Versal ACAP GTY Transceivers

Architecture Manual

AM002 (v1.0) July 16, 2020

XILINX®

Revision History

The following table shows the revision history for this document.

Section	Revision Summary
07/16/2020 Version 1.0	
Initial release.	N/A

Table of Contents

Revision History **2**

Chapter 1: Transceiver and Tool Overview **6**
- Introduction to Versal ACAP 6
- Features 8
- Versal ACAPs Transceivers Wizard 12
- Versal ACAPs Transceivers Bridge IP 12
- Versal ACAP Ports and Attributes 12
- Simulation 13
- Implementation 14

Chapter 2: Shared Features **15**
- Reference Clock Input/Output Structure 15
- Reference Clock Selection and Distribution 19
- Ring PLL 29
- LC-Tank PLL 35
- Reset and Initialization 39
- Rate Change 66
- Power Down 70
- Loopback 72
- Fabric Configuration Interface 73
- Digital Monitor 74

Chapter 3: Transmitter **76**
- TX Interface 77
- TX 8B/10B Encoder 83
- TX 128B/130B Encoder 87
- TX Buffer 88
- TX Buffer Bypass 91
- TX Synchronous Gearbox 118
- TX Asynchronous Gearbox 122
- TX Pattern Generator 127

TX CRC Generator....131
TX Polarity Control....134
TX Fabric Clock Output Control....134
TX Phase Interpolator PPM Controller....140
TX Configurable Driver....142
TX Receiver Detect Support for PCI Express Designs....147
TX Out-of-Band Signaling....149

Chapter 4: Receiver....152
RX Analog Front End....153
RX Out-of-Band Signaling....159
RX Equalizer (DFE and LPM)....164
RX CDR....166
RX Interface....168
RX Fabric Clock Output Control....173
RX Margin Analysis....179
RX Polarity Control....197
RX Pattern Checker....197
RX CRC Checker....200
RX Byte and Word Alignment....203
RX 8B/10B Decoder....211
RX 128/130B Decoder....216
RX Buffer....217
RX Buffer Bypass....219
RX Clock Correction....231
RX Channel Bonding....238
RX Synchronous Gearbox....244
RX Asynchronous Gearbox....247

Chapter 5: 8B/10B Valid Characters....253

Chapter 6: Board Design Guidelines....261
Pin Description and Design Guidelines....261
Reference Clock....263
GTY Transceiver Reference Clock Checklist....265
Reference Clock Interface....266
Power Supply....268
Power Supply Distribution Network Staged Decoupling....272
PCB Design Checklist....273

Appendix A: Additional Resources and Legal Notices 276
Xilinx Resources 276
Documentation Navigator and Design Hubs 276
References 276
Please Read: Important Legal Notices 277

XILINX

Chapter 1

Transceiver and Tool Overview

Introduction to Versal ACAP

Versal™ adaptive compute acceleration platforms (ACAPs) combine Scalar Engines, Adaptable Engines, and Intelligent Engines with leading-edge memory and interfacing technologies to deliver powerful heterogeneous acceleration for any application. Most importantly, Versal ACAP hardware and software are targeted for programming and optimization by data scientists and software and hardware developers. Versal ACAPs are enabled by a host of tools, software, libraries, IP, middleware, and frameworks to enable all industry-standard design flows.

Built on the TSMC 7 nm FinFET process technology, the Versal portfolio is the first platform to combine software programmability and domain-specific hardware acceleration with the adaptability necessary to meet today's rapid pace of innovation. The portfolio includes six series of devices uniquely architected to deliver scalability and AI inference capabilities for a host of applications across different markets—from cloud—to networking—to wireless communications—to edge computing and endpoints.

The Versal architecture combines different engine types with a wealth of connectivity and communication capability and a network on chip (NoC) to enable seamless memory-mapped access to the full height and width of the device. Intelligent Engines are SIMD VLIW AI Engines for adaptive inference and advanced signal processing compute, and DSP Engines for fixed point, floating point, and complex MAC operations. Adaptable Engines are a combination of programmable logic blocks and memory, architected for high-compute density. Scalar Engines, including Arm® Cortex™-A72 and Cortex-R5F processors, allow for intensive compute tasks.

The Versal AI Core series delivers breakthrough AI inference acceleration with AI Engines that deliver over 100x greater compute performance than current server-class of CPUs. This series is designed for a breadth of applications, including cloud for dynamic workloads and network for massive bandwidth, all while delivering advanced safety and security features. AI and data scientists, as well as software and hardware developers, can all take advantage of the high-compute density to accelerate the performance of any application.

The Versal Prime series is the foundation and the mid-range of the Versal platform, serving the broadest range of uses across multiple markets. These applications include 100G to 200G networking equipment, network and storage acceleration in the Data Center, communications test equipment, broadcast, and aerospace & defense. The series integrates mainstream 58G transceivers and optimized I/O and DDR connectivity, achieving low-latency acceleration and performance across diverse workloads.

The Versal Premium series provides breakthrough heterogeneous integration, very high-performance compute, connectivity, and security in an adaptable platform with a minimized power and area footprint. The series is designed to exceed the demands of high-bandwidth, compute-intensive applications in wired communications, data center, test & measurement, and other applications. Versal Premium series ACAPs include 112G PAM4 transceivers and integrated blocks for 600G Ethernet, 600G Interlaken, PCI Express® Gen5, and high-speed cryptography.

The Versal architecture documentation suite is available at: https://www.xilinx.com/versal.

Navigating Content by Design Process

Xilinx® documentation is organized around a set of standard design processes to help you find relevant content for your current development task. This document covers the following design processes:

- **System and Solution Planning:** Identifying the components, performance, I/O, and data transfer requirements at a system level. Includes application mapping for the solution to PS, PL, and AI Engine. Topics in this document that apply to this design process include:
  - Versal ACAPs Transceivers Wizard
  - LC-Tank PLL
  - Features
- **Hardware, IP, and Platform Development:** Creating the PL IP blocks for the hardware platform, creating PL kernels, subsystem functional simulation, and evaluating the Vivado® timing, resource use, and power closure. Also involves developing the hardware platform for system integration. Topics in this document that apply to this design process include:
  - Versal ACAPs Transceivers Wizard
  - Versal ACAPs Transceivers Bridge IP
  - Simulation
- **System Integration and Validation:** Integrating and validating the system functional performance, including timing, resource use, and power closure. Topics in this document that apply to this design process include:
  - Versal ACAPs Transceivers Wizard
  - Chapter 6: Board Design Guidelines

- **Board System Design:** Designing a PCB through schematics and board layout. Also involves power, thermal, and signal integrity considerations. Topics in this document that apply to this design process include:
  - Chapter 6: Board Design Guidelines
  - Reference Clock Selection and Distribution

Features

The Versal ACAP GTY transceiver provides the greatest performance and integration at 7 nm, including serial I/O bandwidth and logic capacity. As the industry's high-end FPGA at the 7 nm process node, this product family is ideal for applications including 400G networking, large-scale ASIC prototyping, and emulation.

The GTY transceivers in the Versal architecture are power-efficient transceivers, supporting line rates from 1.25 Gb/s to 32.75 Gb/s. The GTY transceiver is highly configurable and tightly integrated with the programmable logic resources of the Versal architecture. The following table summarizes the features by functional group that support a wide variety of applications.

Table 1: **Transceiver Features**

Group	Feature
PCS	2-byte, 4-byte, and 8-byte internal datapath to support different line rate requirements
	8B/10B encoding and decoding
	64B/66B and 64B/67B support
	128B/130B encoding and decoding for PCI Express® Gen3 and Gen4
	Comma detection and byte and word alignment
	TX phase FIFO
	RX elastic FIFO for clock correction and channel bonding
	Buffer bypass support for fixed latency
	PRBS generator and checker
	Programmable FPGA logic interface
	100 Gb attachment unit interface (CAUI) support
	Native multi-lane support for buffer bypass
	TX phase interpolator PPM controller for external voltage-controller crystal oscillator (VCXO) replacement
	Out-of-band (OOB) signaling including COM signal support for serial ATA (SATA) designs

Table 1: **Transceiver Features** *(cont'd)*

Group	Feature
PMA	Two LC tank and two ring oscillator PLLs per Quad for best jitter performance
	Power-efficient adaptive linear equalizer mode called the lower-power mode (LPM) with auto adapt
	15-tap decision feedback equalization (DFE) with auto adapt
	TX pre-emphasis
	Programmable TX output
	Beacon signaling for PCI Express designs
	Line rate support up to 32.75 Gb/s for Versal devices

The GTY transceiver supports the following use modes:

- Aurora
- CCIX 16G/20G/25G
- Common Packet Radio Interface (CPRI)
- DisplayPort
- Fibre Channel
- HDMI 2.0/2.1
- Interlaken
- JESD204b/c
- OC-3/12/48/192
- Optical channel transport unit (OTU): OTU-1, OTU-2, OUT-2e, OTU-3, OTU-4
- PCI Express, revision 1.1, 2.0, 3.0, and 4.0
- Serial RapidIO (SRIO)
- Serial advanced technology attachment (SATA), serial attached SCSI (SAS)
- Serial digital interface (SDI)
- SFF-8431 (SFP+)
- USB 3.0
- VbyOne
- 10GBASE-R/KR
- 10 Gb attachment unit interface (XAUI), reduced pin extended attachment unit interface (RXAUI), 100 Gb attachment unit interface (CAUI), 40 Gb attachment unit interface (XLAUI)

Key Differences from Previous FPGA Generations

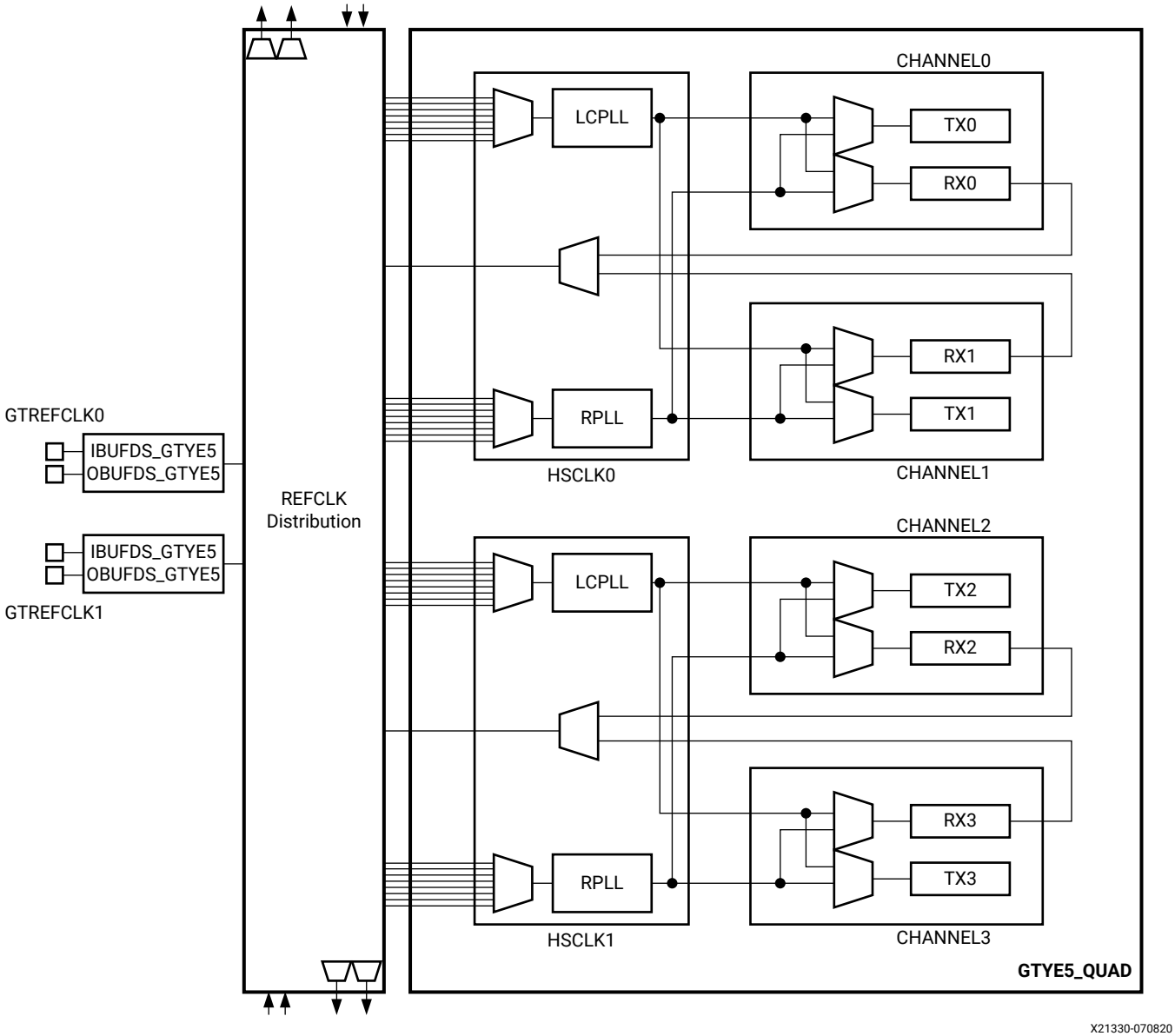
- Additional datapath to support CCIX

- GTY primitive is a single Quad instead of separate COMMON and CHANNEL primitives
- Single USRCLK clocking scheme driven by TX/RXOUTCLK

The figure below illustrates the clustering of four transceiver channel (CHANNEL) blocks and two high speed clocking (HSCLK) blocks to form the GTYE5_QUAD primitive.

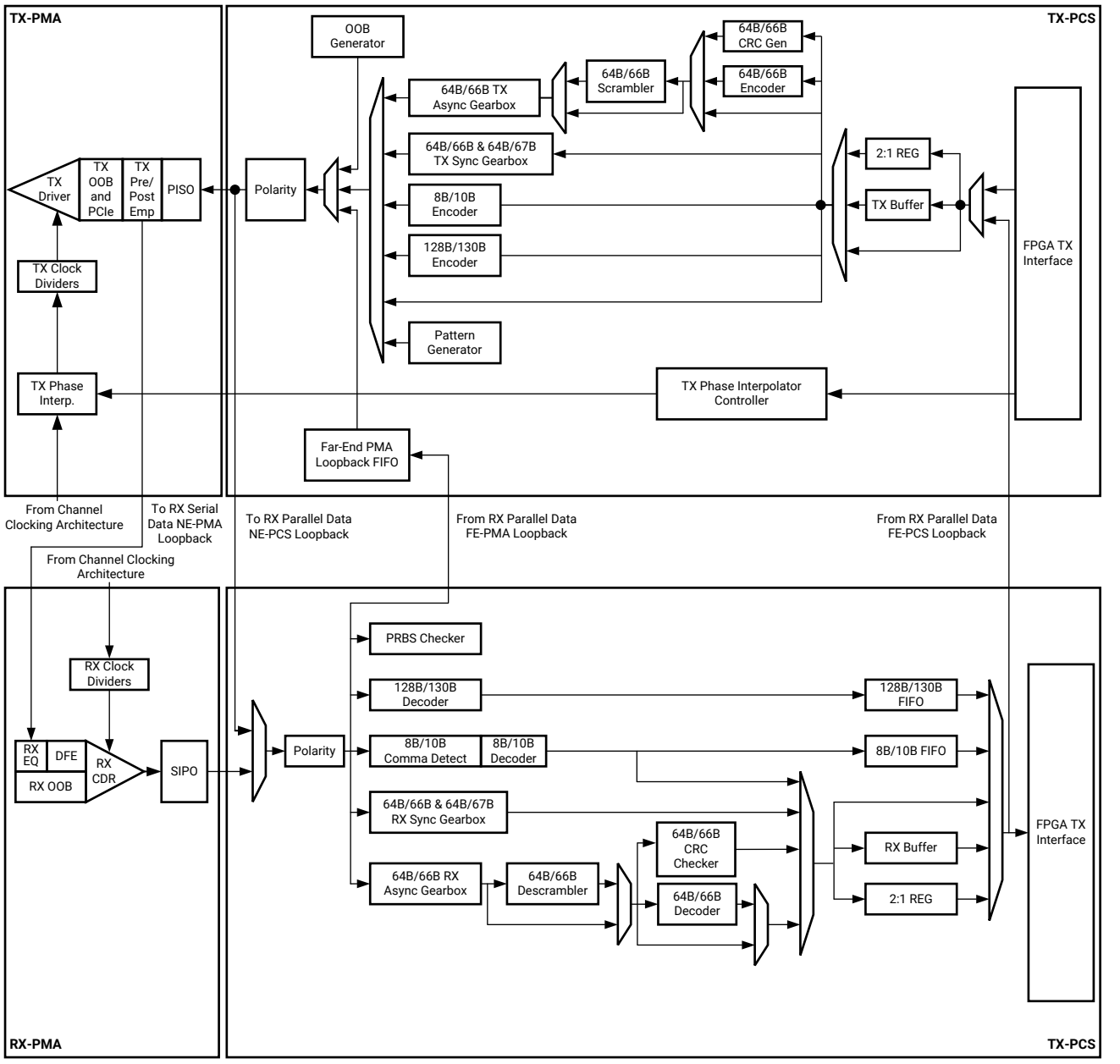
***Note*:** The GTY Quad primitive is called GTYE5_QUAD in Versal ACAPs.

Figure 1: **Transceiver Quad Configuration**



Four CHANNEL blocks clustered together with two HSCLK blocks form a *Quad* or *Q*. Each HSCLK block contains one LC-tank PLL (LCPLL) and one ring oscillator PLL (RPLL). PLLs inside HSCLK0 can only provide a clock to CHANNEL0/1 and PLLs inside HSCLK1 can only provide a clock to CHANNEL2/3. Each CHANNEL block consists of a transmitter and a receiver. The following figure illustrates the topology of the GTY channel.

Figure 2: **Channel Topology**



Refer to Ring PLL for the description of the channel clocking architecture, which provides clocks to the RX and TX clock dividers.

Versal ACAPs Transceivers Wizard

The Versal ACAPs Transceivers Wizard (hereinafter called the Wizard) is the preferred tool to generate a wrapper to instantiate the GTYE5_QUAD primitives in Versal devices. The Wizard is located in the IP catalog under the I/O interfaces category.

RECOMMENDED: *Download the most recent IP update before using the Wizard.*

Versal ACAPs Transceivers Bridge IP

The Versal ACAPs transceivers Bridge IP is the preferred way to generate designs that do not rely on Xilinx IP cores but still require instantiation of multiple Quads in Versal devices. The bridge IP is located in the IP catalog.

RECOMMENDED: *Download the most recent IP update before using the Bridge IP.*

Versal ACAP Ports and Attributes

The Versal ACAP transceiver primitive contains ports and attributes that must be configured correctly for optimal performance. The user should rely on the Wizard to configure the transceiver when possible, and take care when performing manual configuration following the port and attribute descriptions in this document. Any port or attribute that is not explicitly described is assumed to be reserved, and its value should be left at the default generated by example designs from the Wizard or other IP cores.

The attributes in the Versal ACAP transceiver primitive also carry *labels* for each of the described sub-fields. These are not actual names in the UNISIMs but descriptive titles of the sub-fields following a similar naming convention in the UltraScale™ architecture documentation. The purpose of these is to help the user become familiar with the sub-fields and improve search efficiency while using this document.

RECOMMENDED: *Download the most recent IP update before using the Wizard. Xilinx recommends using the latest version of the Vivado® Design Suite for best performance.*

Simulation

The simulation environment and the test bench must fulfill specific prerequisites before running simulation using the transceiver primitives. 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 the *Vivado Design Suite User Guide: Logic Simulation* (UG900).

The prerequisites for simulating a design with the GTYE5_QUAD primitives are listed:

- A simulator with support for SecureIP models: SecureIP is an IP encryption methodology. SecureIP models are encrypted versions of the Verilog HDL used for implementation of the modeled block. To support SecureIP models, a simulator that complies with the encryption standards described in the Verilog language reference manual (LRM)—IEEE Standard for Verilog Hardware Description Language (IEEE Std 1364-2005) 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 GTY transceiver SecureIP model.
- The correct setup of the simulator for SecureIP use (initialization file, environment variables).
- The correct simulator resolution (Verilog).

Ports and Attributes

There are no simulation-only ports on the GTYE5_QUAD primitives.

Table 2: **Simulation-Only Attributes**

Attribute	Type	Description
SIM_DEVICE	String	This attribute selects the simulation version to match different revisions of silicon: VERSAL_[*]_ES1: Engineering samples 1 VERSAL_[*]_ES2: Engineering samples 2 VERSAL_[*]: Production silicon Where [*] denotes the actual Versal device: AI_CORE AI_EDGE AI_RF HBM PREMIUM PRIME
QUAD_SIM_MODE	String	This attribute selects the simulation mode. The default value is FAST.

Table 2: **Simulation-Only Attributes** *(cont'd)*

Attribute	Type	Description
QUAD_SIM_RESET_SPEEDUP	String	When set to TRUE (default), an approximate reset sequence is used to speed up the reset time for simulations, where faster reset times and faster simulation time are desired. When set to FALSE, the model emulates hardware reset behavior in detail.
CH[0/1/2/3]_SIM_MODE	String	This attribute selects the simulation mode. The default value is FAST. The value should match QUAD_SIM_MODE.
CH[0/1/2/3]_SIM_RESET_SPEEDUP		When set to TRUE (default), an approximate reset sequence is used to speed up the reset time for simulations, where faster reset times and faster simulation time are desired. When set to FALSE, the model emulates hardware reset behavior in detail. The value should match QUAD_SIM_RESET_SPEEDUP.
CH[0/1/2/3]_SIM_RECEIVER_DETECT_PASS	String	Determines if receiver detection should indicate a pass or fail by setting this attribute to TRUE or FALSE, respectively.
CH[0/1/2/3]_SIM_TX_EIDLE_DRIVE_LEVEL	String	The attribute can be set to LOW, HIGH, X, or Z to allow for the value driven onto TXP and TXN in simulation to represent the electrical idle condition. This is for simulation only and has no impact on actual device operation. The default value for this attribute is Z.

Implementation

It is a common practice to define the location of 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 XDC file.

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

Use the I/O planner in the Vivado I/O planner to set the transceiver locations and be modified manually to change the placement locations. Care must be taken to ensure that all of the parameters needed to configure the transceivers are correctly entered.

Shared Features

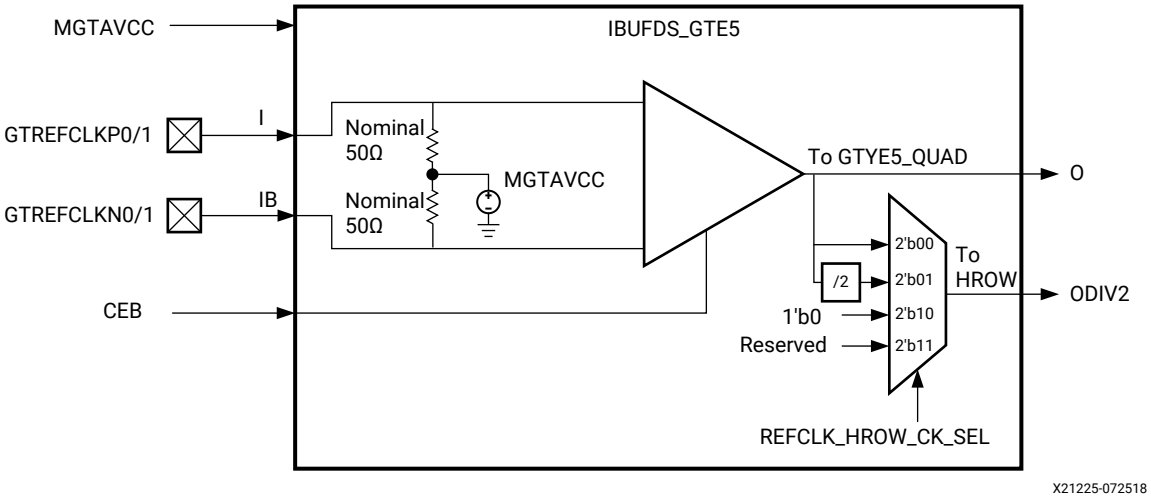
Reference Clock Input/Output Structure

The reference clock structure in the GTY transceiver supports two modes of operation: input mode and output mode. In the input mode of operation, your design provides a clock on the dedicated reference clock I/O pins that are used to drive the LCPLLs and RPLLs. In the output mode of operation, the recovered clocks (HSCLK*_RXRECCLKOUT0/1) from any of the four channels within the same Quad can be routed to the dedicated reference clock I/O pins. This output clock can then be used as the reference clock input at a different location. The mode of operation cannot be changed during run time.

Input Mode

The reference clock input mode structure is illustrated in the following figure. The input is terminated internally with 50Ω on each leg to MGTAVCC. The reference clock is instantiated in software with the IBUFDS_GTE5 software primitive. The ports and attributes controlling the reference clock input are tied to the IBUFDS_GTE5 software primitive.

Figure 3: **Reference Clock Input Structure**



IMPORTANT! *Upon device configuration, the clock output from the IBUFDS_GTE5 which takes inputs from MGTREFCLK[0/1]P and MGTREFCLK[0/1]N can only be used under the following conditions:*

- The GTPOWERGOOD signal has already asserted High.

Ports and Attributes

The following table defines the reference clock input ports in the IBUFDS_GTE5 software primitive.

Table 3: **Reference Clock Input Ports (IBUFDS_GTE5)**

Port	Dir	Clock Domain	Description
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.
I	In (pad)	N/A	These are the reference clock input ports that get mapped to GTREFCLK0P and GTREFCLK1P.
IB	In (pad)	N/A	These are the reference clock input ports that get mapped to GTREFCLK0N and GTREFCLK1N.
O	Out	N/A	This output drives the GTYE5_QUAD primitive. Refer to Reference Clock Selection and Distribution for more details.
ODIV2	Out	N/A	This output can be configured to output either the O signal or a divide-by-2 version of the O signal. It can drive the BUFG_GT via the HROW routing. Refer to Reference Clock Selection and Distribution for more details.

The following table defines the attributes in the IBUFDS_GTE5 software primitive that configure the reference clock input.

Table 4: **Reference Clock Input Attributes (IBUFDS_GTE5)**

Attribute	Type	Description
REFCLK_EN_TX_PATH	1-bit Binary	Reserved. This attribute must always be set to `1'b0`.
REFCLK_HROW_CK_SEL	Integer	Configures the ODIV2 output port: 0: ODIV2 = O. 1: ODIV2 = Divide-by-2 version of O. 2: ODIV2 = `1'b0`. 3: ODIV2 = Reserved.
REFCLK_ICNTL_RX	Integer	Reserved. Use the recommended value from the Wizard.
REFCLK_CTL_DRV_SWING	3-bit Binary	Reserved. Use the recommended value from the Wizard.
REFCLK_EN_DRV	1-bit Binary	Reserved. Use the recommended value from the Wizard.
RXRECCLK_SEL	2-bit Binary	Reserved. Use the recommended value from the Wizard.

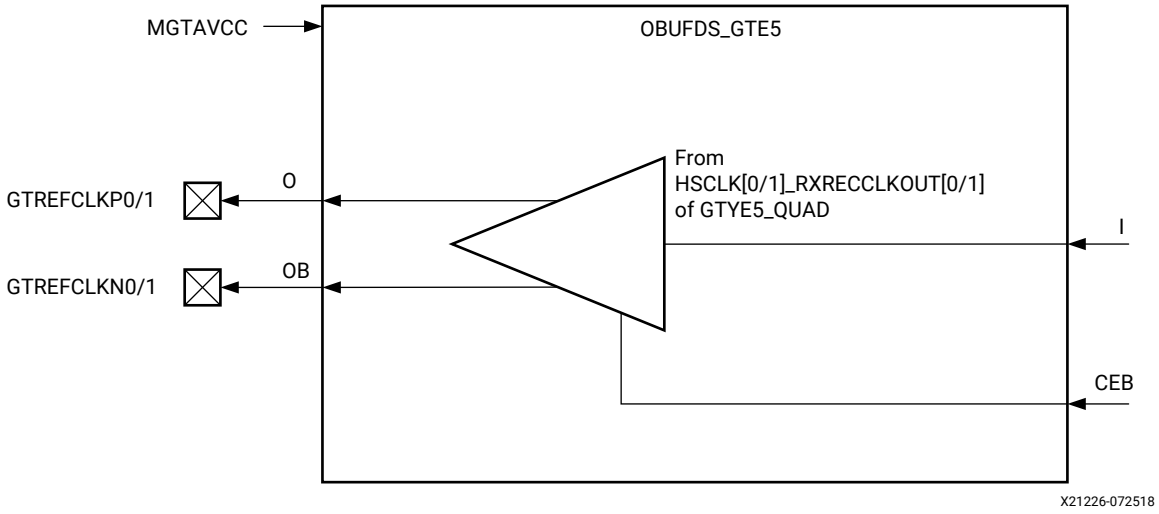
Output Mode

The reference clock output mode can be accessed via one of the two software primitives: OBUFDS_GTE5 and OBUFDS_GTE5_ADV. The choice of the primitive depends on your application. Use OBUFDS_GTE5 when the CH*_RXRECCLKOUT[0/1] is always derived from the same channel. Use OBUFDS_GTE5_ADV if the channel providing CH*_RXRECCLKOUT[0/1] can change during runtime.

OBUFDS_GTE5

The reference clock output mode structure with the OBUFDS_GTE5 primitive is shown in the following figure. The ports and attributes controlling the reference clock output are tied to the OBUFDS_GTE5 software primitive.

Figure 4: **Reference Clock Output Use Model with OBUFDS_GTE5**



Ports and Attributes

The following table defines the ports in the OBUFDS_GTE5 software primitive.

Table 5: **Reference Clock Output Ports (OBUFDS_GTE5)**

Port	Dir	Clock Domain	Description
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.
I	In	N/A	Recovered clock input. Connect to the output port HSCLK[0/1]_RXRECCLKOUT[0/1] of one of the four channels in the GTYE5_QUAD.
O	Out	N/A	Reference clock output port that gets mapped to GTREFCLK0P and GTREFCLK1P.

Table 5: **Reference Clock Output Ports (OBUFDS_GTE5)** *(cont'd)*

Port	Dir	Clock Domain	Description
OB	Out	N/A	Reference clock output port that gets mapped to GTREFCLK0N and GTREFCLK1N.

The following table defines the attributes in the OBUFDS_GTE5 software primitive that configure the reference clock output.

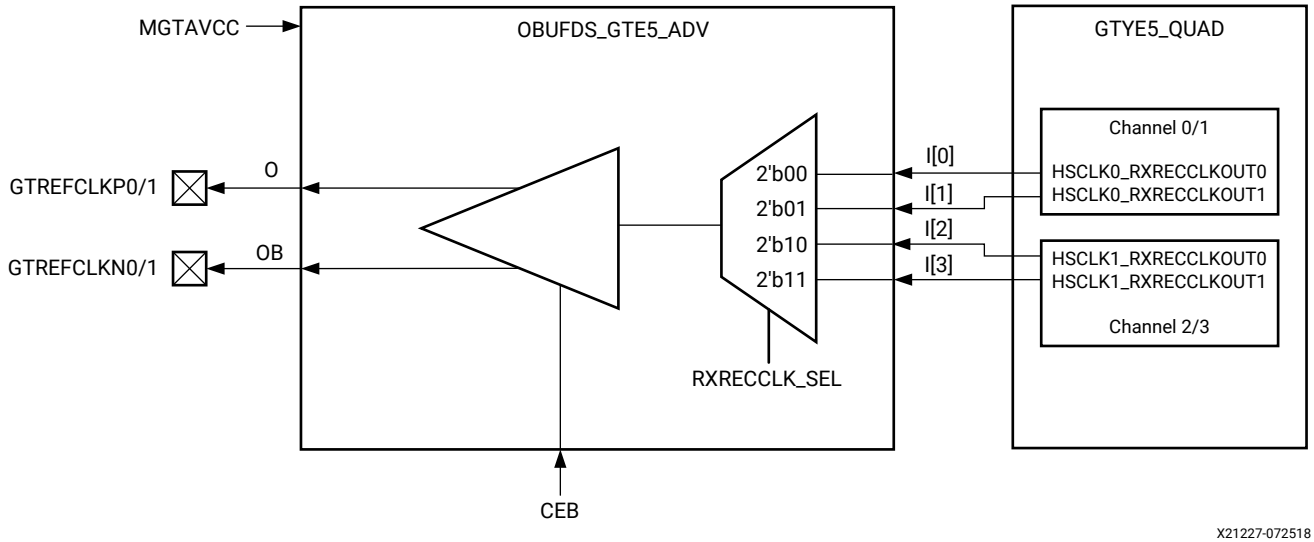
Table 6: **Reference Clock Output Attributes (OBUFDS_GTE5)**

Attribute	Address	Description
REFCLK0_REFCLK_EN_TX_PATH	0x0DBF[17]	Reserved. This attribute must always be set to `1'b1`.
REFCLK1_REFCLK_EN_TX_PATH	0x0FBF[17]	Reserved. This attribute must always be set to `1'b1`.

OBUFDS_GTE5_ADV

The reference clock output mode structure with the OBUFDS_GTE5_ADV primitive is shown in the following figure. The ports and attributes controlling the reference clock output are tied to the OBUFDS_GTE5_ADV software primitives. The attribute RXRECCLK_SEL controls the multiplexer that selects between CH*_RXRECCLKOUT[0/1] from the four different channels in a Quad.

Figure 5: **Reference Clock Output Use Model with OBUFDS_GTE5_ADV**



Ports and Attributes

The following table defines the ports in the OBUFDS_GTE5_ADV software primitive.

Table 7: **Reference Clock Output Ports (OBUFDS_GTE5_ADV)**

Port	Dir	Clock Domain	Description
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.
I[3:0]	In	N/A	Recovered clock input bus. Connect I[0] to the HSCLK0_RXRECCLKOUT0 of GTYE5_QUAD mapping to channel 0. Connect I[1] to the HSCLK0_RXRECCLKOUT1 of GTYE5_QUAD mapping to channel 1. Connect I[2] to the HSCLK1_RXRECCLKOUT0 of GTYE5_QUAD mapping to channel 2. Connect I[3] to the HSCLK1_RXRECCLKOUT1 of GTYE5_QUAD mapping to channel 3.
O	Out	N/A	Reference clock output ports that get mapped to GTREFCLK0P and GTREFCLK1P.
OB	Out	N/A	Reference clock output ports that get mapped to GTREFCLK0N and GTREFCLK1N.

The following table defines the attributes in the OBUFDS_GTE5_ADV software primitive that configure the reference clock output.

Table 8: **Reference Clock Output Attributes (OBUFDS_GTE5_ADV)**

Attribute	Address	Description
REFCLK0_REFCLK_EN_TX_PATH	0x0DBF[17]	Reserved. This attribute must always be set to `1'b1`.
REFCLK0_RXRECCLK_SEL	0x0DBF[14:13]	Recovered clock output selection control on the GTREFCLK0 pin. `2'b00`: From channel 0 RXPROGDIV (HSCLK0_RXRECCLKOUT0) `2'b01`: From channel 1 RXPROGDIV (HSCLK0_RXRECCLKOUT1) `2'b10`: From channel 2 RXPROGDIV (HSCLK1_RXRECCLKOUT0) `2'b11`:From channel 3 RXPROGDIV (HSCLK1_RXRECCLKOUT1)
REFCLK1_REFCLK_EN_TX_PATH	0x0FBF[17]	Reserved. This attribute must always be set to `1'b1`.
REFCLK1_RXRECCLK_SEL	0x0FBF[14:13]	Recovered clock output selection control on pin GTREFCLK1. `2'b00`: From channel 0 RXPROGDIV (HSCLK0_RXRECCLKOUT0) `2'b01`: From channel 1 RXPROGDIV (HSCLK0_RXRECCLKOUT1) `2'b10`: From channel 2 RXPROGDIV (HSCLK1_RXRECCLKOUT0) `2'b11`:From channel 3 RXPROGDIV (HSCLK1_RXRECCLKOUT1)

Reference Clock Selection and Distribution

The Versal ACAP transceivers provide different reference clock input options. Clock selection supports two LCPLLs and two RPLLs.

From an architecture perspective, a Quad (or Q) contains four channels, two HSCLK blocks, two dedicated external reference clock pin pairs, and dedicated reference clock routing. The following list provides additional constraints on the above resources within a Quad.

- There is one RPLL and one LCPLL within each of the HSCLK0/1 blocks.
- PLLs from HSCLK0 can only provide clocking to Channel 0 and 1.
- PLLs from HSCLK1 can only provide clocking to Channel 2 and 3.

The GTYE5_QUAD primitive must be instantiated when any PLL or transceiver channel inside the Quad is used. In general, the reference clock for a Quad (Q(n)) can also be sourced from up to two Quads below (Q(n–1) or Q(n–2)) or from up to two Quads above (Q(n+1) or Q(n+2)). For devices that support stacked silicon interconnect (SSI) technology, the reference clock sharing is limited within its own super logic region (SLR).

See the Versal ACAP data sheets for more information about SSI technology.

For Versal devices, sourcing of the reference clock is limited to two Quads above and below when the channel is operating at or below 16.375 Gb/s. For line rates greater than 16.375 Gb/s, no reference clock sharing is allowed.

Reference clock features include:

- Clock routing for northbound and southbound clocks.
- Flexible clock inputs available for the LCPLL or RPLL.
- Static or dynamic selection of the reference clock for the LCPLL or RPLL.

The Quad architecture has four GTY transceivers, two dedicated reference clock pin pairs, and dedicated north or south reference clock routing. Each GTY transceiver channel in a Quad has six clock inputs available according to the corresponding PLL resource assignments shown in the following table.

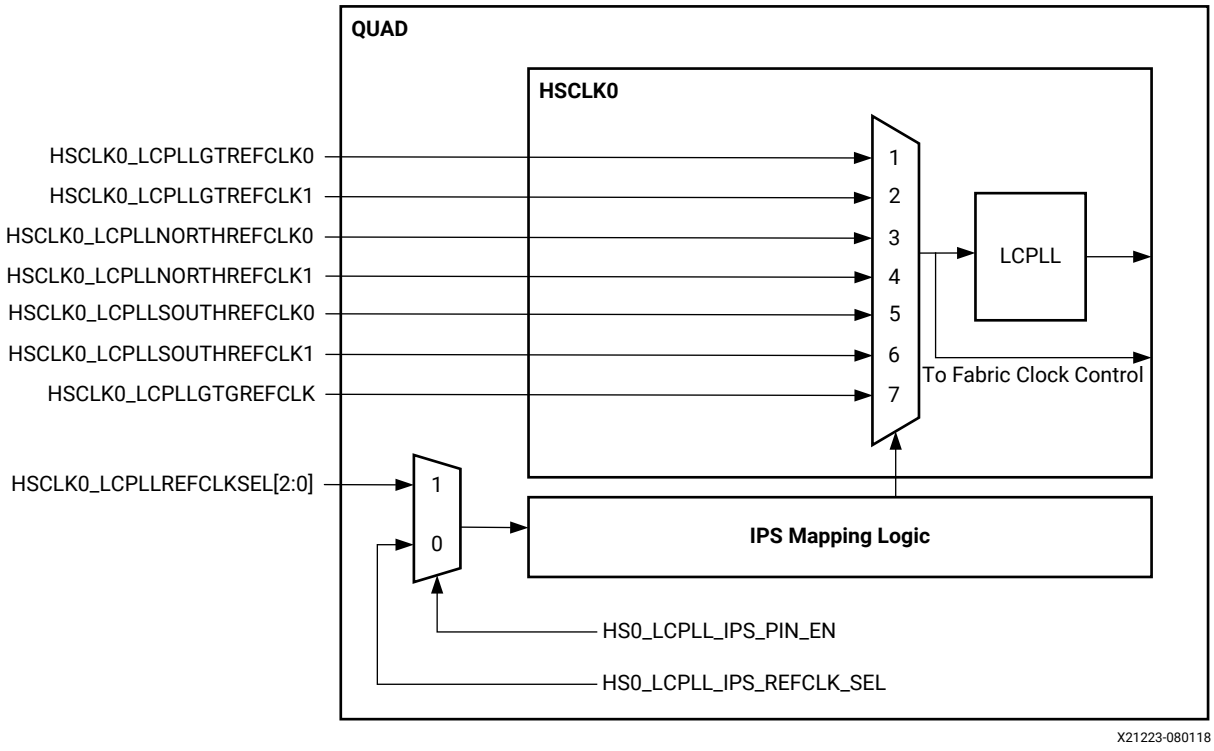
Table 9: **Reference Clock Sharing for Versal ACAP GTY Transceivers**

Clock Source	PLL Used	GTY Transceiver Channel 0/1	GTY Transceiver Channel 2/3
Two local reference clock pin pairs	RPLL	HSCLK0_RPLLGTREFCLK0 HSCLK0_RPLLGTREFCLK1	HSCLK1_RPLLGTREFCLK0 HSCLK1_RPLLGTREFCLK1
	LCPLL	HSCLK0_LCPLLGTREFCLK0 HSCLK0_LCPLLGTREFCLK1	HSCLK1_LCPLLGTREFCLK0 HSCLK1_LCPLLGTREFCLK1
Two reference clock pin pairs from Quads above	RPLL	HSCLK0_RPLLSOUTHREFCLK0 HSCLK0_RPLLSOUTHREFCLK1	HSCLK1_RPLLSOUTHREFCLK0 HSCLK1_RPLLSOUTHREFCLK1
	LCPLL	HSCLK0_LCPLLSOUTHREFCLK0 HSCLK0_LCPLLSOUTHREFCLK1	HSCLK1_LCPLLSOUTHREFCLK0 HSCLK1_LCPLLSOUTHREFCLK1
Two reference clock pin pairs from Quads below	RPLL	HSCLK0_RPLLNORTHREFCLK0 HSCLK0_RPLLNORTHREFCLK1	HSCLK1_RPLLNORTHREFCLK0 HSCLK1_RPLLNORTHREFCLK1
	LCPLL	HSCLK0_LCPLLNORTHREFCLK0 HSCLK0_LCPLLNORTHREFCLK1	HSCLK1_LCPLLNORTHREFCLK0 HSCLK1_LCPLLNORTHREFCLK1

Because there are only two south clock inputs and four potential clock sources from the two Quads above (Q(n+1)) and Q(n+2)), only a maximum of two of the four potential reference clock pin pairs from above can be physically connected up to Q(n) at any given moment. The four potential reference clock pin pairs from above are reduced to two or three if the Quad above (Q(n+1)) is itself sourcing reference clock pin pairs from two above (Q(n+3)). This is because there are a total of two south reference clock routing tracks connecting the Quads. Similar rules apply when sourcing a reference clock from Quads below. Because there are two north clock inputs and four potential clock sources from the two Quads below (Q(n–1) and Q(n–2)), only a maximum of two of the four potential reference clock pin pairs from below can be physically connected up to Q(n) at any given moment. The four potential reference clock pin pairs from below is reduced to two or three if the Quad below (Q(n–1)) is itself sourcing reference clock pin pairs from two below Q(n–3). Again, this is because there are a total of two north reference clock routing tracks connecting the Quads. For example, Q(n–1) is sourcing both reference clocks from Q(n–3). In this example, Q(n) would only be able to source reference clock pins below from Q(n–1). Q(n) would not be able to access the reference clock pins in Q(n–2) because the two routing tracks have already been used to bring the two reference clocks from Q(n–3) to Q(n–1).

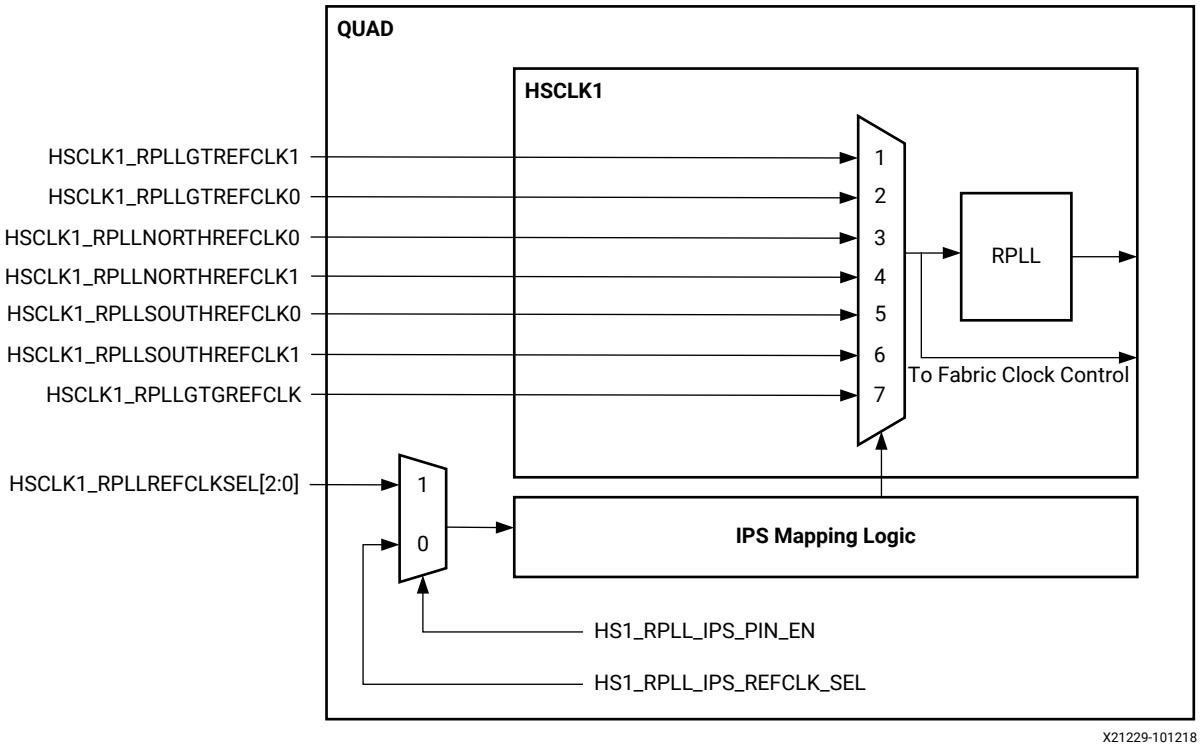
The following figure shows the detailed view of the reference clock multiplexer structure within a single HSCLK block, using LCPLL inside block HSCLK0 as an example. The same structure applies to LCPLL inside HSCLK1. When single or multiple reference clock sources are connected, the designer first needs to make sure that the connections are made to the correct PLLs using the reference clock ports assigned to the each intended PLL. The enhanced intelligent pin selection (IPS) automatically analyzes the design and maps reference clock selection during design implementation to guarantee that clocks are properly connected. For additional details on IPS, see Intelligent Pin Selection.

Figure 6: **LCPLL Reference Clock Selection Multiplexer**



Similarly, the following figure shows the detailed view of the reference clock multiplexer structure within a single HSCLK block, but with RPLL being used inside HSCLK1 as another example. Note that the connections for the two local GTREFCLKs are connected differently inside HSCLK1 compared to HSCLK0. HSCLK1_RPLL/LCPLLGTREFCLK1 is connected to input 1 on the input multiplexer while HSCLK1_RPLL/LCPLLGTREFCLK0 is connected to input 2 on the input multiplexer.

Figure 7: **RPLL Reference Clock Selection Multiplexer**



The following tables list the corresponding reference clock selection multiplexer settings.

Table 10: **LCPLL Reference Clock Selection Multiplexer Settings**

HSCLK[0/1]_LCPLLREFCLKSEL	HS[0/1]_LCPLL_IPS_REFCLK_SEL	Selected Input to LCPLL
3'b001	1	HSCLK0_LCPLLGTREFCLK0 HSCLK1_LCPLLGTREFCLK1
3'b010	2	HSCLK0_LCPLLGTREFCLK1 HSCLK1_LCPLLGTREFCLK0
3'b011	3	HSCLK[0/1]_LCPLLNORTHREFCLK0
3'b100	4	HSCLK[0/1]_LCPLLNORTHREFCLK1
3'b101	5	HSCLK[0/1]_LCPLLSOUTHREFCLK0
3'b110	6	HSCLK[0/1]_LCPLLSOUTHREFCLK1
3'b111	7	HSCLK[0/1]_LCPLLGTGREFCLK

Notes:

1. When HS[0/1]_LCPLL_IPS_PIN_EN = 0, HS[0/1]_LCPLL_IPS_REFCLK_SEL control the multiplexer inputs.
2. When HS[0/1]_LCPLL_IPS_PIN_EN = 1, HSCLK[0/1]_LCPLLREFCLKSEL controls the multiplexer inputs.
3. The HSCLK[0/1]_LCPLLGTGREFCLK input is reserved and should not be used.

Table 11: **RPLL Reference Clock Selection Multiplexer Settings**

HSCLK[0/1]_RPLLREFCLK	HS[0/1]_RPLL_IPS_REFCLK_SEL	Selected Input to RPLL
3'b001	1	HSCLK0_RPLLGTREFCLK0 HSCLK1_RPLLGTREFCLK1
3'b010	2	HSCLK0_RPLLGTREFCLK1 HSCLK1_RPLLGTREFCLK0
3'b011	3	HSCLK[0/1]_RPLLNORTHREFCLK0
3'b100	4	HSCLK[0/1]_RPLLNORTHREFCLK1
3'b101	5	HSCLK[0/1]_RPLLSOUTHREFCLK0
3'b110	6	HSCLK[0/1]_RPLLSOUTHREFCLK1
3'b111	7	HSCLK[0/1]_RPLLGTGREFCLK

Notes:

1. When HS[0/1]_RPLL_IPS_PIN_EN = 0, HS[0/1]_RPLL_IPS_REFCLK_SEL controls the multiplexer inputs.
2. When HS[0/1]_RPLL_IPS_PIN_EN = 1, HSCLK[0/1]_RPLLREFCLK controls the multiplexer inputs.
3. The HSCLK[0/1]_RPLLGTGREFCLK input is reserved and should not be used.

As shown in Figure 6 and Figure 7, the GTY transceivers in Versal ACAPs contain multiple dedicated reference clock input ports for each PLL. The user need to set the muxing of the reference clock input by using the selector port or the selector attribute. The following table lists the reference clock multiplexer selector control signals per PLL.

Table 12: **Reference Clock Input Multiplexer Selection Control**

PLL	Selector Control Attribute	Multiplexer Selector Port/Attribute
RPLL (HSCLK0)	HS0_RPLL_IPS_PIN_EN = 1'b1	HSCLK0_RPLLREFCLKSEL[2:0]
	HS0_RPLL_IPS_PIN_EN = 1'b0	HS0_RPLL_IPS_REFCLK_SEL
RPLL (HSCLK1)	HS1_RPLL_IPS_PIN_EN = 1'b1	HSCLK1_RPLLREFCLKSEL[2:0]
	HS1_RPLL_IPS_PIN_EN = 1'b0	HS1_RPLL_IPS_REFCLK_SEL
LCPLL (HSCLK0)	HS0_LCPLL_IPS_PIN_EN = 1'b1	HSCLK0_LCPLLREFCLKSEL[2:0]
	HS0_LCPLL_IPS_PIN_EN = 1'b0	HS0_LCPLL_IPS_REFCLK_SEL
LCPLL (HSCLK1)	HS1_LCPLL_IPS_PIN_EN = 1'b1	HSCLK1_LCPLLREFCLKSEL[2:0]
	HS1_LCPLL_IPS_PIN_EN = 1'b0	HS1_LCPLL_IPS_REFCLK_SEL

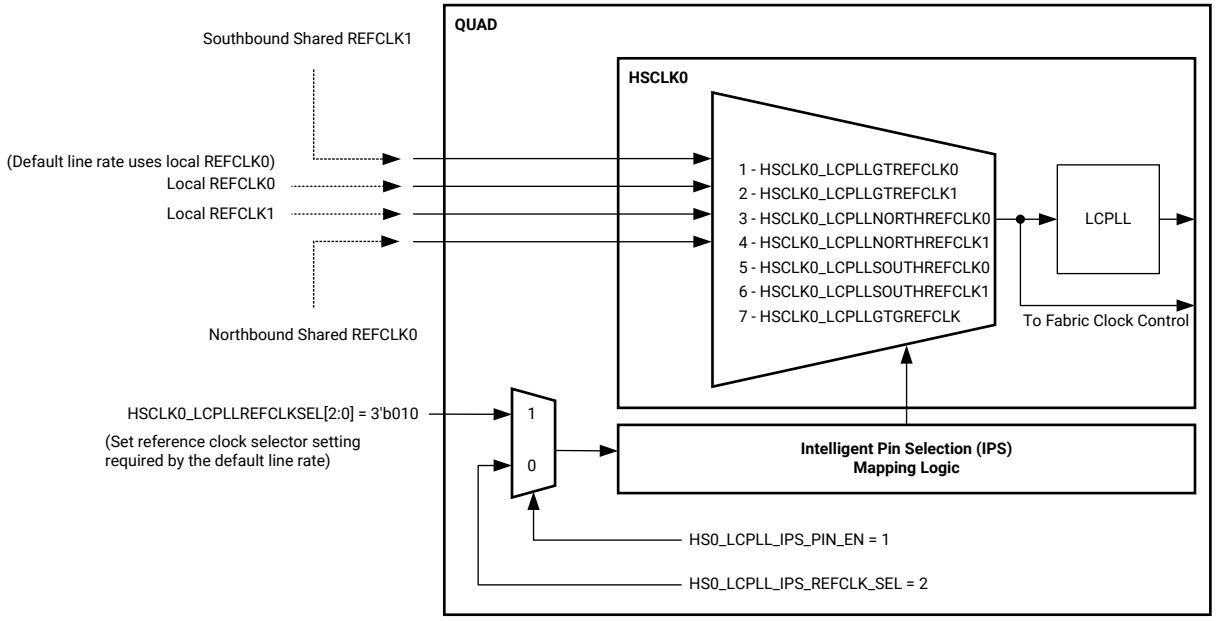
Notes:

1. When HS[0/1]_RPLL_IPS_PIN_EN = 0, HS[0/1]_RPLL_IPS_REFCLK_SEL controls the multiplexer inputs.
2. When HS[0/1]_RPLL_IPS_PIN_EN = 1, HSCLK[0/1]_RPLLREFCLK controls the multiplexer inputs.
3. The HSCLK[0/1]_RPLLGTGREFCLK and HSCLK[0/1]_LCPLLGTGREFCLK inputs are reserved and should not be used.

Intelligent Pin Selection

In Versal devices, the intelligent pin selection (IPS) functionality provides mapping logic during implementation that attempts to route the reference clock to the desired PLL. The IPS provides a layer of mapping logic that assists the user to simplify design creation when multiple reference clocks are connected to any of the input ports on the desired PLL. This is shown in the example figure below.

Figure 8: **IPS Example**



The recommended procedure is to connect each reference clock input from top to bottom on the input multiplexer without leaving any gaps, as shown in the example figure above. IPS automatically provides the necessary mapping during implementation to make sure each of the reference clocks is correcty connected on the PLL input multiplexer.

The list below contains other restrictions or design practices that the user should follow:

- The design must connect the reference clock to the intended PLL.
- The user must configure the reference clock input multiplexer setting to match the reference clock input port that is required by the default line rate.
- The design must not instantiate IBUFDS_GTE5 at a location that violates the reference clock sharing rules. For more information, refer to Reference Clock Selection and Distribution.
- HSCLK*_LCPLLGTREFCLK and HSCLK*_RPLLGTREFCLK are reserved inputs and should not be used.

Ports and Attributes

The following tables define the clocking ports and attributes relevant to reference clock selection and distribution for GTYE5_QUAD primitives.

Table 13: **Reference Clock Selection and Distribution Ports**

Port	Direction	Clock Domain	Description
HSCLK[0/1]_LCPLLGTREFCLK0	Input	CLOCK	Local GTREFCLK0 to LCPLL in HSCLK0/1.
HSCLK[0/1]_LCPLLGTREFCLK1	Input	CLOCK	Local GTREFCLK1 to LCPLL in HSCLK0/1.
HSCLK[0/1]_LCPLLNORTHREFCLK0	Input	CLOCK	North-bound GTREFCLK0 clock from the Quad below.
HSCLK[0/1]_LCPLLNORTHREFCLK1	Input	CLOCK	North-bound GTREFCLK1 clock from the Quad below.
HSCLK[0/1]_LCPLLREFCLKSEL[2:0]	Input	ASYNC	Input to dynamically select the input reference clock to the LCPLL. Set this input to 3'b001 when only one clock source is connected to the LCPLL reference clock selection multiplexer. 3'b001: HSCLK0_LCPLLGTREFCLK0/ HSCLK1_LCPLLGTREFCLK1 3'b010: HSCLK0_LCPLLGTREFCLK1/ HSCLK1_LCPLLGTREFCLK0 3'b011: HSCLK[0/1]_LCPLLNORTHREFCLK0 3'b100: HSCLK[0/1]_LCPLLNORTHREFCLK1 3'b101: HSCLK[0/1]_LCPLLSOUTHREFCLK0 3'b110: HSCLK[0/1]_LCPLLSOUTHREFCLK1 3'b111: HSCLK[0/1]_LCPLLGTGREFCLK Reset must be applied to the LCPLL after changing the reference clock input.
HSCLK[0/1]_LCPLLSOUTHREFCLK0	Input	CLOCK	South-bound GTREFCLK0 clock from the Quad above.
HSCLK[0/1]_LCPLLSOUTHREFCLK1	Input	CLOCK	South-bound GTREFCLK1 clock from the Quad above.
HSCLK[0/1]_RPLLGTREFCLK0	Input	CLOCK	Local GTREFCLK0 to RPLL in HSCLK0/1.
HSCLK[0/1]_RPLLGTREFCLK1	Input	CLOCK	Local GTREFCLK1 to RPLL in HSCLK0/1.
HSCLK[0/1]_RPLLNORTHREFCLK0	Input	CLOCK	North-bound GTREFCLK0 clock from the Quad below.
HSCLK[0/1]_RPLLNORTHREFCLK1	Input	CLOCK	North-bound GTREFCLK1 clock from the Quad below.

Table 13: **Reference Clock Selection and Distribution Ports** *(cont'd)*

Port	Direction	Clock Domain	Description
HSCLK[0/1]_RPLLREFCLKSEL[2:0]	Input	ASYNC	Input to dynamically select the input reference clock to the RPLL. Set this input to 3'b001 when only one clock source is connected to the RPLL reference clock selection multiplexer. 3'b001: HSCLK0_RPLLGTREFCLK0/ HSCLK1_RPLLGTREFCLK1 3'b010: HSCLK0_RPLLGTREFCLK1/ HSCLK1_RPLLGTREFCLK0 3'b011: HSCLK[0/1]_RPLLNORTHREFCLK0 3'b100: HSCLK[0/1]_RPLLNORTHREFCLK1 3'b101: HSCLK[0/1]_RPLLSOUTHREFCLK0 3'b110: HSCLK[0/1]_RPLLSOUTHREFCLK1 3'b111: HSCLK[0/1]_RPLLGTGREFCLK Reset must be applied to the RPLL after changing the reference clock input.
HSCLK[0/1]_RPLLSOUTHREFCLK0	Input	CLOCK	South-bound GTREFCLK0 clock from the Quad above.
HSCLK[0/1]_RPLLSOUTHREFCLK1	Input	CLOCK	South-bound GTREFCLK1 clock from the Quad above.

Table 14: **Reference Clock Selection and Distribution Attributes**

Reference Clock Selection and Distribution Attributes		
Attribute	**Address**	
HS0_LCPLL_IPS_PIN_EN		0x0C12
Label	**Bit Field**	**Description**
HS0_LCPLL_IPS_PIN_EN	[27:27]	Determines whether the reference clock input to the LCPLL in block HSCLK0 is controlled by port HSCLK0_LCPLLREFCLKSEL or attribute HS0_LCPLL_IPS_REFCLK_SEL 0:HS0_LCPLL_IPS_REFCLK_SEL takes control 1:HSCLK0_LCPLLREFCLKSEL takes control
Attribute	**Address**	
HS0_LCPLL_IPS_REFCLK_SEL		0x0C12
Label	**Bit Field**	**Description**
HS0_LCPLL_IPS_REFCLK_SEL	[26:24]	Control selection of reference clock input to LCPLL in block HSCLK0 1:HSCLK0_LCPLLGTREFCLK0 2:HSCLK0_LCPLLGTREFCLK1 3:HSCLK0_LCPLLNORTHREFCLK0 4:HSCLK0_LCPLLNORTHREFCLK1 5:HSCLK0_LCPLLSOUTHREFCLK0 6:HSCLK0_LCPLLSOUTHREFCLK1 7:HSCLK0_LCPLLGTGREFCLK
Attribute	**Address**	
HS0_RPLL_IPS_PIN_EN		0x0C13

Table 14: **Reference Clock Selection and Distribution Attributes** *(cont'd)*

Reference Clock Selection and Distribution Attributes		
Label	**Bit Field**	**Description**
HS0_RPLL_IPS_PIN_EN	[27:27]	Determines whether the reference clock input to the RPLL in block HSCLK0 is controlled by port HSCLK0_RPLLREFCLKSEL or attribute HS0_RPLL_IPS_REFCLK_SEL 0:HS0_RPLL_IPS_REFCLK_SEL takes control 1:HSCLK0_RPLLREFCLKSEL takes control
Attribute	**Address**	
HS0_RPLL_IPS_REFCLK_SEL		0x0C13
Label	**Bit Field**	**Description**
HS0_RPLL_IPS_REFCLK_SEL	[26:24]	Control selection of reference clock input to RPLL in block HSCLK0 1:HSCLK0_RPLLGTREFCLK0 2:HSCLK0_RPLLGTREFCLK1 3:HSCLK0_RPLLNORTHREFCLK0 4:HSCLK0_RPLLNORTHREFCLK1 5:HSCLK0_RPLLSOUTHREFCLK0 6:HSCLK0_RPLLSOUTHREFCLK1 7:HSCLK0_RPLLGTGREFCLK
Attribute	**Address**	
HS1_LCPLL_IPS_PIN_EN		0x0F12
Label	**Bit Field**	**Description**
HS1_LCPLL_IPS_PIN_EN	[27:27]	Determines whether the reference clock input to the LCPLL in block HSCLK1 is controlled by port HSCLK1_LCPLLREFCLKSEL or attribute HS1_LCPLL_IPS_REFCLK_SEL 0:HS1_LCPLL_IPS_REFCLK_SEL takes control 1:HSCLK1_LCPLLREFCLKSEL takes control
Attribute	**Address**	
HS1_LCPLL_IPS_REFCLK_SEL		0x0F12
Label	**Bit Field**	**Description**
HS1_LCPLL_IPS_REFCLK_SEL	[26:24]	Control selection of reference clock input to LCPLL in block HSCLK1 1:HSCLK1_LCPLLGTREFCLK1 2:HSCLK1_LCPLLGTREFCLK0 3:HSCLK1_LCPLLNORTHREFCLK0 4:HSCLK1_LCPLLNORTHREFCLK1 5:HSCLK1_LCPLLSOUTHREFCLK0 6:HSCLK1_LCPLLSOUTHREFCLK1 7:HSCLK1_LCPLLGTGREFCLK
Attribute	**Address**	
HS1_RPLL_IPS_PIN_EN		0x0F13

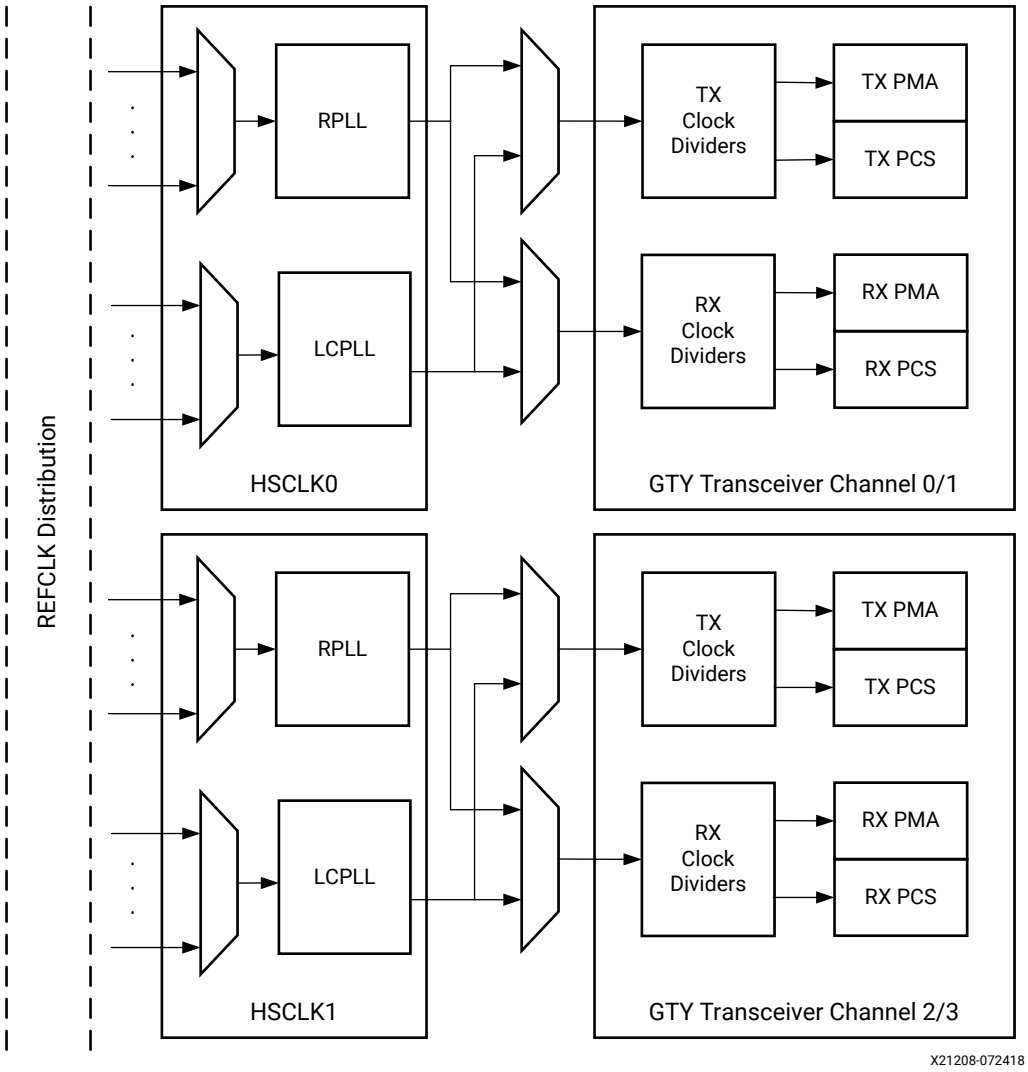
Table 14: **Reference Clock Selection and Distribution Attributes** *(cont'd)*

Reference Clock Selection and Distribution Attributes		
Label	**Bit Field**	**Description**
HS1_RPLL_IPS_PIN_EN	[27:27]	Determines whether the reference clock input to the RPLL in block HSCLK1 is controlled by port HSCLK1_RPLLREFCLKSEL or attribute HS1_RPLL_IPS_REFCLK_SEL 0:HS1_RPLL_IPS_REFCLK_SEL takes control 1:HSCLK1_RPLLREFCLKSEL takes control
Attribute	**Address**	
HS1_RPLL_IPS_REFCLK_SEL		0x0F13
Label	**Bit Field**	**Description**
HS1_RPLL_IPS_REFCLK_SEL	[26:24]	Control selection of reference clock input to RPLL in block HSCLK1 1:HSCLK1_RPLLGTREFCLK1 2:HSCLK1_RPLLGTREFCLK0 3:HSCLK1_RPLLNORTHREFCLK0 4:HSCLK1_RPLLNORTHREFCLK1 5:HSCLK1_RPLLSOUTHREFCLK0 6:HSCLK1_RPLLSOUTHREFCLK1 7:HSCLK1_RPLLGTGREFCLK

Ring PLL

Each GTY transceiver Quad contains two ring-based channel PLLs (RPLL), one RPLL in each of the HSCLK0/1 block. The internal channel clocking architecture is shown in the figure below. The TX and RX clock dividers can individually select the clock from the RPLL or LCPLL assigned to that particular channel to allow the TX and RX datapaths to operate at asynchronous frequencies using different reference clock inputs.

Figure 9: **Channel Clocking Architecture**



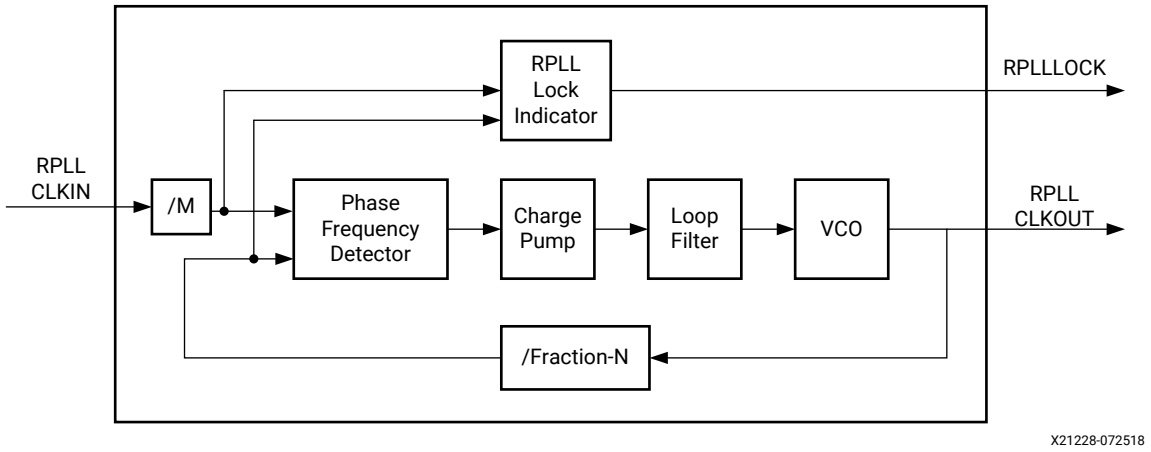
The RPLL input clock selection is described in Reference Clock Selection and Distribution. The RPLL outputs feed the TX and RX clock divider blocks, which control the generation of serial and parallel clocks used by the PMA and PCS blocks. The RPLL can be shared between the TX and RX datapaths if they operate at line rates that are integral multiples of the same VCO frequency, with the limitation that the RPLL from HSCLK0 can only drive channel 0/1 and RPLL from HSCLK1 can only drive channel 2/3.

The figure below illustrates a conceptual view of the RPLL architecture. The input clock is divided by a factor of M before feeding into the phase frequency detector. The feedback divider, N, determines the VCO multiplication ratio and the RPLL output frequency. When the fractional feature of the feedback divider N is enabled, its effective ratio becomes a combination of the N factor plus a fractional part.

IMPORTANT! *The factional-N feature of the RPLL is designed specifically to provide a clocking source and when enabled, the output of the RPLL must not be used to drive the actual datapath (PMA or PCS) of the transceiver channel.*

A number of lock indicators are generated, and the RPLL lock compares the frequencies of the reference clock and the VCO feedback clock to determine if a frequency lock has been achieved.

Figure 10: **RPLL Block Diagram**



The RPLL has a nominal VCO operating range between 4.0 GHz to 8.0 GHz. For additional information regarding the exact RPLL operating range for different device speed grades, refer to the Versal ACAP data sheets. The Versal ACAPs Transceivers Wizard chooses the appropriate RPLL settings based on application requirements.

The following equation shows how to determine the RPLL output frequency (GHz).

Equation 1: **RPLL Output Frequency**

$$f_{PLLCLKOUT} = f_{PLLCLKIN} \times \frac{N.FractionalPart}{M}$$

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

Equation 2: **Line Rate**

$$f_{Linerate} = \frac{f_{PLLCLKOUT} \times 2}{D}$$

The following equation shows how to determine the fractional part of the feedback divider presented in Equation 1: RPLL Output Frequency.

Equation 3: **Fraction Part**

$$FractionalPart = \frac{SDMDATA}{2^{SDMWIDTH}}$$

The following table lists the allowable divider settings for RPLL.

Table 15: **RPLL Divider Settings**

Factor	Attribute/Port	Valid Settings
M	RPLL_REFDIV	Valid settings are 1, 2, 3, 4.
N.FractionalPart	A_HS0_RPLLFBDIV A_HS1_RPLLFBDIV	The valid divider range from the integer N depends on whether or not the fractional component is enabled, and also whether the RPLL is used for clock generation or driving the transceiver datapath. Fractional component disabled: 5–25, 80 Fractional component enabled or fabric clocking: 8–80 When the fractional part is used or if the RPLL output will not drive the datapath, the RPLL is used as a clocking source where the output can be routed to the fabric only. It cannot be used to drive the transceiver datapath (PCS or PMA) in this use model.
D	RXOUT_DIV TXOUT_DIV	1, 2, 2.5[1], 4, 8, 16
SDMDATA	HSCLK[0/1]_RPLLSDMDATA	0 – (2^{24} – 1)
SDMWIDTH	SDM_WIDTHSEL (HSCLK*_RPLL_LGC_CFG1)	16, 20, 24

Notes:

1. 2.5 is only for line rate switching between 10.3125G and 25.78125G.

Ports and Attributes

The following tables define the ports and attributes for the RPLL.

Table 16: **RPLL Ports**

Port	Direction	Clock Domain	Description
HSCLK[0/1]_RPLLGTREFCLK0	Input	CLOCK	Local GTREFCLK0 to RPLL in HSCLK0/1.
HSCLK[0/1]_RPLLGTREFCLK1	Input	CLOCK	Local GTREFCLK1 to RPLL in HSCLK0/1.
HSCLK[0/1]_RPLLLOCK	Output	ASYNC	Active-High RPLL frequency lock signal indicates that the RPLL frequency is within a predetermined tolerance. The GTY transceiver and its clock outputs are not reliable until this condition is met.
HSCLK[0/1]_RPLLNORTHREFCLK0	Input	CLOCK	North-bound GTREFCLK0 clock from the Quad below.
HSCLK[0/1]_RPLLNORTHREFCLK1	Input	CLOCK	North-bound GTREFCLK1 clock from the Quad below.

Table 16: **RPLL Ports** *(cont'd)*

Port	Direction	Clock Domain	Description
HSCLK[0/1]_RPLLREFCLKSEL[2:0]	Input	ASYNC	Input to dynamically select the input reference clock to the RPLL. Set this input to 3'b001 when only one clock source is connected to the RPLL reference clock selection multiplexer. 3'b001: HSCLK0_RPLLGTREFCLK0/ HSCLK1_RPLLGTREFCLK1 3'b010: HSCLK0_RPLLGTREFCLK1/ HSCLK1_RPLLGTREFCLK0 3'b011: HSCLK[0/1]_RPLLNORTHREFCLK0 3'b100: HSCLK[0/1]_RPLLNORTHREFCLK1 3'b101: HSCLK[0/1]_RPLLSOUTHREFCLK0 3'b110: HSCLK[0/1]_RPLLSOUTHREFCLK1 3'b111: HSCLK[0/1]_RPLLGTGREFCLK Reset must be applied to the RPLL after changing the reference clock input.
HSCLK[0/1]_RPLLSDMDATA[25:0]	Input	ASYNC	Input to set the numerator of the fractional part of the RPLL feedback divider.

Table 17: **RPLL Attributes**

RPLL Attributes		
Attribute	**Address**	
A_CFG1		0x0E12
Label	**Bit Field**	**Description**
A_HS1_RPLLFBDIV	[31:24]	Feedback divider N settings for RPLL in HSCLK1. Valid settings depends on whether or not the fractional component is enabled. Fractional component disabled: 5-25, 80 Fractional component enabled: 8 - 80
A_HS0_RPLLFBDIV	[15:8]	Feedback divider N settings for RPLL in HSCLK0. Valid settings depends on whether or not the fractional component is enabled. Fractional component disabled: 5-25, 80 Fractional component enabled: 8 - 80
Attribute	**Address**	
CH0_PIPE_CTRL_CFG3		0x0CA0
CH1_PIPE_CTRL_CFG3		0x0DA0
CH2_PIPE_CTRL_CFG3		0x0EA0
CH3_PIPE_CTRL_CFG3		0x0FA0

Table 17: **RPLL Attributes** *(cont'd)*

RPLL Attributes		
Label	**Bit Field**	**Description**
RXOUT_DIV	[28:26]	This attribute selects the setting for the RX serial clock divider D. 3'b000: /1 3'b001: /2 3'b010: /4 3'b011: /8 3'b100: /16 3'b111: /2.5
TXOUT_DIV	[25:23]	This attribute selects the setting for the TX serial clock divider D. 3'b000: /1 3'b001: /2 3'b010: /4 3'b011: /8 3'b100: /16 3'b111: /2.5
Attribute	**Address**	
HSCLK0_RPLL_CFG0	0x0D0F	
HSCLK1_RPLL_CFG0	0x0F0F	
Label	**Bit Field**	**Description**
RPLL_REFDIV	[4:0]	RPLL reference clock divider M settings. Valid settings are 1, 2, 3, and 4. 00000: 2 00001: 3 00010: 4 00100: 1
Attribute	**Address**	
HSCLK0_RPLL_LGC_CFG0	0x0D0C	
HSCLK1_RPLL_LGC_CFG0	0x0F0C	
Label	**Bit Field**	**Description**
RPLLLOCKEN	[11:11]	Active-High signal enables the RPLL lock detector.
Attribute	**Address**	
HSCLK0_RPLL_LGC_CFG1	0x0D0D	
HSCLK1_RPLL_LGC_CFG1	0x0F0D	
Label	**Bit Field**	**Description**
SDM_WIDTHSEL	[10:9]	This attribute set the denominator of the fractional part of the feedback divider. 00: 24 01: 20 10: 16

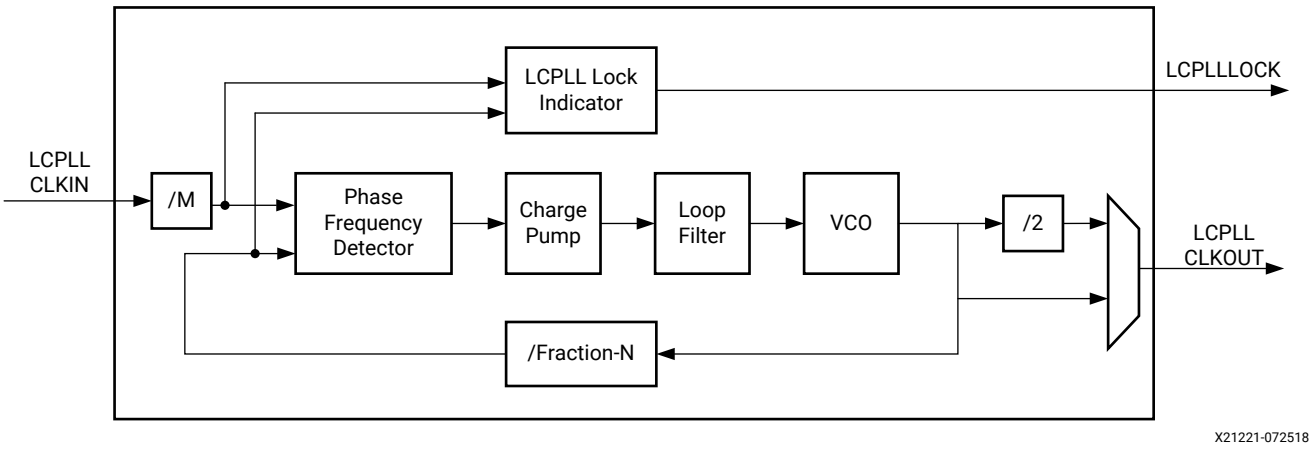
LC-Tank PLL

Each Quad contains two LC-based PLLs, one per HSCLK0/1 block, that are referred to as LCPLLs. Use of an LCPLL is required when operating the channels at line rates above the RPLL operating range. The GTYE5_QUAD primitive encapsulates both the LCPLLs and must be instantiated when either LCPLL is used.

