1G/10G/25G Switching Ethernet Subsystem v2.6

Product Guide

Vivado Design Suite

PG292 (v2.6) February 5, 2021

XILINX®

Table of Contents

Chapter 1: Introduction **4**
- Features 4
- IP Facts 5

Chapter 2: Overview **6**
- Navigating Content by Design Process 6
- Subsystem Overview 6
- Feature Summary 7
- Applications 8
- Licensing and Ordering 8

Chapter 3: Product Specification **10**
- Standards 12
- Performance and Resource Utilization 12
- Latency 12
- Port Descriptions 13
- Register Space 49

Chapter 4: Designing with the Subsystem **65**
- Clocking 65
- Resets 68
- LogiCORE Example Design Clocking and Resets 70
- Support for IEEE Standard 1588v2 74
- RS-FEC Support 83
- 802.1cm Preemption Feature 86
- Status/Control Interface 91
- Pause Processing 92
- Auto-Negotiation 96
- Link Training 103

Chapter 5: Design Flow Steps **108**
- Customizing and Generating the Subsystem 108

Constraining the Subsystem....116
Simulation....117
Synthesis and Implementation....118

Chapter 6: Example Design....119
Overview....119
Example Design Hierarchy (GT in Example Design)....125
User Interface....129
Core xci Top Level Port List....131
Duplex Mode of Operation....159
AXI4-Lite Interface Implementation....160
PTP 1588 Timer Syncer IP....164

Chapter 7: Batch Mode Test Bench....177

Appendix A: Debugging....178
Finding Help on Xilinx.com....178
Migrating from the Legacy XGEMAC....179
Debug Tools....181
Simulation Debug....182
Hardware Debug....184
Protocol Interface Debug....187

Appendix B: Additional Resources and Legal Notices....192
Xilinx Resources....192
Documentation Navigator and Design Hubs....192
References....193
Revision History....193
Please Read: Important Legal Notices....195

Introduction

The Xilinx® 1G/10G/25G Switching Ethernet Subsystem includes MAC+PCS/PMA switching subsystems in two variants as well as a PCS/ PMA switching subsystem variant. It offers a flexible solution for connection to transmit and receive data interfaces using an AXI4-Stream interface for the MAC+PCS/PMA configuration, and an XGMII/GMII interface for the PCS/PMA configuration.

Features

- Designed to the Ethernet requirements for 1/10 Gb/s operation specified by IEEE 802.3 Clause 49 or Clause 36
- Runtime switchable Ethernet MAC and PCS/ PMA functions for 1/10/25 Gb/s operation
- Supports only GTHE3/GTYE3 and GTHE4/GTYE4 transceiver supported devices
- AXI4-Lite interface for control bus
- 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
- PCS operating at 25.78125 Gb/s or 10.3125 Gb/ s or 1.25 Gb/s
- Optional Clause 73 Auto-Negotiation with Parallel Detection support
- Optional Clause 72 Link Training for 64-bit variant
- Optional Clause 74 BASE-KR FEC sublayer for 64-bit variant
- Optional clause 108 Reed-Solomon for 64-bit variant
- Optional IEEE 1588 two-step hardware timestamping
- Single channel (lane) support

- Optional time sensitive networking (TSN) (for 64-bit variant) feature designed to IEEE standard 802.1CM
  - Supports interspersing express traffic with low priority traffic
  - Supports frame preemption

IP Facts

Facts Table	
Subsystem Specifics	
Supported Device Family[1]	Versal™ ACAP Zynq® UltraScale+™ RFSoC Zynq® UltraScale+™ MPSoC Virtex® UltraScale+™ Kintex® UltraScale+™ Virtex® UltraScale™ Kintex UltraScale
Supported User Interfaces	AXI4-Stream and AXI4-Lite for all variants XGMII and GMII for PCS-only variants
Resources	Performance and Resource Utilization web page
Provided with Subsystem	
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	N/A
Tested Design Flows[2]	
Design Entry	Vivado Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado Synthesis
Support	
Release Notes and Known Issues	Master Answer Record: 72695
All Vivado IP Change Logs	Master Vivado IP Change Logs: 72775
Xilinx Support web page	

Notes:

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

XILINX

Chapter 2

Overview

Navigating Content by Design Process

Xilinx® documentation is organized around a set of standard design processes to help you find relevant content for your current development task. All Versal™ ACAP design process Design Hubs can be found on the Xilinx.com website. This document covers the following design processes:

- **Hardware, IP, and Platform Development:** Creating the PL IP blocks for the hardware platform, creating PL kernels, subsystem functional simulation, and evaluating the Vivado® timing, resource use, and power closure. Also involves developing the hardware platform for system integration. Topics in this document that apply to this design process include:
  - Port Descriptions
    - Port Descriptions
  - Register Space
  - Clocking
  - Resets
  - Customizing and Generating the Subsystem
  - Chapter 6: Example Design

Subsystem Overview

This document details the features of the 1G/10G/25G Ethernet Subsystem dynamically switching PCS/PMA and MAC core. The 10G/25G Ethernet Subsystem is defined by the 25G Ethernet Consortium. 10G PCS functionality is defined by IEEE Standard 802.3, 2015, Clause 49, Physical Coding Sublayer (PCS) for 64B/66B, type 10GBASE-R. 1G PCS functionality is defined in Clause 36. 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 the following table for compatibility of options with the different variants of the LogiCORE IP subsystem.

1G/10G/25G Supported Features

- Complete MAC and PCS functions
- 10G BASE-R mode based on IEEE 802.3 Clause 49 or 1000BASE-X mode based on IEEE 802.3 Clause 36
- 32-bit/64-bit AXI4-Stream user interface for the MAC + PCS mode of operation
- XGMII and GMII interfaces for the PCS-only mode of operation
- AXI4-Lite control and status interface
- Statistics and diagnostics
- Custom preamble and adjustable interframe gap for the 64-bit variant
- Optional Clause 73 Auto-Negotiation with Parallel Detection support
- Optional Clause 72 Link Training for the 64-bit variant
- Optional Clause 74 FEC sublayer: shortened cyclic code (2112, 2080) for the 64-bit variant
- Optional Clause 108 RS-FEC for 64-bit variant
- Optional 802.1CM preemption feature for 64-bit variant
- Pause Processing including IEEE Std 802.3 Annex 31D (Priority based Flow Control) for the 64-bit variant

Table 1: **Feature Compatibility Matrix**

Variant	User Interface	MAC	PCS	Pause Processing	Auto Negotiation Clause 73[3]	Auto Negotiation Clause 37[3]	Link Training[3]	Clause 74 FEC[3]	Clause 108 RS FEC	IEEE 1588 HW Time Stamp
1G/10GMAC with PCS[1]	32-bit AXI4-Stream	X	X		X					X
1G/10G/25G MAC with PCS[2]	64-bit AXI4-Stream	X	X	X	X		X	X	X	
1G/10G PCS only	32-bit XGMII and GMII		X		X	X				

Table 1: **Feature Compatibility Matrix** *(cont'd)*

Variant	User Interface	MAC	PCS	Pause Processing	Auto Negotiation Clause 73[3]	Auto Negotiation Clause 37[3]	Link Training [3]	Clause 74 FEC[3]	Clause 108 RS FEC	IEEE 1588 HW Time Stamp
Preemption (802.1CM) 1G/10G/25G only	64-bit AXI4-Stream	X	X						X	X

Notes:

1. Only two-step timestamping is supported.
2. The 64-bit option is only available for the 1G/10G/25G targeting GTY Transceiver.
3. This feature is not supported for Versal device.

Applications

IEEE Std 802.3 enables several different Ethernet speeds for Local Area Network (LAN) applications. The IP core delivers the capability to switch between 25GBASE-R, 10GBASE-R, and 1000BASE-X PHY.

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 an error. License checkpoints are enforced by the following tools:

- Vivado Synthesis
- Vivado Implementation
- write_bitstream (Tcl 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 1G/10G/25G Switching Ethernet page.

Ordering Information

To purchase any of these IP cores, contact your local Xilinx Sales Representative referencing the appropriate part number(s) in the following table:

Table 2: **Ordering Information**

Description	Part Number	License Key
1G/10G/25G Ethernet MAC + BASE-R PCS/PMA	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
• 1G/10G BASE-KR PCS/PMA (CL74 FEC, AN/LT) • Standalone PCS/PMA only	EF-DI-25GBASE-KR-PROJ EF-DI-25GBASE-KR-SITE ***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
1G/10G BASE-R PCS/PMA	Included with the Vivado design tools. No purchase necessary.	No license key

Notes:

1. Used for 25GBASE-CR, 25GBASE-KR, or 25GBASE-SR applications.

Product Specification

The following figures show the block diagrams of the 1G/10G/25G Switching Ethernet Subsystem, respectively.

Figure 1: **Ethernet 1/10/25G Dynamically Switching 32-bit MAC and PCS/PMA IP Block Diagram**

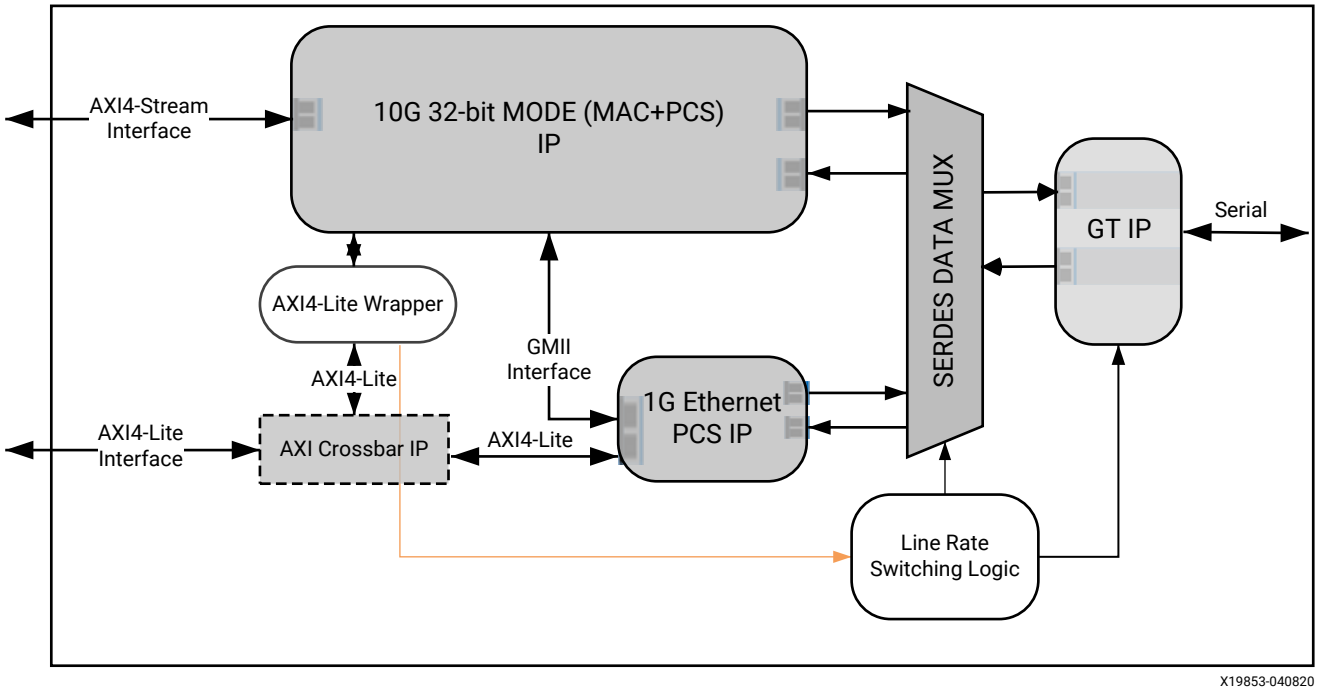


Figure 2: **Ethernet 1/10/25G Dynamically Switching 64-bit MAC and PCS/PMA IP Block Diagram**

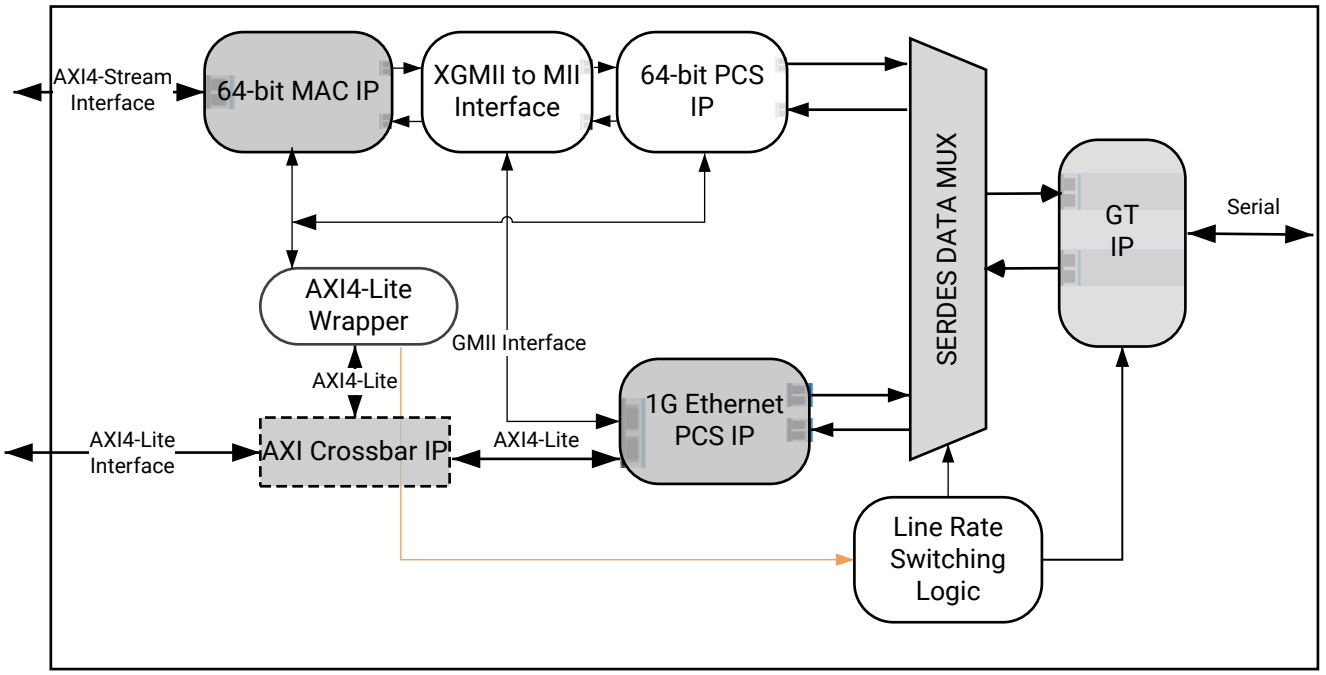
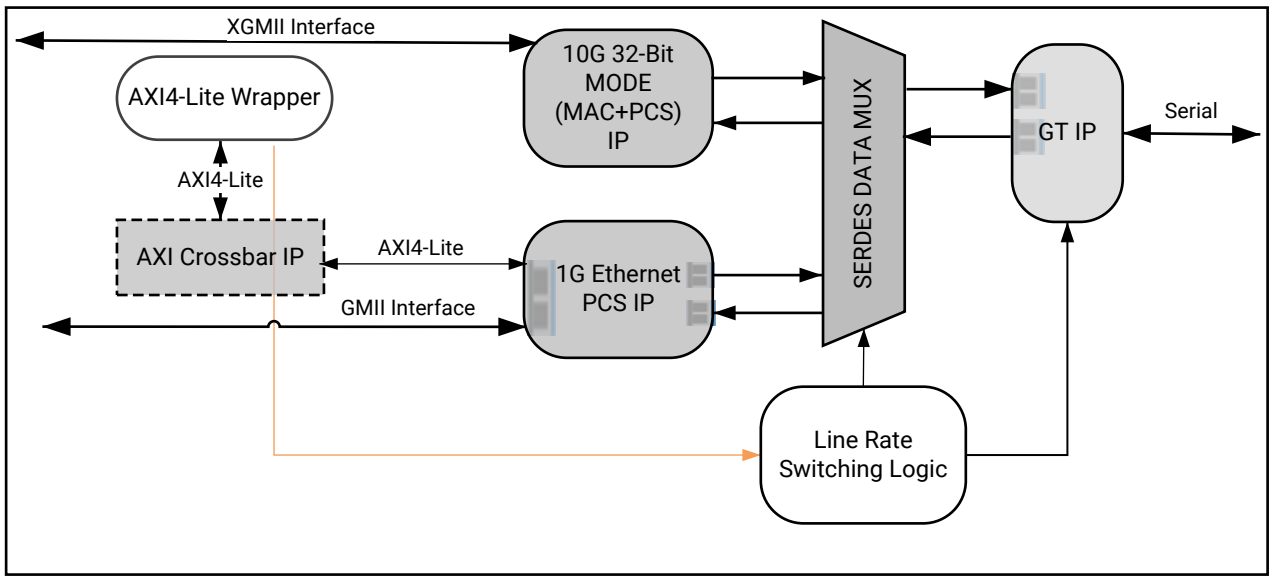


Figure 3: **Ethernet 1/10G Dynamically Switching 32-bit PCS/PMA IP Block Diagram**



Standards

The 1G/10G/25G Ethernet Subsystem is designed to the standard specified in *IEEE Standard for Ethernet* (IEEE Std 802.3-2015).

Performance and Resource Utilization

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

Latency

The following table provides the measured low latency information for the 1G/10G/25G Ethernet Subsystem.

Table 3: **Latency**

Core	Core Configuration	Latency (ns)	User Bus Width (bits)	Core Clock Frequency (MHz)
32-bit MAC + PCS/PMA	10G mode TX latency	28.8	32	312.5
	1G mode TX latency	632	32	125
	10G mode RX latency	25.6	32	312.5
	1G mode TX latency	304	32	125
64-bit MAC + PCS/PMA	25G mode TX latency	77	64	390.625
	10G mode TX latency	192	64	156.25
	1G mode TX latency	864	64	125
	25G mode RX latency	113	64	390.625
	10G mode RX latency	281	64	156.25
	1G mode RX latency	568	64	125
32-bit PCS/PMA	10G mode TX latency	19.2	32	312.5
	1G mode TX latency	80	8	125
	10G mode RX latency	19.2	32	312.5
	1G mode RX latency	88	8	125

Port Descriptions

The following tables list the ports for the 1G/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 using the registers instead of the broadside bus.

Transceiver Interface

The following table shows the transceiver I/O ports for the 1G/10G/25G Ethernet Subsystem. See Clocking for details on each clock domain.

Table 4: **Transceiver I/O**

Name	I/O	Description	Clock Domain
gt_tx_reset	I	Reset for the gigabit transceiver (GT) TX.	Async
gt_rx_reset	I	GT RX reset.	Async
ctl_gt_reset_all	I	Active-High asynchronous reset for the transceiver startup Finite State Machine (FSM). Note that this signal also initiates the reset sequence for the entire 1G/10G/25G Ethernet Subsystem.	Async
refclk_n0	I	Differential reference clock input for the SerDes, negative phase.	See Clocking.
refclk_p0	I	Differential reference clock input for the SerDes, positive phase.	See Clocking.
rx_serdes_data_n0	I	Serial data from the line; negative phase of the differential signal	See Clocking.
rx_serdes_data_p0	I	Serial data from the line; positive phase of the differential signal	See Clocking.
tx_serdes_data_n0	O	Serial data to the line; negative phase of the differential signal.	See Clocking.
tx_serdes_data_p0	O	Serial data to the line; positive phase of the differential signal.	See Clocking.
tx_serdes_clkout	O	When present, same as tx_clk_out.	See Clocking.

Related Information

Clocking

AXI4-Stream Interface

The 1G/10G/25G Switching Ethernet Subsystem IP core provides an option of 32-bit and 64-bit AXI4-Stream interface for the Ethernet MAC+PCS/PMA core configuration. For the 1G/10G switching IP, the 32-bit and 64-bit interfaces are provided. For the 1G/10G/25G switching IP, only the 64-bit interface is provided.

AXI4-Stream Clocks and Resets

Table 5: **AXI4-Stream Interface Clock/Reset Signals**

Name	I/O	Description	Clock Domain
rx_clk_out	O	Receive AXI4-Stream clock. All signals between the 1G/10G/25G Ethernet Subsystem and the user-side logic are synchronized to the positive edge of this signal. This clock is 125 MHz for 1G configuration, 156.25 / 312.5 MHz for 10G core configuration and 390.625 MHz for 25G configuration.	See Clocking.
tx_clk_out	O	Transmit AXI4-Stream clock. All signals between the 1G/10G/25G Ethernet Subsystemand the user-side logic are synchronized to the positive edge of this signal. This clock is 125 MHz for 1G configuration, 165.25 / 312.5 MHz for 10G core configuration and 390.625 MHz for 25G configuration.	See Clocking.
rx_reset	I	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	I	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	I	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

Related Information

Clocking

Transmit AXI4-Stream Interface

The following table shows the AXI4-Stream transmit interface signals.

Table 6: **AXI4-Stream Transmit Interface Signals**

Signal	I/O	Description	Clock Domain
tx_axis_tdata[63 or 31:0]	I	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]	I	AXI4-Stream Data Control. Bus width depends on selection of 64-bit or 32-bit interfaces.	tx_clk_out
tx_axis_tvalid	I	AXI4-Stream Data Valid input	tx_clk_out

Table 6: **AXI4-Stream Transmit Interface Signals** *(cont'd)*

Signal	I/O	Description	Clock Domain
tx_axis_tuser	I	AXI4-Stream User Sideband interface. • 1 indicates a bad packet • 0 indicates a good packet	tx_clk_out
tx_axis_tlast	I	AXI4-Stream signal indicating End of Ethernet Packet.	tx_clk_out
tx_axis_tready	O	AXI4-Stream acknowledge signal to indicate to start the Data transfer.	tx_clk_out

Data Lane Mapping - TX

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

Table 7: **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 8: **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 timings for a normal frame transfer are shown in the following figures 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 4: **Normal Frame Transfer - 32-bit**

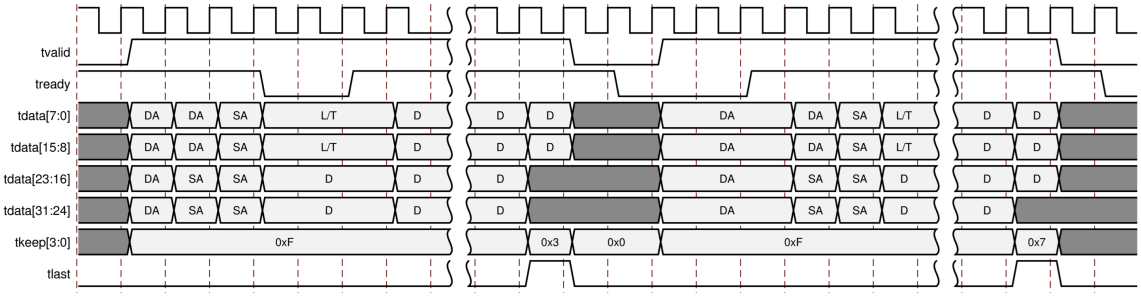
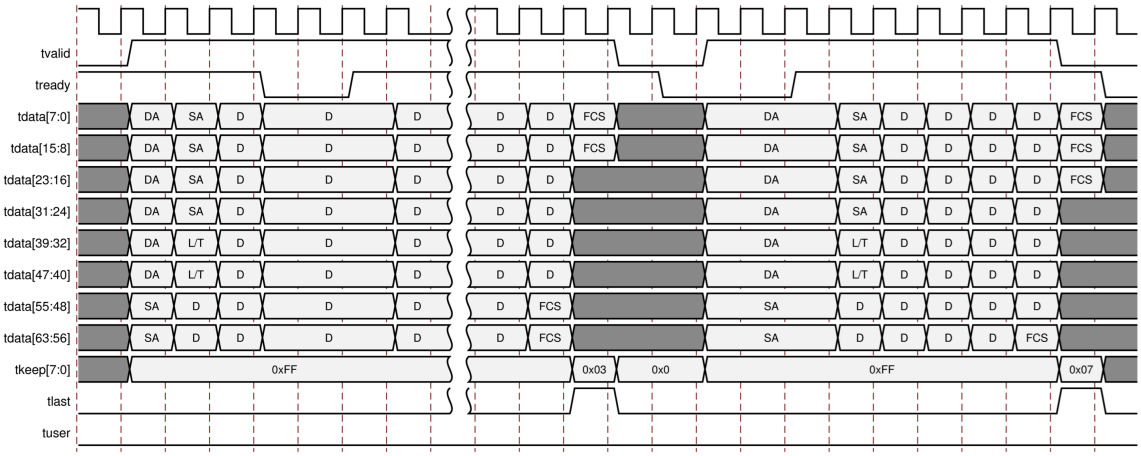


Figure 5: **Normal Frame Transfer - 64-bit**



Back-to-Back Continuous Transfers

Continuous data transfer on the transmit AXI4-Stream interface is possible, because the `tx_axis_tvalid` signal can remain continuously High, with packet boundaries defined solely by the `tx_axis_tlast` signal 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 the following two figures. 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 the `tx_axis_tready` signal.

Figure 6: **Back-to-Back Continuous Transfer on Transmit Client Interface—32-bit**

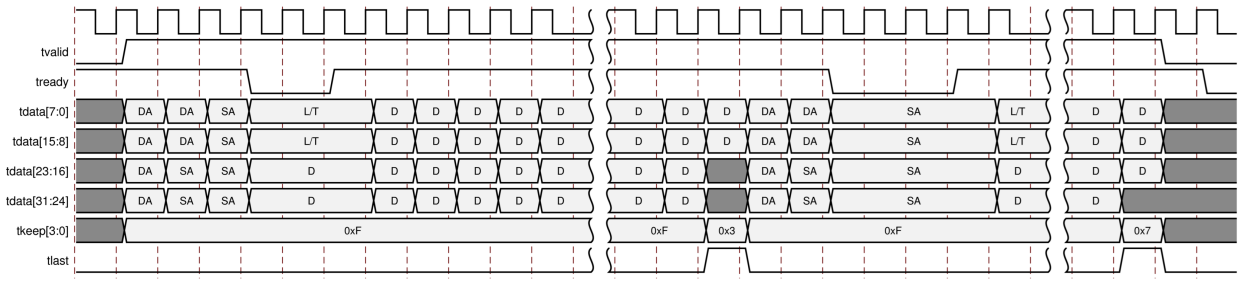
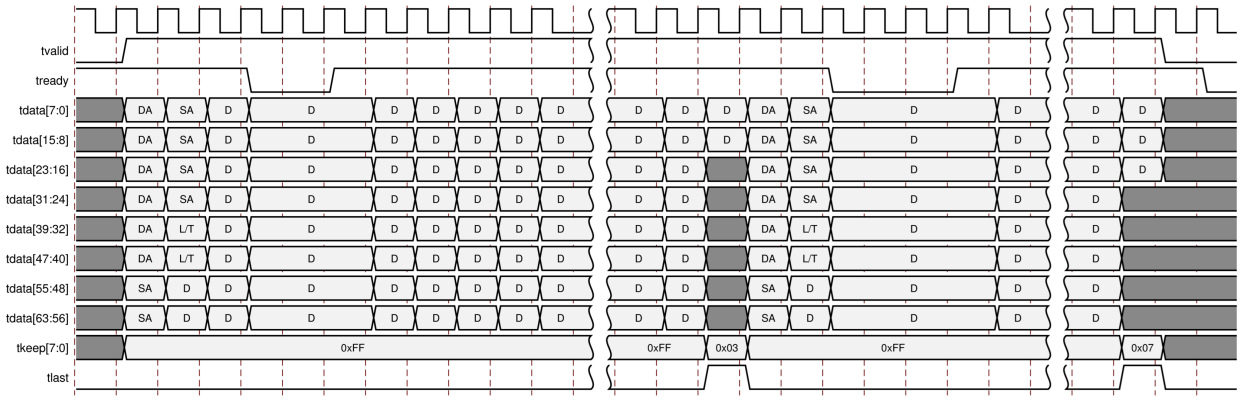


Figure 7: **Back-to-Back Continuous Transfer on Transmit Client Interface—64-bit**



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.

- 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 9: **AXI4-Stream Receive Interface Signals**

Signal	I/O	Description	Clock Domain
rx_axis_tdata[63 or 31:0]	O	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]	O	AXI4-Stream Data Control to upper layer. Bus width depends on 64-bit or 32-bit selection.	rx_core_clk
rx_axis_tvalid	O	AXI4-Stream Data Valid	rx_core_clk
rx_axis_tuser	O	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	O	AXI4-Stream signal indicating an end of packet.	rx_core_clk

Data Lane Mapping - RX

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

Table 10: **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 11: **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 11: **rx_axis_tkeep Lanes - 64-bits** *(cont'd)*

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

Normal Frame Reception

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 8: **Normal Frame Reception – 32-bits**

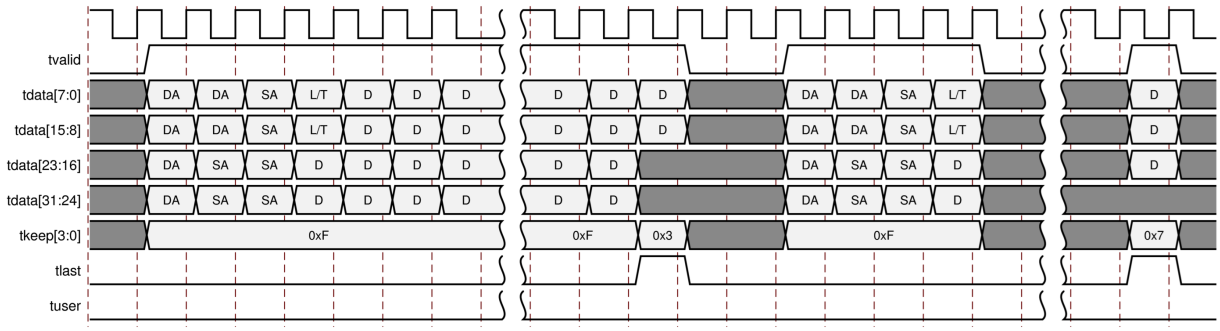
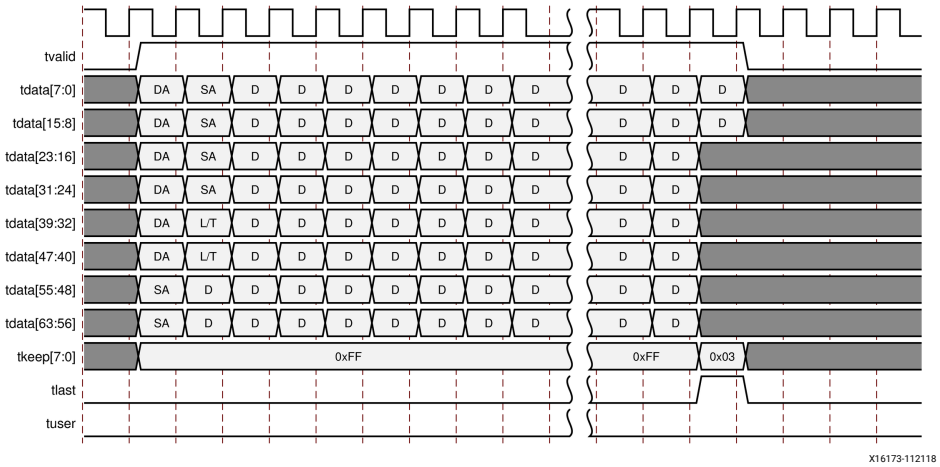


Figure 9: **Normal Frame Reception – 64-bits**



Frame Reception with Errors

An unsuccessful frame reception might take the form of a runt frame or a frame with an incorrect FCS. 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 10: **Frame Reception with Errors – 32-bits**

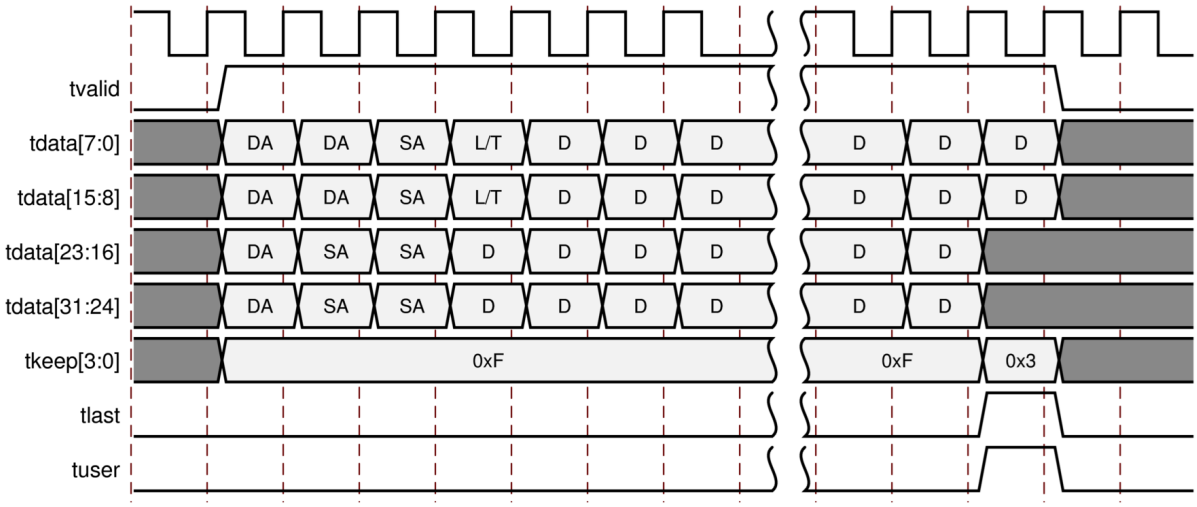
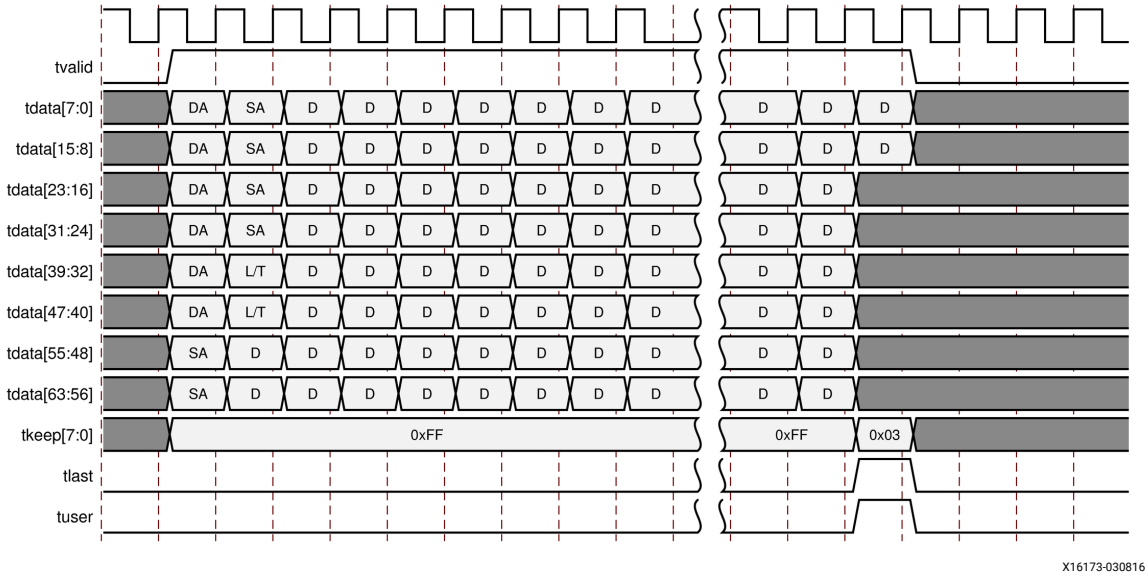


Figure 11: **Frame Reception with Errors – 64-bits**



AXI4-Stream Control and Status Ports

Table 12: **AXI4-Stream Interface–TX Path Control/Status Signals**

Name	I/O	Description	Clock Domain
ctl_tx_custom_preamble_enable[1]	I	When asserted, this signals enables the use of tx_preamblein as a custom preamble instead of inserting a standard preamble. ***Note*:** When the core is switched to 1G, this should always be 0.	tx_clk_out
tx_preamblein [55:0][1]	I	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. ***Note*:** When the core is switched to 1G, this should always be 0. This port is available for non-Versal device only.	tx_clk_out
ctl_tx_ipg_value[3:0][1]	I	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). ***Note*:** When the core is switched to 1G, this should always be 12.	tx_clk_out
ctl_tx_enable	I	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[2]	I	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[2]	I	Transmit Local Fault Indication (LFI) code word. Takes precedence over Remote Fault Indication (RFI).	tx_clk_out
ctl_tx_send_idle	I	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	I	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 12: **AXI4-Stream Interface–TX Path Control/Status Signals** *(cont'd)*

Name	I/O	Description	Clock Domain
ctl_tx_ignore_fcs	I	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
stat_tx_local_fault[2]	O	A value of 1 indicates the transmit decoder state machine is in the TX_INIT state. The output is level-sensitive.	tx_clk_out

Notes:

1. This signal is valid for 64-bit core configurations only.
2. This signal is not valid in 1G mode.

Table 13: **AXI4-Stream Interface–RX Path Control/Status Signals**

Name	I/O	Description	Clock Domain
rx_preambleout [55:0][2]	O	This is the preamble, and now a separate output instead of inline with data as was done with previous releases. ***Note*:** When the core is switched to 1G, this should always be 0. This port is available for non-Versal device only.	rx_core_clk
ctl_rx_enable	I	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[2]	I	When asserted, this input causes the MAC to check the preamble of the received frame.	rx_clk_out
ctl_rx_check_sfd[1]	I	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[1]	I	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
ctl_rx_delete_fcs	I	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

Table 13: **AXI4-Stream Interface–RX Path Control/Status Signals** *(cont'd)*

Name	I/O	Description	Clock Domain
ctl_rx_ignore_fcs	I	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]	I	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]	I	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. The ctl_rx_min_packet_len[7:0] value should be >=64B.	rx_clk_out
stat_rx_framing_err[1:0][1]	O	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[1]	O	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[1]	O	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	O	Indicates current status of the link.	rx_clk_out
stat_rx_block_lock[1]	O	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[1]	O	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]	O	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]	O	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

Table 13: **AXI4-Stream Interface–RX Path Control/Status Signals** *(cont'd)*

Name	I/O	Description	Clock Domain
stat_rx_truncated	O	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[1]	O	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[1]	O	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
stat_rx_hi_ber[1]	O	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[2]	I	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. ***Note*:** When the core is switched to 1G, this should always be 0.	rx_clk_out

Notes:

1. This signal is not valid in 1G mode.
2. This signal is valid for 64-bit core configurations only.

Miscellaneous Status/Control Signals

The following table shows the miscellaneous status and control I/O signals.

Table 14: **Miscellaneous Status/Control Ports**

Name	I/O	Description	Clock Domain
dclk	I	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.	See Clocking.
stat_rx_valid_ctrl_code[1]	O	Indicates that a PCS block with a valid control code was received.	rx_clk_out
ctl_local_loopback	I	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.	Async

Table 14: **Miscellaneous Status/Control Ports** *(cont'd)*

Name	I/O	Description	Clock Domain
stat_rx_got_signal_os[1]	O	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
ctl_rx_process_lfi[1]	I	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[1]	I	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[1]	I	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[1]	O	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. ***Note***: This signal will be present when 802cm Preemption is enabled or when the core type is MAC +PCS/PMA 32-bit type.	rx_clk_out
ctl_rx_data_pattern_select[1]	I	Corresponds to MDIO register bit 3.42.0 as defined in Clause 45.	rx_clk_out
ctl_rx_test_pattern_enable[1]	I	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[1]	I	Corresponds to MDIO register bit 3.42.0 as defined in Clause 45.	tx_clk_out
ctl_tx_test_pattern_enable[1]	I	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][1]	I	Corresponds to MDIO registers 3.34 through to 3.37 as defined in Clause 45.	tx_clk_out
ctl_tx_test_pattern_seed_b[57:0][1]	I	Corresponds to MDIO registers 3.38 through to 3.41 as defined in Clause 45.	tx_clk_out
ctl_tx_test_pattern_select[1]	I	Corresponds to MDIO register bit 3.42.1 as defined in Clause 45.	tx_clk_out
gig_ethernet_pcs_pma_status_vector_0[15:0][1]	O	See the Status Vector Table in *1G/2.5G Ethernet PCS/PMA or SGMII LogiCORE IP Product Guide* (PG047).	

Table 14: **Miscellaneous Status/Control Ports** *(cont'd)*

Name	I/O	Description	Clock Domain
signal_detect	I	This port can be connected to an optical module to detect the presence of light. Logic 1 indicates that the optical module is correctly detecting light; logic 0 indicates a fault. Ensure, therefore, that this is driven with the correct polarity. If not connected to an optical module, the signal must be tied to logic 1. ***Note*:** When signal_detect is set to logic 0, this forces the receiver synchronization state machine of the core to remain in the loss of sync state.	N/A

Notes:

1. This signal is not valid in 1G mode.

Related Information

Clocking

Statistics Interface Ports

The following tables show the Statistics interface I/O ports.

Table 15: **Statistics Interface - RX Path**

Name	I/O	Description	Clock Domain
stat_rx_total_bytes[3:0]	O	Increment for the total number of bytes received.	rx_clk_out
stat_rx_total_packets[1:0]	O	Increment for the total number of packets received.	rx_clk_out
stat_rx_total_good_bytes[13:0]	O	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	O	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	O	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	O	Increment for good and bad packets received that contain 64 bytes.	rx_clk_out
stat_rx_packet_65_127_bytes	O	Increment for good and bad packets received that contain 65 to 127 bytes.	rx_clk_out
stat_rx_packet_128_255_bytes	O	Increment for good and bad packets received that contain 128 to 255 bytes.	rx_clk_out
stat_rx_packet_256_511_bytes	O	Increment for good and bad packets received that contain 256 to 511 bytes.	rx_clk_out
stat_rx_packet_512_1023_bytes	O	Increment for good and bad packets received that contain 512 to 1,023 bytes.	rx_clk_out

Table 15: **Statistics Interface - RX Path** *(cont'd)*

Name	I/O	Description	Clock Domain
stat_rx_packet_1024_1518_bytes	O	Increment for good and bad packets received that contain 1,024 to 1,518 bytes.	rx_clk_out
stat_rx_packet_1519_1522_bytes	O	Increment for good and bad packets received that contain 1519 to 1522 bytes.	rx_clk_out
stat_rx_packet_1523_1548_bytes	O	Increment for good and bad packets received that contain 1,523 to 1,548 bytes.	rx_clk_out
stat_rx_packet_1549_2047_bytes	O	Increment for good and bad packets received that contain 1,549 to 2,047 bytes.	rx_clk_out
stat_rx_packet_2048_4095_bytes	O	Increment for good and bad packets received that contain 2,048 to 4,095 bytes.	rx_clk_out
stat_rx_packet_4096_8191_bytes	O	Increment for good and bad packets received that contain 4,096 to 8,191 bytes.	rx_clk_out
stat_rx_packet_8192_9215_bytes	O	Increment for good and bad packets received that contain 8,192 to 9,215 bytes.	rx_clk_out
stat_rx_packet_small	O	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	O	Increment for all packets that are more than 9,215 bytes long.	rx_clk_out
stat_rx_oversize	O	Increment for packets longer than ctl_rx_max_packet_len with good FCS.	rx_clk_out
stat_rx_toolong	O	Increment for packets longer than ctl_rx_max_packet_len with good and bad FCS.	rx_clk_out
stat_rx_undersize	O	Increment for packets shorter than ctl_rx_min_packet_len with good FCS.	rx_clk_out
stat_rx_fragment	O	Increment for packets shorter than ctl_rx_min_packet_len with bad FCS.	rx_clk_out
stat_rx_jabber	O	Increment for packets longer than ctl_rx_max_packet_len with bad FCS.	rx_clk_out
stat_rx_bad_code[1]	O	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[1]	O	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.	rx_clk_out
stat_rx_bad_preamble[1]	O	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.	rx_clk_out

Notes:

1. This signal is not valid in 1G mode.

Table 16: **Statistics Interface - TX Path**

Name	I/O	Description	Clock Domain
stat_tx_total_bytes[3:0]	O	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	O	Increment for the total number of packets transmitted.	tx_clk_out
stat_tx_total_good_bytes[13:0]	O	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	O	Increment for the total number of good packets transmitted.	tx_clk_out
stat_tx_bad_fcs	O	Increment for packets greater than 64 bytes that have FCS errors.	tx_clk_out
stat_tx_packet_64_bytes	O	Increment for good and bad packets transmitted that contain 64 bytes.	tx_clk_out
stat_tx_packet_65_127_bytes	O	Increment for good and bad packets transmitted that contain 65 to 127 bytes.	tx_clk_out
stat_tx_packet_128_255_bytes	O	Increment for good and bad packets transmitted that contain 128 to 255 bytes.	tx_clk_out
stat_tx_packet_256_511_bytes	O	Increment for good and bad packets transmitted that contain 256 to 511 bytes.	tx_clk_out
stat_tx_packet_512_1023_bytes	O	Increment for good and bad packets transmitted that contain 512 to 1,023 bytes.	tx_clk_out
stat_tx_packet_1024_1518_bytes	O	Increment for good and bad packets transmitted that contain 1,024 to 1,518 bytes.	tx_clk_out
stat_tx_packet_1519_1522_bytes	O	Increment for good and bad packets transmitted that contain 1,519 to 1,522 bytes.	tx_clk_out
stat_tx_packet_1523_1548_bytes	O	Increment for good and bad packets transmitted that contain 1,523 to 1,548 bytes.	tx_clk_out
stat_tx_packet_1549_2047_bytes	O	Increment for good and bad packets transmitted that contain 1,549 to 2,047 bytes.	tx_clk_out
stat_tx_packet_2048_4095_bytes	O	Increment for good and bad packets transmitted that contain 2,048 to 4,095 bytes.	tx_clk_out
stat_tx_packet_4096_8191_bytes	O	Increment for good and bad packets transmitted that contain 4,096 to 8,191 bytes.	tx_clk_out
stat_tx_packet_8192_9215_bytes	O	Increment for good and bad packets transmitted that contain 8,192 to 9,215 bytes.	tx_clk_out
stat_tx_packet_small	O	Increment for all packets that are less than 64 bytes long.	tx_clk_out
stat_tx_packet_large	O	Increment for all packets that are more than 9,215 bytes long.	tx_clk_out
stat_tx_frame_error	O	Increment for packets with tx_axis_tuser set to indicate an End of Packet (EOP) abort or frames aborted by de-asserting tvalid without tlast.	tx_clk_out

XGMII/GMII Interface Ports

The following table shows the XGMII/GMII Interface ports. These ports are available for the Ethernet PCS/PMA 32-bit core configuration only.

