UltraScale Architecture Integrated Block for 100G Ethernet v1.6

LogiCORE IP Product Guide

Vivado Design Suite

PG165 June 24, 2015

XILINX
ALL PROGRAMMABLE™

Table of Contents

Chapter 5: Example Design

Appendix A: Migrating and Upgrading

Appendix B: Debugging

Appendix C: Additional Resources and Legal Notices

Introduction

The Xilinx® UltraScale™ architecture integrated block for the LogiCORE™ 100G Ethernet IP core provides a high performance, low latency 100 Gb/s Ethernet port that allows for a wide range of user customization and statistics gathering. The dedicated block provides both the 100G Ethernet media access control (MAC) and physical coding sublayer (PCS) logic with support for *IEEE 1588-2008* [Ref 1] two-step hardware timestamping.

The 100G Ethernet IP core provides three configurations: (CAUI-10) 10x10.3125G, (CAUI-4) 4x25.78125G, and switchable CAUI-4 and CAUI-10 mode. The 100G Ethernet IP core is designed to the *IEEE std 802.3-2012* [Ref 2] specification.

Features

- Supports CAUI-10, CAUI-4, and switchable CAUI-4 and CAUI-10 modes
- 512-bit segmented local bus (LBUS) user interface at ~322 MHz
- 32-bit interface to the serial transceiver for CAUI-10 lanes and 80-bit interface to the serial transceiver for CAUI-4 lanes
- *IEEE 1588-2008* [Ref 1] one-step and two-step hardware timestamping at ingress and egress at full 80-bits
- Pause frame processing including priority based flow control per *IEEE std 802.3-2012* Annex 31 [Ref 2]
- Dynamic and static deskew support

See Feature Summary in Chapter 1 for a list of more features.

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family(1)	Virtex® UltraScale Architecture
Supported User Interfaces	Segmented LBUS
Resources	Performance and Resource Utilization web page
Provided with Core	
Design Files	Verilog
Example Design	Verilog
Test Bench	Verilog
Constraints File	Xilinx Design Constraints (XDC)
Simulation Model	Verilog
Supported S/W Driver	N/A
Tested Design Flows(2)	
Design Entry	Vivado® Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado synthesis
Support	
Provided by Xilinx at the Xilinx Support web page	

Notes:

1. For a complete list of supported devices, see the Vivado IP catalog.
2. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

This product guide describes the function and operation of the Xilinx® UltraScale™ architecture integrated block for the 100G Ethernet IP core, including how to design, customize, and implement it.

The core is designed to the *IEEE std 802.3-2012* [Ref 2] specification with an option for *IEEE 1588-2008* [Ref 1] hardware timestamping. The core instantiates the UltraScale architecture integrated block for 100G Ethernet. This core simplifies the design process and reduces time to market.

Although the core is a fully-verified solution, implementing a complete design varies depending on the configuration and functionality of the application. See Chapter 2, Product Specification for details about the core.

RECOMMENDED: *For best results, previous experience building high performance, pipelined FPGA designs using Xilinx implementation design tools and constraint files is recommended.*

IMPORTANT: *CAUI-4 and switchable CAUI-10/CAUI-4 require GTY transceivers that are available only in Virtex® UltraScale devices.*

Feature Summary

- One-step and two-step IEEE 1588-2008 [Ref 1] hardware timestamping with transparent clock support
- 20 PCS lanes (PCSLs) for the 100G Ethernet IP core
- GTY or GTH transceivers used for UltraScale devices
- PCS Lane marker framing and de-framing including reordering of each PCS lane
- Link status and alignment monitoring reporting
- 64B/66B decoding and encoding as defined in *IEEE std 802.3-2012* Clause 82 [Ref 2]
- Scrambling and descrambling using $x^{58} + x^{39} + 1$ polynomial
- Inter-Packet gap (IPG) insertion and deletion as required by *IEEE std 802.3-2012* Clause 82 [Ref 2]

- Optional frame check sequence (FCS) calculation and addition in the transmit direction
- FCS checking and optional FCS removal in the receive direction
- Support for 802.3x and priority-based pause operation
- DRP interface for dynamic reconfiguration of the core
- Detailed statistics gathering
  - Total bytes
  - Total packets
  - Good bytes
  - Good packets
  - Unicast packets
  - Multicast packets
  - Broadcast packets
  - Pause packets
  - Virtual local area network (VLAN) tagged packets
  - 64B/66B code violations
  - Bad preambles
  - Bad FCS
  - Packet histogram for packets sized between:
    - 64
    - 65-127
    - 128-255
    - 256-511
    - 512-1023
    - 1024-1518
    - 1519-1522
    - 1523-1548
    - 1549-2047
    - 2048-4095
    - 4096-8191
    - 8192-9215

Licensing and Ordering Information

This Xilinx LogiCORE™ IP module is provided at no additional cost with the Xilinx Vivado® Design Suite under the terms of the Xilinx End User License.

Information about this and other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information about pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

For more information and to generate a no-charge license key, visit the UltraScale Integrated 100G Ethernet MAC/PCS.

Product Specification

Table 2-1 defines the integrated CMAC block for the 100 Gb/s Ethernet solution.

Table 2-1: **Integrated CMAC Block for the 100 Gb/s Ethernet Solution**

Protocol	Lane Width	Line Rate	SerDes	SerDes Width
CAUI-10	x10	10.3125 Gb/s	GTH GTY	32b
CAUI-4	x4	25.78125 Gb/s(2)	GTY(1)	80b
Runtime Switchable CAUI-4/ CAUI-10	CAUI-10: x10 CAUI-4: x4	CAUI-10: 10.3125 Gb/s CAUI-4: 25.78125 Gb/s	GTY(1)	CAUI-10: 32b CAUI-4: 80b

Notes:

1. CAUI-4 and switchable CAUI-10/CAUI-4 require GTY transceivers that are available only in Virtex® UltraScale™ devices.
2. The line rate of 25.78125 Gb/s is available on select devices, Virtex UltraScale devices in typical speed grades.

The core instantiates the CMAC block along with the necessary GTH or GTY transceivers. The core provides an example of how the two blocks are connected together, along with the reset and clocking for those blocks.

The integrated block is designed to *IEEE std 802.3-2012* [Ref 2].

Figure 2-1 illustrates the following interfaces to the integrated CMAC block.

- Serial transceiver interface
- User-side transmit and receive LBUS interface
- Pause processing
- *IEEE 1588-2008* [Ref 1] timestamping interface
- Status/Control interface ‘
- Dynamic reconfiguration port (DRP) interface used for configuration

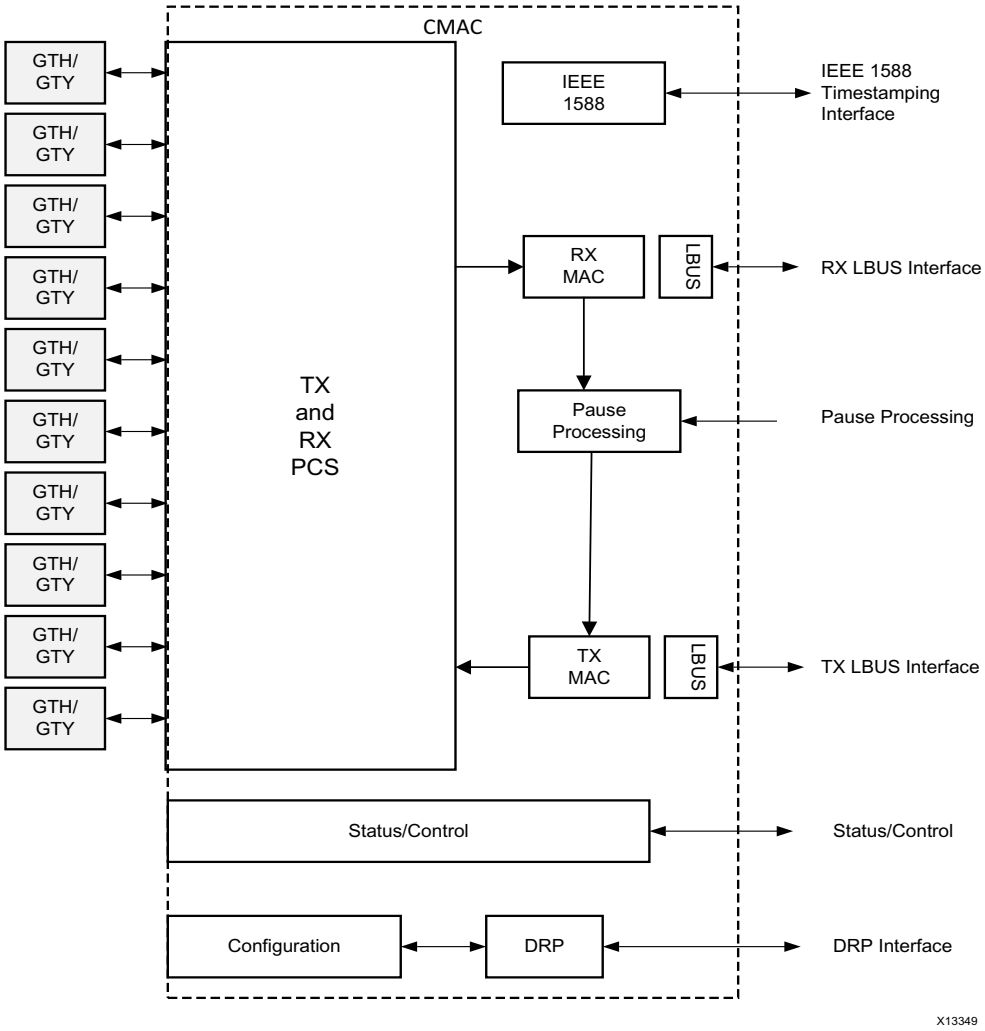

Figure 2-1: **Integrated CMAC Block for 100 Gb/s Ethernet**

Typical Operation

The 100G Ethernet IP core handles all protocol related functions to communicate to the other devices PCS and Ethernet MAC interface. This includes handshaking, synchronizing and error checking. You provide packet data through the Local Bus (LBUS) TX interface and receive packet data from the LBUS RX interface. The LBUS is designed to match commonly used packet bus protocols made common by the SPI4.2 and Interlaken protocols. A detailed description is given in User Side LBUS Interface in Chapter 3.

The core is designed to be flexible and used in many different applications. The RX path does not perform any buffering other than the pipelining required to perform the required operations. Received data is passed directly to the user interface in a cut-through manner, allowing you the flexibility to implement any required buffering scheme. Also, the core TX path consists of a single pipeline with minimal buffering to provide reliable cut-through operation.

Statistics Gathering

The 100G Ethernet IP core provides a flexible and user-friendly mechanism for gathering statistics. For all the supported statistics, the core has an output signal (or bus if needed) that indicates an increment value for the statistic in a given clock cycle. This allows the increment value to build the required counter mechanism. This mechanism allows you to select which statistics are required in the system without having the cost overhead of a full set of counters. Additionally, and more importantly, you can implement any counter and statistics gathering mechanism required by the system. For example, you can build 32-bit or 64-bit counters as needed, or implement clear-on-read or saturated counters, as required.

For the purposes of TX statistics, good packets are defined as packets without FCS or other errors; bad packets are defined as packets with FCS or any other error.

For the purposes of RX statistics, good packets are defined as packets without FCS or other errors including length error. Bad packets are defined as packets with FCS or any other error. The length field error includes length field error, oversize and undersize packets.

Testability Functions

The 100G Ethernet example design implements the test pattern generation and checking as defined in Clause 82.2.10 (Test-pattern generators) and Clause 82.2.17 (Test-pattern checker). See the IEEE 802.3 documents for details.

Pause Operation

The 100G Ethernet IP core is capable of handling 802.3x and priority-based pause operation. The RX path parses pause packets and presents the extracted quanta on the status interface; the TX path can accept pause packet requests from the control interface and will inject the requested packets into the data stream. Both global pause packets and priority-based pause packets are handled. Details are described in Pause Processing Interface in Chapter 3.

Note: "802.3x" and "global pause" are used interchangeably throughout the document.

Standards

The 100G Ethernet IP core is designed to be compliant with the *IEEE std 802.3-2012* [Ref 2] specification. The timestamping feature is designed to be compliant with *IEEE 1588-2008* [Ref 1].

Performance

The 100G Ethernet IP core is designed to operate with the performance characteristics of the CMAC primitive it instantiates.

See the *Virtex UltraScale Architecture Data Sheet: DC and AC Switching Characteristics* (DS893) [Ref 3] for the maximum frequencies allowed on the 100G Ethernet IP core specified by speed grade.

IMPORTANT: *A free-running clock input, `init_clk`, is required for the transceiver portion of the 100G Ethernet IP core. See the UltraScale FPGAs Transceiver Wizards (PG182) [Ref 4] for more information on the `gtwiz_reset_clk_freerun_in` input port.*

Resource Utilization

For full details about performance and resource utilization, visit Performance and Resource Utilization.

Port Descriptions

Table 2-2 provides a detailed description of the 100G Ethernet IP core ports.

IMPORTANT: *CAUI-4 and switchable CAUI-10/CAUI-4 require GTY transceivers that are available only in Virtex UltraScale devices.*

Table 2-2: **Transceiver I/O**

Name	Direction	Description
RX_SERDES_ALT_DATA0[15:0]	Input	16-bit group of the Receive data bus from SerDes0. There are 10 RX_SERDES_DATA buses; one bus for each SerDes lane, and each bus has either 80 or 32 bits depending on whether operation is in CAUI-4 or CAUI-10 mode respectively. The first four SerDes lanes can operate at 80 bits or 32 bits, and the remaining six lanes operate at 32 bits. The 32 LSBs of the first four lanes are used in CAUI-10 mode. The mapping of the 80 bits, comprised of a 16-bit group and a 64-bit group, is not obvious. See PCS Lane Multiplexing in Chapter 3 for details.
RX_SERDES_ALT_DATA1[15:0]	Input	16-bit group of the Receive data bus from SerDes1.
RX_SERDES_ALT_DATA2[15:0]	Input	16-bit group of the Receive data bus from SerDes2.
RX_SERDES_ALT_DATA3[15:0]	Input	16-bit group of the Receive data bus from SerDes3.
RX_SERDES_DATA0[63:0]	Input	64-bit group of the Receive data bus from SerDes0
RX_SERDES_DATA1[63:0]	Input	64-bit group of the Receive data bus from SerDes1.
RX_SERDES_DATA2[63:0]	Input	64-bit group of the Receive data bus from SerDes2
RX_SERDES_DATA3[63:0]	Input	64-bit group of the Receive data bus from SerDes3
RX_SERDES_DATA4[31:0]	Input	Data bus from SerDes4.
RX_SERDES_DATA5[31:0]	Input	Data bus from SerDes5.
RX_SERDES_DATA6[31:0]	Input	Data bus from SerDes6.
RX_SERDES_DATA7[31:0]	Input	Data bus from SerDes7.
RX_SERDES_DATA8[31:0]	Input	Data bus from SerDes8.
RX_SERDES_DATA9[31:0]	Input	Data bus from SerDes9.

Table 2-2: **Transceiver I/O *(Cont'd)***

Name	Direction	Description
TX_SERDES_ALT_DATA0[15:0]	Output	16-bit group of the Transmit data bus to SerDes0. There are 10 TX_SERDES_DATA buses; one bus for each SerDes lane, and each bus has either 80 or 32 bits depending on whether operation is in CAUI-4 or CAUI-10 mode respectively. The first four SerDes lanes can operate at 80 bits or 32 bits, and the remaining six lanes operate at 32 bits. The 32 LSBs of the first four lanes are used in CAUI-10 mode. The mapping of the 80 bits, comprised of a 16-bit group and a 64-bit group, is not obvious. See PCS Lane Multiplexing in Chapter 3 for details.
TX_SERDES_ALT_DATA1[15:0]	Output	16-bit group of the Transmit data bus to SerDes1.
TX_SERDES_ALT_DATA2[15:0]	Output	16-bit group of the Transmit data bus to SerDes2.
TX_SERDES_ALT_DATA3[15:0]	Output	16-bit group of the Transmit data bus to SerDes3.
TX_SERDES_DATA0[63:0]	Output	64-bit group of the Transmit data bus to SerDes0
TX_SERDES_DATA1[63:0]	Output	64-bit group of the Transmit data bus to SerDes1
TX_SERDES_DATA2[63:0]	Output	64-bit group of the Transmit data bus to SerDes2
TX_SERDES_DATA3[63:0]	Output	64-bit group of the Transmit data bus to SerDes3
TX_SERDES_DATA4[31:0]	Output	Data bus to SerDes4.
TX_SERDES_DATA5[31:0]	Output	Data bus to SerDes5.
TX_SERDES_DATA6[31:0]	Output	Data bus to SerDes6.
TX_SERDES_DATA7[31:0]	Output	Data bus to SerDes7.
TX_SERDES_DATA8[31:0]	Output	Data bus to SerDes8.
TX_SERDES_DATA9[31:0]	Output	Data bus to SerDes9.
RX_SERDES_CLK[9:0]	Input	Recovered clock of each SerDes lane. The RX_SERDES_DATA bus for each lane is synchronized to the positive edge of the corresponding bit of this bus.
RX_SERDES_RESET[9:0]	Input	Reset for each RX SerDes lane. The recovered clock for each SerDes lane has associated with it an active-High reset. This signal should be 1 whenever the associated recovered clock is not operating at the correct frequency. Generally this signal is derived from a PLL lock signal. This reset signal should be held in reset until the serial transceiver (GT) is finished its initialization and the RX_SERDES_CLK is stable.

Table 2-3: **LBUS Interface – Clock/Reset Signals**

Name	Direction	Description
TX_CLK	Input	TX clock. All TX signals between the 100G Ethernet IP core and the user-side logic are synchronized to the positive edge of this signal. The clock frequency is equal to the line rate divided by the SerDes width. This frequency is nominally 322.265625 MHz.
RX_CLK	Input	RX clock. All RX signals between the 100G Ethernet IP core and the user-side logic are synchronized to the positive edge of this signal. The frequency of this clock should be the same as the TX clock.
RX_RESET	Input	Reset for the RX circuits. This signal is active-High (1 = reset) and must be held High until RX_CLK is stable.The 100G Ethernet IP core handles synchronizing the RX_RESET input to the appropriate clock domains within the 100G Ethernet IP core.
TX_RESET	Input	Reset for the TX circuits. This signal is active-High (1 = reset) and must be held High until TX_CLK is stable. The 100G Ethernet IP core handles synchronizing the TX_RESET input to the appropriate clock domains within the 100G Ethernet IP core.

Table 2-4: **LBUS Interface – RX Path Signals**

Name	Direction	Description
RX_DATAOUT0[127:0]	Output	Receive segmented LBUS Data for segment 0. The value of this bus is only valid in cycles that RX_ENAOUT0 is sampled as 1.
RX_DATAOUT1[127:0]	Output	Receive segmented LBUS Data for segment1.
RX_DATAOUT2[127:0]	Output	Receive segmented LBUS Data for segment2.
RX_DATAOUT3[127:0]	Output	Receive segmented LBUS Data for segment3.
RX_ENAOUT0	Output	Receive LBUS Enable for segment0. This signal qualifies the other signals of the RX segmented LBUS Interface. Signals of the RX LBUS Interface are only valid in cycles in which RX_ENAOUT is sampled as a 1.
RX_ENAOUT1	Output	Receive LBUS Enable for segment1.
RX_ENAOUT2	Output	Receive LBUS Enable for segment2.
RX_ENAOUT3	Output	Receive LBUS Enable for segment3.

Table 2-4: **LBUS Interface – RX Path Signals *(Cont'd)***

Name	Direction	Description
RX_SOPOUT0	Output	Receive LBUS start of packet (SOP) for segment0. This signal indicates the SOP when it is sampled as a 1 and is only valid in cycles in which RX_ENAOUT is sampled as a 1.
RX_SOPOUT1	Output	Receive LBUS SOP for segment1.
RX_SOPOUT2	Output	Receive LBUS SOP for segment2.
RX_SOPOUT3	Output	Receive LBUS SOP for segment3.
RX_EOPOUT0	Output	Receive LBUS end of packet (EOP) for segment0. This signal indicates the EOP when it is sampled as a 1 and is only valid in cycles in which RX_ENAOUT is sampled as a 1.
RX_EOPOUT1	Output	Receive LBUS EOP for segment1.
RX_EOPOUT2	Output	Receive LBUS EOP for segment2.
RX_EOPOUT3	Output	Receive LBUS EOP for segment3.
RX_ERROUT0	Output	Receive LBUS Error for segment0. This signal indicates that the current packet being received has an error when it is sampled as a 1. This signal is only valid in cycles when both RX_ENAOUT and RX_EOPOUT are sampled as a 1. When this signal is a value of 0, it indicates that there is no error in the packet being received.
RX_ERROUT1	Output	Receive LBUS Error for segment1.
RX_ERROUT2	Output	Receive LBUS Error for segment2.
RX_ERROUT3	Output	Receive LBUS Error for segment3.
RX_MTYOUT0[3:0]	Output	Receive LBUS Empty for segment0. This bus indicates how many bytes of the RX_DATAOUT bus are empty or invalid for the last transfer of the current packet. This bus is only valid in cycles when both RX_ENAOUT and RX_EOPOUT are sampled as 1. When RX_ERROUT and RX_ENAOUT are sampled as 1, the value of RX_MTYOUT[3:0] is always 000. Other bits of RX_MTYOUT are as usual.
RX_MTYOUT1[3:0]	Output	Receive LBUS Empty for segment1.
RX_MTYOUT2[3:0]	Output	Receive LBUS Empty for segment2.
RX_MTYOUT3[3:0]	Output	Receive LBUS Empty for segment3.

Table 2-5: **LBUS Interface – TX Path Signals**

Name	Direction	Description
TX_RDYOUT	Output	Transmit LBUS Ready. This signal indicates whether the dedicated 100G Ethernet IP core TX path is ready to accept data and provides back-pressure to the user logic. A value of 1 means the user logic can pass data to the 100G Ethernet IP core. A value of 0 means the user logic must stop transferring data to the 100G Ethernet IP core within a certain number of cycles or there will be an overflow.
TX_OVFOUT	Output	Transmit LBUS Overflow. This signal indicates whether you have violated the back-pressure mechanism provided by the TX_RDYOUT signal. If TX_OVFOUT is sampled as a 1, a violation has occurred. It is up to you to design the rest of the user logic to not overflow the TX interface. In the event of an overflow condition, the TX path must be reset.
TX_UNFOUT	Output	Transmit LBUS Underflow. This signal indicates whether you have under-run the LBUS interface. If TX_UNFOUT is sampled as 1, a violation has occurred meaning the current packet is corrupted. Error control blocks are transmitted as long as the underflow condition persists. It is up to the user logic to ensure a complete packet is input to the core without under-running the LBUS interface.
TX_DATAIN0[127:0]	Input	Transmit segmented LBUS Data for segment0. This bus receives input data from the user logic. The value of the bus is captured in every cycle that TX_ENAIN is sampled as 1.
TX_DATAIN1[127:0]	Input	Transmit segmented LBUS Data for segment1.
TX_DATAIN2[127:0]	Input	Transmit segmented LBUS Data for segment2.
TX_DATAIN3[127:0]	Input	Transmit segmented LBUS Data for segment3.
TX_ENAIN0	Input	Transmit LBUS Enable for segment0. This signal is used to enable the TX LBUS Interface. All signals on this interface are sampled only in cycles in which TX_ENAIN is sampled as a 1.
TX_ENAIN1	Input	Transmit LBUS Enable for segment1.
TX_ENAIN2	Input	Transmit LBUS Enable for segment2.
TX_ENAIN3	Input	Transmit LBUS Enable for segment3.
TX_SOPIN0	Input	Transmit LBUS SOP for segment0. This signal is used to indicate the SOP when it is sampled as a 1 and is 0 for all other transfers of the packet. This signal is sampled only in cycles in which TX_ENAIN is sampled as a 1.
TX_SOPIN1	Input	Transmit LBUS SOP for segment1.

Table 2-5: **LBUS Interface – TX Path Signals *(Cont'd)***

Name	Direction	Description
TX_SOPIN2	Input	Transmit LBUS SOP for segment2.
TX_SOPIN3	Input	Transmit LBUS SOP for segment3.
TX_EOPIN0	Input	Transmit LBUS EOP for segment0. This signal is used to indicate the EOP when it is sampled as a 1 and is 0 for all other transfers of the packet. This signal is sampled only in cycles in which TX_ENAIN is sampled as a 1.
TX_EOPIN1	Input	Transmit LBUS EOP for segment1.
TX_EOPIN2	Input	Transmit LBUS EOP for segment2.
TX_EOPIN3	Input	Transmit LBUS EOP for segment3.
TX_ERRIN0	Input	Transmit LBUS Error for segment0. This signal is used to indicate a packet contains an error when it is sampled as a 1 and is 0 for all other transfers of the packet. This signal is sampled only in cycles in which TX_ENAIN and TX_EOPIN are sampled as 1. When this signal is sampled as a 1, the last data word is replaced with the IEEE standard 802.3-2012 Error Code control word that guarantees the partner device receives the packet in error. If a packet is input with this signal set to a 1, the FCS checking and reporting is disabled (only for that packet).
TX_ERRIN1	Input	Transmit LBUS Error for segment1.
TX_ERRIN2	Input	Transmit LBUS Error for segment2.
TX_ERRIN3	Input	Transmit LBUS Error for segment3.
TX_MTYIN0[3:0]	Input	Transmit LBUS Empty for segment0. This bus is used to indicate how many bytes of the TX_DATAIN bus are empty or invalid for the last transfer of the current packet. This bus is sampled only in cycles that TX_ENAIN and TX_EOPIN are sampled as 1. When TX_EOPIN and TX_ERRIN are sampled as 1, the value of TX_MTYIN[2:0] is ignored as treated as if it was 000. The other bits of TX_MTYIN are used as usual.
TX_MTYIN1[3:0]	Input	Transmit LBUS Empty for segment1.
TX_MTYIN2[3:0]	Input	Transmit LBUS Empty for segment2.
TX_MTYIN3[3:0]	Input	Transmit LBUS Empty for segment3.

Table 2-6: **LBUS Interface – TX Path Control/Status Signals**

Name	Direction	Description
CTL_TX_ENABLE	Input	TX Enable. This signal is used to enable the transmission of data when it is sampled as a 1. When sampled as a 0, only idles are transmitted by the 100G Ethernet IP core. This input should not be set to 1 until the receiver it is sending data to (that is, the receiver in the other device) is fully aligned and ready to receive data (that is, the other device is not sending a remote fault condition). Otherwise, loss of data can occur. If this signal is set to 0 while a packet is being transmitted, the current packet transmission is completed and then the 100G Ethernet IP core stops transmitting anymore packets.
CTL_TX_SEND_RFI	Input	Transmit Remote Fault Indication (RFI) code word. If this input is sampled as a 1, the TX path only transmits Remote Fault code words. This input should be set to 1 until the RX path is fully aligned and is ready to accept data from the link partner.
CTL_TX_SEND_IDLE	Input	Transmit Idle code words. If this input is sampled as a 1, the TX path only transmits Idle code words. This input should be set to 1 when the partner device is sending Remote Fault Indication (RFI) code words.
STAT_TX_LOCAL_FAULT	Output	A value of 1 indicates the receive decoder state machine is in the TX_INIT state. This output is level sensitive.

Table 2-7: **LBUS Interface – RX Path Control/Status Signals**

Name	Direction	Description
CTL_RX_ENABLE	Input	RX Enable. For normal operation, this input must be set to 1. When this input is set the to 0, after the RX completes the reception of the current packet (if any), it stops receiving packets by keeping the PCS from decoding incoming data. In this mode, there are no statistics reported and the LBUS interface is idle.
CTL_RX_FORCE_RESYNC	Input	RX force resynchronization input. This signal is used to force the RX path to reset, re-synchronize, and realign. A value of 1 forces the reset operation. A value of 0 allows normal operation. ***Note:*** This input should normally be Low and should only be pulsed (one cycle minimum pulse) to force realignment.

Table 2-7: **LBUS Interface – RX Path Control/Status Signals *(Cont'd)***

Name	Direction	Description
STAT_RX_FRAMING_ERR_0[3:0]	Output	RX sync header bits framing error for lane 0. Each PCS Lane has a four-bit bus that indicates how many sync header errors were received for that PCS Lane. The value of the bus is only valid when the corresponding STAT_RX_FRAMING_ERR_VALID_[19:0] is a 1. The values on these buses can be updated at any time and are intended to be used as increment values for sync header error counters.
STAT_RX_FRAMING_ERR_1[3:0]	Output	RX sync header bits framing error for lane 1.
STAT_RX_FRAMING_ERR_2[3:0]	Output	RX sync header bits framing error for lane 2.
STAT_RX_FRAMING_ERR_3[3:0]	Output	RX sync header bits framing error for lane 3.
STAT_RX_FRAMING_ERR_4[3:0]	Output	RX sync header bits framing error for lane 4.
STAT_RX_FRAMING_ERR_5[3:0]	Output	RX sync header bits framing error for lane 5.
STAT_RX_FRAMING_ERR_6[3:0]	Output	RX sync header bits framing error for lane 6.
STAT_RX_FRAMING_ERR_7[3:0]	Output	RX sync header bits framing error for lane 7.
STAT_RX_FRAMING_ERR_8[3:0]	Output	RX sync header bits framing error for lane 8.
STAT_RX_FRAMING_ERR_9[3:0]	Output	RX sync header bits framing error for lane 9.
STAT_RX_FRAMING_ERR_10[3:0]	Output	RX sync header bits framing error for lane 10.
STAT_RX_FRAMING_ERR_11[3:0]	Output	RX sync header bits framing error for lane 11.
STAT_RX_FRAMING_ERR_12[3:0]	Output	RX sync header bits framing error for lane 12.
STAT_RX_FRAMING_ERR_13[3:0]	Output	RX sync header bits framing error for lane 13.
STAT_RX_FRAMING_ERR_14[3:0]	Output	RX sync header bits framing error for lane 14.
STAT_RX_FRAMING_ERR_15[3:0]	Output	RX sync header bits framing error for lane 15.
STAT_RX_FRAMING_ERR_16[3:0]	Output	RX sync header bits framing error for lane 16.
STAT_RX_FRAMING_ERR_17[3:0]	Output	RX sync header bits framing error for lane 17.
STAT_RX_FRAMING_ERR_18[3:0]	Output	RX sync header bits framing error for lane 18.
STAT_RX_FRAMING_ERR_19[3:0]	Output	RX sync header bits framing error for lane 19.
STAT_RX_FRAMING_ERR_VALID_0	Output	Valid indicator for STAT_RX_FRAMING_ERR_0[3:0]. When this output is sampled as a 1, the value on the corresponding STAT_RX_FRAMING_ERR_0[3:0] is valid.
STAT_RX_FRAMING_ERR_VALID_1	Output	Valid indicator for STAT_RX_FRAMING_ERR_1[3:0].
STAT_RX_FRAMING_ERR_VALID_2	Output	Valid indicator for STAT_RX_FRAMING_ERR_2[3:0].
STAT_RX_FRAMING_ERR_VALID_3	Output	Valid indicator for STAT_RX_FRAMING_ERR_3[3:0].
STAT_RX_FRAMING_ERR_VALID_4	Output	Valid indicator for STAT_RX_FRAMING_ERR_4[3:0].
STAT_RX_FRAMING_ERR_VALID_5	Output	Valid indicator for STAT_RX_FRAMING_ERR_5[3:0].
STAT_RX_FRAMING_ERR_VALID_6	Output	Valid indicator for STAT_RX_FRAMING_ERR_6[3:0].
STAT_RX_FRAMING_ERR_VALID_7	Output	Valid indicator for STAT_RX_FRAMING_ERR_7[3:0].
STAT_RX_FRAMING_ERR_VALID_8	Output	Valid indicator for STAT_RX_FRAMING_ERR_8[3:0].

Table 2-7: **LBUS Interface – RX Path Control/Status Signals *(Cont'd)***

Name	Direction	Description
STAT_RX_FRAMING_ERR_VALID_9	Output	Valid indicator for STAT_RX_FRAMING_ERR_9[3:0].
STAT_RX_FRAMING_ERR_VALID_10	Output	Valid indicator for STAT_RX_FRAMING_ERR_10[3:0].
STAT_RX_FRAMING_ERR_VALID_11	Output	Valid indicator for STAT_RX_FRAMING_ERR_11[3:0].
STAT_RX_FRAMING_ERR_VALID_12	Output	Valid indicator for STAT_RX_FRAMING_ERR_12[3:0].
STAT_RX_FRAMING_ERR_VALID_13	Output	Valid indicator for STAT_RX_FRAMING_ERR_13[3:0].
STAT_RX_FRAMING_ERR_VALID_14	Output	Valid indicator for STAT_RX_FRAMING_ERR_14[3:0].
STAT_RX_FRAMING_ERR_VALID_15	Output	Valid indicator for STAT_RX_FRAMING_ERR_15[3:0].
STAT_RX_FRAMING_ERR_VALID_16	Output	Valid indicator for STAT_RX_FRAMING_ERR_16[3:0].
STAT_RX_FRAMING_ERR_VALID_17	Output	Valid indicator for STAT_RX_FRAMING_ERR_17[3:0].
STAT_RX_FRAMING_ERR_VALID_18	Output	Valid indicator for STAT_RX_FRAMING_ERR_18[3:0].
STAT_RX_FRAMING_ERR_VALID_19	Output	Valid indicator for STAT_RX_FRAMING_ERR_19[3:0].
STAT_RX_LOCAL_FAULT	Output	This output is High when STAT_RX_INTERNAL_LOCAL_FAULT or STAT_RX_RECEIVED_LOCAL_FAULT is asserted. This output is level sensitive.
STAT_RX_SYNCED[19:0]	Output	Word Boundary Synchronized. These signals indicate whether a PCS lane is word boundary synchronized. A value of 1 indicates the corresponding PCS lane has achieved word boundary synchronization and it has received a PCS lane marker. Corresponds to management data input/output (MDIO) register bit 3.52.7:0 and 3.53.11:0 as defined in Clause 82.3. This output is level sensitive.
STAT_RX_SYNCED_ERR[19:0]	Output	Word Boundary Synchronization Error. These signals indicate whether an error occurred during word boundary synchronization in the respective PCS lane. A value of 1 indicates that the corresponding PCS lane lost word boundary synchronization due to sync header framing bits errors or that a PCS lane marker was never received. This output is level sensitive.
STAT_RX_MF_LEN_ERR[19:0]	Output	PCS Lane Marker Length Error. These signals indicate whether a PCS Lane Marker length mismatch occurred in the respective lane (that is, PCS Lane Markers were received not every CTL_RX_VL_LENGTH_MINUS1 words apart). A value of 1 indicates that the corresponding lane is receiving PCS Lane Markers at wrong intervals. This output remains High until the error condition is removed.
STAT_RX_MF_REPEAT_ERR[19:0]	Output	PCS Lane Marker Consecutive Error. These signals indicate whether four consecutive PCS Lane Marker errors occurred in the respective lane. A value of 1 indicates an error in the corresponding lane. This output remains High until the error condition is removed.

Table 2-7: **LBUS Interface – RX Path Control/Status Signals *(Cont'd)***

Name	Direction	Description
STAT_RX_MF_ERR[19:0]	Output	PCS Lane Marker Word Error. These signals indicate that an incorrectly formed PCS Lane Marker Word was detected in the respective lane. A value of 1 indicates an error occurred. This output is pulsed for one clock cycle to indicate the error condition. Pulses can occur in back-to-back cycles.
STAT_RX_ALIGNED	Output	All PCS Lanes Aligned/De-Skewed. This signal indicates whether or not all PCS lanes are aligned and de-skewed. A value of 1 indicates all PCS lanes are aligned and de-skewed. When this signal is a 1, the RX path is aligned and can receive packet data. When this signal is 0, a local fault condition exists. Also corresponds to MDIO register bit 3.50.12 as defined in Clause 82.3. This output is level sensitive.
STAT_RX_STATUS	Output	PCS status. A value of 1 indicates that the PCS is aligned and not in HI_BER state. Corresponds to Management Data Input/Output (MDIO) register bit 3.32.12 as defined in Clause 82.3. This output is level sensitive.
STAT_RX_BLOCK_LOCK[19:0]	Output	Block lock status for each PCS lane. A value of 1 indicates that the corresponding lane has achieved block lock as defined in Clause 82. Corresponds to MDIO register bit 3.50.7:0 and 3.51.11:0 as defined in Clause 82.3. This output is level sensitive.
STAT_RX_ALIGNED_ERR	Output	Loss of Lane Alignment/De-Skew. This signal indicates that an error occurred during PCS lane alignment or PCS lane alignment was lost. A value of 1 indicates an error occurred. This output is level sensitive.
STAT_RX_MISALIGNED	Output	Alignment Error. This signal indicates that the lane aligner did not receive the expected PCS lane marker across all lanes. This signal is not asserted until the PCS lane marker has been received at least once across all lanes and at least one incorrect lane marker has been received. This occurs one metaframe after the error. This signal is not asserted if the lane markers have never been received correctly. Lane marker errors are indicated by the corresponding STAT_RX_MF_ERR signal. This output is pulsed for one clock cycle to indicate an error condition. Pulses can occur in back-to-back cycles.
STAT_RX_REMOTE_FAULT	Output	Remote fault indication status. If this bit is sampled as a 1, it indicates a remote fault condition was detected. If this bit is sampled as a 0, a remote fault condition exist does not exist. This output is level sensitive.

Table 2-7: **LBUS Interface – RX Path Control/Status Signals *(Cont'd)***

Name	Direction	Description
STAT_RX_VL_NUMBER_0[4:0]	Output	The signal STAT_RX_VL_NUMBER_0[4:0] indicates which physical lane is receiving PCS lane 0. There are a total of 20 separate STAT_RX_VL_NUMBER[4:0] buses This bus is only valid when the corresponding bit of be STAT_RX_SYNCED[19:0] is a 1. These outputs are level sensitive.
STAT_RX_VL_NUMBER_1[4:0]	Output	This signal indicates which physical lane is receiving PCS lane 1.
STAT_RX_VL_NUMBER_2[4:0]	Output	This signal indicates which physical lane is receiving PCS lane 2.
STAT_RX_VL_NUMBER_3[4:0]	Output	This signal indicates which physical lane is receiving PCS lane 3.
STAT_RX_VL_NUMBER_4[4:0]	Output	This signal indicates which physical lane is receiving PCS lane 4.
STAT_RX_VL_NUMBER_5[4:0]	Output	This signal indicates which physical lane is receiving PCS lane 5.
STAT_RX_VL_NUMBER_6[4:0]	Output	This signal indicates which physical lane is receiving PCS lane 6.
STAT_RX_VL_NUMBER_7[4:0]	Output	This signal indicates which physical lane is receiving PCS lane 7.
STAT_RX_VL_NUMBER_8[4:0]	Output	This signal indicates which physical lane is receiving PCS lane 8.
STAT_RX_VL_NUMBER_9[4:0]	Output	This signal indicates which physical lane is receiving PCS lane 9.
STAT_RX_VL_NUMBER_10[4:0]	Output	This signal indicates which physical lane is receiving PCS lane 10.
STAT_RX_VL_NUMBER_11[4:0]	Output	This signal indicates which physical lane is receiving PCS lane 11.
STAT_RX_VL_NUMBER_12[4:0]	Output	This signal indicates which physical lane is receiving PCS lane 12.
STAT_RX_VL_NUMBER_13[4:0]	Output	This signal indicates which physical lane is receiving PCS lane 13.
STAT_RX_VL_NUMBER_14[4:0]	Output	This signal indicates which physical lane is receiving PCS lane 14.
STAT_RX_VL_NUMBER_15[4:0]	Output	This signal indicates which physical lane is receiving PCS lane 15.
STAT_RX_VL_NUMBER_16[4:0]	Output	This signal indicates which physical lane is receiving PCS lane 16.
STAT_RX_VL_NUMBER_17[4:0]	Output	This signal indicates which physical lane is receiving PCS lane 17.
STAT_RX_VL_NUMBER_18[4:0]	Output	This signal indicates which physical lane is receiving PCS lane 18.

Table 2-7: **LBUS Interface – RX Path Control/Status Signals *(Cont'd)***

Name	Direction	Description
STAT_RX_VL_NUMBER_19[4:0]	Output	This signal indicates which physical lane is receiving PCS lane 19.
STAT_RX_VL_DEMUXED[19:0]	Output	PCS Lane Marker found. If a signal of this bus is sampled as 1, it indicates that the receiver has properly de-multiplexed that PCS lane. These outputs are level sensitive.
STAT_RX_BAD_FCS[4-1:0]	Output	Bad FCS indicator. A value of 1 indicates a packet was received with a bad FCS, but not a stomped FCS. A stomped FCS is defined as the bitwise inverse of the expected good FCS. This output is pulsed for one clock cycle to indicate an error condition. Pulses can occur in back-to-back cycles.
STAT_RX_STOMPED_FCS[4-1:0]	Output	Stomped FCS indicator. A value of 1 or greater indicates that one or more packets were received with a stomped FCS. A stomped FCS is defined as the bitwise inverse of the expected good FCS. This output is pulsed for one clock cycle to indicate the stomped condition. Pulses can occur in back-to-back cycles.
STAT_RX_TRUNCATED	Output	Packet truncation indicator. A value of 1 indicates that the current packet in flight is truncated due to its length exceeding CTL_RX_MAX_PACKET_LEN[14:0]. This output is pulsed for one clock cycle to indicate the truncated condition. Pulses can occur in back-to-back cycles.
STAT_RX_INTERNAL_LOCAL_FAULT	Output	This signal goes High when an internal local fault is generated due to any one of the following: test pattern generation, bad lane alignment, or high bit error rate. This signal remains High as long as the fault condition persists.
STAT_RX_RECEIVED_LOCAL_FAULT	Output	This signal goes High when enough local fault words are received from the link partner to trigger a fault condition as specified by the IEEE fault state machine. This signal remains High as long as the fault condition persists.
STAT_RX_BIP_ERR_0	Output	BIP8 error indicator for PCS lane 0. A non-zero value indicates the BIP8 signature byte was in error for the corresponding PCS lane. A non-zero value is pulsed for one clock cycle. This output is pulsed for one clock cycle to indicate an error condition. Pulses can occur in back-to-back cycles.
STAT_RX_BIP_ERR_1	Output	BIP8 error indicator for PCS lane 1.
STAT_RX_BIP_ERR_2	Output	BIP8 error indicator for PCS lane 2.

Table 2-7: **LBUS Interface – RX Path Control/Status Signals *(Cont'd)***

Name	Direction	Description
STAT_RX_BIP_ERR_3	Output	BIP8 error indicator for PCS lane 3.
STAT_RX_BIP_ERR_4	Output	BIP8 error indicator for PCS lane 4.
STAT_RX_BIP_ERR_5	Output	BIP8 error indicator for PCS lane 5.
STAT_RX_BIP_ERR_6	Output	BIP8 error indicator for PCS lane 6.
STAT_RX_BIP_ERR_7	Output	BIP8 error indicator for PCS lane 7.
STAT_RX_BIP_ERR_8	Output	BIP8 error indicator for PCS lane 8.
STAT_RX_BIP_ERR_9	Output	BIP8 error indicator for PCS lane 9.
STAT_RX_BIP_ERR_10	Output	BIP8 error indicator for PCS lane 10.
STAT_RX_BIP_ERR_11	Output	BIP8 error indicator for PCS lane 11.
STAT_RX_BIP_ERR_12	Output	BIP8 error indicator for PCS lane 12.
STAT_RX_BIP_ERR_13	Output	BIP8 error indicator for PCS lane 13.
STAT_RX_BIP_ERR_14	Output	BIP8 error indicator for PCS lane 14.
STAT_RX_BIP_ERR_15	Output	BIP8 error indicator for PCS lane 15.
STAT_RX_BIP_ERR_16	Output	BIP8 error indicator for PCS lane 16.
STAT_RX_BIP_ERR_17	Output	BIP8 error indicator for PCS lane 17.
STAT_RX_BIP_ERR_18	Output	BIP8 error indicator for PCS lane 18.
STAT_RX_BIP_ERR_19	Output	BIP8 error indicator for PCS lane 19.
STAT_RX_HI_BER	Output	High Bit Error Rate (BER) indicator. When set to 1, the BER is too high as defined by the 802.3. Corresponds to MDIO register bit 3.32.1 as defined in Clause 82.3. This output is level sensitive.

