:0	0–1023	0–1023
0061	15	R/W	RXPRBS_ERR_LOOPBACK	0	0–1	0–1
0061	14:10	R/W	CLK_COR_REPEAT_WAIT	4:0	0–31	0–31
0061	9:0	R/W	CHAN_BOND_SEQ_1_3	9:0	0–1023	0–1023
0062	15:14	R/W	CLK_COR_SEQ_LEN	1:0	1	0
					2	1
					3	2
					4	3

Table D-2: **DRP Map of GTPE2_CHANNEL Primitive *(Cont'd)***

DRP Address	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
0062	13:10	R/W	CLK_COR_SEQ_1_ENABLE	3:0	0–15	0–15
0062	9:0	R/W	CHAN_BOND_SEQ_1_4	9:0	0–1023	0–1023
0063	15	R/W	RXGEARBOX_EN	0	FALSE	0
					TRUE	1
0063	14	R/W	CHAN_BOND_SEQ_2_USE	0	FALSE	0
					TRUE	1
0063	13:10	R/W	CHAN_BOND_SEQ_2_ENABLE	3:0	0–15	0–15
0063	9:0	R/W	CHAN_BOND_SEQ_2_1	9:0	0–1023	0–1023
0064	15:10	R/W	CLK_COR_MAX_LAT	5:0	3–60	3–60
0064	9:0	R/W	CHAN_BOND_SEQ_2_2	9:0	0–1023	0–1023
0065	15:10	R/W	CLK_COR_MIN_LAT	5:0	3–60	3–60
0065	9:0	R/W	CHAN_BOND_SEQ_2_3	9:0	0–1023	0–1023
0066	15	R/W	CLK_COR_PRECEDENCE	0	FALSE	0
					TRUE	1
0066	14	R/W	CLK_COR_KEEP_IDLE	0	FALSE	0
					TRUE	1
0066	9:0	R/W	CHAN_BOND_SEQ_2_4	9:0	0–1023	0–1023
0067	15	R/W	CLK_CORRECT_USE	0	FALSE	0
					TRUE	1
0067	14	R/W	CLK_COR_SEQ_2_USE	0	FALSE	0
					TRUE	1
0067	13:10	R/W	CLK_COR_SEQ_2_ENABLE	3:0	0–15	0–15
0067	9:0	R/W	CLK_COR_SEQ_1_1	9:0	0–1023	0–1023
0068	15	R/W	CBCC_DATA_SOURCE_SEL	0	ENCODED	0
					DECODED	1
0068	14	R/W	FTS_LANE_DESKEW_EN	0	FALSE	0
					TRUE	1
0068	13:10	R/W	FTS_DESKEW_SEQ_ENABLE	3:0	0–15	0–15
0068	9:0	R/W	CLK_COR_SEQ_1_2	9:0	0–1023	0–1023
0069	14	R/W	RX_DEFER_RESET_BUF_EN	0	FALSE	0
					TRUE	1
0069	13:10	R/W	FTS_LANE_DESKEW_CFG	3:0	0–15	0–15

Table D-2: **DRP Map of GTPE2_CHANNEL Primitive *(Cont'd)***

DRP Address	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
0069	9:0	R/W	CLK_COR_SEQ_1_3	9:0	0–1023	0–1023
006A	15:13	R/W	RX_DATA_WIDTH	2:0	16	2
					20	3
					32	4
					40	5
					0–7	0–7
006A	9:0	R/W	CLK_COR_SEQ_1_4	9:0	0–1023	0–1023
006B	15:13	R/W	RXOUT_DIV	2:0	1	0
					2	1
					4	2
					8	3
006B	11:10	R/W	RXSLIDE_MODE	1:0	OFF	0
					AUTO	1
					PCS	2
					PMA	3
006B	9:0	R/W	CLK_COR_SEQ_2_1	9:0	0–1023	0–1023

Table D-2: **DRP Map of GTPE2_CHANNEL Primitive *(Cont'd)***

DRP Address	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
006C	15:11	R/W	RX_SIG_VALID_DLY	4:0	1	0
					2	1
					3	2
					4	3
					5	4
					6	5
					7	6
					8	7
					9	8
					10	9
					11	10
					12	11
					13	12
					14	13
					15	14
					16	15
					17	16
					18	17
					19	18
					20	19
					21	20
					22	21
					23	22
					24	23
					25	24
					26	25
					27	26
					28	27
					29	28
					30	29
					31	30
					32	31