The LCPLL input reference clock selection is described in Reference Clock Selection and Distribution. The LCPLL outputs feed the TX and RX clock divider blocks of the serial transceiver channels within the same Quad, which control the generation of serial and parallel clocks used by the PMA and PCS blocks. The LCPLL from HSCLK0 can only drive channel 0/1, and the LCPLL from HSCLK1 can only drive channel 2/3. When the channel is operating above 16.375 Gb/s, the LCPLL from HSCLK0 must use GTREFCLK0, and the LCPLL from HSCLK1 must use GTREFCLK1.

The following figure illustrates a conceptual view of the LCPLL architecture. The input clock can be divided by a factor of M before it is fed into the phase frequency detector. The feedback divider N determines the VCO multiplication ratio. For line rates below 28.1 Gb/s, a fractional-N divider is supported where the effective ratio is a combination of the N factor plus a fractional part. The LCPLL output frequency depends on the setting of LCPLLCLKOUT_RATE. When LCPLLCLKOUT_RATE is set to HALF, the output frequency is half of the VCO frequency. When it is set to FULL, the output frequency is the same as the VCO 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 11: **LCPLL Block Diagram**



The LCPLL has a nominal operating range between 8.0 GHz to 16.375 GHz. For additional information regarding the exact LCPLL operating range for different device speed grades, refer to the Versal ACAP data sheets. The Versal ACAPs Transceivers Wizard chooses the appropriate LCPLL settings based on application requirements.

The following equation shows how to determine the LCPLL output frequency (GHz). For line rates above 28.1 Gb/s, the fractional part is bypassed.

Equation 4: **LCPLL Output Frequency**

$$f_{PLLCLKOUT} = f_{PLLCLKIN} \times \frac{\text{N.FractionalPart}}{M \times LCPLLCLKOUT\_RATE}$$

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

Equation 5: **Line Rate**

$$f_{Linerate} = \frac{f_{PLLCLKOUT} \times 2}{D}$$

The following equation shows how to determine the fractional part of the feedback divider presented in Equation 4: LCPLL Output Frequency.

Equation 6: **Fractional Part**

$$FractionalPart = \frac{SDMDATA}{2^{SDMWIDTH}}$$

The following table lists the allowable divider values.

Table 18: **LCPLL Divider Settings**

Factor	Attribute/Port	Valid Settings
M	LCPLL_PREDIV	1, 2, 3, 4
N.FractionalPart	A_HS0_LCPLLFBDIV A_HS1_LCPLLFBDIV	The valid divider range depends on whether or not the fractional component is enabled. 1. Fractional component disabled: 13 – 160 2. Fractional component enabled: 16 – 160
D	RXOUT_DIV TXOUT_DIV	1, 2, 2.5[1], 4, 8, 16
LCPLLCLKOUT_RATE	HSDIST_DIV2SEL	Full, Half
SDMDATA	HSCLK[0/1]_LCPLLSDMDATA	0 – (2^{24} – 1)
SDMWIDTH	SDM_WIDTHSEL (HSCLK*_LCPLL_LGC_CFG1)	16, 20, 24

Notes:

1. 2.5 is only for line rate switching between 10.3125G and 25.78125G.

Ports and Attributes

The following tables define the pins and attributes for the LCPLL.

Table 19: **LCPLL Ports**

Port	Direction	Clock Domain	Description
HSCLK[0/1]_LCPLLGTREFCLK0	Input	CLOCK	Local GTREFCLK0 to LCPLL in HSCLK0/1.
HSCLK[0/1]_LCPLLGTREFCLK1	Input	CLOCK	Local GTREFCLK1 to LCPLL in HSCLK0/1.
HSCLK[0/1]_LCPLLLOCK	Output	ASYNC	Active-High LCPLL frequency lock signal indicates that the LCPLL frequency is within a predetermined tolerance. The GTY transceiver and its clock outputs are not reliable until this condition is met.
HSCLK[0/1]_LCPLLNORTHREFCLK0	Input	CLOCK	North-bound GTREFCLK0 clock from the Quad below.
HSCLK[0/1]_LCPLLNORTHREFCLK1	Input	CLOCK	North-bound GTREFCLK1 clock from the Quad below.
HSCLK[0/1]_LCPLLPD	Input	ASYNC	This active-High signal powers down the LCPLL.
HSCLK[0/1]_LCPLLREFCLKSEL[2:0]	Input	ASYNC	Input to dynamically select the input reference clock to the LCPLL. Set this input to 3'b001 when only one clock source is connected to the LCPLL reference clock selection multiplexer. 3'b001: HSCLK0_LCPLLGTREFCLK0/ HSCLK1_LCPLLGTREFCLK1 3'b010: HSCLK0_LCPLLGTREFCLK1/ HSCLK1_LCPLLGTREFCLK0 3'b011: HSCLK[0/1]_LCPLLNORTHREFCLK0 3'b100: HSCLK[0/1]_LCPLLNORTHREFCLK1 3'b101: HSCLK[0/1]_LCPLLSOUTHREFCLK0 3'b110: HSCLK[0/1]_LCPLLSOUTHREFCLK1 3'b111: HSCLK[0/1]_LCPLLGTGREFCLK Reset must be applied to the LCPLL after changing the reference clock input.
HSCLK[0/1]_LCPLLRESET	Input	ASYNC	Active high signal that resets the LCPLL.
HSCLK[0/1]_LCPLLSDMDATA[25:0]	Input	ASYNC	Input to set the numerator of the fractional part of the LCPLL feedback divider.
HSCLK[0/1]_LCPLLSOUTHREFCLK0	Input	CLOCK	South-bound GTREFCLK0 clock from the Quad above.
HSCLK[0/1]_LCPLLSOUTHREFCLK1	Input	CLOCK	South-bound GTREFCLK1 clock from the Quad above.

Table 20: **LCPLL Attributes**

LCPLL Attributes		
Attribute	**Address**	
A_CFG1		0x0E12
Label	**Bit Field**	**Description**
A_HS1_LCPLLFBDIV	[23:16]	Feedback divider N settings for LCPLL in HSCLK1. Valid settings depends on whether or not the fractional component is enabled. Fractional component disabled: 13-160 Fractional component enabled: 16-160

Table 20: **LCPLL Attributes** *(cont'd)*

LCPLL Attributes		
A_HS0_LCPLLFBDIV	[7:0]	Feedback divider N settings for LCPLL in HSCLK0. Valid settings depends on whether or not the fractional component is enabled. Fractional component disabled: 13-160 Fractional component enabled: 16-160
Attribute	**Address**	
HSCLK0_HSDIST_CFG		0x0CBF
HSCLK1_HSDIST_CFG		0x0FBF
Label	**Bit Field**	**Description**
HSDIST_DIV2SEL	[16:16]	Sets the LCPLLCLKOUT_RATE factor either to provide full LCPLL VCO frequency, or half of LCPLL VCO frequency at the output: 0: Full rate 1: Half rate
Attribute	**Address**	
HSCLK0_LCPLL_CFG0		0x0C0F
HSCLK1_LCPLL_CFG0		0x0E0F
Label	**Bit Field**	**Description**
LCPLL_PREDIV	[24:20]	LCPLL reference clock divider M settings. Valid settings are 1, 2, 3, and 4. 00000: 2 00001: 3 00010: 4 00100: 1
Attribute	**Address**	
HSCLK0_LCPLL_LGC_CFG0		0x0C0C
HSCLK1_LCPLL_LGC_CFG0		0x0E0C
Label	**Bit Field**	**Description**
LCPLLLOCKEN	[11:11]	Active-High signal enables the LCPLL lock detector.
Attribute	**Address**	
HSCLK0_LCPLL_LGC_CFG1		0x0C0D
HSCLK1_LCPLL_LGC_CFG1		0x0E0D
Label	**Bit Field**	**Description**
SDM_WIDTHSEL	[10:9]	This attribute set the denominator of the fractional part of the feedback divider. 00: 24 01: 20 10: 16
Attribute	**Address**	
CH0_PIPE_CTRL_CFG3		0x0CA0
CH1_PIPE_CTRL_CFG3		0x0DA0

Table 20: **LCPLL Attributes** *(cont'd)*

LCPLL Attributes		
CH2_PIPE_CTRL_CFG3		0x0EA0
CH3_PIPE_CTRL_CFG3		0x0FA0
Label	**Bit Field**	**Description**
RXOUT_DIV	[28:26]	This attribute selects the setting for the RX serial clock divider D. 3'b000: /1 3'b001: /2 3'b010: /4 3'b011: /8 3'b100: /16 3'b111: /2.5
TXOUT_DIV	[25:23]	This attribute selects the setting for the TX serial clock divider D. 3'b000: /1 3'b001: /2 3'b010: /4 3'b011: /8 3'b100: /16 3'b111: /2.5

Reset and Initialization

The transceiver must be initialized after device power-up and configuration before it can be used. The transmitter (TX) and receiver (RX) can be initialized independently and in parallel as shown in the following figure. Transceiver TX and RX initialization comprises three steps:

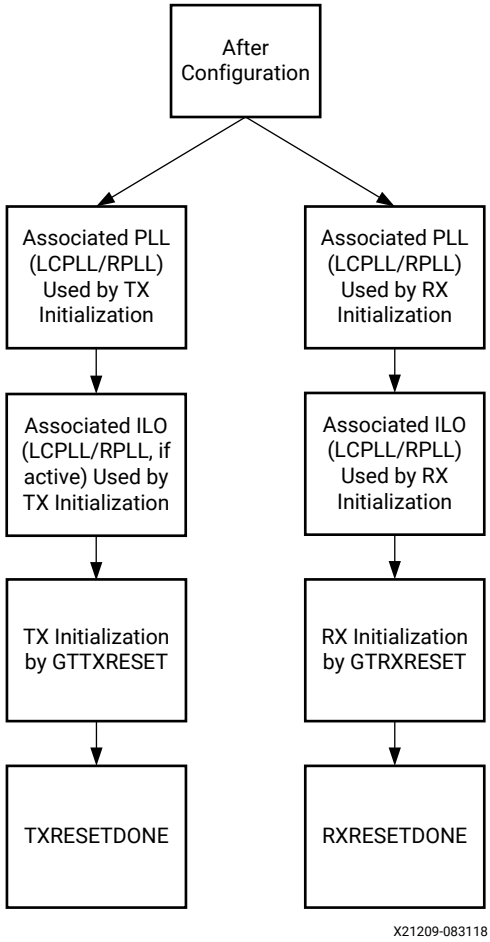
1. Initializing the associated PLL driving TX/RX
2. Initializing the associated ILO
3. Initializing the TX and RX datapaths (PMA, DAPI, Buffer Bypass, PCS)

Transceiver TX and RX can receive a clock from either the LCPLL or the RPLL. The associated PLL (LCPLL/RPLL) used by the TX and RX must be initialized first before ILO, TX and RX initialization. The ILO must be initialized after the associated PLL is locked. Any PLL and ILO used by the TX and RX is reset individually and its reset operation is completely independent from all TX and RX resets. The TX and RX datapaths must be initialized only after the associated PLL and ILO is locked.

For the GTY transceiver, there is a master reset controller available that can run through all reset steps described above automatically. When using the master reset, the associated PLL, ILO, and TX and RX datapaths are reset in the proper sequence by toggling the respective TX and RX master reset signal.

IMPORTANT! *The reference clock to the transceiver must be available and stable prior to start of device programming or start of any manual reset sequence.*

Figure 12: **Transceiver Initialization Overview**



The transceiver TX and RX use a state machine to control the initialization process. They are partitioned into a few reset regions. The partition allows the reset state machine to control the reset process in a sequence that the PMA can be reset first and the PCS can be reset after the assertion of the TXUSERRDY or RXUSERRDY. It also allows the PMA, the PCS, and functional blocks inside them to be reset individually when needed during normal operation.

The transceiver offers two types of reset: initialization and component.

- Initialization Reset: This reset is used for complete transceiver initialization. It must be used after device power-up and configuration. During normal operation, when necessary, GTTXRESET and GTRXRESET can also be used to reinitialize the transceiver TX and RX. GTTXRESET is the initialization reset port for the transceiver TX. GTRXRESET is the initialization reset port for the transceiver RX. During initialization reset, TXRESETMODE and RXRESETMODE should be set to sequential mode. All TX PMA, TX PCS, RX PMA, and RX PCS component resets should be enabled by setting all required component bits of TXPMARESETMASK, TXPCSRESETMASK, RXPMARESETMASK, and RXPCSRESETMASK to High.
- Component Reset: This reset is used for special cases and specific subsection resets while the transceiver is in normal operation. The component that is required to be reset is selected by setting the associated bit within TXPMARESETMASK, TXPCSRESETMASK, RXPMARESETMASK, or RXPCSRESETMASK to High. A TX component reset is triggered by toggling the GTTXRESET port. An RX component reset is triggered by toggling the GTRXRESET port. Separate component reset ports are available. These include EYESCANRESET, RXOOBRESET, RXCDRRESET, and RXPRBSCNTRESET.

All reset ports described in this section initiate the internal reset state machine when driven High. The internal reset state machines are held in the reset state until these same reset ports are driven Low. These resets are all asynchronous. The guideline for the pulse width of these asynchronous resets is one period of the reference clock, unless otherwise noted.

***Note*:** Do not use reset ports for the purpose of power down. For details on proper power-down usage, refer to Power Down.

Resetting Multiple Lanes and Quads

Resetting multiple lanes in a Quad or multiple Quads affects the power supply regulation circuit.

Reset Modes

The transceiver resets can operate in two different modes: sequential mode and single mode.

- Sequential mode: The reset state machine starts with an initialization or component reset input driven High and proceeds through all states after the requested reset states in the reset state machine, as shown in Figure 15 for transceiver TX or Figure 20 for transceiver RX until completion. The completion of sequential mode reset flow is signaled when (TX/RX)RESETDONE transitions from Low to High.
- Single mode: The reset state machine only executes the requested component reset independently for a predetermined time set by its attribute. It does not process any state after the requested state, as shown in Figure 15 for transceiver TX or Figure 20 for transceiver RX. The requested reset can be any component reset to reset the PMA, the PCS, or functional blocks inside them. The completion of a single mode reset is signaled when (TX/RX)RESETDONE transitions from Low to High.

The GTY transceiver initialization reset must use sequential mode. All component resets can be operated in either sequential mode or single mode. The GTY transceiver uses (TX/RX)RESETMODE to select between sequential reset mode and single reset mode. The following table provides configuration details that apply to both the GTY transceiver TX and GTY transceiver RX. Reset modes have no impact on LCPLL and RPLL resets. During normal operation, the GTY transceiver TX or GTY transceiver RX can be reset by applications in either sequential mode or single mode, which provides flexibility to reset a portion of the GTY transceiver. When using either sequential mode or single mode, (TX/RX)RESETMODE must be set to the desired value 50 ns before the assertions of any reset.

Table 21: **Transceiver Reset Modes Operation**

Operation Mode	(TX/RX) RESETMODE
Sequential Mode	`2'b00`
Single Mode	`2'b11`

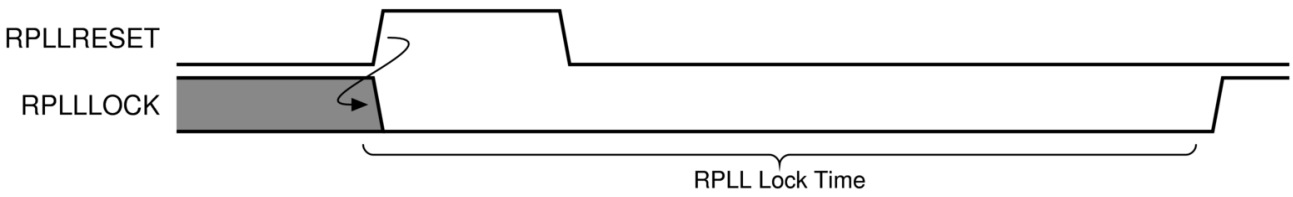
Table 22: **Transceiver Reset Modes Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_RXRESETMODE[1:0]	In	Async	Reset mode port for RX. `2'b00`: Sequential mode (recommended). `2'b01`: Reserved. `2'b10`: Reserved. `2'b11`: Single mode.
CH[0/1/2/3]_TXRESETMODE[1:0]	In	Async	Reset mode port for TX. `2'b00`: Sequential mode (recommended). `2'b01`: Reserved. `2'b10`: Reserved. `2'b11`: Single mode.

RPLL Reset

The RPLL must be reset before it can be used. Each transceiver Quad has dedicated reset ports for each of the two respective RPLLs. As shown in the following figure, RPLLRESET is an input that resets the RPLL. RPLLLOCK is an output that indicates the reset process is done. The guideline for this asynchronous RPLLRESET pulse width is one period of the reference clock. After an RPLLRESET pulse, the internal reset controller generates an internal RPLL reset followed by an internal SDM reset. The time required for the RPLL to lock is affected by a few factors, such as bandwidth setting and clock frequency.

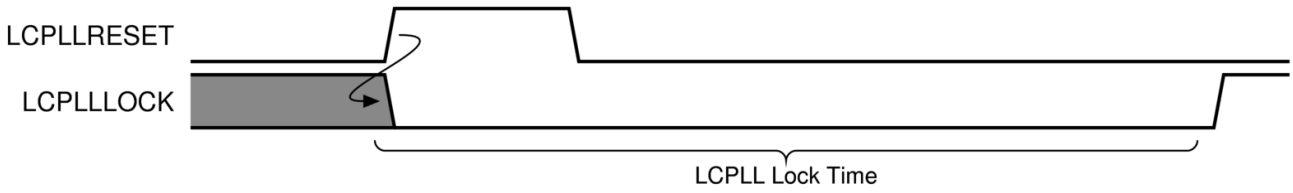
Figure 13: **RPLL Reset Timing Diagram**



LCPLL Reset

The LCPLL must be reset before it can be used. Each transceiver Quad has dedicated reset ports for each of the two respective LCPLLs. As shown in the figure, LCPLLRESET is an input that resets LCPLL. LCPLLLOCK is an output that indicates the reset process is done. The guideline for this asynchronous LCPLLRESET pulse width is one period of the reference clock. After an LCPLLRESET pulse, the internal reset controller generates an internal LCPLL reset followed by an internal SDM reset. The time required for LCPLL to lock is affected by a few factors, such as bandwidth setting and clock frequency.

Figure 14: **LCPLL Reset Timing Diagram**



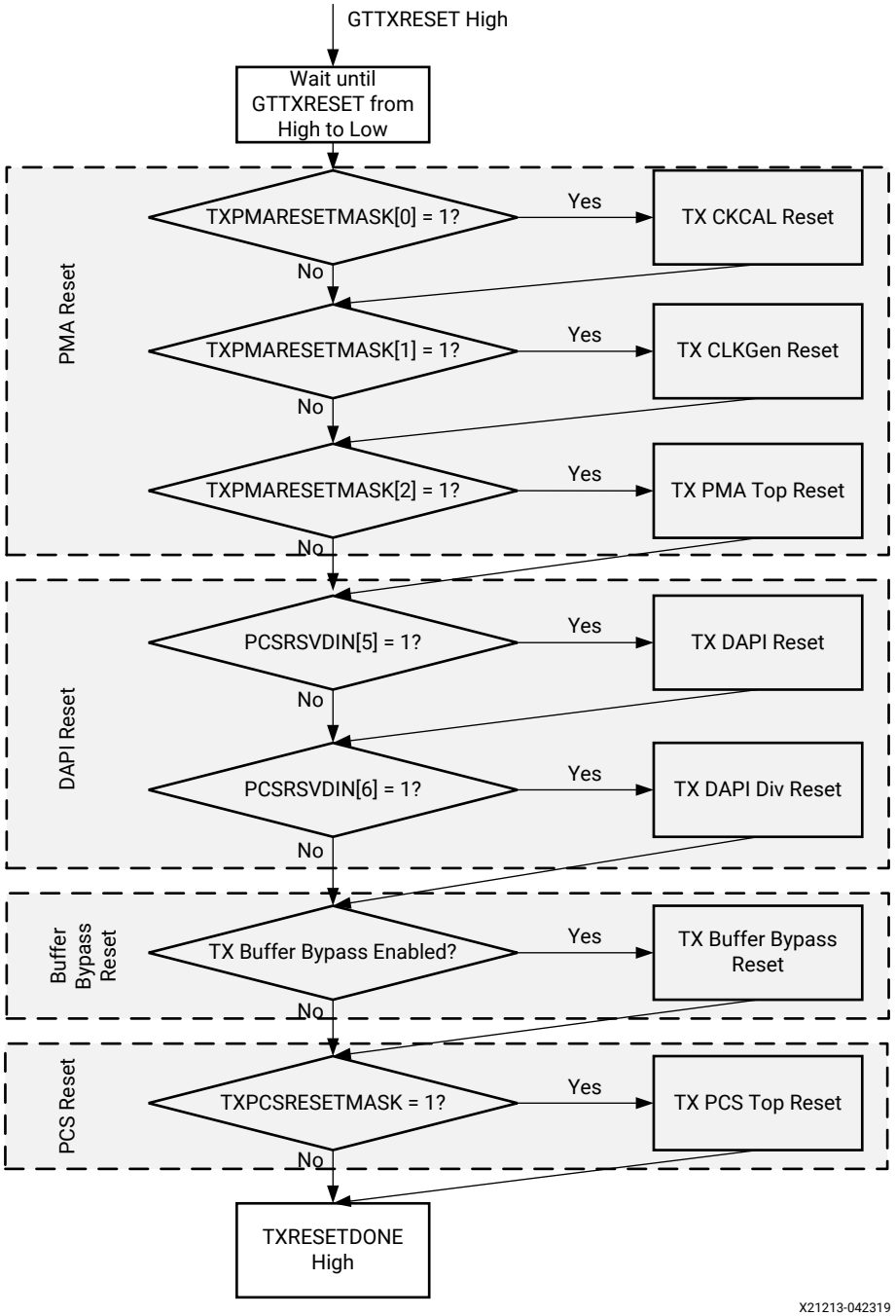
TX Initialization and Reset

The transceiver TX uses a reset state machine to control the reset process. The transceiver TX is partitioned into two reset regions, TX PMA and TX PCS. The partition allows TX initialization and reset to be operated only in sequential mode, as shown in the figure below.

The initializing TX must use GTTXRESET in sequential mode. Activating the GTTXRESET input can automatically trigger a full asynchronous TX reset. The reset state machine executes the reset sequence, as shown in the figure below, covering the whole TX PMA, TX DAPI, TX buffer bypass (if used), and TX PCS. During normal operation, when needed, sequential mode allows you to reset the TX from activating TXPMARESET and continue the reset state machine until TXRESETDONE transitions from Low to High.

The TX reset state machine does not reset the PCS until TXUSERRDY is detected High. Drive TXUSERRDY High after all clocks used by the application including TXUSRCLK are shown as stable.

Figure 15: **Transceiver TX Reset State Machine Sequence**



Transceiver TX Reset in Response to Completion of Configuration

The TX reset sequence shown in TX Initialization and Reset is not automatically started to follow global GSR. It must meet these conditions:

1. TXRESETMODE must be set to sequential mode.
2. GTTXRESET must be used.
3. All TXPMARESETMASK, PCSRSVDIN[6:5] (TXDAPIRESETMASK), and TXPCSRESETMASK bits should be set to High.
4. GTTXRESET cannot be driven Low until the associated PLL and ILO (if ILO is used in the TX) are locked.
5. Ensure that GTPOWERGOOD is High before releasing LC/RPLLRESET, ILORESET (if ILO is used in the TX), and GTTXRESET.

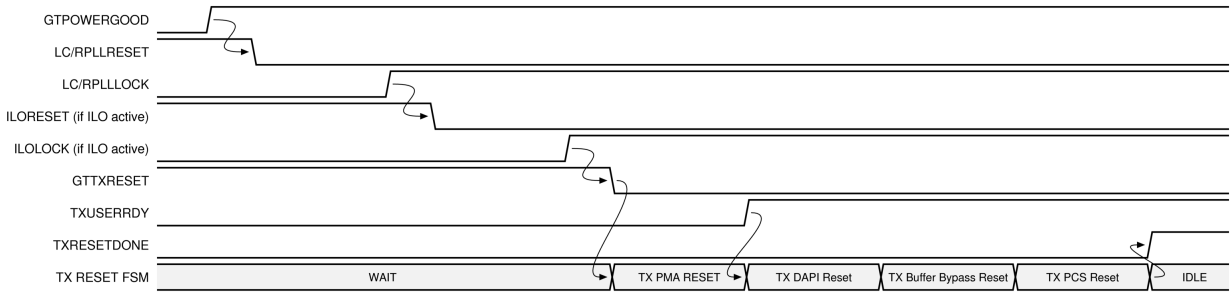
If the reset mode is defaulted to single mode, then you must:

1. Wait another 300–500 ns.
2. Assert LCPLLRESET, ILORESET, RPLLRESET, and GTTXRESET following the reset sequence described in the following figure.

Alternatively, the master reset controller can be used to drive the PLL and TX reset in sequence automatically. Details can be found in Transceiver Master Reset.

RECOMMENDED: *Use the associated PLLLOCK from either LCPLL or RPLL to release GTTXRESET from High to Low as shown in the figure. The TX reset state machine waits when GTTXRESET is detected High and starts the reset sequence until GTTXRESET is released Low.*

Figure 16: **Transmitter Initialization after Configuration**

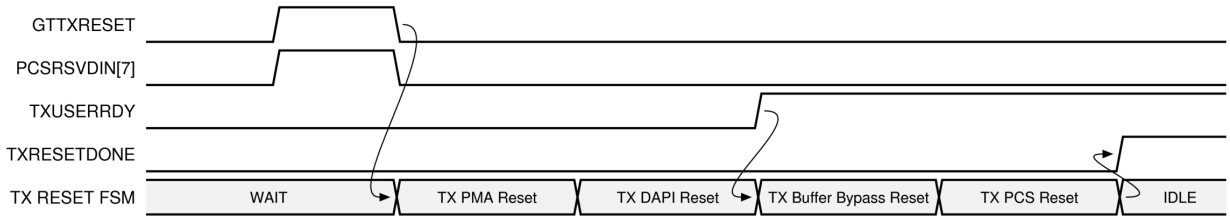


Transceiver TX Reset in Response to GTTXRESET Pulse in Full Sequential Reset

The GTY transceiver allows you to reset the entire TX completely at any time by sending GTTXRESET and PCSRSVDIN[7] (TXDAPIRESET) an active-High pulse. These conditions must be met when using GTTXRESET:

1. TXRESETMODE must be set to sequential mode.
2. All TXPMARESETMASK, PCSRSVDIN[6:5] (TXDAPIRESETMASK), and TXPCSRESETMASK bits should be held High during the reset sequence before TXRESETDONE is detected High.
3. The associated PLL must indicate locked.
4. The guideline for this asynchronous GTTXRESET pulse width is one period of the reference clock.

Figure 17: **Transmitter Reset after GTTXRESET Pulse in Full Sequential Reset**



Transceiver TX Component Reset

TX PMA and TX PCS can be reset individually. Component reset is enabled by setting the appropriate TXPMARESETMASK and TXPCSRESETMASK bits along with TXRESETMODE and then toggling GTTXRESET.

Driving GTTXRESET from High to Low starts the component reset process. All TXPMARESETMASK and TXPCSRESETMASK bits along with TXRESETMODE must be held constant during the reset process.

When TXRESETMODE is set to sequential mode, the internal resets are toggled in sequence depending on TXPMARESTMASK and TXPCSRESETMASK selection. When TXRESETMODE is set to single mode, the internal resets are toggled simultaneously depending on TXPMARESETMASK and TXPCSRESTMASK selection.

In sequential mode, if the TX PCS is to be reset, TXUSERRDY must assert to High prior to the internal PCS reset signal being released allowing TX reset to be completed.

Figure 18: **Transmitter Reset after GTTXRESET Pulse in Component Sequential Mode**

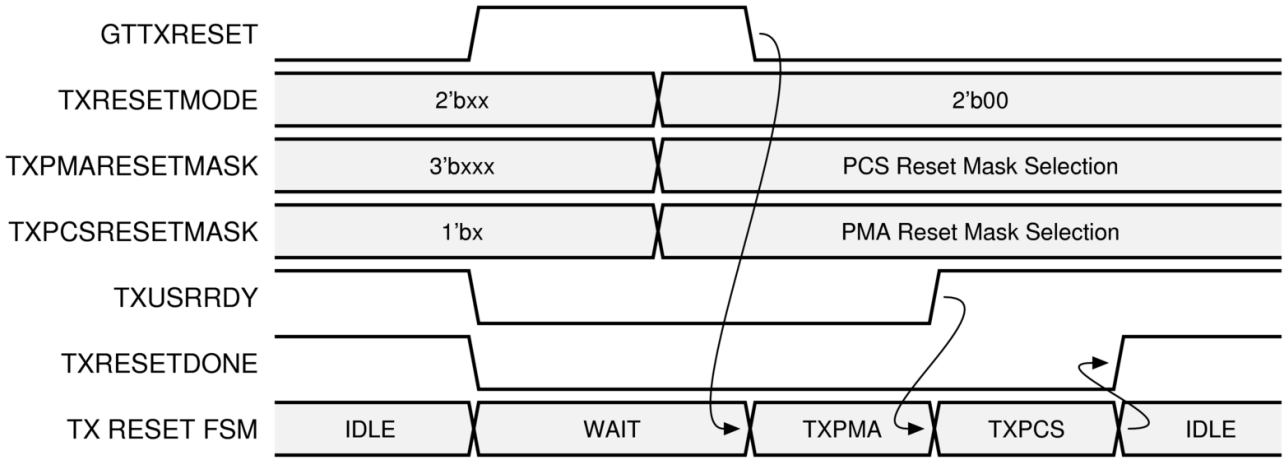
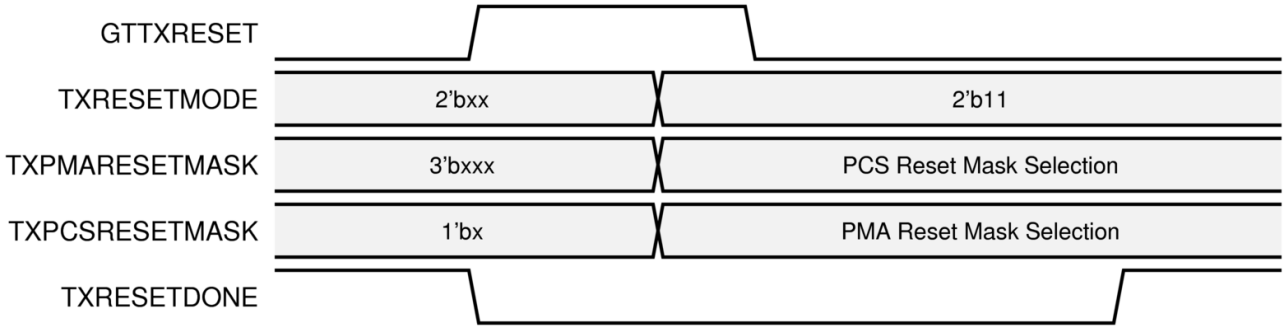


Figure 19: **Transmitter Reset after GTTXRESET Pulse in Component Single Mode**



The following table lists the recommended resets for common situations.

Table 23: **Recommended Transmitter Resets for Common Situations**

Situation	Components to be Reset	Recommended TX Reset Setting		
		TXRESETMODE	TXPMARESETMASK	TXPCSRESETMASK
After power up and confirmation	RPLL, LCPLL, Entire TX	2'b00	3'b111	1'b1
After turning on a reference clock to the LC/RPLL being used	RPLL, LCPLL, Entire TX	2'b00	3'b111	1'b1
After changing the reference clock to the LC/RPLL being used	RPLL, LCPLL, Entire TX	2'b00	3'b111	1'b1
After assertion/ deassertion of LCPLLPD or RPLLPD for the PLL being used	RPLL, LCPLL, Entire TX	2'b00	3'b111	1'b1

Table 23: **Recommended Transmitter Resets for Common Situations** *(cont'd)*

Situation	Components to be Reset	Recommended TX Reset Setting		
		TXRESETMODE	TXPMARESETMASK	TXPCSRESETMASK
After assertion/ deassertion of TXPD[1:0]	Entire TX	2'b00	3'b111	1'b1
TX rate change	TX PMA and TX PCS	2'b00	3'b110	1'b1
TX parallel clock source reset	TX PCS	2'b00/2'b11	3'b000	1'b1

After Power-up and Configuration

The PLL being used and the entire TX require a reset after configuration. See Transceiver TX Reset in Response to Completion of Configuration.

After Turning on a Reference Clock to the LCPLL/RPLL Being Used

If the reference clock(s) changes or the transceiver(s) are powered up after configuration, perform a full TX sequential reset after the PLL fully completes its reset procedure.

After Changing the Reference Clock to the LCPLL/RPLL being Used

Whenever the reference clock input to the PLL is changed, the PLL must be reset afterward to ensure that it locks to the new frequency. Perform a full TX sequential reset after the PLL fully completes its reset procedure.

After Assertion/Deassertion of LC/RPLLPD for the PLL being Used

When the LCPLL or RPLL being used goes back to normal operation after power down, the PLL must be reset. Perform a full TX sequential reset after the PLL fully completes its reset procedure.

After Assertion/Deassertion of TXPD[1:0]

After the TXPD signal is deasserted, perform a full TX sequential reset.

TX Rate Change

When a rate change is performed using the TXRATE port, the required reset sequence is performed automatically. When TXRESETDONE is asserted, it indicates that both a rate change and the necessary reset sequence have been applied and completed.

TX Parallel Clock Source Reset

The clocks driving TXUSRCLK must be stable for correct operation. Perform a TX PCS reset after the clock source re-locks.

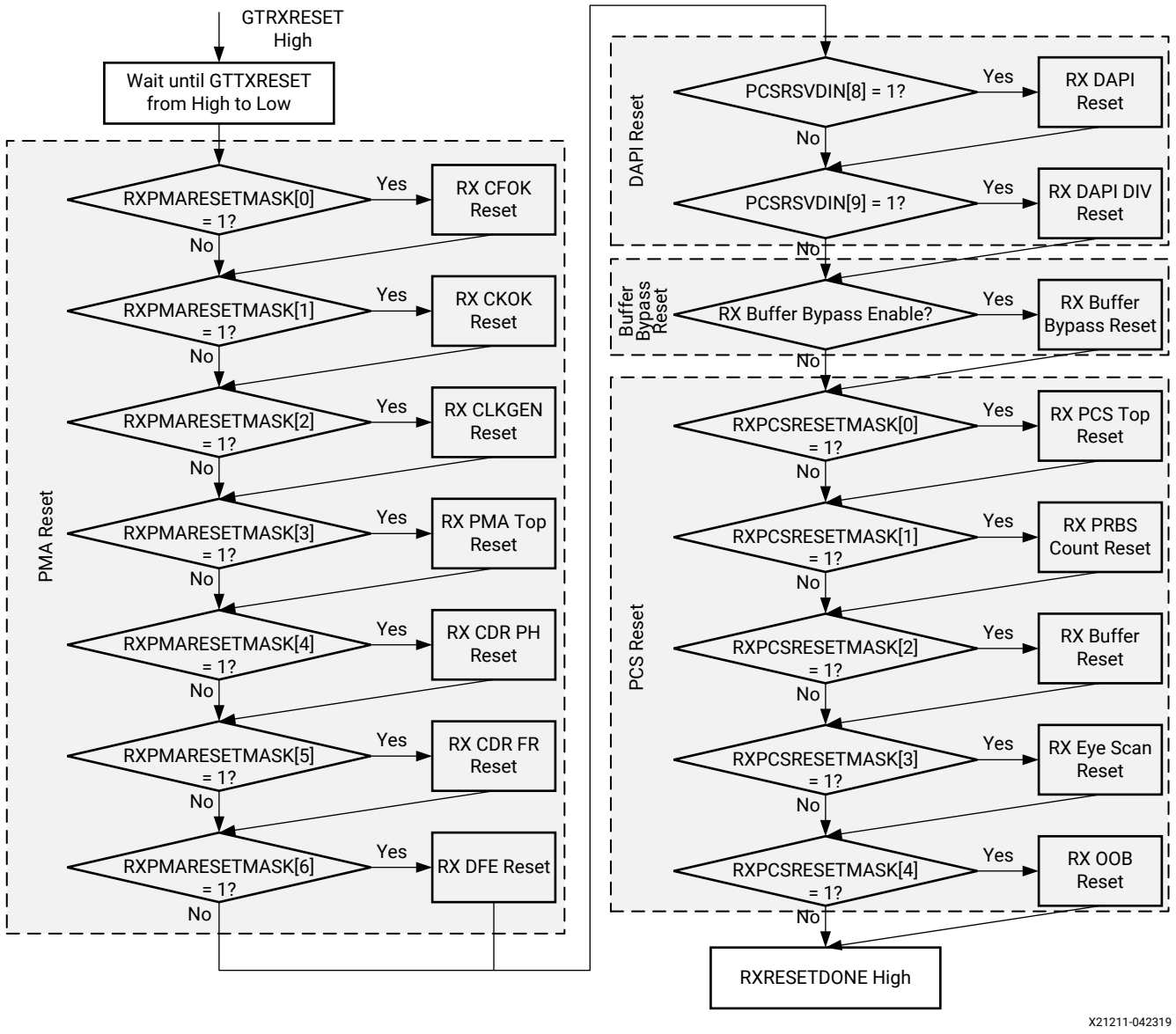
RX Initialization and Reset

The transceiver RX uses a reset state machine to control the reset process. Due to its complexity, the transceiver RX is partitioned into more reset regions than the transceiver TX. The partition allows RX initialization and reset to be operated in either sequential mode, as shown in the following figure, or single mode:

1. RX in Sequential Mode: To initialize the transceiver RX, RXRESETMODE must be set to sequential mode. The RX components that are required to be rest are determined by setting the appropriate RXPMARESETMASK and RXPCSRESETMASK bits to High. The reset sequence is then triggered by toggling GTRXRESET and then internal component resets are triggered sequentially. The reset state machine executes the reset sequence as shown in the following figure, covering the entire RX PMA, RX DPI, RX buffer bypass (if used), and RX PCS. During normal operation, the reset state machine runs until RXRESETDONE transitions from Low to High.
2. RX in Single Mode: When the transceiver RX is in single mode, RXRESETMODE must be set to single mode. The RX components that are required to be rest are determined by setting the appropriate RXPMARESETMASK and RXPCSRESETMASK bits to High. The reset sequence is then triggered by toggling GTRXRESET and the internal component resets are triggered simultaneously. In addition, EYESCANRESET, RXOOBRESET, RXCDRRESET, RXPRBSCNTRESET and HSDPPCSRESET pins are available to reset those components directly in single mode.

In either sequential mode or single mode, the RX reset state machine does not reset the PCS until RXUSERRDY goes High. Drive RXUSERRDY High after all clocks used by the application, including RXUSRCLK, are shown to be stable.

Figure 20: **Transceiver RX Reset State Machine Sequence**



Transceiver RX Reset in Response to Completion of Configuration

The RX reset sequence shown in RX Initialization and Reset is not automatically started to follow the global GSR.

These conditions must be met:

1. RXRESETMODE must be set to use the sequential mode.
2. GTRXRESET must be used.

3. All RXPMARESETMASK, PCSRSVDIN[9:8] (RXDAPIRESETMASK), and RXPCSRESETMASK bits should be set to High.
4. GTRXRESET cannot be driven Low until the associated PLL and ILO are locked.
5. Ensure that GTPOWERGOOD is High before releasing LC/RPLLRESET, ILORESET, GTRXRESET, and PCSRSVDIN[10] (RXDAPIRESET).

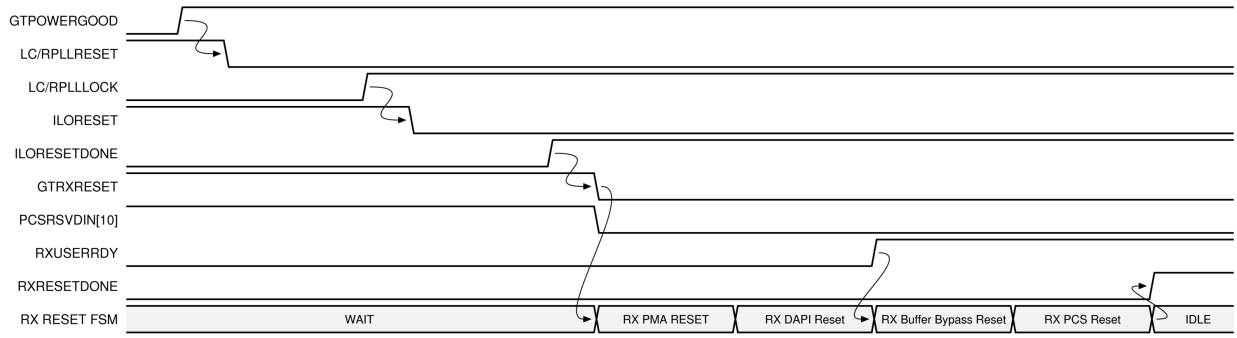
If the reset mode is defaulted to single mode, then you must:

1. Change the reset mode to Sequential mode.
2. All RXPMARESETMASK, PCSRSVDIN[9:8] (RXDAPIRESETMASK), and RXPCSRESETMASK bits should be changed to High.
3. Wait another 300–500 ns.
4. Assert LCPLLRESET, RPLLRESET, ILORESET, GTRXRESET, and PCSRSVIND[10] (RXDAPIRESET) following the reset sequence described in the figure below.

Alternatively, the master reset controller can be used to drive the PLL and RX reset in sequence automatically. Details can be found in Transceiver Master Reset.

RECOMMENDED: *Use the associated PLLLOCK from either the LCPLL or RPLL to release ILORESET from High to Low as shown in the following figure, then use ILORESETDONE to release GTRXRESET from High to Low. The RX reset state machine waits when GTRXRESET is detected High and starts the reset sequence until GTRXRESET is released Low.*

Figure 21: **Receiver Initialization after Configuration**



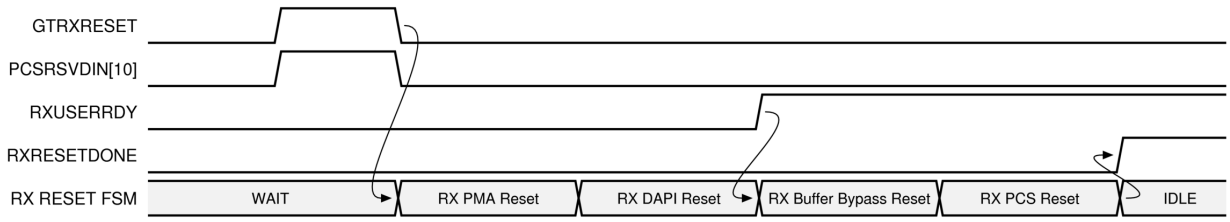
Transceiver RX Reset in Response to GTRXRESET Pulse in Full Sequential Mode

The transceiver allows you to reset the entire RX at any time by sending GTRXRESET and PCSRSVDIN[10] an active-High pulse. These conditions must be met when using GTRXRESET:

1. RXRESETMODE must be set to use sequential mode.

2. All RXPMARESETMASK , PCSRSVDIN[9:8] (RXDAPIRESETMASK), and RXPCSRESETMASK bits should be held to High during the reset sequence before RXRESETDONE is detected High.
3. The associated PLL must indicate locked.
4. The guideline for this asynchronous GTRXRESET pulse width is one period of the reference clock.

Figure 22: **Receiver Reset after GTRXRESET Pulse in Full Sequential Reset**



Transceiver RX Component Reset

Transceiver RX component resets can be reset individually in either sequential mode or single mode. They are primarily used for special cases. These resets are needed when only a specific subsection needs to be reset.

Driving GTRXRESET from High to Low starts the component reset process. All RXPMARESETMASK and RXPCSRESETMASK bits along with RXRESETMODE must be held constant during the rest process.

When RXRESETMODE is set to sequential mode, the internal resets are toggled in sequence depending on the RXPMARESETMASK and RXPCSRESETMASK selection. When RXRESETMODE is set to single mode, the internal resets are toggled simultaneously depending on the RXPMARESETMASK and RXPCSRESETMASK selection.

In sequential mode, if the RX PCS is to be reset, RXUSERRDY must toggle High prior to the internal PCS reset signal being released, allowing RX reset to be completed.

Direct single reset ports EYESCANRESET, RXOOBRESET, RXCDRRESET, RXPRBSCNTRESET, and RXDPPCSRESET are available to perform single resets of the respective RX components. When direct single reset ports are toggled, a single reset is performed regardless of RXPMARESETMASK, RXPCSRESETMASK, and RXRESETMODE selection. These ports must be held Low during any sequential or single rests driven by GTRXRESET.

The following table lists the recommended receiver resets for common situations.

Table 24: **Recommended Receiver Resets for Common Situations**

Situation	Components to be Reset	Recommended RX Reset Setting		
		RXRESETMODE	RXPMARESETMASK	RXPCSRESETMASK
After power up and confirmation	RPLL, LCPLL, ILO, Entire RX	2'b00	7'b1111111	5'b11111
After turning on a reference clock to the LCPLL/RPLL being used	RPLL, LCPLL, ILO, Entire RX	2'b00	7'b1111111	5'b11111
After changing the reference clock to the LCPLL/RPLL being used	RPLL, LCPLL, ILO, Entire RX	2'b00	7'b1111111	5'b11111
After assertion/deassertion of LCPLLPD or RPLLPD for the PLL being used	RPLL, LCPLL, ILO, Entire RX	2'b00	7'b1111111	5'b11111
After assertion/deassertion of RXPD[1:0]	Entire RX	2'b00	7'b1111111	5'b11111
RX rate change	RX PMA and RX PCS	2'b00	7'b1111100	5'b11111
RX parallel clock source reset	RX PCS	2'b00	7'b0000000	5'b11111
After remote power up	Entire RX	2'b00	7'b1111111	5'b11111
Electrical Idle	Entire RX	2'b00	7'b1111111	5'b11111
After connecting RXN/RXP	Entire RX	2'b00	7'b1111111	5'b11111
After recovered clock becomes stable	RX Elastic Buffer	2'b11	7'b0000000	5'b00100
After RX elastic buffer error	RX Elastic Buffer	2'b11	7'b0000000	5'b00100
After changing channel bonding mode in real time	RX Elastic Buffer	2'b11	7'b0000000	5'b00100
After PRBS error	PRBS Error Counter	2'b11	7'b0000000	5'b00010
After comma alignment	RX Elastic Buffer	2'b11	7'b0000000	5'b00100
Eye Scan reset only	Eye Scan	2'b11	7'b0000000	5'b01000

After Power-up and Configuration

The PLL being used and the entire RX require a reset after configuration. See Transceiver RX Reset in Response to Completion of Configuration.

After Turning on a Reference Clock to the LCPLL/RPLL Being Used

If the reference clock(s) changes or transceiver(s) are powered up after configuration, perform a full RX sequential reset after the PLL fully completes its reset procedure.

After Changing the Reference Clock to the LCPLL/RPLL Being Used

Whenever the reference clock input to the PLL is changed, the PLL must be reset afterward to ensure that it locks to the new frequency. Perform a full RX sequential reset after the PLL fully completes its reset procedure.

After Assertion/Deassertion of LC/RPLLPD for the PLL Being Used

When the LCPLL or RPLL being used goes back to normal operation after power down, the PLL must be reset. Perform a full RX sequential reset after the PLL fully completes its reset procedure.

After Assertion/Deassertion of RXPD[1:0]

After the RXPD signal is deasserted, perform a full RX sequential reset.

RX Rate Change

When a rate change is performed using the RXRATE port, the required reset sequence is performed automatically. When RXRESETDONE is asserted, it indicates that both a rate change and then a necessary reset sequence have been applied and completed.

RX Parallel Clock Source Reset

The clocks driving RXUSRCLK must be stable for correct operation. Perform an RX PCS reset after the clock source re-locks.

After Remote Power-Up

If the source of incoming data is powered up after the transceiver that is receiving its data has begun operating, a full RX sequential reset must be performed to ensure a clean lock to the incoming data.

Electrical Idle Reset

For protocols that support OOB and electrical idle, when the differential voltage of the RX input to the transceiver drops to OOB or electrical idle levels, the RX CDR is managed automatically when the attributes associated with electrical idle are set to appropriate values. Use the recommended values from the Wizard.

After Connecting RXN/RXP

When the RX data to the transceiver comes from a connector that can be plugged in and unplugged, a full RX sequential reset must be performed when the data source is plugged in to ensure that it can lock to incoming data.

After Recovered Clock Becomes Stable

Depending on the design of the clocking scheme, it is possible for the RX reset sequence to be completed before the CDR is locked to the incoming data. In this case, the recovered clock might not be stable when RXRESETDONE is asserted. When the RX PCS is used, a single mode reset targeting the RX elastic buffer must be triggered after the recovered clock becomes stable.

When RX buffer bypass is used, the alignment procedure should not start until the recovered clock becomes stable.

Refer to the Versal ACAP data sheets for successful CDR lock-to-data criteria.

After an RX Elastic Buffer Error

After an RX elastic buffer overflow or underflow, a single mode reset targeting the RX elastic buffer must be triggered to ensure correct behavior.

After Changing Channel Bonding Mode During Run Time

When set to TRUE, RXBUF_RESET_ON_CB_CHANGE enables automatic reset of the RX elastic buffer when EB8B10B_CHAN_BOND_MASTER, EB8B10B_CHAN_BOND_SLAVE, or EB8B10B_CHAN_BOND_LEVEL change. *naming may still change*

After a PRBS Error

PRBSCNTRESET is asserted to reset the PRBS error counter.

After Comma Realignment

When set to TRUE, EB8B10B_RESET_ON_COMMAALIGN enables automatic reset of the RX elastic buffer during comma realignment.

Transceiver Master Reset

A master reset controller is available in the GTY transceiver. The master reset controller automatically steps through the reset of the LCPLL, RPLL, ILO, TX programmable divider, RX programmable divider, TX channel, and RX channel. The master reset controller state machine operates as shown in Figure 23 for the TX and Figure 24 for the RX.

Figure 23: **Transceiver TX Master Reset State Machine Sequence**

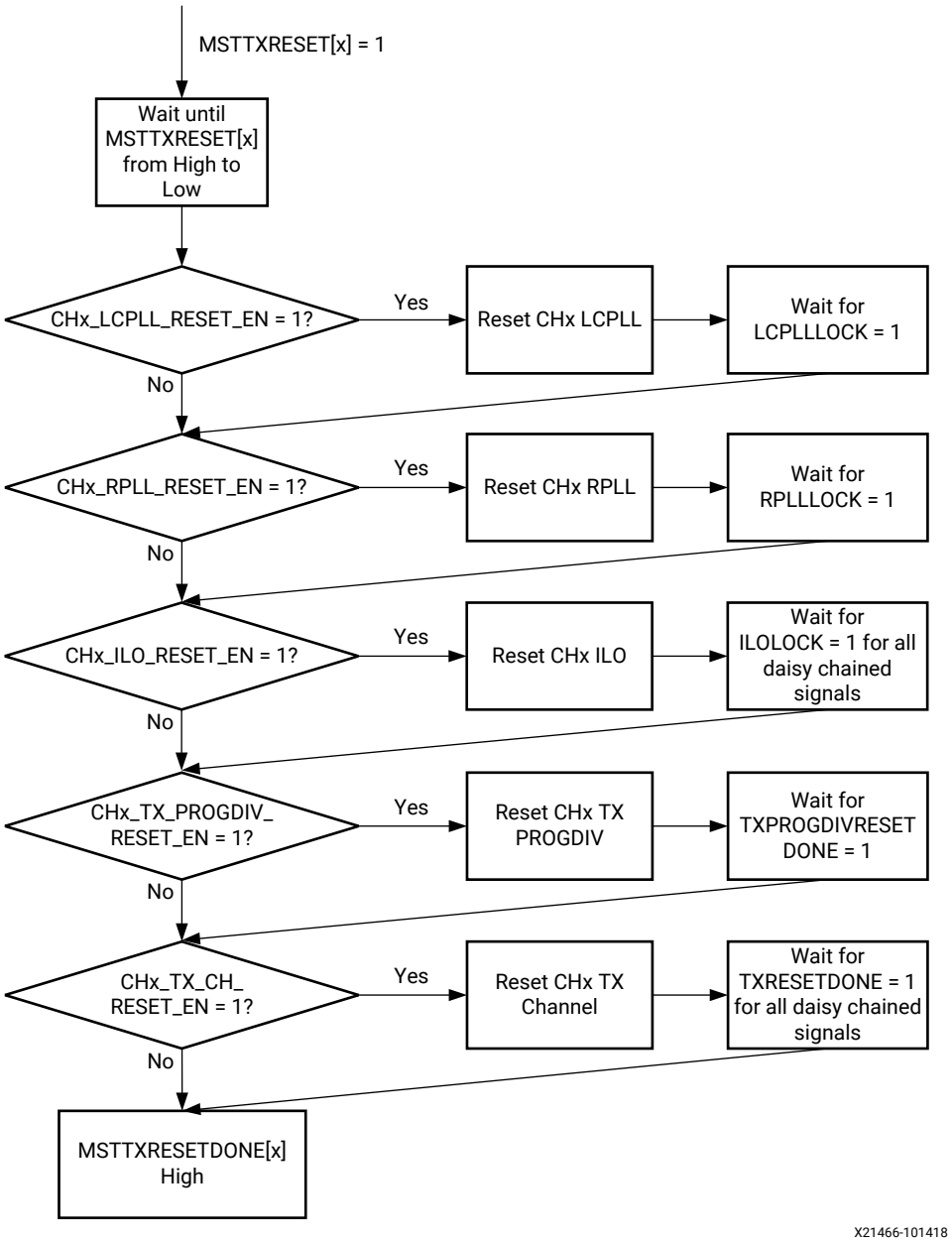
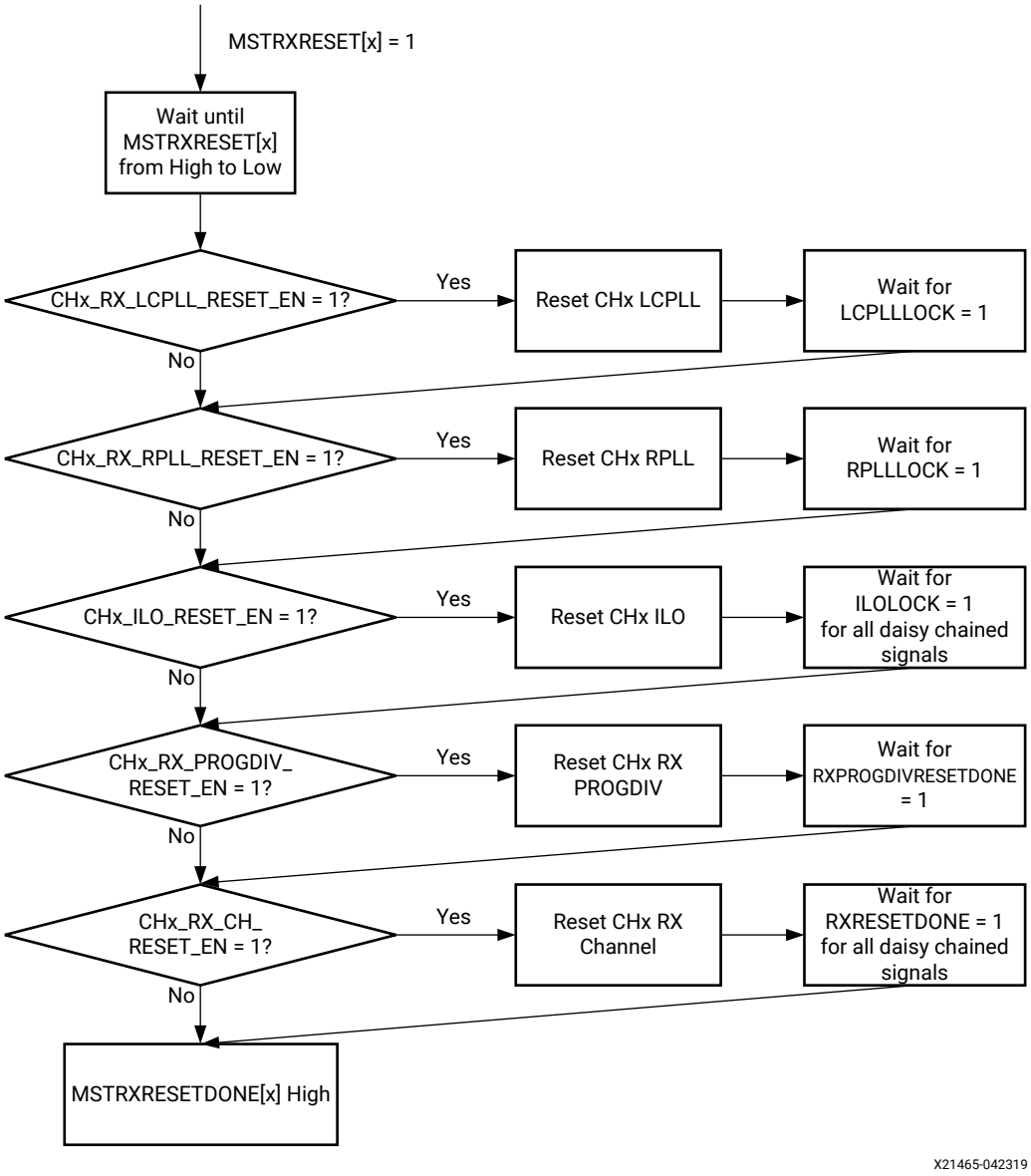


Figure 24: **Transceiver RX Master Reset State Machine Sequence**



Each channel contains a master reset controller port for the given channel. In multi-lane protocols, the master reset signal should be toggled individually on all used lanes. However, ILORESETDONE and TX/RXRESETDONE signals can be daisy-chained to ensure that the previous step is completed on all lanes before the master reset controller proceeds.

Transceiver Master TX Reset in Response to Completion of Configuration

The TX reset sequence shown in Figure 15 is not automatically started to follow the global GSR.

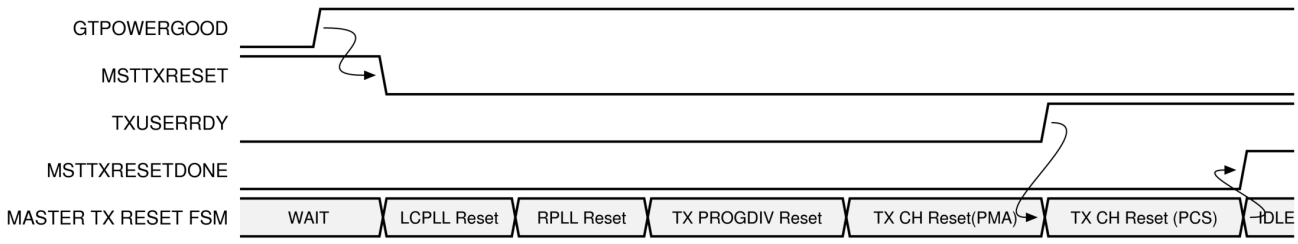
These conditions must be met:

1. TXRESETMODE must be set to use the sequential mode.
2. All TXPMARESETMASK and TXPCSRESETMASK bits should be set to High.
3. Set the appropriate MST_RESET_CFG attribute bits based on PLL selection, PROGDIV selection, and channel usage.
4. MSTTXRESET is to be used.
5. Ensure LC/RPLLRESET, TXPROGDIVRESET, PCSRSVDIN[7] (TXDAPIRESET), and GTTXRESET are held Low during the master reset process.
6. Ensure that GTPOWERGOOD is High before releasing MSTTXRESET.

If the reset mode is defaulted to single mode, then you must:

1. Change reset mode to Sequential mode.
2. All TXPMARESETMASK and TXPCSRESETMASK bits should be changed to High.
3. Wait another 300–500 ns.
4. Release MSTTXRESET.

Figure 25: **Transmitter Initialization after Configuration Using Master Reset Controller**



Transceiver Master RX Reset in Response to Completion of Configuration

The RX reset sequence shown in the following figure is not automatically started to follow the global GSR.

These conditions must be met:

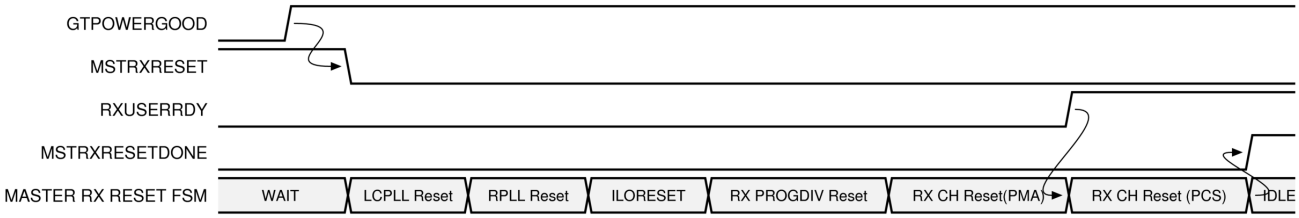
1. RXRESETMODE must be set to use the sequential mode.
2. All RXPMARESETMASK and RXPCSRESETMASK bits should be set to High.
3. Set the appropriate MST_RESET_CFG attribute bits based on PLL selection, PROGDIV selection, and channel usage.
4. MSTRXRESET is to be used.

5. Ensure LC/RPLLRESET, ILORESET, RXPROGDIVRESET, PCSRSVDIN[10] (RXDAPIRESET), and GTRXRESET are held Low during the master reset process.
6. Ensure that GTPOWERGOOD is High before releasing MSTRXRESET.

If the reset mode is defaulted to single mode, then you must:

1. Change reset mode to Sequential mode.
2. All RXPMARESETMASK and RXPCSRESETMASK bits should be changed to High.
3. Wait another 300–500 ns.
4. Release MSTRXRESET.

Figure 26: **Receiver Initialization after Configuration Using Master Reset Controller**



Quad Sharing with Multiple IP Cores

When multiple IP cores are located within the same Quad, there are limitations on how the reset can be performed. When an IP core asserts the master reset, the corresponding PLL that it uses also goes through reset. If any other IP within the same Quad shares the same PLL, its operation will be affected. In this situation, any IP that shares the same PLL within the same Quad must perform reset at the same time.

Ports and Attributes

The following table lists ports required by the TX and RX initialization process.

Table 25: **Reset and Initialization Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_EYESCANRESET	Input	ASYNC	This active-High port resets the eye scan block.
CH[0/1/2/3]_GTRXRESET	Input	ASYNC	This port is driven high and then deasserted to start a RX reset sequence. The components to be reset are determined by RXPMARESETMASK and RXPCSRESETMASK. In sequential mode, the resets are performed sequentially. In single mode, the resets are performed simultaneously.

Table 25: **Reset and Initialization Ports** *(cont'd)*

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_GTTXRESET	Input	ASYNC	This port is driven high and then deasserted to start a TX reset sequence. The components to be reset are determined by TXPMARESETMASK and TXPCSRESETMASK. In sequential mode, the resets are performed sequentially. In single mode, the resets are performed simultaneously.
CH[0/1/2/3]_HSDPPCSRESET	Input	ASYNC	This port is driven high and then deasserted to start a single mode reset on the RX PCS. The reset is not dependent on RXRESETMODE or RXPCSRESETMASK setting.
CH[0/1/2/3]_ILORESET	Input	ASYNC	Active high signal that resets the ILO.
CH[0/1/2/3]_ILORESETDONE	Output	ASYNC	This active-high signal indicates the GTY transceiver ILO has finished reset and is ready for use. This port is driven Low when ILORESET goes High and is not driven High until the GTY transceiver has completed all ILO reset steps.
HSCLK[0/1]_LCPLLLOCK	Output	ASYNC	Active-High LCPLL frequency lock signal indicates that the LCPLL frequency is within a predetermined tolerance. The GTY transceiver and its clock outputs are not reliable until this condition is met.
HSCLK[0/1]_LCPLLRESET	Input	ASYNC	Active high signal that resets the LCPLL.
MSTRXRESET[3:0]	Input	ASYNC	This port is driven high and then deasserted to start a RX master reset sequence. Bit 3: CH3 RX master reset Bit 2: CH2 RX master reset Bit 1: CH1 RX master reset Bit 0: CH0 RX master reset
MSTRXRESETDONE[3:0]	Output	ASYNC	This active-high signal indicates the GTY transceiver RX master has finished reset and is ready for use. This port is driven Low when MSTRXRESET goes High and is not driven High until the GTY transceiver has completed all RX master reset steps, including channel PCS reset gated by RXUSERRDY. Bit 3: CH3 RX master reset done Bit 2: CH2 RX master reset done Bit 1: CH1 RX master reset done Bit 0: CH0 RX master reset done
MSTTXRESET[3:0]	Input	ASYNC	This port is driven high and then deasserted to start a TX master reset sequence. Bit 3: CH3 TX master reset Bit 2: CH2 TX master reset Bit 1: CH1 TX master reset Bit 0: CH0 TX master reset

Table 25: **Reset and Initialization Ports** *(cont'd)*

Port	Direction	Clock Domain	Description
MSTTXRESETDONE[3:0]	Output	ASYNC	This active-high signal indicates the GTY transceiver TX master has finished reset and is ready for use. This port is driven Low when MSTTXRESET goes High and is not driven High until the GTY transceiver has completed all RX master reset steps, including channel PCS reset gated by TXUSERRDY. Bit 3: CH3 TX master reset done Bit 2: CH2 TX master reset done Bit 1: CH1 TX master reset done Bit 0: CH0 TX master reset done
CH[0/1/2/3]_PCSRSVDIN[10]	Input	ASYNC	This port is driven high and then deasserted to start a reset sequence on only the RX DAPI.
CH[0/1/2/3]_PCSRSVDIN[6:5]	Input	ASYNC	TX DAPI reset mask selection Bit 1: TX DAPI Divider Bit 0: TX DAPI
CH[0/1/2/3]_PCSRSVDIN[7]	Input	ASYNC	This port is driven high and then deasserted to start a reset sequence on only the TX DAPI.
CH[0/1/2/3]_PCSRSVDIN[9:8]	Input	ASYNC	RX DAPI reset mask selection Bit 1: RX DAPI Divider Bit 0: RX DAPI
CH[0/1/2/3]_PCSRSVDOUT[10]	Output	ASYNC	This active-high signal indicates TX DAPI reset is complete.
CH[0/1/2/3]_PCSRSVDOUT[11]	Output	ASYNC	This active-high signal indicates RX DAPI reset is complete.
HSCLK[0/1]_RPLLLOCK	Output	ASYNC	Active-High RPLL frequency lock signal indicates that the RPLL frequency is within a predetermined tolerance. The GTY transceiver and its clock outputs are not reliable until this condition is met.
HSCLK[0/1]_RPLLRESET	Input	ASYNC	Active high signal that resets the RPLL.
CH[0/1/2/3]_RXCDRRESET	Input	ASYNC	This port is driven high and then deasserted to start a single mode reset on CDR. The reset is not dependent on RXRESETMODE or RXPMARESETMASK setting.
CH[0/1/2/3]_RXOOBRESET	Input	ASYNC	This port is driven high and then deasserted to start a single mode reset on RX OOB. The reset is not dependent on RXRESETMODE or RXPCSRESETMASK setting.
CH[0/1/2/3]_RXPCSRESETMASK[4:0]	Input	ASYNC	RX PCS reset mask selection Bit 4: RX OOB Bit 3: Eye Scan Bit 2: RX Buffer Bit 1: PRBS Counter Bit 0: RX PCS Top
CH[0/1/2/3]_RXPMARESETDONE	Output	ASYNC	This active-high signal indicates RX PMA reset is complete.

Table 25: **Reset and Initialization Ports** *(cont'd)*

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_RXPMARESETMASK[6:0]	Input	ASYNC	RX PMA reset mask selection Bit 6: DFE Bit 5: CDR FR Bit 4: CDR PH Bit 3: RX PMA Top Bit 2: RX CLKGEN Bit 1: CKOK Bit 0: CFOK
CH[0/1/2/3]_RXPRBSCNTRESET	Input	ASYNC	Assert this port High to reset the PRBS error counter.
CH[0/1/2/3]_RXRESETDONE	Output	ASYNC	This active-high signal indicates the GTY transceiver RX has finished reset and is ready for use. This port is driven Low when GTRXRESET goes High and is not driven High until the GTY transceiver has completed all RX reset steps, including PCS reset gated by RXUSERRDY.
CH[0/1/2/3]_RXRESETMODE[1:0]	Input	ASYNC	Reset mode port for RX. 2'b00: Sequential mode (recommended) 2'b01: Reserved 2'b10: Reserved 2'b11: Single mode
CH[0/1/2/3]_RXUSERRDY	Input	ASYNC	This port is driven high by the user application when RXUSRCLK is stable
CH[0/1/2/3]_TXPCSRESETMASK	Input	ASYNC	TX PCS reset mask
CH[0/1/2/3]_TXPMARESETDONE	Output	ASYNC	This active-high signal indicates TX PMA reset is complete.
CH[0/1/2/3]_TXPMARESETMASK[2:0]	Input	ASYNC	TX PMA reset mask selection Bit 2: TX PMA Bit 1: TX CLKGEN Bit 0: CKCAL
CH[0/1/2/3]_TXRESETDONE	Output	ASYNC	This active-high signal indicates the GTY transceiver TX has finished reset and is ready for use. This port is driven Low when GTTXRESET goes High and is not driven High until the GTY transceiver has completed all TX reset steps, including PCS reset gated by TXUSERRDY.
CH[0/1/2/3]_TXRESETMODE[1:0]	Input	ASYNC	Reset mode port for TX. 2'b00: Sequential mode (recommended) 2'b01: Reserved 2'b10: Reserved 2'b11: Single mode
CH[0/1/2/3]_TXUSERRDY	Input	ASYNC	This port is driven high by the user application when TXUSRCLK is stable

The following table lists attributes required by GTY transceiver TX and RX initialization. In general cases, the reset time required by the TX PMA or the TX PCS varies depending on line rate. The factors affecting PMA reset time and PCS reset time are the user-configurable attributes TX_PMA_RESET_TIME and TX_PCS_RESET_TIME.

Table 26: **Reset and Initialization Attributes**

Reset and Initialization Attributes		
Attribute	**Address**	
CTRL_RSV_CFG0		0x0C03
Label	**Bit Field**	**Description**
CH3_RX_RPLL_RESET_EN	[31:31]	Master reset control component selection for Channel 3 PLL when RPLL is used by Channel 3 RX.
CH2_RX_RPLL_RESET_EN	[30:30]	Master reset control component selection for Channel 2 PLL when RPLL is used by Channel 2 RX.
CH1_RX_RPLL_RESET_EN	[29:29]	Master reset control component selection for Channel 1 PLL when RPLL is used by Channel 1 RX.
CH0_RX_RPLL_RESET_EN	[28:28]	Master reset control component selection for Channel 0 PLL when RPLL is used by Channel 0 RX.
Attribute	**Address**	
HSCLK0_LCPLL_LGC_CFG0		0x0C0C
HSCLK1_LCPLL_LGC_CFG0		0x0E0C
Label	**Bit Field**	**Description**
LCPLLLOCKEN	[11:11]	Active-High signal enables the LCPLL lock detector.
Attribute	**Address**	
MST_RESET_CFG		0x0D12
Label	**Bit Field**	**Description**
CH3_RX_LCPLL_RESET_EN	[31:31]	Master reset control component selection for Channel 3 PLL when LCPLL is used by Channel 3 RX.
CH2_RX_LCPLL_RESET_EN	[30:30]	Master reset control component selection for Channel 2 PLL when LCPLL is used by Channel 2 RX.
CH1_RX_LCPLL_RESET_EN	[29:29]	Master reset control component selection for Channel 1 PLL when LCPLL is used by Channel 1 RX.
CH0_RX_LCPLL_RESET_EN	[28:28]	Master reset control component selection for Channel 0 PLL when LCPLL is used by Channel 0 RX.
CH3_RX_CH_RESET_EN	[27:27]	Master reset control component selection for Channel 3 RX.
CH3_RX_PROGDIV_RESET_EN	[26:26]	Master reset control component selection for Channel 3 RX PROGDIV.
CH3_TX_CH_RESET_EN	[25:25]	Master reset control component selection for Channel 3 TX.
CH3_TX_PROGDIV_RESET_EN	[24:24]	Master reset control component selection for Channel 3 TX PROGDIV.
CH3_ILO_RESET_EN	[23:23]	Master reset control component selection for Channel 3 ILO.
CH2_RX_CH_RESET_EN	[22:22]	Master reset control component selection for Channel 2 RX.
CH2_RX_PROGDIV_RESET_EN	[21:21]	Master reset control component selection for Channel 2 RX PROGDIV.
CH2_TX_CH_RESET_EN	[20:20]	Master reset control component selection for Channel 2 TX.

Table 26: **Reset and Initialization Attributes** *(cont'd)*

Reset and Initialization Attributes		
CH2_TX_PROGDIV_RESET_EN	[19:19]	Master reset control component selection for Channel 2 TX PROGDIV.
CH2_ILO_RESET_EN	[18:18]	Master reset control component selection for Channel 2 ILO.
CH1_RX_CH_RESET_EN	[17:17]	Master reset control component selection for Channel 1 RX.
CH1_RX_PROGDIV_RESET_EN	[16:16]	Master reset control component selection for Channel 1 RX PROGDIV.
CH1_TX_CH_RESET_EN	[15:15]	Master reset control component selection for Channel 1 TX.
CH1_TX_PROGDIV_RESET_EN	[14:14]	Master reset control component selection for Channel 1 TX PROGDIV.
CH1_ILO_RESET_EN	[13:13]	Master reset control component selection for Channel 1 ILO.
CH0_RX_CH_RESET_EN	[12:12]	Master reset control component selection for Channel 0 RX.
CH0_RX_PROGDIV_RESET_EN	[11:11]	Master reset control component selection for Channel 0 RX PROGDIV.
CH0_TX_CH_RESET_EN	[10:10]	Master reset control component selection for Channel 0 TX.
CH0_TX_PROGDIV_RESET_EN	[9:9]	Master reset control component selection for Channel 0 TX PROGDIV.
CH0_ILO_RESET_EN	[8:8]	Master reset control component selection for Channel 0 ILO.
CH3_RPLL_RESET_EN	[7:7]	Master reset control component selection for Channel 3 PLL when RPLL is used by Channel 3 TX.
CH2_RPLL_RESET_EN	[6:6]	Master reset control component selection for Channel 2 PLL when RPLL is used by Channel 2 TX.
CH1_RPLL_RESET_EN	[5:5]	Master reset control component selection for Channel 1 PLL when RPLL is used by Channel 1 TX.
CH0_RPLL_RESET_EN	[4:4]	Master reset control component selection for Channel 0 PLL when RPLL is used by Channel 0 TX.
CH3_LCPLL_RESET_EN	[3:3]	Master reset control component selection for Channel 3 PLL when LCPLL is used by Channel 3 TX.
CH2_LCPLL_RESET_EN	[2:2]	Master reset control component selection for Channel 2 PLL when LCPLL is used by Channel 2 TX.
CH1_LCPLL_RESET_EN	[1:1]	Master reset control component selection for Channel 1 PLL when LCPLL is used by Channel 1 TX.
CH0_LCPLL_RESET_EN	[0:0]	Master reset control component selection for Channel 0 PLL when LCPLL is used by Channel 0 TX.
Attribute	**Address**	
CH0_RESET_TIME_CFG0		0x0C1E
CH1_RESET_TIME_CFG0		0x0D1E
CH2_RESET_TIME_CFG0		0x0E1E
CH3_RESET_TIME_CFG0		0x0F1E
Label	**Bit Field**	**Description**
TX_PCS_RESET_TIME	[29:25]	Reserved. Use the recommended value from the Wizard.
TX_PMA_RESET_TIME	[24:20]	Reserved. Use the recommended value from the Wizard.
Attribute	**Address**	
CH0_RESET_TIME_CFG1		0x0C1F

Table 26: **Reset and Initialization Attributes** *(cont'd)*

Reset and Initialization Attributes		
CH1_RESET_TIME_CFG1		0x0D1F
CH2_RESET_TIME_CFG1		0x0E1F
CH3_RESET_TIME_CFG1		0x0F1F
Label	**Bit Field**	**Description**
RX_PMA_RESET_TIME	[29:25]	Reserved. Use the recommended value from the Wizard.
Attribute	**Address**	
CH0_RESET_TIME_CFG2		0x0C20
CH1_RESET_TIME_CFG2		0x0D20
CH2_RESET_TIME_CFG2		0x0E20
CH3_RESET_TIME_CFG2		0x0F20
Label	**Bit Field**	**Description**
RX_PCS_RESET_TIME	[19:15]	Reserved. Use the recommended value from the Wizard.
Attribute	**Address**	
HSCLK0_RPLL_LGC_CFG0		0x0D0C
HSCLK1_RPLL_LGC_CFG0		0x0F0C
Label	**Bit Field**	**Description**
RPLLLOCKEN	[11:11]	Active-High signal enables the RPLL lock detector.
Attribute	**Address**	
RXRSTDONE_DIST_SEL		0x0D11
Label	**Bit Field**	**Description**
CH3_CONSUMPT_SEL	[30:28]	RX Master reset controller RESETDONE daisy chain consumption selection for Channel 3.
CH3_DISTSOUTH_SEL	[27:26]	RX Master reset controller RESETDONE daisy chain distribution south for selection Channel 3.
CH3_DISTNORTH_SEL	[25:24]	RX Master reset controller RESETDONE daisy chain distribution north for selection Channel 3.
CH2_CONSUMPT_SEL	[22:20]	RX Master reset controller RESETDONE daisy chain consumption selection for Channel 2.
CH2_DISTSOUTH_SEL	[19:18]	RX Master reset controller RESETDONE daisy chain distribution south for selection Channel 2.
CH2_DISTNORTH_SEL	[17:16]	RX Master reset controller RESETDONE daisy chain distribution north for selection Channel 2.
CH1_CONSUMPT_SEL	[14:12]	RX Master reset controller RESETDONE daisy chain consumption selection for Channel 1.
CH1_DISTSOUTH_SEL	[11:10]	RX Master reset controller RESETDONE daisy chain distribution south for selection Channel 1.
CH1_DISTNORTH_SEL	[9:8]	RX Master reset controller RESETDONE daisy chain distribution north for selection Channel 1.
CH0_CONSUMPT_SEL	[6:4]	RX Master reset controller RESETDONE daisy chain consumption selection for Channel 0.
CH0_DISTSOUTH_SEL	[3:2]	RX Master reset controller RESETDONE daisy chain distribution south for selection Channel 0.

Table 26: **Reset and Initialization Attributes** *(cont'd)*

Reset and Initialization Attributes		
CH0_DISTNORTH_SEL	[1:0]	RX Master reset controller RESETDONE daisy chain distribution north for selection Channel 0.
Attribute	**Address**	
TXRSTDONE_DIST_SEL		0x0D10
Label	**Bit Field**	**Description**
CH3_CONSUMPT_SEL	[30:28]	TX Master reset controller RESETDONE daisy chain consumption selection for Channel 3.
CH3_DISTSOUTH_SEL	[27:26]	TX Master reset controller RESETDONE daisy chain distribution south for selection Channel 3.
CH3_DISTNORTH_SEL	[25:24]	TX Master reset controller RESETDONE daisy chain distribution north for selection Channel 3.
CH2_CONSUMPT_SEL	[22:20]	TX Master reset controller RESETDONE daisy chain consumption selection for Channel 2.
CH2_DISTSOUTH_SEL	[19:18]	TX Master reset controller RESETDONE daisy chain distribution south for selection Channel 2.
CH2_DISTNORTH_SEL	[17:16]	TX Master reset controller RESETDONE daisy chain distribution north for selection Channel 2.
CH1_CONSUMPT_SEL	[14:12]	TX Master reset controller RESETDONE daisy chain consumption selection for Channel 1.
CH1_DISTSOUTH_SEL	[11:10]	TX Master reset controller RESETDONE daisy chain distribution south for selection Channel 1.
CH1_DISTNORTH_SEL	[9:8]	TX Master reset controller RESETDONE daisy chain distribution north for selection Channel 1.
CH0_CONSUMPT_SEL	[6:4]	TX Master reset controller RESETDONE daisy chain consumption selection for Channel 0.
CH0_DISTSOUTH_SEL	[3:2]	TX Master reset controller RESETDONE daisy chain distribution south for selection Channel 0.
CH0_DISTNORTH_SEL	[1:0]	TX Master reset controller RESETDONE daisy chain distribution north for selection Channel 0.