Table 2-8: **Miscellaneous Status/Control Signals**

Name	Direction	Description
STAT_RX_GOT_SIGNAL_OS	Output	Signal Ordered Sets (OS) indication. If this bit is sampled as a 1, it indicates that a Signal OS word was received. Signal OS should not be received in an Ethernet network.
CTL_RX_TEST_PATTERN	Input	Test pattern checking enable for the RX core. A value of 1 enables test mode as defined in Clause 82.2.18. Corresponds to MDIO register bit 3.42.2 as defined in Clause 82.3. Checks for scrambled idle pattern.
CTL_TX_TEST_PATTERN	Input	Test pattern generation enable for the TX core. A value of 1 enables test mode as defined in Clause 82.2.18. Corresponds to MDIO register bit 3.42.3 as defined in Clause 82.3. Generates a scrambled idle pattern.

Table 2-8: **Miscellaneous Status/Control Signals *(Cont'd)***

Name	Direction	Description
STAT_RX_TEST_PATTERN_MISMATCH[2:0]	Output	Test pattern mismatch increment. A non-zero value in any cycle indicates how many mismatches occurred for the test pattern in the RX core. This output is only active when CTL_RX_TEST_PATTERN is set to a 1. This output can be used to generate MDIO register 3.43.15:0 as defined in Clause 82.3. This output is pulsed for one clock cycle.
CTL_CAUI4_MODE	Input	When this input is High, the dedicated 100G Ethernet IP core operates in CAUI-4 mode and when Low in CAUI-10 mode.
CTL_TX_LANE0_VLM_BIP7_OVERRIDE	Input	When this input is High, the bip7 byte of the PCS lane0 marker is over-ridden by CTL_TX_LANE0_VLM_BIP7_OVERRIDE_VALUE[7:0]
CTL_TX_LANE0_VLM_BIP7_OVERRIDE_ VALUE[7:0]	Input	This input is the override value of the bip7 byte of PCS lane0 marker when CTL_TX_LANE0_VLM_BIP7_OVERRIDE is asserted.
STAT_RX_LANE0_VLM_BIP7[7:0]	Output	This output is the received value of the bip7 byte in the PCS lane0 marker.
STAT_RX_LANE0_VLM_BIP7_VALID	Output	This output, when asserted, indicates that the value of STAT_RX_LANE0_VLM_BIP[7:0] is valid.

Table 2-9: **Statistics Interface – RX Path**

Name	Direction	Description
STAT_RX_TOTAL_BYTES[7:0]	Output	Increment for the total number of bytes received.
STAT_RX_TOTAL_PACKETS[4-1:0]	Output	Increment for the total number of packets received.
STAT_RX_TOTAL_GOOD_BYTES[13:0]	Output	Increment for the total number of good bytes received. This value is only non-zero when a packet is received completely and contains no errors.
STAT_RX_TOTAL_GOOD_PACKETS	Output	Increment for the total number of good packets received. This value is only non-zero when a packet is received completely and contains no errors.

Table 2-9: **Statistics Interface – RX Path *(Cont'd)***

Name	Direction	Description
STAT_RX_PACKET_BAD_FCS	Output	Increment for packets between 64 and ctl_rx_max_packet_len bytes that have FCS errors.
STAT_RX_PACKET_64_BYTES	Output	Increment for good and bad packets received that contain 64 bytes.
STAT_RX_PACKET_65_127_BYTES	Output	Increment for good and bad packets received that contain 65 to 127 bytes.
STAT_RX_PACKET_128_255_BYTES	Output	Increment for good and bad packets received that contain 128 to 255 bytes.
STAT_RX_PACKET_256_511_BYTES	Output	Increment for good and bad packets received that contain 256 to 511 bytes.
STAT_RX_PACKET_512_1023_BYTES	Output	Increment for good and bad packets received that contain 512 to 1,023 bytes.
STAT_RX_PACKET_1024_1518_BYTES	Output	Increment for good and bad packets received that contain 1,024 to 1,518 bytes.
STAT_RX_PACKET_1519_1522_BYTES	Output	Increment for good and bad packets received that contain 1,519 to 1,522 bytes.
STAT_RX_PACKET_1523_1548_BYTES	Output	Increment for good and bad packets received that contain 1,523 to 1,548 bytes.
STAT_RX_PACKET_1549_2047_BYTES	Output	Increment for good and bad packets received that contain 1,549 to 2,047 bytes.
STAT_RX_PACKET_2048_4095_BYTES	Output	Increment for good and bad packets received that contain 2,048 to 4,095 bytes.
STAT_RX_PACKET_4096_8191_BYTES	Output	Increment for good and bad packets received that contain 4,096 to 8,191 bytes.
STAT_RX_PACKET_8192_9215_BYTES	Output	Increment for good and bad packets received that contain 8,192 to 9,215 bytes.
STAT_RX_PACKET_SMALL[4-1:0]	Output	Increment for all packets that are less than 64 bytes long.
STAT_RX_PACKET_LARGE	Output	Increment for all packets that are more than 9,215 bytes long.
STAT_RX_UNICAST	Output	Increment for good unicast packets.
STAT_RX_MULTICAST	Output	Increment for good multicast packets.
STAT_RX_BROADCAST	Output	Increment for good broadcast packets.
STAT_RX_OVERSIZE	Output	Increment for packets longer than CTL_RX_MAX_PACKET_LEN with good FCS.

Table 2-9: **Statistics Interface – RX Path *(Cont'd)***

Name	Direction	Description
STAT_RX_TOOLONG	Output	Increment for packets longer than CTL_RX_MAX_PACKET_LEN with good and bad FCS.
STAT_RX_UNDERSIZE[4-1:0]	Output	Increment for packets shorter than STAT_RX_MIN_PACKET_LEN with good FCS.
STAT_RX_FRAGMENT[4-1:0]	Output	Increment for packets shorter than stat_rx_min_packet_len with bad FCS.
STAT_RX_VLAN	Output	Increment for good 802.1Q tagged VLAN packets.
STAT_RX_INRANGEERR	Output	Increment for packets with Length field error but with good FCS.
STAT_RX_JABBER	Output	Increment for packets longer than CTL_RX_MAX_PACKET_LEN with bad FCS.
STAT_RX_PAUSE	Output	Increment for 802.3x Ethernet MAC Pause packet with good FCS.
STAT_RX_USER_PAUSE	Output	Increment for priority based pause packets with good FCS.
STAT_RX_BAD_CODE[6:0]	Output	Increment for 64B/66B code violations. This signal indicates that the RX PCS receive state machine is in the `RX_E` state as specified by the 802.3 specifications. This output can be used to generate MDIO register 3.33:7:0 as defined in Clause 82.3.
STAT_RX_BAD_SFD	Output	Increment bad SFD. This signal indicates if the Ethernet packet received was preceded by a valid start of frame delimiter (SFD). A value of 1 indicates that an invalid SFD was received.
STAT_RX_BAD_PREAMBLE	Output	Increment bad preamble. This signal indicates if the Ethernet packet received was preceded by a valid preamble. A value of 1 indicates that an invalid preamble was received.

Table 2-10: **Statistics Interface – TX Path**

Name	Direction	Description
STAT_TX_TOTAL_BYTES[6:0]	Output	Increment for the total number of bytes transmitted.
STAT_TX_TOTAL_PACKETS	Output	Increment for the total number of packets transmitted.
STAT_TX_TOTAL_GOOD_BYTES[13:0]	Output	Increment for the total number of good bytes transmitted. This value is only non-zero when a packet is transmitted completely and contains no errors.
STAT_TX_TOTAL_GOOD_PACKETS	Output	Increment for the total number of good packets transmitted.

Table 2-10: **Statistics Interface – TX Path *(Cont'd)***

Name	Direction	Description
STAT_TX_BAD_FCS	Output	Increment for packets greater than 64 bytes that have FCS errors.
STAT_TX_PACKET_64_BYTES	Output	Increment for good and bad packets transmitted that contain 64 bytes.
STAT_TX_PACKET_65_127_BYTES	Output	Increment for good and bad packets transmitted that contain 65 to 127 bytes.
STAT_TX_PACKET_128_255_BYTES	Output	Increment for good and bad packets transmitted that contain 128 to 255 bytes.
STAT_TX_PACKET_256_511_BYTES	Output	Increment for good and bad packets transmitted that contain 256 to 511 bytes.
STAT_TX_PACKET_512_1023_BYTES	Output	Increment for good and bad packets transmitted that contain 512 to 1,023 bytes.
STAT_TX_PACKET_1024_1518_BYTES	Output	Increment for good and bad packets transmitted that contain 1,024 to 1,518 bytes.
STAT_TX_PACKET_1519_1522_BYTES	Output	Increment for good and bad packets transmitted that contain 1,519 to 1,522 bytes.
STAT_TX_PACKET_1523_1548_BYTES	Output	Increment for good and bad packets transmitted that contain 1,523 to 1,548 bytes.
STAT_TX_PACKET_1549_2047_BYTES	Output	Increment for good and bad packets transmitted that contain 1,549 to 2,047 bytes.
STAT_TX_PACKET_2048_4095_BYTES	Output	Increment for good and bad packets transmitted that contain 2,048 to 4,095 bytes.
STAT_TX_PACKET_4096_8191_BYTES	Output	Increment for good and bad packets transmitted that contain 4,096 to 8,191 bytes.
STAT_TX_PACKET_8192_9215_BYTES	Output	Increment for good and bad packets transmitted that contain 8,192 to 9,215 bytes.
STAT_TX_PACKET_SMALL	Output	Increment for all packets that are less than 64 bytes long. Packet transfers of less than 64 bytes are not permitted.
STAT_TX_PACKET_LARGE	Output	Increment for all packets that are more than 9,215 bytes long.
STAT_TX_UNICAST	Output	Increment for good unicast packets.
STAT_TX_MULTICAST	Output	Increment for good multicast packets.
STAT_TX_BROADCAST	Output	Increment for good broadcast packets.
STAT_TX_VLAN	Output	Increment for good 802.1Q tagged VLAN packets.
STAT_TX_PAUSE	Output	Increment for 802.3x Ethernet MAC Pause packet with good FCS.
STAT_TX_USER_PAUSE	Output	Increment for priority based pause packets with good FCS.
STAT_TX_FRAME_ERROR	Output	Increment for packets with tx_errin set to indicate an EOP abort.

Table 2-11: **Pause Interface – Control Signals**

Name	Direction	Description
CTL_RX_PAUSE_ENABLE[8:0]	Input	RX pause enable signal. This input is used to enable the processing of the pause quanta for the corresponding priority. This signal only affects the RX user interface, not the pause processing logic.
CTL_TX_PAUSE_ENABLE[8:0]	Input	TX pause enable signal. This input is used to enable the processing of the pause quanta for the corresponding priority. This signal gates transmission of pause packets.

Table 2-12: **Pause Interface – RX Path**

Name	Direction	Description
CTL_RX_ENABLE_GCP	Input	A value of 1 enables global control packet processing.
CTL_RX_CHECK_MCAST_GCP	Input	A value of 1 enables global control multicast destination address processing.
CTL_RX_CHECK_UCAST_GCP	Input	A value of 1 enables global control unicast destination address processing.
CTL_RX_CHECK_SA_GCP	Input	A value of 1 enables global control source address processing.
CTL_RX_CHECK_ETYPE_GCP	Input	A value of 1 enables global control Ethertype processing.
CTL_RX_CHECK_OPCODE_GCP	Input	A value of 1 enables global control opcode processing.
CTL_RX_ENABLE_PCP	Input	A value of 1 enables priority control packet processing.
CTL_RX_CHECK_MCAST_PCP	Input	A value of 1 enables priority control multicast destination address processing.
CTL_RX_CHECK_UCAST_PCP	Input	A value of 1 enables priority control unicast destination address processing.
CTL_RX_CHECK_SA_PCP	Input	A value of 1 enables priority control source address processing.
CTL_RX_CHECK_ETYPE_PCP	Input	A value of 1 enables priority control Ethertype processing.
CTL_RX_CHECK_OPCODE_PCP	Input	A value of 1 enables priority control opcode processing.
CTL_RX_ENABLE_GPP	Input	A value of 1 enables global pause packet processing.
CTL_RX_CHECK_MCAST_GPP	Input	A value of 1 enables global pause multicast destination address processing.
CTL_RX_CHECK_UCAST_GPP	Input	A value of 1 enables global pause unicast destination address processing.

Table 2-12: **Pause Interface – RX Path *(Cont'd)***

Name	Direction	Description
CTL_RX_CHECK_SA_GPP	Input	A value of 1 enables global pause source address processing.
CTL_RX_CHECK_ETYPE_GPP	Input	A value of 1 enables global pause Ethertype processing.
CTL_RX_CHECK_OPCODE_GPP	Input	A value of 1 enables global pause opcode processing.
CTL_RX_ENABLE_PPP	Input	A value of 1 enables priority pause packet processing.
CTL_RX_CHECK_MCAST_PPP	Input	A value of 1 enables priority pause multicast destination address processing.
CTL_RX_CHECK_UCAST_PPP	Input	A value of 1 enables priority pause unicast destination address processing.
CTL_RX_CHECK_SA_PPP	Input	A value of 1 enables priority pause source address processing.
CTL_RX_CHECK_ETYPE_PPP	Input	A value of 1 enables priority pause Ethertype processing.
CTL_RX_CHECK_OPCODE_PPP	Input	A value of 1 enables priority pause opcode processing.
STAT_RX_PAUSE_REQ[8:0]	Output	Pause request signal. When the RX receives a valid pause frame, it sets the corresponding bit of this bus to a 1 and holds at 1 until the pause packet has been processed. See Pause Processing Interface in Chapter 3.
CTL_RX_PAUSE_ACK[8:0]	Input	Pause acknowledge signal. This bus is used to acknowledge the receipt of the pause frame from the user logic. See Pause Processing Interface in Chapter 3.
STAT_RX_PAUSE_VALID[8:0]	Output	This bus indicates that a pause packet was received and the associated quanta on the STAT_RX_PAUSE_QUANTA[8:0][15:0] bus is valid and must be used for pause processing. If an 802.3x Ethernet MAC Pause packet is received, bit[8] is set to 1.
STAT_RX_PAUSE_QUANTA0[15:0]	Output	This bus indicates the quanta received for priority 0 in priority based pause operation. If an 802.3x Ethernet MAC Pause packet is received, the quanta is placed in STAT_RX_PAUSE_QUANTA8[15:0].
STAT_RX_PAUSE_QUANTA1[15:0]	Output	This bus indicates the quanta received for priority 1 in a priority based pause operation.
STAT_RX_PAUSE_QUANTA2[15:0]	Output	This bus indicates the quanta received for priority 2 in a priority based pause operation.
STAT_RX_PAUSE_QUANTA3[15:0]	Output	This bus indicates the quanta received for priority 3 in a priority based pause operation.
STAT_RX_PAUSE_QUANTA4[15:0]	Output	This bus indicates the quanta received for priority 4 in a priority based pause operation.
STAT_RX_PAUSE_QUANTA5[15:0]	Output	This bus indicates the quanta received for priority 5 in a priority based pause operation.
STAT_RX_PAUSE_QUANTA6[15:0]	Output	This bus indicates the quanta received for priority 6 in a priority based pause operation.

Table 2-12: **Pause Interface – RX Path *(Cont'd)***

Name	Direction	Description
STAT_RX_PAUSE_QUANTA7[15:0]	Output	This bus indicates the quanta received for priority 7 in a priority based pause operation.
STAT_RX_PAUSE_QUANTA8[15:0]	Output	This bus indicates the value of an 802.3x Ethernet MAC Pause packet when received.

Table 2-13: **Pause Interface – TX Path**

Name	Direction	Description
CTL_TX_PAUSE_REQ[8:0]	Input	If a bit of this bus is set to 1, the dedicated 100G Ethernet IP core transmits a pause packet using the associated quanta value on the CTL_TX_PAUSE_QUANTA[8:0][15:0] bus. If bit[8] is set to 1, a global pause packet is transmitted. All other bits cause a priority pause packet to be transmitted. Each bit of this bus must be held at a steady state for a minimum of 16 cycles before the next transition.
CTL_TX_PAUSE_QUANTA0[15:0]	Input	This bus indicates the quanta to be transmitted for priority 0 in a priority based pause operation. If an 802.3x Ethernet MAC Pause packet is to be transmitted, the quanta is placed in CTL_TX_PAUSE_QUANTA8[15:0].
CTL_TX_PAUSE_QUANTA1[15:0]	Input	This bus indicates the quanta to be transmitted for priority 1 in a priority based pause operation.
CTL_TX_PAUSE_QUANTA2[15:0]	Input	This bus indicates the quanta to be transmitted for priority 2 in a priority based pause operation.
CTL_TX_PAUSE_QUANTA3[15:0]	Input	This bus indicates the quanta to be transmitted for priority 3 in a priority based pause operation.
CTL_TX_PAUSE_QUANTA4[15:0]	Input	This bus indicates the quanta to be transmitted for priority 4 in a priority based pause operation.
CTL_TX_PAUSE_QUANTA5[15:0]	Input	This bus indicates the quanta to be transmitted for priority 5 in a priority based pause operation.
CTL_TX_PAUSE_QUANTA6[15:0]	Input	This bus indicates the quanta to be transmitted for priority 6 in a priority based pause operation.
CTL_TX_PAUSE_QUANTA7[15:0]	Input	This bus indicates the quanta to be transmitted for priority 7 in a priority based pause operation.
CTL_TX_PAUSE_QUANTA8[15:0]	Input	This bus indicates the value of an 802.3x MAC Pause packet to be transmitted.
CTL_TX_PAUSE_REFRESH_TIMER0[15:0]	Input	This bus sets the retransmission time of pause packets for priority 0 in a priority based pause operation.
CTL_TX_PAUSE_REFRESH_TIMER1[15:0]	Input	This bus sets the retransmission time of pause packets for priority 1 in a priority based pause operation.

Table 2-13: **Pause Interface – TX Path *(Cont'd)***

Name	Direction	Description
CTL_TX_PAUSE_REFRESH_TIMER2[15:0]	Input	This bus sets the retransmission time of pause packets for priority 2 in a priority based pause operation.
CTL_TX_PAUSE_REFRESH_TIMER3[15:0]	Input	This bus sets the retransmission time of pause packets for priority 3 in a priority based pause operation.
CTL_TX_PAUSE_REFRESH_TIMER4[15:0]	Input	This bus sets the retransmission time of pause packets for priority 4 in a priority based pause operation.
CTL_TX_PAUSE_REFRESH_TIMER5[15:0]	Input	This bus sets the retransmission time of pause packets for priority 5 in a priority based pause operation.
CTL_TX_PAUSE_REFRESH_TIMER6[15:0]	Input	This bus sets the retransmission time of pause packets for priority 6 in a priority based pause operation.
CTL_TX_PAUSE_REFRESH_TIMER7[15:0]	Input	This bus sets the retransmission time of pause packets for priority 7 in a priority based pause operation.
CTL_TX_PAUSE_REFRESH_TIMER8[15:0]	Input	This bus sets the retransmission time of pause packets for a global pause operation.
CTL_TX_RESEND_PAUSE	Input	Re-transmit pending pause packets. When this input is sampled as 1, all pending pause packets are retransmitted as soon as possible (that is, after the current packet in flight is completed) and the retransmit counters are reset. This input should be pulsed to 1 for one cycle at a time.
STAT_TX_PAUSE_VALID[8:0]	Output	If a bit of this bus is set to 1, the dedicated 100G Ethernet IP core has transmitted a pause packet. If bit[8] is set to 1, a global pause packet is transmitted. All other bits cause a priority pause packet to be transmitted.

Table 2-14: **IEEE 1588 Interface – TX Path**

Name	Direction	Description
CTL_TX_SYSTEMTIMERIN[80-1:0]	Input	System timer input for the TX. In normal clock mode, the time format is according to the IEEE 1588 format, with 48 bits for seconds and 32 bits for nanoseconds. In transparent clock mode, bit 63 is expected to be zero, bits 62:16 carry nanoseconds, and bits 15:0 carry fractional nanoseconds. Refer to IEEE 1588v2 for the representational definitions. This input must be in the TX clock domain.
TX_PTP_TSTAMP_VALID_OUT	Output	This bit indicates that a valid timestamp is being presented on the TX.
TX_PTP_PCSLANE_OUT[5-1:0]	Output	This bus identifies which of the 20 PCS lanes that the SOP was detected on for the corresponding timestamp.

Table 2-14: **IEEE 1588 Interface – TX Path *(Cont'd)***

Name	Direction	Description
TX_PTP_TSTAMP_TAG_OUT[15:0]	Output	Tag output corresponding to TX_PTP_TAG_FIELD_IN[15:0].
TX_PTP_TSTAMP_OUT[79:0]	Output	Time stamp for the transmitted packet SOP corresponding to the time at which it passed the capture plane. The representation of the bits contained in this bus is the same as the timer input.
TX_PTP_1588OP_IN[1:0]	Input	• 2'b00 – "No operation": no timestamp will be taken and the frame will not be modified. • 2'b01 – "1-step": a timestamp should be taken and inserted into the frame. • 2'b10 – "2-step": a timestamp should be taken and returned to the client using the additional ports of 2-step operation. The frame itself will not be modified. • 2'b11 – Reserved: act as "No operation".
TX_PTP_TAG_FIELD_IN[15:0]	Input	The usage of this field is dependent on the 1588 operation • For "No operation", this field will be ignored. • For "1-step" and "2-step", this field is a tag field. This tag value will be returned to the client with the timestamp for the current frame using the additional ports of 2-step operation. This tag value can be used by software to ensure that the timestamp can be matched with the precise timing protocol (PTP) frame that it sent for transmission.
TX_PTP_UPD_CHKSUM_IN	Input	The usage of this field is dependent on the 1588 operation • For "No operation" or "2-step", this bit will be ignored. • For "1-step": ◦ 1'b0: The PTP frame does not contain a UDP checksum. ◦ 1'b1: The PTP frame does contain a UDP checksum which the core is required to recalculate.
TX_PTP_CHKSUM_OFFSET_IN[15:0]	Input	The usage of this field is dependent on the "1588 operation" and on the "Update Checksum" bit. • For "No operation", for "2-step" or for "1-step" when "Update Checksum" is set to 1'b0, this field will be ignored. • For "1-step" when "Update Checksum" is set to 1'b1, this field is a numeric value indicating the number of bytes into the frame to where the first byte of the checksum is located (where a value of 0 represents the first byte of the Destination Address, etc). ***Note:*** The IPv6 header size is unbounded, so this field is able to cope with all frames sizes up to 16K jumbo frames. Only even values are supported.

Table 2-14: **IEEE 1588 Interface – TX Path *(Cont'd)***

Name	Direction	Description
TX_PTP_TSTAMP_OFFSET_IN[15:0]	Input	The usage of this field is dependent on the 1588 operation • For "No operation" or "2-step" this field will be ignored. • For "1-step", this field is a numeric value indicating the number of bytes into the frame to where the first byte of the timestamp to be inserted is located (where a value of 0 represents the first byte of the Destination Address, etc). This input is also used to specify the offset for the correction field in 1-step Transparent Clock mode. ***Note:*** The IPv6 header size is unbounded, so this field is able to cope with all frames sizes up to 16K jumbo frames. ***Note:*** Only even values are supported. ***Note:*** In transparent clock mode and when tx_ptp_upd_chksum_in=1, this value cannot be greater than tx_ptp_chksum_offset_in + 34 (decimal).
CTL_TX_PTP_VLANE_ADJUST_MODE	Input	When asserted, this signal applies an adjustment to the TX timestamps according to the PCS lane on which the SOP occurs. When zero, no adjustment is made. This signal only has effect for 1-step operation.
TX_PTP_RXTSTAMP_IN[63:0]	Input	Reserved
STAT_TX_PTP_FIFO_WRITE_ERROR	Output	Transmit PTP FIFO write error. A 1 on this status indicates that an error occurred during the PTP Tag write. A TX Path reset is required to clear the error.
STAT_TX_PTP_FIFO_READ_ERROR	Output	Transmit PTP FIFO read error. A 1 on this status indicates that an error occurred during the PTP Tag read. A TX Path reset is required to clear the error.

Table 2-15: **IEEE 1588 Interface – RX Path**

Name	Direction	Description
CTL_RX_SYSTEMTIMERIN[80-1:0]	Input	System timer input for the RX. In normal clock mode, the time format is according to the IEEE 1588 format, with 48 bits for seconds and 32 bits for nanoseconds. In transparent clock mode, bit 63 is expected to be zero, bits 62:16 carry nanoseconds, and bits 15:0 carry fractional nanoseconds. Refer to IEEE 1588v2 for the representational definitions. This input must be in the same clock domain as the lane 0 RX SerDes.
RX_PTP_TSTAMP_OUT[79:0]	Output	Time stamp for the received packet SOP corresponding to the time at which it passed the capture plane. This signal will be valid starting at the same clock cycle during which the SOP is asserted for one of the LBUS segments. The representation of the bits contained in this bus is the same as the timer input.
RX_PTP_PCSLANE_OUT[5-1:0]	Output	This bus identifies which of the 20 PCS lanes that the SOP was detected on for the corresponding timestamp. This signal will be valid starting at the same clock cycle during which the SOP is asserted for one of the LBUS segments.
RX_LANE_ALIGNER_FILL_0[7-1:0]	Output	This output indicates the fill level of the alignment buffer for PCS lane0. This information can be used by the PTP application, together with the signal RX_PTP_PCSLANE_OUT[4:0], to adjust for the lane skew of the arriving SOP. The units are SerDes clock cycles.
RX_LANE_ALIGNER_FILL_1[7-1:0]	Output	This output indicates the fill level of the alignment buffer for PCS lane1.
RX_LANE_ALIGNER_FILL_2[7-1:0]	Output	This output indicates the fill level of the alignment buffer for PCS lane2.
RX_LANE_ALIGNER_FILL_3[7-1:0]	Output	This output indicates the fill level of the alignment buffer for PCS lane3.
RX_LANE_ALIGNER_FILL_4[7-1:0]	Output	This output indicates the fill level of the alignment buffer for PCS lane4.
RX_LANE_ALIGNER_FILL_5[7-1:0]	Output	This output indicates the fill level of the alignment buffer for PCS lane5.
RX_LANE_ALIGNER_FILL_6[7-1:0]	Output	This output indicates the fill level of the alignment buffer for PCS lane6.
RX_LANE_ALIGNER_FILL_7[7-1:0]	Output	This output indicates the fill level of the alignment buffer for PCS lane7.
RX_LANE_ALIGNER_FILL_8[7-1:0]	Output	This output indicates the fill level of the alignment buffer for PCS lane8.

Table 2-15: **IEEE 1588 Interface – RX Path *(Cont'd)***

Name	Direction	Description
RX_LANE_ALIGNER_FILL_9[7-1:0]	Output	This output indicates the fill level of the alignment buffer for PCS lane9.
RX_LANE_ALIGNER_FILL_10[7-1:0]	Output	This output indicates the fill level of the alignment buffer for PCS lane10.
RX_LANE_ALIGNER_FILL_11[7-1:0]	Output	This output indicates the fill level of the alignment buffer for PCS lane11.
RX_LANE_ALIGNER_FILL_12[7-1:0]	Output	This output indicates the fill level of the alignment buffer for PCS lane12.
RX_LANE_ALIGNER_FILL_13[7-1:0]	Output	This output indicates the fill level of the alignment buffer for PCS lane13.
RX_LANE_ALIGNER_FILL_14[7-1:0]	Output	This output indicates the fill level of the alignment buffer for PCS lane14.
RX_LANE_ALIGNER_FILL_15[7-1:0]	Output	This output indicates the fill level of the alignment buffer for PCS lane15.
RX_LANE_ALIGNER_FILL_16[7-1:0]	Output	This output indicates the fill level of the alignment buffer for PCS lane16.
RX_LANE_ALIGNER_FILL_17[7-1:0]	Output	This output indicates the fill level of the alignment buffer for PCS lane17.
RX_LANE_ALIGNER_FILL_18[7-1:0]	Output	This output indicates the fill level of the alignment buffer for PCS lane18.
RX_LANE_ALIGNER_FILL_19[7-1:0]	Output	This output indicates the fill level of the alignment buffer for PCS lane19.

Table 2-16: **DRP Path/Control Signals**

Name	Direction	Description
DRP_DO[15:0]	Output	Data bus for reading configuration data from the 100G Ethernet IP core to the FPGA logic resources.
DRP_RDY	Output	Indicates operation is complete for write operations and data is valid for read operations.
DRP_ADDR[9:0]	Input	DRP address bus.
DRP_CLK	Input	DRP interface clock. When DRP is not used, this can be tied to GND.
DRP_DI[15:0]	Input	Data bus for writing configuration data from the FPGA logic resources to the 100G Ethernet IP core.

Table 2-16: **DRP Path/Control Signals *(Cont'd)***

Name	Direction	Description
DRP_EN	Input	DRP enable signal. 0: No read or write operations performed. 1: Enables a read or write operation. For write operations, DRP_WE and DRP_EN should be driven High for one DRP_CLK cycle only.
DRP_WE	Input	DRP write enable. 0: Read operation when DRP_EN is 1. 1: Write operation when DRP_EN is 1. For write operations, DRP_WE and DRP_EN should be driven High for one DRP_CLK cycle only.

Attribute Descriptions

Table 2-17 provides detailed descriptions of the 100G Ethernet IP core attributes and their default values.

Table 2-17: **UltraScale Device 100G Ethernet IP Core Attributes**

Name	Type	Description	Default Value
LBUS Interface – TX Path Control/Status			
CTL_TX_FCS_INS_ENABLE	Boolean	Enable FCS insertion by the TX core. • TRUE: 100G Ethernet IP core calculates and adds FCS to the packet. • FALSE: 100G Ethernet IP core does not add FCS to packet. This attribute cannot be changed dynamically between packets.	TRUE

Table 2-17: **UltraScale Device 100G Ethernet IP Core Attributes *(Cont'd)***

Name	Type	Description	Default Value
CTL_TX_IGNORE_FCS	Boolean	Enable FCS error checking at the LBUS interface by the TX core. This input only has effect when ctl_tx_fcs_ins_enable is FALSE. • TRUE: A packet with bad FCS transmitted is binned as good. • FALSE: A packet with bad FCS transmitted is not binned as good. The error is flagged on the signals stat_tx_bad_fcs and STAT_RX_STOMPED_FCS, and the packet is transmitted as it was received. Statistics are reported as if there was no FCS error.	FALSE
CTL_TX_VL_LENGTH_MINUS1[15:0]	16-bit Hex	Number of words in between PCS Lane markers minus one. Default value, as defined in IEEE 802.3, should be set to 16,383 (decimal).	16'h3FFF
CTL_TX_VL_MARKER_ID0[63:0]	64-bit Hex	This bus sets the TX PCS Lane marker for PCS lane 0. For IEEE 802.3 default values, see the specification.	64'hc16821003e97de00
CTL_TX_VL_MARKER_ID1[63:0]	64-bit Hex	This bus sets the TX PCS Lane marker for PCS lane 1.	64'h9d718e00628e7100
CTL_TX_VL_MARKER_ID2[63:0]	64-bit Hex	This bus sets the TX PCS Lane marker for PCS lane 2.	64'h594be800a6b41700
CTL_TX_VL_MARKER_ID3[63:0]	64-bit Hex	This bus sets the TX PCS Lane marker for PCS lane 3.	64'h4d957b00b26a8400
CTL_TX_VL_MARKER_ID4[63:0]	64-bit Hex	This bus sets the TX PCS Lane marker for PCS lane 4.	64'hf50709000af8f600
CTL_TX_VL_MARKER_ID5[63:0]	64-bit Hex	This bus sets the TX PCS Lane marker for PCS lane 5.	64'hdd14c20022eb3d00
CTL_TX_VL_MARKER_ID6[63:0]	64-bit Hex	This bus sets the TX PCS Lane marker for PCS lane 6.	64'h9a4a260065b5d900
CTL_TX_VL_MARKER_ID7[63:0]	64-bit Hex	This bus sets the TX PCS Lane marker for PCS lane 7.	64'h7b45660084ba9900
CTL_TX_VL_MARKER_ID8[63:0]	64-bit Hex	This bus sets the TX PCS Lane marker for PCS lane 8.	64'ha02476005fdb8900
CTL_TX_VL_MARKER_ID9[63:0]	64-bit Hex	This bus sets the TX PCS Lane marker for PCS lane 9.	64'h68c9fb0097360400
CTL_TX_VL_MARKER_ID10[63:0]	64-bit Hex	This bus sets the TX PCS Lane marker for PCS lane 10.	64'hfd6c990002936600

Table 2-17: **UltraScale Device 100G Ethernet IP Core Attributes *(Cont'd)***

Name	Type	Description	Default Value
CTL_TX_VL_MARKER_ID11[63:0]	64-bit Hex	This bus sets the TX PCS Lane marker for PCS lane 11.	64'hb9915500466eaa00
CTL_TX_VL_MARKER_ID12[63:0]	64-bit Hex	This bus sets the TX PCS Lane marker for PCS lane 12.	64'h5cb9b200a3464d00
CTL_TX_VL_MARKER_ID13[63:0]	64-bit Hex	This bus sets the TX PCS Lane marker for PCS lane 13.	64'h1af8bd00e5074200
CTL_TX_VL_MARKER_ID14[63:0]	64-bit Hex	This bus sets the TX PCS Lane marker for PCS lane 14.	64'h83c7ca007c383500
CTL_TX_VL_MARKER_ID15[63:0]	64-bit Hex	This bus sets the TX PCS Lane marker for PCS lane 15.	64'h3536cd00cac93200
CTL_TX_VL_MARKER_ID16[63:0]	64-bit Hex	This bus sets the TX PCS Lane marker for PCS lane 16.	64'hc4314c003bceb300
CTL_TX_VL_MARKER_ID17[63:0]	64-bit Hex	This bus sets the TX PCS Lane marker for PCS lane 17.	64'hadd6b70052294800
CTL_TX_VL_MARKER_ID18[63:0]	64-bit Hex	This bus sets the TX PCS Lane marker for PCS lane 18.	64'h5f662a00a099d500
CTL_TX_VL_MARKER_ID19[63:0]	64-bit Hex	This bus sets the TX PCS Lane marker for PCS lane 19.	64'hc0f0e5003f0f1a00
LBUS Interface – RX Path Control/Status Signals			
CTL_RX_CHECK_PREAMBLE	Boolean	When set to TRUE, this attribute causes the Ethernet MAC to check the preamble of the received frame.	FALSE
CTL_RX_CHECK_SFD	Boolean	When set to TRUE, this attribute causes the Ethernet MAC to check the Start of Frame Delimiter of the received frame.	FALSE
CTL_RX_DELETE_FCS	Boolean	Enable FCS removal by the RX core. • TRUE: 100G Ethernet IP core deletes the FCS of the incoming packet. • FALSE: 100G Ethernet IP core does not remove the FCS of the incoming packet. FCS is not deleted for packets that are less than or equal to 8 bytes long.	TRUE

Table 2-17: **UltraScale Device 100G Ethernet IP Core Attributes *(Cont'd)***

Name	Type	Description	Default Value
CTL_RX_IGNORE_FCS	Boolean	Enable FCS error checking at the LBUS interface by the RX core. • TRUE: 100G Ethernet IP core does not flag an FCS error at the LBUS interface. • FALSE: A packet received with an FCS error is sent with the RX_ERROUT pin asserted during the last transfer (RX_EOPOUT and RX_ENAOUT are sampled as 1). ***Note:*** The statistics are reported as if the packet is good. The signal stat_rx_bad_fcs, however, reports the error.	FALSE
CTL_RX_MAX_PACKET_LEN[14:0]	15-bit Hex	Any packet longer than this value is considered to be oversized. If a packet has a size greater than this value, the packet is truncated to this value and the RX_ERROUT signal is asserted along with the rx_eopout signal. ctl_rx_max_packet_len[14] is reserved and must be set to 0. Packets less than 64 bytes are dropped. The allowed value for this bus can range from 64 to 16,383.	15'h2580
CTL_RX_MIN_PACKET_LEN[7:0]	8-bit Hex	Any packet shorter than the default value of 64 (decimal) is considered to be undersized. If a packet has a size less than this value, the rx_errout signal is asserted during the rx_eopout asserted cycle. Packets less than 64 bytes are dropped. The value of this bus must be less than or equal to the value of CTL_RX_MAX_PACKET_LEN[14:0].	8'h40
CTL_RX_VL_LENGTH_MINUS1[15:0]	16-bit Hex	Number of words in between PCS Lane markers minus one. Default value, as defined in IEEE 802.3, should be set to 16,383 (decimal).	16'h3FFF
CTL_RX_VL_MARKER_ID0[63:0]	64-bit Hex	This bus sets the RX PCS Lane marker for PCS lane 0. For IEEE 802.3 default values, see the specification.	64'hc16821003e97de00
CTL_RX_VL_MARKER_ID1[63:0]	64-bit Hex	This bus sets the RX PCS Lane marker for PCS lane 1.	64'h9d718e00628e7100
CTL_RX_VL_MARKER_ID2[63:0]	64-bit Hex	This bus sets the RX PCS Lane marker for PCS lane 2.	64'h594be800a6b41700

Table 2-17: **UltraScale Device 100G Ethernet IP Core Attributes *(Cont'd)***

Name	Type	Description	Default Value
CTL_RX_VL_MARKER_ID3[63:0]	64-bit Hex	This bus sets the RX PCS Lane marker for PCS lane 3.	64'h4d957b00b26a8400
CTL_RX_VL_MARKER_ID4[63:0]	64-bit Hex	This bus sets the RX PCS Lane marker for PCS lane 4.	64'hf50709000af8f600
CTL_RX_VL_MARKER_ID5[63:0]	64-bit Hex	This bus sets the RX PCS Lane marker for PCS lane 5.	64'hdd14c20022eb3d00
CTL_RX_VL_MARKER_ID6[63:0]	64-bit Hex	This bus sets the RX PCS Lane marker for PCS lane 6.	64'h9a4a260065b5d900
CTL_RX_VL_MARKER_ID7[63:0]	64-bit Hex	This bus sets the RX PCS Lane marker for PCS lane 7.	64'h7b45660084ba9900
CTL_RX_VL_MARKER_ID8[63:0]	64-bit Hex	This bus sets the RX PCS Lane marker for PCS lane 8.	64'ha02476005fdb8900
CTL_RX_VL_MARKER_ID9[63:0]	64-bit Hex	This bus sets the RX PCS Lane marker for PCS lane 9.	64'h68c9fb0097360400
CTL_RX_VL_MARKER_ID10[63:0]	64-bit Hex	This bus sets the RX PCS Lane marker for PCS lane 10.	64'hfd6c990002936600
CTL_RX_VL_MARKER_ID11[63:0]	64-bit Hex	This bus sets the RX PCS Lane marker for PCS lane 11.	64'hb9915500466eaa00
CTL_RX_VL_MARKER_ID12[63:0]	64-bit Hex	This bus sets the RX PCS Lane marker for PCS lane 12.	64'h5cb9b200a3464d00
CTL_RX_VL_MARKER_ID13[63:0]	64-bit Hex	This bus sets the RX PCS Lane marker for PCS lane 13.	64'h1af8bd00e5074200
CTL_RX_VL_MARKER_ID14[63:0]	64-bit Hex	This bus sets the RX PCS Lane marker for PCS lane 14.	64'h83c7ca007c383500
CTL_RX_VL_MARKER_ID15[63:0]	64-bit Hex	This bus sets the RX PCS Lane marker for PCS lane 15.	64'h3536cd00cac93200
CTL_RX_VL_MARKER_ID16[63:0]	64-bit Hex	This bus sets the RX PCS Lane marker for PCS lane 16.	64'hc4314c003bceb300
CTL_RX_VL_MARKER_ID17[63:0]	64-bit Hex	This bus sets the RX PCS Lane marker for PCS lane 17.	64'hadd6b70052294800
CTL_RX_VL_MARKER_ID18[63:0]	64-bit Hex	This bus sets the RX PCS Lane marker for PCS lane 18.	64'h5f662a00a099d500
CTL_RX_VL_MARKER_ID19[63:0]	64-bit Hex	This bus sets the RX PCS Lane marker for PCS lane 19.	64'hc0f0e5003f0f1a00

Table 2-17: **UltraScale Device 100G Ethernet IP Core Attributes *(Cont'd)***

Name	Type	Description	Default Value
Miscellaneous Status/Control			
CTL_RX_PROCESS_LFI	Boolean	TRUE: The 100G Ethernet IP core RX core will expect and process LF control codes coming in from the SERDES. FALSE: The 100G Ethernet IP core RX core ignores Local Fault Indication (LFI) control codes coming in from the SERDES. ***Note:*** If an LFI condition is detected, the core will stop receiving packets until the LFI is cleared. Packets in progress will be terminated and an error will be indicated on the LBUS. A START block must be received before packets are received again.	FALSE
Pause Interface – RX Path			
CTL_RX_PAUSE_DA_UCAST[47:0]	48-bit Hex	Unicast destination address for pause processing.	48'h000000000000
CTL_RX_PAUSE_SA[47:0]	48-bit Hex	Source address for pause processing.	48'h000000000000
CTL_RX_OPCODE_MIN_GCP[15:0]	16-bit Hex	Minimum global control opcode value.	16'h0000
CTL_RX_OPCODE_MAX_GCP[15:0]	16-bit Hex	Maximum global control opcode value.	16'hffff
CTL_RX_ETYPE_GCP[15:0]	16-bit Hex	Ethertype field for global control processing.	16'h8808
CTL_RX_PAUSE_DA_MCAST[47:0]	48-bit Hex	Multicast destination address for pause processing.	48'h0180c2000001
CTL_RX_ETYPE_PCP[15:0]	16-bit Hex	Ethertype field for priority control processing.	16'h8808
CTL_RX_OPCODE_MIN_PCP[15:0]	16-bit Hex	Minimum priority control opcode value.	16'h0000
CTL_RX_OPCODE_MAX_PCP[15:0]	16-bit Hex	Maximum priority control opcode value.	16'hffff
CTL_RX_ETYPE_GPP[15:0]	16-bit Hex	Ethertype field for global pause processing.	16'h8808
CTL_RX_OPCODE_GPP[15:0]	16-bit Hex	Global pause opcode value.	16'h0001
CTL_RX_ETYPE_PPP[15:0]	16-bit Hex	Ethertype field for priority pause processing.	16'h8808
CTL_RX_OPCODE_PPP[15:0]	16-bit Hex	Priority pause opcode value.	16'h0001

Table 2-17: **UltraScale Device 100G Ethernet IP Core Attributes *(Cont'd)***

Name	Type	Description	Default Value
CTL_RX_CHECK_ACK	Boolean	Wait for acknowledge. • TRUE: 100G Ethernet IP core uses the CTL_RX_PAUSE_ACK[8:0] bus for pause processing. • FALSE: CTL_RX_PAUSE_ACK[8:0] is not used.	TRUE
CTL_RX_FORWARD_CONTROL	Boolean	TRUE: 100G Ethernet IP core will forward control packets. FALSE: 100G Ethernet IP core will drop control packets. See Pause Processing Interface in Chapter 3.	FALSE
Pause Interface – TX Path			
CTL_TX_DA_GPP[47:0]	48-bit Hex	Destination address for transmitting global pause packets.	48'h0180c2000001
CTL_TX_SA_GPP[47:0]	48-bit Hex	Source address for transmitting global pause packets.	48'h000000000000
CTL_TX_ETHERTYPE_GPP[15:0]	16-bit Hex	Ethertype for transmitting global pause packets.	16'h8808
CTL_TX_OPCODE_GPP[15:0]	16-bit Hex	Opcode for transmitting global pause packets.	16'h0001
CTL_TX_DA_PPP[47:0]	48-bit Hex	Destination address for transmitting priority pause packets.	48'h0180c2000001
CTL_TX_SA_PPP[47:0]	48-bit Hex	Source address for transmitting priority pause packets.	48'h000000000000
CTL_TX_ETHERTYPE_PPP[15:0]	16-bit Hex	Ethertype for transmitting priority pause packets.	16'h8808
CTL_TX_OPCODE_PPP[15:0]	16-bit Hex	Opcode for transmitting priority pause packets.	16'h0001

Table 2-17: **UltraScale Device 100G Ethernet IP Core Attributes *(Cont'd)***

Name	Type	Description	Default Value
IEEE 1588 Interface – TX Path			
CTL_TX_PTP_1STEP_ENABLE	Boolean	TRUE: Enable 1-step operation. FALSE: Disable 1-step operation.	FALSE
CTL_PTP_TRANSPCLK_MODE	Boolean	This attribute, when set to TRUE, places the timestamping logic into transparent clock mode and enables correction field updates on the TX. In transparent clock mode, the system timer input is interpreted as the correction value. In this mode, the sign bit is assumed to be 0 (positive time). The TX will sample a TX timestamp for the PTP packet and add it to its correction field value. It is expected that the incoming PTP packet's original correction field has already been subtracted with its RX timestamp by the user before being transferred to the CMAC TX LBUS interface. ***Note:*** Both RX and TX timer inputs are expected to be in correction field format as well as timestamps.	FALSE
CTL_TX_PTP_LATENCY_ADJUST[10:0]	11'bit Hex	This bus can be used to adjust the 1-step TX timestamp with respect to the 2-step timestamp. The units of the bus bits [10:3] are nanoseconds. The 3 LSB bits in this input are fractional nanoseconds. In normal mode, the usual value is 705 decimal (2C1 hex), corresponding to the delay between the 1-step logic and the 2-step timestamp capture plane. In transparent clock mode, the value of 802 decimal (322 hex) is recommended.	11'h2C1
Testing Attributes			
CTL_TEST_MODE_PIN_CHAR	Boolean	Reserved. Set to FALSE.	FALSE
TEST_MODE_PIN_CHAR	Boolean	Reserved. Set to FALSE.	FALSE

Designing with the Core

This chapter includes guidelines and additional information to facilitate designing with the core.