Table D-2: **DRP Map of GTPE2_CHANNEL Primitive *(Cont'd)***

DRP Address	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
006C	9:0	R/W	CLK_COR_SEQ_2_2	9:0	0–1023	0–1023
006D	15:12	R/W	RXSLIDE_AUTO_WAIT	3:0	0–15	0–15
006D	11	R/W	RXBUF_EN	0	FALSE	0
					TRUE	1
006D	10	R/W	RX_XCLK_SEL	0	RXREC	0
					RXUSR	1
006D	9:0	R/W	CLK_COR_SEQ_2_3	9:0	0–1023	0–1023
006E	15	R/W	TX_XCLK_SEL	0	TXOUT	0
					TXUSR	1

Table D-2: **DRP Map of GTPE2_CHANNEL Primitive *(Cont'd)***

DRP Address	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
006E	14:10	R/W	TX_CLK25_DIV	4:0	1	0
					2	1
					3	2
					4	3
					5	4
					6	5
					7	6
					8	7
					9	8
					10	9
					11	10
					12	11
					13	12
					14	13
					15	14
					16	15
					17	16
					18	17
					19	18
					20	19
					21	20
					22	21
					23	22
					24	23
					25	24
					26	25
					27	26
					28	27
					29	28
					30	29
					31	30
					32	31

Table D-2: **DRP Map of GTPE2_CHANNEL Primitive *(Cont'd)***

DRP Address	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
006E	9:0	R/W	CLK_COR_SEQ_2_4	9:0	0–1023	0–1023
006F	15:13	R/W	TX_DATA_WIDTH	2:0	16	2
					20	3
					32	4
					40	5
006F	12	R/W	PCS_PCIE_EN	0	FALSE	0
					TRUE	1
006F	10	R/W	ALIGN_MCOMMA_DET	0	FALSE	0
					TRUE	1
006F	9:0	R/W	ALIGN_MCOMMA_VALUE	9:0	0–1023	0–1023
0070	15:13	R/W	TXOUT_DIV	2:0	1	0
					2	1
					4	2
					8	3
0070	12	R/W	TXBUF_RESET_ON_RATE_CHANGE	0	FALSE	0
					TRUE	1
0070	11	R/W	TXBUF_EN	0	FALSE	0
					TRUE	1
0070	10	R/W	ALIGN_PCOMMA_DET	0	FALSE	0
					TRUE	1
0070	9:0	R/W	ALIGN_PCOMMA_VALUE	9:0	0–1023	0–1023
0071	15:13	R/W	TX_EIDLE_DEASSERT_DELAY	2:0	0–7	0–7
0071	11:0	R/W	PD_TRANS_TIME_FROM_P2	11:0	0–4095	0–4095
0072	15:8	R/W	PD_TRANS_TIME_TO_P2	7:0	0–255	0–255
0072	7:0	R/W	PD_TRANS_TIME_NONE_P2	7:0	0–255	0–255
0073	15:13	R/W	TX_EIDLE_ASSERT_DELAY	2:0	0–7	0–7
0073	12:8	R/W	TX_DRIVE_MODE	4:0	DIRECT	0
					PIPE	1
					PIPEGEN3	2
0073	7:0	R/W	TRANS_TIME_RATE	7:0	0–255	0–255
0074	15:0	R/W	TST_RSV	15:0	0–65535	0–65535
0075	15:0	R/W	TST_RSV	31:16	0–65535	0–65535

Table D-2: **DRP Map of GTPE2_CHANNEL Primitive *(Cont'd)***

DRP Address	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
0076	14	R/W	TXGEARBOX_EN	0	FALSE	0
					TRUE	1
0076	13	R/W	TX_MAINCURSOR_SEL	0	0–1	0–1
0076	11:6	R/W	TX_DEEMPH1	5:0	0–63	0–63
0076	5:0	R/W	TX_DEEMPH0	5:0	0–63	0–63
0077	15	R/W	RXSYNC_OVRD	0	0–1	0–1
0077	14	R/W	LOOPBACK_CFG	0	0–1	0–1
0077	13:0	R/W	TX_RXDETECT_CFG	13:0	0–16383	0–16383
0078	15	R/W	TXOOB_CFG	0	0–1	0–1
0078	14	R/W	TXSYNC_OVRD	0	0–1	0–1
0078	13:7	R/W	TX_MARGIN_FULL_1	6:0	0–127	0–127
0078	6:0	R/W	TX_MARGIN_FULL_0	6:0	0–127	0–127
0079	15	R/W	TXSYNC_SKIP_DA	0	0–1	0–1
0079	14	R/W	RXSYNC_SKIP_DA	0	0–1	0–1
0079	13:7	R/W	TX_MARGIN_FULL_3	6:0	0–127	0–127
0079	6:0	R/W	TX_MARGIN_FULL_2	6:0	0–127	0–127
007A	14	R/W	RXOOB_CLK_CFG	0	PMA	0
					FABRIC	1
007A	13:7	R/W	TX_MARGIN_LOW_0	6:0	0–127	0–127
007A	6:0	R/W	TX_MARGIN_FULL_4	6:0	0–127	0–127
007B	15	R/W	TXSYNC_MULTILANE	0	0–1	0–1
007B	14	R/W	RXSYNC_MULTILANE	0	0–1	0–1
007B	13:7	R/W	TX_MARGIN_LOW_2	6:0	0–127	0–127
007B	6:0	R/W	TX_MARGIN_LOW_1	6:0	0–127	0–127
007C	15:14	R/W	SATA_PLL_CFG	1:0	0–3	0–3
007C	13:7	R/W	TX_MARGIN_LOW_4	6:0	0–127	0–127
007C	6:0	R/W	TX_MARGIN_LOW_3	6:0	0–127	0–127
0084	15:0	R/W	PCS_RSVD_ATTR	15:0	0–65535	0–65535
0085	15:0	R/W	PCS_RSVD_ATTR	31:16	0–65535	0–65535
0086	15:0	R/W	PCS_RSVD_ATTR	47:32	0–65535	0–65535
008A	15:13	R/W	TX_RXDETECT_REF	2:0	0–7	0–7
008A	9	R/W	CLK_COMMON_SWING	0	0–1	0–1