Rate Change

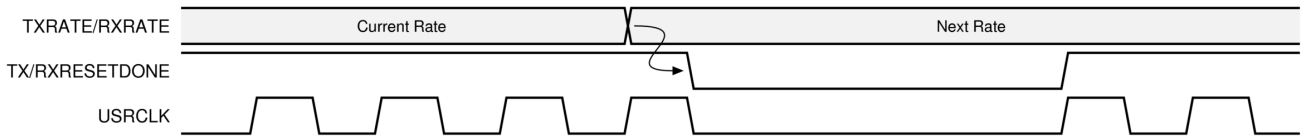
The Versal ACAP GTY transceiver provides great flexibility by allowing users to dynamically change the operating line rate. To ensure all attributes are correctly configured during the rate change, Xilinx recommends generating wrappers from the Versal ACAPs Transceivers Wizard (Wizard) that allows the designer to preconfigure a set of line rates at which the device is intend to operate.

When a rate change is performed using the CH*_TXRATE or CH*_RXRATE port to any of the preconfigured line rates set using the Wizard, the required reset sequence and necessary attribute update is performed automatically. The user should wait for the assertion of CH*_TXRESETDONE or CH*_RXRESETDONE as an indication that the current rate change process and necessary reset sequence have completed.

Rate Change Use Mode Without Reference Clock Change

If the reference clock frequency does not need to change, the user shall use the rate change port and wait for the CH*_TXRESETDONE or CH*_RXRESETDONE assertion to signal the rate change process completion. The following timing diagram shows the sequence details.

Figure 27: **Transceiver Rate Change with No Changes in Reference Clock**

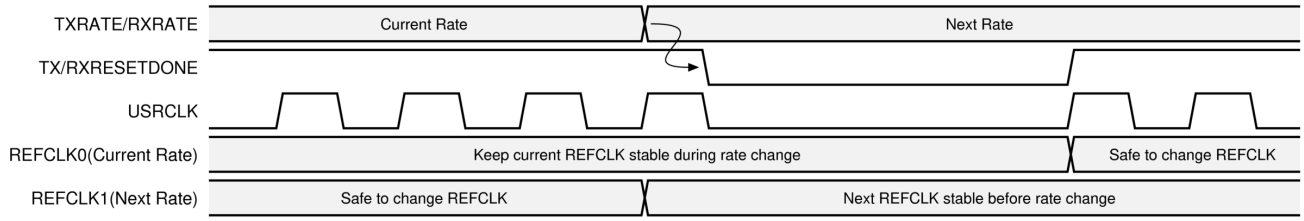


Rate Change Use Mode with Reference Clock Changes

If the user decides to use a different reference clock, whether it is an entirely different clock source or the same clock source but with only changes in the actual frequency, additional steps need to be followed when performing the rate change.

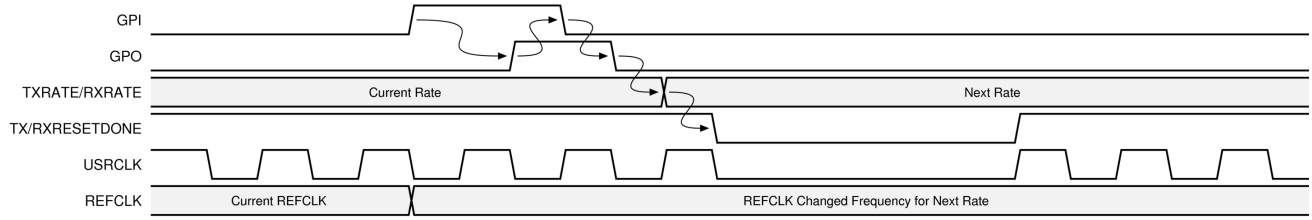
If the user decides to use a different reference clock port/source, the current reference clock source must be kept unchanged and stable during the entire rate change sequence. At the same time, the new reference clock source should be set and stable when the rate change procedure starts. The following timing diagram shows the required sequence.

Figure 28: **Transceiver Rate Change with Changes in REFCLK Source**



If the user decide to use the same reference clock source but the actual frequency will change, the corresponding GPI ports need to be toggled according to the following timing diagram below. The new reference clock frequency should be set and stable when the GPI port is toggled. See the following timing diagram for the proper rate change sequence under this use mode.

Figure 29: **Transceiver Rate Change with Changes in REFCLK Frequency without Port Change**



Rate Change Notes

In a design generated through the Wizard containing multiple rates, if the device registers were manually modified by the user, this might cause unexpected behaviors. If a register manually set by the user is part of the rate change sequence, its value will be overwritten during the rate change procedure.

The rate change algorithm does not keep a record of the original device state before any manual register overwrites. This means that if the user modifies a register that is not part of the rate change sequence, it will not be refreshed back to its default state even after each rate change procedure. This could lead to a situation where the register that was manually modified by the user might not be the most optimal and could cause performance issues in the new line rate configuration. Therefore, it is up to the user to track all manually modified register values before and after each rate change and correct them if necessary following the recommendations in this manual.

Ports and Attributes

The following table defines the rate change ports.

Table 27: **Rate Change Ports**

Port	Direction	Clock Domain	Description
GPI[15:0]	Input	ASYNC	Assert corresponded GPI ports before rate change to indicate the reference clock changes frequency on same port GPI[0]: Ch0 TX Simplex and Duplex GPI[1]: Ch1 TX Simplex and Duplex GPI[2]: Ch2 TX Simplex and Duplex GPI[3]: Ch3 TX Simplex and Duplex GPI[4]: Ch0 RX Simplex GPI[5]: Ch1 RX Simplex GPI[6]: Ch2 RX Simplex GPI[7]: Ch3 RX Simplex GPI[15:8]: Reserved
GPO[15:0]	Output	ASYNC	Set corresponded GPO ports in response to assertions of GPI ports GPO[0]: Ch0 TX Simplex and Duplex GPO[1]: Ch1 TX Simplex and Duplex GPO[2]: Ch2 TX Simplex and Duplex GPO[3]: Ch3 TX Simplex and Duplex GPO[4]: Ch0 RX Simplex GPO[5]: Ch1 RX Simplex GPO[6]: Ch2 RX Simplex GPO[7]: Ch3 RX Simplex GPO[15:8]: Reserved
CH[0/1/2/3]_RXRATE[7:0]	Input	RXUSRCLK	This port is used to perform rate change on the Transceiver RX. The Wizard will preconfigure a list of desired line rates, and this port will be used to dynamically adjust the running line rate based on the preconfigured list. Set this port to the matching preconfigured line rate option value to obtained the proper line rate.
CH[0/1/2/3]_TXRATE[7:0]	Input	TXUSRCLK	This port is used to perform rate change on the Transceiver TX. The Wizard will preconfigure a list of desired line rates, and this port will be used to dynamically adjust the running line rate based on the preconfigured list. Set this port to the matching preconfigured line rate option value to obtained the proper line rate.

There are no rate change attributes in Versal ACAPs.

Power Down

The GTY transceiver supports a range of power-down modes. These modes support both generic power management capabilities as well as those defined in the PCI Express® and SATA standards.

The GTY transceiver offers different levels of power control. Each channel in each direction can be powered down separately using CH*_TXPD and CH*_RXPD. The HSCLK*_RPLLPD port directly affects the RPLL while the HSCLK*_LCPLLPD port directly affects the LCPLL.

PLL Power Down

To activate the LCPLL power-down mode, the active-High HSCLK*_LCPLLPD signal is asserted. Similarly, to activate the RPLL power-down mode, the active-High HSCLK*_RPLLPD signal is asserted. When either HSCLK*_LCPLLPD or HSCLK*_RPLLPD is asserted, the corresponding PLL is powered down. As a result, all clocks derived from the respective PLL are stopped.

Recovery from this power state is indicated by the assertion of the corresponding PLL lock signal that is either the HSCLK*_LCPLLLOCK signal of the LCPLL or the HSCLK*_RPLLLOCK signal of the RPLL.

TX and RX Power Down

When the TX and RX power control signals are used in non-PCI Express implementations, CH*_TXPD and CH*_RXPD can be used independently. Also, when these interfaces are used in non-PCI Express applications, only two power states are supported, as shown in the following table. When using this power-down mechanism, these must be true:

- CH*_TXPD[1] and CH*_TXPD[0] are connected together.
- CH*_RXPD[1] and CH*_RXPD[0] are connected together.
- CH*_TXDETECTRX must be tied Low.
- CH*_TXELECIDLE must be tied to CH*_TXPD[1] and CH*_TXPD[0].

Table 28: **TX and RX Power States for Operation in Non-PCI Express Designs**

CH*_TXPD[1:0] or CH*_RXPD[1:0]	Description
`00`	Normal mode. Transceiver TX or RX is actively sending or receiving data.
`11`	Power-down mode. Transceiver TX or RX is idle.

Power savings can be achieved by setting the CH*_TXPD and CH*_RXPD ports to `2'b11`, and setting TX_CLKMUX_EN and ILO_CLKMUX_EN to `1'b0`.

Ports and Attributes

The following table defines the power-down ports.

Table 29: **Power Down Ports**

Port	Direction	Clock Domain	Description
HSCLK[0/1]_LCPLLPD	Input	ASYNC	This active-High signal powers down the LCPLL.
HSCLK[0/1]_RPLLPD	Input	ASYNC	This active-High signal powers down the RPLL.
CH[0/1/2/3]_RXPD[1:0]	Input	RXUSRCLK	Powers down the RX lane according to the PCI Express encoding. In PCI Express mode, tie TXPD and RXPD 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.
CH[0/1/2/3]_RXPHDLYPD	Input	ASYNC	RX phase and delay alignment circuit power down. Tied High when a) RX buffer bypass is not in use; b) RXPD is asserted, or c) RXOUTCLKSEL is set to 3'b011 or 3'b100 but the reference clock is not connected. Tied Low during RX buffer bypass mode normal operation. 0: Power-up the RX phase and delay alignment circuit. 1: Power-down the RX phase and delay alignment circuit.
CH[0/1/2/3]_TXPD[1:0]	Input	TXUSRCLK	Powers down the TX lane according to the PCI Express encoding. 2'b00: P0 normal operation 2'b01: P0s low recovery time power down 2'b10: P1 longer recovery time, RecDet still on 2'b11: P2 lowest power state. Transition times between the power states are controlled by the following attributes: PD_TRANS_TIME_FROM_P2 PD_TRANS_TIME_NONE_P2 PD_TRANS_TIME_TO_P2

Table 29: **Power Down Ports** *(cont'd)*

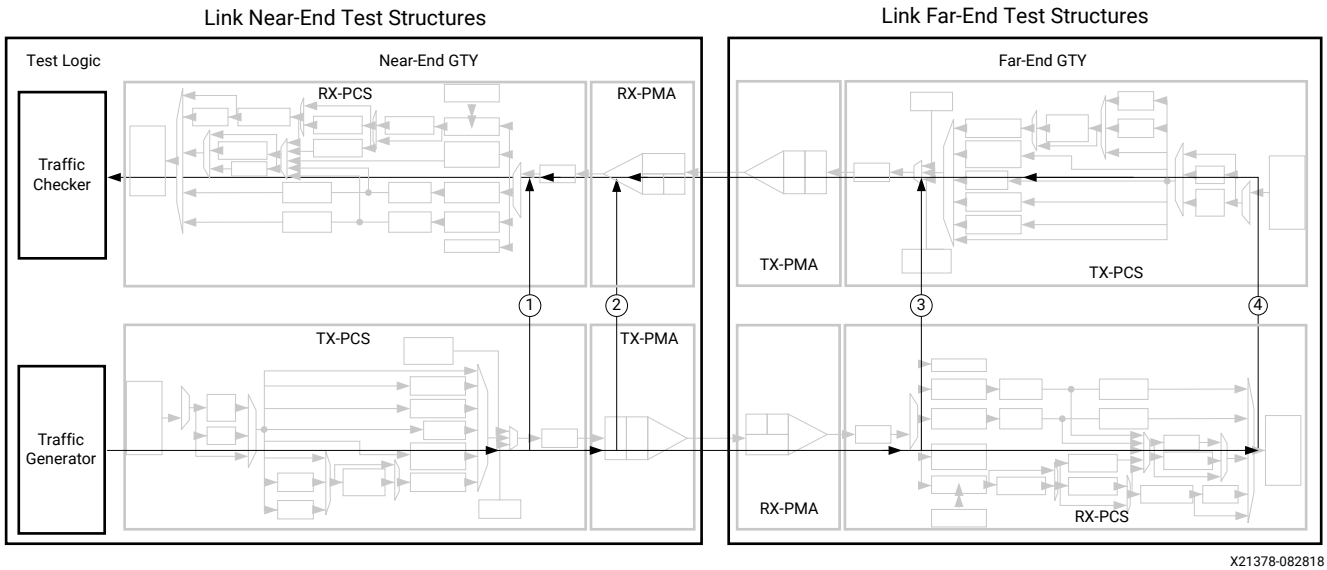
Port	Direction	Clock Domain	Description
CH[0/1/2/3]_TXPHDLYPD	Input	ASYNC	TX phase and delay alignment circuit power down. Tied High when a) TX buffer bypass is not in use; b) TXPD is asserted, or c) TXOUTCLKSEL is set to 3'b011 or 3'b100 but the reference clock is not connected. Tied Low during TX buffer bypass mode normal operation. 0: Power-up the TX phase and delay alignment circuit. 1: Power-down the TX phase and delay alignment circuit.

The are no power-down attributes in Versal ACAPs.

Loopback

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. The following figure illustrates a loopback test configuration with four different loopback modes.

Figure 30: **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. For Versal ACAP GTY transceivers, the serial data is still available on the TX differential output pairs.
- Far-end loopback modes loop received data back in the transceiver at the far end of the link. For Versal ACAP GTY transceivers, the receive data is visible on the RXDATA interface as in normal operation.

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 transceiver has a built-in PRBS generator and checker.

Ports and Attributes

The following table defines the loopback ports.

Table 30: **Loopback Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_LOOPBACK[2:0]	Input	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

There are no loopback attributes in Versal devices.

Fabric Configuration Interface

The fabric configuration interface (APB3) allows the user to dynamically update the attributes of the GTYE5_QUAD primitive. The APB3 interface is a processor-friendly synchronous interface with an address bus (APB3PADDR) and separated data buses for reading (APB3PRDATA) and writing (APB3PWDATA) configuration data to the primitive. An enable signal (APB3PENABLE), a read/write signal (APB3PWRITE), and a ready/valid signal (APB3PREADY) are the control signals that implement read and write operations, indicate operation completion, or indicate the availability of data.

Ports and Attributes

The following table shows the APB3 related ports.

Table 31: **APB3 Ports**

Port	Direction	Clock Domain	Description
APB3CLK	Input	CLOCK	This is the clock used for the APB3 interface.
APB3PADDR[15:0]	Input	APB3CLK	Configuration data address.
APB3PENABLE	Input	APB3CLK	Set to 1'b1 to enable read/write requests.
APB3PRDATA[31:0]	Output	APB3CLK	Data bus for reading configuration data from the transceiver to the interconnect logic resources.
APB3PREADY	Output	APB3CLK	Indicates operation is completed for write operations and data is valid for read operations.
APB3PRESETN	Input	ASYNC	Active low reset. This port resets the APB3 read/write access logic.
APB3PSEL	Input	APB3CLK	The APB master generates this signal to each bus slave. It indicates that the slave device is selected and that a data transfer is required.
APB3PSLVERR	Output	APB3CLK	Indicates slave error when invalid address or when request is unable to complete.
APB3PWDATA[31:0]	Input	APB3CLK	Data bus for writing configuration data from the interconnect logic resources to the transceiver.
APB3PWRITE	Input	APB3CLK	Write Enable. 0: Set to 0 for read request. 1: Set to 1 for write request.

Digital Monitor

The two receiver modes (LPM and DFE) use an adaptive algorithm in optimizing a link. The digital monitor provides visibility into the current state of these adaptation loops. Digital monitor requires a clock: RXUSRCLK can be used for this. The output port CH*_DMONITOROUT contains the current code(s) for a selected loop.

To enable digital monitor, DMON_EN must be driven High. When DMON_EN is driven Low, internal clock CH*_DMONITOROUTCLK from the digital monitor is gated, and the output port CH*_DMONITOROUT is driven to 0.

Ports and Attributes

The following table shows the digital monitor ports.

Table 32: **DMON Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_DMONITORCLK	Input	CLOCK	Digital monitor clock.
CH[0/1/2/3]_DMONITOROUT[31:0]	Output	ASYNC	Digital monitor output bus.
CH[0/1/2/3]_DMONITOROUTCLK	Output	ASYNC	Internal clock from digital monitor.

The following table shows the digital monitor attributes.

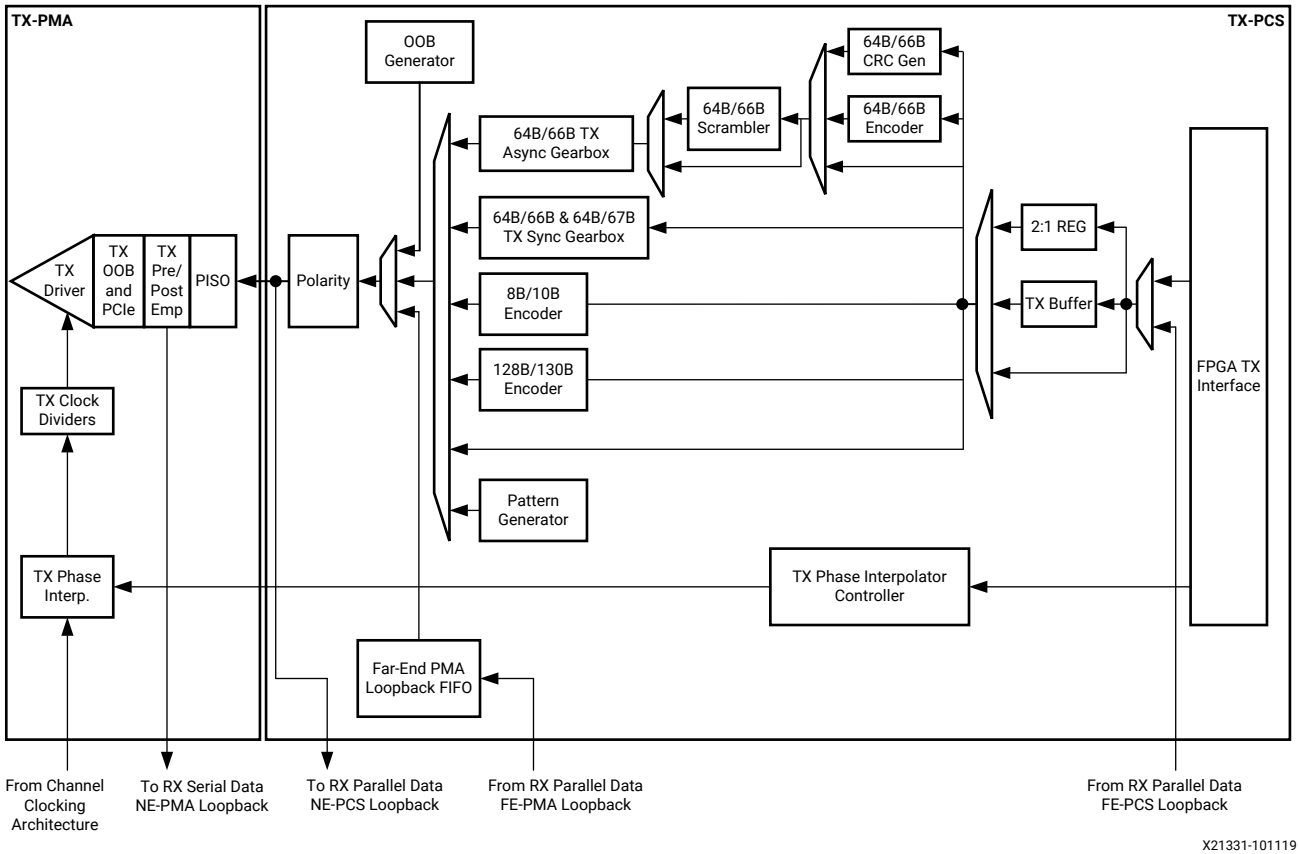
Table 33: **DMON Attributes**

DMON Attributes		
Attribute	**Address**	
CH0_MONITOR_CFG		0x0CBA
CH1_MONITOR_CFG		0x0DBA
CH2_MONITOR_CFG		0x0EBA
CH3_MONITOR_CFG		0x0FBA
Label	**Bit Field**	**Description**
DMON_CLK_SRC	[6:4]	Set to 0 when monitoring adaptation loops.
DMON_DATA_SRC	[3:1]	Set to 0 when monitoring adaptation loops.
DMON_EN	[0:0]	0: Disable digital monitor. 1: Enable digital monitor.

Transmitter

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. The following figure shows the functional blocks of the transmitter. Parallel data flows from the device logic into the TX interface, through the PCS and PMA, and then out the TX driver as high-speed serial data.

Figure 31: **Transceiver TX Block Diagram**



The key elements within the transceiver TX are:

1. TX Interface
2. TX 8B/10B Encoder
3. TX Synchronous Gearbox

4. TX Buffer
5. TX Buffer Bypass
6. TX Pattern Generator
7. TX Polarity Control
8. TX Fabric Clock Output Control
9. TX Phase Interpolator PPM Controller
10. TX Configurable Driver
11. TX Out-of-Band Signaling

TX Interface

The TX interface is the gateway to the TX datapath of the GTY transceiver. Applications transmit data through the GTY transceiver by writing data to the TXDATA port on the positive edge of TXUSRCLK. The width of the port can be configured to be two, four, or eight bytes wide. Port widths can be 16, 20, 32, 40, 64, 80, and 128. The total data width can be extended from eight byte to sixteen bytes wide by using the CH*_TXCTRL0 and CH*_TXCTRL1 ports together, which can provide 160 bits combined. The rate of the parallel clock TXUSRCLK at the interface is determined by the TX line rate, the width of the CH*_TXDATA port, and whether or not 8B/10B encoding is enabled. This section shows how to drive the parallel clock and explains the constraint on this clock for correct operation.

Interface Width Configuration

The GTY transceiver contains 2-byte, 4-byte, and 8-byte internal datapaths and is configurable by setting TX_INT_DATA_WIDTH. The interface width is configurable by setting TX_DATA_WIDTH. When the 8B/10B encoder is enabled, TX_DATA_WIDTH must be configured to 20, 40, or 80 bits, and in this case, the TX interface only uses the CH*_TXDATA ports. For example, CH*_TXDATA[15:0] is used when the interface width is 16. When the 8B/10B encoder is bypassed, TX_DATA_WIDTH can be configured to any of the available widths: 16, 20, 32, 40, 64, 80, 128, or 160 bits.

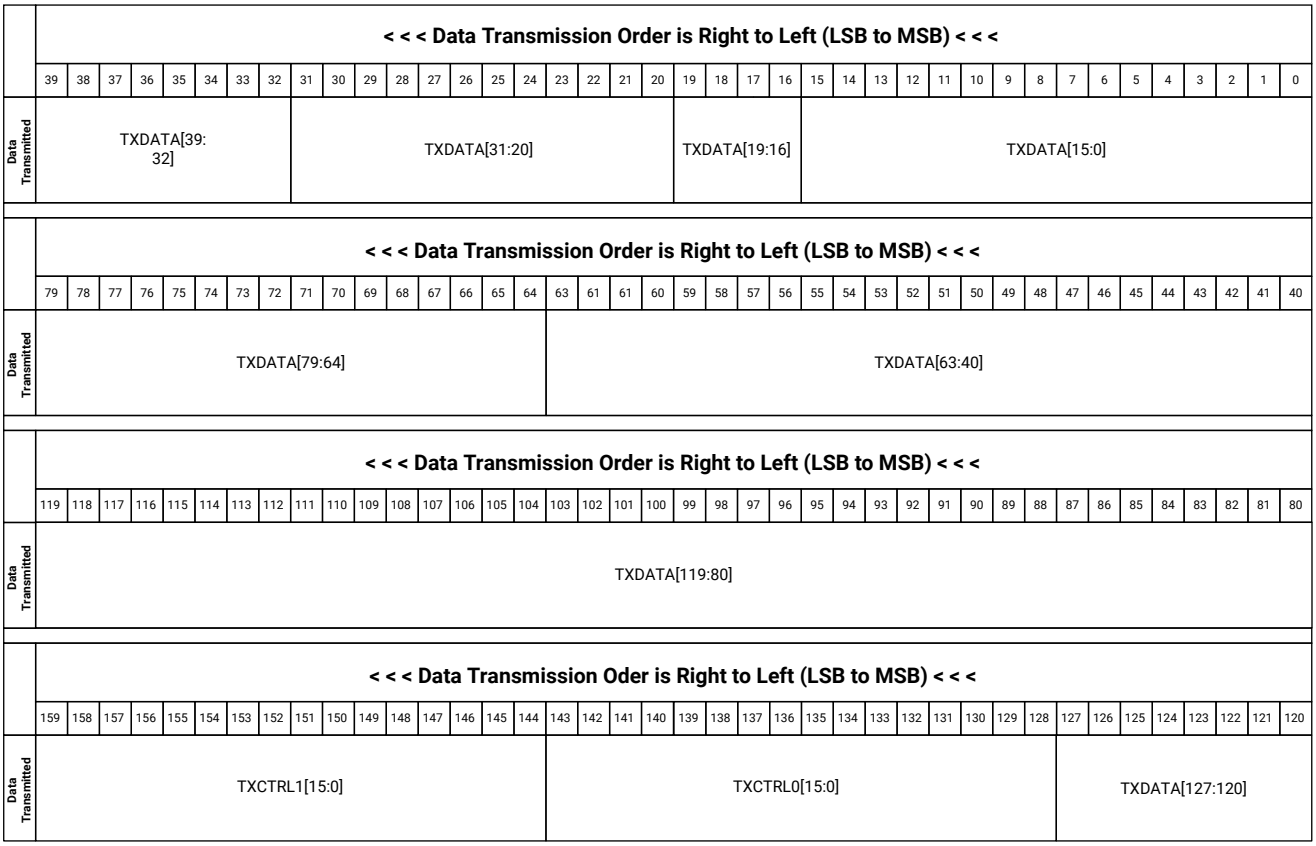
The following table shows how the interface width for the TX datapath is selected. 8B/10B encoding is described in more detail in TX 8B/10B Encoder.

Table 34: **TX Interface Datapath Configuration**

8B/10B	TX_DATA_WIDTH	TX_INT_DATA_WIDTH	Interface Width	Internal Data Width
Disabled	4'b0010	2'b00	16	16
	4'b0011	2'b00	20	20
	4'b0100	2'b00	32	16
	4'b0100	2'b01	32	32
	4'b0101	2'b00	40	20
	4'b0101	2'b01	40	40
	4'b0110	2'b01	64	32
	4'b0110	2'b10	64	64
	4'b0111	2'b01	80	40
	4'b0111	2'b10	80	80
	4'b1000	2'b10	128	64
	4'b1001	2'b10	160	80
Enabled	4'b0011	2'b00	16	20
	4'b0101	2'b00	32	20
	4'b0101	2'b01	32	40

When the 8B/10B encoder is bypassed and the TX_DATA_WIDTH is set to 160 bits, the CH*_TXCTRL1 and CH*_TXCTRL0 ports are used to extend the CH*_TXDATA port from 128 to 160 bits. The following figure shows the data transmitted when the 8B/10B encoder is disabled. When the TX gearbox is used, refer to TX Synchronous Gearbox for data transmission order.

Figure 32: **TX Data Transmitted When the 8B10B Encoder is Bypassed**



X21469-110119

TXUSRCLK Generation

The TX interface includes the parallel clock TXUSRCLK. The required rate for TXUSRCLK depends on the interface width of the GTYE5_QUAD primitive and the TX line rate of the GTY transmitter. Equation 7: TXUSRCLK shows how to calculate the required rate for TXUSRCLK for all cases except when the TX asynchronous gearbox is enabled.

Equation 7: **TXUSRCLK**

$$TXUSRCLK = \frac{Line\ Rate}{Interface\ Width}$$

TXUSRCLK is the internal synchronization clock for all signals into the TX side of the GTY transceiver. Most signals into the TX side of the GTY transceiver are sampled on the positive edge of TXUSRCLK.

Ports and Attributes

The following table defines the TX Interface ports.

Table 35: **TX Interface Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_TXCTRL0[15:0]	Input	TXUSRCLK	When using 8B/10B encoder: Works with TXCTRL1 to provide running disparity control. TXCTRL0[15:8] are unused. TXCTRL0[3] corresponds to TXDATA[31:24] TXCTRL0[2] corresponds to TXDATA[23:16] TXCTRL0[1] corresponds to TXDATA[15:8] TXCTRL0[0] corresponds to TXDATA[7:0] When using 128B/130B encoder: CH*_TXCTRL0[6]: this signal instructs the transceiver to ignore the TX data interface for one clock cycle. 1: CH*_TXDATA will be used. 0: CH*_TXDATA will be ignored. CH*_TXCTRL0[7]: this signal tells the transceiver the starting byte for a 128b block. CH*_TXCTRL0[5:4]: this signal will be used as the sync header for the next 130b block when CH*_TXCTRL0[7] is driven High. When 8B/10B encoding and 128B/130B encoding is disabled, TXCTRL0/1 is used to extend the data bus from 128-bit to 160-bit TX interfaces.
CH[0/1/2/3]_TXCTRL1[15:0]	Input	TXUSRCLK	When using 8B/10B encoder: Set High to work with TXCTRL0 to force running disparity negative or positive when encoding TXDATA. Set Low to use normal running disparity. TXCTRL1[15:8] are unused. TXCTRL1[3] corresponds to TXDATA[31:24] TXCTRL1[2] corresponds to TXDATA[23:16] TXCTRL1[1] corresponds to TXDATA[15:8] TXCTRL1[0] corresponds to TXDATA[7:0] When 8B/10B encoding and 128B/130B encoding is disabled, TXCTRL0/1 is used to extend the data bus from 128-bit to 160-bit TX interfaces.

Table 35: **TX Interface Ports** *(cont'd)*

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_TXCTRL2[7:0]	Input	TXUSRCLK	When using 8B/10B encoder: When High, indicates the corresponding data byte on TXDATA is a valid K character. TXCTRL2[3] corresponds to TXDATA[31:24] TXCTRL2[2] corresponds to TXDATA[23:16] TXCTRL2[1] corresponds to TXDATA[15:8] TXCTRL2[0] corresponds to TXDATA[7:0] A TXCTRL2 bit should be driven Low when the corresponding data byte from TXDATA is set to bypass the 8B/10B encoder.
CH[0/1/2/3]_TXDATA[127:0]	Input	TXUSRCLK	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 TX_DATA_WIDTH = 64, 80: TXDATA[63:0] = 64 bits wide TX_DATA_WIDTH = 128, 160: TXDATA[127:0] = 128 bits wide When a 160-bits bus is required, the TXCTRL0 and TXCTRL1 Port is concatenated with the TXDATA port to extend the 128-bits to 160-bits.
CH[0/1/2/3]_TXUSRCLK	Input	CLOCK	This port is used to provide a clock for the internal PCS datapath.

The following table defines the TX interface attributes.

Table 36: **TX Interface Attributes**

TX Interface Attributes		
Attribute	**Address**	
CH0_TX_PCS_CFG0		0x0C77
CH1_TX_PCS_CFG0		0x0D77
CH2_TX_PCS_CFG0		0x0E77
CH3_TX_PCS_CFG0		0x0F77
Label	**Bit Field**	**Description**
TX_INT_DATA_WIDTH	[25:24]	Controls the width of the internal datapath. 2'b00: 2-byte internal datapath 2'b01: 4-byte internal datapath 2'b10: 8-byte internal datapath

Table 36: **TX Interface Attributes** *(cont'd)*

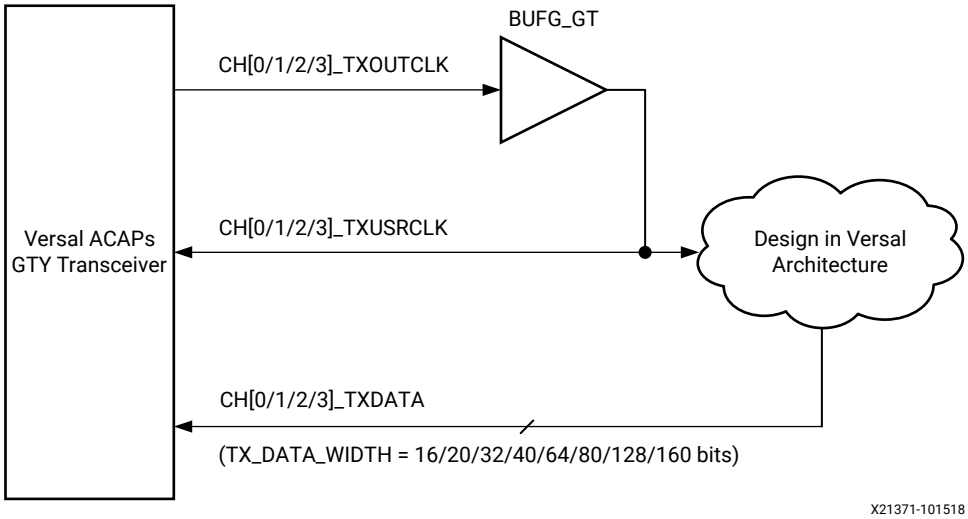
TX Interface Attributes		
TX_DATA_WIDTH	[23:20]	Sets the bit width of the CH*_TXDATA port. When 8B/10B encoding is enabled, RX_DATA_WIDTH must be set to 20, 40 or 80. Valid settings are 16, 20, 32, 40, 64, and 80. 4'0010: 16-bit 4'0011: 20-bit 4'0100: 32-bit 4'0101: 40-bit 4'0110: 64-bit 4'0111: 80-bit 4'1000: 128-bit 4'1001: 160-bit

Driving the TX Interface

Depending on the TXUSRCLK frequency, there are different ways Versal architecture clock resources can be used to drive the parallel clock for the TX interface. Figure 33 shows that TXOUTCLK is from the PMA, and the TXOUTCLKCTL = `3'b010` to select the TXPHYCLK path as indicated in TX Fabric Clock Output Control.

- Depending on the input reference clock frequency and the required line rate, a BUFG_GT with the appropriate TXOUTCLKCTL setting is required. The Versal ACAPs Transceivers Wizard creates a sample design based on different design requirements for most cases.
- In use models where the TX buffer is bypassed, there are additional restrictions on the clocking resources. Refer to TX Buffer Bypass for more information.

Figure 33: **TXOUTCLK Drives TXUSRCLK**



TX 8B/10B Encoder

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 GTY transceiver has a built-in 8B/10B TX path to encode TX data without consuming device 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.

TX 8B/10B Bit and Byte Ordering

The order of the bits after the 8B/10B encoder is the opposite of the order shown in Chapter 5: 8B/10B Valid Characters, because 8B/10B encoding requires bit a0 to be transmitted first, and the GTY transceiver always transmits the right-most bit first. To match with 8B/10B, the 8B/10B encoder in the GTY transceiver automatically reverses the bit order. The 8B/10B encoder does not support TX_DATA_WIDTH = 80 or 160. TX_INT_DATAWIDTH must be set to 0 (2-byte internal datapath) or 1 (4-byte internal datapath). To match with 8B/10B, the 8B/10B encoder in the GTY transceiver automatically reverses the bit order.

The number of bits used by CH*_TXDATA and corresponding byte orders are determined by TX_DATA_WIDTH.

- Only use CH*_TXDATA[15:0] if TX_DATA_WIDTH = 20
- Only use CH*_TXDATA[31:0] if TX_DATA_WIDTH = 40

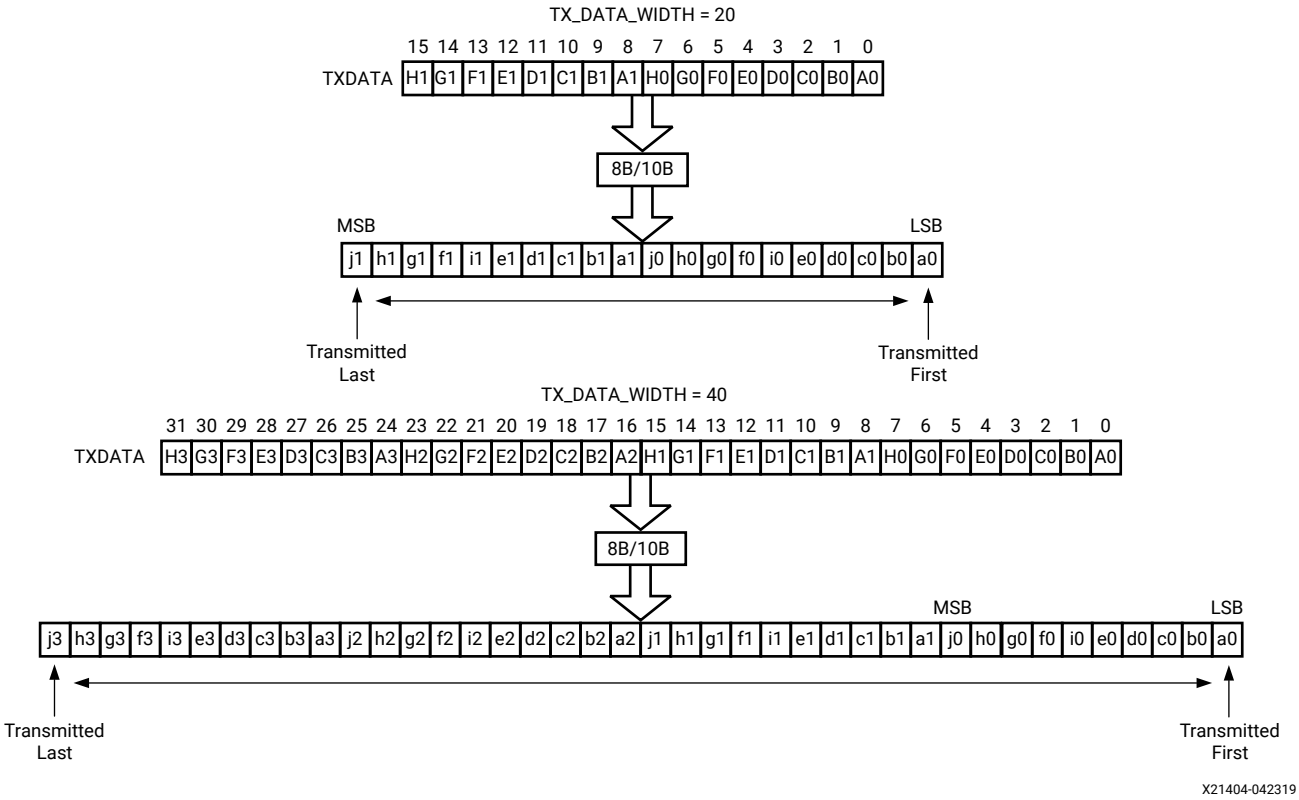
When the 8B/10B encoder is bypassed and TX_DATA_WIDTH is set to a multiple of 10, 10-bit characters are passed to TX data interface with this format:

- The corresponding TXDATA byte represents the [7:0] bits
- TXDATA[9:8] represents the 9th and 8th bits, respectively

K Characters

The 8B/10B table includes special characters (K characters) that are often used for control functions. CH*_TXCTRL2 ports are used to indicate if data on TXDATA are K characters or regular data. The 8B/10B encoder checks received CH*_TXDATA bytes to match any K character if corresponding CH*_TXCTRL2 bit is driven High.

Figure 34: **8B10B 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 can not only be generated by the 8B/10B encoder, but it is also controllable through CH*_TXCTRL1 and CH*_TXCTRL0, as shown in the following table. For example, an idle character sent with reversed disparity might be used to trigger clock correction.

Table 37: **TXCTRL0 and TXCTRL1 versus Outgoing Disparity**

CH*_TXCTRL0	CH*_TXCTRL1	Outgoing Disparity
0	0	Calculated by the 8B/10B encoder.
1	0	Inverts running disparity when encoding TXDATA.
0	1	Forces running disparity negative when encoding CH*_TXDATA.
1	1	Forces running disparity positive when encoding CH*_TXDATA.

Enabling and Disabling 8B/10B Encoding

To enable the 8B/10B encoder, TX_8B10B_EN must be driven High. The TX 8B/10B encoder allows byte interleaved data to bypass the encoder on a per-byte basis. When TX_8B10B_EN is driven Low, all encoders are turned off and no data from CH*_TXDATA can be encoded. When TX_8B10B_EN is High, driving a bit from TX_8B10B_BYPASS 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 TX interface.

Ports and Attributes

The following table lists the ports required by the TX 8B/10B encoder.

Table 38: **8B10BEN Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_TXCTRL0[15:0]	Input	TXUSRCLK	When using 8B/10B encoder: Works with TXCTRL1 to provide running disparity control. TXCTRL0[15:8] are unused. TXCTRL0[3] corresponds to TXDATA[31:24] TXCTRL0[2] corresponds to TXDATA[23:16] TXCTRL0[1] corresponds to TXDATA[15:8] TXCTRL0[0] corresponds to TXDATA[7:0] When using 128B/130B encoder: CH*_TXCTRL0[6]: this signal instructs the transceiver to ignore the TX data interface for one clock cycle. 1: CH*_TXDATA will be used. 0: CH*_TXDATA will be ignored. CH*_TXCTRL0[7]: this signal tells the transceiver the starting byte for a 128b block. CH*_TXCTRL0[5:4]: this signal will be used as the sync header for the next 130b block when CH*_TXCTRL0[7] is driven High. When 8B/10B encoding and 128B/130B encoding is disabled, TXCTRL0/1 is used to extend the data bus from 128-bit to 160-bit TX interfaces.

Table 38: **8B10BEN Ports** *(cont'd)*

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_TXCTRL1[15:0]	Input	TXUSRCLK	When using 8B/10B encoder: Set High to work with TXCTRL0 to force running disparity negative or positive when encoding TXDATA. Set Low to use normal running disparity. TXCTRL1[15:8] are unused. TXCTRL1[3] corresponds to TXDATA[31:24] TXCTRL1[2] corresponds to TXDATA[23:16] TXCTRL1[1] corresponds to TXDATA[15:8] TXCTRL1[0] corresponds to TXDATA[7:0] When 8B/10B encoding and 128B/130B encoding is disabled, TXCTRL0/1 is used to extend the data bus from 128-bit to 160-bit TX interfaces.
CH[0/1/2/3]_TXCTRL2[7:0]	Input	TXUSRCLK	When using 8B/10B encoder: When High, indicates the corresponding data byte on TXDATA is a valid K character. TXCTRL2[3] corresponds to TXDATA[31:24] TXCTRL2[2] corresponds to TXDATA[23:16] TXCTRL2[1] corresponds to TXDATA[15:8] TXCTRL2[0] corresponds to TXDATA[7:0] A TXCTRL2 bit should be driven Low when the corresponding data byte from TXDATA is set to bypass the 8B/10B encoder.

The following table lists the attributes required by the TX 8B/10B encoder.

Table 39: **8B10BEN Attributes**

8B10BEN Attributes		
Attribute	**Address**	
CH0_TX_PCS_CFG0		0x0C77
CH1_TX_PCS_CFG0		0x0D77
CH2_TX_PCS_CFG0		0x0E77
CH3_TX_PCS_CFG0		0x0F77
Label	**Bit Field**	**Description**
TX_DATA_WIDTH	[23:20]	Sets the bit width of the CH*_TXDATA port. When 8B/10B encoding is enabled, RX_DATA_WIDTH must be set to 20, 40 or 80. Valid settings are 16, 20, 32, 40, 64, and 80. 4'0010: 16-bit 4'0011: 20-bit 4'0100: 32-bit 4'0101: 40-bit 4'0110: 64-bit 4'0111: 80-bit 4'1000: 128-bit 4'1001: 160-bit

Table 39: **8B10BEN Attributes** *(cont'd)*

8B10BEN Attributes		
TX_8B10B_EN	[17:17]	8B10B encoder enable. 0: disable 8B10B encoder. 1: enable 8B10B encoder.
TX_8B10B_BYPASS	[16:13]	8B10B encoder bypass control, one bit per byte. 0: do not bypass encoder (normal operation). 1: bypass encoder for the byte. CH*_TX_PCS_CFG0[16] corresponds to CH*_TXDATA[31:24] CH*_TX_PCS_CFG0[15] corresponds to CH*_TXDATA[23:16] CH*_TX_PCS_CFG0[14] corresponds to CH*_TXDATA[15:8] CH*_TX_PCS_CFG0[13] corresponds to CH*_TXDATA[7:0]

TX 128B/130B Encoder

128B/130B is an industry standard encoding scheme, and protocols like PCI Express® use 128B/130B encoding on outgoing data when a PCI Express link is operating at a data rate of 8.0 GT/s or higher. The GTY transceiver has a built-in 128B/130B TX path to encode TX data without consuming device resources.

Ports and Attributes

The following table lists the ports required by the TX 128B/130B encoder.

Table 40: **128B130BEN Ports**

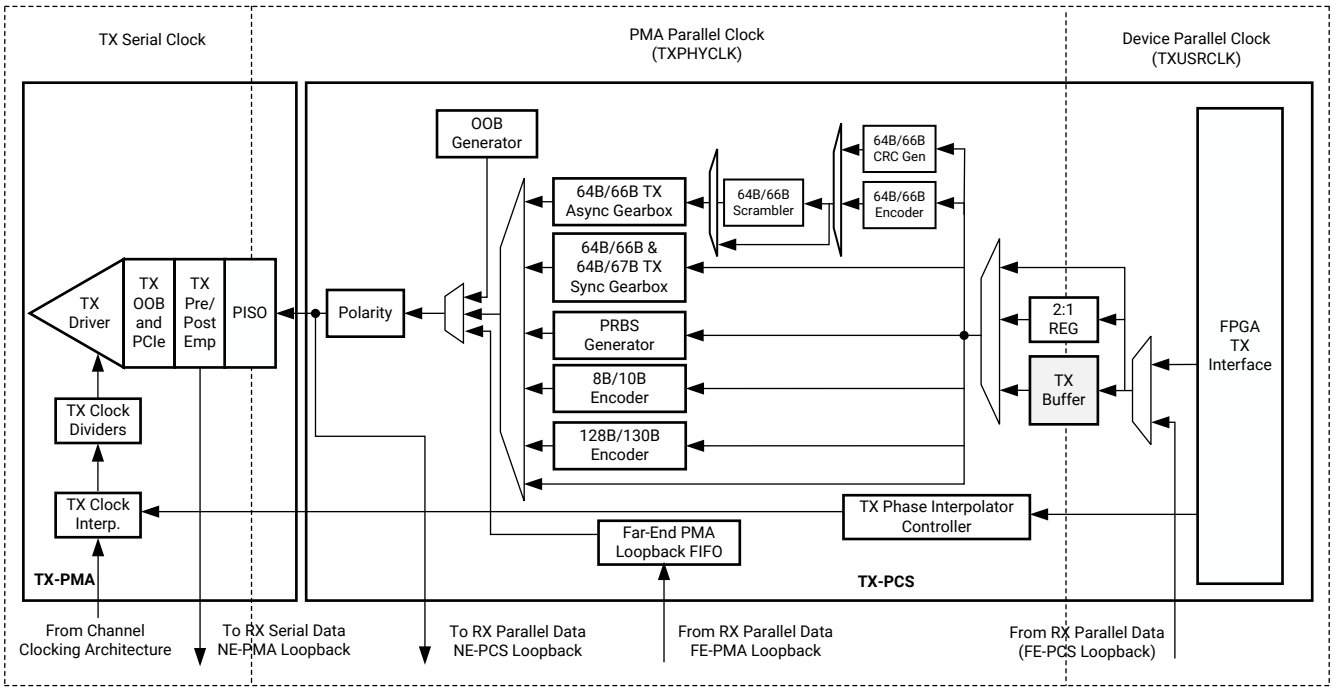
Port	Direction	Clock Domain	Description
CH[0/1/2/3]_TXCTRL0[15:0]	Input	TXUSRCLK	When using 8B/10B encoder: Works with TXCTRL1 to provide running disparity control. TXCTRL0[15:8] are unused. TXCTRL0[3] corresponds to TXDATA[31:24] TXCTRL0[2] corresponds to TXDATA[23:16] TXCTRL0[1] corresponds to TXDATA[15:8] TXCTRL0[0] corresponds to TXDATA[7:0] When using 128B/130B encoder: CH*_TXCTRL0[6]: this signal instructs the transceiver to ignore the TX data interface for one clock cycle. 1: CH*_TXDATA will be used. 0: CH*_TXDATA will be ignored. CH*_TXCTRL0[7]: this signal tells the transceiver the starting byte for a 128b block. CH*_TXCTRL0[5:4]: this signal will be used as the sync header for the next 130b block when CH*_TXCTRL0[7] is driven High. When 8B/10B encoding and 128B/130B encoding is disabled, TXCTRL0/1 is used to extend the data bus from 128-bit to 160-bit TX interfaces.

***Note*:** There are no TX 128B/130B encoder attributes.

TX Buffer

In the transceiver TX datapath, the TX buffer acts as a buffer between two clock domains: the fabric (TXUSRCLK), and the PCS parallel clock (TXPHYCLK). To transmit data, the TXPHYCLK rate must match the TXUSRCLK rate, and all phase differences between the two domains must be resolved. The following figure shows the TXUSRCLK and TXPHYCLK domains. Additionally, the TX buffer supports 2:1 data width conversion.

Figure 35: **TX Clock Domains**



X21389-101118

The transmitter includes a TX buffer and a TX phase alignment circuit to resolve phase differences between the TXPHYCLK and TXUSRCLK domains. The TX phase alignment circuit is used when the TX buffer is bypassed (see TX Buffer Bypass). All TX datapaths must use either the TX buffer or the TX phase-alignment circuit. The following table shows trade-offs between buffering and phase alignment.

Table 41: **TX Buffering versus Phase Alignment**

Parameter	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 GTY transceiver reference clock as the source of CH*_TXOUTCLK to drive TXUSRCLK.
Latency	If low latency is critical, the TX buffer must be bypassed.	Phase alignment uses fewer registers 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 GTY transceivers. All GTY transceivers involved must use the same line rate.
TXUSRCLK jitter sensitivity	No sensitivity to TXUSRCLK jitter.	Sensitive to TXUSRCLK jitter.

Using the TX Buffer

Reset the TX buffer whenever CH*_TXBUFSTATUS indicates an overflow or underflow condition. The TX buffer can be reset by using the TX reset procedure or PCS component reset described in Transceiver TX Component Reset. This setting is used to enable the TX buffer to resolve phase differences between the PHYCLK and TXUSRCLK domains:

- TX_PHASE_BUFFER_USE = `1'b1`

Ports and Attributes

The following table defines the TX buffer ports.

Table 42: **TX Buffer Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_TXBUFSTATUS[1:0]	Output	ASYNC	TXBUFSTATUS provides status for the TX Buffer and the port status is as follows: Bit[1] = 1'b1: TX FIFO has overflow or underflow error. Bit[1] = 1'b0: TX FIFO does not have overflow or underflow error. Bit[0] = 1'b1: TX FIFO is at least half full. Bit[0] = 1'b0: TX FIFO is less than half full.

The following table defines the TX buffer attributes.

Table 43: **TX Buffer Attributes**

TX Buffer Attributes		
Attribute	**Address**	
CH0_TX_PCS_CFG3		0x0C7A
CH1_TX_PCS_CFG3		0x0D7A
CH2_TX_PCS_CFG3		0x0E7A
CH3_TX_PCS_CFG3		0x0F7A
Label	**Bit Field**	**Description**
TXBUF_ADDR_CFG	[21:21]	TX Phase Buffer address config mode select
TX_PHASE_BUFFER_USE	[20:20]	Enable control for TX Phase Compensation FIFO
Attribute	**Address**	
CH0_TX_PHALIGN_CFG0		0x0C97
CH1_TX_PHALIGN_CFG0		0x0D97
CH2_TX_PHALIGN_CFG0		0x0E97
CH3_TX_PHALIGN_CFG0		0x0F97

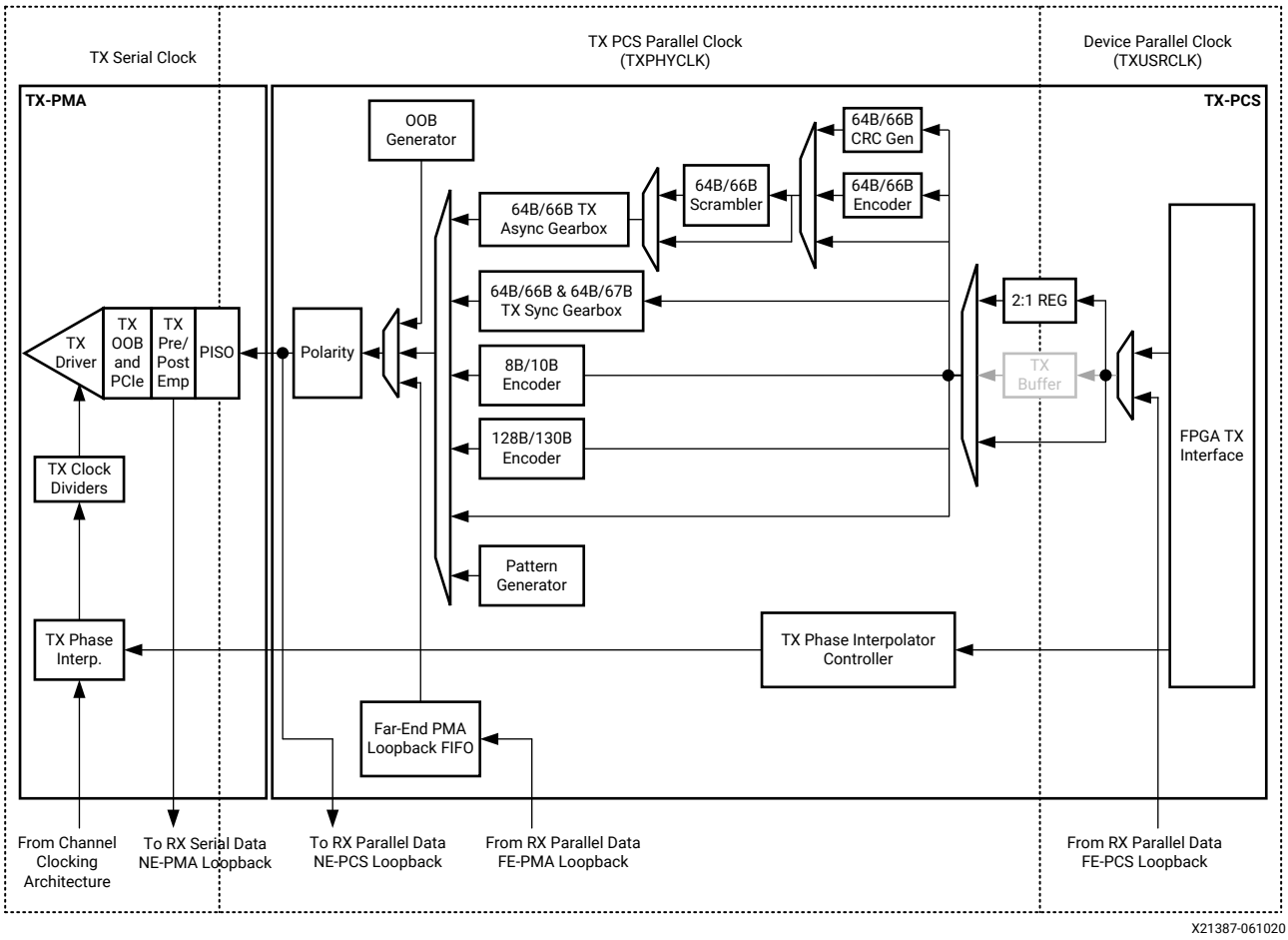
Table 43: **TX Buffer Attributes** *(cont'd)*

TX Buffer Attributes		
Label	**Bit Field**	**Description**
TXBUF_BYPASS_MODE	[14:14]	Buffer bypass enable. Set to 1'b0 when buffer bypass is not used. Set to 1'b1 when buffer bypass is used.

TX Buffer Bypass

The TX phase alignment circuit is used to adjust the phase difference between the PCS parallel clock domain (PHYCLK) and the TXUSRCLK domain when the TX buffer is bypassed. It also performs the TX delay alignment by continuously adjusting the TXUSRCLK to compensate for the temperature and voltage variations. The combined TX phase and delay alignments can be automatically performed by the GTY transceiver. Refer to Table 41 for trade-offs between buffering and phase alignment. The following figure shows how TX phase alignment allows the TX buffer to be bypassed. Before TX phase alignment, there is no guaranteed phase relationship between the PCS parallel clock domain (PHYCLK) and the TXUSRCLK domain.

Figure 36: **TX Buffer Bypass**



***Note*:** In order to use multi-lane buffer bypass, the Quad placement must be contiguous.

Ports and Attributes

The following table defines the TX buffer bypass ports.

Table 44: **TX Buffer Bypass Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_TXPHDLYPD	Input	ASYNC	TX phase and delay alignment circuit power down. Tied High when a) TX buffer bypass is not in use; b) TXPD is asserted, or c) TXOUTCLKSEL is set to 3'b011 or 3'b100 but the reference clock is not connected. Tied Low during TX buffer bypass mode normal operation. 0: Power-up the TX phase and delay alignment circuit. 1: Power-down the TX phase and delay alignment circuit.

Table 44: **TX Buffer Bypass Ports** *(cont'd)*

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_TXPHDLYRESET	Input	ASYNC	This reset is used to initiate the TX Phase alignment process. Assertion of this signal stops all ongoing phase and delay alignment process. De-assertion of this signal will start the auto alignment process between USRCLK and XCLK.
CH[0/1/2/3]_TXPHDLYRESETDONE	Output	ASYNC	This signal indicates that the auto phase alignment process request has been received and phase alignment has started.
CH[0/1/2/3]_TXSYNCALLIN	Input	ASYNC	This input is used for common clock buffer bypass where TXUSRCLK and RXUSRCLK is shared from the same source. Tie to 1'b0 in all other modes.
CH[0/1/2/3]_TXSYNCDONE	Output	ASYNC	Indicates TX buffer bypass procedure is complete. In multi-lane case, signal should be read from the lane designated as the initial master.

The following table defines the TX buffer bypass attributes.

Table 45: **TX Buffer Bypass Attributes**

TX Buffer Bypass Attributes		
Attribute	**Address**	
CH0_RX_PHALIGN_CFG5		0x0C96
CH1_RX_PHALIGN_CFG5		0x0D96
CH2_RX_PHALIGN_CFG5		0x0E96
CH3_RX_PHALIGN_CFG5		0x0F96
Label	**Bit Field**	**Description**
CMN_FAB_CLK_PHALIGN_MODE	[27:26]	Buffer Bypass Common Clock Mode 2'b00: Common clock disabled, or asynchronous gearbox enabled 2'b01: TX initial master common clock mode 2'b10: RX initial master common clock mode 2'b11: Reserved
Attribute	**Address**	
CH0_TX_PHALIGN_CFG0		0x0C97
CH1_TX_PHALIGN_CFG0		0x0D97
CH2_TX_PHALIGN_CFG0		0x0E97
CH3_TX_PHALIGN_CFG0		0x0F97
Label	**Bit Field**	**Description**
DLY_ALIGN_EN	[31:31]	Delay alignment enable. Set to 1'b0 when buffer bypass is not used. Set to 1'b1 when buffer bypass is used.
PH_ALIGN_EN	[30:30]	Phase alignment enable. Set to 1'b0 when buffer bypass is not used. Set to 1'b1 when buffer bypass is used.

Table 45: **TX Buffer Bypass Attributes** *(cont'd)*

TX Buffer Bypass Attributes		
SYNC_MODE	[17:16]	Multi-lane mode lane type select 0x0: Slave or Single Lane Mode 0x1: Initial Master 0x2: Maintenance Master
SYNC_MULTI_LANE	[15:15]	Multi-lane enable 1'b0: Single lane mode 1'b1: Multi lane mode
TXBUF_BYPASS_MODE	[14:14]	Buffer bypass enable. Set to 1'b0 when buffer bypass is not used. Set to 1'b1 when buffer bypass is used.
Attribute	**Address**	
CH0_TX_PHALIGN_CFG1		0x0C98
CH1_TX_PHALIGN_CFG1		0x0D98
CH2_TX_PHALIGN_CFG1		0x0E98
CH3_TX_PHALIGN_CFG1		0x0F98
Label	**Bit Field**	**Description**
CHAIN_MODE	[2:1]	Multi-lane daisy chain configuration 2'b00: Unused 2'b01: Top lane of multi-lane block 2'b10: Bottom lane of multi-lane block 2'b11: Middle lane(s) of multi-lane block
ASYNC_GBOX_PHALIGN_EN	[0:0]	Asynchronous Gearbox TX Buffer Bypass Enable 1'b0: Asynchronous gearbox not enabled 1'b1: Asynchronous gearbox enabled

TX Buffer Bypass Use Modes

TX phase alignment can be performed on one channel (single lane) or a group of channels sharing a single TXOUTCLK (multi-lane). See the following table for use modes.

Table 46: **TX Buffer Bypass Use Modes**

TX Buffer Bypass	TX Buffer Bypass Modes
Single-Lane	Auto
Multi-Lane	Auto

When the TX asynchronous gearbox is used, the exact use mode depends on the TX fabric interface data width and TX internal data width configuration. See the following table for buffer bypass with asynchronous gearbox applied to both single-lane and multi-lane use modes.

Table 47: **TX Buffer Bypass Use Modes with Asynchronous Gearbox**

TX Buffer Bypass with Asynchronous Gearbox Use Modes	TX Data Width Configuration	Note
1:1 Mode	Fabric Data Width = Internal Data Width	Phase alignment procedure not required.
2:1 Mode	Fabric Data Width = 2 x Internal Data Width	Phase alignment procedure required.

TX Buffer Bypass in Single-Lane Auto Mode without Asynchronous Gearbox

Use these transceiver settings to bypass the TX buffer in single lane mode:

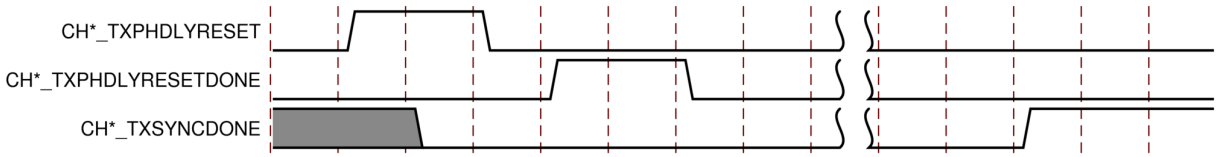
- CH*_TX_PHALIGN_CFG0[31] = `1'b1` (DLY_ALIGN_EN)
- CH*_TX_PHALIGN_CFG0[30] = `1'b1` (PH_ALIGN_EN)
- CH*_TX_PHALIGN_CFG0[17:16] = `2'b00` (SYNC_MODE)
- CH*_TX_PHALIGN_CFG0[15] = `1'b0` (SYNC_MULTI_LANE)
- CH*_TX_PHALIGN_CFG0[14] = `1'b1` (TXBUF_BYPASS_MODE)
- CH*_TX_PHALIGN_CFG1[2:1] = `2'b00` (CHAIN_MODE)
- CH*_TX_PHALIGN_CFG1[0] = `1'b0` (ASYNC_GBOX_PHALIGN_EN)
- CH*_PIPE_CTRL_CFG7[2:0] = `3'b011`, `3'b100`, `3'b101` (TXOUTCLKCTL) to select either the transceiver reference clock or the programmable divider clock as the source of TXOUTCLK

With the transceiver reference clock selected, TXOUTCLK is used as the source of TXUSRCLK. You must ensure that TXOUTCLK and the selected transceiver reference clock are operating at the desired frequency. When the TX buffer is bypassed, the TX phase alignment procedure must be performed after these conditions:

- Resetting or powering up the transceiver TX
- Resetting or powering up the RPLL and/or LCPLL
- Change of the transceiver reference clock source or frequency
- Change of the TX line rate

The following figure shows the required steps to perform the auto TX phase alignment and use the TX delay alignment to adjust TXUSRCLK to compensate for temperature and voltage variations.

Figure 37: **TX Buffer Bypass—Single-Lane Auto Mode**



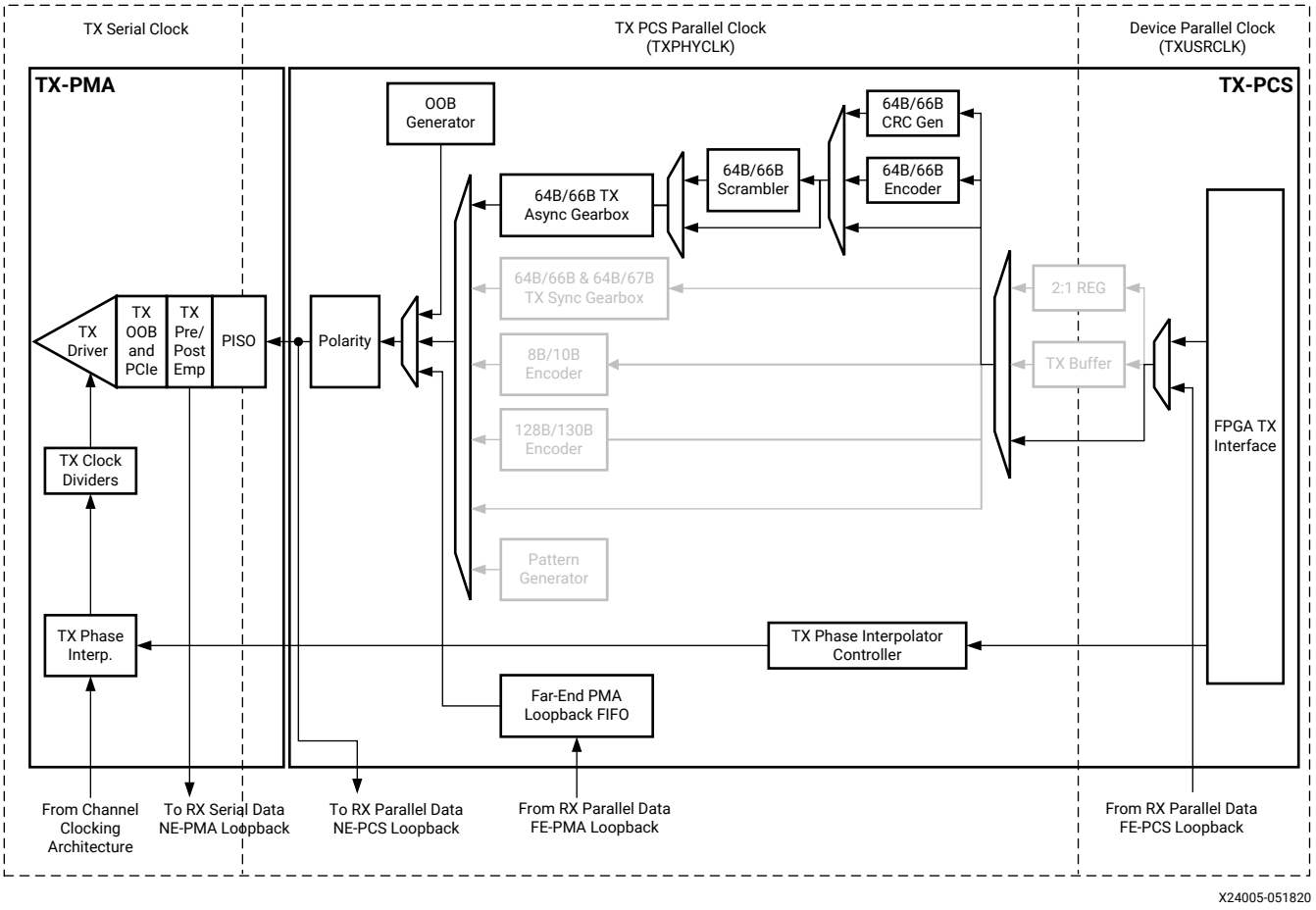
***Note*:**

1. The sequence of events in the figure is not drawn to scale.
2. After conditions such as a transmitter reset or TX rate change, TX phase alignment must be performed to align PHYCLK and TXUSRCLK. The TX phase and delay alignments are initiated by asserting CH*_TXPHDLYRESET.
3. TX phase alignment is done when the rising edge of CH*_TXSYNCDONE is detected. This signal should remain asserted until another alignment procedure is initiated.
4. An assertion/deassertion of GTTXRESET is required if CH*_TXSYNCDONE does not follow the sequence shown in the figure.
5. TX delay alignment continues to adjust TXUSRCLK to compensate for temperature and voltage variations.

TX Buffer Bypass in Single-Lane with Asynchronous Gearbox (1:1 Mode)

In single-lane with asynchronous gearbox mode, the TXUSRCLK domain is used to drive logic in the TXPHYCLK domain. When the TX internal data width is identical to the fabric interface data width, the TX buffer is directly bypassed, and clock phase compensation is provided by the asynchronous gearbox FIFO. In this mode, the latency is deterministic and the asynchronous gearbox FIFO can provide measured latency. For more details on how to read the asynchronous gearbox FIFO latency, refer to TX Asynchronous Gearbox.

Figure 38: **TX Buffer Bypass in Single-Lane with Asynchronous Gearbox 1:1 Mode**



Use these transceiver settings to bypass the TX buffer with asynchronous gearbox enabled in single-lane 1:1 mode:

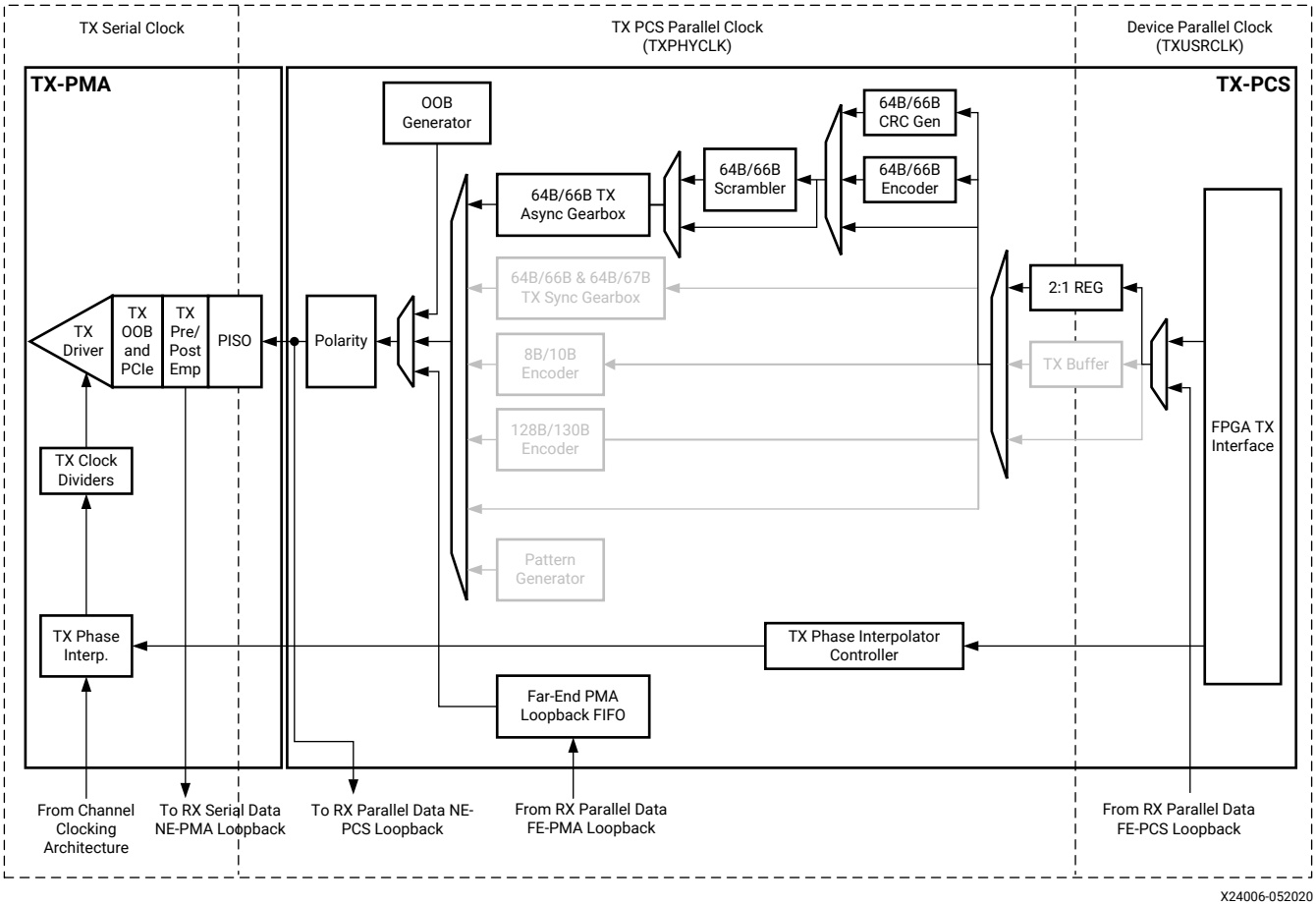
- CH*_TX_PCS_CFG0[5] = `1'b1` (USE_BG)
- CH*_TX_PHALIGN_CFG0[31] = `1'b0` (DLY_ALIGN_EN)
- CH*_TX_PHALIGN_CFG0[30] = `1'b0` (PH_ALIGN_EN)
- CH*_TX_PHALIGN_CFG0[17:16] = `2'b00` (SYNC_MODE)
- CH*_TX_PHALIGN_CFG0[15] = `1'b0` (SYNC_MULTI_LANE)
- CH*_TX_PHALIGN_CFG0[14] = `1'b1` (TXBUF_BYPASS_MODE)
- CH*_TX_PHALIGN_CFG1[2:1] = `2'b00` (CHAIN_MODE)
- CH*_TX_PHALIGN_CFG1[0] = `1'b1` (ASYNC_GBOX_PHALIGN_EN)
- CH*_PIPE_CTRL_CFG7[2:0] = `3'b011`, `3'b100`, or `3'b101` (TXOUTCLKCTL) to select either the transceiver reference clock or the programmable divider clock as the source of TXOUTCLK

In this particular use mode, because the asynchronous gearbox FIFO provides the phase compensation, there is no need to perform the TX phase alignment procedure as shown in Figure 37.

TX Buffer Bypass in Single-Lane with Asynchronous Gearbox (2:1 Mode)

In single-lane with asynchronous gearbox mode, the TXUSRCLK domain is used to drive logic in the TXPHYCLK domain. When the TX fabric interface data width is twice the internal data width, the TX buffer is bypassed through the 2:1 REG. The 2:1 REG requires the phase alignment procedure to be performed identical to Figure 37. In this use case, note that the latency is still deterministic because the continuous phase alignment at 2:1 REG does not impact the data latency through the data path.

Figure 39: **TX Buffer Bypass in Single-Lane with Asynchronous Gearbox 2:1 Mode**



Use these transceiver settings to bypass the TX buffer with asynchronous gearbox enabled in single-lane 2:1 mode:

- CH*_TX_PCS_CFG0[5] = `1'b1` (USE_BG)
- CH*_TX_PHALIGN_CFG0[31] = `1'b1` (DLY_ALIGN_EN)
- CH*_TX_PHALIGN_CFG0[30] = `1'b1` (PH_ALIGN_EN)
- CH*_TX_PHALIGN_CFG0[17:16] = `2'b00` (SYNC_MODE)
- CH*_TX_PHALIGN_CFG0[15] = `1'b0` (SYNC_MULTI_LANE)
- CH*_TX_PHALIGN_CFG0[14] = `1'b1` (TXBUF_BYPASS_MODE)
- CH*_TX_PHALIGN_CFG1[2:1] = `2'b00` (CHAIN_MODE)
- CH*_TX_PHALIGN_CFG1[0] = `1'b1` (ASYNC_GBOX_PHALIGN_EN)
- CH*_PIPE_CTRL_CFG7[2:0] = `3'b011`, `3'b100`, or `3'b101` (TXOUTCLKCTL) to select either the transceiver reference clock or the programmable divider clock as the source of TXOUTCLK

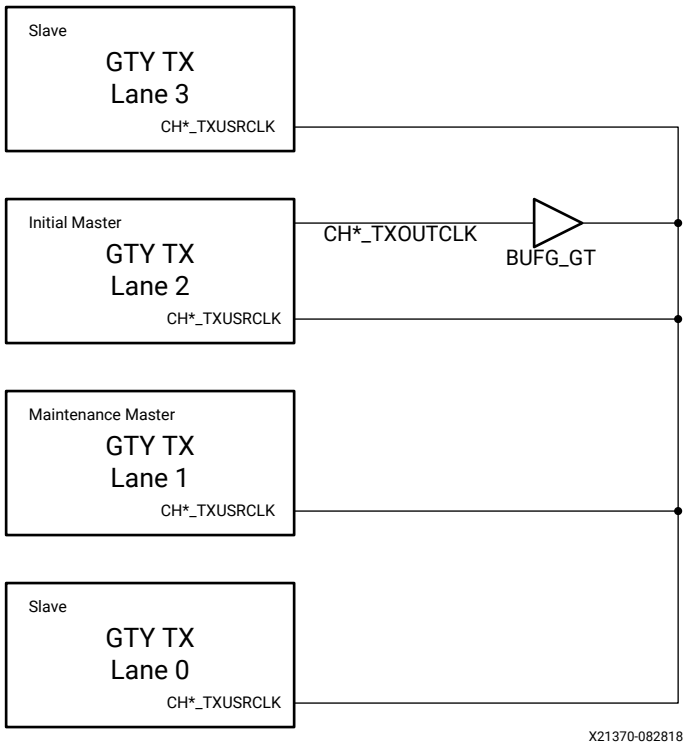
TX Buffer Bypass in Multi-Lane Auto Mode without Asynchronous Gearbox

When a multi-lane application requires TX buffer bypass, phase alignment is performed automatically. This section describes the steps required to perform the multi-lane TX buffer bypass alignment procedure automatically.

- Initial Master: In a multi-lane application, the buffer bypass initial master is the lane that is the source of the TXOUTCLK.
  - CH*_TX_PHALIGN_CFG0[17:16] = `2'b01` (SYNC_MODE)
- Maintenance Master: This lane shares the same TXUSRCLK generated from the TXOUCLK of the buffer bypass initial master. The maintenance master also provides delay skew information, which is forwarded internally to the initial master lane.
  - CH*_TX_PHALIGN_CFG0[17:16] = `2'b11` (SYNC_MODE)
- Slave: These are all the lanes that share the same TXUSRCLK, which is generated from the TXOUCLK of the buffer bypass initial master.
  - CH*_TX_PHALIGN_CFG0[17:16] = `2'b00` (SYNC_MODE)

The following figure shows an example of buffer bypass initial master, maintenance master, and slave lanes.