Clocking

The UltraScale™ architecture integrated CMAC block has up to 13 clock inputs for the CAUI-10 interface and up to seven clock inputs for the CAUI-4 interface. These clocks include the `RX_SERDES_CLK[9:0]` and `RX_SERDES_CLK[3:0]` respectively for the CAUI-10 and CAUI-4 modes, `TX_CLK`, `RX_CLK` and the `DRP_CLK`. The `DRP_CLK` is optional and is necessary only during a DRP operation.

The 10 CAUI-10 or 4 CAUI-4 `RX_SERDES_CLK` clocks must not have an FPGA induced dynamic skew of more than 400 ps. More details on the clocks are provided in the following sections.

The Runtime Switchable mode follows the same clocking structure as the one from CAUI-10 described previously.

RX GT/Lane Logic Clock (RX_SERDES_CLK)

These clocks are provided to the CMAC block from the serial transceiver (GT) to clock the Lane Logic RX interface. The clocks must be 322.266 MHz for both CAUI-10 and CAUI-4 operation. The GT interface datapath is 32 bits per lane for CAUI-10 and 80 bits per lane for CAUI-4.

The other implementation allows only one `RX_SERDES_CLK` to go to the Ethernet MAC `RX_SERDES_CLK` inputs. The serial transceiver will also be in raw mode but this time the buffer is used. This mode is used when you can tolerate higher latency and are interested in saving FPGA clocking resources.

TX CLK

This clock is provided to both the CMAC block and serial transceiver to clock the GT/ lane logic TX interface as well as the whole Ethernet MAC. The clock must be 322.266 MHz for both CAUI-10 and CAUI-4 operation. The GT lane logic interface datapath is 32 bits per lane for CAUI-10 and 80 bits per lane for CAUI-4. Only one `TX_CLK` is needed regardless of the CAUI-10 or CAUI-4 implementation. This clock also clocks the transmit Ethernet MAC, LBUS, interface and the Control/Status port.

RX CLK

This clock is provided to the CMAC block. The clock must be 322.266 MHz for both CAUI-10 and CAUI-4 operation, and must be the same as `TX_CLK`. This clock is used in the receive Ethernet MAC, LBUS interface, and the Control/Status port.

DRP Clock (drp_clk)

This signal clocks the DRP port. Any convenient frequency can be chosen, up to 250 MHz.

Resets

The 100G Ethernet IP core has a total of 12 resets. They are `TX_RESET`, `RX_RESET`, `RX_SERDES_RESET[9:0]`. During configuration `TX_RESET`, `RX_RESET`, and `RX_SERDES_RESET[9:0]` need to be asserted High and after the clocks are stable, the resets are released. During normal operation the RX and TX paths can be asserted independently. Within the RX and TX logic paths, there are separate resets to the core and the lane logic. The reset procedure is simple and the only requirement is that a reset must be asserted until the corresponding clock(s) are stable. The 100G Ethernet IP core takes care of ensuring that the different resets are properly synchronized to the required domain. It is up to you to ensure a reset is held until the corresponding clock is fully stable.

Note: Some of the control inputs to the 100G Ethernet IP core can only be modified while the core is held in reset. If one of these inputs needs changing, the appropriate RX or TX LBUS reset input (`RX_RESET` or `TX_RESET`) must be asserted until the control input is stabilized. All resets within the block are asynchronously asserted, and synchronously deasserted. Standard cell synchronizers are used, where applicable per guidelines to synchronize assertion and release of resets to respective clock inputs.

See Figure 3-1, Figure 3-2, Figure 3-3, and Figure 3-4 for diagrams of the clocking and resets. Any of the four modes are available based on the Vivado IDE selection and configuration.

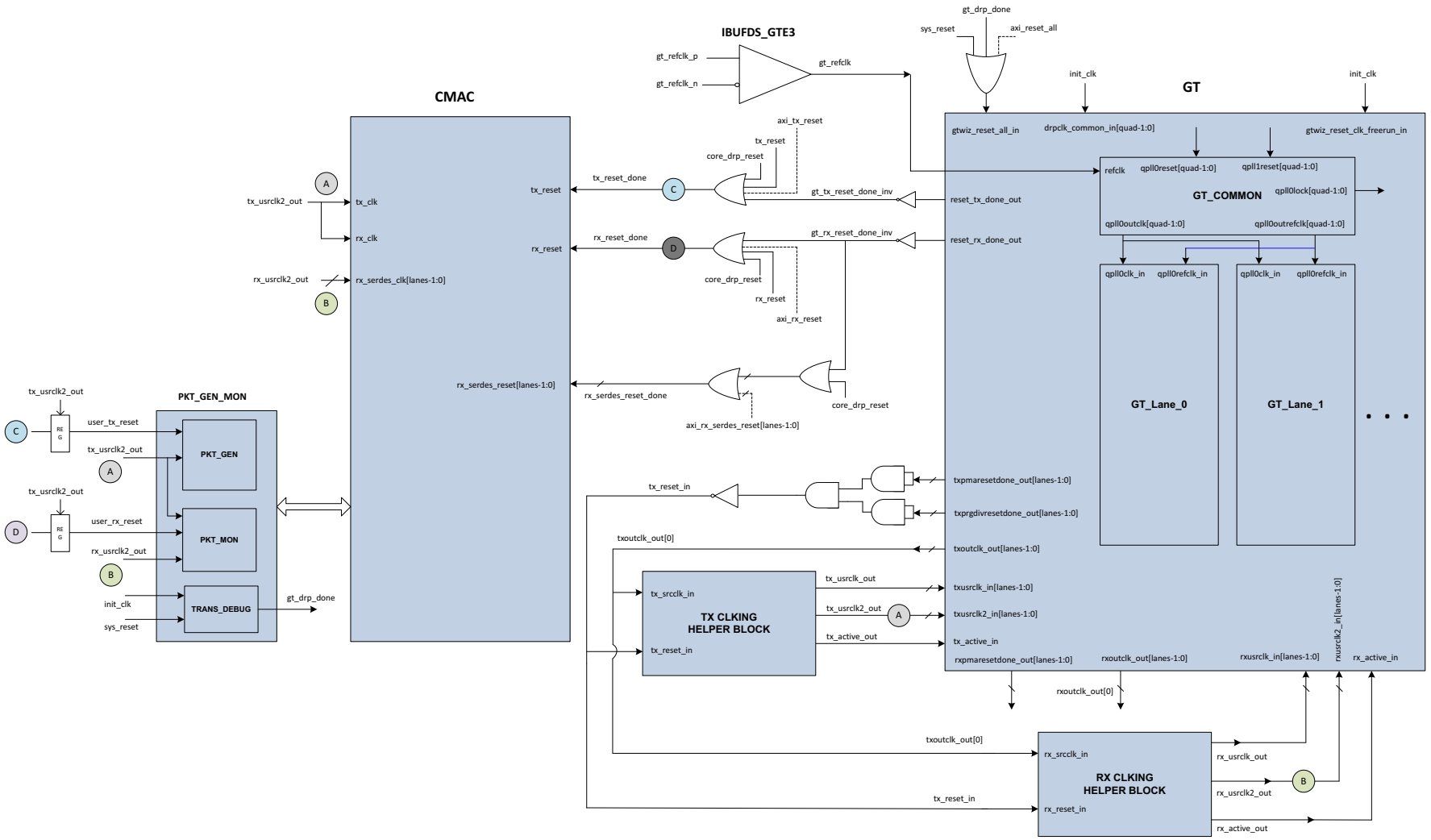

Figure 3-1: **CMAC Clocking and Reset - Sync Clock Mode - MultiLane**

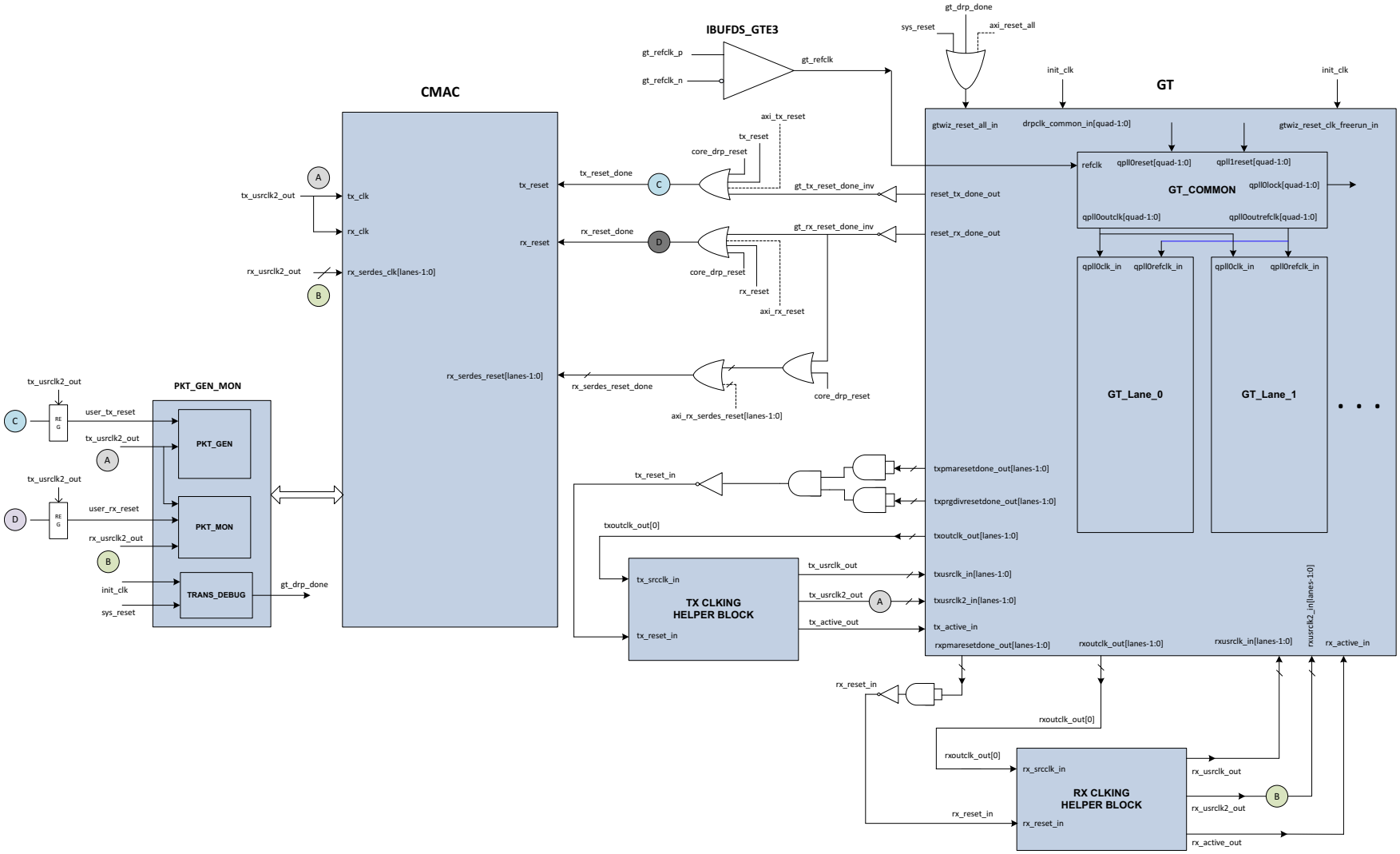

Figure 3-2: **CMAC Clocking and Reset - Async Clock Mode - MultiLane**

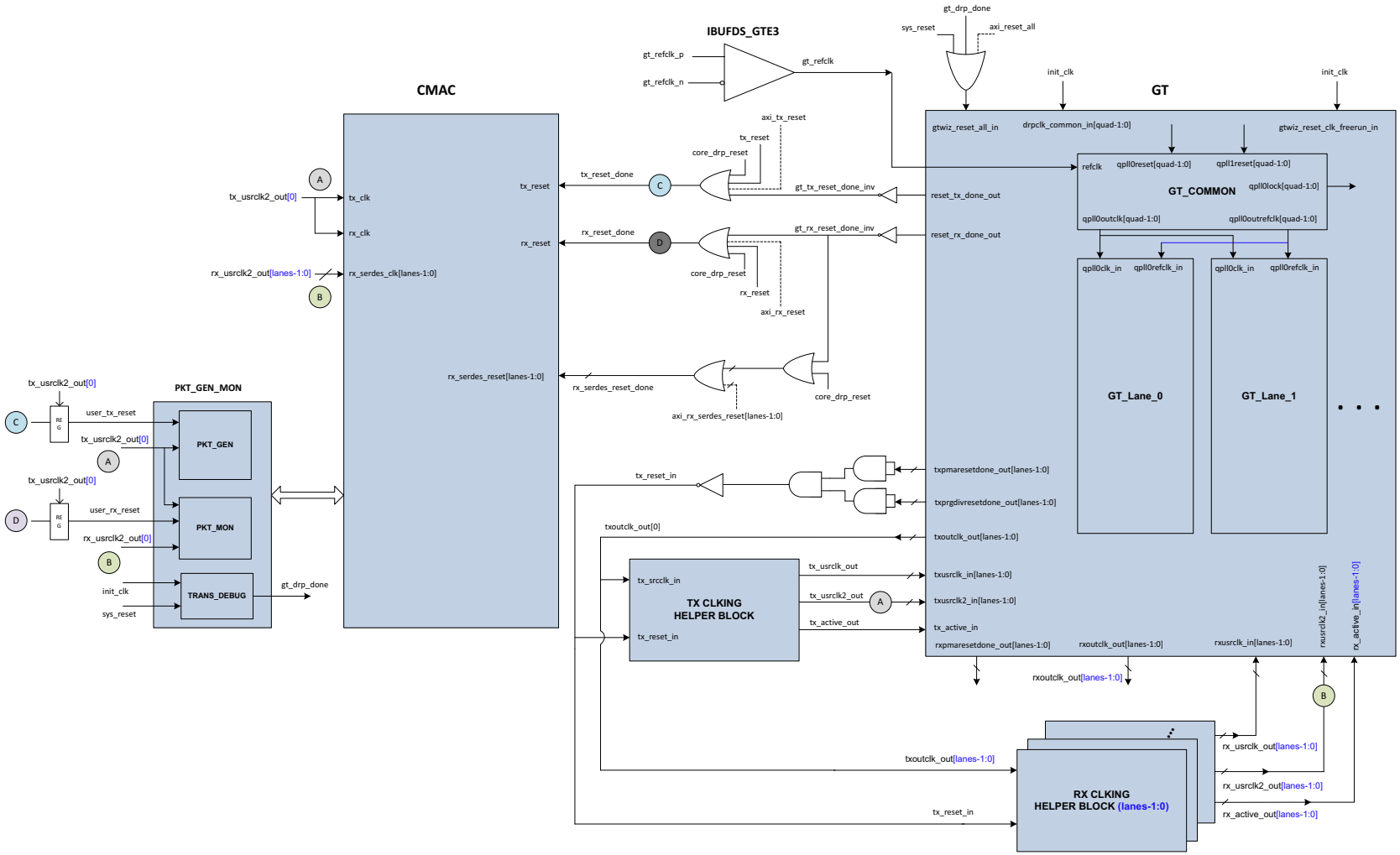

Figure 3-3: **CMAC Clocking and Reset - Sync Clock Mode - Single Lane**

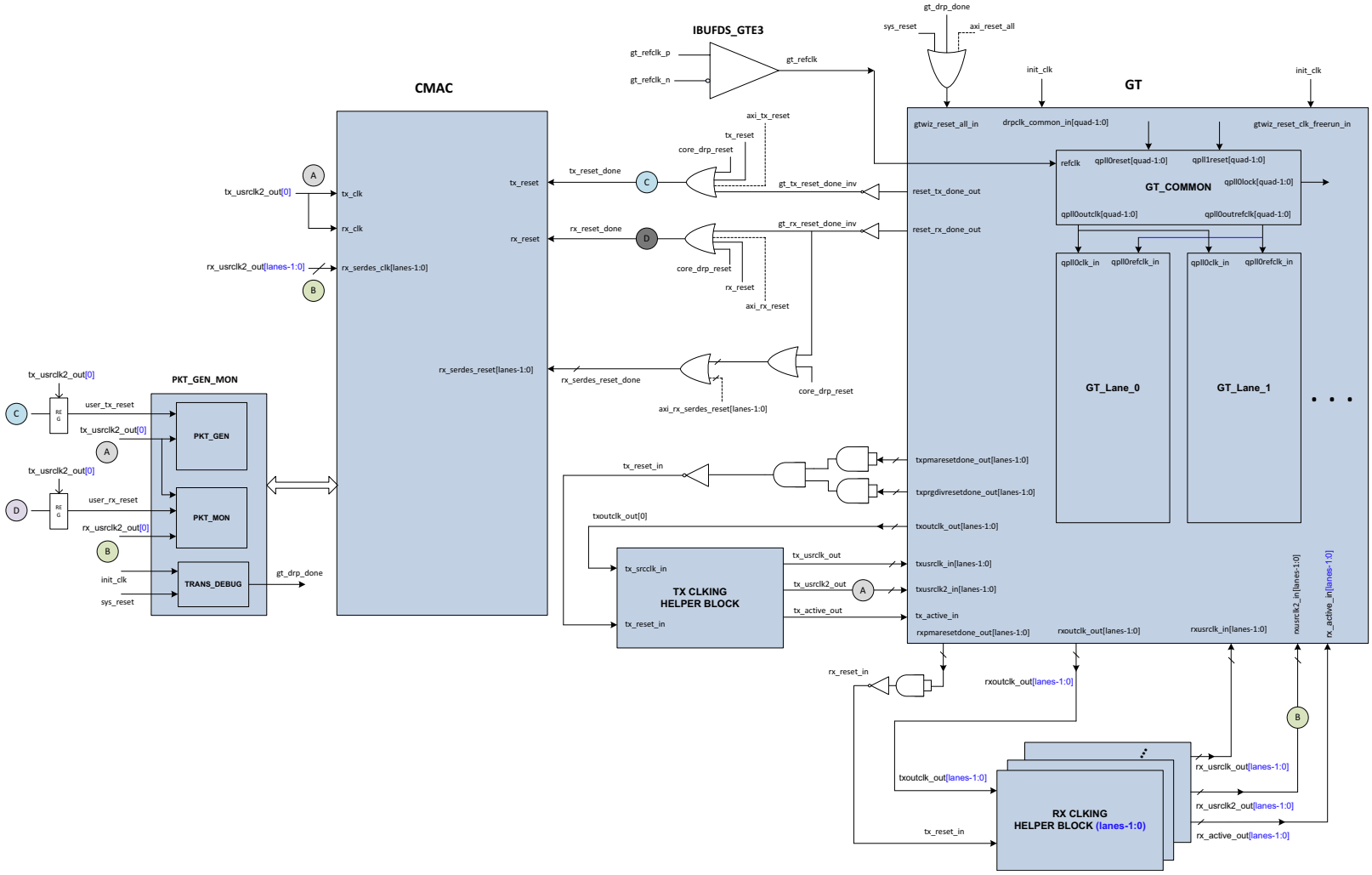

Figure 3-4: **CMAC Clocking and Reset - Async Clock Mode - Single Lane**

Protocol Description

The 100G Ethernet IP core is fully designed to IEEE 802.3 specifications for the 100G Ethernet protocol. The 100G Ethernet IP core instantiates the CMAC block, GTH (CAUI-10) or GTY (CAUI-10 or CAUI-4) transceivers and clocking resources to implement the 100G Ethernet IP core protocol.

PCS

This section refers to the PCS lane logic within the CMAC block and not the PCS within the serial transceiver. The PCS lane logic architecture is based on distributing (or striping) parts of a packet over several (relatively) lower speed physical interfaces by the transmitting device.

The receiving device PCS lane logic is then responsible for de-striping the different parts and rebuilding the packet before handing it off to the CMAC block.

The receiver PCS lane logic must also deskew the data from the different physical interfaces as these might see different delays as they are transported throughout the network. Additionally, the core handles PCS lane swapping across all received PCS lanes, allowing the 100G Ethernet IP core to be used with all optical transport systems.

The PCS lane logic includes scrambling/descrambling and 64B/66B encoders/decoders capable of supporting the 100 Gb/s line rate. The frequency at which the PCS runs at is shown in Table 3-1.

Table 3-1: **100G PCS Frequencies**

Configuration	GT Interface Width	100G PCS Frequency
100G (4 x 25.78125)	80	322.266 MHz
100G (10 x 10.3125)	32	322.266 MHz

PCS Lane Multiplexing

Between the CAUI-10 and CAUI-4 modes, the PCS multiplexer blocks combine and distribute the Physical Medium Dependent (PMD) lanes from the SerDes to the internal PCS lane logic. Figure 3-5 illustrates the multiplexing and demultiplexing function contained in the RX and TX PCS multiplexer blocks for the SerDes interfaces which are 80-bits wide. The lower 32 bits are used in CAUI-10 mode.

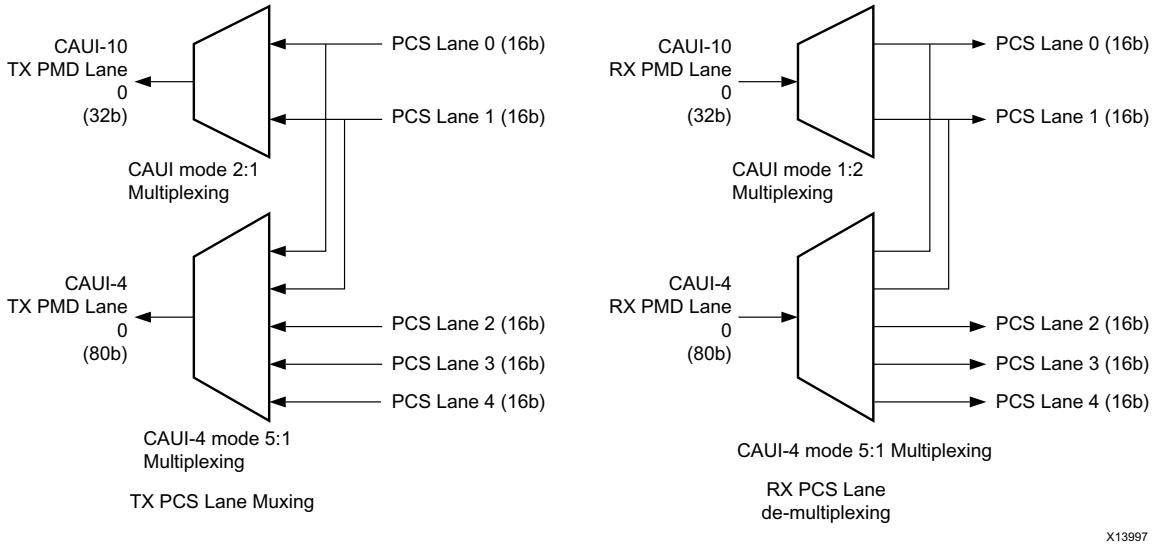

Figure 3-5: **PCS Multiplexing in CAUI-10 and CAUI-4 Modes**

The preceding pattern is repeated for the other three 80-bit SerDes interfaces.

Each 80-bit SerDes interface is actually composed of a 16-bit group and a 64-bit group. The mapping of these two groups onto the 80-bit interface is illustrated in Figure 3-6 and Figure 3-7 for RX and TX respectively.

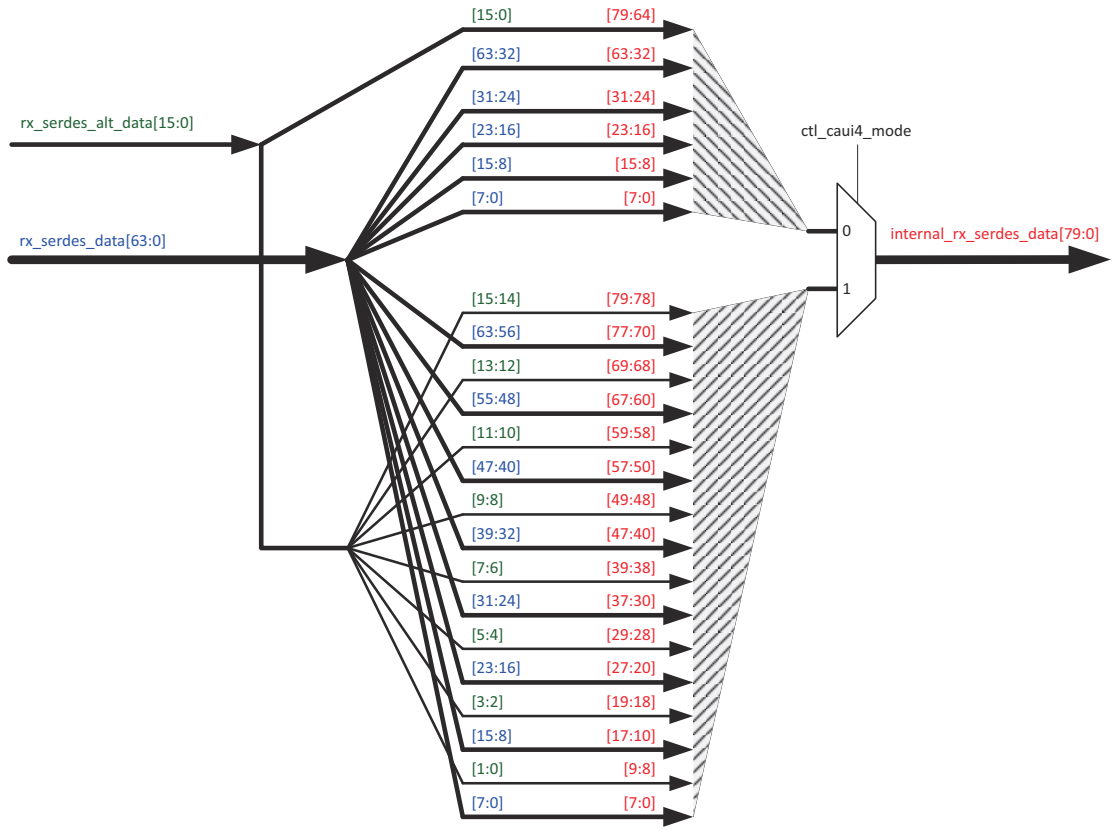

Figure 3-6: **RX GTY Mapping**

Note: The connectivity between the 100G Ethernet IP RX SerDes data interface to the GTY transceiver RX datapath for CAUI-10 and CAUI-4 operation is taken care of in the 100G Ethernet IP core.

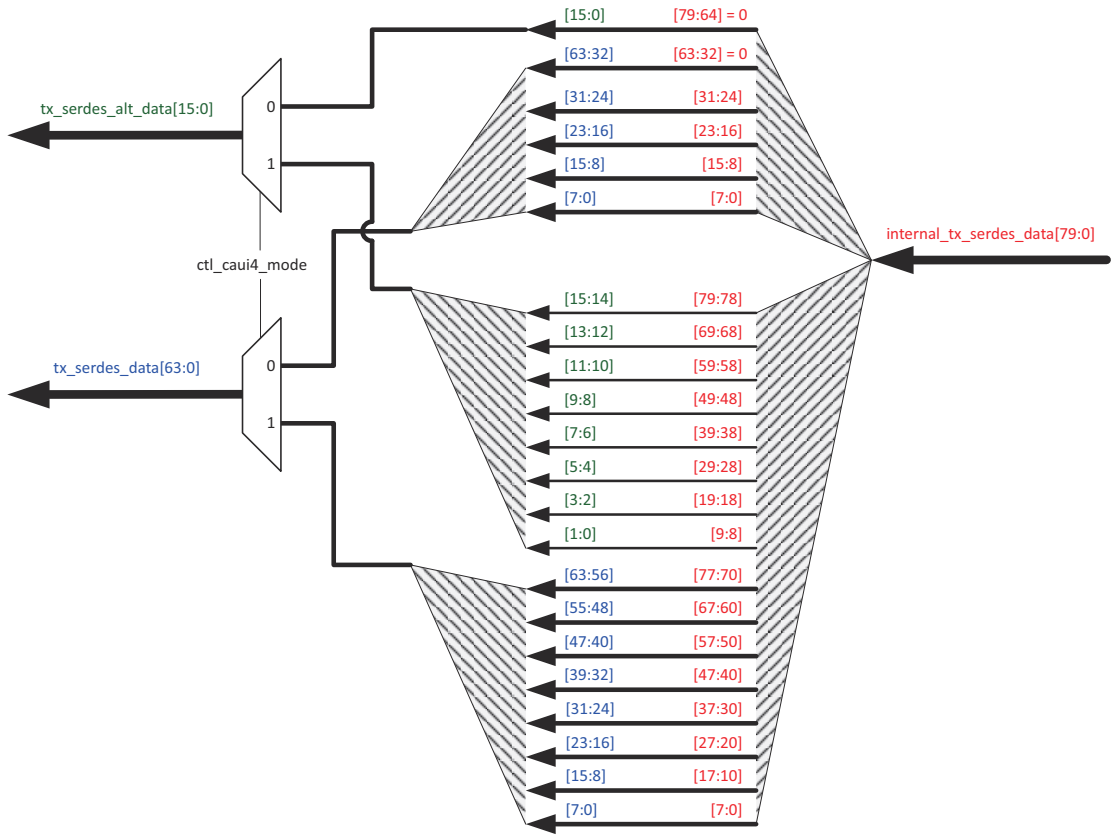

Figure 3-7: **TX GTY Mapping**

Note: The connectivity between the 100G Ethernet IP TX SerDes data interface to the GTY transceiver TX datapath for CAUI-10 and CAUI-4 operation is taken care of in the 100G Ethernet IP core.

PCS Lane Clock Distribution

The TX interface uses a common clock for all SerDes lanes. However in the RX direction, similar to the distribution of the data streams from the SerDes interface to the PCS lane, the RX PCS lane clocks also change with the operating mode. A hardened clock multiplexer block is used to change the clocking. Figure 3-8 illustrates this clock multiplexing by looking at the clock multiplexing required for PCS lanes 0 and 1.

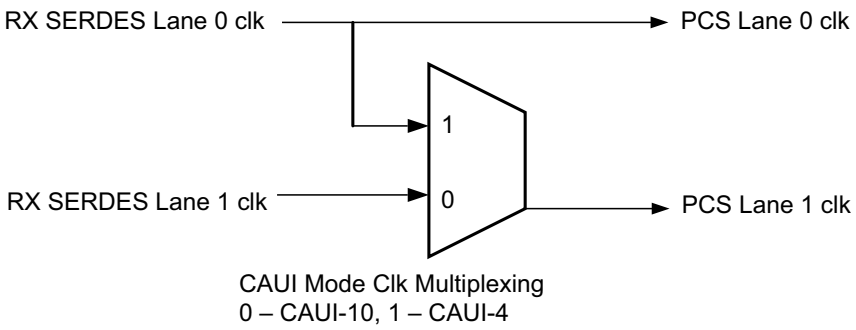

Figure 3-8: **RX PCS Lane0 and Lane1 Clocking**

MAC

The 100G Ethernet IP core provides several interfaces to interact with it. These consist of the following.

- User Side LBUS Interface (for RX and TX data and RX and TX control signals)
- Pause Processing Interface
- Status and Control Interface

User Side LBUS Interface

The user side interface of the UltraScale architecture 100G dedicated IP core is a simple packet interface referred to as the LBUS. The LBUS interface implemented in the 100G Ethernet IP core is 512-bits segmented.

The LBUS consists of three separate interfaces:

- Transmitter (TX) interface
- Receiver (RX) interface
- Status/Control interface

The transmitter accepts packet-oriented data, packages the data in accordance with the IEEE 802.3 Specification and sends that packaged data to the serial transceiver interface. The transmitter has control/configuration inputs to shape the data packaging to meet design-specific requirements.

The receiver accepts IEEE 802.3 data streams from the serial transceiver interface and provides packet-oriented data to the user side.

The status/control interface is used to set the characteristics of the interface and monitor its operation.

The 100G Ethernet IP core employs a segmented LBUS interface to prevent the loss of potential bandwidth that occurs at the end of a packet when the size of the packet is not a multiple of the LBUS width.

The segmented LBUS is a collection of narrower LBUSs, each 128 bits wide, with multiple transfers presented in parallel during the same clock cycle. Each segment has all the control signals associated with a complete 128-bit LBUS. The 512-bit segmented LBUS has four 128-bit segments with the signals for each segment shown in Table 3-2.

Table 3-2: **Segmented LBUS Signals**

Segment Number	TX Signals	RX Signals
0	tx_datain0[127:0] tx_enain0 tx_sopin0 tx_eopin0 tx_errin0 tx_mtyin0[3:0]	rx_dataout0[127:0] rx_enaout0 rx_sopout0 rx_eopout0 rx_errout0 rx_mtyout0[3:0]
1	tx_datain1[127:0] tx_enain1 tx_sopin1 tx_eopin1 tx_errin1 tx_mtyin1[3:0]	rx_dataout1[127:0] rx_enaout1 rx_sopout1 rx_eopout1 rx_errout1 rx_mtyout1[3:0]
2	tx_datain2[127:0] tx_enain2 tx_sopin2 tx_eopin2 tx_errin2 tx_mtyin2[3:0]	rx_dataout2[127:0] rx_enaout2 rx_sopout2 rx_eopout2 rx_errout2 rx_mtyout2[3:0]
3	tx_datain3[127:0] tx_enain3 tx_sopin3 tx_eopin3 tx_errin3 tx_mtyin3[3:0]	rx_dataout3[127:0] rx_enaout3 rx_sopout3 rx_eopout3 rx_errout3 rx_mtyout3[3:0]

The transmit and receive signals are defined as follows:

- `tx_datain0[127:0]`: Transmit LBUS Data. This bus receives input data from the user logic. The value of the bus is captured in every cycle for which `tx_enain` is sampled as 1.
- `tx_enain0`: Transmit LBUS Enable. This signal is used to enable the TX LBUS Interface. All signals on the LBUS interface are sampled only in cycles during which `tx_enain` is sampled as 1.
- `tx_sopin0`: Transmit LBUS Start Of Packet. This signal is used to indicate the Start Of Packet (SOP) when it is sampled as a 1 and is 0 for all other transfers of the packet. This signal is sampled only in cycles during which `tx_enain` is sampled as 1.
- `tx_eopin0`: Transmit LBUS End Of Packet. This signal is used to indicate the EOP when it is sampled as a 1 and is 0 for all other transfers of the packet. This signal is sampled only in cycles during which `tx_enain` is sampled as 1.

- `tx_errin0`: Transmit LBUS Error. This signal is used to indicate a packet contains an error when it is sampled as a 1 and is 0 for all other transfers of the packet. This signal is sampled only in cycles during which `tx_enain` and `tx_eopin` are sampled as 1.
- `tx_mtyin0[3:0]`: Transmit LBUS Empty. This bus is used to indicate how many bytes of the `tx_datain` bus are empty or invalid for the last transfer of the current packet. This bus is sampled only in cycles that `tx_enain` and `tx_eopin` are sampled as 1.

  When `tx_eopin` and `tx_errin` are sampled as 1, the value of `tx_mtyin[2:0]` is ignored as treated as if it is 000. The other bits of `tx_mtyin` are used as usual.
- `rx_dataout0[127:0]`: Receive LBUS Data. The value of the bus is only valid in cycles during which rx_enaout is sampled as 1.
- `rx_enaout0`: Receive LBUS Enable. This signal qualifies the other signal of the RX LBUS Interface. Signals of the RX LBUS Interface are only valid in cycles during which `rx_enaout` is sampled as 1.
- `rx_sopout0`: Receive LBUS SOP. This signal indicates the SOP when it is sampled as 1 and is only valid in cycles during which `rx_enaout` is sampled as a 1.
- `rx_eopout0`: Receive LBUS EOP. This signal indicates the EOP when it is sampled as 1 and is only valid in cycles during which `rx_enaout` is sampled as a 1.
- `rx_errout0`: Receive LBUS Error. This signal indicates that the current packet being received has an error when it is sampled as 1. This signal is only valid in cycles when both `rx_enaout` and `rx_eopout` are sampled as a 1. When this signal is 0, it indicates that there is no error in the packet being received.
- `rx_mtyout0[3:0]`: Receive LBUS Empty. This bus indicates how many bytes of the `rx_dataout` bus are empty or invalid for the last transfer of the current packet. This bus is only valid in cycles when both `rx_enaout` and `rx_eopout` are sampled as 1.

  When `rx_errout` and `rx_enaout` are sampled as 1, the value of `rx_mtyout[2:0]` is always 000. Other bits of `rx_mtyout` are as usual.

The transmitter accepts packet-oriented data. The transmitter has control/configuration inputs to shape the data packaging to meet design-specific requirements. The receiver accepts Ethernet bitstreams from the SerDes and provides packet-oriented data to the user side segmented LBUS.

IMPORTANT: *In the following section, the term "asserting" is used to mean "assigning a value of 1," and the term "negating" is used to mean "assigning a value of 0."*

TX LBUS Interface

The synchronous TX Local bus interface accepts packet-oriented data of arbitrary length. All signals are synchronous relative to the rising-edge of the `clk` port. Figure 3-9 shows a sample waveform for data transactions for two consecutive 65-byte packets using a 512-bit segmented bus. Each of the four segments is 128-bits wide.

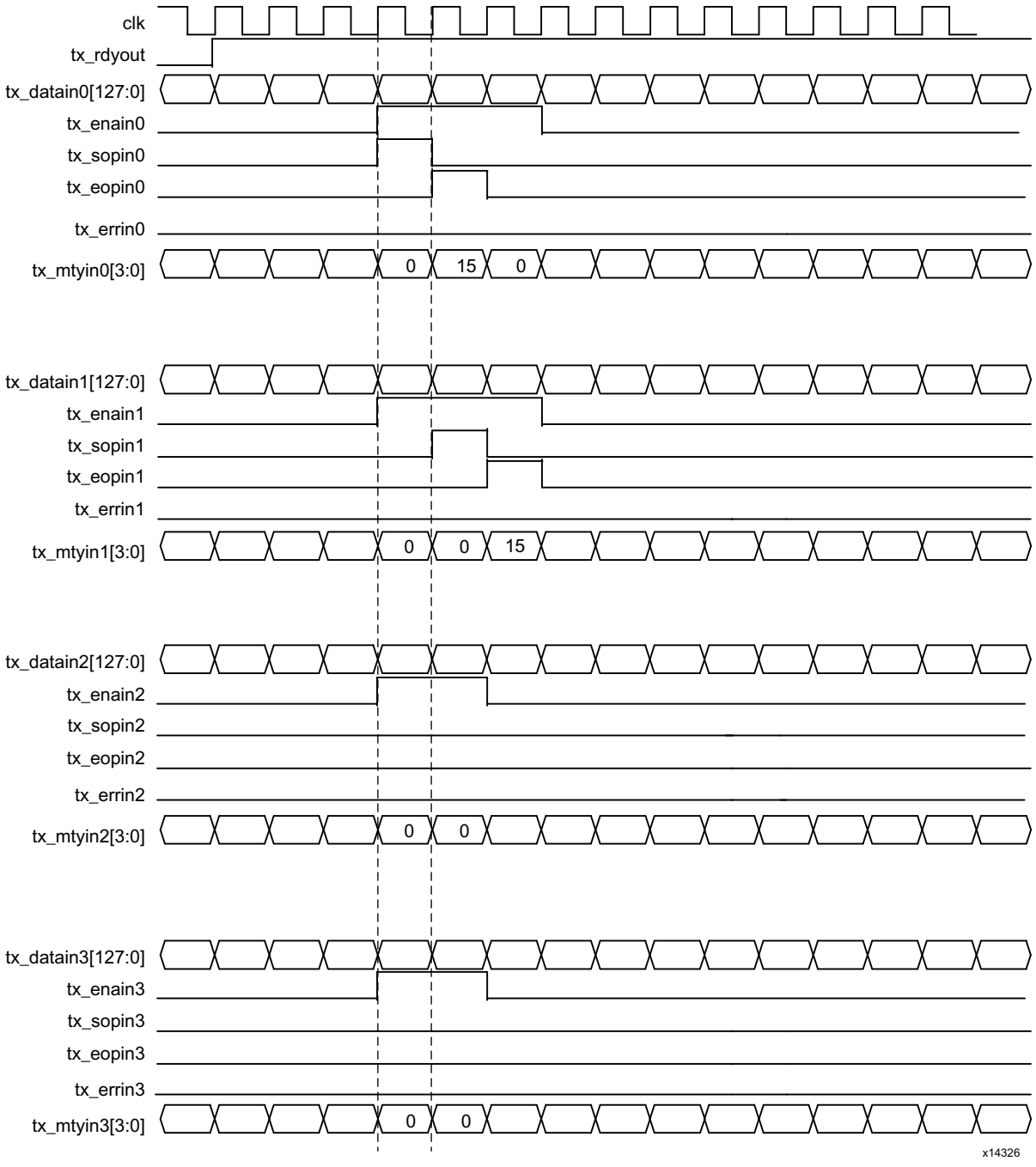

Figure 3-9: **Transmit Timing Diagram**

TX Transactions

Data is written into the interface on every clock cycle when tx_enain is asserted. This signal qualifies the other inputs of the TX Local bus interface. This signal must be valid every clock cycle. When tx_enain is deasserted, data on the other buses is ignored.

The start of a packet is identified by asserting tx_sopin with tx_enain. The end of a packet is identified by asserting tx_eopin with tx_enain. Both tx_sopin and tx_eopin can be asserted during the same cycle provided there are no empty segments between them. This is done for packets that are less than or equal to the bus width.

Data is presented on the tx_datain inputs. For a given segment, the first byte of the packet is written on bits [127:120], the second byte on bits [119:112], and so forth.