Table 17: **XGMII/GMII Interface Ports**

Name	I/O	Description	Clock Domain
rx_mii_d[31:0]	O	Receive XGMII Data bus.	rx_mii_clk
rx_mii_c[3:0]	O	Receive XGMII Control bus.	rx_mii_clk
rx_mii_clk	I	Receive XGMII Clock input.	See Clocking for more information.
tx_mii_d[31:0]	I	Transmit XGMII Data bus.	rx_mii_clk
tx_mii_c[3:0]	I	Transmit XGMII Control bus.	rx_mii_clk
tx_mii_clk	I	Transmit XGMII Clock input.	See Clocking for more information.
gmii_rxd[7:0]	O	Receive GMII Data bus.	rx_core_clk
gmii_rx_dv	O	Receive GMII Control signal.	rx_core_clk
gmii_rx_er	O	Receive GMII error signal.	rx_core_clk
gmii_txd[7:0]	O	Transmit GMII Data bus.	tx_out_clk
gmii_tx_en	O	Transmit GMII enable signal.	tx_out_clk
gmii_rx_er	O	Transmit GMII error signal.	tx_out_clk

Related Information

Clocking

XGMII Interfaces

Internal 32-bit SDR Client-Side Interface

The mapping of lanes to data bits is shown in the following table. The lane number is also the index of the control bit for that particular lane; for example, `tx_mii_c[2]` and `tx_mii_d[23:16]` are the control and data bits respectively for lane 2.

Table 18: **Lanes for Internal 32-Bit Client-Side Interface**

Lane	tx_mii_d, rx_mii_d Bits
0	7:0
1	15:8
2	23:16
3	31:24

Definitions of Control Characters

Reference is regularly made to certain XGMII control characters signifying Start, Terminate, Error, and others. These control characters all have in common that the control line for that lane is 1 for the character and a certain data byte value. The relevant characters are defined in the IEEE Std. 802.3 and are reproduced in the following table for reference.

Table 19: **Partial List of XGMII Characters**

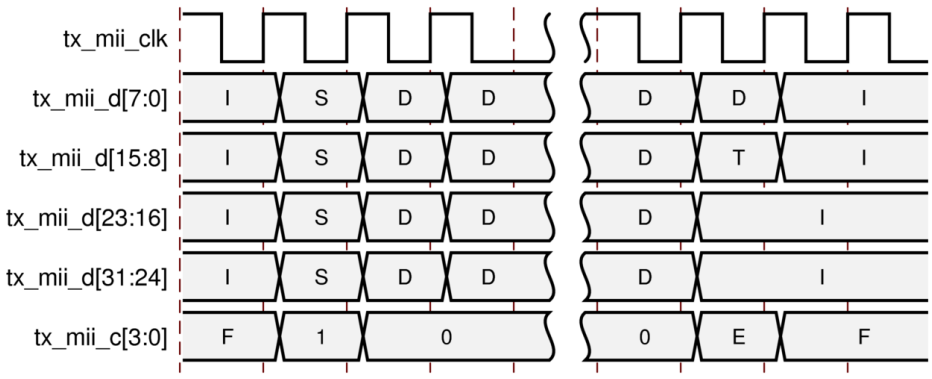
Data (Hex)	Control	Name, Abbreviation
00 to FF	0	Data (D)
07	1	Idle (I)
FB	1	Start (S)
FD	1	Terminate (T)
FE	1	Error (E)

Interfacing to the Transmit Client Interface

The timing of a data frame transmission through the internal 32-bit client-side interface is shown in the following figure. The beginning of the data frame is shown by the presence of the Start character (the /S/ codegroup in lane 0 in the following figure) followed by data characters in lanes 1, 2, and 3.

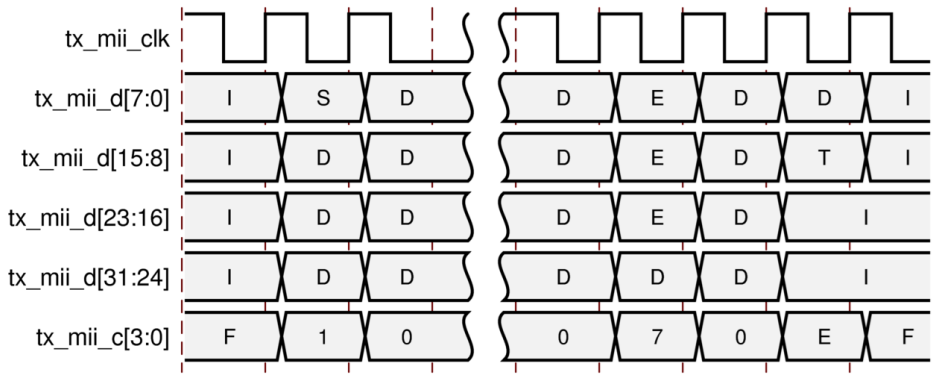
When the frame is complete, it is completed by a Terminate character (the T in lane 1 in the following figure). The Terminate character can occur in any lane; the remaining lanes are padded by the XGMII idle characters.

Figure 12: **Normal Frame Transmission Across the Internal 32-Bit Client Interface**



The following figure shows a similar frame to that in the preceding figure, with the exception that this frame is propagating an error. The error code is denoted by the letter E, with the relevant control bits set.

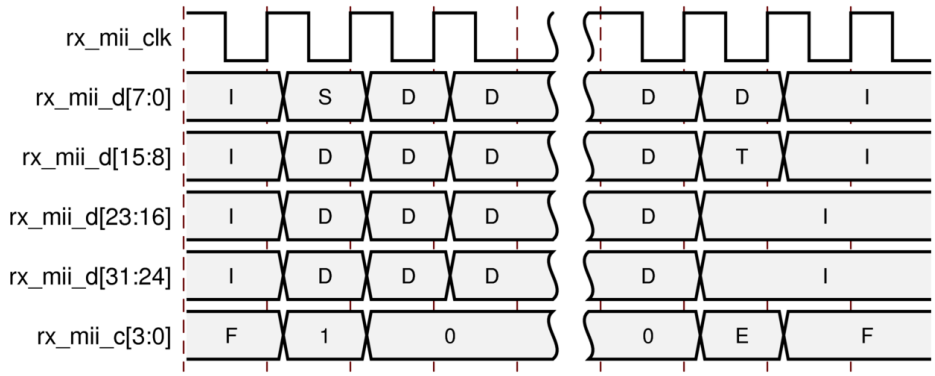
Figure 13: **Frame Transmission with Error Across the Internal 32-bit Client Interface**



Interfacing the Receive Client Interface

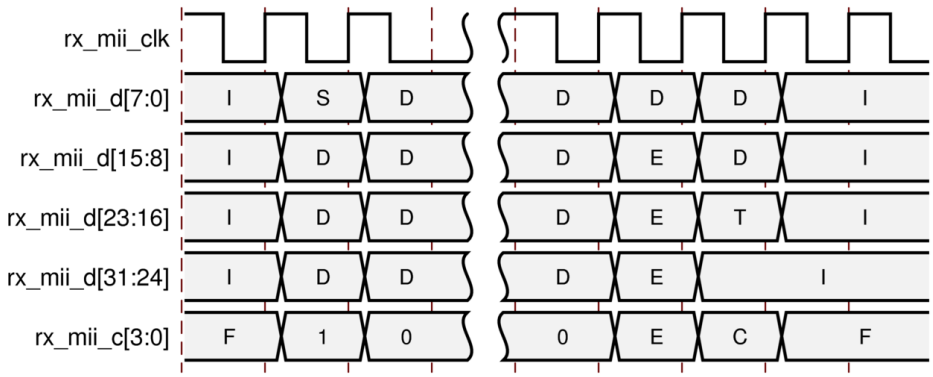
The timing of a normal inbound frame transfer is shown in the following figure. As in the transmit case, the frame is delimited by a Start character (S) and by a Terminate character (T). The Start character in this implementation can occur only on lane 0. The Terminate character, T, can occur in any lane.

Figure 14: **Frame Reception Across the Internal 32-Bit Client Interface**



The following figure shows an inbound frame of data propagating an error. In this instance, the error is propagated in lanes 1 to 3, shown by the letter E.

Figure 15: **Frame Reception with Error Across the Internal 32-Bit Client Interface**



Using the Client-Side GMII Interface

It is not within the scope of this document to define the Gigabit Media Independent Interface (GMII)— see Clause 35 of the IEEE 802.3 specification for information about the GMII. Timing diagrams and descriptions are provided only as an informational guide.

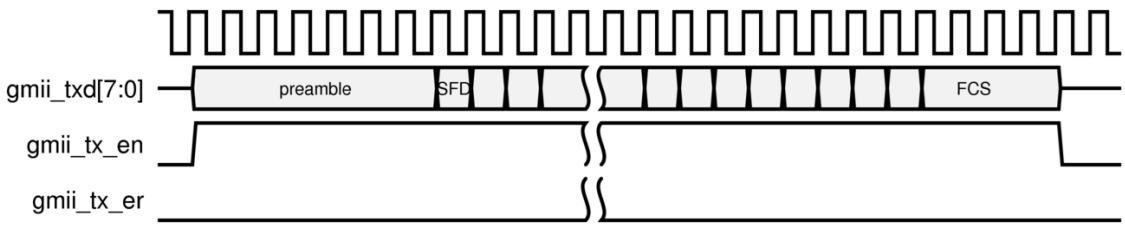
GMI Transmission

This section includes figures that illustrate GMII transmission. In these figures the clock is not labeled. The source of this clock signal varies, depending on the options selected when the core is generated.

Normal Frame Transmission

Normal outbound frame transfer timing is shown in the following figure. This figure shows that an Ethernet frame is proceeded by an 8-byte preamble field (inclusive of the Start of Frame Delimiter (SFD)), and completed with a 4-byte Frame Check Sequence (FCS) field. This frame is created by the MAC connected to the other end of the GMII. The PCS logic itself does not recognize the different fields within a frame and treats any value placed on `gmii_txd[7:0]` within the `gmii_tx_en` assertion window as data.

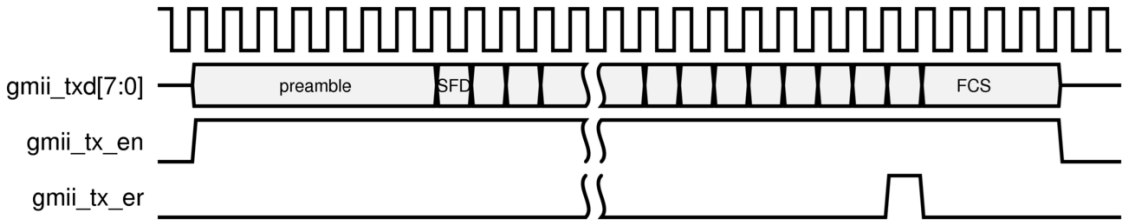
Figure 16: **GMII Normal Frame Transmission**



Error Propagation

A corrupted frame transfer is shown in the following figure. An error can be injected into the frame by asserting `gmii_tx_er` at any point during the `gmii_tx_en` assertion window. The core ensures that all errors are propagated through both transmit and receive paths so that the error is eventually detected by the link partner.

Figure 17: **GMII Error Propagation Within a Frame**



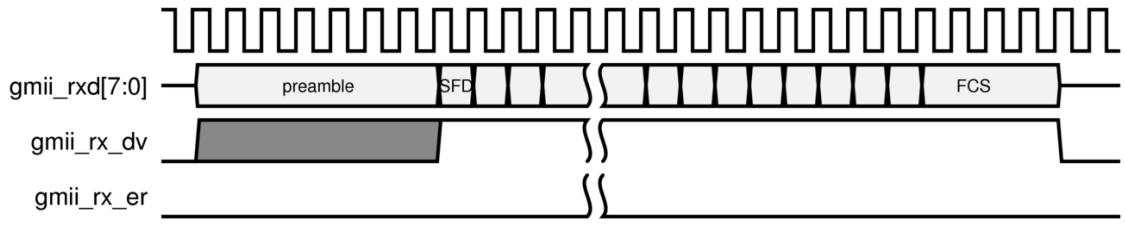
GMII Reception

This section includes figures that illustrate GMII reception. In these figures the clock is not labeled. The source of this clock signal varies, depending on the options used when the core is generated.

Normal Frame Reception

The timing of normal inbound frame transfer is shown in the following figure. This shows that Ethernet frame reception is preceded by a preamble field. The IEEE 802.3 specification (see Clause 35) allows for up to all of the seven preamble bytes that proceed the Start of Frame Delimiter (SFD) to be lost in the network. The SFD is always present in well-formed frames.

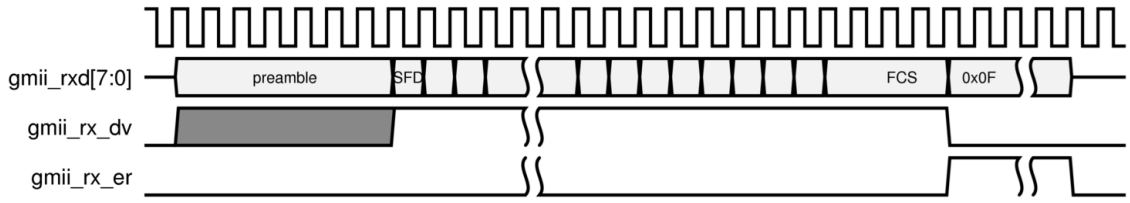
Figure 18: **GMII Normal Frame Reception**



Normal Frame Reception with Extension Field

In accordance with the IEEE 802.3 specification, state machines for the 1000BASE-XPCS, `gmii_rx_er` can be driven High following reception of the end frame in conjunction with `gmii_rxd[7:0]` containing the hexadecimal value of `0x0F` to signal carrier extension. This is shown in the following figure. This is not an error condition and can occur even for full-duplex frames.

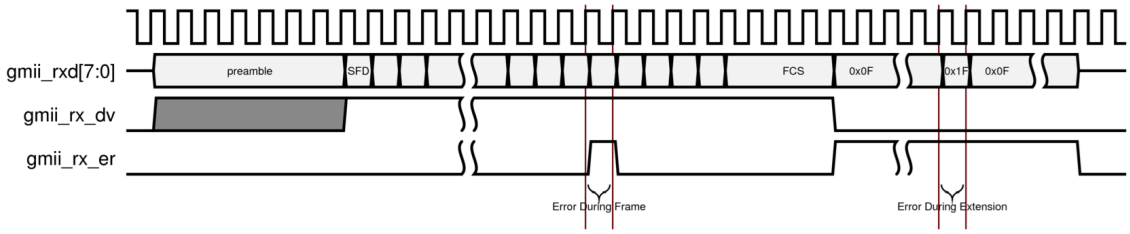
Figure 19: **GMII Normal Frame Reception**



Frame Reception with Errors

The signal `gmii_rx_er` when asserted within the assertion window signals that a frame was received with a detected error (see the following figure). In addition, a late error can also be detected during the Carrier Extension interval. This is indicated by `gmii_rxd[7:0]` containing the hexadecimal value `0x1F`, also shown in the following figure.

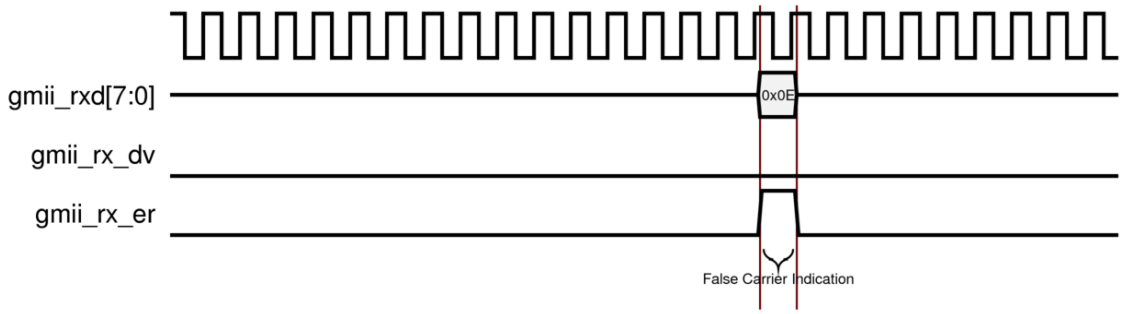
Figure 20: **GMII Frame Reception with Errors**



False Carrier

The following figure shows the GMII signaling for a False Carrier condition. False Carrier is asserted by the core in response to certain error conditions, such as a frame with a corrupted start code, or for random noise.

Figure 21: **False Carrier Indication**



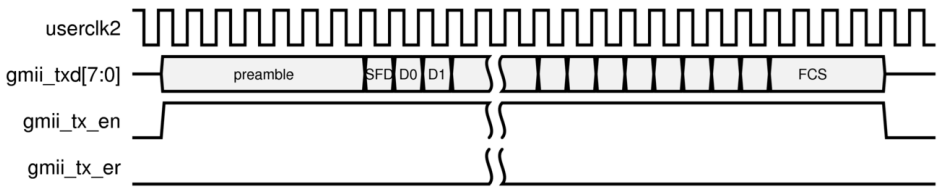
Using the Client-Side GMII for the SGMII Standard

GMII Transmission

1/2.5 Gb/s Frame Transmission

The timing of normal outbound frame transfer is shown in the following figure.

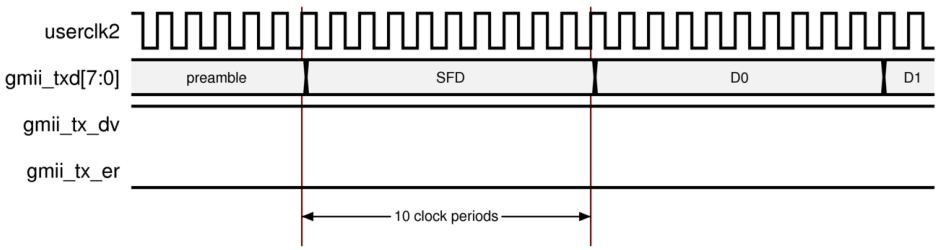
Figure 22: **GMII Frame Transmission at 1 Gb/s**



100 Mb/s Frame Transmission

At 100 Mb/s the operation of the core remains unchanged. It is the responsibility of the client logic (for example, an Ethernet MAC) to enter data at the correct rate. When operating at 100 Mb/s, every byte of the MAC frame (from preamble to the Frame Check Sequence field, inclusive) should be repeated for 10 clock periods to achieve the desired bit rate, as shown in the following figure. The core always expects eight bits from the client logic.

Figure 23: **GMII Data Transmission at 100 Mb/s**



10 Mb/s Frame Transmission

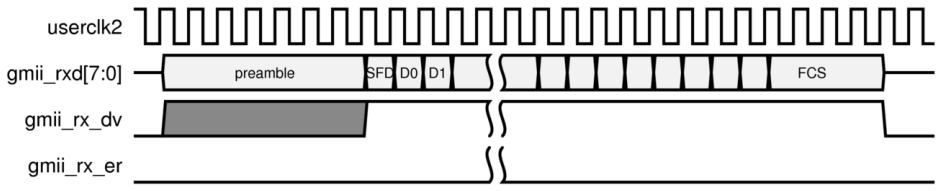
At 10 Mb/s the operation of the core remains unchanged. It is the responsibility of the client logic (for example, an Ethernet MAC), to enter data at the correct rate. When operating at 10 Mb/s, every byte of the MAC frame (from preamble to the frame check sequence field, inclusive) should be repeated for 100 clock periods to achieve the desired bit rate. It is also the responsibility of the client logic to ensure that the interframe gap period is legal for the current speed of operation. The core always expects eight bits from the client logic.

GMII Reception

1/2.5 Gb/s Frame Reception

The timing of a normal inbound frame transfer is shown in the following figure.

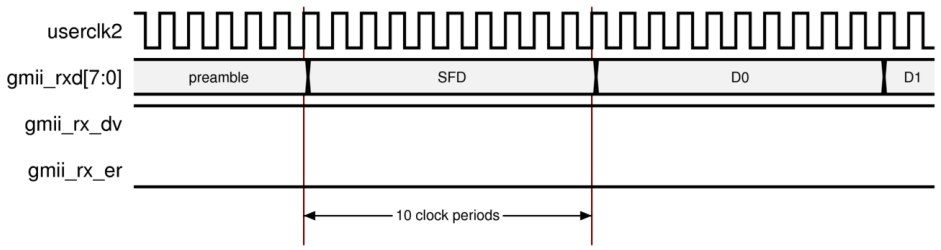
Figure 24: **GMII Frame Reception at 1 Gb/s**



100 Mb/s Frame Reception

At 100 Mb/s the operation of the core is unchanged. When operating at a speed of 100 Mb/s, every byte of the MAC frame (from preamble to the frame check sequence field, inclusive) is repeated for 10 clock periods to achieve the desired bit rate. See the following figure.

Figure 25: **GMII Data Reception at 100 Mb/s**



10 Mb/s Frame Reception

The operation of the core remains unchanged. When operating at a speed of 10 Mb/s, every byte of the MAC frame (from preamble to the frame check sequence field, inclusive) is repeated for 100 clock periods to achieve the desired bit rate.

Pause Interface

The following tables show the Pause interface I/O ports.

Table 20: **Pause Interface - Control Ports**

Name	I/O	Description	Clock Domain
ctl_rx_pause_enable[8:0]	I	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]	I	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 21: **Pause Interface - RX Path**

Name	I/O	Description	Clock Domain
ctl_rx_enable_gcp	I	A value of 1 enables global control packet processing.	rx_clk_out
ctl_rx_check_mcast_gcp	I	A value of 1 enables global control multicast destination address processing.	rx_clk_out
ctl_rx_check_ucast_gcp	I	A value of 1 enables global control unicast destination address processing.	rx_clk_out
ctl_rx_pause_da_ucast[47:0]	I	Unicast destination address for pause processing.	rx_clk_out
ctl_rx_check_sa_gcp	I	A value of 1 enables global control source address processing.	rx_clk_out
ctl_rx_pause_sa[47:0]	I	Source address for pause processing.	rx_clk_out

Table 21: **Pause Interface - RX Path** *(cont'd)*

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

Table 21: **Pause Interface - RX Path** *(cont'd)*

Name	I/O	Description	Clock Domain
ctl_rx_check_etype_ppp	I	A value of 1 enables priority pause ethertype processing.	rx_clk_out
ctl_rx_etype_ppp[15:0]	I	Ethertype field for priority pause processing.	rx_clk_out
ctl_rx_check_opcode_ppp	I	A value of 1 enables priority pause opcode processing.	rx_clk_out
ctl_rx_opcode_ppp[15:0]	I	Priority pause opcode value.	rx_clk_out
stat_rx_pause_req[8:0]	O	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]	I	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	I	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	I	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]	O	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]	O	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 22: **Pause Interface–TX Path**

Name	I/O	Description	Clock Domain
ctl_tx_pause_req[8:0]	I	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]	I	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

Table 22: **Pause Interface–TX Path** *(cont'd)*

Name	I/O	Description	Clock Domain
ctl_tx_pause_refresh_timer [8:0][15:0]	I	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]	I	Destination address for transmitting global pause packets.	tx_clk_out
ctl_tx_sa_gpp[47:0]	I	Source address for transmitting global pause packets.	tx_clk_out
ctl_tx_ethertype_gpp[15:0]	I	Ethertype for transmitting global pause packets.	tx_clk_out
ctl_tx_opcode_gpp[15:0]	I	Opcode for transmitting global pause packets.	tx_clk_out
ctl_tx_da_ppp[47:0]	I	Destination address for transmitting priority pause packets.	tx_clk_out
ctl_tx_sa_ppp[47:0]	I	Source address for transmitting priority pause packets.	tx_clk_out
ctl_tx_ethertype_ppp[15:0]	I	Ethertype for transmitting priority pause packets.	tx_clk_out
ctl_tx_opcode_ppp[15:0]	I	Opcode for transmitting priority pause packets.	tx_clk_out
ctl_tx_resend_pause	I	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]	O	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

The following table shows the additional ports used for Auto-Negotiation. These signals are found at the `*wrapper.v` hierarchy file.

Table 23: **Additional Ports for Auto-Negotiation**

Port Name	I/O	Description	Clock Domain
an_clk	I	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. ***Note*:** This port is not accessible to you. This is tied to dclk, which is a free-running clock.	Refer to Clocking.
an_reset	I	Asynchronous active-High reset.	Async
ctl_autoneg_enable	I	Enable signal for auto-negotiation.	an_clk

Table 23: **Additional Ports for Auto-Negotiation** *(cont'd)*

Port Name	I/O	Description	Clock Domain
ctl_autoneg_bypass	I	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]	I	8-bit seed to initialize the nonce field Polynomial generator. 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	I	Selects the polynomial generator for the bit 49 random BitGenerator. 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
ctl_restart_negotiation	I	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	I	This input signal is used to set the remote_fault bit of the transmit link codeword.	an_clk
Signals Used for Pause Ability Advertising			
ctl_an_pause	I	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	I	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	I	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	I		an_clk
ctl_an_ability_100gbase_cr4	I		an_clk
ctl_an_ability_100gbase_kp4	I		an_clk
ctl_an_ability_100gbase_kr4	I		an_clk
ctl_an_ability_10gbase_kr	I		an_clk
ctl_an_ability_10gbase_kx4	I		an_clk
ctl_an_ability_25gbase_cr	I		an_clk
ctl_an_ability_25gbase_cr1	I		an_clk
ctl_an_ability_25gbase_kr	I		an_clk
ctl_an_ability_25gbase_kr1	I		an_clk
ctl_an_ability_40gbase_cr4	I		an_clk
ctl_an_ability_40gbase_kr4	I		an_clk
ctl_an_ability_50gbase_cr2	I		an_clk
ctl_an_ability_50gbase_kr2	I		an_clk
ctl_an_fec_request	I	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 23: **Additional Ports for Auto-Negotiation** *(cont'd)*

Port Name	I/O	Description	Clock Domain
ctl_an_fec_ability_override	I	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	I	This bit is used to indicate clause 91 FEC ability.	an_clk
ctl_an_cl91_fec_request	I	This bit is used to request clause 91 FEC.	an_clk
stat_an_link_cntl_1000base_kx[1:0]	O	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]	O		an_clk
stat_an_link_cntl_100gbase_cr4[1:0]	O		an_clk
stat_an_link_cntl_100gbase_kp4[1:0]	O		an_clk
stat_an_link_cntl_100gbase_kr4[1:0]	O		an_clk
stat_an_link_cntl_10gbase_kr[1:0]	O		an_clk
stat_an_link_cntl_10gbase_kx4[1:0]	O		an_clk
stat_an_link_cntl_25gbase_cr[1:0]	O		an_clk
stat_an_link_cntl_25gbase_cr1[1:0]	O		an_clk
stat_an_link_cntl_25gbase_kr[1:0]	O		an_clk
stat_an_link_cntl_25gbase_kr1[1:0]	O		an_clk
stat_an_link_cntl_40gbase_cr4[1:0]	O		an_clk
stat_an_link_cntl_40gbase_kr4[1:0]	O		an_clk
stat_an_link_cntl_50gbase_cr2[1:0]	O		an_clk
stat_an_link_cntl_50gbase_kr2[1:0]	O		an_clk
stat_an_fec_enable	O	Used to enable the use of clause 74 FEC on the link.	an_clk
stat_an_rs_fec_enable	O	Used to enable the use of clause 91 FEC on the link.	an_clk
stat_an_tx_pause_enable	O	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	O	Used to enable station-to-station (global) pause packet interpretation in the receive path, to control data flow from the transmitter.	an_clk
stat_an_autoneg_complete	O	Indicates the auto-negotiation is complete and RX link status from the PCS has been received.	an_clk
stat_an_parallel_detection_fault	O	Indicated a parallel detection fault during auto-negotiation.	an_clk

Table 23: **Additional Ports for Auto-Negotiation** *(cont'd)*

Port Name	I/O	Description	Clock Domain
stat_an_lp_ability_1000base_kx	O	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	O		an_clk
stat_an_lp_ability_100gbase_cr4	O		an_clk
stat_an_lp_ability_100gbase_kp4	O		an_clk
stat_an_lp_ability_100gbase_kr4	O		an_clk
stat_an_lp_ability_10gbase_kr	O		an_clk
stat_an_lp_ability_10gbase_kx4	O		an_clk
stat_an_lp_ability_25gbase_cr	O		an_clk
stat_an_lp_ability_25gbase_kr	O		an_clk
stat_an_lp_ability_40gbase_cr4	O		an_clk
stat_an_lp_ability_40gbase_kr4	O		an_clk
stat_an_lp_ability_25gbase_cr1	O	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	O	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	O	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_kr2	O	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	O	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	O	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	O	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	O	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	O	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

Table 23: **Additional Ports for Auto-Negotiation** *(cont'd)*

Port Name	I/O	Description	Clock Domain
stat_an_lp_ability_valid	O	This signal indicates when all of the link partner advertisements become valid.	an_clk
an_loc_np_data[47:0]	I	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]	O	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	I	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	I	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
stat_an_loc_np_ack	O	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	O	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]	O	This output indicates the extended FEC abilities as defined in Schedule 3.	an_clk
stat_an_lp_extended_ability_valid	O	When this bit is 1, it indicates that the detected extended abilities are valid.	an_clk
stat_an_lp_rf	O	This bit indicates link partner remote fault.	an_clk

Table 23: **Additional Ports for Auto-Negotiation** *(cont'd)*

Port Name	I/O	Description	Clock Domain
stat_an_start_tx_disable	O	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	O	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

The following table shows the Link Training ports.

Table 24: **Link Training Ports**

Port Name	I/O	Description	Clock Domain
ctl_lt_training_enable	I	Enables link training. When link training is disabled, all PCS lanes function in mission mode.	tx_serdes_clk
ctl_lt_restart_training	I	This signal triggers a restart of link training regardless of the current state.	tx_serdes_clk
ctl_lt_rx_trained	I	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
stat_lt_signal_detect	O	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	O	This signal indicates when the respective link training state machine is performing link training.	tx_serdes_clk
stat_lt_training_fail	O	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	O	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	O	This signal reflects the value of the preset control bit received in the control block from the link partner.	rx_serdes_clk

Table 24: **Link Training Ports** *(cont'd)*

Port Name	I/O	Description	Clock Domain
stat_lt_initialize_from_rx	O	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]	O	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]	O	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]	O	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]	O	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]	O	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
stat_lt_stat_m1_from_rx0[1:0]	O	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]	I	This 11-bit signal seeds the training pattern generator.	tx_serdes_clk
ctl_lt_preset_to_tx	I	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	I	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]	I	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]	I	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]	I	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]	I	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]	I	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]	I	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]	O	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

The following table shows the IEEE 802.3 Clause 74 FEC Control/Status and Statistics signals.

Table 25: **IEEE 802.3 Clause 74 FEC Interface Control/Status/Statistics Signals**

Signal	I/O	Clock	Description
ctl_fec_tx_enable	I	tx_serdes_clk	Asserted to enable the clause 74 FEC encoding on the transmitted data
ctl_fec_rx_enable	I	rx_serdes_clk	Asserted to enable the clause 74 FEC decoding of the received data
ctl_fec_enable_error_to_pcs	I	rx_serdes_clk	Clause 74 FEC enable error to PCS
stat_fec_inc_correct_count[3:0]	O	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]	O	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]	O	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]	O	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

Ports under this section are available when IEEE Clause 108 (RS-FEC) is selected from Configuration tab.

Table 26: **IEEE Clause 108 (RS-FEC) Control/Status/Statistics Signals**

Name	Size	I/O	Description
ctl_rx_rsfec_enable_correction_*	1	I	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	I	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.

Table 26: **IEEE Clause 108 (RS-FEC) Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
ctl_rsfec_enable_*	1	I	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 1G/10G/25G Subsystem can have individual RX and TX enable signals.
ctl_rsfec_ieee_error_indication_mode_*	1	I	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.
ctl_rsfec_consortium_25g_*	1	I	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 1G/10G/25G Subsystem can have individual RX and TX consortium signals.
stat_rx_rsfec_hi_ser_*	1	O	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	O	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	O	Increment for corrected errors.
stat_rx_rsfec_uncorrected_cw_inc_*	1	O	Increment for uncorrected errors.
stat_rx_rsfec_err_count0_inc_*	3	O	Increment for detected errors.
stat_tx_rsfec_lane_alignment_status_*	1	O	A value of 1 indicates that the TX RS-FEC block has achieved alignment on the incoming PCS data.

Register Space

The 1G/10G/25G Ethernet Subsystem is configured with AXI4-Lite registers to access the configuration and status signals.

For more information, see *1G/2.5G Ethernet PCS/PMA or SGMII LogiCORE IP Product Guide* (PG047) and *10G/25G High Speed Ethernet Subsystem Product Guide* (PG210). AXI Crossbar module is initiated within the IP to access the AXI4-Lite interface control and statistics registers for Gigabit Ethernet PCS-PMA and 10G MAC+PCS subsystem. This module is provided with a single user AXI4-Lite interface.

The AXI Crossbar IP is configured with 1-master and 2-slave interfaces. Both Gigabit Ethernet PCS-PMA and 10G/25G MAC+PCS subsystem control and status registers can be accessed with AXI4-Lite interface through AXI Crossbar. Refer to the *AXI Interconnect LogiCORE IP Product Guide* (PG059) for AXI Crossbar soft IP functionality.

The following are the configured base address locations for Gigabit Ethernet PCS-PMA and 10G/25G MAC+PCS subsystem control and status registers in the AXI Crossbar soft IP:

- **0x0000_0000 to 0x0000_0FFF:** Address locations for 10G/25G MAC+PCS subsystem
- **0x0000_1000 to 0x0000_1FFF:** Address locations for Gigabit Ethernet PCS-PMA

AXI4-Lite Ports

The following table describes the port list for the AXI processor interface.

Table 27: **AXI Ports**

Signal	I/O	Description
s_axi_aclk	I	AXI4-Lite clock. Range between 10 MHz and 300 MHz
s_axi_aresetn	I	Asynchronous active-Low reset
s_axi_awaddr[31:0]	I	Write address Bus
s_axi_awvalid	I	Write address valid
s_axi_awready	O	Write address acknowledge
s_axi_wdata[31:0]	I	Write data bus
s_axi_wstrb[3:0]	I	Strobe signal for the data bus byte lane
s_axi_wvalid	O	Write data valid
s_axi_wready	O	Write data acknowledge
s_axi_bresp[1:0]	O	Write transaction response
s_axi_bvalid	O	Write response valid
s_axi_bready	I	Write response acknowledge
s_axi_araddr[31:0]	I	Read address bus
s_axi_arvalid	I	Read address valid
s_axi_arready	O	Read address acknowledge
s_axi_rdata[31:0]	O	Read data output
s_axi_rresp[1:0]	O	Read data response
s_axi_rvalid	O	Read data/response valid
s_axi_rready	I	Read data acknowledge
pm_tick	I	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 Arm AMBA AXI Protocol v2.0 Specification (Arm IHI 0022C).

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 and Status Register Map

Configuration Register Map 1G/10G/25G Ethernet Subsystem

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.

Table 28: **Configuration Register Map**

Hex Address	Register Name	Notes
0x0000[1]	GT_RESET_REG: 0000	
0x0004[1]	RESET_REG: 0004	
0x0008[1]	MODE_REG: 0008	
0x000C	CONFIGURATION_TX_REG1: 000C	
0x0014[1]	CONFIGURATION_RX_REG1: 0014	
0x0018[1]	CONFIGURATION_RX_MTU: 0018	Only in MAC+PCS variant and MAC-only variants
0x001C	CONFIGURATION_VL_LENGTH_REG: 001C	
0x0020[1]	TICK_REG: 0020	
0x0024[1]	CONFIGURATION_REVISION_REG: 0024	
0x0028[1]	CONFIGURATION_TX_TEST_PAT_SEED_A_LSB: 0028	Only in MAC+PCS and PCS-only variants
0x002C[1]	CONFIGURATION_TX_TEST_PAT_SEED_A_MSB: 002C	Only in MAC+PCS and PCS-only variants
0x0030[1]	CONFIGURATION_TX_TEST_PAT_SEED_B_LSB: 0030	Only in MAC+PCS and PCS-only variants
0x0034[1]	CONFIGURATION_TX_TEST_PAT_SEED_B_MSB: 0034	Only in MAC+PCS and PCS-only variants
0x0038[1]	CONFIGURATION_1588_REG: 0038	Only in MAC+PCS/PMA 64-bit variant when time stamping is enabled.
0x0040[1]	CONFIGURATION_TX_FLOW_CONTROL_REG1: 0040	Only in MAC+PCS and MAC-only variants
0x0044[1]	CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG1: 0044	Only in MAC+PCS and MAC-only variants
0x0048[1]	CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG2: 0048	Only in MAC+PCS and MAC-only variants
0x004C[1]	CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG3: 004C	Only in MAC+PCS and MAC-only variants
0x0050[1]	CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG4: 0050	Only in MAC+PCS and MAC-only variants
0x0054[1]	CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG5: 0054	Only in MAC+PCS and MAC-only variants
0x0058[1]	CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG1: 0058	Only in MAC+PCS and MAC-only variants
0x005C[1]	CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG2: 005C	Only in MAC+PCS and MAC-only variants
0x0060[1]	CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG3: 0060	Only in MAC+PCS and MAC-only variants
0x0064[1]	CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG4: 0064	Only in MAC+PCS and MAC-only variants

Table 28: **Configuration Register Map** *(cont'd)*

Hex Address	Register Name	Notes
0x0068[1]	CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG5: 0068	Only in MAC+PCS and MAC-only variants
0x006C[1]	CONFIGURATION_TX_FLOW_CONTROL_PPP_ETYPE_OP_REG: 006C	Only in MAC+PCS and MAC-only variants
0x0070[1]	CONFIGURATION_TX_FLOW_CONTROL_GPP_ETYPE_OP_REG: 0070	Only in MAC+PCS and MAC-only variants
0x0074[1]	CONFIGURATION_TX_FLOW_CONTROL_GPP_DA_REG_LSB: 0074	Only in MAC+PCS and MAC-only variants
0x0078[1]	CONFIGURATION_TX_FLOW_CONTROL_GPP_DA_REG_MSB: 0078	Only in MAC+PCS and MAC-only variants
0x007C[1]	CONFIGURATION_TX_FLOW_CONTROL_GPP_SA_REG_LSB: 007C	Only in MAC+PCS and MAC-only variants
0x0080[1]	CONFIGURATION_TX_FLOW_CONTROL_GPP_SA_REG_MSB: 0080	Only in MAC+PCS and MAC-only variants
0x0084[1]	CONFIGURATION_TX_FLOW_CONTROL_PPP_DA_REG_LSB: 0084	Only in MAC+PCS and MAC-only variants
0x0088[1]	CONFIGURATION_TX_FLOW_CONTROL_PPP_DA_REG_MSB: 0088	Only in MAC+PCS and MAC-only variants
0x008C[1]	CONFIGURATION_TX_FLOW_CONTROL_PPP_SA_REG_LSB: 008C	Only in MAC+PCS and MAC-only variants
0x0090[1]	CONFIGURATION_TX_FLOW_CONTROL_PPP_SA_REG_MSB: 0090	Only in MAC+PCS and MAC-only variants
0x0094[1]	CONFIGURATION_RX_FLOW_CONTROL_REG1: 0094	Only in MAC+PCS and MAC-only variants
0x0098[1]	CONFIGURATION_RX_FLOW_CONTROL_REG2: 0098	Only in MAC+PCS and MAC-only variants
0x009C[1]	CONFIGURATION_RX_FLOW_CONTROL_PPP_ETYPE_OP_REG: 009C	Only in MAC+PCS and MAC-only variants
0x00A0[1]	CONFIGURATION_RX_FLOW_CONTROL_GPP_ETYPE_OP_REG: 00A0	Only in MAC+PCS and MAC-only variants
0x00A4[1]	CONFIGURATION_RX_FLOW_CONTROL_GCP_PCP_TYPE_REG: 00A4	Only in MAC+PCS and MAC-only variants
0x00A8[1]	CONFIGURATION_RX_FLOW_CONTROL_PCP_OP_REG: 00A8	Only in MAC+PCS and MAC-only variants
0x00AC[1]	CONFIGURATION_RX_FLOW_CONTROL_GCP_OP_REG: 00AC	Only in MAC+PCS and MAC-only variants
0x00B0[1]	CONFIGURATION_RX_FLOW_CONTROL_DA_REG1_LSB: 00B0	Only in MAC+PCS and MAC-only variants
0x00B4[1]	CONFIGURATION_RX_FLOW_CONTROL_DA_REG1_MSB: 00B4	Only in MAC+PCS and MAC-only variants
0x00B8[1]	CONFIGURATION_RX_FLOW_CONTROL_DA_REG2_LSB: 00B8	Only in MAC+PCS and MAC-only variants
0x00BC[1]	CONFIGURATION_RX_FLOW_CONTROL_DA_REG2_MSB: 00BC	Only in MAC+PCS and MAC-only variants
0x00C0[1]	CONFIGURATION_RX_FLOW_CONTROL_SA_REG1_LSB: 00C0	Only in MAC+PCS and MAC-only variants
0x00C4[1]	CONFIGURATION_RX_FLOW_CONTROL_SA_REG1_MSB: 00C4	Only in MAC+PCS and MAC-only variants
0x00D0[1]	CONFIGURATION_RSFEC_REG: 00D0	Only in MAC+PCS/PMA 64 bit variant.
0x00D4[1]	CONFIGURATION_FEC_REG: 00D4	Only in MAC+PCS and PCS-only variants
0x00E0[1]	CONFIGURATION_AN_CONTROL_REG1: 00E0	Only in MAC+PCS and PCS-only variants
0x00E4[1]	CONFIGURATION_AN_CONTROL_REG2: 00E4	Only in MAC+PCS and PCS-only variants
0x00F8[1]	CONFIGURATION_AN_ABILITY: 00F8	Only in MAC+PCS and PCS-only variants
0x0100[1]	CONFIGURATION_LT_CONTROL_REG1: 0100	Only in MAC+PCS and PCS-only variants
0x0104[1]	CONFIGURATION_LT_TRAINED_REG: 0104	Only in MAC+PCS and PCS-only variants
0x0108[1]	CONFIGURATION_LT_PRESET_REG: 0108	Only in MAC+PCS and PCS-only variants
0x010C[1]	CONFIGURATION_LT_INIT_REG: 010C	Only in MAC+PCS and PCS-only variants
0x0110[1]	CONFIGURATION_LT_SEED_REG0: 0110	Only in MAC+PCS and PCS-only variants
0x0130[1]	CONFIGURATION_LT_COEFFICIENT_REG0: 0130	Only in MAC+PCS and PCS-only variants
0x0134[1]	USER_REG_0: 0134	
0x014C[1]	VERSAL_CHANNEL_NUM_REG:0x014C	Only when timestamp is enabled for Versal platform.