Figure 40: **TX Buffer Bypass Initial Master, Maintenance Master, and Slave Lanes**



Use these transceiver settings to bypass the TX buffer in multi-lane mode:

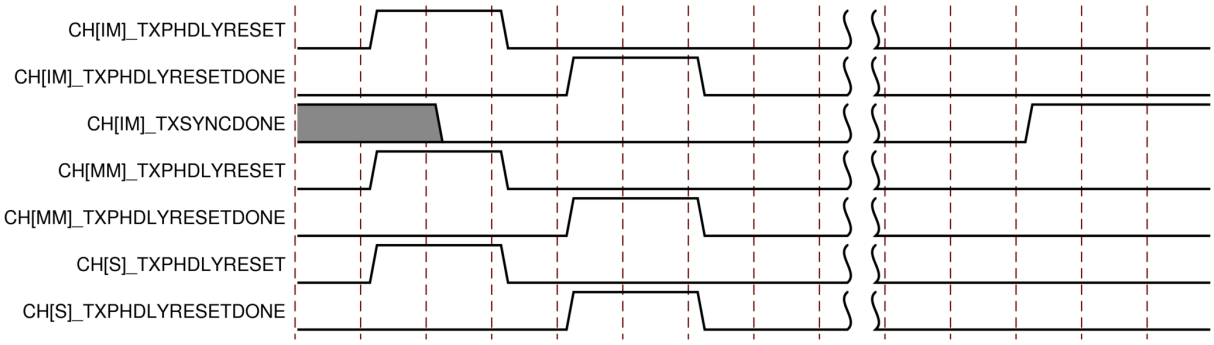
- CH*_TX_PHALIGN_CFG0[31] = `1'b1` (DLY_ALIGN_EN)
- CH*_TX_PHALIGN_CFG0[30] = `1'b1` (PH_ALIGN_EN)
- CH*_TX_PHALIGN_CFG0[15] = `1'b1` (SYNC_MULTI_LANE)
- CH*_TX_PHALIGN_CFG0[14] = `1'b1` (TXBUF_BYPASS_MODE)
- CH*_TX_PHALIGN_CFG1[0] = `1'b0` (ASYNC_GBOX_PHALIGN_EN)
- CH*_PIPE_CTRL_CFG7[2:0] = `3'b011`, `3'b100`, or `3'b101` (TXOUTCLKCTL) to select either the transceiver reference clock or the programmable divider clock as the source of TXOUTCLK.

Multi-lane buffer bypass must only be used on lanes that have physical locations that are directly adjacent to one another. CH*_TX_PHALIGN_CFG1[2:1] (CHAIN_MODE) must be set according to the physical location of the multi-lane group.

- Top location: CH*_TX_PHALIGN_CFG1[2:1] = `2'b01` (CHAIN_MODE)
- Middle location(s): CH*_TX_PHALIGN_CFG1[2:1] = `2'b11` (CHAIN_MODE)
- Bottom location: CH*_TX_PHALIGN_CFG1[2:1] = `2'b10` (CHAIN_MODE)

The following figure shows the required steps to perform auto TX phase and delay alignment.

Figure 41: **TX Buffer Bypass—Multi-Lane Auto Mode**



Notes relevant to the figure:

1. The sequence of events in the figure is not drawn to scale.
2. CH[IM]_* denotes ports related to the initial master lane.
3. CH[MM]_* denotes ports related to the maintenance master lane.
4. CH[S]_* denotes ports related to the slave lane(s).
5. After conditions such as a transmitter reset or TX rate change, TX phase alignment must be performed to align TXPHYCLK and TXUSRCLK. The TX phase and delay alignments are initiated by asserting CH*_TXPHDLYRESET.
6. TX phase alignment is done when the rising edge of CH[IM]_TXSYNCDONE is detected. This signal should remain asserted until another alignment procedure is initiated.
7. An assertion/deassertion of GTTXRESET is required if CH[IM]_TXSYNCDONE does not follow the sequence shown in the figure.
8. TX delay alignment continues to adjust TXUSRCLK to compensate for the temperature and voltage variations.

TX Buffer Bypass in Multi-Lane Auto Mode with Asynchronous Gearbox (1:1 Mode)

In multi-lane with asynchronous gearbox mode, the TXUSRCLK domain is used to drive logic in the TXPHYCLK domain. When the TX internal data width is identical to the fabric interface data width, the TX buffer is directly bypassed, and clock phase compensation is provided by the asynchronous gearbox FIFO. In this mode, the latency is deterministic, and the asynchronous gearbox FIFO can provide measured latency. For more details on how to read the asynchronous gearbox FIFO latency, refer to TX Asynchronous Gearbox.

In this particular use mode, because the asynchronous gearbox FIFO provides the phase compensation, there is no need to perform the TX phase alignment procedure.

- CH*_TX_PCS_CFG0[5] = `1'b1` (USE_BG)

- CH*_TX_PHALIGN_CFG0[31] = `1'b0` (DLY_ALIGN_EN)
- CH*_TX_PHALIGN_CFG0[30] = `1'b0` (PH_ALIGN_EN)
- CH*_TX_PHALIGN_CFG0[17:16] = `2'b00` (SYNC_MODE) (All lanes are configured as slaves in multi-lane asynchronous gearbox mode)
- CH*_TX_PHALIGN_CFG0[15] = `1'b1` (SYNC_MULTI_LANE)
- CH*_TX_PHALIGN_CFG0[14] = `1'b1` (TXBUF_BYPASS_MODE)
- CH*_TX_PHALIGN_CFG1[2:1] = `2'b00` (CHAIN_MODE)
- CH*_TX_PHALIGN_CFG1[0] = `1'b1` (ASYNC_GBOX_PHALIGN_EN)
- CH*_PIPE_CTRL_CFG7[2:0] = `3'b011`, `3'b100`, or `3'b101` (TXOUTCLKCTL) to select either the transceiver reference clock or the programmable divider clock as the source of TXOUTCLK

TX Buffer Bypass in Multi-Lane Auto Mode with Asynchronous Gearbox (2:1 Mode)

In multi-lane with asynchronous gearbox mode, the TXUSRCLK domain is used to drive logic in the TXPHYCLK domain. When the TX fabric interface data width is twice the internal data width, the TX buffer is bypassed through the 2:1 REG. The 2:1 REG requires the phase alignment procedure to be performed identical to Figure 41. In this use case, note that the latency is still deterministic because the continuous phase alignment at 2:1 REG does not impact the data latency through the data path.

- CH*_TX_PCS_CFG0[5] = `1'b1` (USE_BG)
- CH*_TX_PHALIGN_CFG0[31] = `1'b1` (DLY_ALIGN_EN)
- CH*_TX_PHALIGN_CFG0[30] = `1'b1` (PH_ALIGN_EN)
- CH*_TX_PHALIGN_CFG0[17:16] = `2'b00` (SYNC_MODE) (All lanes are configured as slaves in multi-lane asynchronous gearbox mode)
- CH*_TX_PHALIGN_CFG0[15] = `1'b1` (SYNC_MULTI_LANE)
- CH*_TX_PHALIGN_CFG0[14] = `1'b1` (TXBUF_BYPASS_MODE)
- CH*_TX_PHALIGN_CFG1[2:1] = `2'b00` (CHAIN_MODE)
- CH*_TX_PHALIGN_CFG1[0] = `1'b1` (ASYNC_GBOX_PHALIGN_EN)
- CH*_PIPE_CTRL_CFG7[2:0] = `3'b011`, `3'b100`, or `3'b101` (TXOUTCLKCTL) to select either the transceiver reference clock or the programmable divider clock as the source of TXOUTCLK

TXUSRCLK and RXUSRCLK Sharing Using Both TX and RX Buffer Bypass in Single-Lane Auto Mode with TX Master

Use these transceiver settings to bypass the TX and RX buffer in single-lane mode with a common clock, TX master.

Common clock settings:

- CH*_RX_PHALIGN_CFG5[27:26] = `2'b01` (CMN_FAB_CLK_PHALIGN_MODE)

TX buffer bypass settings:

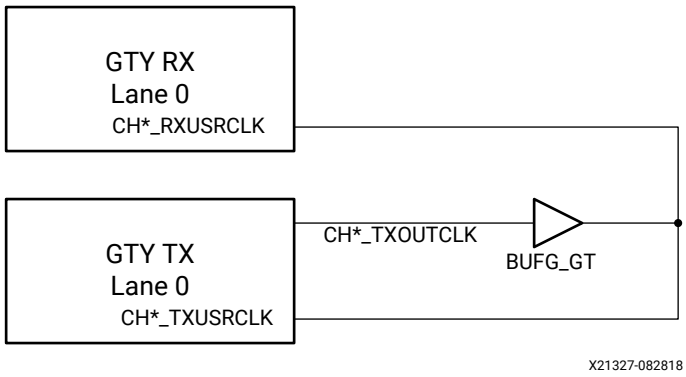
- CH*_TX_PHALIGN_CFG0[31] = `1'b1` (DLY_ALIGN_EN)
- CH*_TX_PHALIGN_CFG0[30] = `1'b1` (PH_ALIGN_EN)
- CH*_TX_PHALIGN_CFG0[17:16] = `2'b00` (SYNC_MODE)
- CH*_TX_PHALIGN_CFG0[15] = `1'b0` (SYNC_MULTI_LANE)
- CH*_TX_PHALIGN_CFG0[14] = `1'b1` (TXBUF_BYPASS_MODE)
- CH*_TX_PHALIGN_CFG1[2:1] = `2'b00` (CHAIN_MODE)
- CH*_TX_PHALIGN_CFG1[0] = `1'b0` (ASYNC_GBOX_PHALIGN_EN)
- CH*_PIPE_CTRL_CFG7[2:0] = `3'b011`, `3'b100`, `3'b101` (TXOUTCLKCTL) to select either the transceiver reference clock or the programmable divider clock as the source of TXOUTCLK.

RX buffer bypass settings:

- CH*_RX_PHALIGN_CFG0[31] = `1'b1` (DLY_ALIGN_EN)
- CH*_RX_PHALIGN_CFG0[30] = `1'b1` (PH_ALIGN_EN)
- CH*_RX_PHALIGN_CFG0[17:16] = `2'b00` (SYNC_MODE)
- CH*_RX_PHALIGN_CFG0[15] = `1'b0` (SYNC_MULTI_LANE)
- CH*_RX_PHALIGN_CFG0[14] = `1'b1` (RXBUF_BYPASS_MODE)
- CH*_RX_PHALIGN_CFG1[3:2] = `2'b00` (CHAIN_MODE)
- CH*_RX_PHALIGN_CFG1[1] = `1'b0` (ASYNC_GBOX_PHALIGN_EN)

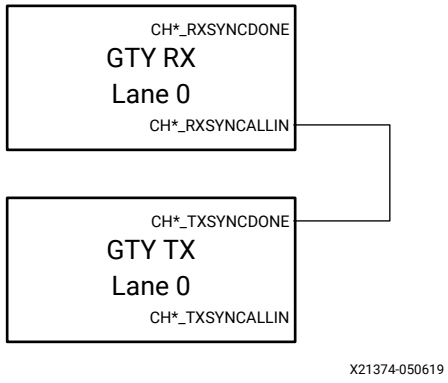
The following figure shows an example of a single-lane common clock buffer bypass with a TX master.

Figure 42: **Common Clock Buffer Bypass Lanes in Single-Lane TX Master Mode**



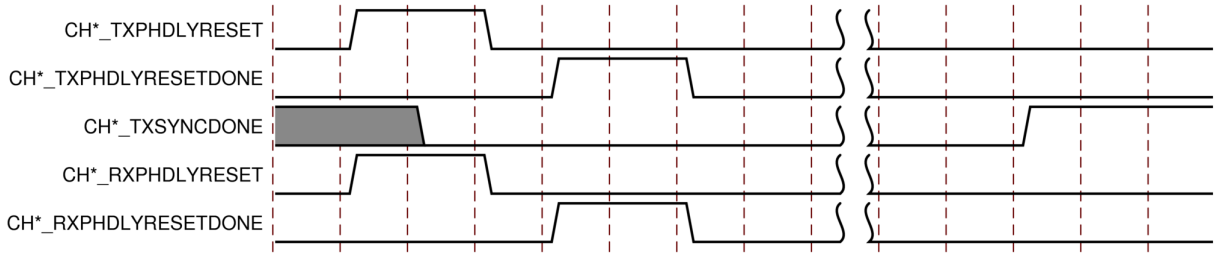
The following figure shows the port connection for a single-lane common clock buffer bypass with a TX master.

Figure 43: **Port Connection for Common Clock Buffer Bypass Lanes in Single-Lane TX Master Mode**



The following timing diagram shows the required steps to perform common clock phase and delay alignment in single-lane TX master mode.

Figure 44: **Common Clock Buffer Bypass - Single-Lane TX Master Mode**



Notes relevant to the figure:

1. The sequence of events in the figure is not drawn to scale.

2. After conditions such as a reset or rate change, common clock phase alignment must be performed. The common clock phase and delay alignments are initiated by asserting CH*_TXPHDLYRESET and CH*_RXPHDLYRESET.
3. Common clock phase alignment with TX master is done when the rising edge of CH*_TXSYNCDONE is detected. This signal should remain asserted until another alignment procedure is initiated.
4. An assertion/deassertion of GTTXRESET and GTRXRESET is required if CH*_TXSYNCDONE does not follow the sequence shown in the figure.
5. TX delay alignment continues to adjust TXUSRCLK to compensate for temperature and voltage variations.

TXUSRCLK and RXUSRCLK Sharing Using Both TX and RX Buffer Bypass in Single-Lane Auto Mode with RX Master

Use these transceiver settings to bypass the TX and RX buffer in single-lane mode with a common clock, RX master.

Common clock settings:

- CH*_RX_PHALIGN_CFG5[27:26] = `2'b10` (CMN_FAB_CLK_PHALIGN_MODE)

TX buffer bypass settings:

- CH*_TX_PHALIGN_CFG0[31] = `1'b1` (DLY_ALIGN_EN)
- CH*_TX_PHALIGN_CFG0[30] = `1'b1` (PH_ALIGN_EN)
- CH*_TX_PHALIGN_CFG0[17:16] = `2'b00` (SYNC_MODE)
- CH*_TX_PHALIGN_CFG0[15] = `1'b0` (SYNC_MULTI_LANE)
- CH*_TX_PHALIGN_CFG0[14] = `1'b1` (TXBUF_BYPASS_MODE)
- CH*_TX_PHALIGN_CFG1[2:1] = `2'b00` (CHAIN_MODE)
- CH*_TX_PHALIGN_CFG1[0] = `1'b0` (ASYNC_GBOX_PHALIGN_EN)
- CH*_PIPE_CTRL_CFG7[2:0] = `3'b011`, `3'b100`, `3'b101` (TXOUTCLKCTL) to select either the transceiver reference clock or the programmable divider clock as the source of TXOUTCLK.

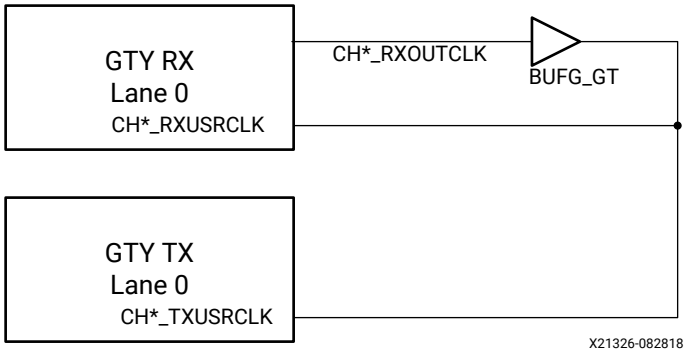
RX buffer bypass settings:

- CH*_RX_PHALIGN_CFG0[31] = `1'b1` (DLY_ALIGN_EN)
- CH*_RX_PHALIGN_CFG0[30] = `1'b1` (PH_ALIGN_EN)
- CH*_RX_PHALIGN_CFG0[17:16] = `2'b00` (SYNC_MODE)

- CH*_RX_PHALIGN_CFG0[15] = `1'b0` (SYNC_MULTI_LANE)
- CH*_RX_PHALIGN_CFG0[14] = `1'b1` (RXBUF_BYPASS_MODE)
- CH*_RX_PHALIGN_CFG1[2:1] = `2'b00` (CHAIN_MODE)
- CH*_RX_PHALIGN_CFG1[0] = `1'b0` (ASYNC_GBOX_PHALIGN_EN)
- CH*_PIPE_CTRL_CFG7[14:12] = `3'b011` to select TXOUTCLK as the source of RXOUTCLK.

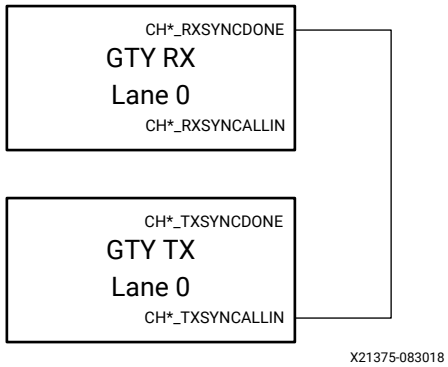
The following figure shows an example of a single-lane common clock buffer bypass with an RX master.

Figure 45: **Common Clock Buffer Bypass Lanes in Single-Lane RX Master Mode**



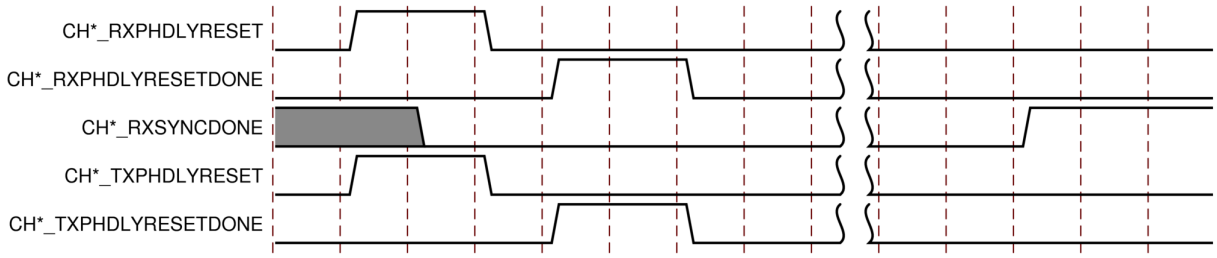
The following figure shows the port connection for a single-lane common clock buffer bypass with an RX master.

Figure 46: **Port Connection for Common Clock Buffer Bypass Lanes in Single-Lane RX Master Mode**



The following timing diagram shows the required steps to perform common clock phase and delay alignment in single-lane RX master mode.

Figure 47: **Common Clock Buffer Bypass - Single-Lane RX Master Mode**



Notes relevant to the figure:

1. The sequence of events in the figure is not drawn to scale.
2. After conditions such as a reset or rate change, common clock phase alignment must be performed. The common clock phase and delay alignments are initiated by asserting CH*_TXPHDLYRESET and CH*_RXPHDLYRESET.
3. Common clock phase alignment with the TX master is done when the rising edge of CH*_RXSYNCDONE is detected. This signal should remain asserted until another alignment procedure is initiated.
4. An assertion/deassertion of GTTXRESET and GTRXRESET is required if CH*_RXSYNCDONE does not follow the sequence shown in the figure.
5. RX delay alignment continues to adjust RXUSRCLK to compensate for temperature and voltage variations.

TXUSRCLK and RXUSRCLK Sharing Using Both TX and RX Buffer Bypass in Multi-Lane Auto Mode with TX Master

In multi-lane common clock buffer bypass with TX master, a TX lane must be designated as an initial master lane, and another as a maintenance master lane. All remaining TX lanes and all RX lanes must be designated as slave lanes.

Use these transceiver settings to bypass the TX and RX buffer in single-lane mode with a common clock, TX master.

Common clock settings:

- CH*_RX_PHALIGN_CFG5[27:26] = `2'b01` (CMN_FAB_CLK_PHALIGN_MODE)

TX buffer bypass settings:

- CH*_TX_PHALIGN_CFG0[31] = `1'b1` (DLY_ALIGN_EN)
- CH*_TX_PHALIGN_CFG0[30] = `1'b1` (PH_ALIGN_EN)

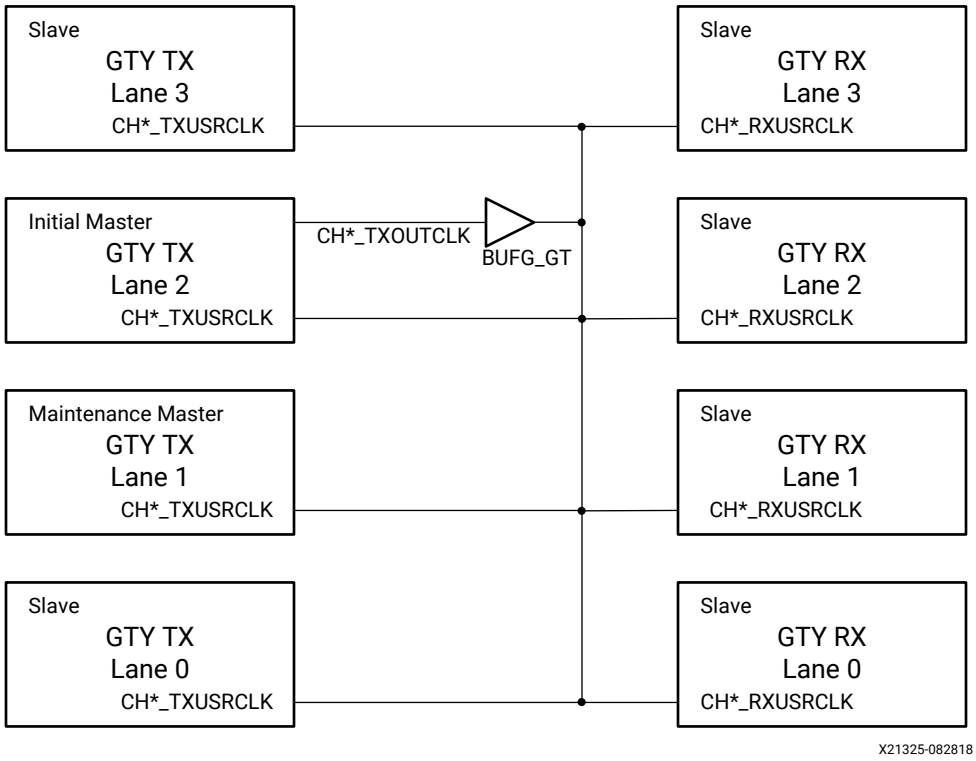
- CH*_TX_PHALIGN_CFG0[15] = `1'b1` (SYNC_MULTI_LANE)
- CH*_TX_PHALIGN_CFG0[14] = `1'b1` (TXBUF_BYPASS_MODE)
- CH*_TX_PHALIGN_CFG1[0] = `1'b0` (ASYNC_GBOX_PHALIGN_EN)
- CH*_PIPE_CTRL_CFG7[2:0] = `3'b011`, `3'b100`, `3'b101` (TXOUTCLKCTL) to select either the transceiver reference clock or the programmable divider clock as the source of TXOUTCLK.

RX buffer bypass settings:

- CH*_RX_PHALIGN_CFG0[31] = `1'b1` (DLY_ALIGN_EN)
- CH*_RX_PHALIGN_CFG0[30] = `1'b1` (PH_ALIGN_EN)
- CH*_RX_PHALIGN_CFG0[15] = `1'b1` (SYNC_MULTI_LANE)
- CH*_RX_PHALIGN_CFG0[14] = `1'b1` (RXBUF_BYPASS_MODE)
- CH*_RX_PHALIGN_CFG1[1] = `1'b0` (ASYNC_GBOX_PHALIGN_EN)

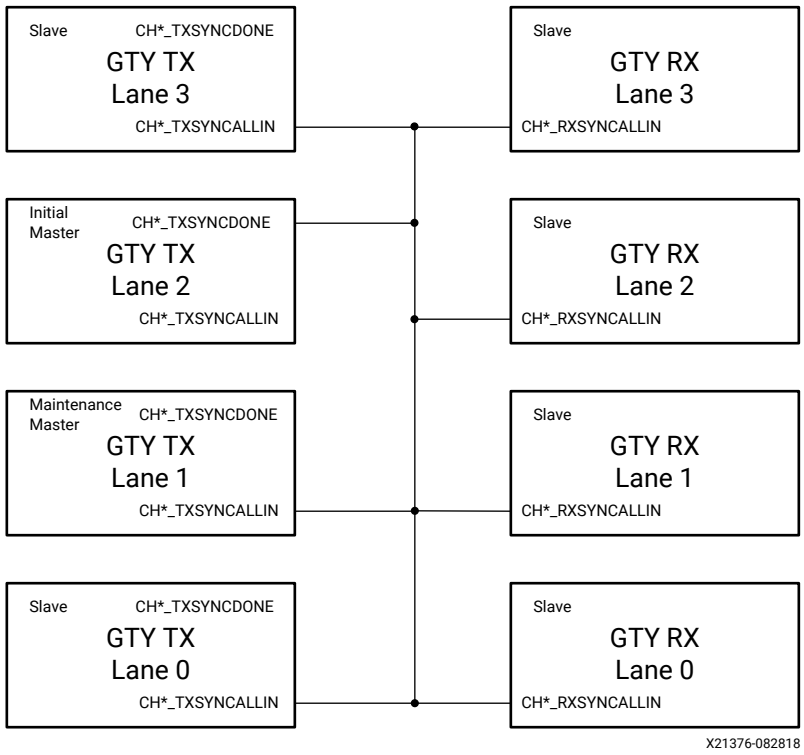
The following figure shows an example of a single-lane common clock buffer bypass with a TX master.

Figure 48: **Common Clock Buffer Bypass Lanes in Multi-Lane TX Master Mode**



The following figure shows the port connection for multi-lane common clock buffer bypass with a TX master.

Figure 49: **Port Connection for Common Clock Buffer Bypass Lanes in Multi-Lane TX Master Mode**

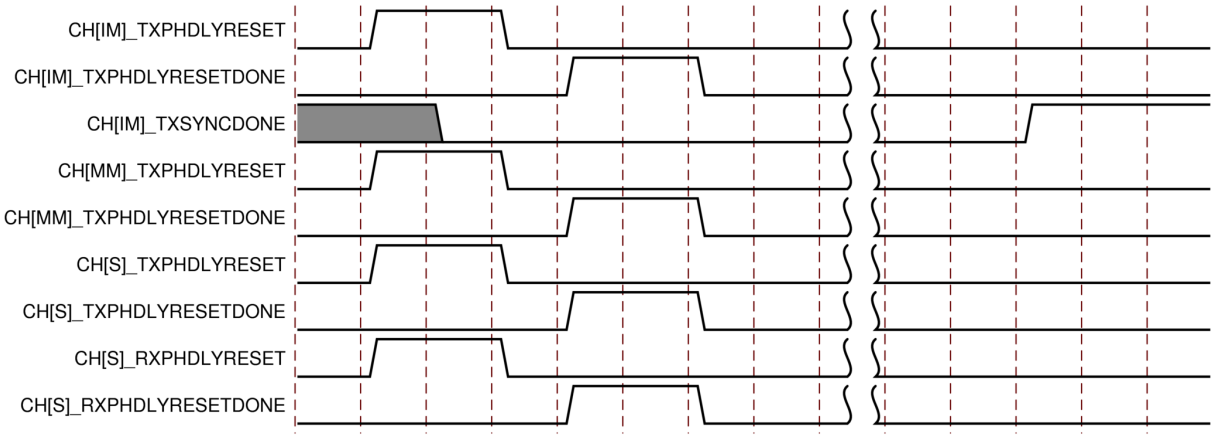


Multi-lane buffer bypass must only be used on lanes that have physical locations that are directly adjacent to one another. CH*_TX_PHALIGN_CFG1[2:1] (CHAIN_MODE) and CH*_RX_PHALIGN_CFG1[3:2] (CHAIN_MODE) must be set according to the physical location of the multi-lane group:

- Top location: CH*_TX_PHALIGN_CFG1[2:1]/CH*_RX_PHALIGN_CFG1[3:2] (CHAIN_MODE) = `2'b01`
- Middle location(s): CH*_TX_PHALIGN_CFG1[2:1]/CH*_RX_PHALIGN_CFG1[3:2] (CHAIN_MODE) = `2'b11`
- Bottom location: CH*_TX_PHALIGN_CFG1[2:1]/CH*_RX_PHALIGN_CFG1[3:2] (CHAIN_MODE) = `2'b10`

The following figure shows the required steps to perform common clock phase and delay alignment in multi-lane TX master mode.

Figure 50: **Common Clock Buffer Bypass - Multi-Lane with TX Master**



Notes relevant to the figure:

1. The sequence of events in the figure is not drawn to scale.
2. CH[IM]_* denotes ports related to the initial master lane.
3. CH[MM]_* denotes ports related to the maintenance master lane.
4. CH[S]_* denotes ports related to the slave lane(s).
5. After conditions such as a reset or rate change, common clock phase alignment must be performed. The common clock phase and delay alignments are initiated by asserting CH*_TXPHDLYRESET and CH*_RXPHDLYRESET.
6. Common clock phase alignment with TX master is done when the rising edge of CH*_TXSYNCDONE on the initial master lane is detected. This signal should remain asserted until another alignment procedure is initiated.
7. An assertion/deassertion of GTTXRESET and GTRXRESET is required if CH*_TXSYNCDONE does not follow the sequence shown in the figure.
8. TX delay alignment continues to adjust TXUSRCLK to compensate for temperature and voltage variations.

TXUSRCLK and RXUSRCLK Sharing Using Both TX and RX Buffer Bypass in Multi-Lane Auto Mode with RX Master

In multi-lane common clock buffer bypass with RX master, an RX lane must be designated as an initial master lane, and another as a maintenance master lane. All remaining RX lanes and all TX lanes must be designated as slave lanes.

Use these transceiver settings to bypass the TX and RX buffer in single-lane mode with a common clock, RX master.

Common clock settings:

- CH*_RX_PHALIGN_CFG5[27:26] = `2'b10` (CMN_FAB_CLK_PHALIGN_MODE)

TX buffer bypass settings:

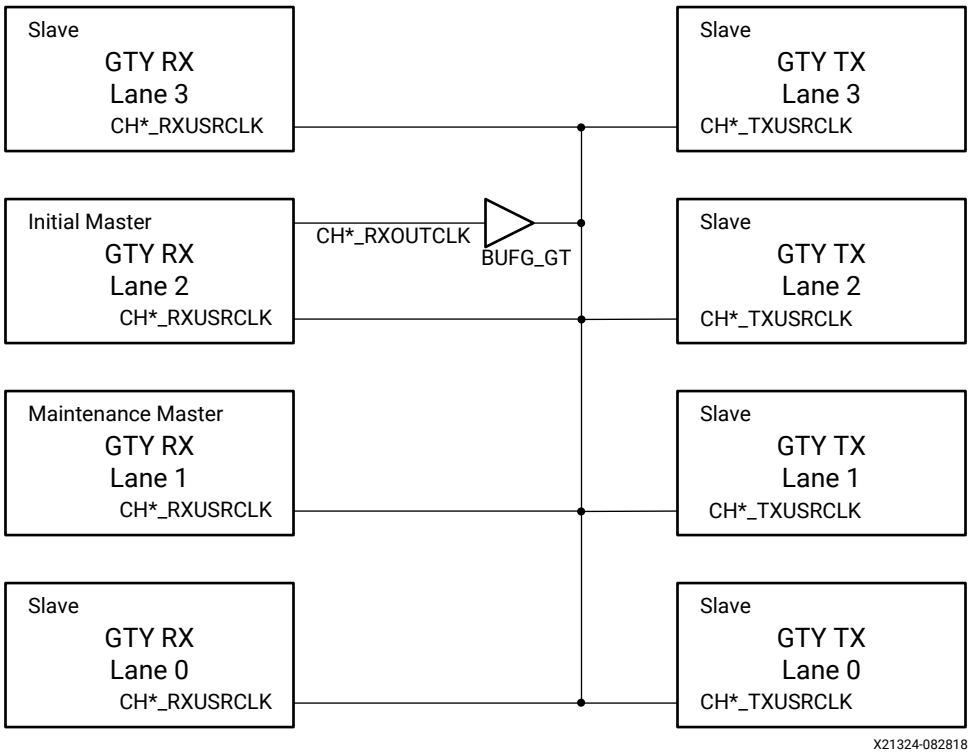
- CH*_TX_PHALIGN_CFG0[31] = `1'b1` (DLY_ALIGN_EN)
- CH*_TX_PHALIGN_CFG0[30] = `1'b1` (PH_ALIGN_EN)
- CH*_TX_PHALIGN_CFG0[15] = `1'b1` (SYNC_MULTI_LANE)
- CH*_TX_PHALIGN_CFG0[14] = `1'b1` (TXBUF_BYPASS_MODE)
- CH*_TX_PHALIGN_CFG1[0] = `1'b0` (ASYNC_GBOX_PHALIGN_EN)
- CH*_PIPE_CTRL_CFG7[2:0] = `3'b011`, `3'b100`, `3'b101` (TXOUTCLKCTL) to select either the transceiver reference clock or the programmable divider clock as the source of TXOUTCLK.

RX buffer bypass settings:

- CH*_RX_PHALIGN_CFG0[31] = `1'b1` (DLY_ALIGN_EN)
- CH*_RX_PHALIGN_CFG0[30] = `1'b1` (PH_ALIGN_EN)
- CH*_RX_PHALIGN_CFG0[15] = `1'b1` (SYNC_MULTI_LANE)
- CH*_RX_PHALIGN_CFG0[14] = `1'b1` (RXBUF_BYPASS_MODE)
- CH*_RX_PHALIGN_CFG1[1] = `1'b0` (ASYNC_GBOX_PHALIGN_EN)
- CH*_PIPE_CTRL_CFG7[14:12] = `3'b110` to select TXOUTCLK as the source of RXOUTCLK

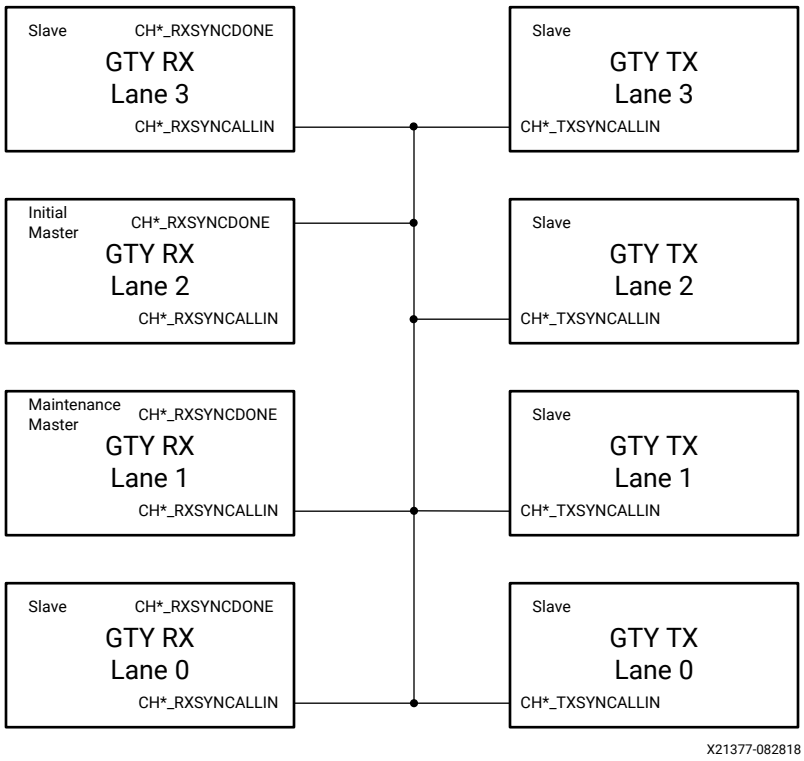
The following figure shows an example of a single-lane common clock buffer bypass with a TX master.

Figure 51: **Common Clock Buffer Bypass Lanes in Multi-Lane RX Master Mode**



The following figure shows the port connection for a multi-lane common clock buffer bypass with a TX master.

Figure 52: **Port Connection for Common Clock Buffer Bypass Lanes in Multi-Lane RX Master Mode**

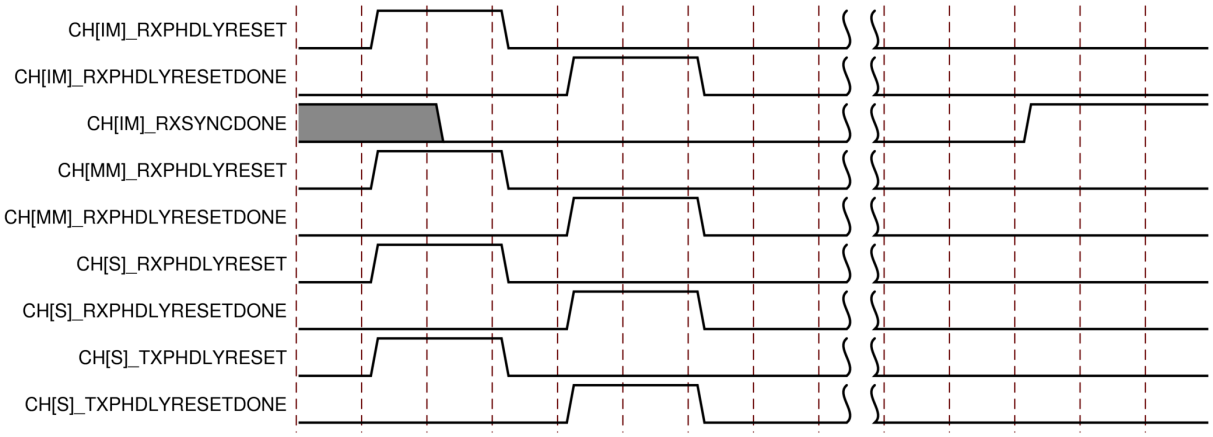


Multi-lane buffer bypass must only be used on lanes that have physical locations that are directly adjacent to one another. CH*_TX_PHALIGN_CFG1[2:1] (CHAIN_MODE) and CH*_RX_PHALIGN_CFG1[3:2] (CHAIN_MODE) must be set according to the physical location of the multi-lane group:

- Top location: CH*_TX_PHALIGN_CFG1[2:1]/CH*_RX_PHALIGN_CFG1[3:2] (CHAIN_MODE) = `2'b01`
- Middle location(s): CH*_TX_PHALIGN_CFG1[2:1]/CH*_RX_PHALIGN_CFG1[3:2] (CHAIN_MODE) = `2'b11`
- Bottom location: CH*_TX_PHALIGN_CFG1[2:1]/CH*_RX_PHALIGN_CFG1[3:2] (CHAIN_MODE) = `2'b10`

The following figure shows the required steps to perform common clock phase and delay alignment in multi-lane TX master mode.

Figure 53: **Common Clock Buffer Bypass - Multi-Lane with RX Master**



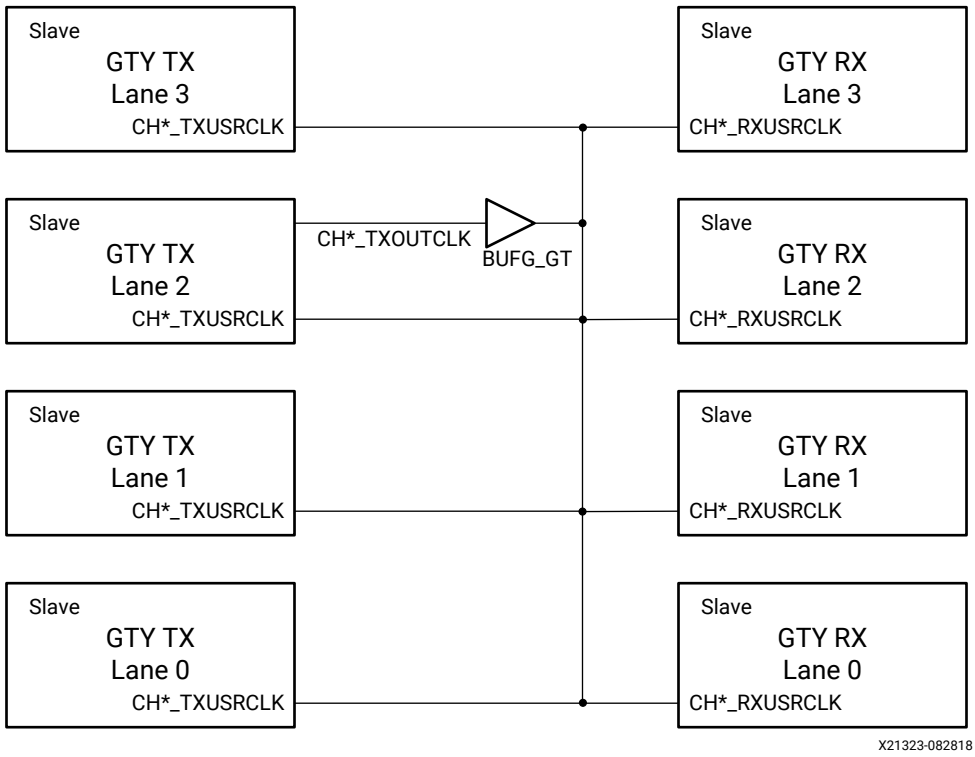
Notes relevant to the figure:

1. The sequence of events in the figure is not drawn to scale.
2. CH[IM]_* denotes ports related to the initial master lane.
3. CH[MM]_* denotes ports related to the maintenance master lane.
4. CH[S]_* denotes ports related to the slave lane(s).
5. After conditions such as a reset or rate change, common clock phase alignment must be performed. The common clock phase and delay alignments are initiated by asserting CH*_TXPHDLYRESET and CH*_RXPHDLYRESET.
6. Common clock phase alignment with a TX master is done when the rising edge of CH*_TXSYNCDONE on the initial master lane is detected. This signal should remain asserted until another alignment procedure is initiated.
7. An assertion/deassertion of GTTXRESET and GTRXRESET is required if CH*_TXSYNCDONE does not follow the sequence shown in the figure.
8. TX delay alignment continues to adjust TXUSRCLK to compensate for temperature and voltage variations.

TXUSRCLK and RXUSRCLK Sharing Using Both TX and RX Buffer Bypass with Asynchronous Gearbox Mode

In multi-lane with asynchronous gearbox mode with TX and RX common clock, the RX/TXUSRCLK domain is used to drive logic in the RX/TXPHYCLK domain. The following figure shows an example of asynchronous mode common clock buffer bypass lanes.

Figure 54: **Common Clock Buffer Bypass Lanes in Multi-Lane Asynchronous Gearbox Mode**



When the TX/RX internal data width is identical to the fabric interface data width, the TX/RX buffer is directly bypassed. Use these transceiver settings to bypass the RX and TX buffer with asynchronous gearbox in common clock multi-lane 1:1 mode:

Common clock settings:

- CH*_RX_PHALIGN_CFG5[27:26] = `2'b00` (CMN_FAB_CLK_PHALIGN_MODE) (Disabled because each lane works individually)

TX buffer bypass settings:

- CH*_TX_PHALIGN_CFG0[31] = `1'b0` (DLY_ALIGN_EN)
- CH*_TX_PHALIGN_CFG0[30] = `1'b0` (PH_ALIGN_EN)
- CH*_TX_PHALIGN_CFG0[17:16] = `2'b00` (SYNC_MODE) (All lanes are configured as slaves in multi-lane asynchronous gearbox mode)
- CH*_TX_PHALIGN_CFG0[15] = `1'b1` (SYNC_MULTI_LANE)
- CH*_TX_PHALIGN_CFG0[14] = `1'b1` (TXBUF_BYPASS_MODE)
- CH*_TX_PHALIGN_CFG1[2:1] = `2'b00` (CHAIN_MODE)
- CH*_TX_PHALIGN_CFG1[0] = `1'b1` (ASYNC_GBOX_PHALIGN_EN)

- CH*_PIPE_CTRL_CFG7[2:0] = `3'b011`, `3'b100`, `3'b101` (TXOUTCLKCTL) to select either the transceiver reference clock or the programmable divider clock as the source of TXOUTCLK

RX buffer bypass settings:

- CH*_RX_PHALIGN_CFG0[31] = `1'b0` (DLY_ALIGN_EN)
- CH*_RX_PHALIGN_CFG0[30] = `1'b0` (PH_ALIGN_EN)
- CH*_RX_PHALIGN_CFG0[17:16] = `2'b00` (SYNC_MODE) (All lanes are configured as slaves in multi-lane asynchronous gearbox mode)
- CH*_RX_PHALIGN_CFG0[15] = `1'b1` (SYNC_MULTI_LANE)
- CH*_RX_PHALIGN_CFG0[14] = `1'b1` (RXBUF_BYPASS_MODE)
- CH*_RX_PHALIGN_CFG1[3:2] = `2'b00` (CHAIN_MODE)
- CH*_RX_PHALIGN_CFG1[1] = `1'b1` (ASYNC_GBOX_PHALIGN_EN)
- CH*_PIPE_CTRL_CFG7[14:12] = `3'b110` to select TXOUTCLK as source of RXOUTCLK

In RX and TX buffer bypass with asynchronous gearbox in common clock multi-lane 1:1 mode, because the asynchronous gearbox FIFO provides the phase compensation, there is no need to perform the phase alignment procedure.

When the TX/RX fabric interface data width is twice the internal data width, the TX/RX buffer is bypassed through the 2:1 REG. Use these transceiver settings to bypass the RX and TX buffer with asynchronous gearbox in common clock multi-lane 2:1 mode:

Common clock settings:

- CH*_RX_PHALIGN_CFG5[27:26] = `2'b00` (CMN_FAB_CLK_PHALIGN_MODE) (Disabled because each lane works individually)

TX buffer bypass settings:

- CH*_TX_PHALIGN_CFG0[31] = `1'b1` (DLY_ALIGN_EN)
- CH*_TX_PHALIGN_CFG0[30] = `1'b1` (PH_ALIGN_EN)
- CH*_TX_PHALIGN_CFG0[17:16] = `2'b00` (SYNC_MODE) (All lanes are configured as slaves in multi-lane asynchronous gearbox mode)
- CH*_TX_PHALIGN_CFG0[15] = `1'b1` (SYNC_MULTI_LANE)
- CH*_TX_PHALIGN_CFG0[14] = `1'b1` (TXBUF_BYPASS_MODE)
- CH*_TX_PHALIGN_CFG1[2:1] = `2'b00` (CHAIN_MODE)
- CH*_TX_PHALIGN_CFG1[0] = `1'b1` (ASYNC_GBOX_PHALIGN_EN)

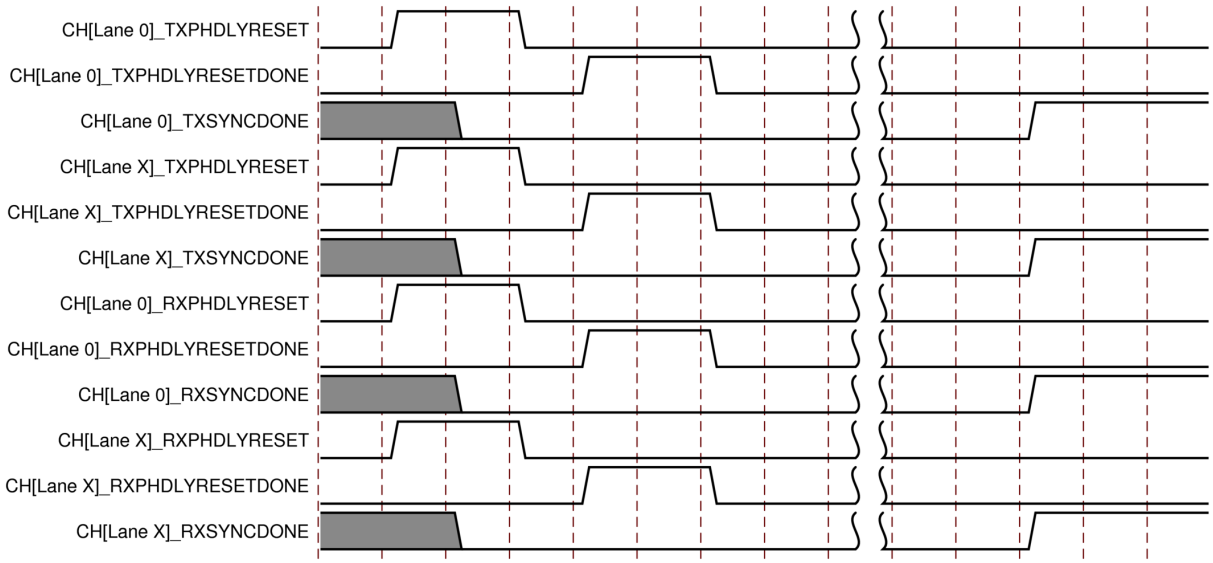
- CH*_PIPE_CTRL_CFG7[2:0] = `3'b011`, `3'b100`, `3'b101` (TXOUTCLKCTL) to select either the transceiver reference clock or the programmable divider clock as the source of TXOUTCLK.

RX buffer bypass settings:

- CH*_RX_PHALIGN_CFG0[31] = `1'b1` (DLY_ALIGN_EN)
- CH*_RX_PHALIGN_CFG0[30] = `1'b1` (PH_ALIGN_EN)
- CH*_RX_PHALIGN_CFG0[17:16] = `2'b00` (SYNC_MODE) (All lanes are configured as slaves in multi-lane asynchronous gearbox mode)
- CH*_RX_PHALIGN_CFG0[15] = `1'b1` (SYNC_MULTI_LANE)
- CH*_RX_PHALIGN_CFG0[14] = `1'b1` (RXBUF_BYPASS_MODE)
- CH*_RX_PHALIGN_CFG1[3:2] = `2'b00` (CHAIN_MODE)
- CH*_RX_PHALIGN_CFG1[1] = `1'b1` (ASYNC_GBOX_PHALIGN_EN)
- CH*_PIPE_CTRL_CFG7[14:12] = `3'b110` to select TXOUTCLK as source of RXOUTCLK

When bypassing the RX and TX buffer with asynchronous gearbox in common clock multi-lane 2:1 mode, the following timing diagram shows the required steps to perform auto phase and delay alignment.

Figure 55: **Common Clock Buffer Bypass - Multi-Lane Asynchronous Gearbox 2:1 Mode**



Notes relevant to the figure:

1. The sequence of events in the figure is not drawn to scale.
2. CH[Lane *]_* denotes ports of various lanes.

3. After conditions such as a reset or rate change, common clock phase alignment must be performed. The common clock phase and delay alignments are initiated by asserting CH*_TXPHDLYRESET and CH*_RXPHDLYRESET.
4. Common clock phase alignment is done when the rising edge of CH*_TXSYNCDONE and CH*_RXSYNCDONE for every lane is detected. This signal should remain asserted until another alignment procedure is initiated.
5. An assertion/deassertion of GTTXRESET and GTRXRESET is required if CH*_TXSYNCDONE and CH*_RXSYNCDONE for every lane does follow the sequence shown in the figure.
6. Common clock delay alignment continues to adjust RX/TXUSRCLK to compensate for temperature and voltage variations.

TX Synchronous Gearbox

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 synchronous 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 TX synchronous gearbox supports 2-byte, 4-byte, or 8-byte interfaces. Scrambling of the data is done in the interconnect logic.

Enabling the TX Synchronous Gearbox

The following table describes the attribute settings to enable the TX synchronous gearbox.

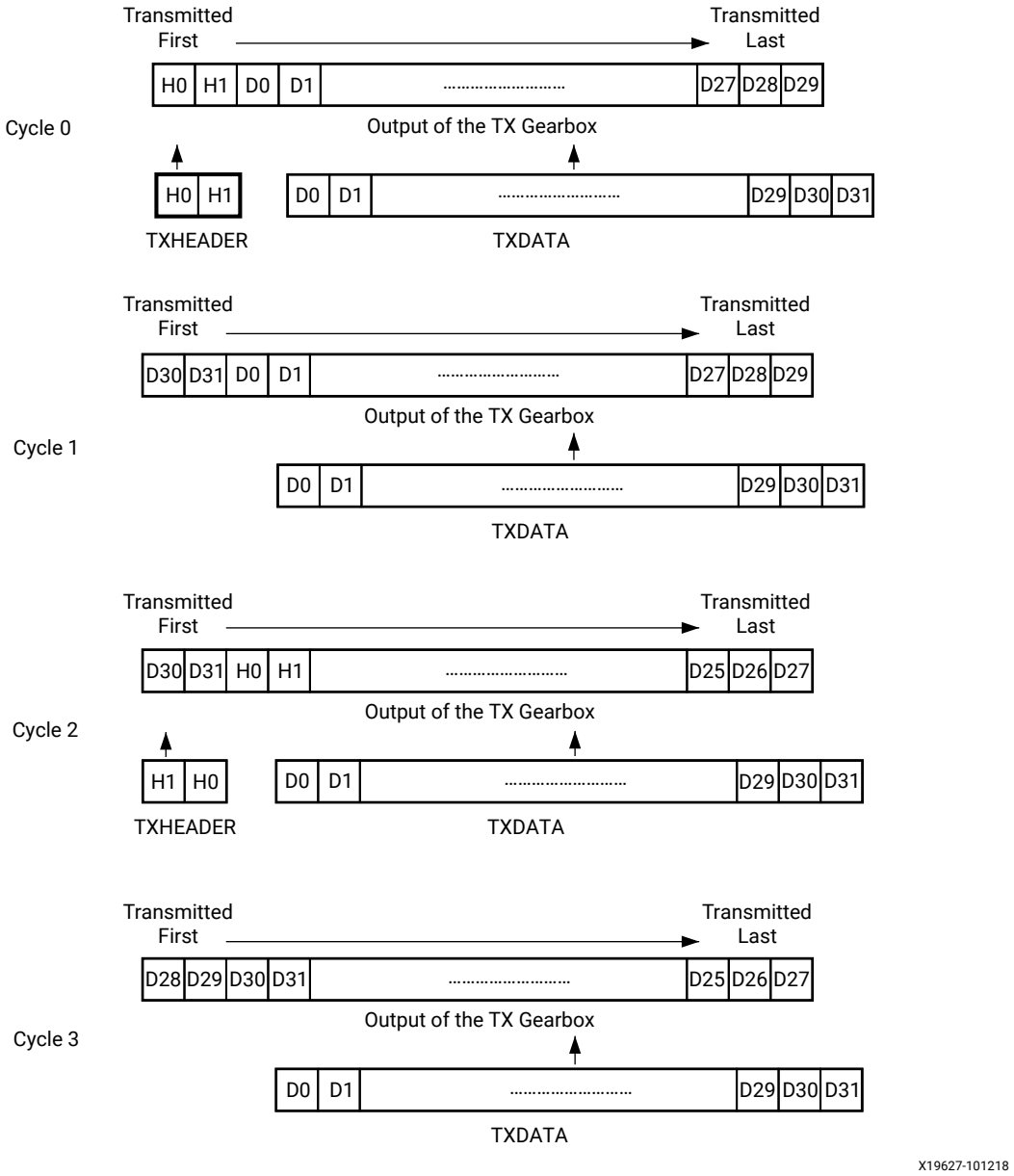
Table 48: **Enabling the TX Synchronous Gearbox**

Attribute	Label	Description
CH*_TX_PCS_CFG0[5]	USE_GB	This bit must be set to `1'b1`.
CH*_TX_PCS_CFG0[4:0]	MODE	Bit[4] - Must be set to `1'b0`. Bit[3] - Reserved. Must be set to `1'b0`. Bit[2] - Reserved. Must be set to `1'b0`. Bit[1] - Reserved. Must be set to `1'b0`. Bit[0] - Set to `1'b0` to use 64B/67B gearbox or `1'b1` to use 64B/66B gearbox.

TX Synchronous Gearbox Bit and Byte Ordering

The following figure shows an example of the first four cycles of data entering and exiting the TX gearbox for 64B/66B encoding when using a 4-byte logic interface (TX_DATA_WIDTH = 32 (4-byte), TX_INT_DATAWIDTH = 1 (4-byte)) in normal mode (CH*_TX_PCS_CFG0[2] = `1'b0`). The input consists of a 2-bit header and 32 bits of data. On the first cycle, the header and 30 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 30 data bits from the current TXDATA input exit the TX gearbox. On the third 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 56: **TX Gearbox Bit Ordering in Normal Mode (CH*_TX_PCS_CFG0[2] = `1'b0`)**



Note relevant to the figure:

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

Ports and Attributes

The following table defines the TX synchronous gearbox ports.

Table 49: **TXSYNCGEARBOX Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_TXHEADER[5:0]	Input	TXUSRCLK	Input port to provide header. TXHEADER[1:0] is used in normal mode. When normal mode is used with a 16-byte interface, TXHEADER[4:3] is also used to provide header information in addition to TXHEADER[1:0].
CH[0/1/2/3]_TXSEQUENCE[6:0]	Input	TXUSRCLK	TXSEQUENCE[0] is used to indicate on which TXUSRCLK cycle a header is provided onto the interface. On cycles where TXSEQUENCE[0] = 1'b0, the header is present on TXHEADER. When using a 64-bit (8-byte) or 128-bit (16-byte) TXDATA interface to interconnect logic, tie TXSEQUENCE[0] to 1'b0. When using a 32-bit (4-byte) TXDATA interface to interconnect logic, toggle TXSEQUENCE[0] every TXUSRCLK cycle.

The following table defines the TX synchronous gearbox attributes.

Table 50: **TXSYNCGEARBOX Attributes**

TXSYNCGEARBOX Attributes		
Attribute	**Address**	
CH0_TX_PCS_CFG0	0x0C77	
CH1_TX_PCS_CFG0	0x0D77	
CH2_TX_PCS_CFG0	0x0E77	
CH3_TX_PCS_CFG0	0x0F77	
Label	**Bit Field**	**Description**
USE_GB	[5:5]	This attribute must be set to 1'b1 to enable either the TX synchronous or asynchronous gearbox.
MODE	[4:0]	This attribute indicates the TX gearbox modes: Bit[4]: 1'b0 = Select Synchronous Gearbox. 1'b1 = Select Asynchronous Gearbox. Bit[3]: Reserved. Set to 1'b0. Bit[2]: Reserved. Set to 1'b0. Bit[1]: Reserved. Set to 1'b0. Bit[0]: 1'b0 = 64B/67B gearbox mode for Interlaken (Only valid for synchronous gearbox) 1'b1 = 64B/66B.

TX Asynchronous Gearbox

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 asynchronous gearbox provides support for 64B/66B header and payload combining. 64B/67B is not supported by the TX asynchronous gearbox.

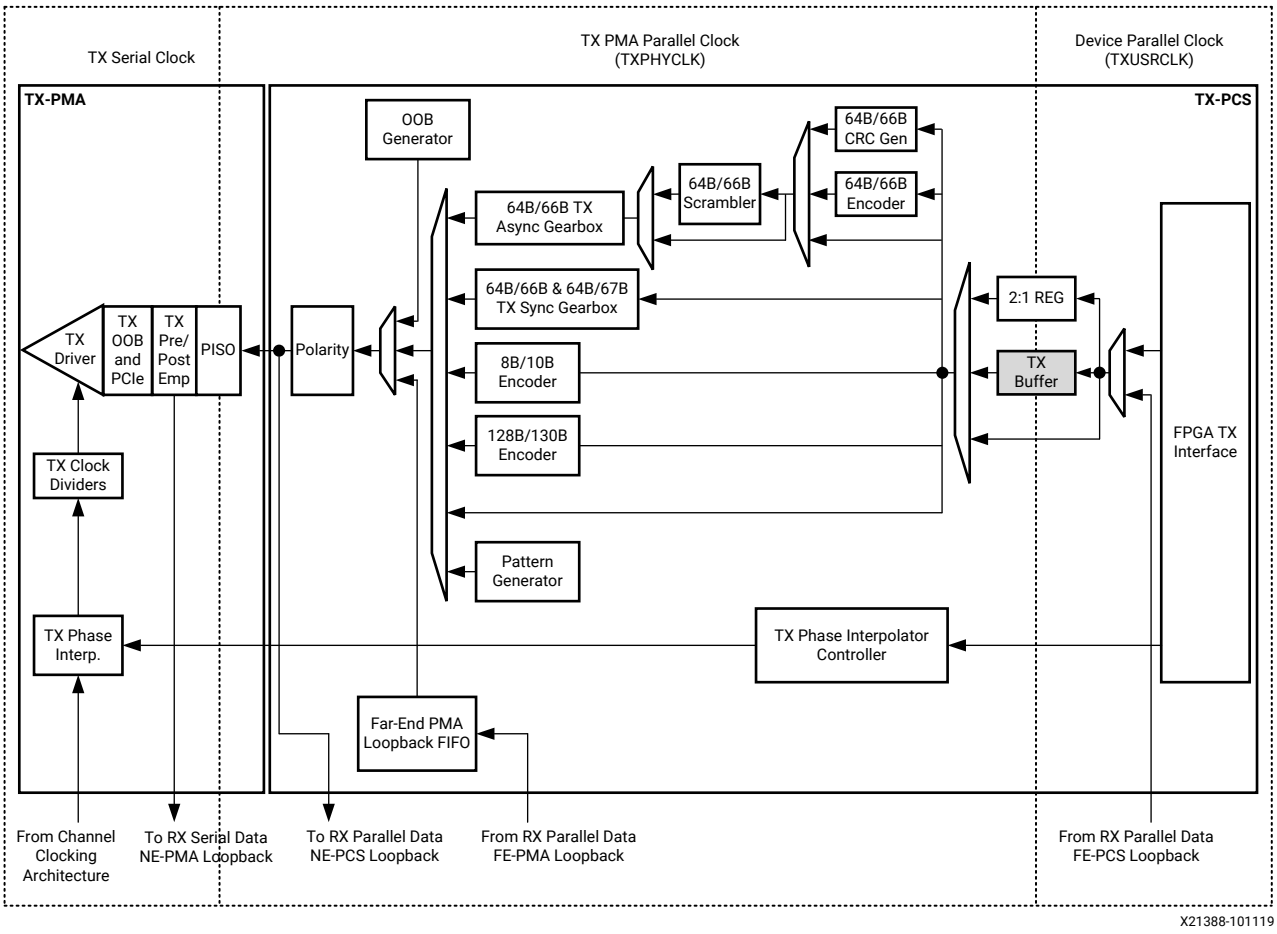
The TX asynchronous gearbox supports 4-byte, 8-byte, and 16-byte TX data interfaces to interconnect logic and requires the use of the 4-byte or 8-byte internal datapath. Scrambling of the data is done in the interconnect logic. The following table shows the valid data width combinations for the asynchronous gearbox.

Table 51: **Valid Data Width Combinations for TX Asynchronous Gearbox**

Internal Datapath Width	Interface Width	PHYCLK (MHz)	TXUSRCLK (MHz)
32	32	TX Line Rate/32	TX Line Rate/33
32	64	TX Line Rate/32	TX Line Rate/66
64	64	TX Line Rate/64	TX Line Rate/66
64	128	TX Line Rate/64	TX Line Rate/132

While the TX synchronous gearbox requires you to pause transmission of your data during various sequence counter values, the TX asynchronous gearbox allows data to be continuously applied every TXUSRCLK cycle. TX buffer bypass is required when using the TX asynchronous gearbox. The following figure shows the location of the TX asynchronous gearbox. When a 4-byte internal datapath is selected (TX_INT_DATA_WIDTH = 1), 32 bits of data are always output by the TX asynchronous gearbox on every TXPHYCLK cycle. Alternating 34 bits (2-bit header and 32-bit payload) and 32 bits (32 bits payload) of data enter the TX asynchronous gearbox every TXUSRCLK cycle. For an 8-byte internal datapath, 64 bits of data are always output by the TX asynchronous gearbox on every TXPHYCLK cycle. 66 bits (2-bit header and 64-bit payload) of data enter the TX asynchronous gearbox every TXUSRCLK cycle.

Figure 57: **TX Clock Domain Example (TX_INT_DATA_WIDTH = 1 (4-byte) and TX_DATA_WIDTH = 64)**



When in normal mode, the datapath latency through the TX asynchronous gearbox is measured internally, and the reported latency can be accessed by reading a read-only register via APB3. The TX asynchronous gearbox is used in conjunction with the TX programmable dividers. TXOUTCLKCTL must be set to `3'b101` and an appropriate divide value must be selected to create the required clock frequency for TXUSRCLK.

Enabling the TX Asynchronous Gearbox

The following table describes the attribute settings to enable the TX asynchronous gearbox.

Table 52: **Enabling the TX Asynchronous Gearbox**

Attribute	Label	Description
CH*_TX_PCS_CFG0[5]	USE_GB	This bit must be set to `1'b1`.

Table 52: **Enabling the TX Asynchronous Gearbox** *(cont'd)*

Attribute	Label	Description
CH*_TX_PCS_CFG0[4:0]	MODE	Bit[4] - Must be set to `1'b1`. Bit[3] - Reserved. Must be set to `1'b0`. Bit[2] - Reserved. Must be set to `1'b0`. Bit[1] - Reserved. Must be set to `1'b0`. Bit[0] - Set to `1'b1` to use 64B/66B gearbox when using the asynchronous gearbox.

TX Asynchronous Gearbox Bit and Byte Ordering

The TX asynchronous gearbox uses the same bit ordering as the TX synchronous gearbox. Refer to TX Synchronous Gearbox Bit and Byte Ordering for additional details.

Reading Datapath Latency

The datapath latency through the TX async gearbox FIFO is calculated statistically using TXLATCLK, which is asynchronous to TX_PHYCLK. SAMPLE_PERIOD in CH*_TX_PCS_CFG0 determines the number of TXLATCLK cycles over which averaging takes place. The measured latency value in TXGBOX_FIFO_LATENCY is updated once per sampling period, which is defined in SAMPLE_PERIOD. The latency measurement is not supported in CAUI mode.

These settings are used to read the latency:

- Enable TX asynchronous gearbox under normal mode.
- Set CH*_TX_PCS_CFG0[12:10] (SAMPLE_PERIOD):
  - A higher averaging period gives a more accurate latency value.
- Read CH*_TXGBOX_FIFO_LATENCY[29:16] (TXGBOX_FIFO_LATENCY): The value is in units of 1/8 UI.
- The actual latency is TXGBOX_FIFO_LATENCY plus a fixed value.

Ports and Attributes

The following table defines the TX asynchronous gearbox ports.

Table 53: **TXASYNCGEARBOX Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_TXHEADER[5:0]	Input	TXUSRCLK	Input port to provide header. TXHEADER[1:0] is used in normal mode. When normal mode is used with a 16-byte interface, TXHEADER[4:3] is also used to provide header information in addition to TXHEADER[1:0].
CH[0/1/2/3]_TXLATCLK	Input	CLOCK	Input port used to provide a clock for the TX asynchronous gearbox latency calculation.
CH[0/1/2/3]_TXSEQUENCE[6:0]	Input	TXUSRCLK	TXSEQUENCE[0] is used to indicate on which TXUSRCLK cycle a header is provided onto the interface. On cycles where TXSEQUENCE[0] = 1'b0, the header is present on TXHEADER. When using a 64-bit (8-byte) or 128-bit (16-byte) TXDATA interface to interconnect logic, tie TXSEQUENCE[0] to 1'b0. When using a 32-bit (4-byte) TXDATA interface to interconnect logic, toggle TXSEQUENCE[0] every TXUSRCLK cycle.

The following table defines the TX asynchronous gearbox attributes.

Table 54: **TXASYNCGEARBOX Attributes**

TXASYNCGEARBOX Attributes		
Attribute	**Address**	
CH0_TXGBOX_FIFO_LATENCY	0x0863	
CH1_TXGBOX_FIFO_LATENCY	0x0963	
CH2_TXGBOX_FIFO_LATENCY	0x0a63	
CH3_TXGBOX_FIFO_LATENCY	0x0b63	
Label	**Bit Field**	**Description**
TXGBOX_FIFO_LATENCY	[29:16]	Measured latency in UI through the TX asynchronous gearbox averaged over SAMPLE_PERIOD (CH*_TX_PCS_CFG0[12:10]) cycles. The reported latency is in units of 1/8 UI. The TXGBOX_FIFO_LATENCY is read-only.
Attribute	**Address**	
CH0_TXGBOX_FIFO_OVERFLOW	0x0861	
CH1_TXGBOX_FIFO_OVERFLOW	0x0961	
CH2_TXGBOX_FIFO_OVERFLOW	0x0a61	
CH3_TXGBOX_FIFO_OVERFLOW	0x0b61	
Label	**Bit Field**	**Description**
TXGBOX_FIFO_OVERFLOW	[24:24]	TX asynchronous gearbox FIFO overflow status. A High indicates that overflow error has occured. The TXGBOX_FIFO_OVERFLOW is read-only.

Table 54: **TXASYNCGEARBOX Attributes** *(cont'd)*

TXASYNCGEARBOX Attributes		
Attribute	**Address**	
CH0_TXGBOX_FIFO_UNDERFLOW		0x0861
CH1_TXGBOX_FIFO_UNDERFLOW		0x0961
CH2_TXGBOX_FIFO_UNDERFLOW		0x0a61
CH3_TXGBOX_FIFO_UNDERFLOW		0x0b61
Label	**Bit Field**	**Description**
TXGBOX_FIFO_UNDERFLOW	[25:25]	TX asynchronous gearbox FIFO underflow status. A High indicates that underflow error has occured. The TXGBOX_FIFO_UNDERFLOW is read-only.
Attribute	**Address**	
CH0_TX_PCS_CFG0		0x0C77
CH1_TX_PCS_CFG0		0x0D77
CH2_TX_PCS_CFG0		0x0E77
CH3_TX_PCS_CFG0		0x0F77
Label	**Bit Field**	**Description**
SAMPLE_PERIOD	[12:10]	Number of CH*_TXLATCLK cycles over which averaging take place for latency calculation: 3'b000: 256 3'b001: 512 3'b010: 1024 3'b011: 2048 3'b100: 4096 3'b101: 8192 3'b110: 16384 3'b111: 32768
USE_GB	[5:5]	This attribute must be set to 1'b1 to enable either the TX synchronous or asynchronous gearbox.
MODE	[4:0]	This attribute indicates the TX gearbox modes: Bit[4]: 1'b0 = Select Synchronous Gearbox. 1'b1 = Select Asynchronous Gearbox. Bit[3]: Reserved. Set to 1'b0. Bit[2]: Reserved. Set to 1'b0. Bit[1]: Reserved. Set to 1'b0. Bit[0]: 1'b0 = 64B/67B gearbox mode for Interlaken (Only valid for synchronous gearbox) 1'b1 = 64B/66B.

TX Pattern Generator

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 GTY transceiver pattern generator block can generate several industry-standard PRBS patterns listed in the following table.

Table 55: **Supported PRBS Patterns**

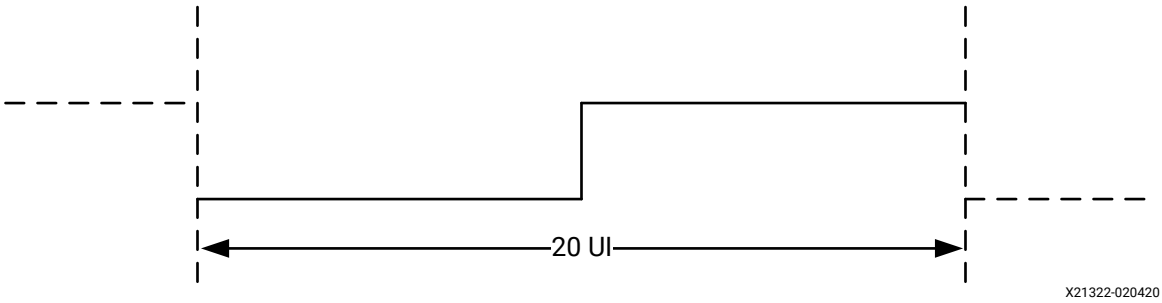
Name	Polynomial	Length of Sequence	Description
PRBS-7	$1 + X^6 + X^7$	$2^7 - 1$ bits	Used to test channels with 8B/10B.
PRBS-9	$1 + X^5 + X^9$	$2^9 - 1$ bits	ITU-T Recommendation O.150, Section 5.1. PRBS-9 is one of the recommended test partterns for SFP+.
PRBS-15	$1 + X^{14} + X^{15}$	$2^{15} - 1$ bits	ITU-T Recommendation O.150, Section 5.3. PRBS-15 is often used for jitter measurement because it is the longest pattern that the Keysight DCA-X sampling oscilloscope 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 Std 802.3ae-2002.

In addition to PRBS patterns, the transceiver supports 16 UI, 20 UI, 32 UI, 40 UI, 64 UI, or 80 UI square wave test patterns, depending on internal data width, as well as a 2 UI square wave test pattern and PCI Express compliance pattern generation. Clocking patterns are usually used to check PLL random jitter often done with a spectrum analyzer.

Table 56: **PCI Express Compliance Pattern**

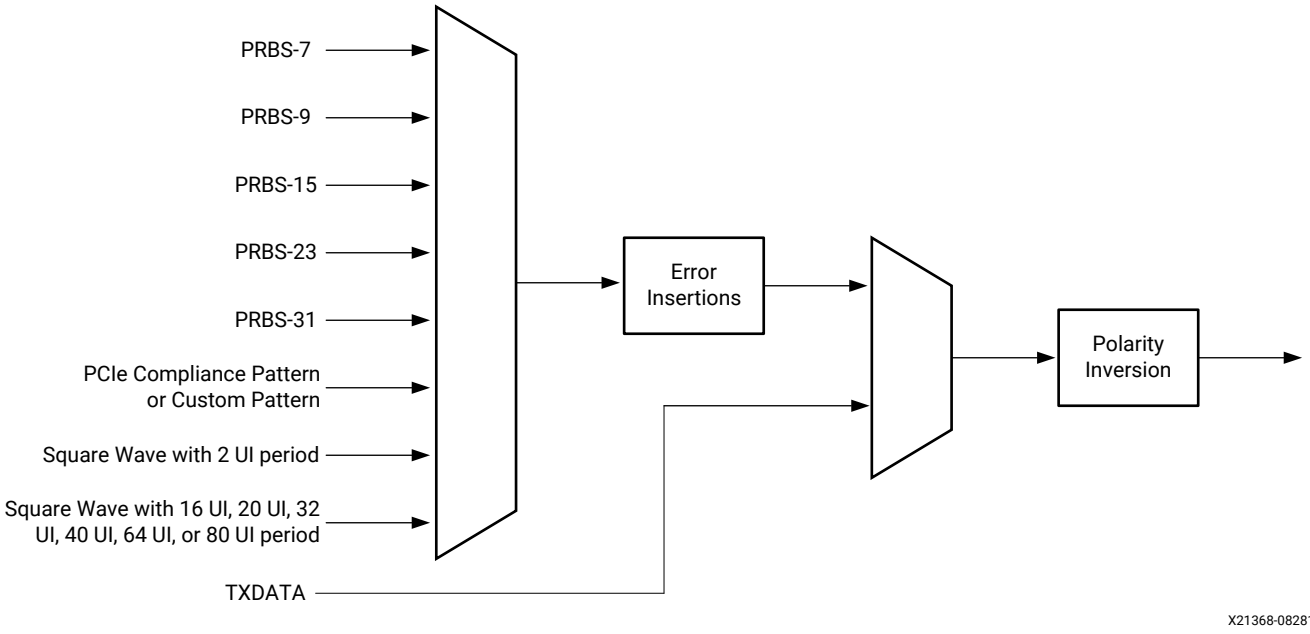
Symbol	K28.5	D21.5	K28.5	D21.5
Disparity	0	1	1	0
Pattern	0011111010	1010101010	1100000101	0101010101

Figure 58: **20 UI Square Wave**



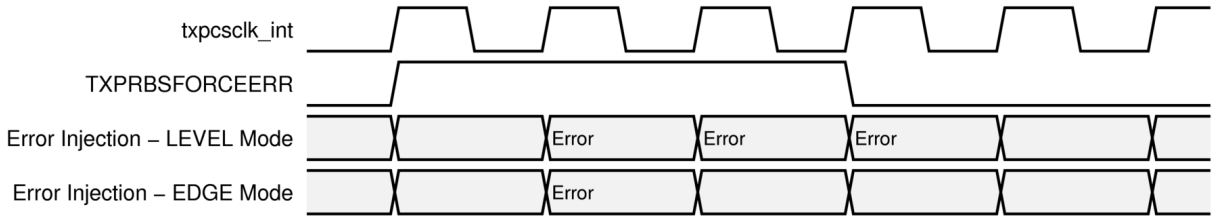
The error insertion function is supported to verify link connection and also for jitter tolerance tests. When an inverted PRBS pattern is necessary, the CH*_TXPOLARITY signal is used to control polarity.

Figure 59: **TX Pattern Generator Block**



The pattern generator might control when an injection of an error in the generated pattern occurs by choosing between two modes. In LEVEL mode, the generator injects an error after any cycle in which CH*_TXPRBSFORCEERR is High. In EDGE mode, an error is injected only after CH*_TXPRBSFORCEERR has just gone High after being Low in the previous cycle. The following figure shows the error injection in both operating modes.

Figure 60: **LEVEL and EDGE Mode Error Injection**



Ports and Attributes

The following table defines the pattern generator ports.

Table 57: **TX Pattern Generator Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_TXPRBSFORCEERR	Input	ASYNC	When this port is driven High, a single error is forced in the PRBS transmitter for every TXUSRCLK2 clock cycle that the port is asserted. While this port is asserted, the output data pattern contains one error for every TXUSRCLK2 clock cycle while the port is asserted. When TXPRBSSEL is set to 4'b0000, this port does not affect TXDATA.
CH[0/1/2/3]_TXPRBSSEL[3:0]	Input	ASYNC	Transmitter PRBS generator test pattern control. 4'b0000: Standard operation mode (test pattern generation is off) 4'b0001: PRBS-7 4'b0010: PRBS-9 4'b0011: PRBS-15 4'b0100: PRBS-23 4'b0101: PRBS-31 4'b1000: User configurable data pattern. Defaults to PCI Express compliance pattern. Only works with internal data width 20 bit, 40 bit, and 80 bit modes. 4'b1001: Square wave with 2 UI (alternating 0s/1s) 4'b1010: Square wave with 16 UI, 20 UI, 32 UI, 40 UI, 64 UI, or 80 UI period (based on internal data width)

The following table defines the pattern generator attribute.

Table 58: **TX Pattern Generator Attributes**

TX Pattern Generator Attributes	
Attribute	**Address**
CH0_TX_PCS_CFG1	0x0C78

Table 58: **TX Pattern Generator Attributes** *(cont'd)*

TX Pattern Generator Attributes		
CH1_TX_PCS_CFG1		0x0D78
CH2_TX_PCS_CFG1		0x0E78
CH3_TX_PCS_CFG1		0x0F78
Label	**Bit Field**	**Description**
TX_PRBS_FORCE_MODE	[31:31]	Pattern Generator error operation mode 0: Level mode 1: Edge mode
RXPRBS_ERR_LOOPBACK	[30:30]	Enable RXPRBSERR error injection
TX_PRBS_USERPATTERN2	[29:20]	TX PRBS user configurable data pattern byte2. Default is K28.5
TX_PRBS_USERPATTERN1	[19:10]	TX PRBS user configurable data pattern byte1. Default is D21.5
TX_PRBS_USERPATTERN0	[9:0]	TX PRBS user configurable data pattern byte0. Default is K28.5
Attribute	**Address**	
CH0_TX_PCS_CFG2		0x0C79
CH1_TX_PCS_CFG2		0x0D79
CH2_TX_PCS_CFG2		0x0E79
CH3_TX_PCS_CFG2		0x0F79
Label	**Bit Field**	**Description**
TX_PRBS_USERPATTERN5	[29:20]	TX PRBS user configurable data pattern byte5. Default is K28.5
TX_PRBS_USERPATTERN4	[19:10]	TX PRBS user configurable data pattern byte4. Default is D21.5
TX_PRBS_USERPATTERN3	[9:0]	TX PRBS user configurable data pattern byte3. Default is D10.2
Attribute	**Address**	
CH0_TX_PCS_CFG3		0x0C7A
CH1_TX_PCS_CFG3		0x0D7A
CH2_TX_PCS_CFG3		0x0E7A
CH3_TX_PCS_CFG3		0x0F7A
Label	**Bit Field**	**Description**
TX_PRBS_USERPATTERN7	[19:10]	TX PRBS user configurable data pattern byte7. Default is D10.2
TX_PRBS_USERPATTERN6	[9:0]	TX PRBS user configurable data pattern byte6. Default is D10.2

Using TX Pattern Generator

In all but one use mode, the TX pattern generator can be enabled by changing the value of the CH*_TXPRBSSEL port to select the desired pattern. When the TX asynchronous gearbox is enabled, these additional steps must be taken to enable the TX pattern generator:

1. Put the PCS into reset by following the corresponding TX component reset procedure.
2. Set attribute TXGEARBOX_EN to `1'b0` and TXBUF_EN to `1'b1` via the APB3.
3. Set attribute TXOUTCLKCTL to `3'b010` (TXPHYCLK).
4. Set port CH*_TXPRBSSEL to the desired pattern.
5. Release the PCS from reset by following the corresponding TX component reset procedure.

