10G/25G High Speed Ethernet Subsystem v2.4

Product Guide

Vivado Design Suite

PG210 June 6, 2018

Table of Contents

Introduction

The Xilinx® 10G/25G High Speed Ethernet Subsystem implements the 25G Ethernet Media Access Controller (MAC) with a Physical Coding Sublayer (PCS) as specified by the 25G Ethernet Consortium. MAC and physical coding sublayer/physical medium attachment (PCS/PMA) or PCS/PMA alone are available. Legacy operation at 10 Gb/s is supported.

Features

- Designed to the Ethernet requirements for 10/25 Gb/s operation specified by IEEE 802.3 Clause 49, IEEE 802.3by, and the 25G Ethernet Consortium
- Includes complete Ethernet MAC and PCS/PMA functions or standalone PCS/PMA for 25 Gb/s operation
- Includes complete Ethernet MAC and PCS/PMA functions, standalone MAC or standalone PCS/PMA for 10 Gb/s operation. Includes standalone 64-bit Ethernet MAC
- Simple packet-oriented user interface
- Comprehensive statistics gathering
- Status signals for all major functional indicators
- Delivered with a top-level wrapper including functional transceiver wrapper, IP netlist, sample test scripts, and Vivado® Design Suite tools compile scripts
- BASE-R PCS sublayer operating at 10.3125 Gb/s or 25.78125 Gb/s
- Optional clause 74 BASE-KR FEC sublayer
- Optional Auto-Negotiation
- Optional clause 108 25G Reed-Solomon Forward Error Correction (RS-FEC) sublayer
- Custom Preamble mode
- Optional IEEE 1588 1-step and 2-step timestamping
- Runtime switchable between 10G and 25G
- Optional fee-based Time Sensitive Networking (TSN) feature designed to IEEE standard 802.1 CM
  - Supports interspersing express traffic with low priority traffic
  - Supports frame preemption

Facts Table	
Core Specifics	
Supported Device Family[1]	Zynq® UltraScale+™ MPSoC Virtex® UltraScale+, Kintex® UltraScale+ Virtex UltraScale™, Kintex UltraScale
Supported User Interfaces	AXI4-Stream for Variants with MAC XGMII or 25GMII for PCS-only variants
Resources	Performance and Resource Utilization web page
Provided with Core	
Design Files	Encrypted register transfer level (RTL)
Example Design	Verilog
Test Bench	Verilog
Constraints File	Xilinx Design Constraints (XDC)
Simulation Model	Verilog
Supported S/W Driver	Linux
Tested Design Flows[2]	
Design Entry	Vivado Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Synopsis or 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.

Note: To access the 25G specification, go to the 25G Ethernet Consortium website.

Overview

This document details the features of the 10G/25G Ethernet Subsystem as defined by the 25G Ethernet Consortium [Ref 1]. PCS functionality is defined by *IEEE Standard 802.3, 2015, Clause 49, Physical Coding Sublayer (PCS) for 64B/66B, type 10GBASE-R* [Ref 2]. For 25G operation, clock frequencies are increased to provide a serial interface operating at 25.78125 Gb/s to leverage the latest high-speed serial transceivers. The low latency design is optimized for UltraScale™ architecture devices.

Feature Summary

See Table 1-1 for compatibility of options with the different variants of the LogiCORE™ IP core.

25G Supported Features

- Complete Ethernet MAC and PCS functions
- Designed to Schedule 3 of the 25G Consortium
- Statistics and diagnostics
- 66-bit serializer/deserializer (SerDes) interface using the Xilinx® GTY transceiver operating with Asynchronous Gearbox enabled
- Pause Processing including *IEEE std. 802.3* Annex 31D (Priority based Flow Control)
- Low latency
- Custom preamble and adjustable Inter Frame Gap
- Configurable for operation at 10.3125 Gb/s (Clause 49)
- Optional Clause 73 Auto-negotiation
- Optional Clause 72.6.10 Link Training
- Optional Clause 74 FEC – shortened cyclic code (2112, 2080)
- Optional clause 108 25G Reed-Solomon Forward Error Correction (RS-FEC) sublayer
- PCS only version with 25GMII Interface

- 64-bit AXI4-Stream Interface
- Optional AXI4-Lite control and status interface

10G Supported Features

- Complete MAC and PCS functions
- Base-KR mode based on IEEE 802.3 Clause 49
- Pause Processing
- Optional 64-bit or 32-bit AXI4-Stream user interfaces
- Optional Standalone MAC with 64-bit AXI4-Stream interface and XGMII pin out
- Optional Clause 73 Auto-negotiation
- Optional Clause 72.6.10 Link Training
- Optional Clause 74 FEC - shortened cyclic code (2112, 2080)
- PCS only version with XGMII Interface
- Optional AXI4-Lite control and status interface
- Statistics and diagnostics
- 66-bit SerDes interface
- Custom preamble and adjustable Inter Frame Gap
- Supports 802.1CM (802.3br/802.1bu) preemption feature for MAC+PCS/PMA 64-bit Base-R

10G/25G Runtime Switchable IP features

- Complete MAC and PCS features
- Base-KR mode based on IEEE 802.3 Clause 49
- Statistics and Diagnostics
- 66-bit SerDes interface using Xilinx GTY transceiver
- Custom preamble and adjustable InterFrame Gap
- Pause Processing
- 64-bit AXI4-streaming interface
- Optional Clause 73 Auto-negotiation
- Optional Clause 72 Link Training
- Optional Clause 74 FEC

Table 1-1: **Feature Compatibility Matrix**

Variant	**User Interface**	**MAC**	**PCS**	**Pause Processing**	**Auto-Negotiation and Link Training**	**Clause 74 FEC**	**Clause 108 RS-FEC**	**IEEE 1588 Time Stamp**
10G MAC with PCS	64-bit AXI4-Stream	X	X	X	X	X		X
Low Latency 10G MAC with PCS	64-bit AXI4-Stream	X	X					
Low Latency 10G MAC with PCS	32-bit AXI4-Stream	X	X		X(2)			
Low Latency 25G MAC with PCS	64-bit AXI4-Stream	X	X					
25G MAC with PCS	64-bit AXI4-Stream	X	X	X	X	X	X	X
Runtime switchable 10G/ 25G MAC+PCS	64-bit AXI4-Stream	X	X	X	X	X	X	X
10G PCS-only	XGMII		X		X	X		X(1)
25G PCS-only	25GMII		X		X	X	X	X(1)
Runtime switchable 10/ 25G PCS-only	XGMII/ 25GMII		X		X	X	X	X(1)
10G MAC-only	64-bit AXI4-Stream	X		X				
Preemption (802.1CM) 10G only	64-bit AXI4-Stream	X	X		X			X

Notes:

1. Only 2-step timestamping is supported with the PCS-only configurations.
2. Only auto-negotiation logic.

Applications

IEEE Std 802.3 enables several different Ethernet speeds for Local Area Network (LAN) applications, and 25 Gb/s is the latest addition to the standard. The capability to interconnect devices at 25 Gb/s Ethernet rates becomes especially relevant for next-generation data center networks where:

- To keep up with increasing CPU and storage bandwidth, rack or blade servers need to support aggregate throughputs faster than 10 Gb/s (single lane) or 20 Gb/s (dual lane) from their Network Interface Card (NIC) or LAN-on-Motherboard (LOM) networking ports;
- Given the increased bandwidth to endpoints, uplinks from Top-of-Rack (TOR) or Blade switches need to transition from 40 Gb/s (four lanes) to 100 Gb/s (four lanes) while ideally maintaining the same per-lane breakout capability;
- (Due to the expected adoption of 100GBASE-CR4/KR4/SR4/LR4, SerDes and cabling technologies are already being developed and deployed to support 25 Gb/s per physical lane, twisted pair, or fiber.

Licensing and Ordering

License Checkers

If the IP requires a license key, the key must be verified. The Vivado® design tools have several license checkpoints for gating licensed IP through the flow. If the license check succeeds, the IP can continue generation. Otherwise, generation halts with error. License checkpoints are enforced by the following tools:

- Vivado Synthesis
- Vivado Implementation
- write_bitstream (Tcl Console command)

IMPORTANT: *IP license level is ignored at checkpoints. The test confirms a valid license exists. It does not check IP license level.*

License Type

10G/25G Ethernet PCS/PMA (10G/25G BASE-R)

This Xilinx 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 IP modules is available at the Xilinx Intellectual Property page. For information about pricing and availability of other Xilinx IP modules and tools, contact your local Xilinx sales representative. For more information, visit the 10G/25G Ethernet Subsystem page.

Standalone 10G/25G Ethernet MAC and PCS/PMA (10G/25G EMAC + 10G/25G BASE-R/KR), 10G/25G BASE-KR and 10G TSN IEEE802.1CM

Note: The 10G/25G Ethernet MAC + BASE-R, 10GBASE-KR/25GBASE-KR and 10G TSN IP features require separate fee-based licensing.

These Xilinx IP module is provided under the terms of the Xilinx Core License Agreement. The module is shipped as part of the Vivado Design Suite. For full access to all core functionalities in simulation and in hardware, you must purchase one or more licenses for the core. Contact your local Xilinx sales representative for information about pricing and availability. For more information, visit the 10G/25G Ethernet Subsystem page.

Ordering Information

To purchase any of these IP cores, contact your local Xilinx Sales Representative referencing the appropriate part number(s) in Table 1-2.

Table 1-2: **Ordering Information**

Description	Part Number	License Key
• 10G/25G Ethernet MAC + BASE-R PCS/PMA (64-bit) • 10E MAC + BASE-R PCS/PMA (32-bit)	EF-DI-25GEMAC-PROJ[1] EF-DI-25GEMAC-SITE[1] ***Note:*** These part numbers do not include the BASE-KR/CR/SR functionality which is comprised of FEC and AN/LT. To include support for this feature, please order EF-DI-25GBASE-KR-PROJ or EF-DI-25GBASE-KR-SITE.	xxv_eth_mac_pcs x_eth_mac
• 25GBASE-KR PCS/PMA (CL108 RS-FEC, CL74 FEC, AN)[2] • 10GBASE-KR PCS/PMA (CL74 FEC, AN) Standalone PCS/PMA only	EF-DI-25GBASE-KR-PROJ[2][3] EF-DI-25GBASE-KR-SITE[2][3] ***Note:*** The 10G/25G Ethernet MAC is sold separately. To include the Xilinx MAC, also order the EF-DI-25GEMAC-PROJ or EF-DI-25GEMAC-SITE.	xxv_eth_basekr ieee802d3_25g_rs_fec_full ieee802d3_25g_rs_fec_only[4]
• All of the configurations in row 1 and 2 are included plus 10G/25G Ethernet MAC + BASE-KR PCS/PMA (64-bit).	EF-DI-25GEMAC-PROJ[1] EF-DI-25GEMAC-SITE [1] and EF-DI-25GBASE-KR-PROJ[2][3] EF-DI-25GBASE-KR -SITE[2][3]	xxv_eth_mac_pcs x_eth_mac xxv_eth_basekr ieee802d3_25g_rs_fec_full ieee802d3_25g_rs_fec_only[4]
• 25G/10GBASE-R PCS/PMA only • 10GBASE-R PCS/PMA only • Switchable 25G and 10G BASE-R PCS/PMA	Included with the Vivado design tools. No purchase necessary.	No license key
• 25G/10G Ethernet MAC+PCS/PMA • 10G Ethernet TSN 802.1CM (Preemption) Feature	EF-DI-25G-TSN-802-1-CM-PROJ ***Note:*** Bundle includes Ethernet MAC IP. Current support in Vivado is for 10GE TSN only. For 25G inquiries, please email ethernet_mgmt@xilinx.com.	xxv_tsn_802d1cm xxv_eth_mac_pcs x_eth_mac

Notes:

1. Includes access to legacy 10 Gigabit Ethernet Media Access Controller - 10GEMAC for 7 series and UltraScale devices (key name: ten_gig_eth_mac).
2. Used for 25GBASE-CR, 25GBASE-KR, or 25GBASE-SR applications.
3. Includes access to 10 Gigabit Ethernet PCS/PMA with FEC/Auto-Negotiation - 10GBASE-KR for 7 series and UltraScale devices (key name: ten_gig_eth_pcs_pma_basekr).
4. iee802d3_25g_rs_fec_only license key enables transcode bypass mode for custom applications only. For more information, refer to the *25G IEEE 802.3by Reed-Solomon Forward Error Correction LogiCORE IP Product Guide* (PG217) [Ref 12]. To request access, email ethernet_mgmt@xilinx.com. If you do not require a custom standalone FEC only implementation without MAC and/or PCS, ignore the ieee802d3_25g_rs_fec_only license key warning message in Vivado.

Product Specification

Figure 2-1 and Figure 2-2 show the block diagrams of the 10G/25G Ethernet IP Subsystem, respectively.

Note: GT transceivers are not shown.

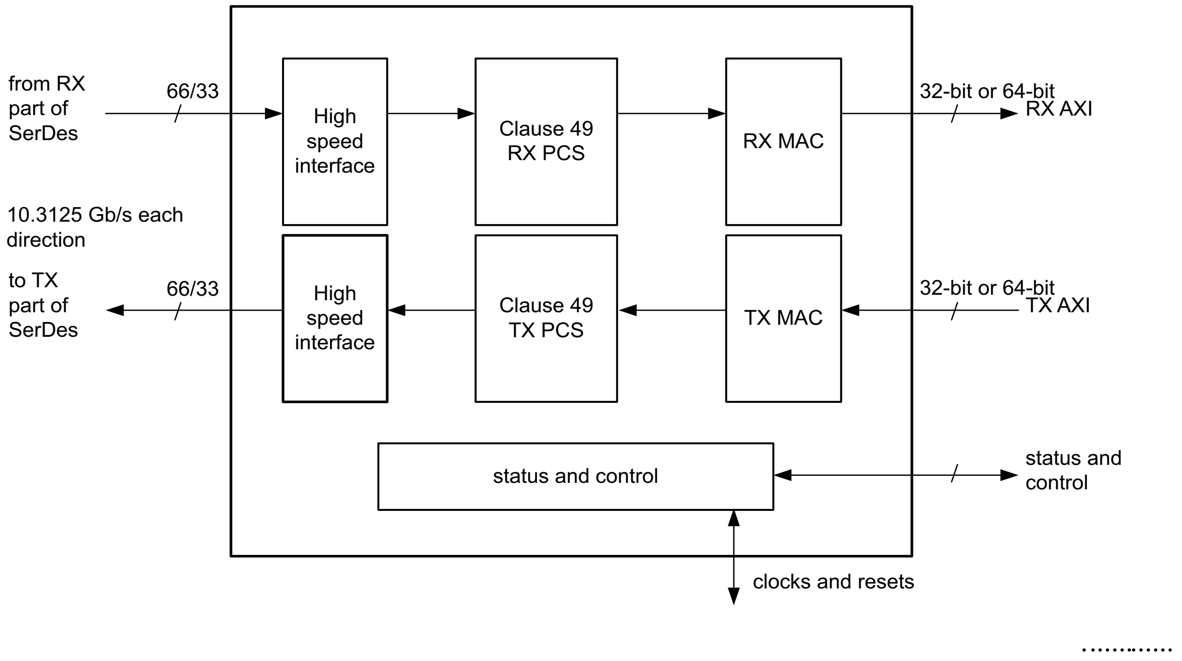


Figure 2-1: **10 Gb/s Core Block Diagram**

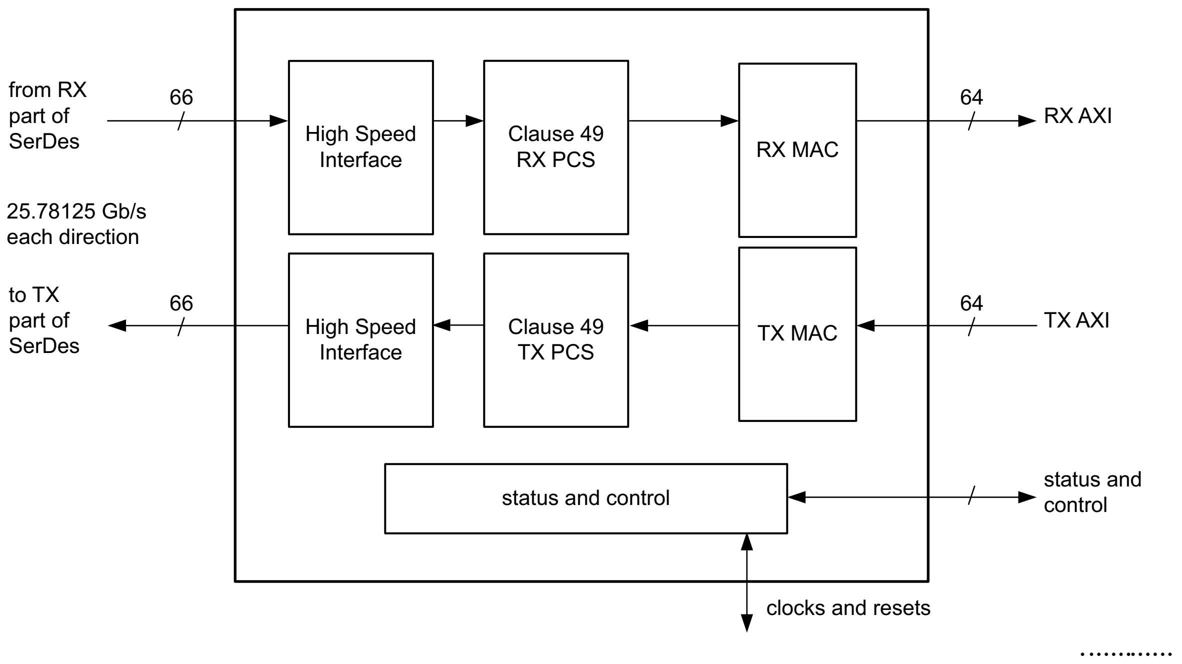


Figure 2-2: **25 Gb/s Core Block Diagram**

A PCS-only variant of the core is also available. The block diagram is shown in Figure 2-3.

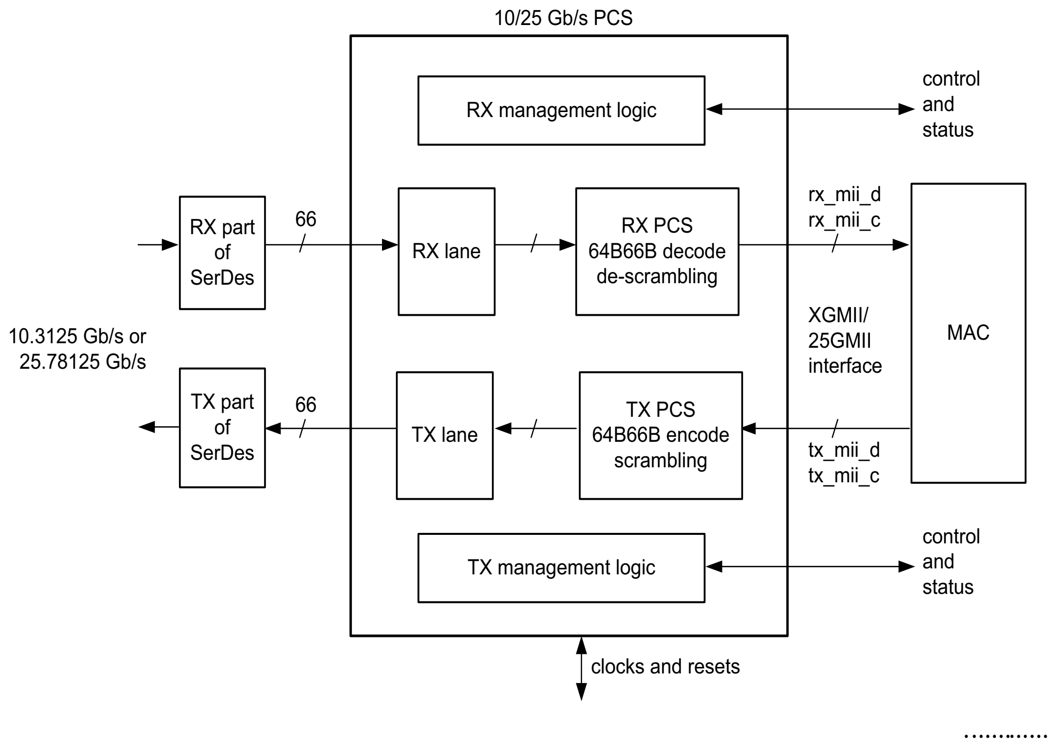


Figure 2-3: **Block Diagram of PCS-Only Core Variant**

Additionally, you can optionally generate a 64-bit standalone version of the MAC for 10 Gb/s operation. The block diagram is shown in Figure 2-4.

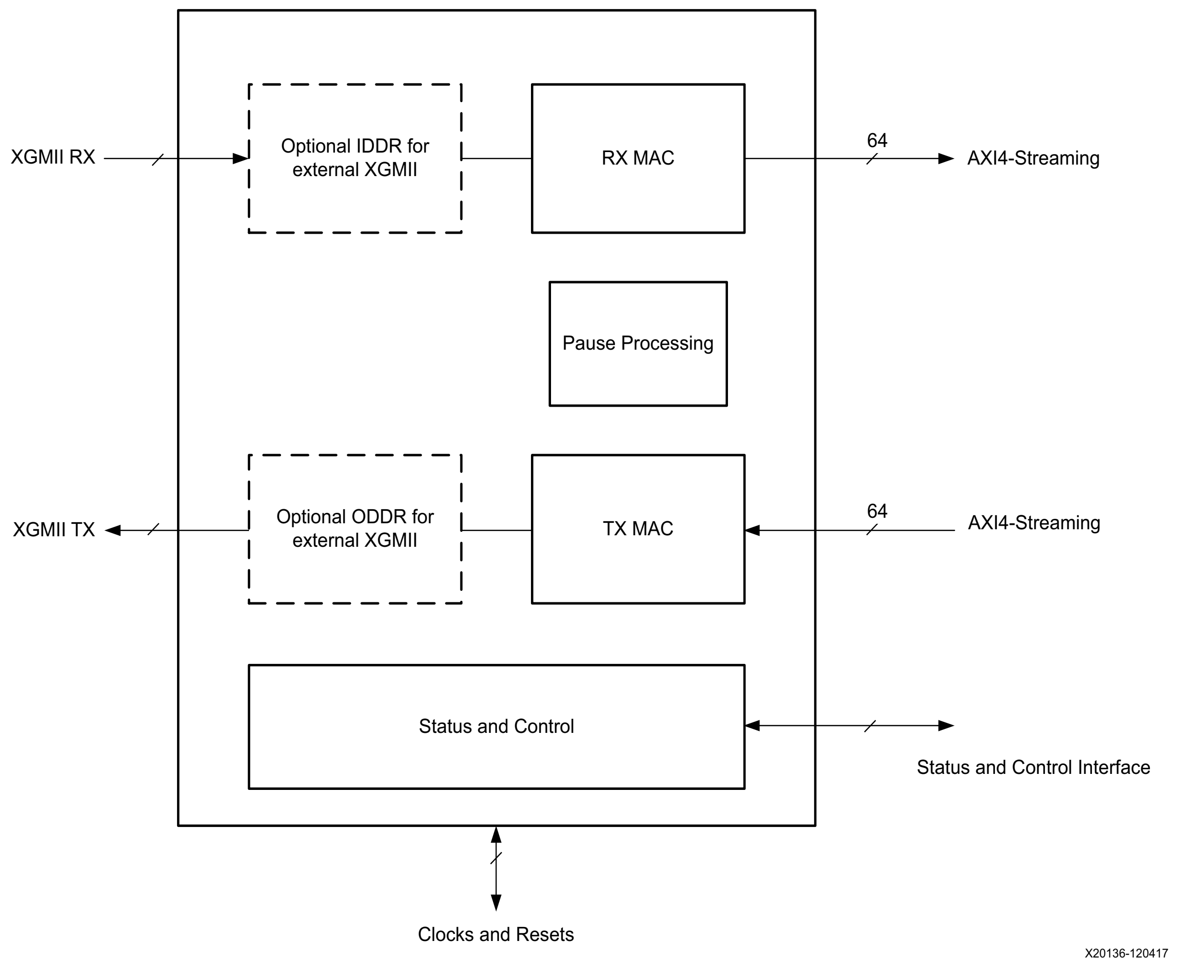


Figure 2-4: **64-bit Standalone Version of the MAC for 10 Gb/s Operation**

Standards

The 10G/25G Ethernet core is designed to the standard specified in the *25G and 50G Ethernet Consortium* [Ref 1] and the *IEEE Std 802.3* [Ref 2] including *IEEE 802.3by* [Ref 3]

Performance and Resource Utilization

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

Latency

Table 2-1 provides the measured low latency information for the 10G/25G IP core.

Table 2-1: **Latency**

Core	Latency (ns)	User Bus Width (bits)	Core Clock Frequency (MHz)
10GE MAC + PCS	115.2	64	156.25
25GE MAC + PCS	46.08	64	390.625
10G PCS	177.32	64	156.25
25G PCS	71.03	64	390.625
10GE MAC + PCS	36.8	32	312.5

Port Descriptions – MAC+PCS Variant

The following tables list the ports for the 10G/25G Ethernet Subsystem with integrated MAC and PCS. These signals are usually found at the wrapper.v hierarchy.

These ports are applicable for both the 64-bit integrated MAC+PCS for 25 Gb/s and 10 Gb/s line rates and the low-latency 32-bit integrated MAC + PCS for the 10 Gb/s line rate. When the AXI register interface is included, some of these ports are accessed via the registers instead of the broadside bus.

Transceiver Interface

Table 2-2 shows the transceiver I/O ports for the 10G/25G Ethernet Subsystem. Refer to Clocking in Chapter 3 for details regarding each clock domain.

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

Name	Direction	Description	Clock Domain
gt_tx_reset	Input	Reset for the gigabit transceiver (GT) TX.	Async
gt_rx_reset	Input	GT RX reset.	Async
ctl_gt_reset_all	Input	Active-High asynchronous reset for the transceiver startup Finite State Machine (FSM). Note that this signal also initiates the reset sequence for the entire 10G/25G Ethernet Subsystem.	Async
refclk_n0	Input	Differential reference clock input for the SerDes, negative phase.	Refer to Clocking.
refclk_p0	Input	Differential reference clock input for the SerDes, positive phase.	Refer to Clocking.
rx_serdes_data_n0	Input	Serial data from the line; negative phase of the differential signal	Refer to Clocking.
rx_serdes_data_p0	Input	Serial data from the line; positive phase of the differential signal	Refer to Clocking.
tx_serdes_data_n0	Output	Serial data to the line; negative phase of the differential signal.	Refer to Clocking.
tx_serdes_data_p0	Output	Serial data to the line; positive phase of the differential signal.	Refer to Clocking.
tx_serdes_clkout	Output	When present, same as tx_clk_out.	Refer to Clocking.

AXI4-Stream Interface

The 10/25G High Speed Ethernet Subsystem IP core provides an option of 32-bit and 64-bit AXI4-Stream interface for systems operating at 10G and 64-bit only for systems operating at 25G. For 10G systems, 32 and 64-bit interfaces are provided. For 25G systems, only a 64-bit interface is provided.

AXI4-Stream Clocks and Resets

Table 2-3: **AXI4-Stream Interface–Clock/Reset Signals**

Name	Direction	Description	Clock Domain
rx_clk_out	Output	rx_serdes_clk. Clocks RX interface between GT and the core. When in low latency buffer bypass mode this clock also clocks the AXI4-Stream RX interface.	Refer to Clocking.
tx_clk_out	Output	Clocks TX AXI4-Stream Interface and full TX datapath.	Refer to Clocking.
rx_reset	Input	Reset for the RX circuits. This signal is active-High (1 = reset) and must be held High until clk is stable.The core handles synchronizing the rx_reset input to the appropriate clock domains within the core.	Async
tx_reset	Input	Reset for the TX circuits. This signal is active-High (1 = reset) and must be held High until clk is stable. The core handles synchronizing the tx_reset input to the appropriate clock domains within the core.	Async
rx_core_clk	Input	The rx_core_clk signal is used to clock the receive AXI4-Stream interface. It is an input when the FIFO is included and is not an input port when in low latency mode with FIFO not included; instead it is driven internally by rx_clk_out.	rx_core_clk

Transmit AXI4-Stream Interface

Table 2-4 shows the AXI4-Stream transmit interface signals.

Table 2-4: **AXI4-Stream Transmit Interface Signals**

Signal	Direction	Description	Clock Domain
tx_axis_tdata[63 or 31:0]	In	AXI4-Stream data. 32-bit and 64-bit interfaces are available. Bus width depends on the selection of 64-bit or 32-bit interfaces.	tx_clk_out
tx_axis_tkeep[7:0 or 3:0]	In	AXI4-Stream Data Control. Bus width depends on selection of 64-bit or 32-bit interfaces.	tx_clk_out
tx_axis_tvalid	In	AXI4-Stream Data Valid input	tx_clk_out
tx_axis_tuser	In	AXI4-Stream User Sideband interface. Equivalent to the tx_errin signal. 1 indicates a bad packet 0 indicates a good packet	tx_clk_out
tx_axis_tlast	In	AXI4-Stream signal indicating End of Ethernet Packet.	tx_clk_out
tx_parityin[7:0]	In	AXI4-Stream user-generated parity. Follows the same data lane mapping as tx_axis_tkeep.	tx_clk_out
tx_axis_tready	Out	AXI4-Stream acknowledge signal to indicate to start the Data transfer.	tx_clk_out

Data Lane Mapping

For transmit data tx_axis_tdata, the port is logically divided into lane 0 to lane 3 for the 32-bit interface (See Table 2-5) or lane 0 to lane 7 for the 64-bit interface (See Table 2-6) with the corresponding bit of the tx_axis_tkeep word signifying valid data on tx_axis_tdata.

Table 2-5: **tx_axis_tdata Lanes – 32 Bits**

Lane/ tx_axis_tkeep	tx_axis_tdata[31:0]
0	7:0
1	15:8
2	23:16
3	31:24

Table 2-6: **tx_axis_tdata Lanes – 64 Bits**

Lane/ tx_axis_tkeep	tx_axis_tdata[63:0]
0	7:0
1	15:8
2	23:16
3	31:24
4	39:32
5	47:40
6	55:48
7	63:56

Normal Transmission

The timing of a normal frame transfer is shown in Figure 2-5 and Figure 2-6 for the 32-bit and 64-bit variants respectively. When the client wants to transmit a frame, it asserts the tx_axis_tvalid and places the data and control in tx_axis_tdata and tx_axis_tkeep in the same clock cycle. When this data is accepted by the core, indicated by tx_axis_tready being asserted, the client must provide the next cycle of data. If tx_axis_tready is not asserted by the core, the client must hold the current valid data value until it is. The end of packet is indicated to the core by tx_axis_tlast asserted for one cycle. The bits of tx_axis_tkeep are set appropriately to indicate the number of valid bytes in the final data transfer. tx_axis_tuser is also asserted to indicate a bad packet.

After tx_axis_tlast is deasserted, any data and control is deemed invalid until tx_axis_tvalid is next asserted.

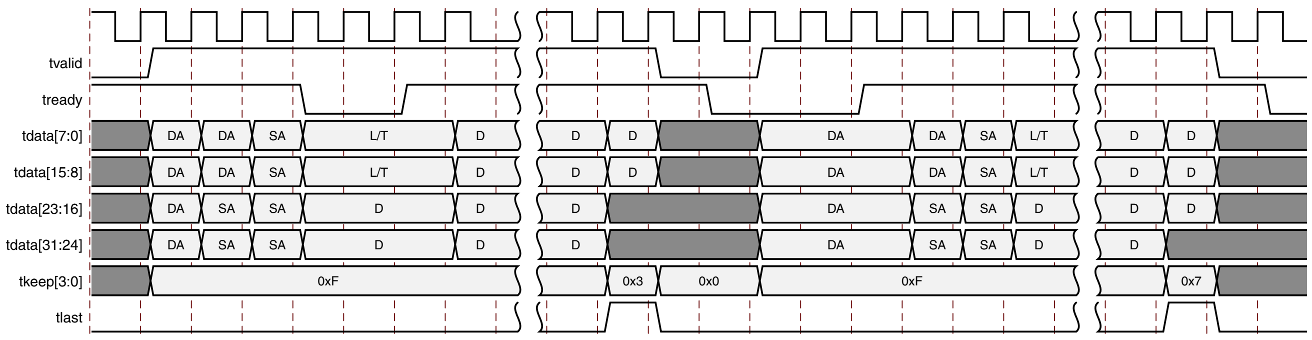


Figure 2-5: **Normal Frame Transfer – 32 Bit**

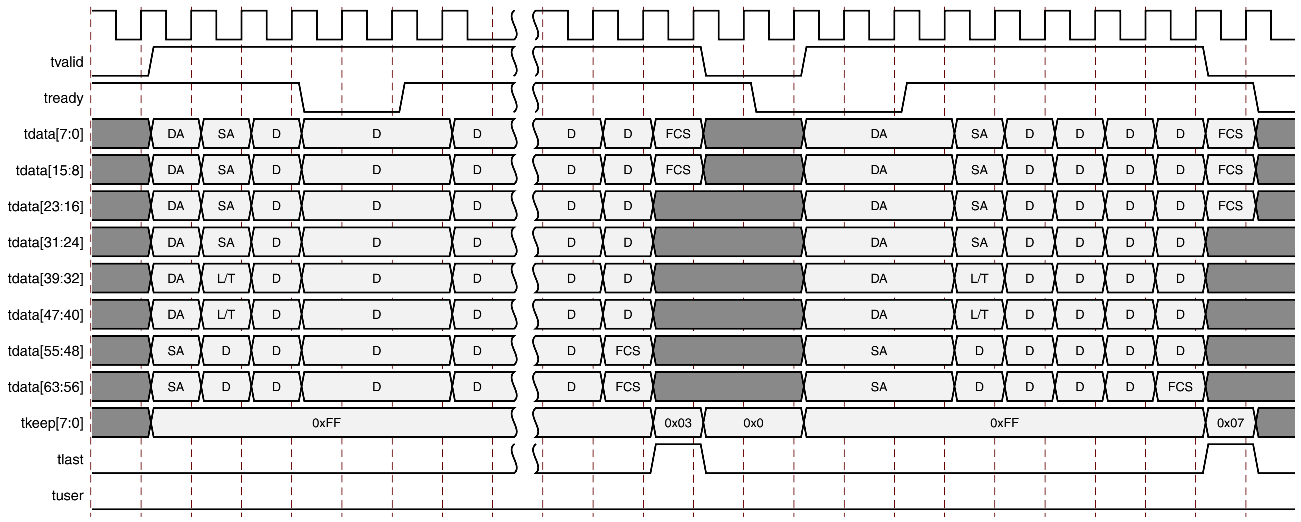


Figure 2-6: **Normal Frame Transfer – 64 Bit**

Back-to-Back Continuous Transfers

Continuous data transfer on the transmit AXI4-Stream interface is possible, as the signal `tx_axis_tvalid` can remain continuously High, with packet boundaries defined solely by `tx_axis_tlast` asserted for the end of the Ethernet packet. However, the core can deassert the `tx_axis_tready` acknowledgment signal to throttle the client data rate as required. See Figure 2-7 and Figure 2-8. The client data logic can update the AXI4-Stream interface with valid data while the core has deasserted the `tx_axis_tready` acknowledgment signal. However, after `valid` is asserted and new data has been placed on the AXI4-Stream, it should remain there until the core has asserted `tx_axis_tready`.

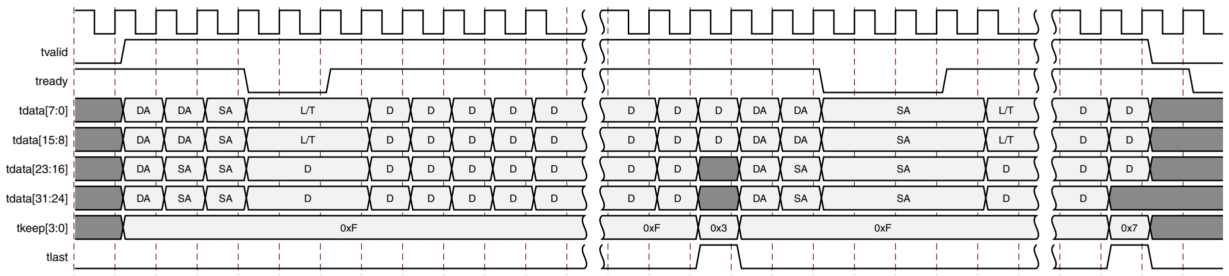


Figure 2-7: **Back-to-Back Continuous Transfer on Transmit Client Interface—32-bit**

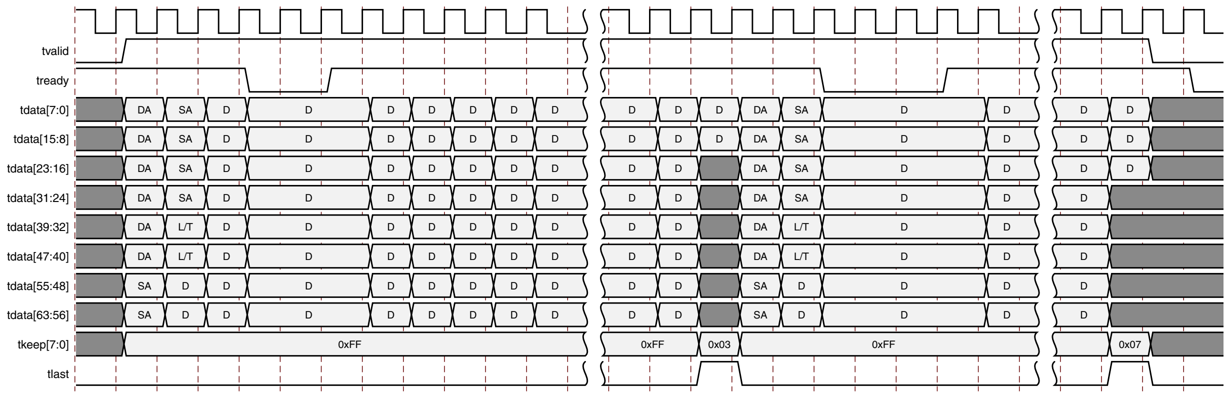


Figure 2-8: **Back-to-Back Continuous Transfer on Transmit Client Interface—64-bit**

Aborting a Transmission

The aborted transfer of a packet on the client interface is called an underrun. This can happen if a FIFO in the AXI Transmit client interface empties before a frame is completed.

This is indicated to the core in one of two ways:

- An explicit error in which a frame transfer is aborted by asserting `tx_axis_tuser` High while `tx_axis_tlast` is High (See Figure 2-12).
- An implicit underrun, in which a frame transfer is aborted by deasserting `tx_axis_tvalid` without asserting `tx_axis_tlast`.

When either of the two scenarios occurs during a frame transmission, the core inserts error codes into the data stream to flag the current frame as an errored frame. It remains the responsibility of the client to re-queue the aborted frame for transmission, if necessary.

Receive AXI4-Stream Interface

Table 2-7: **AXI4-Stream Receive Interface Signals**

Signal	Direction	Description	Clock Domain
rx_axis_tdata[63 or 31:0]	Out	AXI4-Stream Data to upper layer. Bus width depends on 64-bit or 32-bit selection.	rx_core_clk
rx_axis_tkeep[7 or 3:0]	Out	AXI4-Stream Data Control to upper layer. Bus width depends on 64-bit or 32-bit selection.	rx_core_clk
rx_axis_tvalid	Out	AXI4-Stream Data Valid	rx_core_clk
rx_axis_tuser	Out	AXI4-Stream User Sideband interface. 1 indicates a bad packet has been received. 0 indicates a good packet has been received.	rx_core_clk
rx_axis_tlast	Out	AXI4-Stream signal indicating an end of packet.	rx_core_clk
rx_parityout[7:0]	Out	AXI4-Stream core-generated parity. Follows the same data lane mapping as rx_axis_tkeep.	rx_core_clk

Data Lane Mapping

For receive data `rx_axis_tdata`, the port is logically divided into lane 0 to lane 3 for the 32-bit interface (See Table 2-8) or lane 0 to lane 7 for the 64-bit interface (See Table 2-9) with the corresponding bit of the `rx_axis_tkeep` word signifying valid data on `rx_axis_tdata`.

Table 2-8: **rx_axis_tdata Lanes - 32 bits**

Lane/rx_axis_tkeep	rx_axis_tdata[31:0] bits
0	7:0
1	15:8
2	23:16
3	31:24

Table 2-9: **rx_axis_tkeep Lanes - 64 bits**

Lane/ rx_axis_tkeep	rx_axis_tdata Bits
0	7:0
1	15:8
2	23:16
3	31:24
4	39:32
5	47:40

Table 2-9: **rx_axis_tkeep Lanes - 64 bits**

Lane/ rx_axis_tkeep	rx_axis_tdata Bits
6	55:48
7	63:56

Normal Frame Reception

The timing of a normal inbound frame transfer is represented in Figure 2-10. The client must be prepared to accept data at any time; there is no buffering within the core to allow for latency in the receive client. When frame reception begins, data is transferred on consecutive clock cycles to the receive client.

During frame reception, `rx_axis_tvalid` is asserted to indicate that valid frame data is being transferred to the client on `rx_axis_tdata`. All bytes are always valid throughout the frame, as indicated by all `rx_axis_tkeep` bits being set to 1, except during the final transfer of the frame when `rx_axis_tlast` is asserted. During this final transfer of data for a frame, `rx_axis_tkeep` bits indicate the final valid bytes of the frame using the mapping from above. The valid bytes of the final transfer always lead out from `rx_axis_tdata[7:0]` (`rx_axis_tkeep[0]`) because Ethernet frame data is continuous and is received least significant byte first.

The `rx_axis_tlast` is asserted and `rx_axis_tuser` is deasserted, along with the final bytes of the transfer, only after all frame checks are completed. This is after the frame check sequence (FCS) field has been received. The core keeps the `rx_axis_tuser` signal deasserted to indicate that the frame was successfully received and that the frame should be analyzed by the client. This is also the end of packet signaled by `rx_axis_tlast` asserted for one cycle.

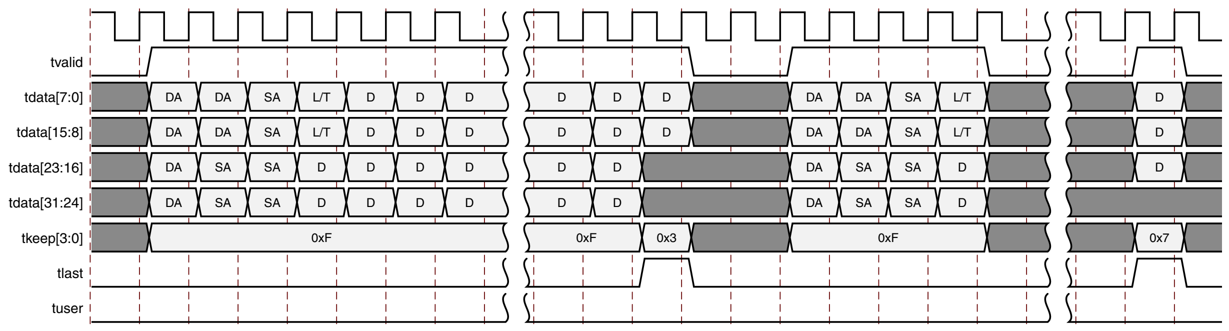


Figure 2-9: **Normal Frame Reception – 32 Bits**

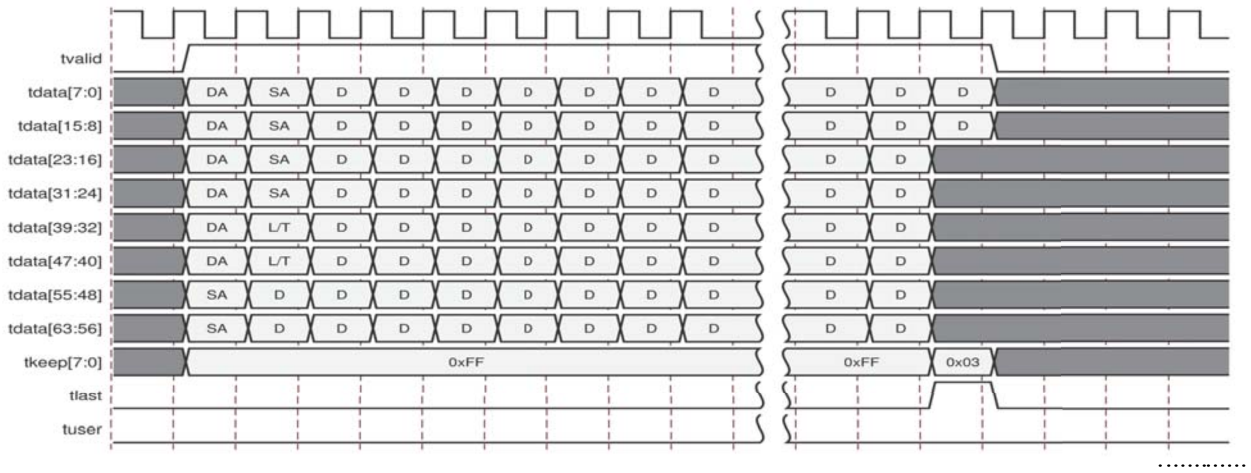


Figure 2-10: **Normal Frame Reception – 64 Bits**

Frame Reception with Errors

The case of an unsuccessful frame reception (for example, a runt frame or a frame with an incorrect FCS) is shown in Figure 2-12. In this case, the bad frame is received and the signal `rx_axis_tuser` is asserted to the client at the end of the frame. It is then the responsibility of the client to drop the data already transferred for this frame.

The following conditions cause the assertion of `rx_axis_tlast` along with `rx_axis_tuser` = 1 signifying a bad_frame:

- FCS errors occur.
- Packets are shorter than 64 bytes (undersize or fragment frames).
- Frames of length greater than the maximum transmission unit (MTU) Size programmed are received.
- Any control frame that is received is not exactly the minimum frame length.
- The XGMII data stream contains error codes.

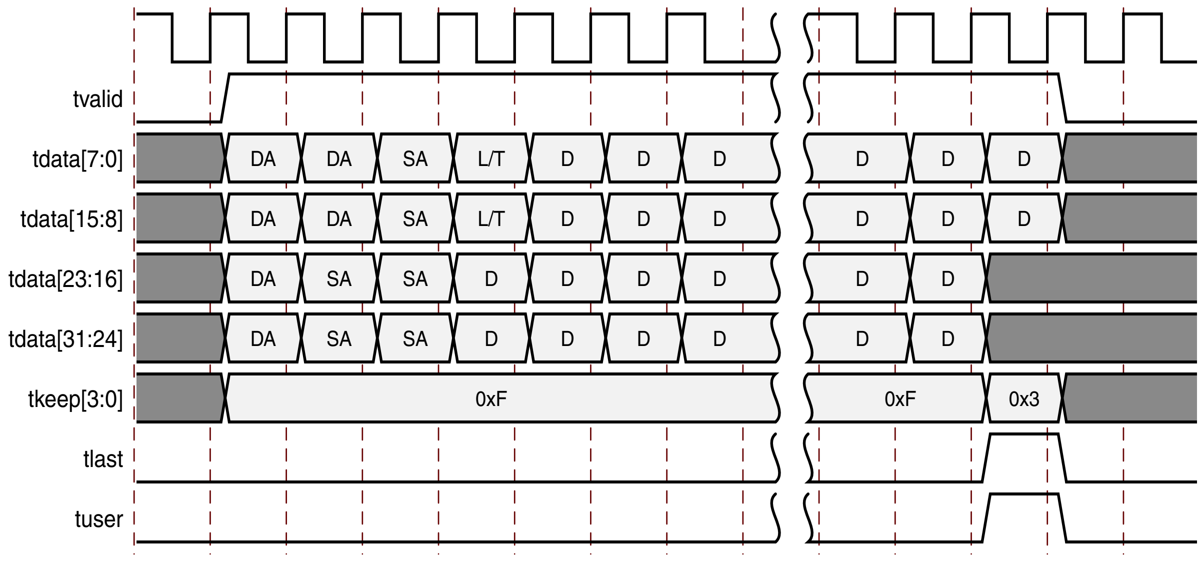


Figure 2-11: **Frame Reception with Errors – 32 Bits**

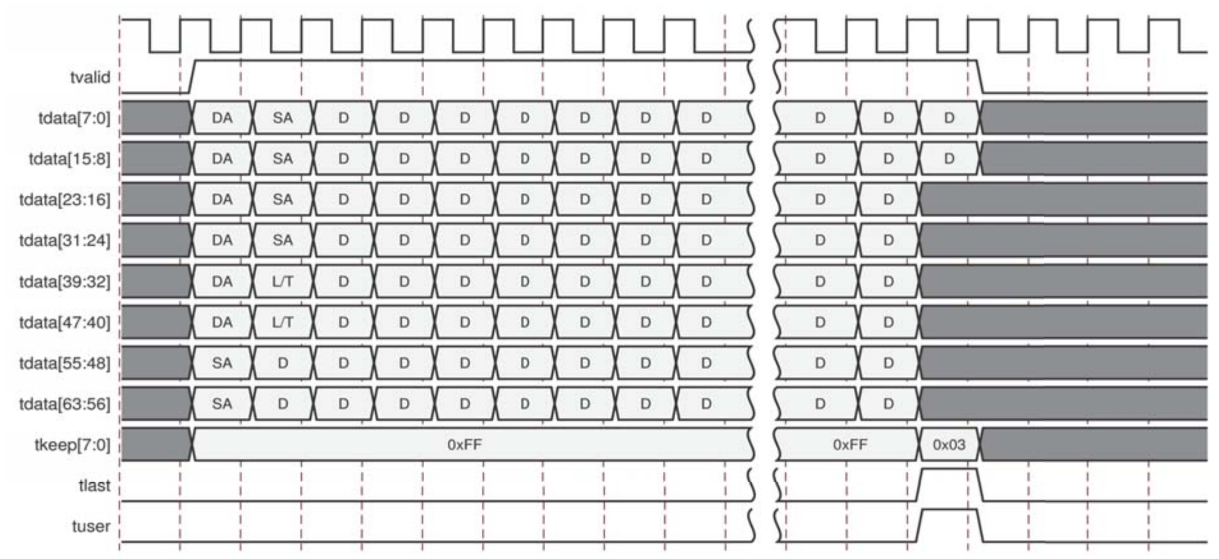


Figure 2-12: **Frame Reception with Errors – 64 Bits**

AXI4-Stream Control and Status Ports

Table 2-10: **AXI4-Stream Interface–TX Path Control/Status Signals**

Name	**Direction**	**Description**	**Clock Domain**
ctl_tx_custom_preamble_enable	Input	When asserted, this signals enables the use of tx_preamblein as a custom preamble instead of inserting a standard preamble.	tx_clk_out
tx_preamblein [55:0]	Input	This is the custom preamble which is a separate input port rather than being in-line with the data. It should be valid during the start of packet.	tx_clk_out
ctl_tx_ipg_value[3:0]	Input	This signal can be optionally present. The ctl_tx_ipg_value defines the target average minimum Inter Packet Gap (IPG, in bytes) inserted between AXI4-Stream packets. Valid values are 8 to 12. The ctl_tx_ipg_value can be programmed to a value in the 0 to 7 range, but in that case, it is interpreted as 8 (the minimum valid value).	tx_clk_out
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 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 synchronized 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 core stops transmitting any more packets.	tx_clk_out
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 synchronized and is ready to accept data from the link partner.	tx_clk_out
ctl_tx_send_lfi	Input	Transmit Local Fault Indication (LFI) code word. Takes precedence over Remote Fault Indication (RFI).	tx_clk_out
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 RFI code words.	tx_clk_out
ctl_tx_fcs_ins_enable	Input	Enable FCS insertion by the TX core. If this bit is set to 0, the core does not add FCS to packet. If this bit is set to 1, the core calculates and adds the FCS to the packet. This input cannot be changed dynamically between packets.	tx_clk_out

Table 2-10: **AXI4-Stream Interface–TX Path Control/Status Signals *(Cont'd)***

Name	Direction	Description	Clock Domain
ctl_tx_ignore_fcs	Input	Enable FCS error checking at the AXI4-Stream interface by the TX core. This input only has effect when ctl_tx_fcs_ins_enable is Low. If this input is Low and a packet with bad FCS is being transmitted, it is not binned as good. If this input is High, a packet with bad FCS is binned as good. The error is flagged on the signals stat_tx_bad_fcs and stomped_fcs, and the packet is transmitted as it was received. ***Note:*** Statistics are reported as if there was no FCS error.	tx_clk_out
ctl_tx_parity_err_response	Input	Parity error response by the TX Core. If this bit is set to 0, the core does not take any action if any parity errors are detected. If this bit is set to 1, the core stomps the outgoing FCS (i.e., bit-wise inverse) and asserts stat_tx_bad_fcs. The stat_tx_bad_parity output is asserted when parity errors are detected regardless of the ctl_tx_parity_err_response configuration.	tx_clk_out

Table 2-11: **AXI4-Stream Interface–RX Path Control/Status Signals**

Name	Direction	Description	Clock Domain
rx_preambleout [55:0]	Output	This is the preamble, and now a separate output instead of inline with data as was done with previous releases.	rx_core_clk
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 AXI4-Stream interface is idle.	rx_clk_out
ctl_rx_check_preamble	Input	When asserted, this input causes the MAC to check the preamble of the received frame.	rx_clk_out
ctl_rx_check_sfd	Input	When asserted, this input causes the MAC to check the Start of Frame Delimiter of the received frame.	rx_clk_out
ctl_rx_force_resync	Input	RX force resynchronization input. This signal is used to force the RX path to reset and re-synchronize. A value of 1 forces the reset operation. A value of 0 allows normal operation. Note that this input should normally be Low and should only be pulsed (1 cycle minimum pulse).	rx_clk_out

Table 2-11: **AXI4-Stream Interface–RX Path Control/Status Signals *(Cont'd)***

Name	Direction	Description	Clock Domain
ctl_rx_delete_fcs	Input	Enable FCS removal by the RX core. If this bit is set to 0, the core does not remove the FCS of the incoming packet. If this bit is set to 1, the core deletes the FCS to the received packet. Note that FCS is not deleted for packets that are less than or equal to 8 bytes long. This input should only be changed while the corresponding reset input is asserted.	rx_clk_out
ctl_rx_ignore_fcs	Input	Enable FCS error checking at the AXI4-Stream interface by the RX core. If this bit is set to 0, a packet received with an FCS error is sent with the rx_axis_tuser pin asserted during the last transfer (rx_axis_tlast sampled 1). If this bit is set to 1, the core does not flag an FCS error at the AXI4-Stream interface. ***Note:*** The statistics are reported as if the packet is good. The signal stat_rx_bad_fcs, however, reports the error.	rx_clk_out
ctl_rx_max_packet_len[14:0]	Input	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_axis_tuser signal is asserted along with the rx_axis_tlast signal. Packets less than 4 bytes are dropped. ctl_rx_max_packet_len[14] is reserved and must be set to 0.	rx_clk_out
ctl_rx_min_packet_len[7:0]	Input	Any packet shorter than this value is considered to be undersized. If a packet has a size less than this value, the rx_axis_tuser signal is asserted during the rx_axis_tlast asserted cycle. Packets less than 4 bytes are dropped.	rx_clk_out
stat_rx_framing_err[1:0]	Output	The RX sync header bits framing error is a bus that indicates how many sync header errors were received. The value of the bus is only valid when stat_rx_framing_err_valid is a 1. The values can be updated at any time and are intended to be used as increment values for sync header error counters.	rx_clk_out
stat_rx_framing_err_valid	Output	Valid indicator for stat_rx_framing_err. When sampled as a 1, the value on stat_rx_framing_err is valid.	rx_clk_out
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.	rx_clk_out
stat_rx_status	Output	Indicates current status of the link.	rx_clk_out

Table 2-11: **AXI4-Stream Interface–RX Path Control/Status Signals *(Cont'd)***

Name	Direction	Description	Clock Domain
stat_rx_block_lock	Output	Block lock status. A value of 1 indicates that block lock is achieved as defined in Clause 49.2.14 and MDIO register 3.32.0 This output is level sensitive.	rx_clk_out
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, remote fault condition does not exist. This output is level sensitive.	rx_clk_out
stat_rx_bad_fcs[1:0]	Output	Bad FCS indicator. The value on this bus indicates packets received with a bad FCS, but not a stomped FCS during a cycle. 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. Note that pulses can occur in back-to-back cycles.	rx_clk_out
stat_rx_stomped_fcs[1:0]	Output	Stomped FCS indicator. The value on this bus indicates the 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. Note that pulses can occur in back to back cycles.	rx_clk_out
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. Note that pulses can occur in back to back cycles.	rx_clk_out
stat_rx_internal_local_fault	Output	High when an internal local fault is generated due to any one of the following: test pattern generation or high bit error rate. Note that this signal remains High as long as the fault condition persists.	rx_clk_out
stat_rx_received_local_fault	Output	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. Remains High as long as the fault condition persists.	rx_clk_out

Table 2-11: **AXI4-Stream Interface–RX Path Control/Status Signals *(Cont'd)***

Name	**Direction**	**Description**	**Clock Domain**
stat_rx_hi_ber	Output	High Bit Error Rate (BER) indicator. When set to 1, the BER is too high as defined by IEEE Std. 802.3. Corresponds to MDIO register bit 3.32.1 as defined in Clause 49.2.14. This output is level sensitive.	rx_clk_out
ctl_rx_custom_preamble_enable	Input	When asserted, this signal causes the side band of a packet presented on the AXI4-Stream to be the preamble as it appears on the line.	rx_clk_out

Miscellaneous Status/Control Signals

Table 2-12 shows the miscellaneous status and control I/O signals.

Table 2-12: **Miscellaneous Status/Control Ports**

Name	**Direction**	**Description**	**Clock Domain**
dclk	Input	Dynamic Reconfiguration Port (DRP) clock input. The required frequency is set by providing the value in the **GT DRP Clock** field in the Vivado® Integrated Design Environment (IDE) GT Selection and Configuration tab. This must be a free running input clock.	Refer to Clocking.
stat_rx_valid_ctrl_code	Output	Indicates that a PCS block with a valid control code was received.	rx_clk_out
ctl_local_loopback	Input	Loopback enable. A value of 1 enables loopback as defined in Clause 49. Corresponds to management data input/output (MDIO) register bit 3.0.14 as defined in Clause 45. This input should only be changed while the corresponding reset input is asserted. ***Note:*** This option will be available only when the **Include AXI4-Lite** enable option is available in the Vivado Design Suite.	Async
stat_rx_got_signal_os	Output	Signal OS indication. If this bit is sampled as a 1, it indicates that a Signal OS word was received. Note that Signal OS should not be received in an Ethernet network.	rx_clk_out

Table 2-12: **Miscellaneous Status/Control Ports *(Cont'd)***

Name	Direction	Description	Clock Domain
ctl_rx_process_lfi	Input	When this input is set to 1, the RX core expects and processes LF control codes coming in from the transceiver. When set to 0, the RX core ignores LF control codes coming in from the transceiver.	rx_clk_out
ctl_rx_test_pattern	Input	Test pattern checking enable for the RX core. A value of 1 enables test mode as defined in Clause 49. Corresponds to MDIO register bit 3.42.2 as defined in Clause 45. Checks for scrambled idle pattern.	rx_clk_out
ctl_tx_test_pattern	Input	Test pattern generation enable for the TX core. A value of 1 enables test mode as defined in Clause 49. Corresponds to MDIO register bit 3.42.3 as defined in Clause 45. Generates a scrambled idle pattern.	tx_clk_out
stat_rx_test_pattern_mismatch	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 as defined in Clause 45. This output is pulsed for one clock cycle.	rx_clk_out
ctl_rx_data_pattern_select	Input	Corresponds to MDIO register bit 3.42.0 as defined in Clause 45.	rx_clk_out
ctl_rx_test_pattern_enable	Input	Test pattern enable for the RX core. A value of 1 enables test mode. Corresponds to MDIO register bit 3.42.2 as defined in Clause 45. Takes second precedence.	rx_clk_out
ctl_tx_data_pattern_select	Input	Corresponds to MDIO register bit 3.42.0 as defined in Clause 45.	tx_clk_out
ctl_tx_test_pattern_enable	Input	Test pattern generation enable for the TX core. A value of 1 enables test mode. Corresponds to MDIO register bit 3.42.3 as defined in Clause 45. Takes second precedence.	tx_clk_out
ctl_tx_test_pattern_seed_a[57:0]	Input	Corresponds to MDIO registers 3.34 through to 3.37 as defined in Clause 45.	tx_clk_out

Table 2-12: **Miscellaneous Status/Control Ports *(Cont'd)***

Name	Direction	Description	Clock Domain
ctl_tx_test_pattern_seed_b[57:0]	Input	Corresponds to MDIO registers 3.38 through to 3.41 as defined in Clause 45.	tx_clk_out
ctl_tx_test_pattern_select	Input	Corresponds to MDIO register bit 3.42.1 as defined in Clause 45.	tx_clk_out

Statistics Interface Ports

Table 2-13 and Table 2-14 show the Statistics interface I/O ports.

Table 2-13: **Statistics Interface - RX Path**

Name	Direction	Description	Clock Domain
stat_rx_total_bytes[3:0]	Output	Increment for the total number of bytes received.	rx_clk_out
stat_rx_total_packets[1:0]	Output	Increment for the total number of packets received.	rx_clk_out
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.	rx_clk_out
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.	rx_clk_out
stat_rx_packet_bad_fcs	Output	Increment for packets between 64 and ctl_rx_max_packet_len bytes that have Frame Check Sequence (FCS) errors.	rx_clk_out
stat_rx_packet_64_bytes	Output	Increment for good and bad packets received that contain 64 bytes.	rx_clk_out
stat_rx_packet_65_127_bytes	Output	Increment for good and bad packets received that contain 65 to 127 bytes.	rx_clk_out
stat_rx_packet_128_255_bytes	Output	Increment for good and bad packets received that contain 128 to 255 bytes.	rx_clk_out
stat_rx_packet_256_511_bytes	Output	Increment for good and bad packets received that contain 256 to 511 bytes.	rx_clk_out
stat_rx_packet_512_1023_bytes	Output	Increment for good and bad packets received that contain 512 to 1,023 bytes.	rx_clk_out
stat_rx_packet_1024_1518_bytes	Output	Increment for good and bad packets received that contain 1,024 to 1,518 bytes.	rx_clk_out
stat_rx_packet_1519_1522_bytes	Output	Increment for good and bad packets received that contain 1519 to 1522 bytes.	rx_clk_out

Table 2-13: **Statistics Interface - RX Path *(Cont'd)***

Name	Direction	Description	Clock Domain
stat_rx_packet_1523_1548_bytes	Output	Increment for good and bad packets received that contain 1,523 to 1,548 bytes.	rx_clk_out
stat_rx_packet_1549_2047_bytes	Output	Increment for good and bad packets received that contain 1,549 to 2,047 bytes.	rx_clk_out
stat_rx_packet_2048_4095_bytes	Output	Increment for good and bad packets received that contain 2,048 to 4,095 bytes.	rx_clk_out
stat_rx_packet_4096_8191_bytes	Output	Increment for good and bad packets received that contain 4,096 to 8,191 bytes.	rx_clk_out
stat_rx_packet_8192_9215_bytes	Output	Increment for good and bad packets received that contain 8,192 to 9,215 bytes.	rx_clk_out
stat_rx_packet_small	Output	Increment for all packets that are less than 64 bytes long. Packets that are less than four bytes are dropped.	rx_clk_out
stat_rx_packet_large	Output	Increment for all packets that are more than 9,215 bytes long.	rx_clk_out
stat_rx_unicast	Output	Increment for good unicast packets.	rx_clk_out
stat_rx_multicast	Output	Increment for good multicast packets.	rx_clk_out
stat_rx_broadcast	Output	Increment for good broadcast packets.	rx_clk_out
stat_rx_oversize	Output	Increment for packets longer than ctl_rx_max_packet_len with good FCS.	rx_clk_out
stat_rx_toolong	Output	Increment for packets longer than ctl_rx_max_packet_len with good and bad FCS.	rx_clk_out
stat_rx_undersize	Output	Increment for packets shorter than ctl_rx_min_packet_len with good FCS.	rx_clk_out
stat_rx_fragment	Output	Increment for packets shorter than ctl_rx_min_packet_len with bad FCS.	rx_clk_out
stat_rx_vlan	Output	Increment for good 802.1Q tagged VLAN packets.	rx_clk_out
stat_rx_inrangeerr	Output	Increment for packets with Length field error but with good FCS.	rx_clk_out
stat_rx_jabber	Output	Increment for packets longer than ctl_rx_max_packet_len with bad FCS.	rx_clk_out
stat_rx_pause	Output	Increment for 802.3x MAC Pause packet with good FCS	rx_clk_out
stat_rx_user_pause	Output	Increment for priority based pause packets with good FCS.	rx_clk_out

Table 2-13: **Statistics Interface - RX Path *(Cont'd)***

Name	Direction	Description	Clock Domain
stat_rx_bad_code	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 IEEE Std. 802.3. This output can be used to generate MDIO register as defined in Clause 45.	rx_clk_out
stat_rx_bad_sfd	Output	Increment bad SFD. This signal indicates if the Ethernet packet received was preceded by a valid SFD. A value of 1 indicates that an invalid SFD was received. ***Note:*** When an invalid SFD is detected, the stat_rx_bad_sfd signal is asserted regardless of the setting of the ctl_rx_check_sfd signal.	rx_clk_out
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. ***Note:*** When an invalid preamble is detected, the stat_rx_bad_preamble signal is asserted regardless of the setting of the ctl_rx_check_preamble signal.	rx_clk_out

Table 2-14: **Statistics Interface - TX Path**

Name	Direction	Description	Clock Domain
stat_tx_total_bytes[3:0]	Output	Increment for the total number of bytes transmitted. The signal width for stat_tx_total_bytes will be [2:0] when the 32-bit AXI4-Stream option is selected.	tx_clk_out
stat_tx_total_packets	Output	Increment for the total number of packets transmitted.	tx_clk_out
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.	tx_clk_out
stat_tx_total_good_packets	Output	Increment for the total number of good packets transmitted.	tx_clk_out
stat_tx_bad_fcs	Output	Increment for packets greater than 64 bytes that have FCS errors.	tx_clk_out
stat_tx_packet_64_bytes	Output	Increment for good and bad packets transmitted that contain 64 bytes.	tx_clk_out
stat_tx_packet_65_127_bytes	Output	Increment for good and bad packets transmitted that contain 65 to 127 bytes.	tx_clk_out

Table 2-14: **Statistics Interface - TX Path *(Cont'd)***

Name	Direction	Description	Clock Domain
stat_tx_packet_128_255_bytes	Output	Increment for good and bad packets transmitted that contain 128 to 255 bytes.	tx_clk_out
stat_tx_packet_256_511_bytes	Output	Increment for good and bad packets transmitted that contain 256 to 511 bytes.	tx_clk_out
stat_tx_packet_512_1023_bytes	Output	Increment for good and bad packets transmitted that contain 512 to 1,023 bytes.	tx_clk_out
stat_tx_packet_1024_1518_bytes	Output	Increment for good and bad packets transmitted that contain 1,024 to 1,518 bytes.	tx_clk_out
stat_tx_packet_1519_1522_bytes	Output	Increment for good and bad packets transmitted that contain 1,519 to 1,522 bytes.	tx_clk_out
stat_tx_packet_1523_1548_bytes	Output	Increment for good and bad packets transmitted that contain 1,523 to 1,548 bytes.	tx_clk_out
stat_tx_packet_1549_2047_bytes	Output	Increment for good and bad packets transmitted that contain 1,549 to 2,047 bytes.	tx_clk_out
stat_tx_packet_2048_4095_bytes	Output	Increment for good and bad packets transmitted that contain 2,048 to 4,095 bytes.	tx_clk_out
stat_tx_packet_4096_8191_bytes	Output	Increment for good and bad packets transmitted that contain 4,096 to 8,191 bytes.	tx_clk_out
stat_tx_packet_8192_9215_bytes	Output	Increment for good and bad packets transmitted that contain 8,192 to 9,215 bytes.	tx_clk_out
stat_tx_packet_small	Output	Increment for all packets that are less than 64 bytes long.	tx_clk_out
stat_tx_packet_large	Output	Increment for all packets that are more than 9,215 bytes long.	tx_clk_out
stat_tx_unicast	Output	Increment for good unicast packets.	tx_clk_out
stat_tx_multicast	Output	Increment for good multicast packets.	tx_clk_out
stat_tx_broadcast	Output	Increment for good broadcast packets.	tx_clk_out
stat_tx_vlan	Output	Increment for good 802.1Q tagged VLAN packets.	tx_clk_out
stat_tx_pause	Output	Increment for 802.3x MAC Pause packet with good FCS.	tx_clk_out
stat_tx_user_pause	Output	Increment for priority based pause packets with good FCS.	tx_clk_out

Table 2-14: **Statistics Interface - TX Path *(Cont'd)***

Name	Direction	Description	Clock Domain
stat_tx_frame_error	Output	Increment for packets with tx_axis_tuser set to indicate an End of Packet (EOP) abort.	tx_clk_out
stat_tx_bad_parity	Output	Increment on any clock cycle where the user-generated parity is calculated as incorrect by the Tx parity checking logic.	tx_clk_out

Pause Interface

Table 2-15 through Table 2-17 show the Pause interface I/O ports.