For a 128-bit segment, the first 16 bytes of a packet are presented on the bus during the cycle that tx_sopin and tx_enain are asserted. Subsequent 16-byte chunks are written during successive cycles with tx_sopin negated. The last bytes of the packet are written with tx_eopin asserted. Unless tx_eopin is asserted, all 128 bits must be presented with valid data whenever tx_enain is asserted.

During the last cycle of a packet, the tx_mtyin signals might be asserted. The value of tx_mtyin must be 0 for all but the last cycle. The tx_mtyin signals indicate how many byte lanes in the data bus are invalid (or empty). The tx_mtyin signals only have meaning during cycles when both tx_enain and tx_eopin are asserted. For a 128-bit wide segment, tx_mtyin is 4 bits wide.

If tx_mtyin has a value of 0x0, there are no empty byte lanes, or in other words, all bits of the data bus are valid. If tx_mtyin has a value of 0x1, then the 1-byte lane is empty. Specifically bits [7:0] of tx_datain do not contain valid data. If tx_mtyin has a value of 0x2, then the 2-byte lanes are empty. Specifically bits [15:0] do not contain valid data. If tx_mtyin has a value of 0x3, then 3-byte lanes are empty, and specifically bits [23:0] do not contain valid data This pattern continues until 15 of 16 bytes are invalid or empty. Table 3-3 shows the relation of tx_mytin and empty byte lanes.

Table 3-3: **tx_mtyin Values**

tx_mtyin Value	Empty Byte Lane(s)	Empty Bits of tx_datain
0x0	None	None
0x1	1 byte	[7:0]
0x2	2 byte	[15:0]
0x3	3 byte	[23:0]
...	...	...
0x15	15 byte	[119:0]

During the last cycle of a packet, when tx_eopin is asserted with tx_enain, tx_errin might also be asserted. This marks the packet as being in error and the last data word is replaced with the 802.3 Error Code. When tx_errin is asserted, the value of tx_mtyin is ignored.

tx_rdyout

Data can be safely written, that is, `tx_enain` asserted, when `tx_rdyout` is asserted. After `tx_rdyout` is negated, additional writes using `tx_enain` can be safely performed provided `tx_ovfout` is never asserted. When `tx_rdyout` is asserted again, additional data can be written. If at any time the back-pressure mechanism is violated, the `tx_ovfout` is asserted to indicate the violation. The threshold for an overflow indication is fixed at a value of 11. Up to eight write cycles might be safely performed after `tx_rdyout` is negated, but no more until `tx_rdyout` is asserted again.

RX LBUS Interface

The synchronous RX Local bus interface provides packet-oriented data much like the TX Local bus interface accepts. All signals are synchronous with the rising-edge of the Local bus clock. Figure 3-10 shows a sample waveform for two data transactions for 65-byte packets using a 512-bit segmented LBUS.

Figure 3-10: **Receive Timing Diagram**

Data is supplied by the 100G Ethernet IP core on every `clk` clock cycle when `rx_enaout` is asserted. This signal qualifies the other outputs of the RX Local bus interface.

The RX is similar to the TX, in that `rx_sopout` identifies the start of a packet and `rx_eopout` identifies the end of a packet. Both `rx_sopout` and `rx_eopout` are asserted during the same cycle for packets that are less than or equal to the bus width.

As in the TX, the first byte of a packet is supplied on the most significant bits of `rx_dataout`. For a 128-bit wide segment, the first byte of the packet is written on bits [127:120], the second byte on bits [119:112], and so forth.

As in the TX, portions of packets are written on the bus in the full width of the bus unless `rx_eopout` is asserted. When `rx_eopout` is asserted, the `rx_mtyout` bus indicates how many byte lanes in the data bus are invalid. The encoding is the same as for `tx_mtyin`.

During the last cycle of a packet, when `rx_eopout` is asserted with `rx_enaout`, `rx_errout` might also be asserted. This indicates the packet received had one of the following errors:

- FCS error
- Length out of the valid range (64 to CTL_RX_MAX_PACKET_LEN[14:0] bytes)
- Bad 64B/66B code received during receipt of the packet

There is no mechanism to back-pressure the RX Local bus interface. The user logic must be capable of receiving data when `rx_enaout` is asserted.

Bus Rules

This section describes the rules that govern the successful use of the segmented LBUS protocol.

Segment Ordering

The 128-bit segments are ordered 0 to 3 (for a 512-bit segmented LBUS). The first of the 128-bit transfers occurs on segment 0, the second on segment 1, and so forth. During each local bus clock cycle that data is transferred on the segmented LBUS, segment 0 must be active. The segmented bus is aligned so that the first bit of the incoming data is placed at the MSB of segment 0.

Active Segments

Data is transferred in a segment on the TX interface when the corresponding `tx_enainS` is a value of 1. The TX interface buffers data, but packets must be written in their entirety unless backpressure is applied (see Gaps). Therefore, it is acceptable to have clock cycles in which none of the `tx_enainS` signals are active during backpressure. However, during a clock cycle with `tx_enain0` active, segments must be filled in sequence with no gaps between active segments. The following are some of the illegal combinations of tx_enainX:

- tx_enain0=0, tx_enain1=1, tx_enain2=1, tx_enain3=1
- tx_enain0=1, tx_enain1=0, tx_enain2=1, tx_enain3=1
- tx_enain0=1, tx_enain1=1, tx_enain2=0, tx_enain3=1

Data is transferred in a segment on the RX interface when the corresponding `rx_enainS` is a value of 1. Similarly, the RX interface buffers data and does not forward until it has a sufficient quantity. Therefore, there are clock cycles in which none of the `rx_enainS` signals are active.

TX Backpressure

The optimal use of bandwidth requires that TX local bus data can be written at a rate faster than it can be delivered on the serial interface. This means that there must be backpressure, or flow-control, on the TX segmented LBUS. The signals used to implement backpressure are `tx_rdyout` and `tx_ovfout`. These signals are common for all segments. When responding to backpressure during a clock cycle, none of the `tx_enainS` can be active.

Gaps

The purpose of the segmented LBUS is to provide a means to optimally use the data bus. Therefore, as discussed in Active Segments, segments must be filled in sequence with no gaps between used segments. However, if a segment has an EOP, the following segments might be inactive. For example, the following combinations are permitted during a single clock cycle:

- tx_enain0=1 tx_eopin0=0 tx_enain1=1 tx_eopin1=0
  tx_enain2=1 tx_eopin2=1 tx_enain3=0 tx_eopin3=0
- tx_enain0=1 tx_eopin0=0 tx_enain1=1 tx_eopin1=1
  tx_enain2=0 tx_eopin2=0 tx_enain3=0 tx_eopin3=0
- tx_enain0=1 tx_eopin0=1 tx_enain1=0 tx_eopin1=0
  tx_enain2=0 tx_eopin2=0 tx_enain3=0 tx_eopin3=0

Examples

This section contains examples that illustrate segmented LBUS cycles covering various combinations of SOP, Dat (data in the middle of a packet), EOP, and idle (no data on the bus). Valid and invalid cycles are shown.

The segmented LBUS is assumed to be 512 bits wide and each segment is 128 bits wide (16 bytes). The TX direction is illustrated. The RX direction has analogous behavior, but there are no invalid cycles on the receive segmented LBUS.

Valid Cycles

Table 3-4 shows possible valid TX segmented LBUS cycles.

Table 3-4: **Valid TX Segmented LBUS Cycles**

Clock Cycle	1	2	3	4	5	6	7	8	9	10
seg0	Dat	Idle	SOP	SOP	Dat	Dat	Idle	Dat	SOP	Idle
seg1	Dat	Idle	Dat	Dat	EOP	Dat	Idle	Dat	Dat	Idle
seg2	Dat	Idle	Dat	Dat	SOP	Dat	Idle	Dat	Dat	Idle
seg3	EOP	Idle	EOP	Dat	Dat	Dat	Idle	EOP	Dat	Idle
tx_rdyout	1	1	1	1	1	1	0	1	0	0
tx_ovfout	0	0	0	0	0	0	0	0	0	0

Cycle 1 shows the end of a packet transfer. If segment 3 (the EOP) is 16 bytes, then `tx_mtyin3` is 0. If segment 3 is less than 16 bytes, then `tx_mtyin3` is a value ranging from 0001b to 1111b.

Cycle 2 is idle and no data is transferred.

Cycle 3 shows the transfer of a packet having a length of 64 bytes.

CAUTION! *Packets less than 64 bytes are considered undersized according to the Ethernet 802.3-2012 specification, and they are marked as undersized by the signal `stat_tx_packet_small` (for the transmit direction). Undersized packets might cause the core to lock up and must be avoided.*

Cycle 4 shows the first part of the transfer of a packet greater than 64 bytes.

Cycle 5 shows the transfer of the end of the packet started in Cycle 4, as indicated by the EOP in Segment 1. Another packet might start during the same clock cycle, as indicated by the SOP in segment 2. There is no idle segment between the EOP and SOP.

Cycle 6 shows the transfer of additional data corresponding to the packet started during Cycle 5.

Cycle 7 is idle, even though the packet has not been completely transferred, due to the deassertion of `tx_rdyout`. This is the only instance where a packet transfer might be interrupted by idle cycles.

Cycle 8 shows the completion of the transfer of the packet started during Cycle 5.

During Cycle 9, `tx_rdyout` is deasserted. It is still possible to write data during that cycle because this is the first cycle it has been deasserted.

IMPORTANT: *Xilinx recommends that no additional data be written in subsequent cycles until* `tx_rdyout` *is asserted again, or there can be an overflow condition indicated by* `tx_ovfout`*. This must be avoided.*

Cycle 10 is idle due to the continued deassertion of `tx_rdyout`.

Invalid Cycles

Table 3-5 shows several invalid TX segmented LBUS cycles as indicated by the shading.

Table 3-5: **Invalid Segmented LBUS Cycles**

Clock Cycle	1	2	3	4	5	6	7	8	9	10	--	14	15
seg0	SOP	Idle	Sop	Dat	Dat	SOP	Idle	Dat	SOP	SOP		Dat	Dat
seg1	Dat	Idle	Dat	Dat	Dat	Dat	Idle	Dat	Dat	Dat		Dat	Dat
seg2	Dat	Idle	EOP	Dat	Dat	Dat	Idle	Dat	Idle	Dat	--	Dat	Dat
seg3	EOP	Idle	SOP	Dat	Dat	Dat	Idle	EOP	EOP	Dat		Dat	Dat
tx_rdyout	1	1	1	1	1	1	1	1	1	0		0	0
tx_ovfout	0	0	0	0	0	0	0	0	0	0		0	1

Cycle 3 is not valid because it contains two SOPs.

Cycle 5 does not contain an EOP even though there is an SOP in the next cycle.

Cycle 6 has an SOP even though the preceding packet was not closed with an EOP. This sequence is not permitted by the LBUS rules and results in undefined behavior.

Cycle 7 is idle even though `tx_rdyout` is asserted, and a packet transfer is already under way. This can result in buffer under-run. If this occurs, the Ethernet packet is not sent in its entirety without interruption, and a malfunction of the FCS calculation occurs.

Cycle 9 contains an idle segment during a packet transfer which is not permitted by the segmented LBUS rules.

Cycle 14 is not recommended because a data transfer is being performed even though `tx_rdyout` has been deasserted for the fifth consecutive cycle.

Cycle 15 must never be performed because `tx_ovfout` has been asserted. In the event of `tx_ovfout` being asserted, the 100G Ethernet IP core should be reset.

Pause Processing Interface

The dedicated 100G Ethernet IP core provides a comprehensive mechanism for pause packet termination and generation. The TX and RX have independent interfaces for processing pause information as described in this section.

TX Pause Generation

You can request a pause packet to be transmitted using the `CTL_TX_PAUSE_REQ[8:0]` and `CTL_TX_PAUSE_ENABLE[8:0]` input buses. Bit 8 corresponds to global pause packets and bits [7:0] correspond to priority pause packets.

Each bit of this bus must be held at a steady state for a minimum of 16 cycles before the next transition.

IMPORTANT: *The 100G Ethernet IP core does not support assertion of global and priority pause packets at the same time.*

The contents of the pause packet are determined using the following input pins:

Global pause packets:

- CTL_TX_DA_GPP[47:0]
- CTL_TX_SA_GPP[47:0]
- CTL_TX_ETHERTYPE_GPP[15:0]
- CTL_TX_OPCODE_GPP[15:0]
- CTL_TX_PAUSE_QUANTA8[15:0]

Priority pause packets:

- CTL_TX_DA_PPP[47:0]
- CTL_TX_SA_PPP[47:0]
- CTL_TX_ETHERTYPE_PPP[15:0]
- CTL_TX_OPCODE_PPP[15:0]
- CTL_TX_PAUSE_QUANTA0[15:0]
- CTL_TX_PAUSE_QUANTA1[15:0]
- CTL_TX_PAUSE_QUANTA2[15:0]
- CTL_TX_PAUSE_QUANTA3[15:0]
- CTL_TX_PAUSE_QUANTA4[15:0]
- CTL_TX_PAUSE_QUANTA5[15:0]
- CTL_TX_PAUSE_QUANTA6[15:0]
- CTL_TX_PAUSE_QUANTA7[15:0]

The dedicated 100G Ethernet IP core automatically calculates and adds the FCS to the packet. For priority pause packets, the dedicated 100G Ethernet IP core also automatically generates the enable vector based on the priorities that are requested.

To request a pause packet, you must set the corresponding bit of the `CTL_TX_PAUSE_REQ[8:0]` and `CTL_TX_PAUSE_ENABLE[8:0]` bus to a 1 and keep it at 1 for the duration of the pause request (that is, if these inputs are set to 0, all pending pause packets are cancelled).

The dedicated 100G Ethernet IP core will transmit the pause packet immediately after the current packet in flight is completed. Each bit of this bus must be held at a steady state for a minimum of 16 cycles before the next transition.

To retransmit pause packets, the dedicated 100G Ethernet IP core maintains a total of nine independent timers: one for each priority and one for global pause. These timers are loaded with the value of the corresponding input buses. After a pause packet is transmitted the corresponding timer is loaded with the corresponding value of the `CTL_TX_PAUSE_REFRESH_TIMER[8:0]` input bus. When a timer times out, another packet for that priority (or global) is transmitted as soon as the current packet in flight is completed. Additionally, you can manually force the timers to 0, and therefore, force a retransmission by setting the `CTL_TX_RESEND_PAUSE` input to 1 for one clock cycle.

To reduce the number of pause packets for priority mode operation, a timer is considered "timed out" if any of the other timers time out. Additionally, while waiting for the current packet in flight to be completed, any new timer that times out or any new requests from the you will be merged into a single pause frame. For example, if two timers are counting down, and you send a request for a third priority, the two timers are forced to be timed out and a pause packet for all three priorities is sent as soon as the current in-flight packet (if any) is transmitted.

Similarly, if one of the two timers times out without an additional request from you, both timers are forced to be timed out and a pause packet for both priorities is sent as soon as the current in-flight packet (if any) is transmitted.

You can stop pause packet generation by setting the appropriate bits of `CTL_TX_PAUSE_REQ[8:0]` or `CTL_TX_PAUSE_ENABLE[8:0]` to 0.

RX Pause Termination

The dedicated 100G Ethernet IP core terminates global and priority pause frames and provides a simple hand-shaking interface to allow user logic to respond to pause packets.

Determining Pause Packets

There are three steps in determining pause packets:

1. Checks are performed to see if a packet is a global or a priority control packet. Packets that pass step 1 are forwarded to you only if CTL_RX_FORWARD_CONTROL is set to 1.
2. If step 1 passes, the packet is checked to determine if it is a global pause packet.
3. If step 2 fails, the packet is checked to determine if it is a priority pause packet.

For step 1, the following pseudo code shows the checking function:

```
assign da_match_gcp = (!ctl_rx_check_mcast_gcp && !ctl_rx_check_ucast_gcp) || ((DA
== ctl_rx_pause_da_ucast) && ctl_rx_check_ucast_gcp) || ((DA == 48'h0180c2000001) &&
ctl_rx_check_mcast_gcp);

assign sa_match_gcp = !ctl_rx_check_sa_gcp || (SA == ctl_rx_pause_sa);

assign etype_match_gcp = !ctl_rx_check_etype_gcp || (ETYPE == ctl_rx_etype_gcp);

assign opcode_match_gcp = !ctl_rx_check_opcode_gcp || ((OPCODE >=
ctl_rx_opcode_min_gcp) && (OPCODE <= ctl_rx_opcode_max_gcp));

assign global_control_packet = da_match_gcp && sa_match_gcp && etype_match_gcp &&
opcode_match_gcp && ctl_rx_enable_gcp;

assign da_match_pcp = (!ctl_rx_check_mcast_pcp && !ctl_rx_check_ucast_pcp) || ((DA
== ctl_rx_pause_da_ucast) && ctl_rx_check_ucast_pcp) || ((DA ==
ctl_rx_pause_da_mcast) && ctl_rx_check_mcast_pcp);

assign sa_match_pcp = !ctl_rx_check_sa_pcp || (SA == ctl_rx_pause_sa);

assign etype_match_pcp = !ctl_rx_check_etype_pcp || (ETYPE == ctl_rx_etype_pcp);

assign opcode_match_pcp = !ctl_rx_check_opcode_pcp || ((OPCODE >=
ctl_rx_opcode_min_pcp) && (OPCODE <= ctl_rx_opcode_max_pcp));

assign priority_control_packet = da_match_pcp && sa_match_pcp && etype_match_pcp &&
opcode_match_pcp && ctl_rx_enable_pcp;

assign control_packet = global_control_packet || priority_control_packet;
```

where DA is the destination address, SA is the source address, OPCODE is the opcode, and ETYPE is the ethertype/length field that is extracted from the incoming packet.

For step 2, the following pseudo code shows the checking function:

```
assign da_match_gpp = (!ctl_rx_check_mcast_gpp && !ctl_rx_check_ucast_gpp) || ((DA
== ctl_rx_pause_da_ucast) && ctl_rx_check_ucast_gpp) || ((DA == 48'h0180c2000001) &&
ctl_rx_check_mcast_gpp);

assign sa_match_gpp = !ctl_rx_check_sa_gpp || (SA == ctl_rx_pause_sa);

assign etype_match_gpp = !ctl_rx_check_etype_gpp || (ETYPE == ctl_rx_etype_gpp);

assign opcode_match_gpp = !ctl_rx_check_opcode_gpp || (OPCODE == ctl_rx_opcode_gpp);

assign global_pause_packet = da_match_gpp && sa_match_gpp && etype_match_gpp &&
opcode_match_gpp && ctl_rx_enable_gpp;
```

where DA is the destination address, SA is the source address, OPCODE is the opcode, and ETYPE is the ethertype/length field that is extracted from the incoming packet.

For step 3, the following pseudo code shows the checking function:

```
assign da_match_ppp = (!ctl_rx_check_mcast_ppp && !ctl_rx_check_ucast_ppp) || ((DA
== ctl_rx_pause_da_ucast) && ctl_rx_check_ucast_ppp) || ((DA ==
ctl_rx_pause_da_mcast) && ctl_rx_check_mcast_ppp);

assign sa_match_ppp = !ctl_rx_check_sa_ppp || (SA == ctl_rx_pause_sa);

assign etype_match_ppp = !ctl_rx_check_etype_ppp || (ETYPE == ctl_rx_etype_ppp);

assign opcode_match_ppp = !ctl_rx_check_opcode_ppp || (OPCODE == ctl_rx_opcode_ppp);

assign priority_pause_packet = da_match_ppp && sa_match_ppp && etype_match_ppp &&
opcode_match_ppp && ctl_rx_enable_ppp;
```

where DA is the destination address, SA is the source address, OPCODE is the opcode, and ETYPE is the ethertype/length field that is extracted from the incoming packet.

User Interface

A simple hand-shaking protocol alerts you of the reception of pause packets using the `CTL_RX_PAUSE_ENABLE[8:0]`, `STAT_RX_PAUSE_REQ[8:0]` and `CTL_RX_PAUSE_ACK[8:0]` buses. For both buses, bit [8] corresponds to global pause packets and bits [7:0] correspond to priority pause packets.

The following steps occur when a pause packet is received:

1. If the corresponding bit of `CTL_RX_PAUSE_ENABLE[8:0]` is 0, the quanta is ignored and the dedicated 100G Ethernet IP core stays in step 1. Otherwise, the corresponding bit of the `STAT_RX_PAUSE_REQ[8:0]` bus is set to 1, and the received quanta is loaded into a timer.

   Note: If one of the bits of `CTL_RX_PAUSE_ENABLE[8:0]` is set to 0 (that is, disabled) when the pause processing is in step 2 or later, the dedicated 100G Ethernet IP core completes the steps as normal until it comes back to step 1.

2. If CTL_RX_CHECK_ACK input is 1, the dedicated 100G Ethernet IP core waits for you to set the appropriate bit of the CTL_RX_PAUSE_ACK[8:0] bus to 1.
3. After you set the proper bit of CTL_RX_PAUSE_ACK[8:0] to 1, or if CTL_RX_CHECK_ACK is 0, the dedicated 100G Ethernet IP core starts counting down the timer.
4. When the timer times out, the dedicated 100G Ethernet sets the appropriate bit of STAT_RX_PAUSE_REQ[8:0] back to 0.
5. If CTL_RX_CHECK_ACK input is 1, the operation is complete when you set the appropriate bit of CTL_RX_PAUSE_ACK[8:0] back to 0.

   If you do not set the appropriate bit of CTL_RX_PAUSE_ACK[8:0] back to 0, the dedicated 100G Ethernet IP core deems the operation complete after 32 clock cycles.

The preceding steps are demonstrated in Figure 3-11 with each step shown on the wave form.

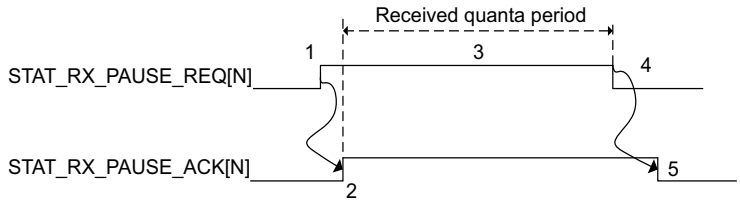

Figure 3-11: **RX Pause Interface Example**

If at any time during steps 2 to 5 a new pause packet is received, the timer is loaded with the newly acquired quanta value and the and the process continues.

Status and Control Interface

The status/control interface allows you to set up the 100G Ethernet IP core configuration and to monitor the status of the core. The following subsections describe the various status and control signals.

RX and TX

The 802.3-2012 defines the PCS Lane marker values. These are shown in Table 3-6.

Table 3-6: **PCS Lane Marker Values**

PCS Lane Marker Attributes	Value
CTL_RX_VL_MARKER_ID[0][63:0] CTL_TX_VL_MARKER_ID[0][63:0]	64'hc1_68_21_00_3e_97_de_00
CTL_RX_VL_MARKER_ID[1][63:0] CTL_TX_VL_MARKER_ID[1][63:0]	64'h9d_71_8e_00_62_8e_71_00
CTL_RX_VL_MARKER_ID[2][63:0] CTL_TX_VL_MARKER_ID[2][63:0]	64'h59_4b_e8_00_a6_b4_17_00
CTL_RX_VL_MARKER_ID[3][63:0] CTL_TX_VL_MARKER_ID[3][63:0]	64'h4d_95_7b_00_b2_6a_84_00
CTL_RX_VL_MARKER_ID[4][63:0] CTL_TX_VL_MARKER_ID[4][63:0]	64'hf5_07_09_00_0a_f8_f6_00
CTL_RX_VL_MARKER_ID[5][63:0] CTL_TX_VL_MARKER_ID[5][63:0]	64'hdd_14_c2_00_22_eb_3d_00
CTL_RX_VL_MARKER_ID[6][63:0] CTL_TX_VL_MARKER_ID[6][63:0]	64'h9a_4a_26_00_65_b5_d9_00
CTL_RX_VL_MARKER_ID[7][63:0] CTL_TX_VL_MARKER_ID[7][63:0]	64'h7b_45_66_00_84_ba_99_00
CTL_RX_VL_MARKER_ID[8][63:0] CTL_TX_VL_MARKER_ID[8][63:0]	64'ha0_24_76_00_5f_db_89_00
CTL_RX_VL_MARKER_ID[9][63:0] CTL_TX_VL_MARKER_ID[9][63:0]	64'h68_c9_fb_00_97_36_04_00
CTL_RX_VL_MARKER_ID[10][63:0] CTL_TX_VL_MARKER_ID[10][63:0]	64'hfd_6c_99_00_02_93_66_00
CTL_RX_VL_MARKER_ID[11][63:0] CTL_TX_VL_MARKER_ID[11][63:0]	64'hb9_91_55_00_46_6e_aa_00
CTL_RX_VL_MARKER_ID[12][63:0] CTL_TX_VL_MARKER_ID[12][63:0]	64'h5c_b9_b2_00_a3_46_4d_00
CTL_RX_VL_MARKER_ID[13][63:0] CTL_TX_VL_MARKER_ID[13][63:0]	64'h1a_f8_bd_00_e5_07_42_00
CTL_RX_VL_MARKER_ID[14][63:0] CTL_TX_VL_MARKER_ID[14][63:0]	64'h83_c7_ca_00_7c_38_35_00
CTL_RX_VL_MARKER_ID[15][63:0] CTL_TX_VL_MARKER_ID[15][63:0]	64'h35_36_cd_00_ca_c9_32_00
CTL_RX_VL_MARKER_ID[16][63:0] CTL_TX_VL_MARKER_ID[16][63:0]	64'hc4_31_4c_00_3b_ce_b3_00
CTL_RX_VL_MARKER_ID[17][63:0] CTL_TX_VL_MARKER_ID[17][63:0]	64'had_d6_b7_00_52_29_48_00

Table 3-6: **PCS Lane Marker Values *(Cont'd)***

PCS Lane Marker Attributes	Value
CTL_RX_VL_MARKER_ID[18][63:0] CTL_TX_VL_MARKER_ID[18][63:0]	64'h5f_66_2a_00_a0_99_d5_00
CTL_RX_VL_MARKER_ID[19][63:0] CTL_TX_VL_MARKER_ID[19][63:0]	64'hc0_f0_e5_00_3f_0f_1a_00

RX PCS Lane Alignment Status

The 100G Ethernet IP core provides status bits to indicate the state of word boundary synchronization and PCS lane alignment. All signals are synchronous with the rising-edge of `RX_CLK`. A detailed description of each signal follows.

STAT_RX_SYNCED[19:0]

When a bit of this bus is 0, it indicates that word boundary synchronization of the corresponding lane is not complete or that an error has occurred as identified by another status bit.

When a bit of this bus is 1, it indicates that the corresponding lane is word boundary synchronized and is receiving PCS Lane Marker Words as expected.

STAT_RX_SYNCED_ERR[19:0]

When a bit of this bus is 1, it indicates one of several possible failures on the corresponding lane.

- Word boundary synchronization in the lane was not possible using Framing bits [65:64].
- After word boundary synchronization in the lane was achieved, errors were detected on Framing bits [65:64].
- After word boundary synchronization in the lane was achieved, a valid PCS Lane Marker Word was never received.

The bits of the bus remain asserted until word boundary synchronization occurs or until some other error/failure is signaled for the corresponding lane.

STAT_RX_MF_LEN_ERR[19:0]

When a bit of this bus is 1, it indicates that PCS Lane Marker Words are being received but not at the expected rate in the corresponding lane. The transmitter and receiver must be re-configured with the same Meta Frame length.

The bits of the bus remain asserted until word boundary synchronization occurs or until some other error/failure is signaled for the corresponding lane.

STAT_RX_MF_REPEAT_ERR[19:0]

After word boundary synchronization is achieved in a lane, if a bit of this bus is a 1, it indicates that four consecutive invalid PCS Lane Marker Words were detected in the corresponding lane.

The bits of the bus remain asserted until re- synchronization occurs or until some other error/failure is signaled for the corresponding lane.

STAT_RX_MF_ERR[19:0]

When a bit of this bus is 1, it indicates that an invalid PCS Lane Marker Word was received on the corresponding lane. This bit is only asserted after word boundary synchronization is achieved. This output is asserted for one clock period each time an invalid Meta Packet Synchronization Word is detected.

STAT_RX_ALIGNED

When `STAT_RX_ALIGNED` is a value of 1, all of the lanes are aligned/de-skewed and the receiver is ready to receive packet data.

STAT_RX_ALIGNED_ERR

When `STAT_RX_ALIGNED_ERR` is a value of 1, one of two things occurred. Lane alignment failed after several attempts, or lane alignment was lost (`STAT_RX_ALIGNED` was asserted and then it was negated).

STAT_RX_MISALIGNED

When `STAT_RX_MISALIGNED` is a value of 1, a valid PCS Lane Marker Word was not received on all PCS lanes simultaneously. This output is asserted for one clock period each time this error condition is detected.

STAT_RX_FRAMING_ERR_[0-19][3:0] and STAT_RX_FRAMING_ERR_VALID_[0-19]

This set of buses is intended to be used to keep track of sync header errors. There is a pair of outputs for each PCS Lane. The `STAT_RX_FRAMING_ERR_[0-19]` output bus indicates how many sync header errors were received and it is qualified (that is, the value is only valid) when the corresponding `STAT_RX_FRAMING_ERR_VALID[0-19]` is sampled as a 1.

STAT_RX_VL_NUMBER_[0-19][4:0]

Each bus indicates which PCS lane will have its status reflected on a specific status pins. For example, `STAT_RX_VLANE_NUMBER_0` indicates which PCS lane will have its status reflected on pin 0 of the other status signals. These buses can be used to detect if a PCS lane has not been found or if one has been mapped to multiple status pins.

STAT_RX_VL_DEMUXED[19:0]

After word boundary synchronization is achieved on each lane, if a bit of this bus is 1 it indicates that the corresponding PCS lane was properly found and demultiplexed.

STAT_RX_BLOCK_LOCK[19:0]

Each bit indicates that the corresponding PCS lane has achieved sync header lock as defined by the 802.3-2012. A value of 1 indicates block lock is achieved.

STAT_RX_STATUS

This output is set to a 1 when `STAT_RX_ALIGNED` is a 1 and `STAT_RX_HI_BER` is a 0. This is defined by the 802.3-2012.

STAT_RX_LOCAL_FAULT

This output is High when `STAT_RX_INTERNAL_LOCAL_FAULT` or `STAT_RX_RECEIVED_LOCAL_FAULT` is asserted. This output is level sensitive.

RX Error Status

The 100G Ethernet IP core provides status signals to identify 64B/66B words and sequences violations and CRC32 checking failures. All signals are synchronous with the rising-edge of `CLK`. A detailed description of each signal follows.

STAT_RX_BAD_FCS

When this signal is a value of 1, it indicates that the error detection logic has identified a mismatch between the expected and received value of CRC32 in the received packet.

When a CRC32 error is detected, the received packet is marked as containing an error and it is sent with `RX_ERROUT` asserted during the last transfer (the cycle with `RX_EOPOUT` asserted), unless `CTL_RX_IGNORE_FCS` is asserted. This signal is asserted for one clock period each time a CRC32 error is detected.

STAT_RX_BAD_CODE[6:0]

This signal indicates how many cycles the RX PCS receive state machine is in the RX_E state as defined by the 802.3-2012 specifications.

1588v2 Timestamping

The integrated block for the 100G Ethernet IP core supports 1588v2 timestamping. All the necessary signals are provided to allow external soft logic to make precise corrections to the timestamp captured by the IP. The core supports 2-step 1588v2 clocks through ingress and egress timestamp captures.

According to the IEEE 1588v2 standard, there are various PTP message encapsulations [Ref 1]. In the case of 2-step clocks, all types of encapsulation are possible with the 100G Ethernet IP core if the design includes a PTP-specific (software) implementation.

For future implementation of a 1-step clock, the encapsulation protocol (PTP message offset) has to be defined. Therefore the integrated CMAC will support the following encapsulations for 1-step operation:

- Ethernet
- IPv4 UDP
- IPv6 UDP

Inputs are provided for the timestamp offset value in the message, and for the RX path timestamp to use for the field adjustment. Further details on the function of the command fields are found in Table 2-2.

Receive Timestamp Function

The ingress logic does not parse the ingress packets to search for 1588 (PTP) frames. Instead, it takes a timestamp for every received frame and outputs this value to the user logic. The feature is always enabled, but the timestamp output can be ignored by users not requiring this function.

See Table 2-2 for a detailed description of signals related to the RX timestamping function.

To compensate for lane skew, the alignment buffer fill levels for each PCS lane are provided as outputs. The RX timestamp function is shown in Figure 3-12.

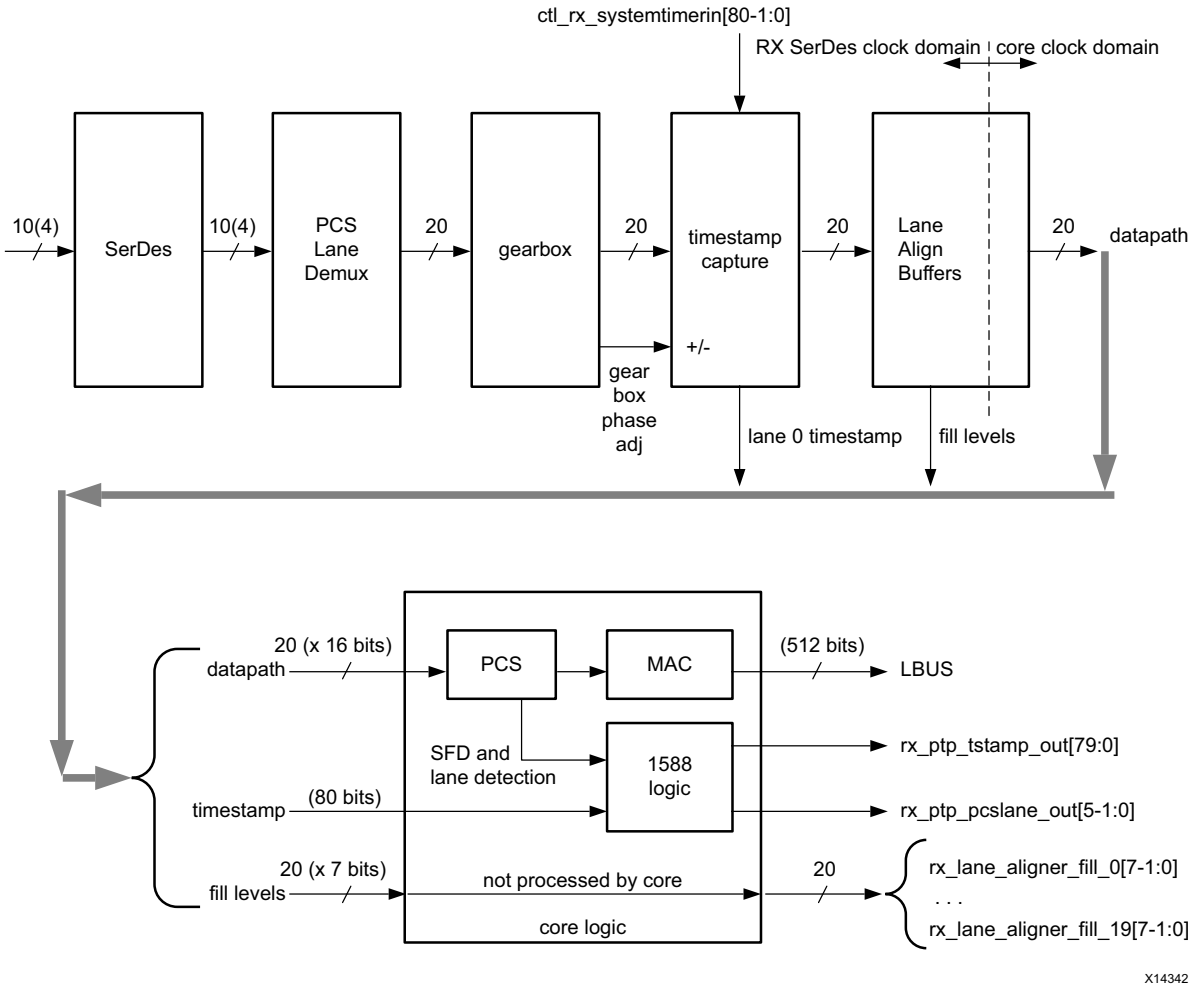

Figure 3-12: **RX Timestamping**

In Figure 3-12, timestamps are captured for each word of lane 0 which is exiting the gearbox plane. The capture logic accounts for the gearbox dead cycle which occurs every 33 cycles.

IMPORTANT: *The RX system timer input must be in lane 0 of the RX SerDes clock domain.*

Timestamps are filtered after the PCS decoder to retain only those timestamps corresponding to an SOP. The PCS also identifies the PCS lane on which the SOP occurred.

The lane alignment fill buffers are carried through to the user interface output. These average values of the fill levels are not expected to vary over time. The average value should be taken to the required accuracy to remove the clock cycle jitter. The alignment fill values reflect the static skew present in each lane.

The signals `stat_rx_vl_number_0[4:0]` to `stat_rx_vl_number_19[4:0]` can be used to correlate each PCS lane to a physical lane.

Soft logic can improve timestamp accuracy and compensate for the lane alignment FIFO fill levels by adding or subtracting the relative fill level of the selected lane. The reference fill level can be the minimum, average, or maximum fill levels. The relationship between the 100G Ethernet IP core and the soft logic is shown in Figure 3-13.

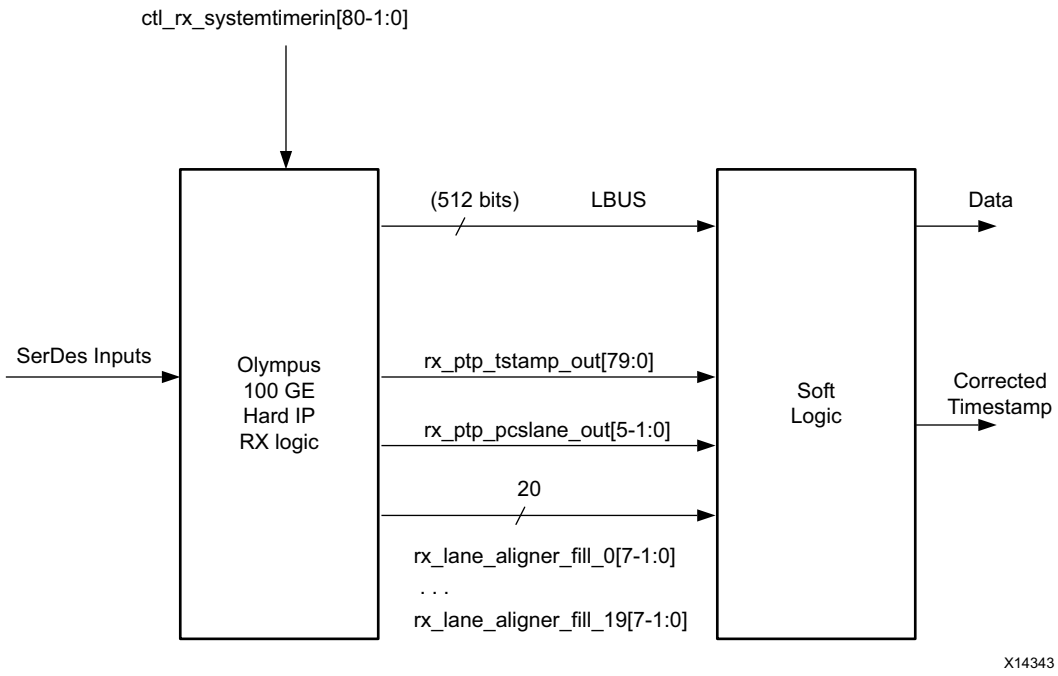

Figure 3-13: **Soft Logic**

The corrected timestamp is computed as:

rx_ptp_tstamp_out +/- rx_lane_aligner_fill_0 +/- Reference Fill Level

Where:

- `rx_ptp_tstamp_out` is the timestamp at the first gearbox, and is filtered by the PCS to correspond to the start of the SOP.
- `rx_lane_aligner_fill_0` is the time average of the alignment buffer fill level for the lane on which the timestamp was taken.
- Reference fill level is a constant value that is defined as the minimum, maximum, or average fill level.

Transmit 1588 Insertion and Timestamp Function

The egress logic uses an operation/command bus to identify frames that require time stamping returned to the user, or frames for which a timestamp should be inserted. See Table 2-2 for a description of the command fields.

Transmit timestamping is illustrated in Figure 3-14.

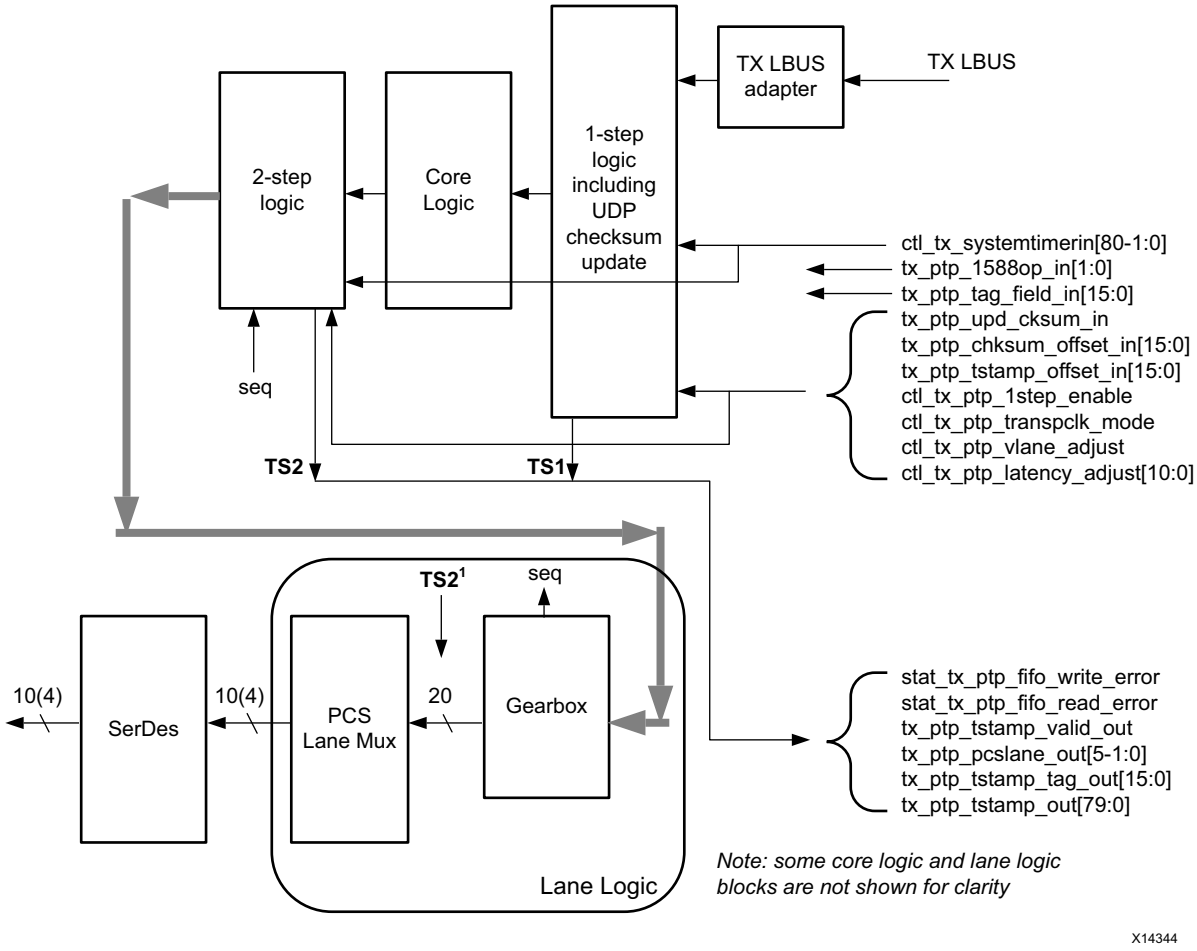

Figure 3-14: **TX Timestamping**

As seen on the diagram, timestamping logic exists in two locations depending on whether 1-step or 2-step operation is desired. 1-step operation requires user datagram protocol (UDP) checksum and FCS updates and therefore the FCS core logic is re-used.