Table 28: **Configuration Register Map** *(cont'd)*

Hex Address	Register Name	Notes
0x0154[1]	GT_WIZ_CHANNEL_LOOPBACK_REG:0154	Versal device only.
0x0190	CONFIGURATION_1588_REG	Only in MAC+PCS and PCS-only variants
0x0194	TX_CONFIGURATION_1588_REG	Only in MAC+PCS and PCS-only variants.
0x0198	RX_CONFIGURATION_1588_REG	Only in MAC+PCS and PCS-only variants.
0x019C[1]	CONFIGURATION_TSN_REG: 0x019C	Only when Preemption Feature is enabled.

Notes:

1. Refer to *10G/25G High Speed Ethernet Subsystem Product Guide* (PG210).

Status Register Map for 1G/10G/25G Ethernet Subsystem

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 29: **Status Register Map**

Hex Address	Register Name	Notes
0x0180	STAT_CORE_SPEED_REG	
0x0400[1]	STAT_TX_STATUS_REG1: 0400	
0x0404	STAT_RX_STATUS_REG1: 0404	
0x0408[1]	STAT_STATUS_REG1: 0408	Only in MAC+PCS and PCS-only variants
0x040C[1]	STAT_RX_BLOCK_LOCK_REG: 040C	Only in MAC+PCS and PCS-only variants
0x043C[1]	STAT_RX_RSFEC_STATUS_REG: 043C	Only for MAC+PCS/PMA 64-bit variant
0x0448[1]	STAT_RX_FEC_STATUS_REG: 0448	Only in MAC+PCS and PCS-only variants
0x044C[1]	STAT_TX_RSFEC_STATUS_REG: 044C	Only for MAC+PCS/PMA 64-bit variant
0x0450[1]	STAT_TX_FLOW_CONTROL_REG1: 0450	Only in MAC+PCS variant and MAC-only variants
0x0454[1]	STAT_RX_FLOW_CONTROL_REG1: 0454	Only in MAC+PCS variant and MAC-only variants
0x0458[1]	STAT_AN_STATUS: 0458	Only in MAC+PCS and PCS-only variants
0x045C[1]	STAT_AN_ABILITY: 045C	Only in MAC+PCS and PCS-only variants
0x0460[1]	STAT_AN_LINK_CTL: 0460	Only in MAC+PCS and PCS-only variants
0x0464[1]	STAT_LT_STATUS_REG1: 0464	Only in MAC+PCS and PCS-only variants

Table 29: **Status Register Map** *(cont'd)*

Hex Address	Register Name	Notes
0x0468[1]	STAT_LT_STATUS_REG2: 0468	Only in MAC+PCS and PCS-only variants
0x046C[1]	STAT_LT_STATUS_REG3: 046C	Only in MAC+PCS and PCS-only variants
0x0470[1]	STAT_LT_STATUS_REG4: 0470	Only in MAC+PCS and PCS-only variants
0x0474[1]	STAT_LT_COEFFICIENT0_REG: 0474	Only in MAC+PCS and PCS-only variants
0x0494[1]	STAT_RX_VALID_CTRL_CODE: 0494	Only in MAC+PCS and PCS-only variants
0x04A0[1]	STAT_GT_WIZ_REG	
0x049C[1]	STAT_TSN_REG: 0x049C	Only when 802.1cm Preemption feature enabled

Notes:

1. Refer to *10G/25G High Speed Ethernet Subsystem Product Guide* (PG210)

Configuration and Status Register Map for 1G Ethernet PCS/PMA

Table 30: **Registers for 1G Ethernet PCS/PMA**

Register Address	Register Name
0x0000	Register 0: Control Register[1][2]
0x0004	Register 1: Status Register[2]
0x0008	Register 2: PHY Identifier[2]
0x000C	Register 3: PHY Identifier[2]
0x0010	Register 4: Auto-Negotiation Advertisement[2]
0x0014	Register 5: Auto-Negotiation Link Partner Ability[2]
0x0018	Register 6: Auto-Negotiation Expansion Register[2]
0x003C	Register 15: Extended Status[2]
0x0040	Auto-Negotiation Interrupt Control Register: Vendor Specific Register 16[2]
0x004C	1588 Control: Vendor Specific Register 19
0x0050	RX PHY Fixed Latency: Vendor Specific Register 20
0x0054	RX PHY Variable Latency: Vendor Specific Register 21

Notes:

1. Power down and Loopback feature is not supported.
2. For 1000 BASEX, refer to the MDIO Registers for 1000 BASEX with Auto-Negotiation table in the *1G/2.5G Ethernet PCS/PMA or SGMII LogiCORE IP Product Guide* (PG047). For SGMII, refer to the MDIO Registers for SGMII with Auto-Negotiation table in the same document.

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 the TICK_REG register, depending on the value of MODE_REG[30] (`tick_reg_mode_sel`). The `pm_tick` signal is used when MODE_REG[30] = 0 and TICK_REG is 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.

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

Implementation of statistics counters for 1G and 10G modes of operation is common to both the 10G/25G Ethernet subsystem and 1G/2.5G Ethernet PCS/PMA. The current values of statistics show the value pertaining to the current mode of operation. When there is a change in line rate, the system resets statistics counters.

The statistics counters, detailed in the following table, are present in the design when the Include Statistics Counters option is selected in the Configuration tab in the IDE.

Table 31: **Statistics Counters**

Hex Address	Register Name	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	
0x0670	STAT_RX_RSFEC_CORRECTED_CW_INC_LSB: 0670	Only for MAC+PCS/PMA 64-bit variant
0x0674	STAT_RX_RSFEC_CORRECTED_CW_INC_MSB: 0674	Only for MAC+PCS/PMA 64-bit variant
0x0678	STAT_RX_RSFEC_UNCORRECTED_CW_INC_LSB: 0678	Only for MAC+PCS/PMA 64-bit variant
0x067C	STAT_RX_RSFEC_UNCORRECTED_CW_INC_MSB: 067C	Only for MAC+PCS/PMA 64-bit variant
0x0680	STAT_RX_RSFEC_ERR_COUNT0_INC_LSB: 0680	Only for MAC+PCS/PMA 64-bit variant
0x0684	STAT_RX_RSFEC_ERR_COUNT0_INC_MSB: 0684	Only for MAC+PCS/PMA 64-bit variant
0x06A0	STAT_TX_FRAME_ERROR_LSB: 06A0	
0x06A4	STAT_TX_FRAME_ERROR_MSB: 06A4	Only in MAC+PCS and PCS-only variants
0x0700	STAT_TX_TOTAL_PACKETS_LSB: 0700	Only in MAC+PCS and PCS-only variants
0x0704	STAT_TX_TOTAL_PACKETS_MSB: 0704	Only in MAC+PCS and PCS-only variants
0x0708	STAT_TX_TOTAL_GOOD_PACKETS_LSB: 0708	Only in MAC+PCS and PCS-only variants

Table 31: **Statistics Counters** *(cont'd)*

Hex Address	Register Name	Notes
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
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
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_BYETS_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
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

Table 31: **Statistics Counters** *(cont'd)*

Hex Address	Register Name	Notes
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_LSB: 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
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
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

Table 31: **Statistics Counters** *(cont'd)*

Hex Address	Register Name	Notes
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
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_MSB: 08FC	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

Table 31: **Statistics Counters** *(cont'd)*

Hex Address	Register Name	Notes
0x0918	STAT_RX_TEST_PATTERN_MISMATCH_LSB: 0918	Only in MAC+PCS variant
0x091C	STAT_RX_TEST_PATTERN_MISMATCH_MSB: 91C	Only in MAC+PCS variant
0x0920	STAT_FEC_INC_CORRECT_COUNT_LSB: 0920	Only in MAC+PCS and PCS-only variants
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	Only when 802.1cm Preemption feature enabled
0x0984	STAT_TX_MM_STATUS_MSB: 0x0984	Only when 802.1cm Preemption feature enabled
0x0988	STAT_TX_MM_STATUS_LSB: 0x0988	Only when 802.1cm Preemption feature enabled
0x098C	STAT_TX_MM_FRAGMENT_MSB: 0x098C	Only when 802.1cm Preemption feature enabled
0x0990	STAT_TX_MM_HOLD_LSB: 0x0990	Only when 802.1cm Preemption feature enabled
0x0994	STAT_TX_MM_HOLD_MSB: 0x0994	Only when 802.1cm Preemption feature enabled
0x0998	STAT_RX_MM_ASSEMBLY_ERROR_LSB: 0x0998	Only when 802.1cm Preemption feature enabled
0x099C	STAT_RX_MM_ASSEMBLY_ERROR_MSB: 0x099C	Only when 802.1cm Preemption feature enabled
0x09A0	STAT_RX_MM_FRAME_SMD_ERROR_LSB: 0x09A0	Only when 802.1cm Preemption feature enabled
0x09A4	STAT_RX_MM_FRAME_SMD_ERROR_MSB: 0x09A4	Only when 802.1cm Preemption feature enabled
0x09A8	STAT_RX_MM_FRAME_ASSEMBLY_OK_LSB: 0x09A8	Only when 802.1cm Preemption feature enabled
0x09AC	STAT_RX_MM_FRAME_ASSEMBLY_OK_MSB: 0x09AC	Only when 802.1cm Preemption feature enabled
0x09B0	STAT_RX_MM_FRAGMENT_LSB: 0x09B0	Only when 802.1cm Preemption feature enabled
0x09B4	STAT_RX_MM_FRAGMENT_MSB: 0x09B4	Only when 802.1cm Preemption feature enabled
0x1000	GIG_EHERNET_PCS_PMA_CONFIGURATION_VECTOR:1000	
0x1004	GIG_ETHERNET_PCS_PMA_STATUS_VECTOR:1004	

For a description of statistics counters, see the statistics counter tables in the *10G/25G High Speed Ethernet Subsystem Product Guide* (PG210) for the bit assignments for the statistics counters.

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. The following descriptions cover only the registers that have been modified with respect to default operation of the respective IPs in applicable mode. For further information on register interface definitions (not defined in this guide), see *10G/25G High Speed Ethernet Subsystem Product Guide* (PG210) or *1G/2.5G Ethernet PCS/PMA or SGMII LogiCORE IP Product Guide* (PG047).

Configuration Registers for 1G/10G/25G Ethernet Subsystem

See the *10G/25G High Speed Ethernet Subsystem Product Guide* (PG210) for information on configuration registers for the 1G/10G/25G Ethernet Subsystem.

CONFIGURATION_TX_REG1: 000C

Table 32: **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]
7:6	2	RW	ctl_core_speed_sel[3]
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
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.
3. 2'b10: This is the default configuration. You have to write this to configure the core in 10G mode.
   2'b01: You have to write this to configure the core in 1G mode.
   2'b00: You have to write this to configure the core in 25G mode.
   Others: Reserved.
   The values provided by these two register bits are ignored when Auto-Negotiation is enabled. Auto-switching is performed according to the resolved speed between the local device and the link partner.

STAT_CORE_SPEED_REG: 0180

Table 33: **STAT_CORE_SPEED_REG: 0180**

Bits	Default	Type	Signal
1:0	2	RO	state_core_speed[1]

Notes:

1. Each mode indicate different configurations.
   - **2'b10:** Indicates that the core is configured in 10G mode.
   - **2'b01:** Indicates that the core is configured in 1G mode.
   - **2'b00:** Indicates that the core is configured in 25G mode.
   - **Others:** Reserved.

CONFIGURATION_1588_REG: 0x0190

Table 34: **CONFIGURATION_1588_REG: 0x0190**

Bits	Default	Type	Signal
0	1	RW	ctl_tx_lat_adj_enb
1	1	RW	ctl_tx_lat_adj_enb ***Note*:** For 1G mode, this register is RESERVED.
2	0	RW	ctl_ptp_transpclk_mode
3	0	RW	ctl_tx_timestamp_adj_enb ***Note*:** For 1G mode, this register is RESERVED.
4	1	RW	ctl_tx_timestamp_adj_enb ***Note*:** For 1G mode, this register is RESERVED.

TX_CONFIGURATION_1588_REG: 0x0194

Table 35: **TX_CONFIGURATION_1588_REG: 0x0194**

Bits	Default	Type	Signal
31:0	0	RW	ctl_tx_latency

RX_CONFIGURATION_1588_REG: 0x0198

Table 36: **RX_CONFIGURATION_1588_REG: 0x0198**

Bits	Default	Type	Signal
31:0	0	RW	ctl_rx_latency ***Note*:** For 1G mode, the register is RESERVED.

GIG_EHERNET_PCS_PMA_CONFIGURATION_VECTOR:0x1000

Table 37: **GIG_EHERNET_PCS_PMA_CONFIGURATION_VECTOR:0x1000**

Bits	Default	Type	Signal
4:0	0	RW	configuration_vector ***Note*:** For 1G mode, the register is RESERVED.

GIG_ETHERNET_PCS_PMA_STATUS_VECTOR:0x1004

Table 38: **GIG_ETHERNET_PCS_PMA_STATUS_VECTOR:0x1004**

Bits	Default	Type	Signal
15:0	0	RO	status_vector ***Note*:** For 1G mode, the register is RESERVED.

Status Registers for 1G/10G/25G Ethernet Subsystem

See *1G/2.5G Ethernet PCS/PMA or SGMII LogiCORE IP Product Guide* (PG047) for more information on Status Registers for 1G/10G/25G Subsystem.

STAT_RX_STATUS_REG1: 0404

Table 39: **STAT_RX_STATUS_REG1: 0404**

Bits	Default	Type	Signal
0	0	RO LL	stat_rx_status
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
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.

Configuration and Status Registers for 1G/2.5G Ethernet PCS/PMA

AXI4-Lite support has been added in the 1G Ethernet PCS/PMA IP to assist you in programming the control and status registers. The addresses of the registers are word aligned for AXI4-Lite accesses. Strobing for data while accessing these registers is disabled.

Any read/write operations to given addresses lead to read/write operations to the corresponding 16-bit register as defined in *1G/2.5G Ethernet PCS/PMA or SGMII LogiCORE IP Product Guide* (PG047).

The following vendor-specific registers have been added to the MDIO PCS Address space when configured for 1000BASE-X operation. See the *1G/2.5G Ethernet PCS/PMA or SGMII LogiCORE IP Product Guide* (PG047).

For registers 0x0000-0x003C, see 1000BASE-X or 2500BASE-X Standard Without Optional Auto-Negotiation Table in *1G/2.5G Ethernet PCS/PMA or SGMII LogiCORE IP Product Guide* (PG047)

Table 40: **1588 Control: Vendor Specific Register 19**

Bits	Default Value	Access	Description
15.4	N/A	RO	Reserved
3	1	RW	Timestamp correction enable. When 1, the RX timestamp is adjusted to compensate for enabled PHY fixed and variable latencies. When 0, no adjustment is made to the timestamp.
2	1	RW	Fixed RX PHY latency correction enable. When 1, the RX timestamp is adjusted to compensate for fixed PHY latency by using the correction value specified in Table 2-30. When 0, no adjustment is made to compensate for fixed known latencies.
1	0	RO	Reserved
0	1	RW	Variable RX transceiver latency correction enable. When 1, the RX timestamp is adjusted to compensate for measurable variable transceiver latency (for 1000BASE-X this is the barrel shift position of the serial-to-parallel converter in the GTX transceiver PMA). This only varies when the subsystem is initialized following a power-on, reset, or recovery from loss of synchronization; it then remains constant for normal operation. When 0, no adjustment is made to compensate for measurable variable known latencies.

Table 41: **RX PHY Fixed Latency: Vendor Specific Register 20**

Bits	Default Value	Access	Description
15:0	0xC8	RW	RX 1000BASE-X Fixed Delay in ns. This value is initialized to the known RX latency from the serial wire input into the FPGA, through the transceiver fixed latency components prior to the timestamping position.

Table 42: **RX PHY Variable Latency: Vendor Specific Register 21**

Bits	Default Value	Access	Description
15:0	N/A	RO	RX 1000BASE-X variable RX Delay in UI. This value is measured within the subsystem following RX synchronization (for 1000BASE-X this is the barrel shift position of the serial-to-parallel converted in the transceiver PMA). This only varies when the subsystem is initialized following a power-on, reset, or recovery from loss of synchronization; it then remains constant for normal operation.

See *1G/2.5G Ethernet PCS/PMA or SGMII LogiCORE IP Product Guide* (PG047) for information on Configuration Registers for 1G/2.5G Ethernet PCS-PMA.

Designing with the Subsystem

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

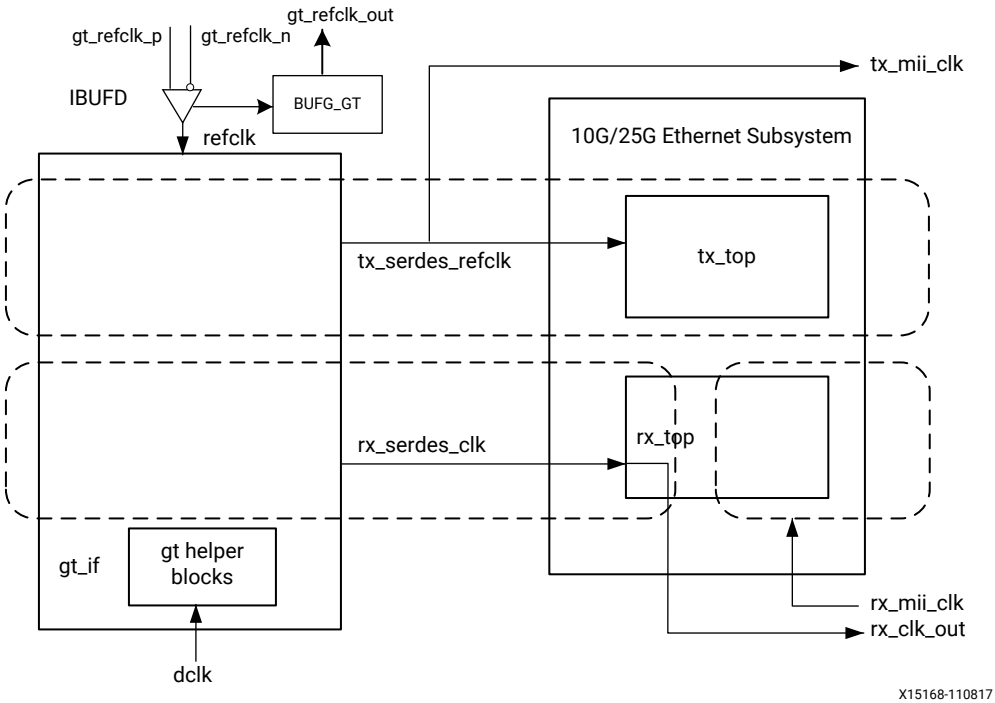
Clocking

This section describes the clocking for all the 1G configurations at the component support wrapper layer.

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 the following figure.

Figure 26: **PCS/PMA Clocking**

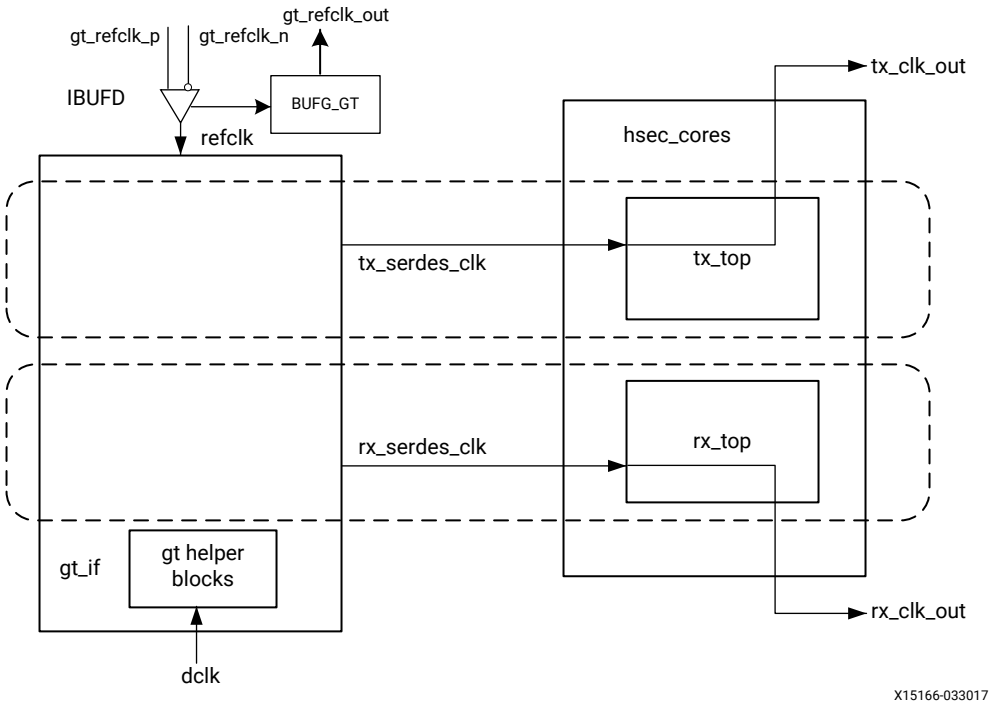


- **refclk_p0, refclk_n0, tx_serdes_refclk:** The `refclk` differential pair is required to be an input to the FPGA. The example design includes a buffer to convert this clock to a single-ended signal `refclk`, which is used as the reference clock for the GT block. The `tx_serdes_refclk` is directly 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.

32 Bit 1/10/25G Ethernet MAC with PCS/PMA Clocking

The clocking architecture for the Low Latency 10/25G MAC with PCS/PMA clocking is illustrated in the following figure. 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 the following figure.

Figure 27: **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.

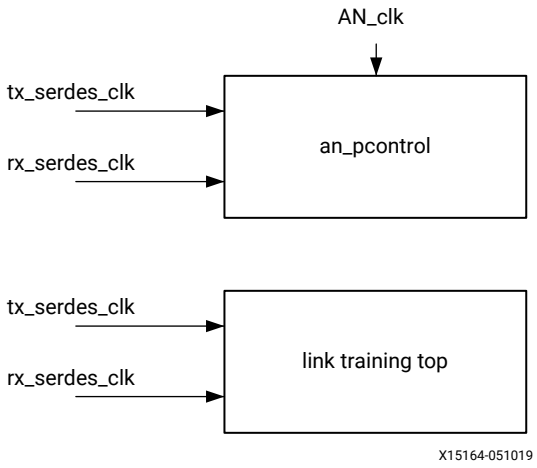
- **rx_clk_out:** The `rx_clk_out` output signal is presented as a reference for the RX control and status signals processed by the RX core. It is the same frequency as the `rx_serdes_clk`. Because there is no RX FIFO, this is also the clock which drives the RX AXI4-Stream Interface. In this arrangement, `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.

Auto-Negotiation and Link Training Clocking

The clocking architecture for the Auto-Negotiation and Link Training blocks are illustrated in the following figure. 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 28: **Auto-Negotiation and Link Training Clocking**

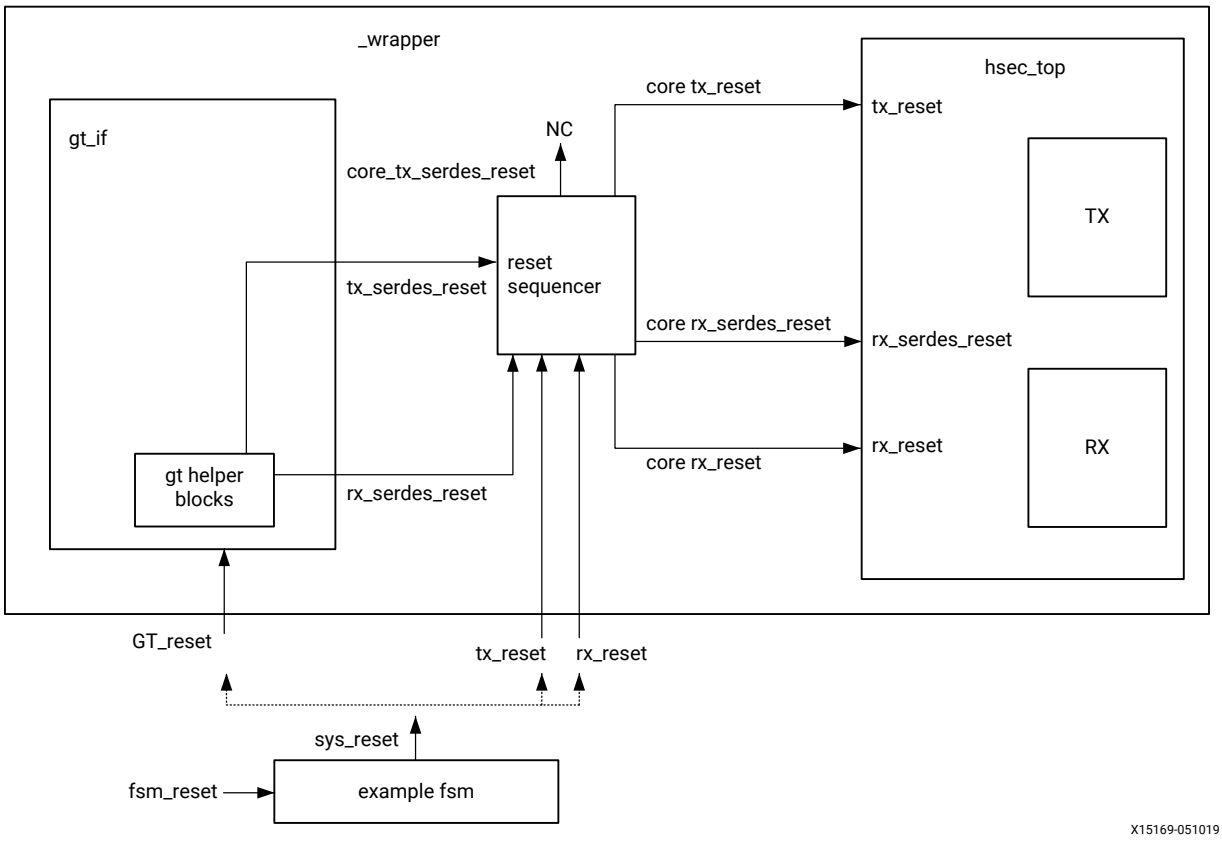


- **rx_serdes_clk:** The `rx_serdes_clk` drives the RX line side logic for the Auto-Negotiation and Link Training.
- **tx_serdes_clk:** The `tx_serdes_clk` drives the TX line side logic for the Auto-Negotiation and Link Training. The DME frame is generated on this clock domain.
- **AN_clk:** The `AN_clk` 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

The following figure shows the reset structure for the 1G/10G/25G Ethernet Subsystem MAC with PCS/PMA as implemented at the component support wrapper layer. Clocks are not shown for clarity.

Figure 29: **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

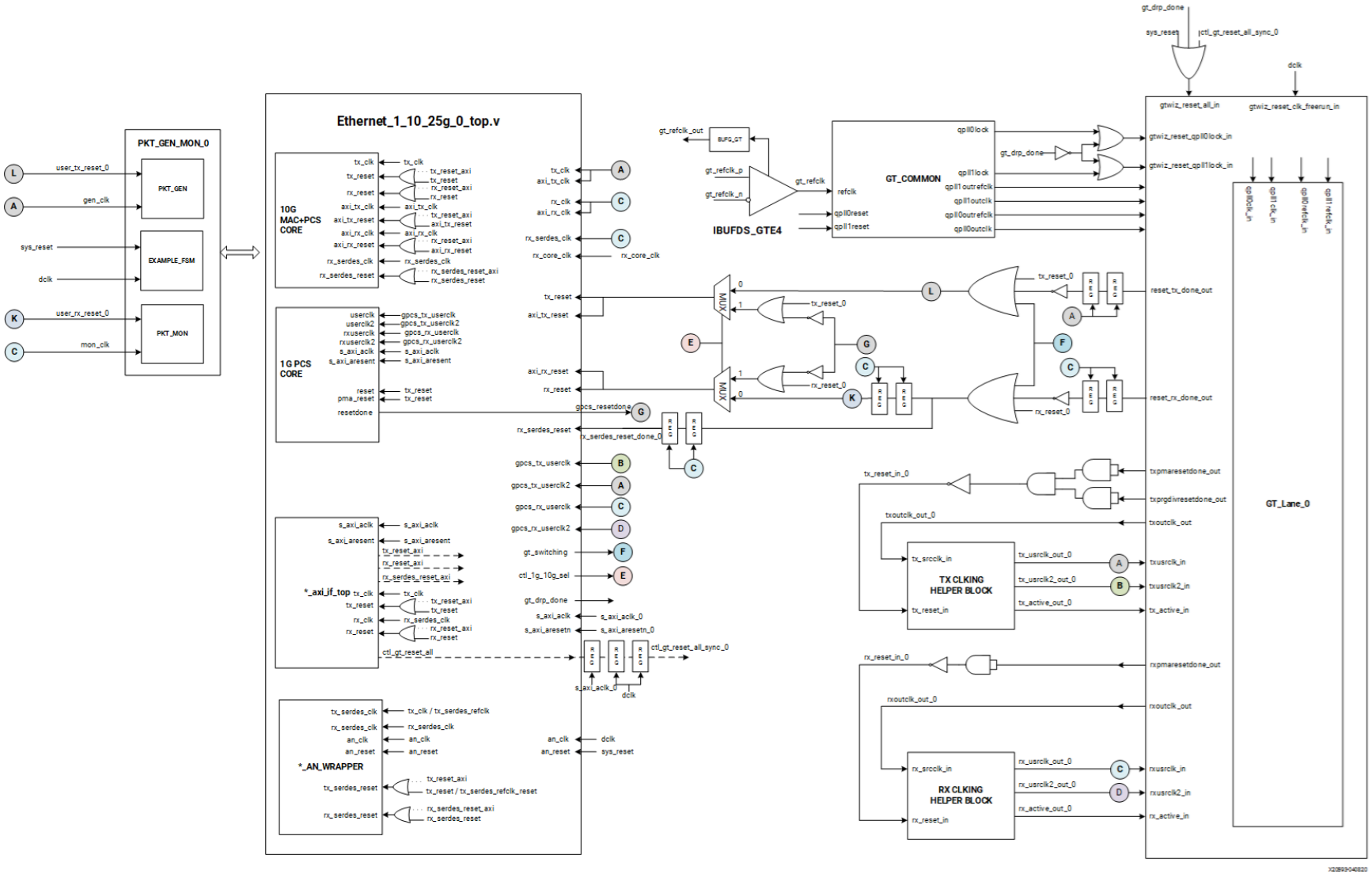
Timing of the reset signals is handled by the reset sequencer block.

- **GT_reset:** The `GT_reset` is the asynchronous active-High reset input to the GT. The internal resets of the GT are handled by the GT helper blocks.
- **tx_reset:** The `tx_reset` is the asynchronous active-High reset for the TX path logic of the 1G/10G/25G Ethernet Subsystem. 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 1G/10G/25G Ethernet Subsystem. 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

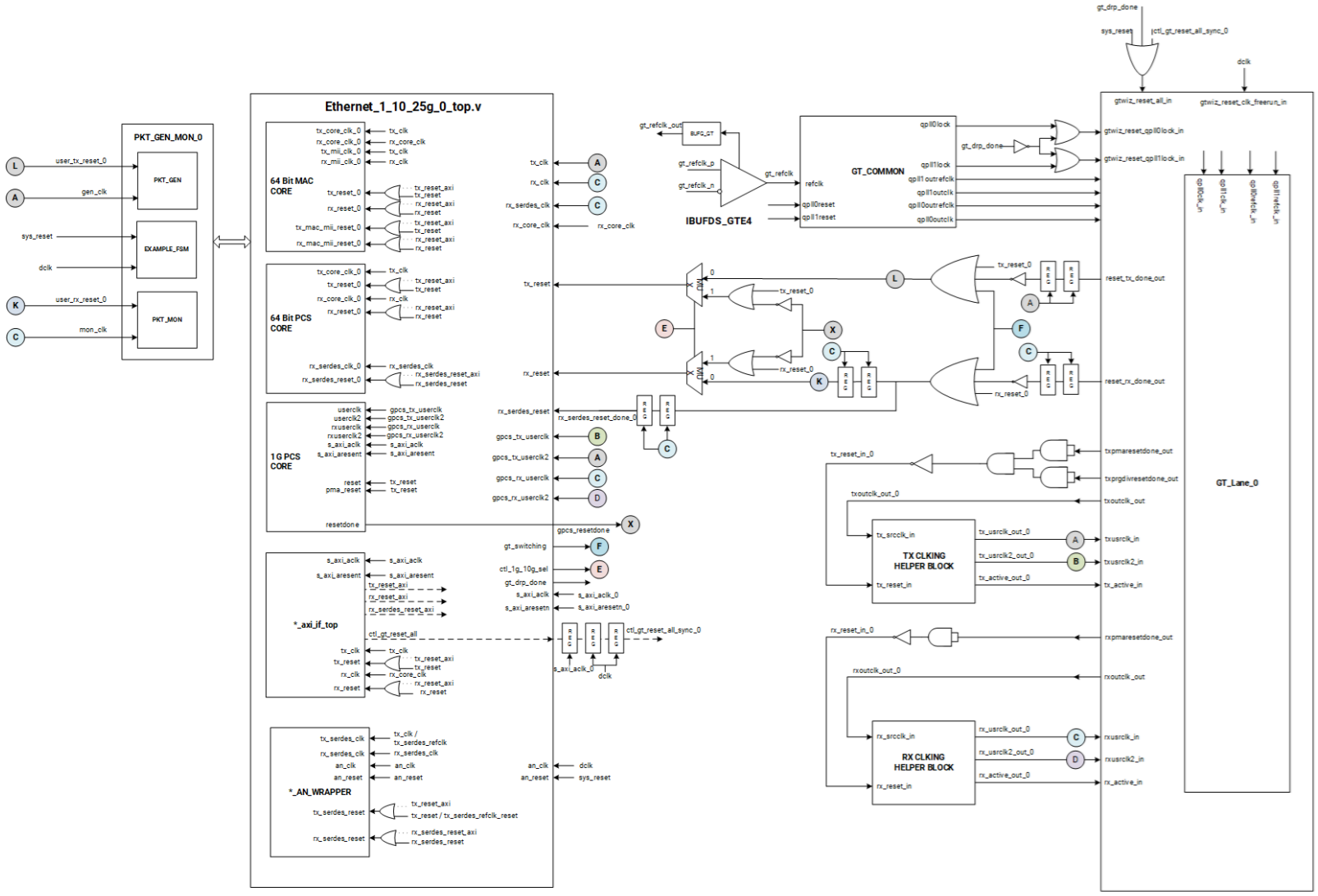
Detailed Diagram of 32-bit MAC+PCS/PMA Single Core (UltraScale/ UltraScale+)

Figure 30: **Detailed Diagram of 32-bit MAC+PCS/PMA Single Core (UltraScale/UltraScale+)**



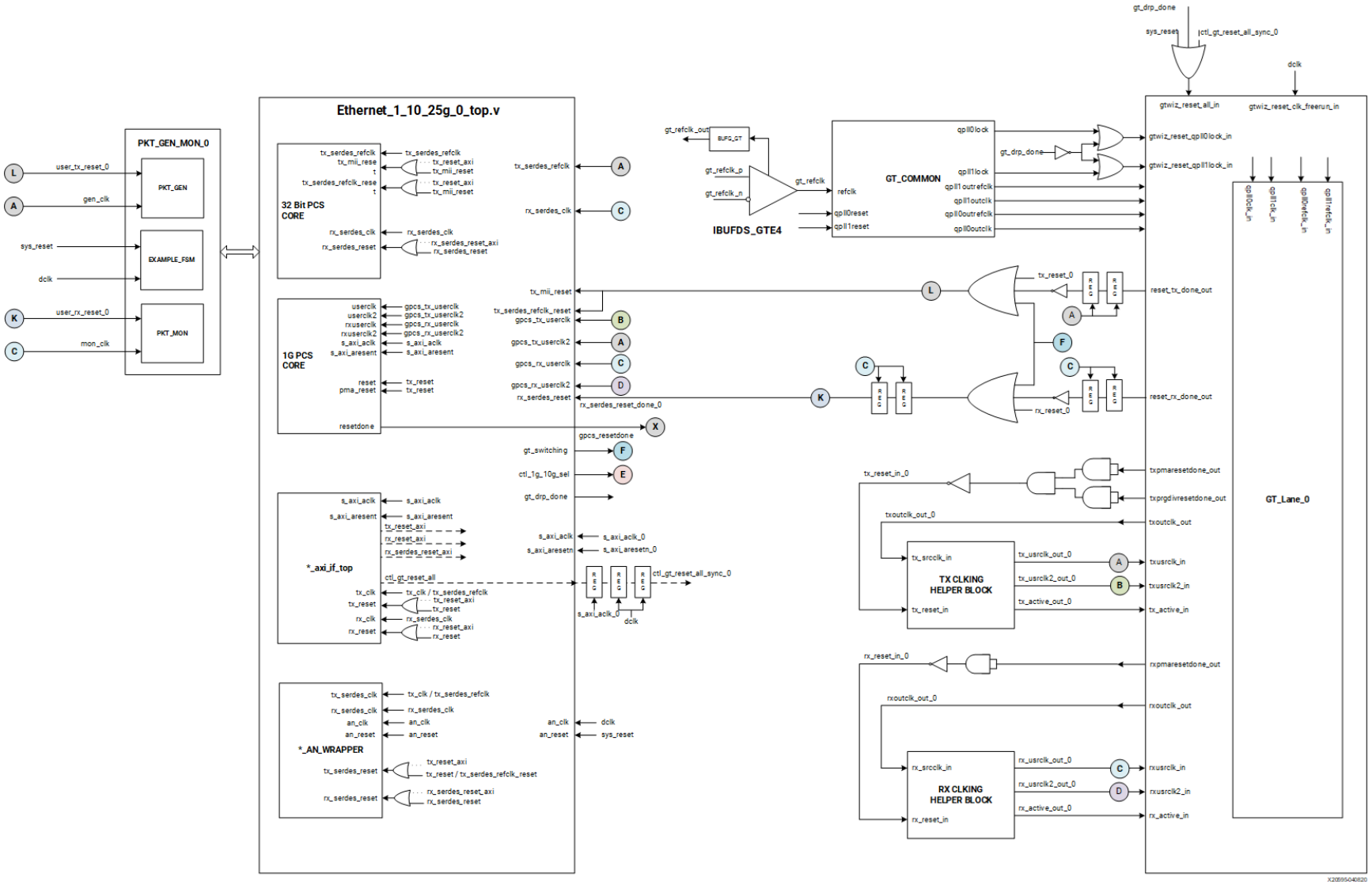
Detailed Diagram of 64-bit MAC+PCS/PMA Single Core (UltraScale/ UltraScale+)

Figure 31: **Detailed Diagram of 64-bit MAC+PCS/PMA Single Core (UltraScale/UltraScale+)**



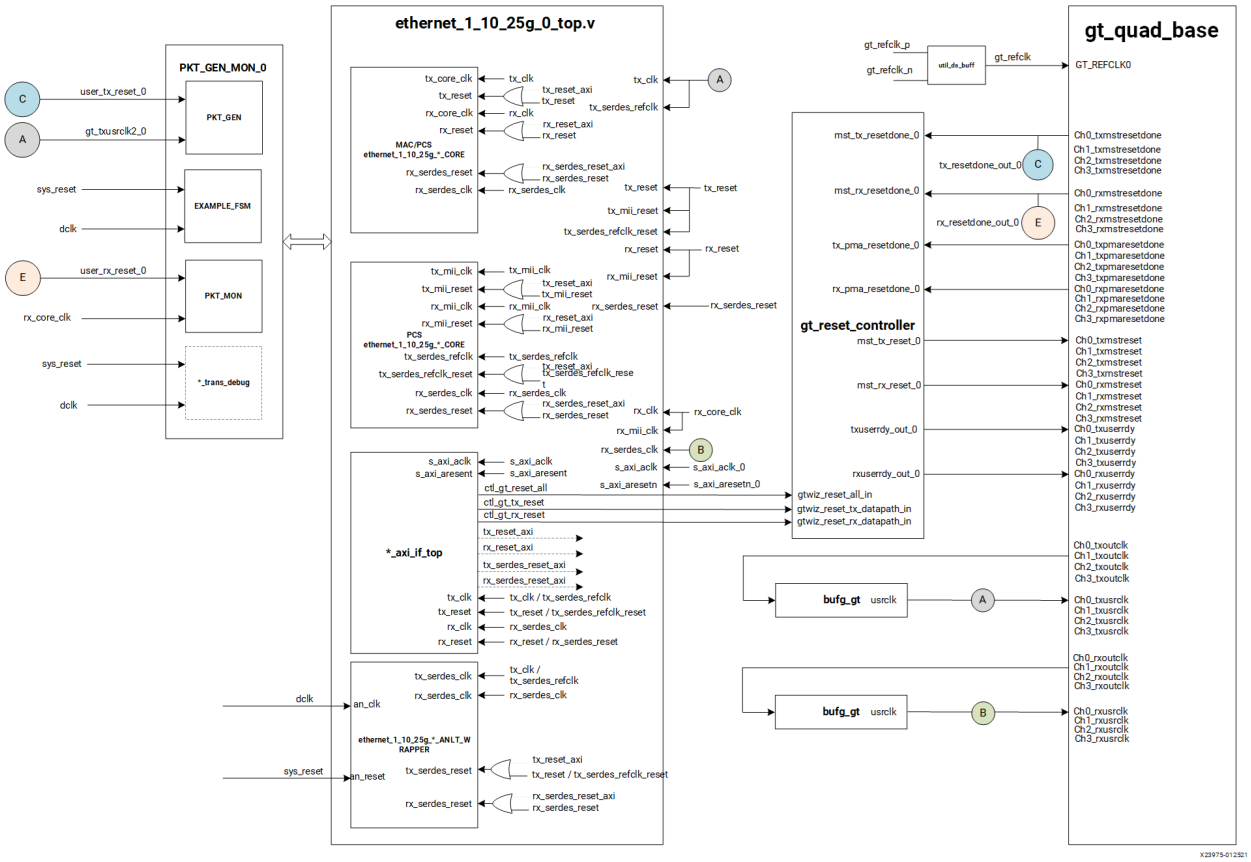
Detailed Diagram of 32-bit PCS-Only Single Core (UltraScale/UltraScale+)

Figure 32: **Detailed Diagram of 32-bit PCS-Only Single Core (UltraScale/UltraScale+)**



Detailed Diagram of Single Core (Versal)

Figure 33: **Detailed Diagram of Single Core (Versal)**



Support for IEEE Standard 1588v2

Overview

This section details the packet timestamping function of the 1G/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 support only for two-step IEEE 1588v2 functionality.

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.