Table 2-15: **Pause Interface–Control Ports**

Name	Direction	Description	Clock Domain
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. Note that this signal only affects the RX user interface, not the pause processing logic.	rx_clk_out
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.	tx_clk_out

Table 2-16: **Pause Interface–RX Path**

Name	Direction	Description	Clock Domain
ctl_rx_enable_gcp	Input	A value of 1 enables global control packet processing.	rx_clk_out
ctl_rx_check_mcast_gcp	Input	A value of 1 enables global control multicast destination address processing.	rx_clk_out
ctl_rx_check_ucast_gcp	Input	A value of 1 enables global control unicast destination address processing.	rx_clk_out
ctl_rx_pause_da_ucast[47:0]	Input	Unicast destination address for pause processing.	rx_clk_out
ctl_rx_check_sa_gcp	Input	A value of 1 enables global control source address processing.	rx_clk_out
ctl_rx_pause_sa[47:0]	Input	Source address for pause processing.	rx_clk_out
ctl_rx_check_etype_gcp	Input	A value of 1 enables global control ethertype processing.	rx_clk_out
ctl_rx_check_opcode_gcp	Input	A value of 1 enables global control opcode processing.	rx_clk_out

Table 2-16: **Pause Interface–RX Path *(Cont'd)***

Name	Direction	Description	Clock Domain
ctl_rx_opcode_min_gcp[15:0]	Input	Minimum global control opcode value.	rx_clk_out
ctl_rx_opcode_max_gcp[15:0]	Input	Maximum global control opcode value.	rx_clk_out
ctl_rx_etype_gcp[15:0]	Input	Ethertype field for global control processing.	rx_clk_out
ctl_rx_enable_pcp	Input	A value of 1 enables priority control packet processing.	rx_clk_out
ctl_rx_check_mcast_pcp	Input	A value of 1 enables priority control multicast destination address processing.	rx_clk_out
ctl_rx_check_ucast_pcp	Input	A value of 1 enables priority control unicast destination address processing.	rx_clk_out
ctl_rx_pause_da_mcast[47:0]	Input	Multicast destination address for pause processing.	rx_clk_out
ctl_rx_check_sa_pcp	Input	A value of 1 enables priority control source address processing.	rx_clk_out
ctl_rx_check_etype_pcp	Input	A value of 1 enables priority control ethertype processing.	rx_clk_out
ctl_rx_etype_pcp[15:0]	Input	Ethertype field for priority control processing.	rx_clk_out
ctl_rx_check_opcode_pcp	Input	A value of 1 enables priority control opcode processing.	rx_clk_out
ctl_rx_opcode_min_pcp[15:0]	Input	Minimum priority control opcode value.	rx_clk_out
ctl_rx_opcode_max_pcp[15:0]	Input	Maximum priority control opcode value.	rx_clk_out
ctl_rx_enable_gpp	Input	A value of 1 enables global pause packet processing.	rx_clk_out
ctl_rx_check_mcast_gpp	Input	A value of 1 enables global pause multicast destination address processing.	rx_clk_out
ctl_rx_check_ucast_gpp	Input	A value of 1 enables global pause unicast destination address processing.	rx_clk_out
ctl_rx_check_sa_gpp	Input	A value of 1 enables global pause source address processing.	rx_clk_out
ctl_rx_check_etype_gpp	Input	A value of 1 enables global pause ethertype processing.	rx_clk_out
ctl_rx_etype_gpp[15:0]	Input	Ethertype field for global pause processing.	rx_clk_out
ctl_rx_check_opcode_gpp	Input	A value of 1 enables global pause opcode processing.	rx_clk_out
ctl_rx_opcode_gpp[15:0]	Input	Global pause opcode value.	rx_clk_out
ctl_rx_enable_ppp	Input	A value of 1 enables priority pause packet processing.	rx_clk_out

Table 2-16: **Pause Interface–RX Path *(Cont'd)***

Name	Direction	Description	Clock Domain
ctl_rx_check_mcast_ppp	Input	A value of 1 enables priority pause multicast destination address processing.	rx_clk_out
ctl_rx_check_ucast_ppp	Input	A value of 1 enables priority pause unicast destination address processing.	rx_clk_out
ctl_rx_check_sa_ppp	Input	A value of 1 enables priority pause source address processing.	rx_clk_out
ctl_rx_check_etype_ppp	Input	A value of 1 enables priority pause ethertype processing.	rx_clk_out
ctl_rx_etype_ppp[15:0]	Input	Ethertype field for priority pause processing.	rx_clk_out
ctl_rx_check_opcode_ppp	Input	A value of 1 enables priority pause opcode processing.	rx_clk_out
ctl_rx_opcode_ppp[15:0]	Input	Priority pause opcode value.	rx_clk_out
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 keep it at 1 until the pause packet has been processed.	rx_clk_out
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.	rx_clk_out
ctl_rx_check_ack	Input	Wait for acknowledge. If this input is set to 1, the core uses the ctl_rx_pause_ack[8:0] bus for pause processing. If this input is set to 0, ctl_rx_pause_ack[8:0] is not used.	rx_clk_out
ctl_rx_forward_control	Input	A value of 1 indicates that the core forwards control packets. A value of 0 causes core to drop control packets.	rx_clk_out
stat_rx_pause_valid[8:0]	Output	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 MAC Pause packet is received, bit[8] is set to 1.	rx_clk_out
stat_rx_pause_quanta[8:0][15:0]	Output	These nine buses indicate the quanta received for each of the eight priorities in priority based pause operation and global pause operation. If an 802.3x MAC Pause packet is received, the quanta is placed in value [8].	rx_clk_out

Table 2-17: **Pause Interface–TX Path**

Name	Direction	Description	Clock Domain
ctl_tx_pause_req[8:0]	Input	If a bit of this bus is set to 1, the 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.	tx_clk_out
ctl_tx_pause_quanta[8:0][15:0]	Input	These nine buses indicate the quanta to be transmitted for each of the eight priorities in priority based pause operation and the global pause operation. The value for ctl_tx_pause_quanta[8] is used for global pause operation. All other values are used for priority pause operation.	tx_clk_out
ctl_tx_pause_refresh_timer [8:0][15:0]	Input	These nine buses set the retransmission time of pause packets for each of the eight priorities in priority based pause operation and the global pause operation. The value for ctl_tx_pause_refresh_timer[8] is used for global pause operation. All other values are used for priority pause operation.	tx_clk_out
ctl_tx_da_gpp[47:0]	Input	Destination address for transmitting global pause packets.	tx_clk_out
ctl_tx_sa_gpp[47:0]	Input	Source address for transmitting global pause packets.	tx_clk_out
ctl_tx_ethertype_gpp[15:0]	Input	Ethertype for transmitting global pause packets.	tx_clk_out
ctl_tx_opcode_gpp[15:0]	Input	Opcode for transmitting global pause packets.	tx_clk_out
ctl_tx_da_ppp[47:0]	Input	Destination address for transmitting priority pause packets.	tx_clk_out
ctl_tx_sa_ppp[47:0]	Input	Source address for transmitting priority pause packets.	tx_clk_out
ctl_tx_ethertype_ppp[15:0]	Input	Ethertype for transmitting priority pause packets.	tx_clk_out
ctl_tx_opcode_ppp[15:0]	Input	Opcode for transmitting priority pause packets.	tx_clk_out

Table 2-17: **Pause Interface–TX Path *(Cont'd)***

Name	Direction	Description	Clock Domain
ctl_tx_resend_pause	Input	Retransmit 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.	tx_clk_out
stat_tx_pause_valid[8:0]	Output	If a bit of this bus is set to 1, the 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.	tx_clk_out

Auto-Negotiation Ports

Table 2-18 shows the additional ports used for Auto-Negotiation. These signals are found at the `*wrapper.v` hierarchy file.

Table 2-18: **Additional Ports for Auto-Negotiation**

Port Name	Direction	Description	Clock Domain
an_clk	Input	Input Clock for the auto-negotiation circuit. The required frequency is indicated in the readme file for the release. It should be a free running clock.	Refer to Clocking.
an_reset	Input	Asynchronous active-High reset.	Async
ctl_autoneg_enable	Input	Enable signal for auto-negotiation.	an_clk
ctl_autoneg_bypass	Input	Input to disable auto-negotiation and bypass the auto-negotiation function. If this input is asserted, then auto-negotiation is turned off, but the PCS is connected to the outputs to allow operation.	an_clk
ctl_an_nonce_seed[7:0]	Input	8-bit seed to initialize the nonce field Polynomial generator. ***Note:*** This input should always be set to a unique non-zero value for every instance of the auto-negotiator.	an_clk
ctl_an_pseudo_sel	Input	Selects the polynomial generator for the bit 49 random bit generator. If this input is 1, then the polynomial is x7+x6+1. If this input is zero, then the polynomial is x7+x3+1.	an_clk

Table 2-18: **Additional Ports for Auto-Negotiation *(Cont'd)***

Port Name	Direction	Description	Clock Domain
ctl_restart_negotiation	Input	This input is used to trigger a restart of the auto-negotiation, regardless of what state the circuit is currently in.	an_clk
ctl_an_local_fault	Input	This input signal is used to set the local_fault bit of the transmit link codeword.	an_clk
Signals Used for Pause Ability Advertising			
ctl_an_pause	Input	This input signal is used to set the PAUSE bit, (C0), of the transmit link codeword. This signal might not be present if the core does not support pause.	an_clk
ctl_an_asmdir	Input	This input signal is used to set the ASMDIR bit, (C1), of the transmit link codeword. This signal might not be present if the core does not support pause.	an_clk
Ability Signal Inputs			
ctl_an_ability_1000base_kx	Input	These inputs identify the Ethernet protocol abilities that is advertised in the transmit link codeword to the link partner. A value of 1 indicates that the interface advertises that it supports the protocol.	an_clk
ctl_an_ability_100gbase_cr10	Input		an_clk
ctl_an_ability_100gbase_cr4	Input		an_clk
ctl_an_ability_100gbase_kp4	Input		an_clk
ctl_an_ability_100gbase_kr4	Input		an_clk
ctl_an_ability_10gbase_kr	Input		an_clk
ctl_an_ability_10gbase_kx4	Input		an_clk
ctl_an_ability_25gbase_cr	Input		an_clk
ctl_an_ability_25gbase_cr1	Input		an_clk
ctl_an_ability_25gbase_kr	Input		an_clk
ctl_an_ability_25gbase_kr1	Input		an_clk
ctl_an_ability_40gbase_cr4	Input		an_clk
ctl_an_ability_40gbase_kr4	Input		an_clk
ctl_an_ability_50gbase_cr2	Input		an_clk
ctl_an_ability_50gbase_kr2	Input		an_clk
ctl_an_fec_request	Input	Used to control the clause 74 FEC request bit in the transmit link codeword. This signal might not be present if the IP core does not support clause 74 FEC.	an_clk

Table 2-18: **Additional Ports for Auto-Negotiation *(Cont'd)***

Port Name	Direction	Description	Clock Domain
ctl_an_fec_ability_override	Input	Used to control the clause 74 FEC ability bit in the transmit link codeword. If this input is set, then the FEC ability bit in the transmit link codeword is cleared. This signal might not be present if the IP core does not support clause 74 FEC.	an_clk
ctl_an_cl91_fec_ability	Input	This bit is used to indicate clause 91 FEC ability.	an_clk
ctl_an_cl91_fec_request	Input	This bit is used to request clause 91 FEC.	an_clk
stat_an_link_cntl_1000base_kx[1:0]	Output	Link Control outputs from the auto-negotiation controller for the various Ethernet protocols. Settings are as follows: • 00: DISABLE; PCS is disconnected • 01: SCAN_FOR_CARRIER; RX is connected to PCS • 11: ENABLE; PCS is connected for mission mode operation • 10: not used	an_clk
stat_an_link_cntl_100gbase_cr10[1:0]	Output		an_clk
stat_an_link_cntl_100gbase_cr4[1:0]	Output		an_clk
stat_an_link_cntl_100gbase_kp4[1:0]	Output		an_clk
stat_an_link_cntl_100gbase_kr4[1:0]	Output		an_clk
stat_an_link_cntl_10gbase_kr[1:0]	Output		an_clk
stat_an_link_cntl_10gbase_kx4[1:0]	Output		an_clk
stat_an_link_cntl_25gbase_cr[1:0]	Output		an_clk
stat_an_link_cntl_25gbase_cr1[1:0]	Output		an_clk
stat_an_link_cntl_25gbase_kr[1:0]	Output		an_clk
stat_an_link_cntl_25gbase_kr1[1:0]	Output		an_clk
stat_an_link_cntl_40gbase_cr4[1:0]	Output		an_clk
stat_an_link_cntl_40gbase_kr4[1:0]	Output		an_clk
stat_an_link_cntl_50gbase_cr2[1:0]	Output		an_clk
stat_an_link_cntl_50gbase_kr2[1:0]	Output		an_clk
stat_an_fec_enable	Output	Used to enable the use of clause 74 FEC on the link.	an_clk
stat_an_rs_fec_enable	Output	Used to enable the use of clause 91 FEC on the link.	an_clk
stat_an_tx_pause_enable	Output	Used to enable station-to-station (global) pause packet generation in the transmit path to control data flow in the receive path.	an_clk
stat_an_rx_pause_enable	Output	Used to enable station-to-station (global) pause packet interpretation in the receive path, to control data flow from the transmitter.	an_clk

Table 2-18: **Additional Ports for Auto-Negotiation *(Cont'd)***

Port Name	Direction	Description	Clock Domain
stat_an_autoneg_complete	Output	Indicates the auto-negotiation is complete and RX link status from the PCS has been received.	an_clk
stat_an_parallel_detection_fault	Output	Indicated a parallel detection fault during auto-negotiation.	an_clk
stat_an_lp_ability_1000base_kx	Output	These signals indicate the advertised protocol from the link partner. They all become valid when the output signal stat_AN_lp_Ability_Valid is asserted. A value of 1 indicates that the protocol is advertised as supported by the link partner.	an_clk
stat_an_lp_ability_100gbase_cr10	Output		an_clk
stat_an_lp_ability_100gbase_cr4	Output		an_clk
stat_an_lp_ability_100gbase_kp4	Output		an_clk
stat_an_lp_ability_100gbase_kr4	Output		an_clk
stat_an_lp_ability_10gbase_kr	Output		an_clk
stat_an_lp_ability_10gbase_kx4	Output		an_clk
stat_an_lp_ability_25gbase_cr	Output		an_clk
stat_an_lp_ability_25gbase_kr	Output		an_clk
stat_an_lp_ability_40gbase_cr4	Output		an_clk
stat_an_lp_ability_40gbase_kr4	Output		an_clk
stat_an_lp_ability_25gbase_cr1	Output	Indicates the advertised protocol from the link partner. Becomes valid when the output signal stat_AN_lp_Extended_Ability_Valid is asserted. A value of 1 indicates that the protocol is advertised as supported by the link partner.	an_clk
stat_an_lp_ability_25gbase_kr1	Output	Indicates the advertised protocol from the link partner. Becomes valid when the output signal stat_AN_lp_Extended_Ability_Valid is asserted. A value of 1 indicates that the protocol is advertised as supported by the link partner.	an_clk
stat_an_lp_ability_50gbase_cr2	Output	Indicates the advertised protocol from the link partner. Becomes valid when the output signal stat_AN_lp_Extended_Ability_Valid is asserted. A value of 1 indicates that the protocol is advertised as supported by the link partner.	an_clk

Table 2-18: **Additional Ports for Auto-Negotiation *(Cont'd)***

Port Name	Direction	Description	Clock Domain
stat_an_lp_ability_50gbase_kr2	Output	Indicates the advertised protocol from the link partner. Becomes valid when the output signal stat_AN_lp_Extended_Ability_Valid is asserted. A value of 1 indicates that the protocol is advertised as supported by the link partner.	an_clk
stat_an_lp_pause	Output	This signal indicates the advertised value of the PAUSE bit, (C0), in the receive link codeword from the link partner. It becomes valid when the output signal stat_AN_lp_Ability_Valid is asserted.	an_clk
stat_an_lp_asm_dir	Output	This signal indicates the advertised value of the ASMDIR bit, (C1), in the receive link codeword from the link partner. It becomes valid when the output signal stat_AN_lp_Ability_Valid is asserted.	an_clk
stat_an_lp_fec_ability	Output	This signal indicates the advertised value of the FEC ability bit in the receive link codeword from the link partner. It becomes valid when the output signal stat_AN_lp_Ability_Valid is asserted.	an_clk
stat_an_lp_fec_request	Output	This signal indicates the advertised value of the FEC Request bit in the receive link codeword from the link partner. It becomes valid when the output signal stat_AN_lp_Ability_Valid is asserted.	an_clk
stat_an_lp_autoneg_able	Output	This output signal indicates that the link partner is able to perform auto-negotiation. It becomes valid when the output signal stat_AN_lp_Ability_Valid is asserted.	an_clk
stat_an_lp_ability_valid	Output	This signal indicates when all of the link partner advertisements become valid.	an_clk

Table 2-18: **Additional Ports for Auto-Negotiation *(Cont'd)***

Port Name	Direction	Description	Clock Domain
an_loc_np_data[47:0]	Input	Local Next Page codeword. This is the 48-bit codeword used if the loc_np input is set. In this data field, the bits NP, ACK, and T, bit positions 15, 14, 12, and 11, are not transferred as part of the next page codeword. These bits are generated in the Auto-Negotiation Intellectual Property Core (ANIPC). However, the Message Protocol bit, MP, in bit position 13, is transferred.	an_clk
an_lp_np_data[47:0]	Output	Link Partner Next Page Data. This 48-bit word is driven by the ANIPC with the 48-bit next page codeword from the remote link partner.	an_clk
ctl_an_loc_np	Input	Local Next Page indicator. If this bit is 1, the ANIPC transfers the next page word at input loc_np_data to the remote link partner. If this bit is 0, the ANIPC does not initiate the next page protocol. If the link partner has next pages to send, and the loc_np bit is clear, the ANIPC transfers null message pages.	an_clk
ctl_an_lp_np_ack	Input	Link Partner Next Page Acknowledge. This is used to signal the ANIPC that the next page data from the remote link partner at output pin lp_np_data has been read by the local host. When this signal goes High, the ANIPC acknowledges reception of the next page codeword to the remote link partner and initiate transfer of the next codeword. During this time, the ANIPC will remove the lp_np signal until the new next page information is available.	an_clk

Table 2-18: **Additional Ports for Auto-Negotiation *(Cont'd)***

Port Name	Direction	Description	Clock Domain
stat_an_loc_np_ack	Output	This signal is used to indicate to the local host that the local next page data, presented at input pin loc_np_data, has been taken. This signal pulses High for 1 clock period when the ANIPC samples the next page data on input pin loc_np_data. When the local host detects this signal High, it must replace the 48-bit next page codeword at input pin loc_np_data with the next 48-bit codeword to be sent. If the local host has no more next pages to send, it must clear the loc_np input.	an_clk
stat_an_lp_np	Output	Link Partner Next Page. This signal is used to indicate that there is a valid 48-bit next page codeword from the remote link partner at output pin lp_np_data. This signal is driven Low when the lp_np_ack input signal is driven High, indicating that the local host has read the next page data. It remains Low until the next codeword becomes available on the lp_np_data output pin, the lp_np output is driven High again.	an_clk
stat_an_lp_ability_extended_fec[1:0]	Output	This output indicates the extended FEC abilities as defined in Schedule 3.	an_clk
stat_an_lp_extended_ability_valid	Output	When this bit is 1, it indicates that the detected extended abilities are valid.	an_clk
stat_an_lp_rf	Output	This bit indicates link partner remote fault.	an_clk

Table 2-18: **Additional Ports for Auto-Negotiation *(Cont'd)***

Port Name	Direction	Description	Clock Domain
stat_an_start_tx_disable	Output	When ctl_autoneg_enable is High and ctl_autoneg_bypass is Low, this signal, stat_an_start_tx_disable, cycles High for 1 clock cycle at the very start of the TX_DISABLE phase of auto-negotiation. That is, when auto-negotiation enters state TX_DISABLE, this output will cycle High for 1 clock period. It effectively signals the start of auto-negotiation.	an_clk
stat_an_start_an_good_check	Output	When ctl_autoneg_enable is High and ctl_autoneg_bypass is Low, this signal, stat_an_start_an_good_check, cycles High for 1 clock cycle at the very start of the AN_GOOD_CHECK phase of auto-negotiation. That is, when auto-negotiation enters the state AN_GOOD_CHECK, this output will cycle High for 1 clock period. It effectively signals the start of link training. However, if link training is not enabled, that is. if the input ctl_lt_training_enable is Low, the stat_an_start_an_good_check output effectively signals the start of mission-mode operation.	an_clk

Link Training Ports

Table 2-19 shows the Link Training ports.

Table 2-19: **Link Training Ports**

Port Name	Direction	Description	Clock Domain
ctl_lt_training_enable	Input	Enables link training. When link training is disabled, all PCS lanes function in mission mode.	tx_serdes_clk
ctl_lt_restart_training	Input	This signal triggers a restart of link training regardless of the current state.	tx_serdes_clk
ctl_lt_rx_trained	Input	This signal is asserted to indicate that the receiver finite impulse response (FIR) filter coefficients have all been set, and that the receiver portion of training is complete.	tx_serdes_clk

Table 2-19: **Link Training Ports *(Cont'd)***

Port Name	Direction	Description	Clock Domain
stat_lt_signal_detect	Output	This signal indicates when the respective link training state machine has entered the SEND_DATA state, in which normal PCS operation can resume.	tx_serdes_clk
stat_lt_training	Output	This signal indicates when the respective link training state machine is performing link training.	tx_serdes_clk
stat_lt_training_fail	Output	This signal is asserted during link training if the corresponding link training state machine detects a time-out during the training period.	tx_serdes_clk
stat_lt_frame_lock	Output	When link training has begun, these signals are asserted, for each physical medium dependent (PMD) lane, when the corresponding link training receiver is able to establish a frame synchronization with the link partner.	rx_serdes_clk
stat_lt_preset_from_rx	Output	This signal reflects the value of the preset control bit received in the control block from the link partner.	rx_serdes_clk
stat_lt_initialize_from_rx	Output	This signal reflects the value of the initialize control bit received in the control block from the link partner.	rx_serdes_clk
stat_lt_k_p1_from_rx0[1:0]	Output	This 2-bit field indicates the update control bits for the k+1 coefficient, as received from the link partner in the control block.	rx_serdes_clk
stat_lt_k0_from_rx0[1:0]	Output	This 2-bit field indicates the update control bits for the k0 coefficient, as received from the link partner in the control block.	rx_serdes_clk
stat_lt_k_m1_from_rx0[1:0]	Output	This 2-bit field indicates the update control bits for the k-1 coefficient, as received from the link partner in the control block.	rx_serdes_clk
stat_lt_stat_p1_from_rx0[1:0]	Output	This 2-bit field indicates the update status bits for the k+1 coefficient, as received from the link partner in the status block.	rx_serdes_clk
stat_lt_stat0_from_rx0[1:0]	Output	This 2-bit field indicates the update status bits for the k0 coefficient, as received from the link partner in the status block.	rx_serdes_clk

Table 2-19: **Link Training Ports *(Cont'd)***

Port Name	Direction	Description	Clock Domain
stat_lt_stat_m1_from_rx0[1:0]	Output	This 2-bit field indicates the update status bits for the k-1 coefficient, as received from the link partner in the status block.	rx_serdes_clk
ctl_lt_pseudo_seed0[10:0]	Input	This 11-bit signal seeds the training pattern generator.	tx_serdes_clk
ctl_lt_preset_to_tx	Input	This signal is used to set the value of the preset bit that is transmitted to the link partner in the control block of the training frame.	tx_serdes_clk
ctl_lt_initialize_to_tx	Input	This signal is used to set the value of the initialize bit that is transmitted to the link partner in the control block of the training frame.	tx_serdes_clk
ctl_lt_k_p1_to_tx0[1:0]	Input	This 2-bit field is used to set the value of the k+1 coefficient update field that is transmitted to the link partner in the control block of the training frame.	tx_serdes_clk
ctl_lt_k0_to_tx0[1:0]	Input	This 2-bit field is used to set the value of the k0 coefficient update field that is transmitted to the link partner in the control block of the training frame.	tx_serdes_clk
ctl_lt_k_m1_to_tx0[1:0]	Input	This 2-bit field is used to set the value of the k-1 coefficient update field that is transmitted to the link partner in the control block of the training frame.	tx_serdes_clk
ctl_lt_stat_p1_to_tx0[1:0]	Input	This 2-bit field is used to set the value of the k+1 coefficient update status that is transmitted to the link partner in the status block of the training frame.	tx_serdes_clk
ctl_lt_stat0_to_tx0[1:0]	Input	This 2-bit field is used to set the value of the k0 coefficient update status that is transmitted to the link partner in the status block of the training frame.	tx_serdes_clk
ctl_lt_stat_m1_to_tx0[1:0]	Input	This 2-bit field is used to set the value of the k-1 coefficient update status that is transmitted to the link partner in the status block of the training frame.	tx_serdes_clk
stat_lt_rx_sof[1-1:0]	Output	This output is High for 1 RX SerDes clock cycle to indicate the start of the link training frame.	rx_serdes_clk

IEEE 802.3 Clause 74 FEC Interface

Table 2-20 shows the IEEE 802.3 Clause 74 FEC Control/Status and Statistics signals.

Table 2-20: **IEEE 802.3 Clause 74 FEC Interface Control/Status/Statistics Signals**

Signal	Direction	Clock	Description
ctl_fec_tx_enable	Input	tx_serdes_clk	Asserted to enable the clause 74 FEC encoding on the transmitted data
ctl_fec_rx_enable	Input	rx_serdes_clk	Asserted to enable the clause 74 FEC decoding of the received data
ctl_fec_enable_error_to_pcs	Input	rx_serdes_clk	Clause 74 FEC enable error to PCS
stat_fec_inc_correct_count[3:0]	Output	rx_serdes_clk	This signal will be asserted roughly every 32 words, while the ctl_rx_fec_enable is asserted, if the FEC decoder detected and corrected a bit errors in the corresponding frame.
stat_fec_inc_cant_correct_count[3:0]	Output	rx_serdes_clk	This signal will be asserted roughly every 32 words, while the ctl_rx_fec_enable is asserted, if the FEC decoder detected bit.
stat_fec_lock_error[3:0]	Output	rx_serdes_clk	This signal is asserted if the FEC decoder has been unable to detect the frame boundary after about 5 ms. It is cleared when the frame boundary is detected.
stat_fec_rx_lock[3:0]	Output	rx_serdes_clk	This signal is asserted while the ctl_fec_rx_enable is asserted when the FEC decoder detects the frame boundary.

IEEE 802.3 Clause 108 RS-FEC Interface

Table 2-21 shows the IEEE 802.3 Clause 108 RS-FEC Control/Status and Statistics signals.

Table 2-21: **IEEE 802.3 Clause 108 (RS-FEC) Control/Status/Statistics Signals**

Signal	Direction	Clock	Description
ctl_rx_rsfec_enable_correction	Input	rx_serdes_clk	Equivalent to MDIO register 1.200.0 • 0: Decoder performs error detection without error correction (see IEEE 802.3by Clause 91.5.3.3). • 1: Decoder also performs error correction.
ctl_rx_rsfec_enable_indication	Input	rx_serdes_clk	Equivalent to MDIO register 1.200.1 • 0: Bypass the error indication function (see IEEE Std 802.3by Clause 91.5.3.3). • 1: Decoder indicates errors to the PCS sublayer.
ctl_rsfec_enable	Input	rx_serdes_clk	Enable RS-FEC function. ***Note:*** Some variants of the 10G/25G Ethernet IP Subsystem can have separate TX and RX enable signals.
ctl_rsfec_ieee_error_indication_mode	Input	rx_serdes_clk	This signal indicates that the core conforms to the IEEE RS-FEC specification • 1: Core conforms to the IEEE RS-FEC specification. • 0: If ctl_rx_rsfec_enable_correction and ctl_rx_rsfec_enable_indication are set to zero, the RS decoder is bypassed.
ctl_rsfec_consortium_25g	Input	rx_serdes_clk	This signal switches between IEEE Clause 108 and 25G Ethernet Consortium modes • 1 = 25G Consortium specification mode. • 0 = IEEE 802.3by mode. ***Note:*** Some variants of the 10G/25G Subsystem can have individual RX and TX consortium signals.

Table 2-21: **IEEE 802.3 Clause 108 (RS-FEC) Control/Status/Statistics Signals *(Cont'd)***

Signal	Direction	Clock	Description
stat_rx_rsfec_hi_ser	Output	rx_serdes_clk	Indicates high symbol error. Set to 1 if the number of RS-FEC symbol errors in a window of 8192 codewords exceeds the threshold of 417. Set to 0 otherwise.
stat_rx_rsfec_lane_alignment_status	Output	rx_serdes_clk	A value of 1 indicates that the RX RS-FEC block has achieved alignment on the data from the transceiver.
stat_rx_rsfec_corrected_cw_inc	Output	rx_serdes_clk	Increment for corrected errors.
stat_rx_rsfec_uncorrected_cw_inc	Output	rx_serdes_clk	Increment for uncorrected errors.
stat_rx_rsfec_err_count0_inc[2:0]	Output	rx_serdes_clk	Increment for detected errors.
stat_tx_rsfec_lane_alignment_status	Output	tx_serdes_clk	A value of 1 indicates that the TX RS-FEC block has achieved alignment on the incoming PCS data.

Port Descriptions – PCS Variant

This section shows the 10G/25G PCS core ports. These are the ports when the PCS-only option is provided. There are no FCS functions. The PCS does not contain the Pause and Flow Control ports. The system interface is XGMII/25GMII. Table 2-22 shows the PCS variant I/O ports.

Table 2-22: **PCS Variant I/O Ports**

Name	Direction	Description	Clock Domain
stat_tx_local_fault	Output	A value of 1 indicates the transmit encoder state machine is in the TX_INIT state. This output is level sensitive.	tx_mii_clk
ctl_rx_prbs31_test_pattern_enable	Input	Corresponds to MDIO register bit 3.42.5 as defined in Clause 45. Takes first precedence.	rx_clk_out
ctl_rx_test_pattern_enable	Input	Test pattern enable for the RX core. A value of 1 enables test mode. Corresponds to MDIO register bit 3.42.2 as defined in Clause 45. Takes second precedence.	rx_clk_out

Table 2-22: **PCS Variant I/O Ports *(Cont'd)***

Name	Direction	Description	Clock Domain
ctl_rx_data_pattern_select	Input	Corresponds to MDIO register bit 3.42.0 as defined in Clause 45.	rx_clk_out
ctl_rx_test_pattern	Input	Test pattern enable for the RX core to receive scrambled idle pattern. Takes third precedence.	rx_clk_out
ctl_tx_prbs31_test_pattern_enable	Input	Corresponds to MDIO register bit 3.42.4 as defined in Clause 45. Takes first precedence.	tx_mii_clk
ctl_tx_test_pattern_enable	Input	Test pattern generation enable for the TX core. A value of 1 enables test mode. Corresponds to MDIO register bit 3.42.3 as defined in Clause 45. Takes second precedence.	tx_mii_clk
ctl_tx_test_pattern_select	Input	Corresponds to MDIO register bit 3.42.1 as defined in Clause 45.	tx_mii_clk
ctl_tx_data_pattern_select	Input	Corresponds to MDIO register bit 3.42.0 as defined in Clause 45.	tx_mii_clk
ctl_tx_test_pattern_seed_a[57:0]	Input	Corresponds to MDIO registers 3.34 through to 3.37 as defined in Clause 45.	tx_mii_clk
ctl_tx_test_pattern_seed_b[57:0]	Input	Corresponds to MDIO registers 3.38 through to 3.41 as defined in Clause 45.	tx_mii_clk
ctl_tx_test_pattern	Input	Scrambled idle Test pattern generation enable for the TX core. A value of 1 enables test mode. Takes third precedence.	tx_mii_clk
stat_tx_fifo_error	Output	Transmit clock compensation FIFO error indicator. A value of 1 indicates the clock compensation FIFO under or overflowed. If this output is sampled as a 1 in any clock cycle, the corresponding port must be reset to resume proper operation.	tx_mii_clk

Table 2-22: **PCS Variant I/O Ports *(Cont'd)***

Name	Direction	Description	Clock Domain
stat_rx_fifo_error	Output	Receive clock compensation FIFO error indicator. A value of 1 indicates the clock compensation FIFO under or overflowed. This condition only occurs if the PPM difference between the recovered clock and the local reference clock is greater than ±200 ppm. If this output is sampled as a 1 in any clock cycle, the corresponding port must be reset to resume proper operation.	rx_clk_out
stat_rx_local_fault	Output	A value of 1 indicates the receive decoder state machine is in the RX_INIT state. This output is level sensitive.	rx_clk_out
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 45. This output is level sensitive.	rx_clk_out
stat_rx_block_lock	Output	Block lock status for each PCS lane. A value of 1 indicates the corresponding lane has achieved a block lock as defined in Clause 49. Corresponds to MDIO register bit 3.50.7:0 and 3.51.11:0 as defined in Clause 45. This output is level sensitive.	rx_clk_out
stat_rx_error	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 45. This output is pulsed for one clock cycle.	rx_clk_out
stat_rx_valid_ctrl_code	Output	Indicates that a PCS block with a valid control code was received.	rx_clk_out

Table 2-22: **PCS Variant I/O Ports *(Cont'd)***

Name	Direction	Description	Clock Domain
stat_rx_error_valid	Output	Increment valid indicator. If this signal is a 1 in any clock cycle, the value of stat_rx_error_valid[0:0] is valid.	rx_clk_out
stat_rx_bad_code	Output	Increment for 64B/66B code violations. This signal indicates the number of 64b/66b words received with an invalid block or if a wrong 64b/66b block sequence was detected. This output can be used to generate MDIO register 3.33:7:0 as defined in Clause 45.	rx_clk_out
stat_rx_bad_code_valid	Output	Increment valid indicator. If this signal is a 1 in any clock cycle, the value of stat_rx_bad_code[0:0] is valid.	rx_clk_out
stat_rx_framing_err	Output	Increment value for number of bad sync header bits detected. The value of this bus is only valid in the same cycle that the corresponding stat_rx_framing_err_valid is a 1.	rx_clk_out
stat_rx_framing_err_valid	Output	Increment valid indicator. If this signal is a 1 in any clock cycle, the value of stat_rx_framing_err[0:0] is valid.	rx_clk_out

Transceiver Interface Ports

Table 2-23 shows the transceiver I/O ports.

Table 2-23: **Transceiver I/O**

Name	Direction	Description	Clock Domain
GT_reset	Input	Active-High reset for the transceiver startup FSM. Note that this signal also initiates the reset sequence for the entire 10G/25G Ethernet IP core.	Async
refclk_n0	Input	Differential reference clock input for the SerDes, negative phase.	Refer to Clocking.
refclk_p0	Input	Differential reference clock input for the SerDes, negative phase.	Refer to Clocking.
rx_serdes_data_n0	Input	Serial data from the line; negative phase of the differential signal.	Refer to Clocking.
rx_serdes_data_p0	Input	Serial data from the line; positive phase of the differential signal.	Refer to Clocking.
tx_serdes_data_n0	Output	Serial data to the line; negative phase of the differential signal.	Refer to Clocking.
tx_serdes_data_p0	Output	Serial data to the line; positive phase of the differential signal.	Refer to Clocking.
tx_serdes_clkout	Output	When present, same as tx_clk_out.	Refer to Clocking.

XGMII/25GMII Interface Ports

Table 2-24 shows the XGMII/25GMII I/O ports.

Table 2-24: **XGMII/25GMII Interface Ports**

Name	Direction	Description	Clock Domain
rx_mii_d[63:0]	Output	Receive XGMII/25GMII Data bus.	rx_mii_clk
rx_mii_c[7:0]	Output	Receive XGMII/25GMII Control bus.	rx_mii_clk
rx_mii_clk	Input	Receive XGMII/25GMII Clock input.	Refer to Clocking.
tx_mii_d[63-1:0]	Input	Transmit XGMII/25GMII Data bus.	tx_mii_clk
tx_mii_c[7:0]	Input	Transmit XGMII/25GMII Control bus.	tx_mii_clk
rx_clk_out	Output	This is the reference clock for RX PCS stats.	Refer to Clocking.
tx_clk_out (or tx_mii_clk)	Output	This output is used to clock the TX MII bus. Data is clocked on the positive edge of this signal.	Refer to Clocking.
rx_mii_reset	Input	Reset input for the RX MII interface.	Async
tx_mii_reset	Input	Reset input for the TX MII interface.	Async

Miscellaneous Status/Control Ports

Table 2-25 shows the miscellaneous status/control ports.

Table 2-25: **Miscellaneous Status/Control Ports**

Name	Direction	Description	Clock Domain
dclk	Input	Dynamic Reconfiguration Port (DRP) clock input. The required frequency is set by providing the value in the **GT DRP Clock** field in the Vivado® IDE GT Selection and Configuration tab. This must be a free running input clock.	Refer to Clocking.
ctl_local_loopback	Input	When High, this signal places the transceiver into the PMA loopback state.	Async

IEEE 802.3 Clause 74 FEC Interface

Table 2-26 shows the IEEE 802.3 Clause 74 FEC Control/Status and Statistics signals.

Table 2-26: **IEEE 802.3 Clause 74 FEC Interface Control/Status/Statistics Signals**

Signal	Direction	Clock	Description
ctl_fec_tx_enable	Input	tx_serdes_clk	Asserted to enable the clause 74 FEC encoding on the transmitted data.
ctl_fec_rx_enable	Input	rx_serdes_clk	Asserted to enable the clause 74 FEC decoding of the received data.
ctl_fec_enable_error_to_pcs	Input	rx_serdes_clk	Clause 74 FEC enable error to PCS.
stat_fec_inc_correct_count[3:0]	Output	rx_serdes_clk	This signal will be asserted roughly every 32 words, while the ctl_rx_fec_enable is asserted, if the FEC decoder detected and corrected a bit errors in the corresponding frame.
stat_fec_inc_cant_correct_count[3:0]	Output	rx_serdes_clk	This signal will be asserted roughly every 32 words, while the ctl_rx_fec_enable is asserted, if the FEC decoder detected bit.
stat_fec_lock_error[3:0]	Output	rx_serdes_clk	This signal is asserted if the FEC decoder has been unable to detect the frame boundary after about 5 ms. It is cleared when the frame boundary is detected.
stat_fec_rx_lock[3:0]	Output	rx_serdes_clk	This signal is asserted while the ctl_fec_rx_enable is asserted when the FEC decoder detects the frame boundary.

IEEE 802.3 Clause 108 RS-FEC Interface

Table 2-27 shows the IEEE 802.3 Clause 108 RS-FEC Control/Status and Statistics signals.

Table 2-27: **IEEE 802.3 Clause 108 (RS-FEC) Control/Status/Statistics Signals**

Signal	Direction	Clock	Description
ctl_rx_rsfec_enable_correction	Input	rx_serdes_clk	Equivalent to MDIO register 1.200.0 • 0: Decoder performs error detection without error correction (see IEEE 802.3by Clause 91.5.3.3). • 1: the decoder also performs error correction.
ctl_rx_rsfec_enable_indication	Input	rx_serdes_clk	Equivalent to MDIO register 1.200.1 • 0: Bypass the error indication function (see IEEE Std 802.3by Clause 91.5.3.3). • 1: Decoder indicates errors to the PCS sublayer
ctl_rsfec_enable	Input	rx_serdes_clk	Enable RS-FEC function. **Note:** Some variants of the 10G/25G Ethernet IP Subsystem can have separate TX and RX enable signals.
ctl_rsfec_ieee_error_indication_mode	Input	rx_serdes_clk	This signal indicates that the core conforms to the IEEE RS-FEC specification • 1: Core conforms to the IEEE RS-FEC specification. • 0: If ctl_rx_rsfec_enable_correction and ctl_rx_rsfec_enable_indication are set to zero, the RS decoder is bypassed.

Table 2-27: **IEEE 802.3 Clause 108 (RS-FEC) Control/Status/Statistics Signals *(Cont'd)***

Signal	Direction	Clock	Description
ctl_rsfec_consortium_25g	Input	rx_serdes_clk	This signal switches between IEEE Clause 108 and 25G Ethernet Consortium modes • 1 = 25G Consortium specification mode; • 0 = IEEE 802.3by mode **Note:** Some variants of the 10G/ 25G Subsystem can have individual RX and TX consortium signals.
stat_rx_rsfec_hi_ser	Output	rx_serdes_clk	Indicates high symbol error. Set to 1 if the number of RS-FEC symbol errors in a window of 8192 codewords exceeds the threshold of 417. Set to 0 otherwise
stat_rx_rsfec_lane_alignment_status	Output	rx_serdes_clk	A value of 1 indicates that the RX RS-FEC block has achieved alignment on the data from the transceiver.
stat_rx_rsfec_corrected_cw_inc	Output	rx_serdes_clk	Increment for corrected errors
stat_rx_rsfec_uncorrected_cw_inc	Output	rx_serdes_clk	Increment for uncorrected errors
stat_rx_rsfec_err_count0_inc[2:0]	Output	rx_serdes_clk	Increment for detected errors
stat_tx_rsfec_lane_alignment_status	Output	tx_serdes_clk	A value of 1 indicates that the TX RS-FEC block has achieved alignment on the incoming PCS data.

Port Descriptions – 10G Ethernet MAC (64-bit) Variant

MII Interface

This interface is used to connect to the physical layer, where this is a separate device or implemented in the FPGA beside the Ethernet MAC core. Table 2-28 shows the port associated with this interface.

Table 2-28: **MII Interface**

Name	Direction	Clock Domain	Description
rx_mac_mii_d[63:0]/rx_mii_d	Input	rx_mii_clk	Receive Data from PHY
rx_mac_mii_c[7:0]/rx_mii_c	Input	rx_mii_clk	Receive Control from PHY
rx_mac_mii_clk/rx_mii_clk	Input		Received clock connected from PHY
rx_mac_mii_reset/rx_mii_reset	Input	rx_mii_clk	Reset signal received from PHY
tx_mac_mii_d[63:0]/tx_mii_d	Output	tx_mii_clk	Transmit Data to PHY
tx_mac_mii_c[7:0]/tx_mii_c	Output	tx_mii_clk	Transmit Control to PHY
tx_mac_mii_clk/tx_mii_clk	Output		XGMII output clock sent to external PHY
tx_mac_mii_reset/tx_mii_reset	Output	tx_mii_clk	Reset signal sent to external PHY

AXI4-Stream Interface

The AXI4-Stream interface clock and reset signals are shown in Table 2-29.

Table 2-29: **AXI4-Stream- Clocks and Resets**

Name	Direction	Clock Domain	Description
rx_reset	Input	Async	Reset for the RX circuits. This signal is active-High (1=Reset) and must be held High until the clock, clk is stable. The core handles synchronizing the rx_reset input to the appropriate clock domains within the core.
tx_reset	Input	Async	Reset for the TX circuits. This signal is active-High (1=Reset) and must be held High until the clock, clk is stable. The core handles synchronizing the tx_reset input to the appropriate clock domains within the core.
clk	Input		All signals between the 10G/25G High Speed Ethernet Subsystem and the user-side logic are synchronized to the positive edge of this signal.

AXI4-Stream Interface - TX

Table 2-30 shows the AXI4-Stream transmit interface signals.

Table 2-30: **AXI4-Stream Transmit Interface Signal**

Name	Direction	Clock Domain	Description
tx_axis_tdata[63:0]	Input	clk	AXI4-Stream Data
tx_axis_tkeep[7:0]	Input	clk	AXI4-Stream Data Control
tx_axis_tlast	Input	clk	AXI4-Stream signal indicating End of Packet
tx_axis_tvalid	Input	clk	AXI4-Stream Data Valid
tx_axis_tuser	Input	clk	AXI4-Stream User Sideband interface signal 1 indicates a bad packet 0 indicates a good packet
tx_axis_tready	Output	clk	AXI4-Stream acknowledge signal to indicate to start the Data transfer
tx_parityin[7:0]	Input	clk	AXI4-Stream user-generated parity. Follows the same data lane mapping as tx_axis_tkeep.

Data Lane Mapping

For transmit data, `tx_axis_tdata[63:0]`, the port is logically divided into lane 0 to lane 7. See Table 2-31.

Table 2-31: **tx_axis_tdata Lanes**

Lane/tx_axis_tkeep	tx_axis_tdata[63:0] bits
0	7:0
1	15:8
2	23:16
3	31:24
4	39:32
5	47:40
6	55:48
7	63:56

Normal Transmission

The timing of a normal frame transfer is shown in Figure 2-13. When the client wants to transmit a frame, it asserts the `tx_axis_tvalid` and places the data and control in `tx_axis_tdata` and `tx_axis_tkeep` in the same clock cycle. When this data is accepted by the core, indicated by `tx_axis_tready` being asserted, the client must provide the next cycle of data. If `tx_axis_tready` is not asserted by the core, the client must hold the current valid data value until it is. The end of the packet is indicated to the core by `tx_axis_tlast` asserted for 1 cycle. The bits of `tx_axis_tkeep` are set appropriately to indicate the number of valid bytes in the final data transfer. `tx_axis_tuser` is also asserted to indicate a bad packet. After `tx_axis_tlast` is deasserted, any data and control is deemed invalid until `tx_axis_tvalid` is next asserted.

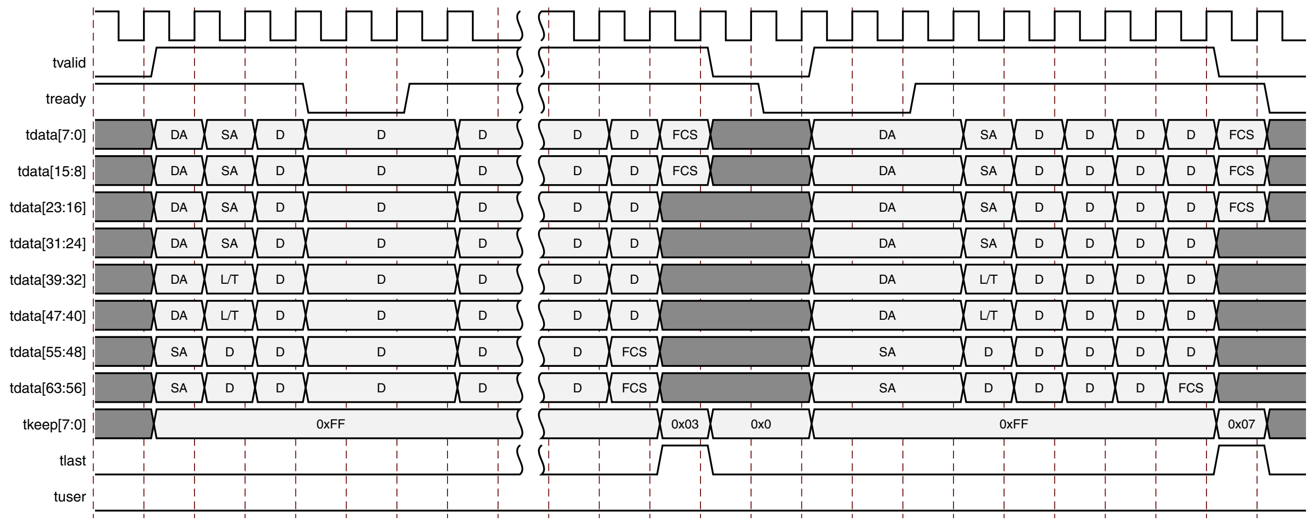


Figure 2-13: **Normal Frame Transfer – 64 Bits**

Aborting a Transmission

The aborted transfer of a packet on the client interface is called an underrun. This can happen if a FIFO in the AXI Transmit client interface empties before a frame is completed.

This is indicated to the core in one of two ways:

- An explicit error in which a frame transfer is aborted by deasserting `tx_axis_tuser` High while `tx_axis_tlast` is High. [see Figure 2-13]
- An implicit underrun in which a frame transfer is aborted by deasserting `tx_axis_tvalid` without asserting `tx_axis_tlast`.

AXI4-Stream Interface – RX

Table 2-32 shows the AXI4-Stream receive interface signals.

Table 2-32: **AXI4-Stream Receive Interface Signals**

Name	Direction	Clock Domain	Description
rx_axis_tdata[63:0]	Output	clk	AXI4-Stream Data to upper layer
rx_axis_tkeep[7:0]	Output	clk	AXI4-Stream Data Control to upper layer
rx_axis_tlast	Output	clk	AXI4-Stream signal indicating an end of packet
rx_axis_tvalid	Output	clk	AXI4-Stream Data Valid
rx_axis_tuser	Output	clk	AXI4-Stream User Sideband interface 1 indicates a bad packet has been received 0 indicates a good packet has been received
rx_parityout[7:0]	Output	clk	AXI4-Stream core-generated parity. Follows the same data lane mapping as rx_axis_tkeep.

Data Lane Mapping

For receive data, rx_axis_tdata[63:0], the port is logically divided into lane 0 to lane 7. See Table 2-33.

Table 2-33: **rx_axis_tdata Lanes**

Lane/rx_axis_tkeep	rx_axis_tdata[63:0] bits
0	7:0
1	15:8
2	23:16
3	31:24
4	39:32
5	47:40
6	55:48
7	63:56

Normal Frame Reception

The timing of a normal inbound frame transfer is represented in Figure 2-14. The client must be prepared to accept data at any time; there is no buffering within the core to allow for latency in the receive client. When frame reception begins, data is transferred on consecutive clock cycles to the receive client.

During frame reception, `rx_axis_tvalid` is asserted to indicate that valid frame data is being transferred to the client on `rx_axis_tdata`. All bytes are always valid throughout the frame, as indicated by all `rx_axis_tkeep` bits being set to 1, except during the final transfer of the frame when `rx_axis_tlast` is asserted. During this final transfer of data for a frame, `rx_axis_tkeep` bits indicate the final valid bytes of the frame using the mapping from above. The valid bytes of the final transfer always lead out from `rx_axis_tdata[7:0]` (`rx_axis_tkeep[0]`) because Ethernet frame data is continuous and is received least significant byte first.

The `rx_axis_tlast` is asserted and `rx_axis_tuser` is deasserted, along with the final bytes of the transfer, only after all the frame checks are completed. This is after the frame check sequence (FCS) field has been received. The core keeps the `rx_axis_tuser` signal deasserted to indicate that the frame was successfully received and that the frame should be analyzed by the client. This is also the end of the packet signaled by `rx_axis_tlast` asserted for one cycle.

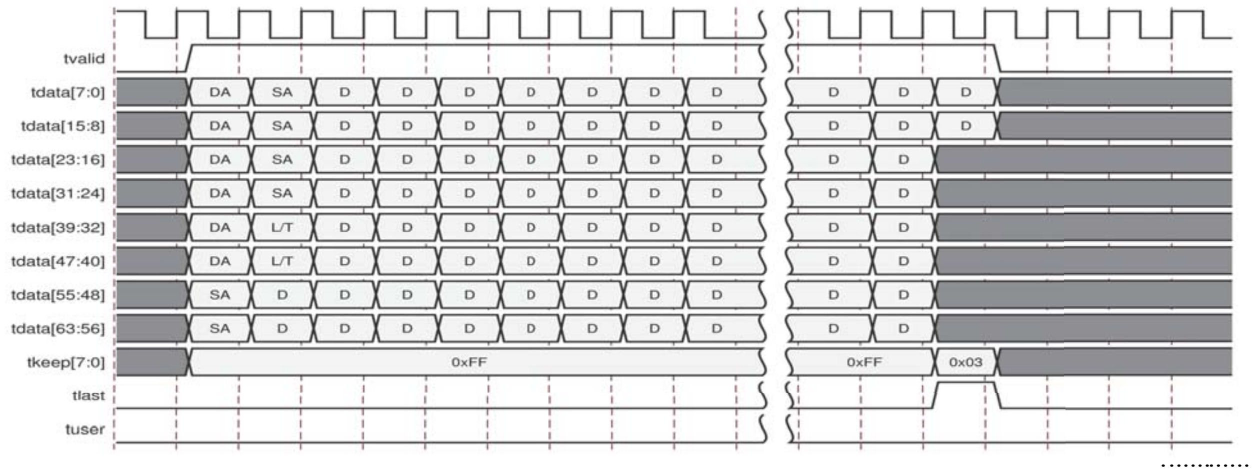


Figure 2-14: **Normal Frame Reception – 64 Bits**

Frame Reception with Errors

The case of an unsuccessful frame reception (for example, a runt frame or a frame with an incorrect FCS) is shown in Figure 2-11 for 32-bit and Figure 2-12 for 64-bit. In this case the bad frame is received and the signal `rx_axis_tuser` is asserted to the client at the end of the frame. It is then the responsibility of the client to drop the data already transferred for this frame.

The following conditions cause the assertion of `rx_axis_tlast` along with `rx_axis_tuser` = 1 signifying a bad frame:

- FCS errors occur
- Packets are shorter than 64 bytes (undersize or fragment frames)
- Frames of length greater than the maximum transmission unit (MTU) size programmed are received
- Any control frame that is received is not exactly the minimum frame length
- The XGMII data stream contains error codes

AXI4-Stream Control and Status Ports

Table 2-34: **AXI4-Stream Interface - TX path Control/Status Signals**

Name	Direction	Clock Domain	Description
ctl_tx_enable	Input	clk	TX Enable. When sampled as a 1, this signal is used to enable the transmission of data. When sampled as a 0, only IDLEs are transmitted by the core. This input should not be set to 1 until the receiver it is sending data to is fully synchronized and ready to receive data. (that is, the receiver on the link partner 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 core stops transmitting any more packets.
ctl_tx_custom_preamble_enable	Input	clk	When asserted, this signal enables the use of tx_preamblein as a custom preamble instead of inserting a standard preamble.
tx_preamblein[55:0]	Input	clk	This is the custom preamble which is a separate input port rather than being in-line with the data. It should be valid during the start of the packet.
ctl_tx_ipg_value[3:0]	Input	clk	The ctl_tx_ipg_value defines the target average minimum Inter Packet Gap (IPG, in bytes) inserted between packets. Typical value is 12. The ctl_tx_ipg_value can also be programmed to a value in the 0 to 7 range, but in that case, it is interpreted as meaning "minimal IPG", so only Terminate code word IPG is inserted; no Idles are ever added in that case and that produces an average IPG of around four bytes when random-size packets are transmitted. ***Note:*** This signal can be optionally present.
ctl_tx_fcs_ins_enable	Input	clk	Enable FCS insertion by the TX core. If set to 0, the core does not FCS to the packet. If set to 1, the core calculates and adds FCS to the packet. This input cannot be dynamically changed between packets
ctl_tx_send_lfi	Input	clk	Transmit Local Fault Indication (LFI) code word. Takes precedence over Remote Fault Indication (RFI).

Table 2-34: **AXI4-Stream Interface - TX path Control/Status Signals *(Cont'd)***

Name	Direction	Clock Domain	Description
ctl_tx_send_rfi	Input	clk	Transmit Remote Fault Indication (RFI) code word. If sampled as a 1, the TX path transmits only RFI code words. This input should be set to 1 until the RX path is fully synchronized and is ready to accept data from the link partner.
ctl_tx_send_idle	Input	clk	Transmit IDLE code words. If sampled as a 1, the TX path only transmits IDLE code words. This input should be set to 1 when the partner is sending RFI code words.
ctl_tx_ignore_fcs	Input	clk	Enable FCS error checking at the AXI4-Stream interface by the TX core. This input only has effect when ctl_tx_fcs_ins_enable is Low. If set to 0 and a packet with bad FCS is being transmitted, it is not binned as good. If set to 1, a packet with bad FCS is binned as good. The error is flagged on the signals stat_tx_bad_fcs and stomped_fcs and the packet is transmitted as it was received. Note: Statistics are reported as if there was no FCS error.
ctl_tx_parity_err_response	Input	clk	Parity error response by the TX Core. If this bit is set to 0, the core does not take any action if any parity errors are detected. If this bit is set to 1, the core stomps the outgoing FCS (i.e., bit-wise inverse) and asserts stat_tx_bad_fcs.

Table 2-35: **AXI4-Stream Interface - RX path Control/Status Signals**

Name	Direction	Clock Domain	Description
ctl_rx_enable	Input	clk	RX enable. For normal operation this input must be set to 1. When set 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 AXI4-Stream interface is idle.
ctl_rx_custom_preamble_enable	Input	clk	When asserted, this signal causes the side band of a packet presented on the AXI4-Stream to be the preamble as it appears on the line.
rx_preambleout[55:0]	Output	clk	This is the preamble, and now a separate output instead of inline with data.
ctl_rx_delete_fcs	Input	clk	Enable FCS removal by the RX core. If set to 0, the core does not remove the FCS of the incoming packet. If set to 1, the core deletes the FCS to the received packet. ***Note:*** FCS is not deleted for packets that are less than eight bytes. This input should only be changed while the corresponding reset input is asserted.
ctl_rx_ignore_fcs	Input	clk	Enable FCS error checking at the AXI4-Stream interface by the RX core. If set to 0, a packet received with an FCS error is indicated as an errored frame (rx_axis_tuser=1 when rx_axis_tlast=1) If set to 1, the core does not flag an FCS error at the AXI4-Stream Interface. ***Note:*** The statistics are reported as if the packet is good. The signal stat_rx_bad_fcs, however reports the error.
ctl_rx_max_packet_len[14:0]	Input	clk	Any packet longer than this value is considered to be oversized. If a packet has a size greater than this value, it is truncated to this value and the rx_axis_tuser signal is asserted along with the rx_axis_tlast signal. ctl_rx_max_packet_len[14] is reserved and must be set to 0.
ctl_rx_min_packet_len[7:0]	Input	clk	Any packet shorter than this value is considered to be undersized. If a packet has a size shorter than this value, the rx_axis_tuser signal is asserted along with the rx_axis_tlast signal. Packets less than four bytes are dropped.

Table 2-35: **AXI4-Stream Interface - RX path Control/Status Signals *(Cont'd)***

Name	Direction	Clock Domain	Description
ctl_rx_check_sfd	Input	clk	When asserted, this input causes the MAC to check the Start of Frame Delimiter (SFD) of the received frame.
ctl_rx_check_preamble	Input	clk	When asserted, this input causes the MAC to check the preamble of the received frame.
stat_rx_local_fault	Output	clk	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_remote_fault	Output	clk	Remote fault indication status. If this bit is sampled as 1, indicates a remote fault condition was detected. If this bit is sampled as 0, remote fault condition does not exist. This output is level sensitive.

Miscellaneous Status/Control Signals

Table 2-36 shows the miscellaneous status and control signals.

Table 2-36: **Miscellaneous Status/Control Signals**

Name	Direction	Clock Domain	Description
ctl_rx_process_lfi	Input	clk	When this input is set to 1, the RX core expects and processes LF control codes coming in from the transceiver. When set to 0, the RX core ignores LF control codes coming in from the transceiver.
stat_tx_gmii_fifo_unf(1)	Output	clk	TX FIFO underflow
stat_tx_gmii_fifo_ovf(1)	Output	clk	TX FIFO overflow

Notes:

1. Available only in 10G MAC-only variant.

Statistics Interface Ports

Table 2-37 and Table 2-38 show the statistics interface ports for the RX and TX paths respectively.

Table 2-37: **Statistics Interface Ports - RX**

Name	Direction	Clock Domain	Description
stat_rx_bad_code	Output	clk	Increment for 64B/66B code violations. This signal indicates that the RX PCS receive state machines is in the RX_E state as specified by IEEE Std 802.3. This output can be used to generate MDIO register as defined in Clause 45.
stat_rx_total_packets[1:0]	Output	clk	Increment for the total number of packets received.
stat_rx_total_good_packets	Output	clk	Increment for the total number of good packets received. This value is non-zero only when a packet is received completely and contains no errors.
stat_rx_total_bytes[3:0]	Output	clk	Increment for the total number of bytes received.
stat_rx_total_good_bytes[13:0]	Output	clk	Increment for the total number of good bytes received. This value is non-zero only when a packet is received completely and contains no errors.
stat_rx_packet_small	Output	clk	Increment for all packets that are less than 64 bytes long. Packets that are less than 4 bytes are dropped.
stat_rx_jabber	Output	clk	Increment for packets longer than ctl_rx_max_packet_len with bad FCS.
stat_rx_packet_large	Output	clk	Increment for all packets that are more than 9215 bytes long.
stat_rx_oversize	Output	clk	Increment for packets longer than ctl_rx_max_packet_len with good FCS.
stat_rx_undersize	Output	clk	Increment for packets shorter than ctl_rx_min_packet_len with good FCS
stat_rx_toolong	Output	clk	Increment for packets longer than ctl_rx_max_packet_len with good and bad FCS
stat_rx_fragment	Output	clk	Increment for packets shorter than ctl_rx_min_packet_len with bad FCS
stat_rx_packet_64_bytes	Output	clk	Increment for good and bad packets received that contain 64 bytes.

Table 2-37: **Statistics Interface Ports - RX *(Cont'd)***

Name	Direction	Clock Domain	Description
stat_rx_packet_65_127_bytes	Output	clk	Increment for good and bad packets received that contain between 65 and 127 bytes.
stat_rx_packet_128_255_bytes	Output	clk	Increment for good and bad packets received that contain between 128 and 255 bytes.
stat_rx_packet_256_511_bytes	Output	clk	Increment for good and bad packets received that contain between 256 and 511 bytes.
stat_rx_packet_512_1023_bytes	Output	clk	Increment for good and bad packets received that contain between 512 and 1023 bytes.
stat_rx_packet_1024_1518_bytes	Output	clk	Increment for good and bad packets received that contain between 1024 and 1518 bytes.
stat_rx_packet_1519_1522_bytes	Output	clk	Increment for good and bad packets received that contain between 1519 and 1522 bytes.
stat_rx_packet_1523_1548_bytes	Output	clk	Increment for good and bad packets received that contain between 1523 and 1548 bytes.
stat_rx_packet_1549_2047_bytes	Output	clk	Increment for good and bad packets received that contain between 1549 and 2047 bytes.
stat_rx_packet_2048_4095_bytes	Output	clk	Increment for good and bad packets received that contain between 2048 and 4095 bytes.
stat_rx_packet_4096_8191_bytes	Output	clk	Increment for good and bad packets received that contain between 4096 and 8191 bytes.
stat_rx_packet_8192_9215_bytes	Output	clk	Increment for good and bad packets received that contain between 8192 and 9215 bytes.

Table 2-37: **Statistics Interface Ports - RX *(Cont'd)***

Name	Direction	Clock Domain	Description
stat_rx_bad_fcs [1:0]	Output	clk	When this signal is positive, it indicates that the error detection logic has identified mismatches 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 is sent with rx_axis_tuser asserted during the last transfer (the cycle with rx_axis_tlast asserted), unless ctl_rx_ignore_fcs is asserted. This signal is asserted for one clock period for each CRC32 error detected.
stat_rx_packet_bad_fcs	Output	clk	Increment for packets between 64 and ctl_rx_max_packet_len bytes that have Frame Check Sequence (FCS) errors.
stat_rx_stomped_fcs [1:0]	Output	clk	Stomped FCS indicator. The value on this bus indicates the packets 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. Note that pulses can occur in back to back cycles.
stat_rx_unicast	Output	clk	Increment for good unicast packets
stat_rx_multicast	Output	clk	Increment for good multicast packets
stat_rx_broadcast	Output	clk	Increment for good broadcast packets
stat_rx_vlan	Output	clk	Increment for good 802.1Q tagged VLAN packets.
stat_rx_pause	Output	clk	Increment for 802.3x MAC Pause Packet with good FCS.
stat_rx_user_pause	Output	clk	Increment for priority based pause packets with good FCS.
stat_rx_inrangeerr	Output	clk	Increment for packets with Length field error but with good FCS.
stat_rx_bad_preamble	Output	clk	Increment for packets received with 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. ***Note:*** When an invalid preamble is detected, the stat_rx_bad_preamble signal is asserted regardless of the setting of the ctl_rx_check_preamble signal.

Table 2-37: **Statistics Interface Ports - RX *(Cont'd)***

Name	Direction	Clock Domain	Description
stat_rx_bad_sfd	Output	clk	Increment for packets received with bad SFD. This signal indicates if the Ethernet packet received was preceded by a valid SFD. A value of 1 indicates that an invalid SFD was received. ***Note:*** When an invalid SFD is detected, the stat_rx_bad_sfd signal is asserted regardless of the setting of the ctl_rx_check_sfd signal.
stat_rx_got_signal_os	Output	clk	Signal OS indication. If this bit is sampled as a 1, it indicates that a signal OS word was received. Note: Signal OS should not be received in an Ethernet network.
stat_rx_truncated	Output	clk	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.

Table 2-38: **Statistics Interface - TX Path**

Name	Direction	Clock Domain	Description
stat_tx_total_packets	Output	clk	Increment for total number of packets transmitted.
stat_tx_total_bytes[2:0]	Output	clk	Increment for total number of bytes transmitted.
stat_tx_total_good_packets	Output	clk	Increment for the total number of good packets transmitted.
stat_tx_total_good_bytes[13:0]	Output	clk	Increment for the total number of good bytes transmitted. This is signal is non-zero only when a packet is transmitted completely and contains no errors.
stat_tx_packet_64_bytes	Output	clk	Increment for good and bad packets transmitted that contain 64 bytes.
stat_tx_packet_65_127_bytes	Output	clk	Increment for good and bad packets transmitted that contain between 65 and 127 bytes.
stat_tx_packet_128_255_bytes	Output	clk	Increment for good and bad packets transmitted that contain between 128 and 255 bytes.

Table 2-38: **Statistics Interface - TX Path *(Cont'd)***

Name	Direction	Clock Domain	Description
stat_tx_packet_256_511_bytes	Output	clk	Increment for good and bad packets transmitted that contain between 256 and 511 bytes.
stat_tx_packet_512_1023_bytes	Output	clk	Increment for good and bad packets transmitted that contain between 512 and 1023 bytes.
stat_tx_packet_1024_1518_bytes	Output	clk	Increment for good and bad packets transmitted that contain between 1024 and 1518 bytes.
stat_tx_packet_1519_1522_bytes	Output	clk	Increment for good and bad packets transmitted that contain between 1519 and 1522 bytes.
stat_tx_packet_1523_1548_bytes	Output	clk	Increment for good and bad packets transmitted that contain between 1523 and 1548 bytes.
stat_tx_packet_1549_2047_bytes	Output	clk	Increment for good and bad packets transmitted that contain between 1549 and 2047 bytes.
stat_tx_packet_2048_4095_bytes	Output	clk	Increment for good and bad packets transmitted that contain between 2048 and 4095 bytes.
stat_tx_packet_4096_8191_bytes	Output	clk	Increment for good and bad packets transmitted that contain between 4096 and 8191 bytes.
stat_tx_packet_8192_9215_bytes	Output	clk	Increment for good and bad packets transmitted that contain between 8192 and 9215 bytes.
stat_tx_packet_small	Output	clk	Increment for all packets that are less than 64 bytes long.
stat_tx_packet_large	Output	clk	Increment for all packets that are more than 9215 bytes long.
stat_tx_unicast	Output	clk	Increment for good unicast packets.
stat_tx_multicast	Output	clk	Increment for good multicast packets.
stat_tx_broadcast	Output	clk	Increment for good broadcast packets.
stat_tx_vlan	Output	clk	Increment for good 802.1Q tagged VLAN packets.
stat_tx_pause	Output	clk	Increment for 802.3x MAC Pause Packet with good FCS
stat_tx_user_pause	Output	clk	Increment for Priority based pause packets with good FCS.
stat_tx_bad_fcs	Output	clk	Increment for packets greater than 64 bytes that have FCS errors.

Table 2-38: **Statistics Interface - TX Path *(Cont'd)***

Name	Direction	Clock Domain	Description
stat_tx_frame_error	Output	clk	Increment for packets with tx_axis_tuser set to indicate an End of Packet(EOP) abort.
stat_tx_local_fault	Output	clk	A value of 1 indicates the receive decoder state machine is in the TX_INIT state. This output is level sensitive.
stat_tx_bad_parity	Output	clk	Increment on any clock cycle where the user-generated parity is calculated as incorrect by the Tx parity checking logic.

Pause Interface

Table 2-39 through Table 2-41 show the Pause interface I/O ports.

Table 2-39: **Pause Interface - Control Ports**

Name	Direction	Clock Domain	Description
ctl_rx_pause_enable[8:0]	Input	clk	RX pause enable signal. This input is used to enable the processing of the pause quanta for the corresponding priority. Note: This signal only affects the RX user interface and not the pause processing logic
ctl_tx_pause_enable[8:0]	Input	clk	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.

Pause Interface – TX

Table 2-40: **Pause Interface - TX path**

Name	Direction	Clock Domain	Description
ctl_tx_pause_req[8:0]	Input	clk	If a bit of this bus is set to 1, the 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.
ctl_tx_resend_pause	Input	clk	Retransmit 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.
ctl_tx_pause_quanta[8:0][15:0]	Input	clk	These nine buses indicate the quanta to be transmitted for each of the eight priorities in priority based and global pause operations. The value for ctl_tx_pause_quanta[8] is used for global pause operation. All other values are used for priority based pause operation.
ctl_tx_pause_refresh_timer[8:0][15:0]	Input	clk	These nine buses set the retransmission time of pause packets for each of the eight priorities in priority based pause operation and the global pause operation. The values for ctl_tx_pause_refresh_timer[8] is used for global pause operation. All other values are used for priority pause operation.
ctl_tx_da_gpp[47:0]	Input	clk	Destination address for transmitting global pause packets.
ctl_tx_sa_gpp[47:0]	Input	clk	Source address for transmitting global pause packets.
ctl_tx_ethertype_gpp[15:0]	Input	clk	Ethertype for transmitting global pause packets.
ctl_tx_opcode_gpp[15:0]	Input	clk	Opcode for transmitting global pause packets.
ctl_tx_da_ppp[47:0]	Input	clk	Destination address for transmitting priority pause packets.
ctl_tx_sa_ppp[47:0]	Input	clk	Source address for transmitting priority pause packets.

Table 2-40: **Pause Interface - TX path *(Cont'd)***

Name	Direction	Clock Domain	Description
ctl_tx_ethertype_ppp[15:0]	Input	clk	Ethertype for transmitting priority pause packets.
ctl_tx_opcode_ppp[15:0]	Input	clk	Opcode for transmitting priority pause packets.
stat_tx_pause_valid[8:0]	Output	clk	If a bit of this bus is set to 1, the core has transmitted a pause packets. If bit [8] is set to 1, a global pause packet is transmitted. All other bits cause a priority pause packet to be transmitted.

