40G/50G High Speed Ethernet Subsystem v3.1

LogiCORE IP Product Guide

Vivado Design Suite

PG211 (v3.1) June 3, 2020

XILINX®

Table of Contents

XILINX

Chapter 1

Introduction

The Xilinx® High Speed Ethernet IP Subsystem implements the 40G or 50G Ethernet Media Access Controller (MAC) with a Physical Coding Sublayer (PCS) or standalone PCS.

Features

- Designed to the Ethernet requirements for 50 Gb/s operation as defined in Schedule 3 of the 25G Ethernet Consortium. See the note at the bottom of the page.
- Designed to the requirements for 40 Gb/s operation as defined in IEEE 802.3 Clause 82 *IEEE Standard for Ethernet* (IEEE Std 802.3-2015).
- Includes complete Ethernet MAC and PCS functions or standalone PCS.
- Simple packet-oriented user interface.
- Low latency mode.
- 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 tools compile scripts.
- Optional fee based AN/LT/KR Forward Error Correction (FEC) features.
- Optional KP4 FEC feature for GTM devices.

Note: To access the 25G specification, visit the 25G 50G Ethernet Consortium website.

IP Facts

LogiCORE™ IP Facts Table	
Subsystem Specifics	
Supported Device Family	Zynq® UltraScale+™ RFSoC Zynq® UltraScale+™ MPSoC Virtex® UltraScale+™ Kintex® UltraScale+™ Kintex® UltraScale™
Supported User Interfaces	128-bit Straddle Packet AXI4-Stream for 50 Gb/s 128-bit Straddle Packet or 256-bit AXI4-Stream for 40 Gb/s
Resources	Performance and Resource Use web page
Provided with Subsystem	
Design Files	Encrypted RTL
Example Design	Verilog
Test Bench	Verilog
Constraints File	Xilinx Design Constraints (XDC)
Simulation Model	Verilog
Supported S/W Driver	Not Applicable
Tested Design Flows[1]	
Design Entry	Vivado® Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Synopsys or Vivado synthesis
Support	
Release Notes and Known Issues	Master Answer Record: 54690
All Vivado IP Change Logs	Master Vivado IP Change Logs: 72775
Xilinx Support web page	

Notes:

1. For the supported versions of third-party tools, see the Xilinx Design Tools: Release Notes Guide.
2. Contact Xilinx Technical Support for your design requirements.

XILINX

Chapter 2

Overview

Subsystem Overview

The Xilinx® 40G/50G High Speed Ethernet Subsystem implements a 40G/50G Ethernet Media Access Controller (MAC) module with 40G/50G PCS or standalone 40G/50G PCS.

The 40G/50G High Speed Ethernet Subsystem is designed to Schedule 3 of the 25G and 50G Ethernet Consortium specification r1.6 for the 50 Gb/s operation and IEEE 802.3 for 40 Gb/s operation; it is hardware proven, and offers system designers with a risk-free and quick path for systems that implement 40G/50G Ethernet protocols.

This guide also describes the 40G/50G High Speed Ethernet Subsystem in detail and provides the information required to integrate the 40G/50G High Speed Ethernet Subsystem into user designs. The document assumes you are familiar with the IEEE Std 802.3-2015 protocol and FPGA design and methodology. See Chapter 3: Product Specification for detailed information. For Xilinx device platform-specific information, see Xilinx Support. See also the *IEEE Standard for Ethernet* (IEEE Std 802.3-2015)

Feature Summary

- Supports custom preambles
- Programmable Inter Packet Gap (IPG)
- Simple packet oriented 128-bit straddled packet AXI4-Stream Interface for 40 Gb/s and 50 Gb/s operation
- Optional 256-bit regular AXI4-Stream Interface for 40 Gb/s operation
- Optional Clause 74 Forward Error Correction (FEC)
- 50G FEC (Ethernet Consortium Schedule 3 specification, based on IEEE 802.3 Clause 91)
- Optional 1588v2 PTP 1-step and 2-step timestamping
- Optional Auto-Negotiation and Link Training

Table 1: **Feature Compatibility Matrix**

Variant	MAC	PCS	128-bit Straddle Packet [2]	256-bit	Pause Processing	Auto-Negotiation and Link Training	Clause 74 FEC	Clause 91 RS-FEC	IEEE 1588 Time Stamp
			AXI4-Stream						
Low Latency 40G MAC + PCS	X	X	X	X					
40G MAC + PCS	X	X	X	X	X	X	X		X
40G PCS-only		X				X	X		X[1]
Low Latency 50G MAC + PCS	X	X	X						
50G MAC + PCS	X	X	X		X	X	X	X	X
50G PCS-only		X				X	X	X	X[1]
Runtime switchable 40/50G MAC +PCS[3]	X	X	X	-	-	X	X	X	X
Runtime switchable 40/50G PCS-only[3]	-	X	-	-	-	X	X	X	X[1]

Notes:

1. Only 2-step time stamping is supported with PCS-only configurations.
2. UltraScale™/ UltraScale+™ speed grade -1 does not support 40G 128-bit AXI-Stream interface. It is recommended to use 256 bit AXI Stream interface for -1 speed grade.
3. Runtime switching, Auto Negotiation, and Link training is not supported for Transceiver type GTM and it will be supported in future release.

Applications

The Xilinx® 40G/50G High Speed Ethernet Subsystem is designed to function as the network interface for applications that require a very high bit rate, such as:

- Ethernet switches
- IP routers
- Data center switches
- Communications equipment

The capability to interconnect devices at 50 Gb/s Ethernet rates becomes especially relevant for next-generation data center networks where:

- To keep up with increasing CPU and storage bandwidth, rack or blade servers must support aggregate throughputs faster than 10 Gb/s (single lane) or 20 Gb/s (dual lane) from their Network Interface Card (NIC) or LAN-on-Motherboard (LOM) networking ports.

- Given the increased bandwidth to endpoints, uplinks from Top-of-Rack (TOR) or Blade switches need to transition from 40 Gb/s (four lanes) to 100 Gb/s (four lanes) while ideally maintaining the same per-lane breakout capability.
- Due to the expected adoption of 100GBASE-CR4/KR4/SR4/LR4, SerDes and cabling technologies are already being developed and deployed to support 25 Gb/s per physical lane, twin-ax cable, or fiber.

Licensing and Ordering

The 40G/50G Ethernet IP core is provided under the terms of the Xilinx Core Agreement. The module is shipped as part of the Vivado® Design Suite. For full access to all core functionalities in simulation and in hardware, you must purchase one or more licenses for the core.

***Note*:** The 40G/50G Ethernet MAC + BASE-R, XLAUI/LAUI, and 40GBASE-KR4/50GBASE-KR2 IP options require separate part numbers.

Contact your local Xilinx sales representative for more information on the 40G/50G Ethernet core pricing and availability. For more information, see the 40G/50G Ethernet Subsystem page.

Information about additional Xilinx® LogiCORE™ modules are available on the Xilinx Intellectual Property page.

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
40G/50G Ethernet MAC + BASE-R[1]	EF-DI-50GEMAC-PROJ EF-DI-50GEMAC-SITE	l_eth_mac_pcs
50GBASE-KP (RS-FEC (544, 514) AN/LT)[2] 50GBASE-KR2 (RS-FEC (528, 514), AN/LT) 40GBASE-KR4 (Clause 74 FEC, AN/LT) 50G RS-FEC (Standalone or adder to EF-DI-50GEMAC)XLAUI/LAUI (40GBASE-R/50GBASE-R) ***Note*:** If a MAC is also required, order EF-DI-50GEMAC-xxxx in addition to the EF-DI-50GBASE-KR2-xxxx part number.	EF-DI-50GBASE-KR2-PROJ EF-DI-50GBASE-KR2-SITE	l_eth_basekr ieee802d3_50g_rs_fec l_eth_baser

Table 2: **Ordering Information** *(cont'd)*

Description	Part Number	License Key
XLAUI/LAUI PCS/PMA (40GBASE-R/50GBASE-R) ***Note*:** AN/LT/FEC/MAC are not included in this part number. See above if these are required.	EF-DI-LAUI-PROJ EF-DI-LAUI-SITE	l_eth_baser

Notes:

1. For access to standalone PCS/PMA (40GBASE-R/50GBASE-R), you must order EF-DI-LAUI-XXXX or EF-DI-50GBASE-KR2-xxxx.
2. Used for 50GBASE-CR2, 50GBASE-KR2, 50GBASE-SR2, 50GBASE-FR2 applications. 50G RS-FEC (544, 514) is available as a hardened IP inside UltraScale+ 58G devices in the GTM transceiver for no additional charge.

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.*

Product Specification

The 40G/50G High Speed Ethernet IP subsystem provides ease of use connecting to the High Speed Ethernet core. Based on the configuration, this subsystem creates interface ports, instantiates the 40G/50G High Speed Ethernet Subsystem and high speed serialize and deserializer (SerDes) blocks, provides the appropriate clock signals, and connects to the AXI4-Stream user-side interface.

The block diagram for the 40G/50G High Speed Ethernet Subsystem is shown in the following figure. The right-hand side is the user interface and the left-hand side is the external device interface.

Figure 1: **40G/50G High Speed Ethernet Subsystem with 128-bit Straddled AXI4-Stream**

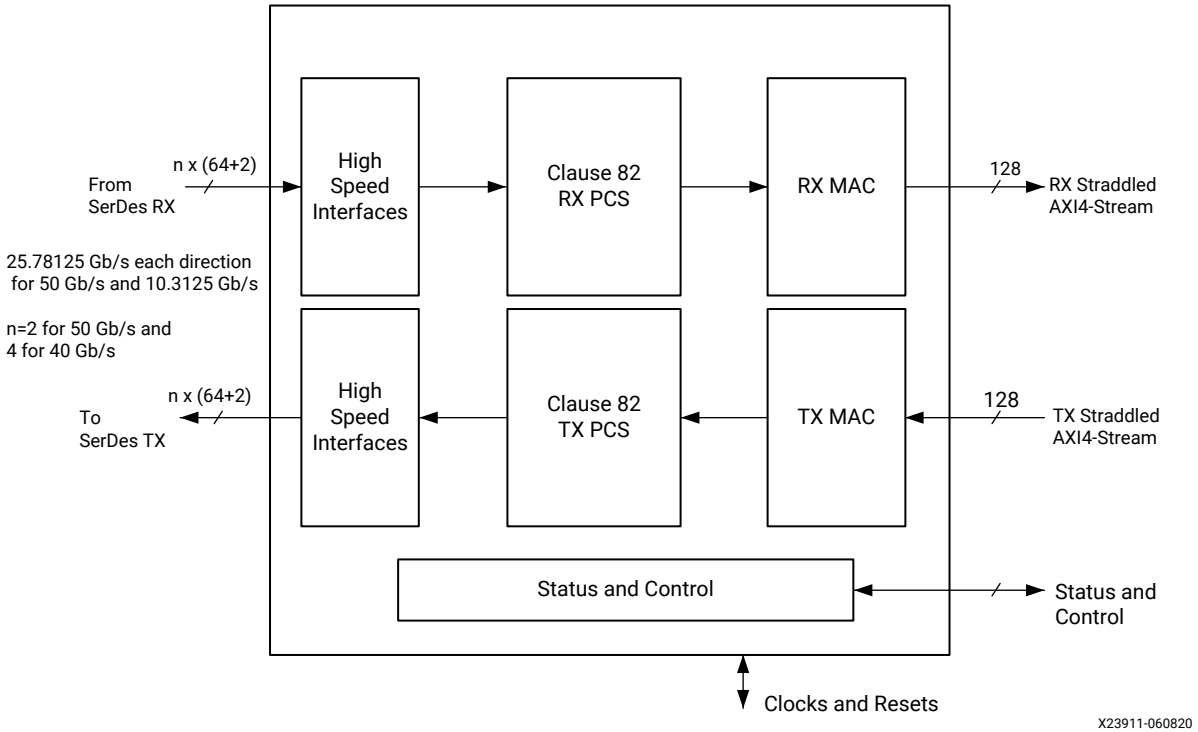
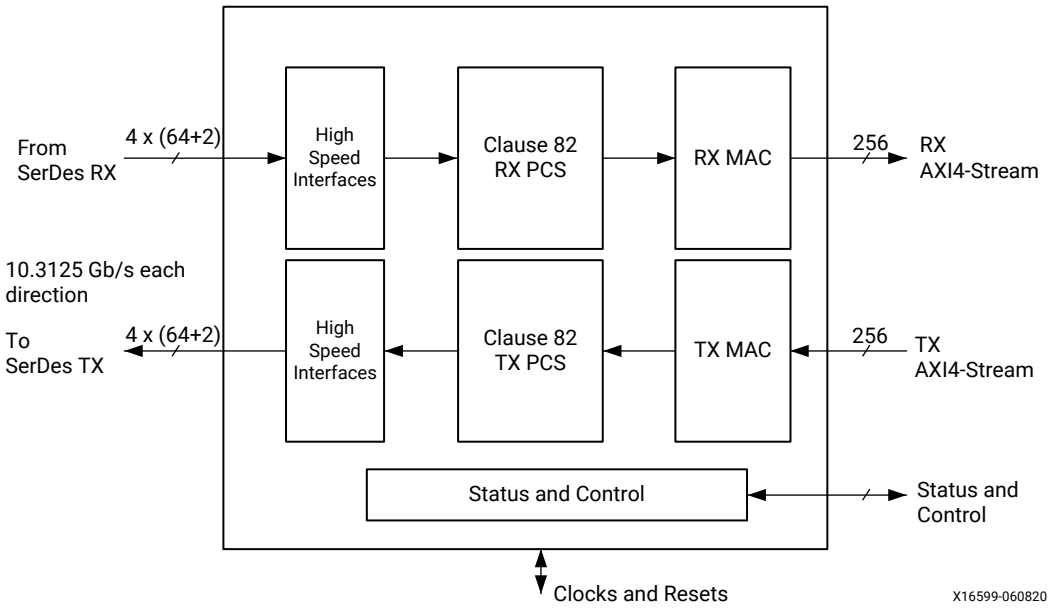


Figure 2: **40G High Speed Ethernet Subsystem with 256-bit AXI4-Stream**



The PCS architecture is based on distributing (or striping) parts of a packet over several (relatively) lower speed physical interfaces by the transmitting device. The receiving device PCS layer is then responsible for stripping the different parts and rebuilding the packet before handing it off to the Ethernet MAC block. The receiver PCS layer must also deskew the data from the different physical interfaces as these might encounter different delays as they are transported throughout the network. Additionally, the core handles PCS Lane swapping across all received PCS Lanes, allowing the 40G/50G High Speed Ethernet Subsystem to be used with all optical transport systems.

The PCS and Ethernet MAC layers of the core operate at the maximum line-rate of the interface, and have been optimized to operate in Xilinx® FPGAs. The PCS layer includes scrambling/descrambling and 64B/66B encoders/decoders operating at full 40G/50G line rate. The Ethernet MAC block includes a high-speed and optimized Frame Check Sequence (FCS) generation and checking module. In addition to checking the FCS integrity of the packet, the FCS module is capable of optionally inserting and deleting the FCS bytes of the packet at full 40G/50G line rate.

The Control and Status block provides several statistics counters for monitoring data traffic. Additionally, the status interface of the 40G/50G High Speed Ethernet Subsystem provides detailed information about the health of the overall interface, each individual physical interface, and every single PCS lane. The status information includes sync header alignment, PCS alignment and PCS deskew status.

Typical Operation

The core handles all protocol-related functions to communicate with the PCS and Ethernet MAC interface of another device. This includes handshaking, synchronizing and error checking. You provide packet data through the AXI4-Stream TX interface, and receive packet data from the AXI4-Stream RX interface. A detailed description is given in AXI4-Stream Interface in AXI4-Stream Interface.

The 40G/50G High Speed Ethernet Subsystem is designed to be as flexible as possible and can be used in many different applications. The RX path does not perform any buffering other than the pipelining required to perform the required operations. Received data is passed directly to the user interface in a cut-through manner, allowing you the flexibility to implement any required buffering scheme. Also, the core TX path consists of a single pipeline with minimal buffering to provide reliable cut-through operation.

Statistics Gathering

The 40G/50G High Speed Ethernet Subsystem provides a flexible and user-friendly mechanism for gathering statistics. For every one of the supported statistics, the core has an output signal (or bus) that indicates an increment value for the statistic in a given clock cycle. This mechanism allows you to select which statistics are required in the system without having the cost overhead of a full set of counters. Additionally, and more importantly, you can implement any counter and statistics gathering mechanism required by the system. For example, you can build 32-bit or 64-bit counters as needed, or implement clear-on-read or saturated counters, as required. An optional AXI4-Lite register implementation is available which includes statistics counters. A detailed description of the option AXI4-Lite implementation is given in Chapter 6: Example Design.

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

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

Testability Functions

The core implements the test pattern generation and checking as defined in Clause 82.2.10 (test-pattern generators) and 82.2.17 (test-pattern checkers). For details, see the *IEEE Standard for Ethernet* (IEEE Std 802.3-2015).

Standards

The 40G/50G High Speed Ethernet Subsystem is designed to Schedule 3 of the 25G and 50G Ethernet Consortium specification r1.6 for the 50 Gb/s operation and IEEE 802.3 for 40 Gb/s operation.

Performance and Resource Utilization

For full details about performance and resource use, visit the Performance and Resource Use web page.

Latency

The following table provides the measured latency information for the low latency design of the 40G/50G IP core. This is the combined RX and TX latency for the core and does not include latency in the transceiver.

Table 3: **Latency Results For Low Latency Design of the 40/50G IP Core**

Core	Total Latency (ns)	TX Latency (ns)	RX Latency (ns)	User bus width (bits)	SerDes data width (bits)	Core Clock Frequency (MHz)
40G MAC_PCS	99.2	35.2	64	128	32	312.5
50G MAC+PCS	84.5	25.6	58.8	128	64	390.625
40G PCS	196.5	97.5	99	128	32	312.5
50G PCS	161.8	79.2	82	128	64	390.625
40G MAC+PCS	153	64	89	256	32	312.5

Notes:

1. These numbers include both RX and TX fabric logic, but do not include the GT.
2. The MAC+PCS numbers are for low latency with FIFO removed. PCS/PMA always has FIFO, due to which the latency numbers are higher.

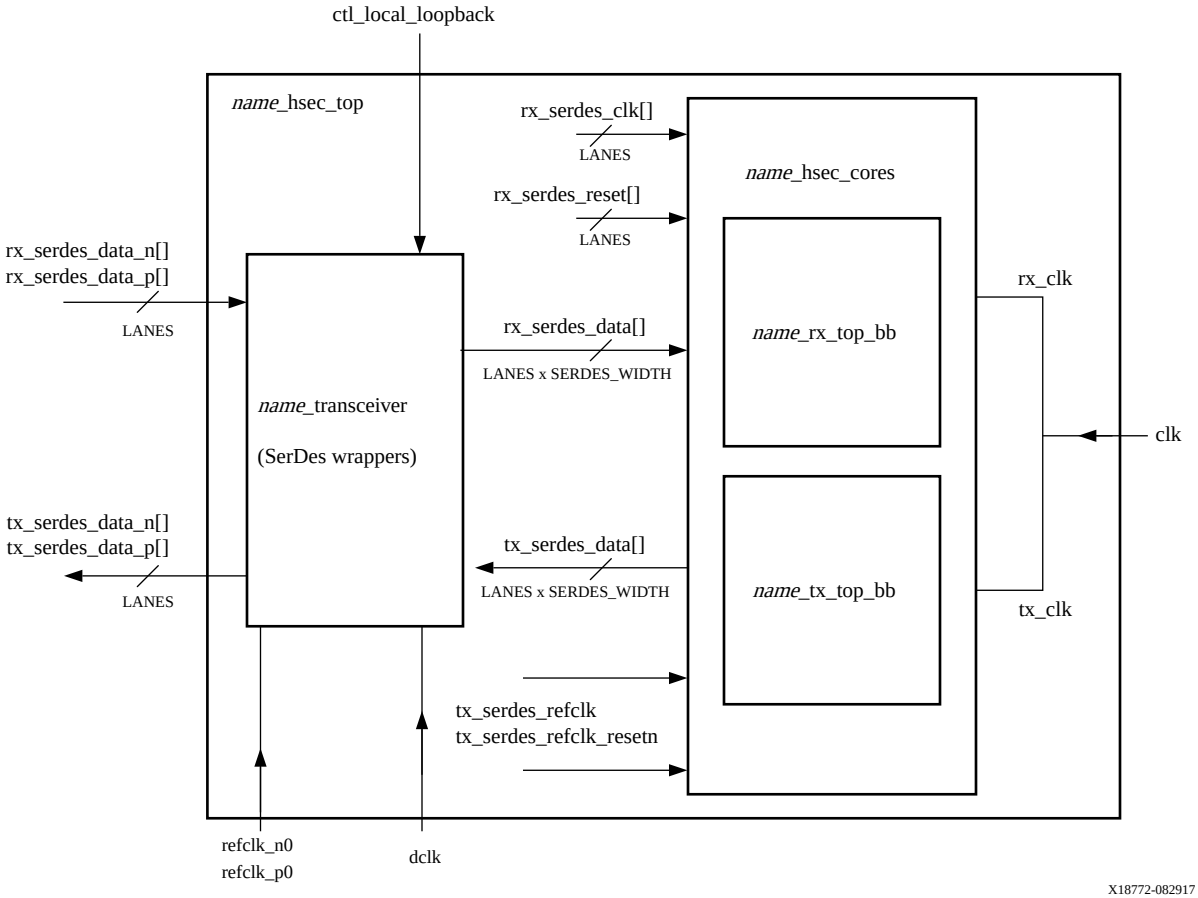
Port Descriptions

The following tables list all ports applicable to the 40G/50G subsystem including the ports used for optional features at the name_hsec_cores hierarchy level. At the xci top level hierarchy the ports change some to include the instantiated transceiver core and other shared logic. The port list at the XCI level of the hierarchy is described in Core xci Top Level Port List in Chapter 6: Example Design. Other Ports are described in PCS Variant and Auto-Negotiation (AN) and Link Training (LT).

The following figure shows the relationship between the hierarchical blocks and illustrates the differences among the blocks in relation to the signals.

Note: When you generate the optional AXI4-Lite registers, some of these ports can be accessed by the corresponding register instead of a broadside bus.

Figure 3: **Port List Hierarchy**



The following sections describe the ports. The VL_LANES parameter is 4 and the LANES parameter is 2 for 50 Gb/s operation and 4 for 40 Gb/s operation. For GTM, see SerDes Data Mapping for GTM.

Transceiver Ports

The following table describes the transceiver I/O ports. Refer to the Clocking topic for clock domain information.

Table 4: **Transceiver I/O Port List**

Name	I/O	Description
rx_serdes_data[LANES-1:0][64-1:0]	I	Data bus from the SerDes macros. There are LANES rx_serdes_data buses; one bus for each SerDes lane and each bus has 64-bits. By definition, bit [64-1] is the first bit received by the 40G/50G High Speed Ethernet Subsystem. Bit [0] is the last bit received. A typical width is 64.
tx_serdes_data[LANES-1:0][64-1:0]	O	Data bus to the SerDes macros. There are LANES tx_serdes_data buses; one bus for each SerDes lane and each bus has 64-bits. By definition, bit [64-1] is the first bit transmitted by the 40G/ 50G High Speed Ethernet Subsystem. Bit [0] is the last bit transmitted. A typical width is 64.
rx_serdes_clk[LANES-1:0]	I	Recovered clock of each SerDes lane. The rx_serdes_data bus for each lane is synchronized to the positive edge of the corresponding bit of this bus.
rx_serdes_reset[LANES-1:0]	I	Reset for each RX SerDes lane. The recovered clock for each SerDes lane has associated with it an active-High reset. This signal should be asserted whenever the associated recovered clock is not operating at the correct frequency. Generally this signal is connected to a phase-locked loop (PLL) lock signal. This is a synchronous reset.
tx_serdes_refclk	I	Reference clock for the TX datapath. This clock must be frequency locked to the tx_serdes_clk inputs. Typically, the same reference clock that is used to drive the TX SerDes is connected to this input.
tx_serdes_refclk_reset	I	Reset for TX Reference clock. This signal should be asserted whenever the tx_serdes_refclk input is not operating at the correct frequency. This is a synchronous reset.

Related Information

Clocking

AXI4-Stream Interface

This section describes how to connect the AXI4-Stream data interfaces of the 40G/50G subsystem. The AXI4-Stream interface follows the standard Arm® AMBA 4 AXI4-Stream Protocol v1.0 Specification but makes extensive use of "user" signals. Refer to the *AMBA AXI4-Stream Protocol Specification* (ARM IHI 0051A) for a detailed description of the interface.

The 40G/50G High Speed Ethernet subsystem provides a 128-bit straddle packet AXI4-Stream interface for operations at 40 Gb/s and 50 Gb/s. Additionally, there is a choice of a 256-bit regular AXI4-Stream interface for the IP operating at 40 Gb/s only.

AXI4-Stream Interface – 128-bit Straddle Packet Interface

In the AXI4-Stream interface, the following tables describe the clock/reset signals, the Receive interface signals, and the Transmit interface signals.

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

Name	I/O	Clock Domain	Description
clk	I		AXI4-Stream clock. All signals between the 40G/50G High Speed Ethernet Subsystem and the user-side logic are synchronized to the positive edge of this signal.
dclk	I		This must be a convenient stable clock, for example 75 MHz. Refer to the current transceiver guide for up to date information.
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 40G/50G High Speed Ethernet Subsystem handles synchronizing the rx_reset input to the appropriate clock domains within the core. This is a synchronous reset.
refclk_n0	I		Differential reference clock for the transceiver (N).
refclk_p0	I		Differential reference clock for the transceiver (P).
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 40G/50G High Speed Ethernet Subsystem handles synchronizing the tx_reset input to the appropriate clock domains within the core. This is a synchronous reset.

Receive AXI4-Stream Interface

The receive AXI4-Stream interface is similar to the transmit side, with the RX data corresponding to the received Ethernet frame. The other signals on the RX AXI bus have a meaning analogous to the signals on the TX bus.

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

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

Signal	I/O	Clock Domain	Description
rx_clk_out	Out		All RX AXI signals are referenced to this clock.
rx_axis_tdata[127:0]	Out	rx_clk_out	AXI4-Stream Data to user logic.
rx_axis_tuser_tvalid	Out	rx_clk_out	AXI4-Stream Data Valid. When this signal is 1, there is valid data on the RX AXI data bus.
rx_axis_tuser_sop0 rx_axis_tuser_sop1	Out	rx_clk_out	This signal, when asserted, indicates the start of a received Ethernet frame.
rx_axis_tuser_eop0 rx_axis_tuser_eop1	Out	rx_clk_out	This signal, when asserted, indicates the end of a received Ethernet frame. There are two bits—one for each segment.
rx_axis_tuser_err0 rx_axis_tuser_err1	Out	rx_clk_out	RX AXI error indication signal. • 1 indicates a bad packet has been received. • 0 indicates a good packet has been received. There are two bits—one for each segment.

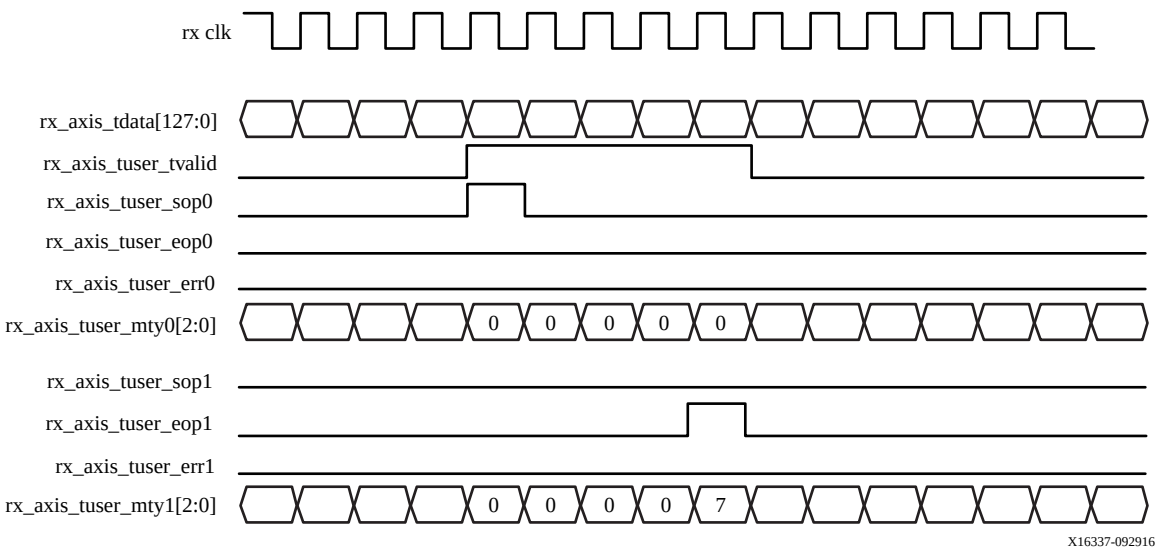
Table 6: **AXI4-Stream Receive Interface Signals** *(cont'd)*

Signal	I/O	Clock Domain	Description
rx_axis_tuser_mty0[2:0] rx_axis_tuser_mty1[2:0]	Out	rx_clk_out	This bus indicates how many bytes of the rx_axis_tdata bus are empty or invalid for the last transfer of the current packet. This bus is only valid during cycles when both rx_axis_tuser_ena and rx_axis_tuser_eop are 1. There are two bits—one for each segment.
rx_axis_tuser_ena0 rx_axis_tuser_ena1	Out	rx_clk_out	Receive AXI4-Steam Enable for each Segment. When asserted, this signal indicates that data for the associated segment is valid.

Normal Frame Reception

The timing of a normal inbound frame transfer is represented below. The client must be prepared to accept data at any time as there is insufficient 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_tuser_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_tuser_mty` bits being set to 0, except during the final transfer of the frame when `rx_axis_tuser_eop` is asserted. During this final transfer of data for a frame, `rx_axis_tuser_mty` bits indicate the final valid bytes of the frame.

Figure 4: **RX Waveform**

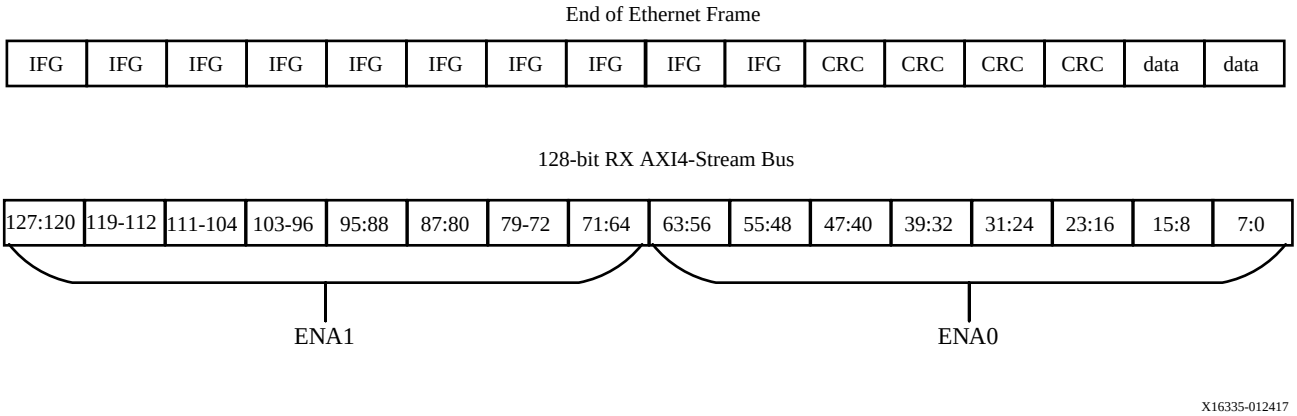


The signals shown in the figure above waveform are described in the following subsections.

- **rx clk:** The `rx_clk_out` signal should be used as the clock reference for all RX AXI4-Stream signals. All RX AXI signals are aligned to the rising edge of this clock.

- **rx_axis_tdata[127:0]:** This bus provides the packet-oriented data corresponding to the received Ethernet frame. Data is clocked by the `rx_clk_out` signal. The following figure illustrates how the end of an Ethernet frame is mapped onto the bit positions of the RX AXI4-Stream interface. Note the positions of the ENA0 and ENA1 signals relative to the bits positions of the RX AXI4-Stream bus. This mapping is for a 128-bit AXI4-Stream bus.

Figure 5: **RX Mapping**



- **rx_axis_tuser_tvalid:** When asserted, this signal indicates that data on the RX AXI bus is valid.
- **rx_axis_tuser_ena0, rx_axi_tuser_ena1:** When asserted, this signal indicates that data for the associated segment is valid.
- **rx_axis_tuser_sop0, rx_axis_tuser_sop1:** This signal indicates the start of an Ethernet frame on the RX AXI bus. There is one SOP signal for each segment.
- **rx_axis_tuser_eop0, rx_axis_tuser_eop1:** This signal indicates the end of an Ethernet frame on the RX AXI bus. There is one EOP signal for each segment.
- **rx_axis_tuser_err0, rx_axis_tuser_err1:** When this signal is asserted, it indicates that there is an error in the received frame. It is valid during the EOP cycle when `ena` and `tvalid` are asserted. There is one error signal for each segment.

  The types of errors that can exist include:

  - There was an FCS error
  - The length was out of the valid range
  - A bad 64B/66B code was received during receipt of the packet
- **rx_axis_tuser_mty0[2:0], rx_axis_tuser_mty1[2:0]:** As in the TX AXI interface, the `mty` signal indicates how many bytes of the current cycle are unused (empty) during the last cycle of a received packet (the EOP cycle). There is one `mty` signal for each RX segment.

Transmit AXI4-Stream Interface

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

Table 7: **AXI4-Stream Transmit Interface Signals**

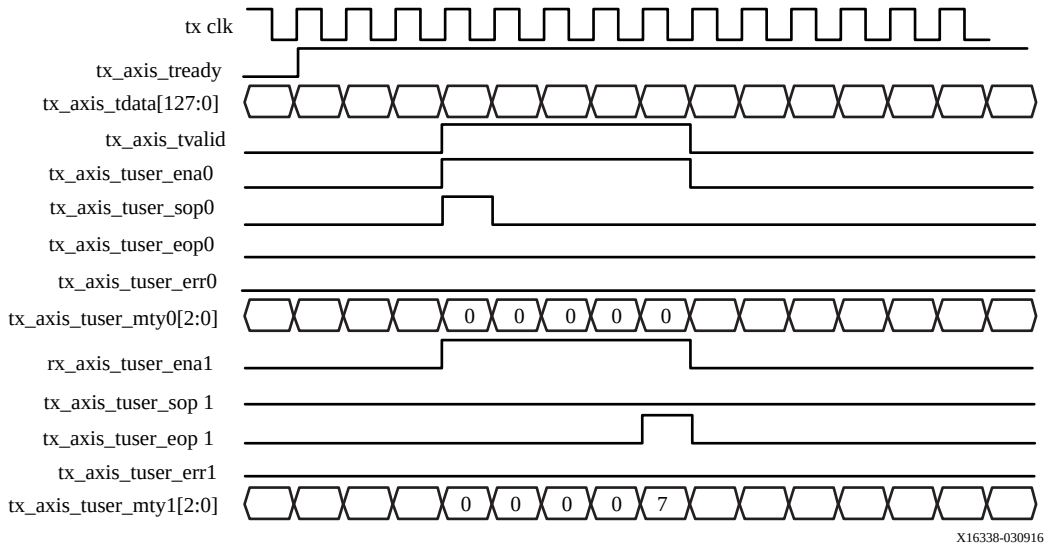
Signal	I/O	Clock Domain	Description
tx_clk_out	O		Transmit AXI clock. All TX signals are referenced to this clock.
tx_axis_tready	O	tx_clk_out	When High, this signal indicates that the TX AXI interface is ready to accept data. You must respond immediately when tx_axis_tready goes Low by stopping data transfers.
tx_axis_tdata[127:0]	I	tx_clk_out	Transmit AXI4-Stream data (128-bit interface). The TX AXI data bus receives user-supplied packet data.
tx_axis_tvalid	I	tx_clk_out	AXI4-Stream Data Valid input. Data transfers are only completed when this signal is 1.
tx_axis_tuser_ena0 tx_axis_tuser_ena1	I	tx_clk_out	Enable signal for the TX AXI bus transfers. A High on this signal enables transfer of data to the TX.
tx_axis_tuser_sop0 tx_axis_tuser_sop1	I	tx_clk_out	AXI4-Stream signal indicating the Start of Ethernet Packet. There is one Start of Packet (SOP) signal per segment.
tx_axis_tuser_eop0 tx_axis_tuser_eop1	I	tx_clk_out	AXI4-Stream signal indicating End of Ethernet Packet. There is one End of Packet (EOP) signal per segment.
tx_axis_tuser_err0 tx_axis_tuser_err0	I	tx_clk_out	This signal is used to indicate that a packet contains an error when it is sampled as a 1 and is 0 for all other transfers of the packet. This signal is sampled only in cycles when tx_enain and tx_eopin are sampled as 1. When this signal is sampled as a 1, the last data word is replaced with the 802.3 Error Code control word that guarantees the partner device receives the packet in error. If a packet is input with this signal set to a 1, the FCS checking and reporting is disabled (only for that packet). There is one signal per segment.
tx_axis_tuser_mty0[2:0] tx_axis_tuser_mty1[2:0]	I	tx_clk_out	Transmit Empty. This bus is used to indicate how many bytes of the tx_datain bus are empty or invalid for the last transfer of the current packet. This bus is sampled only in cycles when tx_axis_valid and tx_axis_user_eopin are sampled as 1.