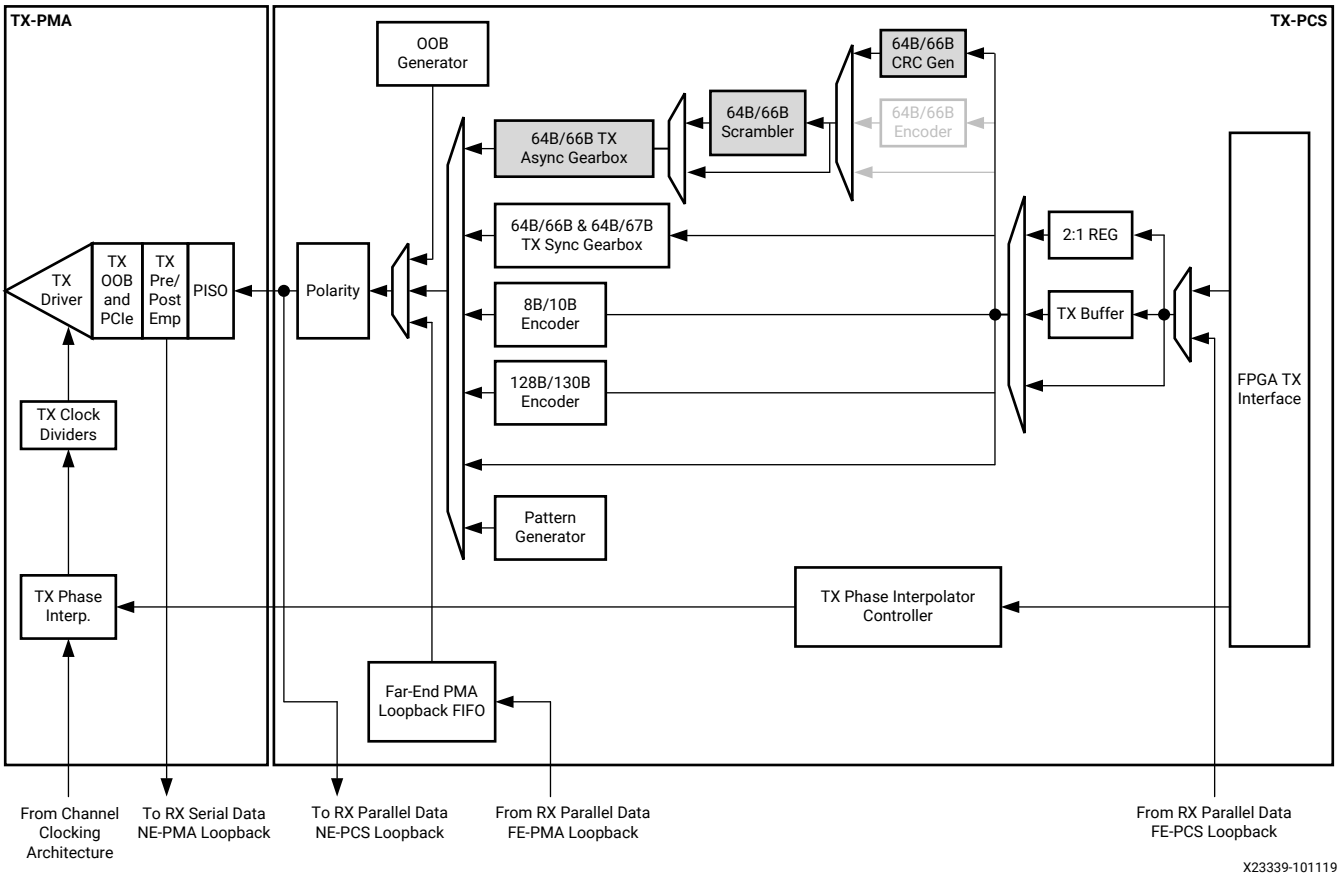
To return to TX asynchronous gearbox mode, the above changes must be reversed as described below:

1. Put the PCS into reset by following the corresponding TX component reset procedure.
2. Set the attribute TXGEARBOX_EN to `1'b1` and TXBUF_EN to `1'b0` via the APB3.
3. Set attribute TXOUTCLKCTL to `3'b101` (TXPROGDIVCLK).
4. Set port CH*_TXPRBSSEL to `4'b0000`.
5. Release the PCS from reset by following the corresponding TX component reset procedure.

TX CRC Generator

The Versal ACAP GTY transceiver contains a cyclic redundancy check (CRC) generator that transmit an error-detecting code or checksum value as part of the transmitted data. The generator supports the generation of CRC-32 for Aurora in 64B/66B mode. It does not support 10G/25G Ethernet or Interlaken.

Figure 61: **TX CRC Generator**



Using the CRC Generator

When the CRC generator is used, the CRC generator enable must be set and the 64B/66B encoder and 64B/66B decoder must be bypassed.

- CRC_EN should be set to `1'b1`.
- DATAPATH_CTRL can only be set to either `2'b01` or `2'b10` depending on whether or not the TX 64B/66B scrambler is bypassed.

Another requirement is that depending on whether or not the TX 64B/66B scrambler is bypassed, the TX CRC generator must be set to provide the expected output endian format. The following must be set:

- When the TX 64B/66B scrambler is bypassed, OUTPUT_BYTE_ORDER must be set to `1'b0`.
- When the TX 64B/66B scrambler is enabled, OUTPUT_BYTE_ORDER must be set to `1'b1`.

Ports and Attributes

There are no ports for the TX CRC generator. The following table defines the TX CRC generator attributes.

Table 59: **TXCRC Attributes**

TXCRC Attributes		
Attribute	**Address**	
CH0_TX_10G_CFG1		0x0C8E
CH1_TX_10G_CFG1		0x0D8E
CH2_TX_10G_CFG1		0x0E8E
CH3_TX_10G_CFG1		0x0F8E
Label	**Bit Field**	**Description**
DATAPATH_CTRL	[30:29]	This setting selects the correct datapath for the PCS data while going through the asynchronous 64B/66B data path. The 64B/66B Encoder and Scrambler can be enabled or disabled independently. 2'b00: Encoder and Scrambler both enabled. 2'b01: Encoder disabled, Scrambler enabled. 2'b10: Encoder and Scrambler both disabled. In this option only the 64B/66B TX Gearbox FIFO is enabled.
Attribute	**Address**	
CH0_TX_CRC_CFG0		0x0C58
CH1_TX_CRC_CFG0		0x0D58
CH2_TX_CRC_CFG0		0x0E58
CH3_TX_CRC_CFG0		0x0F58
Label	**Bit Field**	**Description**
SOP4_CONTROL_WORD	[31:16]	When SOP_DETECT_MODE is enabled, this sets the /S4/ control word that will be used for detection of SOP
SOP_CONTROL_WORD	[15:8]	When SOP_DETECT_MODE is enabled, this sets the /S/ control word that will be used for detection of SOP
SOP_DETECT_MODE	[6:6]	Setting to enable the SOP detection for start of data package 0: Disabled 1: Enabled
BYTE_MODE	[5:4]	Data width selection for the CRC. 00: 8-Byte 01: 4-Byte
CHECKSUM_PACK_MODE	[3:3]	Checksum packing format, the setting need to be set to 1'b1.
CRC_MODE	[2:2]	Setting of the CRC generator mode, the setting need to be set to 1'b0.
OUTPUT_BYTE_ORDER	[1:1]	Setting of the output data format. 0: Little Endian 1: Big Endian

Table 59: **TXCRC Attributes** *(cont'd)*

TXCRC Attributes		
CRC_EN	[0:0]	CRC Generator Enable. 0: Disabled 1: Enabled

TX Polarity Control

If the TXP and TXN differential traces are accidentally swapped on the PCB, the differential data transmitted by the 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 CH*_TXPOLARITY input from the interconnect logic interface. It is driven High to invert the polarity of the outgoing data.

Ports and Attributes

The following table defines the ports required for TX polarity control.

Table 60: **TX Polarity Control Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_TXPOLARITY	Input	TXUSRCLK	This 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.

There are no TX polarity control attributes in Versal ACAPs.

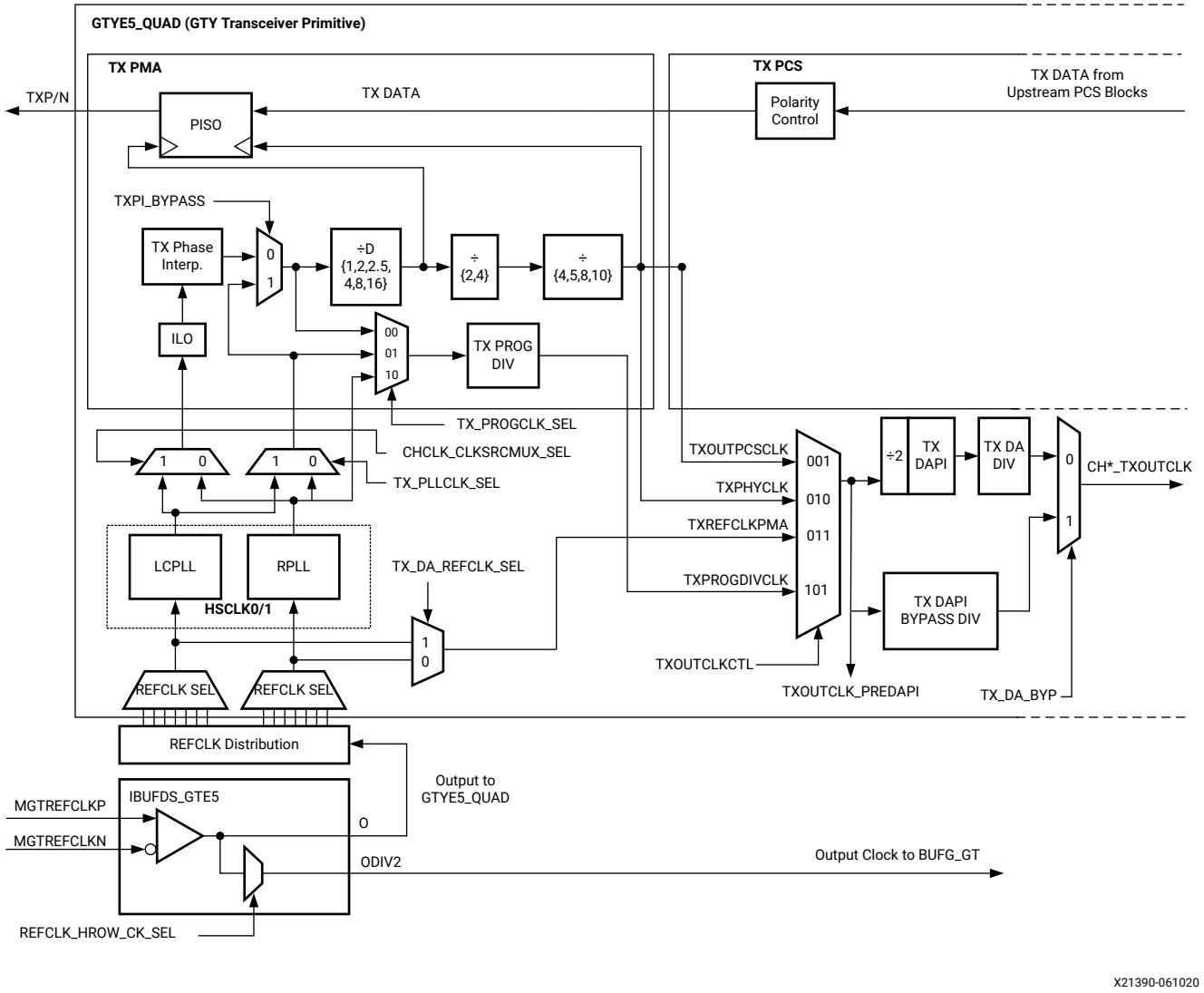
Using TX Polarity Control

CH*_TXPOLARITY can be tied High if the polarity of TXP and TXN needs to be reversed.

TX Fabric Clock Output Control

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 the following figure.

Figure 62: **TX Serial and Parallel Clock Divider**



Note:

1. CH*_TXOUTCLK is used as the source of the interconnect logic clock via BUFG_GT.

   The RPLL and LCPLL from HSCLK0 can only be used by TX channel 0/1, and RPLL and LCPLL from HSCLK1 can only be used by TX channel 2/3.

2. The selection of the /2 and /4 divider block and /4, /5, /8, and /10 divider block is selected based on the TX_DATA_WIDTH and TX_INT_DATA_WIDTH.

3. For details about placement constraints and restrictions on clocking resources (such as BUFG_GT and BUFG_GT_SYNC), refer to the *Versal ACAP Clocking Resources Architecture Manual* (AM003).

4. The clock output from IBUFDS_GTE5 should only be used after GTPOWERGOOD asserts High.

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, TXOUT_DIV must be set to the appropriate value, and the CH*_TXRATE port should be tied to `8'b00000000`.

For multiple line rate applications, the CH*_TXRATE port is used to dynamically select the line rate settings which include the appropriate divider values. See Rate Change for more details.

Parallel Clock Divider and Selector

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

The recommended clock for the interconnect logic is the CH*_TXOUTCLK from one of the GTY transceivers. It is also possible to bring the MGTREFCLK directly to the interconnect logic and use it as the interconnect logic clock. CH*_TXOUTCLK is preferred for general applications because 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 for more details.

TXOUTCLKCTL controls the input selector and allows these clocks to be output via the CH*_TXOUTCLK port:

- `3'b001`: TXOUTPCSCLK path is not recommended to be used because it incurs extra delay from the PCS block.
- `3'b010`: TXPHYCLK 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.
- `3'b011`: TXREFCLKPMA is the input reference clock to the RPLL or LCPLL, depending on the TXOUTCLKCTL setting. TXREFCLKPMA is the recommended clock for general usage and is required for the TX buffer bypass mode.
- `3'b101`: TXPRODIVCLK is the divided down PLL clock after the TX programmable divider. See TX Programmable Divider for more details.

TX Programmable Divider

The TX programmable divider shown in TX Fabric Clock Output Control uses one of the PLL output clocks to generate a parallel output clock. By using the transceiver PLL, TX programmable divider, and BUFG_GT, CH*_TXOUTCLK (TXOUTCLKSEL = `101`) can be used as a clock source for the interconnect logic. The supported divider values are 4, 5, 5.5, 8, 10, 16, 16.5, 20, 32, 33, and 40.

The high-speed clock multiplexer controlled by TX_PROGCLK_SEL is set based on the application requirements:

- `00`: The post TX phase interpolator (PI) clock path can be used to generate a parallel clock with a certain ppm offset created by the TX PI. In this use case, one transceiver PLL is shared for the datapath and clock generation path. The clock signal is interrupted if the channel or the source PLL is being reset.
- `01`: The pre TX PI clock path can be used to generate a system clock to support applications where minimal or fixed latency is needed. In this use case, one transceiver PLL is shared for the datapath and clock generation path. The clock signal is interrupted only if the source PLL is being reset.
- `10`: In applications where the LCPLL clock might be interrupted during reconfiguration, the bypass clock path provides the flexibility to use the RPLL to generate a stable parallel clock for the interconnect logic.

TX DAPI and TX DAPI Bypass Dividers

The TX delay align and phase interpolator (TX DAPI) shown in TX Fabric Clock Output Control can either be enabled or bypassed to provide the CH*_TXOUTCLK.

TX_DA_BYP determines the clock path and is set based on application requirements:

- `0`: The TX DAPI clock path is used to generate the CH*_TXOUTCLK. In this use case, the TX DAPI has a built-in divide-by-2 that is always present. The divider values for TX DA DIV should be left to the default values from the Wizard example design.
- `1`: The clock path where TX DAPI is bypassed generates the CH*_TXOUTCLK. In this use case, the TX DAPI BYPASS DIV divider value should be left to the default values from the Wizard example design.

Ports and Attributes

The following table defines the ports required for TX fabric clock output control.

Table 61: **TX Fabric Clock Output Control Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_TXOUTCLK	Output	CLOCK	TXOUTCLK is the recommended clock output to the interconnect logic.
CH[0/1/2/3]_TXPROGDIVRESET	Input	ASYNC	This active-High port resets the dividers as well as the CH*_TXPROGDIVRESETDONE indicator. A reset must be performed whenever the input clock source is interrupted.
CH[0/1/2/3]_TXPROGDIVRESETDONE	Output	ASYNC	When the input clock is stable and reset is performed, this active-High signal indicates the reset is completed and the output clock is stable.

Table 61: **TX Fabric Clock Output Control Ports** *(cont'd)*

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_TXRATE[7:0]	Input	TXUSRCLK	This port is used to perform rate change on the Transceiver TX. The Wizard will preconfigure a list of desired line rates, and this port will be used to dynamically adjust the running line rate based on the preconfigured list. Set this port to the matching preconfigured line rate option value to obtained the proper line rate.

The following table defines the attributes required for TX fabric clock output control.

Table 62: **TX Fabric Clock Output Control Attributes**

TX Fabric Clock Output Control Attributes		
Attribute	**Address**	
CH0_CHCLK_ILO_CFG		0x0C6B
CH1_CHCLK_ILO_CFG		0x0D6B
CH2_CHCLK_ILO_CFG		0x0E6B
CH3_CHCLK_ILO_CFG		0x0F6B
Label	**Bit Field**	**Description**
CHCLK_CLKSRCMUX_SEL	[18:18]	This attribute selects either the RPLL or LCPLL clock as input to the ILO. 0: Select RPLL 1: Select LCPLL
Attribute	**Address**	
CH0_CHCLK_TXPI_CFG0		0x0C6F
CH1_CHCLK_TXPI_CFG0		0x0D6F
CH2_CHCLK_TXPI_CFG0		0x0E6F
CH3_CHCLK_TXPI_CFG0		0x0F6F
Label	**Bit Field**	**Description**
TX_PROGCLK_SEL	[19:18]	This attribute selects the clocking source for TX Programmable Divider. 2'b00: Select clock path after the TX phase interpolator 2'b01: Select RPLL or LCPLL clock when TXPI Bypass Mode is enabled 2'b10: Select RPLL clock
TXPI_BYPASS	[2:2]	This attribute enables the TXPI Bypass Mode. 1'b0: TXPI Bypass disabled 1'b1: TXPI Bypass enabled
TX_PLLCLK_SEL	[0:0]	This attribute select between the LCPLL and RPLL for TX Clocking for being used in TXPI Bypass Mode. 1'b0: Select RPLL 1'b1: Select LCPLL

Table 62: **TX Fabric Clock Output Control Attributes** *(cont'd)*

TX Fabric Clock Output Control Attributes		
Attribute	**Address**	
CH0_DA_CFG		0x0C5C
CH1_DA_CFG		0x0D5C
CH2_DA_CFG		0x0E5C
CH3_DA_CFG		0x0F5C
Label	**Bit Field**	**Description**
TX_DA_REFCLK_SEL	[19:19]	selects the reference clock for the RX from RPLL or the LCPLL: 1'b0 : RPLL REFCLK 1'b1 : LCPLL REFCLK
Attribute	**Address**	
CH0_PIPE_CTRL_CFG3		0x0CA0
CH1_PIPE_CTRL_CFG3		0x0DA0
CH2_PIPE_CTRL_CFG3		0x0EA0
CH3_PIPE_CTRL_CFG3		0x0FA0
Label	**Bit Field**	**Description**
TXOUT_DIV	[25:23]	This attribute selects the setting for the TX serial clock divider D. 3'b000: /1 3'b001: /2 3'b010: /4 3'b011: /8 3'b100: /16 3'b111: /2.5
Attribute	**Address**	
CH0_PIPE_CTRL_CFG7		0x0CA4
CH1_PIPE_CTRL_CFG7		0x0DA4
CH2_PIPE_CTRL_CFG7		0x0EA4
CH3_PIPE_CTRL_CFG7		0x0FA4
Label	**Bit Field**	**Description**
TX_DA_BYP	[3:3]	This attribute enables the TX DAPI bypass. 0: TX DAPI is not bypassed 1:TX DAPI is bypassed
TXOUTCLKCTL	[2:0]	This attribute selects the source for CH*_TXOUTCLK. 3'b001: TXOUTPCSCLK 3'b010: TXPHYCLK 3'b011: TXREFCLKPMA 3'b101: TXPROGDIVCLK
Attribute	**Address**	
CH0_PIPE_CTRL_CFG8		0x0CA5
CH1_PIPE_CTRL_CFG8		0x0DA5

Table 62: **TX Fabric Clock Output Control Attributes** *(cont'd)*

TX Fabric Clock Output Control Attributes		
CH2_PIPE_CTRL_CFG8		0x0EA5
CH3_PIPE_CTRL_CFG8		0x0FA5
Label	**Bit Field**	**Description**
TXPROGDIVSEL	[9:0]	TX programmable divider ratio. Valid settings are 4, 5, 5.5, 8, 10, 16, 16.5, 20, 32, 33, and 40. 10'b1001011000: /4 10'b1001111000: /5 10'b1110011000: /5.5 10'b0001011000: /8 10'b1001100000: /10 10'b0001000000: /16 10'b1100011000: /16.5 10'b0001100000: /20 10'b0001000010: /32 10'b0100011000: /33 10'b0001100010: /40

TX Phase Interpolator PPM Controller

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 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 interconnect logic can control the TX PI in the TX PMA through the use of the TX phase interpolator PPM controller module in the PCS. The TX phase interpolator PPM controller module is only supported for line rates up to 16.375 Gb/s.

Ports and Attributes

The following table defines the ports required for the TX phase interpolator PPM controller.

Table 63: **TXPIPPM Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_TXPIPPMEN	Input	TXUSRCLK	1'b0: Disables the TX Phase Interpolator PPM Controller block. The TX PI is not updated with a PI code and retains the previous PI code. 1'b1: Enables the TX Phase Interpolator PPM Controller block. The TX PI is updated with a PI code every CH*_TX_PIPPM_CFG [29:25]: TXPI_SYNFREQ_PPM[2:0] cycles.
CH[0/1/2/3]_TXPIPPMSTEPSIZE[4:0]	Input	TXUSRCLK	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.

The following table defines the TX Phase phase interpolator PPM controller attributes.

Table 64: **TXPIPPM Attributes**

TXPIPPM Attributes		
Attribute	**Address**	
CH0_TX_PIPPM_CFG		0x0CA7
CH1_TX_PIPPM_CFG		0x0DA7
CH2_TX_PIPPM_CFG		0x0EA7
CH3_TX_PIPPM_CFG		0x0FA7
Label	**Bit Field**	**Description**
TXPI_SYNFREQ_PPM	[29:25]	This attribute specifies how often PI Code to the TX PI is updated. It is updated every (TXPI_SYNFREQ_PPM[2:0] + 1) cycles. All values are valid except for 5'b00000. Use the Wizard's default value for this attribute.
TXPI_DDR_EN	[24:24]	Enable DDR PI operation.
TXPI_PPM_DDR_CFG	[23:16]	When TXPIPPMOVRDEN=1, program the lower 7 bits of this attribute to one of the 128 values output to the TX PI. The most significant bit needs to be pulse (asserted High and then Low) for the TX PI to register the new 7-bit value of this Attribute.
TXPI_INVSTROBE_SEL	[15:15]	Inversion of pistrobe_ppm_sa for easy timing to TX PI capture flops.
TXPI_PPM_CFG	[11:4]	When TXPIPPMOVRDEN=1, program the lower 7 bits of this attribute to one of the 128 values output to the TX PI. The most significant bit needs to be pulse (asserted High and then Low) for the TX PI to register the new 7-bit value of this Attribute.
TXPIPPMOVRDEN	[3:3]	Enable direct control of the PI Code output to the TX PI in the TX PMA. Use with TXPI_PPM_CFG[6:0] to program the value of PI code.
TXPI_GRAY_SEL	[2:2]	Select gray code for fabric input TXPIPPMSTEPSIZE[3:0].
TXPIPPMSEL	[1:1]	Select TX PI PPM controller in PMA.

Table 64: **TXPIPPM Attributes** *(cont'd)*

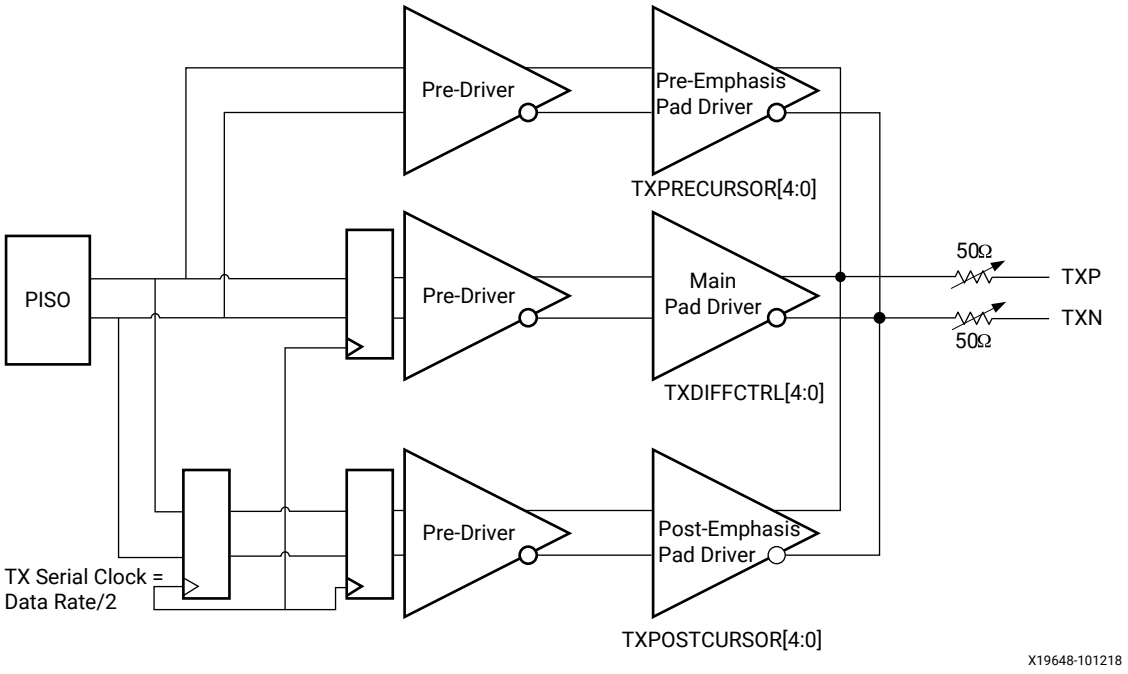
TXPIPPM Attributes		
TXPIPPMPD	[0:0]	TXPI PPM Powerdown.

TX Configurable Driver

The GTY transceiver TX driver is a high-speed voltage-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 63: **TX Configurable Driver Block Diagram**



TX Driver Swing Control

Driver swing can be controlled through CH*_TXDIFFCTRL[4:0]. The default is user specified. All values listed below are in mV_{PDD} and are typical values.

Table 65: **Transmitter Swing Control**

[4:0]	Swing (mV_{PDD})
5'b00000	245
5'b00001	269
5'b00010	293
5'b00011	322
5'b00100	345
5'b00101	370
5'b00110	396
5'b00111	420
5'b01000	445
5'b01001	469
5'b01010	497
5'b01011	525
5'b01100	546
5'b01101	578
5'b01110	602
5'b01111	626
5'b10000	650
5'b10001	678
5'b10010	703
5'b10011	727
5'b10100	750
5'b10101	774
5'b10110	799
5'b10111	825
5'b11000	852
5'b11001	880
5'b11010	907
5'b11011	937
5'b11100	955
5'b11101	981
5'b11110	1015
5'b11111	1033

Notes:

1. The peak-to-peak differential voltage is defined when CH*_TXPOSTCURSOR = 5'b00000 and CH*_TXPRECUSOR = 5'b00000.
2. The output swing described above is obtained using settings from the Wizard design, and the recommended values from the Wizard should not be changed.

TX Driver Post-Cursor Control

Driver swing can be controlled through CH*_TXPOSTCURSOR[4:0]. The default is user specified. All values listed below are in dB and are typical values.

Table 66: **Transmitter Post-Cursor TX Pre-Emphasis Control**

[4:0]	Emphasis (dB)	|Coefficient Units|
5'b00000	0	0
5'b00001	0.41	1
5'b00010	0.81	2
5'b00011	1.27	3
5'b00100	1.76	4
5'b00101	2.26	5
5'b00110	2.86	6
5'b00111	3.48	7
5'b01000	4.14	8
5'b01001	4.88	9
5'b01010	5.69	10
5'b01011	6.60	11
5'b01100	7.60	12
5'b01101	8.76	13
5'b01110	10.11	14
5'b01111	11.72	15
5'b10000	13.65	16
5'b10001	16.21	17
5'b10010	19.37	18

Notes:

1. The above values are defined when CH*_TXPRECURSOR = 5'b00000.
2. Emphasis = $20\log10(V_{high}/V_{low}) = |20\log10(V_{low}/V_{high})|$

TX Driver Pre-Cursor Control

Driver swing can be controlled through CH*_TXPRECURSOR[4:0]. The default is user specified. All values listed below are in dB and are typical values.

Table 67: **Transmitter Pre-Cursor TX Pre-Emphasis Control**

[4:0]	Emphasis (dB)	|Coefficient Units|
5'b00000	0	0
5'b00001	0.41	1
5'b00010	0.84	2
5'b00011	1.31	3

Table 67: **Transmitter Pre-Cursor TX Pre-Emphasis Control** *(cont'd)*

[4:0]	Emphasis (dB)	|Coefficient Units|
5'b00100	1.82	4
5'b00101	2.36	5
5'b00110	2.94	6
5'b00111	3.56	7
5'b01000	4.25	8

Notes:

1. The above values are defined when CH*_TXPOSTCURSOR = 5'b00000.
2. Emphasis = 20log10(V_{high}/V_{low}) = \|20log10(V_{low}/V_{high})\|

Ports and Attributes

The following table defines the TX configurable driver ports.

Table 68: **TX Configurable Driver Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_TXDEEMPH[1:0]	Input	ASYNC	TX de-emphasis control for PCI Express PIPE 3.0 interface. This signal is mapped internally to TXPOSTCURSOR via Attribute.
CH[0/1/2/3]_TXDIFFCTRL[4:0]	Input	ASYNC	Driver Swing Control. The default is user specified.
CH[0/1/2/3]_TXELECIDLE	Input	TXUSRCLK	When High, this signal forces MGTHTXP and MGTHTXN both to Common mode, creating an electrical idle signal.
CH[0/1/2/3]_TXINHIBIT	Input	TXUSRCLK	When High, this signal blocks transmission of TXDATA and forces MGTHTXP to 0 and MGTHTXN to 1.
CH[0/1/2/3]_TXMAINCURSOR[6:0]	Input	ASYNC	Allows the main cursor coefficients to be directly set if the TX_MAINCURSOR_SEL attribute is set to 1'b1.
CH[0/1/2/3]_TXMARGIN[2:0]	Input	ASYNC	TX Margin control for PCI Express PIPE 3.0 Interface. These signals are mapped internally to TXDIFFCTRL via Attribute.
CH[0/1/2/3]_TXPOSTCURSOR[4:0]	Input	ASYNC	Transmitter post-cursor TX pre-emphasis control. The default is user specified.
CH[0/1/2/3]_TXPRECURSOR[4:0]	Input	ASYNC	Transmitter pre-cursor TX pre-emphasis control. The default is user specified.
CH[0/1/2/3]_TXSWING	Input	ASYNC	TX swing control for PCI Express PIPE 3.0 Interface. This signal is mapped internally to TXDIFFCTRL.

The following table defines the TX configurable driver attributes.

Table 69: **TX Configurable Driver Attributes**

TX Configurable Driver Attributes		
Attribute	**Address**	
CH0_PIPE_TX_EQ_CFG0		0x0C73
CH1_PIPE_TX_EQ_CFG0		0x0D73
CH2_PIPE_TX_EQ_CFG0		0x0E73
CH3_PIPE_TX_EQ_CFG0		0x0F73
Label	**Bit Field**	**Description**
TX_MAINCURSOR_SEL	[30:30]	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 pins within the range specified in the pin description.
TX_DRIVE_MODE	[29:28]	This attribute selects whether PCI Express PIPE 3.0 pins or TX Drive Control pins control the TX driver. The default is DIRECT. DIRECT: TXDIFFCTRL, TXPOSTCURSOR, TXPRECURSOR and TXMAINCURSOR (If CH*_PIPE_TX_EQ_CFG0[30] = 1) control the TX driver settings. PIPE0: TXDEEMPH, TXMARGIN, TXSWING, TXPRECURSOR and TXMAINCURSOR (If CH*_PIPE_TX_EQ_CFG0[30] = 1) control the TX driver settings. PIPE1: TXMARGIN, TXSWING, TXPOSTCURSOR, TXPRECURSOR and TXMAINCURSOR (If CH* PIPE_TX_EQ_CFG0[30] = 1) control the TX driver settings.
TX_MARGIN_FULL_3	[27:21]	This attribute has the value of TXDIFFCTRL[4:0] that has to be mapped when TXMARGIN = 011 and TXSWING = 0. CH*_PIPE_TX_EQ_CFG0[27:21] = {2'b0, TXDIFFCTRL[4:0]}.
TX_MARGIN_FULL_2	[20:14]	This attribute has the value of TXDIFFCTRL[4:0] that has to be mapped when TXMARGIN = 010 and TXSWING = 0. CH*_PIPE_TX_EQ_CFG0[20:14] = {2'b0, TXDIFFCTRL[4:0]}.
TX_MARGIN_FULL_1	[13:7]	This attribute has the value of TXDIFFCTRL[4:0] that has to be mapped when TXMARGIN = 001 and TXSWING = 0. CH*_PIPE_TX_EQ_CFG0[13:7] = {2'b0, TXDIFFCTRL[4:0]}.
TX_MARGIN_FULL_0	[6:0]	This attribute has the value of TXDIFFCTRL[4:0] that has to be mapped when TXMARGIN = 000 and TXSWING = 0. CH*_PIPE_TX_EQ_CFG0[6:0] = {2'b0, TXDIFFCTRL[4:0]}.
Attribute	**Address**	
CH0_PIPE_TX_EQ_CFG1		0x0C74
CH1_PIPE_TX_EQ_CFG1		0x0D74
CH2_PIPE_TX_EQ_CFG1		0x0E74
CH3_PIPE_TX_EQ_CFG1		0x0F74
Label	**Bit Field**	**Description**
TX_MARGIN_LOW_2	[27:21]	This attribute has the value of TXDIFFCTRL[4:0] that has to be mapped when TXMARGIN = 010 and TXSWING = 1. CH*_PIPE_TX_EQ_CFG1[27:21] = {2'b0, TXDIFFCTRL[4:0]}.

Table 69: **TX Configurable Driver Attributes** *(cont'd)*

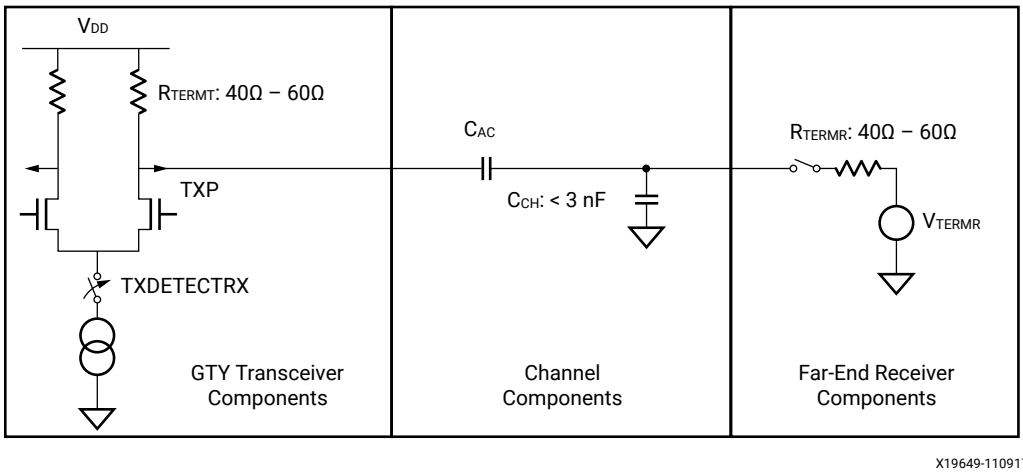
TX Configurable Driver Attributes		
TX_MARGIN_LOW_1	[20:14]	This attribute has the value of TXDIFFCTRL[4:0] that has to be mapped when TXMARGIN = 001 and TXSWING = 1. CH*_PIPE_TX_EQ_CFG1[20:14] = {2'b0, TXDIFFCTRL[4:0]}.
TX_MARGIN_LOW_0	[13:7]	This attribute has the value of TXDIFFCTRL[4:0] that has to be mapped when TXMARGIN = 000 and TXSWING = 1. CH*_PIPE_TX_EQ_CFG1[13:7] = {2'b0, TXDIFFCTRL[4:0]}.
TX_MARGIN_FULL_4	[6:0]	This attribute has the value of TXDIFFCTRL[4:0] that has to be mapped when TXMARGIN = 100 and TXSWING = 1. CH*_PIPE_TX_EQ_CFG1[6:0] = {2'b0, TXDIFFCTRL[4:0]}.
Attribute	**Address**	
CH0_PIPE_TX_EQ_CFG2	0x0C75	
CH1_PIPE_TX_EQ_CFG2	0x0D75	
CH2_PIPE_TX_EQ_CFG2	0x0E75	
CH3_PIPE_TX_EQ_CFG2	0x0F75	
Label	**Bit Field**	**Description**
TX_MARGIN_LOW_4	[13:7]	This attribute has the value of TXDIFFCTRL[4:0] that has to be mapped when TXMARGIN = 100 and TXSWING = 1. CH*_PIPE_TX_EQ_CFG2[13:7] = {2'b0, TXDIFFCTRL[4:0]}.
TX_MARGIN_LOW_3	[6:0]	This attribute has the value of TXDIFFCTRL[4:0] that has to be mapped when TXMARGIN = 011 and TXSWING = 1. CH*_PIPE_TX_EQ_CFG2[6:0] = {2'b0, TXDIFFCTRL[4:0]}.
Attribute	**Address**	
CH0_PIPE_TX_EQ_CFG3	0x0C76	
CH1_PIPE_TX_EQ_CFG3	0x0D76	
CH2_PIPE_TX_EQ_CFG3	0x0E76	
CH3_PIPE_TX_EQ_CFG3	0x0F76	
Label	**Bit Field**	**Description**
TX_DEEMPH_2	[14:10]	Sets transmitter at 0 dB de-emphasis at 5 Gb/s.

TX Receiver Detect Support for PCI Express Designs

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. TX Receiver Detect Support for PCI Express Designs shows the circuit model used for receive detection. The GTY transceiver must be in the P1 power down state to perform receiver detection. Receiver detection requires an external coupling capacitor between the transmitter and receiver, and the receiver must be terminated. Refer to the PCI Express Base Specification for the actual value of the external coupling capacitor in Gen1, Gen2, Gen3, or Gen4

applications. The receiver detection sequence starts with the assertion of CH*_TXDETECTRX. In response, the receiver detection logic drives TXN and TXP to (V_{DD} - V_{SWING}/2) and then releases them. 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 CH*_RXSTATUS when CH*_PHYSTATUS is asserted High for one cycle.

Figure 64: **Receiver Detection Circuit Model**



***Note*:** Check the PCI Express Base Specification for the actual value of the external coupling capacitor in Gen1, Gen2, Gen3, or Gen4 applications.

Ports and Attributes

The following table describes the TX receiver detection ports.

Table 70: **TXDETECTRX Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_PHYSTATUS	Output	RXUSRCLK	In PCI Express mode, this signal is used to communicate completion of several 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 70: **TXDETECTRX Ports** *(cont'd)*

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_RXPD[1:0]	Input	RXUSRCLK	Powers down the RX lane according to the PCI Express encoding. In PCI Express mode, tie TXPD and RXPD 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.
CH[0/1/2/3]_RXSTATUS[2:0]	Output	RXUSRCLK	During receiver detection, this signal is read when PHYSTATUS is asserted High. Only these encoding are valid during receiver detection: 000: Receiver not present. 011: Receiver present.
CH[0/1/2/3]_TXDETECTRX	Input	TXUSRCLK	Used to tell the transceiver to begin a receiver detection operation. 0: Normal operation. 1: Receiver detection.
CH[0/1/2/3]_TXPD[1:0]	Input	TXUSRCLK	Powers down the TX lane according to the PCI Express encoding. 2'b00: P0 normal operation 2'b01: P0s low recovery time power down 2'b10: P1 longer recovery time, RecDet still on 2'b11: P2 lowest power state. Transition times between the power states are controlled by the following attributes: PD_TRANS_TIME_FROM_P2 PD_TRANS_TIME_NONE_P2 PD_TRANS_TIME_TO_P2

There are no TX receiver detection attributes in Versal ACAPs.

TX Out-of-Band Signaling

Each GTY 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

The following table shows the OOB signaling related ports.

Table 71: **TXOOB Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_TXCOMFINISH	Output	TXUSRCLK	Indicates completion of transmission of the last SAS or SATA COM beacon.
CH[0/1/2/3]_TXCOMINIT	Input	TXUSRCLK	Initiates transmission of the COMINIT sequence for SATA/SAS.
CH[0/1/2/3]_TXCOMSAS	Input	TXUSRCLK	Initiates transmission of the COMSAS sequence for SAS.
CH[0/1/2/3]_TXPD[1:0]	Input	TXUSRCLK	Powers down the TX lane according to the PCI Express encoding. 2'b00: P0 normal operation 2'b01: P0s low recovery time power down 2'b10: P1 longer recovery time, RecDet still on 2'b11: P2 lowest power state. Transition times between the power states are controlled by the following attributes: PD_TRANS_TIME_FROM_P2 PD_TRANS_TIME_NONE_P2 PD_TRANS_TIME_TO_P2

The following table shows the OOB signaling attributes.

Table 72: **TXOOB Attributes**

TXOOB Attributes		
Attribute	**Address**	
CH0_PIPE_CTRL_CFG0		0x0C9D
CH1_PIPE_CTRL_CFG0		0x0D9D
CH2_PIPE_CTRL_CFG0		0x0E9D
CH3_PIPE_CTRL_CFG0		0x0F9D
Label	**Bit Field**	**Description**
SATA_BURST_SEQ_LEN	[26:23]	N+1 number of bursts in a COM sequence for SAS/SATA where N is the SATA_BURST_SEQ_LEN value.
TX_PD_ELECIDLE_MODE	[22:22]	Determines if TXELECIDLE and TXPD should be treated as synchronous or asynchronous signals.
Attribute	**Address**	
CH0_TX_PCS_CFG0		0x0C77
CH1_TX_PCS_CFG0		0x0D77
CH2_TX_PCS_CFG0		0x0E77
CH3_TX_PCS_CFG0		0x0F77

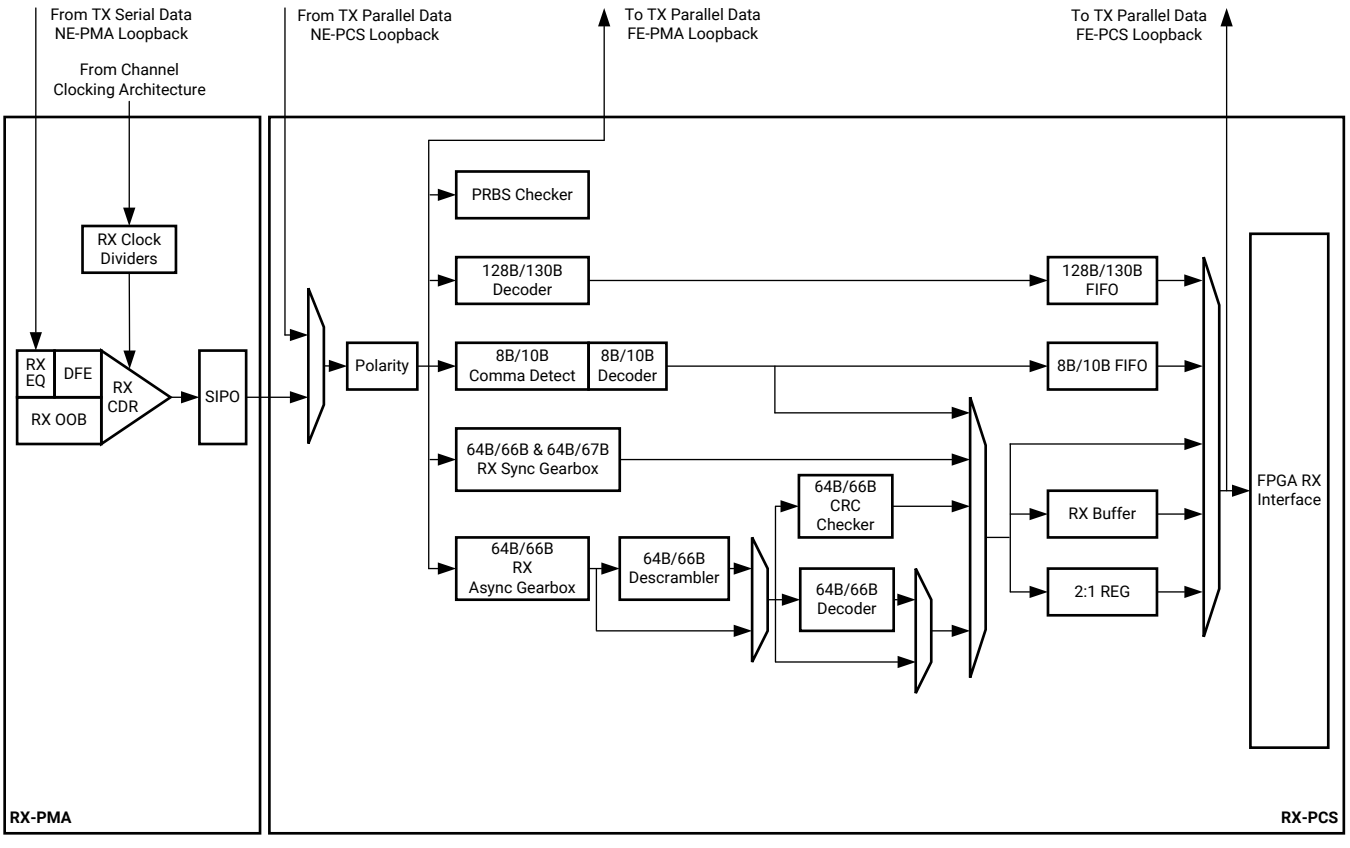
Table 72: **TXOOB Attributes** *(cont'd)*

TXOOB Attributes		
Label	**Bit Field**	**Description**
TX_IDLE_DATA_ZERO	[19:19]	When enabled; data sent to the PMA is all zeros during electrical idle. Use the recommended value from the Wizard.

Receiver

This section shows how to configure and use each of the functional blocks inside the receiver (RX). Each transceiver includes an independent receiver made up of a PCS and a PMA. The following figure shows the blocks of the transceiver RX. High-speed serial data flows from traces on the board into the PMA of the transceiver RX, into the PCS, and finally into the interconnect logic. Refer to Ring PLL and LC-Tank PLL for the description of the channel clocking architecture, which provides clocks to the RX and TX clock dividers.

Figure 65: **Transceiver RX Block Diagram**



X21332-061020

The key elements within the transceiver RX are:

1. RX Analog Front End
2. RX Out-of-Band Signaling

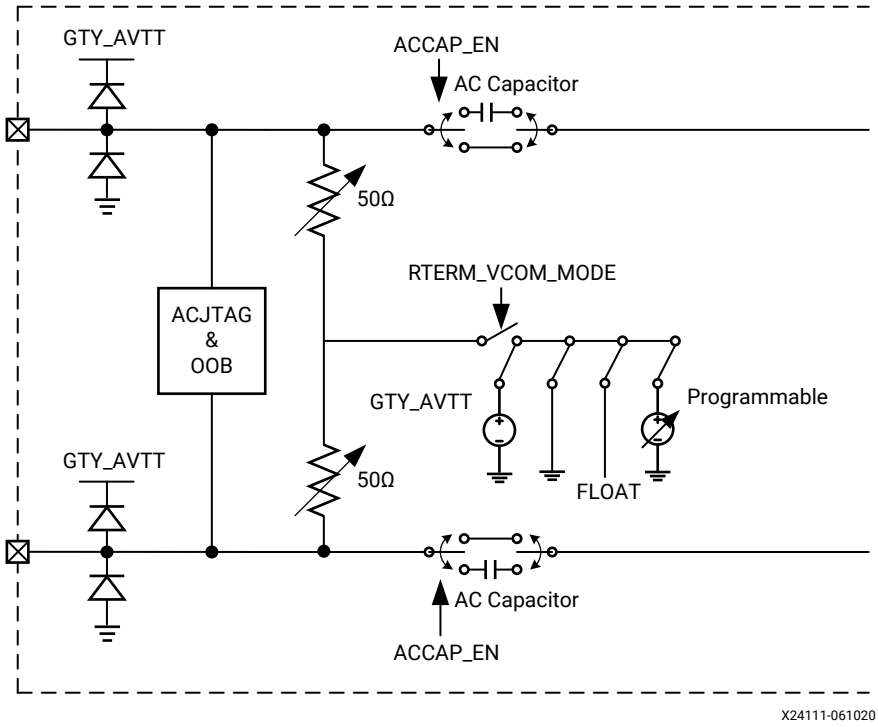
3. RX Equalizer (DFE and LPM)
4. RX CDR
5. RX Fabric Clock Output Control
6. RX Margin Analysis
7. RX Polarity Control
8. RX Pattern Checker
9. RX Byte and Word Alignment
10. RX 8B/10B Decoder
11. RX Buffer Bypass
12. RX Buffer
13. RX Clock Correction
14. RX Channel Bonding
15. RX Synchronous Gearbox
16. RX Interface

RX Analog Front End

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

- Configurable RX termination voltage
- Calibrated termination resistors

Figure 66: **RX Analog Front End**



RX Termination and Use Modes

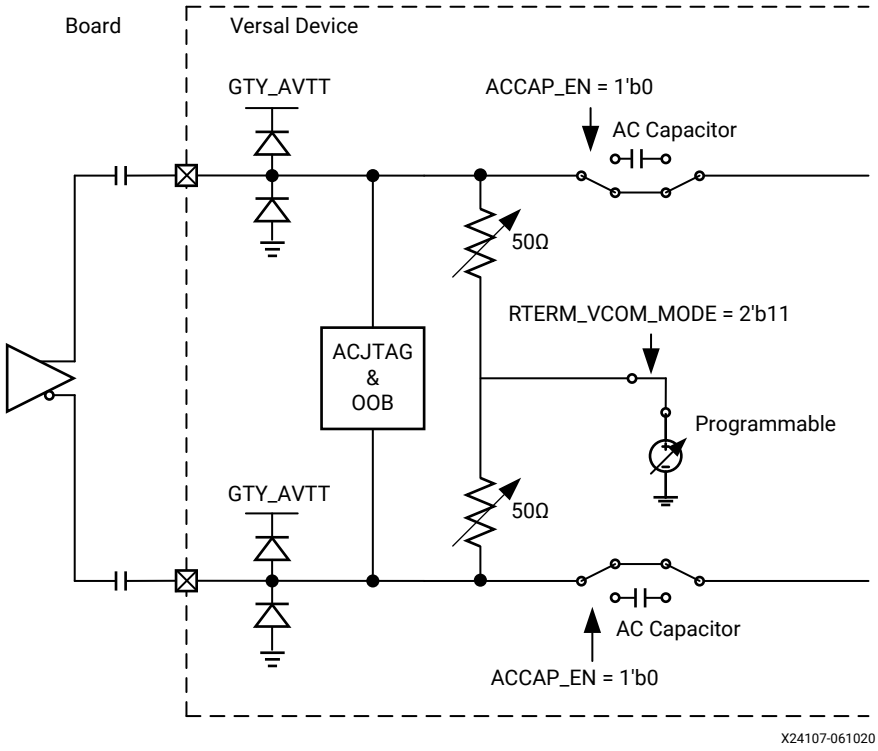
AC Coupling Use Mode with Programmable Termination

This is the typical and default transceiver operation in AC coupling mode. An external AC coupling capacitor is required. The RX termination voltage must be set to 800 mV. In this use mode, the external AC coupling capacitor must be present and should follow data sheet recommendations. The device's internal capacitor is only to be used in DC coupling use modes.

Table 73: **AC Coupling Use Mode with Programmable Termination**

Use Mode	External AC Coupling	Termination Voltage (mV)	Attribute Settings
AC Coupling	Yes	800	• ACCAP_EN = 1'b0 • LSHIFT_EN = 1'b0 • RTERM_VCOM_MODE = 2'b11 • RTERM_VCOM_LVL = 4'b1010 (800 mV)

Figure 67: **AC Coupling with Programmable Termination**



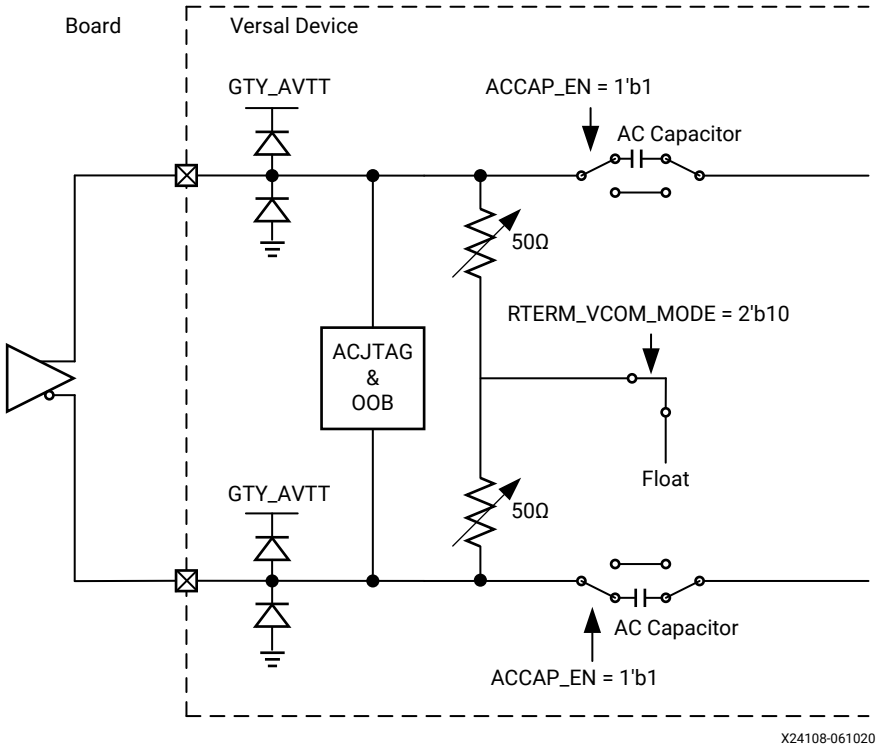
DC Coupling Use Mode with Float Termination

This is the DC coupling use mode with the RX termination set to Float. This mode enables an internal AC capacitor circuit that decouples the analog front end from the equalization block, allowing greater flexibility on the common mode input voltage. The TX or board circuit should ensure that the RX input common mode voltage is set to between 0.2V ~ 1.0V.

Table 74: **DC Coupling Use Mode with Float Termination**

Use Mode	External AC Coupling	Termination Voltage (mV)	Attribute Settings
DC Coupling	No	Float	• ACCAP_EN = 1'b1 • LSHIFT_EN = 1'b1 • RTERM_VCOM_MODE = 2'b10

Figure 68: **DC Coupling with Float Termination**



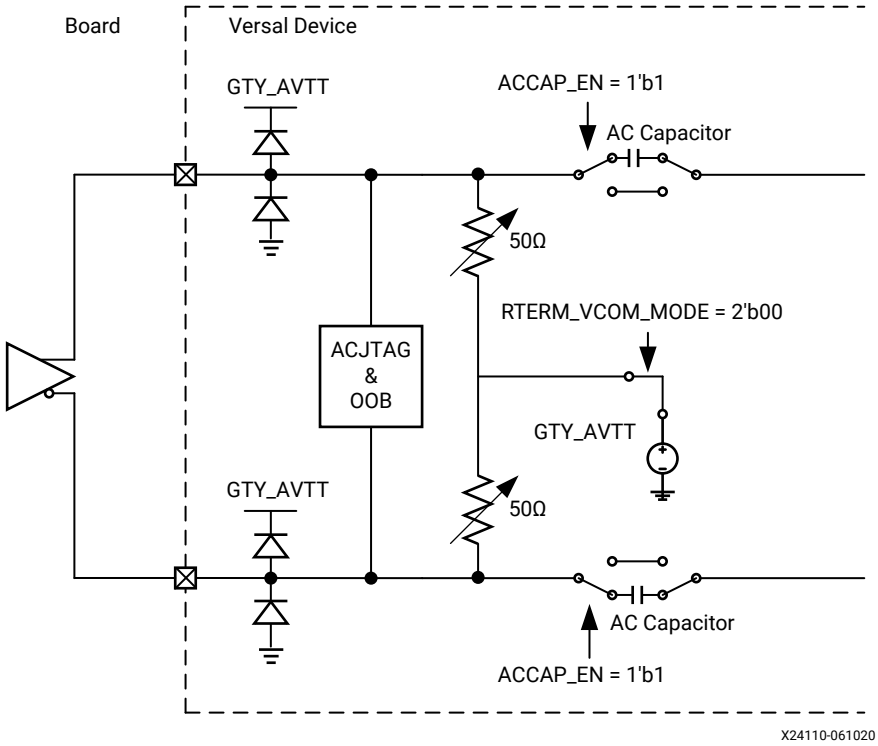
DC Coupling Use Mode with GTY_AVTT Termination

This is the DC coupling use mode with the RX termination set to GTY_AVTT. This mode enables an internal AC capacitor circuit that decouples the analog front end from the equalization block, allowing greater flexibility on the common mode input voltage. The TX or board circuit should ensure that the RX input common mode voltage is set to between 0.2V ~ 1.0V.

Table 75: **DC Coupling Use Mode with GTY_AVTT Termination**

Use Mode	External AC Coupling	Termination Voltage (mV)	Attribute Settings
DC-Coupling	No	GTY_AVTT	• ACCAP_EN = 1'b1 • LSHIFT_EN = 1'b1 • RTERM_VCOM_MODE = 2'b00

Figure 69: **DC Coupling with GTY_AVTT Termination**



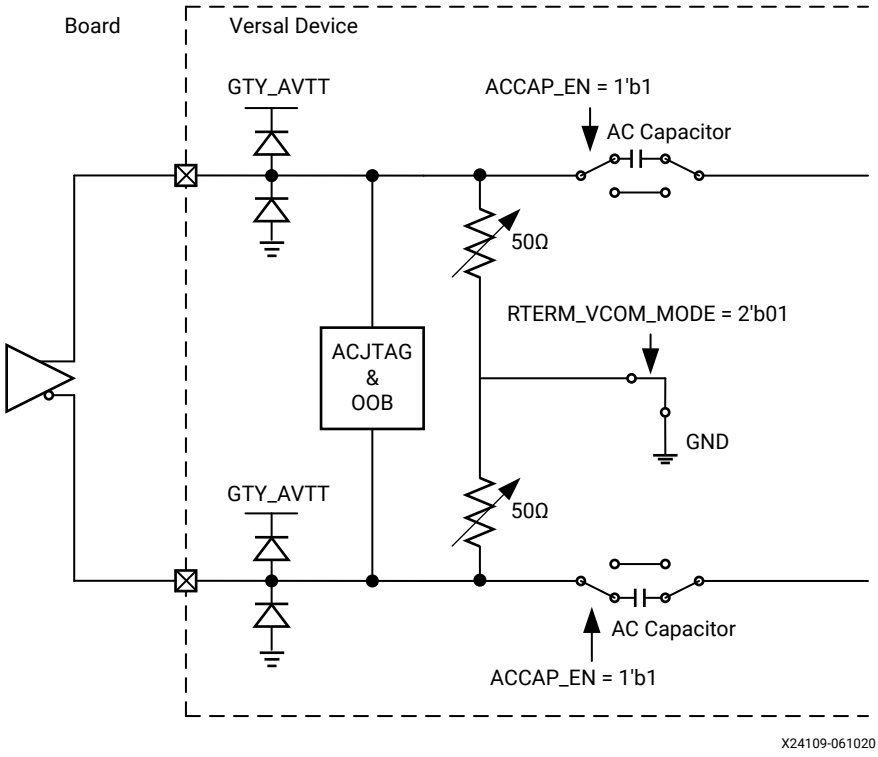
DC Coupling Use Mode with GND Termination

This is the DC coupling use mode with the RX termination set to GND. This mode enables an internal AC capacitor circuit that decouples the analog front end from the equalization block, allowing greater flexibility on the common mode input voltage. The TX or board circuit should ensure that the RX input common mode voltage is set to between 0.2V ~ 1.0V.

Table 76: **DC Coupling Use Mode with GND Termination**

Use Mode	External AC Coupling	Termination Voltage (mV)	Attribute Settings
DC-Coupling	No	GND	• ACCAP_EN = 1'b1 • LSHIFT_EN = 1'b1 • RTERM_VCOM_MODE = 2'b01

Figure 70: **DC Coupling with GND Termination**



Ports and Attributes

The following table defines the RX AFE ports.

Table 77: **RXAFE Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_GTYRXN	Input	ASYNC	Differential complement of GTYRXP forming a differential receiver input pair. Together these ports represent pads.
CH[0/1/2/3]_GTYRXP	Input	ASYNC	Differential complement of GTYRXN forming a differential receiver input pair. Together these ports represent pads.

The following table defines the RX AFE attributes.

Table 78: **RXAFE Attributes**

RXAFE Attributes	
Attribute	**Address**
CH0_RX_PAD_CFG0	0x0C63
CH1_RX_PAD_CFG0	0x0D63
CH2_RX_PAD_CFG0	0x0E63

Table 78: **RXAFE Attributes** *(cont'd)*

RXAFE Attributes		
CH3_RX_PAD_CFG0		0x0F63
Label	**Bit Field**	**Description**
LSHIFT_EN	[12:12]	Internal configuration that must be enabled in DC-Coupling use modes. 1'b0: Disabled. 1'b1: Enabled.
ACCAP_EN	[11:11]	Enables the internal AC capacitor which is required in DC-Coupling use modes. 1'b0: Bypass Mode. Allows for external AC-Coupling capacitor for AC-Coupling use mode. 1'b1: Enables internal AC-Coupling capacitor for all DC-Coupling use modes.
Attribute	**Address**	
CH0_RX_PAD_CFG1		0x0C64
CH1_RX_PAD_CFG1		0x0D64
CH2_RX_PAD_CFG1		0x0E64
CH3_RX_PAD_CFG1		0x0F64
Label	**Bit Field**	**Description**
OOB_PWRDN_B	[18:18]	OOB power up. The OOB circuit can be optionally powered down when not being used. 1'b0 : circuit power down 1'b1 : circuit power up(PCIe, SATA/SAS, protocols/ applications using OOB)
RTERM_VCOM_MODE	[5:4]	Set RX termination mode. 2'b00: GTY_AVTT 2'b01: GND 2'b10: Float 2'b11: Programmable defined by RTERM_VCOM_LVL
RTERM_VCOM_LVL	[3:0]	Controls the common mode voltage when in Programmable mode. Do not change the default setting. 4'b1010: 800 mV(Default)

RX Out-of-Band Signaling

The GTY 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. GTY 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 GTY 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 interconnect logic decodes the beacon sequence.

Ports and Attributes

The following table defines the OOB signaling related ports.

Table 79: **RXOOB Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_RXCDRHOLD	Input	ASYNC	For SATA protocol only: During the initialize electrical idle state, RXCDRHOLD should be set to 1'b1 to prevent the CDR from picking up noise on the RX.
CH[0/1/2/3]_RXCDROVRDEN	Input	ASYNC	For SATA protocol only: During the initialize electrical idle state, RXCDROVRDEN should be set to 1'b0 to prevent the CDR from picking up noise on the RX.
CH[0/1/2/3]_RXCOMINITDET	Output	RXUSRCLK	Indicates reception of the COMINIT sequence for SATA/SAS.
CH[0/1/2/3]_RXCOMSASDET	Output	RXUSRCLK	Indicates reception of the COMSAS sequence for SAS.
CH[0/1/2/3]_RXCOMWAKEDET	Output	RXUSRCLK	Indicates reception of the COMWAKE sequence for SATA/SAS.
CH[0/1/2/3]_RXELECIDLE	Output	ASYNC	This output indicates the status of OOB signal detection and is only valid for PCIe, SATA/SAS, and protocols/applications using OOB. In these cases, the OOB circuit must be powered on. The maximum line rate supported to use RXELECIDLE alone for data detection is 2.5 Gb/s. 1'b0 = Activity is seen on the receiver 1'b1 = No activity is seen For non-OOB protocols, CH0_PIPE_CTRL_CFG0[6:5] : RXELECIDLEMODE[1:0] must be set to 2'b11. RXELECIDLE outputs a static 1'b0 and in this case does not indicate signal detection status.
CH[0/1/2/3]_RXOOBRESET	Input	ASYNC	This port is driven high and then deasserted to start a single mode reset on RX OOB. The reset is not dependent on RXRESETMODE or RXPCSRESETMASK setting.

The following table defines the OOB signaling attributes.

Table 80: **RXOOB Attributes**

RXOOB Attributes		
Attribute	**Address**	
CH0_DA_CFG		0x0C5C
CH1_DA_CFG		0x0D5C
CH2_DA_CFG		0x0E5C
CH3_DA_CFG		0x0F5C
Label	**Bit Field**	**Description**
TX_DA_REFCLK_SEL	[19:19]	selects the reference clock for the RX from RPLL or the LCPLL: 1'b0 : RPLL REFCLK 1'b1 : LCPLL REFCLK
Attribute	**Address**	
CH0_FABRIC_INTF_CFG2		0x0CAB
CH1_FABRIC_INTF_CFG2		0x0DAB
CH2_FABRIC_INTF_CFG2		0x0EAB
CH3_FABRIC_INTF_CFG2		0x0FAB
Label	**Bit Field**	**Description**
A_RXOOBRESET	[25:25]	A_RXOOBRESET
Attribute	**Address**	
CH0_PIPE_CTRL_CFG0		0x0C9D
CH1_PIPE_CTRL_CFG0		0x0D9D
CH2_PIPE_CTRL_CFG0		0x0E9D
CH3_PIPE_CTRL_CFG0		0x0F9D
Label	**Bit Field**	**Description**
RX_ELECIDLE_MODE	[6:5]	Input signal to control the behavior of RXELECIDLE. 2'b00 : RXELECIDLE indicates the status of the OOB signal detection circuit. Use this setting for PCIe, SATA/SAS, and protocols/applications using OOB. In these cases, the OOB circuit must be powered on. 2'b11: RXELECIDLE outputs a static 1'b0. Use this setting for non-OOB protocols.
Attribute	**Address**	
CH0_PIPE_CTRL_CFG1		0x0C9E
CH1_PIPE_CTRL_CFG1		0x0D9E
CH2_PIPE_CTRL_CFG1		0x0E9E
CH3_PIPE_CTRL_CFG1		0x0F9E
Label	**Bit Field**	**Description**
SATA_SPEED	[31:30]	SATA speeds: 2'b00: SATA1; 2'b01: SATA2; 2'b10: SATA3; 2'b11: SAS12G

Table 80: **RXOOB Attributes** *(cont'd)*

RXOOB Attributes		
PD_OOB_PD_OVRD	[22:22]	Attribute to overwrite OOB_PD output from PIPE Powerdown logic.
OOB_PD	[21:21]	OOB power down. The OOB circuit can be optionally powered down when not being used. 1'b1 : Circuit powered down 1'b0 : Circuit powered up (PCIe, SATA/SAS, protocols/ applications using OOB)
Attribute	**Address**	
CH0_PIPE_CTRL_CFG10		0x0CBB
CH1_PIPE_CTRL_CFG10		0x0DBB
CH2_PIPE_CTRL_CFG10		0x0EBB
CH3_PIPE_CTRL_CFG10		0x0FBB
Label	**Bit Field**	**Description**
PIPE_RXPCS_RST_MASK	[20:16]	PIPE RXPCS reset masks: 4: OOB; 3: EYESCAN; 2: RX BUFFER; 1: RX PRBS; 0: RX PCS
Attribute	**Address**	
CH0_PIPE_CTRL_CFG6		0x0CA3
CH1_PIPE_CTRL_CFG6		0x0DA3
CH2_PIPE_CTRL_CFG6		0x0EA3
CH3_PIPE_CTRL_CFG6		0x0FA3
Label	**Bit Field**	**Description**
RXOOB_CLKDIV_VAL	[31:27]	Setting division factor to divide the RXREFCLK to generate 25MHz for RX OOB detection logic. The division factor is RXOOB_CLK_DIV_VAL+1.
Attribute	**Address**	
CH0_RX_OOB_CFG0		0x0CB8
CH1_RX_OOB_CFG0		0x0DB8
CH2_RX_OOB_CFG0		0x0EB8
CH3_RX_OOB_CFG0		0x0FB8
Label	**Bit Field**	**Description**
OOB_SATA_EIDLE_VAL	[29:27]	Number of idles to declare a COM match for SAS/SATA. The default value is 3'b100.
OOB_SATA_BURST_VAL	[26:24]	Number of bursts to declare a COM match for SAS/SATA. The default value is 3'b100.
OOB_SATA_MAX_INIT	[23:18]	Upper bound on idle count during COMINIT/COMRESET for SAS/SATA.
OOB_SATA_MIN_INIT	[17:12]	Lower bound on idle count during COMINIT/COMRESET for SAS/SATA.

Table 80: **RXOOB Attributes** *(cont'd)*

RXOOB Attributes		
OOB_SATA_MAX_BURST	[11:6]	Upper bound on activity burst for COM FSM for SAS/SATA.
OOB_SATA_MIN_BURST	[5:0]	Lower bound on activity burst for COM FSM for SAS/SATA.
Attribute	**Address**	
CH0_RX_OOB_CFG1		0x0CB9
CH1_RX_OOB_CFG1		0x0DB9
CH2_RX_OOB_CFG1		0x0EB9
CH3_RX_OOB_CFG1		0x0FB9
Label	**Bit Field**	**Description**
OOB_SAS_MAX_COM	[24:18]	Upper bound on idle count during COMSAS for SAS.
OOB_SAS_MIN_COM	[17:12]	Lower bound on idle count during COMSAS for SAS.
OOB_SATA_MAX_WAKE	[11:6]	Upper bound on idle count during COMWAKE for SAS/SATA.
OOB_SATA_MIN_WAKE	[5:0]	Lower bound on idle count during COMWAKE for SAS/SATA.
Attribute	**Address**	
CH0_RX_PAD_CFG1		0x0C64
CH1_RX_PAD_CFG1		0x0D64
CH2_RX_PAD_CFG1		0x0E64
CH3_RX_PAD_CFG1		0x0F64
Label	**Bit Field**	**Description**
OOB_VDIFF_INV	[24:24]	Invert the polarity of detection threshold in OOB. 1'b0 : disable(default) 1'b1 : enable
OOB_SIG_VDIFF_SEL	[23:21]	Sets the OOB signal detection threshold. Use the default value from the Wizard.
OOB_OVERRIDE	[20:19]	Debug overdrive feature for oob. 1'b0 : normal oob output 1'b1 : over-ride value from oob_override
OOB_PWRDN_B	[18:18]	OOB power up. The OOB circuit can be optionally powered down when not being used. 1'b0 : circuit power down 1'b1 : circuit power up(PCIe, SATA/SAS, protocols/ applications using OOB)
Attribute	**Address**	
CH0_RX_PCS_CFG2		0x0C67
CH1_RX_PCS_CFG2		0x0D67
CH2_RX_PCS_CFG2		0x0E67
CH3_RX_PCS_CFG2		0x0F67
Label	**Bit Field**	**Description**
PCIE_USB_ERR_REP_DIS	[31:31]	Disable error replacement for 8B10B protocols (PCIe, USB3, SATA...)

RX Equalizer (DFE and LPM)

A serial link bit error rate (BER) performance is a function of the transmitter, the transmission media, and the receiver. The transmission media or channel is bandwidth-limited and the signal traveling through it is subjected to attenuation and distortion.

There are two types of adaptive filtering available to the GTY receiver depending on system-level trade-offs between power and performance. Optimized for power with lower channel loss, the GTY receiver has a power-efficient adaptive mode named low-power mode (LPM), see Figure 71. For equalizing lossier channels, decision feedback equalization (DFE) mode is available. See Figure 72 for the GTY transceiver.

DFE mode allows better compensation of transmission channel losses by providing a closer adjustment of filter parameters than when using a linear equalizer. However, DFE mode cannot remove the pre-cursor of a transmitted bit; it only compensates for the post cursors. A linear equalizer allows pre-cursor and post-cursor gain. The RX DFE mode is a discrete-time adaptive high-pass filter. The TAP values of DFE mode are the coefficients of this filter that are set by the adaptive algorithm.

Figure 71: **LPM Mode**

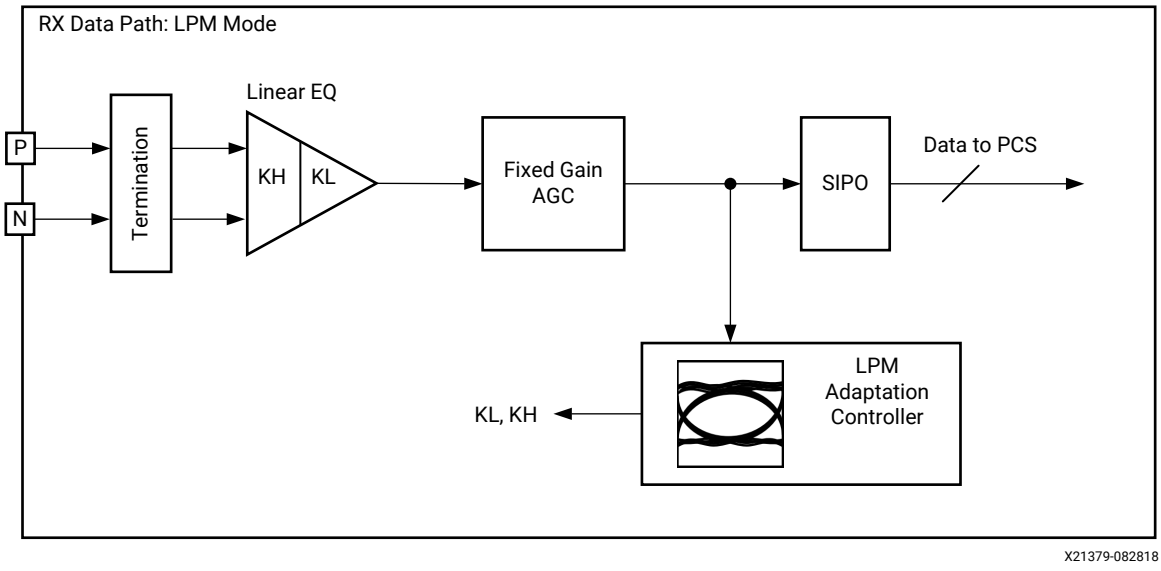
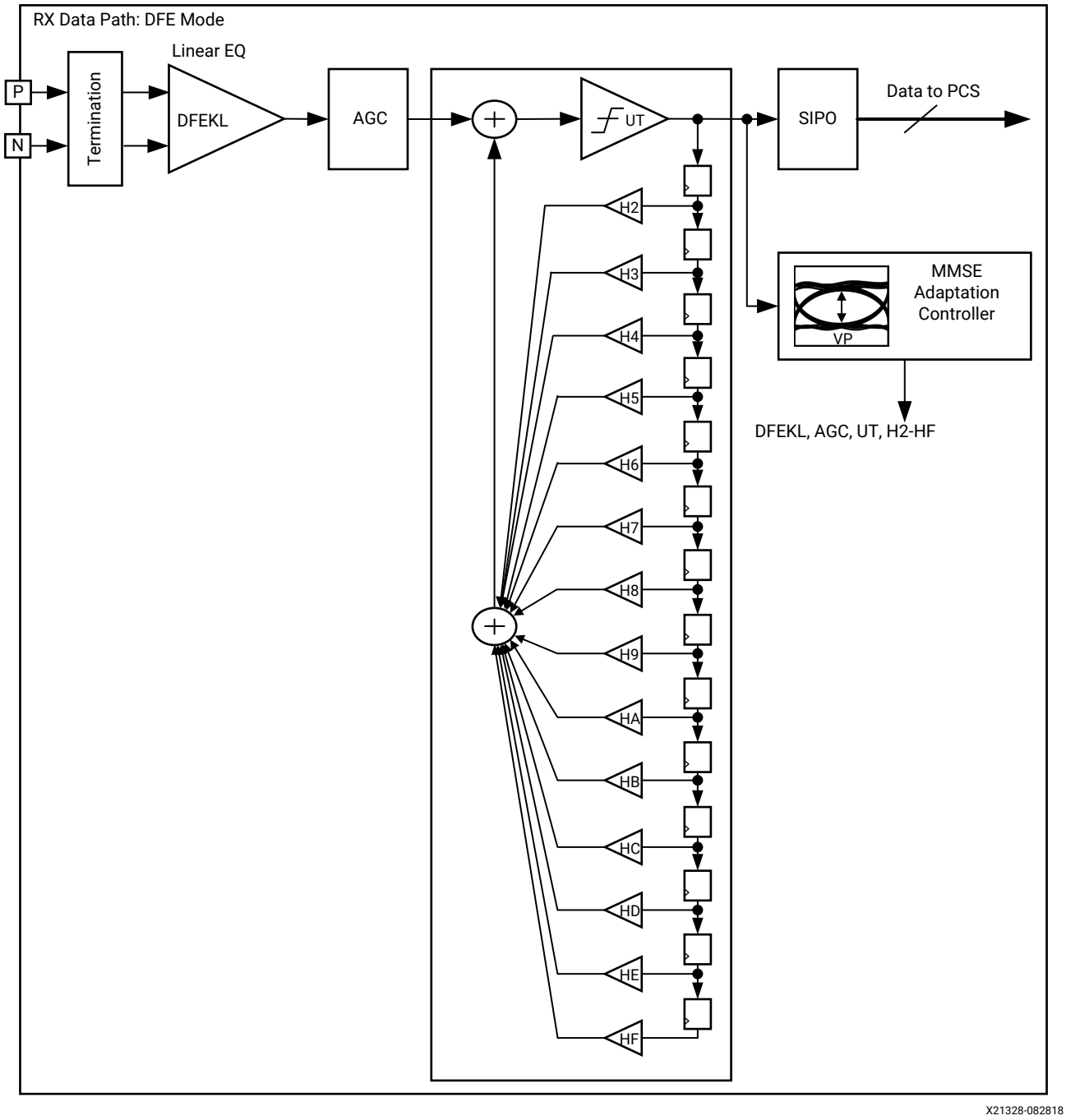


Figure 72: **DFE Mode**



Switching Between LPM and DFE Modes at Run Time

Multi-rate applications might require you to switch between LPM and DFE modes. Follow these steps to manually switch between the modes:

1. To switch from DFE to LPM, set CH*_RXLPMEN = 1.

2. Several attributes change with data rates and insertion loss. Therefore, Xilinx recommends generating DFE and LPM wrappers for the same data rate, comparing all the attributes for the differences, and then writing the corresponding values when switching between the two modes.
3. Reset the receiver's PMA by following the proper reset procedure outlined in Reset and Initialization.