Pause Interface – RX

Table 2-41: **Pause Interface - RX**

Name	Direction	Clock Domain	Description
ctl_rx_pause_ack[8:0]	Input	clk	Pause acknowledge signal. This bus is used to acknowledge the receipt of the pause frame from the user logic.
ctl_rx_check_ack	Input	clk	Wait for acknowledge. IF this input is set to 1, the core uses the ctl_rx_pause_ack[8:0] bus for pause processing. If this input is set to 0, ctl_rx_pause_ack[8:0] is not used.
ctl_rx_enable_gcp	Input	clk	A value of 1 enables global control packet processing.
ctl_rx_check_mcast_gcp	Input	clk	A value of 1 enables global control multicast destination address processing.
ctl_rx_check_ucast_gcp	Input	clk	A value of 1 enables global control unicast destination address processing.
ctl_rx_pause_da_ucast[47:0]	Input	clk	Unicast destination address for pause processing.
ctl_rx_check_sa_gcp	Input	clk	A value of 1 enables global control source address processing.
ctl_rx_pause_sa[47:0]	Input	clk	Source address for pause processing.
ctl_rx_check_etype_gcp	Input	clk	A value of 1 enables global control ethertype processing.
ctl_rx_etype_gcp[15:0]	Input	clk	Ethertype field for global control processing
ctl_rx_check_opcode_gcp	Input	clk	A value of 1 enables global control opcode processing.
ctl_rx_opcode_min_gcp[15:0]	Input	clk	Minimum global control opcode value
ctl_rx_opcode_max_gcp[15:0]	Input	clk	Maximum global control opcode value
ctl_rx_enable_pcp	Input	clk	A value of 1 enables priority control packet processing
ctl_rx_check_mcast_pcp	Input	clk	A value of 1 enables priority control multicast destination address processing
ctl_rx_check_ucast_pcp	Input	clk	A value of 1 enables priority control unicast destination address processing
ctl_rx_pause_da_mcast[47:0]	Input	clk	Multicast destination address for pause processing.
ctl_rx_check_sa_pcp	Input	clk	A value of 1 enables priority control source address processing
ctl_rx_check_etype_pcp	Input	clk	A value of 1 enables priority control ethertype processing
ctl_rx_etype_pcp[15:0]	Input	clk	Ethertype field for priority control processing

Table 2-41: **Pause Interface - RX *(Cont'd)***

Name	Direction	Clock Domain	Description
ctl_rx_check_opcode_pcp	Input	clk	A value of 1 enables priority control opcode processing
ctl_rx_opcode_min_pcp[15:0]	Input	clk	Minimum priority control opcode value
ctl_rx_opcode_max_pcp[15:0]	Input	clk	Maximum priority control opcode value
ctl_rx_enable_gpp	Input	clk	A value of 1 enables global pause packet processing
ctl_rx_check_mcast_gpp	Input	clk	A value of 1 enables global pause multicast destination address processing
ctl_rx_check_ucast_gpp	Input	clk	A value of 1 enables global pause unicast destination address processing
ctl_rx_check_sa_gpp	Input	clk	A value of 1 enables global pause source address processing.
ctl_rx_check_etype_gpp	Input	clk	A value of 1 enables global pause ethertype processing
ctl_rx_etype_gpp[15:0]	Input	clk	Ethertype field for global pause processing
ctl_rx_check_opcode_gpp	Input	clk	A value of 1 enables global pause opcode processing.
ctl_rx_opcode_gpp[15:0]	Input	clk	Global pause opcode value.
ctl_rx_enable_ppp	Input	clk	A value of 1 enables priority pause packet processing
ctl_rx_check_mcast_ppp	Input	clk	A value of 1 enables priority pause multicast destination address processing
ctl_rx_check_ucast_ppp	Input	clk	A value of 1 enables priority pause unicast destination address processing
ctl_rx_check_sa_ppp	Input	clk	A value of 1 enables priority pause source address processing
ctl_rx_check_etype_ppp	Input	clk	A value of 1 enables priority pause ethertype processing
ctl_rx_etype_ppp[15:0]	Input	clk	Ethertype field for priority pause processing
ctl_rx_check_opcode_ppp	Input	clk	A value of 1 enables priority pause opcode processing
ctl_rx_opcode_ppp[15:0]	Input	clk	Priority pause opcode value
ctl_rx_forward_control	Input	clk	A value of 1 indicates that the core forwards control packets. A value of 0 causes core to drop control packets.
stat_rx_pause_valid [8:0]	Output	clk	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 MAC Pause packet is received, bit [8] is set to 1.

Table 2-41: **Pause Interface - RX *(Cont'd)***

Name	Direction	Clock Domain	Description
stat_rx_pause_quanta[8:0] [15:0]	Output	clk	These nine buses indicate the quanta received for each of the eight priorities in priority based pause operation and global pause operation. If an 802.3x MAC Pause packet is received, the quanta is placed in value[8].
stat_rx_pause_req [8:0]	Output	clk	Pause request signal. When the RX receives a valid pause frame, it sets the corresponding bit of this bus to 1 and keep it at 1 until the pause packet has been processed.

Register Space

The 10/25 G Ethernet core can be optionally configured with AXI4-Lite registers to access the configuration and status signals.

AXI4-Lite Ports

Table 2-42 describes the port list for the AXI processor interface.

Table 2-42: **AXI Ports**

Signal	Direction	Description
s_axi_aclk	In	AXI4-Lite clock. Range between 10 MHz and 300 MHz
s_axi_aresetn	In	Asynchronous active-Low reset
s_axi_awaddr[31:0]	In	Write address Bus
s_axi_awvalid	In	Write address valid
s_axi_awready	Out	Write address acknowledge
s_axi_wdata[31:0]	In	Write data bus
s_axi_wstrb[3:0]	In	Strobe signal for the data bus byte lane
s_axi_wvalid	Out	Write data valid
s_axi_wready	Out	Write data acknowledge
s_axi_bresp[1:0]	Out	Write transaction response
s_axi_bvalid	Out	Write response valid
s_axi_bready	In	Write response acknowledge
s_axi_araddr[31:0]	In	Read address bus
s_axi_arvalid	In	Read address valid
s_axi_arready	Out	Read address acknowledge
s_axi_rdata[31:0]	Out	Read data output

Table 2-42: **AXI Ports *(Cont'd)***

Signal	Direction	Description
s_axi_rresp[1:0]	Out	Read data response
s_axi_rvalid	Out	Read data/response valid
s_axi_rready	In	Read data acknowledge
pm_tick	In	Top level signal to read statistics counters; requires MODE_REG[30] (i.e., tick_reg_mode_sel) be set to 0.

Additional information for the operation of the AXI4 bus is found in "Xilinx AXI Memory-Mapped Protocol Version 1.8".

As noted previously, the top level signal `pm_tick` can be used to read statistics counters instead of the configuration register TICK_REG. In this case, configuration register MODE_REG bit 30 (i.e., `tick_reg_mode_sel`) should be set to 0. If `tick_reg_mode_sel` set to 1, `tick_reg` is used to read the statistics counters.

Configuration Register Map

The configuration space provides software with the ability to configure the IP core for various use cases. Certain features are optional and the assigned register might not exist in a particular variant, in which case the applicable registers are considered RESERVED.

In order for the programmed configurations to take effect, it is necessary to issue `s_axi_aresetn`, which is active-Low.

Table 2-43: **Configuration Register Map**

Hex Address	Register Name/Link to Description	Notes
0x0000	GT_RESET_REG: 0000	
0x0004	RESET_REG: 0004	
0x0008	MODE_REG: 0008	
0x000C	CONFIGURATION_TX_REG1: 000C	
0x0014	CONFIGURATION_RX_REG1: 0014	
0x0018	CONFIGURATION_RX_MTU: 0018	Only in MAC+PCS variant and MAC-only variants
0x001C	CONFIGURATION_VL_LENGTH_REG: 001C	
0x0020	TICK_REG: 0020	
0x0024	CONFIGURATION_REVISION_REG: 0024	
0x0028	CONFIGURATION_TX_TEST_PAT_SEED_A_LSB: 0028	Only in MAC+PCS and PCS-only variants
0x002C	CONFIGURATION_TX_TEST_PAT_SEED_A_MSB: 002C	Only in MAC+PCS and PCS-only variants

Table 2-43: **Configuration Register Map *(Cont'd)***

Hex Address	Register Name/Link to Description	Notes
0x0030	CONFIGURATION_TX_TEST_PAT_SEED_B_LSB: 0030	Only in MAC+PCS and PCS-only variants
0x0034	CONFIGURATION_TX_TEST_PAT_SEED_B_MSB: 0034	Only in MAC+PCS and PCS-only variants
0x0038	CONFIGURATION_1588_REG: 0038	Only in MAC+PCS variant
0x0040	CONFIGURATION_TX_FLOW_CONTROL_REG1: 0040	Only in MAC+PCS and MAC-only variants
0x0044	CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG1: 0044	Only in MAC+PCS and MAC-only variants
0x0048	CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG2: 0048	Only in MAC+PCS and MAC-only variants
0x004C	CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG3: 004C	Only in MAC+PCS and MAC-only variants
0x0050	CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG4: 0050	Only in MAC+PCS and MAC-only variants
0x0054	CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG5: 0054	Only in MAC+PCS and MAC-only variants
0x0058	CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG1: 0058	Only in MAC+PCS and MAC-only variants
0x005C	CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG2: 005C	Only in MAC+PCS and MAC-only variants
0x0060	CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG3: 0060	Only in MAC+PCS and MAC-only variants
0x0064	CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG4: 0064	Only in MAC+PCS and MAC-only variants
0x0068	CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG5: 0068	Only in MAC+PCS and MAC-only variants
0x006C	CONFIGURATION_TX_FLOW_CONTROL_PPP_ETYPE_OP_REG: 006C	Only in MAC+PCS and MAC-only variants
0x0070	CONFIGURATION_TX_FLOW_CONTROL_GPP_ETYPE_OP_REG: 0070	Only in MAC+PCS and MAC-only variants
0x0074	CONFIGURATION_TX_FLOW_CONTROL_GPP_DA_REG_LSB: 0074	Only in MAC+PCS and MAC-only variants
0x0078	CONFIGURATION_TX_FLOW_CONTROL_GPP_DA_REG_MSB: 0078	Only in MAC+PCS and MAC-only variant
0x007C	CONFIGURATION_TX_FLOW_CONTROL_GPP_SA_REG_LSB: 007C	Only in MAC+PCS and MAC-only variants
0x0080	CONFIGURATION_TX_FLOW_CONTROL_GPP_SA_REG_MSB: 0080	Only in MAC+PCS and MAC-only variants
0x0084	CONFIGURATION_TX_FLOW_CONTROL_PPP_DA_REG_LSB: 0084	Only in MAC+PCS and MAC-only variants

Table 2-43: **Configuration Register Map *(Cont'd)***

Hex Address	Register Name/Link to Description	Notes
0x0088	CONFIGURATION_TX_FLOW_CONTROL_PPP_DA_REG_MSB: 0088	Only in MAC+PCS and MAC-only variants
0x008C	CONFIGURATION_TX_FLOW_CONTROL_PPP_SA_REG_LSB: 008C	Only in MAC+PCS and MAC-only variants
0x0090	CONFIGURATION_TX_FLOW_CONTROL_PPP_SA_REG_MSB: 0090	Only in MAC+PCS and MAC-only variants
0x0094	CONFIGURATION_RX_FLOW_CONTROL_REG1: 0094	Only in MAC+PCS and MAC-only variants
0x0098	CONFIGURATION_RX_FLOW_CONTROL_REG2: 0098	Only in MAC+PCS and MAC-only variants
0x009C	CONFIGURATION_RX_FLOW_CONTROL_PPP_ETYPE_OP_REG: 009C	Only in MAC+PCS and MAC-only variants
0x00A0	CONFIGURATION_RX_FLOW_CONTROL_GPP_ETYPE_OP_REG: 00A0	Only in MAC+PCS and MAC-only variants
0x00A4	CONFIGURATION_RX_FLOW_CONTROL_GCP_PCP_TYPE_REG: 00A4	Only in MAC+PCS and MAC-only variants
0x00A8	CONFIGURATION_RX_FLOW_CONTROL_PCP_OP_REG: 00A8	Only in MAC+PCS and MAC-only variants
0x00AC	CONFIGURATION_RX_FLOW_CONTROL_GCP_OP_REG: 00AC	Only in MAC+PCS and MAC-only variants
0x00B0	CONFIGURATION_RX_FLOW_CONTROL_DA_REG1_LSB: 00B0	Only in MAC+PCS and MAC-only variants
0x00B4	CONFIGURATION_RX_FLOW_CONTROL_DA_REG1_MSB: 00B4	Only in MAC+PCS and MAC-only variants
0x00B8	CONFIGURATION_RX_FLOW_CONTROL_DA_REG2_LSB: 00B8	Only in MAC+PCS and MAC-only variants
0x00BC	CONFIGURATION_RX_FLOW_CONTROL_DA_REG2_MSB: 00BC	Only in MAC+PCS and MAC-only variants
0x00C0	CONFIGURATION_RX_FLOW_CONTROL_SA_REG1_LSB: 00C0	Only in MAC+PCS and MAC-only variants
0x00C4	CONFIGURATION_RX_FLOW_CONTROL_SA_REG1_MSB: 00C4	Only in MAC+PCS and MAC-only variants
0x00D0	CONFIGURATION_RSFEC_REG: 00D0	Only in MAC+PCS and PCS-only variants
0x00D4	CONFIGURATION_FEC_REG: 00D4	Only in MAC+PCS and PCS-only variants
0x00E0	CONFIGURATION_AN_CONTROL_REG1: 00E0	Only in MAC+PCS and PCS-only variants
0x00E4	CONFIGURATION_AN_CONTROL_REG2: 00E4	Only in MAC+PCS and PCS-only variants

Table 2-43: **Configuration Register Map *(Cont'd)***

Hex Address	Register Name/Link to Description	Notes
0x00F8	CONFIGURATION_AN_ABILITY: 00F8	Only in MAC+PCS and PCS-only variants
0x0100	CONFIGURATION_LT_CONTROL_REG1: 0100	Only in MAC+PCS and PCS-only variants
0x0104	CONFIGURATION_LT_TRAINED_REG: 0104	Only in MAC+PCS and PCS-only variants
0x0108	CONFIGURATION_LT_PRESET_REG: 0108	Only in MAC+PCS and PCS-only variants
0x010C	CONFIGURATION_LT_INIT_REG: 010C	Only in MAC+PCS and PCS-only variants
0x0110	CONFIGURATION_LT_SEED_REG0: 0110	Only in MAC+PCS and PCS-only variants
0x0130	CONFIGURATION_LT_COEFFICIENT_REG0: 0130	Only in MAC+PCS and PCS-only variants
0x0134	USER_REG_0: 0134	
0x0138	SWITCH_CORE_SPEED_REG: 0138	
0x013C	CONFIGURATION_1588_32BIT_REG: 0x013C	Only for MAC+PCS/PMA 32-bit variant 1588 variants
0x0140	TX_CONFIGURATION_1588_REG: 0x140	Only for MAC+PCS/PMA 32-bit variant 1588 variants
0x0144	RX_CONFIGURATION_1588_REG: 0x144	Only for MAC+PCS/PMA 32-bit variant 1588 variants
0x019C	CONFIGURATION_TSN_REG: 0x019C	Only when Preemption Feature is enabled.

Status Register Map

The status registers provide an indication of the health of the system. These registers are Read-Only and a read operation clears the register.

Status registers are cleared according to the following conditions:

- Applying `s_axi_aresetn` clears both TX and RX status registers
- When a particular status register is read (clear on read)
- Applying `rx_reset` clears the RX status registers only
- Applying `tx_reset` clears the TX status registers only

Table 2-44: **Status Register Map**

Hex Address	Register Name/Link to Description	Notes
0x0400	STAT_TX_STATUS_REG1: 0400	
0x0404	STAT_RX_STATUS_REG1: 0404	
0x0408	STAT_STATUS_REG1: 0408	Only in MAC+PCS and PCS-only variants
0x040C	STAT_RX_BLOCK_LOCK_REG: 040C	Only in MAC+PCS and PCS-only variants
0x043C	STAT_RX_RSFEC_STATUS_REG: 043C	Only in MAC+PCS and PCS-only variants
0x0448	STAT_RX_FEC_STATUS_REG: 0448	Only in MAC+PCS and PCS-only variants
0x044C	STAT_TX_RSFEC_STATUS_REG: 044C	Only in MAC+PCS and PCS-only variants
0x0450	STAT_TX_FLOW_CONTROL_REG1: 0450	Only in MAC+PCS variant and MAC-only variants
0x0454	STAT_RX_FLOW_CONTROL_REG1: 0454	Only in MAC+PCS variant and MAC-only variants
0x0458	STAT_AN_STATUS: 0458	Only in MAC+PCS and PCS-only variants
0x045C	STAT_AN_ABILITY: 045C	Only in MAC+PCS and PCS-only variants
0x0460	STAT_AN_LINK_CTL: 0460	Only in MAC+PCS and PCS-only variants
0x0464	STAT_LT_STATUS_REG1: 0464	Only in MAC+PCS and PCS-only variants
0x0468	STAT_LT_STATUS_REG2: 0468	Only in MAC+PCS and PCS-only variants
0x046C	STAT_LT_STATUS_REG3: 046C	Only in MAC+PCS and PCS-only variants
0x0470	STAT_LT_STATUS_REG4: 0470	Only in MAC+PCS and PCS-only variants
0x0474	STAT_LT_COEFFICIENT0_REG: 0474	Only in MAC+PCS and PCS-only variants
0x0494	STAT_RX_VALID_CTRL_CODE: 0494	Only in MAC+PCS and PCS-only variants
0x0498	STAT_CORE_SPEED_REG: 0498	
0x049C	STAT_TSN_REG: 0x049C	Only when 802.1cm Preemption feature enabled

Statistics Counters

The statistics counters provide histograms of the classification of traffic and error counts. These counters can be read either by a 1 on `pm_tick` or by writing a 1 to TICK_REG, depending on the value of MODE_REG[30] (i.e. `tick_reg_mode_sel`). `pm_tick` will be used when MODE_REG[30] = 0 and TICK_REG will be used when MODE_REG[30] = 1 (1 = default).

The counters employ 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 clear the accumulators. The STAT_*_MSB/LSB registers can then be read. In this way all values stored in the statistics counters represent a snap-shot over the same time interval.

Note: These readable STAT_*_MSB/LSB registers are not resettable. This can result in unknown data being present after reset but prior to a TICK_REG write.

The STAT_CYCLE_COUNT_MSB/LSB register contains a count of the number of RX core clock cycles between TICK_REG writes. This allows for easy time-interval based statistics.

Statistic counter registers are cleared according to the following conditions:

- Applying `s_axi_aresetn` clears both TX and RX statistics counter registers
- Applying PM Tick clears both TX and RX statistics counter registers
- Applying `rx_reset` clears the RX statistics counter registers only
- Applying `tx_reset` clears the TX statistics counter registers only

Table 2-45: **Statistics Counters**

Hex Address	Register Name/Link to Description	Notes
0x0500	STATUS_CYCLE_COUNT_LSB: 0500	
0x0504	STATUS_CYCLE_COUNT_MSB: 0504	
0x0648	STAT_RX_FRAMING_ERR_LSB: 0648	Only in MAC+PCS and PCS-only variants
0x064C	STAT_RX_FRAMING_ERR_MSB: 064C	Only in MAC+PCS and PCS-only variants
0x0660	STAT_RX_BAD_CODE_LSB: 0660	
0x0664	STAT_RX_BAD_CODE_MSB: 0664	
0x0668	STAT_RX_ERROR_LSB: 0668	Only in PCS variant
0x066C	STAT_RX_ERROR_MSB: 066C	Only in PCS variant
0x0670	STAT_RX_RSFEC_CORRECTED_CW_INC_LSB: 0670	Only in MAC+PCS and PCS-only variants
0x0674	STAT_RX_RSFEC_CORRECTED_CW_INC_MSB: 0674	Only in MAC+PCS and PCS-only variants

Table 2-45: **Statistics Counters *(Cont'd)***

Hex Address	Register Name/Link to Description	Notes
0x0678	STAT_RX_RSFEC_UNCORRECTED_CW_INC_LSB: 0678	Only in MAC+PCS and PCS-only variants
0x067C	STAT_RX_RSFEC_UNCORRECTED_CW_INC_MSB: 067C	Only in MAC+PCS and PCS-only variants
0x0680	STAT_RX_RSFEC_ERR_COUNT0_INC_LSB: 0680	Only in MAC+PCS and PCS-only variants
0x0684	STAT_RX_RSFEC_ERR_COUNT0_INC_MSB: 0684	Only in MAC+PCS and PCS-only variants
0x06A0	STAT_TX_FRAME_ERROR_LSB: 06A0	Only in MAC+ PCS and MAC-only variants
0x06A4	STAT_TX_FRAME_ERROR_MSB: 06A4	Only in MAC+ PCS and MAC-only variants
0x0700	STAT_TX_TOTAL_PACKETS_LSB: 0700	Only in MAC+ PCS and MAC-only variants
0x0704	STAT_TX_TOTAL_PACKETS_MSB: 0704	Only in MAC+ PCS and MAC-only variants
0x0708	STAT_TX_TOTAL_GOOD_PACKETS_LSB: 0708	Only in MAC+ PCS and MAC-only variants
0x070C	STAT_TX_TOTAL_GOOD_PACKETS_MSB: 070C	Only in MAC+ PCS and MAC-only variants
0x0710	STAT_TX_TOTAL_BYTES_LSB: 0710	Only in MAC+ PCS and MAC-only variants
0x0714	STAT_TX_TOTAL_BYTES_MSB: 0714	Only in MAC+ PCS and MAC-only variants
0x0718	STAT_TX_TOTAL_GOOD_BYTES_LSB: 0718	Only in MAC+ PCS and MAC-only variants
0x071C	STAT_TX_TOTAL_GOOD_BYTES_MSB: 071C	Only in MAC+ PCS and MAC-only variants
0x0720	STAT_TX_PACKET_64_BYTES_LSB: 0720	Only in MAC+ PCS and MAC-only variants
0x0724	STAT_TX_PACKET_64_BYTES_MSB: 0724	Only in MAC+ PCS and MAC-only variants
0x0728	STAT_TX_PACKET_65_127_BYTES_LSB: 0728	Only in MAC+ PCS and MAC-only variants
0x072C	STAT_TX_PACKET_65_127_BYTES_MSB: 072C	Only in MAC+ PCS and MAC-only variants
0x0730	STAT_TX_PACKET_128_255_BYTES_LSB: 0730	Only in MAC+ PCS and MAC-only variants
0x0734	STAT_TX_PACKET_128_255_BYTES_MSB: 0734	Only in MAC+ PCS and MAC-only variants
0x0738	STAT_TX_PACKET_256_511_BYTES_LSB: 0738	Only in MAC+ PCS and MAC-only variants

Table 2-45: **Statistics Counters *(Cont'd)***

Hex Address	Register Name/Link to Description	Notes
0x073C	STAT_TX_PACKET_256_511_BYTES_MSB: 073C	Only in MAC+ PCS and MAC-only variants
0x0740	STAT_TX_PACKET_512_1023_BYTES_LSB: 0740	Only in MAC+ PCS and MAC-only variants
0x0744	STAT_TX_PACKET_512_1023_BYTES_MSB: 0744	Only in MAC+ PCS and MAC-only variants
0x0748	STAT_TX_PACKET_1024_1518_BYTES_LSB: 0748	Only in MAC+ PCS and MAC-only variants
0x074C	STAT_TX_PACKET_1024_1518_BYTES_MSB: 074C	Only in MAC+ PCS and MAC-only variants
0x0750	STAT_TX_PACKET_1519_1522_BYTES_LSB: 0750	Only in MAC+ PCS and MAC-only variants
0x0754	STAT_TX_PACKET_1519_1522_BYTES_MSB: 0754	Only in MAC+ PCS and MAC-only variants
0x0758	STAT_TX_PACKET_1523_1548_BYTES_LSB: 0758	Only in MAC+ PCS and MAC-only variants
0x075C	STAT_TX_PACKET_1523_1548_BYTES_MSB: 075C	Only in MAC+ PCS and MAC-only variants
0x0760	STAT_TX_PACKET_1549_2047_BYTES_LSB: 0760	Only in MAC+ PCS and MAC-only variants
0x0764	STAT_TX_PACKET_1549_2047_BYTES_MSB: 0764	Only in MAC+ PCS and MAC-only variants
0x0768	STAT_TX_PACKET_2048_4095_BYTES_LSB: 0768	Only in MAC+ PCS and MAC-only variants
0x076C	STAT_TX_PACKET_2048_4095_BYTES_MSB: 076C	Only in MAC+ PCS and MAC-only variants
0x0770	STAT_TX_PACKET_4096_8191_BYTES_LSB: 0770	Only in MAC+ PCS and MAC-only variants
0x0774	STAT_TX_PACKET_4096_8191_BYTES_MSB: 0774	Only in MAC+ PCS and MAC-only variants
0x0778	STAT_TX_PACKET_8192_9215_BYTES_LSB: 0778	Only in MAC+ PCS and MAC-only variants
0x077C	STAT_TX_PACKET_8192_9215_BYTES_MSB: 077C	Only in MAC+ PCS and MAC-only variants
0x0780	STAT_TX_PACKET_LARGE_LSB: 0780	Only in MAC+ PCS and MAC-only variants
0x0784	STAT_TX_PACKET_LARGE_MSB: 0784	Only in MAC+ PCS and MAC-only variants
0x0788	STAT_TX_PACKET_SMALL_LSB: 0788	Only in MAC+ PCS and MAC-only variants
0x078C	STAT_TX_PACKET_SMALL_MSB: 078C	Only in MAC+ PCS and MAC-only variants

Table 2-45: **Statistics Counters *(Cont'd)***

Hex Address	Register Name/Link to Description	Notes
0x07B8	STAT_TX_BAD_FCS_LSB: 07B8	Only in MAC+ PCS and MAC-only variants
0x07BC	STAT_TX_BAD_FCS_MSB: 07BC	Only in MAC+ PCS and MAC-only variants
0x07D0	STAT_TX_UNICAST_LSB: 07D0	Only in MAC+ PCS and MAC-only variants
0x07D4	STAT_TX_UNICAST_MSB: 07D4	Only in MAC+ PCS and MAC-only variants
0x07D8	STAT_TX_MULTICAST_LSB: 07D8	Only in MAC+ PCS and MAC-only variants
0x07DC	STAT_TX_MULTICAST_MSB: 07DC	Only in MAC+ PCS and MAC-only variants
0x07E0	STAT_TX_BROADCAST_LSB: 07E0	Only in MAC+ PCS and MAC-only variants
0x07E4	STAT_TX_BROADCAST_MSB: 07E4	Only in MAC+ PCS and MAC-only variants
0x07E8	STAT_TX_VLAN_LSB: 07E8	Only in MAC+ PCS and MAC-only variants
0x07EC	STAT_TX_VLAN_MSB: 07EC	Only in MAC+ PCS and MAC-only variants
0x07F0	STAT_TX_PAUSE_LSB: 07F0	Only in MAC+ PCS and MAC-only variants
0x07F4	STAT_TX_PAUSE_MSB: 07F4	Only in MAC+ PCS and MAC-only variants
0x07F8	STAT_TX_USER_PAUSE_LSB: 07F8	Only in MAC+ PCS and MAC-only variants
0x07FC	STAT_TX_USER_PAUSE_MSB: 07FC	Only in MAC+ PCS and MAC-only variants
0x0808	STAT_RX_TOTAL_PACKETS_LSB: 0808	Only in MAC+ PCS and MAC-only variants
0x080C	STAT_RX_TOTAL_PACKETS_MSB: 080C	Only in MAC+ PCS and MAC-only variants
0x0810	STAT_RX_TOTAL_GOOD_PACKETS_LSB: 0810	Only in MAC+ PCS and MAC-only variants
0x0814	STAT_RX_TOTAL_GOOD_PACKETS_MSB: 0814	Only in MAC+ PCS and MAC-only variants
0x0818	STAT_RX_TOTAL_BYTES_LSB: 0818	Only in MAC+ PCS and MAC-only variants
0x081C	STAT_RX_TOTAL_BYTES_MSB: 081C	Only in MAC+ PCS and MAC-only variants
0x0820	STAT_RX_TOTAL_GOOD_BYTES_LSB: 0820	Only in MAC+ PCS and MAC-only variants

Table 2-45: **Statistics Counters *(Cont'd)***

Hex Address	Register Name/Link to Description	Notes
0x0824	STAT_RX_TOTAL_GOOD_BYTES_MSB: 0824	Only in MAC+ PCS and MAC-only variants
0x0828	STAT_RX_PACKET_64_BYTES_LSB: 0828	Only in MAC+ PCS and MAC-only variants
0x082C	STAT_RX_PACKET_64_BYTES_MSB: 082C	Only in MAC+ PCS and MAC-only variants
0x0830	STAT_RX_PACKET_65_127_BYTES_LSB: 0830	Only in MAC+ PCS and MAC-only variants
0x0834	STAT_RX_PACKET_65_127_BYTES_MSB: 0834	Only in MAC+ PCS and MAC-only variants
0x0838	STAT_RX_PACKET_128_255_BYTES_LSB: 0838	Only in MAC+ PCS and MAC-only variants
0x083C	STAT_RX_PACKET_128_255_BYTES_MSB: 083C	Only in MAC+ PCS and MAC-only variants
0x0840	STAT_RX_PACKET_256_511_BYTES_LSB: 0840	Only in MAC+ PCS and MAC-only variants
0x0844	STAT_RX_PACKET_256_511_BYTES_MSB: 0844	Only in MAC+ PCS and MAC-only variants
0x0848	STAT_RX_PACKET_512_1023_BYTES_LSB: 0848	Only in MAC+ PCS and MAC-only variants
0x084C	STAT_RX_PACKET_512_1023_BYTES_MSB: 084C	Only in MAC+ PCS and MAC-only variants
0x0850	STAT_RX_PACKET_1024_1518_BYTES_LSB: 0850	Only in MAC+ PCS and MAC-only variants
0x0854	STAT_RX_PACKET_1024_1518_BYTES_MSB: 0854	Only in MAC+ PCS and MAC-only variants
0x0858	STAT_RX_PACKET_1519_1522_BYTES_LSB: 0858	Only in MAC+ PCS and MAC-only variants
0x085C	STAT_RX_PACKET_1519_1522_BYTES_MSB: 085C	Only in MAC+ PCS and MAC-only variants
0x0860	STAT_RX_PACKET_1523_1548_BYTES_LSB: 0860	Only in MAC+ PCS and MAC-only variants
0x0864	STAT_RX_PACKET_1523_1548_BYTES_MSB: 0864	Only in MAC+ PCS and MAC-only variants
0x0868	STAT_RX_PACKET_1549_2047_BYTES_LSB: 0868	Only in MAC+ PCS and MAC-only variants
0x086C	STAT_RX_PACKET_1549_2047_BYTES_MSB: 086C	Only in MAC+ PCS and MAC-only variants
0x0870	STAT_RX_PACKET_2048_4095_BYTES_LSB: 0870	Only in MAC+ PCS and MAC-only variants
0x0874	STAT_RX_PACKET_2048_4095_BYTES_MSB: 0874	Only in MAC+ PCS and MAC-only variants

Table 2-45: **Statistics Counters *(Cont'd)***

Hex Address	Register Name/Link to Description	Notes
0x0878	STAT_RX_PACKET_4096_8191_BYTES_LSB: 0878	Only in MAC+ PCS and MAC-only variants
0x087C	STAT_RX_PACKET_4096_8191_BYTES_MSB: 087C	Only in MAC+ PCS and MAC-only variants
0x0880	STAT_RX_PACKET_8192_9215_BYTES_LSB: 0880	Only in MAC+ PCS and MAC-only variants
0x0884	STAT_RX_PACKET_8192_9215_BYTES_MSB: 0884	Only in MAC+ PCS and MAC-only variants
0x0888	STAT_RX_PACKET_LARGE_LSB: 0888	Only in MAC+ PCS and MAC-only variants
0x088C	STAT_RX_PACKET_LARGE_MSB: 088C	Only in MAC+ PCS and MAC-only variants
0x0890	STAT_RX_PACKET_SMALL_LSB: 0890	Only in MAC+ PCS and MAC-only variants
0x0894	STAT_RX_PACKET_SMALL_MSB: 0894	Only in MAC+ PCS and MAC-only variants
0x0898	STAT_RX_UNDERSIZE_LSB: 0898	Only in MAC+ PCS and MAC-only variants
0x089C	STAT_RX_UNDERSIZE_MSB: 089C	Only in MAC+ PCS and MAC-only variants
0x08A0	STAT_RX_FRAGMENT_LSB: 08A0	Only in MAC+ PCS and MAC-only variants
0x08A4	STAT_RX_FRAGMENT_MSB: 08A4	Only in MAC+ PCS and MAC-only variants
0x08A8	STAT_RX_OVERSIZE_LSB: 08A8	Only in MAC+ PCS and MAC-only variants
0x08AC	STAT_RX_OVERSIZE_MSB: 08AC	Only in MAC+ PCS and MAC-only variants
0x08B0	STAT_RX_TOOLONG_LSB: 08B0	Only in MAC+ PCS and MAC-only variants
0x08B4	STAT_RX_TOOLONG_MSB: 08B4	Only in MAC+ PCS and MAC-only variants
0x08B8	STAT_RX_JABBER_LSB: 08B8	Only in MAC+ PCS and MAC-only variants
0x08BC	STAT_RX_JABBER_MSB: 08BC	Only in MAC+ PCS and MAC-only variants
0x08C0	STAT_RX_BAD_FCS_LSB: 08C0	Only in MAC+ PCS and MAC-only variants
0x08C4	STAT_RX_BAD_FCS_MSB: 08C4	Only in MAC+ PCS and MAC-only variants
0x08C8	STAT_RX_PACKET_BAD_FCS_LSB: 08C8	Only in MAC+ PCS and MAC-only variants

Table 2-45: **Statistics Counters *(Cont'd)***

Hex Address	Register Name/Link to Description	Notes
0x08CC	STAT_RX_PACKET_BAD_FCS_MSB: 08CC	Only in MAC+ PCS and MAC-only variants
0x08D0	STAT_RX_STOMPED_FCS_LSB: 08D0	Only in MAC+ PCS and MAC-only variants
0x08D4	STAT_RX_STOMPED_FCS_MSB: 08D4	Only in MAC+ PCS and MAC-only variants
0x08D8	STAT_RX_UNICAST_LSB: 08D8	Only in MAC+ PCS and MAC-only variants
0x08DC	STAT_RX_UNICAST_MSB: 08DC	Only in MAC+ PCS and MAC-only variants
0x08E0	STAT_RX_MULTICAST_LSB: 08E0	Only in MAC+ PCS and MAC-only variants
0x08E4	STAT_RX_MULTICAST_MSB: 08E4	Only in MAC+ PCS and MAC-only variants
0x08E8	STAT_RX_BROADCAST_LSB: 08E8	Only in MAC+ PCS and MAC-only variants
0x08EC	STAT_RX_BROADCAST_MSB: 08EC	Only in MAC+ PCS and MAC-only variants
0x08F0	STAT_RX_VLAN_LSB: 08F0	Only in MAC+ PCS and MAC-only variants
0x08F4	STAT_RX_VLAN_MSB: 08F4	Only in MAC+ PCS and MAC-only variants
0x08F8	STAT_RX_PAUSE_LSB: 08F8	Only in MAC+ PCS and MAC-only variants
0x08FC	STAT_RX_PAUSE_MSB: 08FC	Only in MAC+ PCS and MAC-only variants
0x0900	STAT_RX_USER_PAUSE_LSB: 0900	Only in MAC+ PCS and MAC-only variants
0x0904	STAT_RX_USER_PAUSE_MSB: 0904	Only in MAC+ PCS and MAC-only variants
0x0908	STAT_RX_INRANGEERR_LSB: 0908	Only in MAC+ PCS and MAC-only variants
0x090C	STAT_RX_INRANGEERR_MSB: 090C	Only in MAC+ PCS and MAC-only variants
0x0910	STAT_RX_TRUNCATED_LSB: 0910	Only in MAC+ PCS and MAC-only variants
0x0914	STAT_RX_TRUNCATED_MSB: 0914	Only in MAC+ PCS and MAC-only variants
0x0918	STAT_RX_TEST_PATTERN_MISMATCH_LSB: 0918	Only in MAC+PCS variant
0x091C	STAT_RX_TEST_PATTERN_MISMATCH_MSB: 091C	Only in MAC+PCS variant
0x0920	STAT_FEC_INC_CORRECT_COUNT_LSB: 0920	Only in MAC+PCS and PCS-only variants

Table 2-45: **Statistics Counters *(Cont'd)***

Hex Address	Register Name/Link to Description	Notes
0x0924	STAT_FEC_INC_CORRECT_COUNT_MSB: 0924	Only in MAC+PCS and PCS-only variants
0x0928	STAT_FEC_INC_CANT_CORRECT_COUNT_LSB: 0928	Only in MAC+PCS and PCS-only variants
0x092C	STAT_FEC_INC_CANT_CORRECT_COUNT_MSB: 092C	Only in MAC+PCS and PCS-only variants
0x0980	STAT_TX_MM_STATUS_LSB: 0x0980	MAC+PCS with TSN
0x0984	STAT_TX_MM_STATUS_MSB: 0x0984	MAC+PCS with TSN
0x0988	STAT_TX_MM_FRAGMENT_LSB: 0x0988	MAC+PCS with TSN
0x098C	STAT_TX_MM_FRAGMENT_MSB: 0x098C	MAC+PCS with TSN
0x0990	STAT_TX_MM_HOLD_LSB: 0x0990	MAC+PCS with TSN
0x0994	STAT_TX_MM_HOLD_MSB: 0x0994	MAC+PCS with TSN
0x0998	STAT_RX_MM_ASSEMBLY_ERROR_LSB: 0x0998	MAC+PCS with TSN
0x099C	STAT_RX_MM_ASSEMBLY_ERROR_MSB: 0x099C	MAC+PCS with TSN
0x09A0	STAT_RX_MM_FRAME_SMD_ERROR_LSB: 0x09A0	MAC+PCS with TSN
0x09A4	STAT_RX_MM_FRAME_SMD_ERROR_MSB: 0x09A4	MAC+PCS with TSN
0x09A8	STAT_RX_MM_FRAME_ASSEMBLY_OK_LSB: 0x09A8	MAC+PCS with TSN
0x09AC	STAT_RX_MM_FRAME_ASSEMBLY_OK_MSB: 0x09AC	MAC+PCS with TSN
0x09B0	STAT_RX_MM_FRAGMENT_LSB: 0x09B0	MAC+PCS with TSN
0x09B4	STAT_RX_MM_FRAGMENT_MSB: 0x09B4	MAC+PCS with TSN

Register Descriptions

This section contains descriptions of the configuration registers. In the cases where the features described in the bit fields are not present in the IP core, the bit field reverts to RESERVED.

Configuration Registers

The following tables define the bit assignments for the configuration registers.

Registers or bit fields within registers can be accessed for Read-Write (RW), Write-Only (WO), or Read-Only (RO). Default values shown are decimal values and take effect after a `s_axi_aresetn`.

A description of each signal is found in Port Descriptions – MAC+PCS Variant.

GT_RESET_REG: 0000

Table 2-46: **GT_RESET_REG: 0000**

Bits	Default	Type	Signal
0	0	RW	ctl_gt_reset_all This is a clear on write register.
1	0	RW	ctl_gt_rx_reset
2	0	RW	ctl_gt_tx_reset

RESET_REG: 0004

Table 2-47: **RESET_REG: 0004**

Bits	Default	Type	Signal
0	0	RW	rx_serdes_reset
29	0	RW	tx_serdes_reset
30	0	RW	rx_reset
31	0	RW	tx_reset

MODE_REG: 0008

Table 2-48: **MODE_REG: 0008**

Bits	Default	Type	Signal
30	1	RW	tick_reg_mode_sel
31	0	RW	ctl_local_loopback

CONFIGURATION_TX_REG1: 000C

Table 2-49: **CONFIGURATION_TX_REG1: 000C**

Bits	Default	Type	Signal
0	1	RW	ctl_tx_enable(1)
1	1	RW	ctl_tx_fcs_ins_enable(1)
2	0	RW	ctl_tx_ignore_fcs(1)
3	0	RW	ctl_tx_send_lfi(1)
4	0	RW	ctl_tx_send_rfi(1)
5	0	RW	ctl_tx_send_idle(1)
13:10	12	RW	ctl_tx_ipg_value(1)
14	0	RW	ctl_tx_test_pattern
15	0	RW	ctl_tx_test_pattern_enable
16	0	RW	ctl_tx_test_pattern_select
17	0	RW	ctl_tx_data_pattern_select

Table 2-49: **CONFIGURATION_TX_REG1: 000C *(Cont'd)***

Bits	Default	Type	Signal
18	0	RW	ctl_tx_custom_preamble_enable(1)
23	0	RW	ctl_tx_prbs31_test_pattern_enable(2)

Notes:
1. Only in MAC+PCS variant
2. Only in PCS variant

CONFIGURATION_RX_REG1: 0014

Table 2-50: **CONFIGURATION_RX_REG1: 0014**

Bits	Default	Type	Signal
0	1	RW	ctl_rx_enable(1)
1	1	RW	ctl_rx_delete_fcs(1)
2	0	RW	ctl_rx_ignore_fcs(1)
3	0	RW	ctl_rx_process_lfi(1)
4	1	RW	ctl_rx_check_sfd(1)
5	1	RW	ctl_rx_check_preamble(1)
6	0	RW	ctl_rx_force_resync(1)
7	0	RW	ctl_rx_test_pattern
8	0	RW	ctl_rx_test_pattern_enable
9	0	RW	ctl_rx_data_pattern_select
10	-	-	Reserved
11	0	RW	ctl_rx_custom_preamble_enable
12	0	RW	ctl_rx_prbs31_test_pattern_enable(2)

Notes:
1. Only in MAC+PCS variant
2. Only in PCS variant

CONFIGURATION_RX_MTU: 0018

Table 2-51: **CONFIGURATION_RX_MTU: 0018**

Bits	Default	Type	Signal
7:0	64	RW	ctl_rx_min_packet_len
30:16	9,600	RW	ctl_rx_max_packet_len

CONFIGURATION_VL_LENGTH_REG: 001C

Table 2-52: **CONFIGURATION_VL_LENGTH_REG: 001C**

Bits	Default	Type	Signal
15:0	20,479	RW	ctl_tx_vl_length_minus1
31:16	20,479	RW	ctl_rx_vl_length_minus1

CONFIGURATION_TSN_REG: 0x019C

Table 2-53: **CONFIGURATION_TSN_REG: 0x019C**

Bits	Default	Type	Signal
0	1	RW	ctl_en_preempt
1	0	RW	ctl_hold_request
2	0	RW	ctl_disable_verify
3	0	RW	ctl_restart_verify
5:4	2'b00	RW	ctl_addfrag_size[1:0]
15:8	8'h01	RW	ctl_verify_time[7:0]
19:16	4'h3	RW	ctl_verify_limit[3:0]

TICK_REG: 0020

Table 2-54: **TICK_REG: 0020**

Bits	Default	Type	Signal
0	0	WO	tick_reg ***Note:*** This is a clear on write register.

CONFIGURATION_REVISION_REG: 0024

Table 2-55: **CONFIGURATION_REVISION_REG: 0024**

Bits	Default	Type	Signal
7:0	2	RO	major_rev
15:8	4	RO	minor_rev
31:24	0	RO	patch_rev

CONFIGURATION_TX_TEST_PAT_SEED_A_LSB: 0028

Table 2-56: **CONFIGURATION_TX_TEST_PAT_SEED_A_LSB: 0028**

Bits	Default	Type	Signal
31:0	0	RW	ctl_tx_test_pattern_seed_a[31:0]

CONFIGURATION_TX_TEST_PAT_SEED_A_MSB: 002C

Table 2-57: **CONFIGURATION_TX_TEST_PAT_SEED_A_MSB: 002C**

Bits	Default	Type	Signal
25:0	0	RW	ctl_tx_test_pattern_seed_a[57:32]

CONFIGURATION_TX_TEST_PAT_SEED_B_LSB: 0030

Table 2-58: **CONFIGURATION_TX_TEST_PAT_SEED_B_LSB: 0030**

Bits	Default	Type	Signal
31:0	0	RW	ctl_tx_test_pattern_seed_b[31:0]

CONFIGURATION_TX_TEST_PAT_SEED_B_MSB: 0034

Table 2-59: **CONFIGURATION_TX_TEST_PAT_SEED_B_MSB: 0034**

Bits	Default	Type	Signal
25:0	0	RW	ctl_tx_test_pattern_seed_b[57:32]

CONFIGURATION_1588_REG: 0038

Table 2-60: **CONFIGURATION_1588_REG: 0038**

Bits	Default	Type	Signal
0	0	RW	ctl_tx_ptp_1step_enable
2	0	RW	ctl_ptp_transpclk_mode
26:16	0	RW	ctl_tx_ptp_latency_adjust

CONFIGURATION_TX_FLOW_CONTROL_REG1: 0040

Table 2-61: **CONFIGURATION_TX_FLOW_CONTROL_REG1: 0040**

Bits	Default	Type	Signal
8:0	0	RW	ctl_tx_pause_enable

CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG1: 0044

Table 2-62: **CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG1: 0044**

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

CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG2: 0048

Table 2-63: **CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG2: 0048**

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

CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG3: 004C

Table 2-64: **CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG3: 004C**

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

CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG4: 0050

Table 2-65: **CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG4: 0050**

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

CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG5: 0054

Table 2-66: **CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG5: 0054**

Bits	Default	Type	Signal
15:0	0	RW	ctl_tx_pause_refresh_timer8

CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG1: 0058

Table 2-67: **CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG1: 0058**

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

CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG2: 005C

Table 2-68: **CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG2: 005C**

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

CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG3: 0060

Table 2-69: **CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG3: 0060**

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

CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG4: 0064

Table 2-70: **CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG4: 0064**

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

CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG5: 0068

Table 2-71: **CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG5: 0068**

Bits	Default	Type	Signal
15:0	0	RW	ctl_tx_pause_quanta8

CONFIGURATION_TX_FLOW_CONTROL_PPP_ETYPE_OP_REG: 006C

Table 2-72: **CONFIGURATION_TX_FLOW_CONTROL_PPP_ETYPE_OP_REG: 006C**

Bits	Default	Type	Signal
15:0	34824	RW	ctl_tx_ethertype_ppp
31:16	257	RW	ctl_tx_opcode_ppp

CONFIGURATION_TX_FLOW_CONTROL_GPP_ETYPE_OP_REG: 0070

Table 2-73: **CONFIGURATION_TX_FLOW_CONTROL_GPP_ETYPE_OP_REG: 0070**

Bits	Default	Type	Signal
15:0	34824	RW	ctl_tx_ethertype_gpp
31:16	1	RW	ctl_tx_opcode_gpp

CONFIGURATION_TX_FLOW_CONTROL_GPP_DA_REG_LSB: 0074

Table 2-74: **CONFIGURATION_TX_FLOW_CONTROL_GPP_DA_REG_LSB: 0074**

Bits	Default	Type	Signal
31:0	0	RW	ctl_tx_da_gpp[31:0]

CONFIGURATION_TX_FLOW_CONTROL_GPP_DA_REG_MSB: 0078

Table 2-75: **CONFIGURATION_TX_FLOW_CONTROL_GPP_DA_REG_MSB: 0078**

Bits	Default	Type	Signal
15:0	0	RW	ctl_tx_da_gpp[47:32]

CONFIGURATION_TX_FLOW_CONTROL_GPP_SA_REG_LSB: 007C

Table 2-76: **CONFIGURATION_TX_FLOW_CONTROL_GPP_SA_REG_LSB: 007C**

Bits	Default	Type	Signal
31:0	0	RW	ctl_tx_sa_gpp[31:0]

CONFIGURATION_TX_FLOW_CONTROL_GPP_SA_REG_MSB: 0080

Table 2-77: **CONFIGURATION_TX_FLOW_CONTROL_GPP_SA_REG_MSB: 0080**

Bits	Default	Type	Signal
15:0	0	RW	ctl_tx_sa_gpp[47:32]

CONFIGURATION_TX_FLOW_CONTROL_PPP_DA_REG_LSB: 0084

Table 2-78: **CONFIGURATION_TX_FLOW_CONTROL_PPP_DA_REG_LSB: 0084**

Bits	Default	Type	Signal
31:0	0	RW	ctl_tx_da_ppp[31:0]

CONFIGURATION_TX_FLOW_CONTROL_PPP_DA_REG_MSB: 0088

Table 2-79: **CONFIGURATION_TX_FLOW_CONTROL_PPP_DA_REG_MSB: 0088**

Bits	Default	Type	Signal
15:0	0	RW	ctl_tx_da_ppp[47:32]

CONFIGURATION_TX_FLOW_CONTROL_PPP_SA_REG_LSB: 008C

Table 2-80: **CONFIGURATION_TX_FLOW_CONTROL_PPP_SA_REG_LSB: 008C**

Bits	Default	Type	Signal
31:0	0	RW	ctl_tx_sa_ppp[31:0]

CONFIGURATION_TX_FLOW_CONTROL_PPP_SA_REG_MSB: 0090

Table 2-81: **CONFIGURATION_TX_FLOW_CONTROL_PPP_SA_REG_MSB: 0090**

Bits	Default	Type	Signal
15:0	0	RW	ctl_tx_sa_ppp[47:32]

CONFIGURATION_RX_FLOW_CONTROL_REG1: 0094

Table 2-82: **CONFIGURATION_RX_FLOW_CONTROL_REG1: 0094**

Bits	Default	Type	Signal
8:0	0	RW	ctl_rx_pause_enable
9	0	RW	ctl_rx_forward_control
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	RW	ctl_rx_check_ack

CONFIGURATION_RX_FLOW_CONTROL_REG2: 0098

Table 2-83: **CONFIGURATION_RX_FLOW_CONTROL_REG2: 0098**

Bits	Default	Type	Signal
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

Table 2-83: **CONFIGURATION_RX_FLOW_CONTROL_REG2: 0098 *(Cont'd)***

Bits	Default	Type	Signal
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_mcast_ppp
16	0	RW	ctl_rx_check_ucast_ppp
17	0	RW	ctl_rx_check_sa_ppp
18	0	RW	ctl_rx_check_etype_ppp
19	0	RW	ctl_rx_check_opcode_ppp

CONFIGURATION_RX_FLOW_CONTROL_PPP_ETYPE_OP_REG: 009C

Table 2-84: **CONFIGURATION_RX_FLOW_CONTROL_PPP_ETYPE_OP_REG: 009C**

Bits	Default	Type	Signal
15:0	34,824	RW	ctl_rx_etype_ppp
31:16	257	RW	ctl_rx_opcode_ppp

CONFIGURATION_RX_FLOW_CONTROL_GPP_ETYPE_OP_REG: 00A0

Table 2-85: **CONFIGURATION_RX_FLOW_CONTROL_GPP_ETYPE_OP_REG: 00A0**

Bits	Default	Type	Signal
15:0	34,824	RW	ctl_rx_etype_gpp
31:16	1	RW	ctl_rx_opcode_gpp

CONFIGURATION_RX_FLOW_CONTROL_GCP_PCP_TYPE_REG: 00A4

Table 2-86: **CONFIGURATION_RX_FLOW_CONTROL_GCP_PCP_TYPE_REG: 00A4**

Bits	Default	Type	Signal
15:0	34,824	RW	ctl_rx_etype_gcp
31:16	34,824	RW	ctl_rx_etype_pcp

CONFIGURATION_RX_FLOW_CONTROL_PCP_OP_REG: 00A8

Table 2-87: **CONFIGURATION_RX_FLOW_CONTROL_PCP_OP_REG: 00A8**

Bits	Default	Type	Signal
15:0	257	RW	ctl_rx_opcode_min_pcp
31:16	257	RW	ctl_rx_opcode_max_pcp

CONFIGURATION_RX_FLOW_CONTROL_GCP_OP_REG: 00AC

Table 2-88: **CONFIGURATION_RX_FLOW_CONTROL_GCP_OP_REG: 00AC**

Bits	Default	Type	Signal
15:0	1	RW	ctl_rx_opcode_min_gcp
31:16	6	RW	ctl_rx_opcode_max_gcp

CONFIGURATION_RX_FLOW_CONTROL_DA_REG1_LSB: 00B0

Table 2-89: **CONFIGURATION_RX_FLOW_CONTROL_DA_REG1_LSB: 00B0**

Bits	Default	Type	Signal
31:0	0	RW	ctl_rx_pause_da_ucast[31:0]

CONFIGURATION_RX_FLOW_CONTROL_DA_REG1_MSB: 00B4

Table 2-90: **CONFIGURATION_RX_FLOW_CONTROL_DA_REG1_MSB: 00B4**

Bits	Default	Type	Signal
15:0	0	RW	ctl_rx_pause_da_ucast[47:32]

CONFIGURATION_RX_FLOW_CONTROL_DA_REG2_LSB: 00B8

Table 2-91: **CONFIGURATION_RX_FLOW_CONTROL_DA_REG2_LSB: 00B8**

Bits	Default	Type	Signal
31:0	0	RW	ctl_rx_pause_da_mcast[31:0]

CONFIGURATION_RX_FLOW_CONTROL_DA_REG2_MSB: 00BC

Table 2-92: **CONFIGURATION_RX_FLOW_CONTROL_DA_REG2_MSB: 00BC**

Bits	Default	Type	Signal
15:0	0	RW	ctl_rx_pause_da_mcast[47:32]

CONFIGURATION_RX_FLOW_CONTROL_SA_REG1_LSB: 00C0

Table 2-93: **CONFIGURATION_RX_FLOW_CONTROL_SA_REG1_LSB: 00C0**

Bits	Default	Type	Signal
31:0	0	RW	ctl_rx_pause_sa[31:0]

CONFIGURATION_RX_FLOW_CONTROL_SA_REG1_MSB: 00C4

Table 2-94: **CONFIGURATION_RX_FLOW_CONTROL_SA_REG1_MSB: 00C4**

Bits	Default	Type	Signal
15:0	0	RW	ctl_rx_pause_sa[47:32]

CONFIGURATION_RSFEC_REG: 00D0

Table 2-95: **CONFIGURATION_RSFEC_REG: 00D0**

Bits	Default	Type	Signal
0	0	RW	ctl_rsfec_enable
1	0	RW	ctl_rsfec_consortium_25g
2	0	RW	ctl_rx_rsfec_enable_indication
3	0	RW	ctl_rx_rsfec_enable_correction
5	0	RW	ctl_rsfec_ieee_error_indication_mode

CONFIGURATION_FEC_REG: 00D4

Table 2-96: **CONFIGURATION_FEC_REG: 00D4**

Bits	Default	Type	Signal
0	0	RW	ctl_fec_rx_enable
1	0	RW	ctl_fec_tx_enable
2	0	RW	ctl_fec_enable_error_to_pcs

CONFIGURATION_AN_CONTROL_REG1: 00E0

Table 2-97: **CONFIGURATION_AN_CONTROL_REG1: 00E0**

Bits	Default	Type	Signal
0	0	RW	ctl_autoneg_enable
1	1	RW	ctl_autoneg_bypass(1)
9:2	0	RW	ctl_an_nonce_seed
10	0	RW	ctl_an_pseudo_sel
11	0	RW	ctl_restart_negotiation

Table 2-97: **CONFIGURATION_AN_CONTROL_REG1: 00E0 *(Cont'd)***

Bits	Default	Type	Signal
12	0	RW	ctl_an_local_fault

Notes:

1. For simulation, the `ctl_autoneg_bypass` value is written as 1 during reset. To test with the ANLT enabled configuration, write the register with `ctl_autoneg_enable` to 1 and `ctl_autoneg_bypass` to 0.

CONFIGURATION_AN_CONTROL_REG2: 00E4

Table 2-98: **CONFIGURATION_AN_CONTROL_REG2: 00E4**

Bits	Default	Type	Signal
0	0	RW	ctl_an_pause
1	0	RW	ctl_an_asmdir
16	0	RW	ctl_an_fec_10g_request
17	0	RW	ctl_an_fec_ability_override
18	0	RW	ctl_an_cl91_fec_request
19	0	RW	ctl_an_cl91_fec_ability
20	0	RW	ctl_an_fec_25g_rs_request
21	0	RW	ctl_an_fec_25g_baser_request

CONFIGURATION_AN_ABILITY: 00F8

Table 2-99: **CONFIGURATION_AN_ABILITY: 00F8**

Bits	Default	Type	Signal
0	0	RW	ctl_an_ability_1000base_kx
1	0	RW	ctl_an_ability_10gbase_kx4
2	0	RW	ctl_an_ability_10gbase_kr
3	0	RW	ctl_an_ability_40gbase_kr4
4	0	RW	ctl_an_ability_40gbase_cr4
5	0	RW	ctl_an_ability_100gbase_cr10
6	0	RW	ctl_an_ability_100gbase_kp4
7	0	RW	ctl_an_ability_100gbase_kr4
8	0	RW	ctl_an_ability_100gbase_cr4
9	0	RW	ctl_an_ability_25gbase_krcr_s
10	0	RW	ctl_an_ability_25gbase_krcr
11	0	RW	ctl_an_ability_25gbase_kr1
12	0	RW	ctl_an_ability_25gbase_cr1

Table 2-99: **CONFIGURATION_AN_ABILITY: 00F8 *(Cont'd)***

Bits	Default	Type	Signal
13	0	RW	ctl_an_ability_50gbase_kr2
14	0	RW	ctl_an_ability_50gbase_cr2

CONFIGURATION_LT_CONTROL_REG1: 0100

Table 2-100: **CONFIGURATION_LT_CONTROL_REG1: 0100**

Bits	Default	Type	Signal
0	0	RW	ctl_lt_training_enable
1	0	RW	ctl_lt_restart_training

CONFIGURATION_LT_TRAINED_REG: 0104

Table 2-101: **CONFIGURATION_LT_TRAINED_REG: 0104**

Bits	Default	Type	Signal
0	0	RW	ctl_lt_rx_trained

CONFIGURATION_LT_PRESET_REG: 0108

Table 2-102: **CONFIGURATION_LT_PRESET_REG: 0108**

Bits	Default	Type	Signal
0	0	RW	ctl_lt_preset_to_tx

CONFIGURATION_LT_INIT_REG: 010C

Table 2-103: **CONFIGURATION_LT_INIT_REG: 010C**

Bits	Default	Type	Signal
0	0	RW	ctl_lt_initialize_to_tx

CONFIGURATION_LT_SEED_REG0: 0110

Table 2-104: **CONFIGURATION_LT_SEED_REG0: 0110**

Bits	Default	Type	Signal
10:0	0	RW	ctl_lt_pseudo_seed0

CONFIGURATION_LT_COEFFICIENT_REG0: 0130

Table 2-105: **CONFIGURATION_LT_COEFFICIENT_REG0: 0130**

Bits	Default	Type	Signal
1:0	0	RW	ctl_lt_k_p1_to_tx0
3:2	0	RW	ctl_lt_k0_to_tx0
5:4	0	RW	ctl_lt_k_m1_to_tx0
7:6	0	RW	ctl_lt_stat_p1_to_tx0
9:8	0	RW	ctl_lt_stat0_to_tx0
11:10	0	RW	ctl_lt_stat_m1_to_tx0

USER_REG_0: 0134

Table 2-106: **USER_REG_0: 0134**

Bits	Default	Type	Signal
31:0	0	RW	user_reg0

SWITCH_CORE_SPEED_REG: 0138

Table 2-107: **SWITCH_CORE_SPEED_REG: 0138**

Bits	Default	Type	Signal
0	0	RW	axi_ctl_core_mode_switch

For Runtime Switch mode only. A write 1 enables the mode switch between 10G and 25G. This is a clear on the write register. This is an input to the trans debug module that performs the GT DRP operations.

CONFIGURATION_1588_32BIT_REG: 0x013C

Table 2-108: **CONFIGURATION_1588_32BIT_REG: 0x013C**

Bits	Default	Type	Signal
0	0	RW	ctl_tx_lat_adj_enb
1	0	RW	ctl_rx_lat_adj_enb
2	0	RW	ctl_ptp_transpclk_mode
3	0	RW	ctl_tx_timstamp_adj_enb
4	0	RW	ctl_rx_timstamp_adj_enb
5	0	RW	ctl_core_speed

TX_CONFIGURATION_1588_REG: 0x140

Table 2-109: **TX_CONFIGURATION_1588_REG: 0x140**

Bits	Default	Type	Signal
31:0	0	RW	ctl_tx_latency

RX_CONFIGURATION_1588_REG: 0x144

Table 2-110: **RX_CONFIGURATION_1588_REG: 0x144**

Bits	Default	Type	Signal
31:0	0	RW	ctl_rx_latency

Status Registers

Table 2-212 to Table 2-129 define the bit assignments for the status registers.

Some bits are sticky, that is, latching their value High or Low once set. This is indicated by the type LH (Latched High) or LL (Latched Low).

A description of each signal is found in Port Descriptions – MAC+PCS Variant.

STAT_TX_STATUS_REG1: 0400

Table 2-111: **STAT_TX_STATUS_REG1: 0400**

Bits	Default	Type	Signal
0	0	RO LH	stat_tx_local_fault
1(1)	0	RO LH	stat_tx_gmii_fifo_ovf_1h_r_out
2(1)	0	RO LH	stat_tx_gmii_fifo_unf_1h_r_out
7(2)	0	RO LH	stat_tx_bad_parity

Notes:
1. Bits 1 and 2 are for the 10G MAC-only variant.
2. Bit 7 is applicable only for the **Datapath Parity Feature**.

STAT_RX_STATUS_REG1: 0404

Table 2-112: **STAT_RX_STATUS_REG1: 0404**

Bits	Default	Type	Signal
4	0	RO LH	stat_rx_hi_ber
5	0	RO LH	stat_rx_remote_fault(1)
6	0	RO LH	stat_rx_local_fault

Table 2-112: **STAT_RX_STATUS_REG1: 0404**

Bits	Default	Type	Signal
7	0	RO LH	stat_rx_internal_local_fault[(1)]
8	0	RO LH	stat_rx_received_local_fault[(1)]
9	0	RO LH	stat_rx_bad_preamble[(1)]
10	0	RO LH	stat_rx_bad_sfd[(1)]
11	0	RO LH	stat_rx_got_signal_os[(1)]

Notes:

1. Only in MAC+PCS variant

STAT_STATUS_REG1: 0408

Table 2-113: **STAT_STATUS_REG1: 0408**

Bits	Default	Type	Signal
0	0	RO LH	stat_tx_fifo_error[(1)]
4	0	RO LH	stat_tx_ptp_fifo_read_error
5	0	RO LH	stat_tx_ptp_fifo_write_error
16	0	RO LH	stat_rx_fifo_error[(1)]

Notes:

1. Only in PCS variant

STAT_RX_BLOCK_LOCK_REG: 040C

Table 2-114: **STAT_RX_BLOCK_LOCK_REG: 040C**

Bits	Default	Type	Signal
0	1	RO LL	stat_rx_block_lock

STAT_RX_RSFEC_STATUS_REG: 043C

Table 2-115: **STAT_RX_RSFEC_STATUS_REG: 043C**

Bits	Default	Type	Signal
0	1	RO LL	stat_rx_rsfec_lane_alignment_status
1	1	RO LL	stat_rx_rsfec_hi_ser

STAT_RX_FEC_STATUS_REG: 0448

Table 2-116: **STAT_RX_FEC_STATUS_REG: 0448**

Bits	Default	Type	Signal
0	1	RO LL	stat_fec_rx_lock
16	1	RO LL	stat_fec_lock_error

STAT_TX_RSFEC_STATUS_REG: 044C

Table 2-117: **STAT_TX_RSFEC_STATUS_REG: 044C**

Bits	Default	Type	Signal
0	1	RO LL	stat_tx_rsfec_lane_alignment_status

STAT_TSN_REG: 0x049C

Table 2-118: **STAT_TSN_REG: 0x049C**

Bits	Default	Type	Signal
1:0	0	RO	stat_tx_mm_verify

STAT_TX_FLOW_CONTROL_REG1: 0450

Table 2-119: **STAT_TX_FLOW_CONTROL_REG1: 0450**

Bits	Default	Type	Signal
8:0	0	RO LH	stat_tx_pause_valid

STAT_RX_FLOW_CONTROL_REG1: 0454

Table 2-120: **STAT_RX_FLOW_CONTROL_REG1: 0454**

Bits	Default	Type	Signal
8:0	0	RO LH	stat_rx_pause_req
17:9	0	RO LH	stat_rx_pause_valid

STAT_AN_STATUS: 0458

Table 2-121: **STAT_AN_STATUS: 0458**

Bits	Default	Type	Signal
0	0	RO	stat_an_fec_enable
1	0	RO	stat_an_rs_fec_enable
2	0	RO	stat_an_autoneg_complete

Table 2-121: **STAT_AN_STATUS: 0458 *(Cont'd)***

Bits	Default	Type	Signal
3	0	RO	stat_an_parallel_detection_fault
4	0	RO	stat_an_tx_pause_enable
5	0	RO	stat_an_rx_pause_enable
6	0	RO LH	stat_an_lp_ability_valid
7	0	RO	stat_an_lp_autoneg_able
8	0	RO	stat_an_lp_pause
9	0	RO	stat_an_lp_asm_dir
10	0	RO	stat_an_lp_rf
11	0	RO	stat_an_lp_fec_10g_ability
12	0	RO	stat_an_lp_fec_10g_request
13	0	RO LH	stat_an_lp_extended_ability_valid
17:14	0	RO	stat_an_lp_ability_extended_fec
18	0	RO	stat_an_lp_fec_25g_rs_request
19	0	RO	stat_an_lp_fec_25g_baser_request

STAT_AN_ABILITY: 045C

Table 2-122: **STAT_AN_ABILITY: 045C**

Bits	Default	Type	Signal
0	0	RO	stat_an_lp_ability_1000base_kx
1	0	RO	stat_an_lp_ability_10gbase_kx4
2	0	RO	stat_an_lp_ability_10gbase_kr
3	0	RO	stat_an_lp_ability_40gbase_kr4
4	0	RO	stat_an_lp_ability_40gbase_cr4
5	0	RO	stat_an_lp_ability_100gbase_cr10
6	0	RO	stat_an_lp_ability_100gbase_kp4
7	0	RO	stat_an_lp_ability_100gbase_kr4
8	0	RO	stat_an_lp_ability_100gbase_cr4
9	0	RO	stat_an_lp_ability_25gbase_krcr_s
10	0	RO	stat_an_lp_ability_25gbase_krcr
11	0	RO	stat_an_lp_ability_25gbase_kr1
12	0	RO	stat_an_lp_ability_25gbase_cr1
13	0	RO	stat_an_lp_ability_50gbase_kr2
14	0	RO	stat_an_lp_ability_50gbase_cr2

STAT_AN_LINK_CTL: 0460

Table 2-123: **STAT_AN_LINK_CTL: 0460**

Bits	Default	Type	Signal
1:0	0	RO	stat_an_link_cntl_1000base_kx
3:2	0	RO	stat_an_link_cntl_10gbase_kx4
5:4	0	RO	stat_an_link_cntl_10gbase_kr
7:6	0	RO	stat_an_link_cntl_40gbase_kr4
9:8	0	RO	stat_an_link_cntl_40gbase_cr4
11:10	0	RO	stat_an_link_cntl_100gbase_cr10
13:12	0	RO	stat_an_link_cntl_100gbase_kp4
15:14	0	RO	stat_an_link_cntl_100gbase_kr4
17:16	0	RO	stat_an_link_cntl_100gbase_cr4
19:18	0	RO	stat_an_link_cntl_25gbase_krcr_s
21:20	0	RO	stat_an_link_cntl_25gbase_krcr
23:22	0	RO	stat_an_link_cntl_25gbase_kr1
25:24	0	RO	stat_an_link_cntl_25gbase_cr1
27:26	0	RO	stat_an_link_cntl_50gbase_kr2
29:28	0	RO	stat_an_link_cntl_50gbase_cr2

STAT_LT_STATUS_REG1: 0464

Table 2-124: **STAT_LT_STATUS_REG1: 0464**

Bits	Default	Type	Signal
0	0	RO	stat_lt_initialize_from_rx
16	0	RO	stat_lt_preset_from_rx

STAT_LT_STATUS_REG2: 0468

Table 2-125: **STAT_LT_STATUS_REG2: 0468**

Bits	Default	Type	Signal
0	0	RO	stat_lt_training
16	0	RO	stat_lt_frame_lock

STAT_LT_STATUS_REG3: 046C

Table 2-126: **STAT_LT_STATUS_REG3: 046C**

Bits	Default	Type	Signal
0	0	RO	stat_lt_signal_detect
16	0	RO	stat_lt_training_fail

STAT_LT_STATUS_REG4: 0470

Table 2-127: **STAT_LT_STATUS_REG4: 0470**

Bits	Default	Type	Signal
0	0	RO LH	stat_lt_rx_sof

STAT_LT_COEFFICIENT0_REG: 0474

Table 2-128: **STAT_LT_COEFFICIENT0_REG: 0474**

Bits	Default	Type	Signal
1:0	0	RO	stat_lt_k_p1_from_rx0
3:2	0	RO	stat_lt_k0_from_rx0
5:4	0	RO	stat_lt_k_m1_from_rx0
7:6	0	RO	stat_lt_stat_p1_from_rx0
9:8	0	RO	stat_lt_stat0_from_rx0
11:10	0	RO	stat_lt_stat_m1_from_rx0

STAT_RX_VALID_CTRL_CODE: 0494

Table 2-129: **STAT_RX_VALID_CTRL_CODE: 0494**

Bits	Default	Type	Signal
0	0	RO LH	stat_rx_valid_ctrl_code

STAT_CORE_SPEED_REG: 0498

Table 2-130: **STAT_CORE_SPEED_REG: 0498**

Bits	Default	Type	Signal
0	GUI configured	RO	stat_core_speed
1	GUI configured	RO	runtime_switchable

Notes:

1. This register will be available only for the 64-bit AXI-Stream datapath interface.
   1:0 : 00 - Standalone 25G
   01 - Standalone 10G
   10 - Runtime Switchable 25G
   11 - Runtime Switchable 10G

Statistics Counters

Table 2-131 through Table 2-270 define the bit assignments for the statistics counters.

Counters are 48 bits and require two 32-bit address spaces containing the MSB and LSB as indicated. The default value of all counters is 0. Counters are cleared when read by `tick_reg` (or `pm_tick` if so selected) but the register containing the count retains its value. Each counter saturates at FFFFFFFFFFFF (hex).

A description of each signal is found in Port Descriptions – MAC+PCS Variant.