The synchronous TX AXI bus interface accepts packet-oriented data. All signals are synchronous relative to the rising edge of the `tx_clk_out` port.

The AXI4-Stream transmit interface consists of two segments, where each is 64-bits (8 bytes) wide. This segmented approach or straddled AXI4-Stream approach as it is referred to allows for greater efficiency such that a packet can start and end in any given segment or cycle.

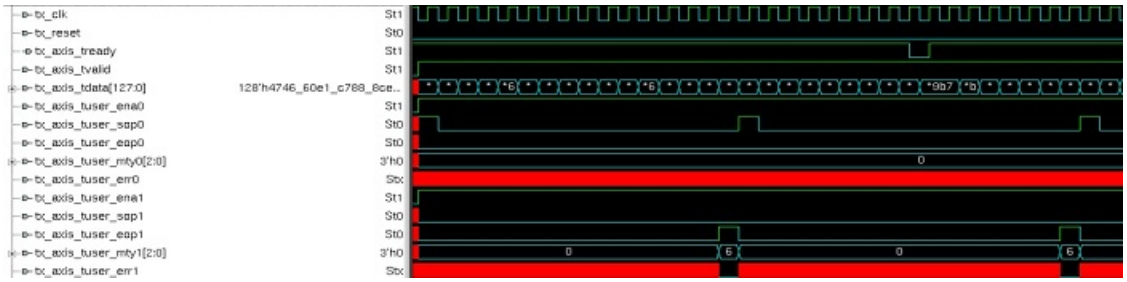
A normal transmit cycle is shown in the following waveform. These waveforms illustrate the transfer of a 73-byte packet over a 128-bit wide AXI4-Stream interface. There are two segments shown, numbered 0 and 1.

Figure 6: **TX Waveform**



The following figure shows a normal transmit cycle with back-to-back continuous transfers.

Figure 7: **TX Waveform with Back-to-Back Continuous Transfers**



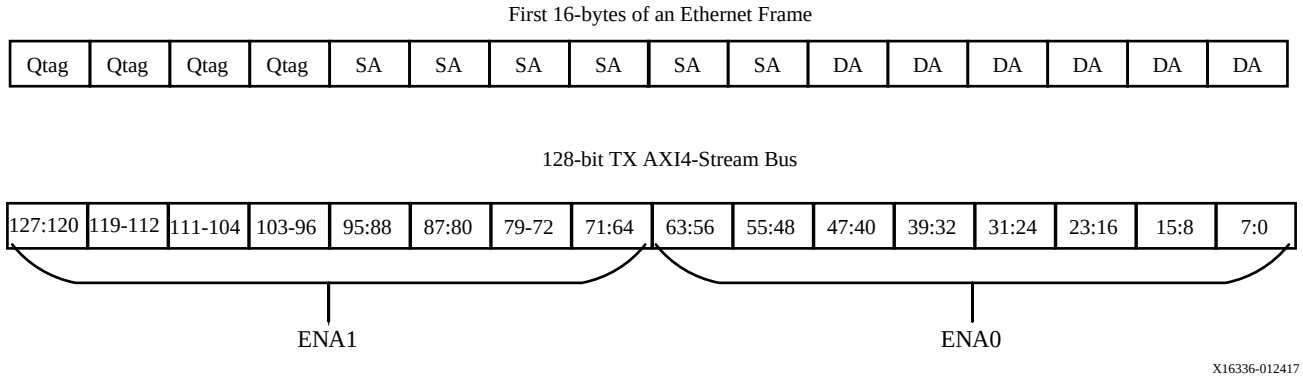
The signals shown in the Figure 6: TX Waveform and Figure 7: TX Waveform with Back-to-Back Continuous Transfers waveforms are described in the following subsections.

- **tx_clk:** This is the signal `tx_clk_out` output of the subsystem. All TX AXI signals are referenced to this clock. The frequency of this clock is normally 390.625 MHz for 50G operation and 312.5 MHz for 40G operation.
- **tx_axis_tready:** When asserted, this signal indicates that the TX AXI4-Stream interface is able to accept data. When `tx_axi_tready` goes Low, you must stop sending data immediately, or it will not be accepted by the TX AXI4-Stream interface.
- **tx_axis_tdata[127:0]:** This is the bus for the frame to be transmitted.

The following figure illustrates how the start of an Ethernet frame is mapped onto the bit positions of the TX AXI4-Stream interface. Note the positions of the ENA0 and ENA1 signals relative to the bits positions of the TX AXI4-Stream bus.

This mapping is for a 128-bit AXI4-Stream bus.

Figure 8: **TX Mapping**



- **tx_axis_tvalid:** When High, this signal indicates that there is valid data on the TX AXI bus.
- **tx_axis_tuser_ena0, tx_axis_tuser_ena1:** These signals enable the transfer of data over the TX bus when asserted. Data transfer has to be validated with the `tvalid` signal in order for a transfer to take place.

There is an enable signal for each AXI segment.

- **tx_axis_tuser_sop0, tx_axis_tuser_sop1:** These signals indicate the start of an Ethernet frame in that cycle. Only one SOP is permitted in a bus cycle. There is a separate SOP signal for each AXI segment.
- **tx_axis_tuser_eop0, tx_axis_tuser_eop1:** These signals indicate the end of an Ethernet frame in that cycle. Only one EOP is permitted in a bus cycle. There is a separate EOP signal for each AXI segment.
- **tx_axis_tuser_mty0[2:0], tx_axis_tuser_mty1[2:0]:** These signals indicate which bytes of the corresponding segment are not used ("empty"). If `tx_mtyin` has a value of 0x0, there are no empty byte lanes, or in other words, all bits of the data bus are valid.For example, if `tx_axis_tuser_mty0[2:0]` = 2, the last 2 bytes of the segment do not contain data and will be ignored. The value of `mty` can only be non-zero during the last cycle of a packet transfer (the EOP cycle).
- **tx_unfout:** Not shown on the waveforms in the previous TX Waveform timing diagram is the `tx_unfout` output indicator. When this signal is High, it indicates that there has not been a sufficient data transfer and the Ethernet interface will underflow. This must not be allowed to occur. You must ensure that you transfer data whenever `tx_axis_tready` is High until you reach the end of the Ethernet frame.

  Note: When this signal sampled as 1, you need to apply `tx_reset/sys_reset` to recover the core from the underflow issue. `tx_reset` resets the TX path only and `sys_reset` recovers the complete system.

- **tx_axis_user_err0, tx_axis_user_err1:** This signal is used to indicate that a packet contains an error when it is sampled as a 1 and is 0 for all other transfers of the packet. This signal is sampled only in cycles when `tx_axis_tuser_ena` and `tx_axis_tuser_eop` are sampled as 1. When this signal is sampled as a 1, the last data word is replaced with the 802.3 Error Code control word that guarantees the partner device receives the packet in error. If a packet is input with this signal set to a 1, the FCS checking and reporting is disabled (only for that packet).There is one `tx_axis_user_err` signal per segment.

AXI4-Stream Interface – 256-bit Packet Interface

The 40G/50G IP Subsystem provides an option of a 256-bit AXI4-Stream packet interface for operation at 40 Gb/s.

AXI4-Stream Clocks and Reset

Table 8: **AXI4-Stream Interface – Clock/Reset Signals**

Name	I/O	Description
clk	I	AXI4-Stream clock. All signals between the 40G/50G High Speed Ethernet Subsystem and the user-side logic are synchronized to the positive edge of this signal.
dclk	I	This must be a convenient stable clock, for example 75 MHz. Refer to the current transceiver guide for up to date information.
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 40G/50G High Speed Ethernet Subsystem handles synchronizing the rx_reset input to the appropriate clock domains within the core. This is a synchronous reset.
refclk_n0	I	Differential reference clock for the transceiver (N).
refclk_p0	I	Differential reference clock for the transceiver (P).
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 40G/50G High Speed Ethernet Subsystem handles synchronizing the tx_reset input to the appropriate clock domains within the core. This is a synchronous reset.

Transmit AXI4-Stream Interface - 256-bit

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

Table 9: **Transmit AXI4-Stream Interface**

Name	I/O	Clock Domain	Description
tx_axis_tready	O	tx_clk_out	AXI4-Stream acknowledge signal to indicate to start the Data transfer
tx_axis_tvalid	I	tx_clk_out	AXI4-Stream Data Valid Input
tx_axis_tdata[255:0]	I	tx_clk_out	AXI4-Stream Data
tx_axis_tuser	I	tx_clk_out	AXI4-Stream User Sideband interface. 1 indicates a bad packet. 0 indicates a good packet.
tx_axis_tlast	I	tx_clk_out	AXI4-Stream signal indicating End of Ethernet Packet
tx_axis_tkeep[31:0]	I	tx_clk_out	AXI4-Stream Data Control

Data Lane Mapping - TX

For transmit data, `tx_axis_tdata[255:0]`, the port is logically divided into lane 0 to lane 31. See the following table.

Table 10: **tx_axis_tdata Lanes**

Lane/tx_axis_tkeep	tx_axis_tdata[255:0] bits
0	7:0
1	15:8
2	23:16
3	31:24
4	39:32
5	47:40
6	55:48
7	63:56
8	71:64
9	79:72
10	87:80
11	95:88
12	103:96
13	111:110
14	119:112
15	127:120
16	135:128
17	143:136
18	151:144
19	159:152
20	167:160
21	175:168
22	183:176
23	191:184
24	199:192
25	207:200
26	215:208
27	223:216
28	231:224
29	239:232
30	247:240
31	255:248

Normal Transmission

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

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

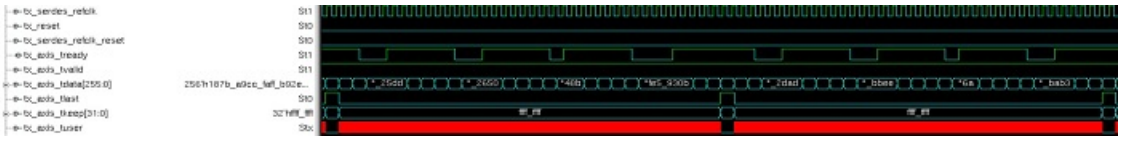
Figure 9: **Normal Frame transmission - 256 bit AXI4-Stream**

Back to Back Continuous Transfer

Continuous data transfer on the transmit AXI4-Stream interface is possible, as the signal tx_axis_tvalid can remain continuously High, with packet boundaries defined solely by tx_axis_tlast asserted for the end of the Ethernet packet. However, the core can deassert the tx_axis_tready acknowledgment signal to throttle the client data rate as required. See the following figure.

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, when valid is asserted and new data has been placed on the AXI4-Stream, it should remain there until the core has asserted tx_axis_tready.

Figure 10: **Back to Back Continuous Transfer – 256-bit AXI4-Stream**



Aborting a Transmission

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

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

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

Receive AXI4-Stream Interface – 256 bit

Table 11: **Receive AXI4-Stream Interface**

Name	I/O	Clock Domain	Description
rx_axis_tdata[255:0]	O	tx_clk_out	AXI4-Stream Data to user logic
rx_axis_tvalid	O	tx_clk_out	AXI4-Stream Data Valid. When this signal is 1, there is valid data on the RX AXI bus
rx_axis_tuser	O	tx_clk_out	AXI4-Stream User Sideband interface. • 1 indicates a bad packet has been received. • 0 indicates a good packet has been received
rx_axis_tlast	O	rx_clk_out	AXI4-Stream signal indicating an end of packet
rx_axis_tkeep[31:0]	O	rx_clk_out	AXI4-Stream Data Control to upper layer.

Data Lane Mapping - RX

For receive data, `rx_axis_tdata[63:0]`, the port is logically divided into lane 0 to lane 7. See the following table.

Table 12: **Data Lane Mapping**

Lane/rx_axis_tkeep	rx_axis_tdata[255:0] bits
0	7:0
1	15:8
2	23:16
3	31:24
4	39:32
5	47:40
6	55:48
7	63:56
8	71:64
9	79:72
10	87:80
11	95:88
12	103:96
13	111:110
14	119:112

Table 12: **Data Lane Mapping** *(cont'd)*

Lane/rx_axis_tkeep	rx_axis_tdata[255:0] bits
15	127:120
16	135:128
17	143:136
18	151:144
19	159:152
20	167:160
21	175:168
22	183:176
23	191:184
24	199:192
25	207:200
26	215:208
27	223:216
28	231:224
29	239:232
30	247:240
31	255:248

Normal Frame Reception

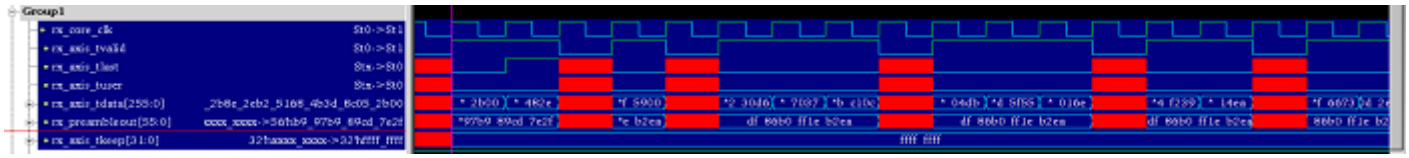
The timing of a normal inbound frame transfer is represented in following figure. The client must be prepared to accept data at any time; there is no buffering within the core to allow for latency in the receive client.

During frame reception, `rx_axis_tvalid` is asserted to indicate that valid frame data is being transferred to the client on `rx_axis_tdata`. All bytes are always valid throughout the frame, as indicated by all `rx_axis_tkeep` bits being set to 1, except during the final transfer of the frame when rx_axis_tlast is asserted. During this final transfer of data for a frame, `rx_axis_tkeep` bits indicate the final valid bytes of the frame using the mapping from above.

The valid bytes of the final transfer always lead out from `rx_axis_tdata[7:0]` (`rx_axis_tkeep[0]`) because Ethernet frame data is continuous and is received least significant byte first.

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

Figure 11: **Normal Frame Reception - 256 bits Non-segmented AXI4-Stream**



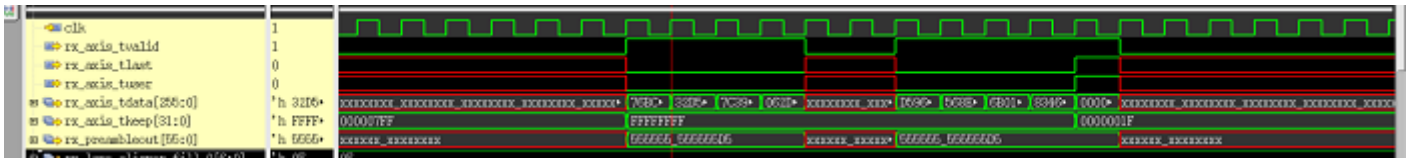
Frame Reception with Errors

The case of an unsuccessful frame reception (for example, a runt frame or a frame with an incorrect FCS) is shown in the following figure. 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 XLGMII data stream contains error codes.

Figure 12: **Frame Reception with Errors - 256-bit Non-Segmented AXI4-Stream**



TX Path Control/Status Ports

The following table describes the other status/control ports.

Table 13: **TX Path Control/Status Ports**

Name	I/O	Clock Domain	Description
ctl_rate_mode	I	static	This signal causes the IP core to switch between 50G operation (0) and 40G operation (1). Note that the clock frequencies must be corrected for the mode chosen.

Table 13: **TX Path Control/Status Ports** *(cont'd)*

Name	I/O	Clock Domain	Description
ctl_tx_enable	I	clk	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 40G/50G High Speed Ethernet Subsystem. This input should not be set to 1 until the receiver it is sending data to (that is, the receiver in the other device) is fully aligned and ready to receive data (that is, the other device is not sending a remote fault condition). Otherwise, loss of data can occur. If this signal is set to 0 while a packet is being transmitted, the current packet transmission is completed and the 40G/50G High Speed Ethernet Subsystem stops transmitting anymore packets.
ctl_tx_send_rfi	I	clk	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	I	clk	Transmit Local Fault Indication (LFI) code word. Takes precedence over RFI.
ctl_tx_send_idle	I	clk	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.
ctl_tx_fcs_ins_enable	I	clk	Enable FCS insertion by the TX core. If this bit is set to 0, the 40G/50G High Speed Ethernet Subsystem does not add FCS to the packet. It this bit is set to 1, the 40G/50G High Speed Ethernet Subsystem calculates and adds the FCS to the packet. This input cannot be changed dynamically between packets.
ctl_tx_ignore_fcs	I	clk	Enable FCS error checking at the AXI4-Stream interface by the TX core. This input only has effect when ctl_tx_fcs_ins_enable is Low. If 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.
ctl_tx_vl_length_minus1[15:0]	I	static	Number of words in between PCS Lane markers minus one. Default value, as defined in IEEE Std 802.3-2015, should be set to 16,383. This input should only be changed while the corresponding reset input is asserted. ***Note*:** When RS-FEC is enabled in the 50G core configuration, this value will be set to 20479.
ctl_tx_vl_marker_id[VL_LANES-1:0][63:0]	I	static	These inputs set the PCS Lane markers for each PCS lane. For 802.3 default values, see the IEEE Std 802.3-2015 [*IEEE Standard for Ethernet* (IEEE Std 802.3-2015)]. This input should only be changed while the corresponding reset input is asserted.
stat_tx_local_fault	O	clk	A value of 1 indicates the transmit encoder state machine is in the TX_INIT state. This output is level sensitive.

Table 13: **TX Path Control/Status Ports** *(cont'd)*

Name	I/O	Clock Domain	Description
ctl_tx_custom_preamble_enable	I	tx_clk	When asserted, this signal treats the first 64 bits of a packet on the rx_serdes_clk as a custom preamble instead of inserting a standard preamble. When asserted, this signals enables the use of tx_preamblein as a custom preamble instead of inserting a standard preamble.
tx_preamblein[55:0]	I	tx_clk	This bus represents the custom preamble when the signal ctl_tx_custom_preamble_enable is asserted. It should be asserted on the first cycle of the packet (start of packet).
stat_tx_underflow_err[1]	O	tx_clk	TX FIFO Underflow
stat_tx_overflow_err[1]	O	tx_clk	TX FIFO Overflow

Notes:

1. These signals are available only in the 256-bit non-segmented AXI4-Stream Variant.

RX Path Control/Status Ports

The following table describes the other status/control ports.

Table 14: **RX Path Control/Status Signals**

Name	I/O	Clock Domain	Description
ctl_rate_mode	I	static	This signal causes the IP core to switch between 50G operation (0) and 40G operation (1). Note that the clock frequencies will need to be correct for the mode chose.
ctl_rx_enable	I	rx_serdes_clk	RX Enable. For normal operation, this input must be set to 1. When this input is set to 0, after the RX completes the reception of the current packet (if any), it stops receiving packets by keeping the PCS from decoding incoming data. In this mode, there are no statistics reported and the AXI4-Stream interface is idle.
ctl_rx_check_preamble	I	rx_serdes_clk	When asserted, this input causes the Ethernet MAC to check the preamble of the received frame.
ctl_rx_check_sfd	I	rx_serdes_clk	When asserted, this input causes the Ethernet MAC to check the start of frame Delimiter of the received frame.
ctl_rx_force_resync	I	rx_serdes_clk	RX force resynchronization input. This signal is used to force the RX path to reset, re-synchronize, and realign. A value of 1 forces the reset operation. A value of 0 allows normal operation. ***Note*:** This input should normally be Low and should only be pulsed (1 cycle minimum pulse) to force realignment. CTL_RX_FORCE_RESYNC restarts the synchronization state machine but does not reset the GT logic. In most cases when there is an RX failure, the GT RX needs to be reset.

Table 14: **RX Path Control/Status Signals** *(cont'd)*

Name	I/O	Clock Domain	Description
ctl_rx_delete_fcs	I	rx_serdes_clk	Enable FCS removal by the RX core. If this bit is set to 0, the 40G/50G High Speed Ethernet Subsystem does not remove the FCS of the incoming packet. If this bit is set to 1, the 40G/50G High Speed Ethernet Subsystem deletes the FCS to the received packet. FCS is not deleted for packets that are £ 8 bytes long. This input should only be changed while the corresponding reset input is asserted.
ctl_rx_ignore_fcs	I	rx_serdes_clk	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_errout pin asserted during the last transfer (rx_eopout and rx_enaout sampled 1). If this bit is set to 1, the 40G/50G High Speed Ethernet Subsystem does not flag an FCS error at the AXI4-Stream interface. ***Note*:** The statistics are reported as if the packet is good. The `stat_rx_bad_fcs` signal, however, reports the error.
ctl_rx_max_packet_len[14:0]	I	rx_serdes_clk	Any packet longer than this value is considered to be oversized. If a packet has a size greater than this value, the packet is truncated to this value and the rx_errout signal is asserted along with the rx_eopout signal. Packets less than 16 bytes are dropped. The allowed value for this bus can range from 64 to 16,383. ctl_rx_max_packet_len[14] is reserved and must be set to 0.
ctl_rx_min_packet_len[7:0]	I	rx_serdes_clk	Any packet shorter than this value is considered to be undersized. If a packet has a size less than this value, the rx_errout signal is asserted during the rx_eopout asserted cycle. Packets that are less than 16 bytes are dropped. ***Note*:** This value should be greater than or equal to 64B.
ctl_rx_vl_length_minus1[15:0]	I	rx_serdes_clk	Number of words in between PCS Lane markers minus one. Default value, as defined in IEEE Std 802.3-2015, should be set to 16,383. This input should only be changed while the corresponding reset input is asserted. ***Note*:** When RS-FEC is enabled in the 50G core configuration, this value will be set to 20479.
ctl_rx_vl_marker_id[VL_LANES-1:0][63:0]	I	rx_serdes_clk	These inputs set the PCS Lane markers for each PCS lane. These inputs should be set to the values as defined in the IEEE Std 802.3-2015. For IEEE 802.3 default values, see Section 5.3 [*IEEE Standard for Ethernet* (IEEE Std 802.3-2015)]. This input should only be changed while the corresponding reset input is asserted.

Table 14: **RX Path Control/Status Signals** *(cont'd)*

Name	I/O	Clock Domain	Description
stat_rx_framing_err_[VL_LANES-1:0][3:0]	O	rx_clk_out	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_[VL_LANES-1:0] is a 1. The values on these buses can be updated at any time and are intended to be used as increment values for sync header error counters.
stat_rx_framing_err_valid_[VL_LANES-1:0]	O	rx_clk_out	Valid indicator for stat_rx_framing_err_[VL_LANES-1:0]. When one of these outputs is sampled as a 1, the value on the corresponding stat_rx_framing_err_[VL_LANES-1:0] is valid.
stat_rx_local_fault	O	rx_clk_out	This output is High when stat_rx_internal_local_fault or stat_rx_received_local_fault is asserted. This output is level sensitive.
stat_rx_synced[VL_LANES-1:0]	O	rx_clk_out	Word Boundary Synchronized. These signals indicate whether a PCS lane is word boundary synchronized. A value of 1 indicates the corresponding PCS lane has achieved word boundary synchronization and it has received a PCS lane marker. Corresponds to management data input/ output (MDIO) register bit 3.52.7:0 and 3.53.11:0 as defined in Clause 82.3. This output is level sensitive.
stat_rx_synced_err[VL_LANES-1:0]	O	rx_clk_out	Word Boundary Synchronization Error. These signals indicate whether an error occurred during word boundary synchronization in the respective PCS lane. A value of 1 indicates that the corresponding PCS lane lost word boundary synchronization due to sync header framing bits errors or that a PCS lane marker was never received. This output is level sensitive.
stat_rx_mf_len_err[VL_LANES-1:0]	O	rx_clk_out	PCS Lane Marker Length Error. These signals indicate whether a PCS Lane Marker length mismatch occurred in the respective lane (that is, PCS Lane Markers were received not every ctl_rx_vl_length_minus1 words apart). A value of 1 indicates that the corresponding lane is receiving PCS Lane Markers at wrong intervals. This remains High until the error condition is removed.
stat_rx_mf_repeat_err[VL_LANES-1:0]	O	rx_clk_out	PCS Lane Marker Consecutive Error. These signals indicate whether four consecutive PCS Lane Marker errors occurred in the respective lane. A value of 1 indicates an error in the corresponding lane. This output remains High until the error condition is removed.
stat_rx_mf_err[VL_LANES-1:0]	O	rx_clk_out	PCS Lane Marker Word Error. These signals indicate that an incorrectly formed PCS Lane Marker Word was detected in the respective lane. A value of 1 indicates an error occurred. This output is pulsed for one clock cycle to indicate the error condition. Pulses can occur in back-to-back cycles.

Table 14: **RX Path Control/Status Signals** *(cont'd)*

Name	I/O	Clock Domain	Description
stat_rx_aligned	O	rx_clk_out	All PCS Lanes Aligned/Deskewed. This signal indicates whether or not all PCS lanes are aligned and deskewed. A value of 1 indicates all PCS lanes are aligned and deskewed. When this signal is a 1, the RX path is aligned and can receive packet data. When this signal is 0, a local fault condition exists. This also corresponds to MDIO register bit 3.50.12 as defined in Clause 82.3. This output is level sensitive.
stat_rx_status	O	rx_clk_out	PCS status. A value of 1 indicates that the PCS is aligned and not in hi_ber state. Corresponds to MDIO register bit 3.32.12 as defined in Clause 82.3. This output is level sensitive.
stat_rx_block_lock[VL_LANES-1:0]	O	rx_clk_out	Block lock status for each PCS lane. A value of 1 indicates that the corresponding lane has achieved block lock as defined in Clause 82. Corresponds to MDIO register bit 3.50.7:0 and 3.51.11:0 as defined in Clause 82.3. This output is level sensitive.
stat_rx_aligned_err	O	rx_clk_out	Loss of Lane Alignment/Deskew. This signal indicates that an error occurred during PCS lane alignment or PCS lane alignment was lost. A value of 1 indicates an error occurred. This output is level sensitive.
stat_rx_misaligned	O	rx_clk_out	Alignment Error. This signal indicates that the lane aligner did not receive the expected PCS lane marker across all lanes. This signal is not asserted until the PCS lane marker has been received at least once across all lanes and at least one incorrect lane marker has been received. This occurs one metaframe after the error. This signal is not asserted if the lane markers have never been received correctly. Lane marker errors are indicated by the corresponding stat_rx_mf_err signal. This output is pulsed for one clock cycle to indicate an error condition. Pulses can occur in back-to-back cycles.
stat_rx_remote_fault	O	rx_clk_out	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.
stat_rx_vl_number_[3:0]|1:0]	O	rx_clk_out	There are a total of VL_LANES separate stat_rx_vl_number[4|1:0] buses. stat_rx_vl_number_# indicates which PCS lane is being received on the corresponding physical lane. This bus is only valid when the corresponding bit of stat_rx_synced[VL_LANES-1:0] is a 1. These outputs are level sensitive.
stat_rx_vl_demuxed[VL_LANES-1:0]	O	rx_clk_out	PCS Lane Marker found. If a signal of this bus is sampled as 1, it indicates that the receiver has properly de-muxed that PCS lane. These outputs are level sensitive.
stat_rx_bad_fcs[n:0]	O	rx_clk_out	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.

Table 14: **RX Path Control/Status Signals** *(cont'd)*

Name	I/O	Clock Domain	Description
stat_rx_stomped_fcs[n:0]	O	rx_clk_out	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_truncated	O	rx_clk_out	Packet truncation indicator. A value of 1 indicates that the current packet in flight is truncated due to its length exceeding ctl_rx_max_packet_len[14:0]. This output is pulsed for one clock cycle to indicate the truncated condition. Pulses can occur in back-to-back cycles.
stat_rx_internal_local_fault	O	rx_clk_out	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	O	rx_clk_out	This signal goes High when enough local fault words are received from the link partner to trigger a fault condition as specified by the IEEE fault state machine. This signal remains High as long as the fault condition persists.
stat_rx_bip_err[VL_LANES-1:0]	O	rx_clk_out	BIP8 error indicator. A non-zero value indicates the BIP8 signature byte was in error for the corresponding PCS lane. A non-zero value is pulsed for one clock cycle. This output is pulsed for one clock cycle to indicate an error condition. Pulses can occur in back-to-back cycles.
stat_rx_hi_ber	O	rx_clk_out	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.
ctl_rx_custom_preamble_enable	I	rx_clk_out	When asserted, this signal causes the preamble to be presented on rx_preambleout.
rx_preambleout[55:0]	O	rx_clk	This bus represents the preamble bytes when the ctl_rx_custom_preamble_enable signal is asserted. It is valid on the first cycle of the packet.

Miscellaneous Status/Control Ports

The following table describes the other status/control signals.

Table 15: **Miscellaneous Status/Control Ports**

Name	I/O	Clock Domain	Description
ctl_tx_ipg_value[3:0]	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 rx_serdes_clk packets. Typical value is 12. The ctl_tx_ipg_value can also be programmed to a value in the 0 to 7 range, but in that case, it is interpreted as meaning "minimal IPG", so only Terminate code word IPG is inserted; no Idles are ever added in that case and that produces an average IPG of around 4 bytes when random-size packets are transmitted.
stat_rx_got_signal_os	O	rx_clk_out	Signal OS indication. If this bit is sampled as a 1, it indicates that a Signal OS word was received. ***Note***: Signal OS should not be received in an Ethernet network
ctl_rx_process_lfi	I	rx_clk_out	When this input is set to 1, the RX core expects and processes Local Fault (LF) control codes coming in from the SerDes. When set to 0, the RX core ignores LF control codes coming in from the SerDes.
ctl_rx_test_pattern	I	rx_clk_out	Test pattern checking enable for the RX core. A value of 1 enables test mode as defined in Clause 82.2.17. Corresponds to MDIO register bit 3.42.2 as defined in Clause 82.3. Checks for scrambled idle pattern.
ctl_tx_test_pattern	I	clk	Test pattern generation enable for the TX core. A value of 1 enables test mode as defined in Clause 82.2.10. Corresponds to MDIO register bit 3.42.7 as defined in Clause 82.3. Generates a scrambled idle pattern.
stat_rx_test_pattern_mismatch[3|2|1|0:0]	O	rx_clk_out	Test pattern mismatch increment. A non zero value in any cycle indicates how many mismatches occurred for the test pattern in the RX core. This output is only active when ctl_rx_test_pattern is set to a 1. This output can be used to generate MDIO register 3.43.15:0 as defined in Clause 82.3. This output is pulsed for one clock cycle.

Statistics Interface Ports

The following two tables describe the RX and TX path ports.

Table 16: **Statistics Interface – RX Path Signals**

Name	I/O	Clock Domain	Description
stat_rx_total_bytes[6|5|3|2:0]	O	clk	Increment for the total number of bytes received.
stat_rx_total_packets[n:0]	O	clk	Increment for the total number of packets received.
stat_rx_total_good_bytes[13:0]	O	clk	Increment for the total number of good bytes received. This value is only non-zero when a packet is received completely and contains no errors.
stat_rx_total_good_packets	O	clk	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_packet_bad_fcs	O	clk	Increment for packets between 64 and ctl_rx_max_packet_len bytes that have FCS errors.

Table 16: **Statistics Interface – RX Path Signals** *(cont'd)*

Name	I/O	Clock Domain	Description
stat_rx_packet_64_bytes	O	clk	Increment for good and bad packets received that contain 64 bytes.
stat_rx_packet_65_127_bytes	O	clk	Increment for good and bad packets received that contain 65 to 127 bytes.
stat_rx_packet_128_255_bytes	O	clk	Increment for good and bad packets received that contain 128 to 255 bytes.
stat_rx_packet_256_511_bytes	O	clk	Increment for good and bad packets received that contain 256 to 511 bytes.
stat_rx_packet_512_1023_bytes	O	clk	Increment for good and bad packets received that contain 512 to 1,023 bytes.
stat_rx_packet_1024_1518_bytes	O	clk	Increment for good and bad packets received that contain 1,024 to 1,518 bytes.
stat_rx_packet_1519_1522_bytes	O	clk	Increment for good and bad packets received that contain 1,519 to 1,522 bytes.
stat_rx_packet_1523_1548_bytes	O	clk	Increment for good and bad packets received that contain 1,523 to 1,548 bytes.
stat_rx_packet_1549_2047_bytes	O	clk	Increment for good and bad packets received that contain 1,549 to 2,047 bytes.
stat_rx_packet_2048_4095_bytes	O	clk	Increment for good and bad packets received that contain 2,048 to 4,095 bytes.
stat_rx_packet_4096_8191_bytes	O	clk	Increment for good and bad packets received that contain 4,096 to 8,191 bytes.
stat_rx_packet_8192_9215_bytes	O	clk	Increment for good and bad packets received that contain 8,192 to 9,215 bytes.
stat_rx_packet_small[n:0]	O	clk	Increment for all packets that are less than 64 bytes long. Packets that are less than 16 bytes are dropped.
stat_rx_packet_large	O	clk	Increment for all packets that are more than 9,215 bytes long.
stat_rx_unicast	O	clk	Increment for good unicast packets.
stat_rx_multicast	O	clk	Increment for good multicast packets.
stat_rx_broadcast	O	clk	Increment for good broadcast packets.
stat_rx_oversize	O	clk	Increment for packets longer than ctl_rx_max_packet_len with good FCS.
stat_rx_toolong	O	clk	Increment for packets longer than ctl_rx_max_packet_len with good and bad FCS.
stat_rx_undersize[n:0]	O	clk	Increment for packets shorter than stat_rx_min_packet_len with good FCS.
stat_rx_fragment[n:0]	O	clk	Increment for packets shorter than ctl_rx_min_packet_len with bad FCS.
stat_rx_vlan	O	clk	Increment for good 802.1Q tagged VLAN packets.
stat_rx_inrangeerr	O	clk	Increment for packets with Length field error but with good FCS.
stat_rx_jabber	O	clk	Increment for packets longer than ctl_rx_max_packet_len with bad FCS.
stat_rx_pause	O	clk	Increment for 802.3x Ethernet MAC Pause packet with good FCS.

Table 16: **Statistics Interface – RX Path Signals** *(cont'd)*

Name	I/O	Clock Domain	Description
stat_rx_user_pause	O	clk	Increment for priority-based pause packets with good FCS.
stat_tx_total_bytes[6|5|3|2:0]	O	clk	Increment for the total number of bytes transmitted.
stat_tx_total_packets	O	clk	Increment for the total number of packets transmitted.
stat_tx_total_good_bytes[13:0]	O	clk	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	O	clk	Increment for the total number of good packets transmitted.
stat_tx_bad_fcs	O	clk	Increment for packets greater than 64 bytes that have FCS errors.
stat_tx_packet_64_bytes	O	clk	Increment for good and bad packets transmitted that contain 64 bytes.
stat_tx_packet_65_127_bytes	O	clk	Increment for good and bad packets transmitted that contain 65 to 127 bytes.
stat_tx_packet_128_255_bytes	O	clk	Increment for good and bad packets transmitted that contain 128 to 255 bytes.