The TS references are defined as follows:

- TS1: The output timestamp signal when a 1-step operation is selected.
- TS2: The output timestamp signal when a 2-step operation is selected.
- TS2': The plane to which both timestamps are corrected.

TS2 always has a correction applied so that it is referenced to the TS2' plane. TS1 might or might not have the TS2' correction applied, depending on the value of the signal `ctl_tx_ptp_latency_adjust[10:0]`. The default value of this signal is 705 (decimal).

Transmit 1588 Gearbox Jitter Compensation

The 2-step 1588 timestamp capture on the TX accounts for the jitter introduced by the transmit gearbox. The gearbox takes in 66-bit timestamped frames in 34/32 bit chunks and outputs data 32 bits at a time. Because 66 bits is not a multiple of 32, the gearbox accumulates excess data that is added to the beginning of subsequent cycles of data output. When data is appended from the gearbox buffer, jitter is introduced to the timestamped frames received by the gearbox. The gearbox has a state that is called the sequence number. For each sequence number, the gearbox has a specific number of bits buffered for adding to the beginning of the output data. The amount of jitter introduced by the gearbox can be represented by the graph in Figure 3-15.

TIP: *The timestamp of a frame aligns with the control bits at the start of the 66-bit frame. As a result, timestamp jitter compensation is applied according to the arrival time of the control bits at the gearbox. The actual compensation is done by multiplying the cycle period (3.103 ns) by the n/32 fraction based on the sequence number and adding that to the timestamp already associated with the 66-bit frame.*

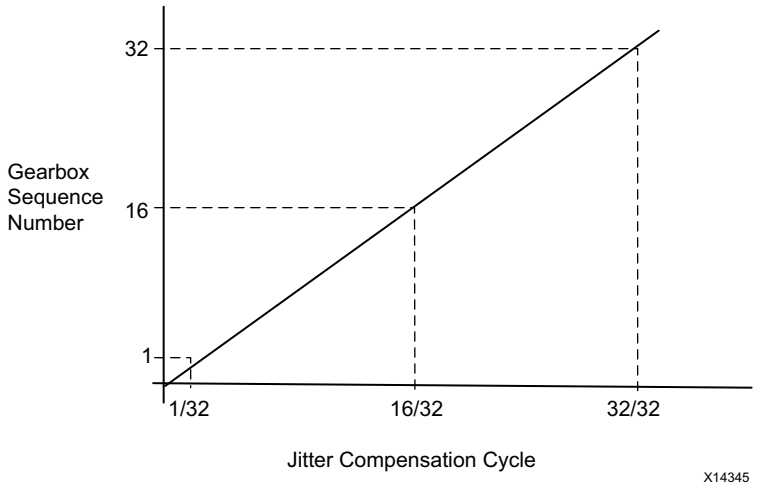

Figure 3-15: **Jitter Compensation**

Transceiver Selection Rules

The design must meet the following rules when connecting the 100G Ethernet IP core to the transceivers.

If implementing CAUI-10:

- CAUI-10 GTs have to be contiguous
- CAUI-10 must include two or four GTs from the quad in the same horizontal Clock Region (CR) as the 100G Ethernet IP
- CAUI-10 must be implemented within an Super Logic Region (SLR)

If implementing CAUI-4:

- CAUI-4 GTs have to be contiguous
- CAUI-4 must use the same CR or one above or below
- CAUI-4 all GTs must come from the same GT quad
- CAUI-4 is only supported in Lanes 1-4
- CAUI-4 must be implemented within an SLR

If implementing Runtime Selectable CAUI-10/CAUI-4, follow the preceding rules for both CAUI-10 and CAUI-4 rules.

TIP: *For transceiver selections outside of these rules, contact Xilinx support or your local FAE.*

See the *UltraScale Architecture Clocking Resource User Guide* (UG572) [Ref 5] for more information on Clock Region.

Dynamic Reconfiguration Port

The dynamic reconfiguration port (DRP) allows the dynamic change of attributes to the 100G Ethernet IP core. The DRP interface is a processor-friendly synchronous interface with an address bus (DRP_ADDR) and separated data buses for reading (DRP_DO) and writing (DRP_DI) configuration data to the CMAC block. An enable signal (DRP_EN), a read/write signal (DRP_WE), and a ready/valid signal (DRP_RDY) are the control signals that implement read and write operations, indicate that the operation is completion, or indicate the availability of data.

For the DRP to work, a clock must be provided to the DRP_CLK port. See the *Virtex UltraScale Architecture Data Sheet: DC and AC Switching Characteristics Data Sheet* (DS893) [Ref 3], for the maximum allowed clock frequency.

The CMAC block must be held in reset when you want to dynamically change the attributes through the DRP. That is, TX_RESET, RX_RESET, and the RX_SERDES_RESET[9:0] need to be asserted High.

DRP Write Operation

Figure 3-16 shows the DRP write operation timing diagram. New DRP operations can be initiated when the DRP_RDY signal is asserted.

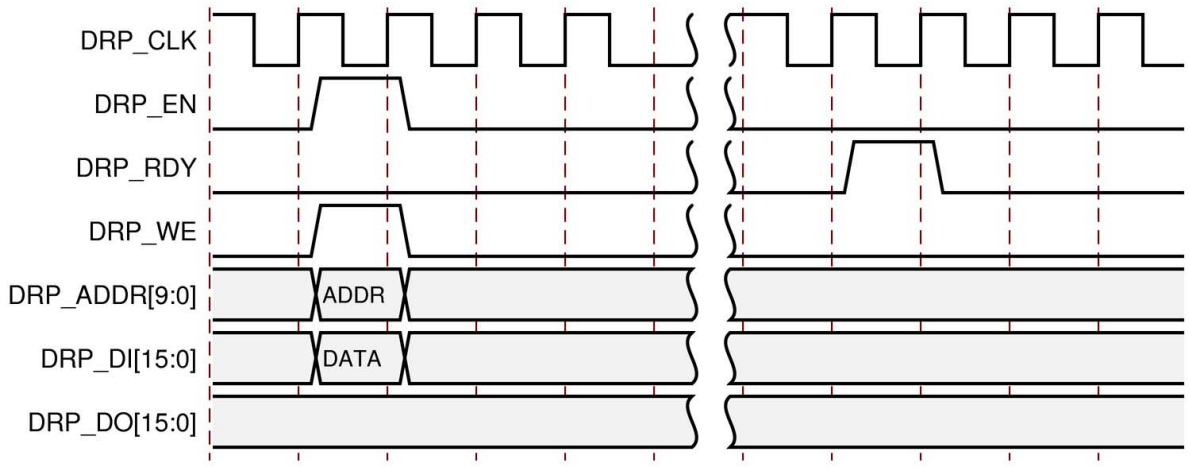

Figure 3-16: **DRP Write Operation Timing Diagram**

DRP Read Operation

Figure 3-17 shows the DRP read operation timing diagram. New DRP operations can be initiated when the DRP_RDY signal is asserted.

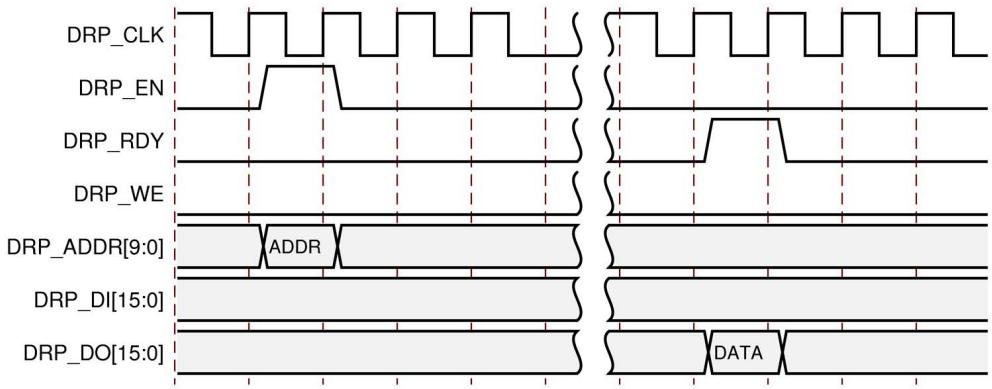

Figure 3-17: **DRP Read Operation Timing Diagram**

DRP Address Map of the CMAC Block

Table 3-7 lists the DRP map of the CMAC block sorted by address.

Table 3-7: **DRP Map of the CMAC Block**

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Encoding (Hex)	DRP Encoding (Hex)
0	0	R/W	CTL_TX_PTP_1STEP_ENABLE	FALSE	0
				TRUE	1
1	0	R/W	CTL_TX_IGNORE_FCS	FALSE	0
				TRUE	1
2	0	R/W	CTL_TX_FCS_INS_ENABLE	FALSE	0
				TRUE	1
8	[15:0]	R/W	CTL_TX_OPCODE_GPP[15:0]	0-FFFF	0-FFFF
9	[15:0]	R/W	CTL_TX_ETHERTYPE_PPP[15:0]]	0-FFFF	0-FFFF
A	[15:0]	R/W	CTL_TX_OPCODE_PPP[15:0]	0-FFFF	0-FFFF
10	[15:0]	R/W	CTL_TX_VL_LENGTH_MINUS1[15:0]	0-FFFF	0-FFFF
18	[15:0]	R/W	CTL_TX_SA_GPP[15:0]	0-FFFF	0-FFFF
19	[15:0]	R/W	CTL_TX_SA_GPP[31:16]	0-FFFF	0-FFFF
1A	[15:0]	R/W	CTL_TX_SA_GPP[47:32]	0-FFFF	0-FFFF
20	[15:0]	R/W	CTL_TX_DA_PPP[15:0]	0-FFFF	0-FFFF
21	[15:0]	R/W	CTL_TX_DA_PPP[31:16]	0-FFFF	0-FFFF

Table 3-7: **DRP Map of the CMAC Block *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Encoding (Hex)	DRP Encoding (Hex)
22	[15:0]	R/W	CTL_TX_DA_PPP[47:32]	0-FFFF	0-FFFF
28	[15:0]	R/W	CTL_TX_SA_PPP[15:0]	0-FFFF	0-FFFF
29	[15:0]	R/W	CTL_TX_SA_PPP[31:16]	0-FFFF	0-FFFF
2A	[15:0]	R/W	CTL_TX_SA_PPP[47:32]	0-FFFF	0-FFFF
30	[15:0]	R/W	CTL_TX_DA_GPP[15:0]	0-FFFF	0-FFFF
31	[15:0]	R/W	CTL_TX_DA_GPP[31:16]	0-FFFF	0-FFFF
32	[15:0]	R/W	CTL_TX_DA_GPP[47:32]	0-FFFF	0-FFFF
38	[15:0]	R/W	CTL_TX_VL_MARKER_ID0[15:0]	0-FFFF	0-FFFF
39	[15:0]	R/W	CTL_TX_VL_MARKER_ID0[31:16]	0-FFFF	0-FFFF
3A	[15:0]	R/W	CTL_TX_VL_MARKER_ID0[47:32]	0-FFFF	0-FFFF
3B	[15:0]	R/W	CTL_TX_VL_MARKER_ID0[63:48]	0-FFFF	0-FFFF
40	[15:0]	R/W	CTL_TX_VL_MARKER_ID1[15:0]	0-FFFF	0-FFFF
41	[15:0]	R/W	CTL_TX_VL_MARKER_ID1[31:16]	0-FFFF	0-FFFF
42	[15:0]	R/W	CTL_TX_VL_MARKER_ID1[47:32]	0-FFFF	0-FFFF
43	[15:0]	R/W	CTL_TX_VL_MARKER_ID1[63:48]	0-FFFF	0-FFFF
48	[15:0]	R/W	CTL_TX_VL_MARKER_ID2[15:0]	0-FFFF	0-FFFF
49	[15:0]	R/W	CTL_TX_VL_MARKER_ID2[31:16]	0-FFFF	0-FFFF
4A	[15:0]	R/W	CTL_TX_VL_MARKER_ID2[47:32]	0-FFFF	0-FFFF
4B	[15:0]	R/W	CTL_TX_VL_MARKER_ID2[63:48]	0-FFFF	0-FFFF
50	[15:0]	R/W	CTL_TX_VL_MARKER_ID3[15:0]	0-FFFF	0-FFFF
51	[15:0]	R/W	CTL_TX_VL_MARKER_ID3[31:16]	0-FFFF	0-FFFF
52	[15:0]	R/W	CTL_TX_VL_MARKER_ID3[47:32]	0-FFFF	0-FFFF
53	[15:0]	R/W	CTL_TX_VL_MARKER_ID3[63:48]	0-FFFF	0-FFFF
58	[15:0]	R/W	CTL_TX_VL_MARKER_ID4[15:0]	0-FFFF	0-FFFF
59	[15:0]	R/W	CTL_TX_VL_MARKER_ID4[31:16]	0-FFFF	0-FFFF
5A	[15:0]	R/W	CTL_TX_VL_MARKER_ID4[47:32]	0-FFFF	0-FFFF
5B	[15:0]	R/W	CTL_TX_VL_MARKER_ID4[63:48]	0-FFFF	0-FFFF
60	[15:0]	R/W	CTL_TX_VL_MARKER_ID5[15:0]	0-FFFF	0-FFFF
61	[15:0]	R/W	CTL_TX_VL_MARKER_ID5[31:16]	0-FFFF	0-FFFF
62	[15:0]	R/W	CTL_TX_VL_MARKER_ID5[47:32]	0-FFFF	0-FFFF
63	[15:0]	R/W	CTL_TX_VL_MARKER_ID5[63:48]	0-FFFF	0-FFFF
68	[15:0]	R/W	CTL_TX_VL_MARKER_ID6[15:0]	0-FFFF	0-FFFF
69	[15:0]	R/W	CTL_TX_VL_MARKER_ID6[31:16]	0-FFFF	0-FFFF

Table 3-7: **DRP Map of the CMAC Block *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Encoding (Hex)	DRP Encoding (Hex)
6A	[15:0]	R/W	CTL_TX_VL_MARKER_ID6[47:32]	0-FFFF	0-FFFF
6B	[15:0]	R/W	CTL_TX_VL_MARKER_ID6[63:48]	0-FFFF	0-FFFF
70	[15:0]	R/W	CTL_TX_VL_MARKER_ID7[15:0]	0-FFFF	0-FFFF
71	[15:0]	R/W	CTL_TX_VL_MARKER_ID7[31:16]	0-FFFF	0-FFFF
72	[15:0]	R/W	CTL_TX_VL_MARKER_ID7[47:32]	0-FFFF	0-FFFF
73	[15:0]	R/W	CTL_TX_VL_MARKER_ID7[63:48]	0-FFFF	0-FFFF
78	[15:0]	R/W	CTL_TX_VL_MARKER_ID8[15:0]	0-FFFF	0-FFFF
79	[15:0]	R/W	CTL_TX_VL_MARKER_ID8[31:16]	0-FFFF	0-FFFF
7A	[15:0]	R/W	CTL_TX_VL_MARKER_ID8[47:32]	0-FFFF	0-FFFF
7B	[15:0]	R/W	CTL_TX_VL_MARKER_ID8[63:48]	0-FFFF	0-FFFF
80	[15:0]	R/W	CTL_TX_VL_MARKER_ID9[15:0]	0-FFFF	0-FFFF
81	[15:0]	R/W	CTL_TX_VL_MARKER_ID9[31:16]	0-FFFF	0-FFFF
82	[15:0]	R/W	CTL_TX_VL_MARKER_ID9[47:32]	0-FFFF	0-FFFF
83	[15:0]	R/W	CTL_TX_VL_MARKER_ID9[63:48]	0-FFFF	0-FFFF
88	[15:0]	R/W	CTL_TX_VL_MARKER_ID10[15:0]	0-FFFF	0-FFFF
89	[15:0]	R/W	CTL_TX_VL_MARKER_ID10[31:16]	0-FFFF	0-FFFF
8A	[15:0]	R/W	CTL_TX_VL_MARKER_ID10[47:32]	0-FFFF	0-FFFF
8B	[15:0]	R/W	CTL_TX_VL_MARKER_ID10[63:48]	0-FFFF	0-FFFF
90	[15:0]	R/W	CTL_TX_VL_MARKER_ID11[15:0]	0-FFFF	0-FFFF
91	[15:0]	R/W	CTL_TX_VL_MARKER_ID11[31:16]	0-FFFF	0-FFFF
92	[15:0]	R/W	CTL_TX_VL_MARKER_ID11[47:32]	0-FFFF	0-FFFF
93	[15:0]	R/W	CTL_TX_VL_MARKER_ID11[63:48]	0-FFFF	0-FFFF
98	[15:0]	R/W	CTL_TX_VL_MARKER_ID12[15:0]	0-FFFF	0-FFFF
99	[15:0]	R/W	CTL_TX_VL_MARKER_ID12[31:16]	0-FFFF	0-FFFF
9A	[15:0]	R/W	CTL_TX_VL_MARKER_ID12[47:32]	0-FFFF	0-FFFF
9B	[15:0]	R/W	CTL_TX_VL_MARKER_ID12[63:48]	0-FFFF	0-FFFF
A0	[15:0]	R/W	CTL_TX_VL_MARKER_ID13[15:0]	0-FFFF	0-FFFF
A1	[15:0]	R/W	CTL_TX_VL_MARKER_ID13[31:16]	0-FFFF	0-FFFF
A2	[15:0]	R/W	CTL_TX_VL_MARKER_ID13[47:32]	0-FFFF	0-FFFF
A3	[15:0]	R/W	CTL_TX_VL_MARKER_ID13[63:48]	0-FFFF	0-FFFF
A8	[15:0]	R/W	CTL_TX_VL_MARKER_ID14[15:0]	0-FFFF	0-FFFF
A9	[15:0]	R/W	CTL_TX_VL_MARKER_ID14[31:16]	0-FFFF	0-FFFF
AA	[15:0]	R/W	CTL_TX_VL_MARKER_ID14[47:32]	0-FFFF	0-FFFF

Table 3-7: **DRP Map of the CMAC Block *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Encoding (Hex)	DRP Encoding (Hex)
AB	[15:0]	R/W	CTL_TX_VL_MARKER_ID14[63:48]	0-FFFF	0-FFFF
B0	[15:0]	R/W	CTL_TX_VL_MARKER_ID15[15:0]	0-FFFF	0-FFFF
B1	[15:0]	R/W	CTL_TX_VL_MARKER_ID15[31:16]	0-FFFF	0-FFFF
B2	[15:0]	R/W	CTL_TX_VL_MARKER_ID15[47:32]	0-FFFF	0-FFFF
B3	[15:0]	R/W	CTL_TX_VL_MARKER_ID15[63:48]	0-FFFF	0-FFFF
B8	[15:0]	R/W	CTL_TX_VL_MARKER_ID16[15:0]	0-FFFF	0-FFFF
B9	[15:0]	R/W	CTL_TX_VL_MARKER_ID16[31:16]	0-FFFF	0-FFFF
BA	[15:0]	R/W	CTL_TX_VL_MARKER_ID16[47:32]	0-FFFF	0-FFFF
BB	[15:0]	R/W	CTL_TX_VL_MARKER_ID16[63:48]	0-FFFF	0-FFFF
C0	[15:0]	R/W	CTL_TX_VL_MARKER_ID17[15:0]	0-FFFF	0-FFFF
C1	[15:0]	R/W	CTL_TX_VL_MARKER_ID17[31:16]	0-FFFF	0-FFFF
C2	[15:0]	R/W	CTL_TX_VL_MARKER_ID17[47:32]	0-FFFF	0-FFFF
C3	[15:0]	R/W	CTL_TX_VL_MARKER_ID17[63:48]	0-FFFF	0-FFFF
C8	[15:0]	R/W	CTL_TX_VL_MARKER_ID18[15:0]	0-FFFF	0-FFFF
C9	[15:0]	R/W	CTL_TX_VL_MARKER_ID18[31:16]	0-FFFF	0-FFFF
CA	[15:0]	R/W	CTL_TX_VL_MARKER_ID18[47:32]	0-FFFF	0-FFFF
CB	[15:0]	R/W	CTL_TX_VL_MARKER_ID18[63:48]	0-FFFF	0-FFFF
D0	[15:0]	R/W	CTL_TX_VL_MARKER_ID19[15:0]	0-FFFF	0-FFFF
D1	[15:0]	R/W	CTL_TX_VL_MARKER_ID19[31:16]	0-FFFF	0-FFFF
D2	[15:0]	R/W	CTL_TX_VL_MARKER_ID19[47:32]	0-FFFF	0-FFFF
D3	[15:0]	R/W	CTL_TX_VL_MARKER_ID19[63:48]	0-FFFF	0-FFFF
D8	0	R/W	CTL_RX_CHECK_PREAMBLE	FALSE	0
				TRUE	1
D9	0	R/W	CTL_RX_IGNORE_FCS	FALSE	0
				TRUE	1
DA	0	R/W	CTL_RX_FORWARD_CONTROL	FALSE	0
				TRUE	1
DB	0	R/W	CTL_RX_DELETE_FCS	FALSE	0
				TRUE	1
E0	0	R/W	CTL_RX_CHECK_ACK	FALSE	0
				TRUE	1
E1	0	R/W	CTL_RX_CHECK_SFD	FALSE	0
				TRUE	1

Table 3-7: **DRP Map of the CMAC Block *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Encoding (Hex)	DRP Encoding (Hex)
E2	0	R/W	CTL_RX_PROCESS_LFI	FALSE	0
				TRUE	1
E8	[7:0]	R/W	CTL_RX_MIN_PACKET_LEN[7:0]	0-FF	40-FF
E9	[14:0]	R/W	CTL_RX_MAX_PACKET_LEN[14:0]	0-7FFF	0-3FFF
EA	[15:0]	R/W	CTL_TX_ETHERTYPE_GPP[15:0]	0-FFFF	0-FFFF
EB	[15:0]	R/W	CTL_RX_OPCODE_GPP[15:0]	0-FFFF	0-FFFF
F0	[15:0]	R/W	CTL_RX_OPCODE_MAX_GCP[15:0]	0-FFFF	0-FFFF
F1	[15:0]	R/W	CTL_RX_ETYPE_PPP[15:0]	0-FFFF	0-FFFF
F2	[15:0]	R/W	CTL_RX_ETYPE_GCP[15:0]	0-FFFF	0-FFFF
F3	[15:0]	R/W	CTL_RX_VL_LENGTH_MINUS1[15:0]	0-FFFF	0-FFFF
F8	[15:0]	R/W	CTL_RX_OPCODE_MAX_PCP[15:0]	0-FFFF	0-FFFF
F9	[15:0]	R/W	CTL_RX_OPCODE_MIN_GCP[15:0]	0-FFFF	0-FFFF
FA	[15:0]	R/W	CTL_RX_ETYPE_GPP[15:0]	0-FFFF	0-FFFF
FB	[15:0]	R/W	CTL_RX_OPCODE_MIN_PCP[15:0]	0-FFFF	0-FFFF
100	[15:0]	R/W	CTL_RX_ETYPE_PCP[15:0]	0-FFFF	0-FFFF
101	[15:0]	R/W	CTL_RX_OPCODE_PPP[15:0]	0-FFFF	0-FFFF
108	[15:0]	R/W	CTL_RX_PAUSE_DA_MCAST[15:0]	0-FFFF	0-FFFF
109	[15:0]	R/W	CTL_RX_PAUSE_DA_MCAST[31:16]	0-FFFF	0-FFFF
10A	[15:0]	R/W	CTL_RX_PAUSE_DA_MCAST[47:32]	0-FFFF	0-FFFF
110	[15:0]	R/W	CTL_RX_PAUSE_DA_UCAST[15:0]	0-FFFF	0-FFFF
111	[15:0]	R/W	CTL_RX_PAUSE_DA_UCAST[31:16]	0-FFFF	0-FFFF
112	[15:0]	R/W	CTL_RX_PAUSE_DA_UCAST[47:32]	0-FFFF	0-FFFF
118	[15:0]	R/W	CTL_RX_PAUSE_SA[15:0]	0-FFFF	0-FFFF
119	[15:0]	R/W	CTL_RX_PAUSE_SA[31:16]	0-FFFF	0-FFFF
11A	[15:0]	R/W	CTL_RX_PAUSE_SA[47:32]	0-FFFF	0-FFFF
120	[15:0]	R/W	CTL_RX_VL_MARKER_ID0[15:0]	0-FFFF	0-FFFF
121	[15:0]	R/W	CTL_RX_VL_MARKER_ID0[31:16]	0-FFFF	0-FFFF
122	[15:0]	R/W	CTL_RX_VL_MARKER_ID0[47:32]	0-FFFF	0-FFFF
123	[15:0]	R/W	CTL_RX_VL_MARKER_ID0[63:48]	0-FFFF	0-FFFF
128	[15:0]	R/W	CTL_RX_VL_MARKER_ID1[15:0]	0-FFFF	0-FFFF
129	[15:0]	R/W	CTL_RX_VL_MARKER_ID1[31:16]	0-FFFF	0-FFFF
12A	[15:0]	R/W	CTL_RX_VL_MARKER_ID1[47:32]	0-FFFF	0-FFFF
12B	[15:0]	R/W	CTL_RX_VL_MARKER_ID1[63:48]	0-FFFF	0-FFFF

Table 3-7: **DRP Map of the CMAC Block *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Encoding (Hex)	DRP Encoding (Hex)
130	[15:0]	R/W	CTL_RX_VL_MARKER_ID2[15:0]	0-FFFF	0-FFFF
131	[15:0]	R/W	CTL_RX_VL_MARKER_ID2[31:16]	0-FFFF	0-FFFF
132	[15:0]	R/W	CTL_RX_VL_MARKER_ID2[47:32]	0-FFFF	0-FFFF
133	[15:0]	R/W	CTL_RX_VL_MARKER_ID2[63:48]	0-FFFF	0-FFFF
138	[15:0]	R/W	CTL_RX_VL_MARKER_ID3[15:0]	0-FFFF	0-FFFF
139	[15:0]	R/W	CTL_RX_VL_MARKER_ID3[31:16]	0-FFFF	0-FFFF
13A	[15:0]	R/W	CTL_RX_VL_MARKER_ID3[47:32]	0-FFFF	0-FFFF
13B	[15:0]	R/W	CTL_RX_VL_MARKER_ID3[63:48]	0-FFFF	0-FFFF
140	[15:0]	R/W	CTL_RX_VL_MARKER_ID4[15:0]	0-FFFF	0-FFFF
141	[15:0]	R/W	CTL_RX_VL_MARKER_ID4[31:16]	0-FFFF	0-FFFF
142	[15:0]	R/W	CTL_RX_VL_MARKER_ID4[47:32]	0-FFFF	0-FFFF
143	[15:0]	R/W	CTL_RX_VL_MARKER_ID4[63:48]	0-FFFF	0-FFFF
148	[15:0]	R/W	CTL_RX_VL_MARKER_ID5[15:0]	0-FFFF	0-FFFF
149	[15:0]	R/W	CTL_RX_VL_MARKER_ID5[31:16]	0-FFFF	0-FFFF
14A	[15:0]	R/W	CTL_RX_VL_MARKER_ID5[47:32]	0-FFFF	0-FFFF
14B	[15:0]	R/W	CTL_RX_VL_MARKER_ID5[63:48]	0-FFFF	0-FFFF
150	[15:0]	R/W	CTL_RX_VL_MARKER_ID6[15:0]	0-FFFF	0-FFFF
151	[15:0]	R/W	CTL_RX_VL_MARKER_ID6[31:16]	0-FFFF	0-FFFF
152	[15:0]	R/W	CTL_RX_VL_MARKER_ID6[47:32]	0-FFFF	0-FFFF
153	[15:0]	R/W	CTL_RX_VL_MARKER_ID6[63:48]	0-FFFF	0-FFFF
158	[15:0]	R/W	CTL_RX_VL_MARKER_ID7[15:0]	0-FFFF	0-FFFF
159	[15:0]	R/W	CTL_RX_VL_MARKER_ID7[31:16]	0-FFFF	0-FFFF
15A	[15:0]	R/W	CTL_RX_VL_MARKER_ID7[47:32]	0-FFFF	0-FFFF
15B	[15:0]	R/W	CTL_RX_VL_MARKER_ID7[63:48]	0-FFFF	0-FFFF
160	[15:0]	R/W	CTL_RX_VL_MARKER_ID8[15:0]	0-FFFF	0-FFFF
161	[15:0]	R/W	CTL_RX_VL_MARKER_ID8[31:16]	0-FFFF	0-FFFF
162	[15:0]	R/W	CTL_RX_VL_MARKER_ID8[47:32]	0-FFFF	0-FFFF
163	[15:0]	R/W	CTL_RX_VL_MARKER_ID8[63:48]	0-FFFF	0-FFFF
168	[15:0]	R/W	CTL_RX_VL_MARKER_ID9[15:0]	0-FFFF	0-FFFF
169	[15:0]	R/W	CTL_RX_VL_MARKER_ID9[31:16]	0-FFFF	0-FFFF
16A	[15:0]	R/W	CTL_RX_VL_MARKER_ID9[47:32]	0-FFFF	0-FFFF
16B	[15:0]	R/W	CTL_RX_VL_MARKER_ID9[63:48]	0-FFFF	0-FFFF
170	[15:0]	R/W	CTL_RX_VL_MARKER_ID10[15:0]	0-FFFF	0-FFFF

Table 3-7: **DRP Map of the CMAC Block *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Encoding (Hex)	DRP Encoding (Hex)
171	[15:0]	R/W	CTL_RX_VL_MARKER_ID10[31:16]	0-FFFF	0-FFFF
172	[15:0]	R/W	CTL_RX_VL_MARKER_ID10[47:32]	0-FFFF	0-FFFF
173	[15:0]	R/W	CTL_RX_VL_MARKER_ID10[63:48]	0-FFFF	0-FFFF
178	[15:0]	R/W	CTL_RX_VL_MARKER_ID11[15:0]	0-FFFF	0-FFFF
179	[15:0]	R/W	CTL_RX_VL_MARKER_ID11[31:16]	0-FFFF	0-FFFF
17A	[15:0]	R/W	CTL_RX_VL_MARKER_ID11[47:32]	0-FFFF	0-FFFF
17B	[15:0]	R/W	CTL_RX_VL_MARKER_ID11[63:48]	0-FFFF	0-FFFF
180	[15:0]	R/W	CTL_RX_VL_MARKER_ID12[15:0]	0-FFFF	0-FFFF
181	[15:0]	R/W	CTL_RX_VL_MARKER_ID12[31:16]	0-FFFF	0-FFFF
182	[15:0]	R/W	CTL_RX_VL_MARKER_ID12[47:32]	0-FFFF	0-FFFF
183	[15:0]	R/W	CTL_RX_VL_MARKER_ID12[63:48]	0-FFFF	0-FFFF
188	[15:0]	R/W	CTL_RX_VL_MARKER_ID13[15:0]	0-FFFF	0-FFFF
189	[15:0]	R/W	CTL_RX_VL_MARKER_ID13[31:16]	0-FFFF	0-FFFF
18A	[15:0]	R/W	CTL_RX_VL_MARKER_ID13[47:32]	0-FFFF	0-FFFF
18B	[15:0]	R/W	CTL_RX_VL_MARKER_ID13[63:48]	0-FFFF	0-FFFF
190	[15:0]	R/W	CTL_RX_VL_MARKER_ID14[15:0]	0-FFFF	0-FFFF
191	[15:0]	R/W	CTL_RX_VL_MARKER_ID14[31:16]	0-FFFF	0-FFFF
192	[15:0]	R/W	CTL_RX_VL_MARKER_ID14[47:32]	0-FFFF	0-FFFF
193	[15:0]	R/W	CTL_RX_VL_MARKER_ID14[63:48]	0-FFFF	0-FFFF
198	[15:0]	R/W	CTL_RX_VL_MARKER_ID15[15:0]	0-FFFF	0-FFFF
199	[15:0]	R/W	CTL_RX_VL_MARKER_ID15[31:16]	0-FFFF	0-FFFF
19A	[15:0]	R/W	CTL_RX_VL_MARKER_ID15[47:32]	0-FFFF	0-FFFF
19B	[15:0]	R/W	CTL_RX_VL_MARKER_ID15[63:48]	0-FFFF	0-FFFF
1A0	[15:0]	R/W	CTL_RX_VL_MARKER_ID16[15:0]	0-FFFF	0-FFFF
1A1	[15:0]	R/W	CTL_RX_VL_MARKER_ID16[31:16]	0-FFFF	0-FFFF
1A2	[15:0]	R/W	CTL_RX_VL_MARKER_ID16[47:32]	0-FFFF	0-FFFF
1A3	[15:0]	R/W	CTL_RX_VL_MARKER_ID16[63:48]	0-FFFF	0-FFFF
1A8	[15:0]	R/W	CTL_RX_VL_MARKER_ID17[15:0]	0-FFFF	0-FFFF
1A9	[15:0]	R/W	CTL_RX_VL_MARKER_ID17[31:16]	0-FFFF	0-FFFF
1AA	[15:0]	R/W	CTL_RX_VL_MARKER_ID17[47:32]	0-FFFF	0-FFFF
1AB	[15:0]	R/W	CTL_RX_VL_MARKER_ID17[63:48]	0-FFFF	0-FFFF
1B0	[15:0]	R/W	CTL_RX_VL_MARKER_ID18[15:0]	0-FFFF	0-FFFF
1B1	[15:0]	R/W	CTL_RX_VL_MARKER_ID18[31:16]	0-FFFF	0-FFFF

Table 3-7: **DRP Map of the CMAC Block *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Encoding (Hex)	DRP Encoding (Hex)
1B2	[15:0]	R/W	CTL_RX_VL_MARKER_ID18[47:32]	0-FFFF	0-FFFF
1B3	[15:0]	R/W	CTL_RX_VL_MARKER_ID18[63:48]	0-FFFF	0-FFFF
1B8	[15:0]	R/W	CTL_RX_VL_MARKER_ID19[15:0]	0-FFFF	0-FFFF
1B9	[15:0]	R/W	CTL_RX_VL_MARKER_ID19[31:16]	0-FFFF	0-FFFF
1BA	[15:0]	R/W	CTL_RX_VL_MARKER_ID19[47:32]	0-FFFF	0-FFFF
1BB	[15:0]	R/W	CTL_RX_VL_MARKER_ID19[63:48]	0-FFFF	0-FFFF
1C0	0	R/W	TEST_MODE_PIN_CHAR	FALSE	0
				TRUE	1
1C1	0	R/W	CTL_PTP_TRANSPCLK_MODE	FALSE	0
				TRUE	1
1C2	0	R/W	CTL_TEST_MODE_PIN_CHAR	FALSE	0
				TRUE	1
1C8	[10:0]	R/W	CTL_TX_PTP_LATENCY_ADJUST[10:0]	0-7FF	0-7FF

Design Flow Steps

This chapter describes customizing and generating the core, constraining the core, and the simulation, synthesis and implementation steps that are specific to this IP core. More detailed information about the standard Vivado® design flows and the IP integrator can be found in the following Vivado Design Suite user guides:

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 6]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 7]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 8]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 9]

Customizing and Generating the Core

This section includes information about using Xilinx tools to customize and generate the core in the Vivado Design Suite.

If you are customizing and generating the core in the Vivado IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 6] for detailed information. IP integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values do change, see the description of the parameter in this chapter. To view the parameter value, run the `validate_bd_design` command in the Tcl Console.

You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

1. Select the IP from the Vivado IP catalog.
2. Double-click the selected IP or select the Customize IP command from the toolbar or right-click menu.

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 7] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 8].

Note: Figures in this chapter are illustrations of the Vivado Integrated Design Environment (IDE). This layout might vary from the current version.

When the 100G Ethernet IP is selected from the IP catalog, a window displays showing the different available configurations. These are organized in various tabs for better readability and configuration purposes. The details related to these tabs follow:

General Configuration Tab

The General Configuration tab configures 100G Ethernet IP core features. See Figure 4-1.

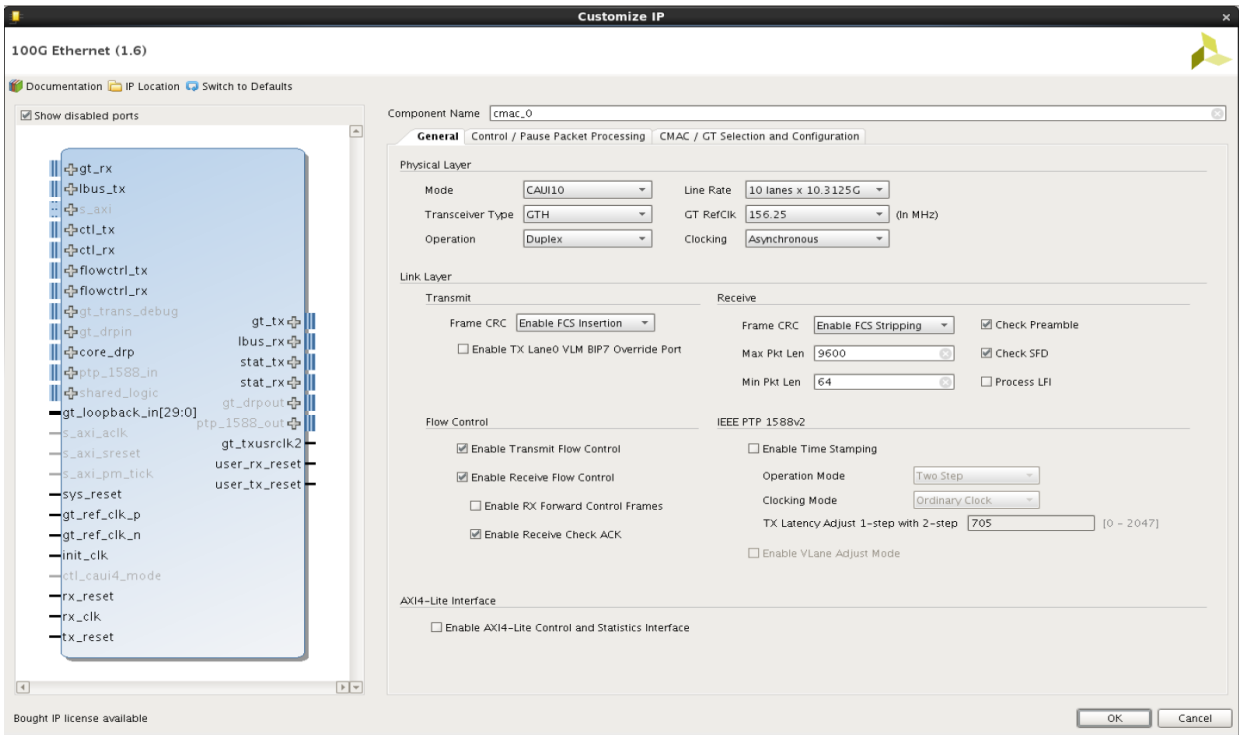

Figure 4-1: **General Configuration Tab**

Table 4-1 describes the General Configuration tab options.

Table 4-1: **General Configuration Tab**

Parameter	Description	Default Value	Range
Physical Layer			
Mode	100G Ethernet Mode	0 (CAUI-10)	0 - CAUI-10 1 - CAUI-4 2 - Runtime Selectable
Line Rate	Number of lanes and line rate	10 lanes x 10.3125 Gb/s	10 lanes x 10.3125 Gb/s 4 lanes x 25.7812 Gb/s
Transceiver Type	Transceiver Type	GTH	GTH GTY
GT RefClk	Reference clock for the GTs used	156.25 MHz	103.12 MHz 128.90 MHz 161.13 MHz 156.25 MHz 161.13 MHz 195.31 MHz 201.41 MHz 206.25 MHz 257.81 MHz 309.37 MHz 312.50 MHz 322.266 MHz
Operation	Operating mode	Duplex	Simplex TX Simplex RX Duplex
Clocking	Clocking mode	Asynchronous	Synchronous Asynchronous
Link Layer — Transmit			
Frame CRC(1)	TX Frame CRC checking	Enable FCS insertion	Enable FCS Insertion Disable FCS Insertion
VLM BIP7 Override (1)	TX Lane0 VLM BIP7 Override	0	0: Disabled 1: Enabled
Link Layer — Receive			
Frame CRC(2)	RX Frame CRC checking	Enable FCS stripping	Enable FCS stripping Disable FCS stripping
Max Pkt Len(2)	Maximum Packet Length	9,600	0x00FF to 0x3FFF
Min Pkt Len(2)	Minimum Packet Length	64	0x40 to 0xFF
Check Preamble(2)	Check Preamble	1	0: Disabled 1: Enabled

Table 4-1: **General Configuration Tab *(Cont'd)***

Parameter	Description	Default Value	Range
Check SFD[2]	Check SFD	1	0: Disabled 1: Enabled
Process LFI[2]	RX Process LFI	0	0: Disabled 1: Enabled
Link Layer — Flow Control			
Enable Transmit Flow Control	Enable Transmit Flow Control	1	0: Disabled 1: Enabled
Enable Receive Flow Control	Enable Receive Flow Control	1	0: Disabled 1: Enabled
Enable RX Forward Control Frames[3]	Forward Control Frames	1	0: Disabled 1: Enabled
Enable Receive Check ACK	Enable Receive Check ACK	0	0: Disabled 1: Enabled
Link Layer — IEEE 1588			
Enable Time Stamping	Enables timestamping	0	0: Disabled 1: Enabled
Operation Mode[4]	Sets the operation mode Unavailable when Enable Time Stamping = 0	Two Step	One Step Two Step Both
Clocking Mode[5]	Clocking Mode	Ordinary Clock	Ordinary Clock Transparent Clock
TX Latency Adjust 1-step with 2-step	Unavailable when Enable Time Stamping = 0 with default value as 0. Only available when Operation Mode is "Both". If the clocking mode is ordinary clock, default value should be 705. If clocking mode is transparent clock, default value should be 803.	0	0 to 2,047
Enable VLane Adjust Mode	Unavailable when Enable Time Stamping = 0. Unavailable when operation mode is "Two Step".	0	0- Disabled 1- Enabled

Table 4-1: **General Configuration Tab *(Cont'd)***

Parameter	Description	Default Value	Range
AXI4-Lite Interface			
Enable AXI4-Lite Interface	When you enable, the AXI4-Lite interface is provided in the core.	0	0: Disabled 1: Enabled

Notes:

1. Requires TX to be enabled. Operation = Simplex TX or Duplex modes only.
2. Requires RX to be enabled. Operation = Simplex RX or Duplex modes only.
3. Option is disabled when receive flow control is disabled.
4. This release supports only one-step and two-step time stamping.
5. This release only supports Ordinary Clock clocking mode.