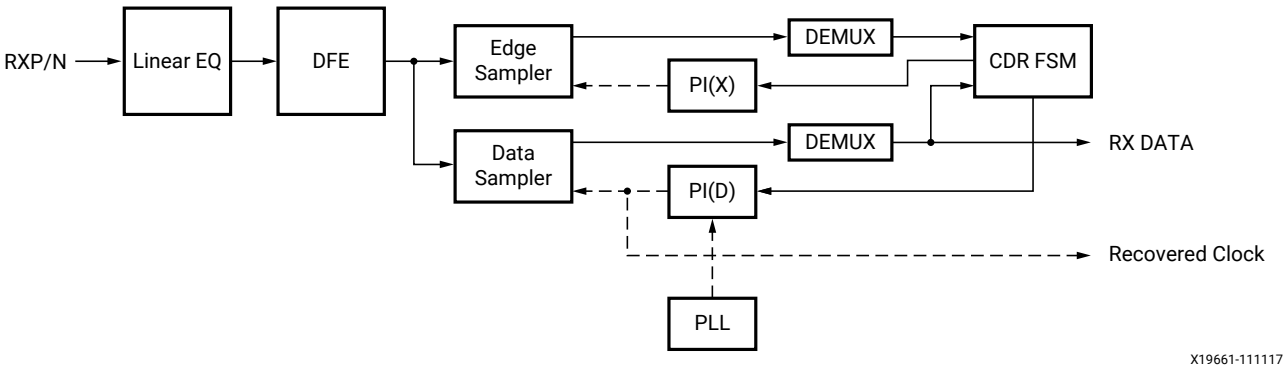
For most applications, the adaptation mode only changes when line rates reach a certain threshold where the physical channel is normally kept intact. In this case, the recommended method is to utilize the rate change function of the transceiver by generating the design through the Wizard that includes all the intended line rates. See Rate Change for more details.

When a rate change is performed the proper attribute settings, including the adaptation mode, are configured automatically, eliminating the need to run multiple single-line-rate projects as required in the manual switching method above. See RX Initialization and Reset for more information regarding the RX reset procedure.

RX CDR

The RX clock data recovery (CDR) circuit for each receiver channel in the GTYE5_QUAD extracts the recovered clock and data from an incoming data stream. The figure below illustrates the architecture of the CDR block. Clock paths are shown with dotted lines for clarity.

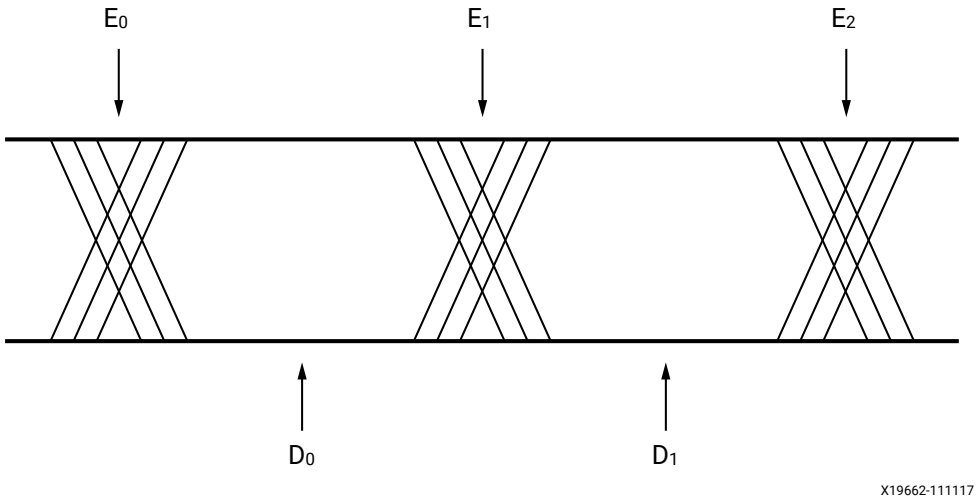
Figure 73: **CDR Detail**



The GTY transceiver employs a 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 of 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 74: **CDR Sampler Positions**



The RPLL or LCPLL 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.

Ports and Attributes

The following table defines the CDR ports.

Table 81: **CDR Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_RXCDRHOLD	Input	ASYNC	For SATA protocol only: During the initialize electrical idle state, RXCDRHOLD should be set to 1'b1 to prevent the CDR from picking up noise on the RX.
CH[0/1/2/3]_RXCDROVRDEN	Input	ASYNC	For SATA protocol only: During the initialize electrical idle state, RXCDROVRDEN should be set to 1'b0 to prevent the CDR from picking up noise on the RX.
CH[0/1/2/3]_RXCDRRESET	Input	ASYNC	This port is driven high and then deasserted to start a single mode reset on CDR. The reset is not dependent on RXRESETMODE or RXPMARESETMASK setting.

There are no CDR attributes in Versal ACAPs.

RX CDR Lock to Reference

To get the CDR to lock to reference, set CH*_RXCDRHOLD = `1'b1` and CH*_RXCDROVRDEN = `1'b0`.

RX Interface

The RX interface is the gateway to the RX datapath of the GTY transceiver. Applications receive data through the GTY transceiver by receiving data from the RXDATA port on the positive edge of RXUSRCLK. The width of the port can be configured to be two, four, or eight bytes wide. The actual width of the port depends on the RX_DATA_WIDTH and RX_INT_DATA_WIDTH attributes and whether or not 8B10B decoding is enabled. Port widths can be 16, 20, 32, 40, 64, 80, and 128 bits. The CH*_RXTRL0 and CH*_RXTRL1 ports must be used together to extend the width from 128 to 160 bits. The rate of the parallel clock (RXUSRCLK) 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. This section shows how to drive the parallel clocks and explains the constraints on those clocks for correct operation. The highest transmitter data rates require an 8-byte interface to achieve an RXUSRCLK rate in the specified operating range.

Interface Width Configuration

The GTY transceiver contains 2-byte, 4-byte, and 8-byte internal datapaths and is configurable by setting the RX_INT_DATA_WIDTH attribute. The 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, 40 bits, or 80 bits, and in this case, the RX interface only uses the CH*_RXDATA ports. For example, CH*_RXDATA[15:0] is used when the 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, 40, 64, 80, 128, or 160 bits.

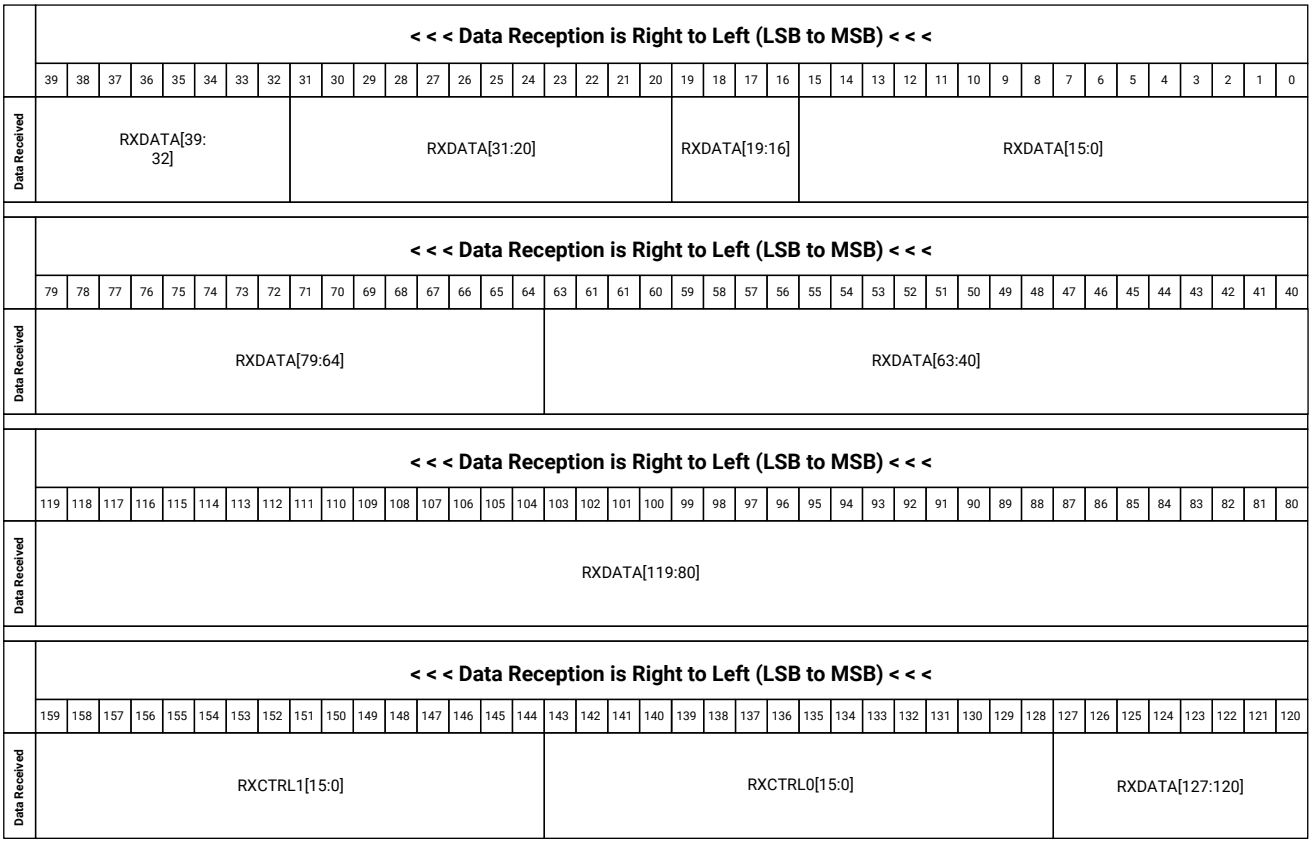
The following table shows how the interface width for the RX datapath is selected. 8B/10B decoding is described in more detail in RX 8B/10B Decoder.

Table 82: **RX Interface Datapath Configuration**

8B10B	RX_DATA_WIDTH	RX_INT_DATA_WIDTH	Interface Width	Internal Data Width
Disabled	4'b0010	2'b00	16	16
	4'b0011	2'b00	20	20
	4'b0100	2'b00	32	16
	4'b0100	2'b01	32	32
	4'b0101	2'b00	40	20
	4'b0101	2'b01	40	40
	4'b0110	2'b01	64	32
	4'b0110	2'b10	64	64
	4'b0111	2'b01	80	40
	4'b0111	2'b10	80	80
	4'b1000	2'b10	128	64
	4'b1001	2'b10	160	80
Enabled	4'b0011	2'b00	16	20
	4'b0101	2'b00	32	20
	4'b0101	2'b01	32	40

When the 8B/10B decoder is bypassed and RX_DATA_WIDTH is 160, the CH*_RXCTRL0 and CH*_RXCTRL1 ports are used to extend the RXDATA port from 128 to 160 bits. The following figure shows the data received when the 8B/10B decoder is disabled. When the RX gearbox is used, refer to RX Synchronous Gearbox for data transmission order.

Figure 75: **RX Data Received When 8B10B Decoder is Bypassed**



X21468-110119

RXUSRCLK Generation

The RX interface includes the parallel clock: RXUSRCLK. RXUSRCLK is the internal clock for the PCS logic in the transmitter. The required rate for RXUSRCLK depends on the interface width of the GTYE5_QUAD primitive and the RX line rate of the GTY transmitter. The following equation shows how to calculate the required rate for RXUSRCLK for all cases except when the RX asynchronous gearbox is used.

Equation 8: **RXUSRCLK**

$$RXUSRCLK = \frac{Line\ Rate}{Interface\ Width}$$

RXUSRCLK is the main synchronization clock for all signals into the RX side of the GTY transceiver. Most signals into the RX side of the GTY transceiver are sampled on the positive edge of RXUSRCLK. Above a given line rate, use of the 4-byte or 8-byte internal datapath is required. For details per speed grade, refer to the Versal ACAP data sheets.

Ports and Attributes

The following table defines the RX interface ports.

Table 83: **RX Interface Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_RXCTRL0[15:0]	Output	RXUSRCLK	When using RX 8B/10B decoder, active High indicates the corresponding byte shown on CH*_RXDATA is a K character when 8B/10B decoding is enabled. CH*_RXCTRL0[3] corresponds to CH*_RXDATA[31:24] CH*_RXCTRL0[2] corresponds to CH*_RXDATA[23:16] CH*_RXCTRL0[1] corresponds to CH*_RXDATA[15:8] CH*_RXCTRL0[0] corresponds to CH*_RXDATA[7:0] When using 128B/130B decoder: CH*_RXCTRL0[2]: when driven High, ignore the RX data interface for one clock cycle. CH*_RXCTRL0[3]: when driven High, indicates the start byte of a 128b block. CH*_RXCTRL0[6:5]: this signal provides the sync header to use in the current 130b block when CH*_RXCTRL0[3] is driven High.
CH[0/1/2/3]_RXCTRL1[15:0]	Output	RXUSRCLK	When using RX 8B/10B decoder, active High indicates the corresponding byte shown on CH*_RXDATA has a disparity error. CH*_RXCTRL1[3] corresponds to CH*_RXDATA[31:24] CH*_RXCTRL1[2] corresponds to CH*_RXDATA[23:16] CH*_RXCTRL1[1] corresponds to CH*_RXDATA[15:8] CH*_RXCTRL1[0] corresponds to CH*_RXDATA[7:0]
CH[0/1/2/3]_RXCTRL2[7:0]	Output	RXUSRCLK	When using RX 8B/10B decoder, active High indicates the corresponding byte shown on CH*_RXDATA is a comma character. CH*_RXCTRL2[3] corresponds to CH*_RXDATA[31:24] CH*_RXCTRL2[2] corresponds to CH*_RXDATA[23:16] CH*_RXCTRL2[1] corresponds to CH*_RXDATA[15:8] CH*_RXCTRL2[0] corresponds to CH*_RXDATA[7:0]

Table 83: **RX Interface Ports** *(cont'd)*

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_RXCTRL3[7:0]	Output	RXUSRCLK	When using RX 8B/10B decoder, active High indicates the corresponding byte shown on CH*_RXDATA was not a valid character in the 8B/10B table. CH*_RXCTRL3[3] corresponds to CH*_RXDATA[31:24] CH*_RXCTRL3[2] corresponds to CH*_RXDATA[23:16] CH*_RXCTRL3[1] corresponds to CH*_RXDATA[15:8] CH*_RXCTRL3[0] corresponds to CH*_RXDATA[7:0]
CH[0/1/2/3]_RXDATA[127:0]	Output	RXUSRCLK	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 RX_DATA_WIDTH = 64, 80: RXDATA[63:0] = 64 bits wide RX_DATA_WIDTH = 128, 160: RXDATA[127:0] = 128 bits wide When a 20-, 40-, or 80-bits bus is required, the RXCTRL0 and RXCTRL1 Port from the 8B/10B encoder is concatenated with the RXDATA port.
CH[0/1/2/3]_RXUSRCLK	Input	CLOCK	This port is used to provide a clock for the internal PCS datapath.

The following table defines the RX interface attributes.

Table 84: **RX Interface Attributes**

RX Interface Attributes		
Attribute	**Address**	
CH0_RX_PCS_CFG0		0x0C65
CH1_RX_PCS_CFG0		0x0D65
CH2_RX_PCS_CFG0		0x0E65
CH3_RX_PCS_CFG0		0x0F65
Label	**Bit Field**	**Description**
RX_INT_DATA_WIDTH	[8:7]	Controls the width of the internal datapath. 2'b00: 2-byte internal datapath 2'b01: 4-byte internal datapath 2'b10: 8-byte internal datapath

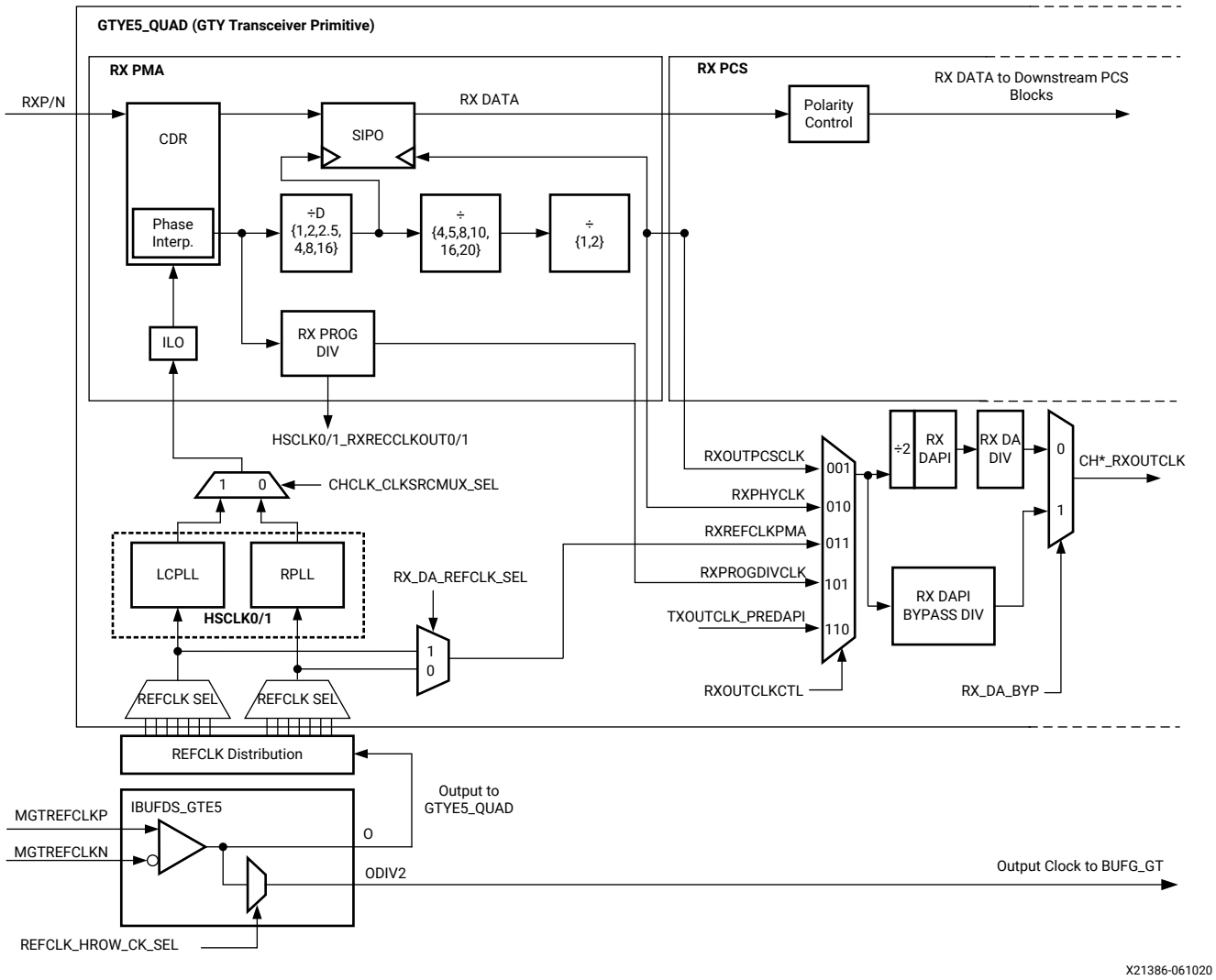
Table 84: **RX Interface Attributes** *(cont'd)*

RX Interface Attributes		
RX_DATA_WIDTH	[6:3]	Sets the bit width of the CH*_RXDATA port. When 8B/10B encoding is enabled, RX_DATA_WIDTH must be set to 20, 40 or 80. Valid settings are 16, 20, 32, 40, 64, and 80. 4'0010: 16-bit 4'0011: 20-bit 4'0100: 32-bit 4'0101: 40-bit 4'0110: 64-bit 4'0111: 80-bit 4'1000: 128-bit 4'1001: 160-bit

RX Fabric Clock Output Control

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 the figure below.

Figure 76: **RX Serial and Parallel Clock Divider**



Notes related to the figure:

1. CH*_RXOUTCLK is used as the source of the interconnect logic clock via BUFG_GT.
2. Note that the RPLL and LCPLL from HSCLK0 can only be used by RX channel 0/1, and RPLL and LCPLL from HSCLK1 can only be used by RX channel 2/3.
3. The selection of the /4, /5, /8, /10, /16, and /20 divider block, and the /1 and /2 divider block is made based on RX_DATA_WIDTH and RX_INT_DATA_WIDTH.
4. For details about placement constraints and restrictions on clocking resources (such as BUFG_GT and BUFG_GT_SYNC), refer to the *Versal ACAP Clocking Resources Architecture Manual* (AM003).
5. The clock output from IBUFDS_GTE5 should only be used after GTPOWERGOOD asserts High.

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 changed dynamically for protocols with multiple line rates.

To use the divider D in fixed line rate applications, RXOUT_DIV must be set to the appropriate value, and the CH*_RXRATE port should be tied to `8'b00000000`.

For multiple line rate applications, the CH*_TXRATE port is used to dynamically select the line rate settings, which include the appropriate divider values. See Rate Change for more details.

Parallel Clock Divider and Selector

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

The recommended clock for the interconnect logic is the CH*_RXOUTCLK from one of the GTY transceivers. It is also possible to bring the MGTREFCLK directly to the interconnect logic and use it as the interconnect logic clock. CH*_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 for more details.

The RXOUTCLKCTL attribute controls the input selector and allows these clocks to be output via the CH*_RXOUTCLK port:

- `3'b001`: RXOUTCLKPCS path is not recommended to be used because it incurs extra delay from the PCS block.
- `3'b010`: RXPHYCLK is the recovered clock that can be brought out to the interconnect 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 interconnect 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.
- `3'b011`: RXREFCLKPMA is the input reference clock to the RPLL or LCPLL, depending on the RXOUTCLKCTL setting. For usages that do not require outputting a recovered clock to the interconnect logic, RXREFCLKPMA can be used as the system clock. However, CH*_TXOUTCLK is usually used as a system clock.
- `3'b101`: RXPRODIVCLK is the divided down PLL clock after the RX programmable divider. See RX Fabric Clock Output Control for more details.
- `3'b110`: TXOUTCLK_PREDAPI is the clock source driving CH*_TXOUTCLK before going through the TX DAPI.

RX Programmable Divider

The RX programmable divider shown in RX Fabric Clock Output Control uses the recovered clock from the CDR to generate a parallel output clock. By using the recovered clock, RX programmable divider, and BUFG_GT, CH*_RXOUTCLK (RXOUTCLKCTL = `101`) can be used as a clock source for the interconnect logic instead of consuming PLL or MMCM resources in the interconnect logic. The output clock of the programmable divider can also be brought out to the transceiver reference clock pin configured as an output. The supported divider values are 4, 5, 5.5, 8, 10, 16, 16.5, 20, 32, 33, and 40.

RX DAPI and RX DAPI Bypass Dividers

The RX delay align and phase interpolator (RX DAPI) shown in RX Fabric Clock Output Control can either be enabled or bypassed to provide the CH*_RXOUTCLK.

RX_DA_BYP determines the clock path and is set based on application requirements:

- `0`: The RX DAPI clock path is used to generate the CH*_RXOUTCLK. In this use case, the RX DAPI has a built-in divide-by-2 that is always present. The divider values for RX DA DIV should be left to the default values from the Wizard example design.
- `1`: The clock path where RX DAPI is bypassed generates the CH*_RXOUTCLK. In this use case, the RX DAPI BYPASS DIV divider value should be left to the default values from the Wizard example design.

Ports and Attributes

The following table defines the ports required for RX fabric clock output control.

Table 85: **RX Fabric Clock Output Control Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_RXOUTCLK	Output	CLOCK	RXOUTCLK is the recommended clock output to the interconnect logic.
CH[0/1/2/3]_RXPROGDIVRESET	Input	ASYNC	This active-High port resets the dividers as well as the CH*_RXPROGDIVRESETDONE indicator. A reset must be performed whenever the input clock source is interrupted.
CH[0/1/2/3]_RXPROGDIVRESETDONE	Output	ASYNC	When the input clock is stable and reset is performed, this active-High signal indicates the reset is completed and the output clock is stable.

Table 85: **RX Fabric Clock Output Control Ports** *(cont'd)*

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_RXRATE[7:0]	Input	RXUSRCLK	This port is used to perform rate change on the Transceiver RX. The Wizard will preconfigure a list of desired line rates, and this port will be used to dynamically adjust the running line rate based on the preconfigured list. Set this port to the matching preconfigured line rate option value to obtained the proper line rate.

The following table defines the attributes required for RX fabric clock output control.

Table 86: **RX Fabric Clock Output Control Attributes**

RX Fabric Clock Output Control Attributes		
Attribute	**Address**	
CH0_CHCLK_ILO_CFG		0x0C6B
CH1_CHCLK_ILO_CFG		0x0D6B
CH2_CHCLK_ILO_CFG		0x0E6B
CH3_CHCLK_ILO_CFG		0x0F6B
Label	**Bit Field**	**Description**
CHCLK_CLKSRCMUX_SEL	[18:18]	This attribute selects either the RPLL or LCPLL clock as input to the ILO. 0: Select RPLL 1: Select LCPLL
Attribute	**Address**	
CH0_DA_CFG		0x0C5C
CH1_DA_CFG		0x0D5C
CH2_DA_CFG		0x0E5C
CH3_DA_CFG		0x0F5C
Label	**Bit Field**	**Description**
RX_DA_REFCLK_SEL	[3:3]	This attribute selects the reference clock input to the RPLL or the LCPLL as TXREFCLKPMA and TXREFCLKPMA_DIV2. 0: Select RPLL's reference clock 1: Select LCPLL's reference clock
Attribute	**Address**	
CH0_PIPE_CTRL_CFG3		0x0CA0
CH1_PIPE_CTRL_CFG3		0x0DA0
CH2_PIPE_CTRL_CFG3		0x0EA0
CH3_PIPE_CTRL_CFG3		0x0FA0

Table 86: **RX Fabric Clock Output Control Attributes** *(cont'd)*

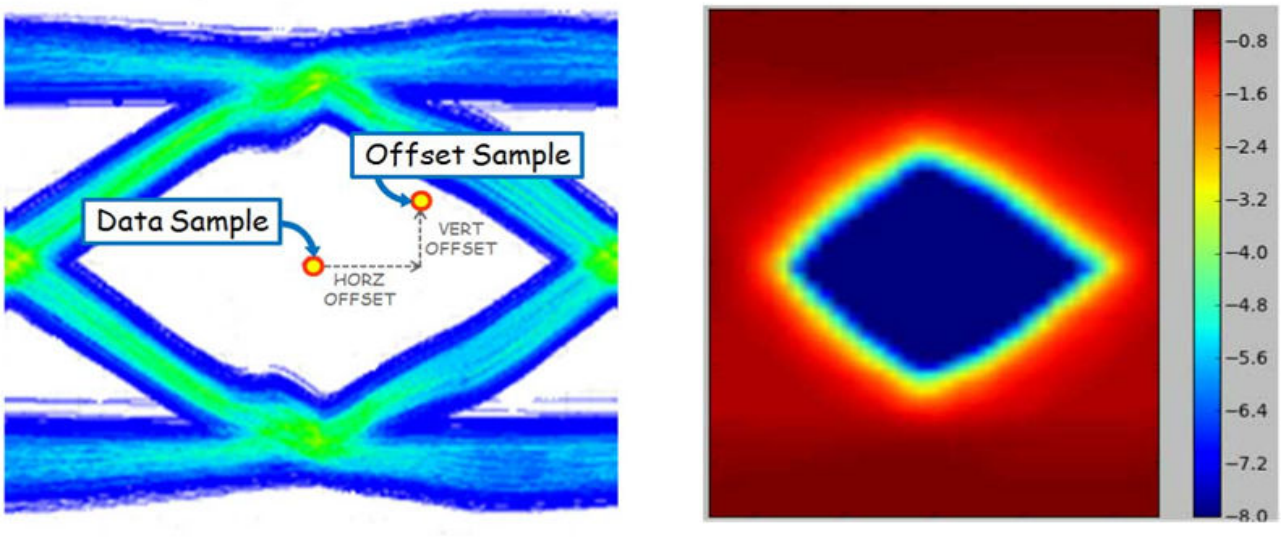
RX Fabric Clock Output Control Attributes		
Label	**Bit Field**	**Description**
RXOUT_DIV	[28:26]	This attribute selects the setting for the RX serial clock divider D. 3'b000: /1 3'b001: /2 3'b010: /4 3'b011: /8 3'b100: /16 3'b111: /2.5
Attribute	**Address**	
CH0_PIPE_CTRL_CFG7	0x0CA4	
CH1_PIPE_CTRL_CFG7	0x0DA4	
CH2_PIPE_CTRL_CFG7	0x0EA4	
CH3_PIPE_CTRL_CFG7	0x0FA4	
Label	**Bit Field**	**Description**
RX_DA_BYP	[15:15]	This attribute enables the RX DAPI bypass. 0: RX DAPI is not bypassed 1: RX DAPI is bypassed
RXOUTCLKCTL	[14:12]	This attribute selects the source for CH*_RXOUTCLK. 3'b001: RXOUTPCSCLK 3'b010: RXPHYCLK 3'b011: RXREFCLKPMA 3'b110: CH*_TXOUTCLK 3'b101: RXPROGDIVCLK
Attribute	**Address**	
CH0_PIPE_CTRL_CFG8	0x0CA5	
CH1_PIPE_CTRL_CFG8	0x0DA5	
CH2_PIPE_CTRL_CFG8	0x0EA5	
CH3_PIPE_CTRL_CFG8	0x0FA5	
Label	**Bit Field**	**Description**
RXPROGDIVSEL	[19:10]	RX programmable divider ratio. Valid settings are 4, 5, 5.5, 8, 10, 16, 16.5, 20, 32, 33, and 40. 10'b1001011000: /4 10'b1001111000: /5 10'b1110011000: /5.5 10'b0001011000: /8 10'b1001100000: /10 10'b0001000000: /16 10'b1100011000: /16.5 10'b0001100000: /20 10'b0001000010: /32 10'b0100011000: /33 10'b0001100010: /40

RX Margin Analysis

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 GTY transceiver 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 77: **Offset Sample and Data Sample to Calculate BER as a Function of Offset—the Statistical Eye**



Eye Scan Theory

The eye scan architecture can support different types of eye margin analysis, including the following:

- Statistical eye view: The eye scan block can count all data errors continuously over some period for a calculation of BER and generation of eye diagrams, as in the right side of RX Margin Analysis.
- Waveform view: Given a data pattern known to be prone to errors (or any other pattern of interest), the eye scan block can statistically determine voltage levels per bit for that pattern, allowing generation of analog waveforms for the recovered pattern.

- Scope view: Data collected by the eye scan block can be post-processed for the generation of standard scope displays such as derivative-based displays, as in the left side of RX Margin Analysis.
- Diagnostic mode: Under a variety of programmable trigger conditions, the instantaneous contents of the data buses are captured and available to be read out. This can be used, for example, to examine the pattern of burst errors due to DFE behavior.

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 RX Margin Analysis.

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 RX Margin Analysis.

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 RX Margin Analysis, 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 RX Margin Analysis) 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.

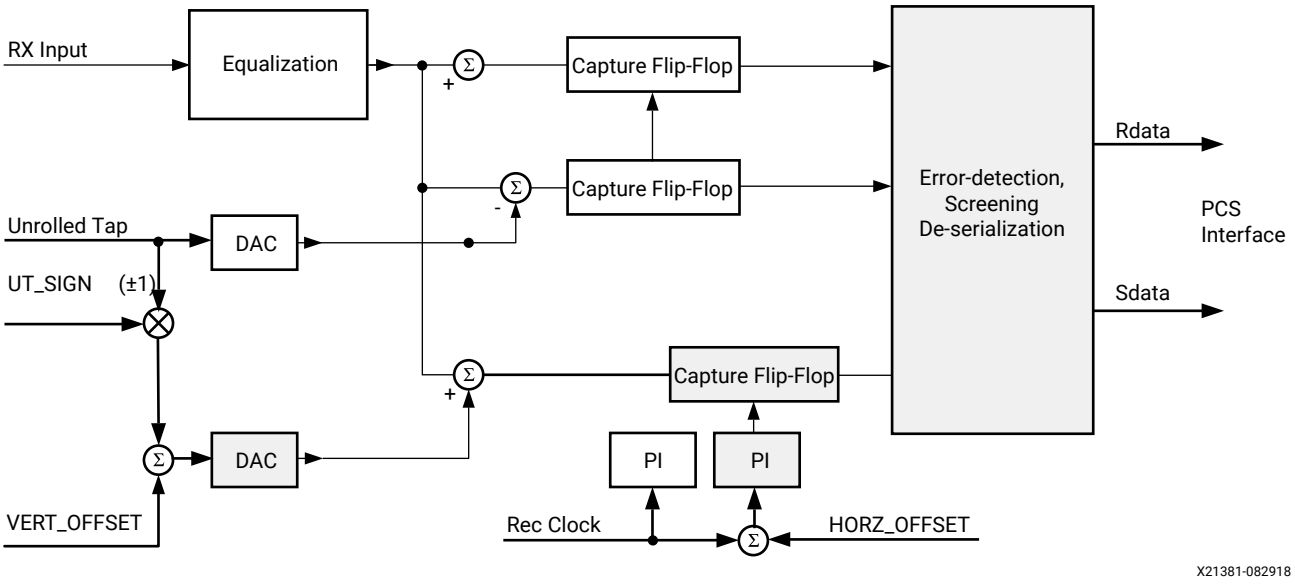
Because this functionality puts no restrictions on the data patterns being received nor requires any changes in the RX settings, it can be performed while application data is being received without error. Furthermore, no interconnect logic is required—only the ability to read and write attributes.

Eye Scan Architecture

The blocks shaded gray in the figure below 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.

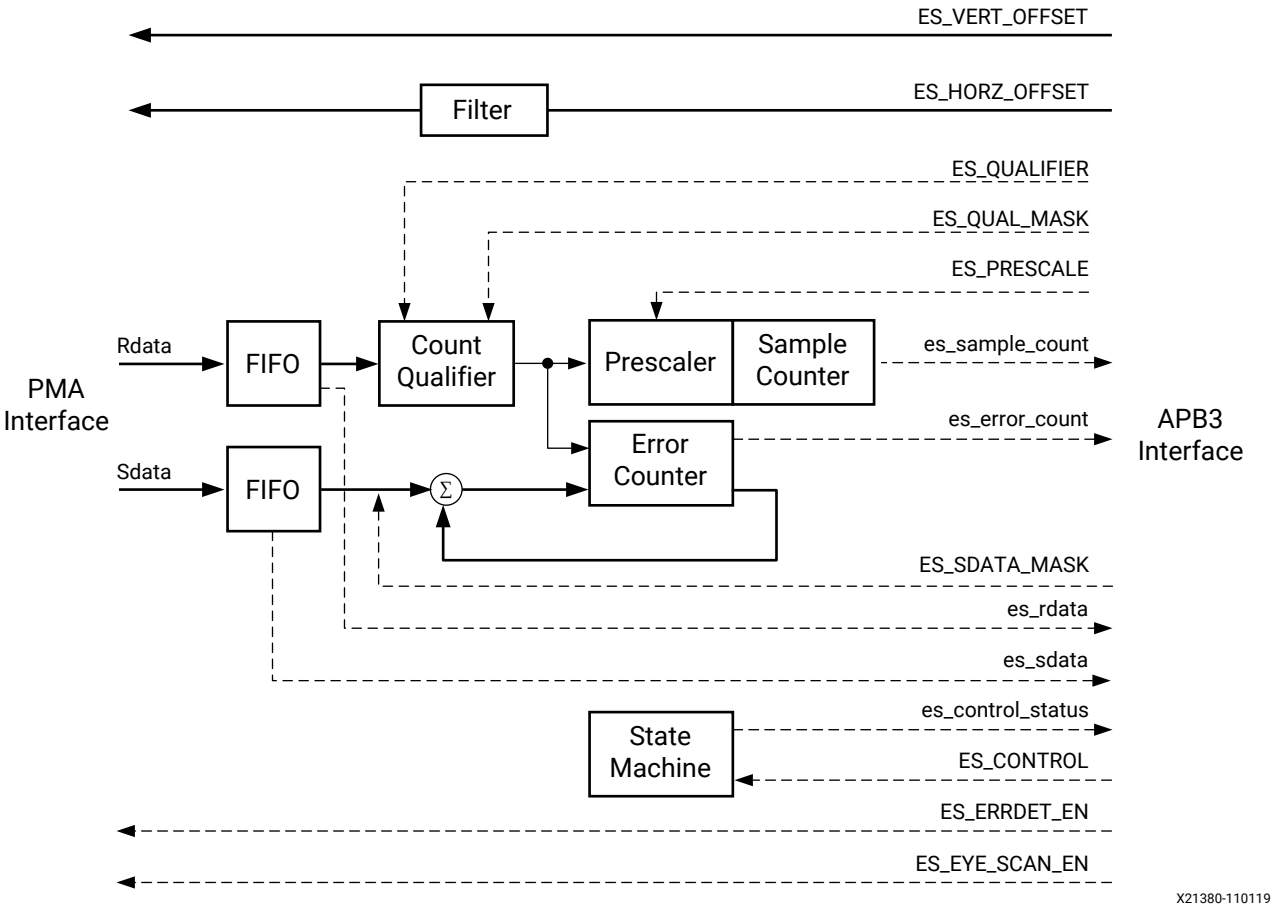
When in DFE mode (CH*_RXLPMEN=0), due to the *unrolled* first DFE tap, two separate eye scan measurements are needed, one at +UT and one at –UT, to measure the TOTAL BER at a given vertical and horizontal offset.

Figure 78: **PMA Architecture to Support Eye Scan**



The figure below describes the portion of the PCS architecture that supports eye scan. The 80-bit Rdata bus contains the data samples, and each bit of the 80-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 94.)

Figure 79: **PCS Architecture to Support Eye Scan**



In the figure above, the sample counter and error counter count the total number of bits examined and the total number of errors observed. (The sample count is scaled by ES_PRESCALE and int_datawidth. See Equation 9: BER and Equation 10: BER in DFE Mode.) The state machine controls the recording of Rdata and Sdata values in the FIFOs and the accumulation of counts in the sample counter and error counter. The functions of the various blocks in the above figure are as follows.

- The FIFOs retain the two most recent cycles (a maximum of 160 bits) of Rdata and Sdata. This data serves the following purposes:
  - Support detection of errors by examination of Sdata.
  - Support detection of desired data patterns by examination of Rdata.
  - Provide data snapshots for external examination. The state machine might stop operation of the FIFOs under certain conditions, after which the FIFO contents can be read out to interconnect logic via the APB3 interface.

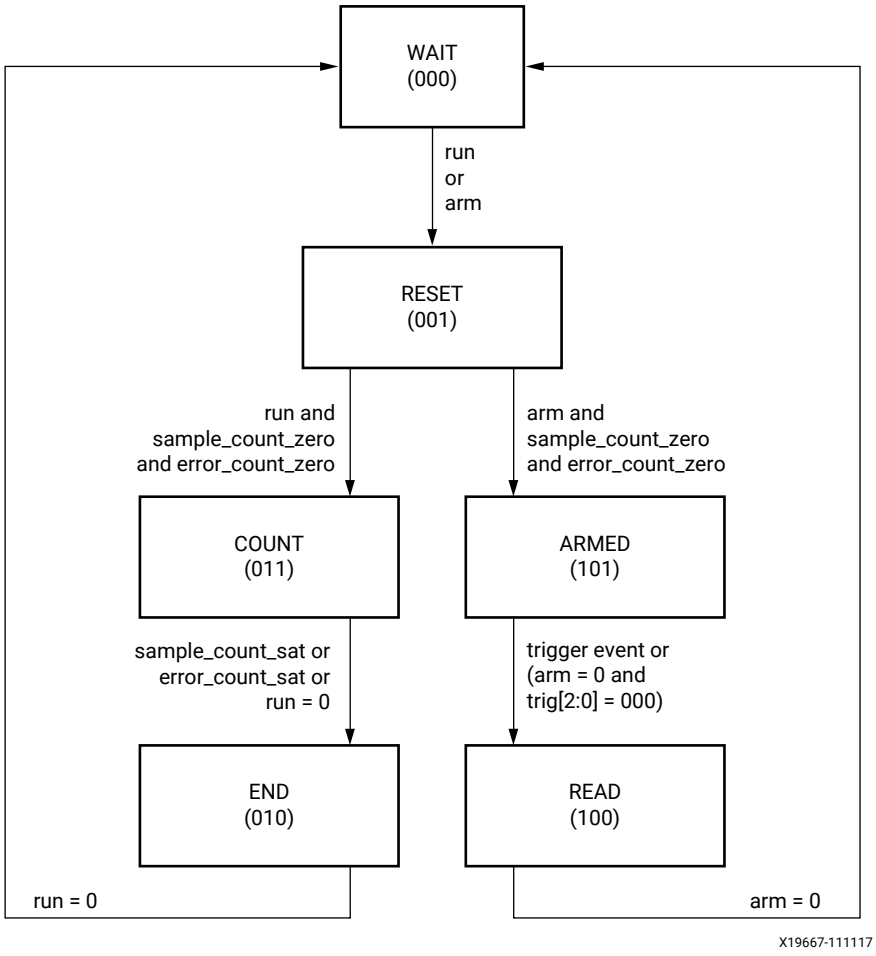
- The count qualifier compares the Rdata FIFO contents against ES_QUALIFIER. Its output goes High to indicate a match, which enables the sample counter (via the prescaler) and the error counter to advance. High ES_QUAL_MASK bits omit selected bits of Rdata from the comparison. If all the bits are High, no bits of Rdata have to match, which means operation of the counters is always enabled. Setting some ES_QUAL_MASK bits Low inhibits counter operation unless the corresponding bits of Rdata match the specified pattern in ES_QUALIFIER. The count qualifier output is High if the condition (Rdata_FIFO[k] EQUALS ES_QUALIFIER[k]) OR ES_QUAL_MASK[k] is satisfied for every k in the range 0 to 159. For a statistical eye view, ES_QUAL_MASK is set to {160{1'b1}}. For a waveform view, some ES_QUAL_MASK bits can be set to 0 to constrain counter operation or to define a trigger for capturing Rdata and Sdata for examination.
- The prescaler receives the count qualifier output signal and passes some sub-multiple of High values to the sample counter so that each increment of the sample counter corresponds to some multiple number of High outputs received from the count qualifier. ES_PRESCALE defines the number of High count qualifier outputs indicated by each sample counter increment, $2^{ES\_PRESCALE+1}$, in the range 2 to 4,294,967,296.
- The sample counter counts the total number of cycles (scaled by ES_PRESCALE) in which counting of bit errors is enabled by the count qualifier.
- The error counter accumulates an error count over time. Counting the total number of bit errors for a statistical eye view requires setting ES_SDATA_MASK[159:80] = {80{1'b1}}. For this usage, the error counter counts the total number of bit errors (1-bits) in Sdata[79:0]. High ES_SDATA_MASK bits can omit selected bits of Sdata from being checked. Commonly, a statistical eye view uses ES_SDATA_MASK[159:0] = {80{1'b1}, 80{1'b0}} for 80-bit data (see the RX Margin Analysis Attributes table). Error bits in Sdata[159:80] are not counted because Sdata[159:80] contains the previous value of Sdata[79:0] that was counted in the previous cycle. The number of bit errors counted in a given cycle is the number of values of k in the range 0 to 79 for which the condition Sdata_FIFO[k] AND NOT ES_SDATA_MASK[k] is satisfied.

  For other usages of eye scan data (such as waveform view), exactly one bit of ES_SDATA_MASK[159:0] is set to 0. The error counter counts the number of errors in Rdata_FIFO[k] (i.e., Sdata_FIFO[k] = 1) for the bit k with ES_SDATA_MASK[k] = 0. Alternatively, if ES_ERRDET_EN is FALSE, the error counter counts the total number of times the offset data Sdata_FIFO[k] is 1, regardless of whether or not this is the erroneous value. If exactly one bit of ES_SDATA_MASK[159:0] is 0, or if at least one bit of ES_SDATA_MASK[159:80] is 0, then the error counter increments by 1 if the count qualifier output is High and the condition Sdata_FIFO[k] AND NOT ES_SDATA_MASK[k] is satisfied for any k in the range 0 to 159. If any bit of ES_SDATA_MASK[159:80] is 0, then error counter increments only by 1, and not by the total number of bit errors that might otherwise be detected within a single cycle.

- The state machine controls the operation of the eye scan block to count errors or to capture snapshots of interest for Rdata and Sdata. It has two modes of operation, Run and Arm:
  - Run mode (the left loop in the figure below) supports statistical data collection for statistical eye, waveform, and scope views. It initiates operation of the sample counter and error counter as described above, stopping the operation when either the sample counter or the error counter saturates (reaches its maximum value), or when a APB3 operation terminates Run mode.
  - Arm mode (the right loop in the figure below) supports the capture of Rdata and Sdata snapshots (by disabling further FIFO operation) that can be read out through the APB3 interface. Arm mode can be used, for example, to determine data patterns that are prone to causing poor eye margin. These patterns can then drive generation of waveform views for further analysis. The state machine can be configured to stop Rdata and Sdata FIFO operation in these cases:
    - An error occurs (anywhere in Sdata_FIFO[159:0], subject to masking by ES_SDATA_MASK[159:0]).
    - Rdata matches a specified pattern (defined by ES_QUALIFIER and ES_QUAL_MASK).
    - Interconnect input EYESCANTRIGGER causes a trigger by going High.
    - A trigger is forced via an attribute write to ES_CONTROL.

The figure below documents the state transitions in the eye scan state machine.

Figure 80: **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 LPM mode, the BER is calculated as:

Equation 9: **BER**

$$BER = \frac{es\_error\_count}{es\_sample\_count \times 2^{(1+ES\_PRESCALE)} \times int\_datawidth}$$

In DFE mode, two error accumulations are required, one with EYESCAN_VS_UT_SIGN = 0 and one with EYESCAN_VS_UT_SIGN = 1. The bit error ratio is then calculated as:

Equation 10: **BER in DFE Mode**

$$BER = \frac{es\_error\_count0}{es\_sample\_count0 \times 2^{(1+ES\_PRESCALE0)} \times int\_datawidth} + \frac{es\_error\_count1}{es\_sample\_count1 \times 2^{(1+ES\_PRESCALE1)} \times int\_datawidth}$$

To maintain resolution and repeatability of these error accumulations for “deep” BER values, ES_PRESCALE must be adjusted dynamically between accumulations, balancing the need for good repeatability against the desire to not take any longer than needed. That is, 1 or 2 errors on one accumulation might be 0 or 3 or 5 errors on a repeated accumulation, changing BER significantly. But 30 errors on one accumulation changing to 27 or 35 errors on another accumulation does not change BER significantly. The following table shows the required maximum ES_PRESCALE to confirm a given BER with a given bus width.

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 APB3 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 APB3 read-only register, es_rdata[159:0], and the last two cycles of Sdata can be read from the APB3 read-only register, es_sdata[159:0].

Table 87: **ES_HORZ_OFFSET Phase Offset Decoding Table**

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`)
Quarter	–128 (`11'b11110000000`)	+0 (`11'b00000000000`)	+128 (`11'b00010000000`)
Octal	–256 (`11'b11100000000`)	+0 (`11'b00000000000`)	+256 (`11'b00100000000`)
Hex	–512 (`11'b11000000000`)	+0 (`11'b00000000000`)	+512 (`11'b01000000000`)

Table 88: **ES_SDATA_MASK Description Table**

ES_SDATA_MASK	Description
ES_SDATA_MASK0 ES_SDATA_MASK1 ES_SDATA_MASK2 ES_SDATA_MASK3 ES_SDATA_MASK4	These five 32-bit quantities comprise the 160-bit ES_SDATA_MASK. (ES_SDATA_MASK4[31:0] holds bits [159:128], etc.) This attribute masks up to two cycles of the 80-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 1's on the most recent cycle of the Sdata bus (masked by ES_SDATA_MASK[79:0]). To support the waveform view, the error counter increments by only one for any non-zero number of unmasked 1's on the previous cycle of the Sdata bus (masked by ES_SDATA_MASK[159:80]). This attribute and ES_QUAL_MASK must also mask out unused bits for bus widths narrower than 80 bits. For the statistical eye view, this attribute would assume the following values as a function of bus width: 80-bit width: ES_SDATA_MASK = ({80{1'b1}}, {80{1'b0}}) 64-bit width: ES_SDATA_MASK = ({80{1'b1}}, {64{1'b0}},{16{1'b1}}) 40-bit width: ES_SDATA_MASK = ({80{1'b1}}, {40{1'b0}},{40{1'b1}}) 32-bit width: ES_SDATA_MASK = ({80{1'b1}}, {32{1'b0}},{48{1'b1}}) 20-bit width: ES_SDATA_MASK = ({80{1'b1}}, {20{1'b0}},{60{1'b1}}) 16-bit width: ES_SDATA_MASK = ({80{1'b1}}, {16{1'b0}},{64{1'b1}}) Scope and waveform views require a sequence of measurements, unmasking only a single bit per measurement.

Table 89: **ES_QUALIFIER Description Table**

ES_QUALIFIER	Description
ES_QUALIFIER0 ES_QUALIFIER1 ES_QUALIFIER2 ES_QUALIFIER3 ES_QUALIFIER4	These five 32-bit quantities comprise the 160-bit ES_QUALIFIER. (ES_QUALIFIER4[31:0] holds bits [159:128], etc.) Eye scan can qualify BER measurement based on patterns up to 80 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 = ({130{1'b?}}, 10'b0011111010, {20{1'b?}}) ES_QUALIFIER = ({129{1'b?}}, 10'b0011111010, {21{1'b?}}) ES_QUALIFIER = ({128{1'b?}}, 10'b0011111010, {22{1'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 (80, 64, 40, 32, 20, or 16). See the description of RX_INT_DATA_WIDTH.

Table 90: **ES_QUAL_MASK Description Table**

ES_QUAL_MASK	Description
ES_QUAL_MASK0 ES_QUAL_MASK1 ES_QUAL_MASK2 ES_QUAL_MASK3 ES_QUAL_MASK4	These five 32-bit quantities comprise the 160-bit ES_QUAL_MASK. (ES_QUAL_MASK4[31:0] holds bits [159:128], etc.) This attribute masks those bits not included in the qualifier pattern. For example, the corresponding value for the K28.5 example above would be: ES_QUAL_MASK = ({130{1'b?}}, {10{1'b0}}, {20{1'b?}}) ES_QUAL_MASK = ({129{1'b?}}, {10{1'b0}}, {21{1'b?}}) ES_QUAL_MASK = ({128{1'b?}}, {10{1'b0}}, {22{1'b?}}) ...etc...

Table 91: **ES_RDATA_DWORD (Read-Only) Description Table**

ES_RDATA_DWORD	Description
ES_RDATA_DWORD0 ES_RDATA_DWORD1 ES_RDATA_DWORD2 ES_RDATA_DWORD3 ES_RDATA_DWORD4	These five 32-bit quantities comprise the 160-bit ES_RDATA. (ES_RDATA_DWORD4[31:0] hold bits [159:128], etc.) When a trigger event occurs in the ARMED state, ES_RDATA[79:0] is the present state of the Rdata bus, and ES_RDATA[159:80] is the previous state of the Rdata bus. ***Note***: This is a read-only attribute.

Table 92: **ES_SDATA_DWORD (Read-Only) Description Table**

ES_SDATA_DWORD	Description
ES_SDATA_DWORD0 ES_SDATA_DWORD1 ES_SDATA_DWORD2 ES_SDATA_DWORD3 ES_SDATA_DWORD4	These five 32-bit quantities comprise the 160-bit ES_SDATA. (ES_SDATA_DWORD4[31:0] hold bits [159:128], etc.) When a trigger event occurs in the ARMED state, ES_SDATA[79:0] is the present state of the Rdata bus, and ES_SDATA[159:80] is the previous state of the Rdata bus. ***Note***: This is a read-only attribute.

Ports and Attributes

The following table defines ports related to the RX margin.

Table 93: **RX Margin Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_EYESCANDATAERROR	Output	ASYNC	Asserts high for one REC_CLK cycle when an (unmasked) error occurs while in the COUNT or ARMED state.
CH[0/1/2/3]_EYESCANRESET	Input	ASYNC	This active-High port resets the eye scan block.
CH[0/1/2/3]_EYESCANTRIGGER	Input	RXUSRCLK	Causes a trigger event. See ES_CONTROL[4] below.
CH[0/1/2/3]_RXLPMEN	Input	ASYNC	When set to 1'b1, the LPM mode with the adaptive linear equalizer is enabled. When set to 1'b0, the high-performance DFE mode is enabled.

The following table defines RX margin attributes.

Table 94: **RX Margin Attributes**

RX Margin Attributes		
Attribute	**Address**	
CH0_ES_CONTROL_STATUS		0x085b
CH1_ES_CONTROL_STATUS		0x095b
CH2_ES_CONTROL_STATUS		0x0a5b
CH3_ES_CONTROL_STATUS		0x0b5b
Label	**Bit Field**	**Description**
ES_CONTROL_STATUS	[3:0]	[0]: DONE. Asserted High only in the WAIT, END, or READ states. [3:1]: Current state of the state machine: 000: WAIT 001: RESET 011: COUNT 010: END 101: ARMED 100: READ Note: This is a read only attribute.
Attribute	**Address**	
CH0_ES_ERROR_COUNT		0x085a
CH1_ES_ERROR_COUNT		0x095a
CH2_ES_ERROR_COUNT		0x0a5a
CH3_ES_ERROR_COUNT		0x0b5a
Label	**Bit Field**	**Description**
ES_ERROR_COUNT	[15:0]	In END and WAIT states, contains the final error count for the preceding BER measurement.
Attribute	**Address**	
CH0_ES_RDATA_DWORD0		0x0850
CH1_ES_RDATA_DWORD0		0x0950
CH2_ES_RDATA_DWORD0		0x0a50
CH3_ES_RDATA_DWORD0		0x0b50
Label	**Bit Field**	**Description**
ES_RDATA_DWORD0	[31:0]	See ES_RDATA_DWORD description table for details.
Attribute	**Address**	
CH0_ES_RDATA_DWORD1		0x0851
CH1_ES_RDATA_DWORD1		0x0951
CH2_ES_RDATA_DWORD1		0x0a51
CH3_ES_RDATA_DWORD1		0x0b51
Label	**Bit Field**	**Description**
ES_RDATA_DWORD1	[31:0]	See ES_RDATA_DWORD description table for details.

Table 94: **RX Margin Attributes** *(cont'd)*

RX Margin Attributes		
Attribute	**Address**	
CH0_ES_RDATA_DWORD2		0x0852
CH1_ES_RDATA_DWORD2		0x0952
CH2_ES_RDATA_DWORD2		0x0a52
CH3_ES_RDATA_DWORD2		0x0b52
Label	**Bit Field**	**Description**
ES_RDATA_DWORD2	[31:0]	See ES_RDATA_DWORD description table for details.
Attribute	**Address**	
CH0_ES_RDATA_DWORD3		0x0853
CH1_ES_RDATA_DWORD3		0x0953
CH2_ES_RDATA_DWORD3		0x0a53
CH3_ES_RDATA_DWORD3		0x0b53
Label	**Bit Field**	**Description**
ES_RDATA_DWORD3	[31:0]	See ES_RDATA_DWORD description table for details.
Attribute	**Address**	
CH0_ES_RDATA_DWORD4		0x0854
CH1_ES_RDATA_DWORD4		0x0954
CH2_ES_RDATA_DWORD4		0x0a54
CH3_ES_RDATA_DWORD4		0x0b54
Label	**Bit Field**	**Description**
ES_RDATA_DWORD4	[31:0]	See ES_RDATA_DWORD description table for details.
Attribute	**Address**	
CH0_ES_SAMPLE_COUNT		0x085a
CH1_ES_SAMPLE_COUNT		0x095a
CH2_ES_SAMPLE_COUNT		0x0a5a
CH3_ES_SAMPLE_COUNT		0x0b5a
Label	**Bit Field**	**Description**
ES_SAMPLE_COUNT	[31:16]	In END and WAIT states, contains the final sample count for the preceding BER measurement.
Attribute	**Address**	
CH0_ES_SDATA_DWORD0		0x0855
CH1_ES_SDATA_DWORD0		0x0955
CH2_ES_SDATA_DWORD0		0x0a55
CH3_ES_SDATA_DWORD0		0x0b55
Label	**Bit Field**	**Description**
ES_SDATA_DWORD0	[31:0]	See ES_SDATA_DWORD description table for details.

Table 94: **RX Margin Attributes** *(cont'd)*

RX Margin Attributes		
Attribute	**Address**	
CH0_ES_SDATA_DWORD1		0x0856
CH1_ES_SDATA_DWORD1		0x0956
CH2_ES_SDATA_DWORD1		0x0a56
CH3_ES_SDATA_DWORD1		0x0b56
Label	**Bit Field**	**Description**
ES_SDATA_DWORD1	[31:0]	See ES_SDATA_DWORD description table for details.
Attribute	**Address**	
CH0_ES_SDATA_DWORD2		0x0857
CH1_ES_SDATA_DWORD2		0x0957
CH2_ES_SDATA_DWORD2		0x0a57
CH3_ES_SDATA_DWORD2		0x0b57
Label	**Bit Field**	**Description**
ES_SDATA_DWORD2	[31:0]	See ES_SDATA_DWORD description table for details.
Attribute	**Address**	
CH0_ES_SDATA_DWORD3		0x0858
CH1_ES_SDATA_DWORD3		0x0958
CH2_ES_SDATA_DWORD3		0x0a58
CH3_ES_SDATA_DWORD3		0x0b58
Label	**Bit Field**	**Description**
ES_SDATA_DWORD3	[31:0]	See ES_SDATA_DWORD description table for details.
Attribute	**Address**	
CH0_ES_SDATA_DWORD4		0x0859
CH1_ES_SDATA_DWORD4		0x0959
CH2_ES_SDATA_DWORD4		0x0a59
CH3_ES_SDATA_DWORD4		0x0b59
Label	**Bit Field**	**Description**
ES_SDATA_DWORD4	[31:0]	See ES_SDATA_DWORD description table for details.
Attribute	**Address**	
CH0_EYESCAN_CFG0		0x0C7B
CH1_EYESCAN_CFG0		0x0D7B
CH2_EYESCAN_CFG0		0x0E7B
CH3_EYESCAN_CFG0		0x0F7B

Table 94: **RX Margin Attributes** *(cont'd)*

RX Margin Attributes		
Label	**Bit Field**	**Description**
ES_ERRDET_EN	[24:24]	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 error occurred, the Sdata bus would be identical to the Rdata bus. This is used for the scope and waveform views.
ES_EYE_SCAN_EN	[23:23]	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_CONTROL	[22:17]	[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 CH*_EYESCANTRIGGER port asserts High. 1000 - In the ARMED state, causes a trigger event (transition to the READ state) immediately.
ES_PRESCALE	[16:12]	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 + register value), so minimum prescale is 2^(1+0) = 2 and maximum prescale is 2^(1 + 31) = 4,294,967,296.
ES_HORZ_OFFSET	[11:0]	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 11'd0 for all data rates. The following table lists the minimum count (representing -0.5 UI) and maximum count (representing +0.5 UI) for each data rate. See ES_HORZ_OFFSET data rate decoding table for more details. [11]: Phase unification. Set this bit to 1'b0 for all positive counts including zero, and set to 1'b1 for all negative counts.

Attribute	**Address**
CH0_EYESCAN_CFG1	0x0C7C

Table 94: **RX Margin Attributes** *(cont'd)*

RX Margin Attributes		
CH1_EYESCAN_CFG1		0x0D7C
CH2_EYESCAN_CFG1		0x0E7C
CH3_EYESCAN_CFG1		0x0F7C
Label	**Bit Field**	**Description**
ES_QUALIFIER0	[31:0]	See ES_QUALIFIER description table for details.
Attribute	**Address**	
CH0_EYESCAN_CFG10		0x0C85
CH1_EYESCAN_CFG10		0x0D85
CH2_EYESCAN_CFG10		0x0E85
CH3_EYESCAN_CFG10		0x0F85
Label	**Bit Field**	**Description**
ES_QUAL_MASK4	[31:0]	See ES_QUAL_MASK description table for details.
Attribute	**Address**	
CH0_EYESCAN_CFG11		0x0C86
CH1_EYESCAN_CFG11		0x0D86
CH2_EYESCAN_CFG11		0x0E86
CH3_EYESCAN_CFG11		0x0F86
Label	**Bit Field**	**Description**
ES_SDATA_MASK0	[31:0]	See ES_SDATA_MASK description table for details
Attribute	**Address**	
CH0_EYESCAN_CFG12		0x0C87
CH1_EYESCAN_CFG12		0x0D87
CH2_EYESCAN_CFG12		0x0E87
CH3_EYESCAN_CFG12		0x0F87
Label	**Bit Field**	**Description**
ES_SDATA_MASK1	[31:0]	See ES_SDATA_MASK description table for details
Attribute	**Address**	
CH0_EYESCAN_CFG13		0x0C88
CH1_EYESCAN_CFG13		0x0D88
CH2_EYESCAN_CFG13		0x0E88
CH3_EYESCAN_CFG13		0x0F88
Label	**Bit Field**	**Description**
ES_SDATA_MASK2	[31:0]	See ES_SDATA_MASK description table for details
Attribute	**Address**	
CH0_EYESCAN_CFG14		0x0C89
CH1_EYESCAN_CFG14		0x0D89
CH2_EYESCAN_CFG14		0x0E89

Table 94: **RX Margin Attributes** *(cont'd)*

RX Margin Attributes		
CH3_EYESCAN_CFG14		0x0F89
Label	**Bit Field**	**Description**
ES_SDATA_MASK3	[31:0]	See ES_SDATA_MASK description table for details
Attribute	**Address**	
CH0_EYESCAN_CFG15		0x0C8A
CH1_EYESCAN_CFG15		0x0D8A
CH2_EYESCAN_CFG15		0x0E8A
CH3_EYESCAN_CFG15		0x0F8A
Label	**Bit Field**	**Description**
ES_SDATA_MASK4	[31:0]	See ES_SDATA_MASK description table for details
Attribute	**Address**	
CH0_EYESCAN_CFG16		0x0C8B
CH1_EYESCAN_CFG16		0x0D8B
CH2_EYESCAN_CFG16		0x0E8B
CH3_EYESCAN_CFG16		0x0F8B
Label	**Bit Field**	**Description**
EYESCAN_VS_UT_SIGN	[12:12]	1-bit binary UT sign: 0: positive unwrapped threshold 1: negative unwrapped threshold Equivalent to ES_VERT_OFFSET[8] in 7 series devices.
	[11:10]	Sets scale factor for eye scan as follows: 00: 1.5 mV/count (default) 01: 1.8 mV/count 10: 2.2 mV/count 11: 2.8 mV/count
EYESCAN_VS_NEG_DIR	[9:9]	1-bit binary offset sign: 1: negative 0: positive Equivalent to ES_VERT_OFFSET[7] in 7 series devices.
EYESCAN_VS_CODE	[8:2]	7-bit binary offset magnitude (centered on +/- UT, the unwrapped threshold). Equivalent to ES_VERT_OFFSET[6:0] in 7 series devices.
Attribute	**Address**	
CH0_EYESCAN_CFG2		0x0C7D
CH1_EYESCAN_CFG2		0x0D7D
CH2_EYESCAN_CFG2		0x0E7D
CH3_EYESCAN_CFG2		0x0F7D
Label	**Bit Field**	**Description**
ES_QUALIFIER1	[31:0]	See ES_QUALIFIER description table for details.

Table 94: **RX Margin Attributes** *(cont'd)*

RX Margin Attributes		
Attribute	**Address**	
CH0_EYESCAN_CFG3		0x0C7E
CH1_EYESCAN_CFG3		0x0D7E
CH2_EYESCAN_CFG3		0x0E7E
CH3_EYESCAN_CFG3		0x0F7E
Label	**Bit Field**	**Description**
ES_QUALIFIER2	[31:0]	See ES_QUALIFIER description table for details.
Attribute	**Address**	
CH0_EYESCAN_CFG4		0x0C7F
CH1_EYESCAN_CFG4		0x0D7F
CH2_EYESCAN_CFG4		0x0E7F
CH3_EYESCAN_CFG4		0x0F7F
Label	**Bit Field**	**Description**
ES_QUALIFIER3	[31:0]	See ES_QUALIFIER description table for details.
Attribute	**Address**	
CH0_EYESCAN_CFG5		0x0C80
CH1_EYESCAN_CFG5		0x0D80
CH2_EYESCAN_CFG5		0x0E80
CH3_EYESCAN_CFG5		0x0F80
Label	**Bit Field**	**Description**
ES_QUALIFIER4	[31:0]	See ES_QUALIFIER description table for details.
Attribute	**Address**	
CH0_EYESCAN_CFG6		0x0C81
CH1_EYESCAN_CFG6		0x0D81
CH2_EYESCAN_CFG6		0x0E81
CH3_EYESCAN_CFG6		0x0F81
Label	**Bit Field**	**Description**
ES_QUAL_MASK0	[31:0]	See ES_QUAL_MASK description table for details.
Attribute	**Address**	
CH0_EYESCAN_CFG7		0x0C82
CH1_EYESCAN_CFG7		0x0D82
CH2_EYESCAN_CFG7		0x0E82
CH3_EYESCAN_CFG7		0x0F82
Label	**Bit Field**	**Description**
ES_QUAL_MASK1	[31:0]	See ES_QUAL_MASK description table for details.
Attribute	**Address**	
CH0_EYESCAN_CFG8		0x0C83

Table 94: **RX Margin Attributes** *(cont'd)*

RX Margin Attributes		
CH1_EYESCAN_CFG8		0x0D83
CH2_EYESCAN_CFG8		0x0E83
CH3_EYESCAN_CFG8		0x0F83
Label	**Bit Field**	**Description**
ES_QUAL_MASK2	[31:0]	See ES_QUAL_MASK description table for details.
Attribute	**Address**	
CH0_EYESCAN_CFG9		0x0C84
CH1_EYESCAN_CFG9		0x0D84
CH2_EYESCAN_CFG9		0x0E84
CH3_EYESCAN_CFG9		0x0F84
Label	**Bit Field**	**Description**
ES_QUAL_MASK3	[31:0]	See ES_QUAL_MASK description table for details.
Attribute	**Address**	
CH0_RX_PCS_CFG0		0x0C65
CH1_RX_PCS_CFG0		0x0D65
CH2_RX_PCS_CFG0		0x0E65
CH3_RX_PCS_CFG0		0x0F65
Label	**Bit Field**	**Description**
RX_INT_DATA_WIDTH	[8:7]	Controls the width of the internal datapath. 2'b00: 2-byte internal datapath 2'b01: 4-byte internal datapath 2'b10: 8-byte internal datapath
RX_DATA_WIDTH	[6:3]	Sets the bit width of the CH*_RXDATA port. When 8B/10B encoding is enabled, RX_DATA_WIDTH must be set to 20, 40 or 80. Valid settings are 16, 20, 32, 40, 64, and 80. 4'0010: 16-bit 4'0011: 20-bit 4'0100: 32-bit 4'0101: 40-bit 4'0110: 64-bit 4'0111: 80-bit 4'1000: 128-bit 4'1001: 160-bit

RX Polarity Control

If the RXP and RXN differential traces are accidentally swapped on the PCB, the differential data received by the GTY transceiver RX is reversed. The GTY transceiver RX allows inversion to be done on parallel bytes in the PCS after the SIPO. This is done to offset reversed polarity on the differential pair. The polarity control function uses the CH*_RXPOLARITY input, which is driven High from the interconnect logic interface to invert the polarity.

Ports and Attributes

The following table defines the ports required by the RX polarity control function.

Table 95: **RX Polarity Control Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_RXPOLARITY	Input	RXUSRCLK	This 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

The are no RX polarity control attributes in Versal ACAPs.

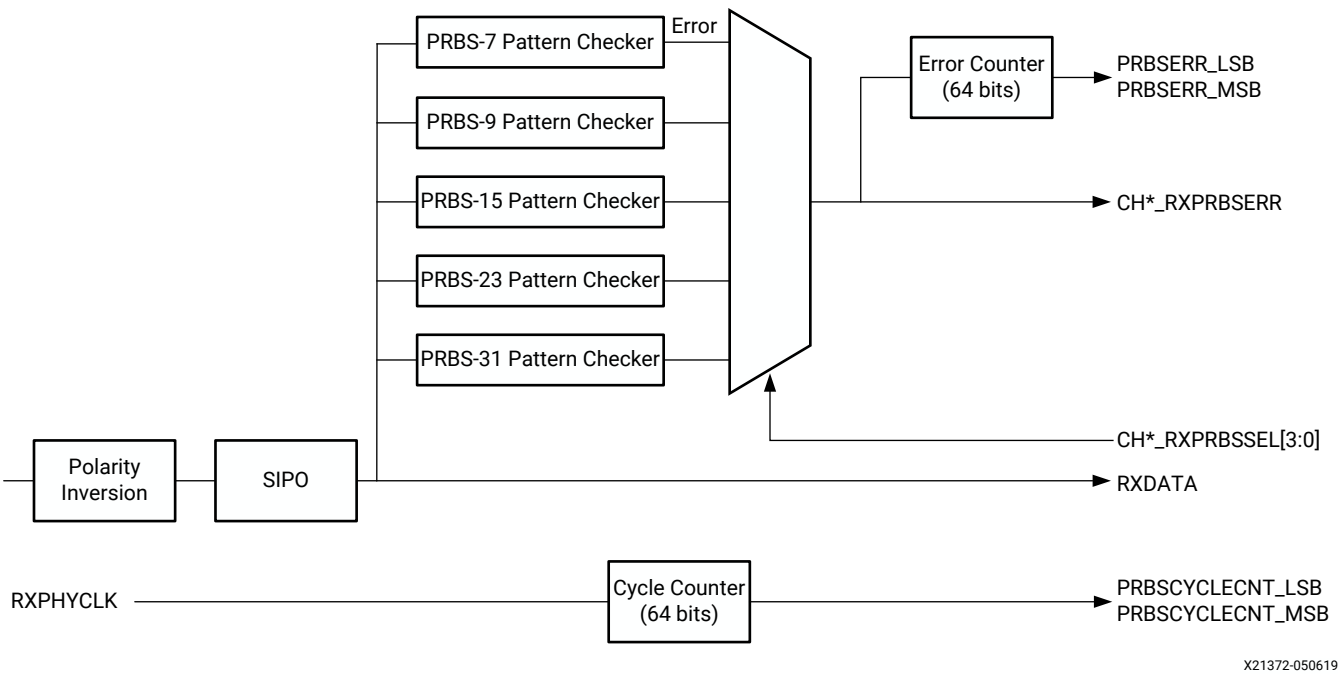
Using RX Polarity Control

CH*_RXPOLARITY can be tied High if the polarity of RXP and RXN needs to be reversed.

RX Pattern Checker

The GTY receiver includes a built-in PRBS checker (see the figure below). This checker can be set to check for one of five 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 81: **RX Pattern Checker Block**



Ports and Attributes

The following table defines the pattern checker ports.

Table 96: **RX Pattern Checker Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_RXPRBSCNTRESET	Input	ASYNC	Assert this port High to reset the PRBS error counter.
CH[0/1/2/3]_RXPRBSERR	Output	RXUSRCLK	This non-sticky status output indicates that PRBS errors have occurred. Only use PRBSERR_LSB and PRBSERR_MSB to read the precise bit error counts.
CH[0/1/2/3]_RXPRBSLOCKED	Output	RXUSRCLK	Output to indicate that the RX PRBS checker has been error free for RXPRBS_LINKACQ_CNT XCLK cycles after reset. Once asserted High, RXPRBSLOCKED does not de-assert until reset of the RX pattern checker via a reset of the RX (GTRXRESET) or a reset of the PRBS error counter (RXPRBSCNTRESET).

Table 96: **RX Pattern Checker Ports** *(cont'd)*

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_RXPRBSSEL[3:0]	Input	RXUSRCLK	Receiver PRBS checker test pattern control. Only these settings are valid: 3'b000 - Standard operation mode. (PRBS check is off) 3'b001 - PRBS-7 3'b010 - PRBS-9 3'b011 - PRBS-15 3'b100 - PRBS-23 3'b101 - PRBS-31 After changing patterns, perform a reset of the RX (GTRXRESET) or a reset of the PRBS error counter (RXPRBSCNTRESET) such that the RX pattern checker can attempt to reestablish the link acquired. No checking is done for non-PRBS patterns.

The following tables define the pattern checker attributes.

Table 97: **RX Pattern Checker Attributes**

RX Pattern Checker Attributes		
Attribute	**Address**	
CH0_PRBSCYCLECNT_LSB		0x085e
CH1_PRBSCYCLECNT_LSB		0x095e
CH2_PRBSCYCLECNT_LSB		0x0a5e
CH3_PRBSCYCLECNT_LSB		0x0b5e
Label	**Bit Field**	**Description**
PRBSCYCLECNT_LSB	[31:0]	Threshold of number of errors expected in the given time window using a cycle count. LSB [31:0] bits of this counter
Attribute	**Address**	
CH0_PRBSCYCLECNT_MSB		0x085f
CH1_PRBSCYCLECNT_MSB		0x095f
CH2_PRBSCYCLECNT_MSB		0x0a5f
CH3_PRBSCYCLECNT_MSB		0x0b5f
Label	**Bit Field**	**Description**
PRBSCYCLECNT_MSB	[31:0]	Threshold of number of errors expected in the given time window using a cycle count. MSB [63:32] bits of this counter
Attribute	**Address**	
CH0_PRBSERR_LSB		0x085c
CH1_PRBSERR_LSB		0x095c
CH2_PRBSERR_LSB		0x0a5c
CH3_PRBSERR_LSB		0x0b5c

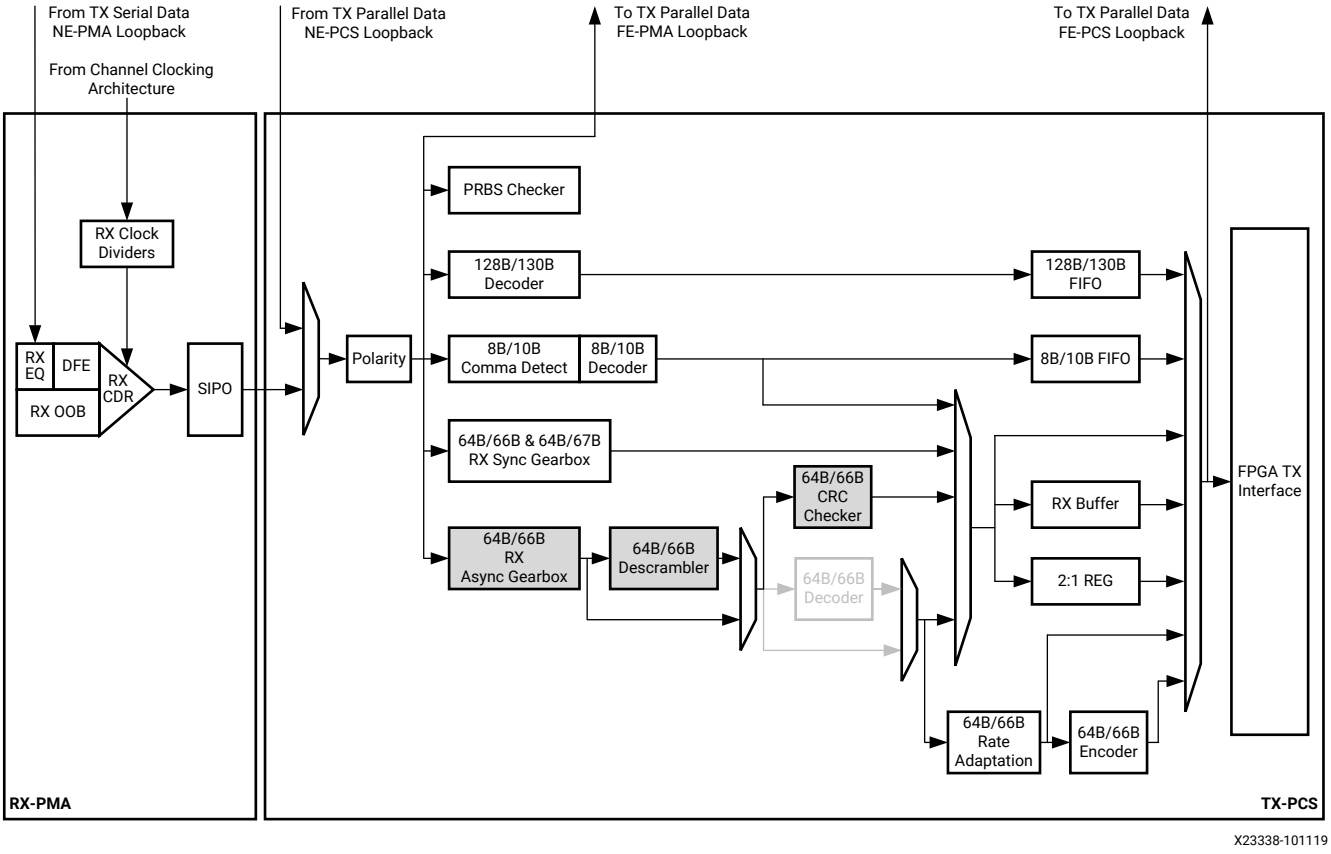
Table 97: **RX Pattern Checker Attributes** *(cont'd)*

RX Pattern Checker Attributes		
Label	**Bit Field**	**Description**
PRBSERR_LSB	[31:0]	Sum of the bit error count of the received data pattern for a time the particular PRBS mode is enabled. LSB [31:0] bits of the PRBS error counter
Attribute	**Address**	
CH0_PRBSERR_MSB	0x085d	
CH1_PRBSERR_MSB	0x095d	
CH2_PRBSERR_MSB	0x0a5d	
CH3_PRBSERR_MSB	0x0b5d	
Label	**Bit Field**	**Description**
PRBSERR_MSB	[31:0]	Sum of the bit error count of the received data pattern for a time the particular PRBS mode is enabled. MSB [63:32] bits of the PRBS error counter
Attribute	**Address**	
CH0_RX_PCS_CFG3	0x0C68	
CH1_RX_PCS_CFG3	0x0D68	
CH2_RX_PCS_CFG3	0x0E68	
CH3_RX_PCS_CFG3	0x0F68	
Label	**Bit Field**	**Description**
RXPRBS_LINKACQ_CNT	[7:0]	RX pattern checker link acquire count. Used in conjunction with output port RXPRBSLOCKED. After the RX PRBS checker has seen RX_PCS_CFG3[7:0] XCLK cycles of error-free PRBS data, RXPRBSLOCKED is asserted High. Valid range is 15-255.
Attribute	**Address**	
CH0_TX_PCS_CFG1	0x0C78	
CH1_TX_PCS_CFG1	0x0D78	
CH2_TX_PCS_CFG1	0x0E78	
CH3_TX_PCS_CFG1	0x0F78	
Label	**Bit Field**	**Description**
RXPRBS_ERR_LOOPBACK	[30:30]	Enable RXPRBSERR error injection

RX CRC Checker

The Versal ACAP GTY transceiver contains a cyclic redundancy check (CRC) checker that analyzes the incoming data checksum value for error detection. The checker supports CRC-32 for Aurora in 64B/66B mode. It does not support 10G/25G Ethernet or Interlaken.

Figure 82: **RX CRC Checker**



Using the CRC Checker

When the CRC checker is used, it needs to be enabled, and the 64B/66B encoder and 64B/66B decoder should be bypassed.

- CRC_EN should be set to `1'b1`.
- DATAPATH_CTRL can only be set to `3'b100` if the TX scrambler/RX descrambler is enabled, or `3'b111` if TX scrambler/RX descrambler is bypassed.

Another requirement is that depending on whether or not the RX 64B/66B descrambler is bypassed, the RX CRC checker must be set to the correct input endian format. The following must be set:

- When the RX 64B/66B descrambler is bypassed, INPUT_BYTE_ORDER must be set to `1'b0`.
- When the RX 64B/66B descrambler is not bypassed, INPUT_BYTE_ORDER must be set to `1'b1`.

Ports and Attributes

There are no ports for the RX CRC checker. The following table defines the RX CRC checker attributes.