Table 17: **Statistics Interface – TX Path Signals**

Name	I/O	Clock Domain	Description
stat_tx_total_bytes[6|5|3|2:0]	O	clk	Increment for the total number of bytes transmitted.
stat_tx_total_packets	O	clk	Increment for the total number of packets transmitted.
stat_tx_total_good_bytes[13:0]	O	clk	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	O	clk	Increment for the total number of good packets transmitted.
stat_tx_bad_fcs	O	clk	Increment for packets greater than 64 bytes that have FCS errors.
stat_tx_packet_64_bytes	O	clk	Increment for good and bad packets transmitted that contain 64 bytes.
stat_tx_packet_65_127_bytes	O	clk	Increment for good and bad packets transmitted that contain 65 to 127 bytes.
stat_tx_packet_128_255_bytes	O	clk	Increment for good and bad packets transmitted that contain 128 to 255 bytes.
stat_tx_packet_256_511_bytes	O	clk	Increment for good and bad packets transmitted that contain 256 to 511 bytes.
stat_tx_packet_512_1023_bytes	O	clk	Increment for good and bad packets transmitted that contain 512 to 1,023 bytes.
stat_tx_packet_1024_1518_bytes	O	clk	Increment for good and bad packets transmitted that contain 1,024 to 1,518 bytes.
stat_tx_packet_1519_1522_bytes	O	clk	Increment for good and bad packets transmitted that contain 1,519 to 1,522 bytes.
stat_tx_packet_1523_1548_bytes	O	clk	Increment for good and bad packets transmitted that contain 1,523 to 1,548 bytes.
stat_tx_packet_1549_2047_bytes	O	clk	Increment for good and bad packets transmitted that contain 1,549 to 2,047 bytes.

Table 17: **Statistics Interface – TX Path Signals** *(cont'd)*

Name	I/O	Clock Domain	Description
stat_tx_packet_2048_4095_bytes	O	clk	Increment for good and bad packets transmitted that contain 2,048 to 4,095 bytes.
stat_tx_packet_4096_8191_bytes	O	clk	Increment for good and bad packets transmitted that contain 4,096 to 8,191 bytes.
stat_tx_packet_8192_9215_bytes	O	clk	Increment for good and bad packets transmitted that contain 8,192 to 9,215 bytes.
stat_tx_packet_small	O	clk	Increment for all packets that are less than 64 bytes long.
stat_tx_packet_large	O	clk	Increment for all packets that are more than 9,215 bytes long.
stat_tx_unicast	O	clk	Increment for good unicast packets.
stat_tx_multicast	O	clk	Increment for good multicast packets.
stat_tx_broadcast	O	clk	Increment for good broadcast packets.
stat_tx_vlan	O	clk	Increment for good 802.1Q tagged VLAN packets.
stat_tx_pause	O	clk	Increment for 802.3x Ethernet MAC Pause packet with good FCS.
stat_tx_user_pause	O	clk	Increment for priority-based pause packets with good FCS.
stat_tx_frame_error	O	clk	Increment for packets with tx_errin set to indicate an EOP abort.

Pause Interface Ports

Pause Interface - Control Ports

The following table describes the control signals.

Table 18: **Pause Interface – Control Ports**

Name	I/O	Clock Domain	Description
ctl_rx_pause_enable[8:0]	I	rx_serdes_clk	RX pause enable signal. This input is used to enable the processing of the pause quanta for the corresponding priority. ***Note***: This signal only affects the RX user interface, not the pause processing logic.
ctl_tx_pause_enable[8:0]	I	clk	TX pause enable signal. This input is used to enable the processing of the pause quanta for the corresponding priority. This signal gates transmission of pause packets.

Pause Interface - RX Path Ports

The following table describes the RX path ports.

Table 19: **Pause Interface – RX Path Ports**

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

Table 19: **Pause Interface – RX Path Ports** *(cont'd)*

Name	I/O	Clock Domain	Description
ctl_rx_check_ucast_ppp	I	rx_serdes_clk	A value of 1 enables priority pause unicast destination address processing.
ctl_rx_check_sa_ppp	I	rx_serdes_clk	A value of 1 enables priority pause source address processing.
ctl_rx_check_etype_ppp	I	rx_serdes_clk	A value of 1 enables priority pause ethertype processing.
ctl_rx_etype_ppp[15:0]	I	rx_serdes_clk	Ethertype field for priority pause processing.
ctl_rx_check_opcode_ppp	I	rx_serdes_clk	A value of 1 enables priority pause opcode processing.
ctl_rx_opcode_ppp[15:0]	I	rx_serdes_clk	Priority pause opcode value.
stat_rx_pause_req[8:0]	O	rx_serdes_clk	Pause request signal. When the RX receives a valid pause frame, it sets the corresponding bit of this bus to a 1 and keeps it at 1 until the pause packet has been processed. See Appendix C: Pause Processing Interface for pause interface details.
ctl_rx_pause_ack[8:0]	I	rx_serdes_clk	Pause acknowledge signal. This bus is used to acknowledge the receipt of the pause frame from the user logic. See Appendix C: Pause Processing Interface for pause interface details.
ctl_rx_check_ack	I	rx_serdes_clk	Wait for acknowledge. If this input is set to 1, the 40G/50G High Speed Ethernet Subsystem uses the ctl_rx_pause_ack[8:0] bus for pause processing. If this input is set to 0, ctl_rx_pause_ack[8:0] is not used.
ctl_rx_forward_control	I	rx_serdes_clk	A value of 1 indicates that the 40G/50G High Speed Ethernet Subsystem forwards control packets to you. A value of 0 causes the 40G/ 50G High Speed Ethernet Subsystem to drop control packets. See Appendix C: Pause Processing Interface for control/pause packet processing.
stat_rx_pause_valid[8:0]	O	rx_serdes_clk	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_quanta[8:0] [15:0]	O	rx_serdes_clk	These nine buses indicate the quanta received for each of the eight priorities in priority-based pause operation and global pause operation. If an 802.3x Ethernet MAC Pause packet is received, the quanta are placed in value [8].

Pause Interface - TX Path Ports

The following table describes the TX path signals.

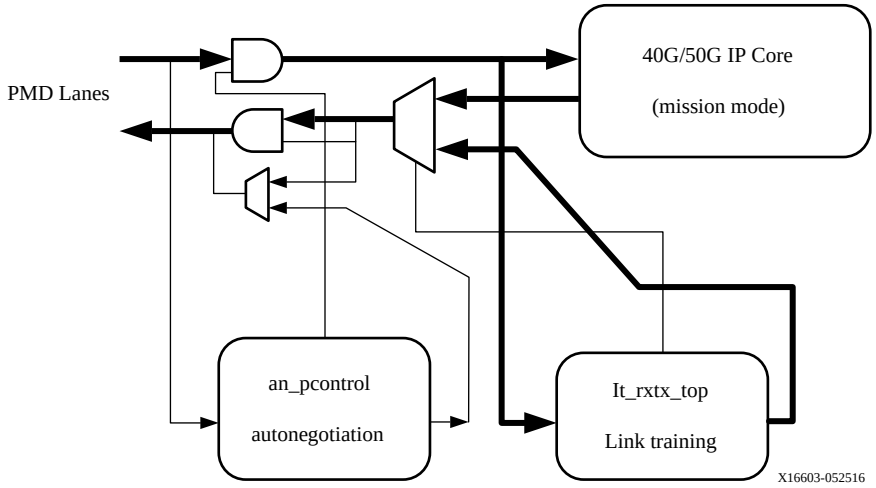
Table 20: **Pause Interface – TX Path Ports**

Name	I/O	Clock Domain	Description
ctl_tx_pause_req[8:0]	I	clk	If a bit of this bus is set to 1, the 40G/50G High Speed Ethernet Subsystem 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_quanta[8:0][15:0]	I	clk	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.
ctl_tx_pause_refresh_timer[8:0][15:0]	I	clk	These nine buses set the retransmission time of pause packets for each of the eight priorities in priority-based pause operation and the global pause operation. The value for ctl_tx_pause_refresh_timer[8] is used for global pause operation. All other values are used for priority pause operation.
ctl_tx_da_gpp[47:0]	I	clk	Destination address for transmitting global pause packets.
ctl_tx_sa_gpp[47:0]	I	clk	Source address for transmitting global pause packets.
ctl_tx_ethertype_gpp[15:0]	I	clk	Ethertype for transmitting global pause packets.
ctl_tx_opcode_gpp[15:0]	I	clk	Opcode for transmitting global pause packets.
ctl_tx_da_ppp[47:0]	I	clk	Destination address for transmitting priority pause packets.
ctl_tx_sa_ppp[47:0]	I	clk	Source address for transmitting priority pause packets.
ctl_tx_ethertype_ppp[15:0]	I	clk	Ethertype for transmitting priority pause packets.
ctl_tx_opcode_ppp[15:0]	I	clk	Opcode for transmitting priority pause packets.
ctl_tx_resend_pause	I	clk	Re-transmit pending pause packets. When this input is sampled as 1, all pending pause packets are retransmitted as soon as possible (that is, after the current packet in flight is completed) and the retransmit counters are reset. This input should be pulsed to 1 for one cycle at a time.
stat_tx_pause_valid[8:0]	O	clk	If a bit of this bus is set to 1, the 40G/50G High Speed Ethernet Subsystem 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.

Auto-Negotiation (AN) and Link Training (LT)

The 40G/50G IP core supports Auto-Negotiation and Link Training. A block diagram of the 40G/50G IP core with AN and LT is illustrated in the following figure.

Figure 13: **40G/50G IP core with Auto-Negotiation and Link Training**



The Auto-Negotiation function allows an Ethernet device to advertise the modes of operation it possesses to another device at the remote end of a Backplane Ethernet link and to detect corresponding operational modes the other device might be advertising. The objective of this Auto-Negotiation function is to provide the means to exchange information between two devices and to automatically configure them to take maximum advantage of their abilities. It has the additional objective of supporting a digital signal detect to ensure that the device is attached to a link partner rather than detecting a signal due to crosstalk. When Auto-Negotiation is complete, the ability is reported according to the available modes of operation.

Link Training (LT) is performed after AN if the LT function is supported by both ends of the link. Link training is typically required due to frequency-dependent losses that can occur as digital signals traverse the backplane. The primary function of the LT block included with this IP core is to provide register information and a training sequence over the backplane link which is then analyzed by a receiving circuit (part of the SerDes).

The other function of the LT block is to communicate training feedback from the receiver to the corresponding transmitter so that its pre-emphasis circuit (part of the SerDes) can be adjusted as required. The decision-making algorithm is not part of this IP core.

When AN and LT are complete, the datapath is switched to mission mode (see the above figure).

Port List — Auto-Negotiation

The following additional signals are used for the auto-negotiation function. These signals are found at the `*wrapper.v` hierarchy.

Table 21: **Auto-Negotiation Ports**

Port Name	I/O	Clock Domain	Description and Notes
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.
an_reset	I	an_clk	Synchronous active-High reset corresponding to an_clk domain.
ctl_autoneg_enable	I	an_clk	Enable signal for auto-negotiation.
ctl_autoneg_bypass	I	an_clk	Input to disable auto-negotiation and bypass the auto-negotiation function. If this input is asserted, auto-negotiation is turned off, but the PCS is connected to the output to allow operation.
ctl_an_nonce_seed[7:0]	I	an_clk	8-bit seed to initialize the nonce field polynomial generator. Non-zero. The auto-negotiation does not function if this is zero.
ctl_an_pseudo_sel	I	an_clk	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, the polynomial is x7+x3+1.
ctl_restart_negotiation	I	an_clk	This input is used to trigger a restart of the auto-negotiation, regardless of what state the circuit is currently in.
ctl_an_local_fault	I	an_clk	This input signal is used to set the remote_fault bit of the transmit link codeword.
Signals Used for PAUSE Ability Advertising			
ctl_an_pause	I	an_clk	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.
ctl_an_asmdir	I	an_clk	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.

Table 21: **Auto-Negotiation Ports** *(cont'd)*

Port Name	I/O	Clock Domain	Description and Notes
Ability Signal Inputs			
ctl_an_ability_1000base_kx	I	an_clk	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_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_krcr	I	an_clk	
ctl_an_ability_25gbase_cr1	I	an_clk	
ctl_an_ability_25gbase_krcr_s	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_ability_2_5gbase_kx	I	an_clk	
ctl_an_ability_5gbase_kr	I	an_clk	
ctl_an_ability_50gbase_krcr	I	an_clk	
ctl_an_ability_200gbase_kr4cr4	I	an_clk	
ctl_an_ability_100gbase_kr2cr2	I	an_clk	
ctl_an_fec_10g_request	I	an_clk	Used to set the clause 74 FEC request bit in the transmit link codeword. This signal applies only to PMDs running at 10 Gb/s. It is ignored in PMDs running any other rate.
ctl_an_fec_ability_override	I	an_clk	Used to set the clause 74 FEC ability bit in the transmit link codeword. If this input is set, 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.
ctl_an_cl91_fec_ability	I	an_clk	This bit is used to set clause 91 FEC ability.
ctl_an_cl91_fec_request	I	an_clk	This bit is used to request clause 91 FEC.
ctl_an_fec_25g_rs_request	I	an_clk	Used to set the RS-FEC request bit in the transmit link codeword. This signal applies only to PMDs running at 25 Gb/s. It is ignored in PMDs running any other rate.
ctl_an_fec_25g_baser_request	I	an_clk	Used to set the clause 74 FEC request bit in the transmit link codeword. This signal applies only to PMDs running at 25 Gb/s. It is ignored in PMDs running any other rate.
stat_an_rxcdrhold	O	an_clk	Used to set the rxcdrhold_in of the GT during auto-negotiation.

Table 21: **Auto-Negotiation Ports** *(cont'd)*

Port Name	I/O	Clock Domain	Description and Notes
stat_an_link_cntl_1000base_kx[1:0]	O	an_clk	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_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_krcr[1:0]	O	an_clk	
stat_an_link_cntl_25gbase_cr1[1:0]	O	an_clk	
stat_an_link_cntl_25gbase_krcr_s[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	an_clk	Used to enable the use of clause 74 FEC on the link.
stat_an_rs_fec_enable	O	an_clk	Used to enable the use of clause 91 FEC on the link.
stat_an_tx_pause_enable	O	an_clk	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	O	an_clk	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	O	an_clk	Indicates the auto-negotiation is complete and rx link status from the PCS has been received.
stat_an_parallel_detection_fault	O	an_clk	Indicated a parallel detection fault during auto-negotiation.
stat_an_lp_ability_1000base_kx	O	an_clk	These signals indicate the advertised protocol from the link partner. They all become valid when the output signal stat_an_lp_ability_valid is asserted. A value of 1 indicates that the protocol is advertised as supported by the link partner.
stat_an_lp_ability_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_krcr	O	an_clk	
stat_an_lp_ability_25gbase_krcr_s	O	an_clk	
stat_an_lp_ability_40gbase_cr4	O	an_clk	
stat_an_lp_ability_40gbase_kr4	O	an_clk	

Table 21: **Auto-Negotiation Ports** *(cont'd)*

Port Name	I/O	Clock Domain	Description and Notes
stat_an_lp_ability_25gbase_cr1	O	an_clk	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_25gbase_kr1	O	an_clk	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	O	an_clk	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	O	an_clk	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_pause	O	an_clk	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	O	an_clk	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_fec_10g_ability	O	an_clk	This signal indicates the advertised value of the clause 74 FEC ability bit in the receive link codeword on the corresponding 10 Gb/s PMD interface from the link partner. It becomes valid when the output signal stat_an_lp_ability_valid is asserted.
stat_an_lp_fec_10g_request	O	an_clk	This signal indicates the advertised value of the clause 74 FEC Request bit in the receive link codeword on the corresponding 10 Gb/s PMD interface from the link partner. It becomes valid when the output signal stat_an_lp_ability_valid is asserted.
stat_an_lp_fec_25g_rs_request	O	an_clk	This signal indicates the advertised value of the RS-FEC request bit in the receive link codeword on the corresponding 25 Gb/s PMD interface from the link partner. It becomes valid when the output signal.
stat_an_lp_fec_25g_baser_request	O	an_clk	This signal indicates the advertised value of the clause 74 FEC request bit in the receive link codeword on the corresponding 25 Gb/s PMD interface from the link partner. It becomes valid when the output signal stat_an_lp_ability_valid is asserted.
stat_an_lp_autoneg_able	O	an_clk	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.

Table 21: **Auto-Negotiation Ports** *(cont'd)*

Port Name	I/O	Clock Domain	Description and Notes
stat_an_lp_ability_valid	O	an_clk	This signal indicates when all of the link partner advertisements become valid.
an_loc_np_data[47:0]	I	an_clk	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, and 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[47:0]	O	an_clk	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.
ctl_an_loc_np	I	an_clk	Local Next Page indicator. If this bit is 1, the AN IP transfers the next page word at input loc_np_data to the remote link partner. If this bit is 0, 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, the AN IP transfers null message pages.
stat_fec_inc_cant_correct_count[3:0]	O	rx_serdes_clk	Logical indication of uncorrectable errors. If an uncorrectable packet is encountered, this output signal cycles once. The signal is High for a minimum of 16 clocks and goes Low for a minimum of 16 clocks. There is one per lane.
stat_fec_inc_correct_count[3:0]	O	rx_serdes_clk	Logical indication of correctable errors. If a correctable packet is encountered, this output signal cycles once. The signal is High for a minimum of 16 clocks and goes Low for a minimum of 16 clocks. There is one per lane.
stat_fec_lock_error[3:0]	O	rx_serdes_clk	Logical indication of a failure to achieve a frame lock. The receiver scans the incoming data stream for about 10,000,000 bits, attempting all possible bit alignments for frame synchronization. After this time, this signal is asserted High and remains High until the receiver achieves a frame lock. There is one per lane.
stat_fec_rx_lock[3:0]	O	rx_serdes_clk	Logical indication of a frame lock. The receiver asserts this signal High when it achieves a frame lock to the incoming bitstream. There is one per lane.
ctl_an_lp_np_ack	I	an_clk	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.

Table 21: **Auto-Negotiation Ports** *(cont'd)*

Port Name	I/O	Clock Domain	Description and Notes
stat_an_loc_np_ack	O	an_clk	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, it must clear the loc_np input.
stat_an_lp_np	O	an_clk	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_extended_fec[1:0]	O	an_clk	This output indicates the extended FEC abilities as defined in Schedule 3.
stat_an_lp_extended_ability_valid	O	an_clk	When this bit is 1, it indicates that the detected extended abilities are valid.
stat_an_lp_rf	O	an_clk	This bit indicates link partner remote fault.
stat_an_start_tx_disable	O	an_clk	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 the state TX_DISABLE, this output will cycle High for 1 clock period. It effectively signals the start of auto-negotiation.
stat_an_start_an_good_check	O	an_clk	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.

Port List — Link Training

The following additional signals are used for the link-training function. These signals are found at the `*wrapper.v` hierarchy.

Table 22: **Link Training Ports**

Port Name	I/O	Clock Domain	Description and Notes
ctl_lt_training_enable	I	tx_serdes_clk	Enables link training. When link training is disabled, all PCS lanes function in mission mode.
ctl_lt_restart_training	I	tx_serdes_clk	This signal triggers a restart of link training regardless of the current state.
ctl_lt_rx_trained[4-1:0]	I	tx_serdes_clk	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.
stat_lt_signal_detect[4-1:0]	O	tx_serdes_clk	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[4-1:0]	O	tx_serdes_clk	This signal indicates when the respective link training state machine is performing link training.
stat_lt_training_fail[4-1:0]	O	tx_serdes_clk	This signal is asserted during link training if the corresponding link training state machine detects a time-out during the training period.
stat_lt_frame_lock[4-1:0]	O	tx_serdes_clk	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[4-1:0]	O	rx_serdes_clk	This signal reflects the value of the preset control bit received in the control block from the link partner.
stat_lt_initialize_from_rx[4-1:0]	O	rx_serdes_clk	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[1:0]	O	rx_serdes_clk	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[1:0]	O	rx_serdes_clk	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[1:0]	O	rx_serdes_clk	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[1:0]	O	rx_serdes_clk	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[1:0]	O	rx_serdes_clk	This 2-bit fields indicates the update status bits for the k0 coefficient, as received from the link partner in the status block.
stat_lt_stat_m1_from_rx0[1:0]	O	rx_serdes_clk	This 2-bit field indicates the update status bits for the k-1 coefficient, as received from the link partner in the status block.
ctl_lt_pseudo_seed0[10:0]	I	tx_serdes_clk	This 11- bit signal seeds the training pattern generator. The training pattern will not be correct if this seed is loaded with a value of zero.
ctl_lt_preset_to_tx[4-1:0]	I	tx_serdes_clk	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.
ctl_lt_initialize_to_tx[4-1:0]	I	tx_serdes_clk	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.

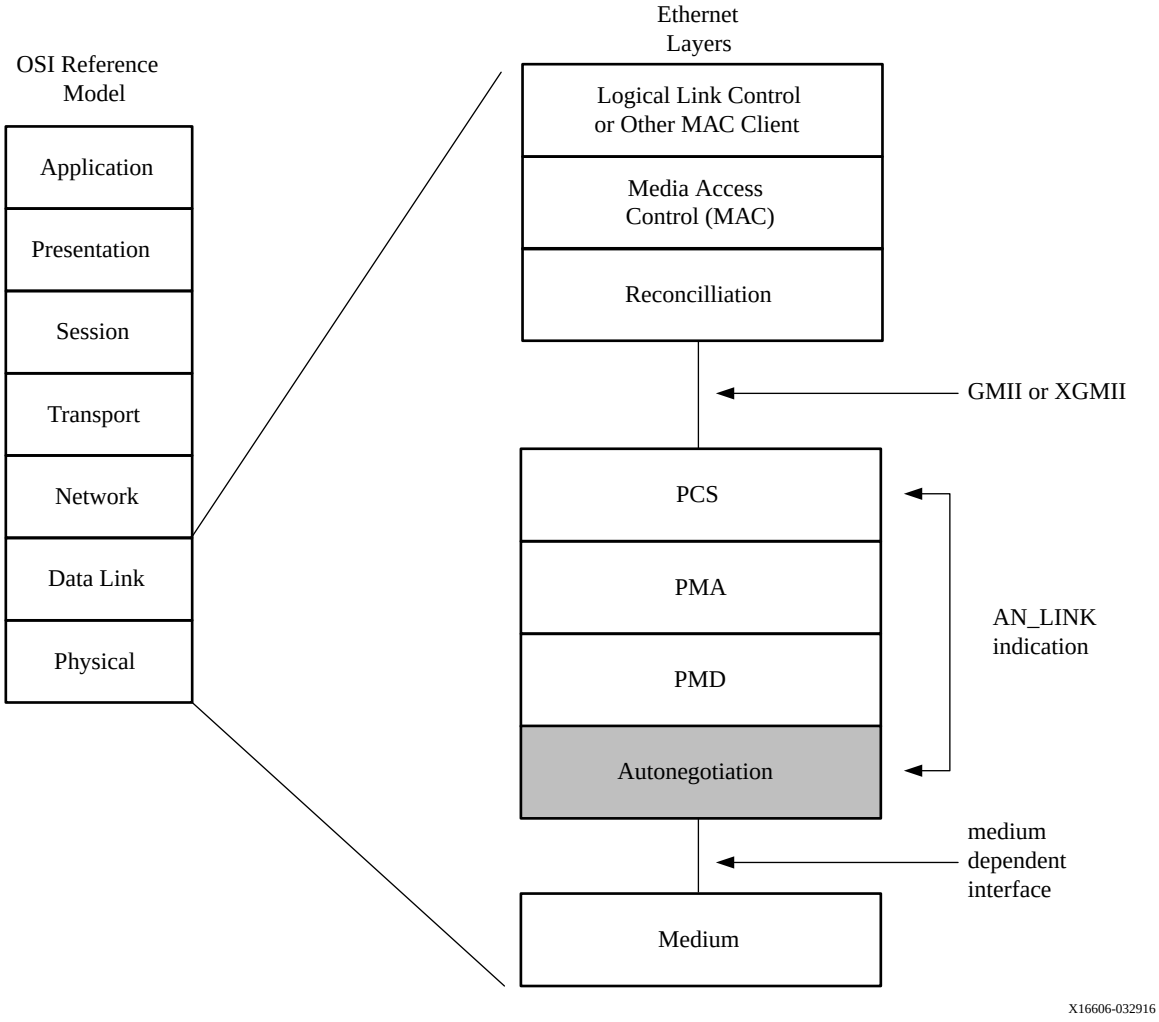
Table 22: **Link Training Ports** *(cont'd)*

Port Name	I/O	Clock Domain	Description and Notes
ctl_lt_k_p1_to_tx0[1:0]	I	tx_serdes_clk	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.
ctl_lt_k0_to_tx0[1:0]	I	tx_serdes_clk	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,.
ctl_lt_k_m1_to_tx0[1:0]	I	tx_serdes_clk	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.
ctl_lt_stat_p1_to_tx0[1:0]	I	tx_serdes_clk	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.
ctl_lt_stat0_to_tx0[1:0]	I	tx_serdes_clk	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.
ctl_lt_stat_m1_to_tx0[1:0]	I	tx_serdes_clk	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,.
stat_lt_rx_sof[4-1:0]	O	rx_serdes_clk	This output is High for 1 RX SerDes clock cycle to indicate the start of the link training frame.

Overview

The following figure as per IEEE P802.3 illustrates the position of the AN function in the OSI reference model.

Figure 14: **Auto-Negotiation in OSI Model**



The Auto-Negotiation IP core implements the requirements as specified in Clause 73, IEEE Std 802.3-2015, including those amendments specified in IEEE P802.3by and Schedule 3 of the 25 GE Consortium.

The functions of the AN IP core are explicitly listed in clause 73, especially in Figure 73-11, Arbitration state diagram, of 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 SerDes 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 quite a bit different than the mission mode operation.

To accommodate this requirement, the AN IP 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.

Auto-Negotiation Description

- **autoneg_enable:** When the `autoneg_enable` input signal is set to a 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 value for every instance of the auto-negotiator. The AN IP will not function if the `nonce_seed` is set to 0. This is important in order to guarantee that no deadlocks 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 in bit position 49 of the DME pages used during auto-negotiation. Any selection on this input is valid and will not result in any adverse behavior.
- **Link Control:** When auto-negotiation has begun, then 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 then monitored by the AN IP hardware for an indication of the state of the various links that can be 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 AN IP hardware sets up a communication link with the link partner and uses this link to negotiate the capabilities of the connection.
- **Autoneg Complete:** When Auto-Negotiation is complete, the `autoneg_complete` output signal is asserted. In addition to this, the output signal `an_fec_enable` is asserted if the Forward Error Correction hardware is to be used; the output signal `tx_pause_en` is asserted if the transmitter hardware is allowed to generate PAUSE control packets, the output signal `rx_pause_en` is asserted if the receiver hardware is allowed to detect PAUSE control packets, and the output link control of the selected link is set to its mission mode value (bin)11.

IMPORTANT! *The autoneg complete signal is not asserted until rx_status is received from the PCS. That means that, where link training is included, the autoneg_complete output signal is not asserted until after link training has completed and rx_status is High.*

Link Training Description

Overview

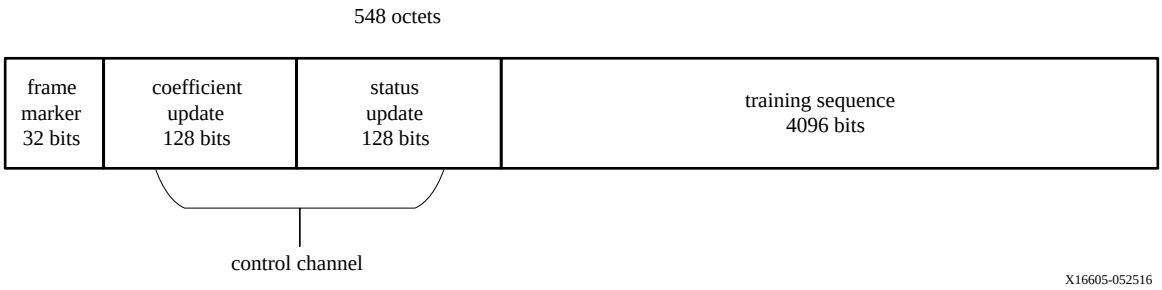
Link Training (LT) is performed after auto-negotiation (AN) converges to a backplane or copper technology. Technology selection can also be the result of a manual entry or parallel detection. Link training might be required due to frequency-dependent losses that can occur as digital signals traverse the backplane or a copper cable. The primary function of the LT IP core is to provide register information and a training sequence over the backplane link which is then analyzed by a receiving circuit that is not part of the IP core. The other function of the IP core is to communicate training feedback from the receiver to the corresponding transmitter so that its equalizer circuit (not part of the IP core) can be adjusted as required. The two circuits comprising the IP core are the receive Link Training block and the transmit Link Training block.

***Note*:** The logic responsible for the adjustment of the transmitter pre-emphasis must be supplied external to this IP core.

Transmit

The LT transmit block constructs a 4,384-bit frame that contains a frame delimiter, control channel, and link training sequence. It is formatted as follows:

Figure 15: **Link Training Frame Structure**



It is recommended 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 will begin when `tx_SOF` is asserted and ends at least 288 bit times later, or approximately 30 ns.

Note that 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 8. 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 8, each coefficient and status transmission contain 128/8=16 bits each numbered from 15:0. The following two tables define these bits in the order in which they are transmitted starting with bit 15 and ending with bit 0.

Table 23: **Coefficient and Update Field Bit Definitions**

Bits	Name	Description
15:14	Reserved	Transmitted as 0, ignored on reception.
13	Preset	1 = Preset coefficients 0 = Normal operation
12	Initialize	1 = Initialize coefficients 0 = Normal operation
11:6	Reserved	Transmitted as 0, ignored on reception.
5:4	Coefficient (+1) update	11 = reserved 10 = decrement 01 = increment 00 = hold
3:2	Coefficient (0) update	11 = reserved 10 = decrement 01 = increment 00 = hold
1:0	Coefficient (-1) update	11 = reserved 10 = decrement 01 = increment 00 = hold

Table 24: **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	11 = maximum 10 = minimum 01 = updated 00 = not updated
3:2	Coefficient (0) update	11 = maximum 10 = minimum 01 = updated 00 = not updated
1:0	Coefficient (-1) update	11 = maximum 10 = minimum 01 = updated 00 = not updated

The functions of each bit are defined in IEEE 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 Port Descriptions

- **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. Seed must be non-zero. Subsequent to the initial seed being loaded, the PRBS generator continues to run with no further intervention being required.The PRBS generator itself is implemented with a circuit that corresponds to the following polynomial:

```
G(x)= 1 + x9 + x11
```

Receive

The receive block implements the frame alignment state diagram illustrated in IEEE 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 802.3 Clause 72, Figure 72-4. When frame lock has been achieved, the signal `frame_lock` is set to a value of TRUE.
- **Received Data:** The receiver outputs the control channel with the bit definitions previously defined in the previous two tables 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 of the 40G/50G High Speed Ethernet Using the 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 (00E0) register. Based on the requirement, you can enable or bypass the auto-negotiation. If auto-negotiation is enabled then you need to write the nonce seed value. For example, write the value 0x16D to address 0x00E0.

3. Read the CONFIGURATION_AN_CONTROL_REG2 (0x00E4) register. Based on the requirement you can enable Pause, FEC, RS-FEC, etc. by writing to address 0x00E4. For example, write the value 0x16D to address 0x00E4.
4. Enable the link training option by setting the CONFIGURATION_LT_CONTROL_REG1 (0x0100) register link training control signal. For example, write the value 0x1 to address 0x0100.
5. Write the CONFIGURATION_LT_SEED_REG0 (0x0110) register with some seed value. For example, write the value 0x0605 to address 0x0110.
6. Write the CONFIGURATION_LT_SEED_REG1(0x0114) register with some seed valueFor example, write the value as 0x0807 to address 0x0114.
7. Write the CONFIGURATION_LT_COEFFICIENT_REG0 (0x0130) register with some coefficient values for the place holder logic. For example, write the value 0x540 to address 0x0130.
8. Write the CONFIGURATION_LT_COEFFICIENT_REG1 (0x0134) register with some coefficient values for the place holder logic. For example, write the value 0x555 to address 0x0134.
9. Issue `sys_reset`/write `1'b1` to `ctl_an_reset` that is, bit 28 of address 0x0004, so that the Auto-Negotiation block looks for the updated nonce seed value.
10. Keep reading `stat_an_autoneg_complete`, bit 2 of address 0x0458, which indicates the successful completion of the Auto negotiation and link training.

Forward Error Correction (FEC)

The 40G/50G IP core has support for any one of three FEC modes of operation as defined in Schedule 3 of the 25G Consortium:

- No FEC
- Clause 74 FEC (shortened cyclic code (2112, 2080))
- Clause 91 FEC (Reed-Solomon (528,514))

The FEC mode is communicated to the link partner during the auto-negotiation phase.

PCS Variant

A PCS-only variant of the 40G/50G Ethernet IP subsystem can be generated from the same wizard.

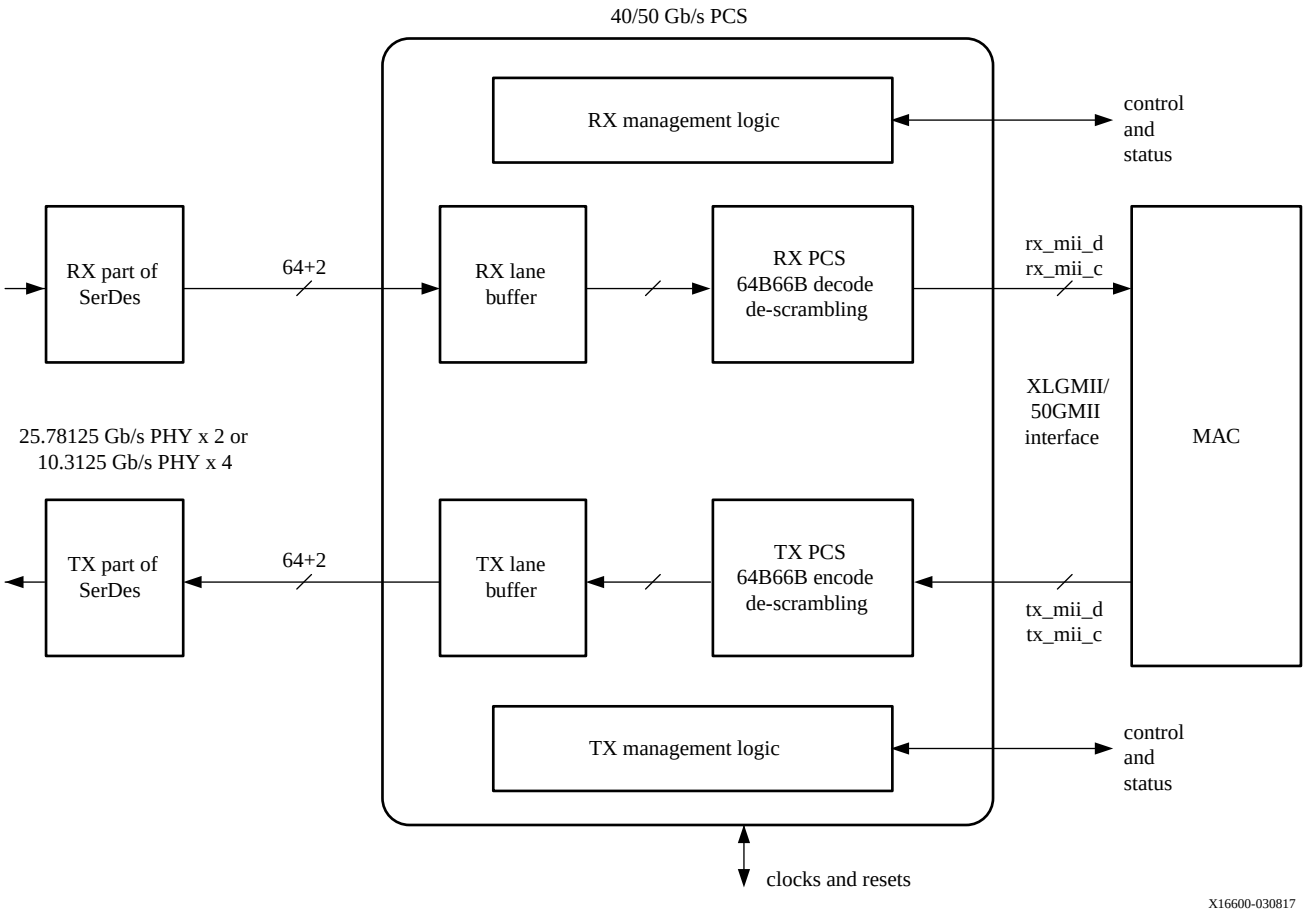
Features

- Designed to the Ethernet requirements for 50 Gb/s operation as defined in Schedule 3 of the 25G Ethernet Consortium [https://25gethernet.org/]
- Designed to the Ethernet requirements for 40 Gb/s operation as defined in IEEE 802.3 Clause 82 [*IEEE Standard for Ethernet* (IEEE Std 802.3-2015)]
- Includes complete PCS functions
- Standard XLGMII or 50GMII system side interface
- Supports Base-R 64b/66b encoding and decoding
- Optional Base-KR FEC
- Optional Auto Negotiation and Link Training

Block Diagram

The following figure is a block diagram of the 40/50 Gb/s PCS-only variant.