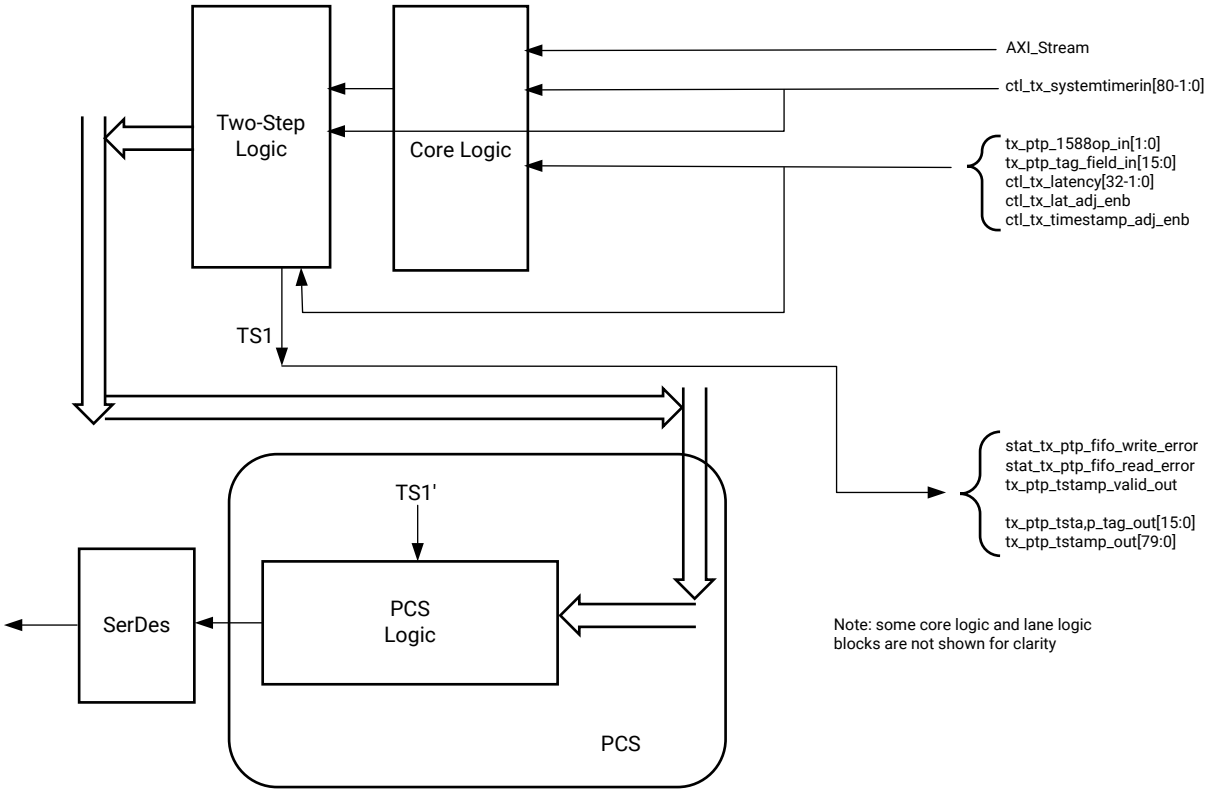
Other timing slave devices on the network receive these timing reference packets from the network timing master and attempt to synchronize their own local timer references to it. This mechanism relies on these Ethernet ports also taking timestamps (samples of their own local timer) when the 1588 timing 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 216 (see IEEE 1588 clause 13.3.2.7). This timer should count from 0 through the full range up to 264 -1 before wrapping around.

Egress

Figure 34: **Egress**



The TS references are defined as follows:

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

TS1 always has a correction applied, so that it is referenced to the TS1' plane.

***Note*:** For 10G 1588, registers defined at address 0x0190, 0x0194 and 0x0198 must be programmed. For 1G 1588, registers defined at address 0x0190, 0x0194, 0x004C, 0x0050 and 0x0054 must be programmed.

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 Port Descriptions.

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 Port Descriptions (with the upper bits of data set to zero as defined in the table).

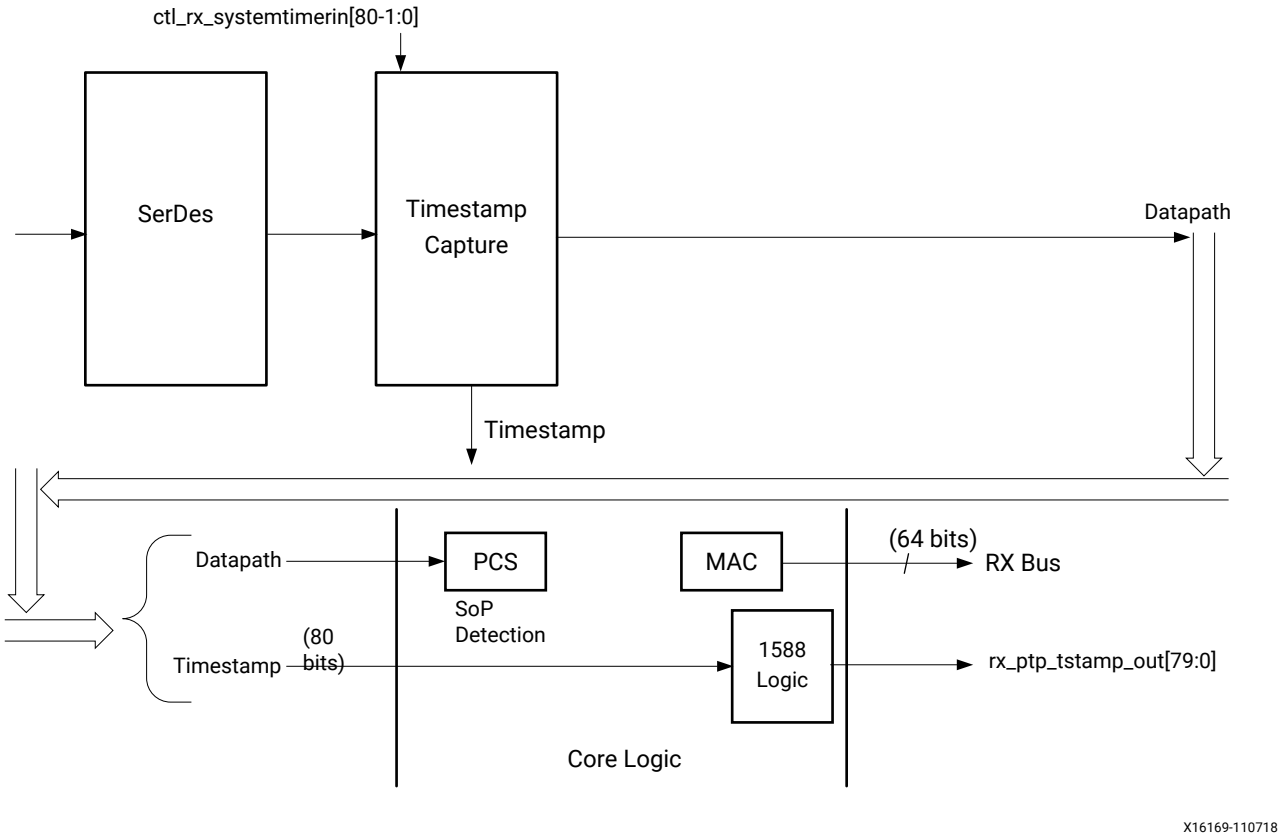
Frame-by-Frame Timestamping Operation

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.

Ingress

Figure 35: **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.

Port Descriptions

The following table details the additional signals present when the packet timestamping feature is included.

Table 43: **1588v2 Port List and Descriptions**

Name	I/O	Description	Clock Domain
Common			
systemtimerin	I	Common System timer input. In TOD 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.	
IEEE 1588 Interface – TX Path			
tx_ptp_tstamp_valid_out	O	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]	O	Tag output corresponding to tx_ptp_tag_field_in[15:0]	tx_clk_out
tx_ptp_tstamp_out[80-1:0]	O	Timestamp for the transmitted packet SOP corresponding to the time at which it passed the capture plane. Time format is the same as timer input.	tx_clk_out
tx_ptp_1588op_in[1:0]	I	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– Reserved. 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
tx_ptp_tag_field_in[15:0]	I	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

Table 43: **1588v2 Port List and Descriptions** *(cont'd)*

Name	I/O	Description	Clock Domain
ctl_tx_ptp_1step_enable	I	When set to 1, this bit enables 1-step operation.	tx_clk_out
ctl_ptp_transpclk_mode	I	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 adds 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 is already adjusted with the proper RX timestamp.	tx_clk_out
stat_tx_ptp_fifo_write_error	O	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	O	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
ctl_tx_latency	I	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 nsb values. The latency is in binary Q16.16 format.	tx_clk_out
ctl_tx_lat_adj_enb	I	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.	tx_clk_out
ctl_tx_timestamp_adj_enb	I	When this signal is enabled, the delay computation on the TX path takes into account the value got from GT DRP read for latency of TX gearbox FIFO. Because the design does not use TX gearbox FIFO, this signal need not be updated.	tx_clk_out
IEEE 1588 Interface – RX Path			
ctl_rx_systemtimerin[80-1:0]	I	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]	O	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.	rx_core_clk
rx_ptp_tstamp_valid_out	O	This bit indicates that a valid timestamp is being presented on the RX	rx_serdes_clk
ctl_rx_latency	I	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 signal is not valid in 1G mode.	rx_clk_out
ctl_rx_lat_adj_enb	I	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 signal is not valid in 1G mode.	rx_clk_out

Table 43: **1588v2 Port List and Descriptions** *(cont'd)*

Name	I/O	Description	Clock Domain
ctl_rx_timestamp_adj_enb	I	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 signal is not valid in 1G mode.	rx_clk_out

IEEE 1588v2 PTP Functional Description

The IEEE 1588v2 feature of the 1G/10G/25G Switching 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. You need 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 1G/10G/25G Switching Ethernet Subsystem. This is shown in the following diagrams.

Figure 36: **Receive**

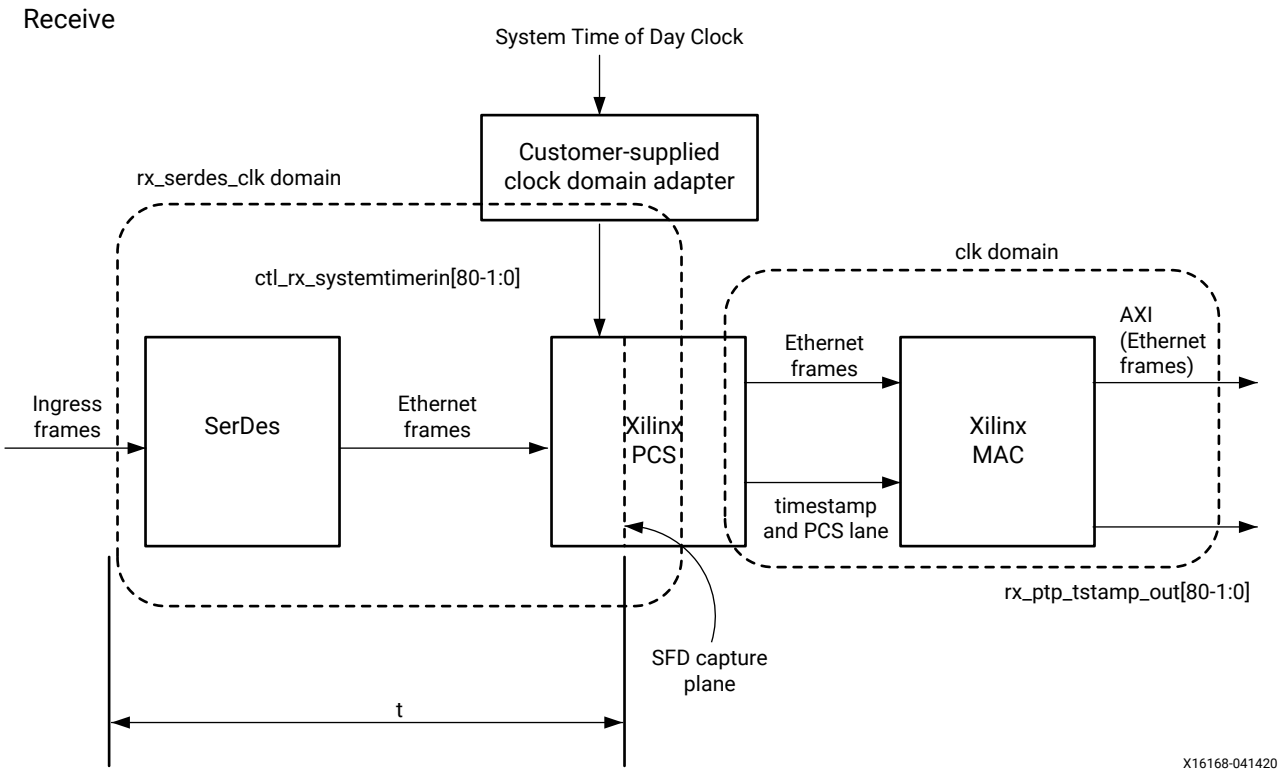
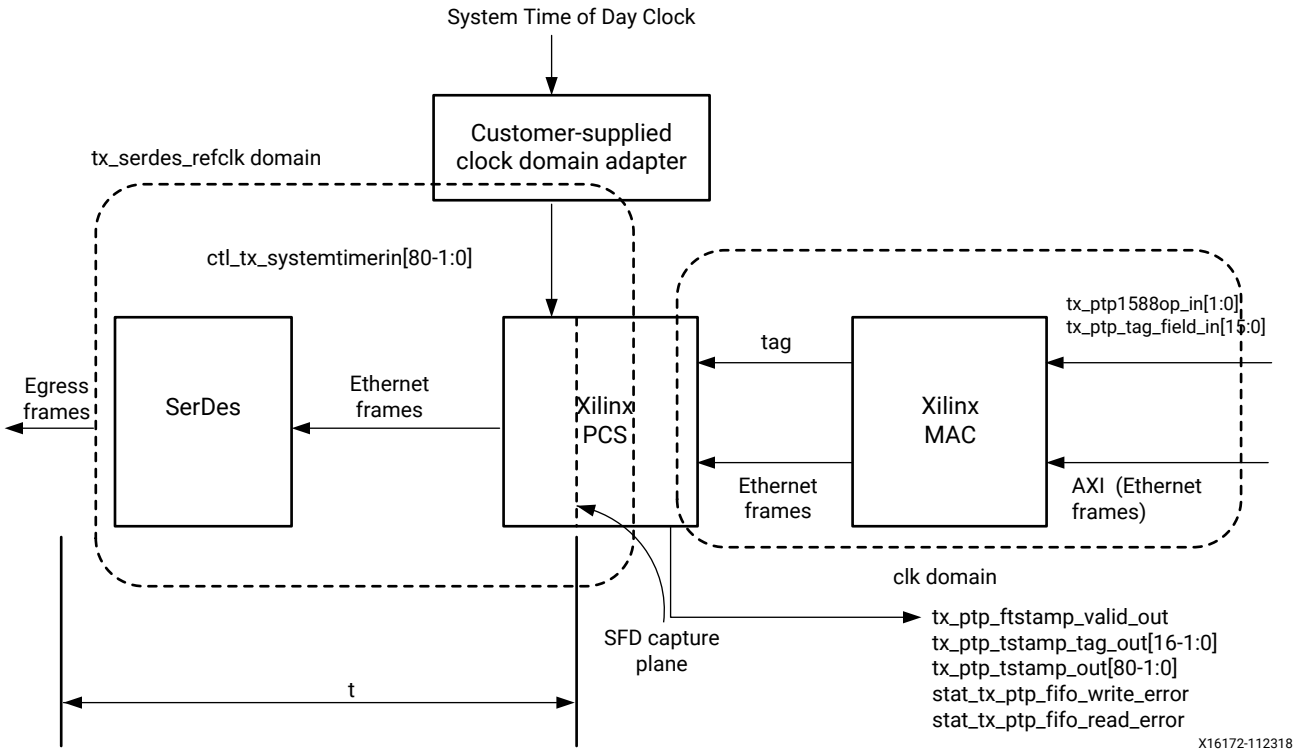


Figure 37: **Transmit**



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.

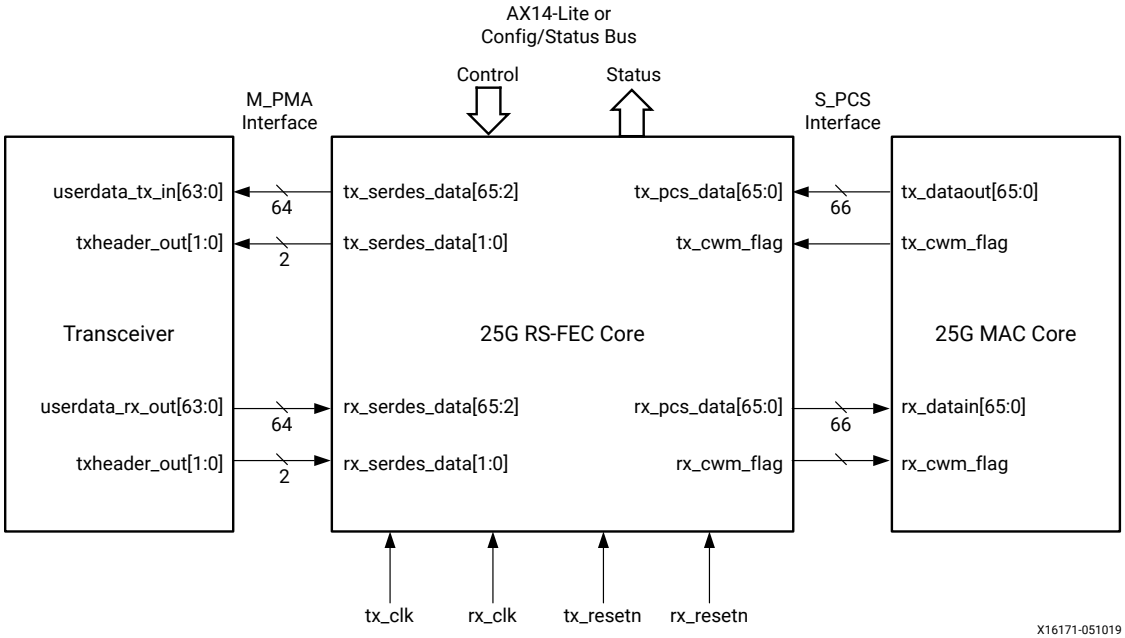
RS-FEC Support

Overview

This section describes the optional RS-FEC function of the 1/10/25G 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 38: **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.

Further information can also be found in the *25G IEEE 802.3by Reed-Solomon Forward Error Correction LogiCORE IP Product Guide* (PG217) (registration required).

Port Descriptions

Table 44: **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 1/10/25G 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 1/10/25G 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
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.	

Table 44: **RS-FEC Port List and Descriptions** *(cont'd)*

Port	Direction	Description	Clock Domain
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
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 1/10/25G 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) (registration required) 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) (registration required) 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.

802.1cm Preemption Feature

Features

Optional fee-based Time Sensitive Networking (TSN) feature based on *IEEE Standard for Local and Metropolitan Area Networks - Time Sensitive Networking for Fronthaul* (IEEE Std 802.1CM-2018).

- Supports frame preemption
- Supports interspersing express traffic with low priority traffic

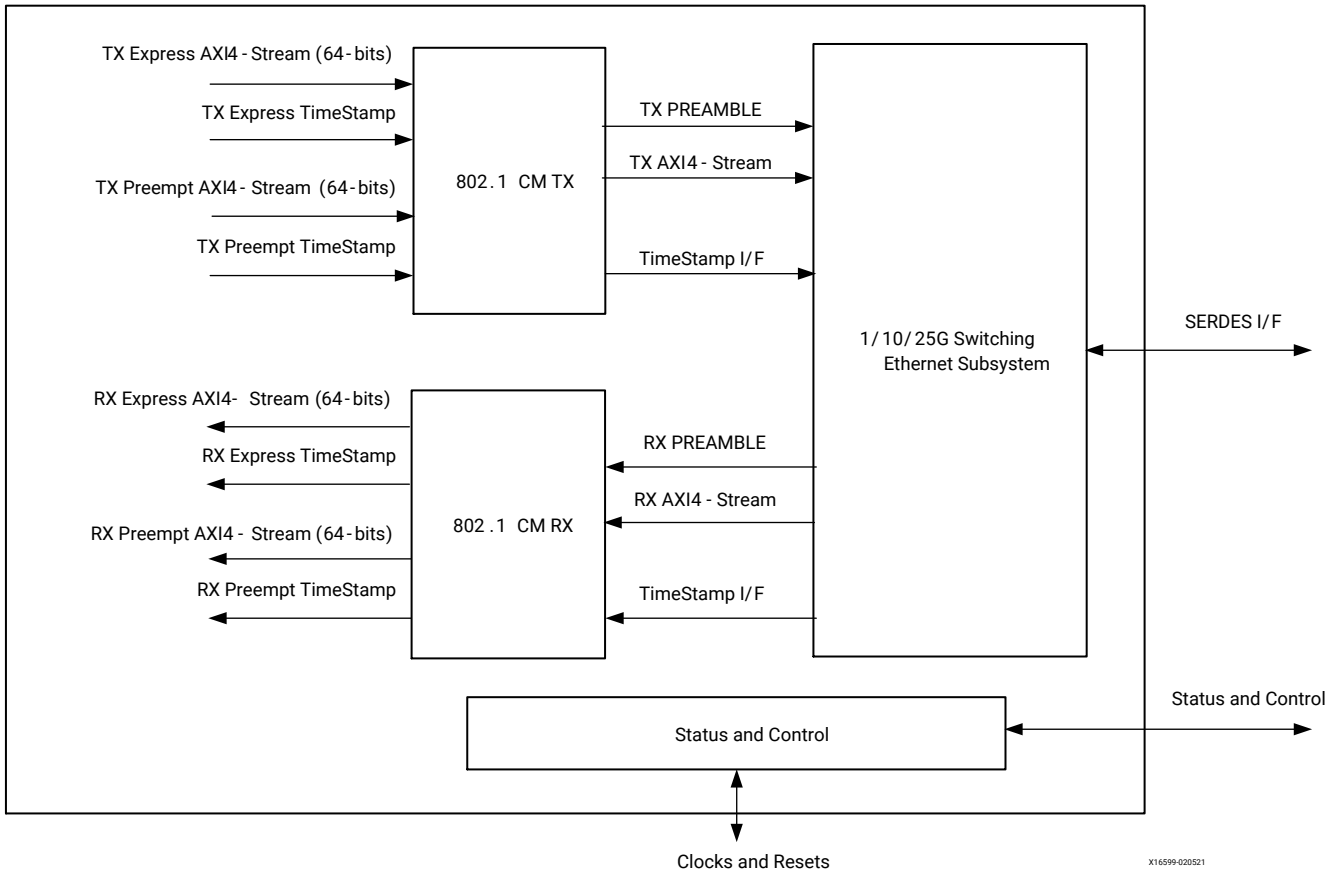
Overview

Frame preemption and Express traffic interspersing is defined by IEEE standard 802.1 CM. The 1G/10G/25G Ethernet Subsystem includes the optional TSN feature based on IEEE 802.1 CM.

Product Specification

The following figure shows the block diagram of the 1/10/25G Ethernet IP subsystem with the optional TSN feature:

Figure 39: **1/10/25G 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 AXI4-Stream Interface.

Note: The same descriptions and rules apply to these signals as the ones in AXI4-Stream Interface.

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 AXI4-Stream interface.

Frame Transmission

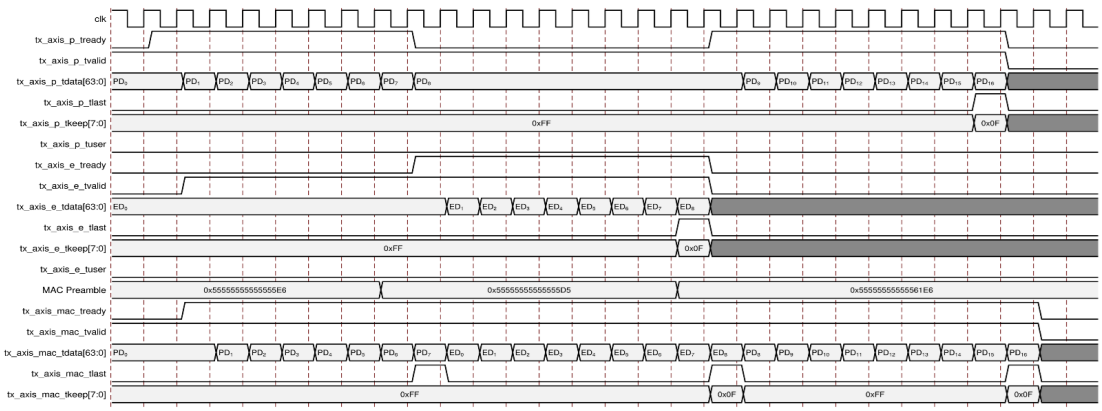
If you disable preemption, the core services the express and preempt traffic on a first-come-first-serve basis. If both interfaces present frames at the same time, the express traffic is serviced first and then the preempt traffic. For details on presenting a frame for transmission, see the AXI4-Stream Interface section.

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 IEEE standard 802.3, section 79.3.7. The core assumes that the you 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 is active only after verification has been successfully completed. You can also disable verification in which case the core does not attempt to start the verification process and makes 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 are transmitted.
- If the express and preempt interface request frame transmission at the same time, the express frames are 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 following timing diagram depicts how preemption and interspersing works. For a detailed description of preemption and interspersing of express traffic, refer to IEEE Std 802.1CM-2018.

Figure 40: **Preemption and Interspersing**



***Note*:** The inter-frame gap (IFG) is not depicted in the previous diagram for compactness. However there is IFG which results in `tx_axis_mac_tready` being deasserted between frames.

Receive AXI4-Stream Interface

Receive AXI4-Stream Interface 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 Receive AXI4-Stream Interface.

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 AXI4-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 `tlast`assembly process; if the assembly process is not successful, it asserts `tuser` along with.

The following figure shows how the preempt frame fragments are presented on the AXI4-Stream interface when the FIFO is not inserted.

Figure 41: **Preempt Frame Fragments: When the FIFO Not Inserted**

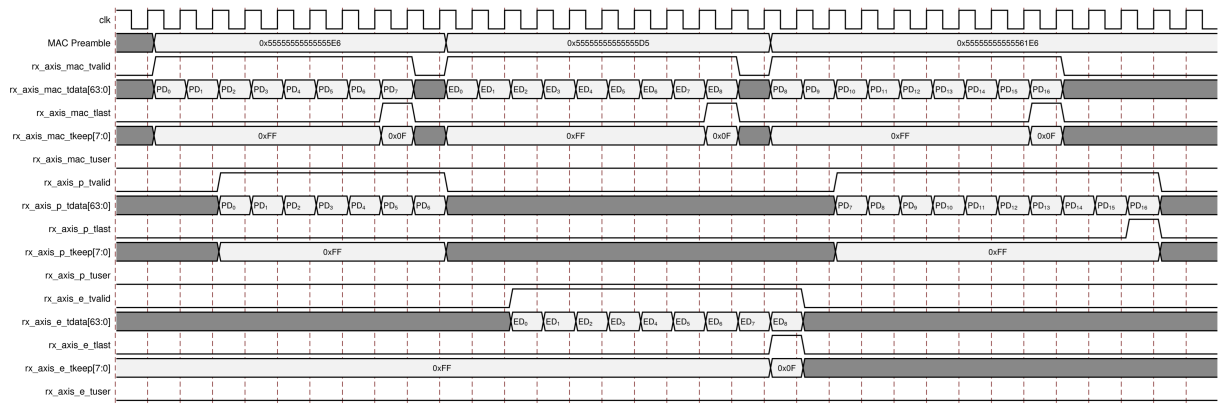


Table 45: **Control and Status Ports**

Name	I/O	Description	Clock Domain
CONTROL			
ctl_en_preempt	I	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	I	If asserted, preempt traffic is withheld.	tx_clk_out
ctl_disable_verify	I	If asserted, it inhibits the verification process.	tx_clk_out
ctl_restart_verify	I	A 0-to-1 transition will trigger Verification if ctl_disable_verify = 1'b0.	tx_clk_out
ctl_addfrag_size[1:0]	I	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]	I	Verification time-out value in milliseconds. Integer range 1-128. Default is 1 ms.	tx_clk_out
ctl_verify_limit[3:0]	I	Number of times core attempts Verification. Integer range 1-15. Default is 3.	tx_clk_out
p_frame_len_0	I	Indicates the length of the frame, in bytes, being presented at the TX Preempt AIX-S interface. Should be valid at preempt frame SOP.	tx_clk_out
STATUS			
stat_tx_mm_verify[1:0] (No Counter)	O	Indicates verification status. [0] – When asserted, indicates that verification is complete. [1] – When asserted, indicates that verification is successful. 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	O	Asserted when a preemptable packet (initial fragment or complete packet) is transmitted.	tx_clk_out
stat_tx_mm_fragment	O	Asserted when a continuation fragment of an preemptable packet is transmitted.	tx_clk_out
stat_tx_mm_hold	O	Asserted when ctl_hold_request transitions from 1'b0 to 1'b1.	tx_clk_out
stat_rx_mm_assembly_error	O	Asserted when errors are detected during fragment assembly.	rx_core_clk
stat_rx_mm_frame_smd_error	O	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
stat_rx_mm_frame_assembly_ok	O	Asserted when all the preemptable frame fragments have been assembled and presented.	rx_core_clk
stat_rx_mm_fragment	O	Asserted when a fragment frame is received.	rx_core_clk
rx_p_frm_drop_0	Output	Asserted by the packet assembly FIFO when it cannot continue with preempt fragment assembly because its full. When this happens all the fragments of the preempt frame received so far will be discarded.	rx_core_clk

Table 45: **Control and Status Ports** *(cont'd)*

Name	I/O	Description	Clock Domain
rx_p_frm_drop_count_0	Output	Count of number of such events.	rx_core_clk

Notes:

1. Timestamp ports description for express and preempt interface is similar to 1588v2 ports description. For more details, see Port Descriptions .

Status/Control Interface

The Status/Control interface allows you to set up the 1G/10G/25G Ethernet Subsystem 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 1G/10G/25G 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 1G/10G/25G Ethernet Subsystem automatically calculates and adds the FCS to the packet. For priority pause packets the 1G/10G/25G Ethernet Subsystem 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 1G/10G/25G Ethernet Subsystem 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 1G/10G/25G Ethernet Subsystem 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 1G/10G/25G Ethernet Subsystem 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 1 passes, the packet is checked to determine if it is a global pause packet.
3. If step 2 fails, the packet is checked to determine if it is a priority pause packet.

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

```
assign da_match_gcp = (!ctl_rx_check_mcast_gcp && !ctl_rx_check_ucast_gcp)
|| ((DA
== ctl_rx_pause_da_ucast) && ctl_rx_check_ucast_gcp) || ((DA ==
48'h0180c2000001) &&
ctl_rx_check_mcast_gcp);
assign sa_match_gcp = !ctl_rx_check_sa_gcp || (SA == ctl_rx_pause_sa);
assign etype_match_gcp = !ctl_rx_check_etype_gcp || (ETYPE ==
ctl_rx_etype_gcp);
assign opcode_match_gcp = !ctl_rx_check_opcode_gcp || ((OPCODE >=
ctl_rx_opcode_min_gcp) && (OPCODE <= ctl_rx_opcode_max_gcp));
assign global_control_packet = da_match_gcp && sa_match_gcp &&
etype_match_gcp &&
opcode_match_gcp && ctl_rx_enable_gcp;
assign da_match_pcp = (!ctl_rx_check_mcast_pcp && !ctl_rx_check_ucast_pcp)
|| ((DA
== ctl_rx_pause_da_ucast) && ctl_rx_check_ucast_pcp) || ((DA ==
ctl_rx_pause_da_mcast) && ctl_rx_check_mcast_pcp);
assign sa_match_pcp = !ctl_rx_check_sa_pcp || (SA == ctl_rx_pause_sa);
assign etype_match_pcp = !ctl_rx_check_etype_pcp || (ETYPE ==
ctl_rx_etype_pcp);
assign opcode_match_pcp = !ctl_rx_check_opcode_pcp || ((OPCODE >=
ctl_rx_opcode_min_pcp) && (OPCODE <= ctl_rx_opcode_max_pcp));
assign priority_control_packet = da_match_pcp && sa_match_pcp &&
etype_match_pcp &&
opcode_match_pcp && ctl_rx_enable_pcp;
assign control_packet = global_control_packet || priority_control_packet;
```

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

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

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

```
assign sa_match_gpp = !ctl_rx_check_sa_gpp || (SA == ctl_rx_pause_sa);
assign etype_match_gpp = !ctl_rx_check_etype_gpp || (ETYPE ==
ctl_rx_etype_gpp);
assign opcode_match_gpp = !ctl_rx_check_opcode_gpp || (OPCODE ==
ctl_rx_opcode_gpp);
assign global_pause_packet = da_match_gpp && sa_match_gpp &&
etype_match_gpp &&
opcode_match_gpp && ctl_rx_enable_gpp;
```

where DA is the destination address, SA is the source address, OPCODE is the opcode and ETYPE is the ethertype/length field that 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 core stays in step 1. Otherwise, the corresponding bit of the `stat_rx_pause_req[8:0]` bus is set to 1, and the received quanta is loaded into a timer.

   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.

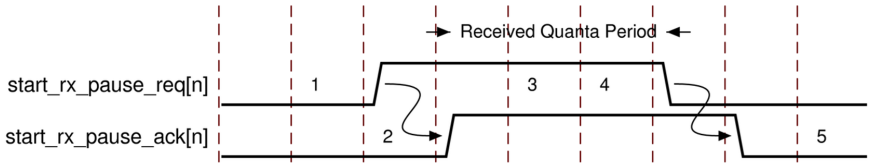
3. After you set the proper bit of `ctl_rx_pause_ack[8:0]` to 1, or if `ctl_rx_check_ack` is 0, the core starts counting down the timer.

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 the following figure with each step shown on the waveform.

Figure 42: **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.

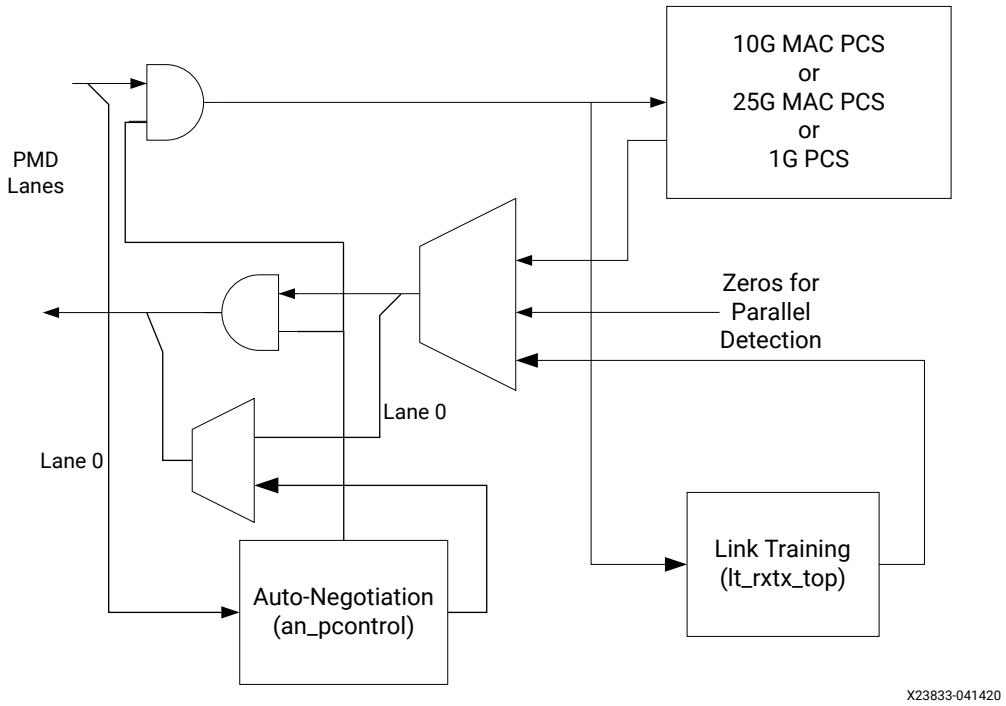
***Note*:** Transmitter MAC should not transmit frames/packets when it receives Pause frames from the receiver until the time duration specified in the Pause timer.

Auto-Negotiation

Auto-Negotiation (Clause 73)

A block diagram of the 1G/10G/25G Ethernet Subsystemwith Auto-Negotiation (AN) with Parallel Detection (PD) is shown in the following figure. The Parallel Detection is done inside the AN RTL, which is encrypted. As a result of PD, only the AN is resolved to 1G speed.

Figure 43: **Core with Auto-Negotiation**

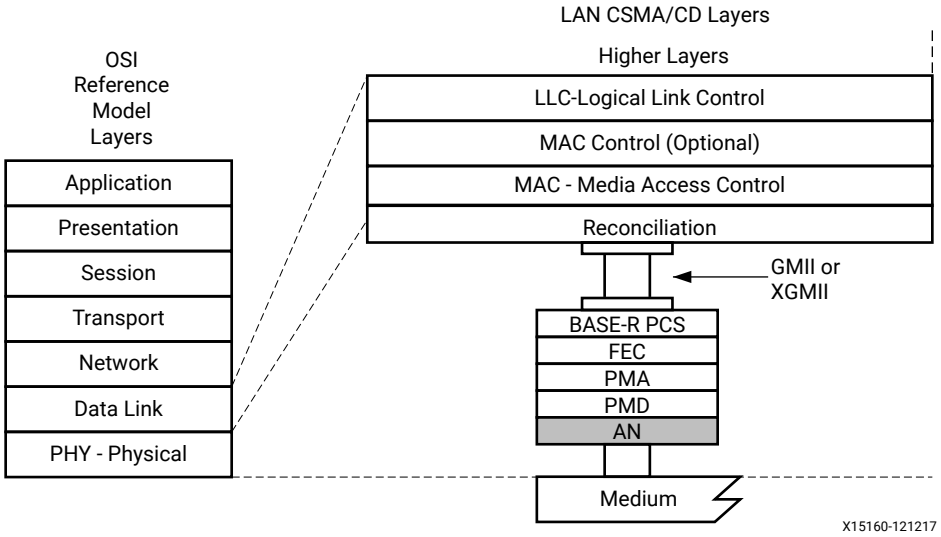


The auto-negotiation function allows an Ethernet device to advertise the modes of operation it possesses to another device at the remote end of a backplane Ethernet link and to detect corresponding operational modes the other device might be advertising. The objective of this auto-negotiation function 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.When Auto-Negotiation is complete, the datapath is switched to mission mode (the PCS), as shown in the preceding figure.

Overview

The following figure shows the position of the auto-negotiation function in the OSI reference model.

Figure 44: **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.

Auto-Negotiation 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.*

Auto-Negotiation (Clause 37)

This section provides general guidelines for using the Auto-Negotiation (Clause 37) function of the core. Auto-Negotiation is controlled and monitored through the PCS management registers.

Overview of Operation

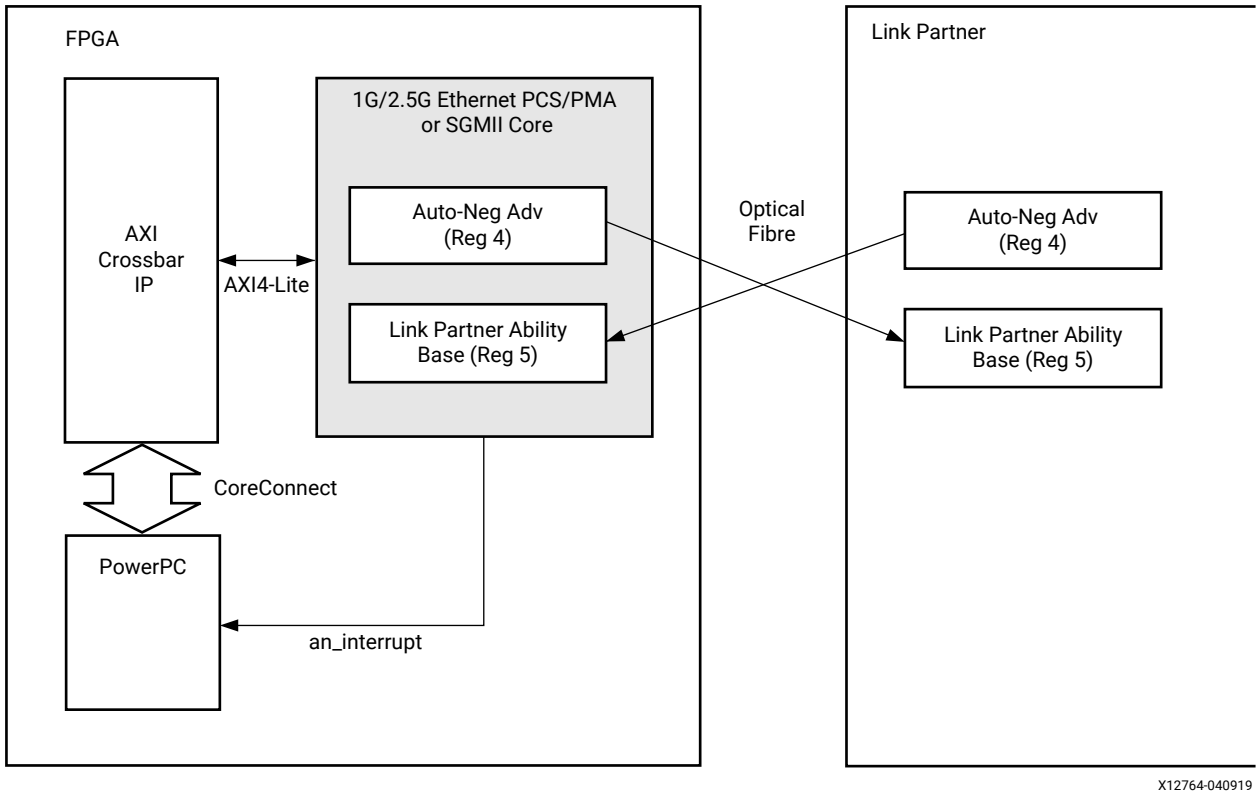
For either standard, when considering Auto-Negotiation between two connected devices, it must be remembered that:

- Auto-Negotiation must be either enabled in *both* devices, or
- Auto-Negotiation must be disabled in *both* devices.

1000BASE-X Standard

IEEE 802.3 Clause 37 describes the 1000BASE-X Auto-Negotiation function that allows a device to advertise the modes of operation that it supports to a device at the remote end of a link segment (the link partner) and to detect corresponding operational modes that the link partner advertises. The following figure shows the operation of 1000BASE-X Auto-Negotiation.

Figure 45: **1000BASE-X Auto-Negotiation Overview**



The following describes typical operation when Auto-Negotiation is enabled.

1. Auto-Negotiation starts automatically when any of the following conditions are met:
   - Power-up/reset.
   - Upon loss of synchronization.
   - The link partner initiates Auto-Negotiation.
   - An Auto-Negotiation restart is requested (see Register 0: Control Register).
2. During Auto-Negotiation, the contents of the Auto-Negotiation Advertisement register are transferred to the link partner.

   This register is writable through AXI4-Lite, therefore enabling software control of the systems advertised abilities. See Register 4: Auto-Negotiation Advertisement for more information.

   Information provided in this register includes:

   - Fault Condition signaling
   - Duplex Mode
   - Flow Control capabilities for the attached Ethernet MAC

3. The advertised abilities of the Link Partner are simultaneously transferred into the Auto-Negotiation Link Partner Ability Base register.

   This register contains the same information as in the Auto-Negotiation Advertisement register. See Register 5: Auto-Negotiation Link Partner Base for more information. Remote Fault and pause status bits of this register are also provided in `status_vector`.

4. Under normal conditions, this completes the Auto-Negotiation information exchange. It is now the responsibility of system management (for example, software running on an embedded PowerPC or MicroBlaze processor) to complete the cycle. The results of the Auto-Negotiation should be read from the Auto-Negotiation Link Partner Ability Base register or by reading the `remote_fault` and pause status bits of `status_vector` if the AXI4-Lite interface is not present.

   There are two methods that a host processor uses to learn of the completion of an Auto-Negotiation cycle:

   - Polling the Auto-Negotiation completion bit 1.5 in the Status register (Register 1).
   - Using the Auto-Negotiation interrupt port of the core (see Using the Auto-Negotiation Interrupt).