STATUS_CYCLE_COUNT_LSB: 0500

Table 2-131: **STATUS_CYCLE_COUNT_LSB: 0500**

Bits	Default	Type	Signal
31:0	0	RO HIST	stat_cycle_count[31:0]

STATUS_CYCLE_COUNT_MSB: 0504

Table 2-132: **STATUS_CYCLE_COUNT_MSB: 0504**

Bits	Default	Type	Signal
15:0	0	RO HIST	stat_cycle_count[47:32]

STAT_RX_FRAMING_ERR_LSB: 0648

Table 2-133: **STAT_RX_FRAMING_ERR_LSB: 0648**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_framing_err_count[31:0]

STAT_RX_FRAMING_ERR_MSB: 064C

Table 2-134: **STAT_RX_FRAMING_ERR_MSB: 064C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_framing_err_count[48-1:32]

STAT_RX_BAD_CODE_LSB: 0660

Table 2-135: **STAT_RX_BAD_CODE_LSB: 0660**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_bad_code_count[31:0]

STAT_RX_BAD_CODE_MSB: 0664

Table 2-136: **STAT_RX_BAD_CODE_MSB: 0664**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_bad_code_count[47:32]

STAT_RX_ERROR_LSB: 0668

Table 2-137: **STAT_RX_ERROR_LSB: 0668**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_error_count[31:0]

STAT_RX_ERROR_MSB: 066C

Table 2-138: **STAT_RX_ERROR_MSB: 066C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_error_count[47:32]

STAT_RX_RSFEC_CORRECTED_CW_INC_LSB: 0670

Table 2-139: **STAT_RX_RSFEC_CORRECTED_CW_INC_LSB: 0670**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_rsfec_corrected_cw_inc_count[31:0]

STAT_RX_RSFEC_CORRECTED_CW_INC_MSB: 0674

Table 2-140: **STAT_RX_RSFEC_CORRECTED_CW_INC_MSB: 0674**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_rsfec_corrected_cw_inc_count[47:32]

STAT_RX_RSFEC_UNCORRECTED_CW_INC_LSB: 0678

Table 2-141: **STAT_RX_RSFEC_UNCORRECTED_CW_INC_LSB: 0678**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_rsfec_uncorrected_cw_inc_count[31:0]

STAT_RX_RSFEC_UNCORRECTED_CW_INC_MSB: 067C

Table 2-142: **STAT_RX_RSFEC_UNCORRECTED_CW_INC_MSB: 067C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_rsfec_uncorrected_cw_inc_count[47:32]

STAT_RX_RSFEC_ERR_COUNT0_INC_LSB: 0680

Table 2-143: **STAT_RX_RSFEC_ERR_COUNT0_INC_LSB: 0680**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_rsfec_err_count_inc_count[31:0]

STAT_RX_RSFEC_ERR_COUNT0_INC_MSB: 0684

Table 2-144: **STAT_RX_RSFEC_ERR_COUNT0_INC_MSB: 0684**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_rsfec_err_count_inc_count[47:32]

STAT_TX_FRAME_ERROR_LSB: 06A0

Table 2-145: **STAT_TX_FRAME_ERROR_LSB: 06A0**

Bits	Default	Type	Signal
31:0	0	HIST	stat_tx_frame_error_count[31:0]

STAT_TX_FRAME_ERROR_MSB: 06A4

Table 2-146: **STAT_TX_FRAME_ERROR_MSB: 06A4**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_frame_error_count[47:32]

STAT_TX_TOTAL_PACKETS_LSB: 0700

Table 2-147: **STAT_TX_TOTAL_PACKETS_LSB: 0700**

Bits	Default	Type	Signal
31:0	0	HIST	stat_tx_total_packets_count[31:0]

STAT_TX_TOTAL_PACKETS_MSB: 0704

Table 2-148: **STAT_TX_TOTAL_PACKETS_MSB: 0704**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_total_packets_count[47:32]

STAT_TX_TOTAL_GOOD_PACKETS_LSB: 0708

Table 2-149: **STAT_TX_TOTAL_GOOD_PACKETS_LSB: 0708**

Bits	Default	Type	Signal
31:0	0	HIST	stat_tx_total_good_packets_count[31:0]

STAT_TX_TOTAL_GOOD_PACKETS_MSB: 070C

Table 2-150: **STAT_TX_TOTAL_GOOD_PACKETS_MSB: 070C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_total_good_packets_count[47:32]

STAT_TX_TOTAL_BYTES_LSB: 0710

Table 2-151: **STAT_TX_TOTAL_BYTES_LSB: 0710**

Bits	Default	Type	Signal
31:0	0	HIST	stat_tx_total_bytes_count[31:0]

STAT_TX_TOTAL_BYTES_MSB: 0714

Table 2-152: **STAT_TX_TOTAL_BYTES_MSB: 0714**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_total_bytes_count[47:32]

STAT_TX_TOTAL_GOOD_BYTES_LSB: 0718

Table 2-153: **STAT_TX_TOTAL_GOOD_BYTES_LSB: 0718**

Bits	Default	Type	Signal
31:0	0	HIST	stat_tx_total_good_bytes_count[31:0]

STAT_TX_TOTAL_GOOD_BYTES_MSB: 071C

Table 2-154: **STAT_TX_TOTAL_GOOD_BYTES_MSB: 071C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_total_good_bytes_count[47:32]

STAT_TX_PACKET_64_BYTES_LSB: 0720

Table 2-155: **STAT_TX_PACKET_64_BYTES_LSB: 0720**

Bits	Default	Type	Signal
31:0	0	HIST	stat_tx_packet_64_bytes_count[31:0]

STAT_TX_PACKET_64_BYTES_MSB: 0724

Table 2-156: **STAT_TX_PACKET_64_BYTES_MSB: 0724**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_packet_64_bytes_count[47:32]

STAT_TX_PACKET_65_127_BYTES_LSB: 0728

Table 2-157: **STAT_TX_PACKET_65_127_BYTES_LSB: 0728**

Bits	Default	Type	Signal
31:0	0	HIST	stat_tx_packet_65_127_bytes_count[31:0]

STAT_TX_PACKET_65_127_BYTES_MSB: 072C

Table 2-158: **STAT_TX_PACKET_65_127_BYTES_MSB: 072C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_packet_65_127_bytes_count[47:32]

STAT_TX_PACKET_128_255_BYTES_LSB: 0730

Table 2-159: **STAT_TX_PACKET_128_255_BYTES_LSB: 0730**

Bits	Default	Type	Signal
31:0	0	HIST	stat_tx_packet_128_255_bytes_count[31:0]

STAT_TX_PACKET_128_255_BYTES_MSB: 0734

Table 2-160: **STAT_TX_PACKET_128_255_BYTES_MSB: 0734**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_packet_128_255_bytes_count[47:32]

STAT_TX_PACKET_256_511_BYTES_LSB: 0738

Table 2-161: **STAT_TX_PACKET_256_511_BYTES_LSB: 0738**

Bits	Default	Type	Signal
31:0	0	HIST	stat_tx_packet_256_511_bytes_count[31:0]

STAT_TX_PACKET_256_511_BYTES_MSB: 073C

Table 2-162: **STAT_TX_PACKET_256_511_BYTES_MSB: 073C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_packet_256_511_bytes_count[47:32]

STAT_TX_PACKET_512_1023_BYTES_LSB: 0740

Table 2-163: **STAT_TX_PACKET_512_1023_BYTES_LSB: 0740**

Bits	Default	Type	Signal
31:0	0	HIST	stat_tx_packet_512_1023_bytes_count[31:0]

STAT_TX_PACKET_512_1023_BYTES_MSB: 0744

Table 2-164: **STAT_TX_PACKET_512_1023_BYTES_MSB: 0744**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_packet_512_1023_bytes_count[47:32]

STAT_TX_PACKET_1024_1518_BYTES_LSB: 0748

Table 2-165: **STAT_TX_PACKET_1024_1518_BYTES_LSB: 0748**

Bits	Default	Type	Signal
31:0	0	HIST	stat_tx_packet_1024_1518_bytes_count[31:0]

STAT_TX_PACKET_1024_1518_BYTES_MSB: 074C

Table 2-166: **STAT_TX_PACKET_1024_1518_BYTES_MSB: 074C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_packet_1024_1518_bytes_count[47:32]

STAT_TX_PACKET_1519_1522_BYTES_LSB: 0750

Table 2-167: **STAT_TX_PACKET_1519_1522_BYTES_LSB: 0750**

Bits	Default	Type	Signal
31:0	0	HIST	stat_tx_packet_1519_1522_bytes_count[31:0]

STAT_TX_PACKET_1519_1522_BYTES_MSB: 0754

Table 2-168: **STAT_TX_PACKET_1519_1522_BYTES_MSB: 0754**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_packet_1519_1522_bytes_count[47:32]

STAT_TX_PACKET_1523_1548_BYTES_LSB: 0758

Table 2-169: **STAT_TX_PACKET_1523_1548_BYTES_LSB: 0758**

Bits	Default	Type	Signal
31:0	0	HIST	stat_tx_packet_1523_1548_bytes_count[31:0]

STAT_TX_PACKET_1523_1548_BYTES_MSB: 075C

Table 2-170: **STAT_TX_PACKET_1523_1548_BYTES_MSB: 075C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_packet_1523_1548_bytes_count[47:32]

STAT_TX_PACKET_1549_2047_BYTES_LSB: 0760

Table 2-171: **STAT_TX_PACKET_1549_2047_BYTES_LSB: 0760**

Bits	Default	Type	Signal
31:0	0	HIST	stat_tx_packet_1549_2047_bytes_count[31:0]

STAT_TX_PACKET_1549_2047_BYTES_MSB: 0764

Table 2-172: **STAT_TX_PACKET_1549_2047_BYTES_MSB: 0764**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_packet_1549_2047_bytes_count[47:32]

STAT_TX_PACKET_2048_4095_BYTES_LSB: 0768

Table 2-173: **STAT_TX_PACKET_2048_4095_BYTES_LSB: 0768**

Bits	Default	Type	Signal
31:0	0	HIST	stat_tx_packet_2048_4095_bytes_count[31:0]

STAT_TX_PACKET_2048_4095_BYTES_MSB: 076C

Table 2-174: **STAT_TX_PACKET_2048_4095_BYTES_MSB: 076C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_packet_2048_4095_bytes_count[47:32]

STAT_TX_PACKET_4096_8191_BYTES_LSB: 0770

Table 2-175: **STAT_TX_PACKET_4096_8191_BYTES_LSB: 0770**

Bits	Default	Type	Signal
31:0	0	HIST	stat_tx_packet_4096_8191_bytes_count[31:0]

STAT_TX_PACKET_4096_8191_BYTES_MSB: 0774

Table 2-176: **STAT_TX_PACKET_4096_8191_BYTES_MSB: 0774**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_packet_4096_8191_bytes_count[47:32]

STAT_TX_PACKET_8192_9215_BYTES_LSB: 0778

Table 2-177: **STAT_TX_PACKET_8192_9215_BYTES_LSB: 0778**

Bits	Default	Type	Signal
31:0	0	HIST	stat_tx_packet_8192_9215_bytes_count[31:0]

STAT_TX_PACKET_8192_9215_BYTES_MSB: 077C

Table 2-178: **STAT_TX_PACKET_8192_9215_BYTES_MSB: 077C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_packet_8192_9215_bytes_count[47:32]

STAT_TX_PACKET_LARGE_LSB: 0780

Table 2-179: **STAT_TX_PACKET_LARGE_LSB: 0780**

Bits	Default	Type	Signal
31:0	0	HIST	stat_tx_packet_large_count[31:0]

STAT_TX_PACKET_LARGE_MSB: 0784

Table 2-180: **STAT_TX_PACKET_LARGE_MSB: 0784**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_packet_large_count[47:32]

STAT_TX_PACKET_SMALL_LSB: 0788

Table 2-181: **STAT_TX_PACKET_SMALL_LSB: 0788**

Bits	Default	Type	Signal
31:0	0	HIST	stat_tx_packet_small_count[31:0]

STAT_TX_PACKET_SMALL_MSB: 078C

Table 2-182: **STAT_TX_PACKET_SMALL_MSB: 078C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_packet_small_count[47:32]

STAT_TX_BAD_FCS_LSB: 07B8

Table 2-183: **STAT_TX_BAD_FCS_LSB: 07B8**

Bits	Default	Type	Signal
31:0	0	HIST	stat_tx_bad_fcs_count[31:0]

STAT_TX_BAD_FCS_MSB: 07BC

Table 2-184: **STAT_TX_BAD_FCS_MSB: 07BC**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_bad_fcs_count[47:32]

STAT_TX_UNICAST_LSB: 07D0

Table 2-185: **STAT_TX_UNICAST_LSB: 07D0**

Bits	Default	Type	Signal
31:0	0	HIST	stat_tx_unicast_count[31:0]

STAT_TX_UNICAST_MSB: 07D4

Table 2-186: **STAT_TX_UNICAST_MSB: 07D4**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_unicast_count[47:32]

STAT_TX_MULTICAST_LSB: 07D8

Table 2-187: **STAT_TX_MULTICAST_LSB: 07D8**

Bits	Default	Type	Signal
31:0	0	HIST	stat_tx_multicast_count[31:0]

STAT_TX_MULTICAST_MSB: 07DC

Table 2-188: **STAT_TX_MULTICAST_MSB: 07DC**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_multicast_count[47:32]

STAT_TX_BROADCAST_LSB: 07E0

Table 2-189: **STAT_TX_BROADCAST_LSB: 07E0**

Bits	Default	Type	Signal
31:0	0	HIST	stat_tx_broadcast_count[31:0]

STAT_TX_BROADCAST_MSB: 07E4

Table 2-190: **STAT_TX_BROADCAST_MSB: 07E4**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_broadcast_count[47:32]

STAT_TX_VLAN_LSB: 07E8

Table 2-191: **STAT_TX_VLAN_LSB: 07E8**

Bits	Default	Type	Signal
31:0	0	HIST	stat_tx_vlan_count[31:0]

STAT_TX_VLAN_MSB: 07EC

Table 2-192: **STAT_TX_VLAN_MSB: 07EC**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_vlan_count[47:32]

STAT_TX_PAUSE_LSB: 07F0

Table 2-193: **STAT_TX_PAUSE_LSB: 07F0**

Bits	Default	Type	Signal
31:0	0	HIST	stat_tx_pause_count[31:0]

STAT_TX_PAUSE_MSB: 07F4

Table 2-194: **STAT_TX_PAUSE_MSB: 07F4**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_pause_count[47:32]

STAT_TX_USER_PAUSE_LSB: 07F8

Table 2-195: **STAT_TX_USER_PAUSE_LSB: 07F8**

Bits	Default	Type	Signal
31:0	0	HIST	stat_tx_user_pause_count[31:0]

STAT_TX_USER_PAUSE_MSB: 07FC

Table 2-196: **STAT_TX_USER_PAUSE_MSB: 07FC**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_user_pause_count[47:32]

STAT_RX_TOTAL_PACKETS_LSB: 0808

Table 2-197: **STAT_RX_TOTAL_PACKETS_LSB: 0808**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_total_packets_count[31:0]

STAT_RX_TOTAL_PACKETS_MSB: 080C

Table 2-198: **STAT_RX_TOTAL_PACKETS_MSB: 080C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_total_packets_count[47:32]

STAT_RX_TOTAL_GOOD_PACKETS_LSB: 0810

Table 2-199: **STAT_RX_TOTAL_GOOD_PACKETS_LSB: 0810**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_total_good_packets_count[31:0]

STAT_RX_TOTAL_GOOD_PACKETS_MSB: 0814

Table 2-200: **STAT_RX_TOTAL_GOOD_PACKETS_MSB: 0814**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_total_good_packets_count[47:32]

STAT_RX_TOTAL_BYTES_LSB: 0818

Table 2-201: **STAT_RX_TOTAL_BYTES_LSB: 0818**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_total_bytes_count[31:0]

STAT_RX_TOTAL_BYTES_MSB: 081C

Table 2-202: **STAT_RX_TOTAL_BYTES_MSB: 081C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_total_bytes_count[47:32]

STAT_RX_TOTAL_GOOD_BYTES_LSB: 0820

Table 2-203: **STAT_RX_TOTAL_GOOD_BYTES_LSB: 0820**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_total_good_bytes_count[31:0]

STAT_RX_TOTAL_GOOD_BYTES_MSB: 0824

Table 2-204: **STAT_RX_TOTAL_GOOD_BYTES_MSB: 0824**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_total_good_bytes_count[47:32]

STAT_RX_PACKET_64_BYTES_LSB: 0828

Table 2-205: **STAT_RX_PACKET_64_BYTES_LSB: 0828**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_packet_64_bytes_count[31:0]

STAT_RX_PACKET_64_BYTES_MSB: 082C

Table 2-206: **STAT_RX_PACKET_64_BYTES_MSB: 082C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_packet_64_bytes_count[47:32]

STAT_RX_PACKET_65_127_BYTES_LSB: 0830

Table 2-207: **STAT_RX_PACKET_65_127_BYTES_LSB: 0830**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_packet_65_127_bytes_count[31:0]

STAT_RX_PACKET_65_127_BYTES_MSB: 0834

Table 2-208: **STAT_RX_PACKET_65_127_BYTES_MSB: 0834**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_packet_65_127_bytes_count[47:32]

STAT_RX_PACKET_128_255_BYTES_LSB: 0838

Table 2-209: **STAT_RX_PACKET_128_255_BYTES_LSB: 0838**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_packet_128_255_bytes_count[31:0]

STAT_RX_PACKET_128_255_BYTES_MSB: 083C

Table 2-210: **STAT_RX_PACKET_128_255_BYTES_MSB: 083C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_packet_128_255_bytes_count[47:32]

STAT_RX_PACKET_256_511_BYTES_LSB: 0840

Table 2-211: **STAT_RX_PACKET_256_511_BYTES_LSB: 0840**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_packet_256_511_bytes_count[31:0]

STAT_RX_PACKET_256_511_BYTES_MSB: 0844

Table 2-212: **STAT_RX_PACKET_256_511_BYTES_MSB: 0844**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_packet_256_511_bytes_count[47:32]

STAT_RX_PACKET_512_1023_BYTES_LSB: 0848

Table 2-213: **STAT_RX_PACKET_512_1023_BYTES_LSB: 0848**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_packet_512_1023_bytes_count[31:0]

STAT_RX_PACKET_512_1023_BYTES_MSB: 084C

Table 2-214: **STAT_RX_PACKET_512_1023_BYTES_MSB: 084C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_packet_512_1023_bytes_count[47:32]

STAT_RX_PACKET_1024_1518_BYTES_LSB: 0850

Table 2-215: **STAT_RX_PACKET_1024_1518_BYTES_LSB: 0850**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_packet_1024_1518_bytes_count[31:0]

STAT_RX_PACKET_1024_1518_BYTES_MSB: 0854

Table 2-216: **STAT_RX_PACKET_1024_1518_BYTES_MSB: 0854**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_packet_1024_1518_bytes_count[47:32]

STAT_RX_PACKET_1519_1522_BYTES_LSB: 0858

Table 2-217: **STAT_RX_PACKET_1519_1522_BYTES_LSB: 0858**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_packet_1519_1522_bytes_count[31:0]

STAT_RX_PACKET_1519_1522_BYTES_MSB: 085C

Table 2-218: **STAT_RX_PACKET_1519_1522_BYTES_MSB: 085C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_packet_1519_1522_bytes_count[47:32]

STAT_RX_PACKET_1523_1548_BYTES_LSB: 0860

Table 2-219: **STAT_RX_PACKET_1523_1548_BYTES_LSB: 0860**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_packet_1523_1548_bytes_count[31:0]

STAT_RX_PACKET_1523_1548_BYTES_MSB: 0864

Table 2-220: **STAT_RX_PACKET_1523_1548_BYTES_MSB: 0864**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_packet_1523_1548_bytes_count[47:32]

STAT_RX_PACKET_1549_2047_BYTES_LSB: 0868

Table 2-221: **STAT_RX_PACKET_1549_2047_BYTES_LSB: 0868**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_packet_1549_2047_bytes_count[31:0]

STAT_RX_PACKET_1549_2047_BYTES_MSB: 086C

Table 2-222: **STAT_RX_PACKET_1549_2047_BYTES_MSB: 086C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_packet_1549_2047_bytes_count[47:32]

STAT_RX_PACKET_2048_4095_BYTES_LSB: 0870

Table 2-223: **STAT_RX_PACKET_2048_4095_BYTES_LSB: 0870**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_packet_2048_4095_bytes_count[31:0]

STAT_RX_PACKET_2048_4095_BYTES_MSB: 0874

Table 2-224: **STAT_RX_PACKET_2048_4095_BYTES_MSB: 0874**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_packet_2048_4095_bytes_count[47:32]

STAT_RX_PACKET_4096_8191_BYTES_LSB: 0878

Table 2-225: **STAT_RX_PACKET_4096_8191_BYTES_LSB: 0878**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_packet_4096_8191_bytes_count[31:0]

STAT_RX_PACKET_4096_8191_BYTES_MSB: 087C

Table 2-226: **STAT_RX_PACKET_4096_8191_BYTES_MSB: 087C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_packet_4096_8191_bytes_count[47:32]

STAT_RX_PACKET_8192_9215_BYTES_LSB: 0880

Table 2-227: **STAT_RX_PACKET_8192_9215_BYTES_LSB: 0880**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_packet_8192_9215_bytes_count[31:0]

STAT_RX_PACKET_8192_9215_BYTES_MSB: 0884

Table 2-228: **STAT_RX_PACKET_8192_9215_BYTES_MSB: 0884**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_packet_8192_9215_bytes_count[47:32]

STAT_RX_PACKET_LARGE_LSB: 0888

Table 2-229: **STAT_RX_PACKET_LARGE_LSB: 0888**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_packet_large_count[31:0]

STAT_RX_PACKET_LARGE_MSB: 088C

Table 2-230: **STAT_RX_PACKET_LARGE_MSB: 088C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_packet_large_count[47:32]

STAT_RX_PACKET_SMALL_LSB: 0890

Table 2-231: **STAT_RX_PACKET_SMALL_LSB: 0890**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_packet_small_count[31:0]

STAT_RX_PACKET_SMALL_MSB: 0894

Table 2-232: **STAT_RX_PACKET_SMALL_MSB: 0894**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_packet_small_count[47:32]

STAT_RX_UNDERSIZE_LSB: 0898

Table 2-233: **STAT_RX_UNDERSIZE_LSB: 0898**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_undersize_count[31:0]

STAT_RX_UNDERSIZE_MSB: 089C

Table 2-234: **STAT_RX_UNDERSIZE_MSB: 089C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_undersize_count[47:32]

STAT_RX_FRAGMENT_LSB: 08A0

Table 2-235: **STAT_RX_FRAGMENT_LSB: 08A0**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_fragment_count[31:0]

STAT_RX_FRAGMENT_MSB: 08A4

Table 2-236: **STAT_RX_FRAGMENT_MSB: 08A4**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_fragment_count[47:32]

STAT_RX_OVERSIZE_LSB: 08A8

Table 2-237: **STAT_RX_OVERSIZE_LSB: 08A8**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_oversize_count[31:0]

STAT_RX_OVERSIZE_MSB: 08AC

Table 2-238: **STAT_RX_OVERSIZE_MSB: 08AC**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_oversize_count[47:32]

STAT_RX_TOOLONG_LSB: 08B0

Table 2-239: **STAT_RX_TOOLONG_LSB: 08B0**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_toolong_count[31:0]

STAT_RX_TOOLONG_MSB: 08B4

Table 2-240: **STAT_RX_TOOLONG_MSB: 08B4**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_toolong_count[47:32]

STAT_RX_JABBER_LSB: 08B8

Table 2-241: **STAT_RX_JABBER_LSB: 08B8**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_jabber_count[31:0]

STAT_RX_JABBER_MSB: 08BC

Table 2-242: **STAT_RX_JABBER_MSB: 08BC**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_jabber_count[47:32]

STAT_RX_BAD_FCS_LSB: 08C0

Table 2-243: **STAT_RX_BAD_FCS_LSB: 08C0**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_bad_fcs_count[31:0]

STAT_RX_BAD_FCS_MSB: 08C4

Table 2-244: **STAT_RX_BAD_FCS_MSB: 08C4**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_bad_fcs_count[47:32]

STAT_RX_PACKET_BAD_FCS_LSB: 08C8

Table 2-245: **STAT_RX_PACKET_BAD_FCS_LSB: 08C8**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_packet_bad_fcs_count[31:0]

STAT_RX_PACKET_BAD_FCS_MSB: 08CC

Table 2-246: **STAT_RX_PACKET_BAD_FCS_MSB: 08CC**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_packet_bad_fcs_count[47:32]

STAT_RX_STOMPED_FCS_LSB: 08D0

Table 2-247: **STAT_RX_STOMPED_FCS_LSB: 08D0**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_stomped_fcs_count[31:0]

STAT_RX_STOMPED_FCS_MSB: 08D4

Table 2-248: **STAT_RX_STOMPED_FCS_MSB: 08D4**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_stomped_fcs_count[47:32]

STAT_RX_UNICAST_LSB: 08D8

Table 2-249: **STAT_RX_UNICAST_LSB: 08D8**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_unicast_count[31:0]

STAT_RX_UNICAST_MSB: 08DC

Table 2-250: **STAT_RX_UNICAST_MSB: 08DC**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_unicast_count[47:32]

STAT_RX_MULTICAST_LSB: 08E0

Table 2-251: **STAT_RX_MULTICAST_LSB: 08E0**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_multicast_count[31:0]

STAT_RX_MULTICAST_MSB: 08E4

Table 2-252: **STAT_RX_MULTICAST_MSB: 08E4**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_multicast_count[47:32]

STAT_RX_BROADCAST_LSB: 08E8

Table 2-253: **STAT_RX_BROADCAST_LSB: 08E8**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_broadcast_count[31:0]

STAT_RX_BROADCAST_MSB: 08EC

Table 2-254: **STAT_RX_BROADCAST_MSB: 08EC**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_broadcast_count[47:32]

STAT_RX_VLAN_LSB: 08F0

Table 2-255: **STAT_RX_VLAN_LSB: 08F0**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_vlan_count[31:0]

STAT_RX_VLAN_MSB: 08F4

Table 2-256: **STAT_RX_VLAN_MSB: 08F4**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_vlan_count[47:32]

STAT_RX_PAUSE_LSB: 08F8

Table 2-257: **STAT_RX_PAUSE_LSB: 08F8**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_pause_count[31:0]

STAT_RX_PAUSE_MSB: 08FC

Table 2-258: **STAT_RX_PAUSE_MSB: 08FC**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_pause_count[47:32]

STAT_RX_USER_PAUSE_LSB: 0900

Table 2-259: **STAT_RX_USER_PAUSE_LSB: 0900**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_user_pause_count[31:0]

STAT_RX_USER_PAUSE_MSB: 0904

Table 2-260: **STAT_RX_USER_PAUSE_MSB: 0904**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_user_pause_count[47:32]

STAT_RX_INRANGEERR_LSB: 0908

Table 2-261: **STAT_RX_INRANGEERR_LSB: 0908**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_inrangeerr_count[31:0]

STAT_RX_INRANGEERR_MSB: 090C

Table 2-262: **STAT_RX_INRANGEERR_MSB: 090C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_inrangeerr_count[47:32]

STAT_RX_TRUNCATED_LSB: 0910

Table 2-263: **STAT_RX_TRUNCATED_LSB: 0910**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_truncated_count[31:0]

STAT_RX_TRUNCATED_MSB: 0914

Table 2-264: **STAT_RX_TRUNCATED_MSB: 0914**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_truncated_count[47:32]

STAT_RX_TEST_PATTERN_MISMATCH_LSB: 0918

Table 2-265: **STAT_RX_TEST_PATTERN_MISMATCH_LSB: 0918**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_test_pattern_mismatch_count[31:0]

STAT_RX_TEST_PATTERN_MISMATCH_MSB: 091C

Table 2-266: **STAT_RX_TEST_PATTERN_MISMATCH_MSB: 091C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_test_pattern_mismatch_count[47:32]

STAT_FEC_INC_CORRECT_COUNT_LSB: 0920

Table 2-267: **STAT_FEC_INC_CORRECT_COUNT_LSB: 0920**

Bits	Default	Type	Signal
31:0	0	HIST	stat_fec_inc_correct_count_count[31:0]

STAT_FEC_INC_CORRECT_COUNT_MSB: 0924

Table 2-268: **STAT_FEC_INC_CORRECT_COUNT_MSB: 0924**

Bits	Default	Type	Signal
15:0	0	HIST	stat_fec_inc_correct_count_count[47:32]

STAT_FEC_INC_CANT_CORRECT_COUNT_LSB: 0928

Table 2-269: **STAT_FEC_INC_CANT_CORRECT_COUNT_LSB: 0928**

Bits	Default	Type	Signal
31:0	0	HIST	stat_fec_inc_cant_correct_count_count[31:0]

STAT_FEC_INC_CANT_CORRECT_COUNT_MSB: 092C

Table 2-270: **STAT_FEC_INC_CANT_CORRECT_COUNT_MSB: 092C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_fec_inc_cant_correct_count_count[47:32]

STAT_TX_MM_STATUS_LSB: 0x0980

Table 2-271: **STAT_TX_MM_STATUS_LSB: 0x0980**

Bits	Default	Type	Signal
31:0	0	HIST	stat_tx_mm_status[31:0]

STAT_TX_MM_STATUS_MSB: 0x0984

Table 2-272: **STAT_TX_MM_STATUS_MSB: 0x0984**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_mm_status[47:32]

STAT_TX_MM_FRAGMENT_LSB: 0x0988

Table 2-273: **STAT_TX_MM_STATUS_LSB: 0x0988**

Bits	Default	Type	Signal
31:0	0	HIST	stat_tx_mm_fragment[31:0]

STAT_TX_MM_FRAGMENT_MSB: 0x098C

Table 2-274: **STAT_TX_MM_FRAGMENT_MSB: 0x098C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_mm_fragment[47:32]

STAT_TX_MM_HOLD_LSB: 0x0990

Table 2-275: **STAT_TX_MM_HOLD_LSB: 0x0990**

Bits	Default	Type	Signal
31:0	0	HIST	stat_tx_mm_hold[31:0]

STAT_TX_MM_HOLD_MSB: 0x0994

Table 2-276: **STAT_TX_MM_HOLD_MSB: 0x0994**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_mm_hold[47:32]

STAT_RX_MM_ASSEMBLY_ERROR_LSB: 0x0998

Table 2-277: **STAT_RX_MM_ASSEMBLY_ERROR_LSB: 0x0998**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_mm_assembly_error[31:0]

STAT_RX_MM_ASSEMBLY_ERROR_MSB: 0x099C

Table 2-278: **STAT_RX_MM_ASSEMBLY_ERROR_MSB: 0x099C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_mm_assembly_error[47:32]

STAT_RX_MM_FRAME_SMD_ERROR_LSB: 0x09A0

Table 2-279: **STAT_RX_MM_FRAME_SMD_ERROR_LSB: 0x09A0**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_mm_frame_smd_error[31:0]

STAT_RX_MM_FRAME_SMD_ERROR_MSB: 0x09A4

Table 2-280: **STAT_RX_MM_FRAME_SMD_ERROR_MSB: 0x09A4**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_mm_frame_smd_error[47:32]

STAT_RX_MM_FRAME_ASSEMBLY_OK_LSB: 0x09A8

Table 2-281: **STAT_RX_MM_FRAME_ASSEMBLY_OK_LSB: 0x09A8**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_mm_frame_assembly_ok[31:0]

STAT_RX_MM_FRAME_ASSEMBLY_OK_MSB: 0x09AC

Table 2-282: **STAT_RX_MM_FRAME_ASSEMBLY_OK_MSB: 0x09AC**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_mm_frame_assembly_ok[47:32]

STAT_RX_MM_FRAGMENT_LSB: 0x09B0

Table 2-283: **STAT_RX_MM_FRAGMENT_LSB: 0x09B0**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_mm_fragment[31:0]

STAT_RX_MM_FRAGMENT_MSB: 0x09B4

Table 2-284: **SSTAT_TX_MM_FRAGMENT_MSB: 0x09B4**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_mm_fragment[47:32]

Designing with the Subsystem

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

Clocking

This section describes the clocking for all the 10G/25G configurations at the component support wrapper layer. There are five fundamentally different clocking architectures depending on the functionality and options:

- PCS/PMA Only Clocking
- 10G/25G MAC with PCS/PMA Clocking
- Low Latency 10G/25G MAC with PCS/PMA Clocking
- Low Latency 32-bit 10 Gb/s MAC with PCS
- 10G MAC-only Clocking

Also described is Auto-Negotiation and Link Training Clocking.

PCS/PMA Only Clocking

The clocking architecture for the 10G/25G PCS is illustrated below. There are three clock domains in the datapath, as illustrated by the dashed lines in Figure 3-1.

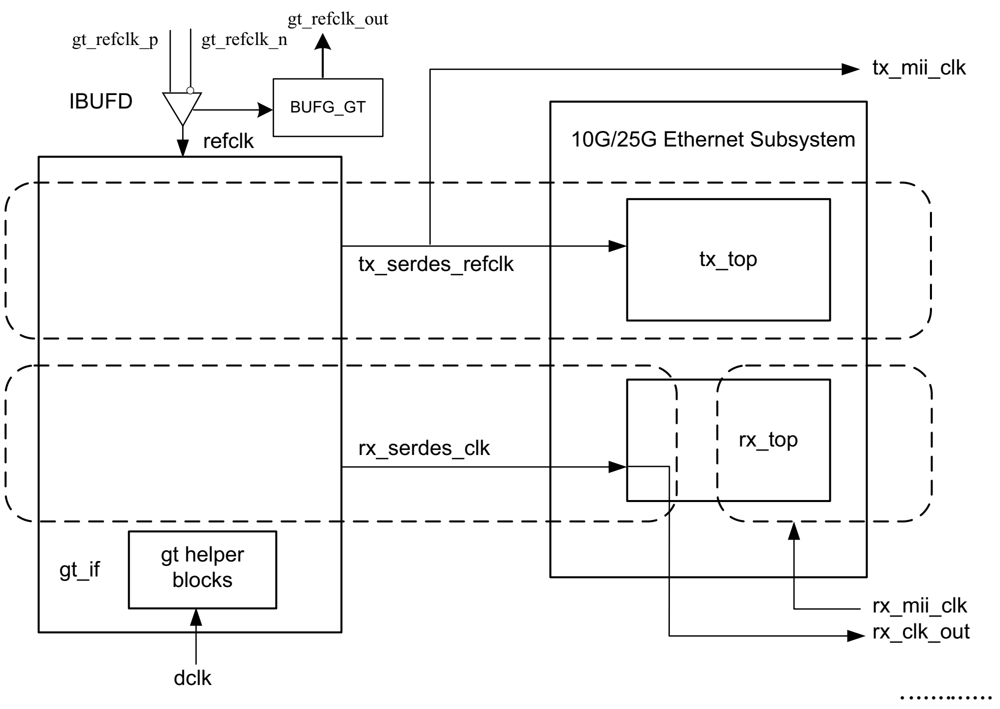


Figure 3-1: **PCS/PMA Clocking**

refclk_p0, refclk_n0, tx_serdes_refclk

The `refclk` differential pair is required to be an input to the FPGA. The example design includes a buffer to convert this clock to a single-ended signal `refclk`, which is used as the reference clock for the GT block. The `tx_serdes_refclk` is directly derived from `refclk`. Note that `refclk` must be chosen so that the `tx_mii_clk` meets the requirements of 802.3, which is within 100 ppm of 390.625 MHz for 25G and 156.25 MHz for 10G.

tx_mii_clk

The `tx_mii_clk` is an output which is the same as the `tx_serdes_refclk`. The entire TX path is driven by this clock. You must synchronize the TX path `mii` bus to this clock output. All TX control and status signals are referenced to this clock.

rx_serdes_clk

The `rx_serdes_clk` is derived from the incoming data stream within the GT block. The incoming data stream is processed by the RX core in this clock domain.

rx_clk_out

The `rx_clk_out` output signal is presented as a reference for the RX control and status signals processed by the RX core. It is the same frequency as the `rx_serdes_clk`.

rx_mii_clk

The `rx_mii_clk` input is required to be synchronized to the RX XGMII/25GMII data bus. This clock and the RX XGMII/25GMII bus must be within 100 ppm of the required frequency, which is 390.625 MHz for 25G and 156.25 MHz for 10G.

dclk

The `dclk` signal must be a convenient, stable clock. It is used as a reference frequency for the GT helper blocks which initiate the GT itself. In the example design, a typical value is 75 MHz, which is readily derived from the 300 MHz clock available on the VCU107 evaluation board. Note that the actual frequency must be known to the GT helper blocks for proper operation.

10G/25G MAC with PCS/PMA Clocking

The clocking architecture for the 10/25G MAC with PCS/PMA clocking is illustrated below. This version of the subsystem includes FIFOs in the RX. There are three clock domains in the datapath, as illustrated by the dashed lines in Figure 3-2.

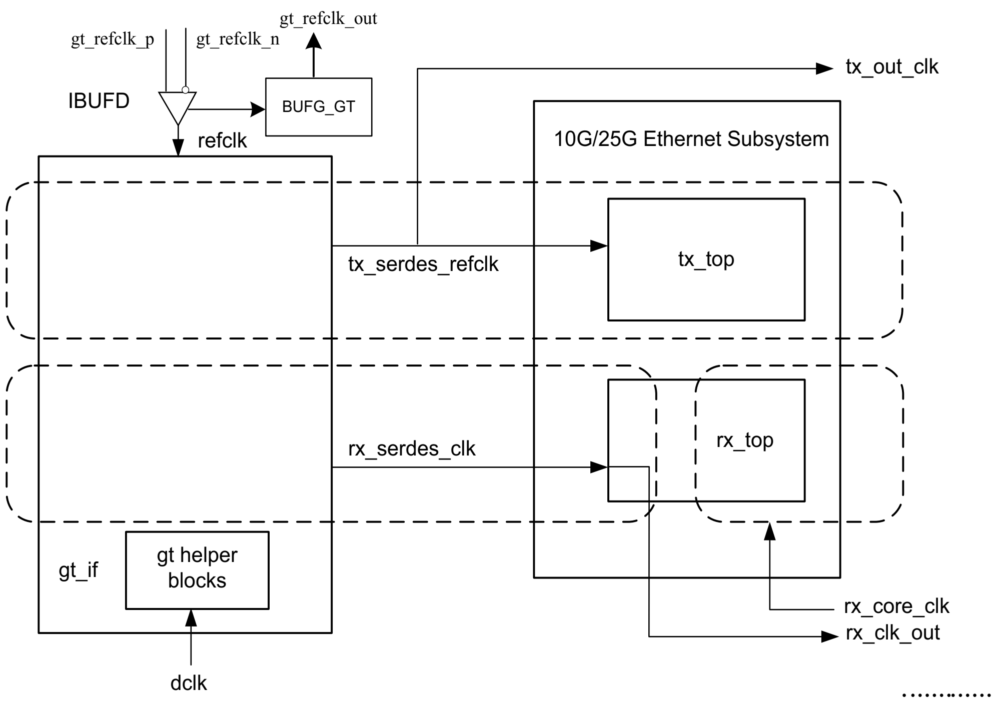


Figure 3-2: **10G/25G MAC with PCS/PMA Clocking**

refclk_p0, refclk_n0, tx_serdes_refclk

The `refclk` differential pair is required to be an input to the FPGA. The example design includes a buffer to convert this clock to a single-ended signal `refclk`, which is used as the reference clock for the GT block. The `tx_serdes_refclk` is directly derived from `refclk`. Note that `refclk` must be chosen so that the `tx_serdes_refclk` meets the requirements of 802.3, which is within 100 ppm of 390.625 MHz for 25G, 156.25 MHz for 64-bit 10G, and 312.5Mhz for 32-bit 10G.

tx_clk_out

This clock is used for clocking data into the TX AXI4-Stream Interface and it is also the reference clock for the TX control and status signals. It is the same frequency as `tx_serdes_refclk`.

rx_clk_out

The `rx_clk_out` output signal is presented as a reference for the RX control and status signals processed by the RX core. It is the same frequency as the `rx_serdes_clk`.

rx_clk

The `rx_clk` is the input clk for RX core. This `rx_clk` is available as `rx_core_clk` to you, which you must drive from the example design. You can drive the `rx_core_clk` with any frequency that must be equal to or greater than the `tx_clk`. When FIFO is enabled, the most preferred mode of operation for system side datapath is to connect the `tx_clk_out` to `rx_core_clk`. When connected in this manner, the RX AXI4-Stream Interface and the TX AXI4-Stream Interface are on the same clock domain.

dclk

The `dclk` signal must be a convenient stable clock. It is used as a reference frequency for the GT helper blocks which initiate the GT itself. In the example design, a typical value is 75 MHz, which is readily derived from the 300 MHz clock available on the VCU107 evaluation board.

Note: The actual frequency must be known to the GT helper blocks for proper operation.

Low Latency 10G/25G MAC with PCS/PMA Clocking

The clocking architecture for the Low Latency 10/25G MAC with PCS/PMA clocking is illustrated in Figure 3-3. Low latency is achieved by omitting the RX FIFOs, which results in different clocking arrangement. There are two clock domains in the datapath, as illustrated by the dashed lines in Figure 3-3.

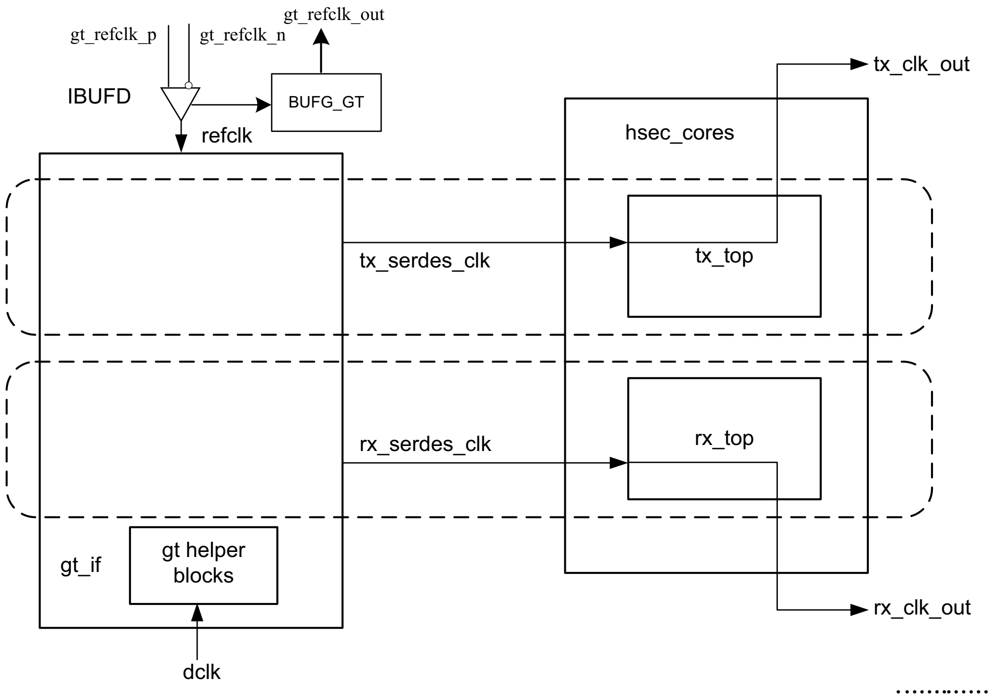


Figure 3-3: **Low Latency 10G/25G MAC with PCS/PMA Clocking**

refclk_p0, refclk_n0, tx_serdes_refclk

The `refclk` differential pair is required to be an input to the FPGA. The example design includes a buffer to convert this clock to a single-ended signal `refclk`, which is used as the reference clock for the GT block. The `tx_serdes_refclk` is directly derived from `refclk`. Note that `refclk` must be chosen so that the `tx_serdes_refclk` meets the requirements of 802.3, which is within 100 ppm of 390.625 MHz for 25G, and 156.25 MHz for 10G.

tx_clk_out

This clock is used for clocking data into the TX AXI4-Stream Interface and it is also the reference clock for the TX control and status signals. It is the same frequency as `tx_serdes_refclk`. Because there is no TX FIFO, you must respond immediately to the `tx_axis_tready` signal.

rx_clk_out

The `rx_clk_out` output signal is presented as a reference for the RX control and status signals processed by the RX core. It is the same frequency as the `rx_serdes_clk`. Because there is no RX FIFO, this is also the clock which drives the RX AXI4-Stream Interface. In this arrangement, `rx_clk_out` and `tx_clk_out` are different frequencies and have no defined phase relationship to each other.

dclk

The `dclk` signal must be a convenient stable clock. It is used as a reference frequency for the GT helper blocks which initiate the GT itself. In the example design, a typical value is 75 MHz, which is readily derived from the 300 MHz clock available on the VCU107 evaluation board. Note that the actual frequency must be known to the GT helper blocks for proper operation.

Low Latency 32-bit 10 Gb/s MAC with PCS

The clocking architecture is identical to that of its 64-bit counterpart, except that the clock to the AXI4-Stream interface will now be 312.5 MHz. Refer to the section on 10G/25G MAC with PCS/PMA Clocking clocking for more details on the clocking architecture.

10G MAC-only Clocking

The clocking architecture for the 10G MAC-only configuration is shown in Figure 3-4. There are three clock domains as illustrated by the dashed lines.

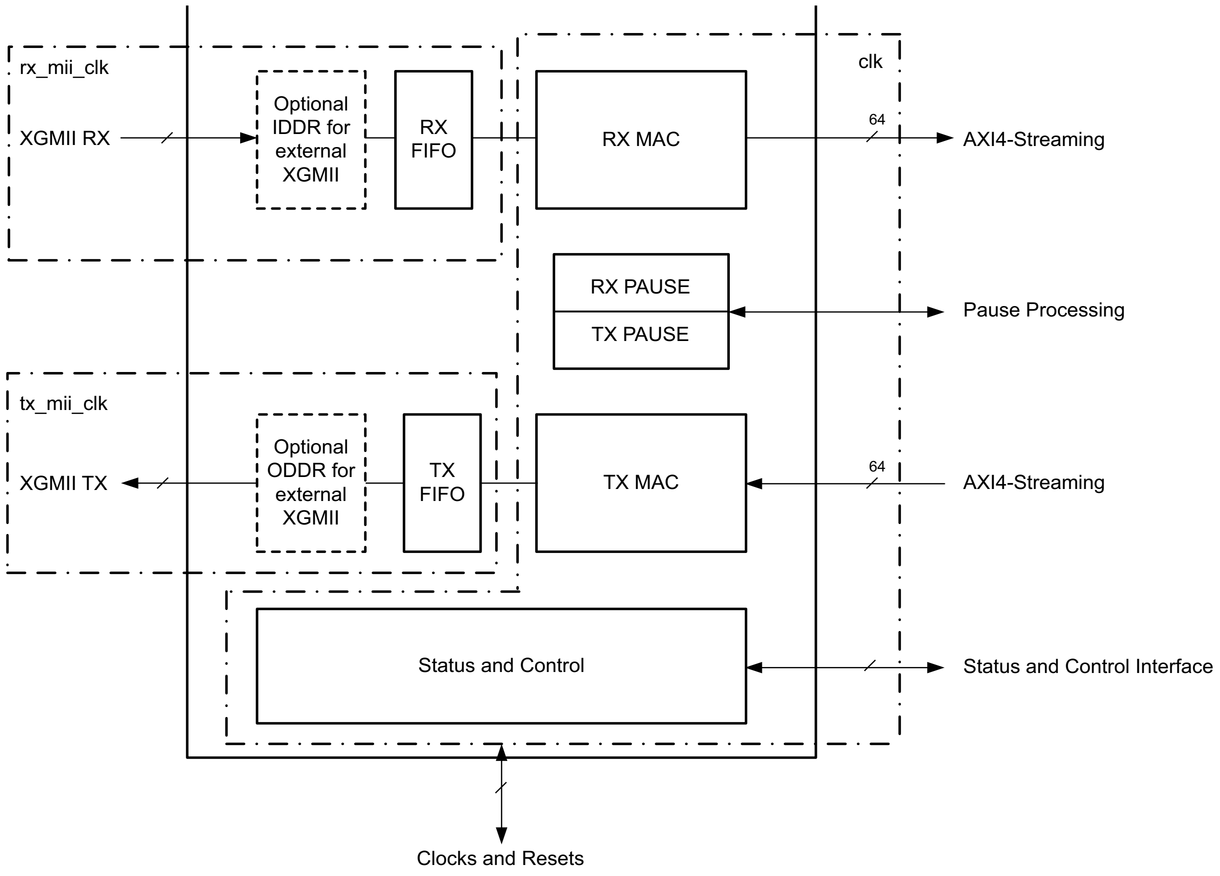


Figure 3-4: **Clocking Architecture for the 10G MAC-only Configuration**

rx_mii_clk

The `rx_mii_clk` can be driven internally or externally. It is required that the clock be chosen to meet the IEEE 802.3 requirements of 156.25 MHz ± 100 ppm for 10 Gb/s operation.

tx_mii_clk

The `tx_mii_clk` can be driven internally or externally. It is required that the clock be chosen to meet the IEEE 802.3 requirements of 156.25 MHz ± 100 ppm for 10 Gb/s operation.

clk

The clock `clk` drives all the internal RX and TX core logic including the AXI4-Stream interface and control and status signals. The clock `clk` should be run at a frequency greater than or equal to 156.25 MHz.

Auto-Negotiation and Link Training Clocking

The clocking architecture for the Auto-Negotiation and Link Training blocks are illustrated in Figure 3-5. Note that these blocks are not included unless the BASE-KR feature is selected. The Auto-Negotiation and Link Training blocks function independently from the MAC and PCS, and therefore they are on different clock domains.

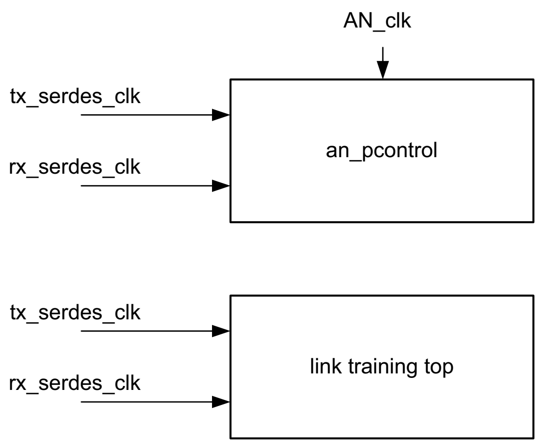


Figure 3-5: **Auto-Negotiation and Link Training Clocking**

tx_serdes_clk

The `tx_serdes_clk` drives the TX line side logic for the Auto-Negotiation and Link Training. The DME frame is generated on this clock domain.

rx_serdes_clk

The `rx_serdes_clk` drives the RX line side logic for the Auto-Negotiation and Link Training.

AN_clk

The `AN_clk` drives the Auto-Negotiation state machine. All ability signals are on this clock domain. The `AN_clk` can be any convenient frequency. In the example design, `AN_clk` is connected to the `dclk` input, which has a typical frequency of 75 MHz. The `AN_clk` frequency must be known to the Auto-Negotiation state machine because it is the reference for all timers.

Resets

Figure 3-6 shows the reset structure for the 10G/25G Ethernet MAC with PCS/PMA as implemented at the component support wrapper layer. Clocks are not shown for clarity.

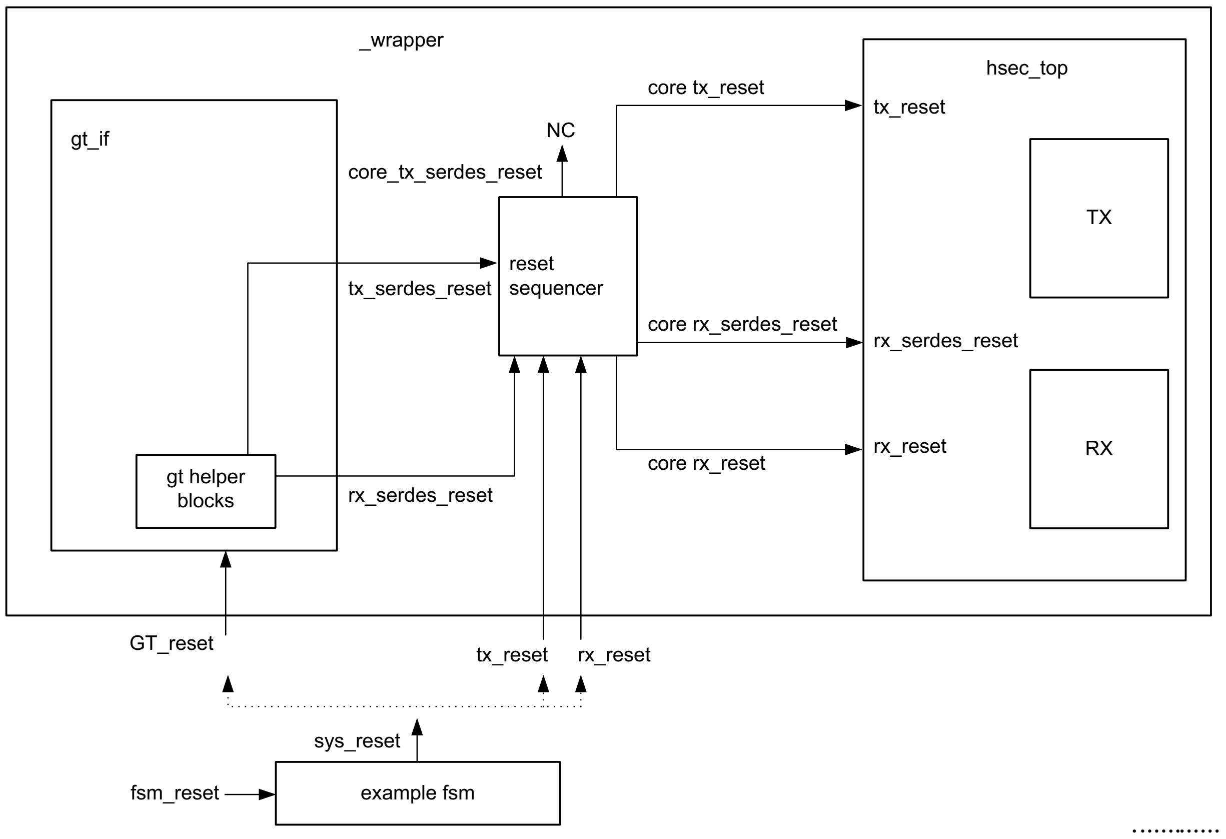


Figure 3-6: **Reset Structure**

Component Support Layer Resets

In the example design, a single reset is used to reset the entire wrapper layer. Using the external stimulus `fsm_reset`, the example_fsm block issues the signal `sys_reset` which is connected to the three `_wrapper` resets. Therefore, the example design demonstrates that all three wrapper resets can be released simultaneously and correct operation follows.

Wrapper Resets

The _wrapper layer of the hierarchy is assumed to be what you instantiate in your own design. There are three resets to be handled as follows:

- GT_reset
- tx_reset
- rx_reset

You do not need to be concerned with timing the reset signals; this is taken care of by the reset_sequencer block.

GT_reset

The `GT_reset` is the asynchronous active-High reset input to the GT. You do not need to be concerned with the internal resets of the GT because this is taken care of by the GT helper blocks.

tx_reset

The `tx_reset` is the asynchronous active-High reset for the TX path logic of the 10G/25G Ethernet IP core. While it is connected to the GT reset in the example design, this reset can be asserted at any time to reset the TX path independently without disturbing the RX path.

rx_reset

The `rx_reset` is the asynchronous active-High reset for the RX path logic of the 10G/25G Ethernet IP core. While it is connected to the GT reset in the example design, this reset can be asserted at any time to reset the RX path independently without disturbing the TX path.

LogiCORE Example Design Clocking and Resets

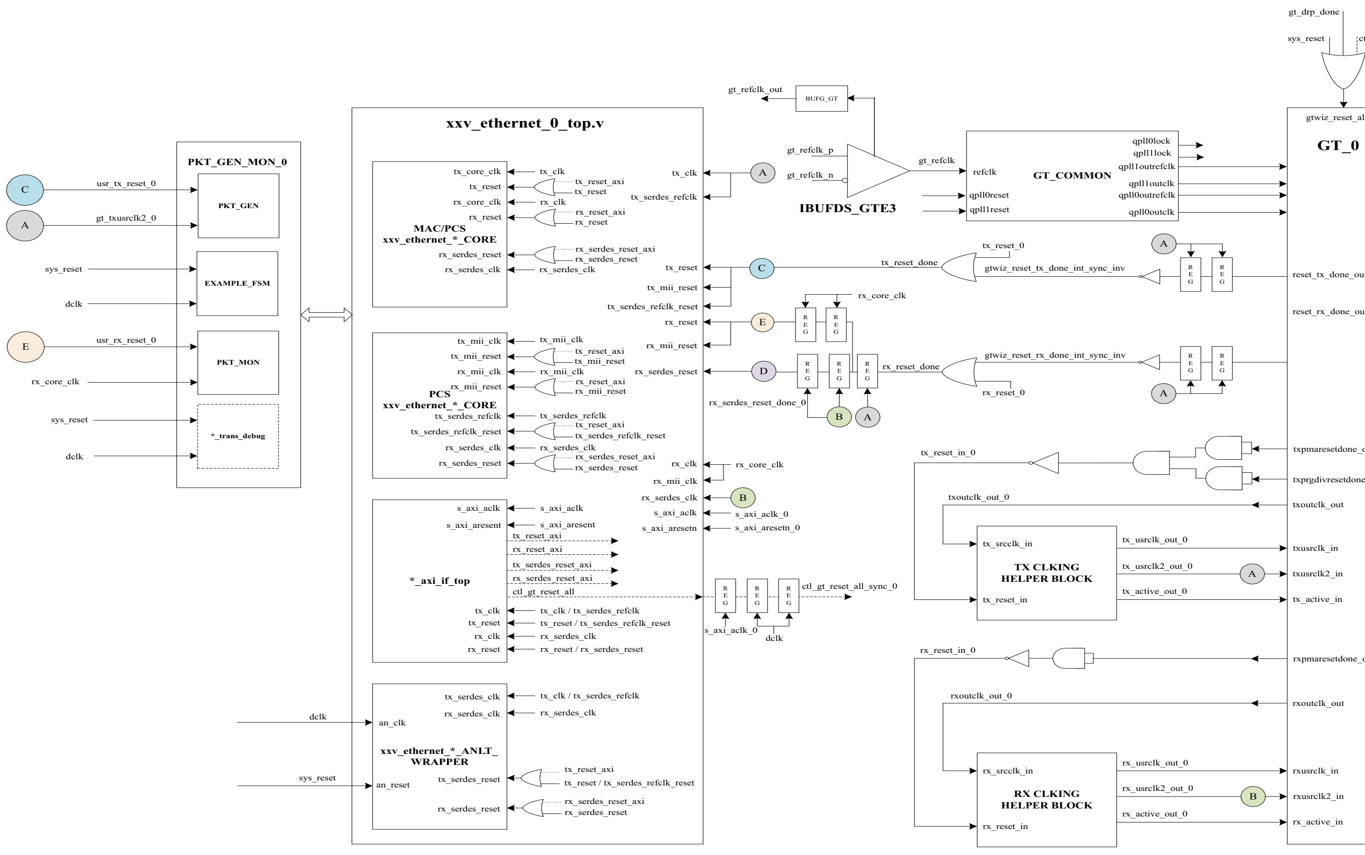


Figure 3-7: **Detailed Diagram of Single Core - Asynchronous Clock Mode**

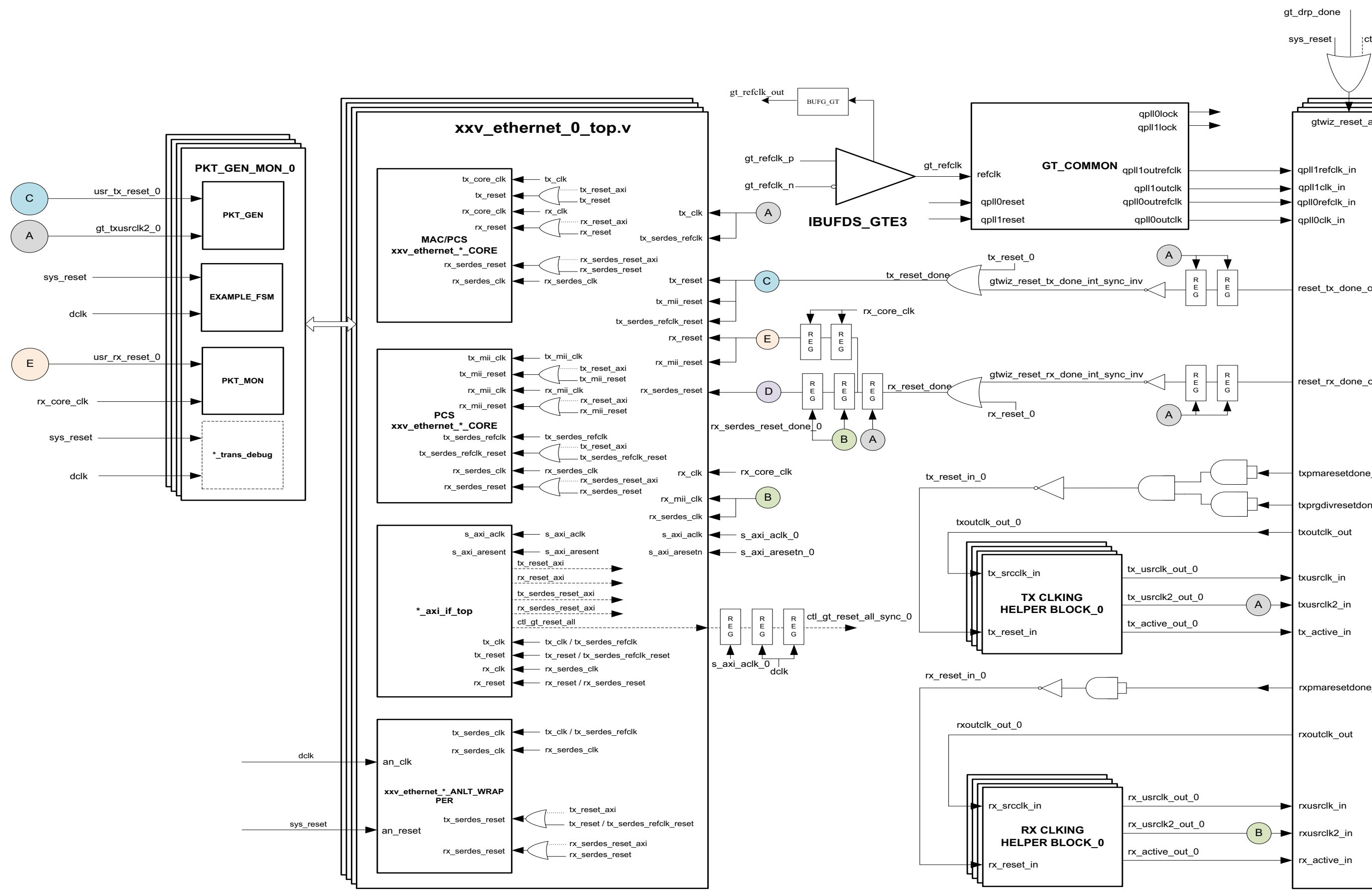


Figure 3-8: **Detailed Diagram of Multiple Cores - Asynchronous Clock Mode**

Support for IEEE Standard 1588v2

Overview

This section details the packet timestamping function of the 10G/25G Ethernet Subsystem when the MAC layer is included. The timestamping option must be specified at the time of generating the subsystem from the IP catalog or ordering the IP Core asynchronously. This feature provides one-step and two-step IEEE 1588v2 functionality.

If you select the IEEE PTP 1588v2 operation mode as "Two Step", then only the two-step related ports will be populated and the core will perform only the two-step time stamping functionality. If you select the IEEE PTP 1588v2 operation mode as "One Step", then all the ports related to one-step and two-step will be populated and both one-step and two-step core functionality will be available.

Ethernet frames are timestamped at both ingress and egress. The option can be used for implementing all kinds of IEEE 1588v2 clocks: Ordinary, Transparent, and Boundary. It can also be used for the generic timestamping of packets at the ingress and egress ports of a system. While this feature can be used for a variety of packet timestamping applications, the rest of this section assumes that you are also implementing the IEEE 1588v2 Precision Time Protocol (PTP).

IEEE 1588v2 defines a protocol for performing timing synchronization across a network. A 1588 network has a single master clock timing reference, usually selected through a best master clock algorithm. Periodically, this master samples its system timer reference counter, and transmits this sampled time value across the network using defined packet formats. This timer should be sampled (a timestamp) when the start of a 1588 timing packet is transmitted. Therefore, to achieve high synchronization accuracy over the network, accurate timestamps are required. If this sampled timer value (the timestamp) is placed into the packet that triggered the timestamp, this is known as one-step operation. Alternatively, the timestamp value can be placed into a follow up packet; this is known as two-step operation.

Other timing slave devices on the network receive these timing reference packets from the network timing master and attempt to synchronize their own local timer references to it. This mechanism relies on these Ethernet ports also taking timestamps (samples of their own local timer) when the 1588 timing packets are received. Further explanation of the operation of 1588 is out of scope of this document. This document now describes the 1588 hardware timestamping features of the subsystem.

The 1588 timer provided to the subsystem and the consequential timestamping taken from it are available in one of two formats which are selected during subsystem generation.

- Time-of-Day (ToD) format: IEEE 1588-2008 format consisting of an unsigned 48-bit second field and a 32-bit nanosecond field.

- Correction Field format: IEEE 1588-2008 numerical format consisting of a 64-bit signed field representing nanoseconds multiplied by 2^{16} (see IEEE 1588 clause 13.3.2.7). This timer should count from 0 through the full range up to 2^{64} -1 before wrapping around.

Egress

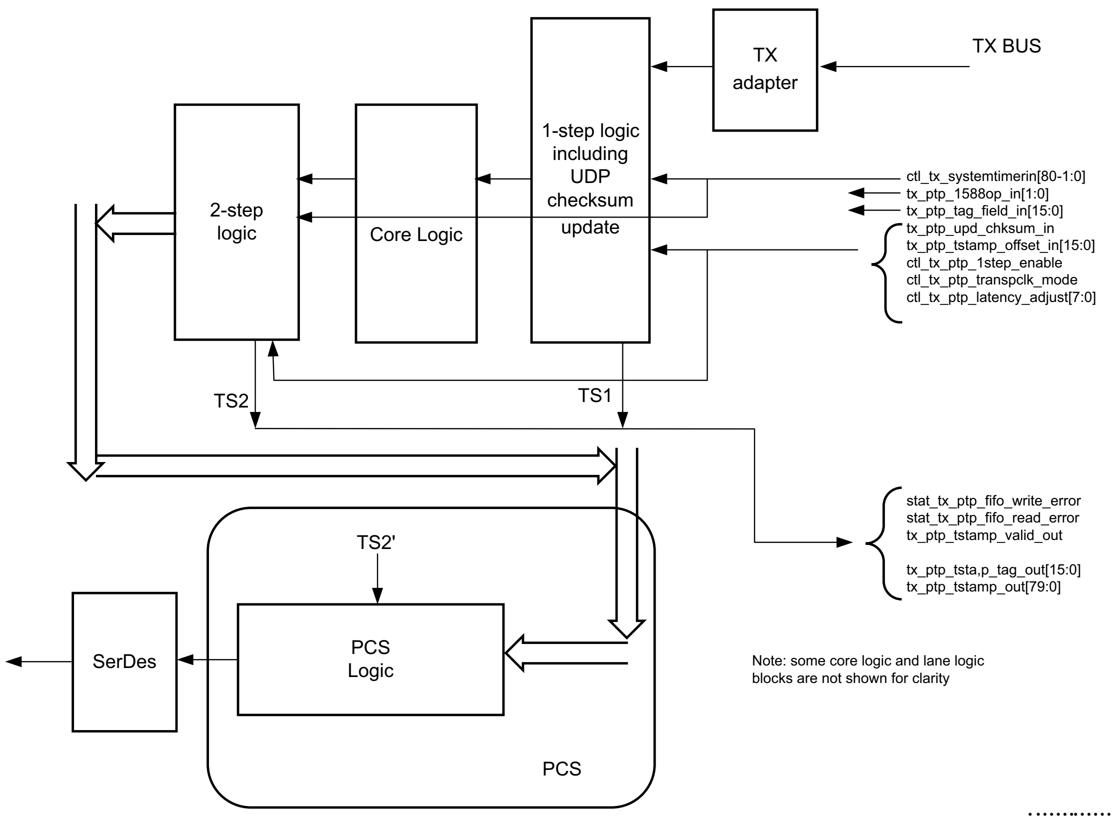

TX BUS
TX adapter
1-step logic including UDP checksum update
Core Logic
2-step logic
ctl_tx_systemtimerin[80-1:0]
tx_ptp_1588op_in[1:0]
tx_ptp_tag_field_in[15:0]
tx_ptp_upd_chksum_in
tx_ptp_tstamp_offset_in[15:0]
ctl_tx_ptp_1step_enable
ctl_tx_ptp_transpclk_mode
ctl_tx_ptp_latency_adjust[7:0]
TS2
TS1
stat_tx_ptp_fifo_write_error
stat_tx_ptp_fifo_read_error
tx_ptp_tstamp_valid_out
tx_ptp_tsta,p_tag_out[15:0]
tx_ptp_tstamp_out[79:0]
TS2'
PCS Logic
SerDes
PCS
Note: some core logic and lane logic blocks are not shown for clarity

Figure 3-9: **Egress**

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

Based on rate and clock mode (Ordinary or Transparent), the suggested default values of the `ctl_tx_ptp_latency_adjust[10:0]` signal are as follows:

- 25G Ordinary Clock = 370
- 25G Transparent Clock = 449
- 10G Ordinary Clock = 925
- 10G Transparent Clock = 1132

On the transmit side, a command field is provided by the client to the subsystem in parallel with the frame sent for transmission. This indicates, on a frame-by-frame basis, the 1588 function to perform (either no-operation, 1-step, or 2-step) and also indicates, for 1-step frames, whether there is a UDP checksum field to update.

If using the ToD format, then for both 1-step and 2-step operation, the full captured 80-bit ToD timestamp is returned to the client logic using the additional ports defined in Table 3-1.

If using the Correction Field format, then for both 1-step and 2-step operation, the full captured 64-bit timestamp is returned to the client logic using the additional ports defined in Table 3-1 (with the upper bits of data set to zero as defined in the table).

If using the ToD format, then for 1-step operation, the full captured 80-bit ToD timestamp is inserted into the frame. If using the Correction Field format, then for 1-step operation, the captured 64-bit timestamp is summed with the existing Correction Field contained within the frame and the summed result is overwritten into the original Correction Field of the frame. Supported frame types for 1-step timestamping are:

- Raw Ethernet
- UDP/IPv4
- UDP/IPv6

For 1-step UDP frame types, the UDP checksum is updated in accordance with IETF RFC 1624. For all 1-step frames, the Ethernet Frame Check Sequence (FCS) field is calculated after all frame modifications have been completed. For 2-step transmit operation, all Precision Time Protocol (PTP) frame types are supported.