Figure 16: **PCS Variant**



Port List — PCS-Only

The following table shows the 40G/50G PCS IP core ports. These are the ports when the PCS-only option is provided. There are no FCS functions and no AXI4-Stream-related ports.

The PCS does not contain the Pause and Flow Control ports. The system interface is XLGMII/50GMII instead of the AXI4-Stream.

These signals are found at the `*wrapper.v` hierarchy. Refer to PCS Clocking for clock domain definitions.

Table 25: **PCS Variant Ports**

Name	I/O	Clock Domain	Description
Transceiver I/O			
rx_serdes_data_n0	I	rx_serdes_clk	Serial data from the line for lane 0; negative phase of the differential signal
rx_serdes_data_p0	I	rx_serdes_clk	Serial data from the line for lane 0; positive phase of the differential signal
tx_serdes_data_n0	O	tx_serdes_clk	Serial data to the line for lane 0; negative phase of the differential signal.
tx_serdes_data_p0	O	tx_serdes_clk	Serial data to the line for lane 0; positive phase of the differential signal.
rx_serdes_data_n1	I	rx_serdes_clk	Serial data from the line for lane 1; negative phase of the differential signal
rx_serdes_data_p1	I	rx_serdes_clk	Serial data from the line for lane 1; positive phase of the differential signal
tx_serdes_data_n1	O	tx_serdes_clk	Serial data to the line for lane 1; negative phase of the differential signal.
tx_serdes_data_p1	O	tx_serdes_clk	Serial data to the line for lane 1; positive phase of the differential signal.
GT_reset (ctl_gt_reset_all)	I	async	Active-High reset for the transceiver startup FSM. Note that this signal also initiates the reset sequence for the entire IP core.
refclk_n0	I		Differential reference clock input for the SerDes, negative phase.
refclk_p0	I		Differential reference clock input for the SerDes, negative phase.
XLGMII/50GMII Interface Signals			
rx_mii_d[127:0]	O	rx_mii_clk	Receive XLGMII/50GMII Data bus.
rx_mii_c[15:0]	O	rx_mii_clk	Receive XLGMII/50GMII Control bus.
rx_mii_clk	I		Receive XLGMII/50GMII Clock input.
tx_mii_d[127:0]	I	tx_mii_clk	Transmit XLGMII/50GMII Data bus.
tx_mii_c[15:0]	I	tx_mii_clk	XLGMII/50GMII Control bus.
tx_mii_clk	I	tx_mii_clk	Transmit XLGMII/50GMII Clock input.
rx_mii_reset	I		Reset input for the RX XLGMII/50GMII interface.
tx_mii_reset	I		Reset input for the TX XLGMII/50GMII interface.

Table 25: **PCS Variant Ports** *(cont'd)*

Name	I/O	Clock Domain	Description
rx_serdes_clk	I		Input clock signal used for clocking the core logic of the RX PCS.
tx_core_clk	I		Input clock signal used for clocking the core logic of the TX PCS.
rx_reset	I	rx_serdes_clk	Reset associated with the rx_serdes_clk logic. Must be synchronous to rx_serdes_clk.
tx_reset	I	tx_core_clk	Reset associated with the tx_core_clk logic. Must be synchronous to tx_core_clk.
XLGMII/50GMII Interface – Control/Status Signals			
ctl_rx_vl_length_minus1[15:0]	I	static	Number of words in between PCS Lane markers minus one for RX. Default value, as defined in the IEEE 802.3, should be set to 16,383. This input should only be changed while the corresponding reset input is asserted. ***Note***: When RS-FEC is enabled in the 50G core configuration, this value will be set to 20479.
ctl_tx_vl_length_minus1[15:0]	I	static	Number of words in between PCS Lane markers minus one for TX. Default value, as defined in the IEEE 802.3, should be set to 16,383. This input should only be changed while the corresponding reset input is asserted. ***Note***: When RS-FEC is enabled in the 50G core configuration, this value will be set to 20479.
ctl_rx_vl_marker_id0[63:0]	I	static	PCS Lane marker for RX PCS lane0. For IEEE 802.3 default values, see RX and TX PCS Lane Marker Values. This input should only be changed while the corresponding reset input is asserted.
ctl_rx_vl_marker_id1[63:0]	I	static	PCS Lane marker for RX PCS lane1.
ctl_rx_vl_marker_id2[63:0]	I	static	PCS Lane marker for RX PCS lane2.
ctl_rx_vl_marker_id3[63:0]	I	static	PCS Lane marker for RX PCS lane3.
ctl_tx_vl_marker_id0[63:0]	I	static	PCS Lane marker for TX PCS lane0. For IEEE 802.3 default values, see RX and TX PCS Lane Marker Values. This input should only be changed while the corresponding reset input is asserted.
ctl_tx_vl_marker_id1[63:0]	I	static	PCS Lane marker for TX PCS lane1.
ctl_tx_vl_marker_id2[63:0]	I	static	PCS Lane marker for TX PCS lane2.
ctl_tx_vl_marker_id3[63:0]	I	static	PCS Lane marker for TX PCS lane3.
ctl_rx_test_pattern	I	rx_serdes_clk	Test pattern enable for the RX core to receive scrambled idle pattern. Takes third precedence.
ctl_tx_test_pattern	I	tx_core_clk	Scrambled idle Test pattern generation enable for the TX core. A value of 1 enables test mode. Takes third precedence.

Table 25: **PCS Variant Ports** *(cont'd)*

Name	I/O	Clock Domain	Description
stat_rx_fifo_error	O	rx_serdes_clk	Receive clock compensation FIFO error indicator. A value of 1 indicates the clock compensation FIFO under or overflowed. This condition only occurs if the PPM difference between the recovered clock and the local reference clock is greater than ±200 ppm. If this output is sampled as a 1 in any clock cycle, the corresponding port must be reset to resume proper operation.
stat_rx_local_fault	O	rx_serdes_clk	A value of 1 indicates the receive decoder state machine is in the RX_INIT state. This output is level sensitive.
stat_rx_hi_ber	O	rx_serdes_clk	High Bit Error Rate (BER) indicator. When set to 1, the BER is too high as defined by the IEEE 802.3. Corresponds to MDIO register bit 3.32.1 as defined in Clause 82.3. This output is level sensitive.
stat_rx_block_lock[3:0]	O	rx_serdes_clk	Block lock status for each PCS lane. A value of 1 indicates the corresponding lane has achieved a 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_error[7:0]	O	rx_serdes_clk	Test pattern mismatch increment. A non-zero value in any cycle indicates how many mismatches occurred for the test pattern in the RX core. This output is only active when ctl_rx_test_pattern is set to a 1. This output can be used to generate MDIO register 3.43.15:0 as defined in Clause 82.3. This output is pulsed for one clock cycle.
stat_rx_error_valid	O	rx_serdes_clk	Increment valid indicator. If this signal is a 1 in any clock cycle, the value of stat_rx_error is valid.
stat_rx_bad_code[1:0]	O	rx_serdes_clk	Increment for 64B/66B code violations. This signal indicates the number of 64b/ 66b words received with an invalid block or if a wrong 64b/66b block sequence was detected. This output can be used to generate MDIO register 3.33:7:0 as defined in Clause 82.3.
stat_rx_bad_code_valid	O	rx_serdes_clk	Increment valid indicator. If this signal is a 1 in any clock cycle, the value of stat_rx_bad_code is valid.
stat_rx_framing_err_0[3:0]	O	rx_serdes_clk	Increment value for number of sync header errors detected for PCS lane 0. The value of this bus is only valid in the same cycle that stat_rx_framing_err_valid_0 is a 1.
stat_rx_framing_err_1[3:0]	O	rx_serdes_clk	Increment value for number of sync header errors detected for PCS lane 1.
stat_rx_framing_err_2[3:0]	O	rx_serdes_clk	Increment value for number of sync header errors detected for PCS lane 2.
stat_rx_framing_err_3[3:0]	O	rx_serdes_clk	Increment value for number of sync header errors detected for PCS lane 3.
stat_rx_valid_0	O	rx_serdes_clk	Increment valid indicator for PCS lane 0. If this signal is a 1 in any clock cycle, the value of stat_rx_framing_err_0 is valid.
stat_rx_valid_1	O	rx_serdes_clk	Increment valid indicator for PCS lane 1.

Table 25: **PCS Variant Ports** *(cont'd)*

Name	I/O	Clock Domain	Description
stat_rx_valid_2	O	rx_serdes_clk	Increment valid indicator for PCS lane 2.
stat_rx_valid_3	O	rx_serdes_clk	Increment valid indicator for PCS lane 3.
stat_rx_aligned	O	rx_serdes_clk	All PCS Lanes Aligned/Deskewed. This signal indicates whether or not all PCS lanes are aligned and deskewed. A value of 1 indicates all PCS lanes are aligned and deskewed. When this signal is a 1, the RX path is aligned and can receive packet data. When this signal is 0, a local fault condition exists. Also corresponds to MDIO register bit 3.50.12 as defined in Clause 82.3. This output is level sensitive.
stat_rx_aligned_err	O	rx_serdes_clk	Loss of Lane Alignment/Deskew. This signal indicates an error occurred during PCS lane alignment or virtual lane alignment was lost. A value of 1 indicates an error occurred. This output is level sensitive.
stat_rx_misaligned	O	rx_serdes_clk	Alignment Error. This signal indicates that the lane aligner did not receive the expected PCS lane marker across all lanes. This signal is not asserted until the PCS lane marker has been received at least once across all lanes. This output is pulsed for one clock cycle to indicate an error condition.
stat_rx_status	O	rx_serdes_clk	PCS status. A value of 1 indicates the PCS is aligned and not in hi_ber state. Corresponds to MDIO register bit 3.32.12 as defined in Clause 82.3. This output is level sensitive.
stat_rx_vl_demuxed[3:0]	O	rx_serdes_clk	PCS Lane Marker found. If a signal of this bus is sampled as 1, it indicates that the receiver has properly de-muxed that PCS lane. This output is level sensitive.
stat_tx_local_fault	O	tx_core_clk	A value of 1 indicates the transmit encoder state machine is in the TX_INIT state. This output is level sensitive.
stat_tx_fifo_error	O	tx_core_clk	Transmit clock compensation FIFO error indicator. A value of 1 indicates the clock compensation FIFO under or overflowed. This condition only occurs if the PPM difference between the transmitter clock and the local reference clock is greater than ±200 ppm. If this output is sampled as a 1 in any clock cycle, the corresponding port must be reset to resume proper operation.
stat_rx_vl_number_0[1:0]	O	rx_serdes_clk	The value of this bus indicates which physical lane appears on PCS lane 0. This bus is only valid when the corresponding bit of stat_rx_synced[PCS_LANES-1:0] is a 1. These outputs are level sensitive.
stat_rx_vl_number_1[1:0]	O	rx_serdes_clk	The value of this bus indicates which physical lane appears on PCS lane 1.
stat_rx_vl_number_2[1:0]	O	rx_serdes_clk	The value of this bus indicates which physical lane appears on PCS lane 2.
stat_rx_vl_number_3[1:0]	O	rx_serdes_clk	The value of this bus indicates which physical lane appears on PCS lane 3.

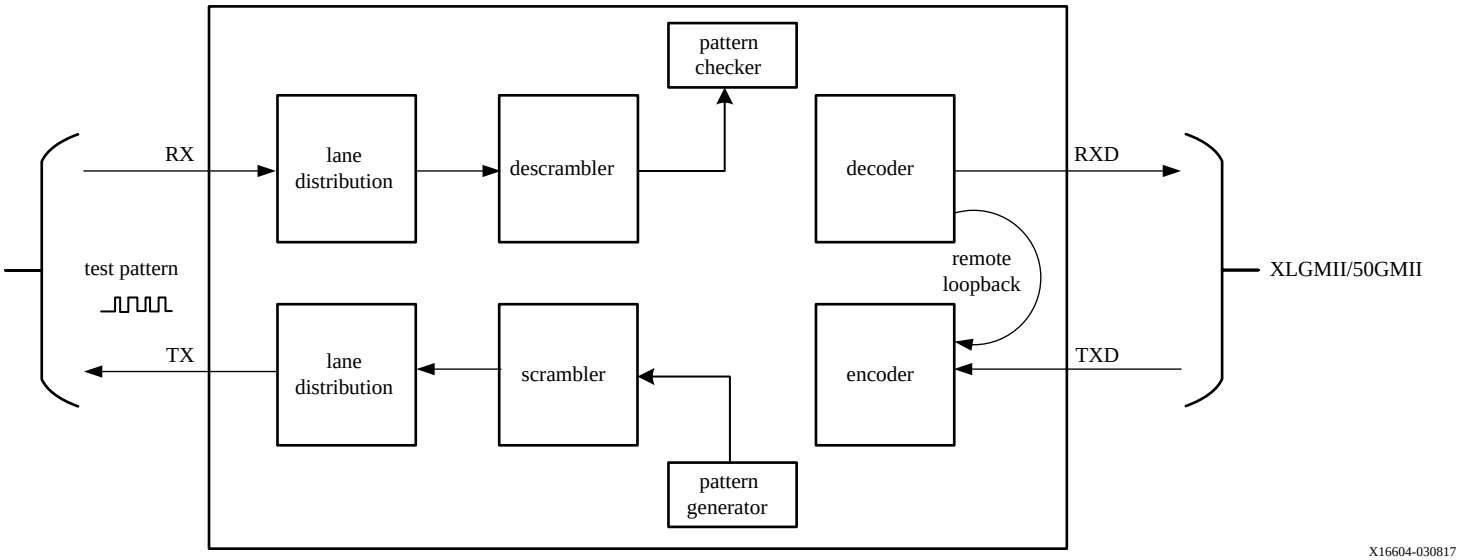
Table 25: **PCS Variant Ports** *(cont'd)*

Name	I/O	Clock Domain	Description
stat_rx_bip_err_0	O	rx_serdes_clk	BIP8 error indicator for PCS lane 0. A non-zero value indicates the BIP8 signature was in error. A non-zero value is pulsed for one clock cycle. This output is pulsed for one clock cycle to indicate an error condition.
stat_rx_bip_err_1	O	rx_serdes_clk	BIP8 error indicator for PCS lane 2.
stat_rx_bip_err_2	O	rx_serdes_clk	BIP8 error indicator for PCS lane 2.
stat_rx_bip_err_3	O	rx_serdes_clk	BIP8 error indicator for PCS lane 3.
stat_rx_synced[3:0]	O	rx_serdes_clk	Word Boundary Synchronized. These signals indicate whether a PCS lane is word boundary synchronized. A value of 1 indicates the corresponding PCS lane has achieved word boundary synchronization and it has received a PCS lane marker. Corresponds to MDIO register bit 3.52.7:0 and 3.53.11:0 as defined in Clause 82.3. This output is level sensitive.
stat_rx_synced_err[3:0]	O	rx_serdes_clk	Word Boundary Synchronization Error. These signals indicate whether an error occurred during word boundary synchronization in the respective PCS lane. A value of 1 indicates the corresponding PCS lane lost word boundary synchronization due to sync header framing bits errors or that a PCS lane marker was never received. This output is level sensitive.
stat_rx_mf_len_err[3:0]	O	rx_serdes_clk	Virtual Lane Marker Length Error. These signals indicate whether a PCS Lane Marker length mismatch occurred in the respective lane (that is, PCS Lane Markers were received not every ctl_rx_vl_length_minus1 words apart). A value of 1 indicates the corresponding lane is receiving PCS Lane Markers at wrong intervals. This output is pulsed for one clock cycle to indicate the error condition.
stat_rx_mf_repeat_err[3:0]	O	rx_serdes_clk	PCS Lane Marker Consecutive Error. These signals indicate whether four consecutive PCS Lane Marker errors occurred in the respective lane. A value of 1 indicates an error in the corresponding lane. This output is pulsed for one clock cycle to indicate the error condition.
stat_rx_mf_err	O	rx_serdes_clk	PCS Lane Marker Word Error. These signals indicate that an incorrectly formed PCS Lane Marker Word was detected in the respective lane. A value of 1 indicates an error occurred. This output is pulsed for one clock cycle to indicate the error condition.
Miscellaneous Status/Control Signals			
dclk	I	rx_serdes_clk	Dynamic Reconfiguration Port (DRP) clock input. The required frequency is set by providing the value in the GT DRP Clock field in the Vivado® IDE GT Selection and Configuration tab. This must be a free running input clock.
gt_loopback_in[12|6:0]	I	async	GT loopback input signal for each transceiver. Refer to the GT user guide. 6-bit width for the 50G single core, 12-bit width for 40G single core/ 50G two cores.

Testability

In addition to the local loopback available at the PMA, the 40G/50G PCS provides test pattern and remote loopback functions as illustrated in the following figure.

Figure 17: **PCS Loopback**

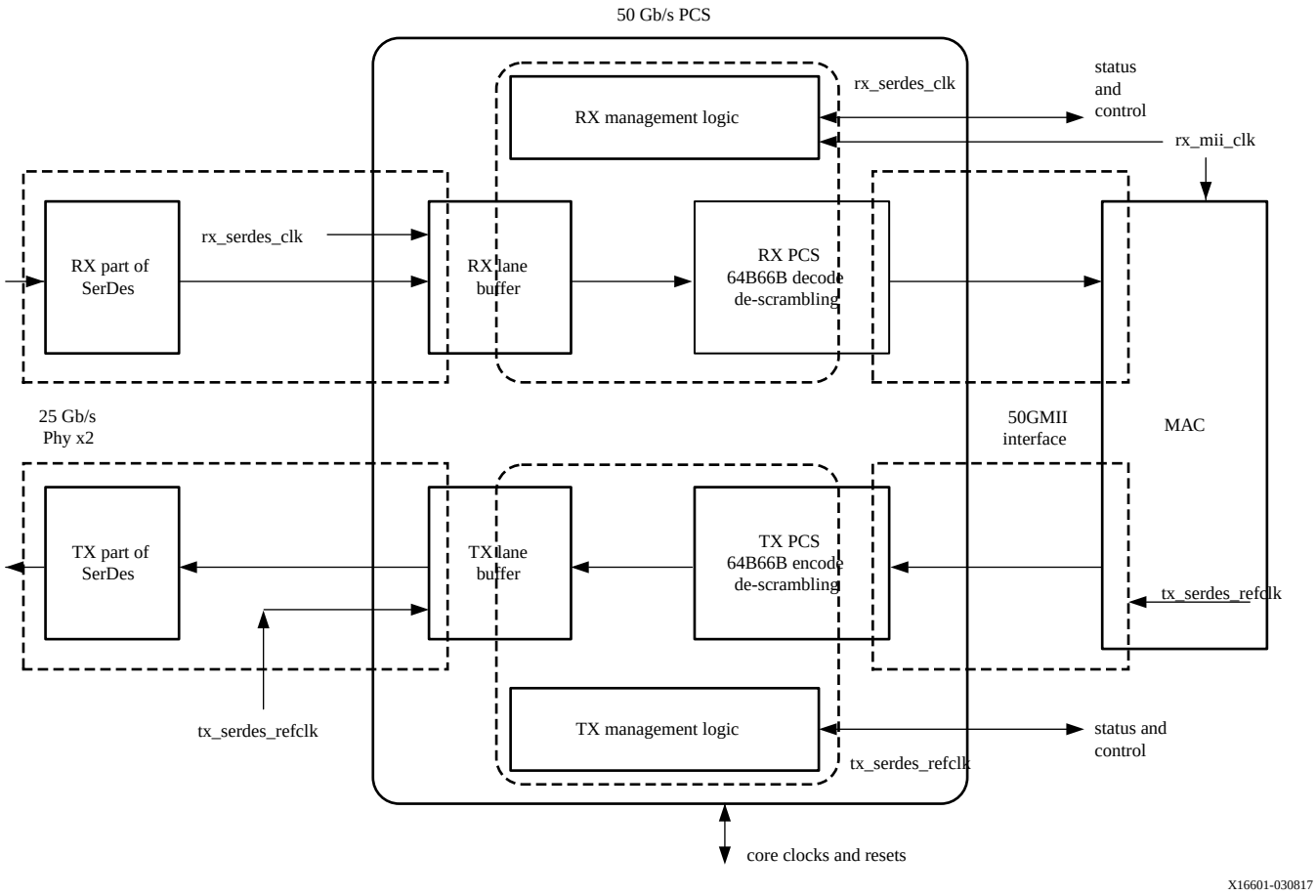


When enabled, the transmitted test pattern is continuous idle characters, which are then scrambled.

PCS Clocking

The 40G/50G High Speed Ethernet Subsystem PCS uses separate RX and TX clock domains with RX and TX comprised of three clock domains each as illustrated in the following figure.

Figure 18: **PCS Clocking**



XLGMII/50GMII Clocks

These clocks drive logic for the XLGMII/50GMII interfaces. The required clock frequency is determined by the Media Independent Interface (MII) bus width and the data rate. For example, for a 50GMII interface (50 Gb/s) and a data bus width of 128 bits, the required clock frequency is 50e9/128 = 390.625 MHz. For 40G the formula is 40e9/128 = 312.5 MHz. This clock frequency is determined according to the IEEE Standard for Ethernet (IEEE Std 802.3-2015) specification.

SerDes Clocks

- **RX:** Each SerDes lane has its own recovered clock. This clock is used for all of the logic for that SerDes lane. The 40G/50G High Speed Ethernet Subsystem synchronizes the received data from all of the SerDes to the RX core clock domains. There is one clock per SerDes lane. The SerDes clock frequency is equal to the data rate divided by the SerDes width. For example, at a data rate of 25.78125 Gb/s per lane and a 66-bit SerDes, the clock frequency is 25.78125e9/66=390.625 MHz.

- **TX:** The TX SerDes domain is associated with the TX lane logic. Both TX transceivers must be clocked with the same frequency. The frequency is calculated in the same way as documented in RX.

AXI4-Lite Register Space

The status and control signals of this Ethernet IP core can be optionally accessed by means of an AXI interface instead of the broadside bus. Detailed descriptions for each signal in the AXI register are found in the Port List — PCS-Only.

AXI4-Lite Ports

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

Table 26: **AXI4-Lite Ports**

Signal	I/O	Description
s_axi_aclk	In	AXI4-Lite clock. Range between 10 MHz and 300 MHz
s_axi_aresetn	In	Asynchronous active-Low reset
s_axi_awaddr[31:0]	In	Write address Bus
s_axi_awvalid	In	Write address valid
s_axi_awready	Out	Write address acknowledge
s_axi_wdata[31:0]	In	Write data bus
s_axi_wstrb[3:0]	In	Strobe signal for the data bus byte lane
s_axi_wvalid	Out	Write data valid
s_axi_wready	Out	Write data acknowledge
s_axi_bresp[1:0]	Out	Write transaction response
s_axi_bvalid	Out	Write response valid
s_axi_bready	In	Write response acknowledge
s_axi_araddr[31:0]	In	Read address bus
s_axi_arvalid	In	Read address valid
s_axi_arready	Out	Read address acknowledge
s_axi_rdata[31:0]	Out	Read data output
s_axi_rresp[1:0]	Out	Read data response
s_axi_rvalid	Out	Read data/response valid
s_axi_rready	In	Read data acknowledge
pm_tick	In	Top level signal to read statistics counters; requires MODE_REG[30] (tick_reg_mode_sel) to be set to 0.

Additional information for the operation of the AXI4 bus is found in “Xilinx AXI Memory-Mapped Protocol Version 1.8”. As noted previously, the top-level signal `pm_tick` can be used to read statistics counters instead of the configuration register TICK_REG. In this case, configuration register MODE_REG bit 30 (tick_reg_mode_sel) should be set to 0. If tick_reg_mode_sel is set to 1, `tick_reg` is used to read the statistics counters.

Base Pages

The Ethernet register map is divided into three sections as follows:

Table 27: **Register Map**

Address Base	Address Space Name
0x0000	IP Configuration Registers
0x0400	Status Registers
0x0500	Statistics Counters

All registers are 32 bits in size, and aligned on 32-bit addressing. The registers are designed such that the full 32b register is read/written (Byte write enables are ignored). In the below register space maps, any holes in the address space should be considered RESERVED and can cause an AXI-Ctl interface IP core to respond with an error if accessed.

When the AXI interface counters are selected, a “tick” register (TICK_REG) write/read is used to capture the statistics from the core clock domain on to the AXI clock domain, at the same time clearing the counters. After the “tick” is issued, the counters contain their updated value and can be read multiple times without destruction of this data.

The register reset signal is `s_axi_aresetn`, which is active-Low. This reset forces all registers to their default values as indicated in these tables.

Configuration Registers

The configuration space provides software with the ability to configure the IP core for various use cases. Certain features are optional (such as Auto-Negotiation, Link Training, and Flow Control), in which case the applicable registers are considered RESERVED.

Hex Address	Name/Link to Description	Note
0x0000	GT_RESET_REG: 0000	
0x0004	RESET_REG: 0004	
0x0008	MODE_REG: 0008	
0x000C	CONFIGURATION_TX_REG1: 000C	
0x0014	CONFIGURATION_RX_REG1: 0014	
0x0018	CONFIGURATION_RX_MTU: 0018	Only in MAC+PCS variant
0x001C	CONFIGURATION_VL_LENGTH_REG: 001C	

Hex Address	Name/Link to Description	Note
0x0020	TICK_REG: 0020	
0x0024	CONFIGURATION_REVISION_REG: 0024	
0x0038	CONFIGURATION_1588_REG: 0038	Only in MAC+PCS variant
0x0040	CONFIGURATION_TX_FLOW_CONTROL_REG1: 0040	Only in MAC+PCS variant
0x0044	CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG1: 0044	Only in MAC+PCS variant
0x0048	CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG2: 0048	Only in MAC+PCS variant
0x004C	CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG3: 004C	Only in MAC+PCS variant
0x0050	CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG4: 0050	Only in MAC+PCS variant
0x0054	CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG5: 0054	Only in MAC+PCS variant
0x0058	CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG1: 0058	Only in MAC+PCS variant
0x005C	CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG2: 005C	Only in MAC+PCS variant
0x0060	CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG3: 0060	Only in MAC+PCS variant
0x0064	CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG4: 0064	Only in MAC+PCS variant
0x0068	CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG5: 0068	Only in MAC+PCS variant
0x006C	CONFIGURATION_TX_FLOW_CONTROL_PPP_ETYPE_OP_REG: 006C	Only in MAC+PCS variant
0x0070	CONFIGURATION_TX_FLOW_CONTROL_GPP_ETYPE_OP_REG: 0070	Only in MAC+PCS variant
0x0074	CONFIGURATION_TX_FLOW_CONTROL_GPP_DA_REG_LSB: 0074	Only in MAC+PCS variant
0x0078	CONFIGURATION_TX_FLOW_CONTROL_GPP_DA_REG_MSB: 0078	Only in MAC+PCS variant
0x007C	CONFIGURATION_TX_FLOW_CONTROL_GPP_SA_REG_LSB: 007C	Only in MAC+PCS variant
0x0080	CONFIGURATION_TX_FLOW_CONTROL_GPP_SA_REG_MSB: 0080	Only in MAC+PCS variant
0x0084	CONFIGURATION_TX_FLOW_CONTROL_PPP_DA_REG_LSB: 0084	Only in MAC+PCS variant
0x0088	CONFIGURATION_TX_FLOW_CONTROL_PPP_DA_REG_MSB: 0088	Only in MAC+PCS variant
0x008C	CONFIGURATION_TX_FLOW_CONTROL_PPP_SA_REG_LSB: 008C	Only in MAC+PCS variant
0x0090	CONFIGURATION_TX_FLOW_CONTROL_PPP_SA_REG_MSB: 0090	Only in MAC+PCS variant
0x0094	CONFIGURATION_RX_FLOW_CONTROL_REG1: 0094	Only in MAC+PCS variant
0x0098	CONFIGURATION_RX_FLOW_CONTROL_REG2: 0098	Only in MAC+PCS variant
0x009C	CONFIGURATION_RX_FLOW_CONTROL_PPP_ETYPE_OP_REG: 009C	Only in MAC+PCS variant
0x00A0	CONFIGURATION_RX_FLOW_CONTROL_GPP_ETYPE_OP_REG: 00A0	Only in MAC+PCS variant
0x00A4	CONFIGURATION_RX_FLOW_CONTROL_GCP_PCP_TYPE_REG: 00A4	Only in MAC+PCS variant
0x00A8	CONFIGURATION_RX_FLOW_CONTROL_PCP_OP_REG: 00A8	Only in MAC+PCS variant
0x00AC	CONFIGURATION_RX_FLOW_CONTROL_GCP_OP_REG: 00AC	Only in MAC+PCS variant
0x00B0	CONFIGURATION_RX_FLOW_CONTROL_DA_REG1_LSB: 00B0	Only in MAC+PCS variant
0x00B4	CONFIGURATION_RX_FLOW_CONTROL_DA_REG1_MSB: 00B4	Only in MAC+PCS variant
0x00B8	CONFIGURATION_RX_FLOW_CONTROL_DA_REG2_LSB: 00B8	Only in MAC+PCS variant
0x00BC	CONFIGURATION_RX_FLOW_CONTROL_DA_REG2_MSB: 00BC	Only in MAC+PCS variant
0x00C0	CONFIGURATION_RX_FLOW_CONTROL_SA_REG1_LSB: 00C0	Only in MAC+PCS variant
0x00C4	CONFIGURATION_RX_FLOW_CONTROL_SA_REG1_MSB: 00C4	Only in MAC+PCS variant
0x00D0	CONFIGURATION_RSFEC_REG: 00D0	
0x00D4	CONFIGURATION_FEC_REG: 00D4	
0x00E0	CONFIGURATION_AN_CONTROL_REG1: 00E0	

Hex Address	Name/Link to Description	Note
0x00E4	CONFIGURATION_AN_CONTROL_REG2: 00E4	
0x00F8	CONFIGURATION_AN_ABILITY: 00F8	
0x0100	CONFIGURATION_LT_CONTROL_REG1: 0100	
0x0104	CONFIGURATION_LT_TRAINED_REG: 0104	
0x0108	CONFIGURATION_LT_PRESET_REG: 0108	
0x010C	CONFIGURATION_LT_INIT_REG: 010C	
0x0110	CONFIGURATION_LT_SEED_REG0: 0110	
0x0114	CONFIGURATION_LT_SEED_REG1: 0114	
0x0130	CONFIGURATION_LT_COEFFICIENT_REG0: 0130	
0x0134	CONFIGURATION_LT_COEFFICIENT_REG1: 0134	
0x0138	USER_REG_0: 0138	

Status Registers

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 the 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 28: **Status Registers**

Hex Address	Name/Link to Description	Notes
0x0400	STAT_TX_STATUS_REG1: 0400	
0x0404	STAT_RX_STATUS_REG1: 0404	
0x0408	STAT_STATUS_REG1: 0408	
0x040C	STAT_RX_BLOCK_LOCK_REG: 040C	
0x0410	STAT_RX_LANE_SYNC_REG: 0410	
0x0414	STAT_RX_LANE_SYNC_ERR_REG: 0414	
0x0418	STAT_RX_AM_ERR_REG: 0418	
0x041C	STAT_RX_AM_LEN_ERR_REG: 041C	
0x0420	STAT_RX_AM_REPEAT_ERR_REG: 0420	
0x0424	STAT_RX_LANE_DEMUXED: 0424	
0x0428	STAT_RX_PCS_LANE_NUM_REG1: 0428	
0x043C	STAT_RX_RSFEC_STATUS_REG: 043C	
0x0440	STAT_RX_RSFEC_STATUS_REG: 0440	
0x0448	STAT_RX_FEC_STATUS_REG: 0448	

Table 28: **Status Registers** *(cont'd)*

Hex Address	Name/Link to Description	Notes
0x044C	STAT_TX_RSFEC_STATUS_REG: 044C	
0x0450	STAT_TX_FLOW_CONTROL_REG1: 0450	Only in MAC+PCS variant
0x0454	STAT_RX_FLOW_CONTROL_REG1: 0454	Only in MAC+PCS variant
0x0458	STAT_AN_STATUS: 0458	
0x045C	STAT_AN_ABILITY: 045C	
0x0460	STAT_AN_LINK_CTL: 0460	
0x9F0	STAT_AN_LINK_CTL2: 09F0	
0x0464	STAT_LT_STATUS_REG1: 0464	
0x0468	STAT_LT_STATUS_REG2: 0468	
0x046C	STAT_LT_STATUS_REG3: 046C	
0x0470	STAT_LT_STATUS_REG4: 0470	
0x0474	STAT_LT_COEFFICIENT0_REG: 0474	
0x0478	STAT_LT_COEFFICIENT1_REG: 0478	
0x047C	STAT_CORE_SPEED_REG: 047C	
0x04A0	STAT_GT_WIZ_REG: 04A0	

Statistics Counters

The statistics counters provide histograms of the classification of traffic and error counts. These counters can be read either by a 1 on `pm_tick` or by writing a 1 to `TICK_REG`, depending on the value of `MODE_REG[30]` (that is tick_reg_mode_sel).

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 snapshot over the same time-interval.

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

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

The counters have a default width of 48b. The counters saturate to 1s. The values in the counters are held until the next write to the TICK_REG register.

The first of the pair of addresses shown for the counters are the addresses of the LSB register, or bits 31:0 of the count. The MSB bits 47:32 of the counter are located at + 0x4 from the LSB.

Statistic counter registers are cleared according to the following conditions.