Control Pause/Packet Processing Tab

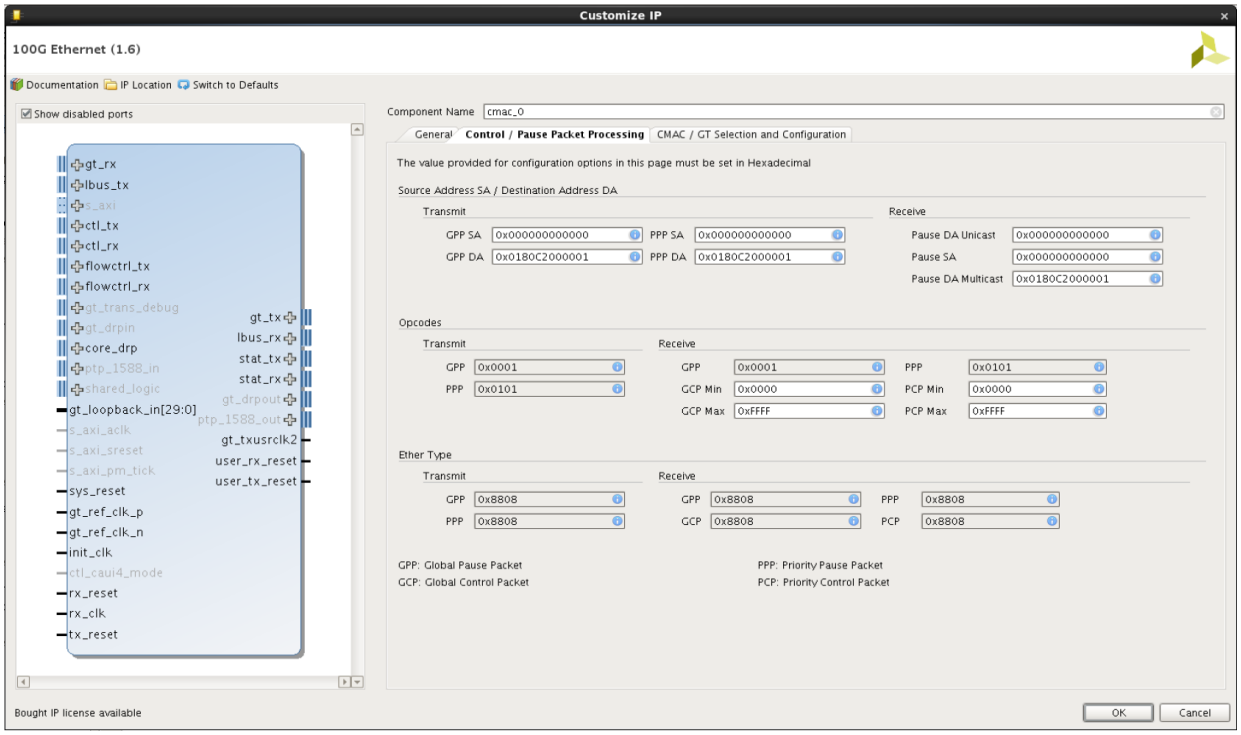

Figure 4-2: **Control Pause/Packet Processing Tab**

Table 4-2 describes the Control/Pause Packet Processing tab options.

Table 4-2: **Control/Pause Packet Processing**

Parameter	Description	Default Value	Range
Source Address (SA)/ Destination Address (DA) — Transmit			
TX GPP SA[47:0](1)	Transmit Global Pause Packet Source Address	0x000000000000	0x000000000000 - 0xFFFFFFFFFFFF
TX GPP DA[47:0](1)	Transmit Global Pause Packet Destination Address	0x0180C2000001	0x000000000000 - 0xFFFFFFFFFFFF
TX PPP SA[47:0](1)	Transmit Priority Pause Packet Source Address	0x000000000000	0x000000000000 - 0xFFFFFFFFFFFF
TX PPP DA[47:0](1)	Transmit Priority Pause Packet Destination Address	0x0180C2000001	0x000000000000 - 0xFFFFFFFFFFFF
Source Address (SA)/ Destination Address (DA) — Receive			
RX Pause DA Unicast[47:0](2)	Receive Pause Destination Address Unicast	0x000000000000	0x000000000000 - 0xFFFFFFFFFFFF
RX Pause SA[47:0](2)	Receive Source Address	0x000000000000	0x000000000000 - 0xFFFFFFFFFFFF
RX Pause DA Multicast[47:0](2)	Receive Pause Destination Address Multicast	0x0180C2000001	0x000000000000 - 0xFFFFFFFFFFFF
Opcodes — Transmit			
TX Opcode GPP[15:0](1)	Transmit Opcode for Global Pause Packet	0x0001	
TX Opcode PPP[15:0](1)	Transmit Opcode for Priority Pause Packet	0x0101	
Opcodes — Receive			
RX Opcode GPP[15:0](1)	Receive Opcode for Global Pause Packet	0x0001	
RX Opcode GCP[15:0] Min (2)	Receive Minimum Opcode for Global Control Packet	0x0000	0x0000 - 0xFFFF
RX Opcode GCP[15:0] Max (2)	Receive Maximum Opcode for Global Control Packet	0xFFFF	0x0000 - 0xFFFF
RX Opcode PPP[15:0](2)	Receive Opcode for Priority Pause Packet	0x0101	
RX Opcode PCP[15:0] Min (2)	Receive Minimum Opcode for Priority Control Packet	0x0000	0x0000 - 0xFFFF
RX Opcode PCP[15:0] Max (2)	Receive Maximum Opcode for Priority Control Packet	0xFFFF	0x0000 - 0xFFFF

Table 4-2: **Control/Pause Packet Processing *(Cont'd)***

Parameter	Description	Default Value	Range
EtherType — Transmit			
TX EtherType GPP[15:0](1)	Transmit EtherType for Global Pause Packet	0x8808	
TX EtherType PPP[15:0](1)	Transmit EtherType for Priority Pause Packet	0x8808	
EtherType — Receive			
RX EtherType GPP[15:0](2)	Receive EtherType for Global Pause Packet	0x8808	
RX EtherType GCP[15:0](3)	Receive EtherType for Global Control Packet	0x8808	
RX EtherType PPP[15:0](2)	Receive EtherType for Priority Pause Packet	0x8808	
RX EtherType PCP[15:0](3)	Receive EtherType for Priority Control Packet	0x8808	

Notes:

1. TX flow control must be enabled to use this feature.
2. RX flow control must be enabled to use this feature.
3. RX must be enabled to use this feature.

GT Selections and Configuration Tab

This configures transceiver resources. See Figure 4-3, The window loads with default values that are pre-populated.

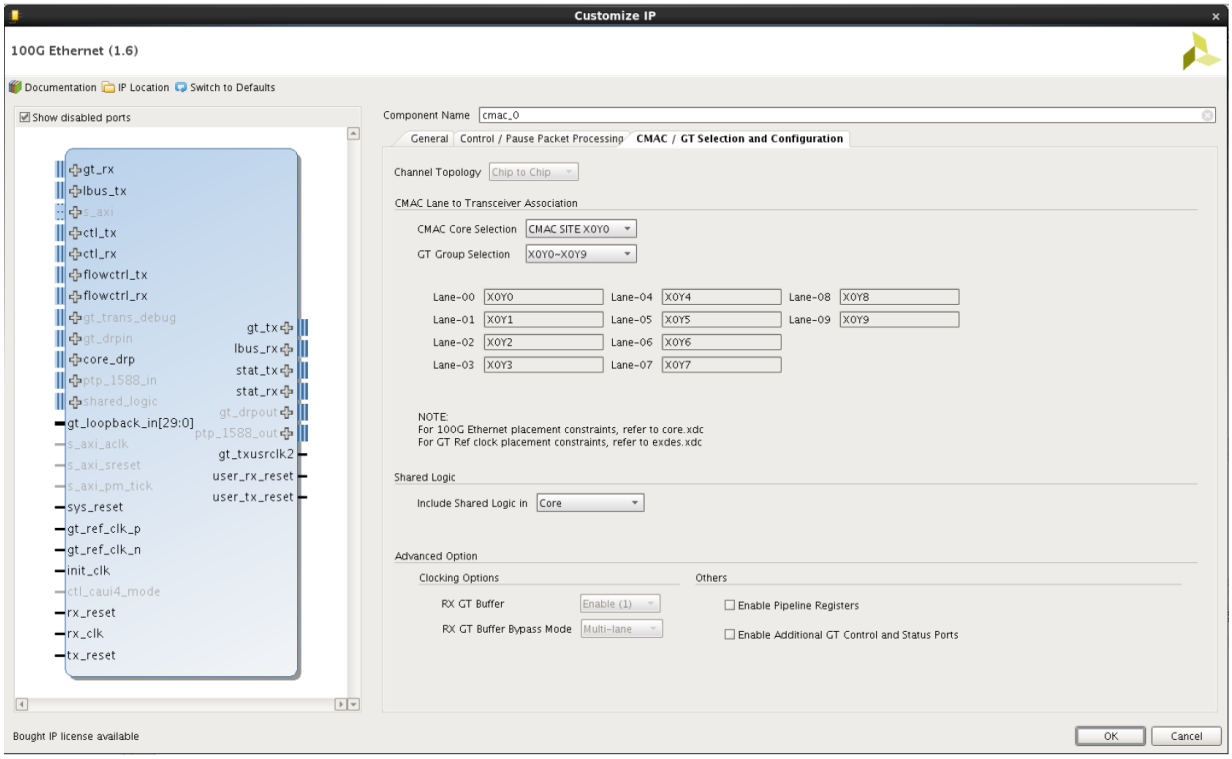

Figure 4-3: **GT Selections and Configuration Tab**

Table 4-3 describes the GT Selections and Configuration tab options.

Table 4-3: **GT Selections and Configuration**

Parameter	Description	Default Value	Range
Channel Topology	Type of channel used.	Chip to Chip	Chip to Chip Backplane
CMAC Lane to Transceiver Association			
CMAC Core Selection	Select 100G Ethernet IP core location		Based on the FPGA, part number, CMAC Mode and GT type selected, all the usable/configurable 100G Ethernet IP cores for that particular device/package will be listed.
GT Group Selection Lane-00 to Lane-10	Select GT location	The best combination of transceivers will be auto filled based on the 100G Ethernet IP core location chosen	Based on the Mode selection (CAUI-10, CAUI-4 or Runtime Selectable), the GT selection guidelines are followed. See Transceiver Selection Rules in Chapter 3 for more details.
Shared Logic			
Include Shared Logic in	Determines the location of the transceiver shared logic	Core	Core Example Design
Advanced Options			
Clocking Options			
RX GT Buffer	Enable Buffer on RX	1	0-Bypass 1-Enable
RX GT Buffer Bypass Mode		1	0-Single-lane 1-Multi-lane
Others			
Enable Pipeline Register	Enable insertion of pipeline registers in the FPGA fabric between the 100G Ethernet IP core and the GTH or GTY transceiver interface to ease timing	0	0-Disable 1-Enable
Enable Additional GT Control and Status Ports	Enable Additional GT Control and Status Ports	0	0-Disable 1-Enable

User Parameters

Table 4-4 shows the relationship between the fields in the Vivado IDE and the User Parameters (which can be viewed in the Tcl Console).

Table 4-4: **Vivado IDE Parameter to User Parameter Relationship**

Vivado IDE Parameter/Value[1]	User Parameter/Value[1]	Default Value
MODE	CMAC_CAUI4_MODE	CAUI10
CAUI4		
CAUI10		
Line Rate	NUM_LANES	0
10x10.3125G		
4x25.7812G		
Transceiver Type	GT_TYPE	GTH
GTH		
GTY		
GT RefClk	GT_REF_CLK_FREQ	156.25
103.12 128.90 161.13 156.25 161.13 195.31 201.41 206.25 257.81 309.37 312.50 322.266		

Notes:

1. Parameter values are listed in the table where the Vivado IDE parameter value differs from the user parameter value. Such values are shown in this table as indented below the associated parameter.

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 7].

Constraining the Core

This section contains information about constraining the core in the Vivado Design Suite.

Required Constraints

The UltraScale™ architecture integrated 100 Gb/s Ethernet IP core solution requires the specification of timing and other physical implementation constraints to meet the specified performance requirements. These constraints are provided in a Xilinx® Device Constraints (XDC) file. Pinouts and hierarchy names in the generated XDC correspond to the provided example design of the 100G Ethernet IP core.

To achieve consistent implementation results, an XDC containing these original, unmodified constraints must be used when a design is run through the Xilinx tools. For additional details on the definition and use of an XDC or specific constraints, see the *Vivado Design Suite User Guide: Using Constraints* (UG903) [Ref 10].

Constraints provided in the 100G Ethernet IP core have been verified through implementation and provide consistent results. Constraints can be modified, but modifications should only be made with a thorough understanding of the effect of each constraint.

Device, Package, and Speed Grade Selections

This section is not applicable for this IP core.

Clock Frequencies

This section is not applicable for this IP core.

Clock Management

This section is not applicable for this IP core.

Clock Placement

This section is not applicable for this IP core.

Banking

This section is not applicable for this IP core.

Transceiver Placement

This section is not applicable for this IP core.

I/O Standard and Placement

This section is not applicable for this IP core.

Simulation

For comprehensive information about Vivado simulation components, as well as information about using supported third-party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 9].

For information regarding simulating the example design, see Simulating the Example Design in Chapter 5.

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 7].

For information regarding synthesizing and implementing the example design, see Synthesizing and Implementing the Example Design in Chapter 5.

Example Design

Overview

This chapter briefly explains the 100G Ethernet IP core example design and the various test scenarios implemented within the example design.

Figure 5-1 shows the example design hierarchy.

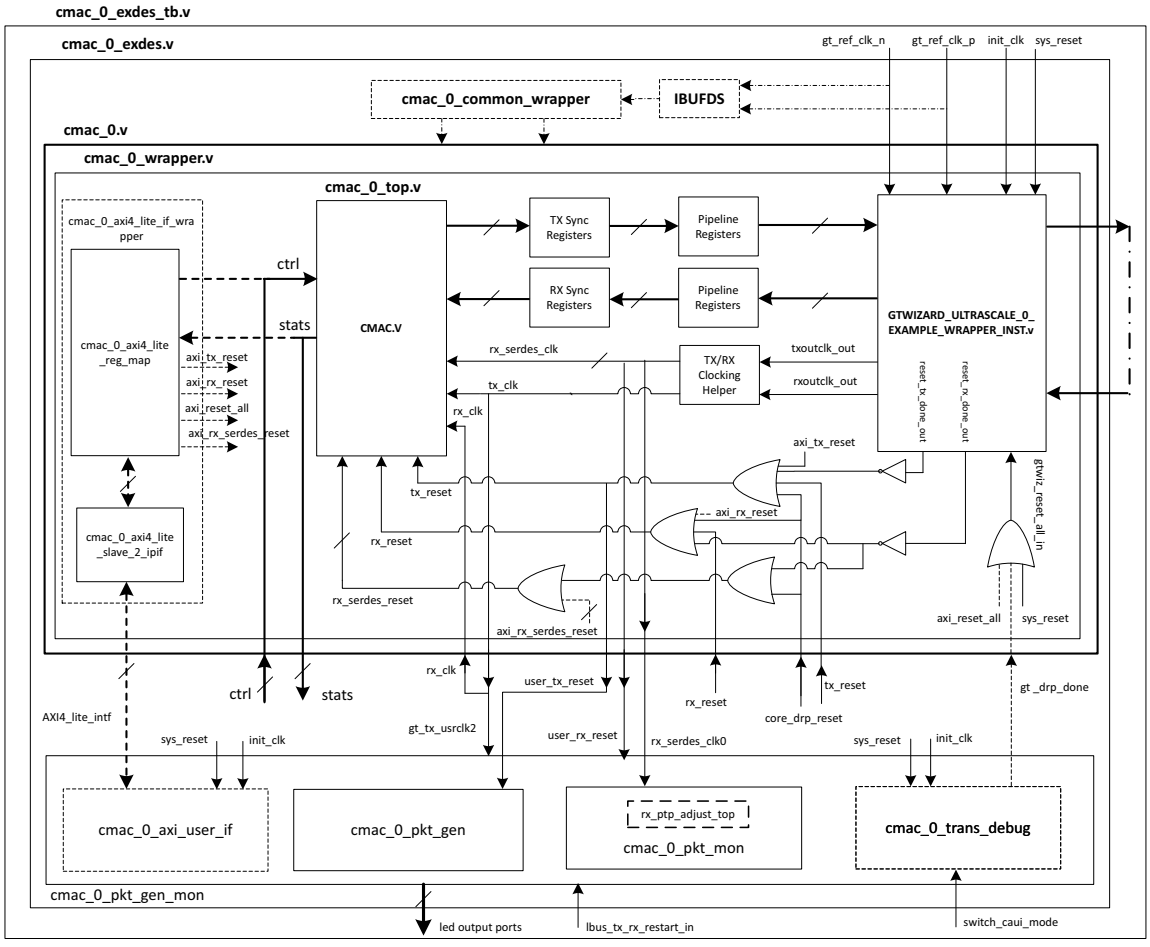

Figure 5-1: **Example Design Hierarchy**

Figure 5-1 shows the instantiation of various modules and their hierarchy in the example design. The `cmac_0` module instantiates the 100G Ethernet IP core and GT along with various helper blocks. Sync registers and pipeline registers are used for the synchronization of data between the 100G Ethernet IP core and the GT. Clocking helper blocks are used to generate the required clock frequency for the 100G Ethernet IP core. The `cmac_0_pkt_gen_mon` module instantiates `cmac_0_pkt_gen` (packet generator) and `cmac_0_pkt_mon` (packet monitor).

`cmac_0_pkt_gen_mon` and `cmac_0` handshake with each other using few signals such as GT locked, RX alignment, and data transfer signals as per the LBUS protocol (more on this will be described in later sections). The `cmac_0_pkt_gen` module is mainly responsible for the generation of packets. It contains a state machine that monitors the status of the GT and the 100G Ethernet IP core (that is, GT lock and RX alignment) and sends traffic to the core. Similarly the `cmac_0_pkt_mon` module is mainly responsible for the reception and checking of packets from the core.

It also contains a state machine that monitors the status of the GT and 100G Ethernet IP core (that is, GT lock and RX alignment) and receives traffic from the core.

Other optional modules instantiated in the example design are as follows:

- **cmac_0_trans_debug**: This module brings out all the DRP ports of the transceiver module out of the 100G Ethernet IP core. This module is present in the example design for the following conditions:
  - When you select the **Runtime Selectable** mode in the 100G Ethernet IP Vivado® Integrated Design Environment (IDE), this module is used to perform the GT DRP writes to change the GT configuration (that is, from CAUI-4 to CAUI-10 / CAUI-10 to CAUI-4). After completion of the DRP write, this module generates the `drp_done_out` signal that is used to reset the GT.
  - When you select **Additional transceiver control and status port** in the 100G Ethernet IP Vivado IDE.
- **cmac_0_common_wrapper**: When you select **Include Shared Logic in example design** in the 100G Ethernet IP Vivado IDE, this module is present in the example design. This module brings the transceiver common module out of the 100G Ethernet IP core.
- **Pipeline registers**: Single-stage pipeline registers are introduced between the 100G Ethernet IP core and the transceiver when you select **Enable Pipeline register** in the General Configuration Tab. This includes a one-stage pipeline register between the 100G Ethernet IP core macro and the transceiver to ease timing, using the `gt_txusrclk2` and `gt_rxusrclk2` for the TX and RX paths respectively.
- **TX / RX Sync register**: The TX Sync register double synchronizes the data between the 100G Ethernet IP core and the transceiver with respect to the `tx_clk`. The RX Sync register double synchronizes the data between the transceiver and the 100G Ethernet IP core with respect to `rx_serdes_clk`.

- **rx_ptp_adjust_top**: When you select **Enable time stamping** in the General Configuration Tab, this module is present inside the packet monitor module. This soft logic can improve timestamp accuracy and compensate for the lane alignment FIFO fill levels by adding or subtracting the relative fill level of the selected lane. The reference fill level can be the minimum, average, or maximum fill levels. This module has a window averaging block with fixed window size of 32.
- **cmac_0_axi4_lite_if_wrapper**: When selected with the **Enable AXI4-Lite Interface** option from the General Configuration Tab, this module will be included inside the `cmac_0_wrapper`. This wrapper contains two modules `cmac_0_axi4_lite_reg_map` and `cmac_0_axi4_lite_slave_2_ipif`. The details of these modules are described in AXI4-Lite Interface Implementation.
- **cmac_0_axi4_lite_user_if**: When selected with the **Enable AXI4-Lite Interface** option from the General Configuration Tab, this module will be present inside the `cmac_0_pkt_gen_mon`. The details about this module are described in section AXI4-Lite Interface Implementation.

User Interface

General purpose I/Os (GPIOs) have been provided to control the example design. I/Os are as listed in Table 5-1.

Note: For all the input and output signals mentioned in Table 5-1, a three-stage registering has been done internally.

Table 5-1: **User I/O Ports**

Name	Size	Direction	Description
sys_reset	1	Input	Reset for cmac_0
gt_ref_clk_p	1	Input	Differential input clk to GT
gt_ref_clk_n	1	Input	Differential input clk to GT
init_clk	1	Input	Stable input clk to GT
simplex_mode_rx_aligned	1	Input	This signal is used to indicate to the generator module that the simplex RX module is aligned and the generator can now start the packet generation. ***Note:*** This input is available only for simplex TX.
switch_caui_mode	1	Input	This signal is used to initiate the GT DRP write operation to switch the operation mode of CMAC. After the GT DRP operation, a normal data sanity check will be performed for the switched mode. ***Note:*** This input is available only for the mode **Runtime Selectable**. This input should be a single pulse. Do not apply another pulse before RX is aligned.

Table 5-1: **User I/O Ports *(Cont'd)***

Name	Size	Direction	Description
lbus_tx_rx_restart_in	1	Input	This signal is used to restart the packet generation and reception for the data sanity test, when the packet generator and the packet monitor are in idle state. (that is, tx_busy_led = 0 and rx_busy_led = 0)
tx_gt_locked_led	1	Output	Indicates that the GT has been locked. (This input is available only for simplex TX mode.)
tx_done_led	1	Output	Indicates that the packet generator has sent all the packets.
caui_mode_led	1	Output	Indicates the CMAC operation mode (CAUI-10/CAUI-4) 1b0: CAUI-10 1b1: CAUI-4 ***Note:*** This output is available only for the mode **Runtime Selectable**.
tx_busy_led	1	Output	Indicates that the generator is busy and is not able to respond to the lbus_tx_rx_restart_in command.
rx_gt_locked_led	1	Output	Indicates that the GT has been locked.
rx_aligned_led	1	Output	Indicates that RX alignment has been achieved.
rx_done_led	1	Output	Indicates that the monitor has received all packets.
rx_data_fail_led	1	Output	Indicates the data comparison failed in the packet monitor.
rx_busy_led	1	Output	Indicates that the monitor is busy and is not able to respond to the lbus_tx_rx_restart_in command.
mode_sel_in	2	Input	This signal is visible when you select the **Enable Time Stamping** option from the General Configuration Tab. This signal is used in the rx_ptp_adjust_top soft logic module to select the type of correction. `2'b00` - Maximum of averaged RX Lane Aligner Fill `2'b01` - Minimum of averaged RX Lane Aligner Fill `2'b10` - Average of averaged RX Lane Aligner Fill `2'b11` - Invalid

Notes:

1. For all the input and output signals mentioned in the table, a three-stage registering has been done internally.

Modes of Operation

Three modes of operations are supported for this example design which are:

- Duplex Mode
- Simplex TX Mode
- Simplex RX Mode

Duplex Mode

In this mode of operation both the 100G Ethernet IP core transmitter and receiver are active and loopback is provided at the GT output interface, that is, output is fed back as input. Packet generation and monitor are also active in this mode.

To enable this mode of operation, select the duplex mode from the Vivado IDE parameters. Figure 5-2 shows the duplex mode of operation.

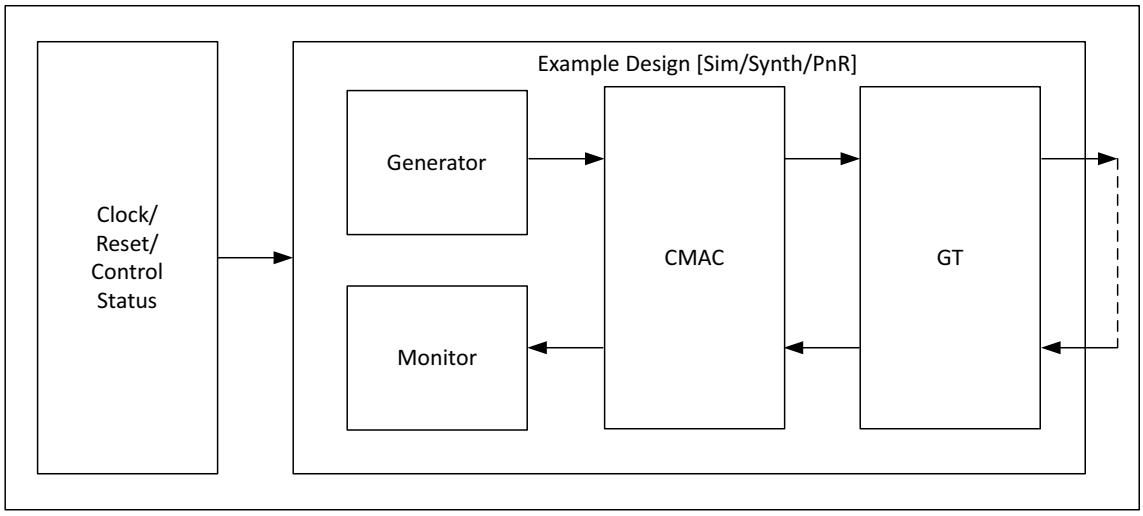

Figure 5-2: **Duplex Mode of Operation**

Simplex TX Mode

In this mode of operation only the 100G Ethernet IP core transmitter is enabled as shown in Figure 5-3. Also, only the packet generator will be enabled for the generation of packets.

To enable this mode of operation, select the simplex TX mode from the Vivado IDE. Figure 5-3 shows the simplex TX mode of operation.

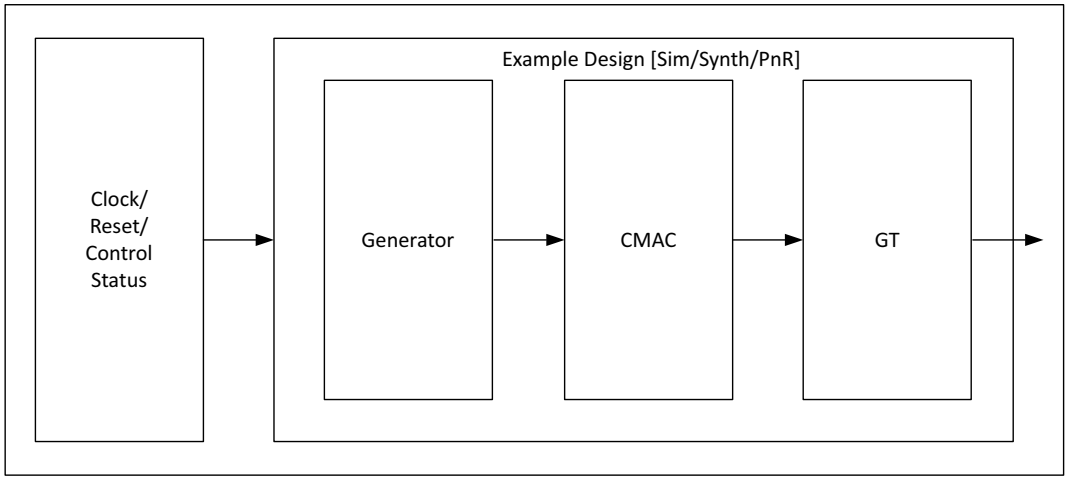

Figure 5-3: **Simplex TX Mode of Operation**

Simplex TX Mode Simulation

As shown in the Figure 5-3 in this mode of operation only the CMAC transmitter is enabled and the packet generator is enabled for the generation of packets. For simulation a partner test bench is instantiated to perform the functionality of the CMAC receiver. This partner test bench will have a CMAC receiver and a packet monitor to verify the received data from the generator.

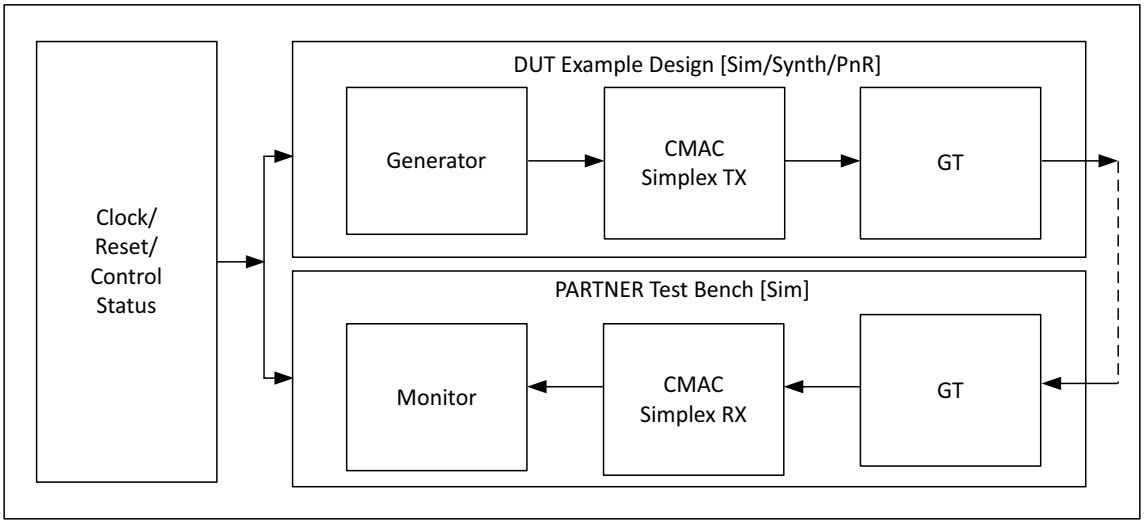

Figure 5-4: **Simplex TX Mode Simulation Block Diagram**

Simplex RX Mode

In this mode of operation only the 100G Ethernet IP core receiver is enabled as shown in Figure 5-5. Also only the packet monitor will be enabled for the reception of packets.

To enable this mode of operation, select the **Simplex Rx** mode from the Vivado IDE parameters.

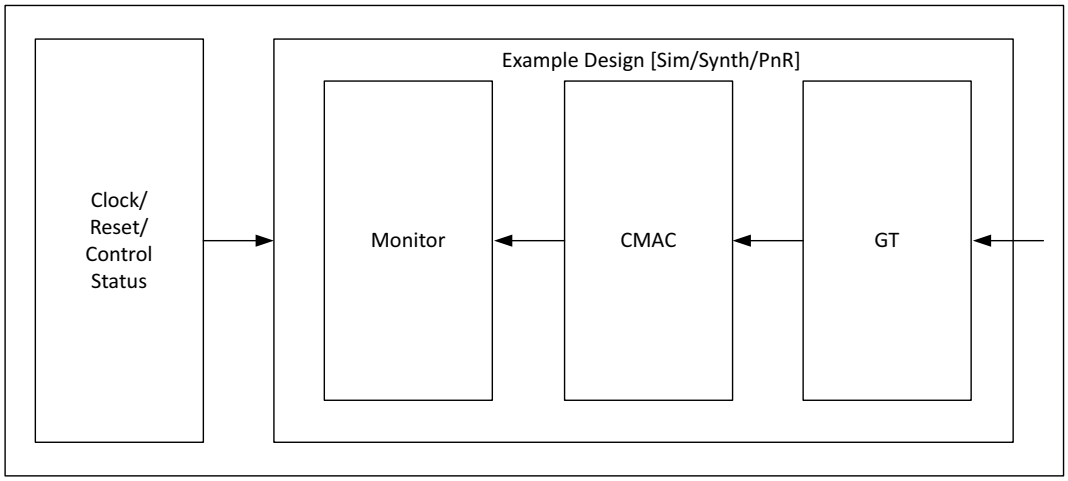

Figure 5-5: **Simplex RX Mode of Operation**

Simplex RX Mode Simulation

As shown in Figure 5-5, in this mode of operation only the CMAC receiver is enabled and the packet monitor will be enabled to verify the received data. For simulation, a partner test bench is instantiated to perform the functionality of the CMAC transmitter. This partner test bench will have a CMAC transmitter and a packet generator to generate the test data.

Figure 5-6 shows the Simplex RX mode for simulation.

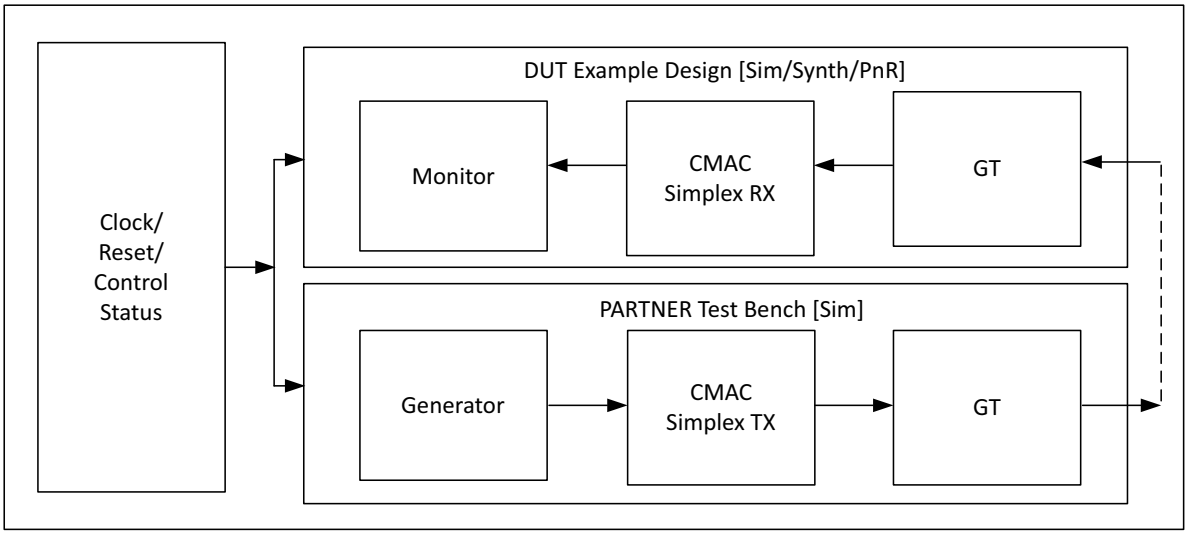

Figure 5-6: **Simplex RX Mode Simulation Block Diagram**

Transaction Flow

This section describes the flow of data between `cmac_0_pkt_gen_mon` and `cmac_0` and various state transitions that happen within `cmac_0_pkt_gen` and `cmac_0_pkt_mon`.

Packet Generation

The module `cmac_0_pkt_gen` is responsible for the generation of LBUS packets. Typically the packet generator waits for the GT to achieve lock and for the core RX to get aligned. After this has occurred, the packet generator sends a predefined number of packets. A Finite State Machine (FSM) is used to generate the LBUS packets. A functional description of each state follows:

- **STATE_TX_IDLE**: By default the controller is in the STATE_TX_IDLE state. When `reset_done` becomes High, it moves to the STATE_GT_LOCKED state.
- **STATE_GT_LOCKED**: This state sets `ctl_tx_send_rfi=1`, `tx_core_busy_led=1` and `gt_lock_led=1`. It then moves to the STATE_WAIT_RX_ALIGNED state.
- **STATE_WAIT_RX_ALIGNED**: This state waits for the 100G Ethernet IP cores to indicate `stat_rx_aligned=1`, which means that the 100G Ethernet IP RX core is locked. After that, it moves to the STATE_PKT_TRANSFER_INIT state.
- **STATE_PKT_TRANSFER_INIT**: This state sets `rx_aligned_led=1` and `tx_core_busy_led=1`. It then initializes all signals to start LBUS packet generation and moves to the STATE_LBUS_TX_ENABLE state.

- **STATE_LBUS_TX_ENABLE**: This state checks for the number of packets to be generated and sends LBUS packets of a predefined size. After sending all the packets, the FSM moves to the STATE_LBUS_TX_DONE state. During transmission of the packets, if `tx_rdyout`=0, `tc_ovfout`=1 or `tx_unfout`=1, the FSM controller moves to the STATE_LBUS_TX_HALT state.
- **STATE_LBUS_TX_HALT**: In this state, the controller generates the `tx_fail_reg` flag if `tc_ovfout` or `tx_unfout` is High. Then the FSM moves to the STATE_LBUS_TX_DONE state. If `tx_rdyout` becomes High, the FSM moves to the STATE_LBUS_TX_ENABLE state to proceed with packet generation.
- **STATE_LBUS_TX_DONE**: This state resets all signals related to packet generation and sets `tx_done_led` =1. If 1588 1-step is enabled, FSM moves to the STATE_PTP_PKT_INIT state; otherwise it checks if TX_FLOW_CONTROL is enabled. If enabled, the FSM moves to the STATE_TX_PAUSE_INIT state. If TX_FLOW_CONTROL is now enabled, the FSM moves to the STATE_WAIT_FOR_RESTART state.
- **STATE_WAIT_FOR_RESTART**: In this state, all the packet generator parameters reset to the default values and reset `tx_busy_led`=0. The FSM moves to STATE_PKT_TRANSFER_INIT at `tx_restart_rising_edge`.
- **STATE_PTP_PKT_INIT**: Reset all the signals used for LBUS transactions. Move to the STATE_PTP_PKT_READ state wait until the initialization counter is done and set the `ptp_pkt_transfer` flag to one. After sending three 1588 PTP packets (Ethernet, IPV4 and IPV6) FSM moves to the STATE_TX_PAUSE_INIT state if the TX_FLOW_CONTROL is enabled; otherwise FSM moves to STATE_WAIT_FOR_RESTART state.
- **STATE_PTP_PKT_READ**: In case of IPV4 or IPV6, increment the `tx_ptp_pkt_index` and move to the STATE_PTP_PKT_TRANSFER state.
- **STATE_TX_PTP_PKT_TRANSFER**: Read the data from the `ptp_pkt_gen` module after sending the complete ptp packet, move to the STATE_PTP_PKT_INIT state.
- **STATE_TX_PAUSE_INIT**: Set the `ctl_tx_pause_enable = 9'h100` and `ctl_tx_pause_req[8]` = 1 and wait for the `stat_tx_pause` signal to become High and the FSM moves to the STATE_TX_PPP_INIT state.
- **STATE_TX_PPP_INIT**: In this state, the controller sets `ctl_tx_pause_enable` = 9'h0ff and `ctl_tx_pause_req[7:0]` one bit at a time in decrementing order (bit 7 to bit 0). It then waits for `stat_tx_pause_valid[0]` to become High and moves to the STATE_TX_PAUSE_DONE state.
- **STATE_TX_PAUSE_DONE**: In this state, all the pause signals are reset. The controller then moves to the STATE_WAIT_FOR_RESTART state.

Notes:

- If any time `stat_rx_aligned` = 0, the FSM moves to STATE_TX_IDLE.
- In the simplex TX mode of operation because RX alignment information will not be available, the state machine waits for you to input `simplex_mode_rx_aligned`. After you assert this input to High, packet transmission starts.
- If you select the **Disable FCS Insertion** option in the General Configuration Tab, the CMAC core will not insert the CRC value for the data packets. So a `CRC_Mapping_LUT` module will be instantiated inside this `cmac_0_pkt_gen` module. This `CRC_Mapping_LUT` module contains the pre-calculated CRC values for the current LBUS predefined data packet of the packet size 522 bytes. These CRC values will be appended at the end of each LBUS packet.

  Therefore, if you are changing the packet size, you must change the CRC values for this new module as appropriate for the new packet and/or packet size.

  In case of 1588 1-step enable, you must provide appropriate CRC values as per the packet and/or packet size.
- If you select the **1588 1-step enable** option in the General Configuration Tab, the `ptp_packet_gen` module will be instantiated inside the `cmac_0_packet_gen` module. This module contains three 1588 PTP packets (Ethernet, IPV4 and IPV6) with or without the CRC values based on the **Disable FCS Insertion** option in the General Configuration Tab.

The state transition occurring during this process is shown in Figure 5-7.

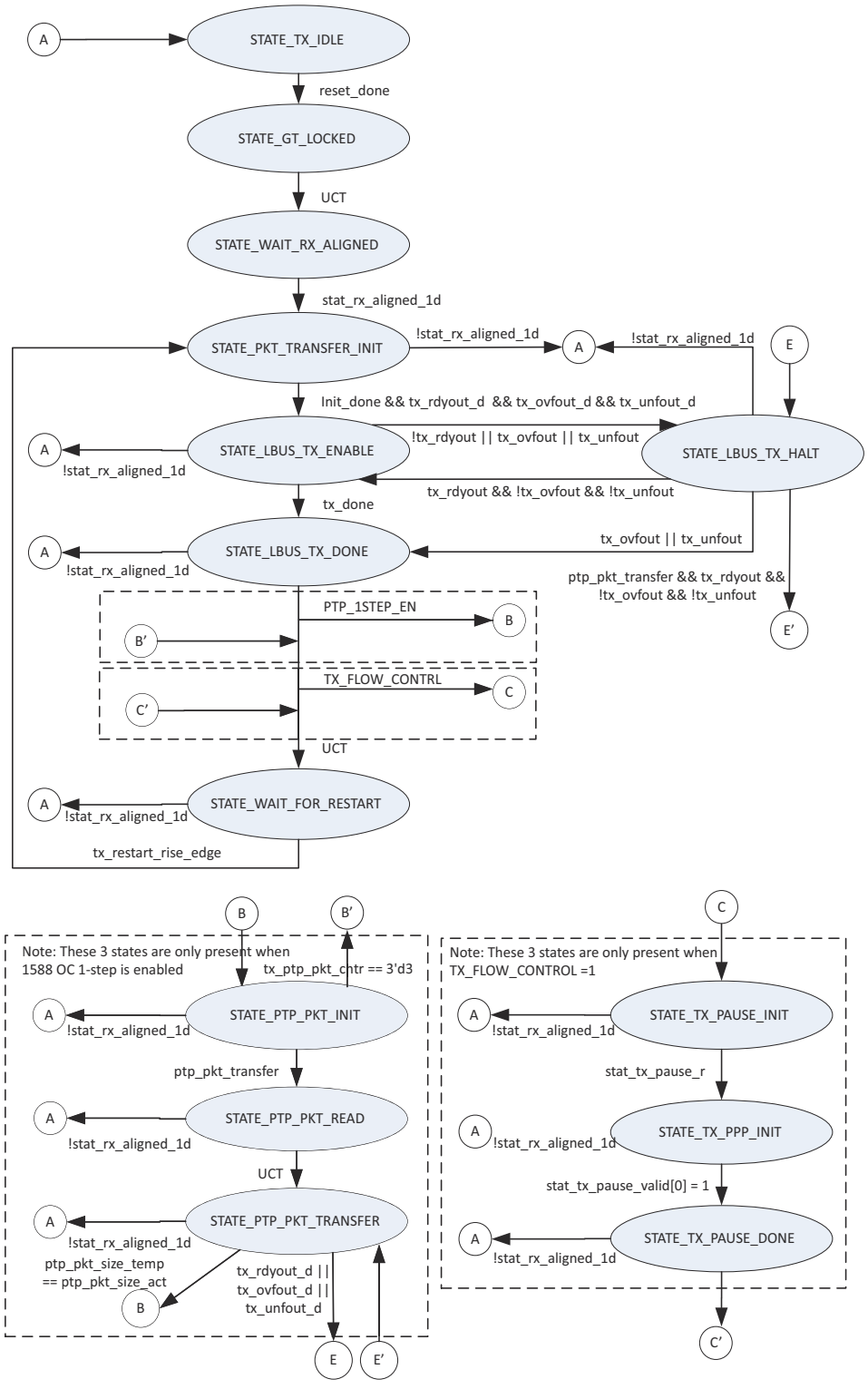

Figure 5-7: **State Transition Diagram for Packet Generator**

Packet Reception

The module cmac_0_pkt_mon is responsible for reception of packets. Typically the packet monitor waits for transceivers to achieve lock and for the 100G Ethernet IP RX to align. After alignment, the packet monitor receives a predefined number of packets. The FSM is used to monitor the RX LBUS signals. A functional description of each state follows:

- **STATE_RX_IDLE**: By default, the FSM is in the IDLE state. When `reset_done` goes High, the FSM moves to the STATE_GT_LOCKED state.
- **STATE_GT_LOCKED**: This state sets `gt_lock_led`=1, `rx_core_busy_led`=1, and `ctl_rx_enable`=1. Then the FSM moves to the STATE_WAIT_RX_ALIGNED state.
- **STATE_WAIT_RX_ALIGNED**: This state waits for `stat_rx_aligned`=1, which indicates that the 100G Ethernet IP RX core is aligned. The FSM then moves to the STATE_PKT_TRANSFER_INIT state.
- **STATE_PKT_TRANSFER_INIT**: This state sets `rx_aligned_led`=1, `rx_core_busy_led`=1, initializes all signals to start LBUS packet generation, and then moves to the STATE_LBUS_RX_ENABLE state.
- **STATE_LBUS_RX_ENABLE**: This state receives LBUS packets and compares them to the expected packets. If there is a mismatch, it sets `rx_data_fail_led`=1. This flag is reset only when `lbus_tx_rx_restart_in`=1. After receiving all the packets, the FSM moves to the STATE_LBUS_RX_DONE state.
- **STATE_LBUS_RX_DONE**: This state resets all the signals related to LBUS packets, sets the `rx_done_led`=1, and moves to the STATE_WAIT_FOR_RESTART state. If the TX Flow Control and RX Flow Control functions are enabled, it waits for `pause_test_done`=1 and then moves to the STATE_WAIT_FOR_RESTART state. If 1588 1-step is enabled, the FSM moves to the STATE_RX_PTP_ENABLE state.
- **STATE_RX_PTP_ENABLE**: Receive the three 1588 PTP packets. After receiving the packets, the FSM moves to the STATE_RX_PTP_DONE state.
- **STATE_RX_PTP_DONE**: This state only displays the time stamps received. If the TX Flow Control and RX Flow Control are enabled, wait for the `pause_test_done`=1 and move to STATE_WAIT_FOR_RESTART.
- **STATE_WAIT_FOR_RESTART**: This state resets all signals related to the LBUS packet monitor and resets `rx_core_busy_led`=0. It then waits for `rx_restart_rise_edge`=1 and `stat_rx_aligned`=1. The FSM then moves to the STATE_PKT_TRANSFER_INIT state. If any time `stat_rx_aligned` = 0, the FSM moves to STATE_RX_IDLE

Notes:

- If any time stat_rx_aligned = 0, then the FSM moves to STATE_RX_IDLE.
- When RX_FLOW_CONTROL is enabled, the corresponding input control signals are initialized to enable Pause and Priority Pause frames reception.
- If you select the **Disable FCS Stripping** option in the General Configuration Tab, the `CRC_Mapping_LUT` module will be instantiated inside the `cmac_0_pkt_mon` module. This `CRC_Mapping_LUT` module contains the pre-calculated CRC values for the received LBUS data packets of packet size 522 bytes. These CRC values will be compared with the received LBUS packet CRC.
- If you change the packet size, you must provide the new CRC values as appropriate for the new packet size.