Related Information

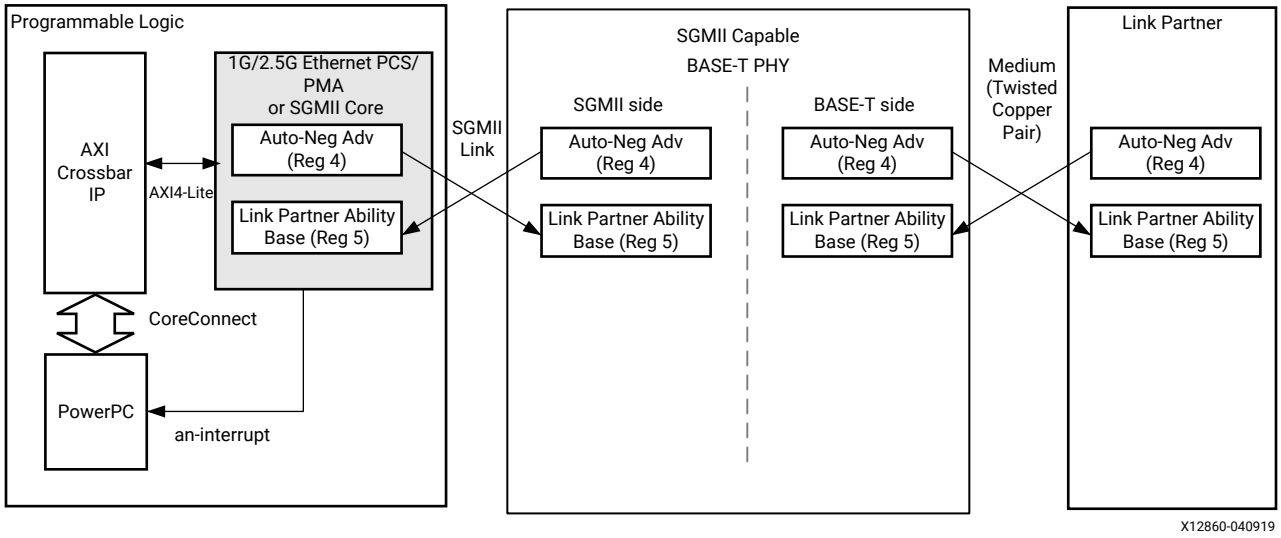
Configuration and Status Register Map for 1G Ethernet PCS/PMA
Using the Auto-Negotiation Interrupt

SGMII Standard

The following figure shows the operation of SGMII Auto-Negotiation as described in Overview of Operation.

Additional information about SGMII Standard Auto-Negotiation is provided in the following sections.

Figure 46: **SGMII Auto-Negotiation**



The SGMII capable PHY has two distinctive sides to Auto-Negotiation.

- The PHY performs Auto-Negotiation with its link partner using the relevant Auto-Negotiation standard for the chosen medium (BASE-T Auto-Negotiation is shown in the preceding figure, using a twisted copper pair as its medium). This resolves the operational speed and duplex mode with the link partner.
- The PHY then passes the results of the Auto-Negotiation process with the link partner to the core (in SGMII mode), by leveraging the 1000BASE-X Auto-Negotiation specification described in the 1000BASE-X Standard topic. This transfers the results of the Link Partner Auto-Negotiation across the SGMII and is the only Auto-Negotiation observed by the core.

This SGMII Auto-Negotiation function, summarized previously, leverages the 1000BASE-X PCS/PMA Auto-Negotiation function but contains two differences.

- The duration of the Link Timer of the SGMII Auto-Negotiation is shrunk from 10 ms to 1.6 ms so that the entire Auto-Negotiation cycle is much faster
- The information exchanged is different and now contains speed resolution in addition to duplex mode. See Register 5: Auto-Negotiation Link Partner Base in Configuration and Status Register Map for 1G Ethernet PCS/PMA. Speed and Duplex status bits of this register are also provided in status_vector.
- There are no other differences and dealing with the results of Auto-Negotiation can be handled as described previously in the 1000BASE-X Standard topic.

Related Information

Configuration and Status Register Map for 1G Ethernet PCS/PMA
1000BASE-X Standard

Using the Auto-Negotiation Interrupt

The Auto-Negotiation function has an an_interrupt port. This is designed to be used with common microprocessor bus architectures (for example, the CoreConnect bus interfacing to a MicroBlaze™ processor).

The operation of this port is enabled or disabled and cleared through the AXI4-Lite Register 16, the Vendor-specific Auto-Negotiation Interrupt Control register.

- When disabled, the port is permanently tied to logic 0.
- When enabled, this port is set to logic 1 following the completion of an Auto-Negotiation cycle. It remains High until it is cleared by writing 0 to bit 16.1 (Interrupt Status bit) of Register 16: Vendor-Specific Auto-Negotiation Interrupt Control.

Related Information

Configuration and Status Register Map for 1G Ethernet PCS/PMA

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.

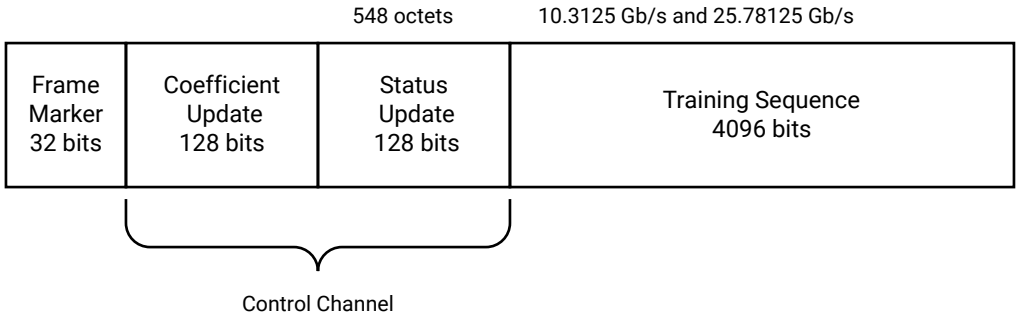
The other function of the core is to communicate training feedback from the receiver to the corresponding transmitter so that its equalizer circuit (not part of the core) can be adjusted as required. The two circuits comprising the core are the receive Link Training block and the transmit Link Training 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 the following figure.

Figure 47: **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. The following tables define these bits in the order in which they are transmitted starting with bit 15 and ending with bit 0.

Table 46: **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.

Table 46: **Coefficient and Update Field Bit Definitions** *(cont'd)*

Bits	Name	Description
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 47: **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	1 1 = maximum 0 1 = updated 1 0 = minimum 0 0 = not_updated
3:2	Coefficient (0) update	1 1 = maximum 0 1 = updated 1 0 = minimum 0 0 = not_updated
1:0	Coefficient (-1) update	1 1 = maximum 0 1 = updated 1 0 = minimum 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 Auto-Negotiation Ports.

Related Information

Auto-Negotiation Ports

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 the Transmit topic and signal names defined in Port Descriptions.

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.

Related Information

Transmit
Port Descriptions

Board Testing Steps for Auto-Negotiation and Link Training Using AXI4-Lite Interface

1. Enable the Abilities register, CONFIGURATION_AN_ABILITY (0x00F8), as per the core configuration or the abilities you want to advertise. For example, write the value 0x1E06 to address 0x00F8.

2. Read the CONFIGURATION_AN_CONTROL_REG1 register (0x00E0), based on the requirement you can enable or bypass the auto-negotiation. If auto-negotiation is enabled than you need to write the nonce seed value into the same register. For example, write the value 0x16D to address 0x00E0.
3. Enable the Link Training option by setting the link training enable signal in the CONFIGURATION_LT_CONTROL_REG1 (0x0100) register. For example, write the value 0x1 to address 0x0100.
4. Write the CONFIGURATION_LT_SEED_REG0 register (0x0110) with some seed value. For example, write the value 0x0605 to address 0x0110.
5. Write the CONFIGURATION_LT_COEFFICIENT_REG0 register (0x0130) with some coefficient values for the place holder logic. For example, write the value 0x540 to address 0x0130.
6. Issue `sys_reset`/write `1'b1` to `ctl_an_reset` (bit 28 of address 0x0004) so that the auto-negotiation block takes the updated nonce seed value into consideration.
7. Keep reading the `stat_an_autoneg_complete` (bit 2 of address 0x0458) which indicates the successful completion of the auto-negotiation and link training.

If the link partner sends next pages, the `ctl_an_lo_np_ack` signal must be set. This port can be tied High.

Design Flow Steps

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

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
- *Vivado Design Suite User Guide: Designing with IP* (UG896)
- *Vivado Design Suite User Guide: Getting Started* (UG910)
- *Vivado Design Suite User Guide: Logic Simulation* (UG900)

Customizing and Generating the Subsystem

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 subsystem in the Vivado IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) 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) and *Vivado Design Suite User Guide: Getting Started* (UG910).

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 provides the basic core configuration options. Default values are pre-populated in all fields.

Figure 48: **Configuration Tab (Versal)**

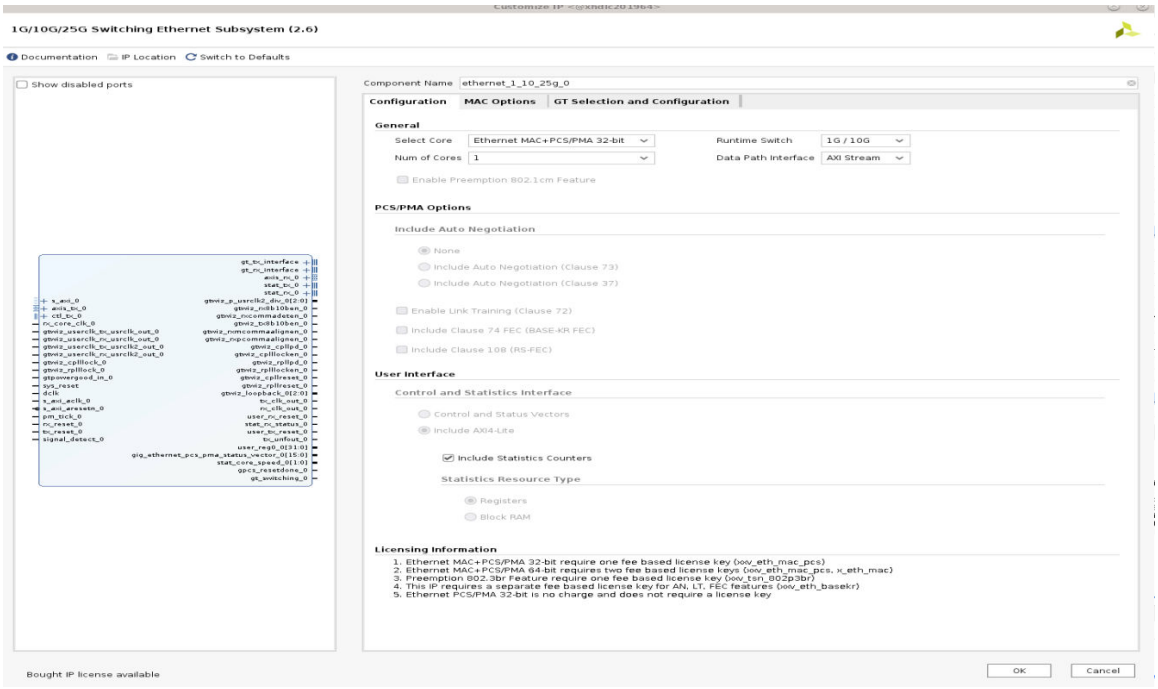


Figure 49: **Configuration Tab (UltraScale/UltraScale+)**

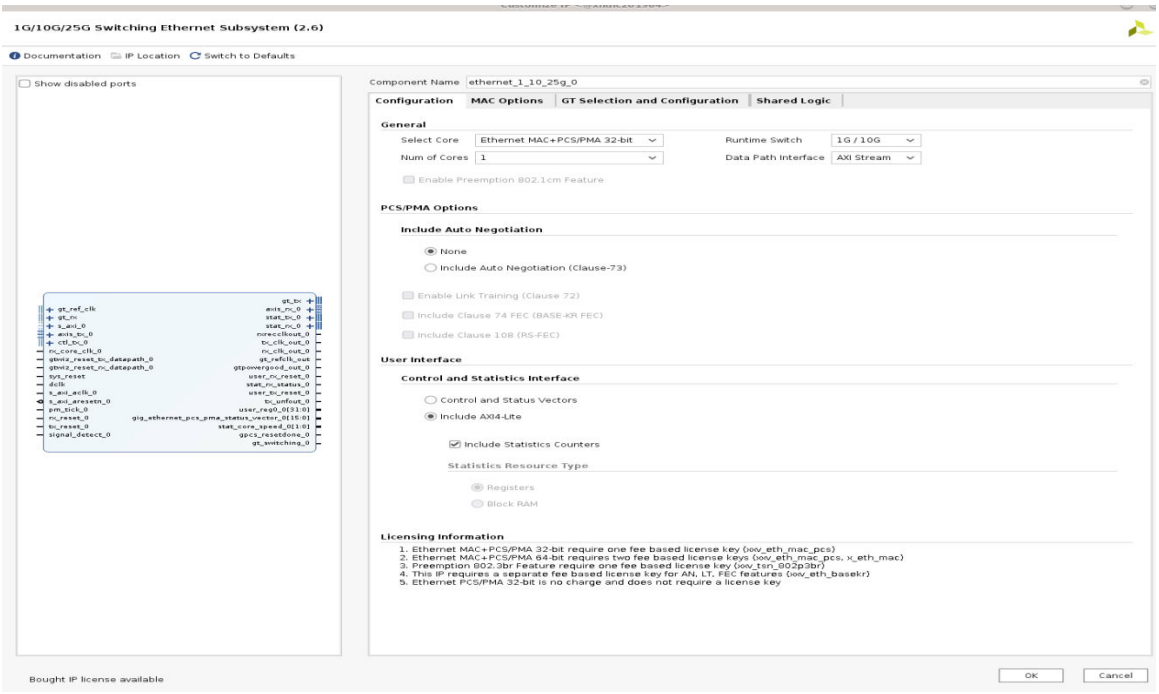


Table 48: **Configuration Options**

Option	Values	Default
General		
Select Core	Ethernet PCS/PMA 32-bit Ethernet MAC+PCS/PMA 32-bit Ethernet MAC+PCS/PMA 64-bit	Ethernet PCS/PMA 32-bit
Runtime Switch	1G/10G/25G	1G/10G
Num of Cores [8]	1 2 3 4	1
Data Path Interface	AXI4-Stream [1] Media Independent Interface (MII) [2]	AXI4-Stream
Enable Preemption 802.3cm Feature	0, 1	0
PCS/PMA Options		
Include Auto Negotiation[9]	None Include Auto Negotiation (Clause-73)[34] Include Auto Negotiation (Clause-37)[5]	None
Enable Link Training (Clause 72)[9]	0,1	0
Include Clause 74 FEC (Clause 72)[9]	0,1	0
Include Clause 108 (RS-FEC)[10]	0, 1	0

Table 48: **Configuration Options** *(cont'd)*

Option	Values	Default
Control and Statistics Interface		
Control and Statistics interface	Control and Status Vectors Include AXI4-Lite	Include AXI4-Lite
Include Statistics Counters [7]	0,1	1
Statistics Resource Type	Registers, Block RAM[6]	Register

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. This feature is visible only for the Ethernet MAC+PCS/PMA 64 bit core configuration.
4. This feature is visible only for the Ethernet MAC+PCS/PMA 32- bit core configuration.
5. This feature is visible only for the PCS/PMA 32-bit core configuration.
6. This feature is not yet supported.
7. Default size of the statistics counter is 48-bit. To reconfigure this statistics counter, a hidden GUI parameter "STATISTICS_COUNTERS_SIZE" is provided. You can generate the IP using this parameter from the Vivado tcl console. Valid values are 32 to 48.
8. The Number of Cores will be 1 only for the Versal devices and multi-core support will not be supported as GT will always be in the example design/outside the core.
9. This support is available for non-Versal devices only.
10. This feature is visible only for the Ethernet MAC+PCS/PMA 64-bit core configuration for:
    a. Disabled Auto-Negotiation feature or,
    b. Disabled FEC logic.

MAC Options Tab

The MAC Options tab provides additional core configuration options.

Figure 50: **MAC Options Tab (Versal)**

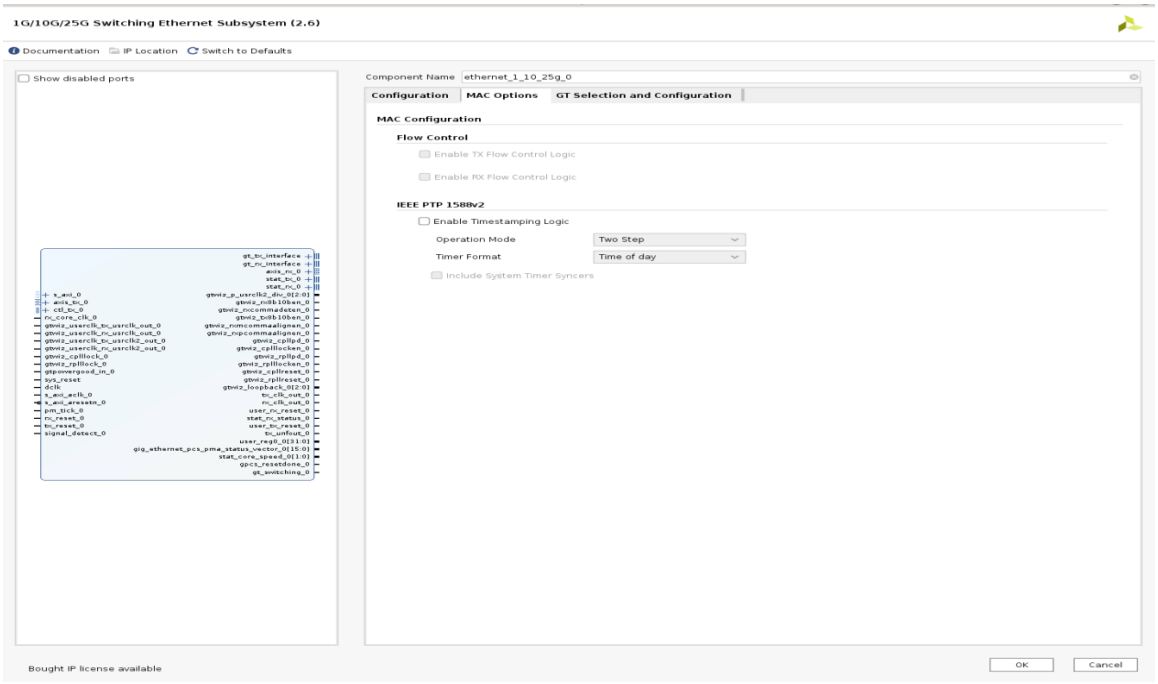


Figure 51: **MAC Options Tab (UltraScale/UltraScale+)**

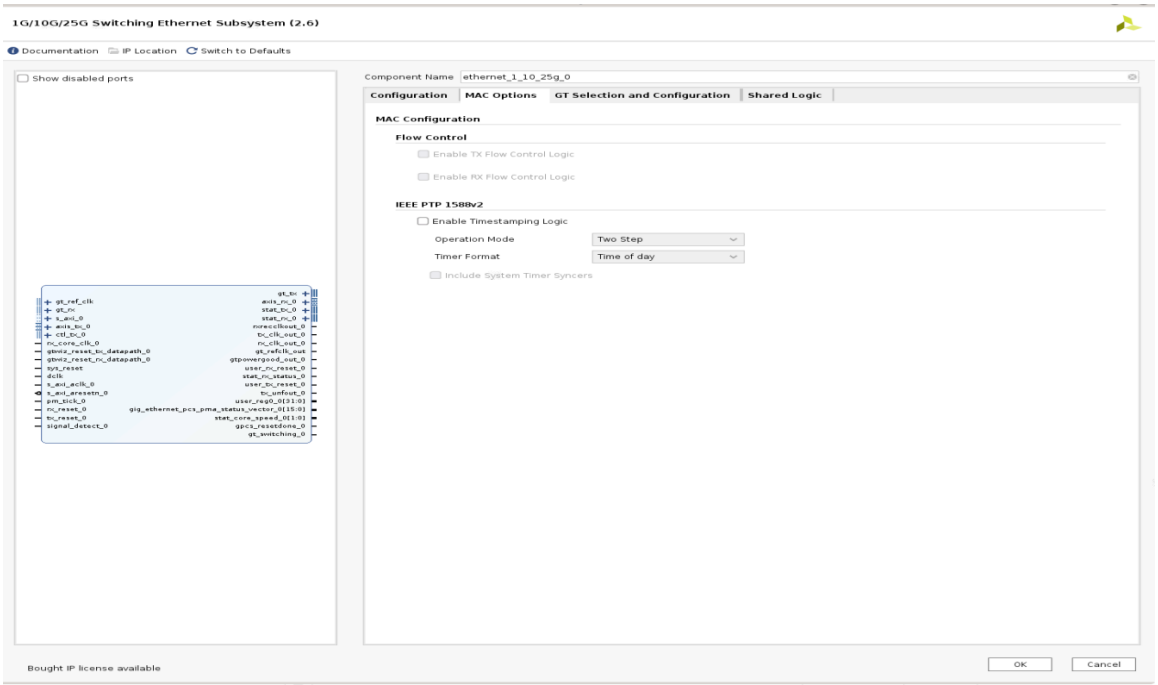


Table 49: **MAC Options**

Option	Values	Default
Flow Control[1]		
Enable TX Flow Control Logic	Checked, Unchecked	Unchecked
Enable RX Flow Control Logic	Checked, Unchecked	Unchecked
IEEE PTP 1588v2[2]		
Enable Timestamping Logic	Checked, Unchecked	Unchecked
Operation Mode	One Step Two Step	Two Step
Timer Format	Time of day, Correction Field Format	Time of day
Include System Timer Syncers [3]	0, 1	0

Notes:

1. This feature is visible only for the Ethernet MAC+PCS/PMA 64-bit core configuration.
2. This feature is visible only for the Ethernet MAC+PCS/PMA 32-bit core configuration.
3. This option is available when Enable Timestamping Logic is selected. This will include the timer syncer modules inside the IP.

GT Selection and Configuration Tab

The GT Selection and Configuration tab enables you to configure the serial transceiver features of the core.

Figure 52: **GT Selection and Configuration Tab (Versal)**

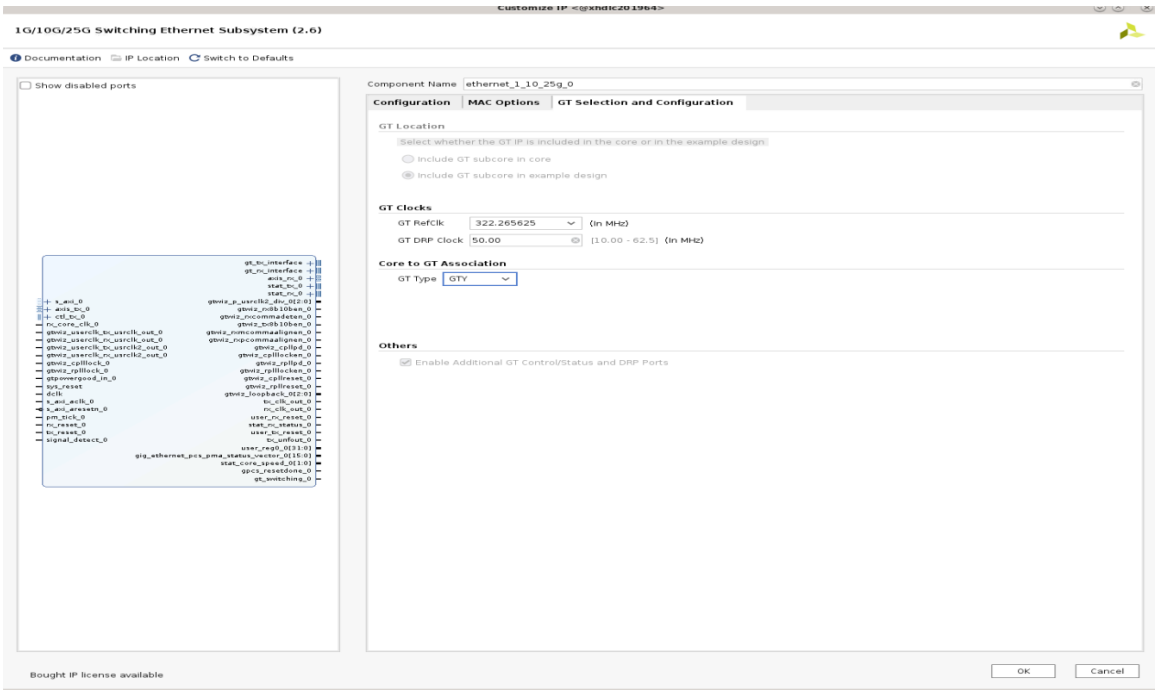


Figure 53: **GT Selection and Configuration Tab (UltraScale/UltraScale+)**

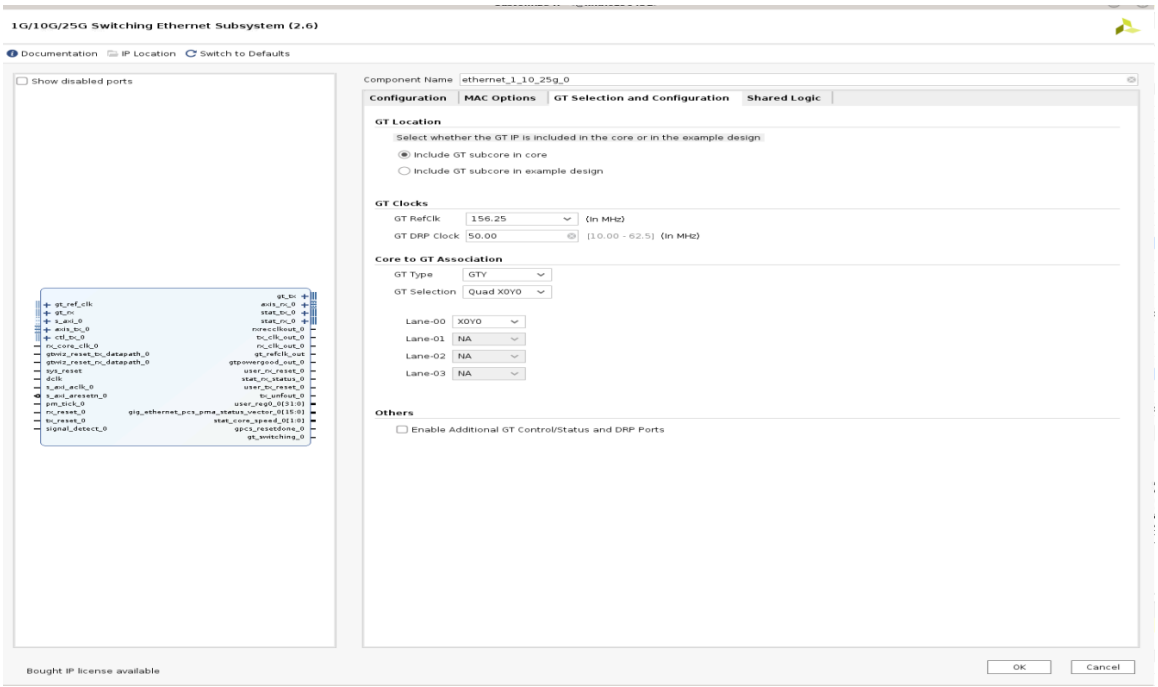


Table 50: **GT Clocks Options**

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[2]		
GT RefClk (In MHz) [1]	156.25	156.25
GT DRP Clock (In MHz)	10 – 62.5	50.00
Core to GT Association		
GT Type	GTY GTH GTM	GTH
GT Selection	Options based on device/package Quad groups. For example: Quad X0Y1 Quad X0Y2 Quad X0Y3 ...	Quad X0Y1

Table 50: **GT Clocks Options** *(cont'd)*

Option	Values	Default
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 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.
2. Non-Versal devices support 156.25 frequency only and Versal devices support 322.265625 frequency only.

Shared Logic Tab

The Shared Logic tab enables you to use shared logic in either the core or the example design.

Note: The Shared Logic tab is available only for UltraScale/UltraScale+ devices.

Figure 54: **Shared Logic Tab (UltraScale/UltraScale+)**

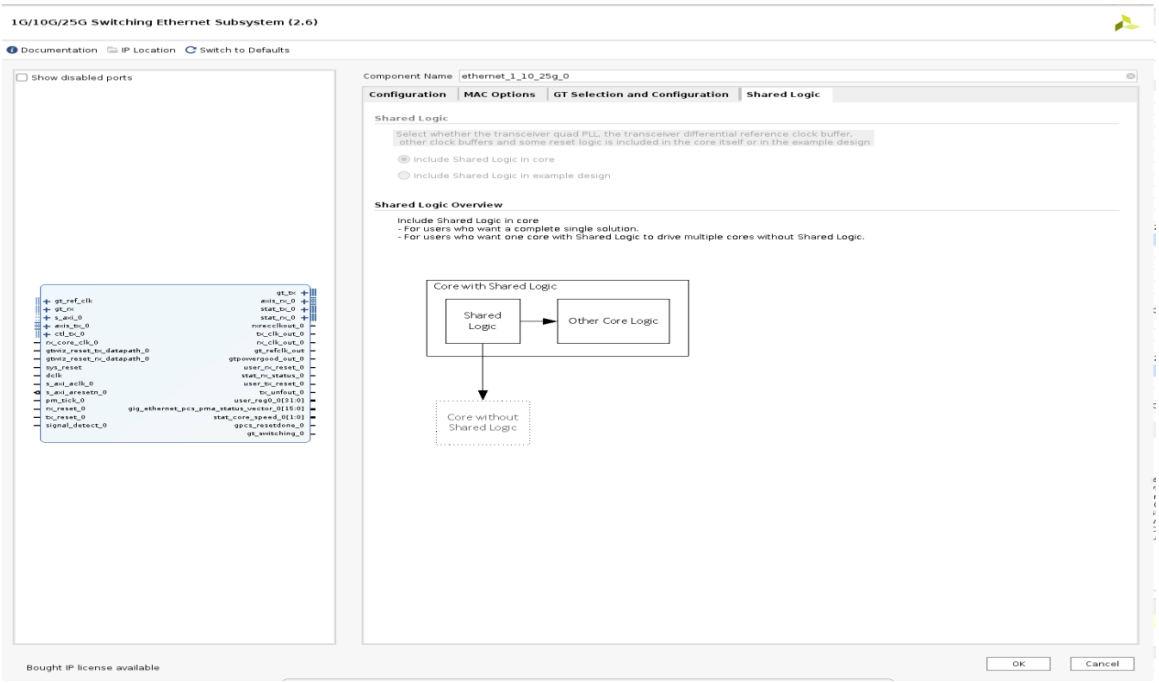


Table 51: **Shared Logic Options**

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).

Constraining the Subsystem

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

Required Constraints

The 1G/10G/25G 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 1G/10G/25G 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). Constraints provided in the 1G/10G/25G 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 IP subsystem.

Clock Frequencies

This section is not applicable for this subsystem.

Clock Management

This section is not applicable for this IP subsystem.

Clock Placement

This section is not applicable for this subsystem.

Banking

This section is not applicable for this IP subsystem.

Transceiver Placement

This section is not applicable for this IP subsystem.

I/O Standard and Placement

This section is not applicable for this IP subsystem.

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).

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.

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).

XILINX

Chapter 6

Example Design

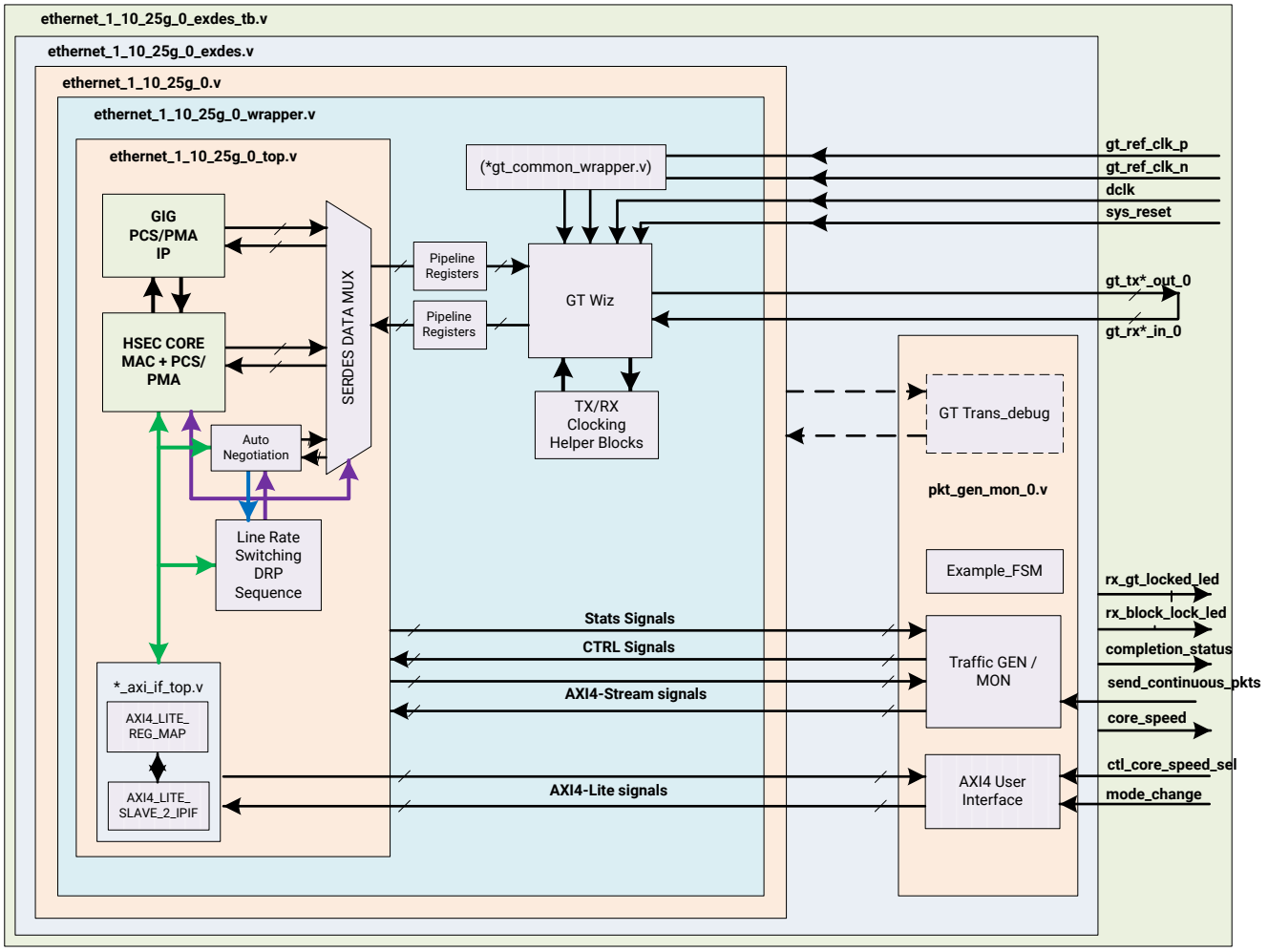
This chapter contains information about the example design provided in the Vivado® Design Suite when using the Vivado Integrated Design Environment (IDE).

Overview

The following figure shows the instantiation of various modules and their hierarchy for a single core configuration of the ethernet_1_10_25g example design for a 32-bit MAC and PCS/PMA core when the GT (serial transceiver) is inside the IP core. (Serial Transceiver will always be a part of the example design for Versal ACAP).

Clocking helper blocks are used to generate the required clock frequency for the core.

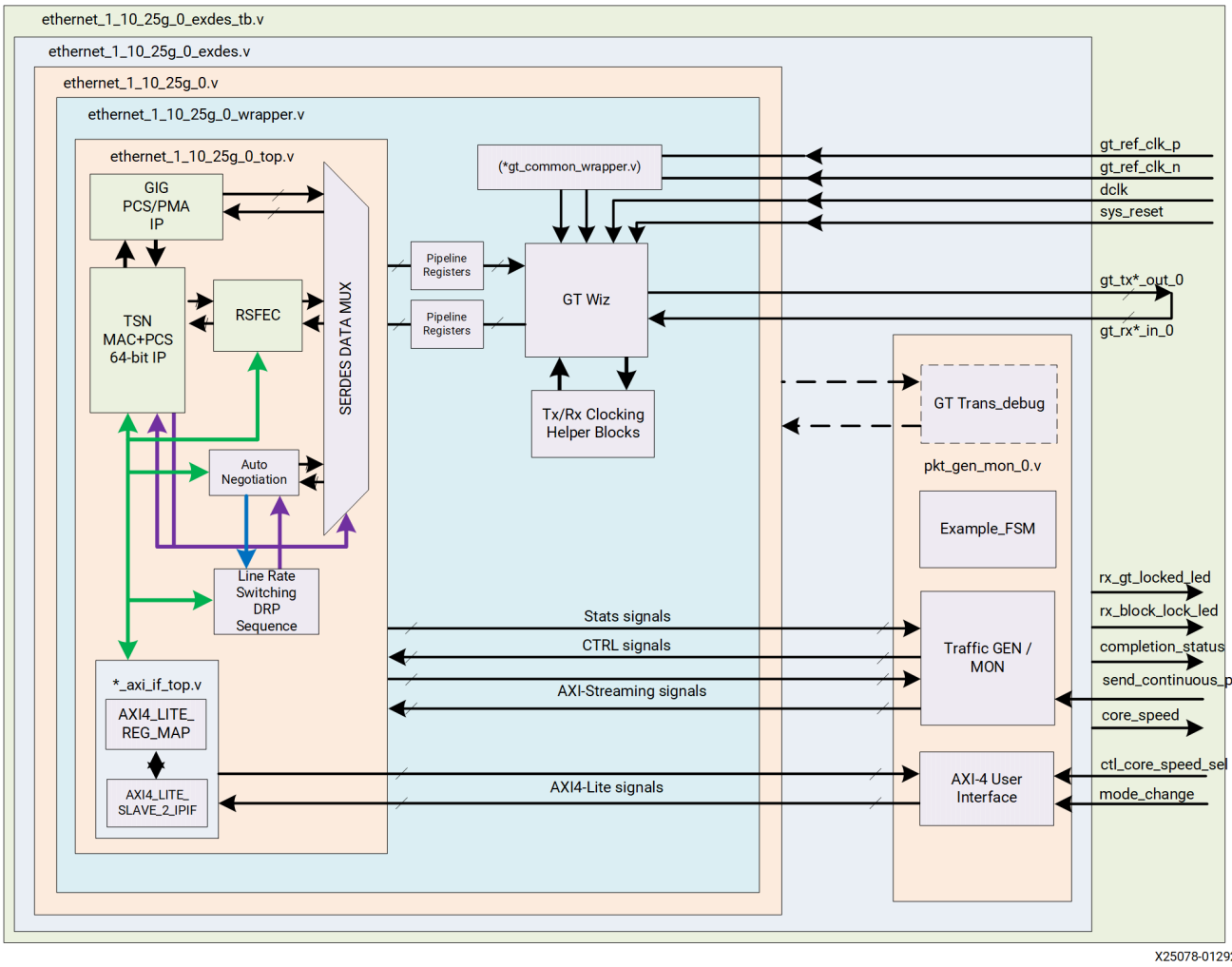
Figure 55: **32-Bit MAC and PCS/PMA Single Core Example Design Hierarchy**



X20625-012821

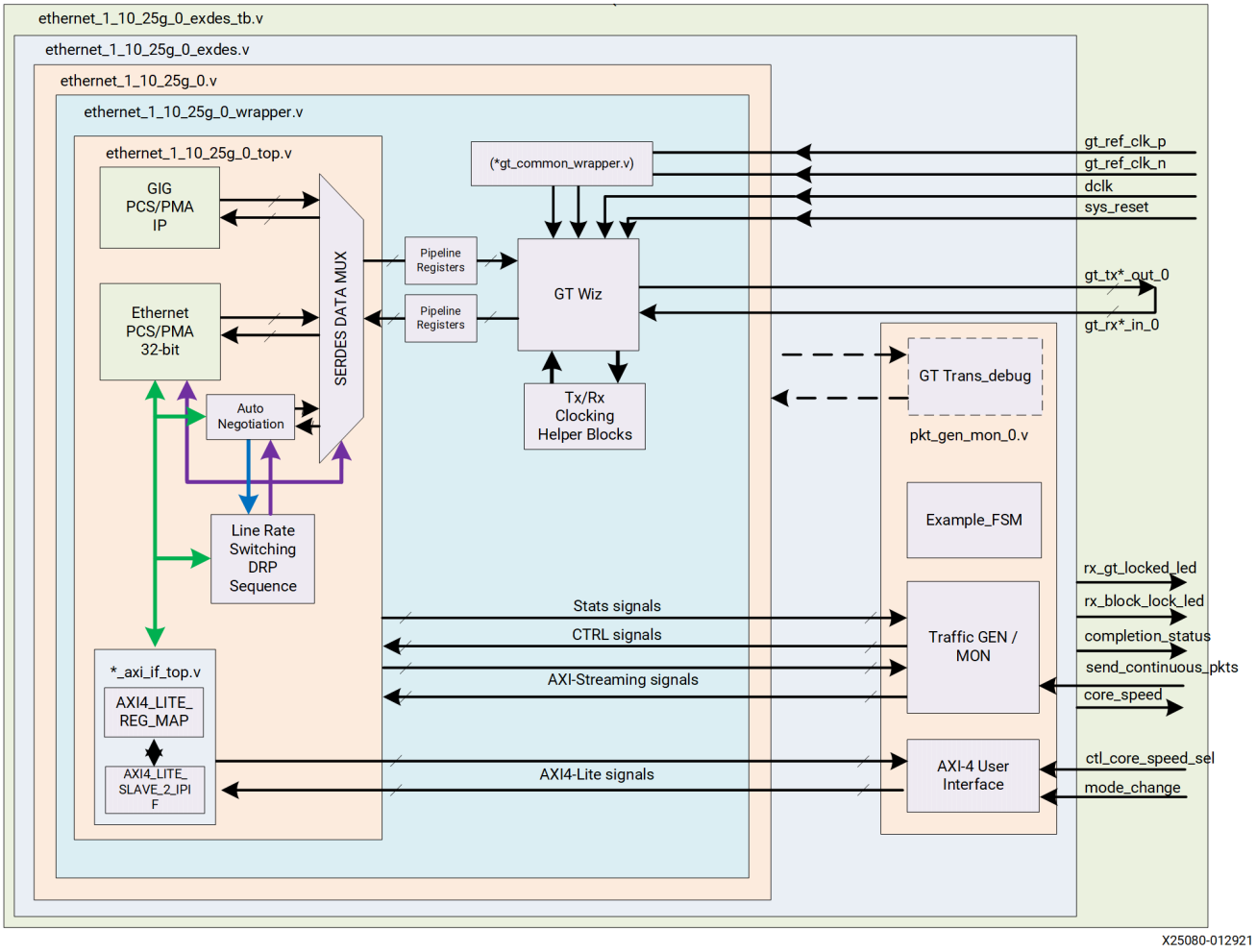
The following figure shows the instantiation of various modules and their hierarchy for a single core configuration of the ethernet_1_10_25g_0 example design for a 64-bit MAC and PCS/PMA core configuration.

Figure 56: **64-Bit MAC and PCS/PMA Single Core Example Design Hierarchy**



The following figure shows the instantiation of various modules when their hierarchy for a single core configuration of ethernet_1_10_25g_0 example design for a 32-bit PCS/PMA core.

Figure 57: **32-Bit PCS Only Single Core Example Design Hierarchy**



The following user interfaces are available for different configurations.

- MAC/PCS configuration:
  - AXI4-Stream for datapath interface
  - AXI4-Lite for control and statistics interface
- PCS configuration:
  - XGMII interface
  - GMII interface
  - AXI4-Lite for control and statistics interface

The `ethernet_1_10_25g_0` module is used to generate the data packets for sanity testing. The packet generation and checking is controlled by a FSM module.