- Applying `s_axi_aresetn` clears both the TX and RX statistics counter registers

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

Table 29: **Statistics Counters**

Hex Address	Name/Link to Description	Notes
0x0500	STATUS_CYCLE_COUNT_LSB: 0500	
0x0504	STATUS_CYCLE_COUNT_MSB: 0504	
0x0508	STAT_RX_BIP_ERR_0_LSB: 0508	
0x050C	STAT_RX_BIP_ERR_0_MSB: 050C	
0x0510	STAT_RX_BIP_ERR_1_LSB: 0510	
0x0514	STAT_RX_BIP_ERR_1_MSB: 0514	
0x0518	STAT_RX_BIP_ERR_2_LSB: 0518	
0x051C	STAT_RX_BIP_ERR_2_MSB: 051C	
0x0520	STAT_RX_BIP_ERR_3_LSB: 0520	
0x0524	STAT_RX_BIP_ERR_3_MSB: 0524	
0x05A8	STAT_RX_FRAMING_ERR_0_LSB: 05A8	
0x05AC	STAT_RX_FRAMING_ERR_0_MSB: 05AC	
0x05B0	STAT_RX_FRAMING_ERR_1_LSB: 05B0	
0x05B4	STAT_RX_FRAMING_ERR_1_MSB: 05B4	
0x05B8	STAT_RX_FRAMING_ERR_2_LSB: 05B8	
0x05BC	STAT_RX_FRAMING_ERR_2_MSB: 05BC	
0x05C0	STAT_RX_FRAMING_ERR_3_LSB: 05C0	
0x05C4	STAT_RX_FRAMING_ERR_3_MSB: 05C4	
0x0660	STAT_RX_BAD_CODE_LSB: 0660	
0x0664	STAT_RX_BAD_CODE_MSB: 0664	
0x0668	STAT_RX_ERROR_LSB: 0668	Only in PCS variant
0x066C	STAT_RX_ERROR_MSB: 066C	Only in PCS variant
0x0670	STAT_RX_RSFEC_CORRECTED_CW_INC_LSB: 0670	
0x0674	STAT_RX_RSFEC_CORRECTED_CW_INC_MSB: 0674	
0x0678	STAT_RX_RSFEC_UNCORRECTED_CW_INC_LSB: 0678	
0x067C	STAT_RX_RSFEC_UNCORRECTED_CW_INC_MSB: 067C	
0x0680	STAT_RX_RSFEC_ERR_COUNT0_INC_LSB: 0680	
0x0684	STAT_RX_RSFEC_ERR_COUNT0_INC_MSB: 0684	
0x0688	STAT_RX_RSFEC_ERR_COUNT1_INC_LSB: 0688	
0x068C	STAT_RX_RSFEC_ERR_COUNT1_INC_LSB: 068C	
0x06A0	STAT_TX_FRAME_ERROR_LSB: 06A0	Only in MAC+PCS variant
0x06A4	STAT_TX_FRAME_ERROR_MSB: 06A4	Only in MAC+PCS variant
0x0700	STAT_TX_TOTAL_PACKETS_LSB: 0700	Only in MAC+PCS variant
0x0704	STAT_TX_TOTAL_PACKETS_MSB: 0704	Only in MAC+PCS variant

Table 29: **Statistics Counters** *(cont'd)*

Hex Address	Name/Link to Description	Notes
0x0708	STAT_TX_TOTAL_GOOD_PACKETS_LSB: 0708	Only in MAC+PCS variant
0x070C	STAT_TX_TOTAL_GOOD_PACKETS_MSB: 070C	Only in MAC+PCS variant
0x0710	STAT_TX_TOTAL_BYTES_LSB: 0710	Only in MAC+PCS variant
0x0714	STAT_TX_TOTAL_BYTES_MSB: 0714	Only in MAC+PCS variant
0x0718	STAT_TX_TOTAL_GOOD_BYTES_LSB: 0718	Only in MAC+PCS variant
0x071C	STAT_TX_TOTAL_GOOD_BYTES_MSB: 071C	Only in MAC+PCS variant
0x0720	STAT_TX_PACKET_64_BYTES_LSB: 0720	Only in MAC+PCS variant
0x0724	STAT_TX_PACKET_64_BYTES_MSB: 0724	Only in MAC+PCS variant
0x0728	STAT_TX_PACKET_65_127_BYTES_LSB: 0728	Only in MAC+PCS variant
0x072C	STAT_TX_PACKET_65_127_BYTES_MSB: 072C	Only in MAC+PCS variant
0x0730	STAT_TX_PACKET_128_255_BYTES_LSB: 0730	Only in MAC+PCS variant
0x0734	STAT_TX_PACKET_128_255_BYTES_MSB: 0734	Only in MAC+PCS variant
0x0738	STAT_TX_PACKET_256_511_BYTES_LSB: 0738	Only in MAC+PCS variant
0x073C	STAT_TX_PACKET_256_511_BYTES_MSB: 073C	Only in MAC+PCS variant
0x0740	STAT_TX_PACKET_512_1023_BYTES_LSB: 0740	Only in MAC+PCS variant
0x0744	STAT_TX_PACKET_512_1023_BYTES_MSB: 0744	Only in MAC+PCS variant
0x0748	STAT_TX_PACKET_1024_1518_BYTES_LSB: 0748	Only in MAC+PCS variant
0x074C	STAT_TX_PACKET_1024_1518_BYTES_MSB: 074C	Only in MAC+PCS variant
0x0750	STAT_TX_PACKET_1519_1522_BYTES_LSB: 0750	Only in MAC+PCS variant
0x0754	STAT_TX_PACKET_1519_1522_BYTES_MSB: 0754	Only in MAC+PCS variant
0x0758	STAT_TX_PACKET_1523_1548_BYTES_LSB: 0758	Only in MAC+PCS variant
0x075C	STAT_TX_PACKET_1523_1548_BYTES_MSB: 075C	Only in MAC+PCS variant
0x0760	STAT_TX_PACKET_1549_2047_BYTES_LSB: 0760	Only in MAC+PCS variant
0x0764	STAT_TX_PACKET_1549_2047_BYTES_MSB: 0764	Only in MAC+PCS variant
0x0768	STAT_TX_PACKET_2048_4095_BYTES_LSB: 0768	Only in MAC+PCS variant
0x076C	STAT_TX_PACKET_2048_4095_BYTES_MSB: 076C	Only in MAC+PCS variant
0x0770	STAT_TX_PACKET_4096_8191_BYTES_LSB: 0770	Only in MAC+PCS variant
0x0774	STAT_TX_PACKET_4096_8191_BYTES_MSB: 0774	Only in MAC+PCS variant
0x0778	STAT_TX_PACKET_8192_9215_BYTES_LSB: 0778	Only in MAC+PCS variant
0x077C	STAT_TX_PACKET_8192_9215_BYTES_MSB: 077C	Only in MAC+PCS variant
0x0780	STAT_TX_PACKET_LARGE_LSB: 0780	Only in MAC+PCS variant
0x0784	STAT_TX_PACKET_LARGE_MSB: 0784	Only in MAC+PCS variant
0x0788	STAT_TX_PACKET_SMALL_LSB: 0788	Only in MAC+PCS variant
0x078C	STAT_TX_PACKET_SMALL_MSB: 078C	Only in MAC+PCS variant
0x07B8	STAT_TX_BAD_FCS_LSB: 07B8	Only in MAC+PCS variant
0x07BC	STAT_TX_BAD_FCS_MSB: 07BC	Only in MAC+PCS variant
0x07D0	STAT_TX_UNICAST_LSB: 07D0	Only in MAC+PCS variant
0x07D4	STAT_TX_UNICAST_MSB: 07D4	Only in MAC+PCS variant
0x07D8	STAT_TX_MULTICAST_LSB: 07D8	Only in MAC+PCS variant

Table 29: **Statistics Counters** *(cont'd)*

Hex Address	Name/Link to Description	Notes
0x07DC	STAT_TX_MULTICAST_MSB: 07DC	Only in MAC+PCS variant
0x07E0	STAT_TX_BROADCAST_LSB: 07E0	Only in MAC+PCS variant
0x07E4	STAT_TX_BROADCAST_MSB: 07E4	Only in MAC+PCS variant
0x07E8	STAT_TX_VLAN_LSB: 07E8	Only in MAC+PCS variant
0x07EC	STAT_TX_VLAN_MSB: 07EC	Only in MAC+PCS variant
0x07F0	STAT_TX_PAUSE_LSB: 07F0	Only in MAC+PCS variant
0x07F4	STAT_TX_PAUSE_MSB: 07F4	Only in MAC+PCS variant
0x07F8	STAT_TX_USER_PAUSE_LSB: 07F8	Only in MAC+PCS variant
0x07FC	STAT_TX_USER_PAUSE_MSB: 07FC	Only in MAC+PCS variant
0x0808	STAT_RX_TOTAL_PACKETS_LSB: 0808	Only in MAC+PCS variant
0x080C	STAT_RX_TOTAL_PACKETS_MSB: 080C	Only in MAC+PCS variant
0x0810	STAT_RX_TOTAL_GOOD_PACKETS_LSB: 0810	Only in MAC+PCS variant
0x0814	STAT_RX_TOTAL_GOOD_PACKETS_MSB: 0814	Only in MAC+PCS variant
0x0818	STAT_RX_TOTAL_BYTES_LSB: 0818	Only in MAC+PCS variant
0x081C	STAT_RX_TOTAL_BYTES_MSB: 081C	Only in MAC+PCS variant
0x0820	STAT_RX_TOTAL_GOOD_BYTES_LSB: 0820	Only in MAC+PCS variant
0x0824	STAT_RX_TOTAL_GOOD_BYTES_MSB: 0824	Only in MAC+PCS variant
0x0828	STAT_RX_PACKET_64_BYTES_LSB: 0828	Only in MAC+PCS variant
0x082C	STAT_RX_PACKET_64_BYTES_MSB: 082C	Only in MAC+PCS variant
0x0830	STAT_RX_PACKET_65_127_BYTES_LSB: 0830	Only in MAC+PCS variant
0x0834	STAT_RX_PACKET_65_127_BYTES_MSB: 0834	Only in MAC+PCS variant
0x0838	STAT_RX_PACKET_128_255_BYTES_LSB: 0838	Only in MAC+PCS variant
0x083C	STAT_RX_PACKET_128_255_BYTES_MSB: 083C	Only in MAC+PCS variant
0x0840	STAT_RX_PACKET_256_511_BYTES_LSB: 0840	Only in MAC+PCS variant
0x0844	STAT_RX_PACKET_256_511_BYTES_MSB: 0844	Only in MAC+PCS variant
0x0848	STAT_RX_PACKET_512_1023_BYTES_LSB: 0848	Only in MAC+PCS variant
0x084C	STAT_RX_PACKET_512_1023_BYTES_MSB: 084C	Only in MAC+PCS variant
0x0850	STAT_RX_PACKET_1024_1518_BYTES_LSB: 0850	Only in MAC+PCS variant
0x0854	STAT_RX_PACKET_1024_1518_BYTES_MSB: 0854	Only in MAC+PCS variant
0x0858	STAT_RX_PACKET_1519_1522_BYTES_LSB: 0858	Only in MAC+PCS variant
0x085C	STAT_RX_PACKET_1519_1522_BYTES_MSB: 085C	Only in MAC+PCS variant
0x0860	STAT_RX_PACKET_1523_1548_BYTES_LSB: 0860	Only in MAC+PCS variant
0x0864	STAT_RX_PACKET_1523_1548_BYTES_MSB: 0864	Only in MAC+PCS variant
0x0868	STAT_RX_PACKET_1549_2047_BYTES_LSB: 0868	Only in MAC+PCS variant
0x086C	STAT_RX_PACKET_1549_2047_BYTES_MSB: 086C	Only in MAC+PCS variant
0x0870	STAT_RX_PACKET_2048_4095_BYTES_LSB: 0870	Only in MAC+PCS variant
0x0874	STAT_RX_PACKET_2048_4095_BYTES_MSB: 0874	Only in MAC+PCS variant
0x0878	STAT_RX_PACKET_4096_8191_BYTES_LSB: 0878	Only in MAC+PCS variant
0x087C	STAT_RX_PACKET_4096_8191_BYTES_MSB: 087C	Only in MAC+PCS variant

Table 29: **Statistics Counters** *(cont'd)*

Hex Address	Name/Link to Description	Notes
0x0880	STAT_RX_PACKET_8192_9215_BYTES_LSB: 0880	Only in MAC+PCS variant
0x0884	STAT_RX_PACKET_8192_9215_BYTES_MSB: 0884	Only in MAC+PCS variant
0x0888	STAT_RX_PACKET_LARGE_LSB: 0888	Only in MAC+PCS variant
0x088C	STAT_RX_PACKET_LARGE_MSB: 088C	Only in MAC+PCS variant
0x0890	STAT_RX_PACKET_SMALL_LSB: 0890	Only in MAC+PCS variant
0x0894	STAT_RX_PACKET_SMALL_MSB: 0894	Only in MAC+PCS variant
0x0898	STAT_RX_UNDERSIZE_LSB: 0898	Only in MAC+PCS variant
0x089C	STAT_RX_UNDERSIZE_MSB: 089C	Only in MAC+PCS variant
0x08A0	STAT_RX_FRAGMENT_LSB: 08A0	Only in MAC+PCS variant
0x08A4	STAT_RX_FRAGMENT_MSB: 08A4	Only in MAC+PCS variant
0x08A8	STAT_RX_OVERSIZE_LSB: 08A8	Only in MAC+PCS variant
0x08AC	STAT_RX_OVERSIZE_MSB: 08AC	Only in MAC+PCS variant
0x08B0	STAT_RX_TOOLONG_LSB: 08B0	Only in MAC+PCS variant
0x08B4	STAT_RX_TOOLONG_MSB: 08B4	Only in MAC+PCS variant
0x08B8	STAT_RX_JABBER_LSB: 08B8	Only in MAC+PCS variant
0x08BC	STAT_RX_JABBER_MSB: 08BC	Only in MAC+PCS variant
0x08C0	STAT_RX_BAD_FCS_LSB: 08C0	Only in MAC+PCS variant
0x08C4	STAT_RX_BAD_FCS_MSB: 08C4	Only in MAC+PCS variant
0x08C8	STAT_RX_PACKET_BAD_FCS_LSB: 08C8	Only in MAC+PCS variant
0x08CC	STAT_RX_PACKET_BAD_FCS_MSB: 08CC	Only in MAC+PCS variant
0x08D0	STAT_RX_STOMPED_FCS_LSB: 08D0	Only in MAC+PCS variant
0x08D4	STAT_RX_STOMPED_FCS_MSB: 08D4	Only in MAC+PCS variant
0x08D8	STAT_RX_UNICAST_LSB: 08D8	Only in MAC+PCS variant
0x08DC	STAT_RX_UNICAST_MSB: 08DC	Only in MAC+PCS variant
0x08E0	STAT_RX_MULTICAST_LSB: 08E0	Only in MAC+PCS variant
0x08E4	STAT_RX_MULTICAST_MSB: 08E4	Only in MAC+PCS variant
0x08E8	STAT_RX_BROADCAST_LSB: 08E8	Only in MAC+PCS variant
0x08EC	STAT_RX_BROADCAST_MSB: 08EC	Only in MAC+PCS variant
0x08F0	STAT_RX_VLAN_LSB: 08F0	Only in MAC+PCS variant
0x08F4	STAT_RX_VLAN_MSB: 08F4	Only in MAC+PCS variant
0x08F8	STAT_RX_PAUSE_LSB: 08F8	Only in MAC+PCS variant
0x08FC	STAT_RX_PAUSE_MSB: 08FC	Only in MAC+PCS variant
0x0900	STAT_RX_USER_PAUSE_LSB: 0900	Only in MAC+PCS variant
0x0904	STAT_RX_USER_PAUSE_MSB: 0904	Only in MAC+PCS variant
0x0908	STAT_RX_INRANGEERR_LSB: 0908	Only in MAC+PCS variant
0x090C	STAT_RX_INRANGEERR_MSB: 090C	Only in MAC+PCS variant
0x0910	STAT_RX_TRUNCATED_LSB: 0910	Only in MAC+PCS variant
0x0914	STAT_RX_TRUNCATED_MSB: 0914	Only in MAC+PCS variant
0x0918	STAT_RX_TEST_PATTERN_MISMATCH_LSB: 0918	Only in MAC+PCS variant

Table 29: **Statistics Counters** *(cont'd)*

Hex Address	Name/Link to Description	Notes
0x091C	STAT_RX_TEST_PATTERN_MISMATCH_MSB: 091C	Only in MAC+PCS variant
0x0920	STAT_FEC_INC_CORRECT_COUNT_LSB: 0920	
0x0924	STAT_FEC_INC_CORRECT_COUNT_MSB: 0924	
0x0928	STAT_FEC_INC_CANT_CORRECT_COUNT_LSB: 0928	
0x092C	STAT_FEC_INC_CANT_CORRECT_COUNT_MSB: 092C	

Register Definitions

Configuration Registers

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. Reserved fields in the configuration registers do not accept any written value, and always return a 0 when read. Registers or bit fields within registers can be accessed for read-write (RW), write-only (WO), or read-only (RO). Default values shown are decimal values and take effect after `s_axi_aresetn`.

A description of each signal is found in the port list section of this document.

GT_RESET_REG: 0000

Table 30: **GT_RESET_REG: 0000**

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

RESET_REG: 0004

This is a clear on write register. This is available when the Include ANLT Logic option selected in the Configuration tab.

Table 31: **RESET_REG: 0004**

Bits	Default	Type	Signal
1:0	0	RW	rx_serdes_reset
28	0	RW	ctl_an_reset
29	0	RW	tx_serdes_reset

Table 31: **RESET_REG: 0004** *(cont'd)*

Bits	Default	Type	Signal
30	0	RW	rx_reset
31	0	RW	tx_reset

MODE_REG: 0008

Table 32: **MODE_REG: 0008**

Bits	Default	Type	Signal
0	1	RW	en_wr_slverr_indication
1	1	RW	en_rd_slverr_indication
30	1	RW	tick_reg_mode_sel
31	0	RW	GT near-end PMA loopback

CONFIGURATION_TX_REG1: 000C

Table 33: **CONFIGURATION_TX_REG1: 000C**

Bits	Default	Type	Signal
0	1	RW	ctl_tx_enable[1]
1	1	RW	ctl_tx_fcs_ins_enable[1]
2	0	RW	ctl_tx_ignore_fcs[1]
3	0	RW	ctl_tx_send_lfi[1]
4	0	RW	ctl_tx_send_rfi[1]
5	0	RW	ctl_tx_send_idle[1]
13:10	12	RW	ctl_tx_ipg_value[1]
14	0	RW	ctl_tx_test_pattern
18	0	RW	ctl_tx_custom_preamble_enable[1]

Notes:

1. Only in MAC+PCS variant

CONFIGURATION_RX_REG1: 0014

Table 34: **CONFIGURATION_RX_REG1: 0014**

Bits	Default	Type	Signal
0	1	RW	ctl_rx_enable[1]
1	1	RW	ctl_rx_delete_fcs[1]
2	0	RW	ctl_rx_ignore_fcs[1]
3	0	RW	ctl_rx_process_lfi[1]
4	1	RW	ctl_rx_check_sfd[1]

Table 34: **CONFIGURATION_RX_REG1: 0014** *(cont'd)*

Bits	Default	Type	Signal
5	1	RW	ctl_rx_check_preamble[1]
6	0	RW	ctl_rx_force_resync[1]
7	0	RW	ctl_rx_test_pattern
18	0	RW	ctl_rx_custom_preamble_enable[1]

Notes:
1. Only in MAC+PCS variant

CONFIGURATION_RX_MTU: 0018

Table 35: **CONFIGURATION_RX_MTU: 0018**

Bits	Default	Type	Signal
7:0	64	RW	ctl_rx_min_packet_len
30:16	9600	RW	ctl_rx_max_packet_len

CONFIGURATION_VL_LENGTH_REG: 001C

Table 36: **CONFIGURATION_VL_LENGTH_REG: 001C**

Bits	Default	Type	Signal
15:0	16383	RW	ctl_tx_vl_length_minus1 ***Note***: When RS-FEC is enabled in the 50G core configuration, this value will be set to 20479.
31:16	16383	RW	ctl_rx_vl_length_minus1 ***Note***: When RS-FEC is enabled in the 50G core configuration, this value will be set to 20479.

TICK_REG: 0020

Table 37: **TICK_REG: 0020**

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

CONFIGURATION_REVISION_REG: 0024

Table 38: **CONFIGURATION_REVISION_REG: 0024**

Bits	Default	Type	Signal
7:0	2	RO	major_rev
15:8	5	RO	minor_rev
31:24	0	RO	patch_rev

CONFIGURATION_1588_REG: 0038

Table 39: **CONFIGURATION_1588_REG: 0038**

Bits	Default	Type	Signal
0	0	RW	ctl_tx_ptp_1step_enable
1	0	RW	ctl_tx_ptp_vlane_adjust_mode
2	0	RW	ctl_ptp_transpclk_mode
26:16	0	RW	ctl_tx_ptp_latency_adjust

CONFIGURATION_TX_FLOW_CONTROL_REG1: 0040

Table 40: **CONFIGURATION_TX_FLOW_CONTROL_REG1: 0040**

Bits	Default	Type	Signal
8:0	0	RW	ctl_tx_pause_enable

CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG1: 0044

Table 41: **CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG1: 0044**

Bits	Default	Type	Signal
15:0	00	RW	ctl_tx_pause_refresh_timer0
31:16	0	RW	ctl_tx_pause_refresh_timer1

CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG2: 0048

Table 42: **CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG2: 0048**

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

CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG3: 004C

Table 43: **CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG2: 004C**

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

CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG4: 0050

Table 44: **CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG2: 0050**

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

CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG5: 0054

Table 45: **CONFIGURATION_TX_FLOW_CONTROL_REFRESH_REG2: 0054**

Bits	Default	Type	Signal
15:0	0	RW	ctl_tx_pause_refresh_timer8

CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG1: 0058

Table 46: **CONFIGURATION_TX_FLOW_CONTRO_QUANTA_REG1: 0058**

Bits	Default	Type	Signal
15:0	00	RW	ctl_tx_pause_quanta0
31:16	0	RW	ctl_tx_pause_quanta1

CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG2: 005C

Table 47: **CONFIGURATION_TX_FLOW_CONTRO_QUANTA_REG1: 005C**

Bits	Default	Type	Signal
15:0	00	RW	ctl_tx_pause_quanta2
31:16	0	RW	ctl_tx_pause_quanta3

CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG3: 0060

Table 48: **CONFIGURATION_TX_FLOW_CONTRO_QUANTA_REG1: 0060**

Bits	Default	Type	Signal
15:0	00	RW	ctl_tx_pause_quanta4
31:16	0	RW	ctl_tx_pause_quanta5

CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG4: 0064

Table 49: **CONFIGURATION_TX_FLOW_CONTRO_QUANTA_REG1: 0064**

Bits	Default	Type	Signal
15:0	00	RW	ctl_tx_pause_quanta6
31:16	0	RW	ctl_tx_pause_quanta7

CONFIGURATION_TX_FLOW_CONTROL_QUANTA_REG5: 0068

Table 50: **CONFIGURATION_TX_FLOW_CONTRO_QUANTA_REG1: 0068**

Bits	Default	Type	Signal
15:0	00	RW	ctl_tx_pause_quanta8

CONFIGURATION_TX_FLOW_CONTROL_PPP_ETYPE_OP_REG: 006C

Table 51: **CONFIGURATION_TX_FLOW_CONTROL_PPP_ETYPE_OP_REG: 006C**

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

CONFIGURATION_TX_FLOW_CONTROL_GPP_ETYPE_OP_REG: 0070

Table 52: **CONFIGURATION_TX_FLOW_CONTROL_PPP_ETYPE_OP_REG: 007C**

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

CONFIGURATION_TX_FLOW_CONTROL_GPP_DA_REG_LSB: 0074

Table 53: **CONFIGURATION_TX_FLOW_CONTROL_PPP_ETYPE_OP_REG: 0074**

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

CONFIGURATION_TX_FLOW_CONTROL_GPP_DA_REG_MSB: 0078

Table 54: **CONFIGURATION_TX_FLOW_CONTROL_PPP_ETYPE_OP_REG: 0078**

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

CONFIGURATION_TX_FLOW_CONTROL_GPP_SA_REG_LSB: 007C

Table 55: **CONFIGURATION_TX_FLOW_CONTROL_PPP_ETYPE_OP_REG: 007C**

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

CONFIGURATION_TX_FLOW_CONTROL_GPP_SA_REG_MSB: 0080

Table 56: **CONFIGURATION_TX_FLOW_CONTROL_PPP_ETYPE_OP_REG: 0080**

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

CONFIGURATION_TX_FLOW_CONTROL_PPP_DA_REG_LSB: 0084

Table 57: **CONFIGURATION_TX_FLOW_CONTROL_PPP_DA_REG_LSB: 0084**

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

CONFIGURATION_TX_FLOW_CONTROL_PPP_DA_REG_MSB: 0088

Table 58: **CONFIGURATION_TX_FLOW_CONTROL_PPP_DA_REG_MSB: 0088**

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

CONFIGURATION_TX_FLOW_CONTROL_PPP_SA_REG_LSB: 008C

Table 59: **CONFIGURATION_TX_FLOW_CONTROL_PPP_DA_REG_LSB: 008C**

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

CONFIGURATION_TX_FLOW_CONTROL_PPP_SA_REG_MSB: 0090

Table 60: **CONFIGURATION_TX_FLOW_CONTROL_PPP_DA_REG_MSB: 0090**

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

CONFIGURATION_RX_FLOW_CONTROL_REG1: 0094

Table 61: **CONFIGURATION_RX_FLOW_CONTROL_REG1: 0094**

Bits	Default	Type	Signal
8:0	0	RW	ctl_rx_pause_enable
9	0	RW	ctl_rx_forward_control
10	0	RW	ctl_rx_enable_gcp
11	0	RW	ctl_rx_enable_pcp
12	0	RW	ctl_rx_enable_gpp
13	0	RW	ctl_rx_enable_ppp
14	0	RW	ctl_rx_check_ack

CONFIGURATION_RX_FLOW_CONTROL_REG2: 0098

Table 62: **CONFIGURATION_RX_FLOW_CONTROL_REG2: 0098**

Bits	Default	Type	Signal
0	0	RW	ctl_rx_check_mcast_gcp
1	0	RW	ctl_rx_check_ucast_gcp
2	0	RW	ctl_rx_check_sa_gcp
3	0	RW	ctl_rx_check_etype_gcp
4	0	RW	ctl_rx_check_opcode_gcp
5	0	RW	ctl_rx_check_mcast_pcp
6	0	RW	ctl_rx_check_ucast_pcp
7	0	RW	ctl_rx_check_sa_pcp
8	0	RW	ctl_rx_check_etype_pcp
9	0	RW	ctl_rx_check_opcode_pcp
10	0	RW	ctl_rx_check_mcast_gpp

Table 62: **CONFIGURATION_RX_FLOW_CONTROL_REG2: 0098** *(cont'd)*

Bits	Default	Type	Signal
11	0	RW	ctl_rx_check_ucast_gpp
12	0	RW	ctl_rx_check_sa_gpp
13	0	RW	ctl_rx_check_etype_gpp
14	0	RW	ctl_rx_check_opcode_gpp
15	0	RW	ctl_rx_check_mcast_ppp
16	0	RW	ctl_rx_check_ucast_ppp
17	0	RW	ctl_rx_check_sa_ppp
18	0	RW	ctl_rx_check_etype_ppp
19	0	RW	ctl_rx_check_opcode_ppp

CONFIGURATION_RX_FLOW_CONTROL_PPP_ETYPE_OP_REG: 009C

Table 63: **CONFIGURATION_RX_FLOW_CONTROL_PPP_ETYPE_OP_REG: 009C**

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

CONFIGURATION_RX_FLOW_CONTROL_GPP_ETYPE_OP_REG: 00A0

Table 64: **CONFIGURATION_RX_FLOW_CONTROL_PPP_ETYPE_OP_REG: 00A0**

Bits	Default	Type	Signal
15:0	34824	RW	ctl_rx_etype_gpp
31:16	1	RW	ctl_rx_opcode_gpp

CONFIGURATION_RX_FLOW_CONTROL_GCP_PCP_TYPE_REG: 00A4

Table 65: **CONFIGURATION_RX_FLOW_CONTROL_GCP_PCP_TYPE_REG: 00A4**

Bits	Default	Type	Signal
15:0	34824	RW	ctl_rx_etype_gcp
31:16	34824	RW	ctl_rx_etypr_pcp

CONFIGURATION_RX_FLOW_CONTROL_PCP_OP_REG: 00A8

Table 66: **CONFIGURATION_RX_FLOW_CONTROL_GCP_PCP_OP_REG: 00A8**

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

CONFIGURATION_RX_FLOW_CONTROL_GCP_OP_REG: 00AC

Table 67: **CONFIGURATION_RX_FLOW_CONTROL_GCP_OP_REG: 00AC**

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

CONFIGURATION_RX_FLOW_CONTROL_DA_REG1_LSB: 00B0

Table 68: **CONFIGURATION_RX_FLOW_CONTROL_GCP_PCP_OP_REG: 00A8**

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

CONFIGURATION_RX_FLOW_CONTROL_DA_REG1_MSB: 00B4

Table 69: **CONFIGURATION_RX_FLOW_CONTROL_DA_REG1_MSB: 00B4**

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

CONFIGURATION_RX_FLOW_CONTROL_DA_REG2_LSB: 00B8

Table 70: **CONFIGURATION_RX_FLOW_CONTROL_DA_REG2_LSB: 00B8**

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

CONFIGURATION_RX_FLOW_CONTROL_DA_REG2_MSB: 00BC

Table 71: **CONFIGURATION_RX_FLOW_CONTROL_DA_REG2_MSB: 00BC**

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

CONFIGURATION_RX_FLOW_CONTROL_SA_REG1_LSB: 00C0

Table 72: **CONFIGURATION_RX_FLOW_CONTROL_SA_REG1_LSB: 00C0**

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

CONFIGURATION_RX_FLOW_CONTROL_SA_REG1_MSB: 00C4

Table 73: **CONFIGURATION_RX_FLOW_CONTROL_SA_REG1_MSB: 00C4**

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

CONFIGURATION_RSFEC_REG: 00D0

Table 74: **CONFIGURATION_RSFEC_REG: 00D0**

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

CONFIGURATION_FEC_REG: 00D4

Table 75: **CONFIGURATION_FEC_REG: 00D4**

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

CONFIGURATION_AN_CONTROL_REG1: 00E0

Table 76: **CONFIGURATION_AN_CONTROL_REG1: 00E0**

Bits	Default	Type	Signal
0	0	RW	ctl_autoneg_enable
1	1	RW	ctl_autoneg_bypass[1]
9:2	0	RW	ctl_an_nonce_seed
10	0	RW	ctl_an_pseudo_sel
11	0	RW	ctl_restart_negotiation
12	0	RW	ctl_an_local_fault

Notes:

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

CONFIGURATION_AN_CONTROL_REG2: 00E4

Table 77: **CONFIGURATION_AN_CONTROL_REG2: 00E4**

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

CONFIGURATION_AN_ABILITY: 00F8

Table 78: **CONFIGURATION_AN_ABILITY: 00F8**

Bits	Default	Type	Signal
0	0	RW	ctl_an_ability_1000base_kx
1	0	RW	ctl_an_ability_10gbase_kx4
2	0	RW	ctl_an_ability_10gbase_kr
3	0	RW	ctl_an_ability_40gbase_kr4
4	0	RW	ctl_an_ability_40gbase_cr4
5	0	RW	ctl_an_ability_100gbase_cr10
6	0	RW	ctl_an_ability_100gbase_kp4
7	0	RW	ctl_an_ability_100gbase_kr4
8	0	RW	ctl_an_ability_100gbase_cr4
9	0	RW	ctl_an_ability_25gbase_krcr_s
10	0	RW	ctl_an_ability_25gbase_krcr
11	0	RW	ctl_an_ability_2_5gbase_kx
12	0	RW	ctl_an_ability_5gbase_kr
13	0	RW	ctl_an_ability_50gbase_krcr
14	0	RW	ctl_an_ability_100gbase_kr2cr2
15	0	RW	ctl_an_ability_200gbase_kr4cr4
16	0	RW	ctl_an_ability_25gbase_kr1
17	0	RW	ctl_an_ability_25gbase_cr1
18	0	RW	ctl_an_ability_50gbase_kr2
19	0	RW	ctl_an_ability_50gbase_cr2

CONFIGURATION_LT_CONTROL_REG1: 0100

Table 79: **CONFIGURATION_LT_CONTROL_REG1: 0100**

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

CONFIGURATION_LT_TRAINED_REG: 0104

Table 80: **CONFIGURATION_LT_TRAINED_REG1: 0104**

Bits	Default	Type	Signal
3:0	0	RW	ctl_lt_rx_trained

CONFIGURATION_LT_PRESET_REG: 0108

Table 81: **CONFIGURATION_LT_TRAINED_REG1: 0104**

Bits	Default	Type	Signal
3:0	0	RW	ctl_lt_preset_to_tx

CONFIGURATION_LT_INIT_REG: 010C

Table 82: **CONFIGURATION_LT_INIT_REG: 010C**

Bits	Default	Type	Signal
3:0	0	RW	ctl_lt_initialize_to_tx

CONFIGURATION_LT_SEED_REG0: 0110

Table 83: **CONFIGURATION_LT_SEED_REG0: 0110**

Bits	Default	Type	Signal
10:0	0	RW	ctl_lt_pseudo_seed0
26:16	0	RW	ctl_lt_pseudo_seed1

CONFIGURATION_LT_SEED_REG1: 0114

Table 84: **CONFIGURATION_LT_SEED_REG1: 0114**

Bits	Default	Type	Signal
10:0	0	RW	ctl_lt_pseudo_seed2
26:16	0	RW	ctl_lt_pseudo_seed3

CONFIGURATION_LT_COEFFICIENT_REG0: 0130

Table 85: **CONFIGURATION_LT_COEFFICIENT_REG0: 0130**

Bits	Default	Type	Signal
1:0	0	RW	ctl_lt_k_p1_to_tx0
3:2	0	RW	ctl_lt_k0_to_tx0
5:4	0	RW	ctl_lt_k_m1_to_tx0
7:6	0	RW	ctl_lt_stat_p1_to_tx0
9:8	0	RW	ctl_lt_stat0_to_tx0
11:10	0	RW	ctl_lt_stat_m1_to_tx0
17:16	0	RW	ctl_lt_k_p1_to_tx1
19:18	0	RW	ctl_lt_k0_to_tx1
21:20	0	RW	ctl_lt_k_m1_to_tx1
23:22	0	RW	ctl_lt_stat_p1_to_tx1
25:24	0	RW	ctl_lt_stat0_to_tx1
27:26	0	RW	ctl_lt_stat_m1_to_tx1

CONFIGURATION_LT_COEFFICIENT_REG1: 0134

Table 86: **CONFIGURATION_LT_COEFFICIENT_REG1: 0134**

Bits	Default	Type	Signal
1:0	0	RW	ctl_lt_k_p1_to_tx2
3:2	0	RW	ctl_lt_k0_to_tx2
5:4	0	RW	ctl_lt_k_m1_to_tx2
7:6	0	RW	ctl_lt_stat_p1_to_tx2
9:8	0	RW	ctl_lt_stat0_to_tx2
11:10	0	RW	ctl_lt_stat_m1_to_tx2
17:16	0	RW	ctl_lt_k_p1_to_tx3
19:18	0	RW	ctl_lt_k0_to_tx3
21:20	0	RW	ctl_lt_k_m1_to_tx3
23:22	0	RW	ctl_lt_stat_p1_to_tx3
25:24	0	RW	ctl_lt_stat0_to_tx3
27:26	0	RW	ctl_lt_stat_m1_to_tx3

USER_REG_0: 0138

Table 87: **USER_REG_0: 0138**

Bits	Default	Type	Signal
31:0	0	RW	user_reg0

Status Registers

The following tables describe the status registers for this Ethernet IP core.

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

STAT_TX_STATUS_REG1: 0400

Table 88: **SWITCH_CORE_SPEED_REG1: 0400**

Bits	Default	Type	Signal
0	0	RO LH	stat_tx_local_fault

For Runtime Switch mode only. A write 1 enables the mode switch between 40G and 50G. This is a clear on the write register. This is an input to the trans debug module that performs the GT DRP operations.