Table 98: **RXCRC Attributes**

RXCRC Attributes		
Attribute	**Address**	
CH0_RX_CRC_CFG0		0x0C54
CH1_RX_CRC_CFG0		0x0D54
CH2_RX_CRC_CFG0		0x0E54
CH3_RX_CRC_CFG0		0x0F54
Label	**Bit Field**	**Description**
SOP4_CONTROL_WORD	[31:16]	When SOP_DETECT_MODE is enabled, this sets the /S4/ control word that will be used for detection of SOP.
SOP_CONTROL_WORD	[15:8]	When SOP_DETECT_MODE is enabled, this sets the /S/ control word that will be used for detection of SOP.
SOP_DETECT_MODE	[6:6]	Setting to enable the SOP detection for start of data package. 0: Disabled 1: Enabled
BYTE_MODE	[5:4]	Data width selection for the CRC. 00: 8-Byte 01: 4-Byte
CHECKSUM_PACK_MODE	[3:3]	Checksum packing format, the setting need to be set to 1'b1.
CRC_MODE	[2:2]	Setting of the CRC checker mode, the setting need to be set to 1'b0.
INPUT_BYTE_ORDER	[1:1]	Setting of the input data format. 0: Little Endian 1: Big Endian
CRC_EN	[0:0]	CRC Checker enable. 0: Disabled 1: Enabled
Attribute	**Address**	
CH0_RX_PCS_CFG3		0x0C68
CH1_RX_PCS_CFG3		0x0D68
CH2_RX_PCS_CFG3		0x0E68
CH3_RX_PCS_CFG3		0x0F68

Table 98: **RXCRC Attributes** *(cont'd)*

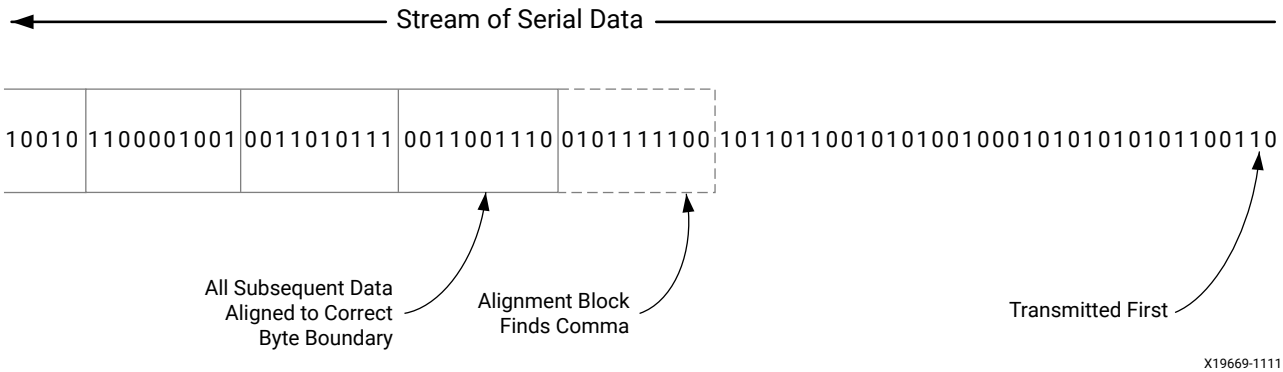
RXCRC Attributes		
Label	**Bit Field**	**Description**
DATAPATH_CTRL	[28:26]	This setting selects the correct datapath for the PCS data while going through the asynchronous 64B/66B data path. The 64B/66B Decoder, Descrambler, Rate Adaptation FIFO, and 64B/66B Encoder can be enabled or disabled independently. 3'b000: Descrambler, Decoder, Rate Adapt FIFO, and Encoder are all enabled. 3'b010: Descrambler, Decoder, and Rate Adapt FIFO are enabled. Encoder is disabled. 3'b011: Only Descrambler and Decoder are enabled. 3'b100: Only Descrambler is enabled. 3'b111: All are bypassed. In this option only the 64B/66B RX Gearbox FIFO is enabled.

RX Byte and Word Alignment

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.

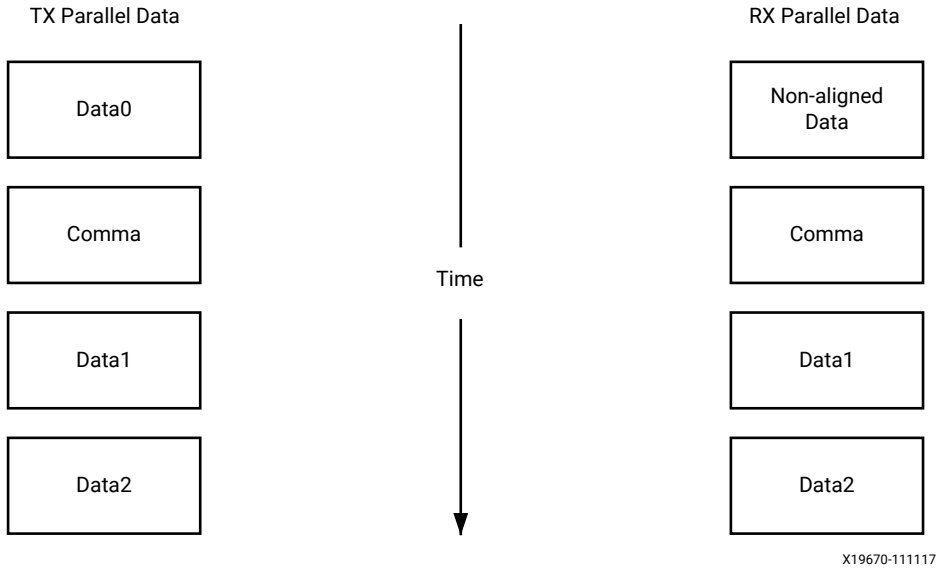
The figure below shows the alignment to a 10-bit comma. The RX receiving unaligned bits are on the right side. The serial data with the comma is highlighted in the middle. Byte aligned RX parallel data is on the left.

Figure 83: **Conceptual View of Comma Alignment (Aligning to a 10-Bit Comma)**



The figure below shows TX parallel data on the left side, and RX receiving recognizable parallel data after comma alignment on the right side.

Figure 84: **Parallel Data View of Comma Alignment (RX_SHOW_REALIGN_COMMA = TRUE)**



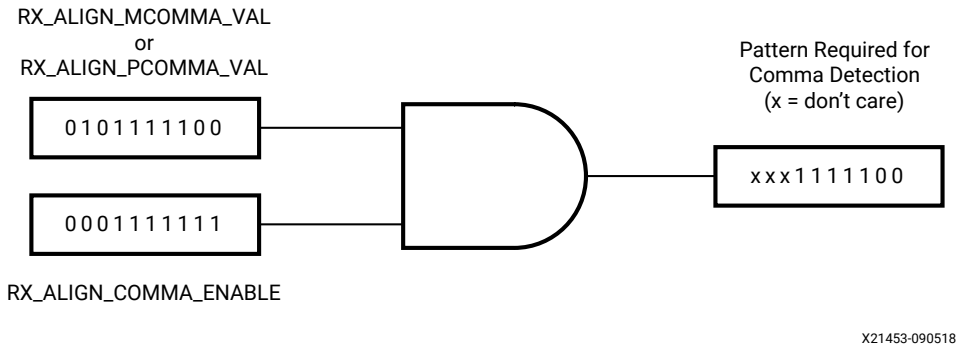
Enabling Comma Alignment

To enable the comma alignment block, attribute RX_COMMADETEN should be set to `1'b1`. RX_COMMADETEN can be set to `1'b0` to bypass the alignment block for minimum latency.

Configuring Comma Patterns

To set the comma pattern that the block searches for in the incoming data stream, the RX_ALIGN_MCOMMA_VAL, RX_ALIGN_PCOMMA_VAL, RX_PCOMMA_ALIGEN, and RX_MCOMMA_ALIGNEN attributes are used. The comma lengths depend on RX_DATA_WIDTH. The following figure shows how RX_ALIGN_COMMA_ENABLE masks each of the comma values to allow partial pattern matching.

Figure 85: **Comma Pattern Masking**

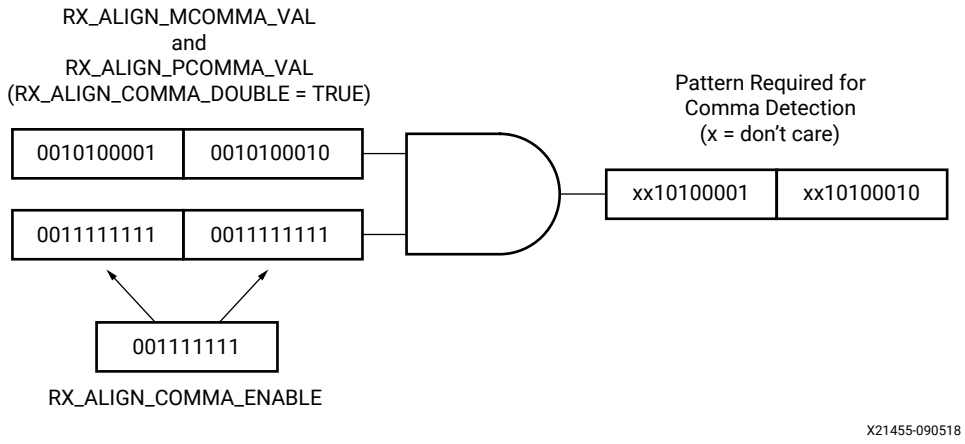


The following figure shows how the commas are combined when RX_ALIGN_COMMA_DOUBLE is TRUE.

Figure 86: **Extended Comma Pattern Definition**

The following figure shows how a comma is combined with RX_ALIGN_COMMA_ENABLE to make a wild-carded comma for a 20-bit internal comma. If RX_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 RX_ALIGN_PCOMMA_VAL, followed by an MCOMMA defined by RX_ALIGN_MCOMMA_VAL with no extra bits in between.

Figure 87: **Extended Comma Pattern Masking**



Activating Comma Alignment

Commas are aligned to a symbol boundary as long as they are found while comma alignment is active. Set attribute RX_MCOMMA_ALIGNEN to `1'b1` to align on the MCOMMA pattern. Set attribute RX_PCOMMA_ALIGNEN to `1'b1` to activate alignment on the PCOMMA pattern. Both attributes can be set to align to either pattern. When RX_ALIGN_COMMA_DOUBLE is TRUE, both enable attributes must always be set to the same value.

Alignment Status Signals

While MCOMMA or PCOMMA alignment is active, any matching comma pattern causes the block to realign to a symbol boundary. After successful alignment, the block holds CH*_RXBYTEISALIGNED High. At this time, RX_MCOMMA_ALIGNEN and RX_PCOMMA_ALIGNEN can be set to `1'b0` to turn off alignment and keep the current alignment position. RX_PCOMMA_ALIGNEN must be `1'b1` for PCOMMAs to cause CH*_RXBYTEISALIGNED to go High. Similarly, RX_MCOMMA_ALIGNEN must be `1'b1` for MCOMMAs to cause CH*_RXBYTEISALIGNED to go High. Commas can arrive while CH*_RXBYTEISALIGNED is High. If the commas arrive aligned to boundaries, there is no change. If the commas arrive out of position, the block deasserts CH*_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 CH*_RXBYTEREALIGN High for one RXUSRCLK cycle.

In applications that operate at a line rate greater than 5 Gb/s and have excessive noise in the system, the byte align block might falsely align to a wrong byte boundary and falsely assert the CH*_RXBYTEISALIGNED signal when no valid data is present. In such applications, a system-level check should be in place for checking the validity of the CH*_RXBYTEISALIGNED indicator and data.

In systems that use the RX OOB block, such as PCIe and SATA, after locking to a valid byte boundary and asserting the CH*_ RXBYTEISALIGNED signal, the byte align block might occasionally deassert the CH*_RXBYTEISALIGNED signal even when there is no change in the byte boundary. In such applications, CH*_RXBYTEISALIGNED should not be used as a valid indicator of the change in byte boundary after the first assertion.

Alignment Boundaries

The allowed boundaries for alignment are defined by RX_ALIGN_COMMA_WORD and RX_INT_DATAWIDTH. 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. The following figure shows the boundaries that can be selected.

Figure 88: **Comma Alignment Boundaries**

RX_DATA_WIDTH	RX_INT_DATAWIDTH	ALIGN_COMMA_WORD	Possible RX Alignments (Grey = Comma Can Appear on Byte)
16/20 (2-byte)	0 (2-byte)	1	Byte1 Byte0
16/20 (2-byte)	0 (2-byte)	2	Byte1 Byte0
16/20 (2-byte)	0 (2-byte)	4	Invalid Configuration
32/40 (4-byte)	0 (2-byte)	1	Byte3 Byte2 Byte1 Byte0
32/40 (4-byte)	0 (2-byte)	2	Byte3 Byte2 Byte1 Byte0
32/40 (4-byte)	0 (2-byte)	4	Invalid Configuration
32/40 (4-byte)	1 (4-byte)	1	Byte3 Byte2 Byte1 Byte0
32/40 (4-byte)	1 (4-byte)	2	Byte3 Byte2 Byte1 Byte0
32/40 (4-byte)	1 (4-byte)	4	Byte3 Byte2 Byte1 Byte0
64/80 (8-byte)	1 (4-byte)	1	Byte7 Byte6 Byte5 Byte4 Byte3 Byte2 Byte1 Byte0
64/80 (8-byte)	1 (4-byte)	2	Byte7 Byte6 Byte5 Byte4 Byte3 Byte2 Byte1 Byte0
64/80 (8-byte)	1 (4-byte)	4	Byte7 Byte6 Byte5 Byte4 Byte3 Byte2 Byte1 Byte0

X19674-111117

Ports and Attributes

The following table defines the RX byte and word alignment ports.

Table 99: **RX Byte and Word Alignment Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_RXCOMINITDET	Output	RXUSRCLK	Indicates reception of the COMINIT sequence for SATA/SAS.
CH[0/1/2/3]_RXCOMMADET	Output	RXUSRCLK	This signal is asserted when the comma alignment block detects a comma. The assertion occurs several cycles before the comma is available at the RX interface. 0: Comma is not detected. 1: Comma detected.

Table 99: **RX Byte and Word Alignment Ports** *(cont'd)*

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_RXCOMSASDET	Output	RXUSRCLK	Indicates reception of the COMSAS sequence for SAS.
CH[0/1/2/3]_RXCOMWAKEDET	Output	RXUSRCLK	Indicates reception of the COMWAKE sequence for SATA/SAS.

The following tables define the RX byte and word alignment attributes.

Table 100: **RX Byte and Word Alignment Attributes**

RX Byte and Word Alignment Attributes		
Attribute	**Address**	
CH0_RXBYTEISALIGNED	0x0874	
CH1_RXBYTEISALIGNED	0x0974	
CH2_RXBYTEISALIGNED	0x0a74	
CH3_RXBYTEISALIGNED	0x0b74	
Label	**Bit Field**	**Description**
RXBYTEISALIGNED	[0:0]	This read-only register indicates 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 CH*_RXBYTEISALIGNED is asserted before aligned data is available at the RX interface. CH*_RXBYTEISALIGNED responds to plus comma alignment when RX_PCOMMA_ALIGNEN is enabled. CH*_RXBYTEISALIGNED responds to minus comma alignment when RX_MCOMMA_ALIGNEN is enabled.
Attribute	**Address**	
CH0_RXBYTEREALIGN	0x0874	
CH1_RXBYTEREALIGN	0x0974	
CH2_RXBYTEREALIGN	0x0a74	
CH3_RXBYTEREALIGN	0x0b74	
Label	**Bit Field**	**Description**
RXBYTEREALIGN	[1:1]	This read-only register 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 replaced when alignment occurs, which can cause data errors (and disparity errors when the 8B/10B decoder is used.)
Attribute	**Address**	
CH0_RXCOMMADET	0x0874	
CH1_RXCOMMADET	0x0974	

Table 100: **RX Byte and Word Alignment Attributes** *(cont'd)*

RX Byte and Word Alignment Attributes		
CH2_RXCOMMADET		0x0a74
CH3_RXCOMMADET		0x0b74
Label	**Bit Field**	**Description**
RXCOMMADET	[2:2]	This read-only register indicates the comma alignment block detects a comma. The assertion occurs several cycles before the comma is available at the RX interface. 0: Comma is not detected. 1: Comma is detected.
Attribute	**Address**	
CH0_RX_PCS_CFG0		0x0C65
CH1_RX_PCS_CFG0		0x0D65
CH2_RX_PCS_CFG0		0x0E65
CH3_RX_PCS_CFG0		0x0F65
Label	**Bit Field**	**Description**
RX_ALIGN_MCOMMA_DET	[31:31]	Controls the raising of RXCOMMADET on comma plus. 0 (FALSE): Do not raise RXCOMMADET when comma minus is detected. 1 (TRUE): Raise RXCOMMADET when comma minus is detected. (This setting does not affect comma alignment.)
RX_ALIGN_PCOMMA_DET	[30:30]	Controls the raising of RXCOMMADET on comma plus. 0 (FALSE): Do not raise RXCOMMADET when comma plus is detected. 1 (TRUE): Raise RXCOMMADET when comma plus is detected. (This setting does not affect comma alignment.)
RX_ALIGN_MCOMMA_VAL	[29:20]	Defines comma plus to raise RXCOMMADET and align parallel data. The reception order is right to left. (RX_ALIGN_MCOMMA_VAL[0] is received first.) The default value is 10'b1010000011 (K28.5). This definition does not affect 8B/10B encoding or decoding.
RX_ALIGN_PCOMMA_VAL	[19:10]	Defines comma plus to raise RXCOMMADET and align parallel data. The reception order is right to left. (RX_ALIGN_PCOMMA_VAL[0] is received first.) The default value is 10'b0101111100 (K28.5). This definition does not affect 8B/10B encoding or decoding.
RX_MCOMMA_ALIGNEN	[2:2]	Aligns the byte boundary when comma minus is detected. 0: Disabled. 1: Enabled.
RX_PCOMMA_ALIGNEN	[1:1]	Aligns the byte boundary when comma plus is detected. 0: Disabled. 1: Enabled.
RX_COMMADETEN	[0:0]	This attribute 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.

Table 100: **RX Byte and Word Alignment Attributes** *(cont'd)*

RX Byte and Word Alignment Attributes		
Attribute	**Address**	
CH0_RX_PCS_CFG1		0x0C66
CH1_RX_PCS_CFG1		0x0D66
CH2_RX_PCS_CFG1		0x0E66
CH3_RX_PCS_CFG1		0x0F66
Label	**Bit Field**	**Description**
RX_SLIDE_MODE	[20:19]	Defines the RXSLIDE mode. 00 (OFF): Default setting. The RXSLIDE feature is not used. 10 (PCS): PCS is used to perform the bit-slipping function. RXSLIDE is driven High for one RXUSRCLK cycle to shift the parallel data (RXDATA) to the left by one bit within the comma alignment boundary determined by the RX_ALIGN_COMMA_WORD, RX_DATA_WIDTH, and RX_INT_DATA_WIDTH settings. In this mode, even if RXOUTCLK is sourcing from the RX PMA, the clock phase remains the same. This option requires RX_SHOW_REALIGN_COMMA to be FALSE. 01 (AUTO): This is an automated PMA mode without using the interconnect 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 RX_SHOW_ALIGN_COMMA to be FALSE. When RX buffer bypass is used, RX_SLIDE_MODE cannot be set to AUTO or PMA.
RX_SHOW_REALIGN_COMMA	[18:18]	Defines if a comma that caused realignment is brought out to the RX. 0 (FALSE): Do not bring the comma that causes realignment to the RX. This setting reduces RX datapath latency. 1 (TRUE): Bring the realignment comma to the RX. RX_SHOW_REALIGN_COMMA = 1'b1 should not be used when ALIGN_COMMA_DOUBLE = 1'b1 or when manual alignment is used.
RX_ALIGN_COMMA_WORD	[13:11]	This attribute controls and alignment of detected commas within a multi-byte datapath. 001: Align comma to either of the 2 bytes for a 2-byte interface, any of the 4 bytes for a 4-byte interface, any of the 8 bytes for an 8-byte interface. 010: 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, RXDATA[9:0]/ RXDATA[29:20]/RXDATA[49:40]/RXDATA[69:60] for an 8-byte interface. 100: Align comma to a 4-byte boundary. This setting is not allowed for RX_INT_DATA_WIDTH = 0. The aligned comma is guaranteed to be aligned to RXDATA[9:0] for a 4-byte interface, and RXDATA[9:0]/RXDATA[49:40] for an 8-byte interface. Protocols that send commas in even and odd positions must set RX_ALIGN_COMMA_WORD to 3'b001.

Table 100: **RX Byte and Word Alignment Attributes** *(cont'd)*

RX Byte and Word Alignment Attributes		
RX_ALIGN_COMMA_DOUBLE	[10:10]	Specifies whether a comma match consists of either a comma plus or a comma minus alone, or if both are required in the sequence. 0 (FALSE): The plus comma (PCOMMA) and minus comma (MCOMMA) are handled separately. An individual match for either can lead to comma detection and alignment. 1 (TRUE): A comma match consists of a comma plus followed immediately by a comma minus. The match pattern is 20 of 16 bits (as determined by RX_DATA_WIDTH). When ALIGN_COMMA_DOUBLE is TRUE, RX_ALIGN_PCOMMA_DET must be the same as RX_ALIGN_MCOMMA_DET, and RX_PCOMMA_ALIGNEN must be the same as RX_MCOMMA_ALIGNEN.
RX_ALIGN_COMMA_ENABLE	[9:0]	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 10'b1111111111. 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.

RX 8B/10B Decoder

If RX received data is 8B/10B encoded, it must be decoded. The GTY transceiver has a built-in 8B/10B encoder in the GTY transceiver TX and an 8B/10B decoder in the GTY transceiver RX. This includes four one-byte 8B/10B decoder modules on the datapath to decode data without consuming device resources. The RX 8B/10B decoder has these features:

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

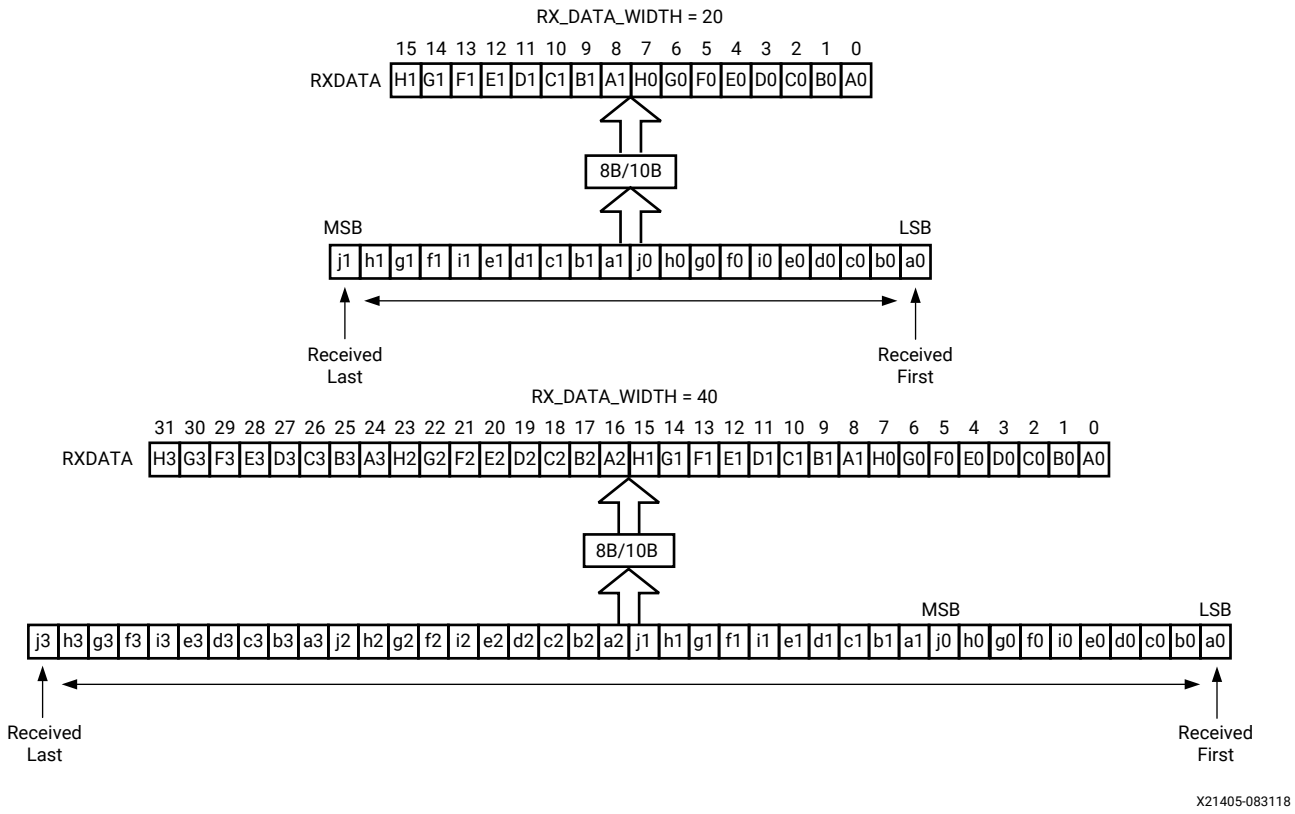
RX 8B/10B Bit and Byte Ordering

The order of the bits into the 8B/10B decoder is the opposite of the order shown in Chapter 5: 8B/10B Valid Characters. 8B/10B decoding requires bit a0 to be received first, but the GTY 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 CH*_RXDATA ports. The 8B/10B decoder does not support RX_DATA_WIDTH = 80 or 160. RX_INT_DATAWIDTH must be set to 0 (2-byte internal datapath) or 1 (4-byte internal datapath). Data is reconstructed into bytes and sent to the CH*_RXDATA interface after the 8B/10B decoder. The number of bits used by CH*_RXDATA and corresponding byte orders are determined by RX_DATA_WIDTH.

- Only use CH*_RXDATA[15:0] if RX_DATA_WIDTH = 20
- Only use CH*_RXDATA[31:0] if RX_DATA_WIDTH = 40

When the 8B/10B decoder is bypassed but RX_DATA_WIDTH is set to multiples of 10, 10-bit characters are passed to the RX data interface with the format shown in the following figure.

Figure 89: **8B10B Decoder Bit and Byte Order**



RX Running Disparity

Disparity check is performed, and the decoder drives the corresponding CH*_RXCTRL1 High when the data byte on CH*_RXDATA arrives with the wrong disparity. The decoder drives the RXCTRL3 port High when the decoder is enabled, but a received 10-bit character cannot be mapped into a valid 8B/10B character listed in Chapter 5: 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 RXCTRL1 represents the 9th bit
- The corresponding RXCTRL0 represents the 8th bit
- The corresponding RXDATA byte represents the [7:0] bits

Special Characters

8B/10B decoding includes special characters (K characters) that are often used for control functions. When CH*_RXDATA is a K character, the decoder drives CH*_RXCTRL0 High.

If RX_DEC_PCOMMA_DET is set to TRUE, the decoder drives the corresponding RXCTRL2 High whenever CH*_RXDATA is a positive 8B/10B comma. If RX_DEC_MCOMMA_DET is TRUE, the decoder drives the corresponding RXCTRL2 bit High whenever CH*_RXDATA is a negative 8B/10B comma.

Enabling and Disabling 8B/10B Decoding

To enable the 8B/10B decoder, RX_8B10B_EN must be driven High. RX_DATA_WIDTH must be set to a multiple of 10 (20 or 40) with the 8B/10B decoder enabled.

To disable the 8B/10B decoder on the GTY receiver path, RX_8B10B_EN must be driven Low. When the decoder is disabled, RX_DATA_WIDTH can be set to a multiple of 8 or 10 (16, 20, 32, 40, 64, 80, 128, or 160). The operation of the CH*_RXDATA port with 8B/10B decoding bypassed is described in RX Interface.

Ports and Attributes

The following table defines the ports required by RX 8B/10B decoder.

Table 101: **8B10BDE Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_RXCTRL0[15:0]	Output	RXUSRCLK	When using RX 8B/10B decoder, active High indicates the corresponding byte shown on CH*_RXDATA is a K character when 8B/10B decoding is enabled. CH*_RXCTRL0[3] corresponds to CH*_RXDATA[31:24] CH*_RXCTRL0[2] corresponds to CH*_RXDATA[23:16] CH*_RXCTRL0[1] corresponds to CH*_RXDATA[15:8] CH*_RXCTRL0[0] corresponds to CH*_RXDATA[7:0] When using 128B/130B decoder: CH*_RXCTRL0[2]: when driven High, ignore the RX data interface for one clock cycle. CH*_RXCTRL0[3]: when driven High, indicates the start byte of a 128b block. CH*_RXCTRL0[6:5]: this signal provides the sync header to use in the current 130b block when CH*_RXCTRL0[3] is driven High.
CH[0/1/2/3]_RXCTRL1[15:0]	Output	RXUSRCLK	When using RX 8B/10B decoder, active High indicates the corresponding byte shown on CH*_RXDATA has a disparity error. CH*_RXCTRL1[3] corresponds to CH*_RXDATA[31:24] CH*_RXCTRL1[2] corresponds to CH*_RXDATA[23:16] CH*_RXCTRL1[1] corresponds to CH*_RXDATA[15:8] CH*_RXCTRL1[0] corresponds to CH*_RXDATA[7:0]
CH[0/1/2/3]_RXCTRL2[7:0]	Output	RXUSRCLK	When using RX 8B/10B decoder, active High indicates the corresponding byte shown on CH*_RXDATA is a comma character. CH*_RXCTRL2[3] corresponds to CH*_RXDATA[31:24] CH*_RXCTRL2[2] corresponds to CH*_RXDATA[23:16] CH*_RXCTRL2[1] corresponds to CH*_RXDATA[15:8] CH*_RXCTRL2[0] corresponds to CH*_RXDATA[7:0]

Table 101: **8B10BDE Ports** *(cont'd)*

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_RXCTRL3[7:0]	Output	RXUSRCLK	When using RX 8B/10B decoder, active High indicates the corresponding byte shown on CH*_RXDATA was not a valid character in the 8B/10B table. CH*_RXCTRL3[3] corresponds to CH*_RXDATA[31:24] CH*_RXCTRL3[2] corresponds to CH*_RXDATA[23:16] CH*_RXCTRL3[1] corresponds to CH*_RXDATA[15:8] CH*_RXCTRL3[0] corresponds to CH*_RXDATA[7:0]

The following table defines the attributes required by RX 8B/10B decoder.

Table 102: **8B10BDE Attributes**

8B10BDE Attributes		
Attribute	**Address**	
CH0_RX_PCS_CFG0		0x0C65
CH1_RX_PCS_CFG0		0x0D65
CH2_RX_PCS_CFG0		0x0E65
CH3_RX_PCS_CFG0		0x0F65
Label	**Bit Field**	**Description**
RX_DATA_WIDTH	[6:3]	Sets the bit width of the CH*_RXDATA port. When 8B/10B encoding is enabled, RX_DATA_WIDTH must be set to 20, 40 or 80. Valid settings are 16, 20, 32, 40, 64, and 80. 4'0010: 16-bit 4'0011: 20-bit 4'0100: 32-bit 4'0101: 40-bit 4'0110: 64-bit 4'0111: 80-bit 4'1000: 128-bit 4'1001: 160-bit
Attribute	**Address**	
CH0_RX_PCS_CFG2		0x0C67
CH1_RX_PCS_CFG2		0x0D67
CH2_RX_PCS_CFG2		0x0E67
CH3_RX_PCS_CFG2		0x0F67
Label	**Bit Field**	**Description**
RX_DEC_MCOMMA_DET	[16:16]	0: CH*_RXCTRL2 is Low when a negative comma is detected. 1: drives the per byte flag CH*_RXCTRL2 High when a MCOMMA is detected.

Table 102: **8B10BDE Attributes** *(cont'd)*

8B10BDE Attributes		
RX_DEC_PCOMMA_DET	[15:15]	0: CH*_RXCTRL2 is Low when a positive comma is detected. 1: drives the per byte flag CH*_RXCTRL2 High when a PCOMMA is detected.
RX_8B10B_EN	[14:14]	RX_8B10B_EN: Enables 10B/8B decoder. 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 can be a multiple of 8 or 10 (16, 20, 32, 40, 64, 80, 128, 160). When clock correction is enabled and RX_DATA_WIDTH is 64 or 80, RX_INT_DATA_WIDTH cannot be set to 2.

RX 128/130B Decoder

If RX received data is 128B/130B encoded, it must be decoded. The GTY transceiver has a built-in 128B/130B encoder in the TX and a 128B/130B decoder in the RX without consuming device resources.

Ports and Attributes

The following table defines the ports required by the RX 128B/130B decoder.

Table 103: **128B130BDE Ports**

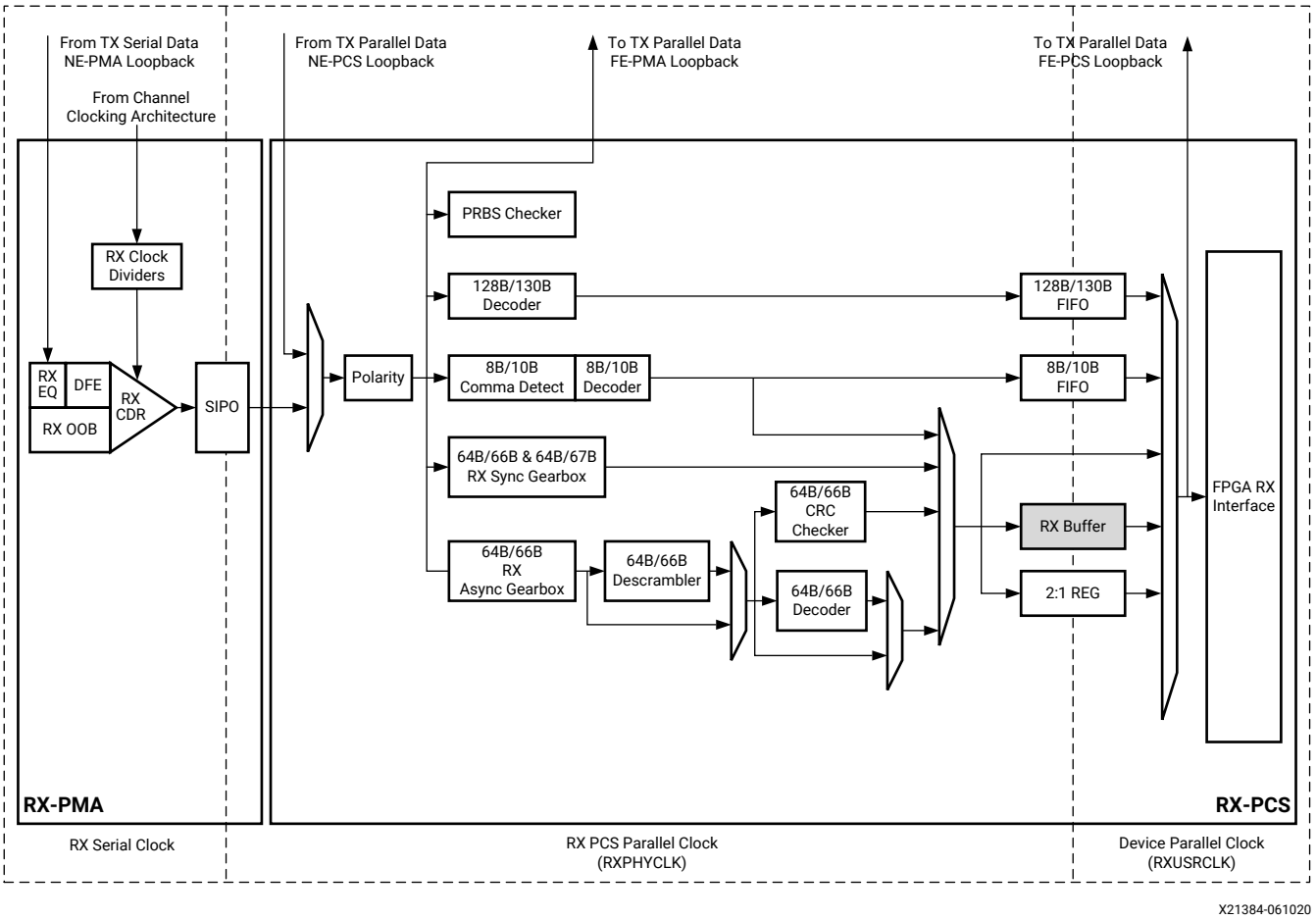
Port	Direction	Clock Domain	Description
CH[0/1/2/3]_RXCTRL0[15:0]	Output	RXUSRCLK	When using RX 8B/10B decoder, active High indicates the corresponding byte shown on CH*_RXDATA is a K character when 8B/10B decoding is enabled. CH*_RXCTRL0[3] corresponds to CH*_RXDATA[31:24] CH*_RXCTRL0[2] corresponds to CH*_RXDATA[23:16] CH*_RXCTRL0[1] corresponds to CH*_RXDATA[15:8] CH*_RXCTRL0[0] corresponds to CH*_RXDATA[7:0] When using 128B/130B decoder: CH*_RXCTRL0[2]: when driven High, ignore the RX data interface for one clock cycle. CH*_RXCTRL0[3]: when driven High, indicates the start byte of a 128b block. CH*_RXCTRL0[6:5]: this signal provides the sync header to use in the current 130b block when CH*_RXCTRL0[3] is driven High.

Note: There are no RX 128B/130B decoder attributes.

RX Buffer

The transceiver RX datapath has two internal parallel clock domains used in the PCS: The PMA parallel clock domain (PHYCLK) 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. The figure below shows the two parallel clock domains: PHYCLK and RXUSRCLK.

Figure 90: **RX Clock Domains**



X21384-061020

The GTY transceiver includes an RX buffer to resolve differences between the PHYCLK and RXUSRCLK domains. The RX buffer's location is highlighted in the previous figure. 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 PHYCLK when the RX buffer is bypassed (see RX Buffer Bypass). The costs and benefits of each approach are shown in the following table.

Table 104: **RX Buffering versus Phase Alignment**

	RX 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 CH*_RXOUTCLK to drive RXUSRCLK.
Clocking Options	Can use RX recovered clock or local clock (with clock correction).	Must use the RX recovered clock.
Initialization	Must wait for all clocks to stabilize before buffer is reset.	Must wait for all clocks to stabilize before performing the RX phase and delay alignment procedure.
Latency	Buffer latency depends on feature use, such as clock correction and channel bonding.	Phase alignment uses fewer registers in the RX datapath to achieve lower and deterministic latency.
RXUSRCLK Jitter Sensitivity	No sensitivity to RXUSRCLK jitter.	Sensitive to RXUSRCLK jitter.

Using the RX Phase Buffer

For 8B/10B and 128B/130B datapaths, there are dedicated elastic FIFOs to resolve phase differences between the PHYCLK and RXUSRCLK domains. For other modes not doing RX clock correction or RX channel bonding and not using the RX buffer bypass, the dedicated RX buffer is used to resolve the phase differences between the PHYCLK and RXUSRCLK domains, and it is enabled by the following:

- RX_PHASE_BUFFER_USE = `1'b1`

The content of the RX buffer becomes invalid if an RX buffer overflow or underflow condition occurs. When any of these conditions occur, reset and reinitialize the RX elastic buffer by using the GTRXRESET procedure or RX component reset procedure. The internally generated RX buffer reset can occur on channel bonding topology change, comma realignment, electrical idle, or rate change conditions.

The dedicated 8B/10B elastic FIFO is used for clock correction (see RX Clock Correction) and channel bonding (see RX Channel Bonding). The RX elastic buffer should be disabled in this use mode. For additional details, refer to the above two sections.

Ports and Attributes

The following table defines the RX buffer ports.

Table 105: **RX Buffer Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_RXBUFSTATUS[2:0]	Output	RXUSRCLK	RXBUFSTATUS provides status for the RX buffer, the port status is as follows: 3'b000: In nominal operating range where the buffer occupancy is within the CLK_COR_MIN_LAT and CLK_COR_MAX_LAT range. 3'b001: RX buffer occupancy is less than CLK_COR_MIN_LAT 3'b010: RX buffer occupancy is greater than CLK_COR_MAX_LAT 3'b101: RX buffer underflow 3'b110: RX buffer overflow.

The following table defines the RX buffer attributes.

Table 106: **RX Buffer Attributes**

RX Buffer Attributes		
Attribute	**Address**	
CH0_RX_PCS_CFG2		0x0C67
CH1_RX_PCS_CFG2		0x0D67
CH2_RX_PCS_CFG2		0x0E67
CH3_RX_PCS_CFG2		0x0F67
Label	**Bit Field**	**Description**
RX_PHASE_BUFFER_USE	[30:30]	Enable control for RX phase buffer. 0: disable control for RX phase buffer. 1: enable control for RX phase buffer.

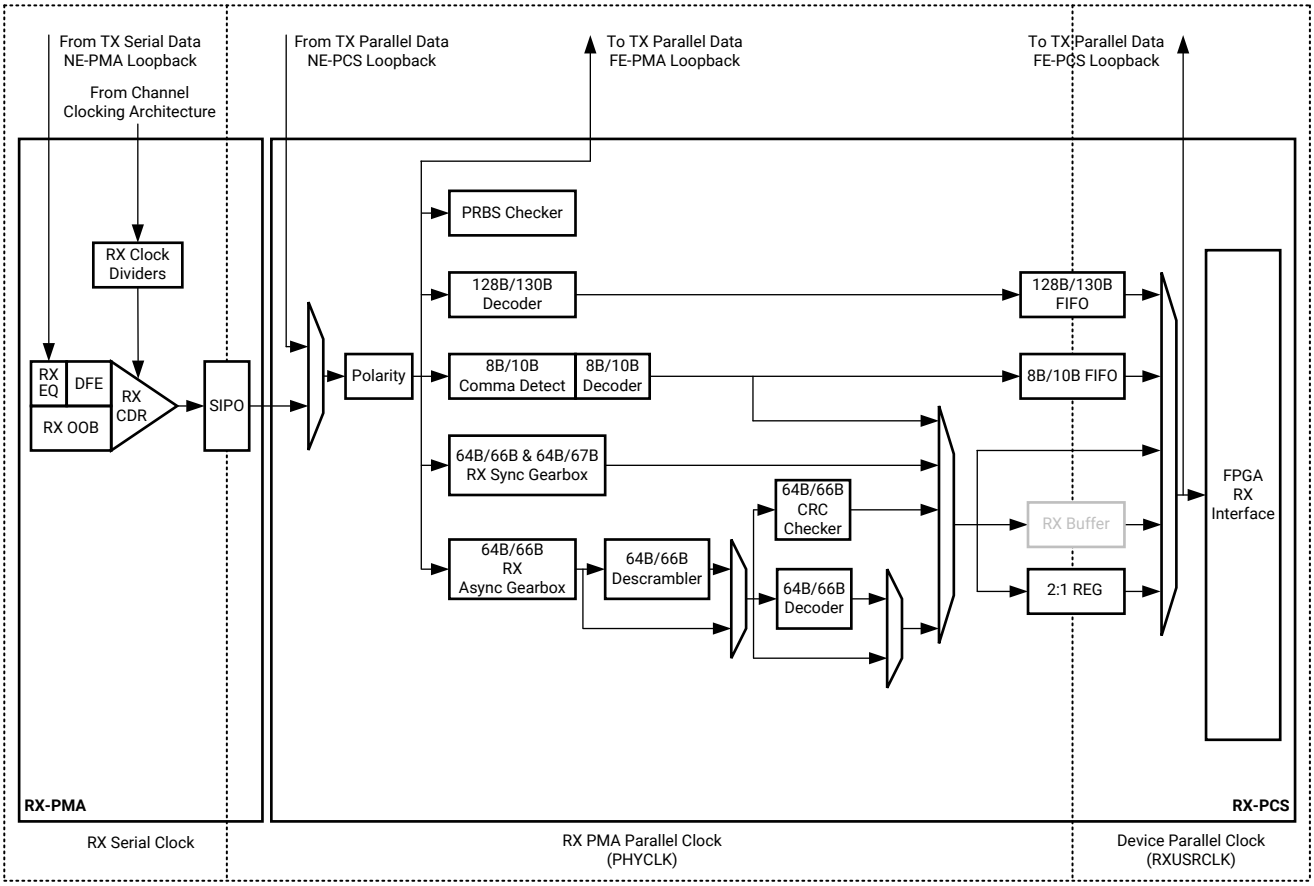
RX Buffer Bypass

The RX phase alignment circuit is used to adjust the phase difference between the PCS parallel clock domain (PHYCLK) and the RXUSRCLK domain when the RX elastic buffer is bypassed. It also performs the RX delay alignment by adjusting RXUSRCLK to compensate for temperature and voltage variations. The combined RX phase and delay alignments can be automatically performed by the GTY transceiver.Table 104 shows trade-offs between buffering and phase alignment.

The RX buffer can be bypassed to reduce latency when the RX recovered clock is used to source RXUSRCLK. When the RX buffer is bypassed, latency through the RX datapath is low and deterministic.

The following figure 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 PCS parallel clock domain (PHYCLK) and the RXUSRCLK domain. RX phase alignment selects a phase shifted version of the RX recovered clock from the CDR so that there is no significant phase difference between PHYCLK and RXUSRCLK.

Figure 91: **RX Buffer Bypass**



X21382-061020

***Note*:** In order to use multi-lane buffer bypass, the Quad placement must be contiguous.

Ports and Attributes

The following table defines the RX buffer bypass ports.

Table 107: **RX Buffer Bypass Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_RXPHDLYPD	Input	ASYNC	RX phase and delay alignment circuit power down. Tied High when a) RX buffer bypass is not in use; b) RXPD is asserted, or c) RXOUTCLKSEL is set to 3'b011 or 3'b100 but the reference clock is not connected. Tied Low during RX buffer bypass mode normal operation. 0: Power-up the RX phase and delay alignment circuit. 1: Power-down the RX phase and delay alignment circuit.
CH[0/1/2/3]_RXPHDLYRESET	Input	ASYNC	This reset is used to initiate the RX Phase alignment process. Assertion of this signal stops all ongoing phase and delay alignment process. De-assertion of this signal will start the auto alignment process between USRCLK and XCLK.
CH[0/1/2/3]_RXPHDLYRESETDONE	Output	ASYNC	This signal indicates that the auto phase alignment process request has been received and phase alignment has started.
CH[0/1/2/3]_RXSYNCALLIN	Input	ASYNC	This input is used for common clock buffer bypass where TXUSRCLK and RXUSRCLK is shared from the same source. Tie to 1'b0 in all other modes.
CH[0/1/2/3]_RXSYNCDONE	Output	RXUSRCLK	Indicates RX buffer bypass procedure is complete. In multi-lane case, signal should be read from the lane designated as the initial master.

The following table defines the RX buffer attributes.

Table 108: **RX Buffer Bypass Attributes**

RX Buffer Bypass Attributes		
Attribute	**Address**	
CH0_RX_PHALIGN_CFG0	0x0C91	
CH1_RX_PHALIGN_CFG0	0x0D91	
CH2_RX_PHALIGN_CFG0	0x0E91	
CH3_RX_PHALIGN_CFG0	0x0F91	
Label	**Bit Field**	**Description**
DLY_ALIGN_EN	[31:31]	Delay alignment enable. Set to 1'b0 when buffer bypass is not used. Set to 1'b1 when buffer bypass is used.
PH_ALIGN_EN	[30:30]	Phase alignment enable. Set to 1'b0 when buffer bypass is not used. Set to 1'b1 when buffer bypass is used.
DAPI_ALIGN_DIR	[29:29]	Multi-lane mode lane type select 0x0: Slave or Single Lane Mode 0x1: Initial Master 0x2: Maintenance Master

Table 108: **RX Buffer Bypass Attributes** *(cont'd)*

RX Buffer Bypass Attributes		
SYNC_MULTI_LANE	[15:15]	Multi-lane enable 1'b0: Single lane mode 1'b1: Multi lane mode
RXBUF_BYPASS_MODE	[14:14]	Buffer bypass enable. Set to 1'b0 when buffer bypass is not used. Set to 1'b1 when buffer bypass is used.
Attribute	**Address**	
CH0_RX_PHALIGN_CFG1		0x0C92
CH1_RX_PHALIGN_CFG1		0x0D92
CH2_RX_PHALIGN_CFG1		0x0E92
CH3_RX_PHALIGN_CFG1		0x0F92
Label	**Bit Field**	**Description**
CHAIN_MODE	[3:2]	Multi-lane daisy chain configuration 2'b00: Unused 2'b01: Top lane of multi-lane block 2'b10: Bottom lane of multi-lane block 2'b11: Middle lane(s) of multi-lane block
ASYNC_GBOX_PHALIGN_EN	[1:1]	Asynchronous Gearbox TX Buffer Bypass Enable 1'b0: Asynchronous gearbox not enabled 1'b1: Asynchronous gearbox enabled
Attribute	**Address**	
CH0_RX_PHALIGN_CFG5		0x0C96
CH1_RX_PHALIGN_CFG5		0x0D96
CH2_RX_PHALIGN_CFG5		0x0E96
CH3_RX_PHALIGN_CFG5		0x0F96
Label	**Bit Field**	**Description**
CMN_FAB_CLK_PHALIGN_MODE	[27:26]	Buffer Bypass Common Clock Mode 2'b00: Common clock disabled, or asynchronous gearbox enabled 2'b01: TX initial master common clock mode 2'b10: RX initial master common clock mode 2'b11: Reserved

RX Buffer Bypass Use Modes

RX phase alignment can be performed on one channel (single lane) or a group of channels sharing a single RXOUTCLK (multi-lane). See the following table for use modes.

Table 109: **RX Buffer Bypass Use Modes**

RX Buffer Bypass	RX Buffer Bypass Modes
Single-Lane	Auto
Multi-Lane	Auto

When the RX asynchronous gearbox is used, the exact use mode depends on the RX fabric interface data width and RX internal data width configuration. See the following table for buffer bypass with asynchronous gearbox applied to both single-lane and multi-lane use modes.

Table 110: **RX Buffer Bypass Use Modes with Asynchronous Gearbox**

RX Buffer Bypass with Asynchronous Gearbox Use Modes	RX Data Width Configuration	Note
1:1 Mode	Fabric Data Width = Internal Data Width	Phase alignment procedure not required.
2:1 Mode	Fabric Data Width = 2 x Internal Data Width	Phase alignment procedure required.

RX Buffer Bypass in Single-Lane Auto Mode without Asynchronous Gearbox

Use these transceiver settings to bypass the RX buffer in single-lane mode:

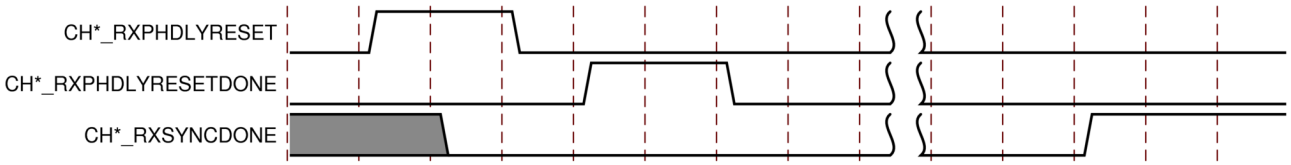
- CH*_RX_PHALIGN_CFG0[31] = `1'b1` (DLY_ALIGN_EN)
- CH*_RX_PHALIGN_CFG0[30] = `1'b1` (PH_ALIGN_EN)
- CH*_RX_PHALIGN_CFG0[17:16] = `2'b00` (SYNC_MODE)
- CH*_RX_PHALIGN_CFG0[15] = `1'b0` (SYNC_MULTI_LANE)
- CH*_RX_PHALIGN_CFG0[14] = `1'b1` (RXBUF_BYPASS_MODE)
- CH*_RX_PHALIGN_CFG1[3:2] = `2'b00` (CHAIN_MODE)
- CH*_RX_PHALIGN_CFG1[1] = `1'b0` (ASYNC_GBOX_PHALIGN_EN)
- CH*_PIPE_CTRL_CFG7[14:12] = `3'b010` or `3'b101` (RXOUTCLKCTL) to select either the recovered clock or the programmable divider clock as the source of RXOUTCLK

With the transceiver reference clock selected, RXOUTCLK is to be used as the source of RXUSRCLK. You must ensure that RXOUTCLK and the selected transceiver reference clocks are operating at the desired frequency. When the RX buffer is bypassed, the RX phase alignment procedure must be performed after these conditions:

- Resetting or powering up the transceiver RX.
- Resetting or powering up the RPLL and/or LCPLL.
- Changing the RX recovered clock source or frequency.
- Changing the TX line rate.

The following figure shows the required steps to perform the auto RX phase alignment and use the RX delay alignment to adjust RXUSRCLK to compensate for temperature and voltage variations.

Figure 92: **RX Buffer Bypass—Single-Lane Auto Mode**



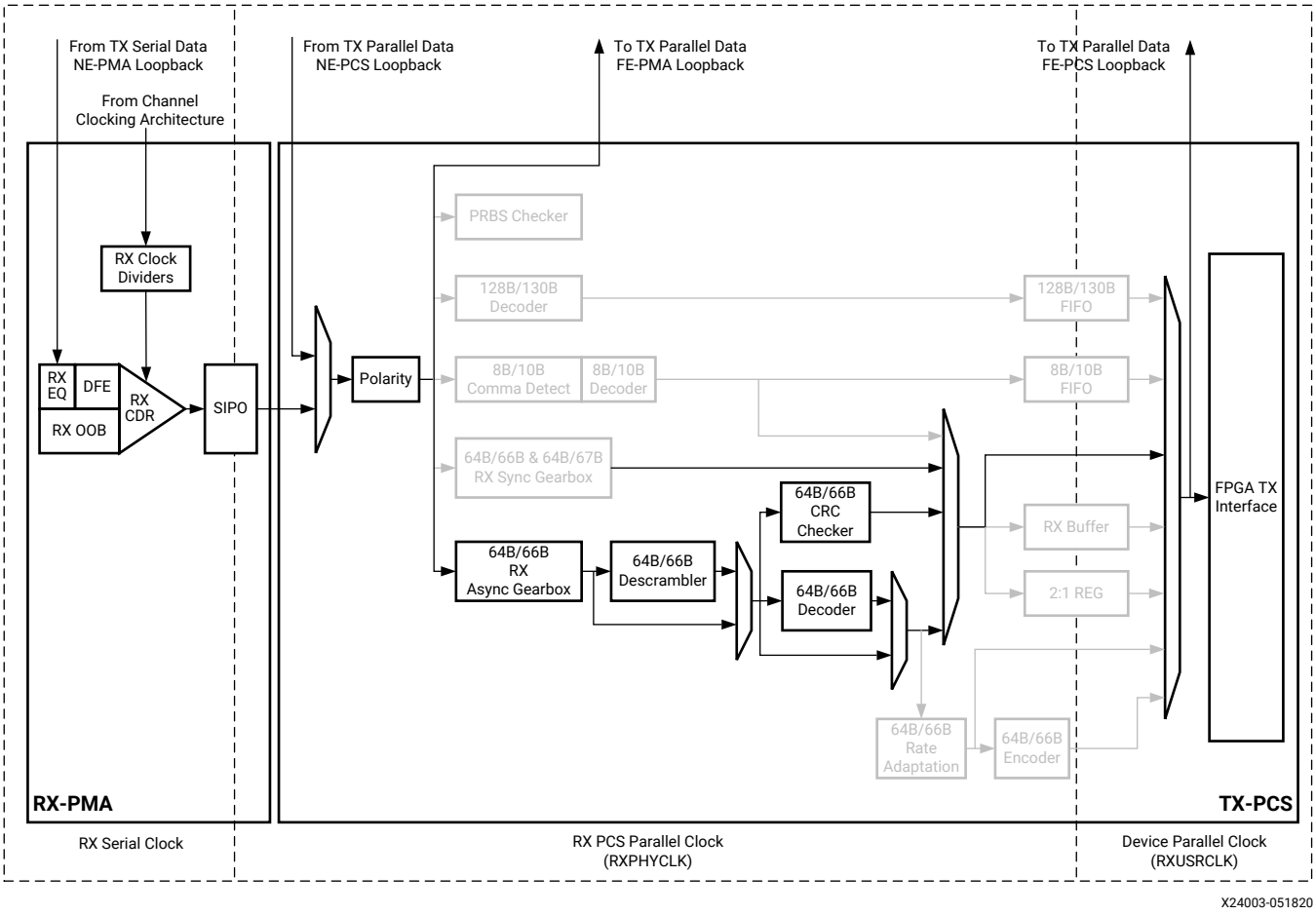
***Note*:**

1. The sequence of events in this figure is not drawn to scale.
2. After conditions such as a receiver reset or RX rate change, RX phase alignment must be performed to align PHYCLK and RXUSRCLK. The RX phase and delay alignments are initiated by asserting CH*_RXPHDLYRESET.
3. RX phase alignment is done when the rising edge of CH*_RXSYNCDONE is detected. This signal should remain asserted until another alignment procedure is initiated.
4. An assertion/deassertion of GTRXRESET is required if CH*_RXSYNCDONE does not follow the sequence shown in this figure.
5. RX delay alignment continues to adjust RXUSRCLK to compensate for temperature and voltage variations.

RX Buffer Bypass in Single-Lane with Asynchronous Gearbox (1:1 Mode)

In single-lane with asynchronous gearbox mode, the RXUSRCLK domain is used to drive logic in the RXPHYCLK domain. In use cases where the RX internal data width is equal to the fabric interface data width, the RX buffer is directly bypassed, and clock phase compensation is provided by the asynchronous gearbox FIFO. In this mode, the latency is deterministic and the asynchronous gearbox FIFO can provide measured latency. For more details on how to read the asynchronous gearbox FIFO latency, refer to RX Asynchronous Gearbox.

Figure 93: **RX Buffer Bypass in Single-Lane with Asynchronous Gearbox 1:1 Mode**



Use these transceiver settings to bypass the RX buffer with asynchronous gearbox enabled in single-lane 1:1 mode:

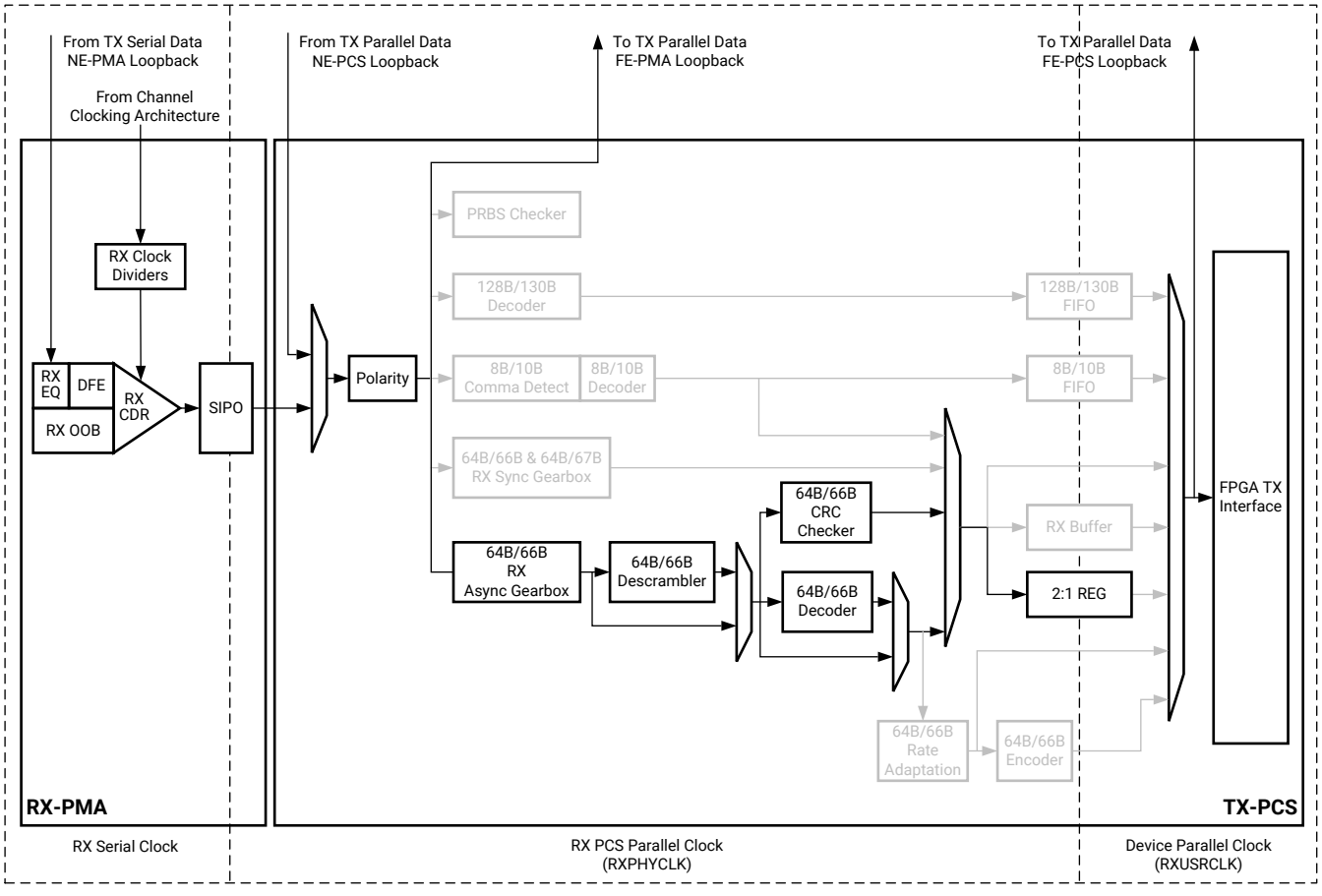
- CH*_RX_PCS_CFG2[5] = `1'b1` (USE_GB)
- CH*_RX_PHALIGN_CFG0[31] = `1'b0` (DLY_ALIGN_EN)
- CH*_RX_PHALIGN_CFG0[30] = `1'b0` (PH_ALIGN_EN)
- CH*_RX_PHALIGN_CFG0[17:16] = `2'b00` (SYNC_MODE)
- CH*_RX_PHALIGN_CFG0[15] = `1'b0` (SYNC_MULTI_LANE)
- CH*_RX_PHALIGN_CFG0[14] = `1'b1` (RXBUF_BYPASS_MODE)
- CH*_RX_PHALIGN_CFG1[3:2] = `2'b00` (CHAIN_MODE)
- CH*_RX_PHALIGN_CFG1[1] = `1'b1` (ASYNC_GBOX_PHALIGN_EN)
- CH*_PIPE_CTRL_CFG7[14:12] = `3'b010` or `3'b101` (RXOUTCLKCTL) to select either the recovered clock or the programmable divider clock as the source of RXOUTCLK

In this particular use mode, because the asynchronous gearbox FIFO provides the phase compensation, there is no need to perform the RX phase alignment procedure.

RX Buffer Bypass in Single-Lane with Asynchronous Gearbox (2:1 Mode)

In single-lane with asynchronous gearbox mode, the RXUSRCLK domain is used to drive logic in the RXPHYCLK domain. In use cases where the RX fabric interface data width is twice the internal data width, the RX buffer is bypassed through the 2:1 REG. The 2:1 REG requires the phase alignment procedure to be performed identical to Figure 92. In this use case, note that the latency is still deterministic because the continuous phase alignment at 2:1 REG does not impact the data latency through the data path.

Figure 94: **RX Buffer Bypass in Single-Lane with Asynchronous Gearbox 2:1 Mode**



X24004-051820

Use these transceiver settings to bypass the RX buffer with asynchronous gearbox enabled in single-lane 2:1 mode:

- `1'b1` (USE_GB)

- CH*_RX_PHALIGN_CFG0[31] = CH*_RX_PCS_CFG2[5] = `1'b1` (DLY_ALIGN_EN)
- CH*_RX_PHALIGN_CFG0[30] = `1'b1` (PH_ALIGN_EN)
- CH*_RX_PHALIGN_CFG0[17:16] = `2'b00` (SYNC_MODE)
- CH*_RX_PHALIGN_CFG0[15] = `1'b0` (SYNC_MULTI_LANE)
- CH*_RX_PHALIGN_CFG0[14] = `1'b1` (RXBUF_BYPASS_MODE)
- CH*_RX_PCS_CFG2[5CH*_RX_PHALIGN_CFG1[3:2] = `2'b00` (CHAIN_MODE)
- CH*_RX_PHALIGN_CFG1[1] = `1'b1` (ASYNC_GBOX_PHALIGN_EN)
- CH*_PIPE_CTRL_CFG7[14:12] = `3'b010` or `3'b101` (RXOUTCLKCTL) to select either the recovered clock or the programmable divider clock as the source of RXOUTCLK

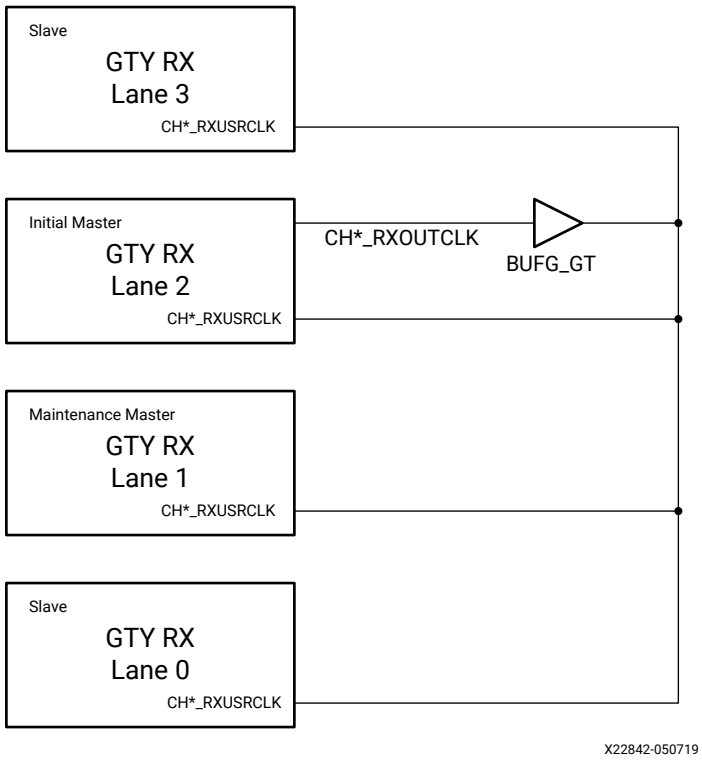
RX Buffer Bypass in Multi-Lane Auto Mode without Asynchronous Gearbox

For GTY transceivers, when a multi-lane application requires RX buffer bypass, phase alignment is performed automatically. This section describes the steps required to perform the multi-lane RX buffer bypass alignment procedure automatically.

- Initial Master: In a multi-lane application, the buffer bypass initial master is the lane that is the source of the RXOUTCLK.
  - CH*_RX_PHALIGN_CFG0[17:16] = `2'b01` (SYNC_MODE)
- Maintenance Master: This lane shares the same RXUSRCLK that is generated from the RXOUCLK of the buffer bypass initial master. The maintenance master also provides delay skew information, which is forwarded internally to the initial master lane.
  - CH*_RX_PHALIGN_CFG0[17:16] = `2'b11` (SYNC_MODE)
- Slave: These are all the lanes that share the same RXUSRCLK, which is generated from the RXOUCLK of the buffer bypass initial master.
  - CH*_RX_PHALIGN_CFG0[17:16] = `2'b00` (SYNC_MODE)

The following figure shows an example of buffer bypass initial master, maintenance master, and slave lanes.

Figure 95: **RX Buffer Bypass Initial Master, Maintenance Master, and Slave Lanes**



Use these transceiver settings to bypass the RX buffer in multi-lane mode:

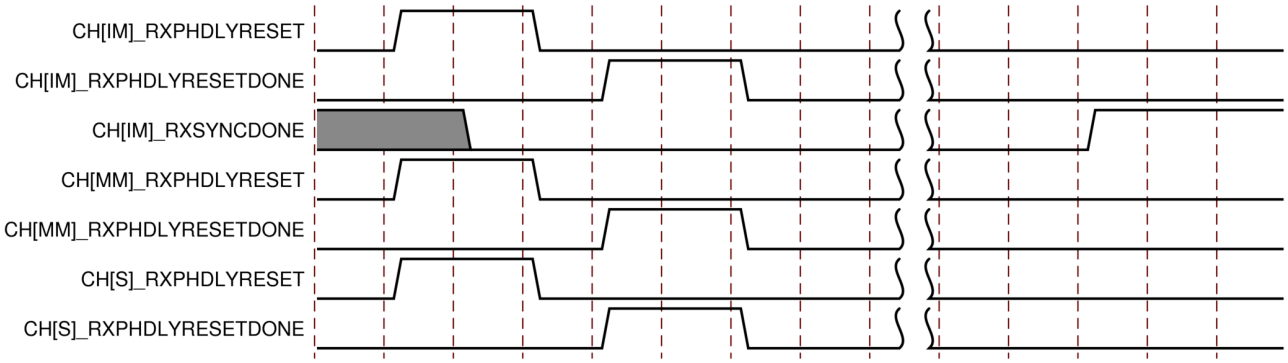
- CH*_RX_PHALIGN_CFG0[31] = `1'b1` (DLY_ALIGN_EN)
- CH*_RX_PHALIGN_CFG0[30] = `1'b1` (PH_ALIGN_EN)
- CH*_RX_PHALIGN_CFG0[15] = `1'b1` (SYNC_MULTI_LANE)
- CH*_RX_PHALIGN_CFG0[14] = `1'b1` (TXBUF_BYPASS_MODE)
- CH*_RX_PHALIGN_CFG1[1] = `1'b0` (ASYNC_GBOX_PHALIGN_EN)
- CH*_PIPE_CTRL_CFG7[14:12] = `3'b010` or `3'b101` (RXOUTCLKCTL) to select either the recovered clock or the programmable divider clock as the source of RXOUTCLK.

Multi-lane buffer bypass must only be used on lanes that have physical locations that are directly adjacent to one another. CH*_RX_PHALIGN_CFG1[3:2](CHAIN_MODE) must be set according to the physical location of the multi-lane group:

- Top location: CH*_RX_PHALIGN_CFG1[3:2] = `2'b01` (CHAIN_MODE)
- Middle location(s): CH*_RX_PHALIGN_CFG1[3:2] = `2'b11` (CHAIN_MODE)
- Bottom location: CH*_RX_PHALIGN_CFG1[3:2] = `2'b10` (CHAIN_MODE)

The following timing diagram shows the required steps to perform auto RX phase and delay alignment.

Figure 96: **RX Buffer Bypass - Multi-Lane Auto Mode**



***Note*:**

1. The sequence of events shown in the figure is not drawn to scale.
2. CH[IM]_* denotes ports related to the initial master lane.
3. CH[MM]_* denotes ports related to the maintenance master lane.
4. CH[S]_* denotes ports related to the slave lane(s).
5. After conditions such as a transmitter reset or RX rate change, RX phase alignment must be performed to align PHYCLK and RXUSRCLK. The RX phase and delay alignments are initiated by asserting CH*_RXPHDLYRESET.
6. RX phase alignment is done when the rising edge of CH[IM]_RXSYNCDONE is detected. This signal should remain asserted until another alignment procedure is initiated.
7. An assertion/deassertion of GTRXRESET is required if CH[IM]_RXSYNCDONE does not follow the sequence shown in Figure 96.
8. RX delay alignment continues to adjust RXUSRCLK to compensate for temperature and voltage variations.

RX Buffer Bypass in Multi-Lane Auto Mode with Asynchronous Gearbox (1:1 Mode)

In multi-lane with asynchronous gearbox mode, the RXUSRCLK domain is used to drive logic in the RXPHYCLK domain. When the RX internal data width is identical to the fabric interface data width, the RX buffer is directly bypassed, and clock phase compensation is provided by the asynchronous gearbox FIFO. In this mode, the latency is deterministic, and the asynchronous gearbox FIFO can provide measured latency. For more details on how to read the asynchronous gearbox FIFO latency, refer to RX Asynchronous Gearbox.

Use these transceiver settings to bypass the RX buffer with asynchronous gearbox in multi-lane 1:1 mode:

- CH*_RX_PCS_CFG2[5] = `1'b1` (USE_GB)

- CH*_RX_PHALIGN_CFG0[31] = `1'b0` (DLY_ALIGN_EN)
- CH*_RX_PHALIGN_CFG0[30] = `1'b0` (PH_ALIGN_EN)
- CH*_RX_PHALIGN_CFG0[17:16] = `2'b00` (SYNC_MODE) (All lanes are configured as slaves in multi-lane asynchronous gearbox mode)
- CH*_RX_PHALIGN_CFG0[15] = `1'b1` (SYNC_MULTI_LANE)
- CH*_RX_PHALIGN_CFG0[14] = `1'b1` (RXBUF_BYPASS_MODE)
- CH*_RX_PHALIGN_CFG1[3:2] = `2'b00` (CHAIN_MODE)
- CH*_RX_PHALIGN_CFG1[1] = `1'b1` (ASYNC_GBOX_PHALIGN_EN)
- CH*_PIPE_CTRL_CFG7[14:12] = `3'b010` or `3'b101` (RXOUTCLKCTL) to select either the recovered clock or the programmable divider clock as the source of RXOUTCLK

In this particular use mode, because the asynchronous gearbox FIFO provides the phase compensation, there is no need to perform the RX phase alignment procedure.

RX Buffer Bypass in Multi-Lane Auto Mode with Asynchronous Gearbox (2:1 Mode)

In multi-lane with asynchronous gearbox mode, the RXUSRCLK domain is used to drive logic in the RXPHYCLK domain. In use cases where the RX fabric interface data width is twice the internal data width, the RX buffer is bypassed through the 2:1 REG. The 2:1 REG requires a phase alignment procedure to be performed identical to Figure 96. In this use case, note that the latency is still deterministic because the continuous phase alignment at 2:1 REG does not impact the data latency through the data path.

Figure 97: **RX Buffer Bypass in Multi-Lane with Asynchronous Gearbox 2:1 Mode**

Use these transceiver settings to bypass the RX buffer with asynchronous gearbox in multi-lane 2:1 mode:

- CH*_RX_PCS_CFG2[5] = 1'b1 (USE_GB)
- CH*_RX_PHALIGN_CFG0[31] = `1'b1` (DLY_ALIGN_EN)
- CH*_RX_PHALIGN_CFG0[30] = `1'b1` (PH_ALIGN_EN)
- CH*_RX_PHALIGN_CFG0[17:16] = `2'b00` (SYNC_MODE) (All lanes are configured as slaves in multi-lane asynchronous gearbox mode)
- CH*_RX_PHALIGN_CFG0[15] = `1'b1` (SYNC_MULTI_LANE)
- CH*_RX_PHALIGN_CFG0[14] = `1'b1` (RXBUF_BYPASS_MODE)
- CH*_RX_PHALIGN_CFG1[3:2] = `2'b00` (CHAIN_MODE)
- CH*_RX_PHALIGN_CFG1[1] = `1'b1` (ASYNC_GBOX_PHALIGN_EN)

- CH*_PIPE_CTRL_CFG7[14:12] = `3'b010` or `3'b101` (RXOUTCLKCTL) to select either the recovered clock or the programmable divider clock as the source of RXOUTCLK

Using TX and RX Buffer Bypass with Common Clock Sharing

In GTY transceivers, using TX and RX buffer bypass with common clock sharing is allowed. For additional details on various common clock sharing usage modes, refer to TX Buffer Bypass.

RX Clock Correction

The dedicated elastic FIFO on the 8B/10B datapath is designed to bridge between two different clock domains, RXUSRCLK and PHYCLK, which is the recovered clock from CDR. Even if RXUSRCLK and PHYCLK are running at same clock frequency, there is always a small frequency difference. Because PHYCLK and RXUSRCLK are not exactly the same, the difference can be accumulated to cause the 8B/10B elastic FIFO to eventually overflow or underflow unless it is corrected. To allow correction, each transceiver TX periodically transmits one or more special characters that the transceiver RX is allowed to remove or replicate in the 8B/10B elastic FIFO, as necessary. By removing characters when the 8B/10B elastic FIFO is too full and replicating characters when the 8B/10B elastic FIFO is too empty, the receiver can prevent overflow or underflow.

Figure 98: **Clock Correction Conceptual View**

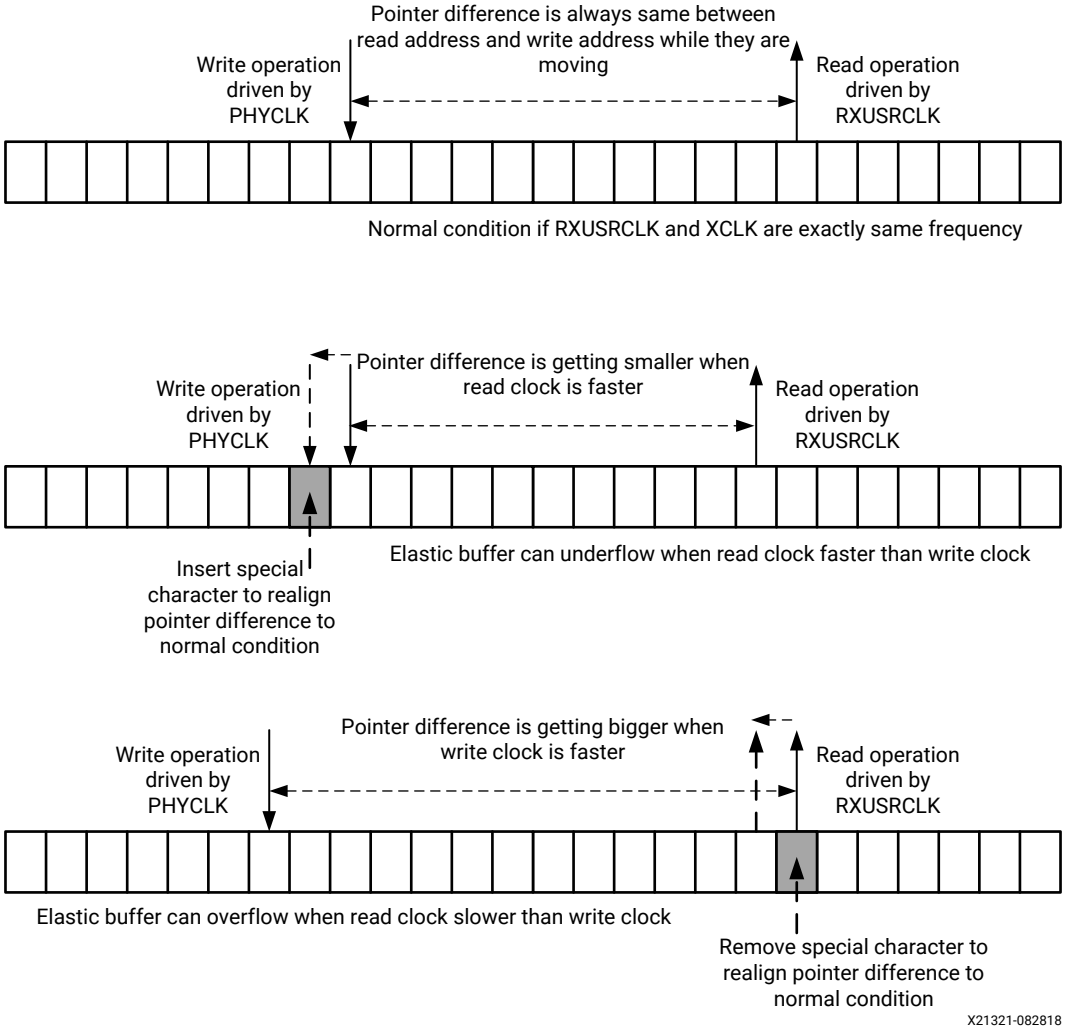


Table 111: **Common Clock Configurations**

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 receiver uses an RX recovered clock.	No
Asynchronous system when separate reference clocks are used and the receiver uses a local clock.	Yes

Ports and Attributes

The following table defines the ports required by RX clock correction functions.