Table D-2: **DRP Map of GTPE2_CHANNEL Primitive *(Cont'd)***

DRP Address	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
008A	8	R/W	USE_PCS_CLK_PHASE_SEL	0	0–1	0–1
008A	7	R/W	ES_CLK_PHASE_SEL	0	0–1	0–1
008A	6	R/W	PMA_LOOPBACK_CFG	0	0–1	0–1
008A	5	R/W	RXLPM_CFG1	0	0–1	0–1
008A	4	R/W	ACJTAG_RESET	0	0–1	0–1
008A	3	R/W	ACJTAG_DEBUG_MODE	0	0–1	0–1
008A	2	R/W	ACJTAG_MODE	0	0–1	0–1
008A	1	R/W	RXPLL_SEL	0	0–1	0–1
008A	0	R/W	TXPLL_SEL	0	0–1	0–1
008B	2	R/W	RESET_POWERSAVE_DISABLE	0	0–1	0–1
008B	1	R/W	PMA_RSV7	0	0–1	0–1
008B	0	R/W	PMA_RSV6	0	0–1	0–1
008D	1:0	R/W	PMA_RSV3	1:0	0–3	0–3
009C	13:7	R/W	CFOK_CFG3	6:0	0–127	0–127
009C	6:0	R/W	CFOK_CFG2	6:0	0–127	0–127
009D	15:0	R/W	CFOK_CFG	15:0	0–65535	0–65535
009E	15:0	R/W	CFOK_CFG	31:16	0–65535	0–65535
009F	10:0	R/W	CFOK_CFG	42:32	0–2047	0–2047
00A0	15:0	R/W	ADAPT_CFG0	15:0	0–65535	0–65535
00A1	8:5	R/W	RXLPM_HF_CFG2	3:0	0–15	0–15
00A1	4	R/W	RXLPM_BIAS_STARTUP_DISABLE	0	0–1	0–1
00A1	3:0	R/W	ADAPT_CFG0	19:16	0–15	0–15
00A2	15:0	R/W	PMA_RSV2	15:0	0–65535	0–65535
00A3	15:0	R/W	PMA_RSV2	31:16	0–65535	0–65535
00A4	9:0	R/W	ES_PMA_CFG	9:0	0–1023	0–1023
00A5	15:0	R/W	RXCDR_CFG	15:0	0–65535	0–65535
00A6	15:0	R/W	RXCDR_CFG	31:16	0–65535	0–65535
00A7	15:0	R/W	RXCDR_CFG	47:32	0–65535	0–65535
00A8	15:0	R/W	RXCDR_CFG	63:48	0–65535	0–65535
00A9	15:0	R/W	RXCDR_CFG	79:64	0–65535	0–65535
00AA	8:3	R/W	RXCDR_LOCK_CFG	5:0	0–63	0–63
00AA	2:0	R/W	RXCDR_CFG	82:80	0–7	0–7

Table D-2: **DRP Map of GTPE2_CHANNEL Primitive *(Cont'd)***

DRP Address	DRP Bits	R/W	Attribute Name	Attribute Bits	Attribute Encoding	DRP Encoding
00AB	12:0	R/W	RX_OS_CFG	12:0	0–8191	0–8191
00AC	15:0	R/W	PMA_RSV	15:0	0–65535	0–65535
00AD	15:0	R/W	PMA_RSV	31:16	0–65535	0–65535
00AE	15:0	R/W	RX_BIAS_CFG	15:0	0–65535	0–65535