STAT_RX_STATUS_REG1: 0404

Table 89: **STAT_RX_STATUS_REG1: 0404**

Bits	Default	Type	Signal
0	1	RO LL	stat_rx_status
1	1	RO LL	stat_rx_aligned
2	0	RO LH	stat_rx_misaligned
3	0	RO LH	stat_rx_aligned_err
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

STAT_STATUS_REG1: 0408

Table 90: **STAT_STATUS_REG1: 0408**

Bits	Default	Type	Signal
0	0	RO LL	stat_rx_status[3]

Table 90: **STAT_STATUS_REG1: 0408** *(cont'd)*

Bits	Default	Type	Signal
1	0	RO LL	stat_rx_aligned[2]
2	0	RO LH	stat_rx_misaligned[2]
3	0	RO LH	stat_rx_aligned_err[1]
4	0	RO LH	stat_rx_hi_ber[1]
5	0	RO LH	stat_rx_remote_fault[3]
16	0	RO LH	stat_rx_local_fault

Notes:
1. Only in MAC+PCS variant
2. Only in 256-bit non-segmented 40 Gb/s MAC+PCS variant
3. Only in PCS only variant

STAT_RX_BLOCK_LOCK_REG: 040C

Table 91: **STAT_RX_BLOCK_LOCK_REG: 040C**

Bits	Default	Type	Signal
3:0	1	RO LL	stat_rx_block_lock

STAT_RX_LANE_SYNC_REG: 0410

Table 92: **STAT_RX_LANE_SYNC_REG: 0410**

Bits	Default	Type	Signal
3:0	1	RO LL	stat_rx_synced

STAT_RX_LANE_SYNC_ERR_REG: 0414

Table 93: **STAT_RX_LANE_SYNC_ERR: 0414**

Bits	Default	Type	Signal
3:0	0	RO LL	stat_rx_synced_err

STAT_RX_AM_ERR_REG: 0418

Table 94: **STAT_RX_AM_ERR_REG: 0418**

Bits	Default	Type	Signal
3:0	0	RO LL	stat_rx_mf_err

STAT_RX_AM_LEN_ERR_REG: 041C

Table 95: **STAT_RX_AM_LEN_ERR_REG: 0418**

Bits	Default	Type	Signal
3:0	0	RO LL	stat_rx_mf_len_err

STAT_RX_AM_REPEAT_ERR_REG: 0420

Table 96: **STAT_RX_AM_REPEAT_ERR_REG: 0420**

Bits	Default	Type	Signal
3:0	0	RO LH	stat_rx_mf_repeat_err

STAT_RX_LANE_DEMUXED: 0424

Table 97: **STAT_RX_LANE_DEMUXED: 0440**

Bits	Default	Type	Signal
3:0	0	RO	stat_rx_vl_demuxed

STAT_RX_PCS_LANE_NUM_REG1: 0428

Table 98: **STAT_RX_PCS_LANE_NUM_REG1: 0428**

Bits	Default	Type	Signal
1:0	0	RO	stat_rx_vl_number_0
6:5	0	RO	stat_rx_vl_number_1
11:10	0	RO	stat_rx_vl_number_2
16:15	0	RO	stat_rx_vl_number_3

STAT_RX_RSFEC_STATUS_REG: 043C

Table 99: **STAT_RX_RSFEC_STATUS_REG: 043C**

Bits	Default	Type	Signal
0	1	RO LH	stat_rx_rsfec_lane_alignment_status
1	0	RO LH	stat_rx_rsfec_lane_hi_ser
4	0	RO	stat_rx_rsfec_am_lock0
5	0	RO	stat_rx_rsfec_am_lock1
9:8	0	RO	stat_rx_rsfec_lane_mapping

STAT_RX_RSFEC_STATUS_REG: 0440

Table 100: **STAT_RX_RSFEC_STATUS_REG: 0440**

Bits	Default	Type	Signal
13:0	0	RO	stat_rx_rsfec_lane_fill_0
29:16	0	RO	stat_rx_rsfec_lane_fill_1

STAT_RX_FEC_STATUS_REG: 0448

Table 101: **STAT_RX_RSFEC_STATUS_REG: 0448**

Bits	Default	Type	Signal
3:0	1	RO LL	stat_fec_rx_lock
19:16	1	RO LL	stat_fec_lock_error

STAT_TX_RSFEC_STATUS_REG: 044C

Table 102: **STAT_RX_RSFEC_STATUS_REG: 044C**

Bits	Default	Type	Signal
0	1	RO LL	stat_tx_rsfec_lane_alignment_status
1	1	RO LL	stat_tx_rsfec_block_lock

STAT_TX_FLOW_CONTROL_REG1: 0450

Table 103: **STAT_TX_FLOW_CONTROL_REG1: 0450**

Bits	Default	Type	Signal
8:0	0	RO LH	stat_tx_pause_valid

STAT_RX_FLOW_CONTROL_REG1: 0454

Table 104: **STAT_TX_FLOW_CONTROL_REG1: 0450**

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

STAT_AN_STATUS: 0458

Table 105: **STAT_AN_STATUS: 0458**

Bits	Default	Type	Signal
0	0	RO	stat_an_fec_enable
1	0	RO	stat_an_rs_fec_enable
2	0	RO	stat_an_autoneg_complete
3	0	RO	stat_an_parallel_detection_fault
4	0	RO	stat_an_tx_pause_enable
5	0	RO	stat_an_rx_pause_enable
6	0	RO LH	stat_an_lp_ability_valid
7	0	RO	stat_an_lp_autoneg_able
8	0	RO	stat_an_lp_pause
9	0	RO	stat_an_lp_asm_dir
10	0	RO	stat_an_lp_rf
11	0	RO	stat_an_lp_fec_10g_ability
12	0	RO	stat_an_lp_fec_10g_request
13	0	RO LH	stat_an_lp_extended_ability_valid
17:14	0	RO	stat_an_lp_ability_extended_fec
18	0	RO	stat_an_lp_fec_25g_rs_request
19	0	RO	stat_an_lp_fec_25g_baser_request

STAT_AN_ABILITY: 045C

Table 106: **STAT_AN_ABILITY: 045C**

Bits	Default	Type	Signal
0	0	RO	stat_an_lp_ability_1000base_kx
1	0	RO	stat_an_lp_ability_10gbase_kx4
2	0	RO	stat_an_lp_ability_10gbase_kr
3	0	RO	stat_an_lp_ability_40gbase_kr4
4	0	RO	stat_an_lp_ability_40gbase_cr4
5	0	RO	stat_an_lp_ability_100gbase_cr10
6	0	RO	stat_an_lp_ability_100gbase_kp4
7	0	RO	stat_an_lp_ability_100gbase_kr4
8	0	RO	stat_an_lp_ability_100gbase_cr4
9	0	RO	stat_an_lp_ability_25gbase_krcr_s
10	0	RO	stat_an_lp_ability_25gbase_krcr
11	0	RO	stat_an_lp_ability_2_5gbase_kx
12	0	RO	stat_an_lp_ability_5gbase_kr
13	0	RO	stat_an_lp_ability_50gbase_krcr

Table 106: **STAT_AN_ABILITY: 045C** *(cont'd)*

Bits	Default	Type	Signal
14	0	RO	stat_an_lp_ability_100gbase_kr2cr2
15	0	RO	stat_an_lp_ability_200gbase_kr4cr4
16	0	RO	stat_an_lp_ability_25gbase_kr1
17	0	RO	stat_an_lp_ability_25gbase_cr1
18	0	RO	stat_an_lp_ability_50gbase_kr2
19	0	RO	stat_an_lp_ability_50gbase_cr2

STAT_AN_LINK_CTL: 0460

Table 107: **STAT_AN_LINK_CTL: 0460**

Bits	Default	Type	Signal
1:0	0	RO	stat_an_link_cntl_1000base_kx
3:2	0	RO	stat_an_link_cntl_10gbase_kx4
5:4	0	RO	stat_an_link_cntl_10gbase_kr
7:6	0	RO	stat_an_link_cntl_40gbase_kr4
9:8	0	RO	stat_an_link_cntl_40gbase_cr4
11:10	0	RO	stat_an_link_cntl_100gbase_cr10
13:12	0	RO	stat_an_link_cntl_100gbase_kp4
15:14	0	RO	stat_an_link_cntl_100gbase_kr4
17:16	0	RO	stat_an_link_cntl_100gbase_cr4
19:18	0	RO	stat_an_link_cntl_25gbase_krcr_s
21:20	0	RO	stat_an_link_cntl_25gbase_krcr
23:22	0	RO	stat_an_link_cntl_2_5gbase_kx
25:24	0	RO	stat_an_link_cntl_5gbase_kr
27:26	0	RO	stat_an_link_cntl_50gbase_krcr
29:28	0	RO	stat_an_link_cntl_100gbase_kr2cr2
31:30	0	RO	stat_an_link_cntl_200gbase_kr4cr4

STAT_AN_LINK_CTL2: 09F0

Table 108: **STAT_AN_LINK_CTL2: 09F0**

Bits	Default	Type	Signal
1:0	0	RO	stat_an_lp_ability_25gbase_kr1
3:2	0	RO	stat_an_link_cntl_25gbase_cr1
5:4	0	RO	stat_an_link_cntl_50gbase_kr2
7:6	0	RO	stat_an_link_cntl_50gbase_cr2

STAT_LT_STATUS_REG1: 0464

Table 109: **STAT_LT_STATUS_REG1: 0464**

Bits	Default	Type	Signal
3:0	0	RO	stat_lt_initialize_from_rx
19:16	0	RO	stat_lt_preset_from_rx

STAT_LT_STATUS_REG2: 0468

Table 110: **STAT_LT_STATUS_REG2: 0468**

Bits	Default	Type	Signal
3:0	0	RO	stat_lt_training
19:16	0	RO	stat_lt_frame_lock

STAT_LT_STATUS_REG3: 046C

Table 111: **STAT_LT_STATUS_REG3: 046C**

Bits	Default	Type	Signal
3:0	0	RO	stat_lt_signal_detect
19:16	0	RO	stat_lt_training_fail

STAT_LT_STATUS_REG4: 0470

Table 112: **STAT_LT_STATUS_REG3: 0470**

Bits	Default	Type	Signal
3:0	0	RO LH	stat_lt_rx_sof

STAT_LT_COEFFICIENT0_REG: 0474

Table 113: **STAT_LT_COEFFICIENT0_REG: 0474**

Bits	Default	Type	Signal
1:0	0	RO	stat_lt_k_p1_from_rx0
3:2	0	RO	stat_lt_k0_from_rx0
5:4	0	RO	stat_lt_k_m1_from_rx0
7:6	0	RO	stat_lt_stat_p1_from_rx0
9:8	0	RO	stat_lt_stat0_from_rx0
11:10	0	RO	stat_lt_stat_m1_from_rx0
17:16	0	RO	stat_lt_k_p1_from_rx1
19:18	0	RO	stat_lt_k0_from_rx1

Table 113: **STAT_LT_COEFFICIENT0_REG: 0474** *(cont'd)*

Bits	Default	Type	Signal
21:20	0	RO	stat_lt_k_m1_from_rx1
23:22	0	RO	stat_lt_stat_p1_from_rx1
25:24	0	RO	stat_lt_stat0_from_rx1
27:26	0	RO	stat_lt_stat_m1_from_rx1

STAT_LT_COEFFICIENT1_REG: 0478

Table 114: **STAT_LT_COEFFICIENT1_REG: 0478**

Bits	Default	Type	Signal
1:0	0	RO	stat_lt_k_p1_from_rx2
3:2	0	RO	stat_lt_k0_from_rx2
5:4	0	RO	stat_lt_k_m1_from_rx2
7:6	0	RO	stat_lt_stat_p1_from_rx2
9:8	0	RO	stat_lt_stat0_from_rx2
11:10	0	RO	stat_lt_stat_m1_from_rx2
17:16	0	RO	stat_lt_k_p1_from_rx3
19:18	0	RO	stat_lt_k0_from_rx3
21:20	0	RO	stat_lt_k_m1_from_rx3
23:22	0	RO	stat_lt_stat_p1_from_rx3
25:24	0	RO	stat_lt_stat0_from_rx3
27:26	0	RO	stat_lt_stat_m1_from_rx3

STAT_CORE_SPEED_REG: 047C

Table 115: **STAT_CORE_SPEED_REG: 047C**

Bits	Default	Type	Signal
0	GUI Configured	RO	stat_core_speed
1	GUI Configured	RO	runtime_switchable[1]

Notes:

1. This register is available only for the 128-bit straddled AXI4-Stream data path interface. Bits 1:0 are defined as:
   00 Standalone 50G
   01 Standalone 40G
   10 Runtime Switchable 50G
   11 Runtime Switchable 40G

STAT_GT_WIZ_REG: 04A0

Table 116: **STAT_GT_WIZ_REG**

Bits	Default	Type	Signal
0	0	RO	gtwiz_reset_tx_done
1	0	RO	gtwiz_reset_rx_done

Statistics Counters

Counters are 48 bits and require two 32-bit address spaces. The following tables provide the first address for each counter corresponding to the 32 LSBs and the next address contains the remaining 16 bits of MSBs in bits [15:0].The default value for all counters is 0.

Counters are cleared when read by `tick_reg` (or `pm_tick` if so selected), but the register retains its value.

Each counter saturates at FFFFFFFFFFFF (hex).

STATUS_CYCLE_COUNT_LSB: 0500

Table 117: **STATUS_CYCLE_COUNT_LSB: 0500**

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

STATUS_CYCLE_COUNT_MSB: 0504

Table 118: **STATUS_CYCLE_COUNT_MSB: 0504**

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

STAT_RX_BIP_ERR_0_LSB: 0508

Table 119: **STAT_RX_BIP_ERR_0_LSB: 0508**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_bip_err_0_count[31:0]

STAT_RX_BIP_ERR_0_MSB: 050C

Table 120: **STAT_RX_BIP_ERR_0_MSB: 050C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_bip_err_0_count[48-1:32]

STAT_RX_BIP_ERR_1_LSB: 0510

Table 121: **STAT_RX_BIP_ERR_1_LSB: 0510**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_bip_err_1_count[31:0]

STAT_RX_BIP_ERR_1_MSB: 0514

Table 122: **STAT_RX_BIP_ERR_1_MSB: 0514**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_bip_err_1_count[48-1:32]

STAT_RX_BIP_ERR_2_LSB: 0518

Table 123: **STAT_RX_BIP_ERR_2_LSB: 0518**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_bip_err_2_count[31:0]

STAT_RX_BIP_ERR_2_MSB: 051C

Table 124: **STAT_RX_BIP_ERR_2_MSB: 051C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_bip_err_2_count[48-1:32]

STAT_RX_BIP_ERR_3_LSB: 0520

Table 125: **STAT_RX_BIP_ERR_3_LSB: 0520**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_bip_err_3_count[31:0]

STAT_RX_BIP_ERR_3_MSB: 0524

Table 126: **STAT_RX_BIP_ERR_3_MSB: 0524**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_bip_err_3_count[48-1:32]

STAT_RX_FRAMING_ERR_0_LSB: 05A8

Table 127: **STAT_RX_FRAMING_ERR_0_LSB: 05A8**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_framing_err_0_count[31:0]

STAT_RX_FRAMING_ERR_0_MSB: 05AC

Table 128: **STAT_RX_FRAMING_ERR_0_MSB: 05AC**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_framing_err_0_count[48-1:32]

STAT_RX_FRAMING_ERR_1_LSB: 05B0

Table 129: **STAT_RX_FRAMING_ERR_1_LSB: 05B0**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_framing_err_1_count[31:0]

STAT_RX_FRAMING_ERR_1_MSB: 05B4

Table 130: **STAT_RX_FRAMING_ERR_1_MSB: 05B4**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_framing_err_1_count[48-1:32]

STAT_RX_FRAMING_ERR_2_LSB: 05B8

Table 131: **STAT_RX_FRAMING_ERR_2_LSB: 05B8**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_framing_err_2_count[31:0]

STAT_RX_FRAMING_ERR_2_MSB: 05BC

Table 132: **STAT_RX_FRAMING_ERR_2_MSB: 05BC**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_framing_err_2_count[48-1:32]

STAT_RX_FRAMING_ERR_3_LSB: 05C0

Table 133: **STAT_RX_FRAMING_ERR_3_LSB: 05C0**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_framing_err_3_count[31:0]

STAT_RX_FRAMING_ERR_3_MSB: 05C4

Table 134: **STAT_RX_FRAMING_ERR_3_MSB: 05C4**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_framing_err_3_count[48-1:32]

STAT_RX_BAD_CODE_LSB: 0660

Table 135: **STAT_RX_BAD_CODE_LSB: 0660**

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

STAT_RX_BAD_CODE_MSB: 0664

Table 136: **STAT_RX_BAD_CODE_MSB: 0664**

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

STAT_RX_ERROR_MSB: 066C

Table 137: **STAT_RX_BAD_CODE_MSB: 066C**

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

STAT_RX_ERROR_LSB: 0668

Table 138: **STAT_RX_BAD_CODE_LSB: 0668**

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

STAT_RX_RSFEC_CORRECTED_CW_INC_LSB: 0670

Table 139: **STAT_RX_RSFEC_CORRECTED_CW_INC_LSB: 0670**

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

STAT_RX_RSFEC_CORRECTED_CW_INC_MSB: 0674

Table 140: **STAT_RX_RSFEC_CORRECTED_CW_INC_MSB: 0674**

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

STAT_RX_RSFEC_UNCORRECTED_CW_INC_LSB: 0678

Table 141: **STAT_RX_RSFEC_CORRECTED_CW_INC_LSB: 0678**

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

STAT_RX_RSFEC_UNCORRECTED_CW_INC_MSB: 067C

Table 142: **STAT_RX_RSFEC_CORRECTED_CW_INC_MSB: 067C**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_rsfec_uncorrected_cw_inc_count[47:32]

STAT_RX_RSFEC_ERR_COUNT0_INC_LSB: 0680

Table 143: **STAT_RX_RSFEC_ERR_COUNT0_INC_LSB: 0680**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_rsfec_err_count0_inc_count[31:0]

STAT_RX_RSFEC_ERR_COUNT0_INC_MSB: 0684

Table 144: **STAT_RX_RSFEC_ERR_COUNT0_INC_MSB: 0684**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_rsfec_err_count0_inc_count[47:32]

STAT_RX_RSFEC_ERR_COUNT1_INC_LSB: 0688

Table 145: **STAT_RX_RSFEC_ERR_COUNT1_INC_LSB: 0688**

Bits	Default	Type	Signal
31:0	0	HIST	stat_rx_rsfec_err_count1_inc_count[31:0]

STAT_RX_RSFEC_ERR_COUNT1_INC_LSB: 068C

Table 146: **STAT_RX_RSFEC_ERR_COUNT1_INC_LSB: 068C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_rsfec_err_count1_inc_count[47:32]

STAT_TX_FRAME_ERROR_LSB: 06A0

Table 147: **STAT_RX_RSFEC_ERR_COUNT1_INC_LSB: 06A0**

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

STAT_TX_FRAME_ERROR_MSB: 06A4

Table 148: **STAT_TX_FRAME_ERROR_MSB: 06A4**

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

STAT_TX_TOTAL_PACKETS_LSB: 0700

Table 149: **STAT_TX_TOTAL_PACKETS_LSB: 0700**

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

STAT_TX_TOTAL_PACKETS_MSB: 0704

Table 150: **STAT_TX_TOTAL_PACKETS_MSB: 0704**

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

STAT_TX_TOTAL_GOOD_PACKETS_LSB: 0708

Table 151: **STAT_TX_TOTAL_GOOD_PACKETS_LSB: 0708**

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

STAT_TX_TOTAL_GOOD_PACKETS_MSB: 070C

Table 152: **STAT_TX_TOTAL_GOOD_PACKETS_MSB: 070C**

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

STAT_TX_TOTAL_BYTES_LSB: 0710

Table 153: **STAT_TX_TOTAL_BYTES_LSB: 0710**

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

STAT_TX_TOTAL_BYTES_MSB: 0714

Table 154: **STAT_TX_TOTAL_BYTES_MSB: 0714**

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

STAT_TX_TOTAL_GOOD_BYTES_LSB: 0718

Table 155: **STAT_TX_TOTAL_GOOD_BYTES_LSB: 0718**

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

STAT_TX_TOTAL_GOOD_BYTES_MSB: 071C

Table 156: **STAT_TX_TOTAL_GOOD_BYTES_MSB: 071C**

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

STAT_TX_PACKET_64_BYTES_LSB: 0720

Table 157: **STAT_TX_PACKET_64_BYTES_LSB: 0720**

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

STAT_TX_PACKET_64_BYTES_MSB: 0724

Table 158: **STAT_TX_PACKET_64_BYTES_MSB: 0724**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_packet_64_bytes_count[48-1:32]

STAT_TX_PACKET_65_127_BYTES_LSB: 0728

Table 159: **STAT_TX_PACKET_65_127_BYTES_LSB: 0728**

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

STAT_TX_PACKET_65_127_BYTES_MSB: 072C

Table 160: **STAT_TX_PACKET_65_127_BYTES_MSB: 072C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_packet_65_127_bytes_count[48-1:32]

STAT_TX_PACKET_128_255_BYTES_LSB: 0730

Table 161: **STAT_TX_PACKET_128_255_BYTES_LSB: 0730**

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

STAT_TX_PACKET_128_255_BYTES_MSB: 0734

Table 162: **STAT_TX_PACKET_128_255_BYTES_MSB: 0734**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_packet_128_255_bytes_count[48-1:32]

STAT_TX_PACKET_256_511_BYTES_LSB: 0738

Table 163: **STAT_TX_PACKET_256_511_BYTES_LSB: 0738**

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

STAT_TX_PACKET_256_511_BYTES_MSB: 073C

Table 164: **STAT_TX_PACKET_256_511_BYTES_MSB: 073C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_packet_256_511_bytes_count[48-1:32]

STAT_TX_PACKET_512_1023_BYTES_LSB: 0740

Table 165: **STAT_TX_PACKET_512_1023_BYTES_LSB: 0740**

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

STAT_TX_PACKET_512_1023_BYTES_MSB: 0744

Table 166: **STAT_TX_PACKET_512_1023_BYTES_MSB: 0744**

Bits	Default	Type	Signal
15:0		HIST	stat_tx_packet_512_1023_bytes_count[48-1:32]

STAT_TX_PACKET_1024_1518_BYTES_LSB: 0748

Table 167: **STAT_TX_PACKET_1024_1518_BYTES_LSB: 0748**

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

STAT_TX_PACKET_1024_1518_BYTES_MSB: 074C

Table 168: **STAT_TX_PACKET_1024_1518_BYTES_MSB: 074C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_packet_1024_1518_bytes_count[48-1:32]

STAT_TX_PACKET_1519_1522_BYTES_LSB: 0750

Table 169: **STAT_TX_PACKET_1519_1522_BYTES_LSB: 0750**

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

STAT_TX_PACKET_1519_1522_BYTES_MSB: 0754

Table 170: **STAT_TX_PACKET_1519_1522_BYTES_MSB: 0754**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_packet_1519_1522_bytes_count[48-1:32]

STAT_TX_PACKET_1523_1548_BYTES_LSB: 0758

Table 171: **STAT_TX_PACKET_1523_1548_BYTES_LSB: 0758**

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

STAT_TX_PACKET_1523_1548_BYTES_MSB: 075C

Table 172: **STAT_TX_PACKET_1523_1548_BYTES_MSB: 075C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_packet_1523_1548_bytes_count[48-1:32]

STAT_TX_PACKET_1549_2047_BYTES_LSB: 0760

Table 173: **STAT_TX_PACKET_1549_2047_BYTES_LSB: 0760**

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

STAT_TX_PACKET_1549_2047_BYTES_MSB: 0764

Table 174: **STAT_TX_PACKET_1549_2047_BYTES_MSB: 0764**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_packet_1549_2047_bytes_count[48-1:32]

STAT_TX_PACKET_2048_4095_BYTES_LSB: 0768

Table 175: **STAT_TX_PACKET_2048_4095_BYTES_LSB: 0768**

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

STAT_TX_PACKET_2048_4095_BYTES_MSB: 076C

Table 176: **STAT_TX_PACKET_2048_4095_BYTES_MSB: 076C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_packet_2048_4095_bytes_count[48-1:32]

STAT_TX_PACKET_4096_8191_BYTES_LSB: 0770

Table 177: **STAT_TX_PACKET_4096_8191_BYTES_LSB: 0770**

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

STAT_TX_PACKET_4096_8191_BYTES_MSB: 0774

Table 178: **STAT_TX_PACKET_4096_8191_BYTES_MSB: 0774**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_packet_4096_8191_bytes_count[48-1:32]

STAT_TX_PACKET_8192_9215_BYTES_LSB: 0778

Table 179: **STAT_TX_PACKET_8192_9215_BYTES_LSB: 0778**

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

STAT_TX_PACKET_8192_9215_BYTES_MSB: 077C

Table 180: **STAT_TX_PACKET_8192_9215_BYTES_MSB: 077C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_tx_packet_8192_9215_bytes_count[48-1:32]

STAT_TX_PACKET_LARGE_LSB: 0780

Table 181: **STAT_TX_PACKET_LARGE_LSB: 0780**

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

STAT_TX_PACKET_LARGE_MSB: 0784

Table 182: **STAT_TX_PACKET_LARGE_MSB: 0784**

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

STAT_TX_PACKET_SMALL_LSB: 0788

Table 183: **STAT_TX_PACKET_SMALL_LSB: 0788**

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

STAT_TX_PACKET_SMALL_MSB: 078C

Table 184: **STAT_TX_PACKET_SMALL_MSB: 078C**

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

STAT_TX_BAD_FCS_LSB: 07B8

Table 185: **STAT_TX_BAD_FCS_LSB: 07B8**

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

STAT_TX_BAD_FCS_MSB: 07BC

Table 186: **STAT_TX_BAD_FCS_MSB: 07BC**

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

STAT_TX_UNICAST_LSB: 07D0

Table 187: **STAT_TX_UNICAST_LSB: 07D0**

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

STAT_TX_UNICAST_MSB: 07D4

Table 188: **STAT_TX_UNICAST_MSB: 07D4**

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

STAT_TX_MULTICAST_LSB: 07D8

Table 189: **STAT_TX_MULTICAST_LSB: 07D8**

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

STAT_TX_MULTICAST_MSB: 07DC

Table 190: **STAT_TX_MULTICAST_MSB: 07DC**

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

STAT_TX_BROADCAST_LSB: 07E0

Table 191: **STAT_TX_BROADCAST_LSB: 07E0**

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

STAT_TX_BROADCAST_MSB: 07E4

Table 192: **STAT_TX_BROADCAST_MSB: 07E4**

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

STAT_TX_VLAN_LSB: 07E8

Table 193: **STAT_TX_VLAN_LSB: 07E8**

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

STAT_TX_VLAN_MSB: 07EC

Table 194: **STAT_TX_VLAN_MSB: 07EC**

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

STAT_TX_PAUSE_LSB: 07F0

Table 195: **STAT_TX_PAUSE_LSB: 07F0**

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

STAT_TX_PAUSE_MSB: 07F4

Table 196: **STAT_TX_PAUSE_MSB: 07F4**

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

STAT_TX_USER_PAUSE_LSB: 07F8

Table 197: **STAT_TX_USER_PAUSE_LSB: 07F8**

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

STAT_TX_USER_PAUSE_MSB: 07FC

Table 198: **STAT_TX_USER_PAUSE_MSB: 07FC**

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

STAT_RX_TOTAL_PACKETS_LSB: 0808

Table 199: **STAT_RX_TOTAL_PACKETS_LSB: 0808**

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

STAT_RX_TOTAL_PACKETS_MSB: 080C

Table 200: **STAT_RX_TOTAL_PACKETS_MSB: 080C**

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

STAT_RX_TOTAL_GOOD_PACKETS_LSB: 0810

Table 201: **STAT_RX_TOTAL_GOOD_PACKETS_LSB: 0810**

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

STAT_RX_TOTAL_GOOD_PACKETS_MSB: 0814

Table 202: **STAT_RX_TOTAL_GOOD_PACKETS_MSB: 0814**

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

STAT_RX_TOTAL_BYTES_LSB: 0818

Table 203: **STAT_RX_TOTAL_BYTES_LSB: 0818**

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

STAT_RX_TOTAL_BYTES_MSB: 081C

Table 204: **STAT_RX_TOTAL_BYTES_MSB: 081C**

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

STAT_RX_TOTAL_GOOD_BYTES_LSB: 0820

Table 205: **STAT_RX_TOTAL_GOOD_BYTES_LSB: 0820**

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

STAT_RX_TOTAL_GOOD_BYTES_MSB: 0824

Table 206: **STAT_RX_TOTAL_GOOD_BYTES_MSB: 0824**

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

STAT_RX_PACKET_64_BYTES_LSB: 0828

Table 207: **STAT_RX_PACKET_64_BYTES_LSB: 0828**

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

STAT_RX_PACKET_64_BYTES_MSB: 082C

Table 208: **STAT_RX_PACKET_64_BYTES_MSB: 082C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_packet_64_bytes_count[48-1:32]

STAT_RX_PACKET_65_127_BYTES_LSB: 0830

Table 209: **STAT_RX_PACKET_65_127_BYTES_LSB: 0830**

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

STAT_RX_PACKET_65_127_BYTES_MSB: 0834

Table 210: **STAT_RX_PACKET_65_127_BYTES_MSB: 0834**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_packet_65_127_bytes_count[48-1:32]

STAT_RX_PACKET_128_255_BYTES_LSB: 0838

Table 211: **STAT_RX_PACKET_128_255_BYTES_LSB: 0838**

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

STAT_RX_PACKET_128_255_BYTES_MSB: 083C

Table 212: **STAT_RX_PACKET_128_255_BYTES_MSB: 083C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_packet_128_255_bytes_count[48-1:32]

STAT_RX_PACKET_256_511_BYTES_LSB: 0840

Table 213: **STAT_RX_PACKET_256_511_BYTES_LSB: 0840**

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

STAT_RX_PACKET_256_511_BYTES_MSB: 0844

Table 214: **STAT_RX_PACKET_256_511_BYTES_MSB: 0844**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_packet_256_511_bytes_count[48-1:32]

STAT_RX_PACKET_512_1023_BYTES_LSB: 0848

Table 215: **STAT_RX_PACKET_512_1023_BYTES_LSB: 0848**

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

STAT_RX_PACKET_512_1023_BYTES_MSB: 084C

Table 216: **STAT_RX_PACKET_512_1023_BYTES_MSB: 084C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_packet_512_1023_bytes_count[48-1:32]

STAT_RX_PACKET_1024_1518_BYTES_LSB: 0850

Table 217: **STAT_RX_PACKET_1024_1518_BYTES_LSB: 0850**

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

STAT_RX_PACKET_1024_1518_BYTES_MSB: 0854

Table 218: **STAT_RX_PACKET_1024_1518_BYTES_MSB: 0854**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_packet_1024_1518_bytes_count[48-1:32]

STAT_RX_PACKET_1519_1522_BYTES_LSB: 0858

Table 219: **STAT_RX_PACKET_1519_1522_BYTES_LSB: 0858**

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

STAT_RX_PACKET_1519_1522_BYTES_MSB: 085C

Table 220: **STAT_RX_PACKET_1519_1522_BYTES_MSB: 085C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_packet_1519_1522_bytes_count[48-1:32]

STAT_RX_PACKET_1523_1548_BYTES_LSB: 0860

Table 221: **STAT_RX_PACKET_1523_1548_BYTES_LSB: 0860**

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

STAT_RX_PACKET_1523_1548_BYTES_MSB: 0864

Table 222: **STAT_RX_PACKET_1523_1548_BYTES_MSB: 0864**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_packet_1523_1548_bytes_count[48-1:32]

STAT_RX_PACKET_1549_2047_BYTES_LSB: 0868

Table 223: **STAT_RX_PACKET_1549_2047_BYTES_LSB: 0868**

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

STAT_RX_PACKET_1549_2047_BYTES_MSB: 086C

Table 224: **STAT_RX_PACKET_1549_2047_BYTES_MSB: 086C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_packet_1549_2047_bytes_count[48-1:32]

STAT_RX_PACKET_2048_4095_BYTES_LSB: 0870

Table 225: **STAT_RX_PACKET_1549_2047_BYTES_MSB: 086C**

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

STAT_RX_PACKET_2048_4095_BYTES_MSB: 0874

Table 226: **STAT_RX_PACKET_2048_4095_BYTES_MSB: 0874**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_packet_2048_4095_bytes_count[48-1:32]

STAT_RX_PACKET_4096_8191_BYTES_LSB: 0878

Table 227: **STAT_RX_PACKET_4096_8191_BYTES_LSB: 0878**

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

STAT_RX_PACKET_4096_8191_BYTES_MSB: 087C

Table 228: **STAT_RX_PACKET_4096_8191_BYTES_MSB: 087C**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_packet_4096_8191_bytes_count[48-1:32]

STAT_RX_PACKET_8192_9215_BYTES_LSB: 0880

Table 229: **STAT_RX_PACKET_8192_9215_BYTES_LSB: 0880**

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

STAT_RX_PACKET_8192_9215_BYTES_MSB: 0884

Table 230: **STAT_RX_PACKET_8192_9215_BYTES_MSB: 0884**

Bits	Default	Type	Signal
15:0	0	HIST	stat_rx_packet_8192_9215_bytes_count[48-1:32]

STAT_RX_PACKET_LARGE_LSB: 0888

Table 231: **STAT_RX_PACKET_LARGE_LSB: 0888**

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

STAT_RX_PACKET_LARGE_MSB: 088C

Table 232: **STAT_RX_PACKET_LARGE_MSB: 088C**

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

STAT_RX_PACKET_SMALL_LSB: 0890

Table 233: **STAT_RX_PACKET_SMALL_LSB: 0890**

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

STAT_RX_PACKET_SMALL_MSB: 0894

Table 234: **STAT_RX_PACKET_SMALL_MSB: 0894**

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

STAT_RX_UNDERSIZE_LSB: 0898

Table 235: **STAT_RX_UNDERSIZE_LSB: 0898**

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

STAT_RX_UNDERSIZE_MSB: 089C

Table 236: **STAT_RX_UNDERSIZE_MSB: 089C**

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

STAT_RX_FRAGMENT_LSB: 08A0

Table 237: **STAT_RX_FRAGMENT_LSB: 08A0**

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

STAT_RX_FRAGMENT_MSB: 08A4

Table 238: **STAT_RX_FRAGMENT_MSB: 08A4**

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

STAT_RX_OVERSIZE_LSB: 08A8

Table 239: **STAT_RX_OVERSIZE_LSB: 08A8**

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

STAT_RX_OVERSIZE_MSB: 08AC

Table 240: **STAT_RX_OVERSIZE_MSB: 08AC**

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

STAT_RX_TOOLONG_LSB: 08B0

Table 241: **STAT_RX_TOOLONG_LSB: 08B0**

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

STAT_RX_TOOLONG_MSB: 08B4

Table 242: **STAT_RX_TOOLONG_MSB: 08B4**

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

STAT_RX_JABBER_LSB: 08B8

Table 243: **STAT_RX_JABBER_LSB: 08B8**

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

STAT_RX_JABBER_MSB: 08BC

Table 244: **STAT_RX_JABBER_MSB: 08BC**

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

STAT_RX_BAD_FCS_LSB: 08C0

Table 245: **STAT_RX_BAD_FCS_LSB: 08C0**

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

STAT_RX_BAD_FCS_MSB: 08C4

Table 246: **STAT_RX_BAD_FCS_MSB: 08C4**

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

STAT_RX_PACKET_BAD_FCS_LSB: 08C8

Table 247: **STAT_RX_PACKET_BAD_FCS_LSB: 08C8**

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

STAT_RX_PACKET_BAD_FCS_MSB: 08CC

Table 248: **STAT_RX_PACKET_BAD_FCS_MSB: 08CC**

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

STAT_RX_STOMPED_FCS_LSB: 08D0

Table 249: **STAT_RX_STOMPED_FCS_LSB: 08D0**

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

STAT_RX_STOMPED_FCS_MSB: 08D4

Table 250: **STAT_RX_STOMPED_FCS_MSB: 08D4**

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

STAT_RX_UNICAST_LSB: 08D8

Table 251: **STAT_RX_UNICAST_LSB: 08D8**

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

STAT_RX_UNICAST_MSB: 08DC

Table 252: **STAT_RX_UNICAST_MSB: 08DC**

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

STAT_RX_MULTICAST_LSB: 08E0

Table 253: **STAT_RX_MULTICAST_LSB: 08E0**

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

STAT_RX_MULTICAST_MSB: 08E4

Table 254: **STAT_RX_MULTICAST_MSB: 08E4**

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

STAT_RX_BROADCAST_LSB: 08E8

Table 255: **STAT_RX_BROADCAST_LSB: 08E8**

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

STAT_RX_BROADCAST_MSB: 08EC

Table 256: **STAT_RX_BROADCAST_MSB: 08EC**

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

STAT_RX_VLAN_LSB: 08F0

Table 257: **STAT_RX_VLAN_LSB: 08F0**

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

STAT_RX_VLAN_MSB: 08F4

Table 258: **STAT_RX_VLAN_MSB: 08F4**

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

STAT_RX_PAUSE_LSB: 08F8

Table 259: **STAT_RX_PAUSE_LSB: 08F8**

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

STAT_RX_PAUSE_MSB: 08FC

Table 260: **STAT_RX_PAUSE_MSB: 08FC**

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

STAT_RX_USER_PAUSE_LSB: 0900

Table 261: **STAT_RX_USER_PAUSE_LSB: 0900**

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

STAT_RX_USER_PAUSE_MSB: 0904

Table 262: **STAT_RX_USER_PAUSE_MSB: 0904**

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

STAT_RX_INRANGEERR_LSB: 0908

Table 263: **STAT_RX_INRANGEERR_LSB: 0908**

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

STAT_RX_INRANGEERR_MSB: 090C

Table 264: **STAT_RX_INRANGEERR_MSB: 090C**

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

STAT_RX_TRUNCATED_LSB: 0910

Table 265: **STAT_RX_TRUNCATED_LSB: 0910**

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

STAT_RX_TRUNCATED_MSB: 0914

Table 266: **STAT_RX_TRUNCATED_MSB: 0914**

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

STAT_RX_TEST_PATTERN_MISMATCH_LSB: 0918

Table 267: **STAT_RX_TEST_PATTERN_MISMATCH_LSB: 0918**

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

STAT_RX_TEST_PATTERN_MISMATCH_MSB: 091C

Table 268: **STAT_RX_TEST_PATTERN_MISMATCH_MSB: 091C**

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

STAT_FEC_INC_CORRECT_COUNT_LSB: 0920

Table 269: **STAT_FEC_INC_CORRECT_COUNT_LSB: 0920**

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

STAT_FEC_INC_CORRECT_COUNT_MSB: 0924

Table 270: **STAT_FEC_INC_CORRECT_COUNT_MSB: 0924**

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

STAT_FEC_INC_CANT_CORRECT_COUNT_LSB: 0928

Table 271: **STAT_FEC_INC_CANT_CORRECT_COUNT_LSB: 0928**

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

STAT_FEC_INC_CANT_CORRECT_COUNT_MSB: 092C

Table 272: **STAT_FEC_INC_CANT_CORRECT_COUNT_MSB: 092C**

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

Designing with the Subsystem

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

General Design Guidelines

Use the Example Design

Each instance of the 40G/50G High Speed Ethernet subsystem created by the Vivado design tool is delivered with an example design that can be implemented in a device and then simulated. This design can be used as a starting point for your own design or can be used to sanity-check your application in the event of difficulty. See the Example Design content for information about using and customizing the example designs for the subsystem.