The state transition occurring during this process is shown in Figure 5-8.

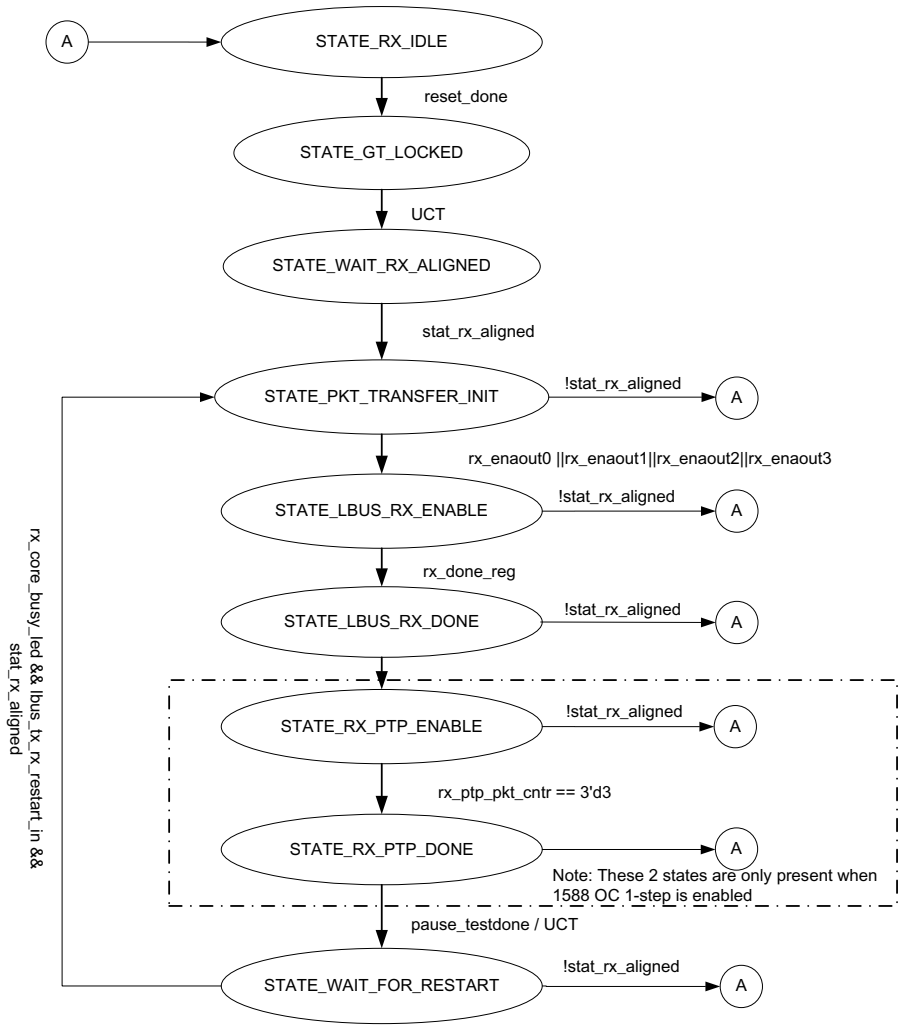

Figure 5-8: **State Transition Diagram for Packet Monitor**

Runtime Selectable

When you select the Mode option as runtime selectable, the cmac_0_runtime_switch module will be present in the example design. This cmac_0_runtime_switch module is responsible for performing the DRP write operation to switch the transceiver operation mode, that is, CAUI10-CAUI4/CAUI4-CAUI10. This module checks the `caui_mode_change` signal from the generator. Whenever it gets a mode_change request it starts the DRP write operation for the transceiver common and transceiver channel and resets the 100G Ethernet IP core. The state transition occurred during this process is shown in Figure 5-9:

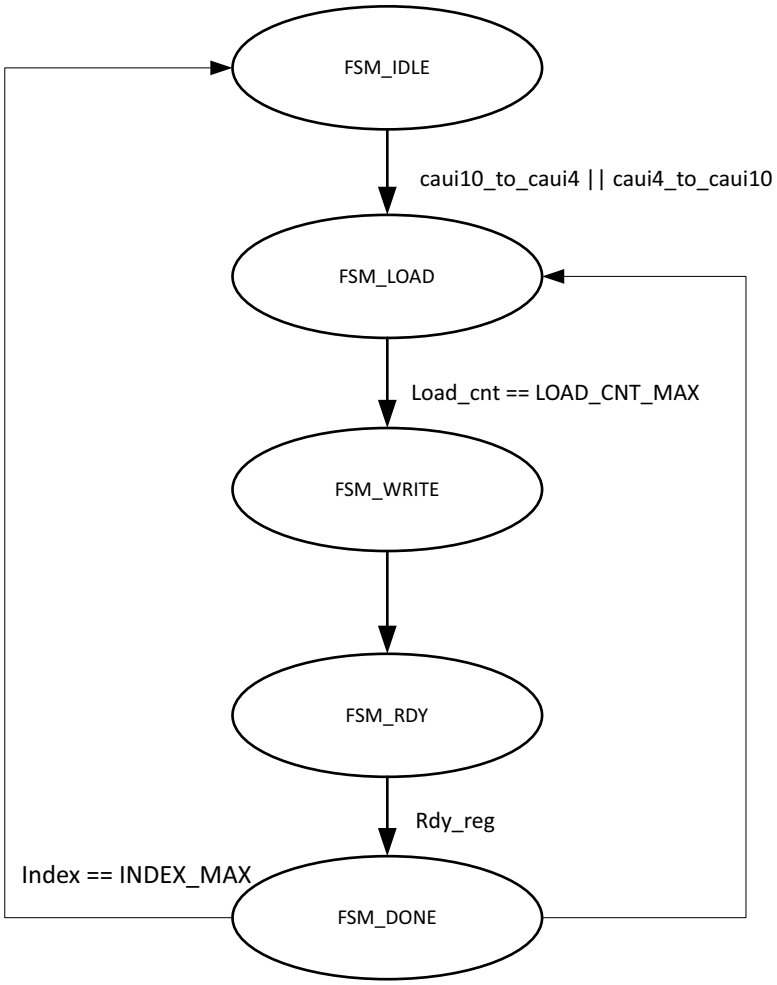

Figure 5-9: **State Transition Diagram for Runtime Selectable DRP Operation**

For reference frequency 161.1328125 MHz, DRP addresses and the Data In (DIN) value for different transceiver modules are listed in Table 5-2 (all values are in hexadecimal).

Table 5-2: **DRP Address and Values to Switch the Mode**

Module	DRP Address	CAUI-10 DRP DIN Value	CAUI-4 DRP DIN Value
GTY Common	00E	0000	0001
GTY Common	014	003E	004E
GTY Channel	03E	A180	A1A2
GTY Channel	085	057C	097C
GTY Channel	003	0881	08E1
GTY Channel	07A	B004	B007
GTY Channel	066	3131	3132

Shared Logic Implementation

Shared logic includes the GT common module which can be present as part of GT or in the example design. By default, shared logic is present inside the GT common. If you want to instantiate shared logic in the example design, select **Include Shared logic in example design** in the Vivado IDE.

Figure 5-10 shows the implementation when shared logic is instantiated in the example design.

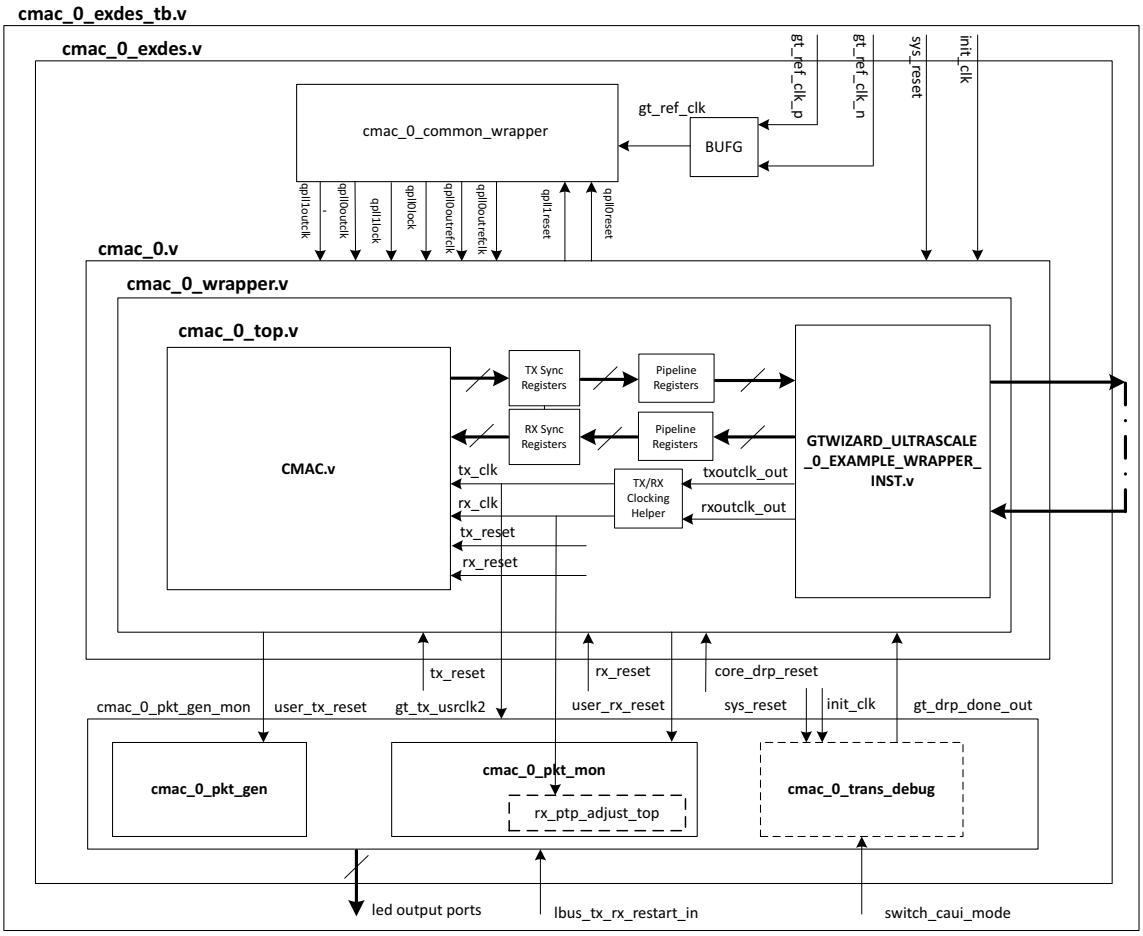

Figure 5-10: **Example Design Hierarchy with Shared Logic Implementation**

CORE DRP Operation

1. Make `core_drp_reset` signal High.
2. Perform the DRP write/read operation.
3. After completion of the DRP operation, make the `core_drp_reset` signal Low.
4. Wait for the `rx_alignment`.

AXI4-Lite Interface Implementation

If you wish to instantiate the AXI4-Lite interface to access the control and status registers of the CMAC core, you need to tick mark the **Enable AXI4-Lite Control and Statistics** check box in the General Configuration Tab. It enables the `axi4_lite_if_wrapper` module (that contains `axi4_lite_reg_map` along with the `axi4_lite_slave_2_ipif` module) in the `cmac_wrapper`. The user interface logic for accessing the registers (control, status and statistics) is present in the `pkt_gen_mon` module.

This mode enables the following features:

- You can configure all the CTL ports of the core through the AXI4-Lite interface. This operation is performed by writing to a set of address locations with the required data to the register map interface to the core. The address location with the configuration register list is mentioned in Table 5-5.
- You can access all the status and statistics registers from the core through the AXI4-Lite interface. This is performed by reading the address locations for the status and statistics registers through register map. Table 5-6 shows the address with the corresponding register descriptions.

The following diagram shows the implementation when the AXI4-Lite interface is instantiated in the example design.

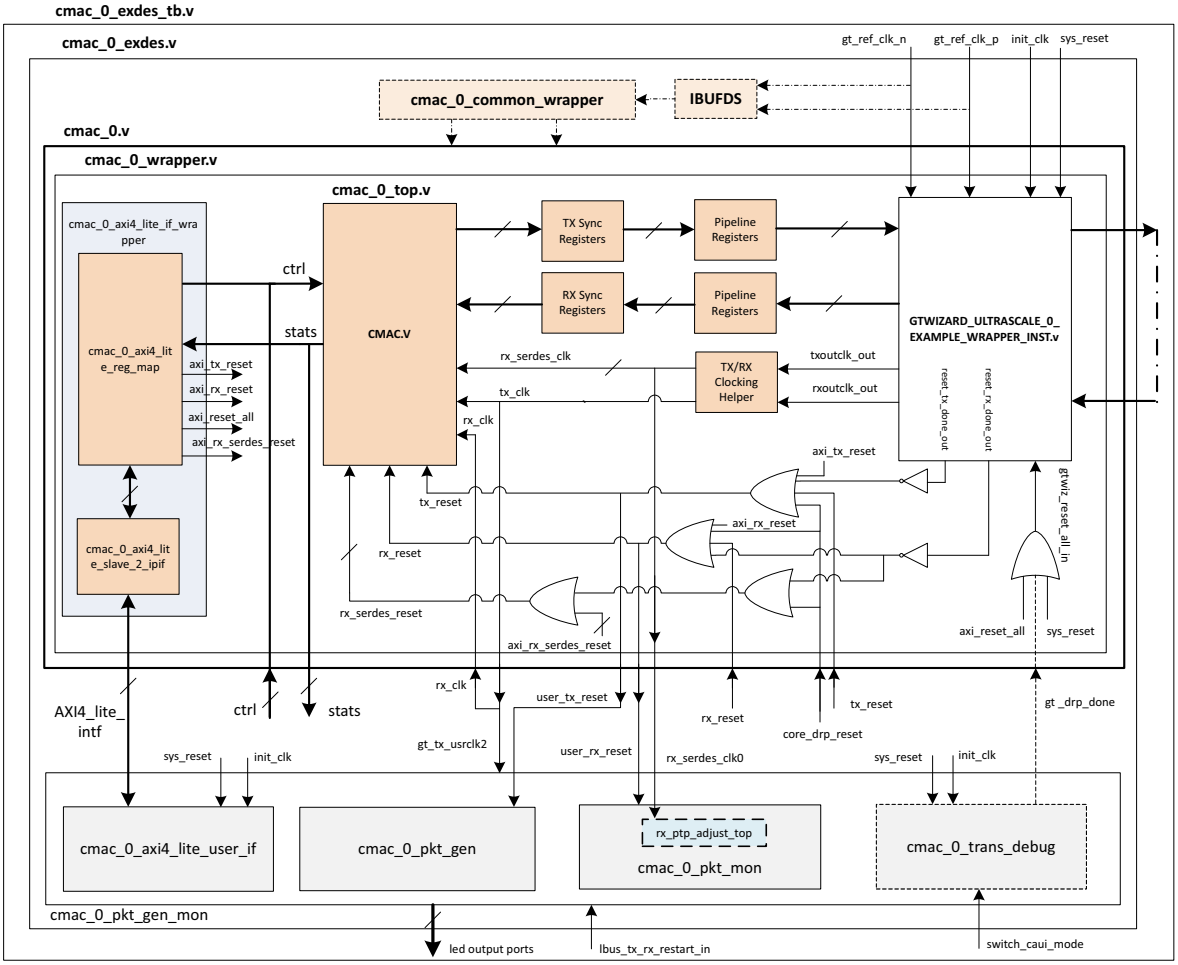

Figure 5-11: **Example Design Hierarchy with AXI4-Lite Interface**

The following sections provide the AXI4-Lite interface state machine control and ports.

User State Machine

The read and write through the AXI4-Lite slave module interface is controlled by a state machine as shown in Figure 5-12.

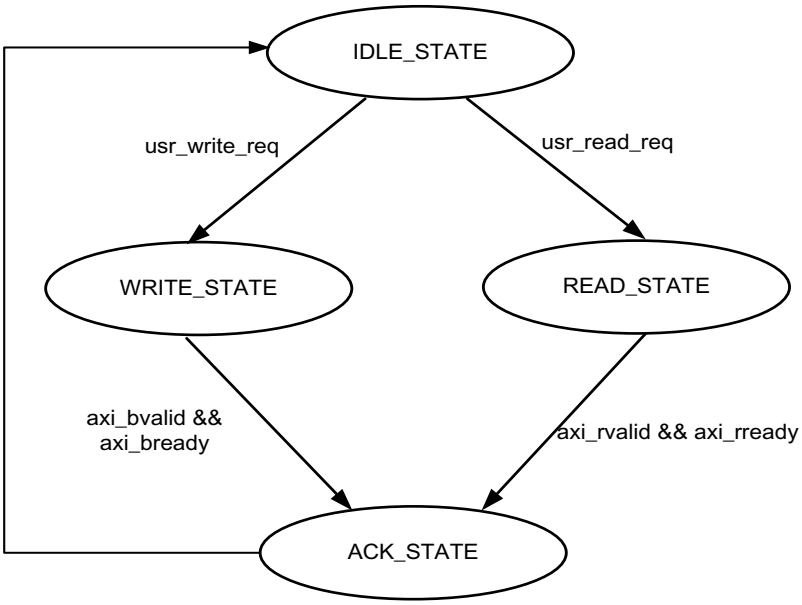

Figure 5-12: **User State Machine for AXi4-Lite Interface**

Following is the functional description of each state.

- **IDLE_STATE**: By default the FSM will be in the IDLE_STATE state. When the `user_read_req` signal becomes High, then it moves to READ_STATE; else if the `user_write_req` signal is High, it moves to the WRITE_STATE; else it remains in IDLE_STATE.
- **WRITE_STATE**: You provide S_AXI_AWVALID, S_AXI_AWADDR, S_AXI_WVALID, S_AXI_WDATA and S_AXI_WSTRB in this state to write to the register map through AXI. When S_AXI_BVALID and S_AXI_BREADY from the AXI slave are High then it moves to ACK_STATE. If there is any write operation than happens in any illegal addresses, the S_AXI_BRESP[1:0] indicates `2'b10` that asserts the write error signal.
- **READ_STATE**: You provide S_AXI_ARVALID and S_AXI_ARADDR in this state to read from the register map through AXI. When S_AXI_RVALID and S_AXI_RREADY are High then it moves to ACK_STATE. If there is any read operation that occurs from any illegal addresses, the S_AXI_RRESP[1:0] indicates `2'b10` that asserts the read error signal
- **ACK_STATE**: The state moves to IDLE_STATE.

AXI User Interface Ports

Table 5-3: **AXI User Interface Ports**

Name	Size	Direction	Description
S_AXI_ACLK	1	Input	AXI clock signal
S_AXI_SRESET	1	Input	AXI active-High synchronous reset
S_AXI_AWADDR	32	Input	AXI write address
S_AXI_AWVALID	1	Input	AXI write address valid
S_AXI_AWREADY	1	Output	AXI write address ready
S_AXI_WDATA	32	Input	AXI write data
S_AXI_WSTRB	4	Input	AXI write strobe. This signal indicates which byte lanes hold valid data.
S_AXI_WVALID	1	Input	AXI write data valid. This signal indicates that valid write data and strobes are available.
S_AXI_WREADY	1	Output	AXI write data ready
S_AXI_BRESP	2	Output	AXI write response. This signal indicates the status of the write transaction. 'b00 = OKAY 'b01 = EXOKAY 'b10 = SLVERR 'b11 = DECERR
S_AXI_BVALID	1	Output	AXI write response valid. This signal indicates that the channel is signaling a valid write response.
S_AXI_BREADY	1	Input	AXI write response ready.
S_AXI_ARADDR	32	Input	AXI read address
S_AXI_ARVALID	1	Input	AXI read address valid
S_AXI_ARREADY	1	Output	AXI read address ready
S_AXI_RDATA	32	Output	AXI read data issued by slave
S_AXI_RRESP	2	Output	AXI read response. This signal indicates the status of the read transfer. 'b00 = OKAY 'b01 = EXOKAY 'b10 = SLVERR 'b11 = DECERR

Table 5-3: **AXI User Interface Ports *(Cont'd)***

Name	Size	Direction	Description
S_AXI_RVALID	1	Output	AXI read data valid
S_AXI_RREADY	1	Input	AXI read ready. This signal indicates that the user/master can accept the read data and response information.

User Side AXI4-Lite Write/Read Transactions

Figure 5-13 through Figure 5-16 show timing diagram waveforms for the AXI4-Lite interface

- Valid Write transactions (Figure 5-13)
- Valid Read transactions (Figure 5-15)
- Invalid Write transactions (Figure 5-14)
- Invalid Read transactions (Figure 5-16)

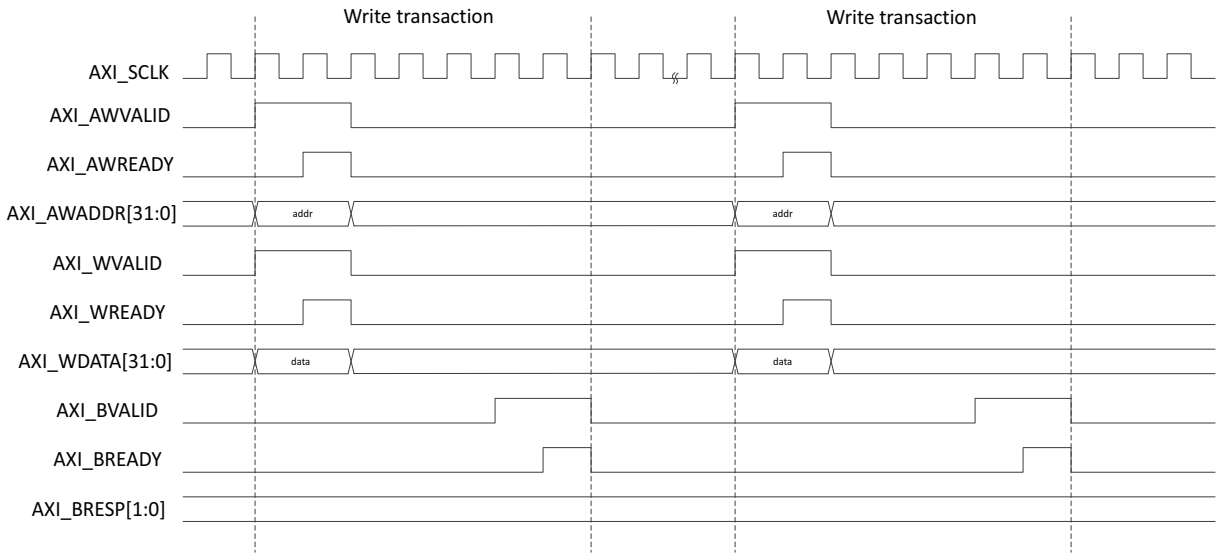

Figure 5-13: **AXI4-Lite User Side Write Transaction**

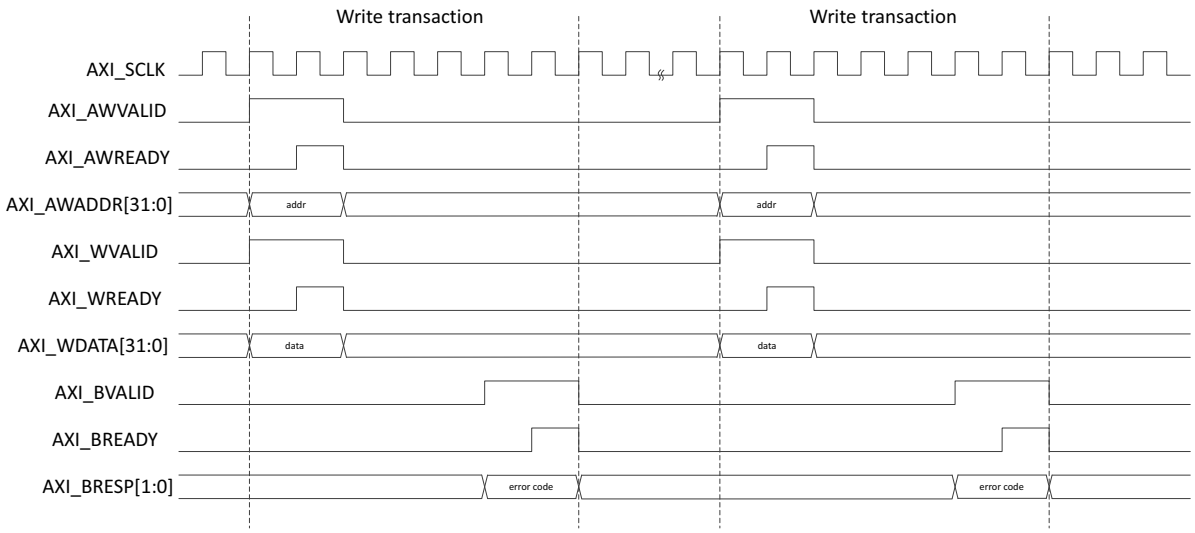

Figure 5-14: **AXI4-Lite User Side Write Transaction with Illegal Write Address**

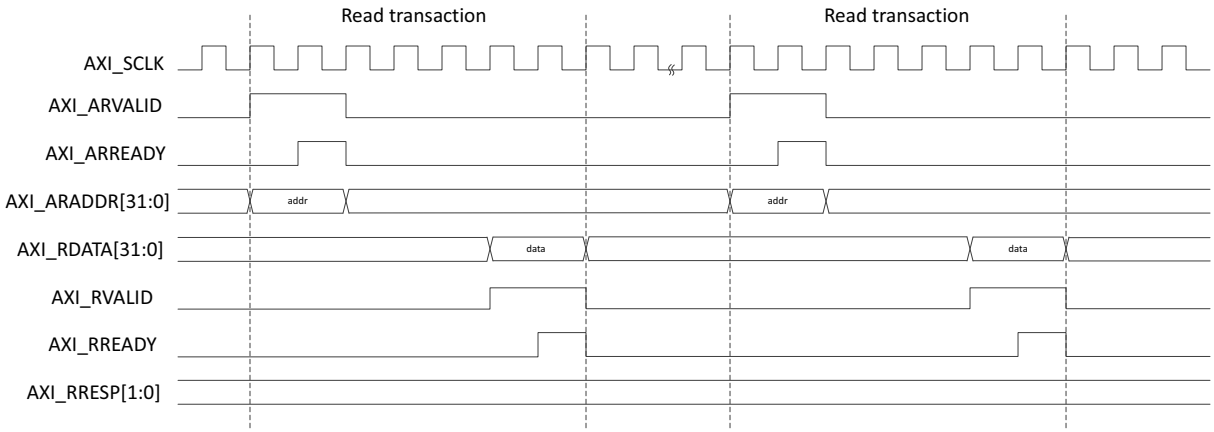

Figure 5-15: **AXI4-Lite User Side Read Transaction**

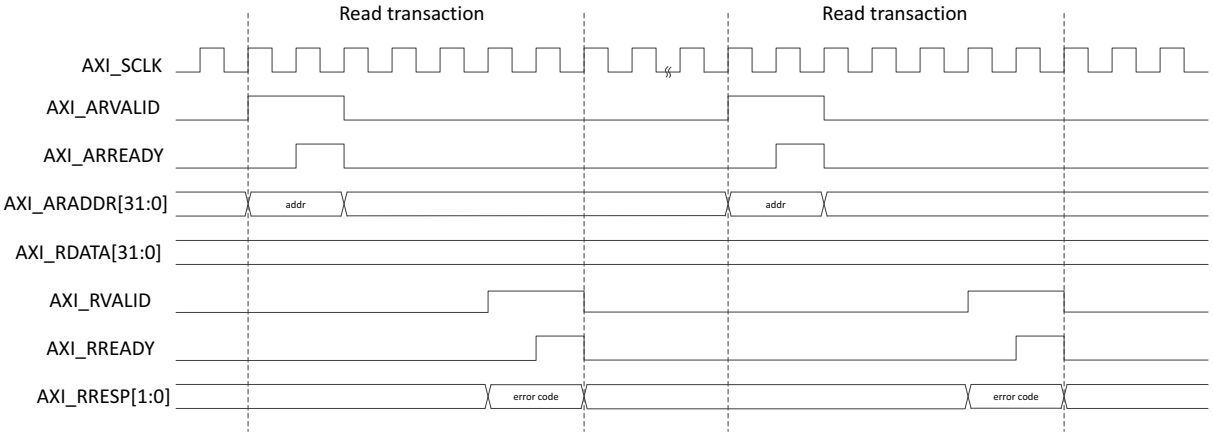

Figure 5-16: **AXI4-Lite User Side Read Transaction with Illegal Read Address**

Register Map

The following sections provide the register map and register descriptions for the core.

Base Pages

The register map is broken into two 512 base address pages to allow for future development and expansion.

Table 5-4: **Register Base Addresses**

Base Address	Space Name
0x0000 0000	IP Configuration Registers
0x0000 0200	Status and Statistics Registers

All registers are 32 bytes in size and aligned on 32-byte addressing. In the following register space maps, any holes in the address space should be considered RESERVED and can cause the AXI Control interface IP to respond with an error if accessed.

Configuration Register Space

The configuration space provides the ability to configure the core for various use cases.

The integrated IP UltraScale™ CMAC makes use of a dynamic reconfiguration port (DRP) to provide the ability to configure aspects of the core without the need for fabric logic connections. In this case those configuration bits in the soft AXI Control register set will become RESERVED (unused) and the software should use the DRP operation registers to configure those attributes of the core. See Table 3-7 for the DRP address map.

Table 5-5: **Configuration Register Map**

Address	Register Name
0x0000	GT_RESET_REG
0x0004	RESET_REG
0x0008	MODE_REG
0x000C	CONFIGURATION_TX_REG1
0x0010	Reserved
0x0014	CONFIGURATION_RX_REG1
0x0018 – 0x0028	Reserved
0x002C	CONFIGURATION_TX_BIP_OVERRIDE
0x0030	CONFIGURATION_TX_FLOW_CONTROL_CONTROL_REG1
0x0034	CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG1
0x0038	CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG2
0x003C	CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG3
0x0040	CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG4
0x0044	CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG5
0x0048	CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG1
0x004C	CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG2
0x0050	CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG3
0x0054	CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG4
0x0058	CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG5
0x005C – 0x0080	Reserved
0x0084	CONFIGURATION_RX_FLOW_CONTROL_CONTROL_REG1
0x0088	CONFIGURATION_RX_FLOW_CONTROL_CONTROL_REG2
0x008C – 0x01FF	Reserved

Status and Statistics Register Space

The status and statistics registers indicate the health of the link and histograms counters to provide classification of the traffic and error counts. The status and counters are all read-only.

Status registers are cleared on read and counters are controlled by a "tick" mechanism.

The counters accumulate their counts in an internal accumulator. A write to the TICK_REG register causes the accumulated counts to be pushed to the readable STAT_*_MSB/LSB registers and simultaneously clears the accumulators. The STAT_*_MSB/LSB registers can then be read. In this way all values stored in the statistics counters represent a snapshot over the same time-interval.

The STAT_CYCLE_COUNT_MSB/LSB register contains a count of the number of SerDes clock cycles between TICK_REG register writes. This allows for easy time-interval based statistics. The counters have a default width of 48 bits. The counters saturate to 1s. The values in the counters are held until the next write to the TICK_REG register.

The addresses shown in Table 5-6 for the counters are the addresses of the LSB register or bits 31:0 of the count. The MSB bits 47:32 of the counter are located at +0x4 from the LSB.

Table 5-6: **Status and Statistics Register Map**

Address	Register Name
0x0200	STAT_TX_STATUS_REG
0x0204	STAT_RX_STATUS_REG
0x0208	STAT_STATUS_REG1
0x020C	STAT_RX_BLOCK_LOCK_REG
0x0210	STAT_RX_LANE_SYNC_REG
0x0214	STAT_RX_LANE_SYNC_ERR_REG
0x0218	STAT_RX_AM_ERR_REG
0x021C	STAT_RX_AM_LEN_ERR_REG
0x0220	STAT_RX_AM_REPEAT_ERR_REG
0x0224	STAT_RX_LANE_DEMUXED
0x0228	STAT_RX_PCS_LANE_NUM_REG1
0x022C	STAT_RX_PCS_LANE_NUM_REG2
0x0230	STAT_RX_PCS_LANE_NUM_REG3
0x0234	STAT_RX_PCS_LANE_NUM_REG4
0x0238	STAT_RX_BIP_OVERRIDE_REG
0x023C – 0x02AF	Reserved

Table 5-6: **Status and Statistics Register Map *(Cont'd)***

Address	Register Name
Histogram / Counter Registers	
0x02B0	TICK_REG
0x02B8	STAT_CYCLE_COUNT
0x02C0	STAT_RX_BIP_ERR_0
0x02C8	STAT_RX_BIP_ERR_1
0x02D0	STAT_RX_BIP_ERR_2
0x02D8	STAT_RX_BIP_ERR_3
0x02E0	STAT_RX_BIP_ERR_4
0x02E8	STAT_RX_BIP_ERR_5
0x02F0	STAT_RX_BIP_ERR_6
0x02F8	STAT_RX_BIP_ERR_7
0x0300	STAT_RX_BIP_ERR_8
0x0308	STAT_RX_BIP_ERR_9
0x0310	STAT_RX_BIP_ERR_10
0x0318	STAT_RX_BIP_ERR_11
0x0320	STAT_RX_BIP_ERR_12
0x0328	STAT_RX_BIP_ERR_13
0x0330	STAT_RX_BIP_ERR_14
0x0338	STAT_RX_BIP_ERR_15
0x0340	STAT_RX_BIP_ERR_16
0x0348	STAT_RX_BIP_ERR_17
0x0350	STAT_RX_BIP_ERR_18
0x0358	STAT_RX_BIP_ERR_19
0x0360	STAT_RX_FRAMING_ERR_0
0x0368	STAT_RX_FRAMING_ERR_1
0x0370	STAT_RX_FRAMING_ERR_2
0x0378	STAT_RX_FRAMING_ERR_3
0x0380	STAT_RX_FRAMING_ERR_4
0x0388	STAT_RX_FRAMING_ERR_5
0x0390	STAT_RX_FRAMING_ERR_6

Table 5-6: **Status and Statistics Register Map *(Cont'd)***

Address	Register Name
0x0398	STAT_RX_FRAMING_ERR_7
0x03A0	STAT_RX_FRAMING_ERR_8
0x03A8	STAT_RX_FRAMING_ERR_9
0x03B0	STAT_RX_FRAMING_ERR_10
0x03B8	STAT_RX_FRAMING_ERR_11
0x03C0	STAT_RX_FRAMING_ERR_12
0x03C8	STAT_RX_FRAMING_ERR_13
0x03D0	STAT_RX_FRAMING_ERR_14
0x03D8	STAT_RX_FRAMING_ERR_15
0x03E0	STAT_RX_FRAMING_ERR_16
0x03E8	STAT_RX_FRAMING_ERR_17
0x03F0	STAT_RX_FRAMING_ERR_18
0x03F8	STAT_RX_FRAMING_ERR_19
0x0400 – 0x0410	Reserved
0x0418	STAT_RX_BAD_CODE
0x0420	Reserved
0x0428	Reserved
0x0430	Reserved
0x0438	Reserved
0x0440	Reserved
0x0448	Reserved
0x0450	Reserved
0x0458	STAT_TX_FRAME_ERROR
0x0460	Reserved
0x0500	STAT_TX_TOTAL_PACKETS
0x0508	STAT_TX_TOTAL_GOOD_PACKETS
0x0510	STAT_TX_TOTAL_BYTES
0x0518	STAT_TX_TOTAL_GOOD_BYTES
0x0520	STAT_TX_PACKET_64_BYTES
0x0528	STAT_TX_PACKET_65_127_BYTES

Table 5-6: **Status and Statistics Register Map *(Cont'd)***

Address	Register Name
0x0530	STAT_TX_PACKET_128_255_BYTES
0x0538	STAT_TX_PACKET_256_511_BYTES
0x0540	STAT_TX_PACKET_512_1023_BYTES
0x0548	STAT_TX_PACKET_1024_1518_BYTES
0x0550	STAT_TX_PACKET_1519_1522_BYTES
0x0558	STAT_TX_PACKET_1523_1548_BYTES
0x0560	STAT_TX_PACKET_1549_2047_BYTES
0x0568	STAT_TX_PACKET_2048_4095_BYTES
0x0570	STAT_TX_PACKET_4096_8191_BYTES
0x0578	STAT_TX_PACKET_8192_9215_BYTES
0x0580	STAT_TX_PACKET_LARGE
0x0588	STAT_TX_PACKET_SMALL
0x0590 – 0x05B0	Reserved
0x05B8	STAT_TX_BAD_FCS
0x05C0	Reserved
0x05C8	Reserved
0x05D0	STAT_TX_UNICAST
0x05D8	STAT_TX_MULTICAST
0x05E0	STAT_TX_BROADCAST
0x05E8	STAT_TX_VLAN
0x05F0	STAT_TX_PAUSE
0x05F8	STAT_TX_USER_PAUSE
0x0600	Reserved
0x0608	STAT_RX_TOTAL_PACKETS
0x0610	STAT_RX_TOTAL_GOOD_PACKETS
0x0618	STAT_RX_TOTAL_BYTES
0x0620	STAT_RX_TOTAL_GOOD_BYTES
0x0628	STAT_RX_PACKET_64_BYTES
0x0630	STAT_RX_PACKET_65_127_BYTES

Table 5-6: **Status and Statistics Register Map *(Cont'd)***

Address	Register Name
0x0638	STAT_RX_PACKET_128_255_BYTES
0x0640	STAT_RX_PACKET_256_511_BYTES
0x0648	STAT_RX_PACKET_512_1023_BYTES
0x0650	STAT_RX_PACKET_1024_1518_BYTES
0x0658	STAT_RX_PACKET_1519_1522_BYTES
0x0660	STAT_RX_PACKET_1523_1548_BYTES
0x0668	STAT_RX_PACKET_1549_2047_BYTES
0x0670	STAT_RX_PACKET_2048_4095_BYTES
0x0678	STAT_RX_PACKET_4096_8191_BYTES
0x0680	STAT_RX_PACKET_8192_9215_BYTES
0x0688	STAT_RX_PACKET_LARGE
0x0690	STAT_RX_PACKET_SMALL
0x0698	STAT_RX_UNDERSIZE
0x06A0	STAT_RX_FRAGMENT
0x06A8	STAT_RX_OVERSIZE
0x06B0	STAT_RX_TOOLONG
0x06B8	STAT_RX_JABBER
0x06C0	STAT_RX_BAD_FCS
0x06C8	STAT_RX_PACKET_BAD_FCS
0x06D0	STAT_RX_STOMPED_FCS
0x06D8	STAT_RX_UNICAST
0x06E0	STAT_RX_MULTICAST
0x06E8	STAT_RX_BROADCAST
0x06F0	STAT_RX_VLAN
0x06F8	STAT_RX_PAUSE
0x0700	STAT_RX_USER_PAUSE
0x0708	STAT_RX_INRANGEERR
0x0710	STAT_RX_TRUNCATED
0x0718 – 0x07FF	Reserved

Register Descriptions

Table 5-7: **GT_RESET_REG**

Bits	Default	Type	Description
0	0	RW	gt_reset_all. A write of 1 issues a RESET to the GTs.
31:1	0	NA	Reserved

Table 5-8: **RESET_REG**

Bits	Default	Type	Description
9:0	0	RW	rx_serdes_reset. RX IP Phy resets. Unused PCS bits are RESERVED. A write of 1 in a given bit location puts that PCS lane logic into reset.
28:10	0	NA	Reserved
29	0	NA	Reserved
30	0	RW	rx_reset. RX core rest. A write of 1 puts the RX path in reset.
31	0	RW	tx_reset. TX core reset. A write of 1 puts the TX path in reset.