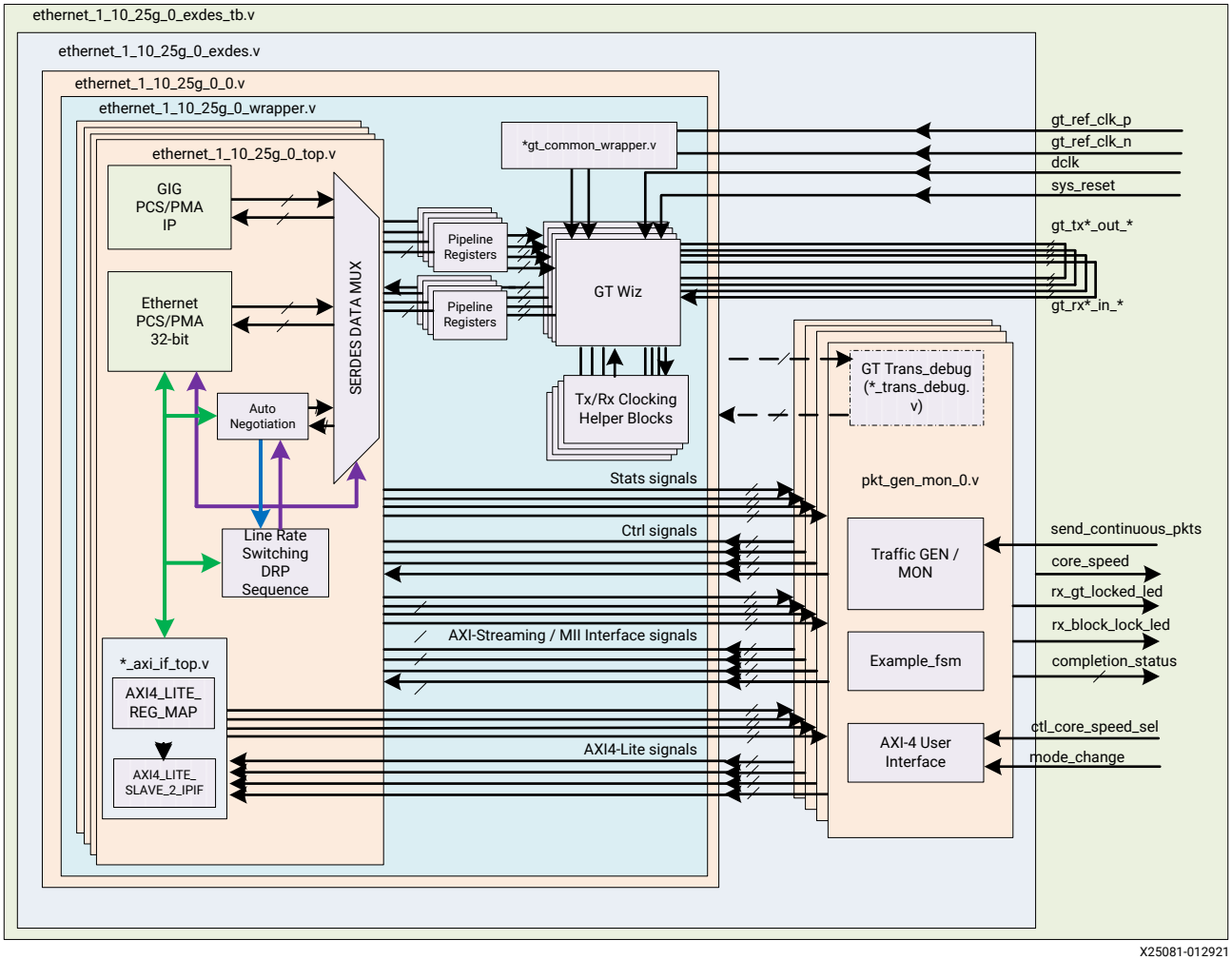
The optional modules are described as follows:

- **TX / RX pipeline register:** The TX pipeline register double synchronizes the data from the core to the GT with respect to the `tx_clk`. The RX pipeline register double synchronizes the data from the GT to the core with respect to the `rx_serdes_clk`.

***Note*:** If you select Auto-Negotiation in the Vivado IDE, the operation is only performed with the 10G data rate. After the Auto-Negotiation is complete, the core is switched to mission mode. For parallel detection, the core is switched to 1G data rate.

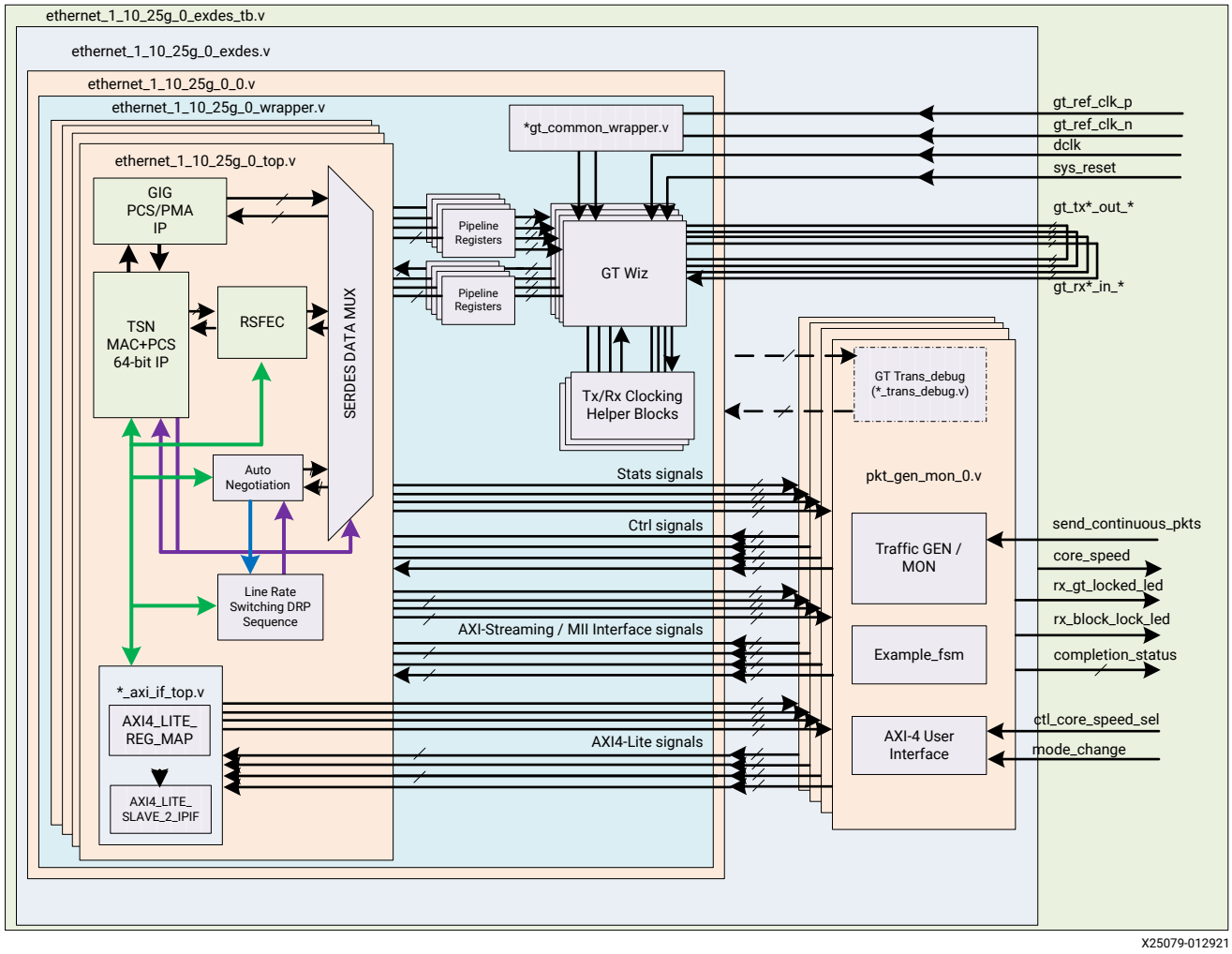
The following figure shows the instantiation of various modules and their hierarchy for multiple core configuration of the `ethernet_1_10_25g_0` example design for a 32-bit PCS/PMA core.

Figure 58: **32-Bit MAC and PCS/PMA Multiple Cores Example Design Hierarchy**



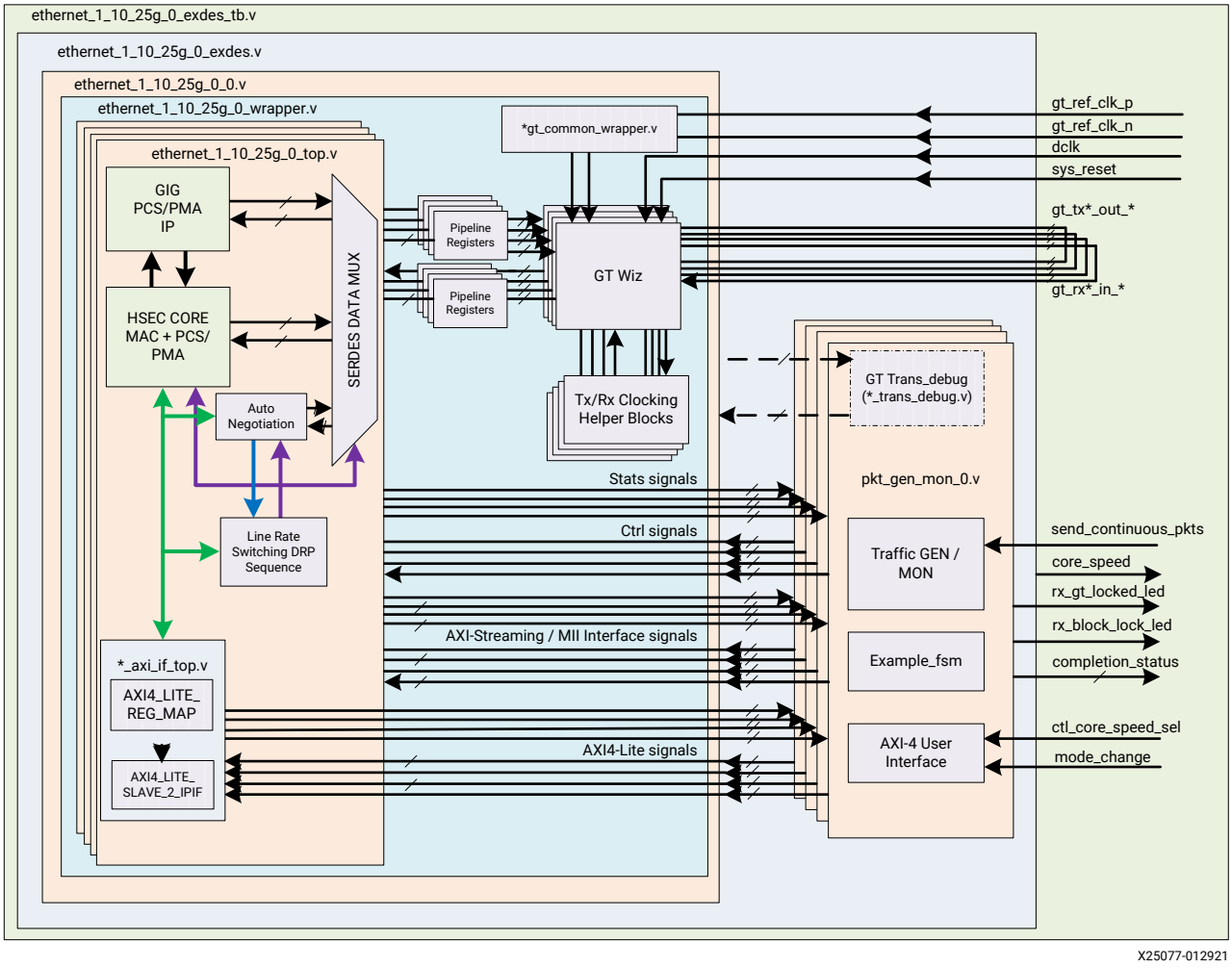
The following figure shows the instantiation of various modules and their hierarchy for the multiple core configuration of the ethernet_1_10_25g_0 example design for a 64-bit MAC and PCS/PMA core.

Figure 59: **64-Bit MAC and PCS/PMA Multiple Core Example Design Hierarchy**



The following figure shows the instantiation of various modules and their hierarchy for the multiple core configuration of the ethernet_1_10_25g_0 example design for a 32-bit PCS/PMA core.

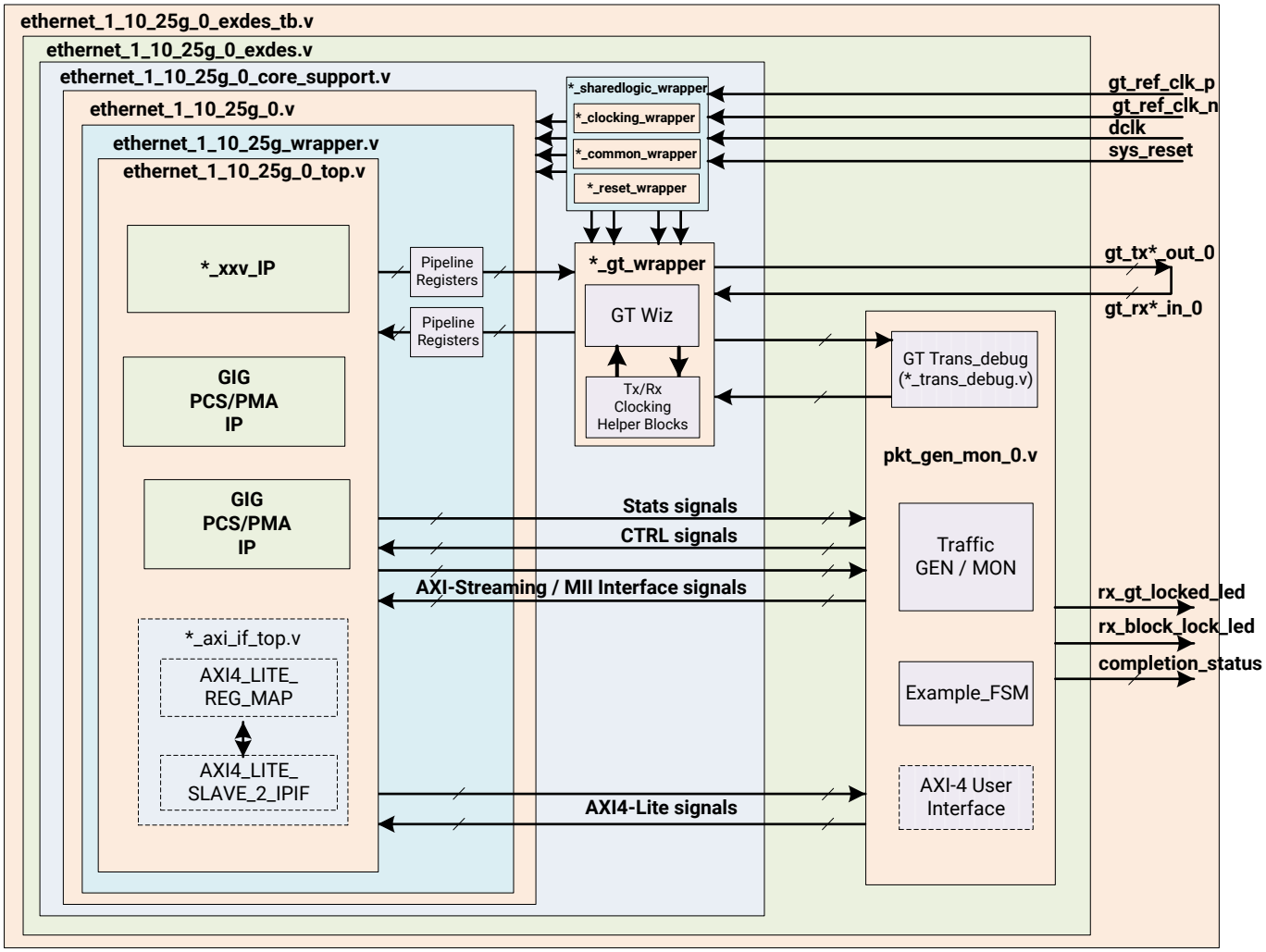
Figure 60: **32-Bit PCS/PMA Only Multiple Core Example Design Hierarchy**



Example Design Hierarchy (GT in Example Design)

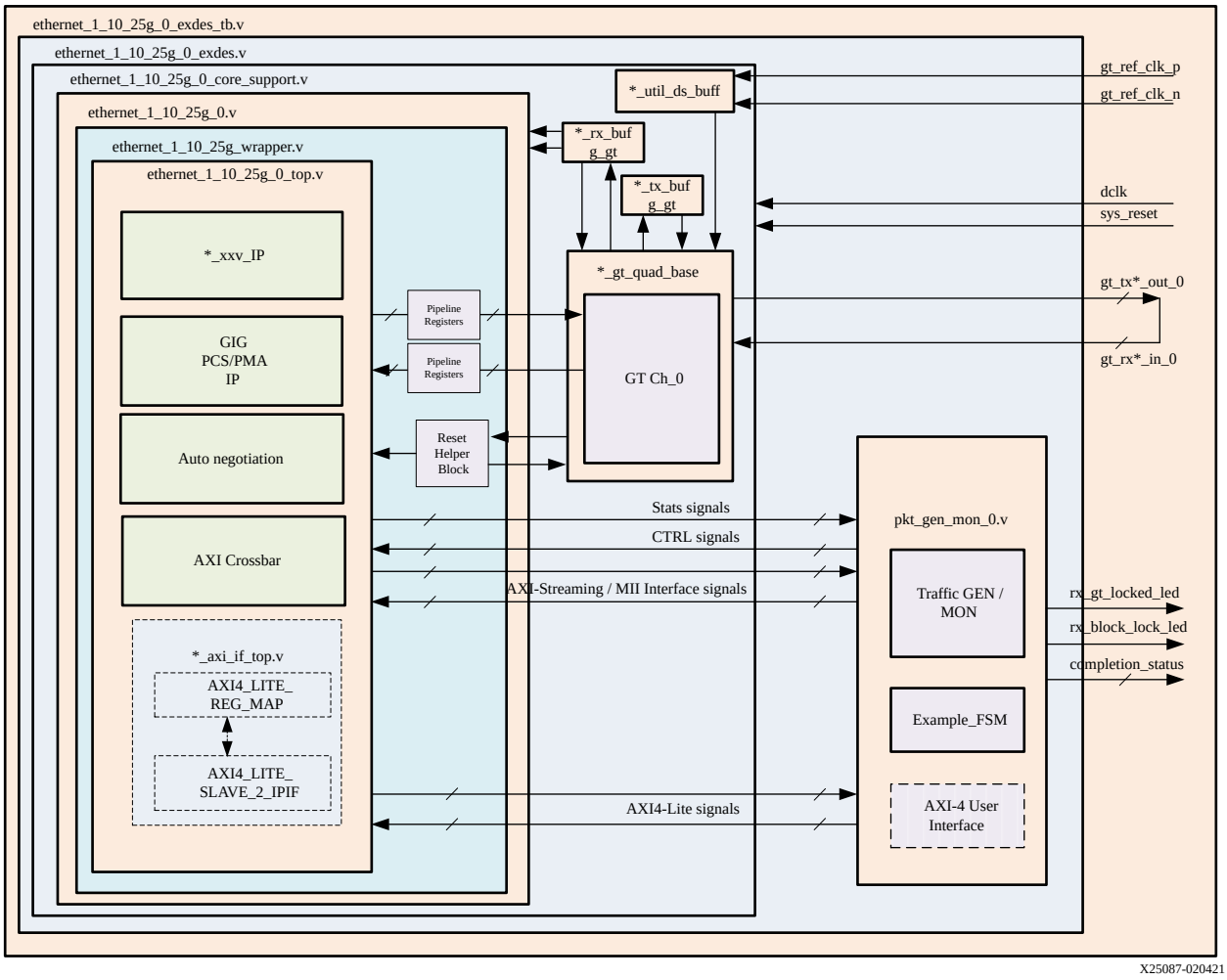
When the 1/10/25G Ethernet subsystem is added to the Vivado IP integrator, the Run Block Automation IP Core and GT (serial transceivers) get connected with some helper blocks as per the core configuration. There is a reset interface IP, internal to 1/10/25G Ethernet IP, used to release TX/RX mstreset to Versal device GT and check for TX/RX mstresetdone status and reset sequencing to GT.

Figure 61: **Single Core with GT in Example Design Hierarchy**



X20631-041420

Figure 62: **Single Core with GT in Example Design Hierarchy (Versal ACAP)**



The previous figures show the instantiation of various modules and their hierarchy for a single core configuration of the ethernet_1_10_25g 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 `ethernet_1_10_25g_0_core_support.v` module 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 `ethernet_1_10_25g_0_sharedlogic_wrapper.v` module and the `ethernet_1_10_25g_0.v` module for the Include Shared Logic in example design option.

The user interface available for MAC/PCS configuration and PCS configuration configurations is the same as mentioned in the Overview topic.

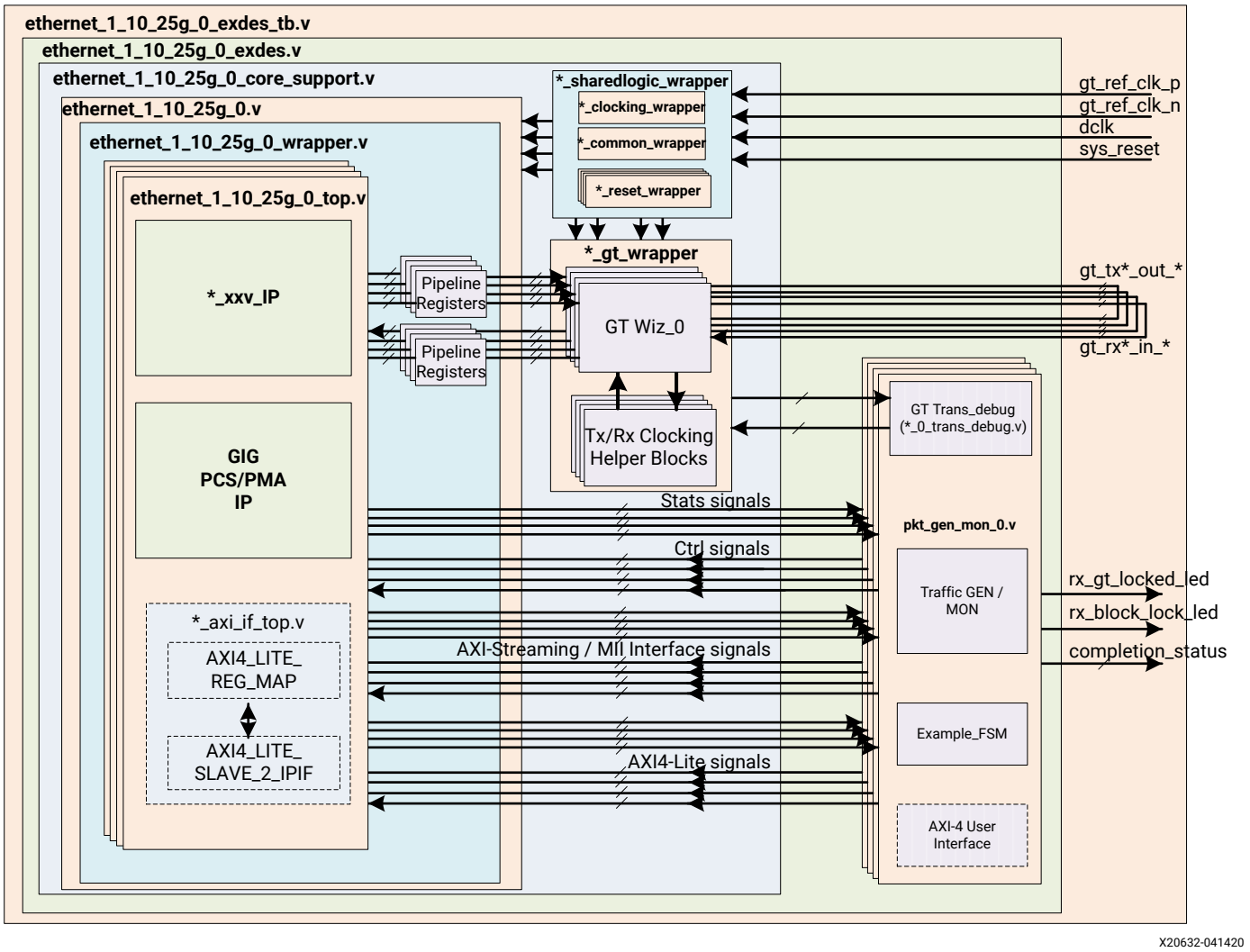
The `ethernet_1_10_25g_0.v` module instantiates the necessary the sync registers/retiming pipeline registers for the synchronization of data between the core and the GT.

The `ethernet_1_10_25g_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:

- **ethernet_1_10_25g_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.
- **ethernet_1_10_25g_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.

The following figure shows the instantiation of various modules and their hierarchy for the multiple core configuration of the ethernet_1_10_25g example design when the GT is in the example design.

Figure 63: **Multiple Cores with GT in Example Design Hierarchy**



For Versal device, the gt_quad_base (GT Wizard for Versal device) is part of the example design only, and 1/10/25G Ethernet Subsystem IP and GT (serial transceiver) IP is connected in the block design using the IP integrator.

Related Information

Overview

User Interface

General purpose I/Os (GPIOs) are provided to control the example design. The user input and output ports are described in the following table.

Table 52: **User I/O Ports**

Name	Size	I/O	Description
sys_reset	1	I	Reset for the core.
gt_ref_clk_p	1	I	Differential input clk to GT.
gt_ref_clk_n	1	I	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	I	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	O	Indicates that GT has been locked.
rx_block_lock_led_0	1	O	Indicates RX block lock has been achieved.
restart_tx_rx_0	1	I	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.
completion_status	5	O	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	I	This is used to switch the core speed.

Table 52: **User I/O Ports** *(cont'd)*

Name	Size	I/O	Description
core_speed_*	1	O	This signal indicates the speed with which the core is working: • 2'b00: 25G • 2'b01: 1G • 2'b10: 10G • 2'b11: Reserved
send_continuous_pkts_*	1	I	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.
ctl_core_speed_sel	2	I	This signal is used to set the operating speed of the core. • 2'b00 = 25G • 2'b01= 1G • 2'b01= 10G • 2'b11 = Reserved

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 tables, an asterisk (*) represents the CORE number, having a value from 0 to 3.

Example: `port_name_*`

- `port_name_0`: for first CORE
- `port_name_1`: for second CORE (present when you select number of cores ≥2)
- `port_name_2`: for third CORE (present when you select number of cores ≥3)
- `port_name_3`: for fourth CORE (present when you select number of cores =4)

Common Clock/Reset Signals

Table 53: **Common Clock/Reset Signals**

Name	Size	I/O	Description
sys_reset	1	I	Reset for core.
dclk	1	I	Stable input clk to GT.

Table 53: **Common Clock/Reset Signals** *(cont'd)*

Name	Size	I/O	Description
gt_refclk_p	1	I	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	I	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 I**nclude Shared Logic in core** option is selected in the Shared Logic tab.
tx_clk_out_*	1	O	TX user clock output from GT. ***Note*:** 1. 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. 2. This port is available when the **Select Core** is Ethernet MAC +PCS/PMA 32/64-bit and GT type is GTM and **Include GT subcore in example design** option is selected in the Selection and Configuration tab.
tx_mii_clk_*	1	O	TX mii clock output from GT. ***Note*:** This port is available when Select Core is Ethernet PCS 32-bit.
tx_stats_clk_out_*	1	O	312.5 MHz / 125 MHz out put clock to be used for the tx statistic.
rx_stats_clk_out_*	1	O	312.5 MHz / 125 MHz out put clock to be used for the rx statistics.
rx_clk_out_*	1	O	RX user clock output from GT.
tx_reset_*	1	I	TX reset input to the core.
user_tx_reset_*	1	O	TX reset output for the user logic.
rx_reset_*	1	I	RX reset input to the core.
user_rx_reset_*	1	O	RX reset output for the user logic.
gtpowergood_out_*	1	O	Refer to the *UltraScale Architecture GTH Transceivers User Guide* (UG576) for the port description.
rxrecclkout_*	1	O	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.
ctl_gt_reset_all_*	1	O	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 (for non-Versal device only).
ctl_gt_tx_reset_*	1	O	gt_tx_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 (for non-Versal device only).

Table 53: **Common Clock/Reset Signals** *(cont'd)*

Name	Size	I/O	Description
ctl_gt_rx_reset_*	1	O	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 (for non-Versal device only).
gtpowergood_in_*	1	I	Refer to the *UltraScale Architecture GTH Transceivers User Guide* (UG576) or *UltraScale Architecture GTY Transceivers User Guide* (UG578) for the port description. ***Note***: For Versal device only.
apb3clk_*	1	I	Refer to the *UltraScale Architecture GTH Transceivers User Guide* (UG576) or *UltraScale Architecture GTY Transceivers User Guide* (UG578) for the port description. ***Note***: For Versal device only when timestamp is enabled.

Common Transceiver Interface Ports

Table 54: **Common Transceiver Interface Ports**

Name	Size	I/O	Description
gt_loopback_in_*	3	I	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_txp_out	1	O	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	1	O	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	1	I	Differential serial GT RX input. ***Note***: This port is available when the IncludeGT subcore in core option and Board support is selected in the GT Selection and Configuration tab.
gt_rxp_in	1	I	Differential serial GT RX input. ***Note***: This port is available when the IncludeGT subcore in core option and Board support is selected

Table 54: **Common Transceiver Interface Ports** *(cont'd)*

Name	Size	I/O	Description
gt_rxp_in_0	1	I	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	I	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.
txuserrdy_out_*	1	O	TX user ready output signal from Core (reset Interface IP) to GT (Versal devices only).
rxuserrdy_out_*	1	O	RX user ready output signal from Core (reset Interface IP) to GT (Versal devices only).
mst_tx_reset_*	1	O	TX master reset output signal from Core to GT (Versal devices only).
mst_rx_reset_*	1	O	RX master reset output signal from Core to GT (Versal devices only).
mst_tx_dp_reset_*	1	O	TX master reset output signal from GT reset IP to the core (Versal devices only).
mst_rx_dp_reset_*	1	O	RX master reset output signal from GT reset IP to the core (Versal devices only).
mst_tx_resetdone_*	1	I	TX master resetdone signal from GT to Core (Versal devices only).
mst_rx_resetdone_*	1	I	RX master resetdone signal from GT to Core (Versal devices only).
gtwiz_loopback_out_*	3	O	GT loopback output signal from AXI4-Lite register map (non-Versal devices only). ***Note*:** This port is available when the Include GT subcore in example design option is selected in the GT Selection and Configuration tab and the AXI4-Lite interface is selected from configuration tab.
gtwiz_tx_rate_*	8	O	GT TX line rate select from the AXI4-Lite register map (Versal devices only). See the appropriate GT user guide. ***Note*:** This port is available when the Include GT subcore in example design option is selected in the GT Selection and Configuration tab and the AXI4-Lite interface is selected from configuration tab.
gtwiz_rx_rate_*	8	O	GT RX line rate select from the AXI4-Lite register map (Versal devices only). ***Note*:** This port is available when the Include GT subcore in example design option is selected in the GT Selection and Configuration tab and the AXI4-Lite interface is selected from configuration tab.
gtwiz_loopback_*	3	O	GT loopback output signal from AXI4-Lite register map (Versal devices only). ***Note*:** This port is available when AXI4-Lite interface is selected from the configuration tab.

Table 54: **Common Transceiver Interface Ports** *(cont'd)*

Name	Size	I/O	Description
gt_txp_out_0	1	O	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	O	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.

Transceiver Core and Status Debug Ports

The ports in the following table are 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.

For the port descriptions, see *UltraScale Architecture GTY Transceivers User Guide* (UG578) and *UltraScale Architecture GTH Transceivers User Guide* (UG576).

Table 55: **Transceiver Core and Status Debug Ports**

Name	Size	I/O
gt_dmonitorout_*	16	O
gt_eyescandataerror_*	1	O
gt_eyescanreset_*	1	I
gt_eyescantrigger_*	1	I
gt_pcsrsvdin_*	16	I
gt_rxbufreset_*	1	I
gt_rxbufstatus_*	3	O
gt_rxcdrhold_*	1	I
gt_rxcommadeten_*	1	I
gt_rxdfeagchold_*	1	I
gt_rxdfelpmreset_*	1	I
gt_rxlatclk_*	1	I
gt_rxlpmen_*	1	I
gt_rxpcsreset_*	1	I
gt_rxpmareset_*	1	I
gt_rxpolarity_*	1	I
gt_rxprbscntreset_*	1	I
gt_rxprbserr_*	1	I
gt_rxprbssel_*	4	I
gt_rxrate_*	3	I

Table 55: **Transceiver Core and Status Debug Ports** *(cont'd)*

Name	Size	I/O
gt_rxslide_in_*	1	I
gt_rxstartofseq_*	2	O
gt_txbufstatus_*	2	O
gt_txdiffctrl_*	5	I
gt_txinhibit_*	1	I
gt_txlatclk_*	1	I
gt_txmaincursor_*	7	I
gt_txpcsreset_*	1	I
gt_txpmareset_*	1	I
gt_txpolarity_*	1	I
gt_txpostcursor_*	5	I
gt_txprbsforceerr_*	1	I
gt_txprbssel_*	4	I
gt_txprecursor_*	5	I
gtwiz_reset_tx_datapath_*	1	I
gtwiz_reset_rx_datapath_*	1	I
gt_drpclk_*[1]	1	I
gt_drpdo_*[1]	16	O
gt_drprdy_*[1]	1	O
gt_drpen_*[1]	1	I
gt_drpwe_*[1]	1	I
gt_drpaddr_*[1]	10	I
gt_drpdi_*[1]	16	I

Notes:

1. This port is available for non-Versal device only.

Include GT Subcore in Example Design Ports

The following ports are available when the **Include GT subcore in Example Design** option is selected in the GT Selection and Configuration tab.

For the port descriptions, see the *UltraScale Architecture GTY Transceivers User Guide* (UG578) and the *UltraScale Architecture GTH Transceivers User Guide* (UG576).

Table 56: **Include GT Subcore in Example Design Ports**

Name	Size	I/O	Description
gtwiz_txpllclksel_*[1]	2	O	-
gtwiz_rxpllclksel_*[1]	2	O	-
gtwiz_txsysclksel_*[1]	2	O	-

Table 56: **Include GT Subcore in Example Design Ports** *(cont'd)*

Name	Size	I/O	Description
gtwiz_rxsysclksel_*[1]	2	O	-
gtwiz_rxoutclksel_*[1]	3	O	-
gtwiz_txoutclksel_*[1]	3	O	-
gtwiz_rxbufstatus_*	3	I	-
gtwiz_txbufstatus_*	2	I	-
gtwiz_userclk_tx_usrclk_out_*	1	I	-
gtwiz_userclk_rx_usrclk_out_*	1	I	-
gtwiz_userclk_tx_usrclk2_out_*	1	I	-
gtwiz_userclk_rx_usrclk2_out_*	1	I	-
gtwiz_buffbypass_tx_reset_*[1]	1	O	-
gtwiz_buffbypass_tx_start_user_* [1]	1	O	-
gtwiz_buffbypass_tx_done_* [1]	1	I	-
gtwiz_buffbypass_rx_reset_*[1]	1	O	-
gtwiz_buffbypass_rx_start_user_* [1]	1	O	-
gtwiz_txdiffctrl_*	4	O	-
gtwiz_cplllock_*	1	I	-
gtwiz_rxresetdone_*	1	I	-
gtwiz_txresetdone_*	1	I	-
gtwiz_rxclkcorcnt_*	2	I	-
gtwiz_rxlpmen_*	1	O	-
gtwiz_p_usrclk2_div_*	3	O	-
gtwiz_rx8b10ben_*	1	O	-
gtwiz_rxcommadeten_*	1	O	-
gtwiz_rxdlybypass_*[1]	1	O	-
gtwiz_tx8b10ben_*	1	O	-
gtwiz_rxmcommaalignen_*	1	O	-
gtwiz_rxpcommaalignen_*	1	O	-
gtwiz_rxphdlypd_*	1	O	-
gtwiz_txdlybypass_* [1]	1	O	-
gtwiz_rpllllock_*[2]	1	I	-
gtwiz_rpllpd_*[2]	1	O	-
gtwiz_rpllllocken_*[2]	1	O	-
gtwiz_rpllreset_*[2]	1	O	-
gtwiz_txphdlypd_*	1	O	-
gtwiz_txpippmpd_* [1]	1	O	-
gtwiz_txpippmsel_*	1	O	-
gtwiz_txpostcursor_*	5	O	-
gtwiz_cpllpd_*	1	O	-
gtwiz_cplllocken_*	1	O	-

Table 56: **Include GT Subcore in Example Design Ports** *(cont'd)*

Name	Size	I/O	Description
gtwiz_rxpd_*	2	O	-
gtwiz_txpd_*	2	O	-
gtwiz_txelecidle_*	1	O	-
gtwiz_reset_rx_done_*	1	I	-
gtwiz_reset_tx_done_*	1	I	-
gtwiz_txctrl0_*	16	O	-
gtwiz_txctrl1_*	16	O	-
gtwiz_txctrl2_*	8	O	-
gtwiz_rxctrl0_*	16	I	-
gtwiz_rxctrl1_*	16	I	-
gtwiz_rxctrl2_*	8	I	-
gtwiz_rxctrl3_*	8	O	-
gtwiz_rxgearboxslip_*	1	O	-
gtwiz_rxdatavalid_*	2	I	-
gtwiz_rxheader_*	6	I	-
gtwiz_rxheadervalid_*	2	I	-
gtwiz_rx_serdes_data_*	128	I	-
gtwiz_tx_serdes_data_*	128	O	-
gtwiz_txheader_*	6	O	-
gtwiz_txsequence_*	7	O	-
gtwiz_rxpmaresetdone_*	1	I	-
gtwiz_txpmaresetdone_*	1	I	-
apb3clk_*[3]	1	I	-
apb3paddr_*[3]	16	O	-
apb3penable_*[3]	1	O	-
apb3prdata_*[3]	32	I	-
apb3pready_*[3]	1	I	-
apb3presetn_*[3]	1	O	-
apb3psel_*[3]	1	O	-
apb3pslverr_*[3]	1	I	-
apb3pwdata_*[3]	32	O	-
apb3pwrite_*[3]	1	O	-
gtwiz_buffbypass_rx_done_* [1]	1	I	-
gtwiz_cpllreset_*	1	O	-
gt_reset_all_* [1]	1	O	Reset signal from the core to the GT.
gtwiz_drpdo_* [1]	16	I	DRP data signal from the GT to the core.
gtwiz_drprdy_*[1]	1	I	DRP ready signal from the GT to the core.
gtwiz_drpen_* [1]	1	O	The drpen signal from the core to the GT.
gtwiz_drpwe_*[1]	1	O	The drpwe signal from the core to the GT.

Table 56: **Include GT Subcore in Example Design Ports** *(cont'd)*

Name	Size	I/O	Description
gtwiz_drpaddr_*[1]	16	O	The drpaddr signal from the core to the GT
gtwiz_drpdi_* [1]	16	O	DRP data signal from the core to the GT.
gt_drpdo_* [1]	16	O	DRP read data from GT to the user, using the DRP arbiter within the core.
gt_drprdy_* [1]	1	O	The drprdy signal from GT to the user, using the DRP arbiter within the core.
gt_drp_gnt_* [1]	1	O	DRP grant signal from the DRP arbiter to the user to enable the GT DRP read/write from the user side.
gt_drp_req_*[1]	1	I	DRP request from the user to the DRP arbiter to perform the GT read/write operation.
gt_drpen_* [1]	1	I	The drpen signal from the user to the DRP arbiter to perform the GT read/write operation.
gt_drpwe_* [1]	1	I	The drpwe signal from the user to the DRP arbiter to perform the GT read/write operation.
gt_drpaddr_* [1]	10	I	The drpaddr signal from the user to the DRP arbiter to perform the GT read/write operation.
gt_drpdi_* [1]	16	I	The drpdi signal from the user to the DRP arbiter to perform the GT read/write operation.
stat_tx_packet_1024_1518_bytes_*	1	O	Increment for good and bad packets transmitted that contain 1,024 to 1,518 bytes.
stat_tx_packet_1519_1522_bytes_*	1	O	Increment for good and bad packets transmitted that contain 1,519 to 1,522 bytes.
stat_tx_packet_1523_1548_bytes_*	1	O	Increment for good and bad packets transmitted that contain 1,519 to 1,522 bytes.
stat_tx_packet_1549_2047_bytes_*	1	O	Increment for good and bad packets transmitted that contain 1,523 to 1,548 bytes.
stat_tx_packet_2048_4095_bytes_*	1	O	Increment for good and bad packets transmitted that contain 1,549 to 2,047 bytes.
stat_tx_packet_4096_8191_bytes_*	1	O	Increment for good and bad packets transmitted that contain 4,096 to 8,191 bytes.
stat_tx_packet_8192_9215_bytes_*	1	O	Increment for good and bad packets transmitted that contain 8,192 to 9,215 bytes.
stat_tx_packet_small_*	1	O	Increment for all packets that are less than 64 bytes long. Packets that are less than 64 bytes are not transmitted.
stat_tx_packet_large_*	1	O	Increment for all packets that are more than 9,215 bytes long.
stat_tx_frame_error_*	1	O	Increment for packets with tx_errin set to indicate an EOP abort.
gig_ethernet_pcs_pma_status_vector_*	16	O	See Status Vector table in *1G/2.5G Ethernet PCS/PMA or SGMII LogiCORE IP Product Guide* (PG047)
stat_core_speed_*	2	O	Indicates the operating core speed: • 2'b10: Core configured in 10G mode. • 2'b01: Core configured in 1G mode. • 2'b00: Core configured in 25G mode.
gpcs_resetdone_*	1	O	Indicates the 1G core is out of reset.

Table 56: **Include GT Subcore in Example Design Ports** *(cont'd)*

Name	Size	I/O	Description
gt_switching_*	1	O	Indicates the GT DRP operation to switch the line rate is in progress.

Notes:

1. This port is available for non-Versal device only.
2. This port is available for Versal device only.
3. This port is available for Versal device only when timestamping is enabled.

AXI4-Lite Interface Ports

Ports in the following table are available when **Include AXI4-Lite** is selected on the Configuration tab.

Table 57: **AXI4-Lite Interface Ports**

Name	Size	I/O	Description
s_axi_aclk_*	1	I	AXI clock signal
s_axi_aresetn_*	1	I	AXI reset signal
pm_tick_*	1	I	PM tick user input
s_axi_awaddr_*	32	I	AXI write address
s_axi_awvalid_*	1	I	AXI write address valid
s_axi_awready_*	1	O	AXI write address ready
s_axi_wdata_*	32	I	AXI write data
s_axi_wstrb_*	4	I	AXI write strobe. This signal indicates which byte lanes hold valid data.
s_axi_wvalid_*	1	I	AXI write data valid. This signal indicates that valid write data and strobes are available.
s_axi_wready_*	1	O	AXI write data ready
s_axi_bresp_*	2	O	AXI write response. This signal indicates the status of the write transaction. 'b00 = OKAY ‘b01 = EXOKAY ‘b10 = SLVERR ‘b11 = DECERR
s_axi_bvalid_*	1	O	AXI write response valid. This signal indicates that the channel is signaling a valid write response.
s_axi_bready_*	1	I	AXI write response ready.
s_axi_araddr_*	32	I	AXI read address
s_axi_arvalid_*	1	I	AXI read address valid
s_axi_arready_*	1	O	AXI read address ready
s_axi_rdata_*	32	O	AXI read data issued by slave

Table 57: **AXI4-Lite Interface Ports** *(cont'd)*

Name	Size	I/O	Description
s_axi_rresp_*	2	O	AXI read response. This signal indicates the status of the read transfer. 'b00 = OKAY 'b01 = EXOKAY 'b10 = SLVERR 'b11 = DECERR
s_axi_rvalid_*	1	O	AXI read data valid
s_axi_rready_*	1	I	AXI read ready. This signal indicates the user/master can accept the read data and response information.
user_reg0_*	32	O	User-defined signal from the AXI4 register map USER_REG_0 register.

AXI4-Stream User Interface Signals

Ports in this section are available when **Ethernet MAC+PCS/PMA-32 bit** is selected from the Configuration tab.