Know the Degree of Difficulty

The Xilinx® 40G/50G High Speed Ethernet subsystem designs are challenging to implement in any technology, and the degree of difficulty is further influenced by:

- Maximum system clock frequency
- Targeted device architecture
- Nature of your application

All 40G/50G High Speed Ethernet subsystem implementations need careful attention to system performance requirements. Pipelining, logic mapping, placement constraints, and logic duplication are all methods that help boost system performance.

Registering Signals

To simplify timing and increase system performance in a programmable device design, keep all inputs and outputs registered between the user application and the subsystem. This means that all inputs and outputs from the user application should come from, or connect to, a flip-flop. While registering signals might not be possible for all paths, it simplifies timing analysis and makes it easier for the Xilinx® tools to place and route the design.

Recognize Timing Critical Signals

The constraints provided with the example design identify the critical signals and timing constraints that should be applied.

Make Only Allowed Modifications

You should not modify the subsystem. Any modifications can have adverse effects on system timing and protocol compliance. Supported user configurations of the subsystem can only be made by selecting the options in the customization IP dialog box when the subsystem is generated.

Clocking

This section describes the clocking for all the 40G/50G configurations at the component support wrapper layer. There are three fundamentally different clocking architectures depending on the functionality and options:

- PCS/PMA Only Clocking
- 40G/50G MAC with PCS/PMA Clocking
- Low Latency 40G/50G MAC with PCS/PMA Clocking

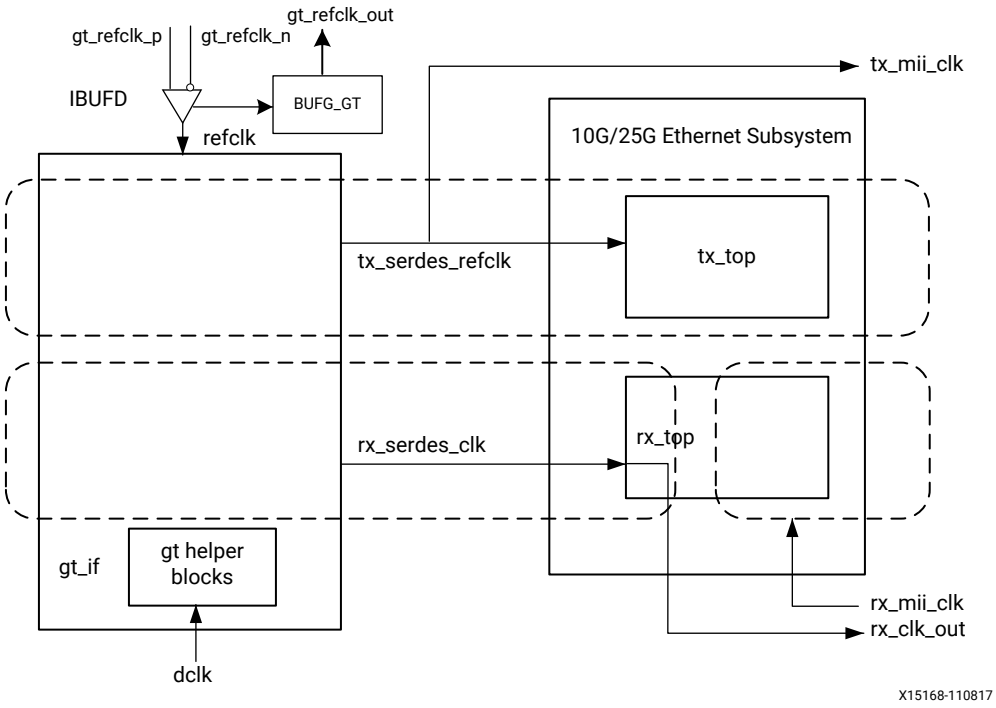
Also described is Auto-Negotiation and Link Training Clocking.

Note: When Data Path Interface is selected as the 256-bit Regular AXI4-Stream in the Vivado IDE, the example design TX/RX AXI4-Stream should use `tx_out_clk`; otherwise this can lead to packet mismatch.

PCS/PMA Only Clocking

The clocking architecture for the 40G/50G PCS is illustrated below. There are three clock domains in the datapath, as illustrated by the dashed lines in the following figure:

Figure 19: **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 312.5 MHz for 40G and 390.625 MHz for 50G.

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 XLGMII/50GMII data bus. This clock and the RX XLGMII/50GMII bus must be within 100 ppm of the required frequency, which is 312.5 MHz for 40G and 390.625 MHz for 50G.

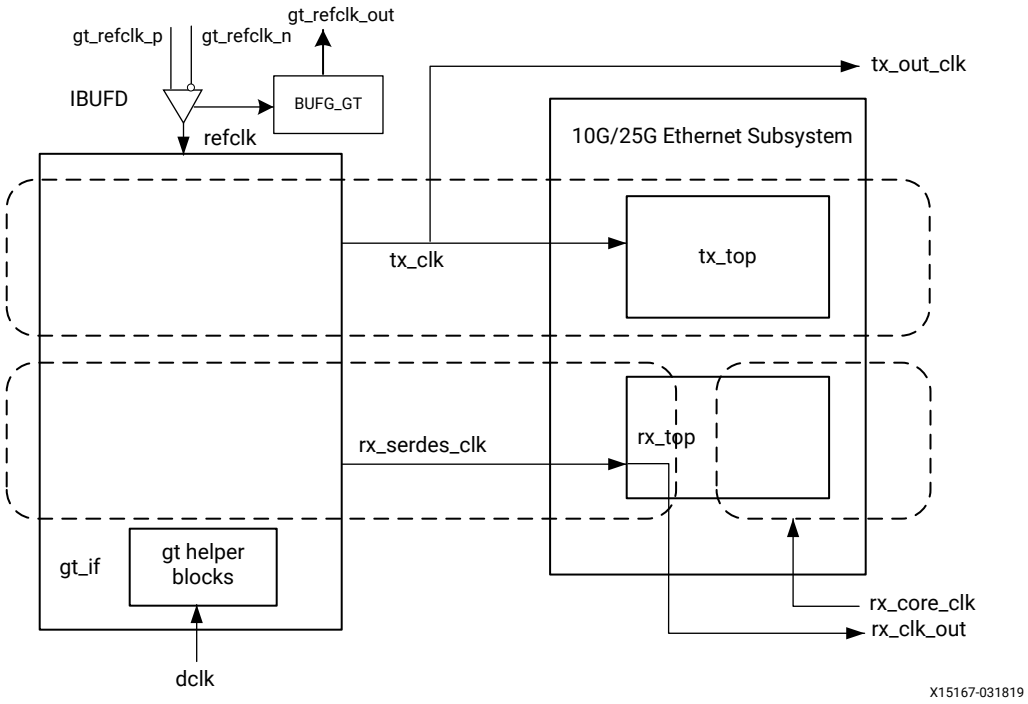
dclk

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.

40G/50G MAC with PCS/PMA Clocking

The clocking architecture for the 40/50G MAC with PCS/PMA clocking is illustrated below. This version of the subsystem includes FIFOs in the RX. There are three clock domains in the data path, as illustrated by the dashed lines in the following figure:

Figure 20: **40G/50G 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 312.5 MHz for 40G and 390.625 MHz for 50G.

rx_clk_out

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

rx_clk_out

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

rx_clk

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

dclk

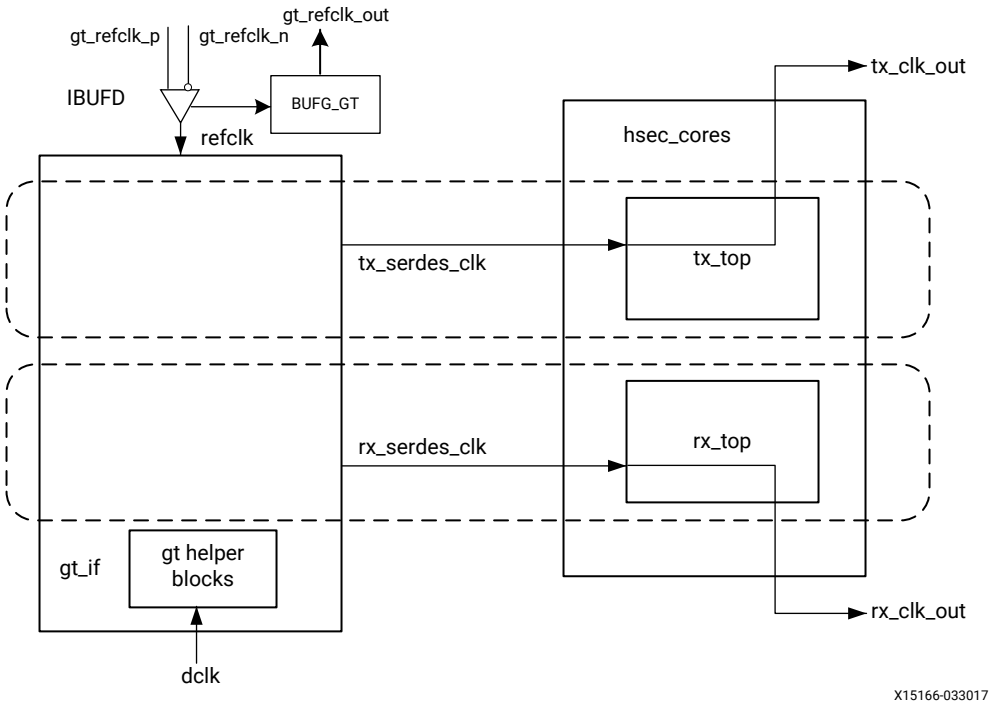
The `dclk` signal must be a convenient stable clock. It is used as a reference frequency for the GT helper blocks which initiate the GT itself. In the example design, a typical value is 75 MHz, which is readily derived from the 300 MHz clock available on the VCU107 evaluation board.

***Note*:** The actual frequency must be known to the GT helper blocks for proper operation.

Low Latency 40G/50G MAC with PCS/PMA Clocking

The clocking architecture for the Low Latency 40/50G MAC with PCS/PMA clocking is illustrated in the following figure. Low latency is achieved by omitting the RX FIFO, 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 21: **Low Latency 40G/50G 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 312.5 MHz for 40G, and 390.625 MHz for 50G.

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

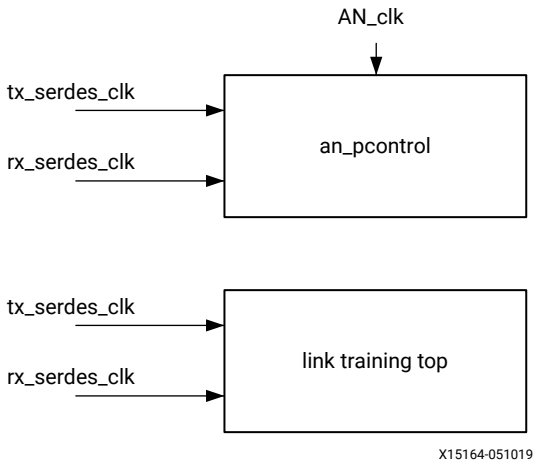
The `dclk` signal must be a convenient stable clock. It is used as a reference frequency for the GT helper blocks which initiate the GT itself. In the example design, a typical value is 75 MHz, which is readily derived from the 300 MHz clock available on the VCU107 evaluation board. Note that the actual 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 50GBASE-KR or 50GBASE-CR 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 22: **Auto-Negotiation and Link Training Clocking**



tx_serdes_clk

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

rx_serdes_clk

The `rx_serdes_clk` drives the RX line side logic for the Auto-Negotiation and Link Training.

AN_clk

The `AN_clk` drives the Auto-Negotiation state machine. All ability signals are on this clock domain. The `AN_clk` can be any convenient frequency. In the example design, `AN_clk` is connected to the `dclk` input, which has a typical frequency of 75 MHz. The `AN_clk` frequency must be known to the Auto-Negotiation state machine because it is the reference for all timers.

LogiCORE Example Design Clocking and Resets

The 40G/50G High Speed Ethernet Subsystem has separate reset inputs for the RX and TX paths that can be asserted independently. Within the RX and TX paths, there are resets for each of the various clock domains. The reset procedure is simple and the only requirement is that a reset must be asserted when the corresponding clock is not stable.

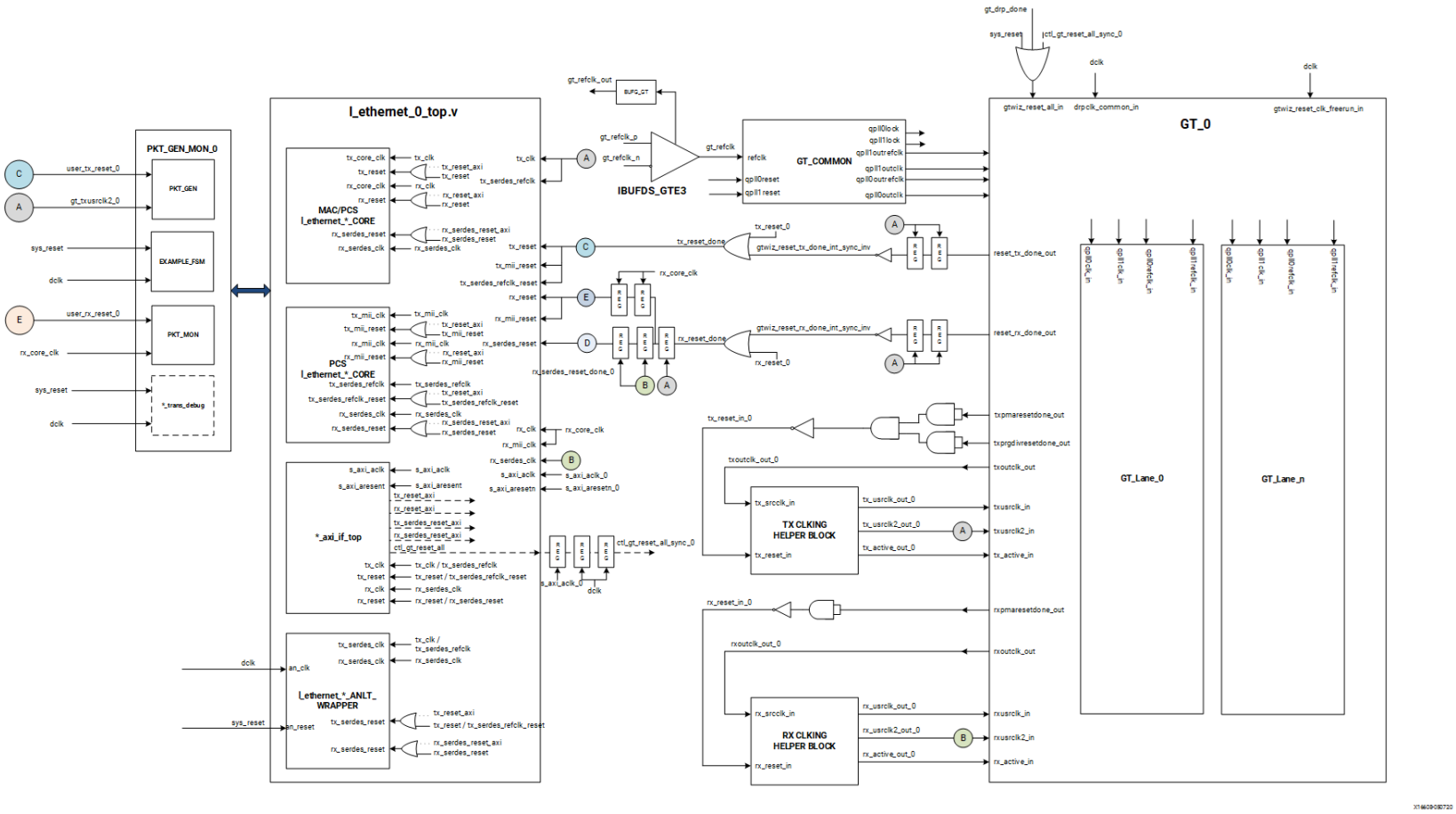
The 40G/50G High Speed Ethernet Subsystem takes care of ensuring that the different resets properly interact with each other internally and the interface operates properly (that is, there is no order required for asserting/deasserting different resets). It is left up to you to ensure a reset is held until the corresponding clock is fully stable.

Note: Some of the control inputs to the 40G/50G High Speed Ethernet Subsystem can only be modified while the core is held in reset. If one of these inputs needs changing, the appropriate RX or TX AXI4-Stream reset input (rx_reset or tx_reset) must be asserted until the control input is stabilized. See Table 2-2 for a list of these inputs. Currently, all resets are synchronous to their corresponding clocks. That is, there must be a 0-1 transition on the corresponding clock while the reset is asserted High in order for the reset to be performed.

The following figures illustrate the clocking and reset structure when you implement the Example Design using the Vivado tools.

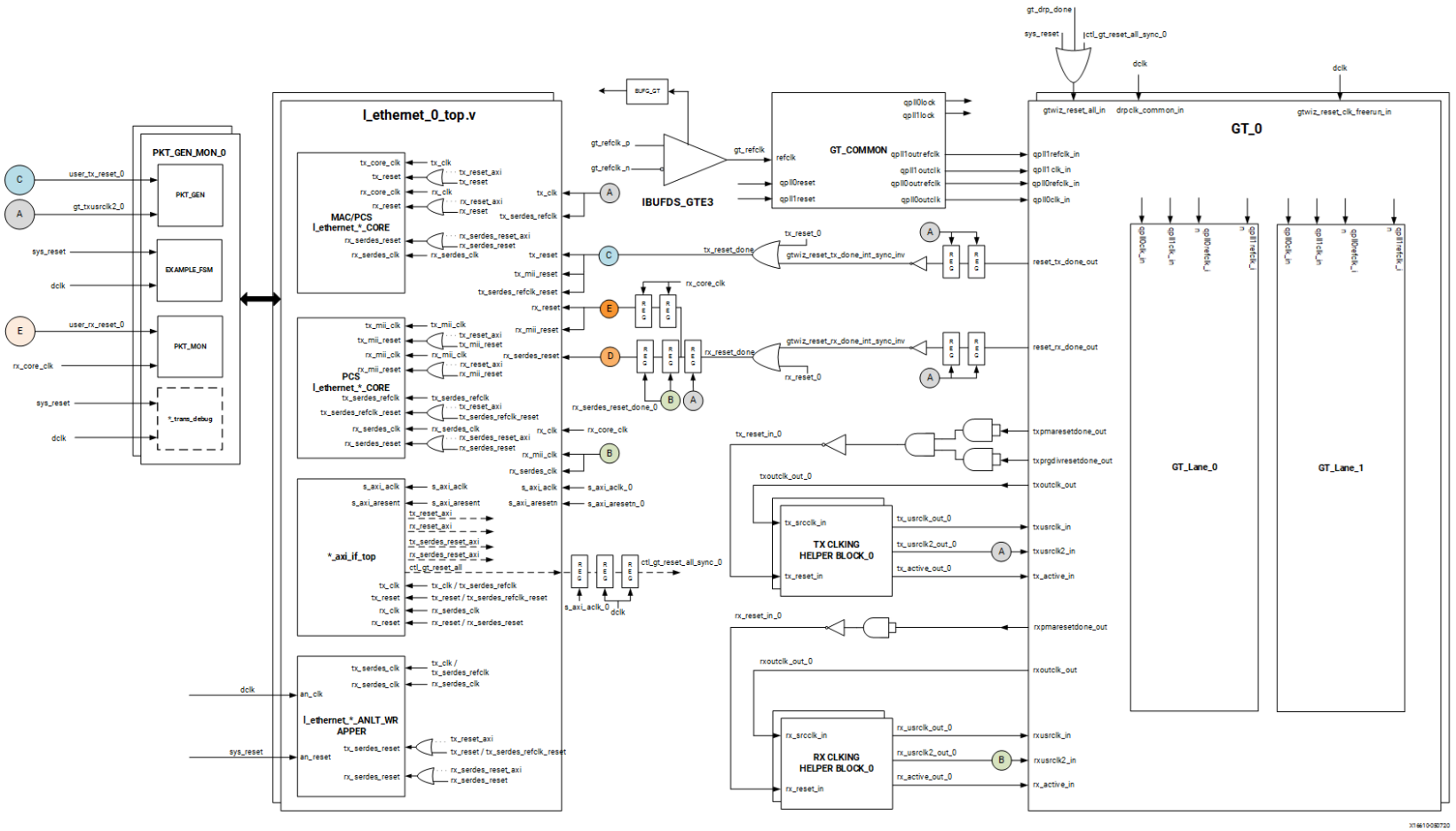
Detailed Diagram of Single Core (1x50G) - Asynchronous Clock Mode (GTY/GTH) (2x25G)

Figure 23: **Detailed Diagram of Single Core (1x50G) - Asynchronous Clock Mode (GTY/GTH) (2x25G)**



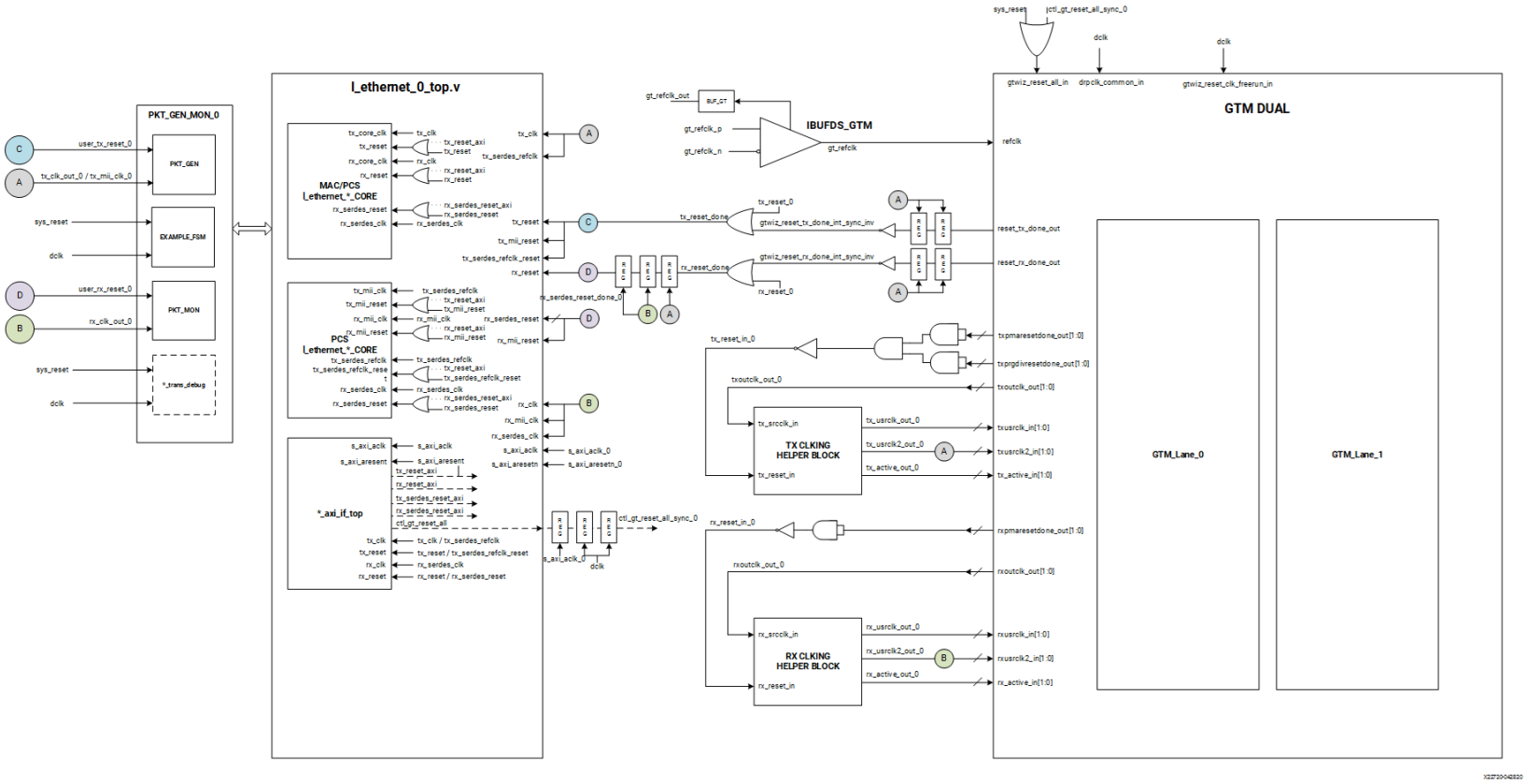
Detailed Diagram of Multiple Cores (2x50G) - Asynchronous Clock Mode GTY (2x25G)

Figure 24: **Detailed Diagram of Multiple Cores (2x50G) - Asynchronous Clock Mode GTY (2x25G)**



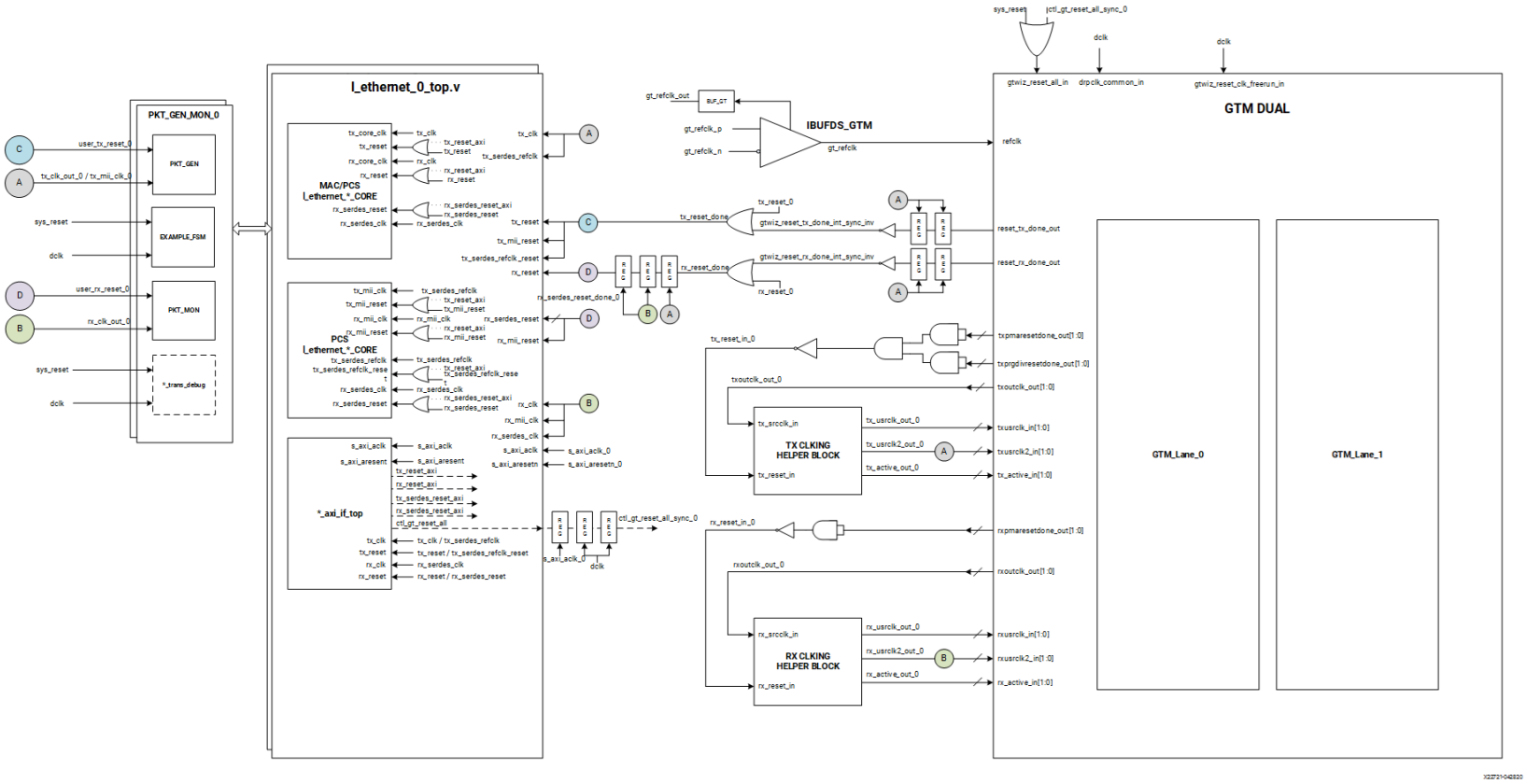
Detailed Diagram of Single Core (1x50G) - Asynchronous Clock Mode GTM (1x50G)

Figure 25: **Detailed Diagram of Single Core (1x50G) - Asynchronous Clock Mode GTM (1x50G)**



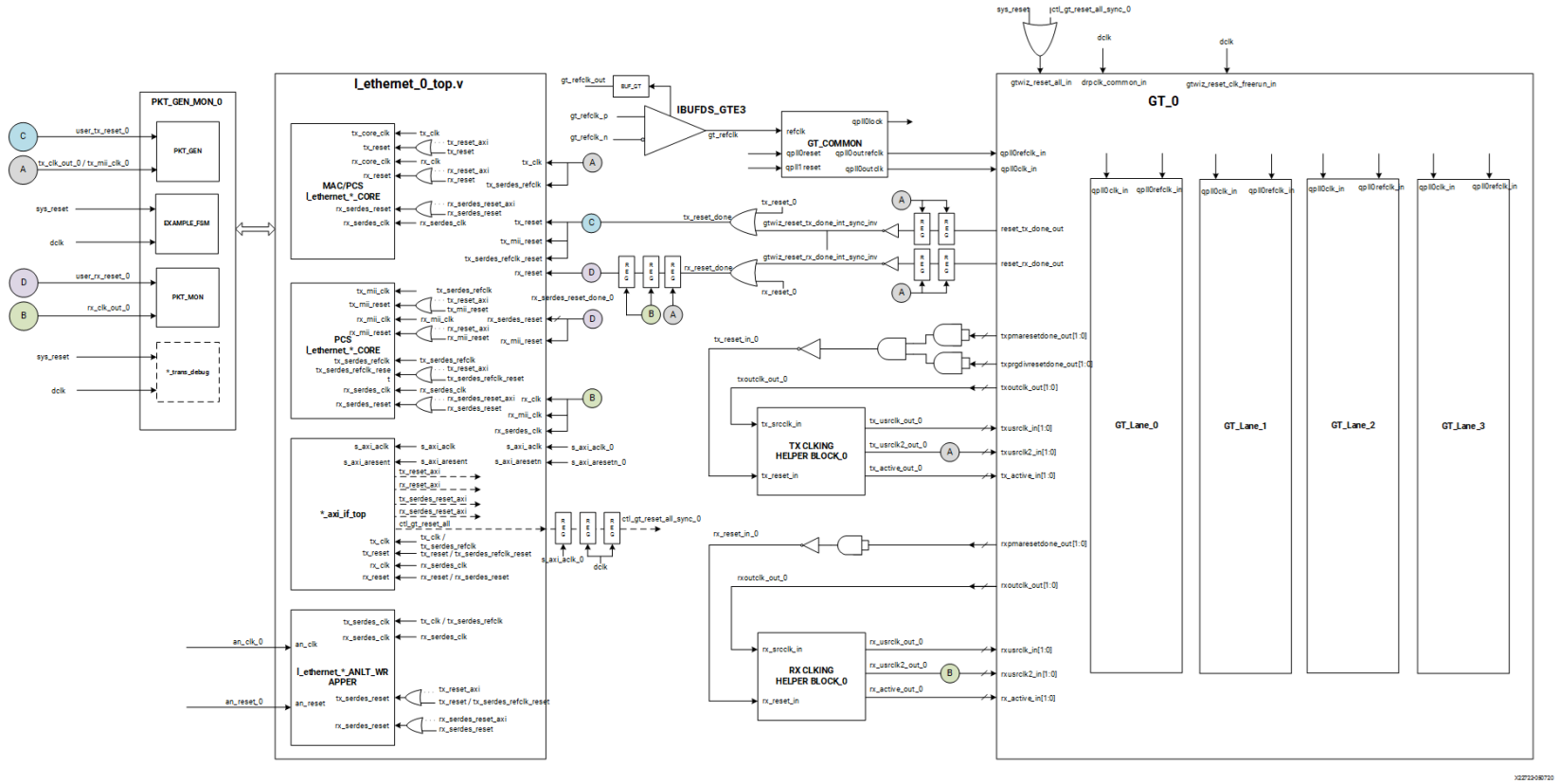
Detailed Diagram of Multiple Cores (2x50G) - Asynchronous Clock Mode GTM (2x50G)

Figure 26: **Detailed Diagram of Multiple Cores (2x50G) - Asynchronous Clock Mode GTM (2x50G)**



Detailed Diagram of Single Core (1x40G) - Asynchronous Clock Mode GTY/GTH (4x10G)

Figure 27: **Detailed Diagram of Single Core (1x40G) - Asynchronous Clock Mode GTY/GTH (4x10G)**