Frame-by-Frame Timestamping Operation

The operational mode of the egress timestamping function is determined by the settings on the 1588 command port. The information contained within the command port indicates one of the following:

- No operation: the frame is not a PTP frame and no timestamp action should be taken.
- Two-step operation is required and a tag value (user-sequence ID) is provided as part of the command field; the frame should be timestamped, and the timestamp made available to the client logic, along with the provided tag value for the frame. The additional MAC transmitter ports provide this function.
- 1-step operation is required.
    - For the ToD timer and timestamp format a timestamp offset value is provided as part of the command port; the frame should be timestamped, and the timestamp should be inserted into the frame at the provided offset (number of bytes) into the frame.
    - For the Correction Field format, a Correction Field offset value is provided as part of the command port; the frame should be timestamped, and the captured 64-bit Timestamp is summed with the existing Correction Field contained within the frame and the summed result is overwritten into original Correction Field of the frame.

      For 1-step operation, following the frame modification, the cyclic redundancy check (CRC) value of the frame should also be updated/recalculated. For UDP IPv4 and IPv6 PTP formatted frames, the checksum value in the header of the frame needs to be updated/recalculated.
- For 1-step UDP frame types, the UDP checksum is updated in accordance with IETF RFC 1624.
    - If using the ToD format, in order for this update function to work correctly, the original checksum value for the frame sent for transmission should be calculated using a zero value for the timestamp data. This particular restriction does not apply when using the Correction Field format.
    - If using the Correction Field format, a different restriction does apply; the separation between the UDP Checksum field and the Correction Field within the 1588 PTP frame header is a fixed interval of bytes, supporting the 1588 PTP frame definition. This is a requirement to minimize the latency through the MAC since both the checksum and the correction field must both be fully contained in the MAC pipeline in order for the checksum to be correctly updated. This particular restriction does not apply to the ToD format because the original timestamp data is calculated as a zero value; consequently the checksum and timestamp position can be independently located within the frame.

Ingress

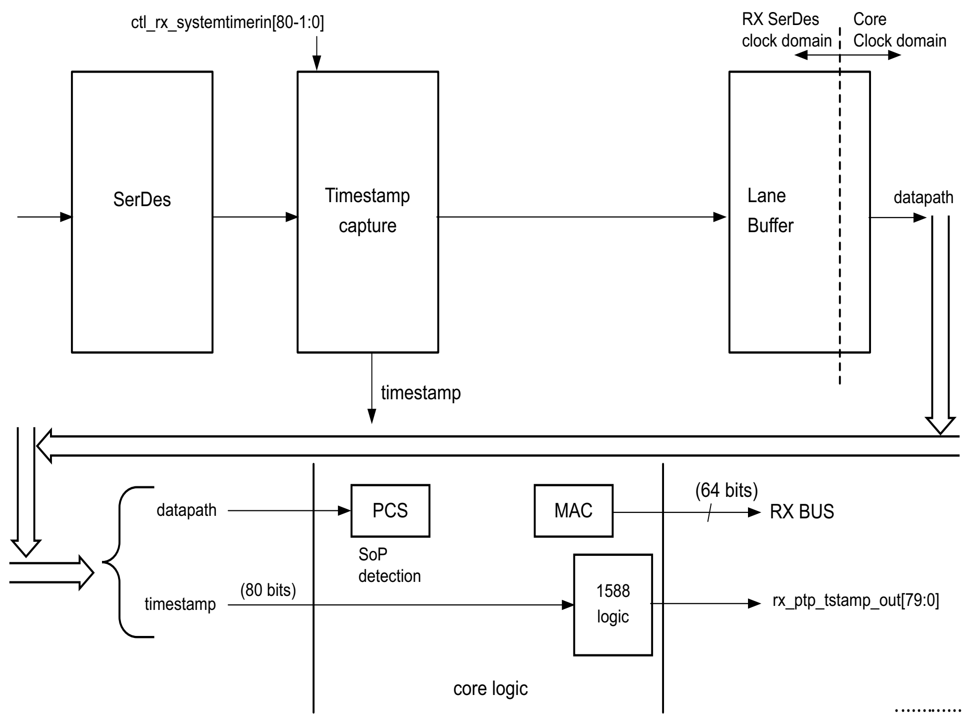


Figure 3-10: **Ingress**

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 if you do not require this function.

Timestamps are filtered after the PCS decoder to retain only those timestamps corresponding to an Start of Packet (SoP). These 80-bit timestamps are output on the system side. The timestamp is valid during the SoP cycle and when ena_out = 1.

1588 time stamp accuracy for different configurations:

- 25G/10G Ordinary Clock: 6 ns
- 25G/10G Transparent clock: 7ns
- 25G Ordinary clock with RSFEC: 9ns
- 25G Transparent clock with RSFEC: 10ns

Port Descriptions

The following table details the additional signals present when the packet timestamping feature is included.

Table 3-1: **1588v2 Port List and Descriptions**

Name	**Direction**	**Description**	**Clock Domain**
		IEEE 1588 Interface – TX Path	
ctl_tx_systemtimerin[80-1:0]	Input	System timer input for the TX. In normal clock mode, the 32 LSBs carry nsec and the 48 MSBs carry seconds. 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 SerDes clock domain.	tx_serdes_clk
tx_ptp_tstamp_valid_out	Output	This bit indicates that a valid timestamp is being presented on the TX system interface.	tx_clk_out
tx_ptp_tstamp_tag_out[15:0]	Output	Tag output corresponding to tx_ptp_tag_field_in[15:0]	tx_clk_out
tx_ptp_tstamp_out[80-1:0]	Output	Timestamp for the transmitted packet SOP corresponding to the time at which it passed the capture plane. Used for 2-step 1588 operation. Time format is the same as timer input.	tx_clk_out
tx_ptp_1588op_in[1:0]	Input	The signal should be valid on the first cycle of the packet. For PCS cores, the first cycle corresponds with the first data word of the packet. 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_clk_out
ctl_tx_ptp_1step_enable	Input	When set to 1, this bit enables 1-step operation.	tx_clk_out
tx_ptp_upd_chksum_in		See tx_ptp_upd_chksum_in_* in Table 5-2.	
tx_ptp_chksum_offset_in		See tx_ptp_chksum_offset_in_* in Table 5-2	

Table 3-1: **1588v2 Port List and Descriptions *(Cont'd)***

Name	Direction	Description	Clock Domain
ctl_ptp_transpclk_mode	Input	When set to 1, this input places the timestamping logic into transparent clock mode. In this mode, the system timer input is interpreted as a correction value. The TX will add the correction value to the TX timestamp according to the process defined in IEEE 1588v2. The sign bit of the correction value is assumed to be 0 (positive time). It is expected that the corresponding incoming PTP packet correction field has already been adjusted with the proper RX timestamp.	tx_clk_out
tx_ptp_tag_field_in[15:0]	Input	The usage of this field is dependent on the 1588 operation. The signal should be valid on the first cycle of the packet. • 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 PTP frame that it sent for transmission.	tx_clk_out
ctl_tx_ptp_latency_adjust[10:0]	Input	This bus can be used to adjust the 1-step TX timestamp with respect to the 2-step timestamp. The units of bits [10:3] are nanoseconds and bits [2:0] are fractional nanoseconds.	tx_clk_out
stat_tx_ptp_fifo_write_error	Output	Transmit PTP FIFO write error. A value of 1 on this status indicates that an error occurred during the PTP Tag write. A TX Path reset is required to clear the error.	tx_clk_out
stat_tx_ptp_fifo_read_error	Output	Transmit PTP FIFO read error. A value of 1 on this status indicates that an error occurred during the PTP Tag read. A TX Path reset is required to clear the error.	tx_clk_out
IEEE 1588 Interface – RX Path			
ctl_rx_systemtimerin[80-1:0]	Input	System timer input for the RX. Same time format as the TX. This input must be in the same clock domain as the RX SerDes.	rx_serdes_clk
rx_ptp_tstamp_out[80-1:0]	Output	Timestamp for the received packet SOP corresponding to the time at which it passed the capture plane. Note that this signal will be valid on the first cycle of the packet. Used for 2-step 1588 operation.	rx_core_clk

Table 3-1: **1588v2 Port List and Descriptions *(Cont'd)***

Name	Direction	Description	Clock Domain
ctl_tx_latency_*	Input	This is the static latency of the TX path of the core including the GT. The MSB 16 bits indicate the delay in ns and the LSB 16 bits indicate sub ns values. The latency is in binary Q16.16 format. ***Note:*** This is applicable only for the MAC+PCS/PMA 32-bit version clock domain: tx_clk_out.	
ctl_tx_lat_adj_enb_*	Input	When this signal is enabled, the delay computation on the TX path takes into account the value provided by the ctl_tx_latency_0 register. ***Note:*** This is applicable only for the MAC+PCS/PMA 32-bit version clock domain: tx_clk_out.	
ctl_rx_latency_0	Input	This is the static latency of the RX path of the core including the GT. The MSB 16 bits indicate the delay in ns and the LSB 16 bits indicate sub ns values. The latency is in binary Q16.16 format. ***Note:*** This is applicable only for the MAC+PCS/PMA 32-bit version clock domain: rx_clk_out.	
ctl_rx_lat_adj_enb_0 (default value: 1)	Input	When this signal is enabled, the delay computation on the RX path takes into account the value provided by the ctl_rx_latency_0 register. ***Note:*** This is applicable only for MAC+PCS/PMA 32-bit version clock domain: rx_clk_out.	
ctl_tx_timestamp_adj_enb_0 (default value: 1)	Input	When this signal is enabled, the delay computation on the TX path takes into account the value obtained from GT DRP read for latency of the TX gearbox FIFO. ***Note:*** This is applicable only for the MAC+PCS/PMA 32-bit version clock domain: tx_clk_out.	
ctl_rx_timestamp_adj_enb_0 (default value: 1)	Input	When this signal is enabled, the delay computation on the RX path takes into account the value obtained from the GT DRP read for latency of the RX gearbox FIFO. ***Note:*** This is applicable only for the MAC+PCS/PMA 32-bit version clock domain: rx_clk_out	

IEEE 1588v2 PTP Functional Description

The IEEE 1588v2 feature of the 10G/25G High Speed Ethernet Subsystem provides accurate timestamping of Ethernet frames at the hardware level for both the ingress and egress directions.

Timestamps are captured according to the input clock source above. However, it is required that this time source be in the same clock domain as the SerDes. This might require re-timing by an external circuit provided by the user.

All ingress frames receive a timestamp. It is up to you to interpret the received frames and determine whether a particular frame contains PTP information (by means of its Ethertype) and if the timestamp needs to be retained or discarded.

Egress frames are timestamped if they are tagged as PTP frames. The timestamps of egress frames are matched to their user-supplied tags.

Timestamps for incoming frames are presented at the user interface during the same clock cycle as the start of packet. You can then append the timestamp to the packet as required.

By definition, a timestamp is captured coincident with the passing of the SOP through the capture plane within the 10G/25G High Speed Ethernet Subsystem. This is illustrated in the following schematic diagrams:

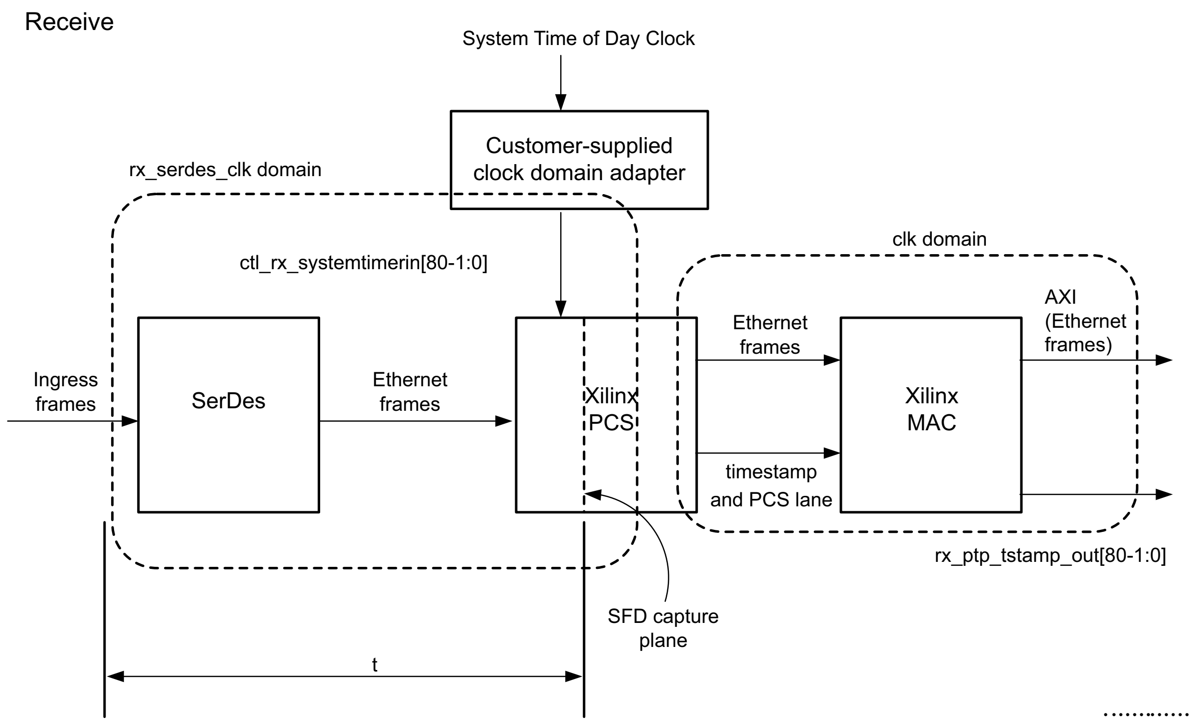


Figure 3-11: **Receive**

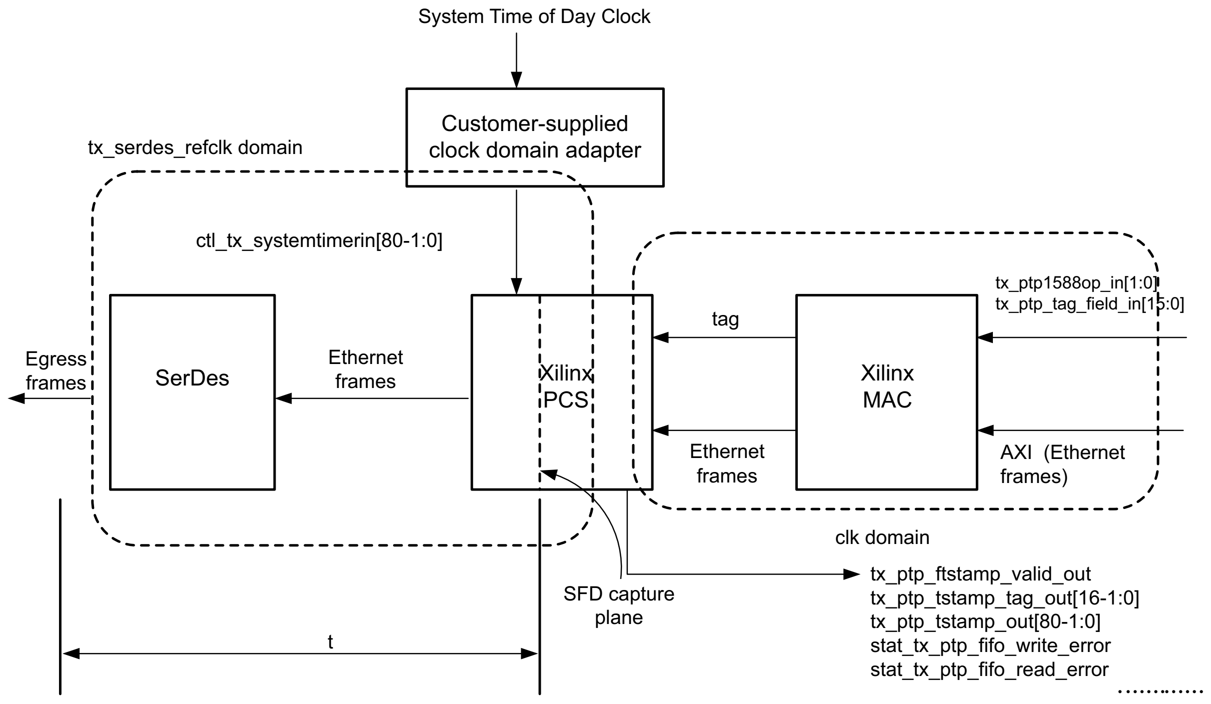


Figure 3-12: **Transmit**

Performance

In a typical application, the difference between the ingress and egress capture times is important for determining absolute time. The PTP algorithm can use asymmetric information to improve accuracy.

The 1588v2 feature requires that all clock frequencies be known in order to make internal calculations. The clock frequencies should be specified at the time the PTP IP core is ordered in order for the timestamp correction to work properly.

In a typical application, the PTP algorithm (or servo, not part of this IP) will remove jitter over the course of time (many packet samples). It is advantageous for the jitter to be as small as possible in order to minimize the convergence time as well as minimizing slave clock drift.

RS-FEC Support

Overview

This section describes the optional RS-FEC function of the 10G/25G Ethernet Subsystem. The RS-FEC option must be specified at the time of generating the subsystem from the IP catalog or ordering the IP core asynchronously.

With reference to the following diagram, the clocks and resets of the RS-FEC core are equivalent to the transceiver signals, with the transceiver resets being active-High. The RS-FEC block is positioned between the PCS and PMA as illustrated in the following figure.

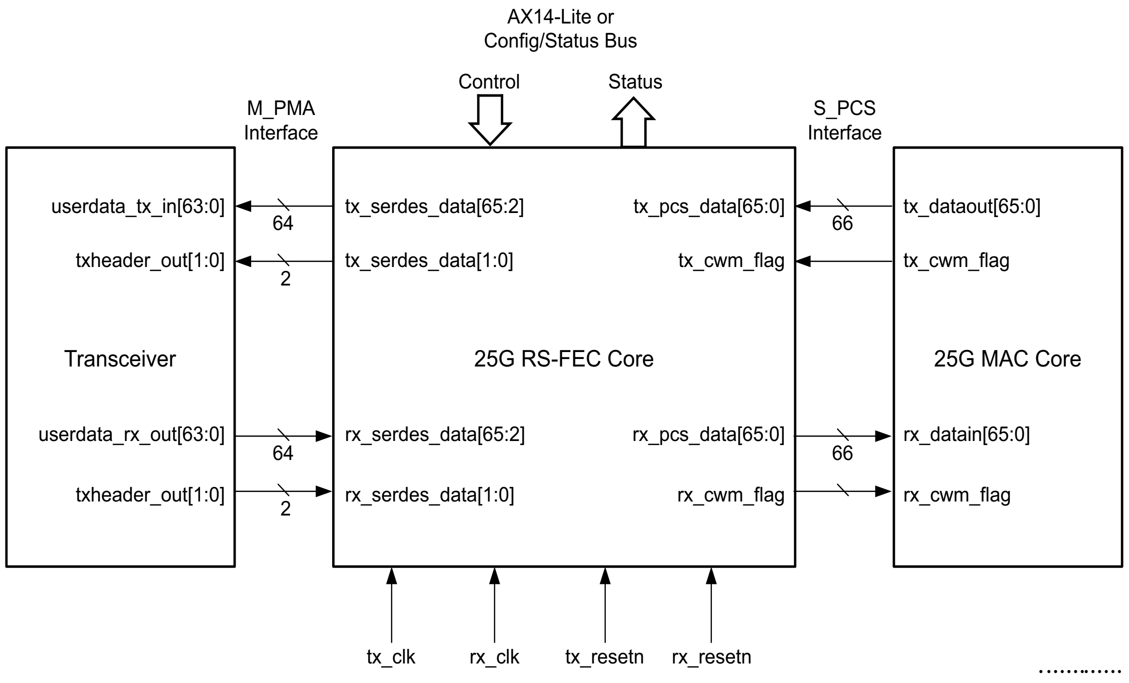


Figure 3-13: **RS-FEC Block Diagram**

The internal details of the RS-FEC are beyond the scope of this document. Refer to IEEE 802.3 Clause 108 and Schedule 3 of the 25G Ethernet Consortium [Ref 1].

Further information can also be found in the *25G IEEE 802.3by Reed-Solomon Forward Error Correction LogiCORE IP Product Guide* (PG217) [Ref 12].

Port Descriptions

Table 3-2: **RS-FEC Port List and Descriptions**

Port	Direction	Description	Clock Domain
RS-FEC Control Signals			
ctl_rsfec_ieee_error_indication_mode	Input	The setting on this bit takes effect after rx_resetn has been asserted Low (~rx_serdes_reset). New value is sampled on first cycle on reset. • 1: Core conforms to the IEEE RS-FEC specification. • 0: If ctl_rx_rsfec_enable_correction and ctl_rx_rsfec_enable_indication are set to zero, the RS decoder is bypassed.	rx_serdes_clk
ctl_rsfec_consortium_25g	Input	Switches between IEEE Clause 108 and 25G Ethernet Consortium mode. The setting on this bit takes effect after rx_resetn has been asserted Low (~rx_serdes_reset). New value is sampled on first cycle on reset. • 1 = 25G Consortium specification mode. • 0 = IEEE 802.3by mode. Note that some variants of the 10G/25G Subsystem can have individual RX and TX consortium signals.	rx_serdes_clk
ctl_rsfec_enable	Input	The setting on this bit takes effect after rx_resetn has been asserted Low (~rx_serdes_reset). New value is sampled on first cycle on reset. Enable RS-FEC function. Note that some variants of the 10G/25G Subsystem can have individual RX and TX enable signals.	rx_serdes_clk
ctl_rx_rsfec_enable_correction	Input	The setting on this bit takes effect after rx_resetn has been asserted Low (~rx_serdes_reset). New value is sampled on first cycle on reset. Equivalent to MDIO register 1.200.0 • 0: Decoder performs error detection without error correction (see IEEE 802.3802.3by section 91.5.3.3). • 1: the decoder also performs error correction.	rx_serdes_clk

Table 3-2: **RS-FEC Port List and Descriptions *(Cont'd)***

Port	Direction	Description	Clock Domain
ctl_rx_rsfec_enable_indication	Input	The setting on this bit takes effect after rx_resetn has been asserted Low (~rx_serdes_reset). New value sampled on first cycle on reset. Equivalent to MDIO register 1.200.1 • 0: Bypass the error indication function (see IEEE Std 802.3by section 91.5.3.3). • 1: Decoder indicates errors to the PCS sublayer,	rx_serdes_clk
ctl_rx_vl_length_minus1[15:0]	Input	Normally set to 20,479 (4FFF hex). The normal value is equivalent to (16,383 x 5 -4) = 81,916.	
ctl_rx_vl_marker_id0[63:0]	Input	Equivalent to the RX PCS lane 0 alignment marker defined in IEEE 802.3 Clause 82 for 40 G Ethernet.	
ctl_rx_vl_marker_id1[63:0]	Input	Equivalent to the PCS lane 1 alignment marker.	
ctl_rx_vl_marker_id2[63:0]	Input	Equivalent to the PCS lane 2 alignment marker.	
ctl_rx_vl_marker_id3[63:0]	Input	Equivalent to the PCS lane 3 alignment marker.	
ctl_tx_vl_length_minus1[15:0]	Input	Normally set to 20479 (decimal). The normal value is equivalent to (16,383 x 5 -4) = 81,916.	
ctl_tx_vl_marker_id0[63:0]	Input	Equivalent to the TX PCS lane 0 alignment marker defined in IEEE 802.3 Clause 82 for 40 G Ethernet.	
ctl_tx_vl_marker_id1[63:0]	Input	Equivalent to the PCS lane 1 alignment marker.	
ctl_tx_vl_marker_id2[63:0]	Input	Equivalent to the PCS lane 2 alignment marker.	
ctl_tx_vl_marker_id3[63:0]	Input	Equivalent to the PCS lane 3 alignment marker	
RS-FEC Status Signals			
stat_rx_rsfec_corrected_cw_inc	Output	Increment for corrected errors.	rx_serdes_clk
stat_rx_rsfec_uncorrected_cw_inc	Output	Increment for uncorrected errors.	rx_serdes_clk
stat_rx_rsfec_err_count_inc[2:0]	Output	Increment for detected errors.	rx_serdes_clk
stat_rx_rsfec_hi_ser	Output	Set to one if the number of RS-FEC symbol errors in a window of 8192 codewords exceeds the threshold K = 417 and is set to zero otherwise.	rx_serdes_clk

Table 3-2: **RS-FEC Port List and Descriptions *(Cont'd)***

Port	Direction	Description	Clock Domain
stat_rx_rsfec_lane_alignment_status	Output	A value of 1 indicates that the RX RS-FEC block has achieved alignment on the data from the transceiver.	rx_serdes_clk
stat_tx_rsfec_lane_alignment_status	Output	A value of 1 indicates that the TX RS-FEC block has achieved alignment on the incoming PCS data.	rx_serdes_clk

RS-FEC Functional Description

The RS-FEC feature of the 10G/25G Subsystem provides error correction capability according to IEEE 802.3 Clause 108 or Schedule 3 of the 25G Ethernet Consortium.

The feature requires the insertion of PCS alignment markers as defined in IEEE 802.3 Table 82-2. Inputs are provided for the alignment markers and also for the value of words between alignment markers.

It is possible to bypass the RS-FEC function by means of the enable signals. This will bypass the RS-FEC function and connect the PCS directly to the transceiver, with the benefit of reduced latency. Refer to *25G IEEE 802.3by Reed-Solomon Forward Error Correction LogiCORE IP Product Guide* (PG217) [Ref 12] for the latest latency performance data in the various bypass modes, defined as follows:

- FEC Bypass Correction: The decoder performs error detection without correction, (see IEEE Std 802.3by section 108.5.3.2. The latency is reduced in this mode (see *25G IEEE 802.3by Reed-Solomon Forward Error Correction LogiCORE IP Product Guide* (PG217) [Ref 12] for latency figures).
- FEC Bypass Indication: In this mode there is correction of the data but no error indication. An additional signal, `rx_hi_ser`, is generated in this mode to reduce the likelihood that errors in a packet are not detected. The RS decoder counts the number of symbol errors detected in consecutive non-overlapping blocks of 8192 codewords (see IEEE Std 802.3by section 108.5.3.2). The latency is reduced in this mode.
- Decoder Bypass: The RS decoder can be bypassed by setting the IEEE Error indication Low when the correction bypass and indication bypass are High.

Ethernet Datapath Parity

The **Datapath Parity Feature** provides soft error detection on datapath logic that resides between the core AXI4-Stream interface and the Ethernet FCS logic.

For each byte of the datapath, a single bit is provided that reflects the calculated parity of that byte.

IMPORTANT: *The IP core implements even parity such that the sum of all ones in the datapath including the parity bit results in an even number (i.e., 0, 2, 4..).*

Datapath parity is implemented in both transmit and receive directions of the core datapath. In both directions, parity bits are considered valid only on valid AXI4-Stream bytes. More specifically, an AXI4-Stream byte is considered valid on cycles where `TKEEP[n]`, `TVALID` and `TREADY` are all asserted. Parity bits associated with invalid AXI4-Stream bytes in the receive datapath should be ignored. Parity bits associated with AXI4-Stream invalid bytes in the transmit direction are not checked.

The **Datapath Parity Feature** adds a single statistic, `stat_tx_bad_parity`, to identify if a soft error has been detected. The statistic is clock-cycle based such that it can assert multiple times for a single packet. And, because it is clock cycle based, it identifies only that at least one soft error has been detected. If two soft errors are detected on a single clock cycle, only one soft error will be indicated. The presence of parity errors can also trigger the assertion of `stat_tx_bad_fcs` errors depending on the configuration.

For variants supporting parity, there is no configuration required for the receive datapath as all packets presented on the AXI4-Stream interface will have parity generated. In the transmit direction, the following table describes configuration bits that affect the behavior of the parity logic:

Table 3-3: **Configuration Bits Affecting Parity Logic Behavior**

ctl_tx_fcs_ins_enable	ctl_tx_parity_err_response	Ethernet FCS Stomped Behavior	stat_tx_bad_parity Behavior	stat_tx_bad_fcs Behavior
0	0	Core will not stomp FCS, but FCS may have been stomped user logic.	stat_tx_bad_parity is asserted if parity errors are detected. (1)	stat_tx_bad_fcs is asserted if the user-generated FCS in the incoming packet is incorrect. Parity errors, which may or may not be present, will not cause a stat_tx_bad_fcs assertion.
0	1	Same as above.	Same as above. (1)	Same as above
1	0	Same as above.	Same as above. (1)	Parity errors, which may or may not be present, will not cause a stat_tx_bad_fcs assertion.
1	1	Same as above.	Same as above. (1)	stat_tx_bad_fcs is asserted if parity errors are detected.(2)

Notes:

1. There is a limitation such that `stat_tx_bad_parity` will not be asserted for parity error(s) received on the final clock cycle of an erred packet or if an LBUS FIFO underflow event occurs.
2. There is a limitation such that `stat_tx_bad_fcs` will not be asserted for parity error(s) received on only the final clock cycle of an erred packet or if an LBUS FIFO underflow event occurs. Parity errors received prior to the final clock cycle will still cause `stat_tx_bad_fcs` to be asserted.

802.1cm Preemption Feature

Features

Optional fee-based Time Sensitive Networking (TSN) feature designed to IEEE standard 802.1 CM

- Supports frame preemption
- Supports interspersing express traffic with low priority traffic

Overview

Frame preemption and Express traffic interspersing is defined by IEEE standard 802.1 CM

The 10G/25G Runtime switchable IP core includes the optional TSN feature based on IEEE 802.1 CM.

Product Specification

Figure 3-14 shows the block diagram of the 10/25G Ethernet IP subsystem with the optional TSN feature:

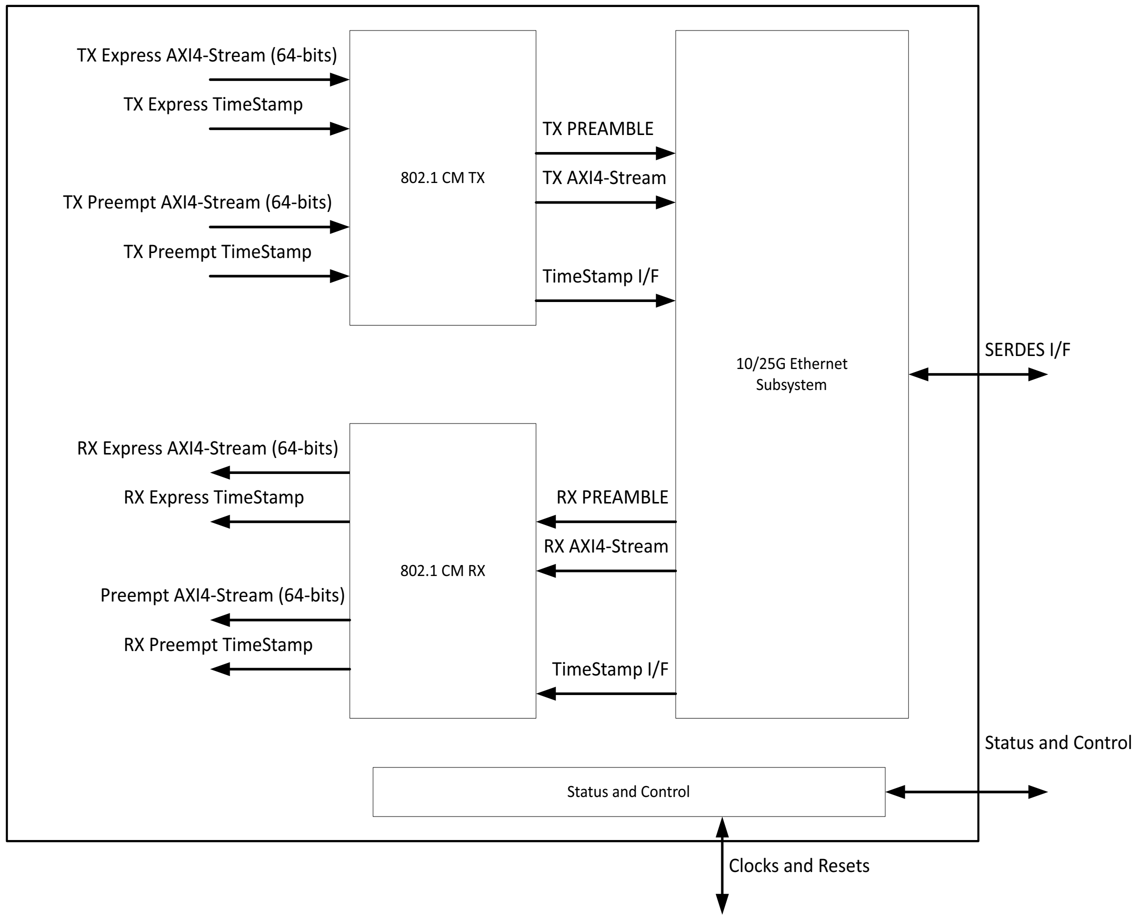


Figure 3-14: **10/25G Ethernet IP Subsystem with Optional TSN**

Transmit AXI4-Stream Interface

The core has two AXI4-Stream Interfaces, for express (tx_axis_e_*) and preempt traffic (tx_axis_p_*), when the core is generated with the optional TSN feature. For details refer to Table 2-4.

Note: The same descriptions and rules apply to these signals as the ones in Table 2-4.

There is an option to insert a FIFO on the preempt interface during core generation. When this FIFO is inserted, the ingress frame will be buffered and only when the complete error free frames is available in this FIFO will it be made available on the AIX4-Stream interface.

Frame Transmission

If you disable preemption, the core services the express and preempt traffic on a first-come-first-serve basis. If both interfaces present frames at the same time, the express traffic is serviced first and then the preempt traffic. For details on presenting a frame for transmission, refer to Normal Transmission in Chapter 2.

Before you enable preemption, the preemption capabilities of the link-partner must be determined first. This is done by the exchange of Additional Ethernet Capabilities TLV as described in 802.3 section 79.3.7. The core assumes that the you will enable preemption only after determining that the link-partner is also capable of preemption.

If preemption is enabled by asserting `ctl_en_preempt`, the core first verifies the preemption operation provided that `ctl_disable_verify` is deasserted.

Preemption will be active only after verification is successfully completed. You can also disable verification in which case the core will not attempt to start the verification process and will make preemption active.

When preemption is active, the core services the frame transmission requests as follows:

- If the express interface is inactive, the frames presented on the preempt interface will be transmitted.
- If the express and preempt interface request frame transmission at the same time, the express frames will be transmitted.
- If the preempt traffic is being transmitted and express interfaces presents a frame, the preempt traffic will be preempted in accordance with this formula specified in IEEE standard 802.3 br-2106:

  pAllow * (eTx + hold) * preemptableFragSize * MIN_REMAIN
- After the express frame has been transmitted and no more express frames are queued-up for transmission, the core resumes the transmission of the preempted frames.

For detailed description of preemption and interspersing of express traffic, refer to IEEE standard 802.1 CM

The timing diagram below depicts how preemption and interspersing works

For detailed description of preemption and interspersing of express traffic, please refer to IEEE standard 802.1 CM

Figure 3-15 depicts how preemption and interspersing works.

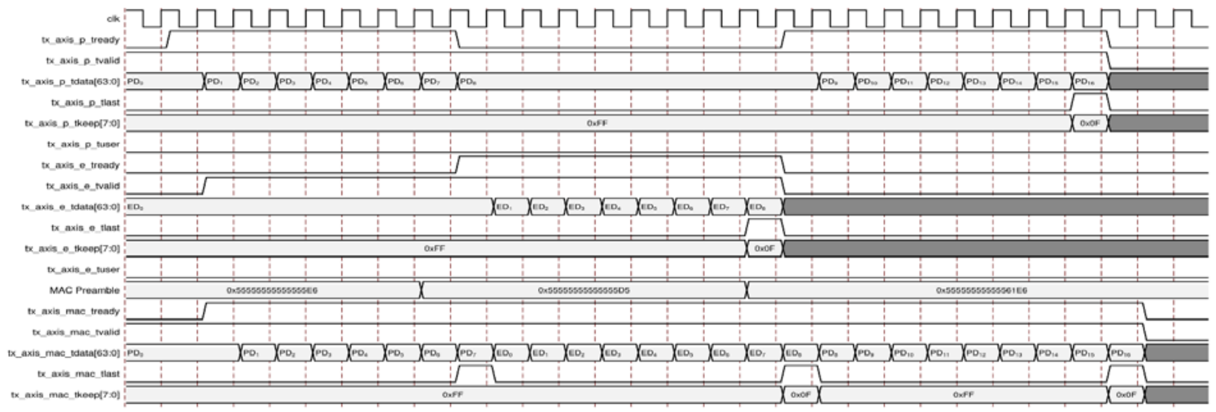


Figure 3-15: **Preemption and Interspersing**

IFG is not depicted in the diagram above for compactness. However there will be IFG which will result in `tx_axis_mac_tready` being deasserted between frames.

Receive AXI4-Stream Interface

Table 2-7 shows the two AXI4-Stream Interfaces, for express (rx_axis_e_*) and preempt traffic (rx_axis_p_*), when the core is generated with the optional TSN feature.

Note: The same descriptions and rules apply to these signals as the ones in Table 2-7.

Frame Reception

The ingress frame can be either preempt or express type. The core determines the type and puts the ingress frame on either the express AXI4-Stream interface or on the preempt AXI4-Stream interface respectively.

Express traffic will be continuous and because the core does not have any buffering mechanism on this interface, you must be ready to accept the express frame at any given time.

Due to the nature of the preempt traffic, the frame can arrive as a set of fragments which will be assembled by the core. There is an option of inserting a FIFO on the preempt interface during core generation. When this FIFO is inserted, all the fragments of the preempt frame will be buffered and only when the assembly process successfully completed, the frame will be made available on the AIX4-Stream interface. If FIFO is not inserted, the `tvalid` on the AXI4-Stream interface can pulsate in-between fragments. Core asserts `tlast` to indicate completion of the fragment assembly process; if the assembly process is not successful, it asserts `tuser` along with `tlast`.

Figure 3-16 shows how the preempt frame fragments are presented on the AXI4-Stream interface when the FIFO is not inserted.

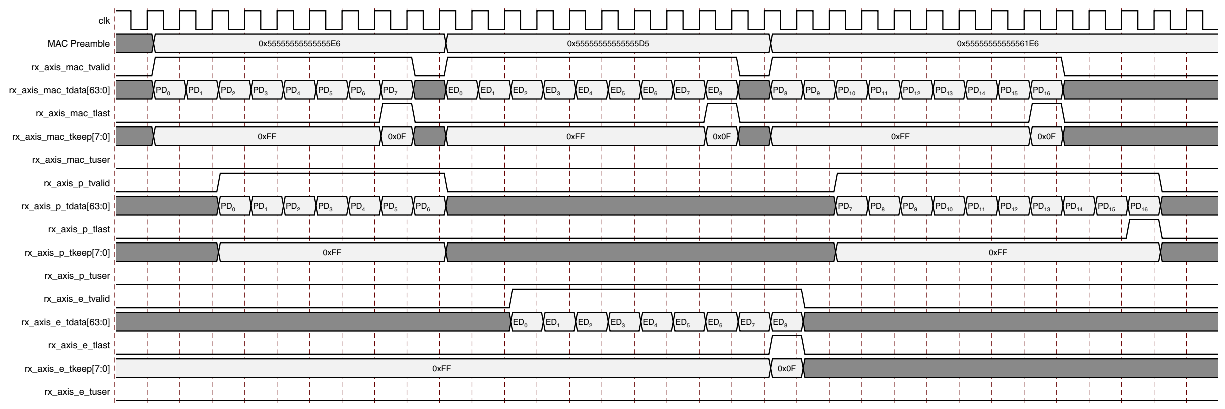


Figure 3-16: **Preempt Frame Fragments: When the FIFO Not Inserted**

Table 3-4: **Control and Status Ports**

Name	Direction	Description	Clock Domain
CONTROL			
ctl_en_preempt	Input	When asserted, it allows preemption. For the very first time it is asserted, it triggers Verification if ctl_disable_verify = 1'b0 and stat_tx_mm_verified[1:0] = 2'b00.	tx_clk_out
ctl_hold_request	Input	If asserted, preempt traffic is withheld.	tx_clk_out
ctl_disable_verify	Input	If asserted, it inhibits the verification process.	tx_clk_out
ctl_restart_verify	Input	A 0-to-1 transition will trigger Verification if ctl_disable_verify = 1'b0.	tx_clk_out
ctl_addfrag_size[1:0]	Input	Fragment size remaining to enable pre-emption: 2'b00 = 64-bytes 2'b01= 128-bytes 2'b10 = 192-bytes 2'b11 = 256-bytes	tx_clk_out
ctl_verify_time[7:0]	Input	Verification time-out value in milliseconds. Integer range 1-128. Default is 1 ms.	tx_clk_out
ctl_verify_limit[3:0]	Input	Number of times core attempts Verification. Integer range 1-15. Default is 3.	tx_clk_out
STATUS			
stat_tx_mm_verify[1:0] (No Counter)	Output	Indicates verification status. [0] – When asserted, indicates that verification is complete. [1] – When asserted, indicates that verification is successful. ***Note:*** The value contained in this status vector is valid only when ctl_en_preempt = 1'b1 and ctl_disable_verify = 1'b0	tx_clk_out
stat_tx_mm_status	Output	Asserted when a preemptable packet (initial fragment or complete packet) is transmitted.	tx_clk_out
stat_tx_mm_fragment	Output	Asserted when a continuation fragment of an preemptable packet is transmitted.	tx_clk_out
stat_tx_mm_hold	Output	Asserted when ctl_hold_request transitions from 1'b0 to 1'b1.	tx_clk_out
stat_rx_mm_assembly_error	Output	Asserted when errors are detected during fragment assembly.	rx_core_clk
stat_rx_mm_frame_smd_error	Output	Asserted when the frame fragment is rejected due to an incorrect SMD value or arriving with SMD-C when no frame is in progress.	rx_core_clk

Table 3-4: **Control and Status Ports *(Cont'd)***

Name	Direction	Description	Clock Domain
stat_rx_mm_frame_assembly_ok	Output	Asserted when all the preemptable frame fragments have been assembled and presented.	rx_core_clk
stat_rx_mm_fragment	Output	Asserted when a fragment frame is received.	rx_core_clk

Notes:

1. Time stamp ports description for express and preempt interface is similar to 1588v2 ports description. For more details see Table 3-1.

Status/Control Interface

The Status/Control interface allows you to set up the 10G/25G Ethernet core configuration and to monitor its status. This sections describes in more detail some of the Status and Control signals.

stat_rx_framing_err and stat_rx_framing_err_valid

These signals are used to keep track of sync header errors. This set of buses is used to keep track of sync header errors. The `stat_rx_framing_err` output 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` is sampled as a 1.

stat_rx_block_lock

This bit indicates that the interface has achieved sync header lock as defined by IEEE Std. 802.3. A value of 1 indicates block lock is achieved.

stat_rx_local_fault

This output is High when `stat_rx_internal_local_fault` or `stat_rx_received_local_fault` is asserted. This is output is level sensitive.

RX Error Status

The 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[1:0]

When this signal is positive, it indicates that the error detection logic has identified mismatches 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 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 for each CRC32 error detected.

stat_rx_bad_code

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

Pause Processing

The 10G/25G High Speed Ethernet Subsystem 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.

CAUTION! *Requesting both global and priority pause packets at the same time results in unpredictable behavior and must be avoided.*

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 10G/25G Ethernet core automatically calculates and adds the FCS to the packet. For priority pause packets the 10G/25G Ethernet 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 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 canceled). The 10G/25G Ethernet core transmits the pause packet immediately after the current packet in flight is completed.

IMPORTANT: *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 10G/25G Ethernet 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 are 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, 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 10G/25G Ethernet 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 one are forwarded to you only if `ctl_rx_forward_control` is set to 1.

2. If step one passes, the packet is checked to determine if it is a global pause packet.
3. If step two 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 are 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 are extracted from the incoming packet.

User Interface

A simple handshaking protocol is used to alert 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 these 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 hard CMAC 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.

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

2. If `ctl_rx_check_ack` input is 1, the 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 core starts counting down the timer.
4. When the timer times out, the core 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 core deems the operation complete after 32 clock cycles.

These steps are demonstrated in Figure 3-17 with each step shown on the waveform.

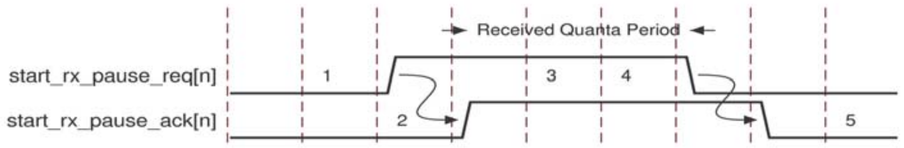


Figure 3-17: **RX Pause Interface Example**

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

Auto-Negotiation

A block diagram of the 10G/25G Ethernet core with Auto-Negotiation (AN) and Link Training (LT) is shown in Figure 3-18.

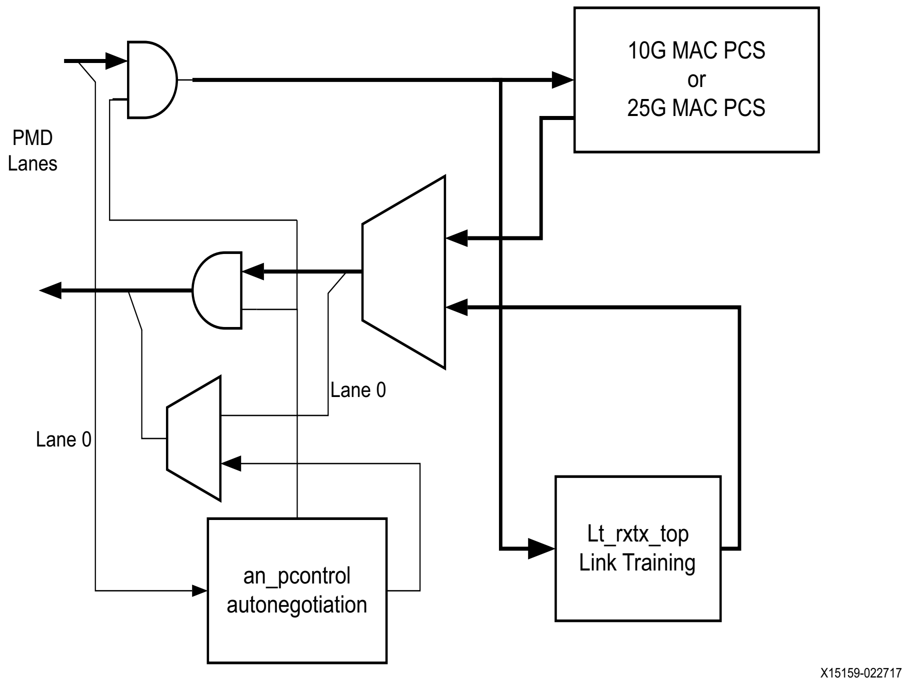


Figure 3-18: **Core with Auto-Negotiation and Link Training**

The auto-negotiation function allows an Ethernet device to advertise the modes of operation it possesses to another device at the remote end of a backplane Ethernet link and to detect corresponding operational modes the other device might be advertising. The objective of this auto-negotiation function is to provide the means to exchange information between two devices and to automatically configure them to take maximum advantage of their abilities. It has the additional objective of supporting a digital signal detect to ensure that the device is attached to a link partner rather than detecting a signal due to crosstalk.

When auto-negotiation is complete, ability is reported according to the available modes of operation.

Link Training is performed after auto-negotiation if the Link Training function is supported by both ends of the link. Link Training is typically required due to frequency-dependent losses which can occur as digital signals traverse the backplane. The primary function of the Link Training block included with this core is to provide register information and a training sequence over the backplane link which is then analyzed by a receiving circuit (part of the transceiver). The other function of the Link Training block is to communicate training feedback from the receiver to the corresponding transmitter so that its equalizer circuit (part of the transceiver) can be adjusted as required. The decision-making algorithm is not part of this core. When auto-negotiation and Link Training are complete, the datapath is switched to mission mode (the PCS), as shown in Figure 3-18.

Overview

Figure 3-19 shows the position of the auto-negotiation function in the OSI reference model.

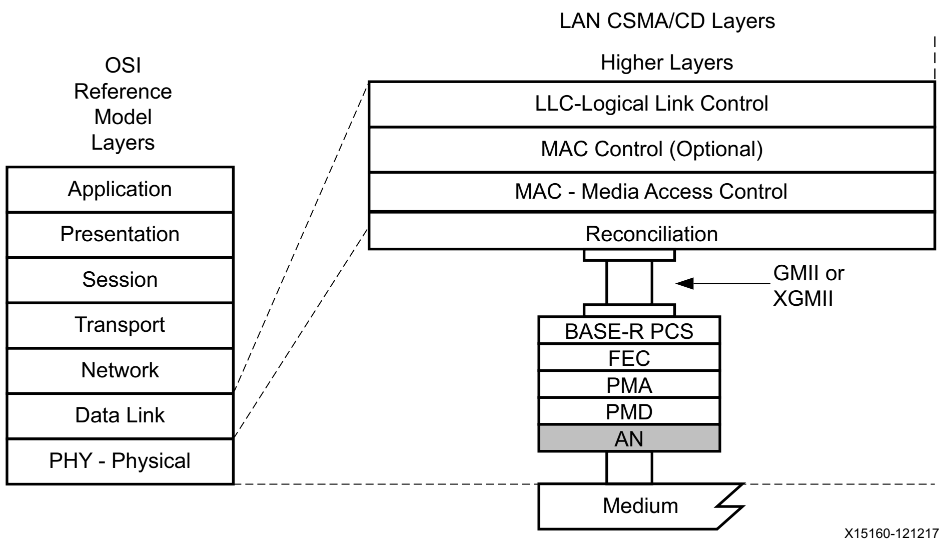


Figure 3-19: **Auto-Negotiation Function in the OSI Model**

The Auto-Negotiation Intellectual Property Core (ANIPC) implements the requirements as specified in Clause 73, IEEE Std 802.3-2015, including those amendments specified in IEEE Std. P802.3ba and 802.3ap. The functions of the ANIPC core are listed in clause 73, specifically Figure 73-11, Arbitration state diagram, in section 73.10.4, State Diagrams.

During normal mission mode operation, with link control outputs set to (bin)11, the bit operating frequency of the transceiver input and output is typically 10.3125 or 25.78125 Gb/s. However, the Dual Manchester Encoding (DME) bit rate used on the lane during Auto-Negotiation is different to the mission mode operation. To accommodate this requirement, the ANIPC core uses over-sampling and over-driving to match the 156.25 Mb/s Auto-Negotiation speed (DME clock frequency 312.5 MHz) with the mission mode 10.3125 or 25.78125 Gb/s physical lane speed.

Functional Description

autoneg_enable

When the autoneg_enable input signal is set to 1, auto-negotiation begins automatically at power-up, or if the carrier signal is lost, or if the input restart_negotiation signal is cycled from a 0 to a 1. All of the Ability input signals as well as the two input signals PAUSE and ASM_DIR are tied Low or High to indicate the capability of the hardware. The nonce_seed[7:0] input must be set to a unique non-zero value for every instance of the auto-negotiator. This is important to guarantee that no dead-locks occur at power-up. If two link partners connected together attempt to auto-negotiate with their nonce_seed[7:0] inputs set to the same value, the auto-negotiation fails continuously. The pseudo_sel input is an arbitrary selection that is used to select the polynomial of the random bit generator used in bit position 49 of the DME pages used during auto-negotiation. Any selection on this input is valid and does not result in adverse behavior.

Link Control

When auto-negotiation begins, the various link control signals are activated, depending on the disposition of the corresponding Ability inputs for those links. Subsequently, the corresponding link status signals are monitored by the ANIPC hardware for an indication of the state of the various links that are connected. If particular links are unused, the corresponding link control outputs are unconnected, and the corresponding link-status inputs should be tied Low. During this time, the ANIPC hardware sets up a communication link with the link partner and uses this link to negotiate the capabilities of the connection.

Autoneg Complete

When Auto-Negotiation is complete, the autoneg_complete output signal is asserted. In addition, the output signal an_fec_enable is asserted if the Forward Error Correction hardware is to be used; the output signal tx_pause_en is asserted if the transmitter hardware is allowed to generate PAUSE control packets, the output signal rx_pause_en is asserted if the receiver hardware is allowed to detect PAUSE control packets, and the output link control of the selected link is set to its mission mode value (bin)11.

IMPORTANT: *The autoneg_complete signal is not asserted until rx_status is received from the PCS. That means that, where link training is included, the autoneg_complete output signal is not asserted until after link training has completed and rx_status is High.*

Link Training

Link Training is performed after auto-negotiation converges to a backplane or copper technology. Technology selection can also be the result of a manual entry or parallel detection. Link training might be required due to frequency-dependent losses that can occur as digital signals traverse the backplane or a copper cable. The primary function of the Link Training core is to provide register information and a training sequence over the backplane link which is then analyzed by a receiving circuit which is not part of the core.

The other function of the core is to communicate training feedback from the receiver to the corresponding transmitter so that its equalizer circuit (not part of the core) can be adjusted as required. The two circuits comprising the core are the receive Link Training block and the transmit Link Training block.

IMPORTANT: *The logic responsible for adjusting the transmitter pre-emphasis taps must be supplied external to this IP core.*

Transmit

The Link Training transmit block constructs a 4,384-bit frame which contains a frame delimiter, control channel, and link training sequence. It is formatted as shown in Figure 3-20.

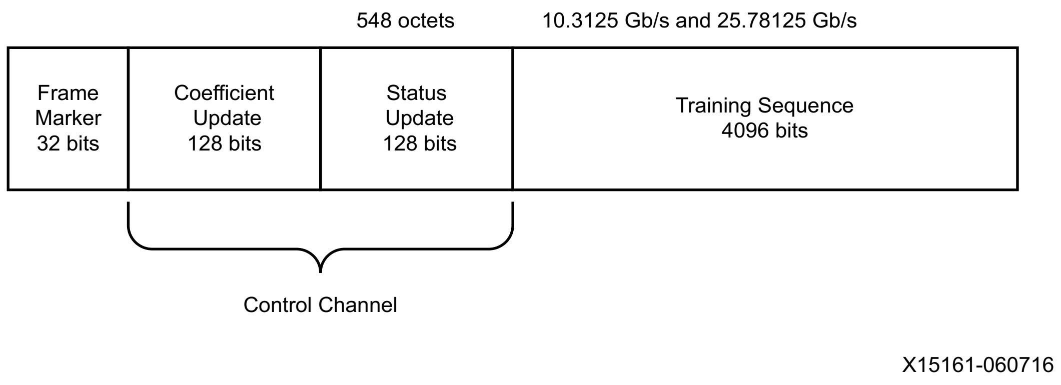

548 octets
10.3125 Gb/s and 25.78125 Gb/s
Frame Marker 32 bits
Coefficient Update 128 bits
Status Update 128 bits
Training Sequence 4096 bits
Control Channel
X15161-060716

Figure 3-20: **Link Training Frame Structure**

Xilinx recommends that the control channel bits not be changed by the Link Training algorithm while the transmit state machine is in the process of transmitting them, or they can be received incorrectly, possibly resulting in a DME error. This time begins when `tx_SOF` is asserted and ends at least 288 bit times later, or approximately 30 ns.

Although the coefficient and status contain 128 bit times at the line rate, the actual signaling rate for these two fields is reduced by a factor of eight. Therefore the DME clock rate is one quarter of the line rate.

Frame Marker

The frame marker consists of 16 consecutive 1s followed by 16 consecutive 0s. This pattern is not repeated in the remainder of the frame.

Coefficient and Status

Because the DME signaling rate for these two fields is reduced by a factor of eight, each coefficient and status transmission contains 128/8=16 bits, each numbered from 15:0. Table 3-5 and Table 3-6 define these bits in the order in which they are transmitted starting with bit 15 and ending with bit 0.

Table 3-5: **Coefficient and Update Field Bit Definitions**

Bits	Name	Description
15:14	Reserved	Transmitted as 0, ignored on reception.
13	Preset	1 = Preset coefficients 0 = Normal operation
12	Initialize	1 = Initialize coefficients 0 = Normal operation
11:6	Reserved	Transmitted as 0, ignored on reception.
5:4	Coefficient (+1) update	1 1 = reserved 0 1 = increment 1 0 = decrease 0 0 = hold
3:2	Coefficient (0) update	1 1 = reserved 0 1 = increment 1 0 = decrease 0 0 = hold
1:0	Coefficient (-1) update	1 1 = reserved 0 1 = increment 1 0 = decrease 0 0 = hold

Table 3-6: **Status Report Field Bit Definitions**

Bits	Name	Description
15	Receiver ready	1 = The local receiver has determined that training is complete and is prepared to receive data. 0 = The local receiver is requesting that training continue.
14:6	Reserved	Transmitted as 0, ignored on reception.
5:4	Coefficient (+1) update	0 1 = minimum 1 1 = maximum 1 0 = updated 0 0 = not_updated
3:2	Coefficient (0) update	1 1 = maximum 0 1 = minimum 1 0 = updated 0 0 = not_updated
1:0	Coefficient (-1) update	1 1 = maximum 0 1 = minimum 1 0 = updated 0 0 = not_updated

The functions of each bit are defined in IEEE Std. 802.3, Clause 72. Their purpose is to communicate the adjustments of the transmit equalizer during the process of link training. The corresponding signal names are defined in Table 2-18.

Training Sequence

The training sequence consists of a pseudo-random bit sequence (PRBS) of 4,094 bits followed by two zeros, for a total of 4,096 bits. The PRBS is transmitted at the line rate of 10.3125 or 25.78125 Gb/s. The PRBS generator receives an 11-bit seed from an external source. Subsequent to the initial seed being loaded, the PRBS generator continues to run with no further intervention required. The PRBS generator is implemented with a circuit which corresponds to the following polynomial:

G(x) = 1 + x9 + x11

Receive

The receive block implements the frame alignment state diagram shown in IEEE Std. 802.3, Clause 72, Figure 72-4.

Frame Lock State Machine

The frame lock state machine searches for the frame marker, consisting of 16 consecutive 1s followed by 16 consecutive 0s. This functionality is fully specified in IEEE Std. 802.3, Clause 72, Fig. 72-4. When frame lock has been achieved, `frame_lock` is set to a value of TRUE.

Received Data

The receiver outputs the control channel with the bit definitions defined in Table 3-5 and Table 3-6 and signal names defined in Port Descriptions – MAC+PCS Variant.

If a DME error has occurred during the reception of a particular DME frame, the control channel outputs are not updated but retain the value of the last received good DME frame and are updated when the next good DME frame is received.

Chapter 4

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 core. More detailed information about the standard Vivado® design flows and the Vivado 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 4]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 5]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 6]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 7]

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 4] 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 core using the following steps:

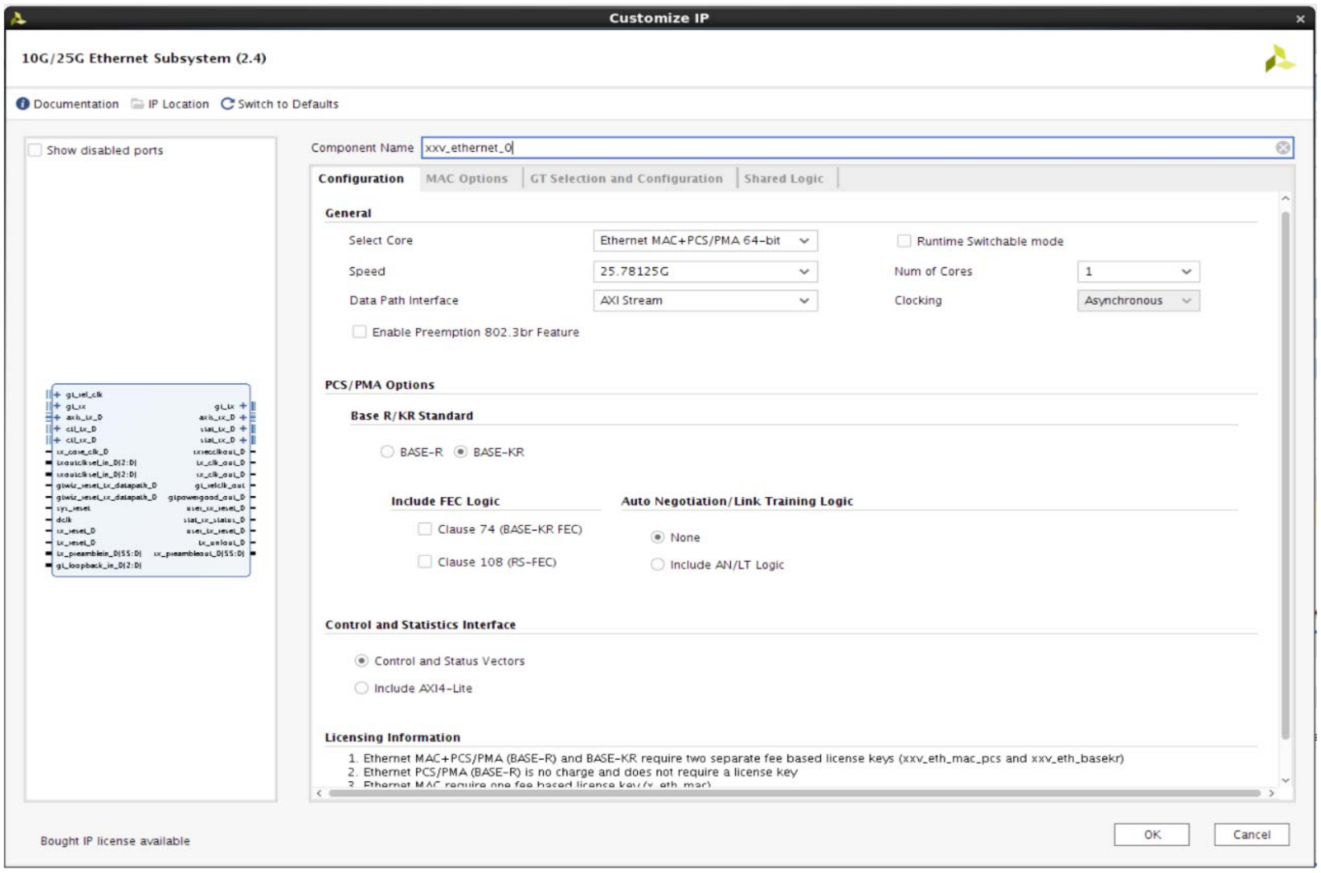
1. Select the IP from the IP catalog.
2. Double-click the selected IP or select the Customize IP command from the toolbar or right-click menu.

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 5] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 6].

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

Configuration Tab

The Configuration tab (Figure 4-1) provides the basic core configuration options. Default values are pre-populated in all fields.



10G/25G Ethernet Subsystem (2.4)

Component Name: xxv_ethernet_0

Configuration | MAC Options | GT Selection and Configuration | Shared Logic

General

Select Core: Ethernet MAC+PCS/PMA 64-bit

Runtime Switchable mode: ☐

Speed: 25.78125G

Num of Cores: 1

Data Path Interface: AXI Stream

Clocking: Asynchronous

☐ Enable Preemption 802.3br Feature

PCS/PMA Options

Base R/KR Standard

BASE-R [✓ BASE-KR]

Include FEC Logic

☐ Clause 74 (BASE-KR FEC)

☐ Clause 108 (RS-FEC)

Auto Negotiation/Link Training Logic

[✓ None] Include AN/LT Logic

Control and Statistics Interface

[✓ Control and Status Vectors] Include AXI4-Lite

Licensing Information

1. Ethernet MAC+PCS/PMA (BASE-R) and BASE-KR require two separate fee based license keys (xxv_eth_mac_pcs and xxv_eth_basekr)
2. Ethernet PCS/PMA (BASE-R) is no charge and does not require a license key
3. Ethernet MAC require one fee based license key (x_eth_mac)

Bought IP license available

Figure 4-1: **Configuration Tab**

Table 4-1: **Configuration Options**

Option	Values	Default
General		
Select Core	Ethernet MAC 64-bit Ethernet PCS/PMA 32-bit Ethernet PCS/PMA 64-bit Ethernet MAC+PCS/PMA 32-bit Ethernet MAC+PCS/PMA 64-bit	Ethernet MAC+PCS/PMA 64-bit
Speed	25.7812G 10.3125G	25.7812G
Runtime Switchable Mode	0, 1	0
Num of Cores	1 2 3 4	1
Clocking	Asynchronous	Asynchronous
Data Path Interface	AXI4-Stream(1) Media Independent Interface (MII)(2)	AXI4-Stream
PCS/PMA 32-bit		
Enable Preemption 802.3br Feature	0, 1	0
PCS/PMA Options		
Base-R Base-KR	Base-R Base-KR	Base-KR
Include FEC Logic		
Clause 74 (BASE-KR FEC)(3)(5)	0,1	0
Clause 108 (RS-FEC)(4)(5)	0,1	0
Auto-Negotiation/Link Training Logic		
Auto-Negotiation/Link Training Logic	None Include AN/LT Logic	None
Control and Statistics Interface		

Table 4-1: **Configuration Options**

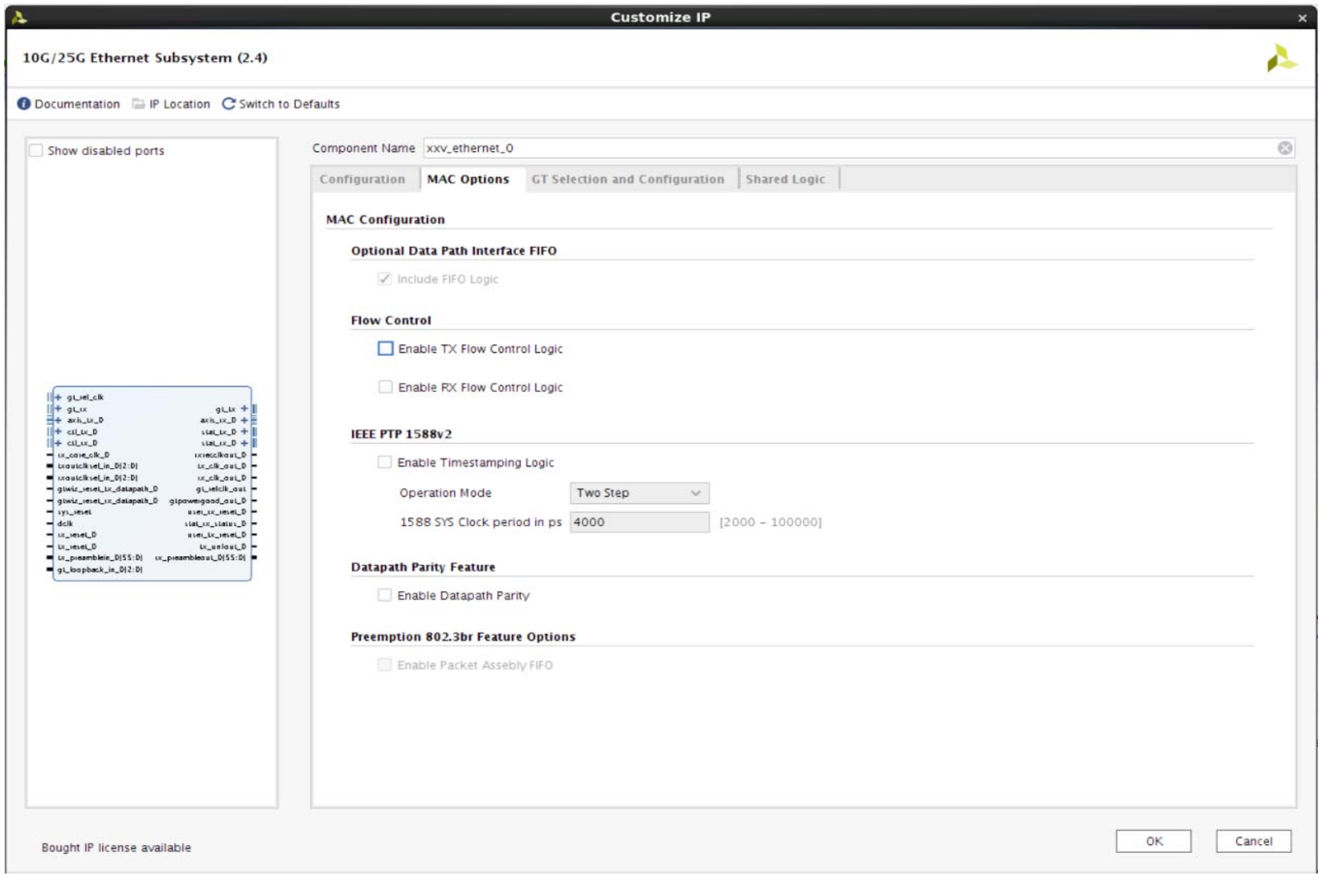
Option	Values	Default
Control and Statistics interface	Control and Status Vectors Include AXI4-Lite	Control and Status Vectors

Notes:

1. The AXI4-Stream interface is visible and is the only option for the Ethernet MAC+PCS/PMA and standalone Ethernet MAC core.
2. The MII interface is visible and is the only option for the Ethernet PCS/PMA core.
3. Clause 74 (BASE-KR FEC) logic is not supported for Base-R.
4. Clause 108 (RS-FEC) is not supported for Base-R,10G speed and also for runtime switchable mode.
5. Clause 74 (BASE-KR FEC) and Clause 108 (RS-FEC) both can be selected in Vivado IDE but during functional operation only one can be enabled at a time using the respective control signals.

MAC Options Tab

The MAC Options tab (Figure 4-2) provides additional core configuration options.