Table 58: **AXI4-Stream User Interface Signals**

Name	Size	I/O	Description
tx_unfout_*	1	O	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. ***Note*:** When this signal sampled as 1, you must apply tx_reset/sys_reset to recover the subsystem from the underflow issue. tx_reset resets the TX path only and sys_reset recovers the complete system.
tx_axis_tready_*	1	O	TX path ready signal from core.
tx_axis_tvalid_*	1	I	Transmit AXI4-Stream Data valid.
tx_axis_tdata_*	32	I	Transmit AXI4-Stream Data bus.
tx_axis_tlast_*	1	I	Transmit AXI4-Stream tlast.
tx_axis_tkeep_*	8/4	I	Transmit AXI4-Stream tkeep.
tx_axis_tuser_*	1	I	Transmit AXI4-Stream tuser.
tx_preamblein_*	56	I	Transmit AXI4-Stream preamble.
rx_axis_tvalid_*	1	O	Receive AXI4-Stream Data valid.
rx_axis_tdata_*	32	O	Receive AXI4-Stream Data bus.
rx_axis_tlast_*	1	I	Receive AXI4-Stream tlast.
rx_axis_tkeep_*	8/4	I	Receive AXI4-Stream tkeep.
rx_axis_tuser_*	1	I	Receive AXI4-Stream tuser.
rx_preamblein_*	56	I	Receive AXI4-Stream preamble.

TX Path Control/Status/Statistics Signals

Table 59: **TX Path Control/Status/Statistics Signals**

Name	Size	I/O	Description
ctl_tx_send_rfi_*	1	I	Transmit Remote Fault Indication (RFI) code word. If this input is sampled as a 1, the TX path only transmits Remote Fault code words. This input should be set to 1 until the RX path is fully aligned and is ready to accept data from the link partner.
ctl_tx_send_lfi_*	1	I	Transmit Local Fault Indication (LFI) code word. Takes precedence over RFI.
ctl_tx_send_idle_*	1	I	Transmit Idle code words. If this input is sampled as a 1, the TX path only transmits Idle code words. This input should be set to 1 when the partner device is sending Remote Fault Indication (RFI) code words.
stat_tx_local_fault_*	1	O	A value of 1 indicates the receive decoder state machine is in the TX_INIT state. This output is level sensitive.
stat_tx_total_bytes_*	5	O	Increment for the total number of bytes transmitted.
stat_tx_total_packets_*	1	O	Increment for the total number of packets transmitted.
stat_tx_total_good_bytes_*	14	O	Increment for the total number of good bytes transmitted. This value is only non-zero when a packet is transmitted completely and contains no errors.
stat_tx_total_good_packets_*	1	O	Increment for the total number of good packets transmitted.
stat_tx_bad_fcs_*	1	O	Increment for packets greater than 64 bytes that have FCS errors.
stat_tx_packet_64_bytes_*	1	O	Increment for good and bad packets transmitted that contain 64 bytes.
stat_tx_packet_65_127_bytes_*	1	O	Increment for good and bad packets transmitted that contain 65 to 127 bytes.
stat_tx_packet_128_255_bytes_*	1	O	Increment for good and bad packets transmitted that contain 128 to 255 bytes.
stat_tx_packet_256_511_bytes_*	1	O	Increment for good and bad packets transmitted that contain 256 to 511 bytes.
stat_tx_packet_512_1023_bytes_*	1	O	Increment for good and bad packets transmitted that contain 512 to 1,023 bytes.
stat_tx_packet_1024_1518_bytes_*	1	O	Increment for good and bad packets transmitted that contain 1,024 to 1,518 bytes.
stat_tx_packet_1519_1522_bytes_*	1	O	Increment for good and bad packets transmitted that contain 1,519 to 1,522 bytes.
stat_tx_packet_1523_1548_bytes_*	1	O	Increment for good and bad packets transmitted that contain 1,523 to 1,548 bytes.
stat_tx_packet_1549_2047_bytes_*	1	O	Increment for good and bad packets transmitted that contain 1,549 to 2,047 bytes.
stat_tx_packet_2048_4095_bytes_*	1	O	Increment for good and bad packets transmitted that contain 2,048 to 4,095 bytes.
stat_tx_packet_4096_8191_bytes_*	1	O	Increment for good and bad packets transmitted that contain 4,096 to 8,191 bytes.
stat_tx_packet_8192_9215_bytes_*	1	O	Increment for good and bad packets transmitted that contain 8,192 to 9,215 bytes.
stat_tx_packet_small_*	1	O	Increment for all packets that are less than 64 bytes long. Packets that are less than 64 bytes are not transmitted.
stat_tx_packet_large_*	1	O	Increment for all packets that are more than 9,215 bytes long.

Table 59: **TX Path Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
stat_tx_frame_error_*	1	O	Increment for packets with tx_axis_tuser set to indicate an End of Packet (EOP) abort or frames aborted by de-asserting tvalid without tlast.
gig_ethernet_pcs_pma_status_vector_*	16	O	See the Status Vector Table in *1G/2.5G Ethernet PCS/PMA or SGMII LogiCORE IP Product Guide* (PG047).
stat_core_speed_*	2	O	Indicates the operating core speed: • 2'b10: Core configured in 10G mode • 2'b01: Core configured in 1G mode • 2'b00: Core configured in 25G mode
gpcs_resetdone_*	1	O	Indicates the 1G core is out-of reset.
gt_switching_*	1	O	Indicates the GT DRP operation to switch the line rate is in progress.

RX Path Control/Status/Statistics Signals

Table 60: **RX Path Control/Status/Statistics Signals**

Name	Size	I/O	Description
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-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-bit** and the **Include FIFO Logic** is disabled.
stat_rx_block_lock_*	4	O	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	O	Valid indicator for stat_rx_framing_err. When 1 stat_rx_framing_err_0 is valid.
stat_rx_framing_err_*	3	O	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	O	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 60: **RX Path Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
stat_rx_bad_code_*	2	O	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_error_valid_*	1	O	Indicates when stat_rx_error is valid.
stat_rx_total_packets_*	2	O	Increment for the total number of packets received.
stat_rx_total_good_packets_*	1	O	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.
stat_rx_total_bytes_*	6	O	Increment for the total number of bytes received.
stat_rx_total_good_bytes_*	14	O	Increment for the total number of good bytes received. This value is only non-zero when a packet is received completely and contains no errors.
stat_rx_packet_small_*	2	O	Increment for all packets that are less than 64 bytes long. Packets that are less than 4 bytes are dropped.
stat_rx_jabber_*	1	O	Increment for packets longer than ctl_rx_max_packet_len with bad FCS.
stat_rx_packet_large_*	1	O	Increment for all packets that are more than 9,215 bytes long.
stat_rx_oversize_*	1	O	Increment for packets longer than ctl_rx_max_packet_len with good FCS.
stat_rx_undersize_*	2	O	Increment for packets shorter than stat_rx_min_packet_len with good FCS.
stat_rx_toolong_*	1	O	Increment for packets longer than ctl_rx_max_packet_len with good and bad FCS.
stat_rx_fragment_*	2	O	Increment for packets shorter than stat_rx_min_packet_len with bad FCS.
stat_rx_packet_64_bytes_*	1	O	Increment for good and bad packets received that contain 64 bytes.
stat_rx_packet_65_127_bytes_*	1	O	Increment for good and bad packets received that contain 65 to 127 bytes.
stat_rx_packet_128_255_bytes_*	1	O	Increment for good and bad packets received that contain 128 to 255 bytes.
stat_rx_packet_256_511_bytes_*	1	O	Increment for good and bad packets received that contain 256 to 511 bytes.
stat_rx_packet_512_1023_bytes_*	1	O	Increment for good and bad packets received that contain 512 to 1,023 bytes.
stat_rx_packet_1024_1518_bytes_*	1	O	Increment for good and bad packets received that contain 1,024 to 1,518 bytes.
stat_rx_packet_1519_1522_bytes_*	1	O	Increment for good and bad packets received that contain 1,519 to 1,522 bytes.
stat_rx_packet_1523_1548_bytes_*	1	O	Increment for good and bad packets received that contain 1,523 to 1,548 bytes.
stat_rx_packet_1549_2047_bytes_*	1	O	Increment for good and bad packets received that contain 1,549 to 2,047 bytes.
stat_rx_packet_2048_4095_bytes_*	1	O	Increment for good and bad packets received that contain 2,048 to 4,095 bytes.
stat_rx_packet_4096_8191_bytes_*	1	O	Increment for good and bad packets received that contain 4,096 to 8,191 bytes.

Table 60: **RX Path Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
stat_rx_packet_8192_9215_bytes_*	1	O	Increment for good and bad packets received that contain 8,192 to 9,215 bytes.
stat_rx_bad_fcs_*	2	O	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.
stat_rx_packet_bad_fcs_*	1	O	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.
stat_rx_stomped_fcs_*	2	O	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.
stat_rx_bad_preamble_*	1	O	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. 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.
stat_rx_bad_sfd_*	1	O	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. 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_*	1	O	Signal OS indication. If this bit is sampled as a 1, it indicates that a Signal OS word was received. Signal OS should not be received in an Ethernet network.
stat_rx_test_pattern_mismatch_*	2	O	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 Preemption is enabled or the core type is MAC+PCS/PMA 32-bit type.
stat_rx_truncated_*	1	O	Packet truncation indicator. A value of 1 indicates that the current packet in flight is truncated due to its length exceeding ctl_rx_max_packet_len[14:0]. This output is pulsed for one clock cycle to indicate the truncated condition. Pulses can occur in back-to-back cycles.
stat_rx_local_fault_*	1	O	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	O	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.

Table 60: **RX Path Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
stat_rx_internal_local_fault_*	1	O	This signal goes High when an internal local fault is generated due to any one of the following: test pattern generation, bad lane alignment, or high bit error rate. This signal remains High as long as the fault condition persists.
stat_rx_received_local_fault_*	1	O	This signal goes High when enough local fault words are received from the link partner to trigger a fault condition as specified by the IEEE fault state machine. This signal remains High as long as the fault condition persists.
stat_rx_valid_ctrl_code_*	1	O	Indicates that a PCS block with a valid control code was received.
stat_rx_status*	1	O	Indicates the link status.

TX Pause Interface Control/Status/Statistics Signals

Ports under this section are available when **Enable TX Flow Control Logic** is selected from the MAC Options tab and **Select Core** is Ethernet MAC+PCS/PMA 64-bit.

Table 61: **TX Pause Interface Control/Status/Statistics Signals**

Name	Size	I/O	Description
ctl_tx_pause_req_*	9	I	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	I	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.[1]
ctl_tx_resend_pause_*	1	I	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	I	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.[1]
ctl_tx_pause_quanta1_*	16	I	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.[1]
ctl_tx_pause_quanta2_*	16	I	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.[1]
ctl_tx_pause_quanta3_*	16	I	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.[1]
ctl_tx_pause_quanta4_*	16	I	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.[1]

Table 61: **TX Pause Interface Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
ctl_tx_pause_quanta5_*	16	I	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.[1]
ctl_tx_pause_quanta6_*	16	I	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.[1]
ctl_tx_pause_quanta7_*	16	I	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.[1]
ctl_tx_pause_quanta8_*	16	I	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.[1]
ctl_tx_pause_refresh_timer0_*	16	I	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.[1]
ctl_tx_pause_refresh_timer1_*	16	I	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.[1]
ctl_tx_pause_refresh_timer2_*	16	I	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.[1]
ctl_tx_pause_refresh_timer3_*	16	I	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.[1]
ctl_tx_pause_refresh_timer4_*	16	I	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.[1]
ctl_tx_pause_refresh_timer5_*	16	I	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.[1]
ctl_tx_pause_refresh_timer6_*	16	I	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.[1]
ctl_tx_pause_refresh_timer7_*	16	I	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.[1]
ctl_tx_pause_refresh_timer8_*	16	I	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.[1]
ctl_tx_da_gpp_*	48	I	Destination address for transmitting global pause packets.[1]
ctl_tx_sa_gpp_*	48	I	Source address for transmitting global pause packets.[1]

Table 61: **TX Pause Interface Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
ctl_tx_ethertype_gpp_*	16	I	Ethertype for transmitting global pause packets.[1]
ctl_tx_opcode_gpp_*	16	I	Opcode for transmitting global pause packets.[1]
ctl_tx_da_ppp_*	48	I	Destination address for transmitting priority pause packets.[1]
ctl_tx_sa_ppp_*	48	I	Source address for transmitting priority pause packets.[1]
ctl_tx_ethertype_ppp_*	16	I	Ethertype for transmitting priority pause packets.[1]
ctl_tx_opcode_ppp_*	16	I	Opcode for transmitting priority pause packets.[1]
stat_tx_pause_valid_*	9	O	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_*[2]	1	O	Increment for good unicast packets.
stat_tx_multicast_*	1	O	Increment for good multicast packets.
stat_tx_broadcast_*	1	O	Increment for good broadcast packets.
stat_tx_vlan_*	1	O	Increment for good 802.1Q tagged VLAN packets.
stat_tx_pause_*	1	O	Increment for 802.3x Ethernet MAC Pause packet with good FCS.
stat_tx_user_pause_*	1	O	Increment for priority-based pause packets with good FCS.

Notes:

1. This port is available when Include AXI4-Lite is *not* selected in the Configuration tab.
2. This port is available when RX flow control is enabled or Preemption is enabled or core type is MAC+PCS/PMA 32-bit.

RX Pause Interface Control/Status/Statistics Signals

Ports in this section are available when Enable RX Flow Control Logic is selected from the MAC Options tab and Select Core is Ethernet MAC+PCS/PMA 64-bit.

Table 62: **RX Pause Interface Control / Status / Statistics Signals**

Name	Size	I/O	Description
ctl_rx_forward_control_*	1	I	A value of 1 indicates that the CORE forwards control packets to you. A value of 0 causes CORE to drop control packets.[1]
ctl_rx_pause_ack_*	9	I	Pause acknowledge signal. This bus is used to acknowledge the receipt of the pause frame from the user logic.
ctl_rx_check_ack_*	1	I	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.[1]
ctl_rx_pause_enable_*	9	I	RX pause enable signal. This input is used to enable the processing of the pause quanta for the corresponding priority.[11]
ctl_rx_enable_gcp_*	1	I	A value of 1 enables global control packet processing.[1]
ctl_rx_check_mcast_gcp_*	1	I	A value of 1 enables global control multicast destination address processing.[1]
ctl_rx_check_ucast_gcp_*	1	I	A value of 1 enables global control unicast destination address processing.[1]
ctl_rx_pause_da_ucast_*	48	I	Unicast destination address for pause processing.[1]
ctl_rx_check_sa_gcp_*	1	I	A value of 1 enables global control source address processing.[1]

Table 62: **RX Pause Interface Control / Status / Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
ctl_rx_pause_sa_*	48	I	Source address for pause processing.[1]
ctl_rx_check_etype_gcp_*	1	I	A value of 1 enables global control ethertype processing.[1]
ctl_rx_etype_gcp_*	16	I	Ethertype field for global control processing.[1]
ctl_rx_check_opcode_gcp_*	1	I	A value of 1 enables global control opcode processing.[1]
ctl_rx_opcode_min_gcp_*	16	I	Minimum global control opcode value.[1]
ctl_rx_opcode_max_gcp_*	16	I	Maximum global control opcode value.[1]
ctl_rx_enable_pcp_*	1	I	A value of 1 enables priority control packet processing.[1]
ctl_rx_check_mcast_pcp_*	1	I	A value of 1 enables priority control multicast destination address processing.[1]
ctl_rx_check_ucast_pcp_*	1	I	A value of 1 enables priority control unicast destination address processing.[1]
ctl_rx_pause_da_mcast_*	48	I	Multicast destination address for pause processing.[1]
ctl_rx_check_sa_pcp_*	1	I	A value of 1 enables priority control source address processing.[1]
ctl_rx_check_etype_pcp_*	1	I	A value of 1 enables priority control ethertype processing.[1]
ctl_rx_etype_pcp_*	16	I	Ethertype field for priority control processing.[1]
ctl_rx_check_opcode_pcp_*	1	I	A value of 1 enables priority control opcode processing.[1]
ctl_rx_opcode_min_pcp_*	16	I	Minimum priority control opcode value.[1]
ctl_rx_opcode_max_pcp_*	16	I	Maximum priority control opcode value.[1]
ctl_rx_enable_gpp_*	1	I	A value of 1 enables global pause packet processing.[1]
ctl_rx_check_mcast_gpp_*	1	I	A value of 1 enables global pause multicast destination address processing.[1]
ctl_rx_check_ucast_gpp_*	1	I	A value of 1 enables global pause unicast destination address processing.[1]
ctl_rx_check_sa_gpp_*	1	I	A value of 1 enables global pause source address processing.[1]
ctl_rx_check_etype_gpp_*	1	I	A value of 1 enables global pause ethertype processing.[1]
ctl_rx_etype_gpp_*	16	I	Ethertype field for global pause processing.[1]
ctl_rx_check_opcode_gpp_*	1	I	A value of 1 enables global pause opcode processing.[1]
ctl_rx_opcode_gpp_*	16	I	Global pause opcode value.[1]
ctl_rx_enable_ppp_*	1	I	A value of 1 enables priority pause packet processing.[1]
ctl_rx_check_mcast_ppp_*	1	I	A value of 1 enables priority pause multicast destination address processing.[1]
ctl_rx_check_ucast_ppp_*	1	I	A value of 1 enables priority pause unicast destination address processing.[1]
ctl_rx_check_sa_ppp_*	1	I	A value of 1 enables priority pause source address processing.[1]
ctl_rx_check_etype_ppp_*	1	I	A value of 1 enables priority pause ethertype processing.[1]
ctl_rx_etype_ppp_*	16	I	Ethertype field for priority pause processing.[1]
ctl_rx_check_opcode_ppp_*	1	I	A value of 1 enables priority pause opcode processing.[1]
ctl_rx_opcode_ppp_*	16	I	Priority pause opcode value.[1]
stat_rx_unicast_*[2]	1	O	Increment for good unicast packets.
stat_rx_multicast_*[2]	1	O	Increment for good multicast packets.
stat_rx_broadcast_*[2]	1	O	Increment for good broadcast packets.

Table 62: **RX Pause Interface Control / Status / Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
stat_rx_vlan_*[2]	1	O	Increment for good 802.1Q tagged VLAN packets.
stat_rx_pause_*	1	O	Increment for 802.3x Ethernet MAC Pause packet with good FCS.
stat_rx_user_pause_*	1	O	Increment for priority-based pause packets with good FCS.
stat_rx_inrangeerr_*[2]	1	O	Increment for packets with Length field error but with good FCS.
stat_rx_pause_valid_*	9	O	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	O	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	O	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	O	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	O	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	O	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_quanta5_*	16	O	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	O	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	O	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	O	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	O	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.

Notes:

1. This port is available when Include AXI4-Lite is *not* selected in the Configuration tab.
2. This port is available when RX flow control is enabled or Preemption is enabled or core type is MAC+PCS/PMA 32-bit.

IEEE 1588 TX/RX Interface Control/Status/Statistics Signals

Ports in the following table are available when **Enable_Time_Stamping** is selected from the MAC Options tab.

Table 63: **IEEE 1588 TX/RX Interface Control/Status/Statistics Signals**

Name	Size	I/O	Description
ctl_tx_systemtimerin_*	80	I	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.
ctl_rx_systemtimerin_*	80	I	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.
stat_tx_ptp_fifo_read_error_*	1	O	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	O	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_ptp_1588op_in_*	2	I	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	I	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	O	This bit indicates that a valid timestamp is being presented on the TX.
tx_ptp_tstamp_tag_out_*	16	O	Tag output corresponding to tx_ptp_tag_field_in[15:0].
tx_ptp_tstamp_out_*	80	O	Timestamp 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.

Table 63: **IEEE 1588 TX/RX Interface Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
rx_ptp_tstamp_valid_out_*	1	O	This bit indicates that a valid timestamp is being presented on the RX. This will be present only when core is Ethernet MAC+PCS/PMA-32/64-bit.
rx_ptp_tstamp_out_*	80	O	Timestamp 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.
tx_ptp_e_rxtstamp_*	64	I	TX PTP RX timestamp.
tx_ptp_p_rxtstamp_*[1]	64	I	TX PTP RX timestamp.
rx_ptp_e_tstamp_*[1]	80	O	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.
rx_ptp_p_tstamp_*[1]	80	O	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.
tx_ptp_e_tstamp_valid_out_*[2]	1	O	This bit indicates that a valid timestamp is presented on the TX.
tx_ptp_p_tstamp_valid_out_*[2]	1	O	This bit indicates that a valid timestamp is presented on the TX.
tx_ptp_e_tstamp_tag_out_*[2]	16	O	Tag output corresponding to tx_ptp_tag_field_in[15:0].
tx_ptp_p_tstamp_tag_out_*[2]	16	O	Tag output corresponding to tx_ptp_tag_field_in[15:0].

Notes:

1. This port is available when Preemption is enabled.
2. This port is available when Preemption is enabled. This ports is valid for non-Versal™ device only.

AN Interface Control/Status/Statistics Signals

The following ports are available when the Include AN Logic option is selected in the Configuration tab.

Table 64: **AN Interface Control/Status/Statistics Signals**

Name	Size	I/O	Description
an_reset_*	1	I	Asynchronous active-High reset corresponding to an_clk domain.
an_loc_np_data_*	48	I	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	O	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.

Table 64: **AN Interface Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
ctl_autoneg_enable_*	1	I	Enable signal for auto-negotiation. ***Note*:** This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_autoneg_bypass_*	1	I	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	I	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	I	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	I	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	I	This input signal is used to set the remote_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	I	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.
ctl_an_asmdir_*	1	I	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	I	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.

Table 64: **AN Interface Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
ctl_an_fec_25g_rs_request_*	1	I	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	I	Indicates the baser FEC request. ***Note***: This port is available when **Include AXI4-Lite** is not selected in the Configuration tab.
ctl_an_fec_ability_override_*	1	I	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	I	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.
ctl_an_lp_np_ack_*	1	I	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	I	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	I	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 64: **AN Interface Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
ctl_an_ability_1000base_kx_*	1	I	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	I	
ctl_an_ability_10gbase_kr_*	1	I	
ctl_an_ability_40gbase_kr4_*	1	I	
ctl_an_ability_40gbase_cr4_*	1	I	
ctl_an_ability_100gbase_cr10_*	1	I	
ctl_an_ability_100gbase_kp4_*	1	I	
ctl_an_ability_100gbase_kr4_*	1	I	
ctl_an_ability_100gbase_cr4_*	1	I	
ctl_an_ability_25gbase_krcr_s_*	1	I	
ctl_an_ability_25gbase_krcr_*	1	I	
ctl_an_ability_25gbase_kr1_*	1	I	
ctl_an_ability_25gbase_cr1_*	1	I	
ctl_an_ability_50gbase_kr2_*	1	I	
ctl_an_ability_50gbase_cr2_*	1	I	
stat_an_link_cntl_1000base_kx_*	2	O	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	O	
stat_an_link_cntl_10gbase_kr_*	2	O	
stat_an_link_cntl_40gbase_kr4_*	2	O	
stat_an_link_cntl_40gbase_cr4_*	2	O	
stat_an_link_cntl_100gbase_cr10_*	2	O	
stat_an_link_cntl_100gbase_kp4_*	2	O	
stat_an_link_cntl_100gbase_kr4_*	2	O	
stat_an_link_cntl_100gbase_cr4_*	2	O	
stat_an_link_cntl_25gbase_krcr_s_*	2	O	
stat_an_link_cntl_25gbase_krcr_*	2	O	
stat_an_link_cntl_25gbase_kr1_*	2	O	
stat_an_link_cntl_25gbase_cr1_*	2	O	
stat_an_link_cntl_50gbase_kr2_*	2	O	
stat_an_link_cntl_50gbase_cr2_*	2	O	
stat_an_fec_enable_*	1	O	Used to enable the use of clause 74 FEC on the link.
stat_an_tx_pause_enable_*	1	O	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	O	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	O	Indicates the auto-negotiation is complete and rx link status from the PCS has been received.
stat_an_parallel_detection_fault_*	1	O	Indicated a parallel detection fault during auto-negotiation.

Table 64: **AN Interface Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
stat_an_lp_ability_1000base_kx_*	1	O	These signals indicate the advertised protocol from the link partner. They all become valid 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	O	
stat_an_lp_ability_10gbase_kr_*	1	O	
stat_an_lp_ability_40gbase_kr4_*	1	O	
stat_an_lp_ability_40gbase_cr4_*	1	O	
stat_an_lp_ability_100gbase_cr10_*	1	O	
stat_an_lp_ability_100gbase_kp4_*	1	O	
stat_an_lp_ability_100gbase_kr4_*	1	O	
stat_an_lp_ability_100gbase_cr4_*	1	O	
stat_an_lp_ability_25gbase_krcr_s_*	1	O	
stat_an_lp_ability_25gbase_krcr_*	1	O	
stat_lp_ability_25gbase_cr1_*	1	O	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	O	Indicates the rx cdr hold signal.
stat_an_lp_pause_*	1	O	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	O	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	O	This bit indicates link partner remote fault.
stat_an_lp_fec_10g_ability_*	1	O	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	O	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	O	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	O	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	O	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	O	This signal indicates when all of the link partner advertisements become valid.

Table 64: **AN Interface Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
stat_an_loc_np_ack_*	1	O	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	O	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	O	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	O	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	O	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_cr2_*	1	O	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	O	This output indicates the extended FEC abilities.
stat_an_rs_fec_enable_*	1	O	Used to enable the use of clause 91 FEC on the link.
stat_an_lp_extended_ability_valid_*	1	O	When this bit is 1, it indicates that the detected extended abilities are valid.
stat_an_switch_speed_*	1	O	This bit indicates that the AN module requests a speed switch to complete the Auto-Negotiation. Core speed switches between 10G and 1G to perform DME transaction or Parallel detection.

LT Interface Control/Status/Signals

Table 65: **LT Interface Control/Status/Signals**

Name	Size	I/O	Description
ctl_lt_training_enable_*	1	I	Enables link training. When link training is disabled, all PCS lanes function in mission mode.[1]
ctl_lt_restart_training_*	1	I	This signal triggers a restart of link training regardless of the current state.[1]
ctl_lt_rx_trained_*	4	I	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.[1]

Table 65: **LT Interface Control/Status/Signals** *(cont'd)*

Name	Size	I/O	Description
ctl_lt_preset_to_tx_*	4	I	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.[1]
ctl_lt_initialize_to_tx_*	4	I	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.[1]
ctl_lt_pseudo_seed0_*	11	I	This 11-bit signal seeds the training pattern generator.[1]
ctl_lt_k_p1_to_tx0_*	2	I	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.[1]
ctl_lt_k0_to_tx0_*	2	I	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.[1]
ctl_lt_k_m1_to_tx0_*	2	I	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.[1]
ctl_lt_stat_p1_to_tx0_*	2	I	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.[1]
ctl_lt_stat0_to_tx0_*	2	I	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.[1]
ctl_lt_stat_m1_to_tx0_*	2	I	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.[1]
stat_lt_signal_detect_*	4	O	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	O	This signal indicates when the respective link training state machine is performing link training.
stat_lt_training_fail_*	4	O	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	O	This output is High for 1 RX SerDes clock cycle to indicate the start of the link training frame.
stat_lt_frame_lock_*	4	O	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	O	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	O	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	O	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	O	This 2-bit field indicates the update control bits for the k0 coefficient, as received from the link partner in the control block.
stat_lt_k_m1_from_rx0_*	2	O	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	O	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	O	This 2-bit fields indicates the update status bits for the k0 coefficient, as received from the link partner in the status block.

Table 65: **LT Interface Control/Status/Signals** *(cont'd)*

Name	Size	I/O	Description
stat_lt_stat_m1_from_rx0_*	2	O	This 2-bit field indicates the update status bits for the k-1 coefficient, as received from the link partner in the status block.
lt_tx_sof_*	4	O	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.

Notes:

1. This port is available when **Include AXI4-Lite** is *not* selected in the Configuration tab.

Clause 74 FEC Interface Control/Status/Statistics Signals

Ports in the following table are available when Clause 74 (BASE-KR FEC) is selected from the Configuration tab.

Table 66: **Clause 74 FEC Interface Control/Status/Statistics Signals**

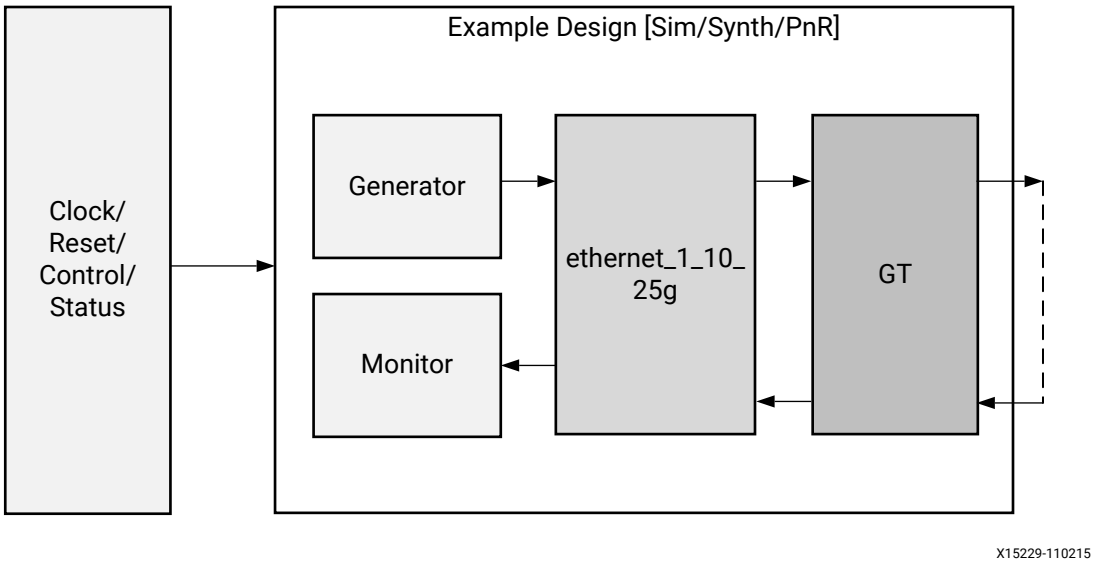
Name	Size	I/O	Description
ctl_fec_tx_enable_*	1	I	Asserted to enable the clause 74 FEC encoding on the transmitted data.
ctl_fec_rx_enable_*	1	I	Asserted to enable the clause 74 FEC decoding of the received data.
ctl_fec_enable_error_to_pcs_*	1	I	Clause 74 FEC enable error to pcs.
stat_fec_inc_correct_count_*	4	O	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	O	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	O	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	O	This signal is asserted while the ctl_fec_rx_enable is asserted when the FEC decoder detects the frame boundary.

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.

The following figure shows the duplex mode of operation.

Figure 64: **Duplex Mode of Operation**



AXI4-Lite Interface Implementation

In order to instantiate the AXI4-Lite interface to access the control and status registers of the ethernet_1_10_25g core, enable the **Include AXI4-Lite** check box in the Configuration Tab of the Vivado IDE. This option enables the `ethernet_1_10_25g_axi_if_top` module (which contains `ethernet_1_10_25g_pif_registers` with the `ethernet_1_10_25g_slave_2_ipif` module). You can access the AXI4-Lite interface logic registers (control, status and statistics) from the `ethernet_1_10_25g_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.

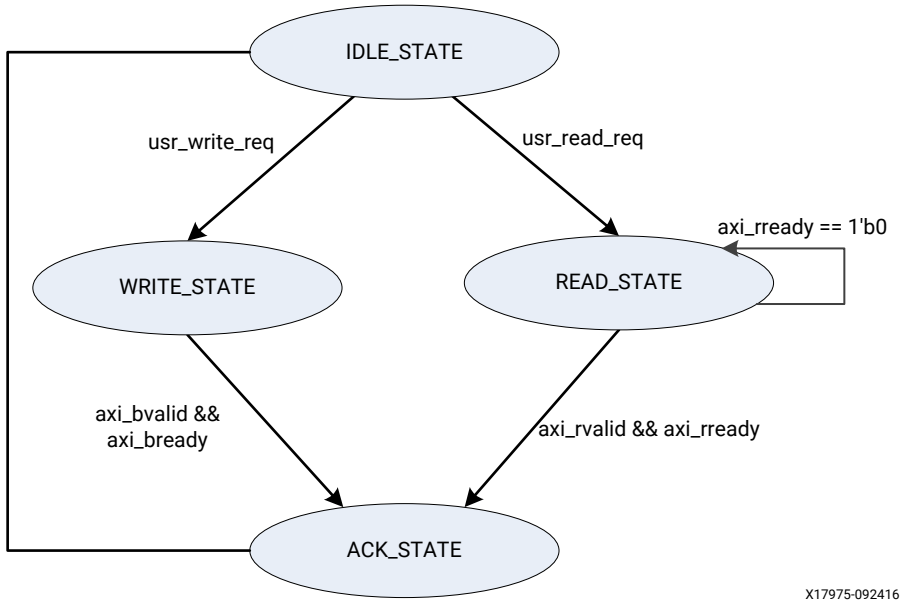
AXI4 Interface User Logic

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 65: **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.

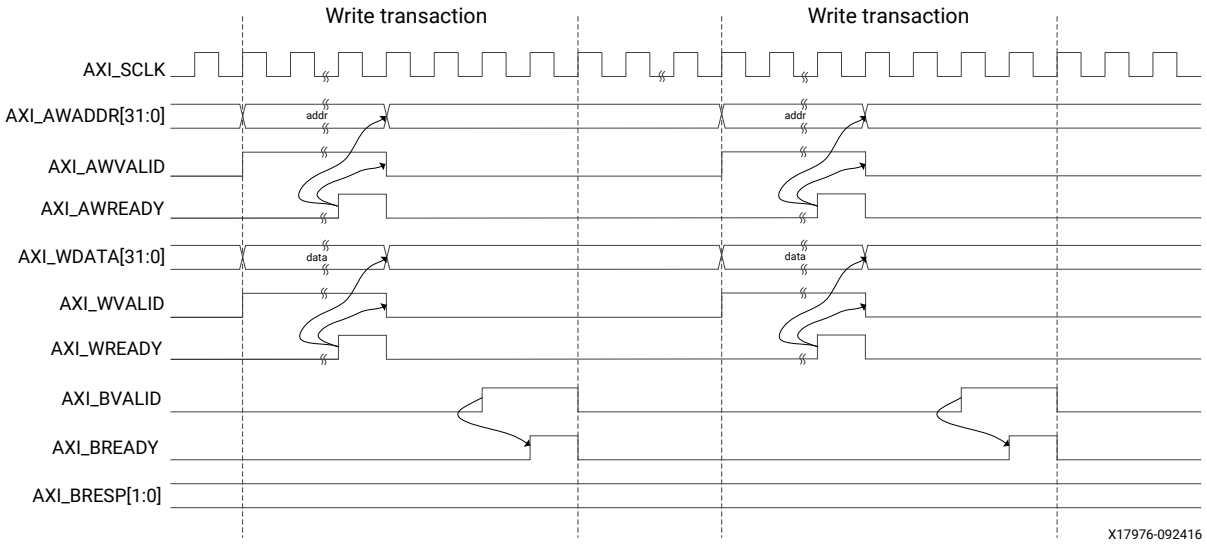
AXI User Interface Ports

Table 67: **AXI User Interface Ports**

Name	Size	I/O	Description
S_AXI_ACLK	1	I	AXI clock signal
S_AXI_ARESETN	1	I	AXI active-Low synchronous reset
S_AXI_PM_TICK	1	I	PM tick user input
S_AXI_AWADDR	32	I	AXI write address
S_AXI_AWVALID	1	I	AXI write address valid
S_AXI_AWREADY	1	O	AXI write address ready
S_AXI_WDATA	32	I	AXI write data
S_AXI_WSTRB	4	I	AXI write strobe. This signal indicates which byte lanes hold valid data.
S_AXI_WVALID	1	I	AXI write data valid. This signal indicates that valid write data and strobes are available.
S_AXI_WREADY	1	O	AXI write data ready
S_AXI_BRESP	2	O	AXI write response. This signal indicates the status of the write transaction. 'b00 = OKAY ‘b01 = EXOKAY ‘b10 = SLVERR ‘b11 = DECERR
S_AXI_BVALID	1	O	AXI write response valid. This signal indicates that the channel is signaling a valid write response.
S_AXI_BREADY	1	I	AXI write response ready.
S_AXI_ARADDR	32	I	AXI read address
S_AXI_ARVALID	1	I	AXI read address valid
S_AXI_ARREADY	1	O	AXI read address ready
S_AXI_RDATA	32	O	AXI read data issued by slave
S_AXI_RRESP	2	O	AXI read response. This signal indicates the status of the read transfer. ‘b00 = OKAY ‘b01 = EXOKAY ‘b10 = SLVERR ‘b11 = DECERR
S_AXI_RVALID	1	O	AXI read data valid
S_AXI_RREADY	1	I	AXI read ready. This signal indicates the user/master can accept the read data and response information.

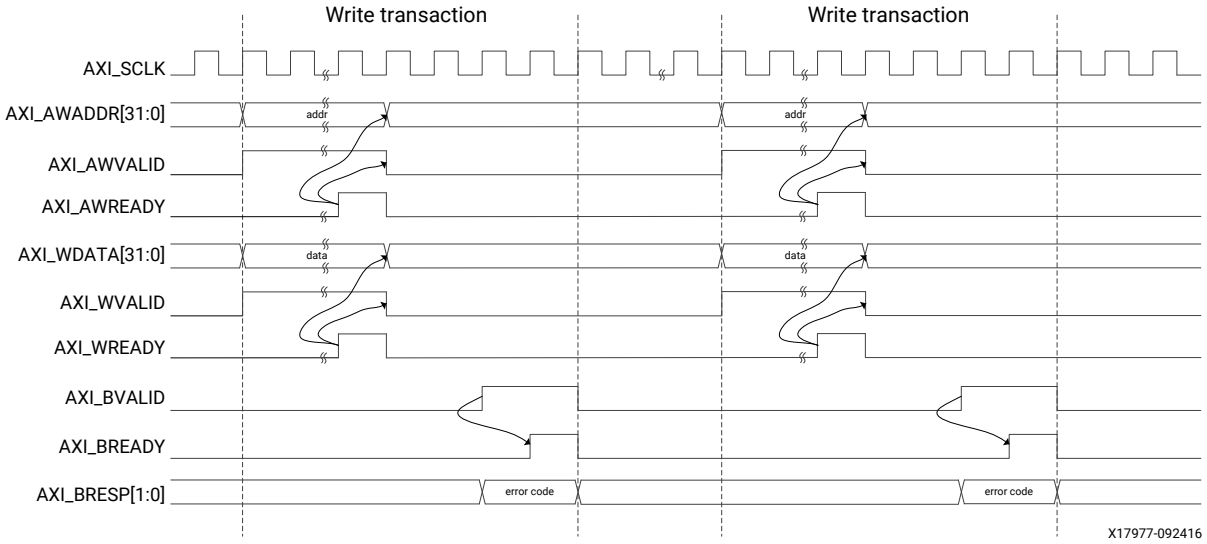
Valid Write Transactions

Figure 66: **AXI4-Lite User-Side Write Transaction**



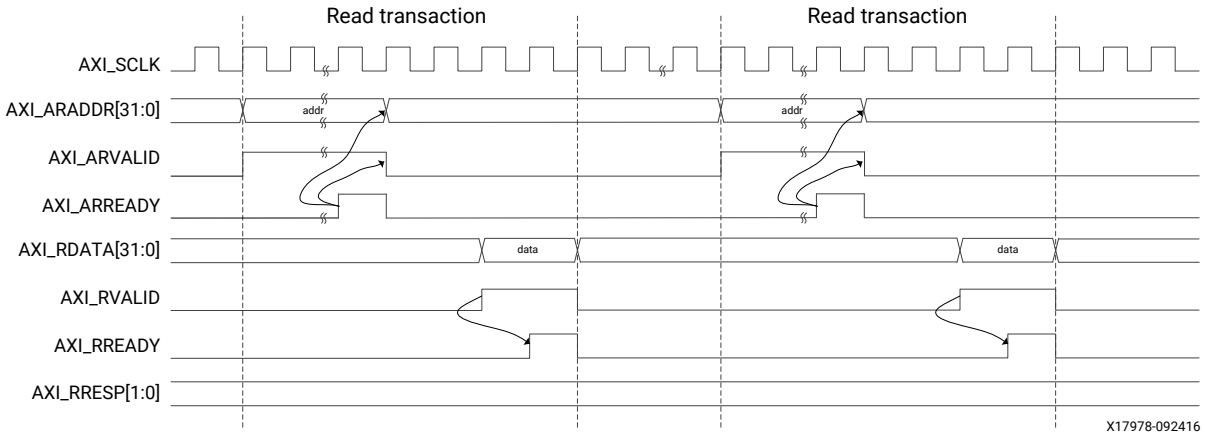
Invalid Write Transactions

Figure 67: **AXI4-Lite User Side Write Transaction with Invalid Write Address**



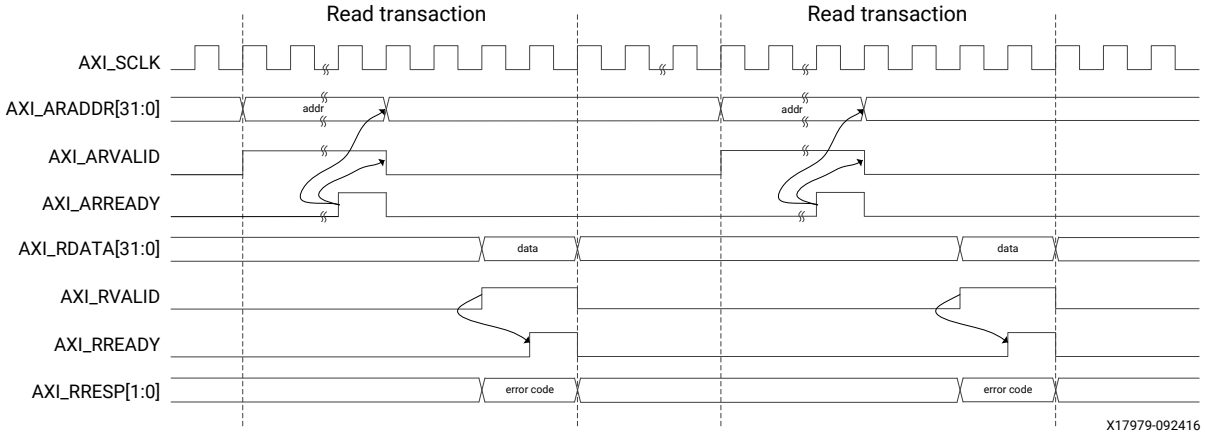
Valid Read Transactions

Figure 68: **AXI4-Lite User Side Read Transaction**



Invalid Read Transactions

Figure 69: **AXI4-Lite User Side Read Transaction with Invalid Read Address**



PTP 1588 Timer Syncer IP

Introduction

The PTP 1588 Timer Syncer IP provides reference time to all the Ethernet ports in the design. It implements an IEEE1588 real time clock timer and can accurately recreate 1588 timer in the local port's clock domain.

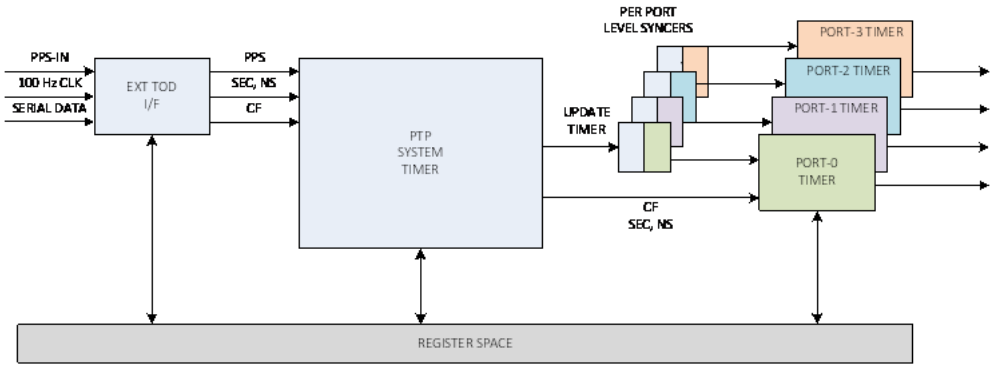
Features

- This IP can be configured as master system timer, master system, and as port timer.
- Supports ToD (Sec – NanoSec) and continuous/correction time formats (CF).
- Implements external ToD bus interface logic for synchronizing high precision clock sources.
- Crosses system ToD to port timer's clock domain.
- AXI4-Lite I/F for configuration and monitoring.