Table 5-9: **MODE_REG**

Bits	Default	Type	Description
0	0	RW	CAUI-4 Mode. A write of 1 puts the 100GE PCS in CAUI-4 mode. A write of 0 puts the 100G PCS in CAUI-10 mode.
31:1	0	NA	Reserved

Table 5-10: **CONFIGURATION_TX_REG1**

Bits	Default	Type	Description
0	0	RW	ctl_tx_enable
3:1	0	NA	Reserved
4	0	RW	ctl_tx_send_rfi
5	0	RW	ctl_tx_send_idle
15:6	0	NA	Reserved
16	0	RW	ctl_tx_test_pattern
31:17	0	NA	Reserved

Table 5-11: **CONFIGURATION_RX_REG1**

Bits	Default	Type	Description
0	0	RW	ctl_rx_enable
6:1	0	NA	Reserved
7	0	RW	ctl_rx_force_resync
8	0	RW	ctl_rx_test_pattern
31:9	0	NA	Reserved

Table 5-12: **CONFIGURATION_TX_BIP_OVERRIDE**

Bits	Default	Type	Description
7:0	0	RW	ctl_tx_lane0_vlm_bip7_override_value
8	0	RW	ctl_tx_lane0_vlm_bip7_override
31:9	0	NA	Reserved

Table 5-13: **CONFIGURATION_TX_FLOW_CONTROL_CONTROL_REG1**

Bits	Default	Type	Description
8:0	0	RW	ctl_tx_pause_enable
31:19	0	NA	Reserved

Table 5-14: **CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG1**

Bits	Default	Type	Description
15:0	0	RW	ctl_tx_pause_refresh_timer0
31:16	0	RW	ctl_tx_pause_refresh_timer1

Table 5-15: **CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG2**

Bits	Default	Type	Description
15:0	0	RW	ctl_tx_pause_refresh_timer2
31:16	0	RW	ctl_tx_pause_refresh_timer3

Table 5-16: **CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG3**

Bits	Default	Type	Description
15:0	0	RW	ctl_tx_pause_refresh_timer4
31:16	0	RW	ctl_tx_pause_refresh_timer5

Table 5-17: **CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG4**

Bits	Default	Type	Description
15:0	0	RW	ctl_tx_pause_refresh_timer6
31:16	0	RW	ctl_tx_pause_refresh_timer7

Table 5-18: **CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG5**

Bits	Default	Type	Description
15:0	0	RW	ctl_tx_pause_refresh_timer8
31:16	0	NA	Reserved

Table 5-19: **CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG1**

Bits	Default	Type	Description
15:0	0	RW	ctl_tx_pause_quanta0
31:16	0	RW	ctl_tx_pause_quanta1

Table 5-20: **CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG2**

Bits	Default	Type	Description
15:0	0	RW	ctl_tx_pause_quanta2
31:16	0	RW	ctl_tx_pause_quanta3

Table 5-21: **CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG3**

Bits	Default	Type	Description
15:0	0	RW	ctl_tx_pause_quanta4
31:16	0	RW	ctl_tx_pause_quanta5

Table 5-22: **CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG4**

Bits	Default	Type	Description
15:0	0	RW	ctl_tx_pause_quanta6
31:16	0	RW	ctl_tx_pause_quanta7

Table 5-23: **CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG5**

Bits	Default	Type	Description
15:0	0	RW	ctl_tx_pause_quanta8
31:16	0	NA	Reserved

Table 5-24: **CONFIGURATION_RX_FLOW_CONTROL_CONTROL_REG1**

Bits	Default	Type	Description
8:0	0	RW	ctl_rx_pause_enable
9	0	NA	Reserved
10	0	RW	ctl_rx_enable_gcp
11	0	RW	ctl_rx_enable_pcp
12	0	RW	ctl_rx_enable_gpp
13	0	RW	ctl_rx_enable_ppp
14	0	NA	Reserved
23:15	0	RW	ctl_rx_pause_ack
31:24	0	NA	Reserved

Table 5-25: **CONFIGURATION_RX_FLOW_CONTROL_CONTROL_REG2**

Bits	Default	Type	Description
0	0	RW	ctl_rx_check_mcast_gcp
1	0	RW	ctl_rx_check_ucast_gcp
2	0	RW	ctl_rx_check_sa_gcp
3	0	RW	ctl_rx_check_etype_gcp
4	0	RW	ctl_rx_check_opcode_gcp
5	0	RW	ctl_rx_check_mcast_pcp
6	0	RW	ctl_rx_check_ucast_pcp
7	0	RW	ctl_rx_check_sa_pcp
8	0	RW	ctl_rx_check_etype_pcp
9	0	RW	ctl_rx_check_opcode_pcp
10	0	RW	ctl_rx_check_mcast_gpp
11	0	RW	ctl_rx_check_ucast_gpp
12	0	RW	ctl_rx_check_sa_gpp
13	0	RW	ctl_rx_check_etype_gpp
14	0	RW	ctl_rx_check_opcode_gpp
15	0	RW	ctl_rx_check_opcode_ppp
16	0	RW	ctl_rx_check_mcast_ppp
17	0	RW	ctl_rx_check_ucast_ppp
18	0	RW	ctl_rx_check_sa_ppp
19	0	RW	ctl_rx_check_etype_ppp
31:20	0	NA	Reserved

Table 5-26: **STAT_TX_STATUS_REG**

Bits	Default	Type	Description
0	0	R/LH	stat_tx_local_fault
31:1	0	R	Reserved

Table 5-27: **STAT_RX_STATUS_REG**

Bits	Default	Type	Description
0	0	R/LL	stat_rx_status
1	0	R/LL	stat_rx_aligned
2	0	R/LH	stat_rx_misaligned
3	0	R/LH	stat_rx_aligned_err
4	0	R/LH	stat_rx_hi_ber
5	0	R/LH	stat_rx_remote_fault
6	0	R/LH	stat_rx_local_fault
7	0	R/LH	stat_rx_internal_local_fault
8	0	R/LH	stat_rx_received_local_fault
11:9	0	R/LH	stat_rx_test_pattern_mismatch
12	0	R/LH	stat_rx_bad_preamble
13	0	R/LH	stat_rx_bad_sfd
14	0	R/LH	stat_rx_got_signal_os
31:15	0	NA	Reserved

Table 5-28: **STAT_STATUS_REG1**

Bits	Default	Type	Description
3:0	0	NA	Reserved
4	0	R/LH	stat_tx_ptp_fifo_read_error
5	0	R/LH	stat_tx_ptp_fifo_write_error
31:6	0	NA	Reserved

Table 5-29: **STAT_RX_BLOCK_LOCK_REG**

Bits	Default	Type	Description
19:0	0	R/LL	stat_rx_block_lock
31:20	0	NA	Reserved

Table 5-30: **STAT_RX_LANE_SYNC_REG**

Bits	Default	Type	Description
19:0	0	R/LL	stat_rx_synced
31:20	0	NA	Reserved

Table 5-31: **STAT_RX_LANE_SYNC_ERR_REG**

Bits	Default	Type	Description
19:0	0	R/LH	stat_rx_synced_err
31:20	0	NA	Reserved

Table 5-32: **STAT_RX_LANE_AM_ERR_REG**

Bits	Default	Type	Description
19:0	0	R/LH	stat_rx_mf_err
31:20	0	NA	Reserved

Table 5-33: **STAT_RX_LANE_AM_LEN_ERR_REG**

Bits	Default	Type	Description
19:0	0	R/LH	stat_rx_mf_len_err
31:20	0	NA	Reserved

Table 5-34: **STAT_RX_LANE_AM_REPEAT_ERR_REG**

Bits	Default	Type	Description
19:0	0	R/LH	stat_rx_mf_repeat_err
31:20	0	NA	Reserved

Table 5-35: **STAT_RX_LANE_DEMUXED**

Bits	Default	Type	Description
19:0	0	R	stat_rx_vl_demuxed
31:20	0	NA	Reserved

Table 5-36: **STAT_RX_PCS_LANE_NUM_REG1**

Bits	Default	Type	Description
4:0	0	R	stat_rx_vl_number_0
9:5	0	R	stat_rx_vl_number_1
14:10	0	R	stat_rx_vl_number_2
19:15	0	R	stat_rx_vl_number_3
24:20	0	R	stat_rx_vl_number_4

Table 5-36: **STAT_RX_PCS_LANE_NUM_REG1 *(Cont'd)***

Bits	Default	Type	Description
29:25	0	R	stat_rx_vl_number_5
31:30	0	NA	Reserved

Table 5-37: **STAT_RX_PCS_LANE_NUM_REG2**

Bits	Default	Type	Description
4:0	0	R	stat_rx_vl_number_6
9:5	0	R	stat_rx_vl_number_7
14:10	0	R	stat_rx_vl_number_8
19:15	0	R	stat_rx_vl_number_9
24:20	0	R	stat_rx_vl_number_10
29:25	0	R	stat_rx_vl_number_11
31:30	0	NA	Reserved

Table 5-38: **STAT_RX_PCS_LANE_NUM_REG3**

Bits	Default	Type	Description
4:0	0	R	stat_rx_vl_number_12
9:5	0	R	stat_rx_vl_number_13
14:10	0	R	stat_rx_vl_number_14
19:15	0	R	stat_rx_vl_number_15
24:20	0	R	stat_rx_vl_number_16
29:25	0	R	stat_rx_vl_number_17
31:30	0	NA	Reserved

Table 5-39: **STAT_RX_PCS_LANE_NUM_REG4**

Bits	Default	Type	Description
4:0	0	R	stat_rx_vl_number_18
9:5	0	R	stat_rx_vl_number_19
31:10	0	NA	Reserved

Table 5-40: **STAT_RX_BIP_OVERRIDE_REG**

Bits	Default	Type	Description
7:0	0	R	stat_rx_lane0_vlm_bip7
8	0	R	stat_rx_lane0_vlm_bip7_valid
31:9	0	NA	Reserved

Table 5-41: **TICK_REG**

Bits	Default	Type	Description
0	0	WO/SC	tick_reg. Writing a 1 to the Tick bit will trigger a snapshot of all the Statistics counters into their readable registers. The bit self-clears, thus only a single write is required.
31:1	0	NA	Reserved

Sample Statistics Counter

A sample statistics counter is illustrated in Table 5-42. The format for the counters is the same for all counter types.

After the 'Tick' is issued, the counters contain their updated value and can be read multiple times without destruction of this data.

Table 5-42: **STAT_RX_BIP_ERR_0[47:0]**

Address	Bits	Default	Type	Description
0x02C0	32	0	R	stat_rx_bip_err_0_lsb[31:0]
0x02C4	16	0	R	stat_rx_bip_err_0_msb[47:32]

Use Case for Different Modes

This section describes the use case for different modes of operation of the 100G Ethernet IP core.

Simulation — Duplex/Simplex RX Mode

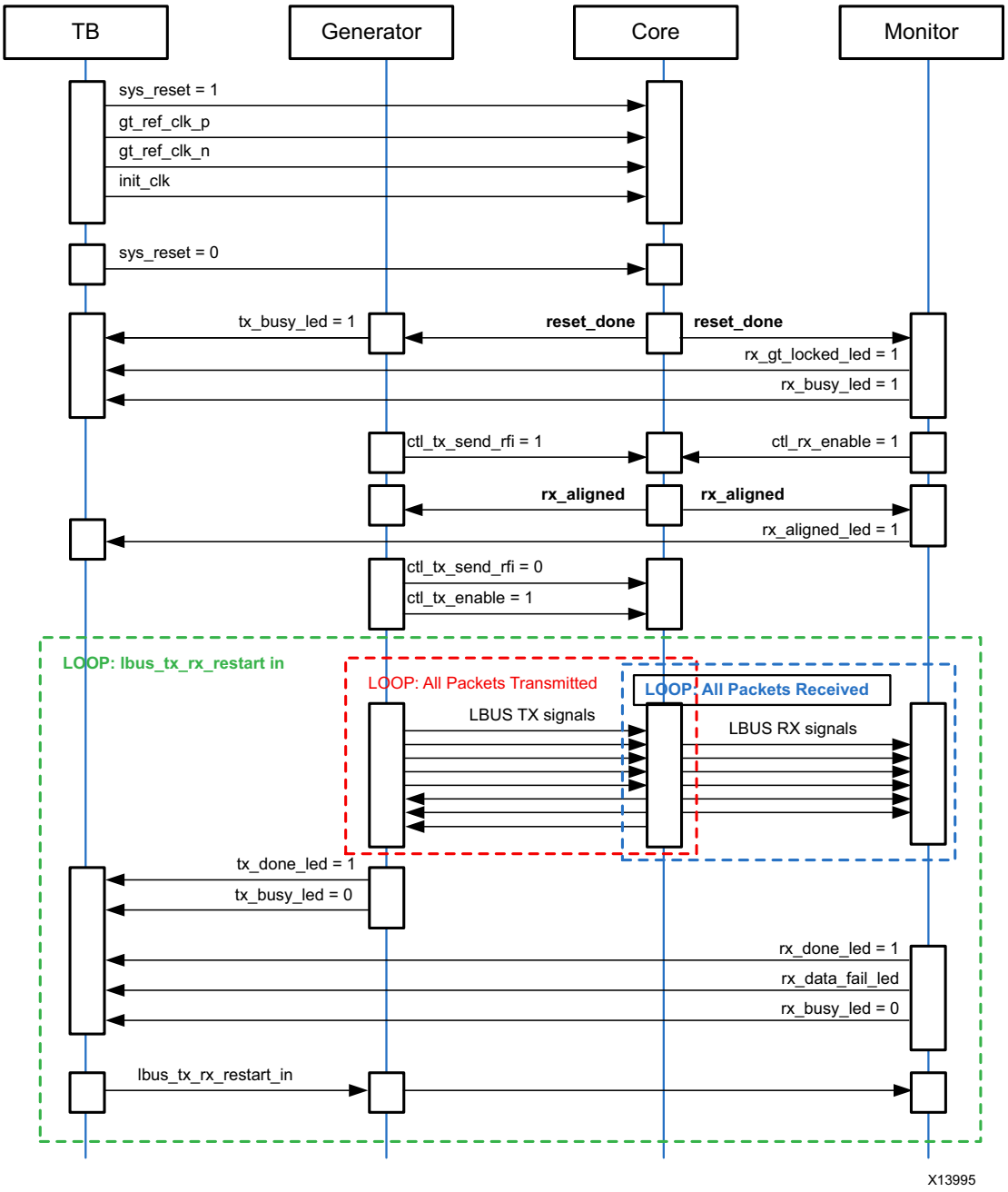

Figure 5-17: **Simulation Use Case for Duplex/Simplex RX Configuration**

Simulation — Simplex TX Mode

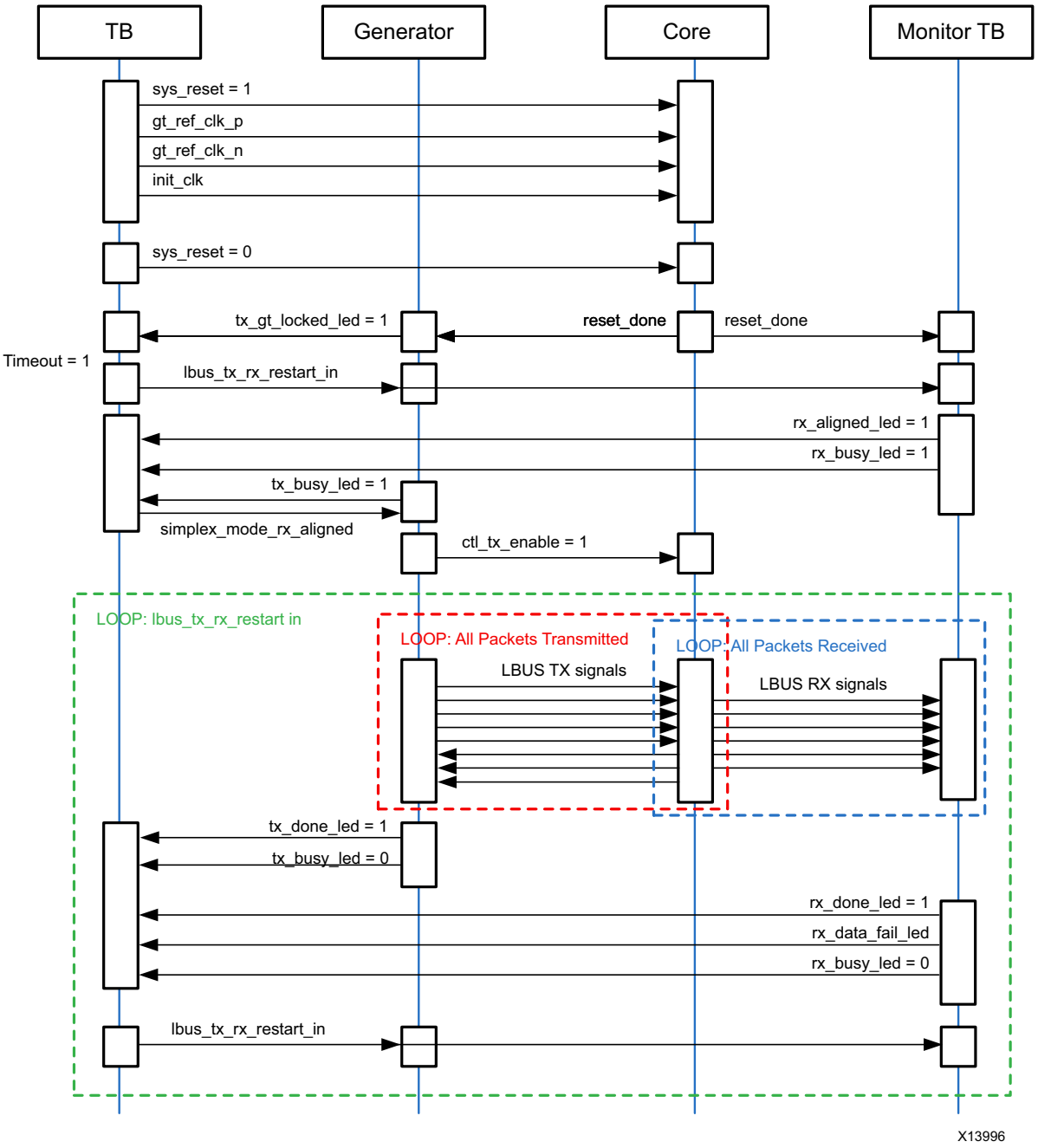

Figure 5-18: **Simulation Use Case for Simplex TX Configuration**

Simulation — Runtime Selectable

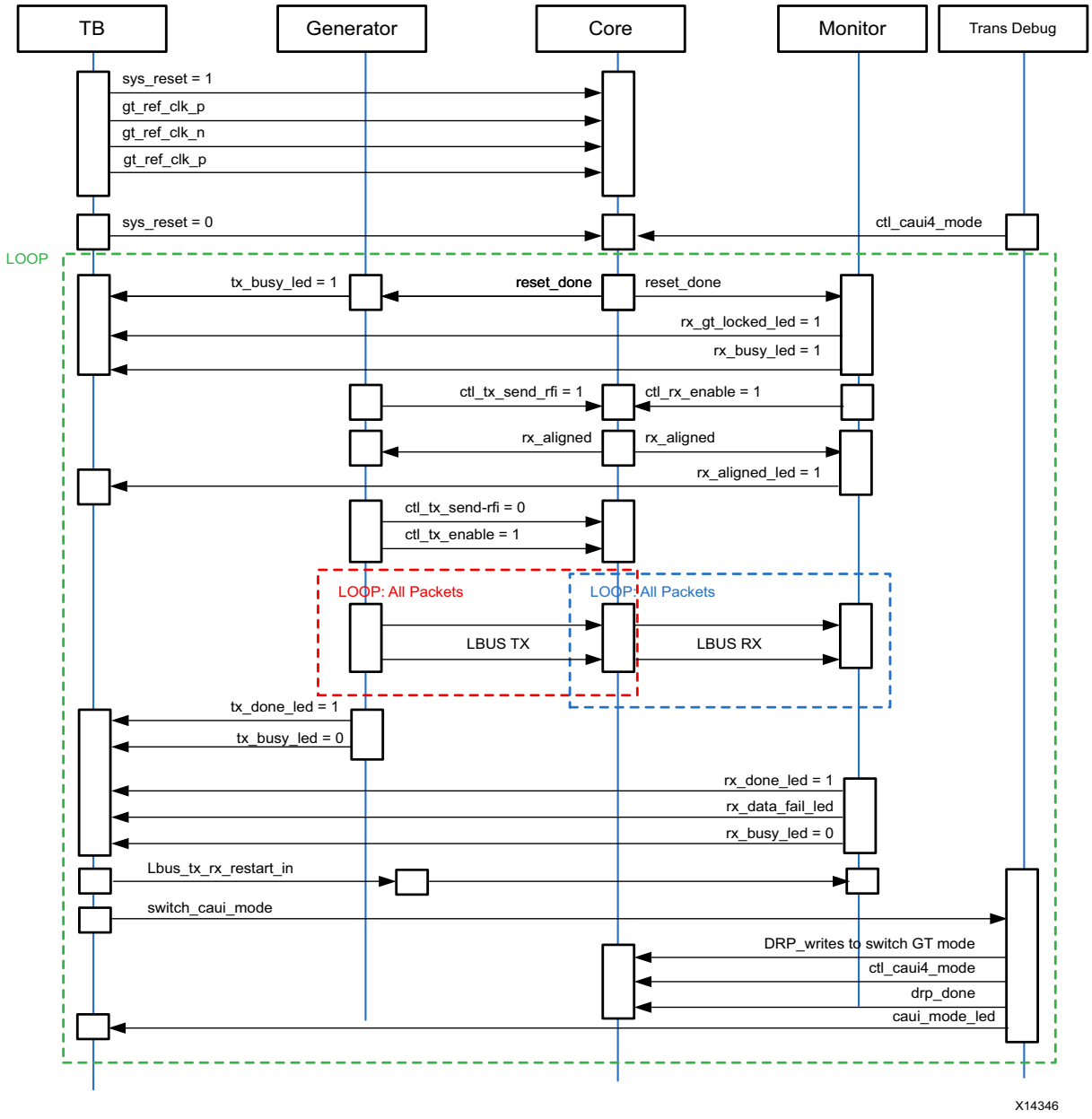

Figure 5-19: **Simulation Use Case for Runtime Selectable Mode**

Validation — Duplex/Simplex RX mode

Figure 5-20 shows the LED behavior and input switch condition for the validation of the 100G Ethernet IP core on the board for the Duplex/Simplex RX mode configuration.

Green color indicates the successful completion of the respective test.

Red color indicates the current process is busy or respective test failed.

Validation — Passing Scenario Duplex/Simplex RX Mode

rx_gt_locked_led	rx_aligned_led	tx_done_led	rx_done_led	rx_data_fail_led	tx_busy_led	rx_busy_led	sys_reset (Switch)	lbus_tx_rx_restart_in (Switch)	Description
○	○	○	○	○	○	○	ON	Off	Board Bring Up
○	○	○	○	○	○	○	Off	Off	On System Reset
●	○	○	○	○	●	●	Off	Off	After GT Locked
●	●	○	○	○	●	●	Off	Off	After Rx_aligned
●	●	●	○	○	○	●	Off	Off	All packets Generated by packet generator
●	●	●	●	○	○	○	Off	Off	All Packets received by the packet monitor without error
●	●	○	○	○	●	●	Off	ON	User restarted the LBUS transaction

X14001

Figure 5-20: **Board Validation for Duplex/Simplex RX Configuration - Passing Scenario**

Validation — Failing Scenario Duplex/Simplex RX Mode

rx_gt_locked_led	rx_aligned_led	tx_done_led	rx_done_led	rx_data_fail_led	tx_busy_led	rx_busy_led	sys_reset (Switch)	lbus_tx_rx_restart_in (Switch)	Description
○	○	○	○	○	○	○	ON	Off	Board Bring Up
○	○	○	○	○	○	○	Off	Off	On System Reset
●	○	○	○	○	●	●	Off	Off	After GT Locked
●	●	○	○	○	●	●	Off	Off	After Rx_aligned
●	●	●	○	○	○	●	Off	Off	All packets generated by packet generator
●	●	●	●	●	○	○	Off	Off	All packets received by the packet monitor without error
●	●	○	○	○	●	●	Off	ON	User restarted the LBUS transaction

X14002

Figure 5-21: **Board Validation for Duplex/Simplex RX Configuration - Failing Scenario**

Validation — Simplex TX Mode

Figure 5-22 describes the LED behavior and input switch condition for the validation of the 100G Ethernet IP core onboard for Simplex TX mode configuration.

Validation — Passing Scenario Simplex TX Mode

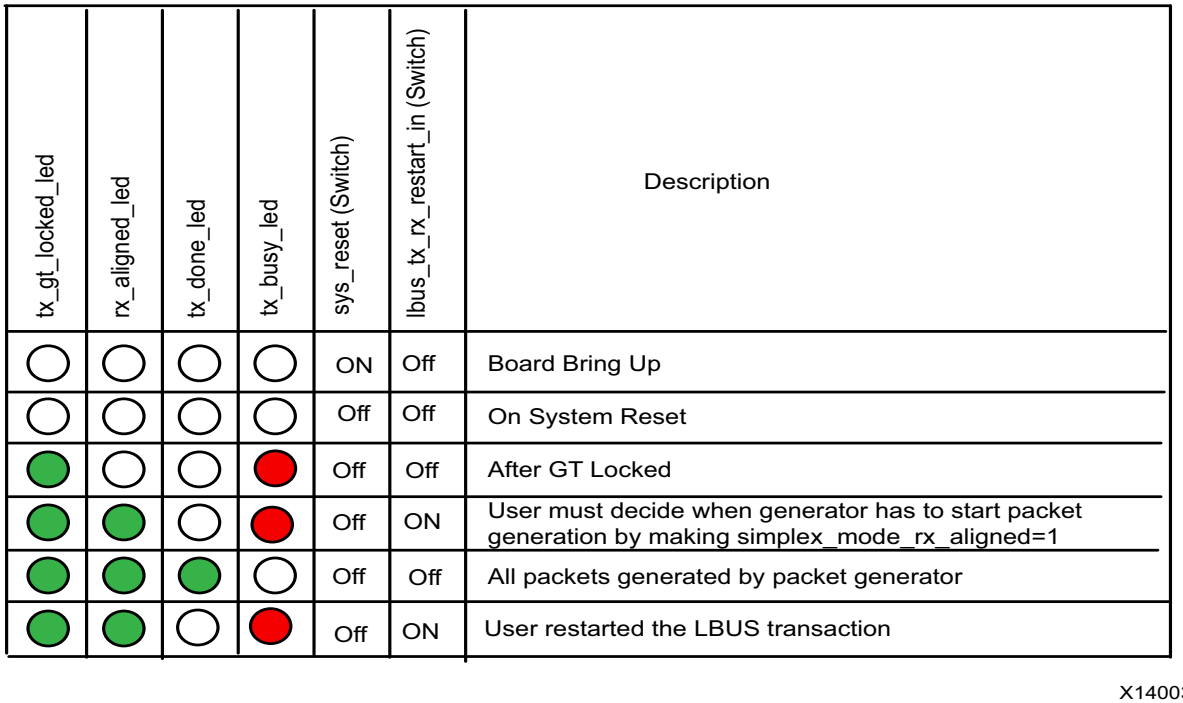

Figure 5-22: **Board Validation for Simplex TX Configuration - Passing Scenario**

Validation — Runtime Selectable Mode (CAUI-10 to CAUI-4) — Passing Scenario

The following table describes the LED behavior and input condition for the validation of the CMAC core on a board for **Runtime Selectable** mode.

rx_gt_locked_led	rx_aligned_led	tx_done_led	caui_mode_led	rx_done_led	rx_data_fail_led	tx_busy_led	rx_busy_led	sys_reset (Switch)	lbus_tx_rx_restart_in (Switch)	switch_caui_mode (Switch)	Description
○	○	○	○	○	○	○	○	ON	Off	Off	Board Bring Up
○	○	○	○	○	○	○	○	Off	Off	Off	On System Reset
●	○	○	○	○	○	●	●	Off	Off	Off	After GT Locked (CAUI-10 mode)
●	●	○	○	○	○	●	●	Off	Off	Off	After Rx_aligned (CAUI-10 mode)
●	●	●	○	○	○	○	●	Off	Off	Off	All packets generated by packet generator for CAUI-10)
●	●	●	○	●	○	○	○	Off	Off	Off	All packets received by the packet monitor without error for CAUI-10
●	●	○	○	○	○	●	●	Off	ON	Off	User restarted the LBUS transaction
●	○	○	●	○	○	●	●	Off	Off	1 Pulse	After GT Locked (CAUI-4 mode)
●	●	○	●	○	○	●	●	Off	Off	Off	After Rx_aligned (CAUI-4 mode)
●	●	●	●	○	○	○	●	Off	Off	Off	All packets generated by packet generator for CAUI-4)
●	●	●	●	●	○	○	○	Off	Off	Off	All packets received by the packet monitor without error for CAUI-4
●	●	○	●	○	○	●	●	Off	ON	Off	User restarted the LBUS transaction

X14004

Figure 5-23: **Board Validation for Runtime Selectable Configuration - Passing Scenario**

Simulating the Example Design

The example design provides a quick way to simulate and observe the behavior of the 100G Ethernet IP core example design projects generated using the Vivado Design Suite.

The currently supported simulators are:

- Vivado simulator (default)
- Mentor Graphics Questa Simulator (QuestaSim)/ModelSim (integrated in the Vivado IDE)
- Cadence Incisive Enterprise Simulator (IES)
- Synopsys VCS and VCS MX

The simulator uses the example design test bench and test cases provided along with the example design.

For any project (100G Ethernet IP core) generated out of the box, the simulations can be run in the following ways:

1. In the Sources Window, right-click the example project file (.xci), and select **Open IP Example Design**. The example project is created.
2. In the Flow Navigator (left-hand pane), under Simulation, right-click **Run Simulation** and select **Run Behavioral Simulation**.

   Note: The post-synthesis and post-implementation simulation options are not supported for the 100G Ethernet IP core.

   After the Run Behavioral Simulation Option is running, you can observe the compilation and elaboration phase through the activity in the **Tcl Console**, and in the **Simulation** tab of the **Log** Window.
3. In **Tcl Console**, type the run all command and press **Enter**. This runs the complete simulation as per the test case provided in example design test bench.

   After the simulation is complete, the result can be viewed in the **Tcl Console**.

To change the simulators:

1. In the Flow Navigator, under Simulation, select **Simulation Settings**.
2. In the Project Settings for Simulation dialog box, change the Target Simulator to **Questa Sim/ModelSim**.
3. When prompted, click **Yes** to change and then run the simulator.

Synthesizing and Implementing the Example Design

To run synthesis and implementation on the example design in the Vivado Design Suite, do the following steps:

1. Go to the XCI file, right-click, and select **Open IP Example Design**.

   A new Vivado tool window opens with the project name "example_project" within the project directory.
2. In the Flow Navigator, click **Run Synthesis** and **Run Implementation**.

TIP: *Click **Run Implementation** first to run both synthesis and implementation.*

Click **Generate Bitstream** to run synthesis, implementation, and then bitstream.

Migrating and Upgrading

This appendix contains information about upgrading to a more recent version of the IP core.

Migrating to the Vivado Design Suite

This section does apply to this core.

Upgrading in the Vivado Design Suite

This section provides information about any updates to the user logic or port designations between core versions.

Changes from v1.5 to v1.6

Port Changes

There were no port or parameter changes from v1.5 to v1.6.

Other Changes

- Updated AXI4 to AXI4-Lite in the General Configuration tab.
- Updated to support AXI4-Lite Simplex and Duplex.

Debugging

This appendix includes details about resources available on the Xilinx Support website and debugging tools.

Finding Help on Xilinx.com

To help in the design and debug process when using the 100G Ethernet IP core, the Xilinx Support web page contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support.

Documentation

This product guide is the main document associated with the 100G Ethernet MAC core. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page or by using the Xilinx Documentation Navigator.

Download the Xilinx® Documentation Navigator from the Design Tools tab on the Downloads page. For more information about this tool and the features available, open the online help after installation.

Solution Centers

See the Xilinx Solution Centers for support on devices, design tools, and intellectual property at all stages of the design cycle. Topics include design assistance, advisories, and troubleshooting tips.

The Solution Center specific to the 100G Ethernet MAC core is Xilinx Ethernet IP Solution Center.

Answer Records

Answer Records include information about commonly encountered problems, helpful information on how to resolve these problems, and any known issues with a Xilinx product. Answer Records are created and maintained daily ensuring that users have access to the most accurate information available.

Answer Records for this core can be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use proper keywords such as

- Product name
- Tool message(s)
- Summary of the issue encountered

A filter search is available after results are returned to further target the results.

Master Answer Record for the 100G Ethernet MAC core

AR: 58696

Contacting Technical Support

Xilinx provides technical support at the Xilinx support web page for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled DO NOT MODIFY.

To contact Xilinx Technical Support:

1. Navigate to the Xilinx support web page.
2. Open a WebCase by selecting the WebCase link located under Additional Resources.

When opening a WebCase, include:

- Target FPGA including package and speed grade.
- All applicable Xilinx design tools and simulator software versions.
- Additional files based on the specific issue might also be required. See the relevant sections in this debug guide for guidelines about which file(s) to include with the WebCase.

Note: Access to WebCase is not available in all cases. Log in to the WebCase tool to see your specific support options.

*

Vivado Design Suite Debug Feature

The Vivado® Design Suite debug feature inserts logic analyzer and virtual I/O cores directly into your design. The debug feature also allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature in the Vivado IDE is used for logic debugging and validation of a design running in Xilinx devices.

The Vivado logic analyzer is used with the logic debug IP cores, including:

- ILA 2.0 (and later versions)
- VIO 2.0 (and later versions)

See the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 12].

Simulation Debug

The 100G Ethernet IP core example design includes a sample simulation test bench. This consists of a loopback from the TX side of the user interface, through the TX circuit, looping back to the RX circuit, and checking the received packets at the RX side of the user interface.

This section contains details about items that should be checked if the simulation does not run properly from the scripts.

Slow Simulation

Simulations can appear to run slowly under some circumstances. If a simulation is unacceptably slow, the following suggestions can improve the run-time performance.

- Use a faster computer with more memory.
- Make use of a Platform LSF (Load Sharing Facility), if available.
- Bypass the Xilinx transceiver (this might require creating your own test bench).
- Send fewer packets. This can be accomplished by modifying the appropriate parameter in the provided sample test bench.
- Specify a shorter time between alignment markers. This should result in a shorter lane alignment phase at the expense of more overhead. However, when the 100G Ethernet IP core is implemented in hardware, the distance between alignment markers should follow the specification recommendations (after every 16,383 words).
  Contact Xilinx technical support for assistance if required.

Simulation Fails Before Completion

If the sample simulation fails or hangs before successfully completing, then it is possible that a timeout has occurred. Ensure that the simulator timeouts are long enough to accommodate the waiting periods in the simulation, for example during the lane alignment phase.

Simulation Completes But Fails

If the sample simulation completes with a failure, contact Xilinx technical support. The test will normally complete successfully. Consult the sample simulation log file for the expected behavior.

Hardware Debug

Hardware issues range from link bring-up to problems seen after hours of testing. This section provides debug steps for common issues.

General Checks

Ensure that all the timing constraints for the core are properly incorporated from the example design and that all constraints are met during implementation.

- Does it work in post-place and route timing simulation? If problems are seen in hardware but not in timing simulation, this could indicate a PCB issue. Ensure that all clock sources are active and clean.
- If using mixed-mode clock managers (MMCMs) in the design, ensure that all MMCMs have obtained lock by monitoring the LOCKED port.

Ethernet Specific Checks

Many issues can occur during the first hardware test. This section details the debugging process. It is assumed that the 100G Ethernet IP core has already passed all simulation testing which is being implemented in hardware. This is a pre-requisite for any kind of hardware debug.

The following sequence helps to isolate ethernet-specific problems:

1. Clean up Signal Integrity.
2. Ensure that each SerDes achieves clock data recovery (CDR) lock.
3. Check that each lane has achieved word alignment.
4. Check that lane alignment has been achieved.
5. Proceed to Interface Debug and Protocol Debug.

Signal Integrity

If you are bringing up a board for the first time and the 100G Ethernet IP core does not seem to be achieving lane alignment, the most likely problem is related to signal integrity. Signal integrity issues must be addressed before any other debugging can take place.

Even if lane alignment is achieved, periodic BIP8 errors create signal integrity issues. Check the BIP8 signals to assist with debugging.

IMPORTANT: *It assumed that the PCB itself has been designed and manufactured in accordance with the required trace impedances and trace lengths, including the requirements for skew set out in the IEEE 802.3 specification.)*

Signal integrity should be debugged independently from the 100G Ethernet IP core. The following checks should be made:

- Transceiver Settings
- Checking For Noise
- Bit Error Rate Testing

If assistance is required for transceiver and signal integrity debugging, contact Xilinx technical support.

Lane Swapping

In Ethernet, physical lanes can be swapped and the protocol will align lanes correctly. Therefore lane swapping should not cause any problems.

N/P Swapping

If the positive and negative signals of a differential pair are swapped, data will not be correctly received on that lane. Verify that each link has the correct polarity of each differential pair.

Clocking and Resets

See Clocking and Resets in Chapter 3 for these requirements.

Ensure that the clock frequencies for both the 100G Ethernet IP core as well as the Xilinx transceiver reference clock match the configuration requested when the IP core was ordered. The core clock has a minimum frequency associated with it. The maximum core clock frequency is determined by timing constraints. The minimum core clock frequency is derived from the required Ethernet bandwidth plus the margin reserved for clock tolerance, wander and jitter.

The first thing to verify during debugging is to ensure that resets remain asserted until the clock is stable. It must be frequency-stable as well as free from glitches before the 100G Ethernet IP core is taken out of reset. This applies to both the SerDes clock as well as the IP core clock.

If any subsequent instability is detected in a clock, the 100G Ethernet IP core must be reset. One example of such instability is a loss of CDR lock. The user logic should determine all external conditions that would require a reset (for example, clock glitches, loss of CDR lock, or power supply glitches).

Configuration changes cannot be made unless the IP core is reset. An example of a configuration change would be setting a different maximum packet length. Check the description for the particular signal on the port list to determine if this requirement applies to the parameter that is being changed (Table 2-2).

Interface Debug

The 100G Ethernet IP core user interface is the segmented LBUS (Local bus). This section details debugging information for the TX and RX interfaces.

TX Debug

TX debugging is assisted using several diagnostic signals. See Table 2-2 for more details.

Data must be written to the TX LBUS so that there are no overflow or underflow conditions.

The LBUS bandwidth must always be greater than the Ethernet bandwidth to guarantee that data can be sent without interruption.

When writing data to the LBUS, the `tx_rdyout` signal must always be observed. This signal indicates whether the fill level of the TX buffer is within an acceptable range or not. If this signal is ever asserted, you must stop writing to the TX LBUS until the signal is deasserted. Because the TX LBUS has greater bandwidth than the TX Ethernet interface, it is not unusual to see this signal being frequently asserted and this is not a cause for concern. You must ensure that TX writes are stopped when `tx_rdyout` is asserted.

The level at which `tx_rdyout` becomes asserted is set by a pre-determined threshold.

IMPORTANT: *If `tx_rdyout` is ignored, the signal `tx_ovfout` might be asserted, indicating a buffer overflow. This should be prevented. Xilinx recommends that the core be reset if `tx_ovfout` is asserted. Do not attempt to continue debugging after `tx_ovfout` has been asserted until the cause of the overflow has been addressed.*

When a packet data transaction has begun in the TX direction, it must continue until completion or there might be a buffer underflow as indicated by the signal `stat_tx_underflow_err`. This must not be allowed to occur. Data must be written on the TX LBUS without interruption. Ethernet packets must be present on the line from start to end with no gaps or idles. If `stat_tx_underflow_err` is asserted, debugging must stop until the condition which caused the underflow has been addressed.

RX Debug

See the port list in Table 2-2 for a complete description of the diagnostic signals that are available to debug the RX.

If the Ethernet packets are being transmitted properly according to the 802.3 protocol, there should not be RX errors. However, the signal integrity of the received signals must be verified first.

The `stat_rx_bip_err` signals provide a per-lane indicator of signal quality. The `stat_rx_hi_ber` signal is asserted when the bit error rate is too high, according to the 802.3 protocol. The threshold is BER = 10-4.

To aid in debug, a local loopback can be performed at the transceiver level. This connects the TX SerDes to the RX SerDes and bypasses potential signal integrity problems. The received data can be checked against the transmitted packets to verify that the logic is operating properly.

Protocol Debug

To achieve error-free data transfers with the 100G Ethernet IP core, the 802.3 specification should be followed. Note that signal integrity should always be ensured before proceeding to the protocol debug.

Alignment Marker Spacing

According to the 802.3 specification, the alignment marker spacing should be set to 16,383 for both the TX and RX. Check that both ends of the link are programmed to this value.

Diagnostic Signals

There are many error indicators available to check for protocol violations. Carefully read the description of each one to see if it is useful for a particular debugging issue. See Table 2-2 for more details.

The following is a suggested debug sequence.

1. Ensure that Word sync has been achieved.
2. Ensure that Lane sync has been achieved (this uses the lane marker alignment words which occur after every 16,383 words).
3. Verify that the BIP8 indicators are clean.
4. Make sure there are no descrambler state errors.
5. Eliminate CRC32 errors, if any.
6. Make sure the LBUS protocol is being followed correctly.
7. Ensure that there are no overflow or underflow conditions when packets are sent.

Statistics Counters

When error-free communication has been achieved, the statistics indicators can be monitored to ensure that traffic characteristics meet expectations. Some signals are strobes only, which means that the counters are not part of the core. This is done so that the counter size can be customized. The counters are optional.

Appendix C

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

References

These documents provide supplemental material useful with this product guide:

1. *IEEE 1588-2008* (standards.ieee.org/findstds/standard/1588-2008.html)
2. *IEEE std 802.3-2012* (standards.ieee.org/findstds/standard/802.3-2012.html)
3. *Virtex UltraScale Architecture Data Sheet: DC and AC Switching Characteristics* (DS893)
4. *UltraScale FPGAs Transceiver Wizards* (PG182)
5. *UltraScale Architecture Clocking Resource User Guide* (UG572)
6. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
7. *Vivado Design Suite User Guide: Designing with IP* (UG896)
8. *Vivado Design Suite User Guide: Getting Started* (UG910)
9. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
10. *Vivado Design Suite User Guide: Using Constraints* (UG903)
11. *ISE to Vivado Design Suite Migration Guide* (UG911)
12. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
13. *Vivado Design Suite User Guide: Implementation* (UG904)
14. *UltraScale FPGAs GTH Transceivers User Guide* (UG576)
15. *UltraScale FPGAs GTY Transceivers User Guide* (UG578)

Revision History

Date	Version	Revision
06/24/2015	1.6	**Chapter 3: Design Flow Steps** • Updated Figure 3-1 through Figure 3-4. **Chapter 4: Design Flow Steps** • Updated screen captures: Figures 4-1 through 4-3. **Chapter 5: Example Design** • Updated Figure 5-1, Figure 5-7, Figure 5-11, Figure 5-17, Figure 5-18, Figure 5-19, Figure 5-20, Figure 5-21, Figure 5-22, Figure 5-23 • Updated switch_caui_mode description in Table 5-1. • Replaced tx_fail_led with caui_mode_led in Table 5-1. • Updated descriptions for STATE_GT_LOCKED, STATE_LBUS_TX_DONE, STATE_WAIT_FOR_RESTART, STATE_LBUS_RX_DONE, STATE_PKT_TRANSFER_INIT. • Removed STATE_SWITCH_CMAC_MODE • Changed Figure 5-23 title to "Board Validation for Runtime Selectable configuration - Passing Scenario."
04/01/2015	1.5	**Chapter 4: Design Flow Steps** • Updated Figures 4-1, 4-2, and 4-3. • Updated Table 4-3: GT Selections and Configuration • Added Enable AXI4-Lite Interface parameter to Table 4-1:General Configuration Tab **Chapter 5: Example Design** • Updated lbus_tx_rx_restart_in description in Table 5-1. • Added two new ports to Table 5-1: simplex_mode_rx_aligned and switch_caui_mode. • Added cmac_0_axi4_lite_if_wrapper, cmac_0_axi4_lite_user_if, STATE_PTP_PKT_INIT, STATE_PTP_PKT_READ, STATE_TX_PTP_PKT_TRANSFER, STATE_RX_PTP_ENABLE, and STATE_RX_PTP_DONE • Updated all of the figures. • Added Simplex TX Mode Simulation, Simplex RX Mode Simulation sections. • Added extensive Notes. • Added new section, AXI4-Lite Interface.
01/22/2015	1.4	• Updated the Vivado® IDE screen captures. • Updated the clocking and reset diagram. • Corrected the maximum packet length.
10/01/2014	1.3	• Updated information on the segmented LBUS in Chapter 3, "Designing with the Core." • Removed Appendix C, Segmented LBUS Protocol.
06/04/2014	1.2	• Updated core to v1.2. • Added example design clocking information.

Date	Version	Revision
04/02/2014	1.1	• Added DRP blocks to Figure 2-1. • Added transceiver selection rules. • Updated General Configuration table • Added resources and performance characteristics. • Updated screen displays in Chapter 4, Figure 5-1, and Figure 5-5. • Provided description of other optional modules instantiated in the example design. • Updated Control/Pause Packet Processing table. • Updated GT Selections and Configurations table. • Added new constraints information. • Added DRP information. • Added Shared Logic Implementation and Runtime Selectable sections.
12/18/2013	1.0	Initial Xilinx release.

Please Read: Important Legal Notices