10G/25G Ethernet Subsystem (2.4)

Show disabled ports

Component Name: xxv_ethernet_0

Configuration | MAC Options | GT Selection and Configuration | Shared Logic

MAC Configuration

Optional Data Path Interface FIFO

[✓] Include FIFO Logic

Flow Control

[ ] Enable TX Flow Control Logic

[ ] Enable RX Flow Control Logic

IEEE PTP 1588v2

[ ] Enable Timestamping Logic

Operation Mode: Two Step

1588 SYS Clock period in ps: 4000 [2000 - 100000]

Datapath Parity Feature

[ ] Enable Datapath Parity

Preemption 802.3br Feature Options

[ ] Enable Packet Assebly FIFO

Bought IP license available

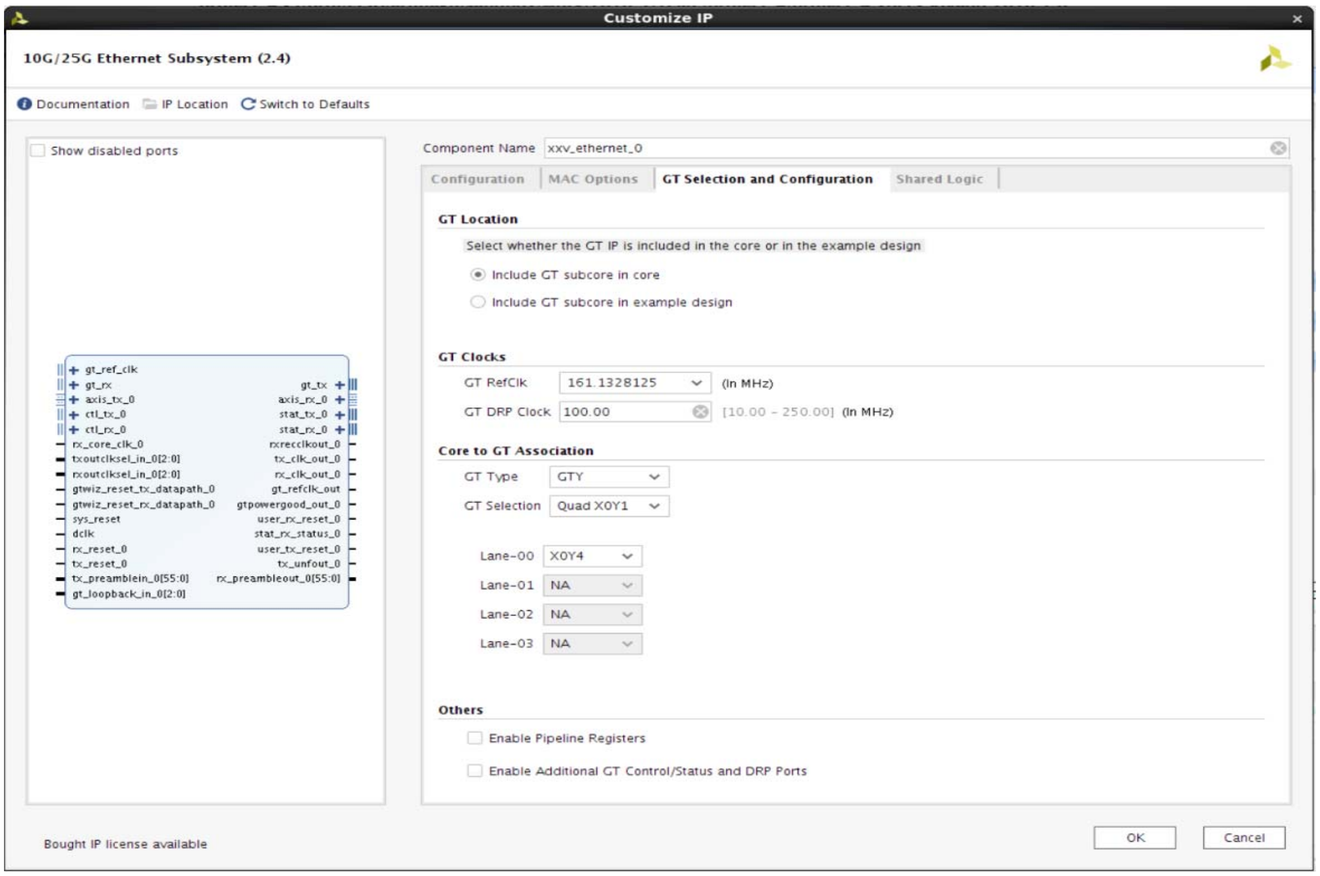
Figure 4-2: **MAC Options Tab**

Table 4-2: **MAC Options**

Option	Values	Default
Optional Data Path Interface FIFO		
Include FIFO Logic	Checked, Unchecked	Checked
Flow Control		
Enable TX Flow Control Logic	Checked, Unchecked	Unchecked
Enable RX Flow Control Logic	Checked, Unchecked	Unchecked
IEEE PTP 1588v2		
Enable Timestamping Logic	Checked, Unchecked	Unchecked
Operation Mode	One Step Two Step	Two Step
1588 SYS Clock period in ps	3333.33 to 20000	10000
Enable Datapath Parity	0, 1	0
Enable Packet Assembly FIFO	0, 1	0

GT Selection and Configuration Tab

The GT Selection and Configuration tab (Figure 4-3) enables you to configure the serial transceiver features of the core.



10G/25G Ethernet Subsystem (2.4)

Show disabled ports

Component Name: xxv_ethernet_0

GT Location

Select whether the GT IP is included in the core or in the example design

(●) Include GT subcore in core

( ) Include GT subcore in example design

GT Clocks

GT RefClk: 161.1328125 (In MHz)

GT DRP Clock: 100.00 [10.00 - 250.00] (In MHz)

Core to GT Association

GT Type: GTY

GT Selection: Quad X0Y1

Lane-00: X0Y4

Lane-01: NA

Lane-02: NA

Lane-03: NA

Others

☐ Enable Pipeline Registers

☐ Enable Additional GT Control/Status and DRP Ports

Bought IP license available

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

Table 4-3: **GT Clocks Options**

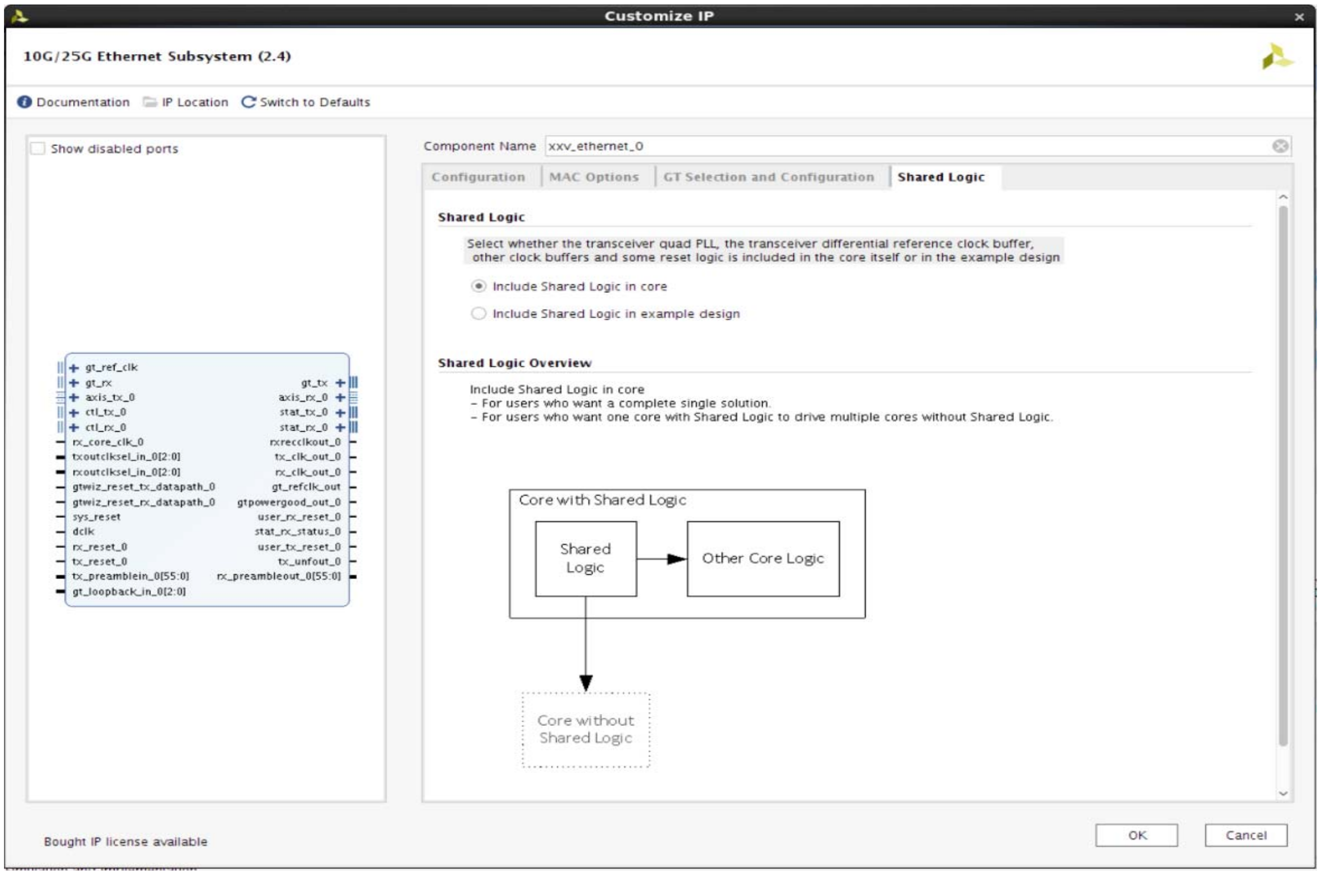
Option	Values	Default
GT Location		
Select whether the GT IP is included in the core or in the example design	Include GT subcore in core Include GT subcore in example design	Include GT subcore in core
GT Clocks		
GT RefClk (In MHz)(1)	161.1328125 195.3125 201.4160156 257.8125 322.265625	161.1328125
GT DRP Clock (In MHz)	10 – 250 MHz	100.00
Core to GT Association		
GT Type	GTY GTH	GTY
GT Selection	Options based on device/package Quad groups. For example: Quad X0Y1 Quad X0Y2 Quad X0Y3 ...	Quad X0Y1
Lane-00 to Lane-03	Auto filled based on device/package. For example, if Num of Core = 4, and GT Selection = Quad X0Y1, four lanes are: X0Y4 X0Y5 X0Y6 X0Y7	
Others		
Enable Pipeline Registers	Checked, Unchecked	Unchecked
Enable Additional GT Control/Status and DRP Ports	Checked, Unchecked	Unchecked

Notes:

1. This list provides frequencies used for the default configurations. See Vivado IDE in the latest version of the tools for a complete list of supported clock frequencies for different speeds.

Shared Logic Tab

The Shared Logic tab (Figure 4-4) enables you to use shared logic in either the core or the example design.



Component Name: xxv_ethernet_0

Shared Logic

Select whether the transceiver quad PLL, the transceiver differential reference clock buffer, other clock buffers and some reset logic is included in the core itself or in the example design

[✓ Include Shared Logic in core]

Include Shared Logic in example design

Shared Logic Overview

Include Shared Logic in core
- For users who want a complete single solution.
- For users who want one core with Shared Logic to drive multiple cores without Shared Logic.

Figure 4-4: **Shared Logic Tab**

Table 4-4: **Shared Logic Options**

Options	Default
Include Shared Logic in core Include Shared Logic in example design	Include Shared Logic in core

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 5].

Constraining the Core

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

Required Constraints

The 10G/25G High Speed Ethernet Subsystem 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 10G/25G High Speed Ethernet Subsystem.

To achieve consistent implementation results, an XDC containing these original, unmodified constraints must be used when a design is run through the Xilinx design tools. For additional details on the definition and use of an XDC specific constraints, see the *Vivado Design Suite User Guide: Using Constraints* (UG903) [Ref 15].

Constraints provided in the 10G/25G High Speed Ethernet Subsystem 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 core.

Clock Frequencies

This section is not applicable for this core.

Clock Management

This section is not applicable for this core.

Clock Placement

This section is not applicable for this core.

Banking

This section is not applicable for this core.

Transceiver Placement

This section is not applicable for this core.

I/O Standard and Placement

This section is not applicable for this 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 7].

Simulation Speed Up

The example design contains wait timers. A `define SIM_SPEED_UP is available to improve simulation time by speeding up these wait times. SIM_SPEED_UP is only available when running RTL simulation. It is not available when running simulation with post synthesis or post implementation netlist.

VCS

Use the vlogan option: +define+SIM_SPEED_UP.

ModelSim

Use the vlog option: +define+SIM_SPEED_UP.

IES

Use the ncvlog option: +define+SIM_SPEED_UP.

Vivado Simulator

Use the xvlog option: -d SIM_SPEED_UP.

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 5].

Example Design

This chapter contains information about the example design provided in the Vivado® Design Suite when using the Vivado Integrated Design Environment (IDE).

Overview

Figure 5-1 shows the instantiation of various modules and their hierarchy for a single core configuration of xxv_ethernet_0 example design when the GT (serial transceiver) is inside the IP core.

Sync registers and pipeline registers are used for to synchronize the data between the core and the GT. Clocking helper blocks are used to generate the required clock frequency for the core.

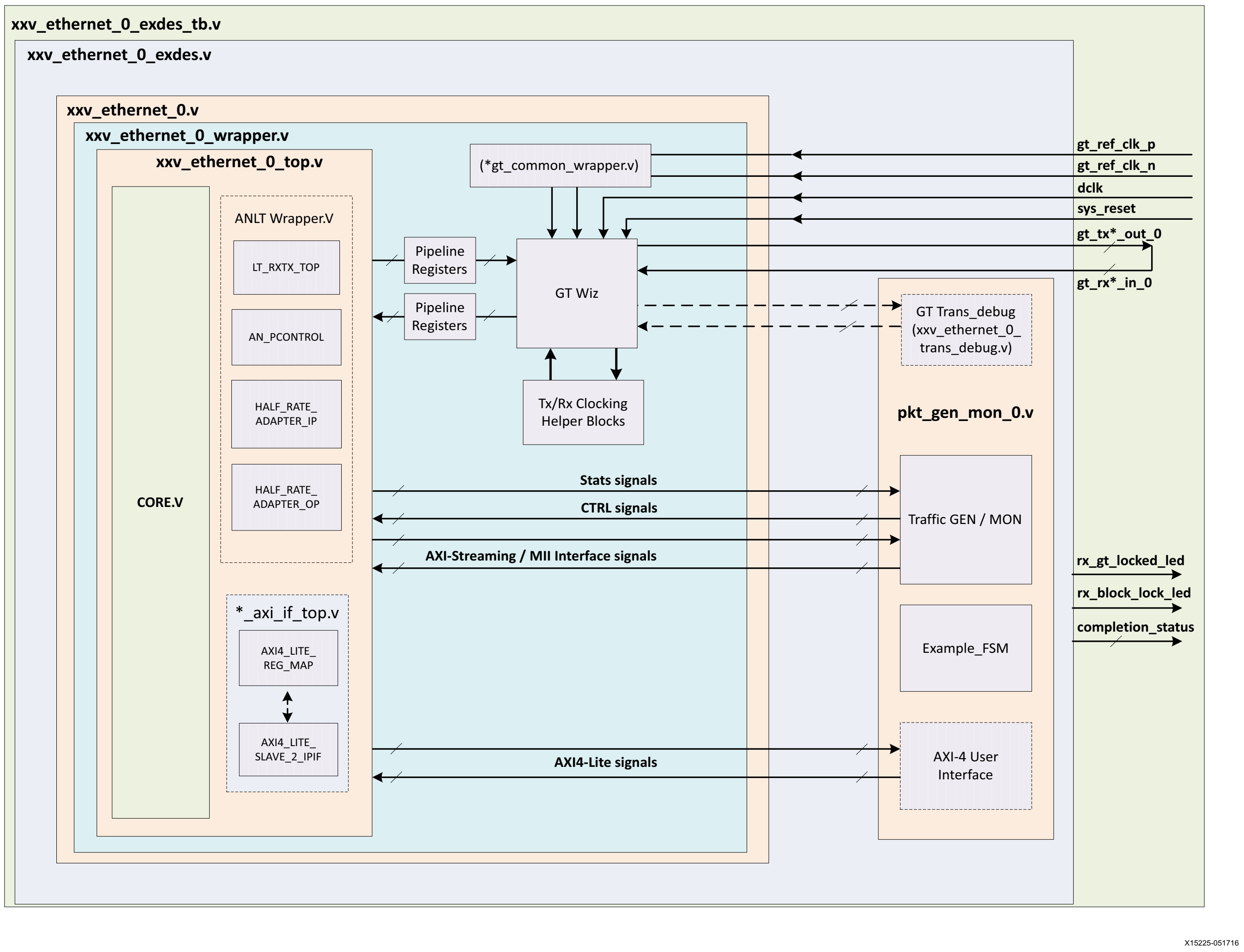


Figure 5-1: **Single Core Example Design Hierarchy**

Following are the user interfaces available for different configurations.

- MAC/PCS configuration:
  - AXI4-Stream for datapath interface
  - AXI4-Lite for control and statistics interface
- PCS configuration:
  - MII for datapath interface
  - AXI4-Lite for control and statistics interface

The `xxv_ethernet_0_pkt_gen_mon` module is used to generate the data packets for sanity testing. The packet generation and checking is controlled by a FSM module.

The optional modules are described as follows:

- **xxv_ethernet _0_trans_debug**: This module is present in the example design when you enable the **Additional GT Control and Status Ports** check box from the GT Selection and Configuration Tab in the Vivado IDE or **Include GT subcore in example design** option in the GT Selection and Configuration tab or the **Runtime Switchable** mode option in the in the Configuration tab. This module brings out all the GT channel DRP ports, and some control and status ports of the transceiver module out of the xxv_ethernet core.
- **Retiming registers**: When you select the **Enable Retiming Register** option from the GT Selection and Configuration Tab, it includes a single stage pipeline register between core and the GT to ease timing, using the `gt_txusrclk2` and `gt_rxusrclk2` for TX and RX paths respectively. However, by default two-stage registering is done for the signals between GT and the core.
- **TX / RX Sync register**: The TX Sync register double synchronizes the data from the core to the GT with respect to the `tx_clk`. The RX Sync register double synchronizes the data from the GT to the core with respect to the `rx_serdes_clk`.

Note: For Runtime Switchable, if Auto-Negotiation/Link training is selected in Vivado IDE, then AN operation will be performed only with the 10G data rate during switchings and LT will be performed in the mission mode.

Note: If Auto-Negotiation/Link training is selected in Vivado IDE and the number of cores >= 3, then a PBLOCK constraint must be applied for the anlt_wrappers. The PBLOCK should be placed near to the selected transceivers (GT) and the size should be sufficient to fit the anlt_wrapper utilization. Refer to example_top.xdc for more information. Following is an example for a xcvu095-ffva2104-2-e device when four cores are selected and the transceivers are x0y4 to x0y7.

Example:

create_pblock pblock_ANLT
add_cells_to_pblock [get_pblocks pblock_ANLT] [get_cells -quiet [list DUT/inst/i_*_top_0/
i_*_*_ANLT_WRAPPER DUT/inst/i_*_top_1/i_*_ANLT_WRAPPER DUT/inst/i_*_*_top_2/
i_*_ANLT_WRAPPER DUT/inst/i_*_top_3/i_*_ANLT_WRAPPER]]resize_pblock [get_pblocks
pblock_ANLT] -add {SLICE_X0Y5:SLICE_X40Y180}

Figure 5-2 shows the instantiation of various modules and their hierarchy for the multiple core configuration of xxv_ethernet_0 example design.

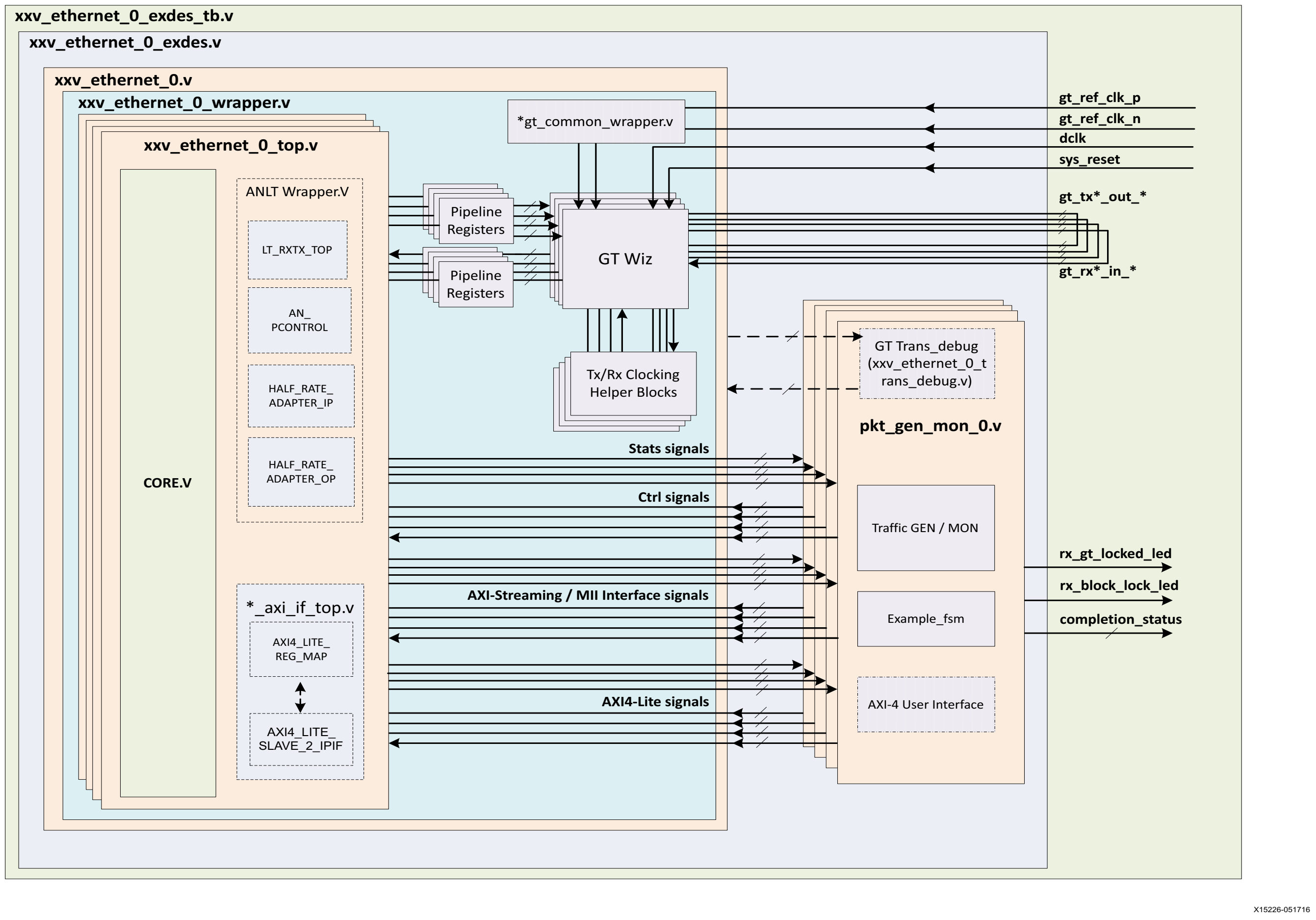


Figure 5-2: **Multiple Core Example Design Hierarchy**

Example Design Hierarchy (GT in Example Design)

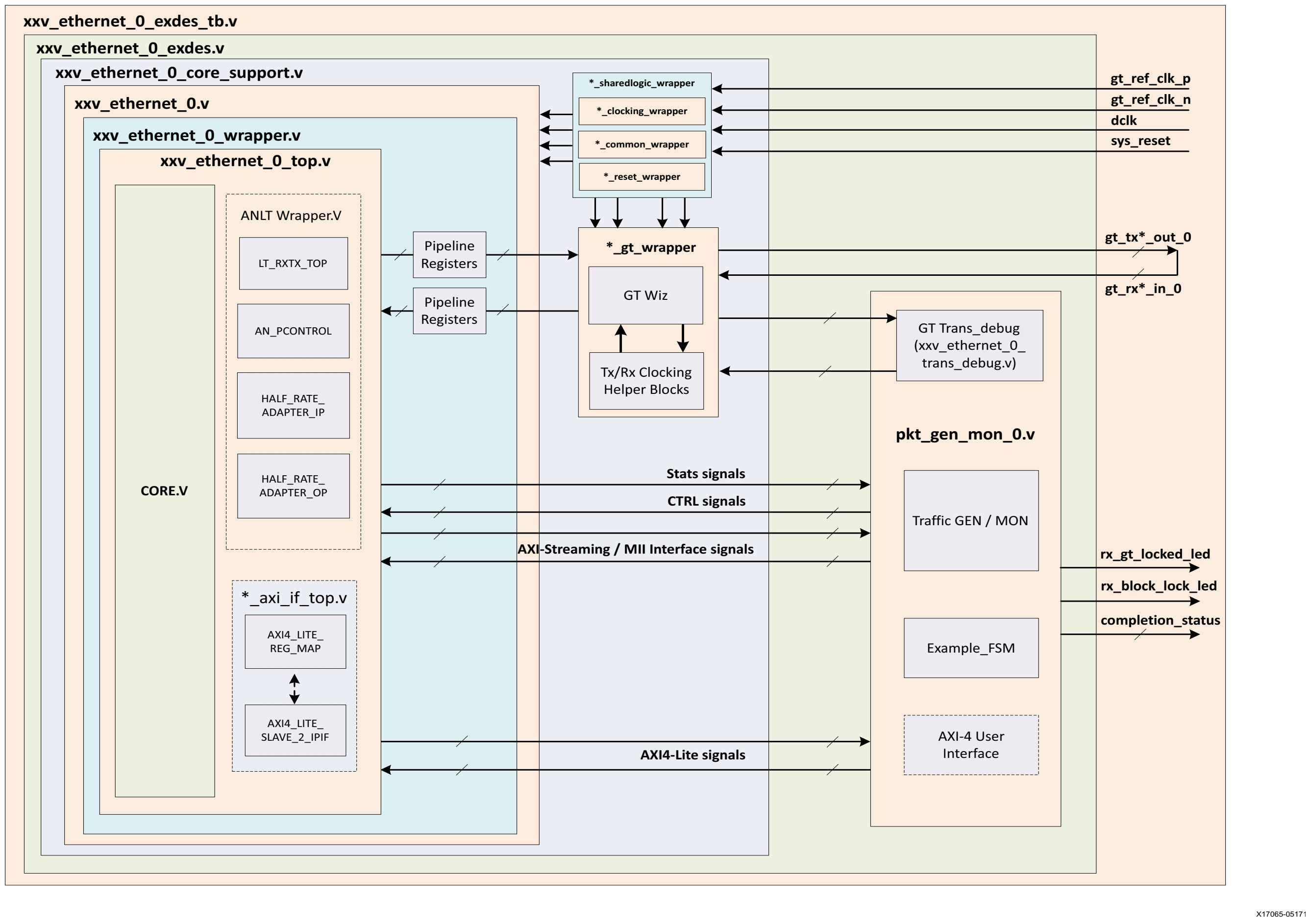


Figure 5-3: **Single Core with GT in Example Design Hierarchy**

Figure 5-3 shows the instantiation of various modules and their hierarchy for a single core configuration of the xxv_ethernet_0 example design when the GT (serial transceiver) is outside the IP Core, that is, in the example design. This hierarchical example design is delivered when you select the **Include GT subcore in example design** option from the GT Selection and Configuration tab.

The `xxv_ethernet_0_core_support.v` is present in the hierarchy when you select the **Include GT subcore in example design** option from the GT Selection and Configuration tab or the **Include Shared Logic in example design** option from the Shared Logic tab. This instantiates the `xxv_ethernet_0_sharedlogic_wrapper.v` module and the `xxv_ethernet_0.v` module for the **Include Shared Logic in example design** option. The `xxv_ethernet_0_gt_wrapper.v` module will be present when you select the **GT subcore in example design** option.

The user interface available for MAC/PCS configuration and PCS configuration configurations is the same as mentioned in the Overview.

The `xxv_ethernet_0.v` module instantiates the necessary the sync registers/retiming pipeline registers for the synchronization of data between the core and the GT.

The `xxv_ethernet_0_pkt_gen_mon` module is used to generate the data packets for sanity testing. The packet generation and checking is controlled by a Finite State Machine (FSM) module.

Description of optional modules are as follows:

- xxv_ethernet _0_sharedlogic_wrapper

  This module is present in the example design when you select the **Include GT subcore in example design** option from the GT Selection and Configuration tab or **Include Shared Logic** in the Example Design from the Shared Logic tab. This module brings all modules that can be shared between multiple IP cores and designs outside the IP core.
- xxv_ethernet _0_gt_wrapper

  This module is present in the example design when you select the **Include GT subcore in example** design option from the GT Selection and Configuration tab. This module is having instantiations of the GT along with various helper blocks. The clocking helper blocks are used to generate the required clock frequency for the core.

Figure 5-4 shows the instantiation of various modules and their hierarchy for the multiple core configuration of the xxv_ethernet_0 example design when the GT is in the example design.

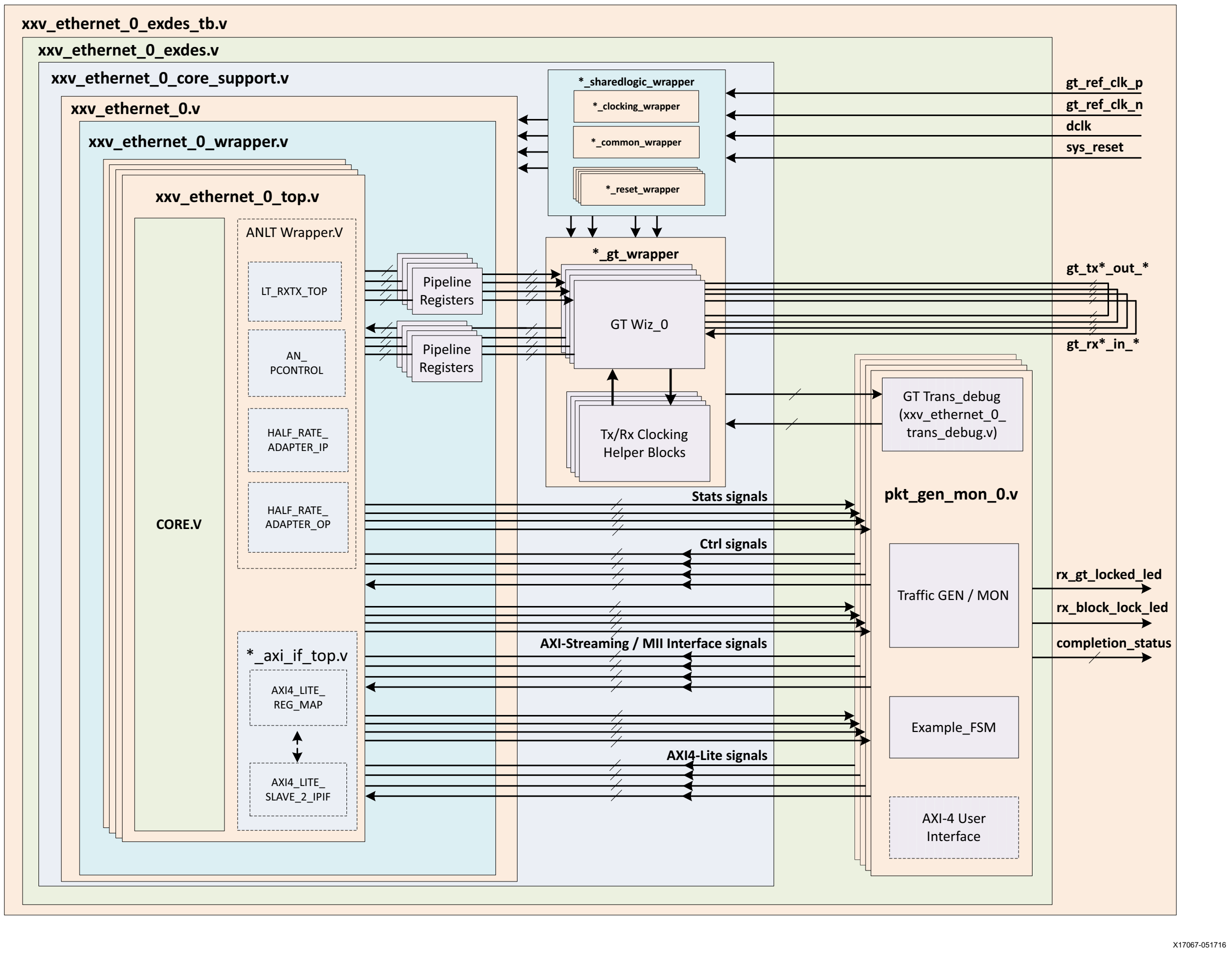


Figure 5-4: **Multiple Core with GT in Example Design Hierarchy**

User Interface

General purpose I/Os (GPIOs) are provided to control the example design. The user input and user output ports are described in Table 5-1.

Table 5-1: **User Input and User Output Ports**

Name	Size	Direction	Description
sys_reset	1	Input	Reset for xxv_ethernet core.
gt_ref_clk_p	1	Input	Differential input clk to GT. This clock frequency should be equal to the GT RefClk frequency mentioned in the Vivado IDE GT Selection and Configuration tab.
gt_ref_clk_n	1	Input	Differential input clk to GT. This clock frequency should be equal to the GT RefClk frequency mentioned in the Vivado IDE GT Selection and Configuration tab.
dclk	1	Input	Stable/free running input clk to GT. This clock frequency should be equal to the GT DRP clock frequency mentioned in the Vivado IDE GT Selection and Configuration tab.
rx_gt_locked_led_0	1	Output	Indicates that GT has been locked.
rx_block_lock_led_0	1	Output	Indicates RX block lock has been achieved.
restart_tx_rx_0	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.

Table 5-1: **User Input and User Output Ports *(Cont'd)***

Name	Size	Direction	Description
completion_status	5	Output	This signal represents the test status/result. • 5'd0 Test did not run. • 5'd1 PASSED 25GE/10GE CORE TEST SUCCESSFULLY COMPLETED. • 5'd2 No block lock on any lanes. • 5'd3 Not all lanes achieved block lock. • 5'd4 Some lanes lost block lock after achieving block lock. • 5'd5 No lane sync on any lanes. • 5'd6 Not all lanes achieved sync. • 5'd7 Some lanes lost sync after achieving sync. • 5'd8 No alignment status or rx_status was achieved. • 5'd9 Loss of alignment status or rx_status after both were achieved. • 5'd10 TX timed out. • 5'd11 No TX data was sent. • 5'd12 Number of packets received did not equal the number of packets sent. • 5'd13 Total number of bytes received did not equal the total number of bytes sent. • 5'd14 A protocol error was detected. • 5'd15 Bit errors were detected in the received packets. • 5'd31 Test is stuck in reset.
mode_change_*	1	Input	This port is available only when **Runtime Switchable** is selected in Vivado IDE and this is used to switch the core speed.
core_speed_*	1	Output	This signal indicates the speed with which the core is working: 1'b1 = 10G and 1'b0 = 25G
send_continuous_pkts_*	1	Input	This port can be used to send continuous packets for board validation. • 1'b0 - Sends fixed 20 packets for simulation. • 1'b1 - Sends continuous packets for board.
stat_reg_compare	1	Output	Indicates TX and RX statistics registers comparison status. • 1'b1 - Indicates both the TX and RX statistics matched. • 1'b0 - Indicates if there is any mismatch between TX and RX statistics. ***Note:*** This output is available when you select **Include AXI4-Lite** option in the General Tab.

Core xci Top Level Port List

The top level port list for the core xci with all features enabled is listed below:

In the following table an asterisk (*) represents CORE number, having value 0 to 3.

Example: port_name_*

- port_name_0: for first CORE
- port_name_1: for second CORE (will be present when user selects number of cores >=2)
- port_name_2: for third CORE (will be present when user selects number of cores >=3)
- port_name_3: for fourth CORE (will be present when user selects number of cores =4)

Table 5-2: **Core xci Top Level Port List**

Name	Size	Direction	Description
Common Clock/Reset Signals			
sys_reset	1	Input	Reset for core. ***Note:*** This port is available when the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab.
dclk	1	Input	Stable input clk to GT. ***Note:*** This port is available when the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab.
gt_refclk_p	1	Input	Differential input clk to GT. ***Note:*** This port is available when the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab and the **Include Shared Logic in core** option is selected in the Shared Logic tab.
gt_refclk_n	1	Input	Differential input clk to GT. ***Note:*** This port is available when the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab and the **Include Shared Logic in core** option is selected in the Shared Logic tab.
qpll0clk_in	2/4	Input	QPLL0 clock input. ***Note:*** This port is available when the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab and the **Include Shared Logic in example design** option is selected in the Shared Logic tab. Port width: 2-bit for 50G single core and 4bit for 40G one core / 50G two core.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
qpll0refclk_in	2/4	Input	QPLL0 ref clock input. ***Note:*** This port is available when the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab and the **Include Shared Logic in example design** option is selected in the Shared Logic tab. Port width: 2-bit for 50G single core and 4bit for 40G one core / 50G two core.
qpll1clk_in	2/4	Input	QPLL1 clock input. ***Note:*** This port is available when the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab and the **Include Shared Logic in example design** option is selected in the Shared Logic tab. Port width: 2-bit for 50G single core and 4bit for 40G one core / 50G two core.
qpll1refclk_in	2/4	Input	QPLL1 ref clock input. ***Note:*** This port is available when the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab and the **Include Shared Logic in example design** option is selected in the Shared Logic tab. Port width: 2-bit for 50G single core and 4bit for 40G one core / 50G two core.
gtwiz_reset_qpll0lock_in	1	Input	QPLL0 lock reset input to the GT. ***Note:*** This port is available when the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab and the **Include Shared Logic in example design** option is selected in the Shared Logic tab.
gtwiz_reset_qpll0reset_out	1	Output	QPLL0 lock reset output from the GT. This port is available when the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab and the **Include Shared Logic in example design** option is selected in the Shared Logic tab.
gtwiz_reset_qpll1lock_in	1	Input	QPLL1 lock reset input to the GT. ***Note:*** This port is available when the **Include GT subcore in core option** is selected in the GT Selection and Configuration tab and the **Include Shared Logic in example design option** is selected in the Shared Logic tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
gtwiz_reset_qpll1reset_out	1	Output	QPLL1 lock reset output from the GT. ***Note:*** This port is available when the **Include GT subcore in core option** is selected in the GT Selection and Configuration tab and the **Include Shared Logic in example design option** is selected in the Shared Logic tab.
tx_clk_out_*	1	Output	TX user clock output from GT. ***Note:*** This port is available when the **Select Core** is Ethernet MAC+PCS/PMA 32/64-bit and the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab.
tx_mii_clk_*	1	Output	TX user clock output from GT. ***Note:*** This port is available when the **Select Core** is Ethernet PCS/PMA 32/64-bit and the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab.
rx_clk_out_*	1	Output	RX user clock output from GT. ***Note:*** This port is available when the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab.
rx_serdes_clk_*	1	Input	RX serdes clock input to core ***Note:*** This port is available when the **Include GT subcore in example design** option is selected in the GT Selection and Configuration tab.
rx_serdes_reset_*	1	Input	RX serdes reset input to core ***Note:*** This port is available when the **Include GT subcore in example design** option is selected in the GT Selection and Configuration tab.
rxrecclkout_*	1	Output	RX recovered clock output from GT. ***Note:*** This port is available when the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab.
tx_core_clk_*	1	Input	TX Core clock input from GT wrapper. ***Note:*** This port is available when the **Include GT subcore in example design** option is selected in the GT Selection and Configuration tab.
rx_core_clk _*	1	Input	RX Core clock input to the core.
tx_reset_*	1	Input	TX reset input to the core.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
user_tx_reset_*	1	Output	TX reset output for the user logic. ***Note:*** This port is available when the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab and the **Include Shared Logic in core** option is selected in the Shared Logic tab.
gt_reset_tx_done_out_*	1	Output	TX reset done signal from the GT. ***Note:*** This port is available when the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab and the **Include Shared Logic in example design** option is selected in the Shared Logic tab.
rx_reset_*	1	Input	RX reset input to the core.
user_rx_reset_*	1	Output	RX reset output for the user logic. ***Note:*** This port is available when the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab and the **Include Shared Logic in core** option is selected in the Shared Logic tab.
gt_reset_rx_done_out_*	1	Output	RX reset done signal from the GT. ***Note:*** This port is available when the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab and the **Include Shared Logic in example design** option is selected in the Shared Logic tab.
ctl_gt_reset_all_*	1	Output	gt_reset_all signal from the AXI4-Lite register map. ***Note:*** This port is available when the **Include AXI4-Lite** is selected from the Configuration tab and the **Include Shared Logic in example design** option is selected in the Shared Logic tab.
ctl_gt_tx_reset_*	1	Output	gt_tx_reset signal from the AXI4-Lite register map. ***Note:*** This port is available when the **Include AXI4-Lite** option is selected from the Configuration tab and the **Include Shared Logic in example design** option is selected in the Shared Logic tab.
ctl_gt_rx_reset_*	1	Output	gt_rx_reset signal from the AXI4-Lite register map. ***Note:*** This port is available when the **Include AXI4-Lite** is selected from the Configuration tab and the **Include Shared Logic in example design** option is selected in the Shared Logic tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
gt_reset_all_in_*	1	Input	gt_reset_all signal from the reset_wrapper of shared logic wrapper. ***Note:*** This port is available when the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab and the **Include Shared Logic in example design** option is selected in the Shared Logic tab.
gt_tx_reset_in_*	1	Input	gt_tx_reset_in signal from reset_wrapper of shared logic wrapper. ***Note:*** This port is available when the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab and the **Include Shared Logic in example design** option is selected in the Shared Logic tab.
gt_rx_reset_in_*	1	Input	gt_rx_reset_in signal from reset_wrapper of shared logic wrapper. ***Note:*** This port is available when the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab and the **Include Shared Logic in example design** option is selected in the Shared Logic tab.
gt_refclk_out	1	Output	Indicates the GT_refclk output This port is available when the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab and the **Include Shared Logic in example design** option is selected in the Shared Logic tab.
gtpowergood_out_*	1	Output	Refer to the *UltraScale Architecture GTH Transceivers User Guide* (UG576) [Ref 13] or the *UltraScale Architecture GTY Transceivers User Guide* (UG578) [Ref 14] for the port description.
TXOUTCLKSEL_IN_*	3	Input	This port is used to select the clock source for the gtwizard TX output clock ***Note:*** This port to be driven with 3'b101 as per preset.
RXOUTCLKSEL_IN_*	3	Input	This port is used to select the clock source for the gtwizard RX output clock ***Note:*** This port to be driven with 3'b101 as per preset.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
Common Transceiver Interface ports			
gt_loopback_in_*	3	Input	GT loopback input signal. Refer to the GT user guide. ***Note:*** This port is available when the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab.
gt_loopback_out_*	1	Output	GT loopback output signal from AXI4-Lite register map. Refer to the GT user guide. ***Note:*** This port is available when **Include AXI4-Lite** is selected from the Configuration tab and the **Include GT subcore in example design** option is selected in the GT Selection and Configuration tab.
gt_txp_out	1	Output	Differential serial GT TX output ***Note:*** This port is available when the **Include GT subcore in core** option and Board support is selected in the GT Selection and Configuration tab.
gt_txn_out		Output	Differential serial GT TX output. ***Note:*** This port is available when the **Include GT subcore in core** option and Board support is selected in the GT Selection and Configuration tab.
gt_rxn_in		Input	Differential serial GT RX input. ***Note:*** This port is available when the **Include GT subcore in core** option and Board support is selected in the GT Selection and Configuration tab.
gt_rxp_in		Input	Differential serial GT RX input. ***Note:*** This port is available when the **Include GT subcore in core** option and Board support is selected
gt_rxp_in_0	1	Input	Differential serial GT RX input for lane 0. ***Note:*** This port is available when the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab.
gt_rxn_in_0	1	Input	Differential serial GT RX input for lane 0. ***Note:*** This port is available when the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
gt_rxp_in_1	1	Input	Differential serial GT RX input for lane 1. ***Note:*** This port is available when **Num of Cores** is >1 and the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab.
gt_rxn_in_1	1	Input	Differential serial GT RX input for lane 1. ***Note:*** This port is available when **Num of Cores** is >1 and the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab.
gt_rxp_in_2	1	Input	Differential serial GT RX input for lane 2. ***Note:*** This port is available when **Num of Cores** is >2 and the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab.
gt_rxn_in_2	1	Input	Differential serial GT RX input for lane 2. ***Note:*** This port is available when **Num of Cores** is >2 and the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab.
gt_rxp_in_3	1	Input	Differential serial GT RX input for lane 3. ***Note:*** This port is available when **Num of Cores** is >3 and the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab.
gt_rxn_in_3	1	Input	Differential serial GT RX input for lane 3. ***Note:*** This port is available when **Num of Cores** is >3 and the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab.
gt_txp_out_0	1	Output	Differential serial GT TX output for lane 0. ***Note:*** This port is available when the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab.
gt_txn_out_0	1	Output	Differential serial GT TX output for lane 0. ***Note:*** This port is available when the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab.
gt_txp_out_1	1	Output	Differential serial GT TX output for lane 1. ***Note:*** This port is available when **Num of Cores** is >1 and the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
gt_txn_out_1	1	Output	Differential serial GT TX output for lane 1. ***Note:*** This port is available when **Num of Cores** is >1 and the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab.
gt_txp_out_2	1	Output	Differential serial GT TX output for lane 2. ***Note:*** This port is available when **Num of Cores** is >2 and the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab.
gt_txn_out_2	1	Output	Differential serial GT TX output for lane 2. ***Note:*** This port is available when **Num of Cores** is >2 and the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab.
gt_txp_out_3	1	Output	Differential serial GT TX output for lane 3. ***Note:*** This port is available when **Num of Cores** is >3 and the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab.
gt_txn_out_3	1	Output	Differential serial GT TX output for lane 3. ***Note:*** This port is available when **Num of Cores** is >3 and the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab.
rxgearboxslip_in_*	1	Output	Rxgearboxslip signal from core to GT. ***Note:*** This port is available when the **Include GT subcore in example design** option is selected in the GT Selection and Configuration tab.
rxdatavalid_out_*	2	Input	rx data valid signal from GT to core. ***Note:*** This port is available when the **Include GT subcore in example design** option is selected in the GT Selection and Configuration tab.
rx_serdes_data_out_*	32/64/128	Input	rx data signal from GT to core. ***Note:*** This port is available when the **Include GT subcore in example design** option is selected in the GT Selection and Configuration tab. The data width is 32/64 bits for 10G configuration and 128 bits for 25G configuration.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
rxheader_out_*	6	Input	rx header signal from GT to core. ***Note:*** This port is available when the **Include GT subcore in example design** option is selected in the GT Selection and Configuration tab.
rxheadervalid_out_*	2	Input	rx header valid signal from GT to core. ***Note:*** This port is available when the **Include GT subcore in example design** option is selected in the GT Selection and Configuration tab.
tx_serdes_data_in_*	32/64/ 128	Output	tx data signal from core to GT. ***Note:*** This port is available when the **Include GT subcore in example design** option is selected in the GT Selection and Configuration tab. The data width is 32/64 bits for 10G configuration and 128 bits for 25G configuration.
txheader_in_*	6	Output	tx header signal from core to GT. ***Note:*** This port is available when the **Include GT subcore in example design** option is selected in the GT Selection and Configuration tab.
Transceiver control and status debug ports Ports under this section will be available when the **Include GT subcore in core** option is selected in the GT Selection and Configuration tab or **Enable Additional GT Control/Status and DRP Ports** is selected from the GT Selection and Configuration tab.			
gt_dmonitorout_*	16	Output	Refer to the GT user guide for the port description.
gt_eyescandataerror_*	1	Output	Refer to the GT user guide for the port description.
gt_eyescanreset_*	1	Input	Refer to the GT user guide for the port description.
gt_eyescantrigger_*	1	Input	Refer to the GT user guide for the port description.
gt_pcsrsvdin_*	16	Input	Refer to the GT user guide for the port description.
gt_rxbufreset_*	1	Input	Refer to the GT user guide for the port description. ***Note:*** Port width: 2-bit for 50G single core and 4-bits for 40G.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
gt_rxbufstatus_*	3	Output	Refer to the GT user guide for the port description.
gt_rxcdrhold_*	1	Input	Refer to the GT user guide for the port description.
gt_rxcommadeten_*	1	Input	Refer to the GT user guide for the port description.
gt_rxdfeagchold_*	1	Input	Refer to the GT user guide for the port description.
gt_rxdfelpmreset_*	1	Input	Refer to the GT user guide for the port description.
gt_rxlatclk_*	1	Input	Refer to the GT user guide for the port description.
gt_rxlpmen_*	1	Input	Refer to the GT user guide for the port description.
gt_rxpcsreset_*	1	Input	Refer to the GT user guide for the port description.
gt_rxpmareset_*	1	Input	Refer to the GT user guide for the port description.
gt_rxpolarity_*	1	Input	Refer to the GT user guide for the port description.
gt_rxprbscntreset_*	1	Input	Refer to the GT user guide for the port description.
gt_rxprbserr_*	1	Input	Refer to the GT user guide for the port description.
gt_rxprbssel_*	4	Input	Refer to the GT user guide for the port description.
gt_rxrate_*	3	Input	Refer to the GT user guide for the port description.
gt_rxslide_in_*	1	Input	Refer to the GT user guide for the port description.
gt_rxstartofseq_*	2	Output	Refer to the GT user guide for the port description.
gt_txbufstatus_*	2	Output	Refer to the GT user guide for the port description.
gt_txdiffctrl_*	5	Input	Refer to the GT user guide for the port description.
gt_txinhibit_*	1	Input	Refer to the GT user guide for the port description.
gt_txlatclk_*	1	Input	Refer to the GT user guide for the port description.
gt_txmaincursor_*	7	Input	Refer to the GT user guide for the port description.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
gt_txpcsreset_*	1	Input	Refer to the GT user guide for the port description.
gt_txpmareset_*	1	Input	Refer to the GT user guide for the port description.
gt_txpolarity_*	1	Input	Refer to the GT user guide for the port description.
gt_txpostcursor_*	5	Input	Refer to the GT user guide for the port description.
gt_txprbsforceerr_*	1	Input	Refer to the GT user guide for the port description.
gt_txprbssel_*	4	Input	Refer to the GT user guide for the port description.
gt_txprecursor_*	5	Input	Refer to the GT user guide for the port description.
gtwiz_reset_tx_datapath_*	1	Input	Refer to the GT user guide for the port description.
gtwiz_reset_rx_datapath_*	1	Input	Refer to the GT user guide for the port description.
gt_ch_drpclk_*	1	Input	Refer to the GT user guide for the port description.
gt_ch_drpdo_*	16	Output	Refer to the GT user guide for the port description.
gt_ch_drprdy_*	1	Output	Refer to the GT user guide for the port description.
gt_ch_drpen_*	1	Input	Refer to the GT user guide for the port description.
gt_ch_drpwe_*	1	Input	Refer to the GT user guide for the port description.
gt_ch_drpaddr_*	10	Input	Refer to the GT user guide for the port description.
gt_ch_drpdi_*	16	Input	Refer to the GT user guide for the port description.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
AXI4-Lite Interface Ports Ports under this section will be available when the **Include AXI4-Lite** is selected from the Configuration tab.			
s_axi_aclk_*	1	Input	AXI clock signal
s_axi_aresetn_*	1	Input	AXI reset signal
pm_tick_*	1	Input	PM tick user input
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

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

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

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
AXI4-Stream User Interface Signals Ports under this section will be available when **Ethernet MAC+PCS/PMA-32/64-bit** is selected from the Configuration tab.			
tx_unfout_*	1	Output	Underflow signal for TX datapath from core. 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 TX datapath interface.
tx_axis_tready_*	1	Output	TX path ready signal from core.
tx_axis_tvalid_*	1	Input	Transmit AXI4-Stream Data valid.
tx_axis_tdata_*	64/32	Input	Transmit AXI4-Stream Data bus.
tx_axis_tlast_*	1	Input	Transmit AXI4-Stream tlast.
tx_axis_tkeep_*	8/4	Input	Transmit AXI4-Stream tkeep.
tx_axis_tuser_*	1	Input	Transmit AXI4-Stream tuser.
tx_preamblein_*	56	Input	Transmit AXI4-Stream preamble.
tx_parityin_*	8	Input	Transmit AXI4-Stream datapath parity.
rx_axis_tvalid_*	1	Output	Receive AXI4-Stream Data valid.
rx_axis_tdata_*	64/32	Output	Receive AXI4-Stream Data bus.
rx_axis_tlast_*	1	Input	Receive AXI4-Stream tlast.
rx_axis_tkeep_*	8/4	Input	Receive AXI4-Stream tkeep.
rx_axis_tuser_*	1	Input	Receive AXI4-Stream tuser.
rx_preamblein_*	56	Input	Receive AXI4-Stream preamble.
rx_parityout_*	8	Output	Receive AXI4-Stream datapath parity.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
MII User Interface Signals Ports under this section will be available when **Ethernet PCS/PMA** is selected from the Configuration tab.			
tx_mii_d_*	64	Input	Transmit XGMII/25GMII Data bus.
tx_mii_c_*	8	Input	XGMII/25GMII Control bus.
rx_mii_d_*	64	Output	Receive XGMII/25GMII Data bus.
rx_mii_c_*	8	Output	Receive XGMII/25GMII Control bus.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
TX Path Control / Status / Statistics Signals			
ctl_tx_enable_*	1	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 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 core stops transmitting any more packets. ***Note:*** This port is available when **Include AXI4-Lite** is not selected and **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
ctl_tx_send_rfi_*	1	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. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
ctl_tx_send_lfi_*	1	Input	Transmit Local Fault Indication (LFI) code word. Takes precedence over RFI. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
ctl_tx_send_idle_*	1	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. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
ctl_tx_fcs_ins_enable_*	1	Input	Enable FCS insertion by the TX core. If this bit is set to 0, the core does not add FCS to the packet. If this bit is set to 1, the HSEC core calculates and adds the FCS to the packet. This input cannot be changed dynamically between packets. ***Note:*** This port is available when **Include AXI4-Lite** is not selected and **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
ctl_tx_ignore_fcs_*	1	Input	Enable FCS error checking at the interface by the TX core. This input only has effect when ctl_tx_fcs_ins_enable is Low. If this input is Low and a packet with bad FCS is being transmitted, it is not binned as good. If this input is High, a packet with bad FCS is binned as good. The error is flagged on the signals stat_tx_bad_fcs and stomped_fcs, and the packet is transmitted as it was received. ***Note:*** Statistics are reported as if there was no FCS error. ***Note:*** This port is available when **Include AXI4-Lite** is not selected and **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
ctl_tx_test_pattern_*	1	Input	Test pattern generation enable for the TX core. A value of 1 enables test mode as defined in Clause 49. Corresponds to MDIO register bit 3.42.7 as defined in Clause 45. Generates a scrambled idle pattern. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_tx_test_pattern_enable_*	1	Input	Test pattern enable for the RX core. A value of 1 enables test mode. Corresponds to MDIO register bit 3.42.2 as defined in Clause 45. Takes second precedence. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_tx_test_pattern_select_*	1	Input	Corresponds to MDIO register bit 3.42.1 as defined in Clause 45. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
ctl_tx_data_pattern_select_*	1	Input	Corresponds to MDIO register bit 3.42.0 as defined in Clause 45. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_tx_test_pattern_seed_a_*	58	Input	Corresponds to MDIO registers 3.34 through to 3.37 as defined in Clause 45. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_tx_test_pattern_seed_b_*	58	Input	Corresponds to MDIO registers 3.38 through to 3.41 as defined in Clause 45. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_tx_prbs31_test_pattern_enable_*	1	Input	Corresponds to MDIO register bit 3.42.4 as defined in Clause 45. Takes first precedence. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the GT Selection and Configuration tab and **Select Core** is **PCS/** in the Configuration tab.
ctl_tx_ipg_value_*	4	Input	***Note:*** This signal can be optionally present. The ctl_tx_ipg_value defines the target average minimum Inter Packet Gap (IPG, in bytes) inserted between rx_serdes_clk packets. Typical value is 12. The ctl_tx_ipg_value can also be programmed to a value in the 0 to 7 range, but in that case, it is interpreted as meaning "minimal IPG", so only Terminate code word IPG is inserted; no Idles are ever added in that case and that produces an average IPG of around 4 bytes when random-size packets are transmitted. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab **and Select Core** is **Ethernet MAC+PCS/ PMA-32/64-bit** and **Include FIFO Logic** is enabled in the MAC Options tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
ctl_tx_custom_preamble_enable_*	1	Input	When asserted, this signal enables the use of tx_preamblein as a custom preamble instead of inserting a standard preamble. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab and **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** and the **Include FIFO Logic** is disabled in the MAC Options tab or **Select Core** is **Ethernet MAC**.
ctl_tx_parity_err_response_*	1	Input	Parity error response by the TX Core. If this bit is set to 0, the core does not take any action if any parity errors are detected. If this bit is set to 1, the core stomps the outgoing FCS (i.e., bit-wise inverse) and asserts stat_tx_bad_fcs.
stat_tx_bad_parity_*	1	Output	Increment on any clock cycle where the user-generated parity is calculated as incorrect by the Tx parity checking logic.
stat_tx_local_fault_*	1	Output	A value of 1 indicates the receive decoder state machine is in the TX_INIT state. This output is level sensitive.
stat_tx_fifo_error_*	1	Output	Indicates when TX FIFO goes into an underflow or overflow condition. ***Note:*** This port is available when **Select Core** is **Ethernet PCS/PMA** in the Configuration tab.
stat_tx_total_bytes_*	5	Output	Increment for the total number of bytes transmitted. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_tx_total_packets_*	1	Output	Increment for the total number of packets transmitted. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_tx_total_good_bytes_*	14	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. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
stat_tx_total_good_packets_*	1	Output	Increment for the total number of good packets transmitted. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the GT Selection and Configuration tab,
stat_tx_bad_fcs_*	1	Output	Increment for packets greater than 64 bytes that have FCS errors. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_tx_packet_64_bytes_*	1	Output	Increment for good and bad packets transmitted that contain 64 bytes. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_tx_packet_65_127_bytes_*	1	Output	Increment for good and bad packets transmitted that contain 65 to 127 bytes. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_tx_packet_128_255_bytes_*	1	Output	Increment for good and bad packets transmitted that contain 128 to 255 bytes. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_tx_packet_256_511_bytes_*	1	Output	Increment for good and bad packets transmitted that contain 256 to 511 bytes. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_tx_packet_512_1023_bytes_*	1	Output	Increment for good and bad packets transmitted that contain 512 to 1,023 bytes. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_tx_packet_1024_1518_bytes_*	1	Output	Increment for good and bad packets transmitted that contain 1,024 to 1,518 bytes. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
stat_tx_packet_1519_1522_bytes_*	1	Output	Increment for good and bad packets transmitted that contain 1,519 to 1,522 bytes. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_tx_packet_1523_1548_bytes_*	1	Output	Increment for good and bad packets transmitted that contain 1,523 to 1,548 bytes. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_tx_packet_1549_2047_bytes_*	1	Output	Increment for good and bad packets transmitted that contain 1,549 to 2,047 bytes. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_tx_packet_2048_4095_bytes_*	1	Output	Increment for good and bad packets transmitted that contain 2,048 to 4,095 bytes. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_tx_packet_4096_8191_bytes_*	1	Output	Increment for good and bad packets transmitted that contain 4,096 to 8,191 bytes. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_tx_packet_8192_9215_bytes_*	1	Output	Increment for good and bad packets transmitted that contain 8,192 to 9,215 bytes. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_tx_packet_small_*	1	Output	Increment for all packets that are less than 64 bytes long. Packets that are less than 64 bytes are not transmitted. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
stat_tx_packet_large_*	1	Output	Increment for all packets that are more than 9,215 bytes long. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_tx_frame_error_*	1	Output	Increment for packets with tx_errin set to indicate an EOP abort. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
RX Path Control / Status / Statistics Signals			
ctl_rx_enable_*	1	Input	RX Enable. For normal operation, this input must be set to 1. When this input is set 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 user interface is idle. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab and **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC**.
ctl_rx_check_preamble_*	1	Input	When asserted, this input causes the Ethernet MAC to check the preamble of the received frame. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab and **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC**.
ctl_rx_check_sfd_*	1	Input	When asserted, this input causes the Ethernet MAC to check the start of frame Delimiter of the received frame. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab and **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC**.
ctl_rx_force_resync_*	1	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 (1 cycle minimum pulse) to force realignment. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab and **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC**.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
ctl_rx_delete_fcs_*	1	Input	Enable FCS removal by the RX core. If this bit is set to 0, the HSEC core does not remove the FCS of the incoming packet. If this bit is set to 1, the HSEC core deletes the FCS to the received packet. FCS is not deleted for packets that are =<8 bytes long. This input should only be changed while the corresponding reset input is asserted. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab and **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC**.
ctl_rx_ignore_fcs_*	1	Input	Enable FCS error checking at the user interface by the RX core. If this bit is set to 0, a packet received with an FCS error is sent with the rx_errout pin asserted during the last transfer (rx_eopout and rx_enaout sampled 1). If this bit is set to 1, the HSEC core does not flag an FCS error at the user interface. ***Note:*** The statistics are reported as if the packet is good. The stat_rx_bad_fcs signal, however, reports the error. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab and **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC**.
ctl_rx_max_packet_len_*	15	Input	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. Packets less than 4 bytes are dropped. The allowed value for this bus can range from 64 to 16,383. ctl_rx_max_packet_len[14] is reserved and must be set to 0. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab and **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC**.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
ctl_rx_min_packet_len_*	8	Input	Any packet shorter than this value 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 that are less than 4 bytes are dropped. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab and **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC**.
ctl_rx_process_lfi_*	1	Input	When this input is set to 1, the RX core expects and processes LF control codes coming in from the transceiver. When set to 0, the RX core ignores LF control codes coming in from the transceiver. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab and **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC**.
ctl_rx_test_pattern_*	1	Input	Test pattern checking enable for the RX core. A value of 1 enables test mode as defined in Clause 49. Corresponds to MDIO register bit 3.42.2 as defined in Clause 45. Checks for scrambled idle pattern. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_data_pattern_select_*			Corresponds to MDIO register bit 3.42.0 as defined in Clause 45. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab and **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** and the **Include FIFO Logic** is disabled.
ctl_rx_test_pattern_enable_*			Test pattern enable for the RX core. A value of 1 enables test mode. Corresponds to MDIO register bit 3.42.2 as defined in Clause 45. Takes second precedence. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab and **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** and the **Include FIFO Logic** is disabled.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
ctl_rx_rate_10g_25gn_*			This port indicates the lane rate of the mission mode. 0 corresponds to 25G and 1 corresponds to 10G. ***Note:*** This port is available when **Runtime Switching Mode** is selected in the Configuration tab.
ctl_tx_prbs31_test_pattern_enable_*	1	Input	Corresponds to MDIO register bit 3.42.1 as defined in Clause 45. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab and the **Select Core** is **PCS/**.
ctl_rx_custom_preamble_enable_*	1	Input	When asserted, this signal causes the preamble to be presented on rx_preambleout. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab and **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** and the **Include FIFO Logic** is disabled or **Select Core** is **Ethernet MAC**.
stat_rx_block_lock_*	4	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_framing_err_valid_*	1	Output	Valid indicator for stat_rx_framing_err. When 1 stat_rx_framing_err_0 is valid.
stat_rx_framing_err_*	3	Output	RX sync header bits framing error. 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 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_hi_ber_*	1	Output	High Bit Error Rate (BER) indicator. When set to 1, the BER is too high as defined by IEEE Std 802.3-2015. Corresponds to MDIO register bit 3.32.1 as defined in Clause 82.3. This output is level sensitive.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
stat_rx_bad_code_*	2	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 IEEE Std 802.3-2015. This output can be used to generate MDIO register 3.33:7:0 as defined in Clause 82.3.
stat_rx_bad_code_valid_*	1	Output	Indicates when stat_rx_bad_code is valid. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** in the Configuration tab.
stat_rx_error_valid_*	1	Output	Indicates when stat_rx_error is valid. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** in the Configuration tab.
stat_rx_error_*	8	Output	Test pattern mismatch increment. A non-zero value in any cycle indicates a mismatch occurred for the test pattern in the RX core. This output is only active when ctl_rx_test_pattern is set. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** in the Configuration tab.
stat_rx_fifo_error_*	1	Output	Indicates when RX FIFO goes into an underflow or overflow condition. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** in the Configuration tab.
stat_rx_total_packets_*	2	Output	Increment for the total number of packets received. ***Note:*** This port is available when **Select Core** is **Ethernet PCS/PMA** or **Ethernet MAC** in the Configuration tab.
stat_rx_total_good_packets_*	1	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. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_rx_total_bytes_*	6	Output	Increment for the total number of bytes received. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
stat_rx_total_good_bytes_*	14	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. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_rx_packet_small_*	2	Output	Increment for all packets that are less than 64 bytes long. Packets that are less than 4 bytes are dropped. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_rx_jabber_*	1	Output	Increment for packets longer than ctl_rx_max_packet_len with bad FCS. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_rx_packet_large_*	1	Output	Increment for all packets that are more than 9,215 bytes long. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_rx_oversize_*	1	Output	Increment for packets longer than ctl_rx_max_packet_len with good FCS. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_rx_undersize_*	2	Output	Increment for packets shorter than stat_rx_min_packet_len with good FCS. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_rx_toolong_*	1	Output	Increment for packets longer than ctl_rx_max_packet_len with good and bad FCS. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_rx_fragment_*	2	Output	Increment for packets shorter than stat_rx_min_packet_len with bad FCS. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
stat_rx_packet_64_bytes_*	1	Output	Increment for good and bad packets received that contain 64 bytes. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_rx_packet_65_127_bytes_*	1	Output	Increment for good and bad packets received that contain 65 to 127 bytes. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_rx_packet_128_255_bytes_*	1	Output	Increment for good and bad packets received that contain 128 to 255 bytes. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_rx_packet_256_511_bytes_*	1	Output	Increment for good and bad packets received that contain 256 to 511 bytes. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_rx_packet_512_1023_bytes_*	1	Output	Increment for good and bad packets received that contain 512 to 1,023 bytes. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_rx_packet_1024_1518_bytes_*	1	Output	Increment for good and bad packets received that contain 1,024 to 1,518 bytes. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_rx_packet_1519_1522_bytes_*	1	Output	Increment for good and bad packets received that contain 1,519 to 1,522 bytes. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_rx_packet_1523_1548_bytes_*	1	Output	Increment for good and bad packets received that contain 1,523 to 1,548 bytes. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
stat_rx_packet_1549_2047_bytes_*	1	Output	Increment for good and bad packets received that contain 1,549 to 2,047 bytes. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_rx_packet_2048_4095_bytes_*	1	Output	Increment for good and bad packets received that contain 2,048 to 4,095 bytes. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_rx_packet_4096_8191_bytes_*	1	Output	Increment for good and bad packets received that contain 4,096 to 8,191 bytes. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_rx_packet_8192_9215_bytes_*	1	Output	Increment for good and bad packets received that contain 8,192 to 9,215 bytes. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_rx_bad_fcs_*	2	Output	Bad FCS indicator. The value on this bus indicates packets 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. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_rx_packet_bad_fcs_*	1	Output	Increment for packets between 64 and ctl_rx_max_packet_len bytes that have FCS errors. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
stat_rx_stomped_fcs_*	2	Output	Stomped FCS indicator. The value on this bus indicates 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. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_rx_bad_preamble_*	1	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. ***Note:*** When an invalid preamble is detected, the stat_rx_bad_preamble signal is asserted regardless of the setting of the ctl_rx_check_preamble signal. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_rx_bad_sfd_*	1	Output	Increment bad SFD. This signal indicates if the Ethernet packet received was preceded by a valid SFD. A value of 1 indicates that an invalid SFD was received. ***Note:*** When an invalid SFD is detected, the stat_rx_bad_sfd signal is asserted regardless of the setting of the ctl_rx_check_sfd signal. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_rx_got_signal_os_*	1	Output	Signal OS indication. If this bit is sampled as a 1, it indicates that a Signal OS word was received. ***Note:*** Signal OS should not be received in an Ethernet network. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
stat_rx_test_pattern_mismatch_*	2	Output	Test pattern mismatch increment. A nonzero 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. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** in the Configuration tab.
stat_rx_truncated_*	1	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. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** or **Ethernet MAC** in the Configuration tab.
stat_rx_local_fault_*	1	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_remote_fault_*	1	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 does not exist. This output is level sensitive. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** in the Configuration tab.
stat_rx_internal_local_fault_*	1	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. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** in the Configuration tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
stat_rx_received_local_fault_*	1	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. ***Note:*** This port is available when **Select Core** is **Ethernet MAC+PCS/PMA-32/64-bit** in the Configuration tab.
stat_rx_valid_ctrl_code_*	1	Output	Indicates that a PCS block with a valid control code was received.
stat_rx_status	1	Output	Indicates the link status. ***Note:*** This port is available only for MAC+PCS/PMA-32/64 bit in the Configurations tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
TX Pause Interface Control / Status / Statistics Signals Ports under this section will be available when **Enable TX Flow Control Logic** is selected from the MAC Options tab and **Select Core** is **Ethernet MAC+PCS/PMA 64-bit**.			
ctl_tx_pause_req_*	9	Input	If a bit of this bus is set to 1, the 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.
ctl_tx_pause_enable_*	9	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. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the GT Selection and Configuration tab.
ctl_tx_resend_pause_*	1	Input	Retransmit 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.
ctl_tx_pause_quanta0_*	16	Input	These buses indicate the quanta to be transmitted for each of the eight priorities in priority-based pause operation and the global pause operation. The value for ctl_tx_pause_quanta[8] is used for global pause operation. All other values are used for priority pause operation. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_tx_pause_quanta1_*	16	Input	These buses indicate the quanta to be transmitted for each of the eight priorities in priority-based pause operation and the global pause operation. The value for ctl_tx_pause_quanta[8] is used for global pause operation. All other values are used for priority pause operation. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
ctl_tx_pause_quanta2_*	16	Input	These buses indicate the quanta to be transmitted for each of the eight priorities in priority-based pause operation and the global pause operation. The value for ctl_tx_pause_quanta[8] is used for global pause operation. All other values are used for priority pause operation. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_tx_pause_quanta3_*	16	Input	These buses indicate the quanta to be transmitted for each of the eight priorities in priority-based pause operation and the global pause operation. The value for ctl_tx_pause_quanta[8] is used for global pause operation. All other values are used for priority pause operation. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_tx_pause_quanta4_*	16	Input	These buses indicate the quanta to be transmitted for each of the eight priorities in priority-based pause operation and the global pause operation. The value for ctl_tx_pause_quanta[8] is used for global pause operation. All other values are used for priority pause operation. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_tx_pause_quanta5_*	16	Input	These buses indicate the quanta to be transmitted for each of the eight priorities in priority-based pause operation and the global pause operation. The value for ctl_tx_pause_quanta[8] is used for global pause operation. All other values are used for priority pause operation. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
ctl_tx_pause_quanta6_*	16	Input	These buses indicate the quanta to be transmitted for each of the eight priorities in priority-based pause operation and the global pause operation. The value for ctl_tx_pause_quanta[8] is used for global pause operation. All other values are used for priority pause operation. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_tx_pause_quanta7_*	16	Input	These buses indicate the quanta to be transmitted for each of the eight priorities in priority-based pause operation and the global pause operation. The value for ctl_tx_pause_quanta[8] is used for global pause operation. All other values are used for priority pause operation. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_tx_pause_quanta8_*	16	Input	These buses indicate the quanta to be transmitted for each of the eight priorities in priority-based pause operation and the global pause operation. The value for ctl_tx_pause_quanta[8] is used for global pause operation. All other values are used for priority pause operation. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_tx_pause_refresh_timer0_*	16	Input	This bus sets the retransmission time of pause packets for each of the eight priorities in priority-based pause operation and the global pause operation. The value for ctl_tx_pause_refresh_timer[8] is used for global pause operation. All other values are used for priority pause operation. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
ctl_tx_pause_refresh_timer1_*	16	Input	This bus sets the retransmission time of pause packets for each of the eight priorities in priority-based pause operation and the global pause operation. The value for ctl_tx_pause_refresh_timer[8] is used for global pause operation. All other values are used for priority pause operation. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_tx_pause_refresh_timer2_*	16	Input	This bus sets the retransmission time of pause packets for each of the eight priorities in priority-based pause operation and the global pause operation. The value for ctl_tx_pause_refresh_timer[8] is used for global pause operation. All other values are used for priority pause operation. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_tx_pause_refresh_timer3_*	16	Input	This bus sets the retransmission time of pause packets for each of the eight priorities in priority-based pause operation and the global pause operation. The value for ctl_tx_pause_refresh_timer[8] is used for global pause operation. All other values are used for priority pause operation. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_tx_pause_refresh_timer4_*	16	Input	This bus sets the retransmission time of pause packets for each of the eight priorities in priority-based pause operation and the global pause operation. The value for ctl_tx_pause_refresh_timer[8] is used for global pause operation. All other values are used for priority pause operation. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
ctl_tx_pause_refresh_timer5_*	16	Input	This bus sets the retransmission time of pause packets for each of the eight priorities in priority-based pause operation and the global pause operation. The value for ctl_tx_pause_refresh_timer[8] is used for global pause operation. All other values are used for priority pause operation. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_tx_pause_refresh_timer6_*	16	Input	This bus sets the retransmission time of pause packets for each of the eight priorities in priority-based pause operation and the global pause operation. The value for ctl_tx_pause_refresh_timer[8] is used for global pause operation. All other values are used for priority pause operation. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_tx_pause_refresh_timer7_*	16	Input	This bus sets the retransmission time of pause packets for each of the eight priorities in priority-based pause operation and the global pause operation. The value for ctl_tx_pause_refresh_timer[8] is used for global pause operation. All other values are used for priority pause operation. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_tx_pause_refresh_timer8_*	16	Input	This bus sets the retransmission time of pause packets for each of the eight priorities in priority-based pause operation and the global pause operation. The value for ctl_tx_pause_refresh_timer[8] is used for global pause operation. All other values are used for priority pause operation. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_tx_da_gpp_*	48	Input	Destination address for transmitting global pause packets. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
ctl_tx_sa_gpp_*	48	Input	Source address for transmitting global pause packets. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_tx_ethertype_gpp_*	16	Input	Ethertype for transmitting global pause packets. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_tx_opcode_gpp_*	16	Input	Opcode for transmitting global pause packets. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_tx_da_ppp_*	48	Input	Destination address for transmitting priority pause packets. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_tx_sa_ppp_*	48	Input	Source address for transmitting priority pause packets. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_tx_ethertype_ppp_*	16	Input	Ethertype for transmitting priority pause packets. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_tx_opcode_ppp_*	16	Input	Opcode for transmitting priority pause packets. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
stat_tx_pause_valid_*	9	Output	If a bit of this bus is set to 1, the HSEC 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.
stat_tx_unicast_*	1	Output	Increment for good unicast packets.
stat_tx_multicast_*	1	Output	Increment for good multicast packets.
stat_tx_broadcast_*	1	Output	Increment for good broadcast packets.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
stat_tx_vlan_*	1	Output	Increment for good 802.1Q tagged VLAN packets.
stat_tx_pause_*	1	Output	Increment for 802.3x Ethernet MAC Pause packet with good FCS.
stat_tx_user_pause_*	1	Output	Increment for priority-based pause packets with good FCS.
RX Pause Interface Control / Status / Statistics Signals Ports under this section will be available when **Enable RX Flow Control Logic** is selected from the MAC Options tab and **Select Core** is **Ethernet MAC+PCS/PMA 64-bit**.			
ctl_rx_forward_control_*	1	Input	A value of 1 indicates that the CORE forwards control packets to you. A value of 0 causes CORE to drop control packets. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_pause_ack_*	9	Input	Pause acknowledge signal. This bus is used to acknowledge the receipt of the pause frame from the user logic.
ctl_rx_check_ack_*	1	Input	Wait for acknowledge. If this input is set to 1, the CORE uses the ctl_rx_pause_ack[8:0] bus for pause processing. If this input is set to 0, ctl_rx_pause_ack[8:0] is not used. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_pause_enable_*	9	Input	RX pause enable signal. This input is used to enable the processing of the pause quanta for the corresponding priority. ***Note:*** This signal only affects the RX user interface, not the pause processing logic. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_enable_gcp_*	1	Input	A value of 1 enables global control packet processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_check_mcast_gcp_*	1	Input	A value of 1 enables global control multicast destination address processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
ctl_rx_check_ucast_gcp_*	1	Input	A value of 1 enables global control unicast destination address processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_pause_da_ucast_*	48	Input	Unicast destination address for pause processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_check_sa_gcp_*	1	Input	A value of 1 enables global control source address processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_pause_sa_*	48	Input	Source address for pause processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_check_etype_gcp_*	1	Input	A value of 1 enables global control ethertype processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_etype_gcp_*	16	Input	Ethertype field for global control processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_check_opcode_gcp_*	1	Input	A value of 1 enables global control opcode processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_opcode_min_gcp_*	16	Input	Minimum global control opcode value. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_opcode_max_gcp_*	16	Input	Maximum global control opcode value. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
ctl_rx_enable_pcp_*	1	Input	A value of 1 enables priority control packet processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_check_mcast_pcp_*	1	Input	A value of 1 enables priority control multicast destination address processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_check_ucast_pcp_*	1	Input	A value of 1 enables priority control unicast destination address processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_pause_da_mcast_*	48	Input	Multicast destination address for pause processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_check_sa_pcp_*	1	Input	A value of 1 enables priority control source address processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_check_etype_pcp_*	1	Input	A value of 1 enables priority control ethertype processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_etype_pcp_*	16	Input	Ethertype field for priority control processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_check_opcode_pcp_*	1	Input	A value of 1 enables priority control opcode processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_opcode_min_pcp_*	16	Input	Minimum priority control opcode value. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
ctl_rx_opcode_max_pcp_*	16	Input	Maximum priority control opcode value. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_enable_gpp_*	1	Input	A value of 1 enables global pause packet processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_check_mcast_gpp_*	1	Input	A value of 1 enables global pause multicast destination address processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_check_ucast_gpp_*	1	Input	A value of 1 enables global pause unicast destination address processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_check_sa_gpp_*	1	Input	A value of 1 enables global pause source address processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_check_etype_gpp_*	1	Input	A value of 1 enables global pause ethertype processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_etype_gpp_*	16	Input	Ethertype field for global pause processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_check_opcode_gpp_*	1	Input	A value of 1 enables global pause opcode processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_opcode_gpp_*	16	Input	Global pause opcode value. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
ctl_rx_enable_ppp_*	1	Input	A value of 1 enables priority pause packet processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_check_mcast_ppp_*	1	Input	A value of 1 enables priority pause multicast destination address processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_check_ucast_ppp_*	1	Input	A value of 1 enables priority pause unicast destination address processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_check_sa_ppp_*	1	Input	A value of 1 enables priority pause source address processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_check_etype_ppp_*	1	Input	A value of 1 enables priority pause ethertype processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_etype_ppp_*	16	Input	Ethertype field for priority pause processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_check_opcode_ppp_*	1	Input	A value of 1 enables priority pause opcode processing. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_rx_opcode_ppp_*	16	Input	Priority pause opcode value. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
stat_rx_unicast_*	1	Output	Increment for good unicast packets.
stat_rx_multicast_*	1	Output	Increment for good multicast packets.
stat_rx_broadcast_*	1	Output	Increment for good broadcast packets.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
stat_rx_vlan_*	1	Output	Increment for good 802.1Q tagged VLAN packets.
stat_rx_pause_*	1	Output	Increment for 802.3x Ethernet MAC Pause packet with good FCS.
stat_rx_user_pause_*	1	Output	Increment for priority-based pause packets with good FCS.
stat_rx_inrangeerr_*	1	Output	Increment for packets with Length field error but with good FCS.
stat_rx_pause_valid_*	9	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_*	16	Output	These buses indicate the quanta received for each of the eight priorities in priority-based pause operation and global pause operation. If an 802.3x Ethernet MAC Pause packet is received, the quanta are placed in value [8].
stat_rx_pause_quanta1_*	16	Output	These buses indicate the quanta received for each of the eight priorities in priority-based pause operation and global pause operation. If an 802.3x Ethernet MAC Pause packet is received, the quanta are placed in value [8].
stat_rx_pause_quanta2_*	16	Output	These buses indicate the quanta received for each of the eight priorities in priority-based pause operation and global pause operation. If an 802.3x Ethernet MAC Pause packet is received, the quanta are placed in value [8].
stat_rx_pause_quanta3_*	16	Output	These buses indicate the quanta received for each of the eight priorities in priority-based pause operation and global pause operation. If an 802.3x Ethernet MAC Pause packet is received, the quanta are placed in value [8].
stat_rx_pause_quanta4_*	16	Output	These buses indicate the quanta received for each of the eight priorities in priority-based pause operation and global pause operation. If an 802.3x Ethernet MAC Pause packet is received, the quanta are placed in value [8].