Core Overview

The following figure shows major functional blocks of the Timer Syncer IP. In this figure, only four instances of the port timer are shown as an example. The number of port instances are user selectable at the time of generating the core. This core supports up to 16 instances of port timers.

Figure 70: **Timer Syncer IP Functional Block Diagram**



The Timer Syncer IP contains all functions and interfaces needed to implement a variety of ToD topologies and applications. It can be controlled with either software or hardware devices. The System Timer maintains time on the free-running system time clock (`ts_clk`) and provides mechanism to synchronize the timer values with various other port timers, each of which may be clocked on their respective clocks (`phy_clk`).

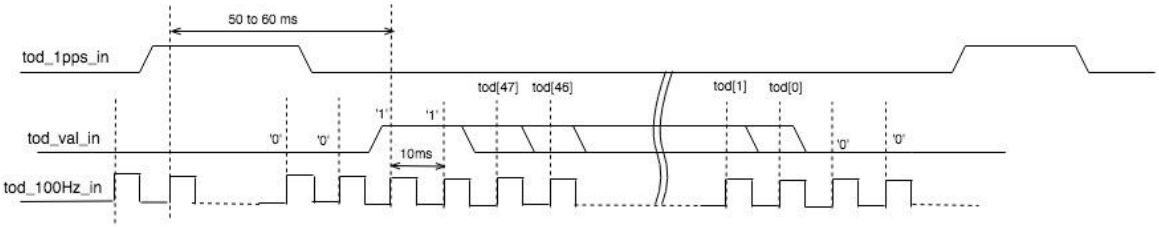
The System Timer IP provides timer values in two formats namely, Timestamp format (or ToD format) and Correction Field (CF) format. The Timestamp format is comprised of 80 bits containing fields of unsigned positive seconds and nanoseconds such as {seconds[47:0], nanoseconds[31:0] }. The Correction Field (CF) format is a signed 64b value in nanoseconds multiplied by 2^{+16}. Subsequent sections describe functional blocks and interfaces.

External ToD Interface Sub Block

This module is optional, it is present only in the example design when you select **Enable Ext ToD Bus I/F** option while generating the IP. Its function is to load the reference seconds value onto the PTP System Timer and synchronize the System Timer to an external 1PPS source. It is intended to interface with a high precision clock source/timing device.

Inputs to this module are 1PPS pulse, serial 1-bit data bus carrying reference ToD seconds time that is clocked by a serial clock input. The serial clock is expected to be less than or equal to the free-running clock (`ts_clk`) of the Timer Sync block. The process of loading the reference 48-bits second is shown in the following figure.

Figure 71: **ToD Value (seconds) Interface**



The External ToD I/F block stores serial input ToD seconds value in a holding register. At the next received positive edge of the 1PPS signal, the holding register's value is incremented by +1 second, and the result be presented to the PTP System Timer sub-block. The positive edge of the 1PPS signal is re-timed within the External ToD I/F block to the `ts_clk` clock and output for use by the PTP System Timer block.

PTP System Timer Sub Block

This is the master or the main ToD timer clocked by a free running clock (`ts_clk`). The system timer maintains time in the ToD (48-bit seconds and 30-bits nano-sec) and continuous time/correction field (63-bits CF) formats.

Registers are provided to initialize the timer seconds and nanoseconds counter values or to read back a snapshot of the values. A set of offset registers are provided for seconds and nanoseconds values which are added to the ToD timer value prior to being output to the port timers.

After initialization, the PTP System Timers internal ToD counters can be optionally synchronized with external devices by either the External ToD interface block's output 1PPS signal, or by the software control via register operations.

1PPS output indicates when the system timer's ToD ns field rolls over from 999_999_999 ns to 1 sec. Also, the values of system timer can be read from snapshot registers.

The process of transferring the system timer's ToD counters to the port timers is configurable and can be triggered by a 1PPS pulse of the external bus, or by a write to the TOD_SW_LOAD register. When a transfer is triggered, the PTP System Timer provides a load-pulse output (synchronous to the `ts_clk` domain) and places the value of its internal timer on the output bus.

PTP Port Timer Sub Block

These sub blocks contain per port TX and RX timers that are typically clocked by their respective TX and RX PHY clocks (`tx_phy_clk` and `rx_phy_clk`). The timer synchronizer IP allows for a maximum of sixteen port timers to be enabled at the time of IP generation.

These port timers maintain time in both the ToD (48-bit seconds and 30-bits nano-sec) and continuous time/correction field (63-bits CF) formats, and provide outputs synchronized to the `tx_phy_clk` and `rx_phy_clk` for use by Xilinx's Ethernet IP.

It is expected that each port's TX and RX clock domains can be asynchronous with respect to the Master timer's `ts_clk` clock. The port timers contain synchronization logic to domain cross the PTP System Timer's load-pulse and timer update values.

As the PTP System Timer is initialized, or synchronized to a high precision reference clock, it in turn pushes its updated timer value to all the port timers that are connected to it. After initialization, the Timer Syncer IP can be configured such that the port timers are continuously kept in-sync with the PTP System Timer's master ToD value, or the port timers may be independently controlled.

Port Descriptions

Timer Syncer IP Interface Ports

Following are the top level Timer Syncer IP ports:

Clock Reset

Table 68: **Clock Reset**

Signal	Direction	Clock Domain	Description
ts_clk	IN	N/A	It is a free running clock which clocks system timer's counters
ts_rst	IN	ts_clk	System timer reset active-High
tod_intr	OUT	ts_clk	Interrupt asserted on 1-PPS event

External ToD Bus Interface

Table 69: **External ToD Bus Interface**

Signal	Direction	Clock Domain	Description
tod_1pps_in	IN	N/A	External bus 1-PPS input
tod_sec_clk_in	IN	N/A	External bus clock, used for tod_sec_in
tod_sec_in	IN	tod_sec_clk_in	External bus seconds serial data
tod_1pps_out	Output	ts_clk	1-PPS output Asserted when the system timer's nano-second field rolls over
tod_sec_clk_out	IN	N/A	Echo of external bus clock
tod_sec_out	IN	tod_sec_clk_in	Serial output of internal ToD seconds

Per-Port Port Timer Interface

Table 70: **Per-Port Port Timer Interface**

Signal	Direction	Clock Domain	Description
tx_phy_clk_m	IN		Port TX PHY clock
rx_phy_clk_m	IN		Port RX PHY clock
tx_phy_rst_m	IN	tx_phy_clk_m	Port TX reset
rx_phy_rst_m	IN	rx_phy_clk_m	Port RX reset
tx_tod_sec_m[47:0]	OUT	tx_phy_clk_m	Port TX timer seconds field
tx_tod_ns_m[31:0]	OUT	tx_phy_clk_m	Port TX timer nano-seconds field
tx_tod_corr_m[63:0]	OUT	tx_phy_clk_m	Port TX timer CF field
rx_tod_sec_m[47:0]	OUT	rx_phy_clk_m	Port RX timer seconds field
rx_tod_ns_m[31:0]	OUT	rx_phy_clk_m	Port RX timer nano-seconds field
rx_tod_corr_m[63:0]	OUT	rx_phy_clk_m	Port RX timer CF field

AXI4-Lite Interface

Table 71: **AXI4-Lite Interface**

Signal	Direction	Clock Domain	Description
s_axi_aclk	IN		AXI4-Lite I/F clock
s_axi_aresetn	IN	s_axi_aclk	AXI4-LiteI/F reset
s_axi_awaddr[31:0]	IN	s_axi_aclk	Write address
s_axi_awvalid	IN	s_axi_aclk	Write address Valid
s_axi_awready	OUT	s_axi_aclk	Write address Ready
s_axi_wdata[31:0]	IN	s_axi_aclk	Write data

Table 71: **AXI4-Lite Interface** *(cont'd)*

Signal	Direction	Clock Domain	Description
s_axi_wstrb[3:0]	IN	s_axi_aclk	Write data byte valid. Tie to 4'hF
s_axi_wvalid	IN	s_axi_aclk	Write valid
s_axi_wready	OUT	s_axi_aclk	Write ready
s_axi_bresp	OUT	s_axi_aclk	Write response
s_axi_bvalid	OUT	s_axi_aclk	Write response valid
s_axi_bready	IN	s_axi_aclk	Write response ready
s_axi_araddr[31:0]	IN	s_axi_aclk	Read address
s_axi_arvalid	IN	s_axi_aclk	Read address valid
s_axi_arready	OUT	s_axi_aclk	Read address ready
s_axi_rdata[31:0]	OUT	s_axi_aclk	Read data
s_axi_rresp	OUT	s_axi_aclk	Read response
s_axi_rvalid	OUT	s_axi_aclk	Read data valid
s_axi_rready	IN	s_axi_aclk	Read ready

Internal Sub-Block Ports

Following are the select ports present on the IP sub-blocks to assist you with debugging:

External ToD Bus Ports

Table 72: **External ToD Bus Ports**

Signal	Direction	Clock Domain	Description
ext_tod_bus_sec	OUT	ts_clk	Seconds value sent to PTP system timer
ext_tod_bus_ns	OUT	ts_clk	Nanoseconds value sent to PTP system timer
ext_tod_bus_1pps	OUT	ts_clk	1PPS sent to PTP system timer

PTP System Timer Ports

Table 73: **PTP System Timer Ports**

Signal	Direction	Clock Domain	Description
sys_tod_sec[48:0]	OUT	ts_clk	System timer ToD seconds field
sys_tod_ns[31:0]	OUT	ts_clk	System timer ToD nano-seconds field
sys_tod_corr[63:0]	OUT	ts_clk	System timer ToD CF format

Table 73: **PTP System Timer Ports** *(cont'd)*

Signal	Direction	Clock Domain	Description
update_timer	OUT	ts_clk	Sync update pulse to the port timer blocks Asserted when the master timer is synchronized either by the Ext ToD I/F or via register updates
sys_timer_1pps_out	OUT	ts_clk	1-PPS output to external ToD bus block This port is asserted when system timer's nano-second field rolls over

Port Timer Ports

Table 74: **Port Timer Ports**

Signal	Direction	Clock Domain	Description
update_port_timer	IN	ts_clk	Sync update pulse driven by system timer
sys_tod_sec_in[48:0]	IN	ts_clk	System timer seconds field Present only when the timer format is ToD or both
sys_tod_ns_in[31:0]	IN	ts_clk	System timer nano-seconds field Present only when the timer format is ToD or both
sys_tod_corr_in[63:0]	IN	ts_clk	System timer CF field. Present only when the timer format is CF or both

Register Space

All the control and status registers are memory mapped. After power-up or reset, you can reconfigure the core parameters from their defaults at any time.

Register Access Type Definitions

Following are the register access type definitions:

Table 75: **Register Access Type Definitions**

Access Type	Description
RO	Read only Readable register; write has no effect
RW	Read and write Readable and writable register

Table 75: **Register Access Type Definitions** *(cont'd)*

Access Type	Description
RW1T	Read and write. '1' to trigger Write '1' to trigger. Write '0' is ignored. Read returns '0'.
RW1C	Read, write '1's to Clear Writing a '1' clears the corresponding bit position in the register to '0'; Writing a '0' leaves the corresponding bit unchanged For example, it can be used to acknowledge interrupt status.
WO	Write only Writable register, the value is not readable (returns '0')

Register Map

Following are the register Map descriptions:

Table 76: **Register Map**

Offset	Register Name	Access	Description
0x0000	TOD_CONFIG	RW	Main configuration [0] – Enable system timer A '0' to this bit field disables system timer IP [1] – Enable external ToD bus Write '1' to enable ToD bus signals. The overwrite mode field further defines how signaling is used This is present only when the core is generated with external ToD bus I/F support [3:2] – Overwrite Mode: • 0x0 - System timer counter is not overwritten at 1-PPS event from external ToD bus • 0x1 - system timer counter is overwritten with the external ToD bus seconds input at 1-PPS event from external ToD bus • 0x2 - system timer counter is overwritten with value stored in the software TOD_SW_SEC_0/1 register at 1-PPS event from the external ToD bus • 0x3 - Reserved The above modes are only present when the core is generated with Ext ToD bus interface support [4 to 19] – Enable port TX and RX timers • [4] = Enable Port-0 TX and RX • [5] = Enable Port-1 TX and RX • [16] = Enable Port-15 TX and RX The upper limit for port number depends on the number of ports enabled at the time of generating core [31:20] - Reserved
0x0004	TOD_SNAPSHOT	RW1T	[0] – Snapshot all timers Writing 1'b1 will snapshot all counters (system, external ToD bus, and all enabled ports) [31:1] - Reserved

Table 76: **Register Map** *(cont'd)*

Offset	Register Name	Access	Description
0x0008	TOD_INTR_ENABLE	RW	Interrupt enable register [0] - 1-PPS interrupt [31:1] - Reserved
0x000C	TOD_INTR_STATUS	RW1C	Interrupt clear register [0] - 1-PPS interrupt [31:1] - Reserved
0x0010	TOD_SW_SEC_0	RW	[31:0] - Overwrite master timer's second field bits [31:0]
0x0014	TOD_SW_SEC_1	RW	[15:0] - Overwrite master timer's second field bits [47:32] [31:16] - Reserved
0x0018	TOD_SW_NS	RW	[29:0] - Overwrite master timer's second field bits [31:30] - Reserved
0x001C	TOD_SW_LOAD	RW1T	[0] - Write '1' initiates a load of the system timer's ToD values from the TOD_SW_SEC_0/1, TOD_SW_NS, and TOD_SW_CTIME_0/1 registers [1] - Write '1' initiates a load of the system timer's ToD offset value from the TOD_SEC_SYS_OFFSET_0/1, and TOD_NS_SYS_OFFSET_0 registers ***Note*:** Offset is added by logic prior to system timer's output to the port timers. As such, offset is not reflected by system timer ToD read backs. [31:0] - Reserved
0x0020	TOD_SW_CTIME_0	RW	[31:0] - Overwrite master timer's CF field bits [31:0]
0x0024	TOD_SW_CTIME_1	RW	[30:0] - Overwrite master timer's second field bits [63:32] [31] - Reserved
0x0028	TOD_SEC_SYS_OFFSET_0	RW	{TOD_SEC_SYS_OFFSET_1[15:0], TOD_SEC_SYS_OFFSET_0[31:0]} - Represents system timer 48b seconds field signed offset value TOD_NS_SYS_OFFSET_0[29:0] - Represents system timer 30b nano second field signed offset value Signed bit interpreted as follows for TOD_SEC_SYS_OFFSET If [47] = 1'b1 then subtract from system timer If [47] = 1'b0 then add to system timer Need to apply trigger bit [1] at register 0x001C
0x002C	TOD_SEC_SYS_OFFSET_1	RW	
0x0030	TOD_NS_SYS_OFFSET_0	RW	
0x0100	TOD_SYS_SEC_0	RO	[31:0] - Snapshot of system timer's second field [31:0]
0x0104	TOD_SYS_SEC_1	RO	[15:0] - Snapshot of system timer's second field [47:32] [31:16] - Reserved
0x0108	TOD_SYS_NS	RO	[29:0] - Snapshot of system timer's Nano-second Field [29:0] [31:30] - Reserved
0x010C	TOD_SYS_OFFSET	RW	[31:0] - Signed offset to be applied to seconds value loaded from the external ToD bus, and to be added to 0 nanoseconds field when a 1PPS event occurs on the External ToD bus. The signed value is expressed in 2^{-16} ns This is present only when the core is generated with Ext ToD Bus support
0x0110	TOD_SYS_CTIME_0	RO	[31:0] - Snapshot of system timer's CF Field [31:0]
0x0114	TOD_SYS_CTIME_1	RO	[30:0] - Snapshot of system timer's CF Field [63:32] [31] - Reserved

Table 76: **Register Map** *(cont'd)*

Offset	Register Name	Access	Description
0x0120	TODBUS_SEC_0	RO	Current value of the Ext ToD bus's second Field [31:0]
0x0124	TODBUS_SEC_1	RO	[15:0] - Current value of the Ext ToD bus's second field [47:32] [31:16] - Reserved
0x012C	TODBUS_SYS_DIFF	RO	[31:0] - Once a second comparison of external ToD bus captured value and system Timer's internal ToD value in signed unit of 2^{-16}ns

Port Timer Registers

Port timer register set is replicated for every port timer present (0 to 15) at the offsets listed for ports 1 to 15. Following are the port timer registers description:

Table 77: **Port Timer Registers**

Offset	Register Name	Access	Description
Port 0 Registers: 0x0200 to 0x023F			
0x0200	TX0_CTIME_0	RW	[31:0] - Snapshot of Port0 TX Timer's CF Field [31:0]
0x0204	TX0_CTIME_1	RW	[30:0] - Snapshot of Port0 TX Timer's CF Field [63:32] [31] - Reserved
0x0208	TX0_PERIOD_0	RW	Port0 TX clock period expressed in 2^{-48} ns For example, 3.2 ns is represented as TX0_PERIOD_0[31:0] = 0x3333_3333 TX0_PERIOD_1[23:0] = 0x0003_3333 TX0_PERIOD_1[31:24] reserved
0x020C	TX0_PERIOD_1	RW	
0x0210	TX0_SYS_OFFSET	RW	[31:0] - Signed offset applied to Port0 TX Timer's ToD output expressed in 2^{-16} ns.
0x0214	TX0_NS_SNAP	RO	[29:0] - Snapshot of Port0 TX Timer's Nano-second Field [29:0] [31:30] - Reserved
0x0218	TX0_SEC_0_SNAP	RO	[31:0] - Snapshot of Port0 TX Timer's Second Field [31:0]
0x021C	TX0_SEC_1_SNAP	RO	[15:0] - Snapshot of Port0 TX Timer's Second Field [47:32] [31:16] - Reserved
0x0220	RX0_CTIME_0	RW	[31:0] - Snapshot of Port0 RX Timer's CF Field [31:0]
0x0224	RX0_CTIME_1	RW	[30:0] - Snapshot of Port0 RX Timer's CF Field [63:32] [31] - Reserved

Table 77: **Port Timer Registers** *(cont'd)*

Offset	Register Name	Access	Description
0x0228	RX0_PERIOD_0	RW	Port0 RX clock period expressed in 2^{-48} ns For example, 3.2 ns is represented as RX0_PERIOD_0[31:0] = 0x3333_3333 RX0_PERIOD_1[23:0] = 0x0003_3333 RX0_PERIOD_1[31:24] reserved
0x022C	RX0_PERIOD_1	RW	
0x0230	RX0_SYS_OFFSET	RW	[31:0] - Signed offset applied to Port0 RX Timer's ToD output expressed in 2^{-16} ns.
0x0234	RX0_NS_SNAP	RO	[29:0] - Snapshot of Port0 RX Timer's Nano-second Field [29:0] [31:30] - Reserved
0x0238	RX0_SEC_0_SNAP	RO	[31:0] - Snapshot of Port0 RX Timer's Second Field [31:0]
0x023C	RX0_SEC_1_SNAP	RO	[15:0] - Snapshot of Port0 RX Timer's Second Field [47:32] [31:16] - Reserved
Port 1 Registers: 0x0240 to 0x027F			
Port 2 Registers: 0x0280 to 0x02BF			
Port 3 Registers: 0x02C0 to 0x02FF			
Port 4 Registers: 0x0300 to 0x033F			
Port 5 Registers: 0x0340 to 0x027F			
Port 6 Registers: 0x0380 to 0x03BF			
Port 7 Registers: 0x03C0 to 0x03FF			
Port 8 Registers: 0x0400 to 0x043F			
Port 9 Registers: 0x0440 to 0x047F			
Port 10 Registers: 0x0480 to 0x04BF			
Port 11 Registers: 0x04C0 to 0x04FF			
Port 12 Registers: 0x0500 to 0x053F			
Port 13 Registers: 0x0540 to 0x057F			
Port 14 Registers: 0x0580 to 0x05BF			
Port 15 Registers: 0x05C0 to 0x05FF			

Software Driver Core Initialization

Ensure that the software drivers follow these steps to properly initialize the core:

1. Initialize other elements of the design such that the clocks including `ts_clk` and `tx/rx_phy_clk*` are present and stable. Release resets for these clock domains such as `ts_rst`, `tx/rx_phy_rst*`.
2. Configure `TOD_CONFIG` register:
   a. In Timer or Timer Syncer mode set bit [0] to enable system timer.

b. If an external ToD bus (1PPS synchronization and optionally serial seconds input) is to be used, then set bit [1] to 1.

c. Set the mode bit field [3:2] to the setting matching the intended synchronization method used in your system. For example, if your system uses an external device connected to the External ToD bus (1PPS input and second's value serial input). The mode field should be set to 0x1.

d. Enable (set to 1) appropriate bits of the Port Timer enable bitfield [19:4] to enable the port TX and RX timers.

3. If the system timer's initial value is to be set by the software via register programming, an alternative to the external ToD bus interface, then write appropriate initial values to the software loading registers such as `TOD_SW_SEC_0/1` and `TOD_SW_NS`, `TOD_SW_CTIME_0/1`. Also configure any static offset to be loaded by writing the `TOD_SEC_SYS_OFFSET_0/1` and `TOD_NS_SYS_OFFSET_0` registers.
4. If the software registers are updated in Step 2, then a write of 1 to the appropriate bits of `TOD_SW_LOAD[1:0]` triggers system timer to load software values.
5. Program port TX/RX Timer. This includes the period values and any required static offset to the appropriate values by writing to the Port timer's registers: TX<M>_PERIOD_0/1, RX<M>_PERIOD_0/1, and TX<M>/RX<M>_SYS_OFFSET.

Continue step 4 for all Ports (M) present in the design.

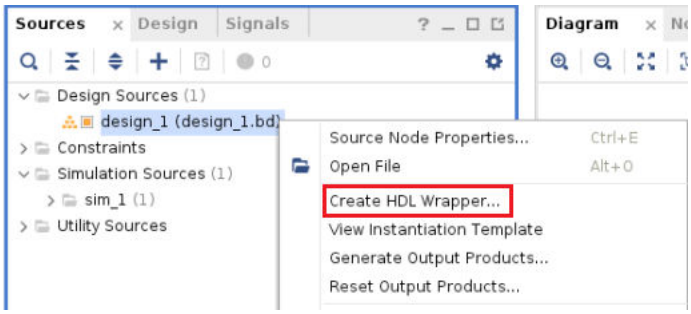
Instantiating the IP

The Timer Syncer IP is a hidden IP and therefore you need to use the following Tcl commands for instantiating the IP:

1. ```
   create_bd_design "design_1"
   ```
2. ```
   set_param bd.skipSupportedIPCheck true
   ```
3. ```
   set ptp_1588_timer_syncer_0 [ create_bd_cell -type ip -vlnv
   xilinx.com:ip:ptp_1588_timer_syncer ptp_1588_timer_syncer_0 ]
   ```
4. ```
   set_property -dict [ list \
   CONFIG.AXI4LITE_FREQ {100.00} \
   CONFIG.CORE_MODE {Timer_Syncer} \
   CONFIG.ENABLE_EXT_TOD_BUS {0} \
   CONFIG.ENABLE_HIGH_MODE {1} \
   CONFIG.NUM_PORTS {1} \
   CONFIG.TIMER_FORMAT {Time_of_Day} \
   CONFIG.TS_CLK_PERIOD {4.0} \
   ] $ptp_1588_timer_syncer_0
   ```

   Note: TS_CLK_PERIOD : 4.0 (250MHz).
5. ```
   startgroup
   ```
6. ```
   make_bd_pins_external  [get_bd_cells ptp_1588_timer_syncer_0] -quiet
   ```

7. `make_bd_intf_pins_external [get_bd_cells ptp_1588_timer_syncer_0] -quiet`
8. `endgroup`
9. `assign_bd_address`
10. `validate_bd_design`
11. `save_bd_design`
12. Click **Create HDL Wrapper**.



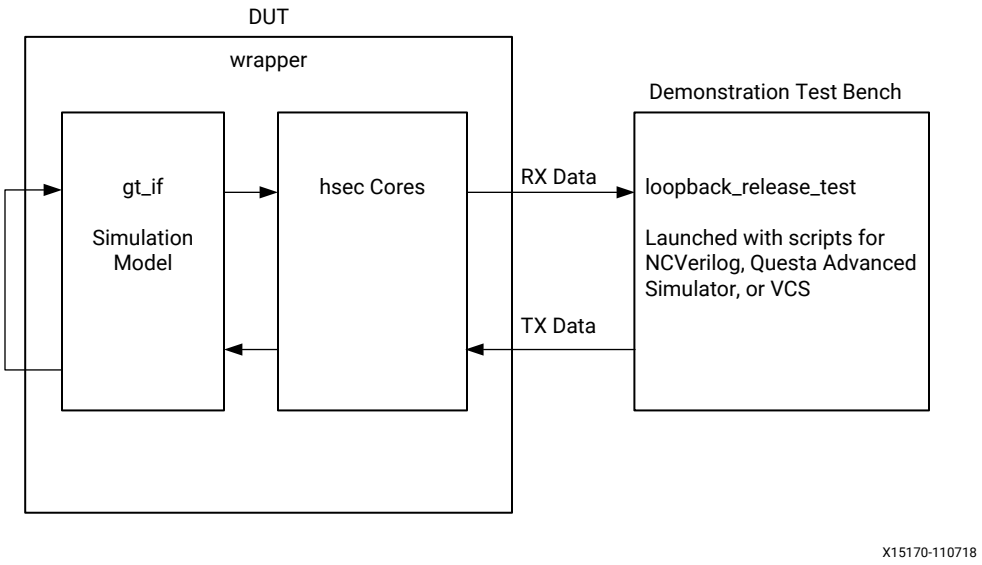
13. `update_compile_order -fileset sources_1`
14. Instantiate the design_1_wrapper() in the design and connect it appropriately with the Ethernet IP and AXI4-Lite interface.

    Note: When you select Enable Timestamping Logic from the GUI Tab-2 of the XXV ethernet IP, the Timer Syncer IP is generated automatically and appropriately connected in the example design. For reference, you can generate example design of the XXV Ethernet IP core.

Batch Mode Test Bench

Each batch mode release of the 1G/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 72: **Test Bench**



Appendix A

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 for more details.*

Finding Help on Xilinx.com

To help in the design and debug process when using the subsystem, 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. The Xilinx Community Forums are also available where members can learn, participate, share, and ask questions about Xilinx solutions.

Documentation

This product guide is the main document associated with the subsystem. 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 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.

Technical Support

Xilinx provides technical support on the Xilinx Community Forums for this LogiCORE™ IP 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 ask questions, navigate to the Xilinx Community Forums.

Migrating from the Legacy XGEMAC

This section contains information about upgrading from the legacy version of the Xilinx 10G EMAC IP to the new 1G/10G/25G Switching Ethernet Subsystem IP core.

***Note*:** For all new designs, Xilinx recommends that you use the most recent version of the 1G/10G/25G Switching Ethernet Subsystem IP Core.

The Xilinx 10G Ethernet MAC and the 1G/10G/25G Switching Ethernet Subsystem 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 1G/10G/25G Switching Ethernet Subsystem IP core. For a list of new features or features exclusive to the new IP, refer to Chapter 3: Product Specification.

The following features are different in the new1G/10G/25G Switching Ethernet Subsystem IP.

- Padding. The Pad field is not added by the 1G/10G/25G Switching Ethernet Subsystem 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). The 10G/25G High Speed Ethernet IP has the DIC always enabled.
- Management Data Input/Output (MDIO) master. The 1G/10G/25G Switching Ethernet Subsystem 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. 1G/10G/25G Switching Ethernet Subsystem IP requires you to control if LF/RF are transmitted. You must provide the fault status signals as well.
- VLAN The new 1G/10G/25G Ethernet Subsystem 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 1G/10G/25G Switching Ethernet Subsystem 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 1G/10G/25G Switching Ethernet Subsystem 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 1G/10G/25G Switching Ethernet Subsystem IP does not pause TX packet transmission when global pause frames are received this is left to user logic.

For more details, see *10G/25G High Speed Ethernet Subsystem Product Guide* (PG210).

Debug Tools

There are many tools available to address 1G/10G/25G Ethernet Subsystem 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 to interact with the logic debug LogiCORE 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).

Reference Boards

Various Xilinx development boards support the 1G/10G/25G Ethernet Subsystem. These boards can be used to prototype designs and establish that the core can communicate with the system.

- **UltraScale FPGA evaluation boards :**
  - ZCU102

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 to speed up wait timers in the example design.

Simulation Fails Before Completion

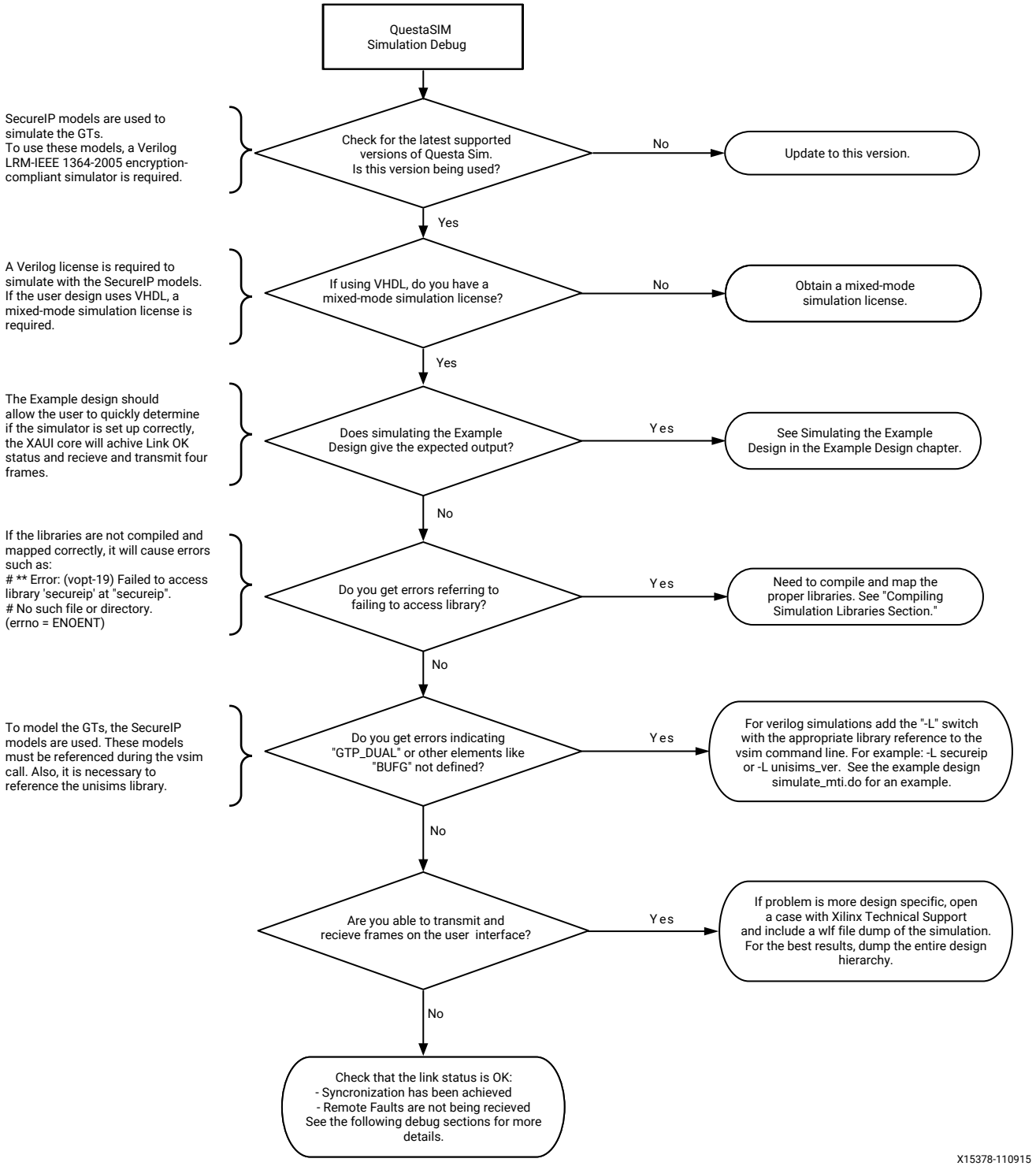
If the sample simulation fails or hangs before successfully completing, it is possible that a timeout has occurred. Ensure that the simulator timeouts are long enough to accommodate the waiting periods in the simulation, for example during the lane alignment phase.

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® Advanced Simulator is illustrated in the following figure. A similar approach can be used with other simulators.

Figure 73: **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 1G/10G/25G Switching Ethernet Subsystem. These should be checked as indicated below.

It is assumed that the 1G/10G/25G Ethernet Subsystem 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 1G/10G/25G Ethernet Subsystem 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 1G/10G/25G Ethernet Subsystem 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 1G/10G/25G Ethernet Subsystem. 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 for these requirements.

Ensure that the clock frequencies for both the 1G/10G/25G Ethernet Subsystem 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 1G/10G/25G Ethernet Subsystem 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 1G/10G/25G Ethernet Subsystem 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 1G/10G/25G Ethernet Subsystem 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 IEEE Std. 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 1G/10G/25G Ethernet Subsystem, 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.

Debugging Auto-Negotiation and Link Training

Auto-Negotiation

To enable auto-negotiation:

- `ctl_autoneg_enable` = 1
- `ctl_autoneg_bypass` = 0

Set `ctl_an_*` to advertise desired auto-negotiation settings.

When using the control and status interface the example design ties off the `ctl_an_*` values to valid settings. If using the register interface see Board Testing Steps for Auto-Negotiation and Link Training Using AXI4-Lite Interface for the register sequence.

Link Training

To enable link training set `ctl_lt_training_enable` to 1.

- The core does not actually do any training. It only provides the control protocol required by section 72.6.10. The training algorithm is a user responsibility.
- The core does not monitor the RX eye nor does it send any presets, initializations, or coefficient control requests to the link partner TX. It is recommended to set `ctl_lt_rx_trained` to 1. Setting `ctl_lt_rx_trained` tells the link partner that your RX training is completed, and that you will not be sending any more presets, initializations, or coefficient changes.
- The core does not adjust any of the GT TX amplitude or coefficient control settings in response to any training messages received from the link partner. The example design link training Place_Holder logic will indicate that maximum limits have been reached. This should allow link training to complete.

Nonce

`nonce_seed` must be set to a non-zero value.

- If connecting two ports with same nonce seed on the same board, resets must be released at different times.

- If the nonce_seed is changed, an `an_reset` is needed to load the new value. This includes changing `nonce_seed` using the AXI4-Lite registers.

Next Pages

If the link partner sends next page, `ctl_an_loc_np_ack` must be set High to acknowledge the next page and allow auto-negotiation to complete. This control signal can be set High after next page is received or tied always High.

Details on Stages and Status Signals

1. At the start of auto-negotiation there is a TX disable state where no data is seen to ensure that the link is down on both sides. The `stat_an_stat_tx_disable` signal toggles for one cycle to indicate the start of this stage.
2. Following the TX disable state, auto-negotiation information is exchanged. During this stage `stat_an_rxcdrhold` is held High. The `stat_an_lp_autoneg_able` and `stat_an_lp_ability_valid` signals will toggle High for one clock cycle to indicate when `stat_an_lp*` information is valid.
3. The `stat_an_start_an_good_check` signal toggles High for one cycle at the start of link training. The `stat_an_rxcdrhold` signal is deasserted and `gtwiz_reset_rx_datapath` toggled. After link training starts there is a 500 ms timer for training and block lock/link up in mission mode/normal PCS operation to complete or auto-negotiation will restart. The `stat_lt_frame_lock` signal goes High and `stat_lt_rx_sof` toggles when the link training block has achieved frame synchronization. The `stat_lt_rx_sof` signal continues to toggle High for one clock duration at the training frame boundary.
4. When link training completes the `stat_lt_signal_detect` signal asserts and indicates the start of normal PCS operation.
5. The `an_autoneg_complete` signal goes High when block lock, synchronization and alignment (if multi-lane core), `stat_rx_status` and `stat_rx_valid_ctrl_code` (`stat_rx_valid_ctrl_code` is only applicable to single lane 10G/25G core) go High.
6. The `an_autoneg_complete` signal must go High within the 500 ms timeout or auto-negotiation will restart. If `stat_rx_status` goes back Low at any time then auto-negotiation restarts.

Simulation and Loopback

Auto-Negotiation TX disable state takes 50 ms of simulation time to complete. Use SIM_SPEED_UP option without pre-compiled IP libraries to speed up the wait time. See AR 73518 for more information on turning off pre-compiled libraries.

Auto-negotiation will not complete in loopback because auto-negotiation requires that the nonce value received from the link partner must be different than the nonce value sent to the link partner.

Starting signal list to add to ILA for debug:

- sys_reset
- an_reset
- ctl_an_*
- ctl_lt_*
- stat_an_start_tx_disable
- stat_an_rxcdrhold
- stat_an_lp_autoneg_able
- stat_an_lp_ability_valid
- stat_an_start_an_good_check
- stat_lt_frame_lock
- stat_lt_signal_detect
- stat_lt_link_training
- stat_lt_link_training_fail
- stat_rx_block_lock
- stat_rx_synced (only available on multi-lane cores)
- stat_rx_aligned (only available on mult-lane cores)
- stat_rx_valid_ctrl_code (only available on 10G/25G core)
- stat_rx_status
- stat_rx_bad_code
- stat_rx_hi_ber

If using line rate that supports Clause 74 Firecode FEC:

- stat_fec_inc_cant_correct_count
- stat_fec_lock_error
- stat_fec_rx_lock
- stat_fec_inc_correct_count
- ctl_an_fec_10g_request
- ctl_fec_rx_enable
- ctl_fec_tx_enable
- stat_an_fec_enable
- stat_an_lp_fec_10g_ability
- stat_an_lp_fec_10g_request

If using line rate that supports RS-FEC:

- ctl_tx_rsfec_enable
- ctl_rx_rsfec_enable
- stat_rx_rsfec_am_lock
- stat_an_rs_fec_enable

***Note*:** If the link partner sends next pages, the `ctl_an_lo_np_ack` signal must be set. This port can be tied High.

Debugging Auto-Negotiation and Link Training Using the AXI4-Lite Interface

To debug auto-negotiation and link training using the AXI4-Lite interface, see Board Testing Steps for Auto-Negotiation and Link Training Using AXI4-Lite Interface.

Appendix B

Additional Resources and Legal Notices

Xilinx Resources

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

Documentation Navigator and Design Hubs

Xilinx® Documentation Navigator (DocNav) provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open DocNav:

- From the Vivado® IDE, select **Help → Documentation and Tutorials**.
- On Windows, select **Start → All Programs → Xilinx Design Tools → DocNav**.
- At the Linux command prompt, enter `docnav`.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In DocNav, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

Note: For more information on DocNav, see the Documentation Navigator page on the Xilinx website.

References

These documents provide supplemental material useful with this product guide:

1. *25/50G Gigabit Ethernet Consortium Schedule 3 (v1.6)* (http://25gethernet.org/)
2. *IEEE Standard for Ethernet* (IEEE Std 802.3-2015)
3. *IEEE Standard for Ethernet - Amendment 2: Media Access Control Parameters, Physical Layers, and Management Parameters for 25 Gb/s Operation* (IEEE Std 802.3by)
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 1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems* (IEEE 1588)
12. *UltraScale Architecture GTH Transceivers User Guide* (UG576)
13. *UltraScale Architecture GTY Transceivers User Guide* (UG578)
14. *10G/25G High Speed Ethernet Subsystem Product Guide* (PG210)
15. *1G/2.5G Ethernet PCS/PMA or SGMII LogiCORE IP Product Guide* (PG047)
16. *AXI Interconnect LogiCORE IP Product Guide* (PG059)
17. Arm AMBA AXI Protocol v2.0 Specification (Arm IHI 0022C) (http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ihi0022c/index.html)

Revision History

The following table shows the revision history for this document.

Section	Revision Summary
02/05/2021 Version 2.6	
General updates	• Added support for Versal ACAP. • Added support for RSFEC(Clause 108) for MAC+PCS/PMA 64-bit. • Added Preemption support for MAC+PCS/PMA 64-bit for non-Versal device.
Register Space	Added new registers.
Register Descriptions	• Added GIG_EHERNET_PCS_PMA_CONFIGURATION_VECTOR:0x1000. • Added GIG_ETHERNET_PCS_PMA_STATUS_VECTOR:0x1004
LogiCORE Example Design Clocking and Resets	Updated figures.
802.1cm Preemption Feature	Added.
Port Descriptions	Updated.
Customizing and Generating the Subsystem	Updated figures.
06/03/2020 Version 2.5	
STAT_RX_STATUS_REG1: 0404	Updated bit 0 Type.
LogiCORE Example Design Clocking and Resets	Reset signal name updated in figures.
Board Testing Steps for Auto-Negotiation and Link Training Using AXI4-Lite Interface	Clarified text.
Customizing and Generating the Subsystem	Screenshots updated.
Debugging Auto-Negotiation and Link Training	New debugging topics added.
10/30/2019 Version 2.4	
Link Training	Added the board testing steps of Ethernet Subsystem core for Auto Negotiation and Link Training with AXI4-Lite Interface.
Configuration Tab	Updated the Notes section with STATISTICS_COUNTERS_SIZE parameter details.
MAC Options Tab	Updated to include the Include System Timer Syncers options.
Appendix A: Debugging	Added a new section Migrating from the Legacy XGEMAC.
05/22/2019 Version 2.3	
Auto-Negotiation (Clause 37)	Added Auto-Negotiation Clause 37 information.
Miscellaneous Status/Control Signals	Added signal_detect port.
Configuration Tab	Updated configuration options table with Statistics Resource Type, Include Statistics Counters, and updated Auto-Negotiation options (Clause 73 and Clause 37).
12/05/2018 Version 2.2	
XGMII Interfaces	Added timing diagrams and descriptions.
Using the Client-Side GMII Interface	Added timing diagrams and descriptions.
Link Training Ports	Added Link Training Ports table.
IEEE 802.3 Clause 74 FEC Interface	Added IEEE 802.3 Clause 74 FEC Interface Control/Status/ Statistics Signals table.
Pause Interface	Added Pause Interface I/O ports.

Section	Revision Summary
Auto-Negotiation and Link Training Clocking	Added Auto-Negotiation Clocking Architecture figure.
Pause Processing	Added Pause Processing section.
Link Training	Added Link Training section.
Configuration Register Map 1G/10G/25G Ethernet Subsystem	Added items to Configuration Register Map.
Status Register Map for 1G/10G/25G Ethernet Subsystem	Added items to Status Register Map.
Statistics Counters	Added items to Statistics Counters.
Configuration Tab	Added further PCS/PMA options.
MAC Options Tab	Added Flow Control section.
Include GT Subcore in Example Design Ports	New table for ports available when the Include GT subcore in Example Design option is selected.
TX Pause Interface Control/Status/Statistics Signals RX Pause Interface Control/Status/Statistics Signals Clause 74 FEC Interface Control/Status/Statistics Signals	New tables for TX Pause Interface Control/Status/Statistics Signals, RX Pause Interface Control/Status/Statistics Signals, and Clause 74 FEC Interface Control/Status/Statistics Signals in the Core XCI Top Level Port List.
06/06/2018 Version 2.1	
General updates Auto-Negotiation (Clause 37) and Auto-Negotiation Ports Auto-Negotiation and Link Training Clocking AN Interface Control/Status/Statistics Signals	• Updated the Example Design Clocking and Reset diagrams. • Added Optional Clause 73 Auto-Negotiation with Parallel Detection support. • Added the clocking architecture for the Auto-Negotiation function. • Added AN Interface Control / Status / Statistics Signals table.
04/04/2018 Version 2.0	
General updates	Initial release.
12/20/2017 Version 1.0	
General updates	Xilinx Confidential. Initial release under NDA only.

Please Read: Important Legal Notices

action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos.

This document contains preliminary information and is subject to change without notice. Information provided herein relates to products and/or services not yet available for sale, and provided solely for information purposes and are not intended, or to be construed, as an offer for sale or an attempted commercialization of the products and/or services referred to herein.

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.

Copyright