Table 112: **RXCC Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_RXBUFSTATUS[2:0]	Output	RXUSRCLK	RXBUFSTATUS provides status for the RX buffer, the port status is as follows: 3'b000: In nominal operating range where the buffer occupancy is within the CLK_COR_MIN_LAT and CLK_COR_MAX_LAT range. 3'b001: RX buffer occupancy is less than CLK_COR_MIN_LAT 3'b010: RX buffer occupancy is greater than CLK_COR_MAX_LAT 3'b101: RX buffer underflow 3'b110: RX buffer overflow.
CH[0/1/2/3]_RXCLKCORCNT[1:0]	Output	RXUSRCLK	RePort the clock correction status of the RX elastic buffer when the first byte of a clock correction sequence is shown in RXDATA. 2'b00: No clock correction 2'b01: One sequence skipped 2'b10: Two sequences skipped 2'b11: One sequence added

The following table defines the attributes required by RX clock correction functions.

Table 113: **RXCC Attributes**

RXCC Attributes		
Attribute	**Address**	
CH0_RX_ELASTIC_BUF_CFG0		0x0C8C
CH1_RX_ELASTIC_BUF_CFG0		0x0D8C
CH2_RX_ELASTIC_BUF_CFG0		0x0E8C
CH3_RX_ELASTIC_BUF_CFG0		0x0F8C
Label	**Bit Field**	**Description**
EB8B10B_CLK_COR_USE	[1:1]	EB8B10B_CLK_COR_USE: 1'b0: FALSE 1'b1:TRUE
Attribute	**Address**	
CH0_RX_ELASTIC_BUF_CFG1		0x0CAF
CH1_RX_ELASTIC_BUF_CFG1		0x0DAF
CH2_RX_ELASTIC_BUF_CFG1		0x0EAF
CH3_RX_ELASTIC_BUF_CFG1		0x0FAF
Label	**Bit Field**	**Description**
EB8B10B_CLK_COR_SEQ_1_3	[31:22]	EB8B10B_CLK_COR_SEQ_1_3: First clock correction sequence 3 to be compared when EB8B10B_CLK_COR_SEQ_1_EN[2] = 1

Table 113: **RXCC Attributes** *(cont'd)*

RXCC Attributes		
EB8B10B_CLK_COR_SEQ_1_2	[21:12]	EB8B10B_CLK_COR_SEQ_1_2: First clock correction sequence 2 to be compared when EB8B10B_CLK_COR_SEQ_1_EN[1] = 1
EB8B10B_CLK_COR_SEQ_1_1	[11:2]	EB8B10B_CLK_COR_SEQ_1_1: First clock correction sequence 1 to be compared when EB8B10B_CLK_COR_SEQ_1_EN[0] = 1.
EB8B10B_CLK_COR_PRECEDENCE	[1:1]	EB8B10B_CLK_COR_PRECEDENCE: 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. 1'b0 : FALSE 1'b1 : TRUE
EB8B10B_CLK_COR_KEEP_IDLE	[0:0]	EB8B10B_CLK_COR_KEEP_IDLE: 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. 1'b0 : FALSE 1'b1 : TRUE

Attribute	Address
CH0_RX_ELASTIC_BUF_CFG2	0x0CB0
CH1_RX_ELASTIC_BUF_CFG2	0x0DB0
CH2_RX_ELASTIC_BUF_CFG2	0x0EB0
CH3_RX_ELASTIC_BUF_CFG2	0x0FB0

Label	Bit Field	Description
EB8B10B_CLK_COR_SEQ_LEN	[31:30]	EB8B10B_CLK_COR_SEQ_LEN: 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.
EB8B10B_CLK_COR_SEQ_2_2	[29:20]	EB8B10B_CLK_COR_SEQ_2_2: Second clock correction sequence 2 to be compared when EB8B10B_CLK_COR_SEQ_2_EN[1] = 1.
EB8B10B_CLK_COR_SEQ_2_1	[19:10]	EB8B10B_CLK_COR_SEQ_2_1: Second clock correction sequence 1 to be compared when EB8B10B_CLK_COR_SEQ_2_EN[0] = 1.
EB8B10B_CLK_COR_SEQ_1_4	[9:0]	EB8B10B_CLK_COR_SEQ_1_4: First clock correction sequence 4 to be compared when EB8B10B_CLK_COR_SEQ_1_EN[3] = 1.

Attribute	Address
CH0_RX_ELASTIC_BUF_CFG3	0x0CB1
CH1_RX_ELASTIC_BUF_CFG3	0x0DB1

Table 113: **RXCC Attributes** *(cont'd)*

RXCC Attributes		
CH2_RX_ELASTIC_BUF_CFG3		0x0EB1
CH3_RX_ELASTIC_BUF_CFG3		0x0FB1
Label	**Bit Field**	**Description**
EB8B10B_DISPERR_SEQ_MATCH	[31:31]	EB8B10B_DISPERR_SEQ_MATCH: 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. 1'b0 : FALSE 1'b1 : TRUE
EB8B10B_CLK_COR_SEQ_2_EN	[28:25]	EB8B10B_CLK_COR_SEQ_2_EN: Mask enable bit for the second clock correction sequence. EB8B10B_CLK_COR_SEQ_2_EN[0] is the mask bit for EB8B10B_CLK_COR_SEQ_2_1. EB8B10B_CLK_COR_SEQ_2_EN[1] is the mask bit for EB8B10B_CLK_COR_SEQ_2_2. EB8B10B_CLK_COR_SEQ_2_EN[2] is the mask bit for EB8B10B_CLK_COR_SEQ_2_3. EB8B10B_CLK_COR_SEQ_2_EN[3] is the mask bit for EB8B10B_CLK_COR_SEQ_2_4. When EB8B10B_CLK_COR_SEQ_2_EN[*] is 0, the corresponding EB8B10B_CLK_COR_SEQ_2_* is either considered as a don't care or is matched automatically without a comparison. When EB8B10B_CLK_COR_SEQ_2_EN[*] is 1, the corresponding EB8B10B_CLK_COR_SEQ_2_* is compared for a match.
EB8B10B_CLK_COR_SEQ_1_EN	[24:21]	EB8B10B_CLK_COR_SEQ_1_EN: Mask enable bit for the first clock correction sequence. EB8B10B_CLK_COR_SEQ_1_EN[0] is the mask bit for EB8B10B_CLK_COR_SEQ_1_1. EB8B10B_CLK_COR_SEQ_1_EN[1] is the mask bit for EB8B10B_CLK_COR_SEQ_1_2. EB8B10B_CLK_COR_SEQ_1_EN[2] is the mask bit for EB8B10B_CLK_COR_SEQ_1_3. EB8B10B_CLK_COR_SEQ_1_EN[3] is the mask bit for EB8B10B_CLK_COR_SEQ_1_4. When EB8B10B_CLK_COR_SEQ_1_EN[*] is 0, the corresponding EB8B10B_CLK_COR_SEQ_1_* is either considered as a don't care or is matched automatically without a comparison. When EB8B10B_CLK_COR_SEQ_1_EN[*] is 1, the corresponding EB8B10B_CLK_COR_SEQ_1_* is compared for a match.
EB8B10B_CLK_COR_SEQ_2_USE	[20:20]	EB8B10B_CLK_COR_SEQ_2_USE: Set to TRUE if the second clock correction sequence (EB8B10B_CLK_COR_SEQ_2_*) is used in addition to the EB8B10B_CLK_COR_SEQ_1_* that is always used. 1'b0 : FALSE 1'b1 : TRUE

Table 113: **RXCC Attributes** *(cont'd)*

RXCC Attributes		
EB8B10B_CLK_COR_SEQ_2_4	[19:10]	EB8B10B_CLK_COR_SEQ_2_4: Second clock correction sequence 4 to be compared when EB8B10B_CLK_COR_SEQ_2_EN[3] = 1
EB8B10B_CLK_COR_SEQ_2_3	[9:0]	EB8B10B_CLK_COR_SEQ_2_3: Second clock correction sequence 3 to be compared when EB8B10B_CLK_COR_SEQ_2_EN[2] = 1
Attribute	**Address**	
CH0_RX_PCS_CFG0		0x0C65
CH1_RX_PCS_CFG0		0x0D65
CH2_RX_PCS_CFG0		0x0E65
CH3_RX_PCS_CFG0		0x0F65
Label	**Bit Field**	**Description**
RX_INT_DATA_WIDTH	[8:7]	Controls the width of the internal datapath. 2'b00: 2-byte internal datapath 2'b01: 4-byte internal datapath 2'b10: 8-byte internal datapath
RX_DATA_WIDTH	[6:3]	Sets the bit width of the CH*_RXDATA port. When 8B/10B encoding is enabled, RX_DATA_WIDTH must be set to 20, 40 or 80. Valid settings are 16, 20, 32, 40, 64, and 80. 4'0010: 16-bit 4'0011: 20-bit 4'0100: 32-bit 4'0101: 40-bit 4'0110: 64-bit 4'0111: 80-bit 4'1000: 128-bit 4'1001: 160-bit
Attribute	**Address**	
CH0_RX_PCS_CFG1		0x0C66
CH1_RX_PCS_CFG1		0x0D66
CH2_RX_PCS_CFG1		0x0E66
CH3_RX_PCS_CFG1		0x0F66

Table 113: **RXCC Attributes** *(cont'd)*

RXCC Attributes		
Label	**Bit Field**	**Description**
RX_ALIGN_COMMA_WORD	[13:11]	This attribute controls and alignment of detected commas within a multi-byte datapath. 001: Align comma to either of the 2 bytes for a 2-byte interface, any of the 4 bytes for a 4-byte interface, any of the 8 bytes for an 8-byte interface. 010: 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, RXDATA[9:0]/ RXDATA[29:20]/RXDATA[49:40]/RXDATA[69:60] for an 8-byte interface. 100: Align comma to a 4-byte boundary. This setting is not allowed for RX_INT_DATA_WIDTH = 0. The aligned comma is guaranteed to be aligned to RXDATA[9:0] for a 4-byte interface, and RXDATA[9:0]/RXDATA[49:40] for an 8-byte interface. Protocols that send commas in even and odd positions must set RX_ALIGN_COMMA_WORD to 3'b001.
Attribute	**Address**	
CH0_RX_PCS_CFG2	0x0C67	
CH1_RX_PCS_CFG2	0x0D67	
CH2_RX_PCS_CFG2	0x0E67	
CH3_RX_PCS_CFG2	0x0F67	
Label	**Bit Field**	**Description**
RX_CBCC_DATA_SEL	[18:18]	RX_CBCC_DATA_SEL: 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.
RX_8B10B_EN	[14:14]	RX_8B10B_EN: Enables 10B/8B decoder. 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 can be a multiple of 8 or 10 (16, 20, 32, 40, 64, 80, 128, 160). When clock correction is enabled and RX_DATA_WIDTH is 64 or 80, RX_INT_DATA_WIDTH cannot be set to 2.

Using RX Clock Correction

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

Enabling Clock Correction

Each GTY transceiver includes a clock correction circuit that performs clock correction by controlling the pointers of the dedicated 8B/10B elastic FIFO. To use clock correction, the dedicated 8B/10B elastic FIFO needs to be enabled and the general-purpose RX buffer disabled, along with the clock correction enable:

- ELASTICBUF_8B10B_EN = `1'b1`
- RX_PHASE_BUFFER_USE = `1'b0`
- EB8B10B_CLK_COR_USE = `1'b1`

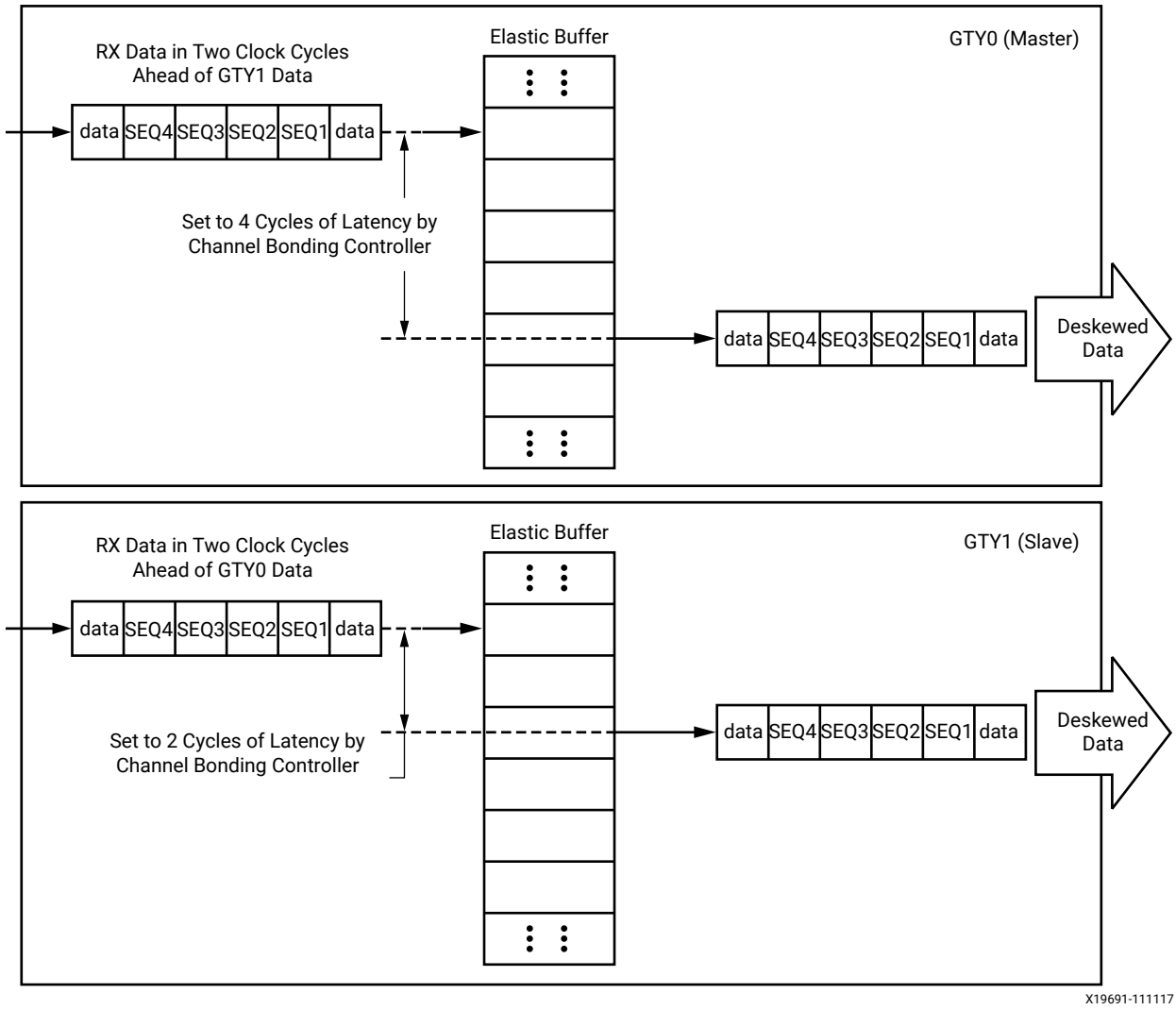
Prior to enabling the clock correction feature, the following settings should also be made (when clock correction is disabled):

- EB8B10B_CLK_COR_SEQ_1_EN = `4'b1111`
- EB8B10B_CLK_COR_SEQ_2_EN = `4'b1111`
- EB8B10B_CLK_COR_SEQ_1_1 = `10'b0100000000`
- EB8B10B_CLK_COR_SEQ_2_1 = `10'b0100000000`

RX Channel Bonding

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. Serial skew, variation in internal clock phases between lanes, and other factors can cause data that is transmitted at the same time to be misaligned across lanes in the received data. Channel bonding compensates for the misalignment by using the RX 8B/10B elastic FIFO as a variable latency block. Channel bonding is also called channel deskew or lane-to-lane deskew. GTY transmitters used for a bonded channel all transmit a channel bonding character (or a sequence of characters) simultaneously. When the sequence is received, the GTY receiver can determine the skew between each lane and adjust the latency of the RX 8B/10B elastic FIFO, so that data is presented without skew at the RX interconnect logic interface.

Figure 99: **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

The following table defines the ports required by RX channel bonding functions.

Table 114: **RXCB Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_RXCHANBONDSEQ	Output	RXUSRCLK	This port goes High when RXDATA contains the start of a channel bonding sequence.
CH[0/1/2/3]_RXCHANISALIGNED	Output	RXUSRCLK	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 or unaligned clock correction sequence is detected, indicating that channel alignment was lost.
CH[0/1/2/3]_RXCHANREALIGN	Output	RXUSRCLK	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.
CH[0/1/2/3]_RXCHBONDI[4:0]	Input	RXUSRCLK	Channel bonding control Port used by slaves only. These Port are used to receive channel bonding and clock correction control information from master GTY transceiver RXCHBONDO Port or from daisy-chained slave GTY transceiver RXCHBONDO Port, which are concatenated from the master GTY transceiver.
CH[0/1/2/3]_RXCHBONDO[4:0]	Output	RXUSRCLK	Channel bonding control Port used to propagate channel bonding and clock correction information to the slave GTY 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 Port. Tie the slave RXCHBONDO to the next level slave RXCHBONDI to form a daisy chain and pass information from the master to each slave.

The following table defines the attributes required by RX channel bonding.

Table 115: **RXCB Attributes**

RXCB Attributes	
Attribute	**Address**
CH0_RX_ELASTIC_BUF_CFG3	0x0CB1
CH1_RX_ELASTIC_BUF_CFG3	0x0DB1
CH2_RX_ELASTIC_BUF_CFG3	0x0EB1
CH3_RX_ELASTIC_BUF_CFG3	0x0FB1

Table 115: **RXCB Attributes** *(cont'd)*

RXCB Attributes		
Label	**Bit Field**	**Description**
EB8B10B_DISPERR_SEQ_MATCH	[31:31]	EB8B10B_DISPERR_SEQ_MATCH: 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. 1'b0 : FALSE 1'b1 : TRUE
Attribute	**Address**	
CH0_RX_ELASTIC_BUF_CFG4		0x0CB2
CH1_RX_ELASTIC_BUF_CFG4		0x0DB2
CH2_RX_ELASTIC_BUF_CFG4		0x0EB2
CH3_RX_ELASTIC_BUF_CFG4		0x0FB2
Label	**Bit Field**	**Description**
EB8B10B_CHAN_BOND_SEQ_1_3	[31:22]	EB8B10B_CHAN_BOND_SEQ_1_3: The CHAN_BOND_SEQ_1 Attribute.
EB8B10B_CHAN_BOND_SEQ_1_2	[21:12]	EB8B10B_CHAN_BOND_SEQ_1_2: The CHAN_BOND_SEQ_1 Attribute.
EB8B10B_CHAN_BOND_SEQ_1_1	[11:2]	EB8B10B_CHAN_BOND_SEQ_1_1: The CHAN_BOND_SEQ_1 Attribute.
EB8B10B_CHAN_BOND_SEQ_LEN	[1:0]	EB8B10B_CHAN_BOND_SEQ_LEN: Defines the length in bytes of the channel bonding sequence that the GT transceiver has to match to find skew. Valid lengths are 1, 2, and 4 bytes.
Attribute	**Address**	
CH0_RX_ELASTIC_BUF_CFG5		0x0CB3
CH1_RX_ELASTIC_BUF_CFG5		0x0DB3
CH2_RX_ELASTIC_BUF_CFG5		0x0EB3
CH3_RX_ELASTIC_BUF_CFG5		0x0FB3
Label	**Bit Field**	**Description**
EB8B10B_CHAN_BOND_SEQ_2_USE	[30:30]	EB8B10B_CHAN_BOND_SEQ_2_USE: 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. 1'b0 : FALSE 1'b1 : TRUE
EB8B10B_CHAN_BOND_SEQ_2_2	[29:20]	EB8B10B_CHAN_BOND_SEQ_2_2: The CHAN_BOND_SEQ_2 Attribute.
EB8B10B_CHAN_BOND_SEQ_2_1	[19:10]	EB8B10B_CHAN_BOND_SEQ_2_1: The CHAN_BOND_SEQ_2 Attribute.
EB8B10B_CHAN_BOND_SEQ_1_4	[9:0]	EB8B10B_CHAN_BOND_SEQ_1_4: The CHAN_BOND_SEQ_1 Attribute.

Table 115: **RXCB Attributes** *(cont'd)*

RXCB Attributes		
Attribute	**Address**	
CH0_RX_ELASTIC_BUF_CFG6		0x0CB4
CH1_RX_ELASTIC_BUF_CFG6		0x0DB4
CH2_RX_ELASTIC_BUF_CFG6		0x0EB4
CH3_RX_ELASTIC_BUF_CFG6		0x0FB4
Label	**Bit Field**	**Description**
EB8B10B_CHAN_BOND_SEQ_2_EN	[27:24]	EB8B10B_CHAN_BOND_SEQ_2_EN
EB8B10B_CHAN_BOND_SEQ_1_EN	[23:20]	EB8B10B_CHAN_BOND_SEQ_1_EN
EB8B10B_CHAN_BOND_SEQ_2_4	[19:10]	EB8B10B_CHAN_BOND_SEQ_2_4: The CHAN_BOND_SEQ_2 Attribute.
EB8B10B_CHAN_BOND_SEQ_2_3	[9:0]	EB8B10B_CHAN_BOND_SEQ_2_3: The CHAN_BOND_SEQ_2 Attribute.
Attribute	**Address**	
CH0_RX_ELASTIC_BUF_CFG7		0x0CB5
CH1_RX_ELASTIC_BUF_CFG7		0x0DB5
CH2_RX_ELASTIC_BUF_CFG7		0x0EB5
CH3_RX_ELASTIC_BUF_CFG7		0x0FB5
Label	**Bit Field**	**Description**
EB8B10B_CHAN_BOND_SLAVE	[22:22]	EB8B10B_CHAN_BOND_SLAVE: Setting channel bonding slave.
EB8B10B_CHAN_BOND_MASTER	[21:21]	EB8B10B_CHAN_BOND_MASTER: Setting channel bonding master.
EB8B10B_CHAN_BOND_LEVEL	[20:18]	EB8B10B_CHAN_BOND_LEVEL: Setting channel bonding pipelining level.
EB8B10B_FTS_LANE_DESKEW_EN	[7:7]	EB8B10B_FTS_LANE_DESKEW_EN: 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.
EB8B10B_CHAN_BOND_MAX_SKEW	[5:2]	EB8B10B_CHAN_BOND_MAX_SKEW: 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.
EB8B10B_CHAN_BOND_EN	[1:1]	EB8B10B_CHAN_BOND_EN : Channel bonding (Deskew) enable for RX 8B10B elastic buffer: 1'b0: FALSE 1'b1:TRUE
Attribute	**Address**	
CH0_RX_PCS_CFG0		0x0C65
CH1_RX_PCS_CFG0		0x0D65

Table 115: **RXCB Attributes** *(cont'd)*

RXCB Attributes		
CH2_RX_PCS_CFG0		0x0E65
CH3_RX_PCS_CFG0		0x0F65
Label	**Bit Field**	**Description**
RX_INT_DATA_WIDTH	[8:7]	Controls the width of the internal datapath. 2'b00: 2-byte internal datapath 2'b01: 4-byte internal datapath 2'b10: 8-byte internal datapath
RX_DATA_WIDTH	[6:3]	Sets the bit width of the CH*_RXDATA port. When 8B/10B encoding is enabled, RX_DATA_WIDTH must be set to 20, 40 or 80. Valid settings are 16, 20, 32, 40, 64, and 80. 4'0010: 16-bit 4'0011: 20-bit 4'0100: 32-bit 4'0101: 40-bit 4'0110: 64-bit 4'0111: 80-bit 4'1000: 128-bit 4'1001: 160-bit
Attribute	**Address**	
CH0_RX_PCS_CFG2		0x0C67
CH1_RX_PCS_CFG2		0x0D67
CH2_RX_PCS_CFG2		0x0E67
CH3_RX_PCS_CFG2		0x0F67
Label	**Bit Field**	**Description**
RX_CBCC_DATA_SEL	[18:18]	RX_CBCC_DATA_SEL: 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.

Using RX Channel Bonding

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

Enabling Channel Bonding

Each GTY transceiver includes a circuit that performs channel bonding by controlling the pointers of the RX 8B/10B elastic FIFO. Because channel bonding requires the use of the RX 8B/10B elastic FIFO, the following settings must be made:

- ELASTICBUF_8B10B_EN = `1'b1`
- RX_PHASE_BUFFER_USE = `1'b0`
- EB8B10B_CHAN_BOND_EN = `1'b1`

Channel bonding is only supported for internal data widths of 2 bytes (RX_INT_DATAWIDTH = 0) and 4 bytes (RX_INT_DATAWIDTH = 1).

RX Synchronous Gearbox

The RX synchronous gearbox provides support for 64B/66B and 64B/67B header and payload separation. The gearbox uses output pins CH*_RXDATA[127:0] and CH*_RXHEADER[5:0] for the payload and header of the received data in normal mode. Similar to TX Synchronous Gearbox, the RX synchronous gearbox operates with the PMA using a single clock. Because of this, occasionally, the output data is invalid. Output pins CH*_RXHEADERVALID and CH*_RXDATAVALID determine if the appropriate header and data are valid. The RX synchronous gearbox supports 2-byte, 4-byte, 8-byte, and 16-byte interfaces.

The data out of the RX synchronous gearbox is not necessarily aligned. Alignment is done in the interconnect logic. The CH*_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 interconnect logic.

Enabling the RX Synchronous Gearbox

The following table describes the attribute settings to enable the RX synchronous gearbox.

Table 116: **Enabling the RX Synchronous Gearbox**

Attribute	Label	Description
CH*_RX_PCS_CFG2[5]	USE_GB	This bit must be set to `1'b1`.

Table 116: **Enabling the RX Synchronous Gearbox** *(cont'd)*

Attribute	Label	Description
CH*_RX_PCS_CFG2[4:0]	MODE	Bit[4] - Must be set to `1'b0`. Bit[3] - Reserved. Must be set to `1'b0`. Bit[2] - Reserved. Must be set to `1'b0`. Bit[1] - Reserved. Must be set to `1'b0`. Bit[0] - Set to `1'b0` to use 64B/67B gearbox or `1'b1` to use 64B/66B gearbox.

Ports and Attributes

The following table defines the RX synchronous gearbox ports.

Table 117: **RXSYNCGEARBOX Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_RXDATAVALID[1:0]	Output	RXUSRCLK	Status output when Gearbox 64B/66B or 64B/67B is used, which indicates that the data appearing on RXDATA is valid. RXDATAVALID[0]: indicates that the data appearing on RXDATA is valid in normal mode for 8-byte, 4-byte, and 2-byte interfaces. For a 16-byte interface, RXDATAVALID[0] indicates RXDATA[63:0] is valid. RXDATAVALID[1]: Indicates that the data appearing on RXDATA[127:64] is valid normal mode for a 16-byte interface.
CH[0/1/2/3]_RXGEARBOXSLIP	Input	RXUSRCLK	When asserted, this port causes the gearbox contents to slip to the next possible alignment. This port is used to achieve proper data alignment within interconnect Port RXDATA and RXHEADER. Asserting this port for one RXUSRCLK 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 interconnect logic.
CH[0/1/2/3]_RXHEADER[5:0]	Output	RXUSRCLK	RXHEADER[1:0]: Header output in normal mode. RXHEADER[4:3]: Header output with 16-byte RXDATA interface. RXHEADER[2] and [5] are used in 64B/67B gearbox

Table 117: **RXSYNCGEARBOX Ports** *(cont'd)*

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_RXHEADERVALID[1:0]	Output	RXUSRCLK	Indicates if RXHEADER is valid. RXHEADERVALID[0]: 1'b1 indicates that RXHEADER is valid for current data. When using an 8-byte RX data interface (RX_DATA_WIDTH = 64) or 16-byte RX data interface (RX_DATA_WIDTH = 128), RXHEADERVALID[0] always outputs 1'b1 indicating RXHEADER is valid for every RXUSRCLK cycle. RXHEADERVALID[0] toggles every RXUSRCLK cycle when using a 4-byte RX data interface. RXHEADERVALID[1]: When using a 16-byte RX data interface RXHEADERVALID[1] always outputs 1'b1 indicating a second header. RXHEADERVALID[1] toggles every RXUSRCLK cycle when using a 4-byte RX data interface.
CH[0/1/2/3]_RXSTARTOFSEQ[1:0]	Output	RXUSRCLK	When the gearbox 64B/66B or 64B/67B is enabled, this output indicates when the sequence counter is 0 for the present RXDATA outputs. RXSTARTOFSEQ[0]: This output indicates when the sequence counter is 0 for the present RXDATA. RXSTARTOFSEQ[0]: Reserved.

The following table defines the RX synchronous gearbox attributes.

Table 118: **RXSYNCGEARBOX Attributes**

RXSYNCGEARBOX Attributes		
Attribute	**Address**	
CH0_RX_PCS_CFG2		0x0C67
CH1_RX_PCS_CFG2		0x0D67
CH2_RX_PCS_CFG2		0x0E67
CH3_RX_PCS_CFG2		0x0F67
Label	**Bit Field**	**Description**
USE_GB	[5:5]	This attribute must be set to 1'b1 to enable either the RX synchronous or asynchronous gearbox.

Table 118: **RXSYNCGEARBOX Attributes** *(cont'd)*

RXSYNCGEARBOX Attributes		
MODE	[4:0]	This attribute indicates the RX gearbox modes: Bit[4]: 1'b0 = Select Synchronous Gearbox. 1'b1 = Select Asynchronous Gearbox. Bit[3]: Reserved. Set to 1'b0. Bit[2]: Reserved. Set to 1'b0. Bit[1]: Reserved. Set to 1'b0. Bit[0]: 1'b0 = 64B/67B gearbox mode for Interlaken (Only valid for synchronous gearbox) 1'b1 = 64B/66B.

RX Asynchronous Gearbox

The RX asynchronous gearbox only provides support for 64B/66B header and payload separation. The gearbox uses the output pins CH*_RXDATA[127:0] and CH*_RXHEADER[4:0] for the payload and header in normal (non-CAUI) mode. 64B/67B is not supported by the RX asynchronous gearbox.

The RX asynchronous gearbox supports a 4-byte, 8-byte, and 16-byte RX data interface to interconnect logic and requires the use of the 4-byte or 8-byte internal datapath. Scrambling of the data is done in the interconnect logic. The following table shows the valid data width combinations for the asynchronous gearbox.

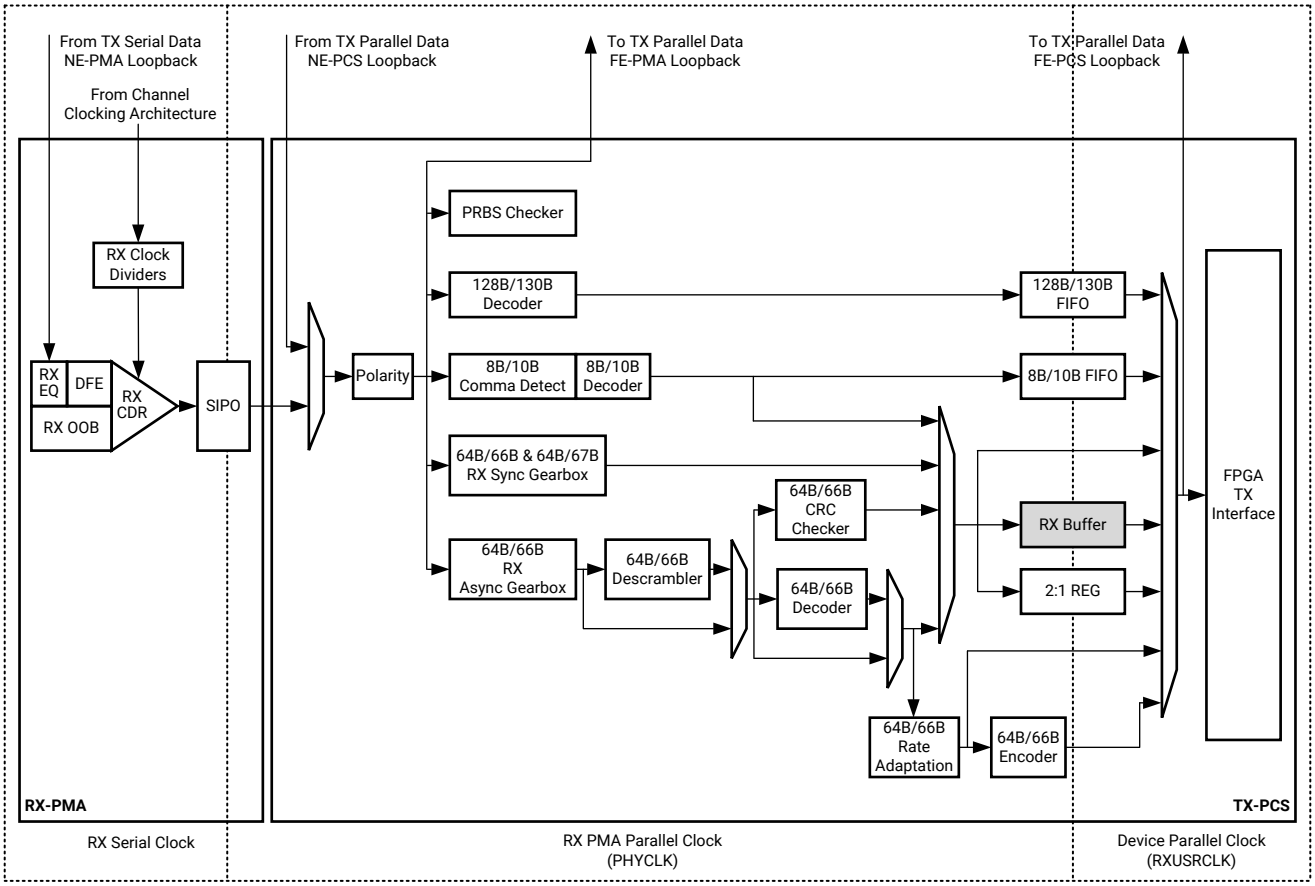
Table 119: **Valid Data Width Combinations for RX Asynchronous Gearbox**

Internal Datapath Width	Interface Width	PHYCLK (MHz)	RXUSRCLK (MHz)
32	32	TX Line Rate/32	TX Line Rate/33
32	64	TX Line Rate/32	TX Line Rate/66
64	64	TX Line Rate/64	TX Line Rate/66
64	128	TX Line Rate/64	TX Line Rate/132

While the RX synchronous gearbox requires you to monitor the CH*_RXDATAVALID port because of invalid data appearing periodically, the RX asynchronous gearbox allows valid data to be continuously received every RXUSRCLK cycle. RX buffer bypass is required when using the RX asynchronous gearbox. The following figure shows the location of the RX asynchronous gearbox. When a 4-byte internal datapath is selected (RX_INT_DATA_WIDTH = 1), 32 bits of

data always enter the RX asynchronous gearbox on every RX PHYCLK cycle. Alternating 34 bits (2-bit header and 32-bit payload) and 32 bits (32-bit payload) of data exit the RX asynchronous gearbox every RXUSRCLK cycle. For an 8-byte internal datapath, 64 bits of data always enter the RX asynchronous gearbox on every RX PHYCLK cycle. 66 bits (2-bit header and 64-bit payload) of data exit the RX asynchronous gearbox every RXUSRCLK cycle.

Figure 100: **RX Clock Domain Example (RX_INT_DATA_WIDTH = 1 (4-byte) and RX_DATA_WIDTH = 64)**



X21383-101119

When in normal mode, the datapath latency through the RX asynchronous gearbox is measured internally, and the reported latency can be accessed by reading a read-only register via the APB3. The RX asynchronous gearbox is used in conjunction with the RX programmable dividers. RXOUTCLKCTL must be set to `3'b101`, and an appropriate divide value must be selected to create the required clock frequency for RXUSRCLK.

Enabling the RX Asynchronous Gearbox

The following table describes the attribute settings to enable the RX asynchronous gearbox.

Table 120: **Enabling the RX Asynchronous Gearbox**

Attribute	Label	Description
CH*_RX_PCS_CFG2[5]	USE_GB	This bit must be set to `1'b1`.
CH*_RX_PCS_CFG2[4:0]	MODE	Bit[4] - Must be set to `1'b1`. Bit[3] - Reserved. Must be set to `1'b0`. Bit[2] - Reserved. Must be set to `1'b0`. Bit[1] - Reserved. Must be set to `1'b0`. Bit[0] - Set to `1'b1` to use 64B/66B gearbox when using the asynchronous gearbox.

Reading Datapath Latency

The datapath latency through the RX async gearbox FIFO is calculated statistically using RXLATCLK, which is asynchronous to RX_PHYCLK. SAMPLE_PERIOD in CH*_RX_PCS_CFG2 determines the number of RXLATCLK cycles over which averaging takes place. The measured latency value in RXGBOX_FIFO_LATENCY is updated once per sampling period, which is defined in SAMPLE_PERIOD.

For the read side of the RX async gearbox FIFO, there is an additional offset that is determined by the gearbox slip count value for the data alignment. Thus, the CH*_RXGEARBOXSLIP must be performed to achieve sync status prior to reading out the latency value. The latency measurement is not supported in CAUI mode.

These settings are used to read the latency:

- Enable RX asynchronous gearbox under normal mode.
- Set CH*_RX_PCS_CFG2[12:10] (SAMPLE_PERIOD):
  - A higher averaging period gives a more accurate latency value.
- Achieve datapath sync status by CH*_RXGEARBOXSLIP.
- Read CH*_RXGBOX_FIFO_LATENCY[29:16] (RXGBOX_FIFO_LATENCY):
  - The value is in units of 1/8 UI.
  - The actual latency is RXGBOX_FIFO_LATENCY plus a fixed value.

Ports and Attributes

The following table defines the RX asynchronous gearbox ports.

Table 121: **RXASYNCGEARBOX Ports**

Port	Direction	Clock Domain	Description
CH[0/1/2/3]_RXGEARBOXSLIP	Input	RXUSRCLK	When asserted, this port causes the gearbox contents to slip to the next possible alignment. This port is used to achieve proper data alignment within interconnect Port RXDATA and RXHEADER. Asserting this port for one RXUSRCLK 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 interconnect logic.
CH[0/1/2/3]_RXHEADER[5:0]	Output	RXUSRCLK	RXHEADER[1:0]: Header output in normal mode. RXHEADER[4:3]: Header output with 16-byte RXDATA interface. RXHEADER[2] and [5] are used in 64B/67B gearbox
CH[0/1/2/3]_RXHEADERVALID[1:0]	Output	RXUSRCLK	Indicates if RXHEADER is valid. RXHEADERVALID[0]: 1'b1 indicates that RXHEADER is valid for current data. When using an 8-byte RX data interface (RX_DATA_WIDTH = 64) or 16-byte RX data interface (RX_DATA_WIDTH = 128), RXHEADERVALID[0] always outputs 1'b1 indicating RXHEADER is valid for every RXUSRCLK cycle. RXHEADERVALID[0] toggles every RXUSRCLK cycle when using a 4-byte RX data interface. RXHEADERVALID[1]: When using a 16-byte RX data interface RXHEADERVALID[1] always outputs 1'b1 indicating a second header. RXHEADERVALID[1] toggles every RXUSRCLK cycle when using a 4-byte RX data interface.
CH[0/1/2/3]_RXLATCLK	Input	CLOCK	Input port used to provide a clock for the RX asynchronous gearbox latency calculation.

The following table defines the RX asynchronous gearbox attributes.

Table 122: **RXASYNCGEARBOX Attributes**

RXASYNCGEARBOX Attributes	
Attribute	**Address**
CH0_RXGBOX_FIFO_LATENCY	0x086d
CH1_RXGBOX_FIFO_LATENCY	0x096d
CH2_RXGBOX_FIFO_LATENCY	0x0a6d
CH3_RXGBOX_FIFO_LATENCY	0x0b6d

Table 122: **RXASYNCGEARBOX Attributes** *(cont'd)*

RXASYNCGEARBOX Attributes		
Label	**Bit Field**	**Description**
RXGBOX_FIFO_LATENCY	[29:16]	Measured latency in UI through the RX asynchronous gearbox averaged over SAMPLE_PERIOD (CH*_RX_PCS_CFG2[12:10]) cycles. The reported latency is in units of 1/8 UI. The RXGBOX_FIFO_LATENCY is read-only.
Attribute	**Address**	
CH0_RXGBOX_FIFO_OVERFLOW	0x086d	
CH1_RXGBOX_FIFO_OVERFLOW	0x096d	
CH2_RXGBOX_FIFO_OVERFLOW	0x0a6d	
CH3_RXGBOX_FIFO_OVERFLOW	0x0b6d	
Label	**Bit Field**	**Description**
RXGBOX_FIFO_OVERFLOW	[31:31]	RX asynchronous gearbox FIFO overflow status. A High indicates that overflow error has occured. The RXGBOX_FIFO_OVERFLOW is read-only.
Attribute	**Address**	
CH0_RXGBOX_FIFO_UNDERFLOW	0x086d	
CH1_RXGBOX_FIFO_UNDERFLOW	0x096d	
CH2_RXGBOX_FIFO_UNDERFLOW	0x0a6d	
CH3_RXGBOX_FIFO_UNDERFLOW	0x0b6d	
Label	**Bit Field**	**Description**
RXGBOX_FIFO_UNDERFLOW	[30:30]	RX asynchronous gearbox FIFO underflow status. A High indicates that underflow error has occured. The RXGBOX_FIFO_UNDERFLOW is read-only.
Attribute	**Address**	
CH0_RX_PCS_CFG2	0x0C67	
CH1_RX_PCS_CFG2	0x0D67	
CH2_RX_PCS_CFG2	0x0E67	
CH3_RX_PCS_CFG2	0x0F67	
Label	**Bit Field**	**Description**
SAMPLE_PERIOD	[12:10]	Number of CH*_RXLATCLK cycles over which averaging take place for latency calculation: 3'b000: 256 3'b001: 512 3'b010: 1024 3'b011: 2048 3'b100: 4096 3'b101: 8192 3'b110: 16384 3'b111: 32768
USE_GB	[5:5]	This attribute must be set to 1'b1 to enable either the RX synchronous or asynchronous gearbox.

Table 122: **RXASYNCGEARBOX Attributes** *(cont'd)*

RXASYNCGEARBOX Attributes		
MODE	[4:0]	This attribute indicates the RX gearbox modes: Bit[4]: 1'b0 = Select Synchronous Gearbox. 1'b1 = Select Asynchronous Gearbox. Bit[3]: Reserved. Set to 1'b0. Bit[2]: Reserved. Set to 1'b0. Bit[1]: Reserved. Set to 1'b0. Bit[0]: 1'b0 = 64B/67B gearbox mode for Interlaken (Only valid for synchronous gearbox) 1'b1 = 64B/66B.

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. The following tables shows the valid Data and K characters.

Table 123: **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
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

Table 123: **Valid Data Characters** *(cont'd)*

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
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
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

Table 123: **Valid Data Characters** *(cont'd)*

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
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
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

Table 123: **Valid Data Characters** *(cont'd)*

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
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
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

Table 123: **Valid Data Characters** *(cont'd)*

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
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
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
D21.5	101 10101	101010 1010	101010 1010
D22.5	101 10110	011010 1010	011010 1010

Table 123: **Valid Data Characters** *(cont'd)*

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
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
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

Table 123: **Valid Data Characters** *(cont'd)*

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
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
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 124: **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

Table 124: **Valid Control K Characters** *(cont'd)*

Special Code Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
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	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

XILINX

Chapter 6

Board Design Guidelines

Topics related to implementing a design on a printed circuit board that uses the GTY transceivers are presented in this chapter. The GTY transceivers are analog circuits that require special consideration and attention when designing and implementing them on a printed circuit board. Besides an understanding of the functionality of the device pins, a design that performs optimally requires attention to issues such as device interfacing, transmission line impedance and routing, power supply design filtering and distribution, component selection, and PCB layout and stackup design.

Pin Description and Design Guidelines

Transceiver Pin Descriptions

The following table defines the GTY transceiver Quad pins.

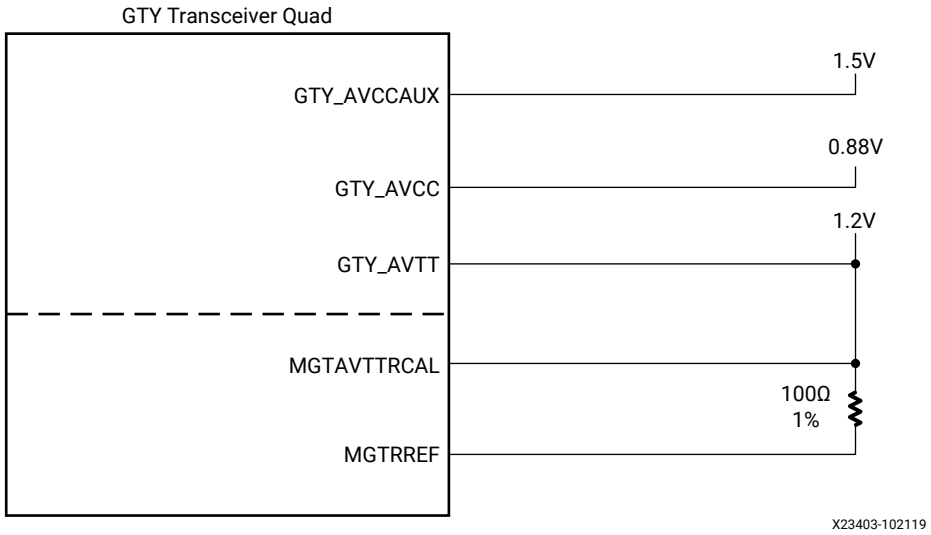
Table 125: **Transceiver Quad Descriptions**

Pins	Dir	Description
MGTREFCLK0P MGTREFCLK0N	In/Out (Pad)	Configured as either reference clock input pins or RX recovered clock output pins for the Quad.
MGTREFCLK1P MGTREFCLK1N	In/Out (Pad)	Configured as either reference clock input pins or RX recovered clock output pins for the Quad.
GTY_RXP[3:0]/GTY_RXN[3:0]	In (Pad)	RXP and RXN are the differential input pairs for each of the receivers in the transceiver Quad.
GTY_TXP[3:0]/GTY_TXN[3:0]	Out (Pad)	TXP and TXN are the differential output pairs for each of the transmitters in the transceiver Quad.
MGTAVTTRCAL	In (Pad)	Bias current supply for the termination resistor calibration circuit. See Termination Resistor Calibration Circuit.
MGTRREF	In (Pad)	Calibration resistor input pin for the termination resistor calibration circuit. See Termination Resistor Calibration Circuit.
GTY_AVCC	In (Pad)	GTY_AVCC is the analog supply for the internal analog circuits of the GTY transceiver Quad tile. This includes the analog circuits for the PLLs, transmitters, and receivers. Most packages have multiple groups of power supply connections in the package for GTY_AVCC. Refer to the package pin definitions to identify in which power supply group a specific GTY transceiver Quad is located. For Versal ACAPs, the nominal voltage is 0.88 V_{DC}.

Table 125: **Transceiver Quad Descriptions** *(cont'd)*

Pins	Dir	Description
GTY_AVTT	In (Pad)	GTY_AVTT is the analog supply for the transmitter and receiver termination circuits of the GTY transceiver Quad tile. Most packages have multiple groups of power supply connections in the package for GTY_AVTT. Refer to the package pin definitions to identify in which power supply group a specific GTY transceiver Quad is located. For Versal ACAPs, the nominal voltage is 1.2 V_{DC}.
GTY_AVCCAUX	In (Pad)	GTY_AVCCAUX is the auxiliary analog LCPLL voltage supply for the transceivers. Most packages have multiple groups of power supply connections in the package for GTY_AVCCAUX. Refer to the package pin definitions to identify in which power supply group a specific GTY transceiver Quad is located. For Versal ACAPs, the nominal voltage is 1.5 V_{DC}.

Figure 101: **Transceivers External Power Supply Connections**



The above figure shows the external power supply connections with the GTY transceivers.

***Note*:** The voltage values are nominal. See the Versal ACAP data sheets for values and tolerances.

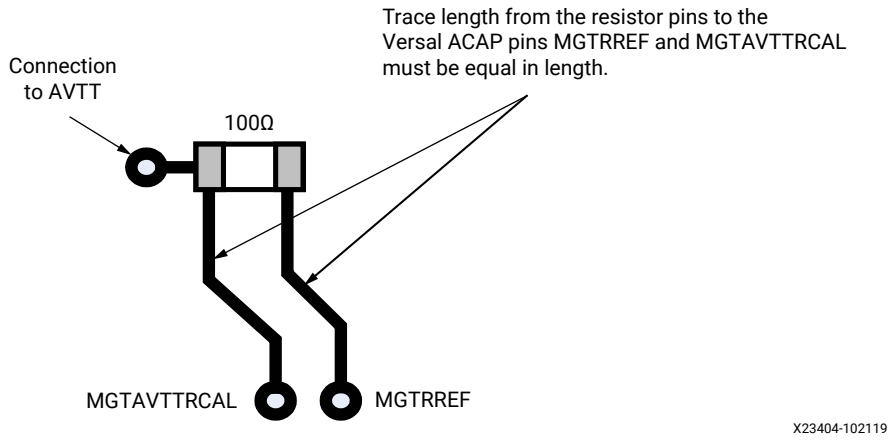
Termination Resistor Calibration Circuit

There is one resistor calibration circuit (RCAL) shared between all transceiver Quad primitives in a transceiver Quad column. The MGTAVTTRCAL and MGTRREF pins connect the bias circuit power and the external calibration resistor to the RCAL circuit. The RCAL circuit performs the resistor calibration only during configuration of the Versal ACAP. Prior to configuration, all analog supply voltages must be present and within the proper tolerance as specified in the Versal ACAP data sheets. If an entire power supply group (PSG) is not used by any Quads, MGTAVTTRCAL and MGTRREF should be tied to ground. See Analog Power Supply Pins for more details regarding RCAL biasing recommendations when there are unused Quads.

The RCAL circuit is associated with the GTY transceiver Quad that is the RCAL master. The RCAL master performs the termination resistor calibration during configuration of the Versal ACAP and then distributes the calibrated values to all of the GTY transceiver Quads in the column. The Quad in which the RCAL circuit is located must be powered on.

Connect the MGTAVTTRCAL pin to the GTY_AVTT supply and to a pin on the 100Ω precision external resistor. The other pin of the resistor is connected to the MGTRREF pin. The resistor calibration circuit provides a controlled current load to the resistor connected to the MGTRREF pin. It then senses the voltage drop across the external calibration resistor and uses that value to adjust the internal resistor calibration setting. The quality of the resistor calibration is dependent on the accuracy of the voltage measurement at the MGTAVTTRCAL and MGTRREF pins. To eliminate errors due to the voltage drop across the traces that lead from the resistor and to the Versal ACAP pins, the trace from the MGTAVTTRCAL pin to the resistor should have the same length and geometry as the trace that connects the other pin of the resistor to the MGTRREF pin. Also, the maximum DC resistance of the PCB trace must be limited to less than 0.5Ω. See the suggested layout in the following figure.

Figure 102: **PCB Layout for the RCAL Resistor**



Reference Clock

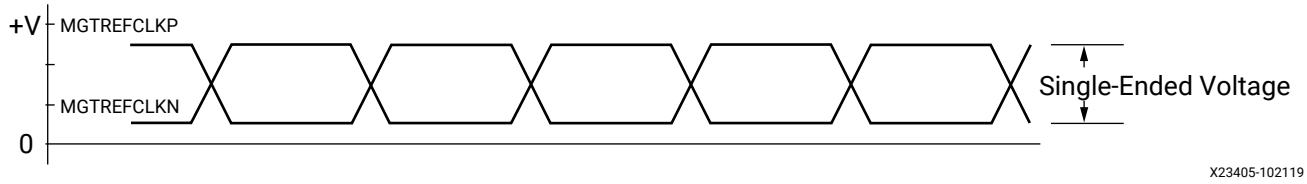
This section focuses on the selection of the reference clock source or oscillator. An oscillator is characterized by:

- Frequency range
- Output voltage swing
- Jitter (deterministic, random, peak-to-peak)
- Rise and fall times
- Supply voltage and current

- Noise specification
- Duty cycle and duty-cycle tolerance
- Frequency stability

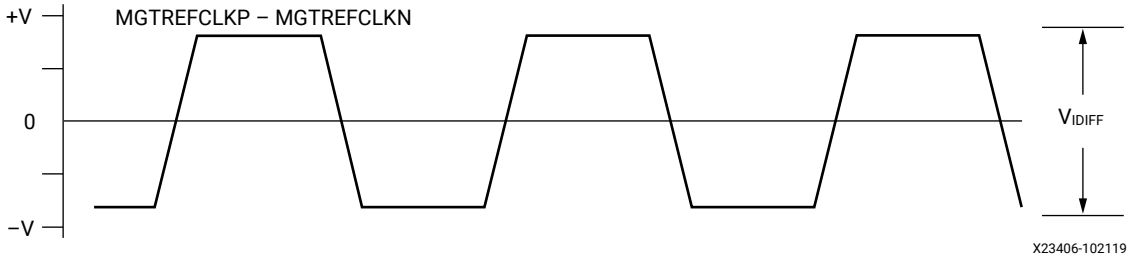
These characteristics are selection criteria when choosing an oscillator for a GTY transceiver design. Figure 103 illustrates the convention for the single-ended clock input voltage swing, peak-to-peak. This figure is provided to show the contrast to the differential clock input voltage swing calculation shown in Figure 104, as used in the GTY transceiver portion of the Versal ACAP data sheets.

Figure 103: **Single-Ended Clock Input Voltage Swing, Peak-to-Peak**



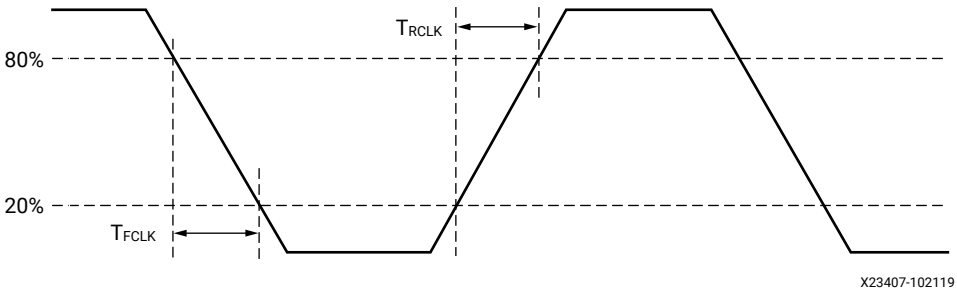
The following figure illustrates the differential clock input voltage swing, which is defined as MGTREFCLKP – MGTREFCLKN.

Figure 104: **Differential Clock Input Voltage Swing, Peak-to-Peak**



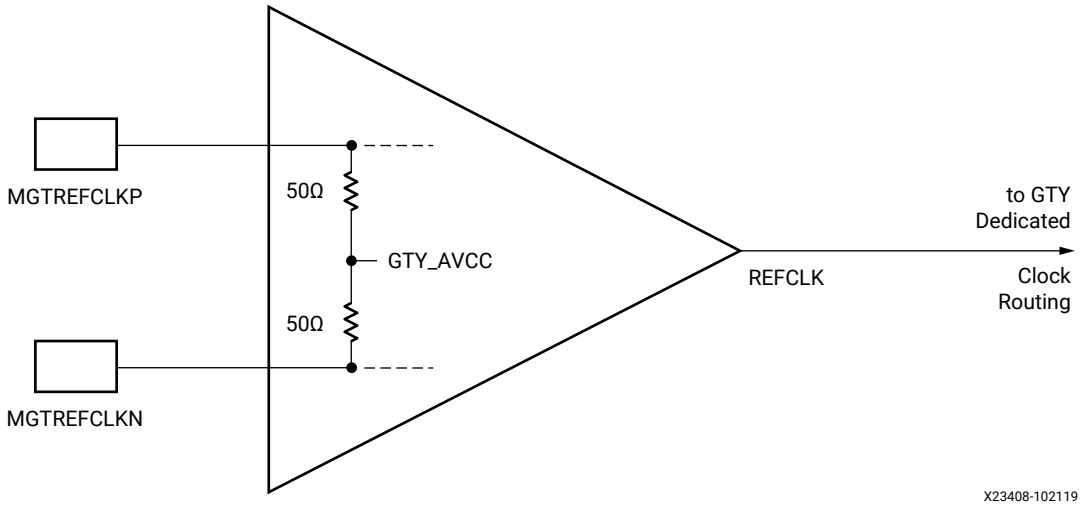
The following figure shows the rise and fall time convention of the reference clock.

Figure 105: **Rise and Fall Times**



The following figure illustrates the internal details of the IBUFDS. The dedicated differential reference clock input pair MGTREFCLKP/MGTREFCLKN is internally terminated with 100Ω differential impedance. The common mode voltage of this differential reference clock input pair is GTY_AVCC, 0.88V. See the Versal ACAP data sheets for exact specifications.

Figure 106: **MGTREFCLK Input Buffer Details**



***Note*:** The resistor values are nominal. See the Versal ACAP data sheets for exact specifications.

GTY Transceiver Reference Clock Checklist

These criteria must be met when choosing an oscillator for a design with GTY transceivers:

- Provide AC coupling between the oscillator output pins and the dedicated GTY transceiver Quad clock input pins.
- Ensure that the differential voltage swing of the reference clock is the range as specified in the Versal ACAP data sheets. The nominal range is 250 mV–2000 mV and the nominal value is 1200 mV).
- Meet or exceed the reference clock characteristics as specified in the Versal ACAP data sheets.
- Meet or exceed the reference clock characteristics as specified in the standard for which the GTY transceiver provides physical layer support.
- Fulfill the oscillator vendor's requirement regarding power supply, board layout, and noise specification.
- Provide a dedicated point-to-point connection between the oscillator and GTY transceiver Quad clock input pins.

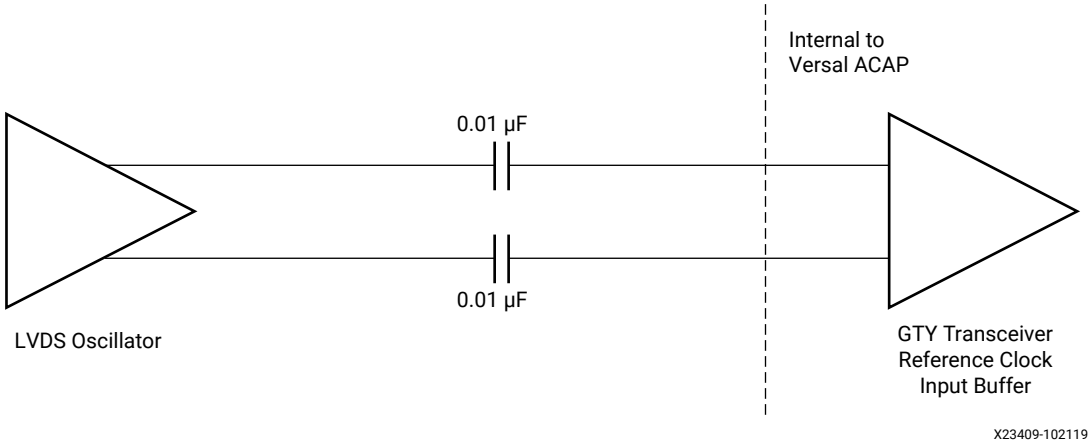
- Keep impedance discontinuities on the differential transmission lines to a minimum (impedance discontinuities generate jitter).

Reference Clock Interface

LVDS

The following figure shows how an LVDS oscillator is connected to a reference clock input of a GTY transceiver.

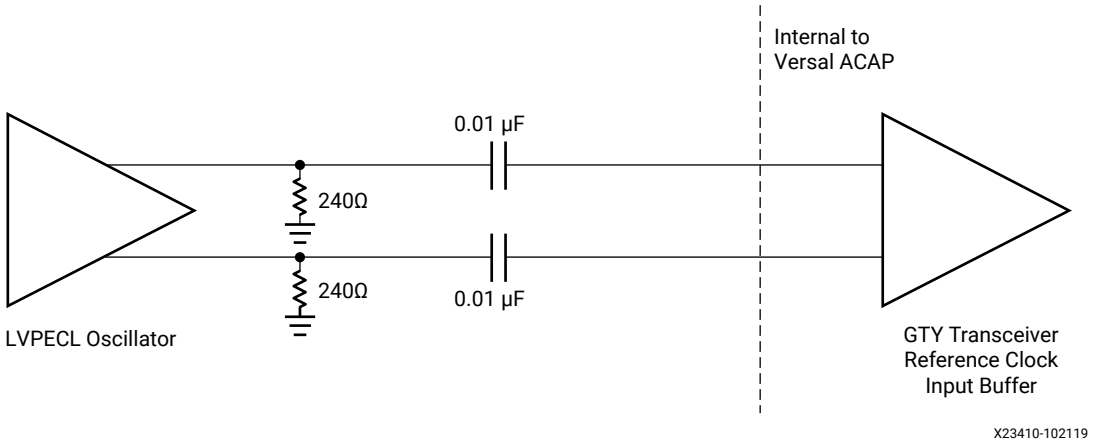
Figure 107: **Interfacing an LVDS Oscillator to the GTY Transceiver Reference Clock Input**



LVPECL

The following figure shows how an LVPECL oscillator is connected to a reference clock input of a GTY transceiver.

Figure 108: **Interfacing an LVPECL Oscillator to the GTY Transceiver Reference Clock Input**



AC Coupled Reference Clock

AC coupling of the oscillator reference clock output to the GTY transceiver Quad reference clock inputs serves multiple purposes:

- Blocking a DC current between the oscillator and the GTY transceiver Quad dedicated clock input pins (which reduces the power consumption of both parts as well)
- Common-mode voltage independence
- The AC coupling capacitor forms a high-pass filter with the on-chip termination that attenuates wander of the reference clock

To minimize noise and power consumption, external AC coupling capacitors between the sourcing oscillator and the GTY transceiver Quad dedicated reference clock input pins are required.

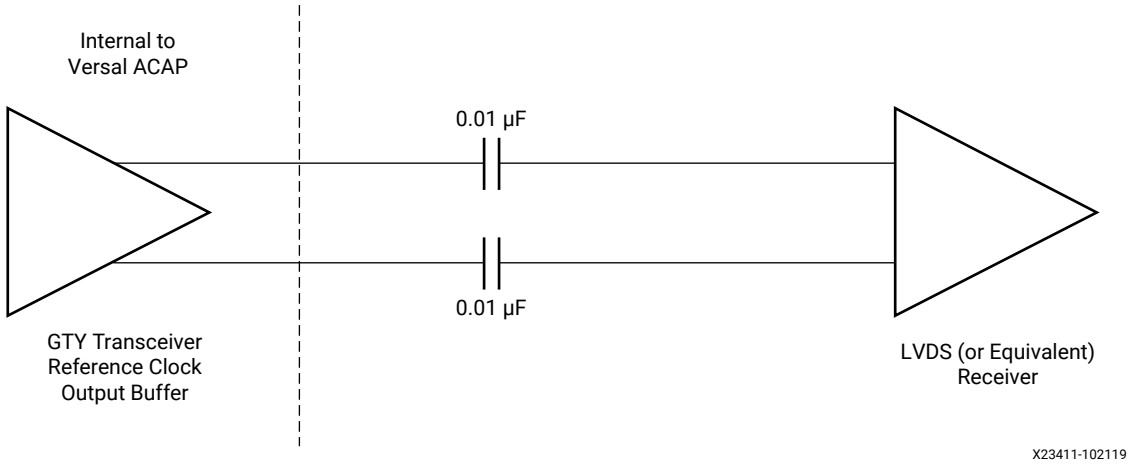
Unused Reference Clocks

If the reference clock input is not used, leave the reference clock input pins unconnected (both MGTREFCLKP and MGTREFCLKN).

Reference Clock Output Buffer

The reference clock pins can be configured to be output pins that drive an RX recovered clock from one of the transceivers in the Quad. Operation and configuration of this buffer is discussed in Reference Clock Input/Output Structure. This output is designed to supply a signal through DC blocking capacitors on the PCB. The signal levels are comparable to those of LVDS after the DC blocking capacitors. See the Versal ACAP data sheets for output levels.

Figure 109: **Versal Architecture GTY Transceiver Reference Clock Output Connection**



Reference Clock Power

The GTY transceiver reference clock input circuit is powered by GTY_AVCC. Excessive noise on this supply has a negative impact on the performance of any GTY transceiver Quad that uses the reference clock from this circuit.

Power Supply

Analog Power Supply Pins

The GTY transceiver Quad analog power supplies (GTY_AVCC, GTY_AVTT, and GTY_AVCCAUX) have planes inside the package. For some of the packages, there are multiple planes for each analog power supply. If there is more than one PSG in the package, the power supply pin names have a suffix (such as _L, or _R) that identifies which pins are associated with which PSG. If all the Quads in a PSG are not used, the associated power pins can be left unconnected or tied to GND. The rules for powering PSGs are as follows:

- Within a package PSG, if no Quads are used, the PSG can be unpowered.
- If any Quads in a PSG are used, the PSG must be powered.
- PSGs on each side (Left or Right) of the package are fully independent. Powering or not powering PSGs on one side of the package does not affect the PSGs on the other sides of the package.
- If a PSG does not have an RCAL master and it is powered, all the PSGs on that side (Left or Right) of the package must be powered.

- If a PSG with an RCAL master is unpowered, any PSG without an RCAL master on that side of the package must also be unpowered.
- A PSG that does not have an RCAL master can be unpowered without affecting other PSGs.

For each GTY transceiver analog power supply group there are three power supplies (GTY_AVCC, GTY_AVTT, and GTY_AVCCAUX). For example, if there are two PSGs in a package, there are a total of six power supply planes in the package for these groups, with three planes in the package for each PSG. The following table shows the power supply groups for Versal ACAPs.

Table 126: **Versal ACAP Transceiver Power Supply Groups and RCAL Master by Package**

Device	Package	GTY Transceiver									
		103	104	105	106	200	201	202	203	204	205	206
VM1802	VC1760	L	L	L RCL	L	RS	RS	RS	RS RCL	RN	RN	RN
	VD1760	L	L	L RCL	L				R RCL	R		
	VA2197	L	L	L RCL	L	R	R	R	R RCL	R	R	R
VC1902	VA1596	L	L	L RCL	L			R	R RCL	R	R	
	VD1760	L	L	L RCL	L				R RCL	R		
	VA2197	L	L	L RCL	L	R	R	R	R RCL	R	R	R

Notes:

1. In each cell, the top row is the power supply group designator. If the second row contains RCL, that Quad is an RCAL master.

Power Supply and Filtering

The GTY transceiver Quad requires three analog power supplies: GTY_AVCC at a nominal voltage level of 0.88 V_{DC}, GTY_VCCAUX at a nominal voltage level of 1.5 V_{DC}, and GTY_AVTT at a nominal voltage level of 1.2 V_{DC} for Versal ACAPs. The pins for each of these analog power supplies are tied to a plane in the package. In some packages, there are two planes (a left plane and a right plane) for each of the analog power supplies. See Chapter 6: Board Design Guidelines for a discussion of the internal power planes in the GTY transceiver packages.

Noise on the GTY transceiver analog power supplies can cause degradation in the performance of the transceivers. The most likely form of degradation is an increase in jitter at the output of the GTY transmitter and reduced jitter tolerance in the receiver. Sources of power supply noise are:

- Power supply regulator noise
- Power distribution network

- Coupling from other circuits

Each of these noise sources must be considered in the design and implementation of the GTY transceiver analog power supplies. The total peak-to-peak noise as measured at the input pin of the Versal ACAP should not exceed 10 mVpp.

Power Up/Down and Reset on Multiple Lanes

The operating state of the GTY transceiver can be controlled through the assertion and deassertion of the power down and reset controls (see Reset and Initialization and Power Down).

When the GTY transceiver's operating state is changed by either changing the power down state or the reset state, the load current as seen by the on-board power distribution network (PDN) and the power supply regulator is also changed. When the load current changes, the power supply regulator must sense the change in the load current and compensate for this change to maintain the design supply voltage. The effect of a delay in the change in the load current can result in a temporary spike or dip in the power supply voltage. When the operating state of the GTY transceiver goes from power down to power up, the load current transient is positive and the voltage from the regulator might dip while the regulator circuit adapts to the new load conditions. Conversely, when the operating state of the GTY transceiver goes from power up to power down, the load current transient is negative and the voltage from the regulator might spike while the regulator circuit adapts to the new load current conditions.

The magnitude and duration of the voltage transient from the power supply regulator depends upon the design of the power supply regulator circuit. In some cases, the voltage might oscillate as the voltage regulator circuit converges to the design voltage setting.

In all of these cases, the important consideration is that the voltage at the input pin of the device must remain within the operating limits as specified in the Versal ACAP data sheets. Use the Xilinx® Power Estimator (XPE) tool to calculate the amount of power required for the transceivers in your application.

Power Supply Regulators

Normally, the GTY transceiver analog voltage supplies have local power supply regulators that provide a final stage of voltage regulation. Preferably these regulators are placed as close as is feasible to the GTY transceiver power supply pins. Minimizing the distance between the analog voltage regulators and the GTY transceiver power supply pins reduces the opportunity for noise coupling into the supply after the regulator and for noise generated by current transients caused by load dynamics.

Linear versus Switching Regulators

The type of power supply regulator can have a significant impact on the complexity, cost, and performance of the power supply circuit. A power supply regulator must provide adequate power to the GTY transceiver with a minimum amount of noise while meeting the overall system thermal and efficiency requirements. There are two major types of power supply voltage regulators available for regulating the GTY transceiver analog voltage rails, linear regulators, and switching regulators. Each of these types of regulators has advantages and disadvantages. The optimal choice of regulator type depends on system requirements such as:

- Physical size
- Thermal budget
- Power efficiency
- Cost

Linear Regulator

A linear regulator is usually the simplest means to provide voltage regulation for the GTY transceiver analog supply rails. Inherently, a linear regulator does not inject significant noise into the regulated output voltage. In fact, some, not all, linear regulators provide noise rejection at the output from noise present on the voltage input. Another advantage of the linear regulator is that it usually requires a minimal number of external components to realize a circuit on the printed circuit board.

There are potentially two major disadvantages to linear regulators, minimum dropout voltage, and limited efficiency. Linear regulators require an input voltage that is higher than the output voltage. This minimum dropout voltage often is dependent on the load current. Even low dropout linear regulators require a minimum difference between the input voltage and the output voltage of the regulator. The system power supply design must consider the minimum dropout voltage requirements of the linear regulators.

The efficiency of a linear regulator is dependent on the voltage difference between the input and output of the linear regulator. For instance, if the input voltage of the regulator is 2.5 V_{DC} and the output voltage of the regulator is 1.2 V_{DC}, the voltage difference is 1.3 V_{DC}. Assuming that the current into the regulator is essentially equal to the current out of the regulator, the maximum efficiency of the regulator is 48%. This means that for every watt delivered to the load, the system must consume an additional watt for regulation. This power consumed by the regulator generates heat that must be dissipated by the system. Providing a means to dissipate the heat generated by the linear regulator can drive up the system cost. So even though from a simple component count and complexity cost, the linear regulator appears to have an advantage over the switching regulator, if the overall system cost is considered, including power consumption and heat dissipation, in high current applications, the linear regulator can actually be at a disadvantage.

Switching Regulator

A switching regulator can provide an efficient means to deliver a well-regulated voltage for the GTY transceiver analog power supply. Unlike the linear regulator, the switching regulator does not depend on the voltage drop between the input voltage of the regulator and the output voltage to provide regulation. Therefore the switching regulator can supply large amounts of current to the load while maintaining high power efficiency. It is not uncommon for a switching regulator to maintain efficiencies of 95% or greater. This efficiency is not severely impacted by the voltage drop between the input of the regulator and the output. It is impacted by the load current in a much lesser degree than that of the linear regulator. Because of the efficiency of the switching regulator, the system does not need to supply as much power to the circuit, and it does not need to provide a means to dissipate power consumed by the regulator.

The disadvantages to the switching regulator are complexity of the circuit and noise generated by the regulator switching function. Switching regulator circuits are usually more complex than linear regulator circuits. This shortcoming in switching regulators has recently been addressed by several switching regulator component vendors. Normally, a switching power supply regulation circuit requires a switching transistor element, an inductor, and a capacitor. Depending on the required efficiency and load requirements, a switching regulator circuit might require external switching transistors and inductors. Besides the component count, these switching regulators require very careful placement and routing on the printed circuit board to be effective.

Switching regulators generate significant noise and therefore usually require additional filtering before the voltage is delivered to the GTY transceiver analog power supply input of the GTY transceiver. As the amplitude of the noise should be limited to less than 10 mVpp, the power supply filter should be designed to attenuate the noise from the switching regulator to meet this requirement.

Power Supply Distribution Network Staged Decoupling

Die

The decoupling capacitance on the die filters the highest frequency noise components on the power supplies. The source of this very high frequency noise is the internal on-die circuits.

Package

The Versal architecture package has additional decoupling. Decoupling capacitors in the package provide attenuation for noise in the package power plane, thereby reducing the interaction between GTY transceiver Quads. These capacitors in the package also aid in maintaining a low-impedance, high-frequency path between the power supply, GTY_AVCC, GTY_VCCAUX, or GTY_AVTT, and GND.

Printed Circuit Board

Because the impedance between the power planes and GND has been kept low on the die and in the package, the board design has a much more relaxed requirement for decoupling on the printed circuit board. The primary purpose of the PCB decoupling capacitors is to provide noise isolation between the transceiver power supply pins and the external noise sources. Some examples of external noise sources are:

- Power supply regulator circuits
- On-board digital switching circuits
- SelectIO signals from the Versal ACAP

Decoupling capacitors should be provided on the PCB near the GTY transceiver power pins. These capacitors reduce the impedance of the PCB power distribution network. The reduced impedance of the PDN provides a means to attenuate noise from external sources before it can get into the device package power planes. The noise at the power pins should be less than 10 mVpp over the band from 10 kHz to 80 MHz.

Use the Xilinx Power Estimator tool (XPE) to determine the PCB capacitor recommendations. The number and usage of GTY transceivers is used by XPE to calculate the decoupling capacitors required for each of the GTY transceiver analog power supplies. The GTY transceiver Quads are organized into power supply groups in the package. See Analog Power Supply Pins for the package being used.

PCB Design Checklist

The following table is a checklist of items that can be used to design and review any GTY transceiver PCB schematic and layout.

Table 127: **GTY Transceiver PCB Design Checklist**

Pins	Recommendation
MGTREFCLK0P MGTREFCLK0N MGTREFCLK1P MGTREFCLK1N	When configured as an input: • Use AC coupling capacitors for connection to oscillator. • For AC coupling capacitors, see Reference Clock Interface. • Reference clock oscillator output must comply with the minimum and maximum input amplitude requirements for these input pins. See the Versal ACAP data sheets. When configured as an output: • Use AC coupling capacitors for connection to receiving device. • For AC coupling capacitors use 0.01 µF. • For output signal characteristics, see the Versal ACAP data sheets. • If reference pins are not used, leave the associated pin pair unconnected. However, if the IBUFDS_GTE3/4 is instantiated in the design but not used, the associated pin pair should be connected to GND.
GTY_RXP[3:0] GTY_RXN[3:0]	• Use AC coupling capacitors for connection to transmitter. The recommended value for AC coupling capacitor is 100 nF. • Receiver data traces should be provided enough clearance to eliminate crosstalk from adjacent signals. • If a receiver will never be used under any conditions, connect the associated pin pair to GND. • If a receiver is not used and not connected to anything under some conditions, but might be connected to something and used under other conditions, then for the conditions when the receiver is unused, either do not instance the GTY transceiver in the Versal ACAP design, or if the GTY transceiver is instanced, set RXPD[1:0] to `2'b11`. • See RX Analog Front End for more details.
GTY_TXP[3:0] GTY_TXN[3:0]	• The transmitter should be AC coupled to the receiver. The recommended value for the AC coupling capacitor is 100 nF. • Transmitter data traces should be provided enough clearance to eliminate crosstalk from adjacent signals. • If a transmitter is not used, leave the associated pin pair unconnected.
MGTAVTTRCAL	• Connect to GTY_AVTT and to a 100Ω resistor that is also connected to MGTRREF. Use identical trace geometry for the connection between the resistor and this pin and for the connection from the other pin of the resistor to MGTRREF. Also, the DC resistance of the PCB trace should be limited to less than 0.5Ω. • See Termination Resistor Calibration Circuit. • If an entire PSG is not used by any Quads, tie MGTAVTTRCAL to ground.
MGTRREF	• Connect a 100Ω resistor that is also connected to MGTAVTTRCAL. Use identical trace geometry for the connection between the resistor and this pin and for the connection from the other pin of the resistor to MGTAVTTRCAL. Also, the DC resistance of the PCB trace should be limited to less than 0.5Ω. • See Termination Resistor Calibration Circuit. • If an entire PSG is not used by any Quads, tie MGTRREF to ground.

Table 127: **GTY Transceiver PCB Design Checklist** *(cont'd)*

Pins	Recommendation
GTY_AVCC	• The nominal voltage is 0.88 V_{DC}. • See the Versal ACAP data sheets for power supply voltage tolerances. • The power supply regulator for this voltage should not be shared with non-transceiver loads. • Many packages have multiple groups of power supply connections in the package for GTY_AVCC. Information on pin locations for each package can be found in *Versal ACAP Packaging and Pinouts Architecture Manual* (AM013). • For optimal performance, power supply noise must be less than 10 mVpp. • If all of the Quads in a power supply group are not used, the associated power pins can be left unconnected or tied to GND. • For power consumption and filter capacitor recommendations, refer to the Xilinx Power Estimator (XPE) at www.xilinx.com/power.
GTY_AVTT	• The nominal voltage is 1.2 V_{DC}. • See the Versal ACAP data sheets for power supply voltage tolerances. • The power supply regulator for this voltage should not be shared with non-transceiver loads. • Many packages have multiple groups of power supply connections in the package for GTY_AVTT. Information on pin locations for each package can be found in *Versal ACAP Packaging and Pinouts Architecture Manual* (AM013). • For optimal performance, power supply noise must be less than 10 mVpp. • If all of the Quads in a power supply group are not used, the associated power pins can be left unconnected or tied to GND. • For power consumption and filter capacitor recommendations, refer to XPE at www.xilinx.com/power.
GTY_AVCCAUX	• The nominal voltage is 1.5 V_{DC}. • See the Versal ACAP data sheets for power supply voltage tolerances. • The power supply regulator for this voltage should not be shared with non-transceiver loads. • Many packages have multiple groups of power supply connections in the package for GTY_AVCCAUX. Information on pin locations for each package can be found in *Versal ACAP Packaging and Pinouts Architecture Manual* (AM013). • For optimal performance, power supply noise must be less than 10 mVpp. • If all the LCPLLs in this power supply group are not used but the Quads are used, the filter capacitors are not necessary, and these pins can be connected to VCCAUX. • If all of the Quads in a power supply group are not used, the associated power pins can be left unconnected or tied to GND. • For power consumption and filter capacitor recommendations, refer to XPE at www.xilinx.com/power.

XILINX

Appendix A

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

Documentation Navigator and Design Hubs

Xilinx® Documentation Navigator (DocNav) provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open DocNav:

- From the Vivado® IDE, select **Help → Documentation and Tutorials**.
- On Windows, select **Start → All Programs → Xilinx Design Tools → DocNav**.
- At the Linux command prompt, enter `docnav`.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In DocNav, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

Note: For more information on DocNav, see the Documentation Navigator page on the Xilinx website.

References

These documents provide supplemental material useful with this guide:

1. Versal ACAP data sheets:
   - *Versal Architecture and Product Data Sheet: Overview* (DS950)
   - *Versal Prime Series Data Sheet: DC and AC Switching Characteristics* (DS956)
   - *Versal AI Core Series Data Sheet: DC and AC Switching Characteristics* (DS957)
2. *Versal ACAP Packaging and Pinouts Architecture Manual* (AM013)
3. *Vivado Design Suite User Guide: Logic Simulation* (UG900)

Please Read: Important Legal Notices

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.

Copyright