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
stat_rx_pause_quanta5_*	16	Output	These buses indicate the quanta received for each of the eight priorities in priority-based pause operation and global pause operation. If an 802.3x Ethernet MAC Pause packet is received, the quanta are placed in value [8].
stat_rx_pause_quanta6_*	16	Output	These buses indicate the quanta received for each of the eight priorities in priority-based pause operation and global pause operation. If an 802.3x Ethernet MAC Pause packet is received, the quanta are placed in value [8].
stat_rx_pause_quanta7_*	16	Output	These buses indicate the quanta received for each of the eight priorities in priority-based pause operation and global pause operation. If an 802.3x Ethernet MAC Pause packet is received, the quanta are placed in value [8].
stat_rx_pause_quanta8_*	16	Output	These buses indicate the quanta received for each of the eight priorities in priority-based pause operation and global pause operation. If an 802.3x Ethernet MAC Pause packet is received, the quanta are placed in value [8].
stat_rx_pause_req_*	9	Output	Pause request signal. When the RX receives a valid pause frame, it sets the corresponding bit of this bus to a 1 and keep it at 1 until the pause packet has been processed.
IEEE 1588 TX/RX Interface Control / Status / Statistics Signals Ports under this section will be available when **Enable_Time_Stamping** is selected from the MAC Options tab.			
ctl_tx_systemtimerin_*	80	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.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
ctl_rx_systemtimerin_*	80	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.
ctl_tx_ptp_1step_enable_*	1	Input	When set to 1, this bit enables 1-step operation.
ctl_tx_ptp_latency_adjust_*	11	Input	This bus can be used to adjust the 1-step TX timestamp with respect to the 2-step timestamp. The units of bits [10:3] are nanoseconds and bits [2:0] are fractional nanoseconds. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_ptp_transpclk_mode_*	1	Input	When set to 1, this input places the timestamping logic into transparent clock mode. In this mode, the system timer input is interpreted as a correction value. The TX will add the correction value to the TX timestamp according to the process defined in IEEE 1588v2. The sign bit of the correction value is assumed to be 0 (positive time). It is expected that the corresponding incoming PTP packet correction field has already been adjusted with the proper RX timestamp. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
stat_tx_ptp_fifo_read_error_*	1	Output	Transmit PTP FIFO write error. A value of 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_write_error_*	1	Output	Transmit PTP FIFO read error. A value of 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 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
tx_ptp_1588op_in_*	2	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_*	16	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 PTP frame that it sent for transmission.
tx_ptp_tstamp_valid_out_*	1	Output	This bit indicates that a valid timestamp is being presented on the tx.
tx_ptp_tstamp_tag_out_*	16	Output	Tag output corresponding to tx_ptp_tag_field_in[15:0].
tx_ptp_tstamp_out_*	80	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.
rx_ptp_tstamp_valid_out_*	1	Output	This bit indicates that a valid timestamp is being presented on the rx. ***Note:*** This will be present only when core is Ethernet MAC+PCS/PMA-32/64-bit.
rx_ptp_tstamp_out_*	80	Output	Time stamp for the received packet SOP corresponding to the time at which it passed the capture plane. Note that this signal will be valid starting at the same clock cycle during which the SOP is asserted for one of the segments. The representation of the bits contained in this bus is the same as the timer input.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
tx_ptp_upd_chksum_in_*	1	Input	TX UPD checksum value. ***Note:*** This port is available when PTP Operation mode is selected as one-step in MAC options tab.
tx_ptp_tstamp_offset_in_*	16	Input	TX PTP timestamp offset. ***Note:*** This port is available when PTP Operation mode is selected as one-step in MAC options tab.
tx_ptp_chksum_offset_in_*	16	Input	TX PTP check sum offset. ***Note:*** This port is available when PTP Operation mode is selected as one-step in MAC options tab.
AN and LT Interface Control / Status / Statistics Signals Ports under this section will be available when the **Include AN/LT Logic** is selected from the Configuration tab.			
an_clk_*	1	Input	Input Clock for the Auto-Negotiation circuit. ***Note:*** This should be free running clock.
an_reset_*	1	Input	Asynchronous active-High reset corresponding to an_clk domain.
an_loc_np_data_*	48	Input	Local Next Page codeword. This is the 48 bit codeword used if the 'loc_np' input is set. In this data field, the bits NP, ACK, & T, bit positions 15, 14, 12, & 11, are not transferred as part of the next page codeword. These bits are generated in the AN IP. However, the Message Protocol bit, MP, in bit position 13, is transferred.
an_lp_np_data_*	48	Output	Link Partner Next Page Data. This 48 bit word is driven by the AN IP with the 48 bit next page codeword from the remote link partner.
lt_tx_sof_*	4	Output	This is a link training signal that is asserted for one tx_serdes_clk period at the start of each training frame. It is provided for applications that need to count training frames or synchronize events to the output of the training frames.
ctl_autoneg_enable_*	1	Input	Enable signal for auto-negotiation. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
ctl_autoneg_bypass_*	1	Input	Input to disable auto-negotiation and bypass the auto-negotiation function. If this input is asserted, then auto-negotiation is turned off, but the PCS is connected to the output to allow operation. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_an_nonce_seed_*	8	Input	8-bit seed to initialize the nonce field polynomial generator. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_an_pseudo_sel_*	1	Input	Selects the polynomial generator for the bit 49 random bit generator. If this input is 1, then the polynomial is x7+x6+1. If this input is zero, then the polynomial is x7+x3+1. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_restart_negotiation_*	1	Input	This input is used to trigger a restart of the auto-negotiation, regardless of what state the circuit is currently in. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_an_local_fault_*	1	Input	This input signal is used to set the local_fault bit of the transmit link codeword. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_an_pause_*	1	Input	This input signal is used to set the PAUSE bit, (C0), of the transmit link codeword. This signal might not be present if the core does not support pause. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
ctl_an_asmdir_*	1	Input	This input signal is used to set the ASMDIR bit, (C1), of the transmit link codeword. This signal might not be present if the core does not support pause. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_an_fec_10g_request_*	1	Input	This signal is used to signal the link partner that the local station is requesting clause 74 FEC on the 10Gb/s lane protocols. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_an_fec_25g_rs_request_*	1	Input	This signal is used to signal the link partner that the local station is requesting rs FEC (clause 91 or 108) on the 25Gb/s lane protocols. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_an_fec_25g_baser_request_*	1	Input	Indicates the baser FEC request. ***Note:*** Note: This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_an_fec_ability_override_*	1	Input	Used to set the clause 74 FEC ability bit in the transmit link codeword. If this input is set, then the FEC ability bit in the transmit link codeword is cleared. This signal might not be present if the IP core does not support clause 74 FEC. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_an_loc_np_*	1	Input	Local Next Page indicator. If this bit is 1, then the AN IP transfers the next page word at input loc_np_data to the remote link partner. If this bit is 0, then the AN IP does not initiate the next page protocol. If the link partner has next pages to send, and the loc_np bit is clear, then the AN IP transfers null message pages.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
ctl_an_lp_np_ack_*	1	Input	Link Partner Next Page Acknowledge. This is used to signal the AN IP that the next page data from the remote link partner at output pin lp_np_data has been read by the local host. When this signal goes High, the AN IP acknowledges reception of the next page codeword to the remote link partner and initiate transfer of the next codeword. During this time, the AN IP removes the lp_np signal until the new next page information is available.
ctl_an_cl91_fec_request_*	1	Input	This bit is used to request clause 91 FEC. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_an_cl91_fec_ability_*	1	Input	This bit is used to set clause 91 FEC ability. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
ctl_an_ability_1000base_kx_*	1	Input	These inputs identify the Ethernet protocol abilities that are advertised in the transmit link codeword to the link partner. A value of 1 indicates that the interface advertises that it supports the protocol.
ctl_an_ability_10gbase_kx4_*	1	Input	
ctl_an_ability_10gbase_kr_*	1	Input	
ctl_an_ability_40gbase_kr4_*	1	Input	
ctl_an_ability_40gbase_cr4_*	1	Input	
ctl_an_ability_100gbase_cr10_*	1	Input	
ctl_an_ability_100gbase_kp4_*	1	Input	
ctl_an_ability_100gbase_kr4_*	1	Input	
ctl_an_ability_100gbase_cr4_*	1	Input	
ctl_an_ability_25gbase_krcr_s_*	1	Input	
ctl_an_ability_25gbase_krcr_*	1	Input	
ctl_an_ability_25gbase_kr1_*	1	Input	
ctl_an_ability_25gbase_cr1_*	1	Input	
ctl_an_ability_50gbase_kr2_*	1	Input	
ctl_an_ability_50gbase_cr2_*	1	Input	
ctl_lt_training_enable_*	1	Input	Enables link training. When link training is disabled, all PCS lanes function in mission mode. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_lt_restart_training_*	1	Input	This signal triggers a restart of link training regardless of the current state. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_lt_rx_trained_*	4	Input	This signal is asserted to indicate that the receiver FIR filter coefficients have all been set, and that the receiver portion of training is complete. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
ctl_lt_preset_to_tx_*	4	Input	This signal is used to set the value of the preset bit that is transmitted to the link partner in the control block of the training frame. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_lt_initialize_to_tx_*	4	Input	This signal is used to set the value of the initialize bit that is transmitted to the link partner in the control block of the training frame. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_lt_pseudo_seed0_*	11	Input	This 11-bit signal seeds the training pattern generator. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_lt_k_p1_to_tx0_*	2	Input	This 2-bit field is used to set the value of the k+1 coefficient update field that is transmitted to the link partner in the control block of the training frame. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_lt_k0_to_tx0_*	2	Input	This 2-bit field is used to set the value of the k0 coefficient update field that is transmitted to the link partner in the control block of the training frame. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_lt_k_m1_to_tx0_*	2	Input	This 2-bit field is used to set the value of the 'k-1' coefficient update field that is transmitted to the link partner in the control block of the training frame. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_lt_stat_p1_to_tx0_*	2	Input	This 2-bit field is used to set the value of the 'k+1' coefficient update status that is transmitted to the link partner in the status block of the training frame. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
ctl_lt_stat0_to_tx0_*	2	Input	This 2-bit field is used to set the value of the 'k0' coefficient update status that is transmitted to the link partner in the status block of the training frame. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_lt_stat_m1_to_tx0_*	2	Input	This 2-bit field is used to set the value of the 'k-1' coefficient update status that is transmitted to the link partner in the status block of the training frame. ***Note:*** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
stat_an_link_cntl_1000base_kx_*	2	Output	Link Control outputs from the auto-negotiation controller for the various Ethernet protocols. Settings are as follows: 00: DISABLE; PCS is disconnected; 01: SCAN_FOR_CARRIER; RX is connected to PCS; 11: ENABLE; PCS is connected for mission mode operation. 10: not used
stat_an_link_cntl_10gbase_kx4_*	2	Output	
stat_an_link_cntl_10gbase_kr_*	2	Output	
stat_an_link_cntl_40gbase_kr4_*	2	Output	
stat_an_link_cntl_40gbase_cr4_*	2	Output	
stat_an_link_cntl_100gbase_cr10_*	2	Output	
stat_an_link_cntl_100gbase_kp4_*	2	Output	
stat_an_link_cntl_100gbase_kr4_*	2	Output	
stat_an_link_cntl_100gbase_cr4_*	2	Output	
stat_an_link_cntl_25gbase_krcr_s_*	2	Output	
stat_an_link_cntl_25gbase_krcr_*	2	Output	
stat_an_link_cntl_25gbase_kr1_*	2	Output	
stat_an_link_cntl_25gbase_cr1_*			
stat_an_link_cntl_50gbase_kr2_*			
stat_an_link_cntl_50gbase_cr2_*			
stat_an_fec_enable_*	1	Output	Used to enable the use of clause 74 FEC on the link.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
stat_an_tx_pause_enable_*	1	Output	Used to enable station-to-station (global) pause packet generation in the transmit path to control data flow in the receive path.
stat_an_rx_pause_enable_*	1	Output	Used to enable station-to-station (global) pause packet interpretation in the receive path, in order to control data flow from the transmitter.
stat_an_autoneg_complete_*	1	Output	Indicates the auto-negotiation is complete and rx link status from the PCS has been received.
stat_an_parallel_detection_fault_*	1	Output	Indicated a parallel detection fault during auto-negotiation.
stat_an_lp_ability_1000base_kx_*	1	Output	These signals indicate the advertised protocol from the link partner. They all become valid when the output signal stat_an_lp_ability_valid is asserted. A value of 1 indicates that the protocol is advertised as supported by the link partner.
stat_an_lp_ability_10gbase_kx4_*	1	Output	
stat_an_lp_ability_10gbase_kr_*	1	Output	
stat_an_lp_ability_40gbase_kr4_*	1	Output	
stat_an_lp_ability_40gbase_cr4_*	1	Output	
stat_an_lp_ability_100gbase_cr10_*	1	Output	
stat_an_lp_ability_100gbase_kp4_*	1	Output	
stat_an_lp_ability_100gbase_kr4_*	1	Output	
stat_an_lp_ability_100gbase_cr4_*	1	Output	
stat_an_lp_ability_25gbase_krcr_s_*	1	Output	
stat_an_lp_ability_25gbase_krcr_*	1	Output	
stat_an_lp_ability_25gbase_cr1_*	1	Output	Indicates the advertised protocol from the link partner. Becomes valid when the output signal stat_AN_lp_Extended_Ability_Valid is asserted. A value of 1 indicates that the protocol is advertised as supported by the link partner.
stat_an_rxcdrhold_*	1	Output	Indicates the rx cdr hold signal.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
stat_an_lp_pause_*	1	Output	This signal indicates the advertised value of the PAUSE bit, (C0), in the receive link codeword from the link partner. It becomes valid when the output signal stat_an_lp_ability_valid is asserted.
stat_an_lp_asm_dir_*	1	Output	This signal indicates the advertised value of the ASMDIR bit, (C1), in the receive link codeword from the link partner. It becomes valid when the output signal stat_an_lp_ability_valid is asserted.
stat_an_lp_rf_*	1	Output	This bit indicates link partner remote fault.
stat_an_lp_fec_10g_ability_*	1	Output	This signal indicates the clause 74 FEC ability associated with 10Gb/s lane protocols that is being advertised by the link partner. It becomes valid when the output signal stat_an_lp_ability_valid is asserted.
stat_an_lp_fec_10g_request_*	1	Output	This signal indicates that the link partner is requesting that the clause 74 FEC be used on the 10Gb/s lane protocols.It becomes valid when the output signal stat_an_lp_ability_valid is asserted.
stat_an_lp_fec_25g_rs_request_*	1	Output	This signal indicates that the link partner is requesting the clause 91 (or 108) rs FEC be used for the 25gb/s lane protocols. It becomes valid when the output signal stat_an_lp_ability_valid is asserted.
stat_an_lp_fec_25g_baser_request_*	1	Output	This signal indicates that the link partner is requesting the clause 74 FEC be used for the 25Gb/s lane base-r protocols. It becomes valid when the output signal stat_an_lp_ability_valid is asserted.
stat_an_lp_autoneg_able_*	1	Output	This output signal indicates that the link partner is able to perform auto-negotiation. It becomes valid when the output signal stat_an_lp_ability_valid is asserted.
stat_an_lp_ability_valid_*	1	Output	This signal indicates when all of the link partner advertisements become valid.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
stat_an_loc_np_ack_*	1	Output	This signal is used to indicate to the local host that the local next page data, presented at input pin loc_np_data, has been taken. This signal pulses High for 1 clock period when the AN IP samples the next page data on input pin loc_np_data. When the local host detects this signal High, it must replace the 48 bit next page codeword at input pin 'loc_np_data' with the next 48 bit codeword to be sent. If the local host has no more next pages to send, then it must clear the loc_np input.
stat_an_lp_np_*	1	Output	Link Partner Next Page. This signal is used to indicate that there is a valid 48 bit next page codeword from the remote link partner at output pin lp_np_data. This signal is driven low when the lp_np_ack input signal is driven high, indicating that the local host has read the next page data. It remains low until the next codeword becomes available on the lp_np_data output pin, then the lp_np output is driven high again.
stat_an_lp_ability_25gbase_kr1_*	1	Output	Indicates the advertised protocol from the link partner. Becomes valid when the output signal stat_an_lp_extended_ability_valid is asserted. A value of 1 indicates that the protocol is advertised as supported by the link partner.
stat_an_link_cntl_25gbase_cr1_*	1	Output	Indicates the advertised protocol from the link partner. Becomes valid when the output signal stat_an_lp_extended_ability_valid is asserted. A value of 1 indicates that the protocol is advertised as supported by the link partner.
stat_an_lp_ability_50gbase_kr2_*	1	Output	Indicates the advertised protocol from the link partner. Becomes valid when the output signal stat_an_lp_extended_ability_valid is asserted. A value of 1 indicates that the protocol is advertised as supported by the link partner.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
stat_an_lp_ability_50gbase_cr2_*	1	Output	Indicates the advertised protocol from the link partner. Becomes valid when the output signal stat_an_lp_extended_ability_valid is asserted. A value of 1 indicates that the protocol is advertised as supported by the link partner.
stat_an_lp_ability_extended_fec_*	4	Output	This output indicates the extended FEC abilities.
stat_an_rs_fec_enable_*	1	Output	Used to enable the use of clause 91 FEC on the link.
stat_an_lp_extended_ability_valid_*	1	Output	When this bit is 1, it indicates that the detected extended abilities are valid.
stat_lt_signal_detect_*	4	Output	This signal indicates when the respective link training state machine has entered the SEND_DATA state, in which normal PCS operation can resume.
stat_lt_training_*	4	Output	This signal indicates when the respective link training state machine is performing link training.
stat_lt_training_fail_*	4	Output	This signal is asserted during link training if the corresponding link training state machine detects a time-out during the training period.
stat_lt_rx_sof_*	4	Output	This output is High for 1 RX SerDes clock cycle to indicate the start of the link training frame.
stat_lt_frame_lock_*	4	Output	When link training has begun, these signals are asserted, for each PMD lane, when the corresponding link training receiver is able to establish a frame synchronization with the link partner.
stat_lt_preset_from_rx_*	4	Output	This signal reflects the value of the preset control bit received in the control block from the link partner.
stat_lt_initialize_from_rx_*	4	Output	This signal reflects the value of the initialize control bit received in the control block from the link partner.
stat_lt_k_p1_from_rx0_*	2	Output	This 2-bit field indicates the update control bits for the 'k+1' coefficient, as received from the link partner in the control block.
stat_lt_k0_from_rx0_*	2	Output	This 2-bit field indicates the update control bits for the 'k0' coefficient, as received from the link partner in the control block.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
stat_lt_k_m1_from_rx0_*	2	Output	This 2-bit field indicates the update control bits for the 'k-1' coefficient, as received from the link partner in the control block.
stat_lt_stat_p1_from_rx0_*	2	Output	This 2-bit field indicates the update status bits for the 'k+1' coefficient, as received from the link partner in the status block.
stat_lt_stat0_from_rx0_*	2	Output	This 2-bit fields indicates the update status bits for the 'k0' coefficient, as received from the link partner in the status block.
stat_lt_stat_m1_from_rx0_*	2	Output	This 2-bit field indicates the update status bits for the 'k-1' coefficient, as received from the link partner in the status block.
Clause 74 FEC Interface Control / Status / Statistics Signals Ports under this section will be available when **Clause 74 (BASE-KR FEC)** is selected from the Configuration tab.			
ctl_fec_tx_enable_*	1	Input	Asserted to enable the clause 74 FEC encoding on the transmitted data.
ctl_fec_rx_enable_*	1	Input	Asserted to enable the clause 74 FEC decoding of the received data.
ctl_fec_enable_error_to_pcs_*	1	Input	Clause 74 FEC enable error to pcs.
stat_fec_inc_correct_count_*	4	Output	This signal will be asserted roughly every 32 words, while the ctl_rx_fec_enable is asserted, if the FEC decoder detected and corrected a bit errors in the corresponding frame.
stat_fec_inc_cant_correct_count_*	4	Output	This signal will be asserted roughly every 32 words, while the ctl_rx_fec_enable is asserted, if the FEC decoder detected bit
stat_fec_lock_error_*	4	Output	stat_fec_lock_error_* is asserted if the FEC decoder has been unable to detect the frame boundary after about 5 ms. It is cleared when the frame boundary is detected.
stat_fec_rx_lock_*	4	Output	This signal is asserted while the ctl_fec_rx_enable is asserted when the FEC decoder detects the frame boundary.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
IEEE Clause 108 (RS-FEC) Control / Status /Statistics Signals Ports under this section will be available when IEEE Clause 108 (RS-FEC) is selected from Configuration tab			
ctl_rx_rsfec_enable_correction_*	1	Input	The setting on this bit takes effect after rx_resetn has been asserted Low (~rx_serdes_reset). New value is sampled on first cycle on reset. Equivalent to MDIO register 1.200.0 • 0: Decoder performs error detection without error correction (see IEEE 802.3by section 91.5.3.3). • 1: the decoder also performs error correction.
ctl_rx_rsfec_enable_indication_*	1	Input	The setting on this bit takes effect after rx_resetn has been asserted Low (~rx_serdes_reset). New value sampled on the first cycle on reset. Equivalent to MDIO register 1.200.1 • 0: Bypass the error indication function (see IEEE Std 802.3by section 91.5.3.3). • 1: Decoder indicates errors to the PCS sublayer.
ctl_rsfec_enable_*	1	Input	The setting on this bit takes effect after rx_resetn has been asserted Low (~rx_serdes_reset). New value is sampled on the first cycle on reset. Enable RS-FEC function. ***Note:*** Some variants of the 10G/25G Subsystem can have individual RX and TX enable signals.
ctl_rsfec_ieee_error_indication_mode_*	1	Input	The setting on this bit takes effect after rx_resetn has been asserted Low (~rx_serdes_reset). New value is sampled on the first cycle on reset. • 1: Core conforms to the IEEE RS-FEC specification. • 0: If ctl_rx_rsfec_enable_correction and ctl_rx_rsfec_enable_indication are set to zero, the RS decoder is bypassed.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
ctl_rsfec_consortium_25g_*	1	Input	Switches between IEEE Clause 108 and 25G Ethernet Consortium mode. The setting on this bit takes effect after rx_resetn has been asserted Low (~rx_serdes_reset). New value is sampled on the first cycle on reset. • 1 = 25G Consortium specification mode. • 0 = IEEE 802.3by mode. ***Note:*** Some variants of the 10G/25G Subsystem can have individual RX and TX consortium signals.
stat_rx_rsfec_hi_ser_*	1	Output	Set to one if the number of RS-FEC symbol errors in a window of 8192 codewords exceeds the threshold K = 417 and is set to zero otherwise.
stat_rx_rsfec_lane_alignment_status_*	1	Output	A value of 1 indicates that the RX RS-FEC block has achieved alignment on the data from the transceiver.
stat_rx_rsfec_corrected_cw_inc_*	1	Output	Increment for corrected errors.
stat_rx_rsfec_uncorrected_cw_inc_*	1	Output	Increment for uncorrected errors.
stat_rx_rsfec_err_count0_inc_*	3	Output	Increment for detected errors.
stat_tx_rsfec_lane_alignment_status_*	1	Output	A value of 1 indicates that the TX RS-FEC block has achieved alignment on the incoming PCS data.
PLL and SYS clock select lines when RUNTIME SWITCH feature selected.			
gt_drp_done_*	1	Input	Indicates the completion of the GT DRP operation. Used to reset the GT module. ***Note:*** This port is available when Runtime Switchable Mode is selected in the Configuration tab.
txpllclksel_in_*	1	Input	TX PLL clock select lines ***Note:*** This port is available when **Runtime Switchable Mode** is selected in the Configuration tab.
rxpllclksel_in_*	1	Input	RX PLL clock select lines ***Note:*** This port is available when **Runtime Switchable Mode** is selected in the Configuration tab.

Table 5-2: **Core xci Top Level Port List *(Cont'd)***

Name	Size	Direction	Description
txsysclksel_in_*	1	Input	Select the TX PLL clock source to drive txoutclk ***Note:*** This port is available when **Runtime Switchable Mode** is selected in the Configuration tab.
rxsysclksel_in_*	1	Input	Select the RX PLL clock source to drive rxoutclk ***Note:*** This port is available when **Runtime Switchable Mode** is selected in the Configuration tab.
rxafecfoken_*	1	Input	Only for UltraScale+™ devices
rxdfecfokfcnum_*	1	Input	Only for UltraScale+ devices
speed_*	1	Input	This signal indicates the speed with which the core is working: 1'b1 = 10G and 1'b0 = 25G ***Note:*** This port is available when Runtime Switchable Mode is selected in the Configuration tab.
anlt_done_*	1	Output	Indicates the completion of Auto-Negotiation and Link Training ***Note:*** This port is available when Runtime Switchable Mode is selected in the Configuration tab.
axi_ctl_core_mode_switch_*	1	Output	This signal can be used to switch the line rate from 10G to 25G and vice-verse when selecting **Include AXI4-Lite** in the Configuration tab and write 1 to the 0x0138 self-clearing register to start the GT DRP operations.
user_reg0_*	32	Output	User-defined signal from the AXI4-Reg map user_reg0 register. ***Note:*** This port is available when **Include AXI4-Lite interface** is selected in the Configuration tab.

Duplex Mode of Operation

In this mode of operation, both the transmitter and receiver of the core are active and loopback functionality is provided at the GT output interface, that is, output is fed back as input. Packet generation and monitor modules are active in this mode. The generator module is responsible for generating the desired number of packets and transmit to the core using the available data interface. The monitor module checks the packets from the receiver.

Figure 5-5 shows the duplex mode of operation.

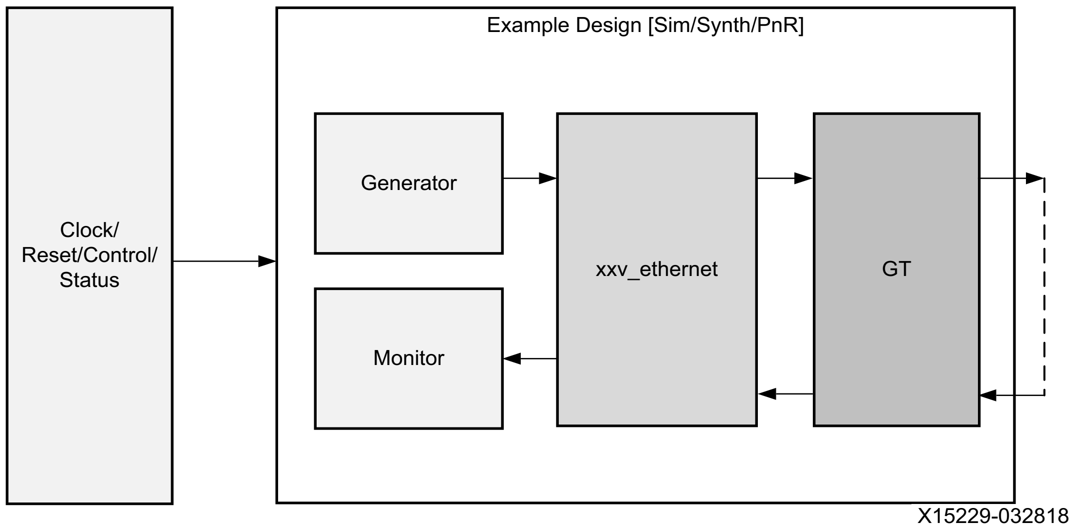


Figure 5-5: **Duplex Mode of Operation**

Runtime Switchable258

This configuration gives the flexibility to switch the line rate between 10G to 25G and vice-versa any time. To activate this feature, select the check box **Runtime Switchable mode** option in the Configuration tab.

When this option is selected the `*_trans_debug` module will be present inside the `*_pkt_gen_mon.v` module of the example design. This `*_trans_debug` module is responsible for performing all the GT DRP write operations to switch the transceiver mode, that is, 10G to 25G or 25G to 10G. When you set the `mode_change_*` input signal High for two clock cycles and then make it Low, it starts the DRP write operation to the GT channel for the specific core and resets the specific core.

The DRP writes are done only for the channel. The QPLL0 of the common is fixed for the line rate 25G and the QPLL1 is fixed for the line rate 10G.

The state transition occurred during this process is shown in Figure 5-6:

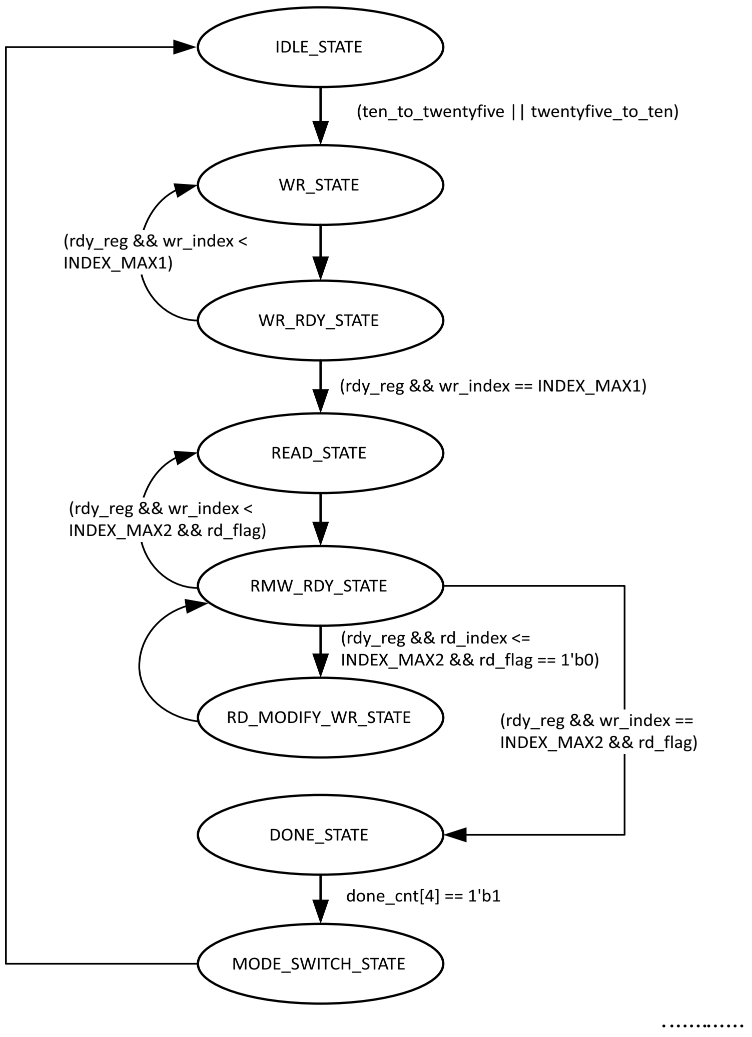

IDLE_STATE
(ten_to_twentyfive || twentyfive_to_ten)
WR_STATE
(rdy_reg && wr_index < INDEX_MAX1)
WR_RDY_STATE
(rdy_reg && wr_index == INDEX_MAX1)
READ_STATE
(rdy_reg && wr_index < INDEX_MAX2 && rd_flag)
RMW_RDY_STATE
(rdy_reg && rd_index <= INDEX_MAX2 && rd_flag == 1'b0)
RD_MODIFY_WR_STATE
(rdy_reg && wr_index == INDEX_MAX2 && rd_flag)
DONE_STATE
done_cnt[4] == 1'b1
MODE_SWITCH_STATE
....

Figure 5-6: **State Transition Diagram for Runtime Switchable DRP Operation without AN/LT**

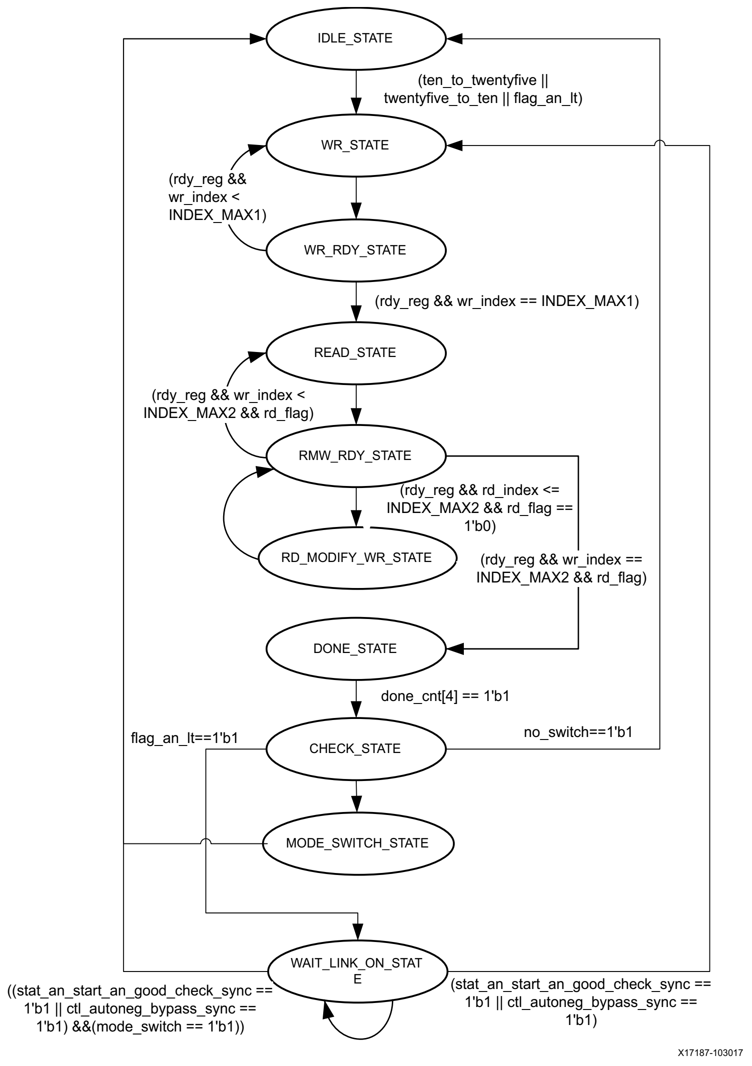


Figure 5-7: **State Transition Diagram for Runtime Switchable DRP Operation with AN/LT**

Shared Logic Implementation

Shared logic includes all the shareable modules that can be present as part of the core or in the Example Design.

By default GT common, reset logic and clocking modules are present inside the IP core. In case of the following conditions, these modules will be placed outside the core so that they can be shared with other designs.

- When you select the **Include GT subcore in example design** option in the GT Selection and Configuration tab.
- When you select the **Include Shared Logic in Example Design** option in the Shared Logic tab.

When the shared logic in the example design is selected, a new `xxv_ethernet_*_core_support.v` module will be instantiated between the `xxv_ethernet_*_exdes.v` and DUT (that is, `xxv_ethernet_*.v`). This module will have all the sub modules that can be shared between multiple designs.

Figure 5-8 shows the implementation when shared logic is instantiated in the example design for single core.

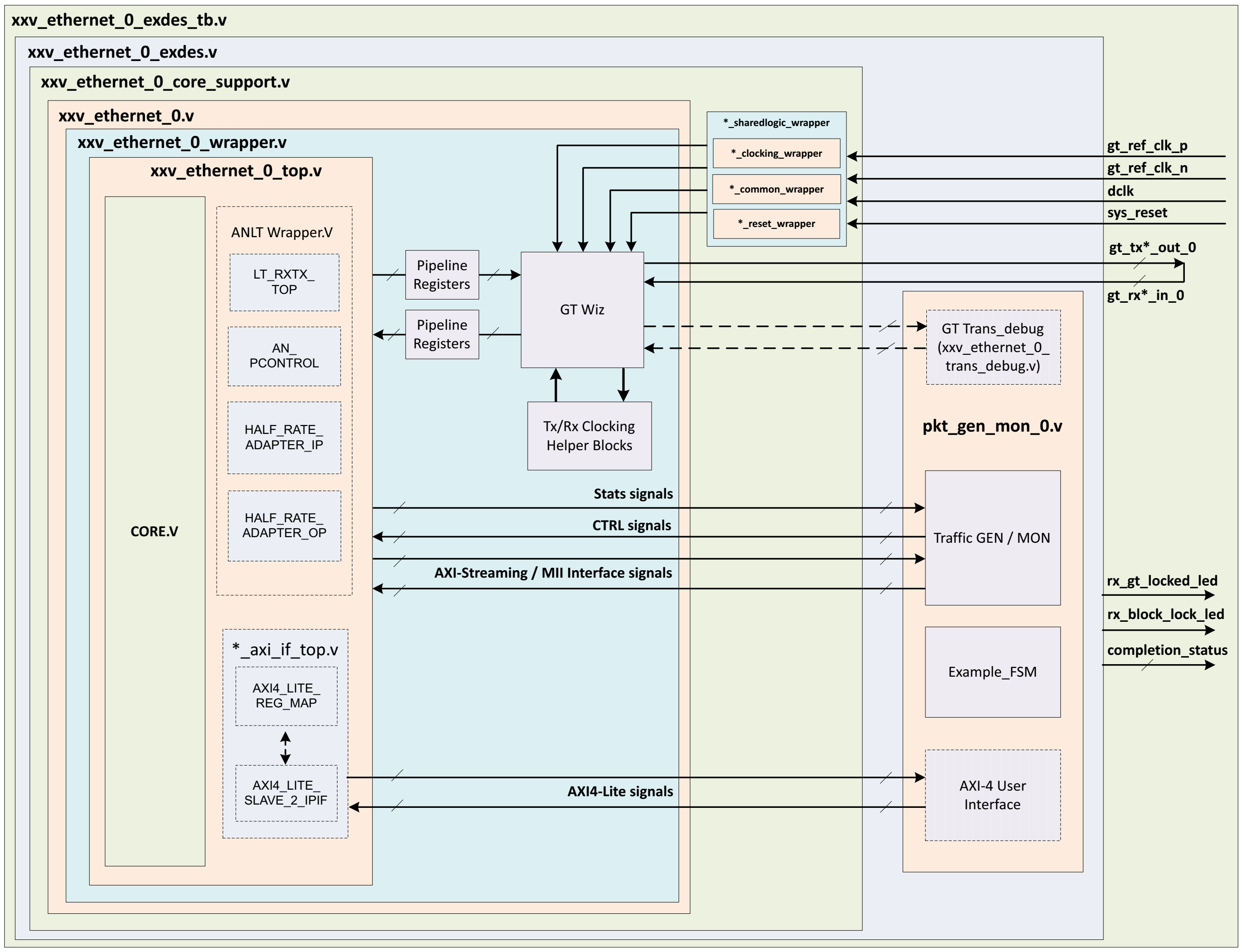


Figure 5-8: **Single Core Example Design Hierarchy With Shared Logic Implementation**

Figure 5-9 shows the implementation when shared logic is instantiated in the example design for multiple cores.

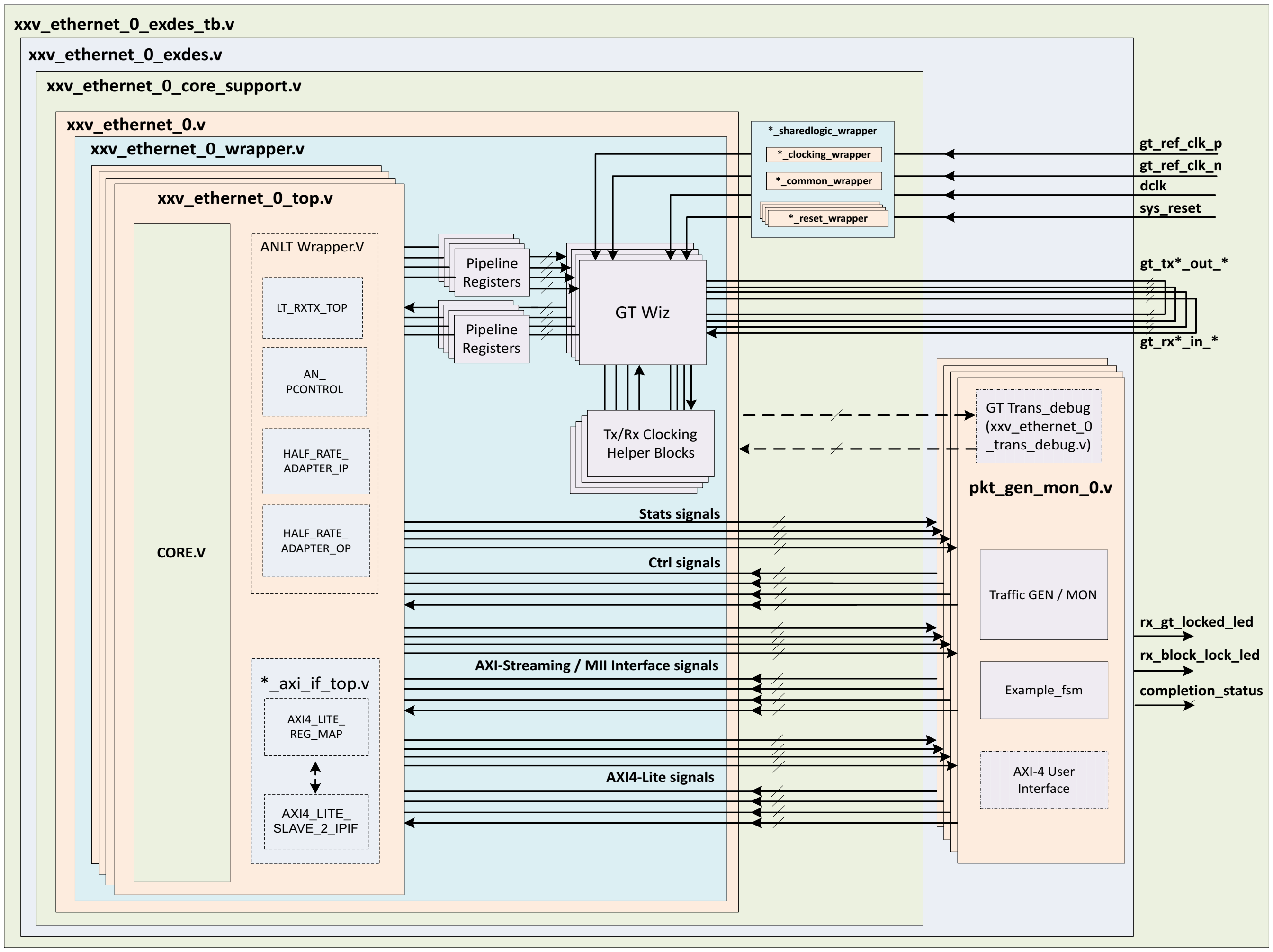


Figure 5-9: **Multiple Core Example Design Hierarchy With Shared Logic Implementation**

The following modules are the part of shared logic wrapper.

- *_clocking_wrapper

  This module contains all the clk resources that can be shared with other designs.

- *_common_wrapper

  This module contains the GT common that can be shared with other designs.

- *_reset_wrapper

  This module contains all the reset logics for the specific selected Vivado IDE configuration.

AXI4-Lite Interface Implementation

In order to instantiate the AXI4-Lite interface to access the control and status registers of the xxv_ethernet core, enable the **Include AXI4-Lite** check box in the Configuration Tab of the Vivado IDE. This option enables the `xxv_ethernet_0_axi_if_top` module (which contains `xxv_ethernet_0_pif_registers` with the `xxv_ethernet_0_slave_2_ipif` module). You can access the AXI4-Lite interface logic registers (control, status and statistics) from the `xxv_ethernet_0_pkt_gen_mon` module.

This mode enables the following features:

- You can configure all the control (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.
- You can access all the status and statistics registers from the core through the AXI4-Lite interface. This operation is performed by reading the address locations for the status and statistics registers through register map.

.h Header File

AXI4 registers information such as register address, register name with bit position, mask value, access type and their default values are provided in header (.h) file format when the IP is generated with **Include AXI4-Lite** enabled in the Vivado Design Suite and the header file can be found under the folder `header_files` of the project path.

AXI4 Interface User Logic

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 below:

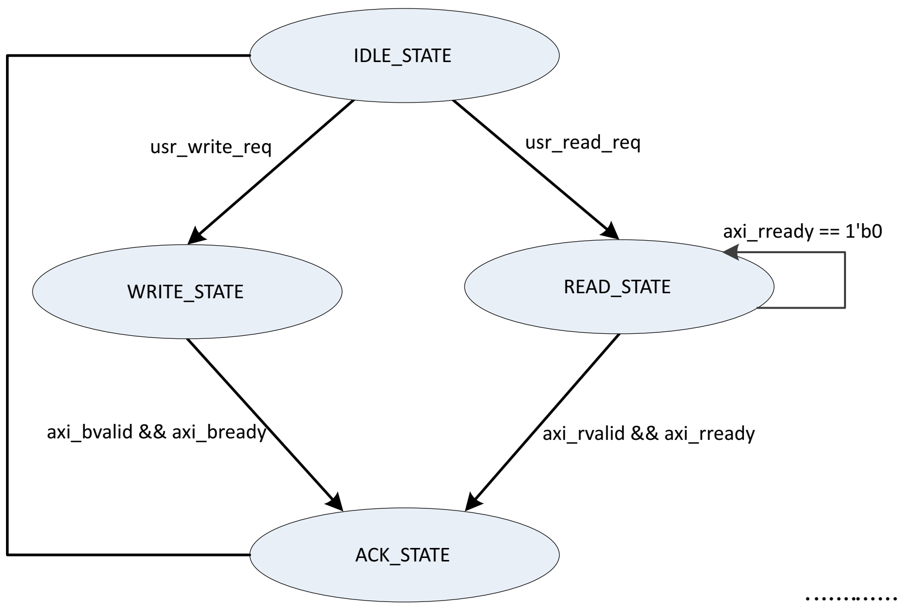


Figure 5-10: **User State Machine for AXI4-Lite Interface**

A functional description of each state is described as below:

- **IDLE_STATE**: By default the FSM will be in IDLE_STATE. When the `user_read_req` signal becomes High, then it moves to the READ_STATE else if the `user_write_req` signal is High, it moves to 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 any write operation 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 any read operation happens 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_ARESETN	1	Input	AXI active-Low synchronous reset

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

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

Valid Write Transactions

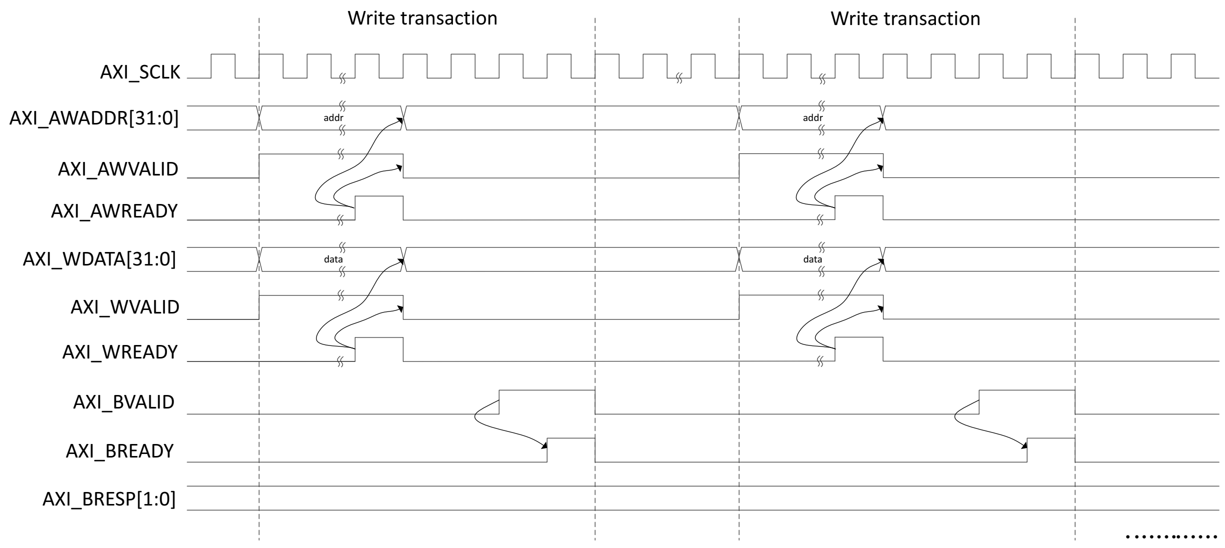


Figure 5-11: **AXI4-Lite User-Side Write Transaction**

Invalid Write Transactions

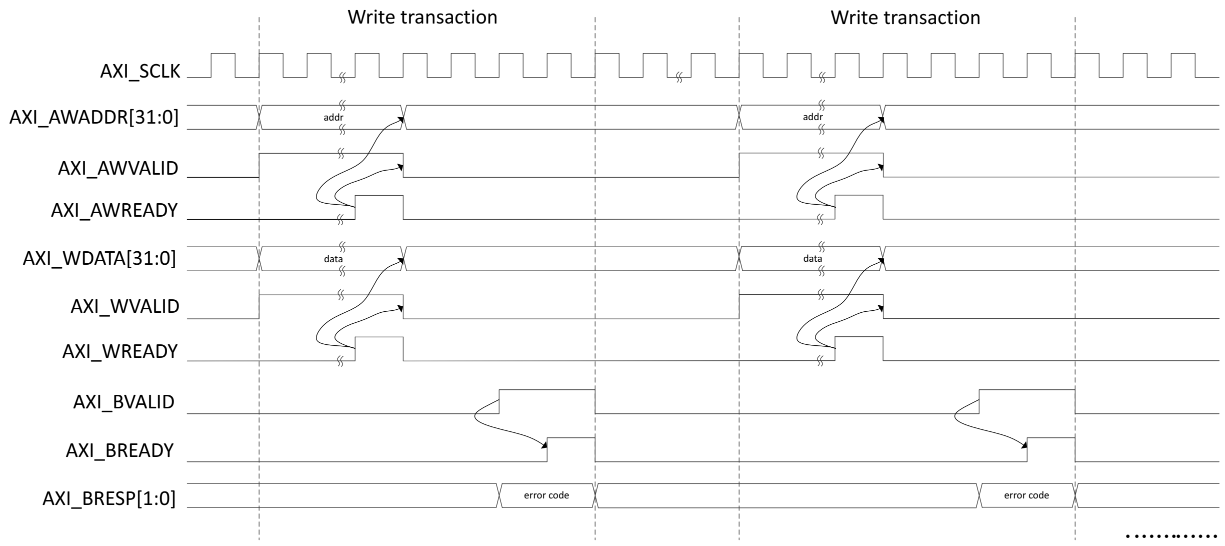


Figure 5-12: **AXI4-Lite User Side Write Transaction with Invalid Write Address**

Valid Read Transactions

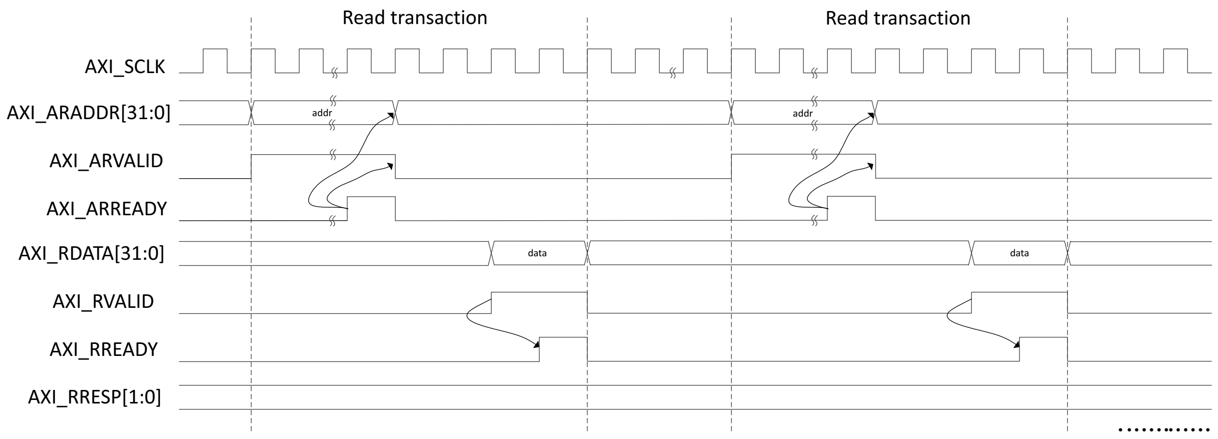


Figure 5-13: **AXI4-Lite User Side Read Transaction**

Invalid Read Transactions

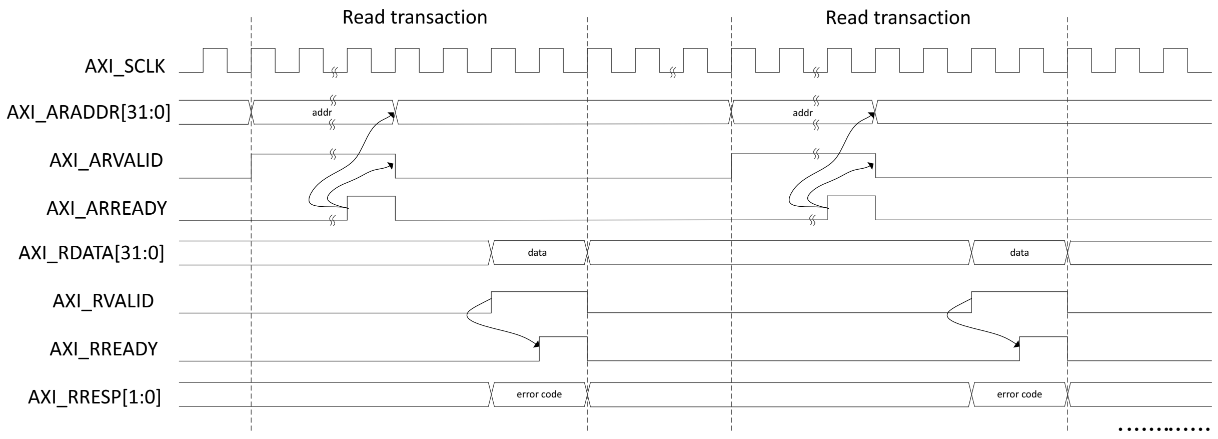


Figure 5-14: **AXI4-Lite User Side Read Transaction with Invalid Read Address**

IEEE Clause 108 (RS-FEC) Integration

If you want to include IEEE clause 108 RS-FEC soft IP (for error correction) in between 25G Ethernet IP and the GT, you must select the **Include Clause 108 (RS-FEC)** check box in the Configuration tab. This option is available for 25G line rate only.

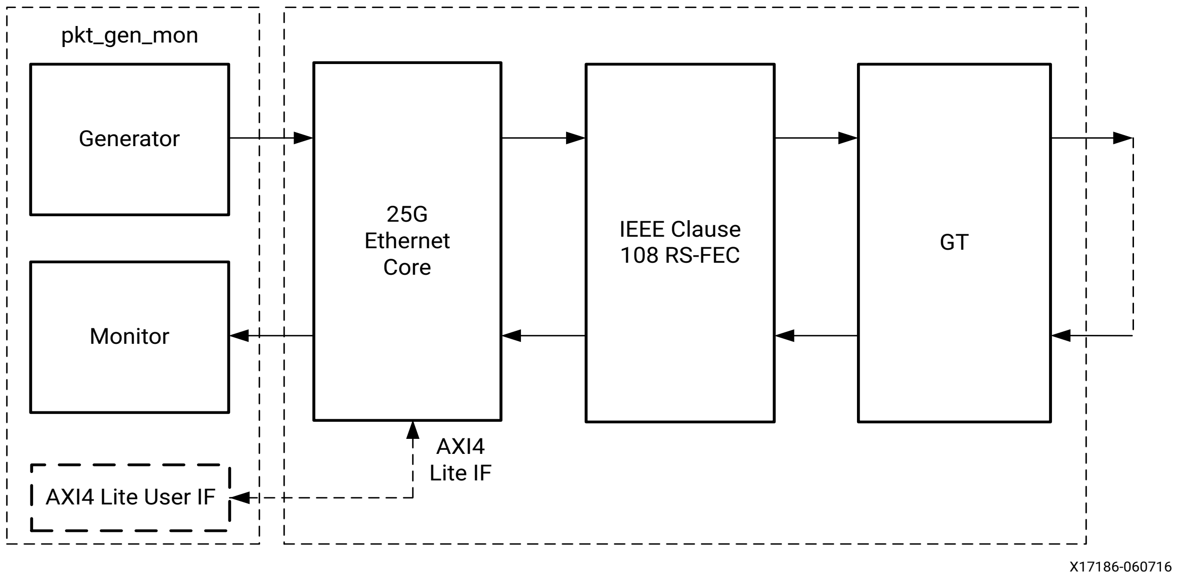


Figure 5-15: **RS-FEC Integration in between 25G and GT**

This feature enables the IEEE Clause108 RS-FEC soft IP component instantiated between the 25G core and the GT. The TX SerDes lines from the 25G core will be input to the RS-FEC soft IP for forward error correction encoding. The output from the RS-FEC module is then fed to GT. Similarly, the RX SerDes lines from the GT will be fed to the RS-FEC module for error correction decoding and then to the 25G core.

Refer to the *25G IEEE 802.3by Reed-Solomon Forward Error Correction LogiCORE IP Product Guide* (PG217) [Ref 12] for IEEE clause 108 Reed-Solomon Forward Error Correction for the LogiCORE™ IP core and its functionality.

Batch Mode Test Bench

Each batch mode release of the 10G/25G Ethernet Subsystem includes a demonstration test bench that performs a loopback test on the complete subsystem. For your convenience, scripts are provided to launch the test bench from several industry-standard simulators. The test program exercises the datapath to check that the transmitted frames are received correctly. Register Transfer Level (RTL) simulation models for the subsystem are included. You must provide the correct path for the transceiver simulation model according to the latest simulation environment settings in your version of the Vivado® Design Suite.

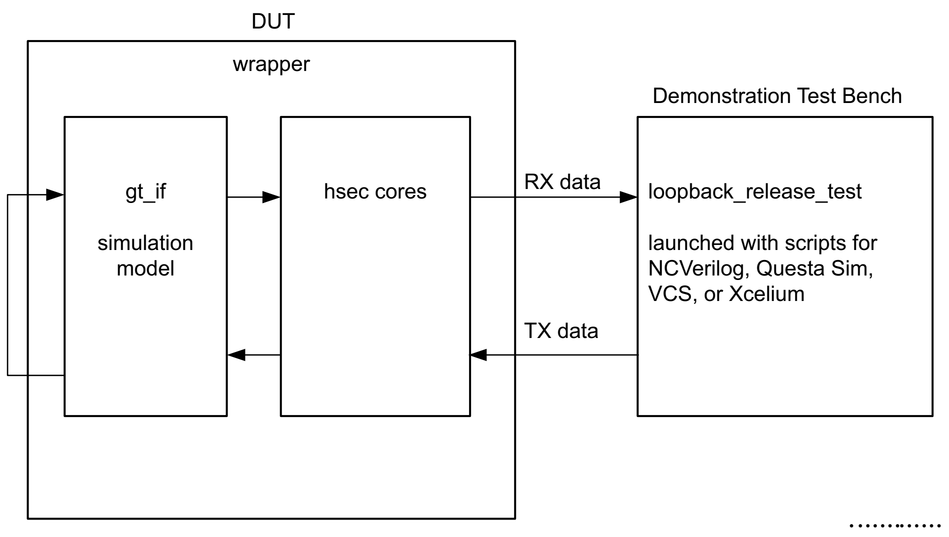


Figure 6-1: **Test Bench**

Upgrading

This appendix contains information about upgrading to a more recent version of the IP core in the Vivado® Design Suite.

A script is provided for upgrading the design to a more recent version of the Vivado Design Suite. The script is found in the `/compile/xilinx/upgrade_IP` directory. Run this script for the following upgrades:

- To upgrade the transceiver wrapper to the latest version.
- To use the latest transceiver simulation model, which if not upgraded can result in simulation errors.

TIP: *Before running the script, save a copy of the original design in the event that you need to revert to the previous version.*

Changes from v2.3 to v2.4

Ports Added

- tx_parityin[7:0]. Added to Table 2-4, Table 2-30, and Table 5-2.
- rx_parityout[7:0]. Added to Table 2-7, Table 2-32, and Table 5-2.
- ctl_tx_parity_err_response. Added to Table 2-10, Table 2-34, and Table 5-2.
- stat_tx_bad_parity. Added to Table 2-14, Table 2-38, and Table 5-2.

Registers Added

Configuration and Status Registers

- CONFIGURATION_TSN_REG to Table 2-43 and the Configuration Registers section
- STAT_TSN_REG to Table 2-44 and to the Status Registers section

Statistics Counters

- STAT_TX_MM_STATUS_LSB: 0x0980
- STAT_TX_MM_STATUS_MSB: 0x0984
- STAT_TX_MM_STATUS_LSB: 0x0988
- STAT_TX_MM_FRAGMENT_MSB: 0x098C
- STAT_TX_MM_HOLD_LSB: 0x0990
- STAT_TX_MM_HOLD_MSB: 0x0994
- STAT_RX_MM_ASSEMBLY_ERROR_LSB: 0x0998
- STAT_RX_MM_ASSEMBLY_ERROR_MSB: 0x099C
- STAT_RX_MM_FRAME_SMD_ERROR_LSB: 0x09A0
- STAT_RX_MM_FRAME_SMD_ERROR_MSB: 0x09A4
- STAT_RX_MM_FRAME_ASSEMBLY_OK_LSB: 0x09A8
- STAT_RX_MM_FRAME_ASSEMBLY_OK_MSB: 0x09AC
- STAT_RX_MM_FRAGMENT_LSB: 0x09B0
- STAT_RX_MM_FRAGMENT_MSB: 0x09B4

Ports Added

Control and Status Ports

- ctl_en_preempt
- ctl_hold_request
- ctl_disable_verify
- ctl_restart_verify
- ctl_addfrag_size
- ctl_verify_time
- ctl_verify_limi
- STATUS
- stat_tx_mm_verify
- stat_tx_mm_status
- stat_tx_mm_fragment
- stat_tx_mm_hold
- stat_rx_mm_assembly_error

- stat_rx_mm_frame_smd_error
- stat_rx_mm_frame_assembly_ok
- stat_rx_mm_fragment

Changes from v2.3 (10/04/2017) to v2.3 (12/20/2017)

Ports Added

The following ports were added to Table 3-1.

- `ctl_tx_latency_*`
- `ctl_tx_lat_adj_enb_*`
- `ctl_rx_latency_0`
- `ctl_rx_lat_adj_enb_0`
- `ctl_rx_timestamp_adj_enb_0`
- `ctl_rx_timestamp_adj_enb_0`

Port Changes

- Updated descriptions of `rx_clk_out`,`rx_core_clk`, and `tx_clk_out` in Table 2-3.
- Updated description of `send_continuous_pkts_*` in Table 5-1.
- Changed 3.42.3 to 3.42.7 for `ctl_tx_test_pattern_*` in Table 5-2.

Ports Removed

The following ports were removed from Table 3-1.

- `rx_ptp_pcslane_out`
- `rx_lane_aligner_fill_0`
- `rx_lane_aligner_fill_1`
- `rx_lane_aligner_fill_2`
- `rx_lane_aligner_fill_3`

The `tx_ptp_rxtstamp_in_*` port was removed from Table 5-2.

Changes from v2.2 to v2.3

Ports Added

- `rx_core_clk` port to Table 2-3
- Table 3-1:
  - `tx_ptp_upd_chksum_in`
  - `tx_ptp_pcslane_out`
  - `tx_ptp_chksum_offset_in`
  - `rx_ptp_pcslane_out`
  - `rx_lane_aligner_fill_0`
  - `rx_lane_aligner_fill_1`
  - `rx_lane_aligner_fill_2`
  - `rx_lane_aligner_fill_3`
- `send_continuous_pkts_*` port to Table 5-1
- `user_reg0_*` in Table 5-2

Ports Updated

Table 5-2:

- `stat_rx_status`
- `user_reg1_*`

Ports Removed

- `tx_ptp_rxtstamp_in` port from Table 3-1

Port Name Changes

- `mode_change_0` to `mode_change_*`
- `core_speed_0` to `core_speed_*`

Registers

- Deleted USER_REG1.
- Changed CORE_SPEED_REG name to SWITCH_CORE_SPEED_REG.
- Added a note about the readable STAT_*_MSB/LSB registers to the Statistics Counters section in Chapter 2.
- Added STAT_CORE_SPEED_REG to Table 2-44. Also added new Table 2-130 for new register.
- Updated USER_REG_0: 0184 to USER_REG_0: 0134.
- Added the following registers: CONFIGURATION_1588_REG, TX_CONFIGURATION_1588_REG, and RX_CONFIGURATION_1588_REG

Changes from v2.1 to v2.2

Ports

Added IEEE 802.3 Clause 74 FEC Interface and IEEE 802.3 Clause 108 RS-FEC Interface sections that include the port description in the new tables.

Ports Added

Added *gtpowergood_out_** to Table 5-2.

Port Descriptions Updated

- dclk in Table 2-23.
- ctl_gt_reset_all signal in Table 2-42.
- tick_reg signal in Table 2-49.
- rx_clk in Chapter 2.
- gt_ref_clk_p, gt_ref_clk_n, and dclk signals in Table 5-1.
- an_clk_*, stat_fec_lock_error_* and ctl_rsfec_enable_* in Table 5-2.

Changes from v2.0 to v2.1

New Variants for 10Gb/s Operation

- 64-bit MAC-only configuration
- 32-bit Low Latency MAC and PCS configuration
- Redesigned auto-negotiation and Link Training features with reduced utilization; all changes are transparent to the user

Feature Updates for 25Gb/s Operation

Redesigned auto-negotiation and Link Training features with reduced utilization; all changes are transparent to the user.

Ports

Ports Added

- stat_rx_status
- axi_ctl_core_mode_switch_*
- rx_mii_clk
- tx_mii_clk
- clk

Ports Deleted

tx_ptp_pcslane_out

Port Changes

- Added 32-bits to tx_axis_tdata_* and rx_axis_tdata_*.
- Updated bus sizes for many signals in Table 2-11 through Table 2-22 and Table 2-24.
- Updated bus sizes of tx_axis_tdata_* and rx_axis_tdata_* in Table 5-2.
- Added 32/64-bit to tx_clk_out_* and tx_mii_out_* in Table 5-2.
- Added 32 bits to rx_serdes_data_out_* and tx_serdes_data_in_* in Table 5-2.
- Replaced “Ethernet MAC+PCS/PMA” with “Ethernet MAC+PCS/PMA-32/64-bit” in most of the signals in Table 5-2.
- Inserted "or Ethernet MAC" in most of the signals in Table 5-2.
- Updated descriptions of the following:
    - rx_axis_tdata[63 or 31:0]
    - rx_axis_tkeep[7 or 3:0]
    - ctl_rx_ignore_fcs
    - ctl_rx_max_packet_len[14:0]
    - ctl_rx_min_packet_len[7:0]
    - stat_rx_undersize
    - stat_rx_fragment
    - stat_tx_total_bytes[3:0]
    - stat_tx_frame_error
    - ctl_tx_pause_quanta[8:0][15:0]
    - ctl_tx_pause_refresh_timer[8:0][15:0]
    - ctl_an_nonce_seed[7:0]
    - rx_mii_clk
    - tx_mii_d_*
    - tx_mii_c_*
    - rx_mii_d_*
    - rx_mii_c_*

Registers Added

- USER_REG_1: 0188
- CORE_SPEED_REG:018C

Register Changes

- Added Bit 1 to CORE_SPEED_REG: 0180.
- Added note for Bits 1 and 2 to STAT_TX_STATUS_REG1: 0400.
- Modified notes for most of the registers in Table 2-43, Table 2-44, and Table 2-45.

Changes from v2.0 (10/05/2016) to v2.0 (11/30/2016) version

Ports Added To Table 5-2, Core xci Top Level Port List

- gtwiz_reset_qpll1lock_in
- gtwiz_reset_qpll1reset_out
- gt_refclk_out
- tx_unfout_*
- stat_rx_valid_ctrl_code_*
- ctl_tx_ptp_1step_enable_*
- ctl_tx_ptp_latency_adjust_*
- ctl_ptp_transpclk_mode_*
- tx_ptp_upd_chksum_in_*
- tx_ptp_tstamp_offset_in_*
- tx_ptp_chksum_offset_in_*
- tx_ptp_rxtstamp_in_*
- ctl_an_fec_25g_baser_request_*
- stat_an_lp_ability_25gbase_cr1_*
- stat_an_rxcdrhold_*
- ctl_fec_enable_error_to_pcs_*
- ctl_rx_rsfec_enable_correction_*

- ctl_rx_rsfec_enable_indication_*
- ctl_rsfec_enable_*
- ctl_rsfec_ieee_error_indication_mode_*
- ctl_rsfec_consortium_25g_*
- stat_rx_rsfec_hi_ser_*
- stat_rx_rsfec_lane_alignment_status_*
- stat_rx_rsfec_corrected_cw_inc_*
- stat_rx_rsfec_uncorrected_cw_inc_*
- stat_rx_rsfec_err_count0_inc_*
- stat_tx_rsfec_lane_alignment_status_*
- gt_drp_done_*
- speed_*
- anlt_done_*

Ports Removed from Table 5-2, Core xci Top Level Port List

- stat_rx_framing_err_valid_1_*
- stat_rx_framing_err_1_*
- stat_rx_framing_err_valid_2_*
- stat_rx_framing_err_2_*
- stat_rx_framing_err_valid_3_*
- stat_rx_framing_err_3_*
- stat_rx_vl_demuxed_*
- stat_rx_vl_number_0_*
- stat_rx_vl_number_1_*
- stat_rx_vl_number_2_*
- stat_rx_vl_number_3_*
- stat_rx_synced_*
- stat_rx_synced_err_*
- stat_rx_mf_len_err_*
- stat_rx_mf_repeat_err_*
- stat_rx_mf_err_*

- stat_rx_misaligned_*
- stat_rx_aligned_err_*
- stat_rx_bip_err_0_*
- stat_rx_bip_err_1_*
- stat_rx_bip_err_2_*
- stat_rx_bip_err_3_*
- stat_rx_aligned_*
- stat_rx_status_*
- tx_ptp_pcslane_out_*
- rx_lane_aligner_fill_0_*
- rx_lane_aligner_fill_1_*
- rx_lane_aligner_fill_2_*
- rx_lane_aligner_fill_3_*
- ctl_lt_pseudo_seed1_*
- ctl_lt_k_p1_to_tx1_*
- ctl_lt_k0_to_tx1_*
- ctl_lt_stat_p1_to_tx1_*
- ctl_lt_stat0_to_tx1_*
- ctl_lt_stat_m1_to_tx1_*
- ctl_lt_pseudo_seed2_*
- ctl_lt_k_p1_to_tx2_*
- ctl_lt_k0_to_tx2_*
- ctl_lt_k_m1_to_tx2_*
- ctl_lt_k_m1_to_tx2_*
- ctl_lt_stat0_to_tx2_*
- ctl_lt_stat_m1_to_tx2_*
- ctl_lt_pseudo_seed3_*
- ctl_lt_k_p1_to_tx3_*
- ctl_lt_k0_to_tx3_*
- ctl_lt_k_m1_to_tx3_*
- ctl_lt_stat_p1_to_tx3_*

- Ctl_lt_stat0_to_tx3_*
- ctl_lt_stat_m1_to_tx3_*
- stat_an_start_tx_disable_*
- stat_an_start_an_good_check_*
- stat_lt_k_p1_from_rx1_*
- stat_lt_k0_from_rx1_*
- stat_lt_k_m1_from_rx1_*
- stat_lt_stat_p1_from_rx1_*
- stat_lt_stat0_from_rx1_*
- stat_lt_stat_m1_from_rx1_*
- stat_lt_k_p1_from_rx2_*
- Stat_lt_k0_from_rx2_*
- stat_lt_k_m1_from_rx2_*
- stat_lt_stat_p1_from_rx2_*
- stat_lt_stat0_from_rx2_*
- stat_lt_stat_m1_from_rx2_*
- stat_lt_k_p1_from_rx3_*
- stat_lt_k0_from_rx3_*
- stat_lt_k_m1_from_rx3_*
- stat_lt_stat_p1_from_rx3_*
- stat_lt_stat0_from_rx3_*
- stat_lt_stat_m1_from_rx3_*

Migrating from the Legacy XGEMAC

This section contains information about upgrading from the legacy version of the Xilinx® 10G EMAC IP to the new 10G/25G High Speed Ethernet IP core.

Note: For all new designs, Xilinx recommends that you use the most recent version of the 10G/25G High Speed Ethernet IP Core.

The Xilinx® 10G Ethernet MAC and the 10G/25G High Speed Ethernet core are both designed to the specifications of the Ethernet IEEE 802.3 standard. There are significant differences in how some features are designed and/or handled. There are also differences in signal and parameter names and the corresponding AXI registers. Note that this section only outlines the features in the legacy 10G EMAC IP and compares them with the new 10G/25G High Speed Ethernet IP core. For a list of new features or features exclusive to the new IP, refer to Chapter 2, Product Specification.

The following features are different in the new 10/25G High Speed Ethernet IP.

- Padding
  The Pad field is not added by the 10/25G High Speed Ethernet IP. You must present a packet that meets the minimum length to the IP core. When the IP core is configured to calculate and add the FCS to the packet (`ctl_tx_fcs_ins_enable` = 1), the minimum packet length is 60 bytes. If the FCS is calculated and added outside the IP core (`ctl_tx_fcs_ins_enable` = 0), the minimum packet length is 64 bytes.
- IFG extension
  The inter-frame gap (IFG/IPG) can be extended up to 12B using the parameter ctl_tx_ipg_value[2:0].
- Deficit Idle Count (DIC)
  This feature is not supported.
- Management Data Input/Output (MDIO) master
  The 10G/25G High Speed Ethernet IP does not provide an MDIO master. The contents of the appropriate MDIO registers are available in the status signals.
- Fault Handling
  The user logic must be designed differently for TX faults. Legacy 10G EMAC TX transmits RF or idles and drops packets by default if LF/RF is received. An option to disable fault transmission is provided. 10/25G High Speed Ethernet IP requires you to control if LF/RF are transmitted. You must provide the fault status signals as well.
- VLAN
  The new 10G/25G IP provides no VLAN specific features. However you can set the ctl_rx_max_packet_len attribute appropriately to allow the standard VLAN frame (1522 B) and also design the user logic to handle any number of stacked VLAN tags.

- Enabling Link Training without Auto-negotiation
  On the legacy 10G IP, it appears that link training was always enabled (see below) whereas the 10G/25G High Speed Ethernet IP performs link training after auto-negotiation and thus both features have to be enabled.
- Link Training Translation
  The legacy 10-BaseKR subsystem included logic that allowed it to be trained by a far-end device without user interaction. This feature is not available in the 10/25G High Speed Ethernet IP and the user logic must be designed to support this feature.
- Standalone MAC with 64-bit internal XGMII interface for connecting to XAUI/RXAUI
  The standalone MAC with the 64-bit internal XGMII interface is now available in 10G/ 25G High Speed Ethernet Subsystem v2.1 and later.
- External XGMII DDR interface to external PHY
  The external XGMII DDR interface to the external PHY option is now available as part of the 64-bit standalone MAC in 10G/25G High Speed Ethernet IP v2.1 and later.
- Pause Interface
  The 10G/25G High Speed Ethernet IP does not support the legacy PFC. However up to eight Priority pause frames and global frames can be generated and terminated.

Port/Signal Comparison

There are significant differences in the names and functions of the signals and parameters. Because it is difficult to draw a one-to-one comparison, the following tables are based on features and their presentation in the two IP cores.

Parameter/Attribute Comparison

Transmitter configuration

The legacy XGEMAC used the mac_tx_configuration_vector[368:0] for all TX configuration whereas the 10G/25G High Speed Ethernet IP deploys various signals for the same purpose. Table A-1 draws a comparison.

Table A-1: **Transmitter Configuration Comparison**

Feature	Legacy XGEMAC	10/25G High Speed Ethernet IP
Legacy Pause Refresh Value	mac_tx_configuration_vector[367:352]	
TX Priority 7 Pause Quanta refresh value	mac_tx_configuration_vector[351:336]	ctl_tx_pause_refresh_timer7[15:0]
TX Priority 7 Pause Quanta	mac_tx_configuration_vector[335:320]	ctl_tx_pause_quanta7[15:0]
TX Priority 6 Pause Quanta refresh value	mac_tx_configuration_vector[319:304]	ctl_tx_pause_refresh_timer6[15:0]
TX Priority 6 Pause Quanta	mac_tx_configuration_vector[303:288]	ctl_tx_pause_quanta6[15:0]

Table A-1: **Transmitter Configuration Comparison *(Cont'd)***

Feature	Legacy XGEMAC	10/25G High Speed Ethernet IP
TX Priority 5 Pause Quanta refresh value	mac_tx_configuration_vector[287:272]	ctl_tx_pause_refresh_timer5[15:0]
TX Priority 5 Pause Quanta	mac_tx_configuration_vector[271:256]	ctl_tx_pause_quanta5[15:0]
TX Priority 4 Pause Quanta refresh value	mac_tx_configuration_vector[255:240]	ctl_tx_pause_refresh_timer4[15:0]
TX Priority 4 Pause Quanta	mac_tx_configuration_vector[239:224]	ctl_tx_pause_quanta4[15:0]
TX Priority 3 Pause Quanta refresh value	mac_tx_configuration_vector[223:208]	ctl_tx_pause_refresh_timer3[15:0]
TX Priority 3 Pause Quanta	mac_tx_configuration_vector[207:192]	ctl_tx_pause_quanta3[15:0]
TX Priority 2 Pause Quanta refresh value	mac_tx_configuration_vector[191:176]	ctl_tx_pause_refresh_timer2[15:0]
TX Priority 2 Pause Quanta	mac_tx_configuration_vector[175:160]	ctl_tx_pause_quanta2[15:0]
TX Priority 1 Pause Quanta refresh value	mac_tx_configuration_vector[159:144]	ctl_tx_pause_refresh_timer1[15:0]
TX Priority 1 Pause Quanta	mac_tx_configuration_vector[143:128]	ctl_tx_pause_quanta1[15:0]
TX Priority 0 Pause Quanta refresh value	mac_tx_configuration_vector[127:112]	ctl_tx_pause_refresh_timer0[15:0]
TX Priority 0 Pause Quanta	mac_tx_configuration_vector[111:96]	ctl_tx_pause_quanta0[15:0]
TX Priority 7 Flow Control Enable	mac_tx_configuration_vector[95]	ctl_tx_pause_enable[7]
TX Priority 6 Flow Control Enable	mac_tx_configuration_vector[94]	ctl_tx_pause_enable[6]
TX Priority 5 Flow Control Enable	mac_tx_configuration_vector[93]	ctl_tx_pause_enable[5]
TX Priority 4 Flow Control Enable	mac_tx_configuration_vector[92]	ctl_tx_pause_enable[4]
TX Priority 3 Flow Control Enable	mac_tx_configuration_vector[91]	ctl_tx_pause_enable[3]
TX Priority 2 Flow Control Enable	mac_tx_configuration_vector[90]	ctl_tx_pause_enable[2]
TX Priority 1 Flow Control Enable	mac_tx_configuration_vector[89]	ctl_tx_pause_enable[1]
TX Priority 0 Flow Control Enable	mac_tx_configuration_vector[88]	ctl_tx_pause_enable[0]
Auto XON enable	mac_tx_configuration_vector[81]	
Priority flow control enable	mac_tx_configuration_vector[80]	
TX Pause Frame Source Address	mac_tx_configuration_vector[79:32]	ctl_tx_pause_da[47:0]

Table A-1: **Transmitter Configuration Comparison *(Cont'd)***

Feature	Legacy XGEMAC	10/25G High Speed Ethernet IP
TX MTU Size	mac_tx_configuration_vector[30:16]	Not required because the TX can handle any size frame presented to the IP.
TX MTU enable	mac_tx_configuration_vector[14]	Not applicable because there is no need to set the TX. MTU does not have to be set.
Deficit Idle Count Enable	mac_tx_configuration_vector[10]	Not supported
TX LAN/WAN mode	mac_tx_configuration_vector[9]	This feature is not available as-is. The 10G/25G High Speed Ethernet IP always maintains the average IPG per the IEEE 802.3 spec. However you can design the user logic to insert idles using ctl_send_idle and increase the IPG value using ctl_tx_ipg_value[3:0] to meet the OC-192 SONET requirements.
TX IFG Adjust enable	mac_tx_configuration_vector[8]	ctl_tx_ipg_value sets the appropriate value of the custom IPG/IFG
TX Preserve Preamble Enable	mac_tx_configuration_vector[7]	ctl_tx_custom_preamble_enable enables the use of a custom preamble. Tx_preamblein presents the custom preamble and rx_preambleout has the preamble field from the received frame.
TX Flow control Enable	mac_tx_configuration_vector[5]	ctl_tx_pause_enable[8:0]
TX Jumbo Frame Enable	mac_tx_configuration_vector[4]	Not required
TX In-band FCS enable	mac_tx_configuration_vector[3]	ctl_tx_fcs_ins_enable
TX VLAN enable	mac_tx_configuration_vector[2]	Not required
TX Enable	mac_tx_configuration_vector[1]	ctl_tx_enable
TX Reset	mac_tx_configuration_vector[0]	tx_reset

Receiver Configuration

The legacy XGEMAC used the rx_configuration_vector[95:0] for all RX configuration whereas the 10G/25G High Speed Ethernet IP core deploys various signals for the same purpose. Table A-2 draws a comparison.

Table A-2: **Receiver Configuration Comparison**

Feature	Legacy XGEMAC	10/25G High Speed Ethernet IP
RX Priority 7-0 Flow control enable	rx_configuration_vector[95:88]	ctl_rx_pause_enable[8:0]
Priority Flow Control Enable	rx_configuration_vector[80]	
RX Pause Frame SA	rx_configuration_vector[79:32]	ctl_rx_pause_sa[47:0]
RX MTU size	rx_configuration_vector[30:16]	Set using ctl_rx_max_packet_len and ctl_rx_min_packet_len signals.
RX MTU Enable	rx_configuration_vector[14]	Not required.
Reconciliation Sublayer Fault Inhibit	rx_configuration_vector[10]	Design user logic to set ctl_tx_send_idle when RFI is received.
Control Frame Length check Disable	rx_configuration_vector[9]	Not available
RX Length/Type Error disable	rx_configuration_vector[8]	The length/type error cannot be disabled on the 10/25G High Speed Ethernet IP core.
RX preserve preamble enable	rx_configuration_vector[7]	ctl_rx_custom_preamble_enable
RX Flow control enable	rx_configuration_vector[5]	ctl_rx_pause_enable[8:0]
RX Jumbo Frame Enable	rx_configuration_vector[4]	Set using the ctl_rx_max_packet_len signal.
RX in-band FCS enable	rx_configuration_vector[3]	ctl_rx_delete_fcs
RX VLAN enable	rx_configuration_vector[2]	Up to the user logic to implement this functionality.
RX enable	rx_configuration_vector[1]	ctl_rx_enable
RX reset	rx_configuration_vector[0]	rx_reset

Status Vector

Table A-3: **Status Vector Comparison**

Feature	Legacy XGEMAC	10/25G High Speed Ethernet IP
Remote Fault RX	status_vector[1]	This feature is not available.
Local Fault RX	status_vector[0]	This feature is not available.

TX Statistics

The legacy XGEMAC used the `tx_statistics_vector[25:0]` for all TX statistics whereas the 10G/25G High Speed Ethernet IP core deploys various signals for the same purpose. Table A-4 draws a comparison.

Table A-4: **TX Statistics Comparison**

Feature	Legacy XGEMAC	10/25G High Speed Ethernet MAC
PFC Frame Transmitted	tx_statistics_vector[26]	stat_tx_user_pause
Pause Frame Transmitted	tx_statistics_vector[25]	stat_tx_pause
Bytes Valid	tx_statistics_vector[24:21]	stat_tx_total_good_bytes
VLAN Frame	tx_statistics_vector[20]	stat_tx_vlan
Frame length count	tx_statistics_vector[19:5]	stat_tx_packet_* signals that can also be used for the packet histogram.
Control Frame	tx_statistics_vector[4]	
Underrun Frame	tx_statistics_vector[3]	
Multicast Frame	tx_statistics_vector[2]	stat_tx_multicast
Broadcast Frame	tx_statistics_vector[1]	stat_tx_broadcast
Successful Frame	tx_statistics_vector[0]	stat_tx_total_good_packets
TX statistics Valid	tx_statistics_valid	Not required

RX Statistics

The legacy XGEMAC used the `rx_statistics_vector[30:0]` for all RX statistics whereas the 10G/25G High Speed Ethernet IP core deploys various signals for the same purpose. Table A-5 draws a comparison.

Table A-5: **RX Statistics Comparison**

Feature	Legacy XGEMAC	10/25G High Speed Ethernet MAC
PFC Frame	rx_statistics_vector[30]	stat_rx_user_pause
Length/Type out of range	rx_statistics_vector[29]	stat_rx_inrange_err
Bad Opcode	rx_statistics_vector[28]	
Flow Control Frame	rx_statistics_vector[27]	stat_rx_pause
Bytes Valid	rx_statistics_vector[26:23]	stat_rx_total_bytes[3:0]
VLAN Frame	rx_statistics_vector[22]	stat_rx_vlan
Out of Bounds	rx_statistics_vector[21]	stat_rx_toolong
Control Frame	rx_statistics_vector[20]	
Frame Length Count	rx_statistics_vector [19:5]	stat_rx_total_good_bytes [13:0]
Multicast Frame	rx_statistics_vector [4]	stat_rx_multicast

Table A-5: **RX Statistics Comparison *(Cont'd)***

Feature	Legacy XGEMAC	10/25G High Speed Ethernet MAC
Broadcast frame	rx_statistics_vector [3]	stat_rx_broadcast
FCS Error	rx_statistics_vector [2]	stat_rx_packet_bad_fcs
Bad frame	rx_statistics_vector [1]	stat_rx_total_good_packets sampled as 0
Good Frame	rx_statistics_vector [0]	stat_rx_total_good_packets sampled as 1
RX statistics Valid	rx_statistics_valid	Not required

Register Space

Both the legacy XGEMAC and the 10/25G High Speed Ethernet IP core provide an AXI User Interface. While much of the configuration parameters are similar, the registers and the memory map of the registers are different between the two IP cores. The following sections discuss the comparison between the configuration, statistics and other registers.

TX Configuration Registers

Table A-6: **TX Configuration Registers Comparison**

Feature	Legacy XGEMAC	10/25G HSEC
TX MTU Size	@0x418: Transmitter MTU Configuration Word [14:0]	Not required because the TX can handle any size frame presented to the IP.
TX MTU enable	@0x418: Transmitter MTU Configuration Word [16]	Not applicable because there is no need to set the TX MTU.
Deficit Idle Count Enable	@0x408: Transmitter Configuration Word[24]	Not supported
TX LAN/WAN mode	@0x408: Transmitter Configuration Word[26]	This feature is not available as-is. The 10G/25G High Speed Ethernet IP Subsystem always maintains the average IPG per the IEEE 802.3 spec. However, you can design the user logic to insert idles using @0x000C: CONFIGURATION_TX_REG1[5] ctl_send_idle and increase the IPG value using @0x000C: CONFIGURATION_TX_REG1[13:10] ctl_tx_ipg_value[3:0] to meet the OC-192 SONET requirements.
TX IFG Adjust enable	@0x408: Transmitter Configuration Word[25]	@0x000C: CONFIGURATION_TX_REG1[13:10] ctl_tx_ipg_value sets the appropriate value of the custom IPG/IFG.
TX Preserve Preamble Enable	@0x408: Transmitter Configuration Word[23]	@0x000C: CONFIGURATION_TX_REG1[18] ctl_tx_custom_preamble_enable enables the use of custom preamble. Tx_preamblein presents the custom preamble and rx_preambleout has the preamble field from the received frame.

Table A-6: **TX Configuration Registers Comparison *(Cont'd)***

Feature	**Legacy XGEMAC**	**10/25G HSEC**
TX Flow control Enable	@0x40C: Flow control Configuration register [30]	ctl_tx_pause_enable[8:0]
TX Jumbo Frame Enable	@0x408: Transmitter Configuration Word[30]	Not required.
TX In-band FCS enable	@0x408: Transmitter Configuration Word[29]	@0x000C: CONFIGURATION_TX_REG1[1] ctl_tx_fcs_ins_enable
TX VLAN enable	@0x408: Transmitter Configuration Word[27]	Not required.
TX Enable	@0x408: Transmitter Configuration Word [28]	@0x000C: CONFIGURATION_TX_REG1[0] ctl_tx_enable
TX Reset	@0x408: Transmitter Configuration Word [31]	@0x0004: RESET_REG[31] tx_reset

RX Configuration

Table A-7: **RX Configuration Comparison**

Feature	Legacy XGEMAC	10/25G High Speed Ethernet IP Core
RX Priority 7-0 Flow control enable	Bit 95:88	@0x0094: CONFIGURATION_RX_FLOW_CONTROL_REG1 [7:0]
Priority Flow Control Enable	Bit 80	Does not support legacy PFC
RX Pause Frame SA	@0x404: Receiver Configuration Word 1[47:32], @0x400: Receiver Configuration Word 0	@0x00C4: CONFIGURATION_RX_FLOW_CONTROL_SA_REG1_MSB[15:0], @0x00C0: CONFIGURATION_RX_FLOW_CONTROL_SA_REG1_LSB
RX MTU size	@0x414: Receiver MTU Configuration Word [14:0]	Set using @0x0018: CONFIGURATION_RX_MTU[30:16] ctl_rx_max_packet_len and @0x0018: CONFIGURATION_RX_MTU[7:0] ctl_rx_min_packet_len signals
RX MTU Enable	@0x414: Receiver MTU Configuration Word [16]	Not required.
Reconciliation Sublayer Fault Inhibit	@0x410: Reconciliation Sublayer Configuration Word [27]	Design user logic to set @0x000C: CONFIGURATION_TX_REG1[5] ctl_tx_send_idle when RFI is received
Control Frame Length check Disable	@0x404: Receiver Configuration Word 1[24]	Not available
RX Length/ Type Error disable	@0x404: Receiver Configuration Word 1[25]	The length/type error cannot be disabled on the 10/25G High Speed Ethernet Subsystem.
RX preserve preamble enable	@0x404: Receiver Configuration Word 1[26]	@0x0014: CONFIGURATION_RX_REG1[11] ctl_rx_custom_preamble_enable
RX Flow control enable	@0x40C: Flow Control Configuration Register [29]	@0x0094: CONFIGURATION_RX_FLOW_CONTROL[8:0] ctl_rx_pause_enable[8:0]
RX Jumbo Frame Enable	@0x404: Receiver Configuration Word 1[30]	Set using ctl_rx_max_packet_len signal
RX in-band FCS enable	@0x404: Receiver Configuration Word 1[29]	@0x0014: CONFIGURATION_RX_REG1[1] ctl_rx_delete_fcs
RX VLAN enable	@0x404: Receiver Configuration Word 1[27]	Up to the user logic to implement this functionality

Table A-7: **RX Configuration Comparison *(Cont'd)***

Feature	Legacy XGEMAC	10/25G High Speed Ethernet IP Core
RX enable	@0x404: Receiver Configuration Word 1[28]	@0x0014: CONFIGURATION_RX_REG1[0] ctl_rx_enable
RX reset	@0x404: Receiver Configuration Word 1[31]	@0x0004: RESET_REG[30] rx_reset

Status Vector

Table A-8: **Status Vector Comparison**

Feature	Legacy XGEMAC	10/25G High Speed Ethernet IP
Remote Fault RX	@0x410: Reconciliation Sublayer Configuration Word[29]	This feature is not available.
Local Fault RX	@0x410: Reconciliation Sublayer Configuration Word[28]	This feature is not available.

Statistics Counters

Table A-9 lists the different statistics counters and their addresses.

Table A-9: **Statistics Counters**

Address (Hex)	Register	Register	Address (Hex)
0x200	Received Bytes (LSW)	STAT_RX_TOTAL_BYTES_LSB	0x0808
0x204	Received Bytes (MSW)	STAT_RX_TOTAL_BYTES_MSB	0x080C
0x208	Transmitted Bytes (LSW)	STAT_TX_TOTAL_BYTES_LSB	0x0710
0x20C	Transmitted Bytes (MSW)	STAT_TX_TOTAL_BYTES_MSB	0x0714
0x210	Undersize Frames Received (LSW)	STAT_RX_UNDERSIZE _LSB	0x0898
0x214	Undersize Frames Received (MSW)	STAT_RX_UNDERSIZE_MSB	0x089C
0x218	Fragment Frames Received (LSW)	STAT_RX_FRAGMENT_LSB	0x08A0
0x21C	Fragment Frames Received (MSW)	STAT_RX_FRAGMENT_MSB	0x08A4
0x220	64-byte frames received OK (LSW)	STAT_RX_PACKET_64_BYTES_LSB	0x0828
0x224	64-byte frames received OK (MSW)	STAT_RX_PACKET_64_BYTES_MSB	0x082C
0x228	65-127 byte frames received OK (LSW)	STAT_RX_PACKET_65_127_BYTES_LSB	0x0830
0x22C	65-127 byte frames received OK (MSW)	STAT_RX_PACKET_65_127_BYTES_MSB	0x0834
0x230	128-255 byte frames received OK (LSW)	STAT_RX_PACKET_128_255_BYTES_LSB	0x0838
0x234	128-255 byte frames received OK (MSW)	STAT_RX_PACKET_128_255_BYTES_MSB	0x083C

Table A-9: **Statistics Counters *(Cont'd)***

Address (Hex)	Register	Register	Address (Hex)
0x238	256-511 byte frames received OK (LSW)	STAT_RX_PACKET_256_511_BYTES_LSB	0x0840
0x23C	256-511 byte frames received OK (MSW)	STAT_RX_PACKET_256_511_BYTES_MSB	0x0844
0x240	512-1023 byte frames received OK (LSW)	STAT_RX_PACKET_512_1023_BYTES_LSB	0x0848
0x244	512-1023 byte frames received OK (MSW)	STAT_RX_PACKET_512_1023_BYTES_MSB	0x084C
0x248	1024 – MaxFrameSize byte frames received OK (LSW)	STAT_RX_PACKET_1024_1518_BYTES_LSB	0x0850
		STAT_RX_PACKET_1519_1522_BYTES_LSB	0x0858
		STAT_RX_PACKET_1523_1548_BYTES_LSB	0x0860
		STAT_RX_PACKET_1549_2047_BYTES_LSB	0x0868
		STAT_RX_PACKET_2048_4095_BYTES_LSB	0x0870
		STAT_RX_PACKET_4096_8191_BYTES_LSB	0x0878
		STAT_RX_PACKET_8192_9215_BYTES_LSB	0x0880
		STAT_RX_PACKET_LARGE_LSB	0x0888
0x24C	1024 – MaxFrameSize byte frames received OK (MSW)	STAT_RX_PACKET_1024_1518_BYTES_MSB	0x0854
		STAT_RX_PACKET_1519_1522_BYTES_MSB	0x085C
		STAT_RX_PACKET_1523_1548_BYTES_MSB	0x0864
		STAT_RX_PACKET_1549_2047_BYTES_MSB	0x086C
		STAT_RX_PACKET_2048_4095_BYTES_MSB	0x0874
		STAT_RX_PACKET_4096_8191_BYTES_MSB	0x087C
		STAT_RX_PACKET_8192_9215_BYTES_MSB	0x0884
		STAT_RX_PACKET_LARGE_MSB	0x088C
0x250	Oversize frames received OK (LSW)	STAT_RX_OVERSIZE_LSB	0x08A8
0x254	Oversize frames received OK (MSW)	STAT_RX_OVERSIZE_MSB	0x08AC
0x258	64-byte frames transmitted OK (LSW)	STAT_TX_PACKET_64_BYTES_LSB	0x0720
0x25C	64-byte frames transmitted OK (MSW)	STAT_TX_PACKET_64_BYTES_MSB	0x0724
0x260	65-127 byte frames transmitted OK (LSW)	STAT_TX_PACKET_65_127_BYTES_LSB	0x0728
0x264	65-127 byte frames transmitted OK (MSW)	STAT_TX_PACKET_65_127_BYTES_MSB	0x072C
0x268	128-255 byte frames transmitted OK (LSW)	STAT_TX_PACKET_128_255_BYTES_LSB	0x0730

Table A-9: **Statistics Counters *(Cont'd)***

Address (Hex)	Register	Register	Address (Hex)
0x26C	128-255 byte frames transmitted OK (MSW)	STAT_TX_PACKET_128_255_BYTES_MSB	0x0734
0x270	256-511 byte frames transmitted OK (LSW)	STAT_TX_PACKET_256_511_BYTES_LSB	0x0738
0x274	256-511 byte frames transmitted OK (MSW)	STAT_TX_PACKET_256_511_BYTES_MSB	0x073C
0x278	512-1023 byte frames transmitted OK (LSW)	STAT_TX_PACKET_512_1023_BYTES_LSB	0x0740
0x27C	512-1023 byte frames transmitted OK (MSW)	STAT_TX_PACKET_512_1023_BYTES_MSB	0x0744
0x280	1024 – MaxFrameSize byte frames transmitted OK (LSW)	STAT_TX_PACKET_1024_1518_BYTES_LSB	0x0748
		STAT_TX_PACKET_1519_1522_BYTES_LSB	0x0750
		STAT_TX_PACKET_1523_1548_BYTES_LSB	0x0758
		STAT_TX_PACKET_1549_2047_BYTES_LSB	0x0760
		STAT_TX_PACKET_2048_4095_BYTES_LSB	0x0768
		STAT_TX_PACKET_4096_8191_BYTES_LSB	0x0770
		STAT_TX_PACKET_8192_9215_BYTES_LSB	0x0778
		STAT_TX_PACKET_LARGE_LSB	0x0780
0x284	1024 – MaxFrameSize byte frames transmitted OK (MSW)	STAT_TX_PACKET_1024_1518_BYTES_MSB	0x074C
		STAT_TX_PACKET_1519_1522_BYTES_MSB	0x0754
		STAT_TX_PACKET_1523_1548_BYTES_MSB	0x075C
		STAT_TX_PACKET_1549_2047_BYTES_MSB	0x0764
		STAT_TX_PACKET_2048_4095_BYTES_MSB	0x076C
		STAT_TX_PACKET_4096_8191_BYTES_MSB	0x0774
		STAT_TX_PACKET_8192_9215_BYTES_MSB	0x077C
		STAT_TX_PACKET_LARGE_MSB	0x0784
0x288	Oversize frames transmitted OK (LSW)	N/A	
0x28C	Oversize frames transmitted OK (MSW)		
0x290	Frames received OK (LSW)	STAT_RX_TOTAL_GOOD_PACKETS_LSB	0x0810
0x294	Frames received OK (MSW)	STAT_RX_TOTAL_GOOD_PACKETS_MSB	0x0814
0x298	Frame Check Sequence Error (LSW)	STAT_RX_PACKET_BAD_FCS_LSB	0x08C8
0x29C	Frame Check Sequence Error (MSW)	STAT_RX_PACKET_BAD_FCS_MSB	0x08CC
0x2A0	Broadcast Frames received OK (LSW)	STAT_RX_BROADCAST_LSB	0x8E8

Table A-9: **Statistics Counters *(Cont'd)***

Address (Hex)	Register	Register	Address (Hex)
0x2A4	Broadcast Frames received OK (MSW)	STAT_RX_BROADCAST_MSB	0x8EC
0x2A8	Multicast Frames received OK (LSW)	STAT_RX_MULTICAST_LSB	0x08E0
0x2AC	Multicast Frames received OK (MSW)	STAT_RX_MULTICAST_MSB	0x08E4
0x2B0	Control Frames received OK (LSW)		
0x2B4	Control Frames received OK (MSW)		
0x2B8	Length/Type out of range (LSW)	STAT_RX_INRANGEERR_LSB	0x0908
0x2BC	Length/Type out of range (MSW)	STAT_RX_INRANGEERR_MSB	0x090C
0x2C0	VLAN tagged frames received OK (LSW)	STAT_RX_VLAN_LSB	0x08F0
0x2C4	VLAN tagged frames received OK (MSW)	STAT_RX_VLAN_MSB	0x08F4
0x2C8	PAUSE frames received OK (LSW)	STAT_RX_PAUSE_LSB	0x08F8
0x2CC	PAUSE frames received OK (MSW)	STAT_RX_PAUSE_MSB	0x08FC
0x2D0	Control frames received with unsupported opcode (LSW)		
0x2D4	Control frames received with unsupported opcode (MSW)		
0x2D8	Frames transmitted OK (LSW)	STAT_TX_TOTAL_GOOD_PACKETS_LSB	0x0708
0x2DC	Frames transmitted OK (MSW)	STAT_TX_TOTAL_GOOD_PACKETS_MSB	0x070C
0x2E0	Broadcast Frames transmitted OK (LSW)	STAT_TX_BROADCAST_LSB	0x07E0
0x2E4	Broadcast Frames transmitted OK (MSW)	STAT_TX_BROADCAST_MSB	0x07E4
0x2E8	Multicast Frames transmitted OK (LSW)	STAT_TX_MULTICAST_LSB	0x07D8
0x2EC	Multicast Frames transmitted OK (MSW)	STAT_TX_MULTICAST_MSB	0x07DC
0x2F0	Underrun errors (LSW)		
0x2F4	Underrun errors (MSW)		
0x2F8	Control Frames transmitted OK (LSW)		
0x2FC	Control Frames transmitted OK (MSW)		
0x300	VLAN tagged frames transmitted OK (LSW)	STAT_TX_VLAN_LSB	0x07E8

Table A-9: **Statistics Counters *(Cont'd)***

Address (Hex)	Register	Register	Address (Hex)
0x304	VLAN tagged frames transmitted OK (MSW)	STAT_TX_VLAN_MSB	0x07EC
0x308	PAUSE frames transmitted OK (LSW)	STAT_TX_PAUSE_LSB	0x07F0
0x30C	PAUSE frames transmitted OK (MSW)	STAT_TX_PAUSE_MSB	0x07F4

Pause Processing

The following section outlines the configuration for priority based flow control.

Table A-10: **Pause Processing**

Address (Hex)	Register	Register	Address (Hex)
0x480	Priority 0 Quanta Register	CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG1 [15:0]	0x0058
0x484	Priority 1 Quanta Register	CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG1 [31:16]	0x0058
0x488	Priority 2 Quanta Register	CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG2 [15:0]	0x005C
0x48C	Priority 3 Quanta Register	CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG2 [31:16]	0x005C
0x490	Priority 4 Quanta Register	CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG3 [15:0]	0x0060
0x494	Priority 5 Quanta Register	CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG3 [31:16]	0x0060
0x498	Priority 6 Quanta Register	CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG4 [15:0]	0x0064
0x49C	Priority 7 Quanta Register	CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG4 [31:16]	0x0064
0x4A0	Legacy Pause Refresh Register	Does not support Legacy PFC	

MDIO Control Registers

The 10G/25G High Speed Ethernet IP does not provide an MDIO station master and thus does not have any of the MDIO control registers.

Interrupt Registers

Typically interrupts are generated after an MDIO operation to indicate completion; because there is no MDIO master there are no interrupt registers.

PCS/PMA MDIO register map

Again, because there is no MDIO interface provided, there are no MDIO registers for the PCS/PMA interface.

AXI4-Stream Interface

The 10G/25G High Speed Ethernet IP Subsystem provides both 64-bit and 32-bit AXI4-Stream interfaces for the datapath as does the Legacy 10G Ethernet IP Subsystem. Note the following difference in the use of `tuser` bits on the RX interface. The table below compares the definitions of the `tuser` signals on both TX and RX.

Table A-11: **Comparison of the Definitions of the tuser Signals in Both TX and RX**

Signal	Legacy XGEMAC	10/25G High Speed Ethernet IP
RX AXI4-Stream tuser	m_axis_rx_tuser AXI4-Stream User Sideband interface • 0 indicates a bad packet has been received. • 1 indicates a good packet has been received	rx_axis_tuser AXI4-Stream User Sideband interface. • 1 indicates a bad packet has been received. • 0 indicates a good packet has been received.
TX AXI4-Stream tuser	s_axis_tx_tuser AXI4-Stream user signal used to indicate explicit underrun	tx_axis_tuser AXI4-Stream User Sideband interface. Equivalent to the tx_errin signal. • 1 indicates a bad packet has been received. • 0 indicates a good packet has been received.

Debugging

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

TIP: *If the IP generation halts with an error, there might be a license issue. See License Checkers in Chapter 1 for more details.*

Finding Help on Xilinx.com

To help in the design and debug process when using the 10G/25G Ethernet, 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 10G/25G Ethernet. 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 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, software tools, and intellectual property at all stages of the design cycle. Topics include design assistance, advisories, and troubleshooting tips.

Refer to the 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 subsystem 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 10G/25G Ethernet

AR: 64710

Technical Support

Xilinx provides technical support in the Xilinx Support web page for this product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support if you do any of the following:

- Implement the solution in devices that are not defined in the documentation.
- Customize the solution beyond that allowed in the product documentation.
- Change any section of the design labeled DO NOT MODIFY.

To contact Xilinx Technical Support, go to the Xilinx Support web page.

Debug Tools

There are many tools available to address 10G/25G Ethernet design issues. It is important to know which tools are useful for debugging various situations.

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 8].

Reference Boards

Various Xilinx development boards support the 10G/25G Ethernet. These boards can be used to prototype designs and establish that the core can communicate with the system.

- UltraScale™ FPGA evaluation boards
  - VCU107
  - VCU108

Simulation Debug

Simulator License Availability

If the simulator does not launch, you might not have a valid license. Ensure that the license is up to date. It is also possible that your organization has a license available for one of the other simulators, so try all the provided scripts.

Slow Simulation

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

- Use a faster computer with more memory.
- Make use of a Platform Load Sharing Facility (LSF) if available in your organization.
- Bypass the Xilinx transceiver (this might require that the customer create their own test bench).
- Send fewer packets.
- If using the example design, see Simulation Speed Up in Chapter 4 to speed up wait timers in the example design.

Simulation Fails Before Completion

If the sample simulation fails or hangs before successfully completing, 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. Each release is tested prior to shipment and normally completes successfully. Consult the sample simulation log file for the expected behavior.

The simulation debug flow for Questa® SIM is illustrated in Figure B-1. A similar approach can be used with other simulators.

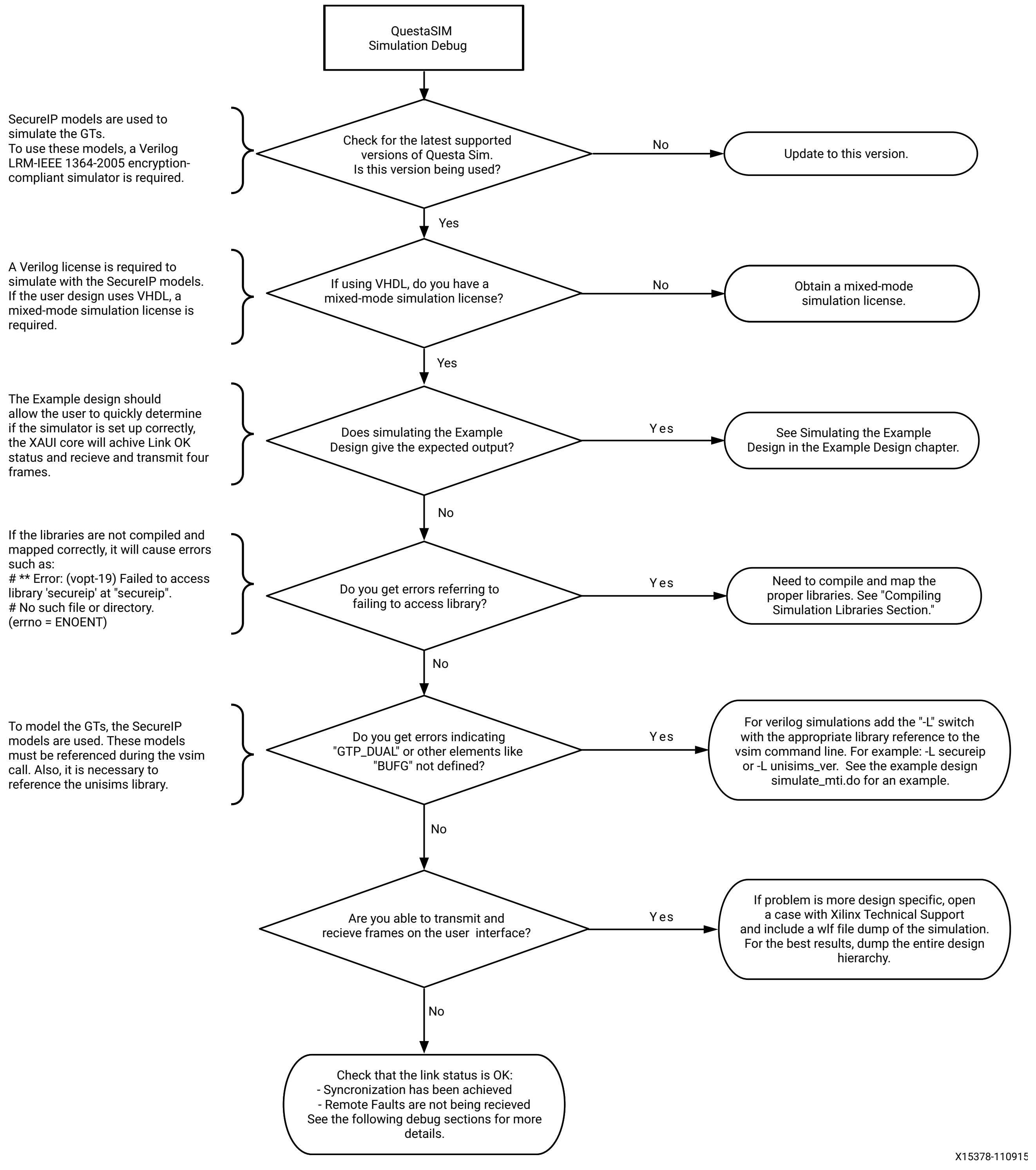

QuestaSIM
Simulation Debug
SecureIP models are used to simulate the GTs. To use these models, a Verilog LRM-IEEE 1364-2005 encryption-compliant simulator is required.
Check for the latest supported versions of Questa Sim. Is this version being used?
No
Update to this version.
Yes
A Verilog license is required to simulate with the SecureIP models. If the user design uses VHDL, a mixed-mode simulation license is required.
If using VHDL, do you have a mixed-mode simulation license?
No
Obtain a mixed-mode simulation license.
Yes
The Example design should allow the user to quickly determine if the simulator is set up correctly, the XAUI core will achive Link OK status and recieve and transmit four frames.
Does simulating the Example Design give the expected output?
Yes
See Simulating the Example Design in the Example Design chapter.
No
If the libraries are not compiled and mapped correctly, it will cause errors such as: # ** Error: (vopt-19) Failed to access library 'secureip' at "secureip". # No such file or directory. (errno = ENOENT)
Do you get errors referring to failing to access library?
Yes
Need to compile and map the proper libraries. See "Compiling Simulation Libraries Section."
No
To model the GTs, the SecureIP models are used. These models must be referenced during the vsim call. Also, it is necessary to reference the unisims library.
Do you get errors indicating "GTP_DUAL" or other elements like "BUFG" not defined?
Yes
For verilog simulations add the "-L" switch with the appropriate library reference to the vsim command line. For example: -L secureip or -L unisims_ver. See the example design simulate_mti.do for an example.
No
Are you able to transmit and recieve frames on the user interface?
Yes
If problem is more design specific, open a case with Xilinx Technical Support and include a wlf file dump of the simulation. For the best results, dump the entire design hierarchy.
No
Check that the link status is OK: - Syncronization has been achieved - Remote Faults are not being recieved See the following debug sections for more details.
X15378-110915

Figure B-1: **Mentor Graphics Questa Advanced Simulator Simulation Debug Flow**

Hardware Debug

Hardware issues can range from link bring-up to problems seen after hours of testing. This section provides debug steps for common issues. The Vivado debug feature is a valuable resource to use in hardware debug. The signal names mentioned in the following individual sections can be probed using the debug feature for debugging the specific problems.

General Checks

Ensure that all the timing constraints for the core were properly incorporated from the example design and that all constraints were 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.
- If your outputs go to 0, check your licensing.

Timing

Ensure that timing is met according to the Vivado tools before attempting to implement the IP in hardware.

Transceiver Specific Checks

- Ensure that the polarities of the txn/txp and rxn/rxp lines are not reversed. If they are, these can be fixed by using the `TXPOLARITY` and `RXPOLARITY` ports of the transceiver.
- Check that the transceiver is not being held in reset or still being initialized. The

RESETDONE outputs from the transceiver indicate when the transceiver is ready.

- Place the transceiver into parallel or serial near-end loopback.
- If correct operation is seen in the transceiver serial loopback, but not when loopback is performed through an optical cable, it might indicate a faulty optical module.
- If the core exhibits correct operation in the transceiver parallel loopback but not in serial loopback, this might indicate a transceiver issue.
- A mild form of bit error rate might be solved by adjusting the transmitter Pre-Emphasis and Differential Swing Control attributes of the transceiver.

Ethernet Specific Checks

A number of issues can commonly occur during the first hardware test of an 10G/25G Ethernet. These should be checked as indicated below.

It is assumed that the 10G/25G Ethernet has already passed all simulation testing which is being implemented in hardware. This is a pre-requisite for any kind of hardware debug.

The usual sequence of debugging is to proceed in the following sequence:

1. Clean up signal integrity.
2. Ensure that the SerDes achieves clock data recovery (CDR) lock.
3. Check that the 10/25G IP has achieved word sync.
4. Proceed to Interface and Protocol debug.

Signal Integrity

When bringing up a board for the first time and the 10/25G Ethernet does not seem to be achieving word sync, the most likely problem is related to signal integrity. Signal integrity issues must be addressed before any other debugging can take place.

Signal integrity should be debugged independently from the 10G/25G Ethernet. The following procedures should be carried out. (Note that 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.)

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

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

N/P Swapping

If the positive and negative signals of a differential pair are swapped, then data cannot be correctly received on that lane. You should verify that the link has the correct polarity of each differential pair.

Clocking and Resets

Refer to the Clocking and Resets in Chapter 3 for these requirements.

Ensure that the clock frequencies for both the 10G/25G Ethernet as well as the Xilinx Transceiver reference clock match the configuration requested when the subsystem 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 10G/25G Ethernet is taken out of reset. This applies to both the SerDes clock as well as the core clock.

If any subsequent instability is detected in a clock, the 10G/25G Ethernet must be reset. One example of such instability is a loss of CDR lock. The user logic should determine all external conditions which would require a reset (e.g. clock glitches, loss of CDR lock, power supply glitches, etc.).

The GT requires a GTRXRESET after the serial data becomes valid to ensure correct CDR lock to the data. This is required after powering on, resetting or reconnecting the link partner. At the core level to avoid interruption on the TX side of the link, the reset can be triggered using `gtwiz_reset_rx_datapath`. If available, signal detect or inversion of loss of signal from the optics can be used to trigger the reset. If signal detect or loss of signal are not available, timeout logic can be added to monitor if alignment has not completed and issue the `gtwiz_reset_rx_datapath` reset.

Configuration changes cannot be made unless the subsystem 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.

RX Debug

Consult the port list section for a description of the diagnostic signals which are available to debug the RX.

stat_rx_block_lock

This signal indicates that the receiver has detected and locked to the word boundaries as defined by a 01 or 10 control or data header. This is the first step to ensure that the 10/25G Ethernet IP is functioning normally.

CAUTION! *Under some conditions of no signal input, the SerDes receiver exhibits a steady pattern of alternating 1010101.... This can cause erroneous block lock, but still indicates that the receiver has detected the pattern.*

stat_rx_bad_fcs

A bad FCS indicates a bit error in the received packet. An FCS error could be due to any number of causes of packet corruption such as noise on the line.

stat_rx_local_fault

A local fault indication can be locally generated or received. Some causes of a local fault are:

- block lock not complete
- high bit error rate
- overflow or underflow

Loopback Check

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

To aid in debug, a local loopback can be performed with the signal `ctl_local_loopback`. This connects the TX SerDes to the RX SerDes, effectively bypassing potential signal integrity problems. The transceiver is placed into "PMA loopback", which is fully described in the transceiver product guide. In this way, the received data can be checked against the transmitted packets to verify that the logic is operating properly

Protocol Interface Debug

To achieve error-free data transfers with the 10G/25G Ethernet, the 802.3 specification should be followed. Note that signal integrity should always be ensured before proceeding to the protocol debug.

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 problem.

The following is a suggested debug sequence:

1. Ensure that Word sync has been achieved.
2. Make sure there are no descrambler state errors.
3. Eliminate CRC32 errors, if any.
4. Make sure the protocol is being followed correctly.
5. Ensure that there are no overflow or underflow conditions when packets are sent.

Statistics Counters

After error-free communication has been achieved, the statistics indicators can be monitored to ensure that traffic characteristics meet expectations. Note that some signals are strobes only, which means that the counters are not part of the subsystem. This is done so that the counter size can be customized. Counters are optionally available with the AXI interface.

Additional Resources and Legal Notices

Xilinx Resources

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

Documentation Navigator and Design Hubs

Xilinx® Documentation Navigator provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open the Xilinx Documentation Navigator (DocNav):

- From the Vivado® IDE, select **Help > Documentation and Tutorials**.
- On Windows, select **Start > All Programs > Xilinx Design Tools > DocNav**.
- At the Linux command prompt, enter `docnav`.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In the Xilinx Documentation Navigator, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

Note: For more information on Documentation Navigator, see the Documentation Navigator page on the Xilinx website.

References

These documents provide supplemental material useful with this product guide:

1. 25G and 50G Ethernet Consortium Schedule 3 version 1.6 (August 18, 2015) (http://25gethernet.org/)
2. IEEE Standard 802.3-2015
3. IEEE Standard for Ethernet (IEEE Std 802.3by™-2016 Amendment 2: Media Access Control Parameters, Physical Layers, and Management Parameters for 25 Gb/s Operation)
4. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
5. *Vivado Design Suite User Guide: Designing with IP* (UG896)
6. *Vivado Design Suite User Guide: Getting Started* (UG910)
7. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
8. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
9. *Vivado Design Suite User Guide - Implementation* (UG904)
10. *Vivado Design Suite AXI Reference Guide* (UG1037)
11. IEEE Standard 1588-2008, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems" (standards.ieee.org/findstds/standard/1588-2008.html)
12. *25G IEEE 802.3by Reed-Solomon Forward Error Correction LogiCORE IP Product Guide* (PG217) Registration required for access.
13. *UltraScale Architecture GTH Transceivers User Guide* (UG576)
14. *UltraScale Architecture GTY Transceivers User Guide* (UG578)
15. *Vivado Design Suite User Guide: Using Constraints* (UG903)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/06/2018	2.4	• Updated 10G TSN license key information. • Added stat_reg_compare port in the Example Design chapter. • Renamed MAC+PCS/PMA 32-bit 1588 register. • Added table note for Table 3-4.
04/04/2018	2.4	• Added fee-based Time Sensitive Networking (TSN) feature to Features section in IP Facts. • Added 25G/10G Ethernet MAC+PCS/PMA description to Table 1-2. • Added support for 802.1cm preemption feature to the 25G Supported Features and 10G Supported Features sections in Chapter 1. • Updated the licensing information in Chapter 1. • Added a note to the ctl_local_loopback description in Table 2-12. • Added a note for the ctl_autoneg_bypass signal in Table 2-96. • Added Bit 7 for stat_tx_bad_parity signal in Table 2-110. • Added a note for Bit 1, runtime_switchable signal, in Table 2-128 • Added option about 1588 time stamp accuracy in the Ingress section in Chapter 3. • Added the new sections, Ethernet Data Path Parity, and 802.1cm Preemption Feature to Chapter 3. • Updated description in the second paragraph of the Ethernet Data Path Parity second in Chapter 3. • Updated Figures 4-1, 4-2, 4-3, and 4-4 • Added Enable Preemption 802.3br option to Table 4-1. • Added the Enable Datapath Parity and Enable Packet Assembly FIFO options to Table 4-2. • Updated a note about default frequencies in Table 3-1. • Added new content to the Required Restraints section in Chapter 4. • Added the .h Header File subsection to the AXI4-Lite Interface Implementation section in Chapter 5. • Updated Figure 6-1 in Chapter 6. • Added *Vivado Design Suite User Guide: Using Constraints* (UG903) to the References section in Appendix C. • For register and port additions see Changes from v2.3 to v2.4 in Appendix A.

Date	Version	Revision
12/20/2017	2.3	• Updated Clause 108 RS-FEC to support the following entries in Table 1-1: Runtime switchable 10G/25G MAC+PCS and Runtime switchable 10/25G PCS-only. • In Table 3-1, changed the name of the first column from "Local Fault Indication" to "Name". • For port and register additions, removal, and changes, see Changes from v2.3 (10/04/2017) to v2.3 (12/20/2017) in Appendix A. • Updated Figure 3-2. • Added 312.5 MHz for 32-bit 10G to refclk_p0, refclk_n0, tx_serdes_refclk description. • Added a note to the following signals about an invalid preamble: stat_rx_bad_sfd, stat_rx_bad_preamble, stat_rx_bad_preamble_*, and stat_rx_bad_sfd_* in tables 2-13, 2-37, and 5-2.
10/04/2017	2.3	• Added a note to Table 1-1 about auto-negotiation. • Added text about client data logic to the Back-to-Back Continuous Transfers section in Chapter 2. • Removed Table 3-2 and supporting text from the Performance section in Chapter 3. • Added a note about the readable STAT_*_MSB/LSB registers to the Statistics Counters section in Chapter 2. • Added Clock Domain columns to Table 2-4 and Table 2-7 and updated clock domain values in Table 2-11. • Added text about runtime switch mode to beginning of Status Registers section in Chapter 2. • Modified ordinary and transparent clock numbers in Egress section in Chapter 3. • Added text about 2-step 1588 operation for tx_ptp_tstamp_out[80-1:0] and rx_ptp_tstamp_out[80-1:0] in Table 3-1. • Updated Legal Notices and Automotive Applications Disclaimer. • Updated screen captures in Chapter 4. • Added text about SIM_SPEED_UP to the Simulation Speed Up section in Chapter 4. • For axi_ctl_core_mode_switch_* in Table 5-2. changed value to 0x0138 • Added, changed and removed some ports and registers. See Changes from v2.2 to v2.3.
06/07/2017	2.2	• Updated the Ordering Information section. • Updated the Supported User Interfaces row and the Supported S/W Driver row in the IP Facts table. • See Changes from v2.1 to v2.2 for new variants, feature updates, and port additions and updates. • Updated Table 1-1, Table 1-2, and Table 2-1. • Updated screen captures in Chapter 4. • Removed the AN/LT Clock option from Table 4-1.

Date	Version	Revision
04/05/2017	2.1	• Changed Supported User Interface in IP Facts table to AXI4-Stream. • Updated feature list with the new features. • Added 10G/25G Runtime Switchable IP Features section and Feature Compatibility Matrix table to Chapter 1. • Added Ordering Information section to Licensing and Ordering Information Chapter 1. Added new license key for standalone 64-bit MAC in new Table 1-2. • Changed Figure 2-1 title to 25 Gb/s Core Block Diagram. • Added new Figure 2-2: 10 Gb/s Core Block Diagram and Figure 2-4: 64-bit Standalone Version of the MAC for 10 Gb/s Operation. • Added AXI4-Stream Interface heading with new 32-bit information throughout the subsections and new timing diagrams for 32-bit operation. • Added new section for the 64-bit 10G MAC offering. • Updated the 64-bit MAC+PCS variant to include the new 32-bit low latency 10G MAC + PCS variant. • Changed "Port Descriptions" name to "Port Descriptions – MAC+PCS Variant" • Added Port Descriptions – 10G Ethernet MAC (64-bit) Variant section and its subsections. • Added new row to Table 2-5 for latency. • Updated most of the Notes in Table 2-39 through Table 2-41.
		• Added Low Latency 32-bit 10 Gb/s MAC with PCS and 10G MAC-only Clocking sections in Chapter 3. • Updated IBUFDS_GTE3 in Figures 3-7 and 3-8. • Removed Figure 3-6 and Figure 3-8 (Synchronous Clock Modes) • Updated the Select Core and Clocking options in Table 4-1. • Updated Figure 4-1 through Figure 4-4. • Replaced “Ethernet MAC+PCS/PMA” with “Ethernet MAC+PCS/PMA-32/64-bit” in all instances and insert "or Ethernet MAC" throughout Table 5-2. • Updated Migration Guide to include the AXI4-Stream Interface • Changed LGMII to XGMII/25GMII throughout. • Changed XXVGMII to 25GMII throughout • For port and register changes, see Appendix A, Migrating and Updating.

Date	Version	Revision
11/30/2016	2.0	• Modified "tx_reset and rx_reset" to "s_axi_aresetn' and "active-High" to "active-Low" in the Configuration Register Map section in Chapter 2. • Added text about clearing status registers to the first paragraph of the Status Register Map section in Chapter 2. • Added text about clearing statistics counters to the first paragraph of the Statistics Counters section in Chapter 2. • Updated table notes 3 and 4 and added table note 5 for Table 4-1, Configuration Options. • Added Note about Auto-Negotiation/Link training to the Overview section in Chapter 5. • Added many ports and deleted many ports in Table 5-2, Core xci Top Level Port List. See the Migrating and Updating appendix. • Removed the text "GT Selection and" throughout the Descriptions in Table 2, Core xci Top Level Port List.
10/05/2016	2.0	• Added migration from legacy 10G EMAC to Appendix A Migrating and Updating. • Added references to tick_reg_mode_sel throughout. • Updated the following figures: 3-6, 3-7, 3-8, 3-9, 4-1, 4-2, 4-3, 4-4 • Updated the description of tx_ptp_1588op_in[1:0] and rx_ptp_tstamp_out[80-1:0] in Table 3-1. • Added support for one-step operation. • Updated several of the port descriptions in Table 3-3. • Added Ethernet MAC value to Select Core option in Table 4-1. • Replaced Include FEC Logic option with Clause 74 (BASE-KR FEC) in Table 4-1. • Added Clause 108 (RS-FEC) option to Table 4-1. • Added the new subsection Simulation Speed Up to the Simulation section in Chapter 4. • Updated the description of ctl_rx_rate_10g_25gn_* in Table 5-2. • Added the rx_ptp_tstamp_valid_out_* to Table 5-2. • Added several new subsection under the AXI4-Lite Interface Implementation section in Chapter 5. • Added Step 5 in the Slow Simulation section in the Debugging appendix. • Added a new paragraph about GTRXRESET in the Clocking and Resets section in the Debugging appendix. • Added the Core xci Top Level Port List section to Chapter 5. • Updated IEEE references to 2015 instead of 2012.

Date	Version	Revision
06/08/2016	1.3	• Changed 10 Gb/s to 10.3125 Gb/s throughout • Updated Figures 2-2, 3-16, 4-1, 4-2, 4-3,4-4, 5-1, 5-2, 5-3, 5-4, 5-5 • Added XGMII to XVGMII throughout. • Changed XXVMII to XVGMII throughout. • Added notes for addresses that support MAC+PCS throughout. • Added Hex Addresses and links in Table 2-24. • Added Bits to Tables 2-25, 2-28, 2-86 • Added new register tables for STAT_TX_RSFEC_STATUS_REG: 044C, STAT_RX_ERROR_LSB: 0668, STAT_RX_ERROR_MSB: 066C, STAT_RX_RSFEC_ERR_COUNT0_INC_LSB: 0680, STAT_RX_RSFEC_ERR_COUNT0_INC_MSB: 0684 • Removed General Design Guidelines section in Chapter 3. • Added tx_ptp_rxtstamp_in to Table 3-1 • Changed "HSEC" to "10G/25G High Speed Ethernet Subsystem" throughout. • Added Control and Statistics Interface section to Table 4-1 • Added GT Location section to Table 4-3 and updated options in the Others section
06/08/2016	1.3	• Updated some descriptions in Table 4-3, GT Clock Options • Updated Overview in the Chapter 5, Example Design • Updated the descriptions of the optional modules. • Added the Example Design Hierarchy (GT in example design), Runtime Switchable, and IEEE Clause 108 (RS-FEC) Integration sections to Chapter 5. • Completely revised the Shared Logic Implementation section. in Chapter 5. • Added descriptions of the modules that are part of the shared logic wrapper. • Completely revised the Simulation Debug section in Appendix B, Debugging. • Changed 802.3-2012 to 802.3-2015 throughout. • Added one-step operation throughout. • Updated several port descriptions in Table 3-3. • Added Simulation Speed Up section to Chapter 4. • Added new port rx_ptp_tstamp_valid_out_* to Table 5-2. • Added Step 5 to the Slow Simulation section in the Debugging appendix. • Added a paragraph about GTRXRESET to the Clocking and Resets section the Debugging appendix.

Date	Version	Revision
04/06/2016	1.2	• Added UltraScale+ support. • Added new section that has RSFEC, 1588 1-step and 2-step support. • Added new IEEE 1588 Timestamping section. • Added rx_preambleout [55:0] for both AXI4-Stream interfaces. • Added tx_preamblein [55:0] for AXI4-Stream interface. • Added registers to the Configuration, Status, and Counter register maps. • Changed custom preamble from in-band to out-of-band. • Added text about pm_tick and TIC_REG to Statistics Counter section • Changed polarity of the tx_axis_tuser and rx_axis_tuser signals. • Updated Figure 3-13 and Figure 3-14. • Removed VLane Adjust Mode from Table 4-2. • Removed LBUS material. • Added ctl_tx_ipg_value[3:0] to Table C-4.
12/02/2015	1.1	• Updated the performance and resource utilization data link.
11/18/2015	1.1	• Added a link to the performance and resource utilization data on the web. • Added the stat_rx_valid_ctrl_code, ctl_tx_custom_preamble_enable, and ctl_rx_custom_preamble_enable signal. • Updated the tx_axis_ tuser signal description. • Updated the Normal Transmission and Aborting a Transmission information in the Transmit AXI4-Stream Interface section. • Added Vivado IDE option details in Design Flow Steps chapter. • Added new information in Example Design chapter.
09/30/2015	1.0	Initial Xilinx release.

Please Read: Important Legal Notices