SerDes Data Mapping for GTM

Figure 28: **50GAUI1 with GTM Integrated Hard RS-FEC**

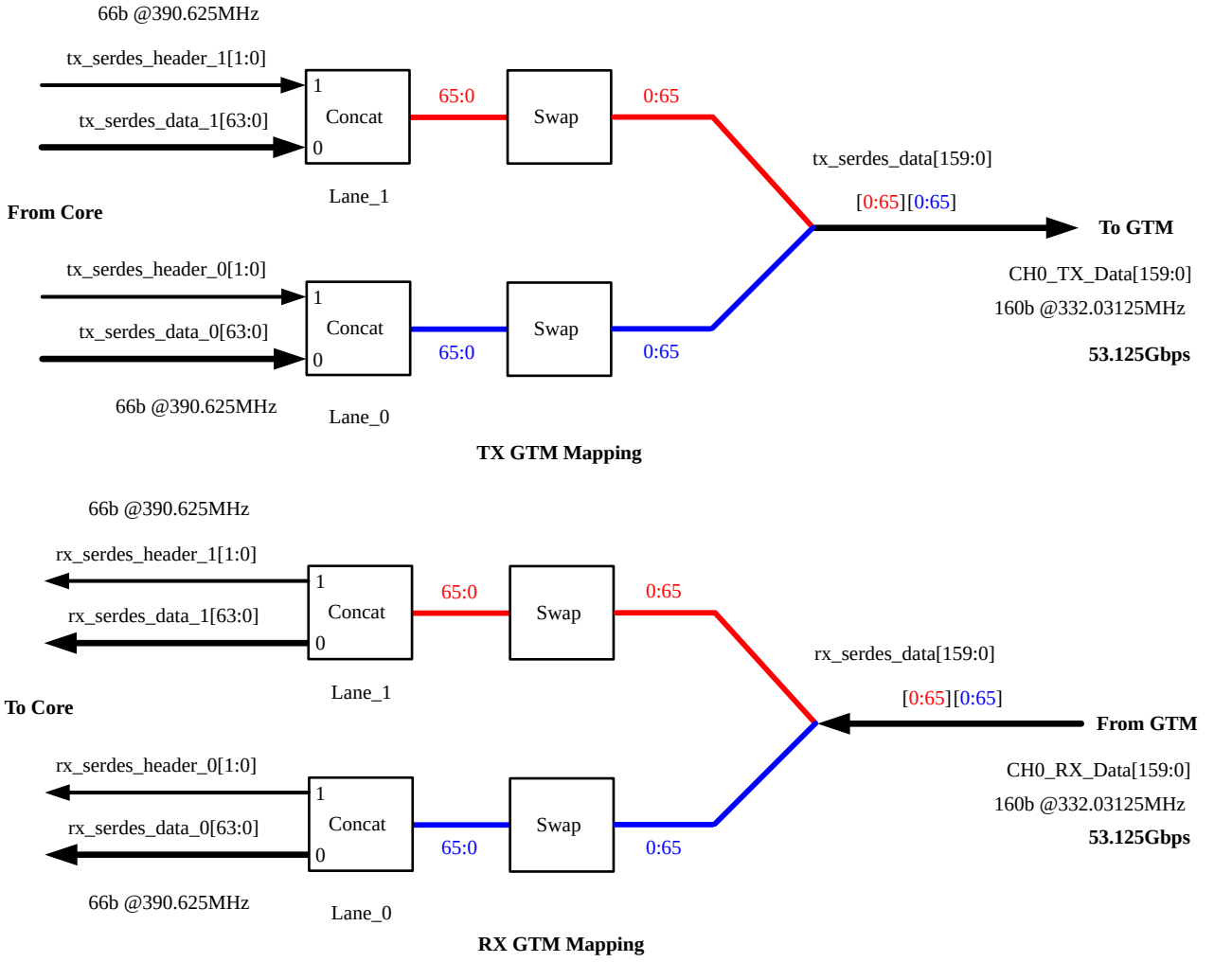


Figure 29: **LAUI2 with Soft RS-FEC**

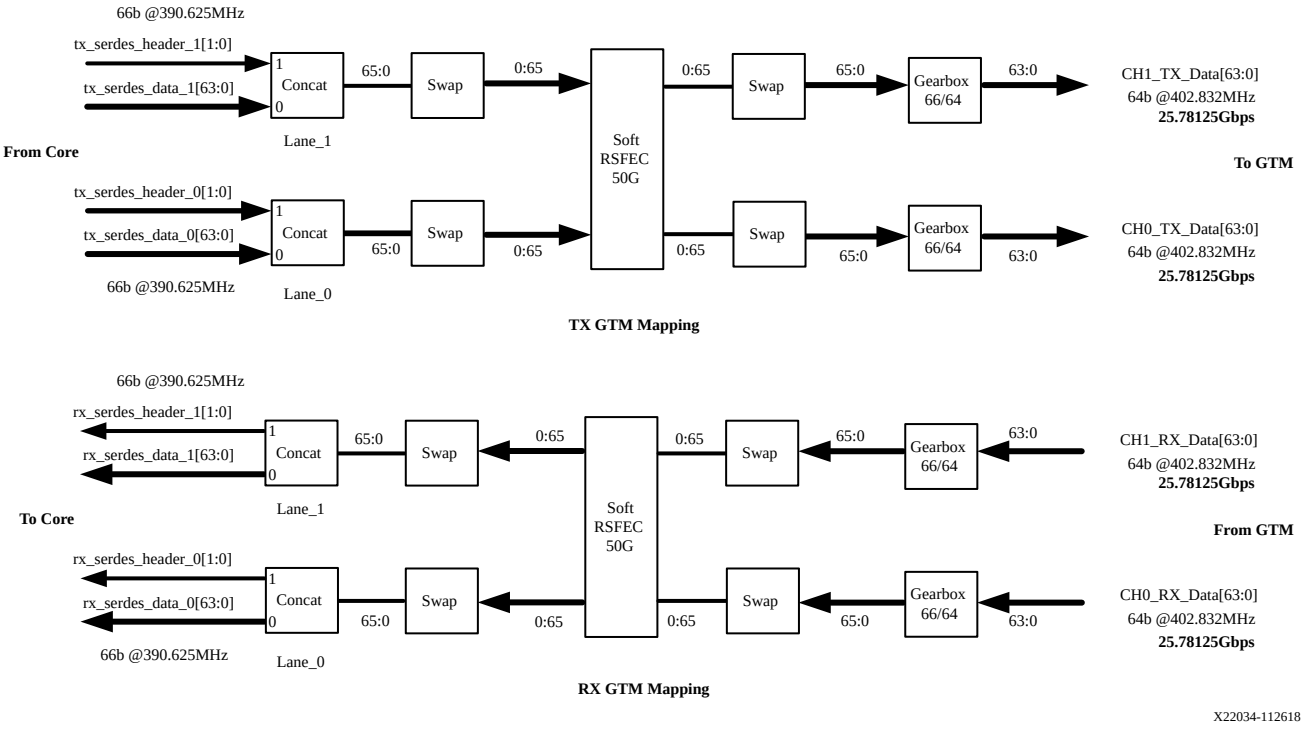
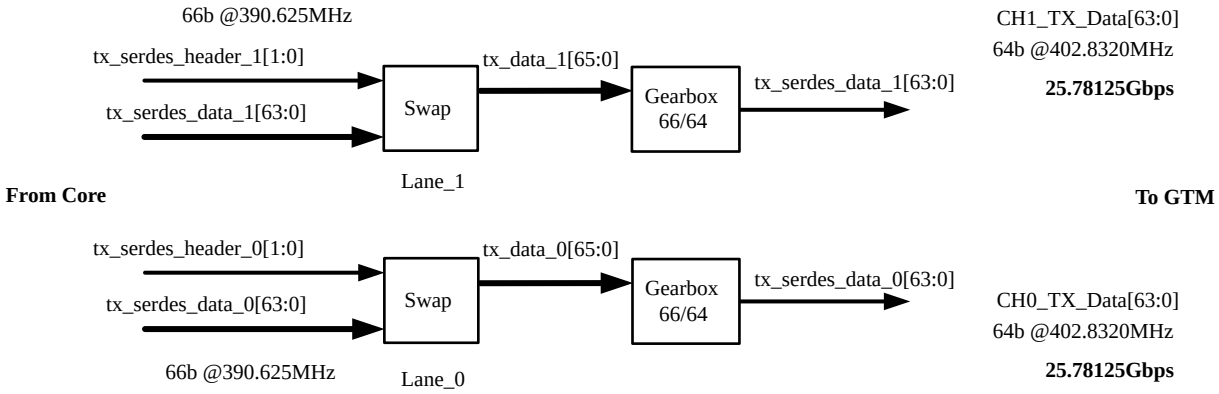


Figure 30: **LAUI2 without Soft RS-FEC**



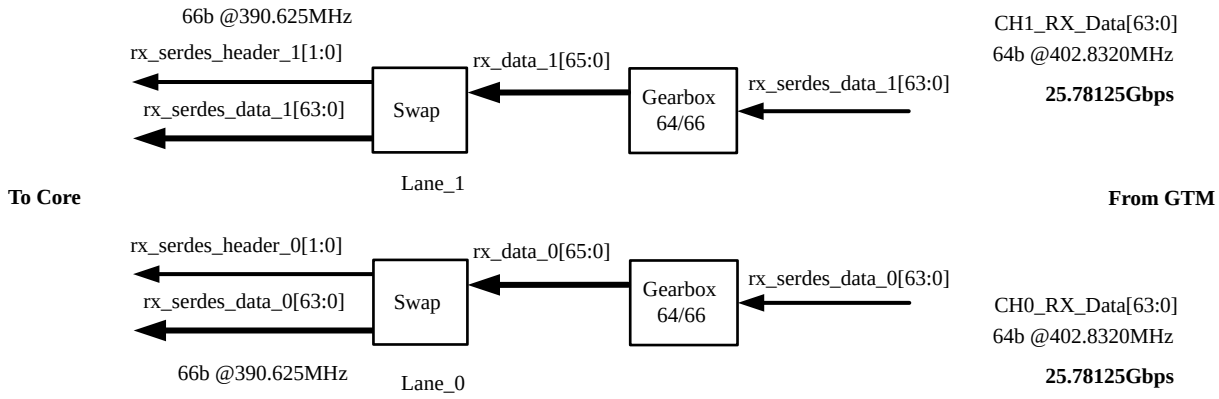
tx_data_1[65:0] = { data[0], data[1], data[2], ……. , data[63], header[0], header[1] }

tx_data_0[65:0] = { data[0], data[1], data[2], ……. , data[63], header[0], header[1] }

tx_serdes_data_1[63:0] = { tx_data_1[63], ……. , tx_data_1[2], tx_data_1[1], tx_data_1[0] }

tx_serdes_data_0[63:0] = { tx_data_0[63], ……. , tx_data_0[2], tx_data_0[1], tx_data_0[0] }

TX GTM Mapping



rx_serdes_header_1[1:0] = { rx_data_1[0], rx_data_1[1] }

rx_serdes_data_1[63:0] = { rx_data_1[2], rx_data_1[3], ……., rx_data_1[65] }

rx_serdes_header_0[1:0] = { rx_data_0[0], rx_data_0[1] }

rx_serdes_data_0[63:0] = { rx_data_0[2], rx_data_0[3], ……., rx_data_0[65] }

RX GTM Mapping

X22033-112618

Table 273: **Supported 40G/50G - GTM Configurations**

Protocol	Lane Width	Line Rate	Encoding	SerDes Width
50GAUI-1[1]	x1	53.125 Gb/s	PAM4	160b
LAUI-2 with soft RSFEC[3]	x2	25.78125 Gb/s	NRZ	64b
LAUI-2 without RSFEC[3]	x2	25.78125 Gb/s	NRZ	64b
XLAUI-1[2][3]	x4	10.3125 Gb/s	NRZ	64b
50GAUI-2 with soft RSFEC	x2	26.5625 Gb/s	NRZ	64b

Notes:

1. Two core options are also available, using two lanes of a single GTM Dual.
2. XLAUI-1 uses two GTM duals.
3. Not yet available.

IEEE PTP 1588v2 for 40G/50G Subsystem

Overview

This section details the packet timestamping function of the 40G/50G 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 presently supports one-step and two-step IEEE PTP 1588 functionality. One-step operation is described in this appendix for reference.

Ethernet frames are timestamped at both ingress and egress. The option can be used for implementing all kinds of IEEE 1588 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 appendix assumes that you are also implementing the IEEE 1588 Precision Time Protocol (PTP).

IEEE 1588 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, then this is known as 1-step operation. Alternatively, the timestamp value can be placed into a follow up packet; this is known as 2-step operation.

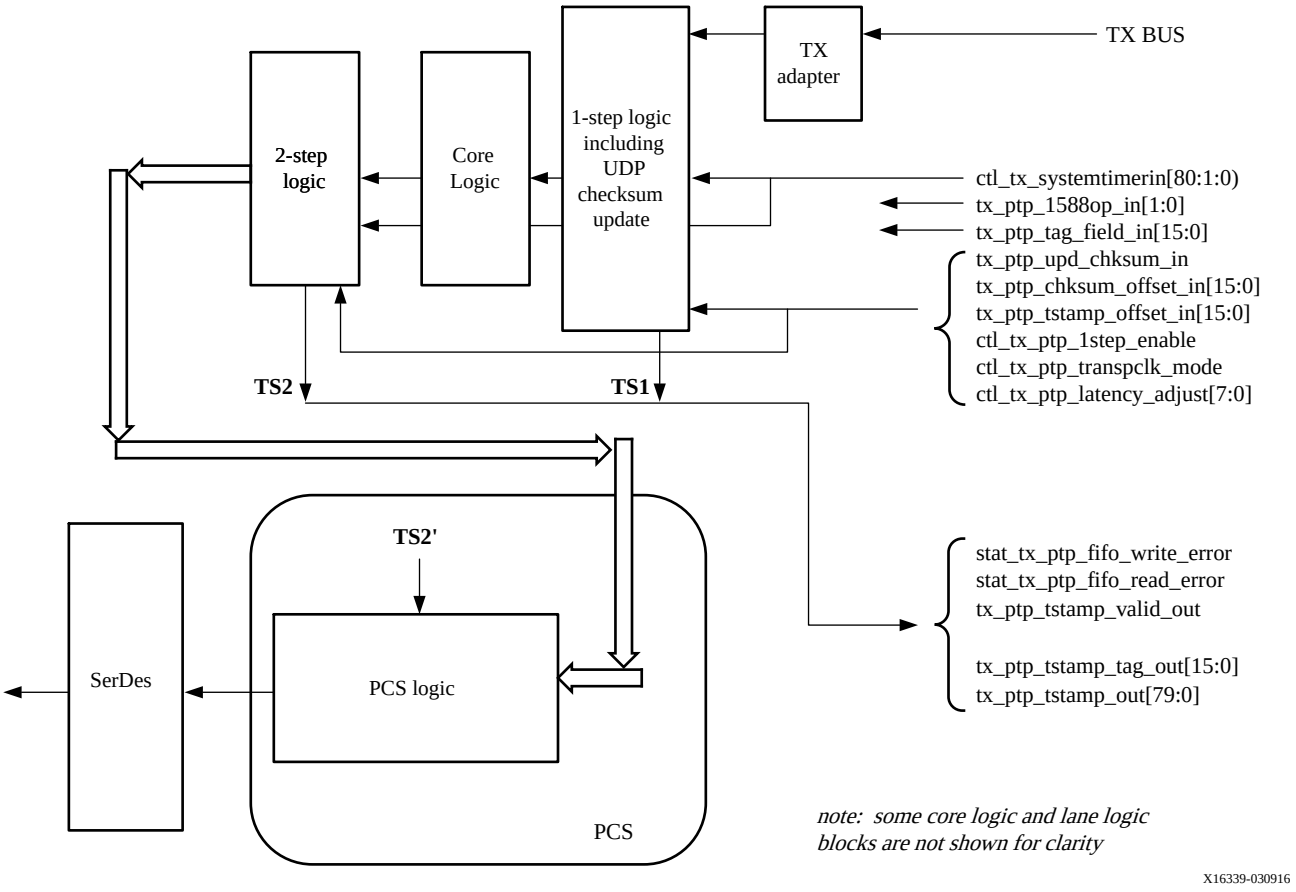
Other timing slave devices on the network receive these timing reference packets from the network timing master and attempt to synchronize their own local timer references to it. This mechanism relies on these Ethernet ports also taking timestamps (samples of their own local timer) when the 1588 timing packets are received. Further explanation of the operation of 1588 is out of the scope of this document. It is assumed that the reader is familiar with the IEEE 1588 specification for the rest of this section.

The 1588 timer provided to the subsystem and the consequential timestamping taken from it are available in one of two formats which are selected during subsystem generation.

- **Time-of-Day (ToD) format:** IEEE 1588-2008 format consisting of an unsigned 48-bit second field and a 32-bit nanosecond field.
- **Correction Field format:** IEEE 1588-2008 numerical format consisting of a 64-bit signed field representing nanoseconds multiplied by 2^{16} (see IEEE 1588 clause 13.3.2.7). This timer should count from 0 through the full range up to $2^{64\ -1}$ before wrapping around.

Egress

Figure 31: **Egress**



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

The TS references are defined as follows:

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

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

Following are the latency adjust values.

- 50G ordinary Clock = 450
- 50G Transparent Clock = 549
- 40G Ordinary Clock = 605
- 40G Transparent Clock = 663
- 40G 256-bit Regular AXI4-Stream Ordinary Clock = 620
- 40G 256-bit Regular AXI4-Stream Transparent Clock = 725

On the transmit side, a command field is provided by the client to the subsystem in parallel with the frame sent for transmission. This indicates, on a frame-by-frame basis, the 1588 function to perform (either no-operation, 1-step, or 2-step) and also indicates, for 1-step frames, whether there is a UDP checksum field to update.

If using the ToD format, then for both 1-step and 2-step operation, the full captured 80-bit ToD timestamp is returned to the client logic using the additional ports defined in the 1588v2 Port List and Descriptions table. 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 the 1588v2 Port List and Descriptions table (with the upper bits of data set to zero as defined in the table).

If using the ToD format, then for 1-step operation, the full captured 80-bit ToD timestamp is inserted into the frame. If using the Correction Field format, then for 1-step operation, the captured 64-bit timestamp is summed with the existing Correction Field contained within the frame and the summed result is overwritten into the original Correction Field of the frame.

Supported frame types for 1-step timestamping are:

- Raw Ethernet

- UDP/IPv4
- UDP/IPv6

For 1-step UDP frame types, the UDP checksum is updated in accordance with IETF RFC 1624. For all 1-step frames, the Ethernet Frame Check Sequence (FCS) field is calculated after all frame modifications have been completed. For 2-step transmit operation, all Precision Time Protocol (PTP) frame types are supported.

The operational mode of the egress timestamping function is determined by the settings on the 1588 command port. The information contained within the command port indicates one of the following:

- No operation: the frame is not a PTP frame and no timestamp action should be taken.
- Two-step operation is required and a tag value (user-sequence ID) is provided as part of the command field; the frame should be timestamped, and the timestamp made available to the client logic, along with the provided tag value for the frame. The additional MAC transmitter ports provide this function.
- 1-step operation is required
  - For the ToD timer and timestamp format a timestamp offset value is provided as part of the command port; the frame should be timestamped, and the timestamp should be inserted into the frame at the provided offset (number of bytes) into the frame.
  - For the Correction Field format, a Correction Field offset value is provided as part of the command port; the frame should be timestamped, and the captured 64-bit Timestamp is summed with the existing Correction Field contained within the frame and the summed result is overwritten into original Correction Field of the frame.

For 1-step operation, following the frame modification, the CRC value of the frame should also be updated/recalculated. For UDP IPv4 and IPv6 PTP formatted frames, the checksum value in the header of the frame needs to updated/recalculated.

For 1-step UDP frame types, the UDP checksum is updated in accordance with IETF RFC 1624.

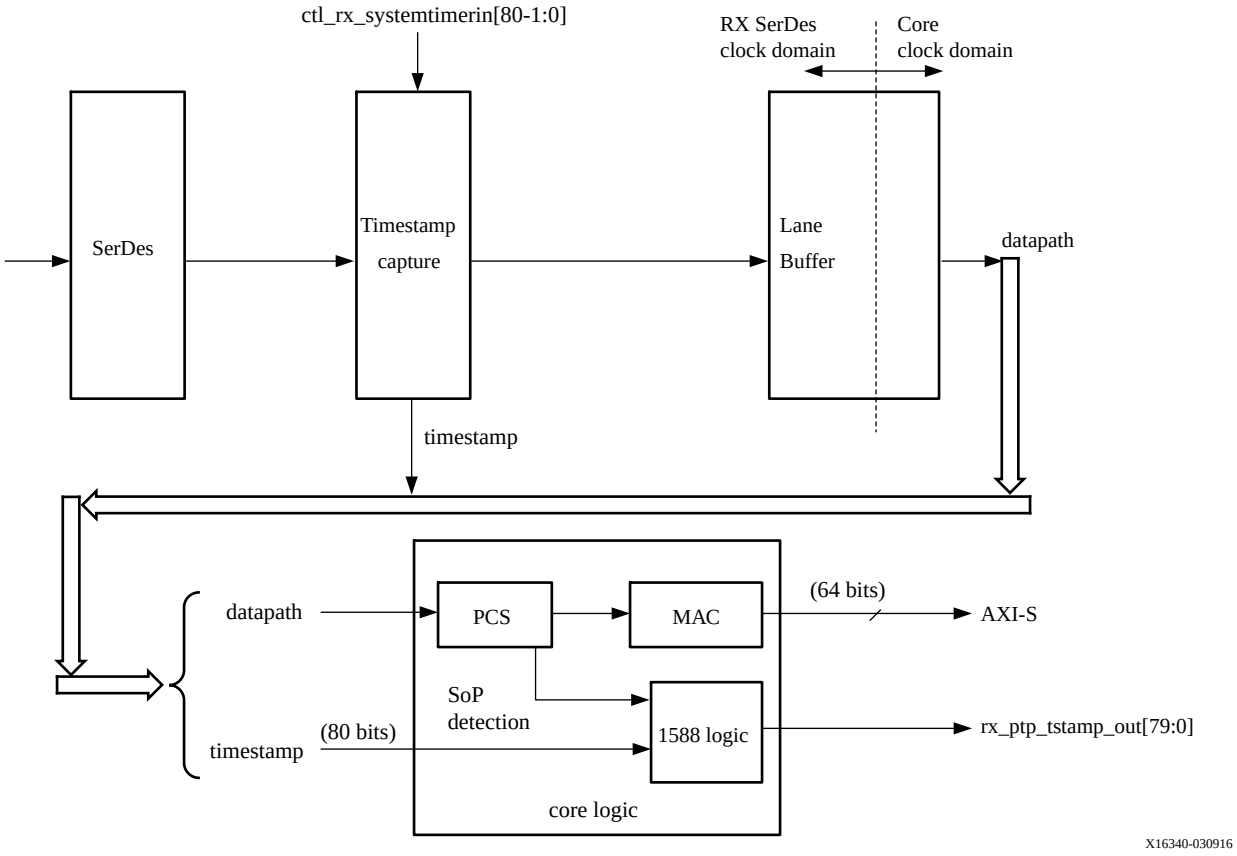
- If using the ToD format, in order for this update function to work correctly, the original checksum value for the frame sent for transmission should be calculated using a zero value for the timestamp data. This particular restriction does not apply when using the Correction Field format.
- If using the Correction Field format then a different restriction does apply; the separation between the UDP Checksum field and the Correction Field within the 1588 PTP frame header is a fixed interval of bytes, supporting the 1588 PTP frame definition. This is a requirement to minimize the latency through the MAC because both the checksum and the correction field must both be fully contained in the MAC pipeline in order for the checksum to be correctly updated. This particular restriction does not apply to the ToD format since the original timestamp data is calculated as a zero value; consequently the checksum and timestamp position can be independently located within the frame.

Related Information

1588v2 Port List and Descriptions

Ingress

Figure 32: **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 are not requiring this function.

Timestamps are filtered after the PCS decoder to retain only those timestamps corresponding to an SOP. These 80-bit timestamps are output on the system side. The timestamp is valid during the SOP cycle and when ena_out = 1.

1588v2 Port List and Descriptions

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

Table 274: **1588v2 Port List and Descriptions**

Signal	I/O	Description	Clock Domain
IEEE 1588 Interface – TX Path			
ctl_tx_systemtimerin[80-1:0]	I	System timer input for the TX. In normal clock mode, the 32 LSBs carry nsec and the 48 MSBs carry seconds. In transparent clock mode, bits 62:16 carry nanoseconds, and bits 15:0 carry fractional nanoseconds. Refer to IEEE 1588v2 for the representational definitions. This input must be in the TX SerDes clock domain.	tx_serdes_clk
tx_ptp_tstamp_valid_out	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	Time stamp for the transmitted packet SOP corresponding to the time at which it passed the capture plane. Used for 2-step 1588 operation. Time format is the same as timer input.	tx_clk_out
tx_ptp_1588op_in[1:0]	I	This signal should be valid on the first cycle of the packet. 2'b00 – No operation: no timestamp will be taken and the frame will not be modified. 2'b01 – 1-step: a timestamp should be taken and inserted into the frame. 2'b10 – 2-step: a timestamp should be taken and returned to the client using the additional ports of 2-step operation. The frame itself will not be modified. 2'b11 – Reserved: act as No operation	tx_clk_out
ctl_tx_ptp_1step_enable	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 will add the correction value to the TX timestamp according to the process defined in IEEE 1588v2. It is expected that the corresponding incoming PTP packet correction field has already been adjusted with the proper RX timestamp.	tx_clk_out
rx_ptp_tstamp_valid_out	O	This bit indicates that a valid time stamp is being presented on the RS system interface. ***Note*:** This is valid for a 256-bit Regular Streaming interface data path only.	
tx_ptp_tag_field_in[15:0]	I	The usage of this field is dependent on the 1588 operation. This signal should be valid on the first cycle of the packet. • For No operation, this field is 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 274: **1588v2 Port List and Descriptions** *(cont'd)*

Signal	I/O	Description	Clock Domain
ctl_tx_ptp_latency_adjust[10:0]	I	This bus can be used to adjust the 1-step TX timestamp with respect to the 2-step timestamp. The units of bits [10:3] are nanoseconds and bits [2:0] are fractional nanoseconds.	tx_clk_out
stat_tx_ptp_fifo_write_error	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
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	Time stamp for the received packet SOP corresponding to the time at which it passed the capture plane. The signal will be valid on the first cycle of the packet.	rx_clk_out
tx_ptp_upd_chksum_in	I	See tx_ptp_upd_chksum_in in IEEE 1588 TX/RX Interface Control /Status /Statistics Signals.	
tx_ptp_pcslane_out	O	See tx_ptp_pcslane_out in IEEE 1588 TX/RX Interface Control /Status /Statistics Signals.	
tx_ptp_chksum_offset_in	I	See tx_ptp_chksum_offset_in in IEEE 1588 TX/RX Interface Control /Status /Statistics Signals.	
rx_ptp_pcslane_out	O	See rx_ptp_pcslane_out in IEEE 1588 TX/RX Interface Control /Status /Statistics Signals.	
rx_lane_aligner_fill	O	See rx_lane_aligner_fill in IEEE 1588 TX/RX Interface Control /Status /Statistics Signals.	

Related Information

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

IEEE PTP 1588v2 Functional Description

The IEEE 1588 feature of the 40G/50G 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 (system timer) defined previously. However, it is required that this time source be in the same clock domain as the SerDes. You might be required to re-time using an external circuit.

In a typical application, the PTP algorithm (or servo, not part of this IP) will remove timestamp errors over the course of time (many packet samples). It is advantageous for the error to be as small as possible to minimize the convergence time as well as minimizing slave clock drift. PTP packets are typically transmitted about 10 times per second.

All ingress frames receive a timestamp. It is up to you to interpret the received frames and determine whether a particular frame contains PTP information (by means of its Ethertype) and if the timestamp needs to be retained or discarded.

Egress frames are timestamped if they are tagged as PTP frames. The timestamps of egress frames are matched to their user-supplied tags.

Timestamps for incoming frames are presented at the user interface in parallel with the AXI4-Stream cycle corresponding to 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 40G/50G High Speed Ethernet Subsystem. This is illustrated in the following schematic diagrams:

Figure 33: **Receive**

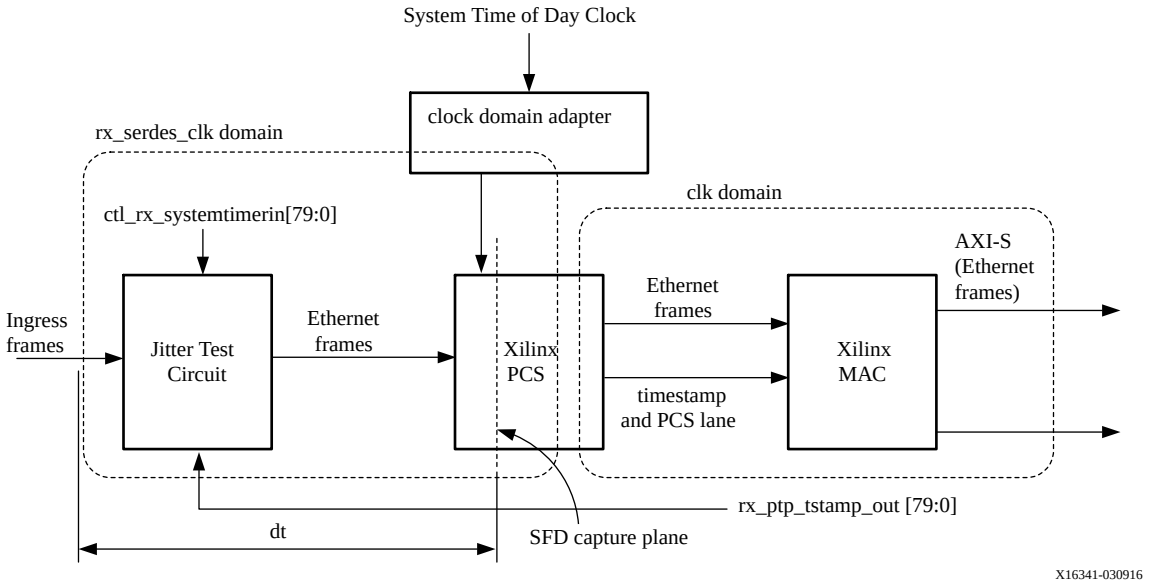
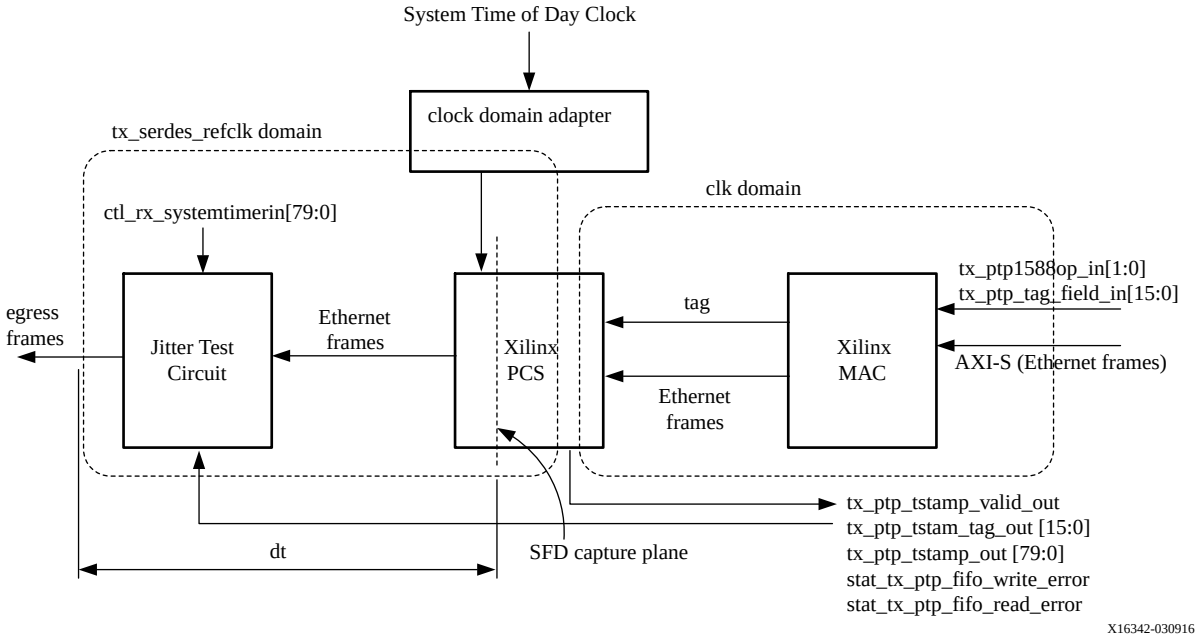


Figure 34: **Transmit**



Performance

Performance of the timestamping logic is tested as shown in the previously referenced diagrams. On the RX side, the jitter test circuit takes note of the system timer at exactly the time when a start-of-frame packet enters the test circuit. Some time later, the timestamp is captured by the RX PCS and is eventually output on the system side AXI4-Stream interface (rx_ptp_tstamp_out[79:0]). The variation of the difference between these two time captures, dt, is defined as the "jitter" performance of the timestamping logic. The TX test is similar for tx_ptp_tstamp_out[79:0].

The 40G/50G subsystem timestamping logic is theoretically capable of determining the time of crossing the SOP capture plane to within the granularity of the 80-bit system timer input. Therefore, if the system timer has a 1 nsec period, the timestamp will be accurate to within a jitter of 1 nsec. 1 nsec is also the granularity of the least significant bit of the 80-bit field as defined by IEEE 1588.

In practice, additional factors will limit the accuracy achievable in a real system.

Clock Domain

In a practical sense, the system timer input is required to have a granularity of the SerDes clock. Therefore, the clock domain crossing of the system timer input should be taken into account. For example, if the SerDes clock has a frequency of 390 MHz, the system timer will have an actual granularity of 2.56 ns, which is also the clock which captures the timestamp. Hence an additional variation of 2.56 ns can be expected.

Transceiver

The addition of a SerDes in the datapath does not impact the jitter performance of the 40G/ 50G subsystem but might result in asymmetry.

In a 1588 clock application, the RX + TX SerDes latency becomes part of the loop delay and is therefore measured by the 1588 protocol. For maximum accuracy of the slave clock it is desirable to take loop asymmetry into account (the difference between the RX and TX SerDes latencies). Xilinx can provide the transceiver latency for various settings of the SerDes specific to your device. You need to contact the vendor of the other transceiver in a datapath if necessary for its characteristics, if you wish to take asymmetry into account in your PTP system.

UltraScale™ and UltraScale+™ transceivers have the ability to report the RX latency. Variation of the RX transceiver latency is mainly due to the fill level of its internal elastic buffer. Refer to the transceiver guide for more details.

Forward Error Correction

Forward Error Correction (for both Clause 74 and Clause 91) takes place on the line side of the timestamp capture. Therefore, the addition of FEC will not impact the accuracy of the timestamp capture in a 40G/50G subsystem. Similar to the SerDes case discussed previously, the additional total (RX + TX) latency of the FEC will be measured by the 1588 protocol.

(Note that the SOP is not visible in a transmitted FEC frame until it has been decoded by the RX FEC function.)

For maximum 1588 slave clock accuracy, it is useful to know the asymmetry of the FEC latency in the RX and TX directions. Contact Xilinx for RX and TX latency of the specific FEC and its configuration. You might also need to obtain this information from the vendor of the link partner in the PTP system if it is not a Xilinx FEC implementation.

Receive Skew Correction

In a multi-lane system such as 40G and 50G, the packet corresponding to the SOP can occur on any lane. Furthermore, lanes can have skew relative to each other. The 40G/50G subsystem provides the ability to take the arrival lane of an SOP frame into account by reporting the SOP lane and its skew. Correction can be performed by hardware or software. The recommended steps are as follows.

Consider the example cases below, for 40G and 50G. The procedure is the same for both except that the number of PMD lanes is 4 and 2 respectively.

Figure 35: **40G Fill Level Correction Example**

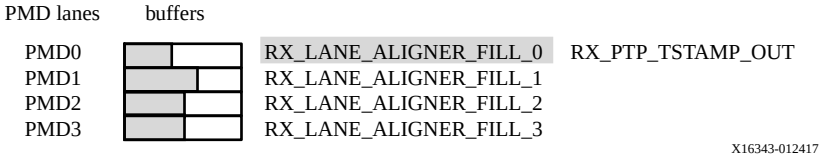


Figure 36: **50G Fill Level Correction Example**

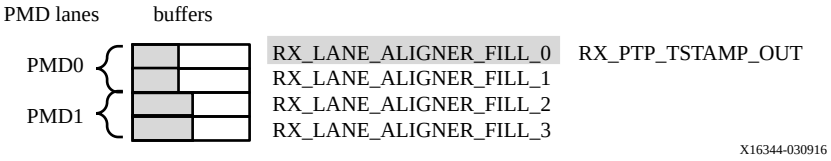


Figure 37: **40G Skew Correction Example**

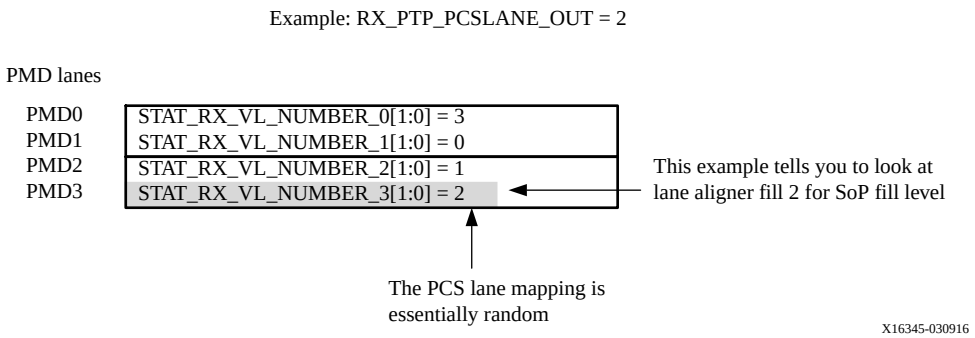


Figure 38: **50G Skew Correction Example**

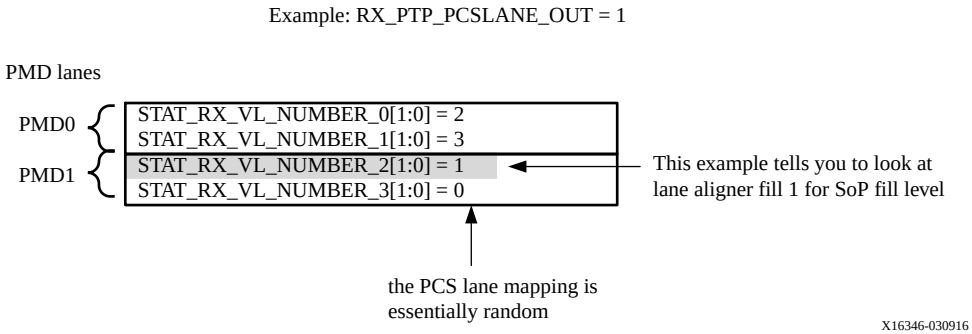
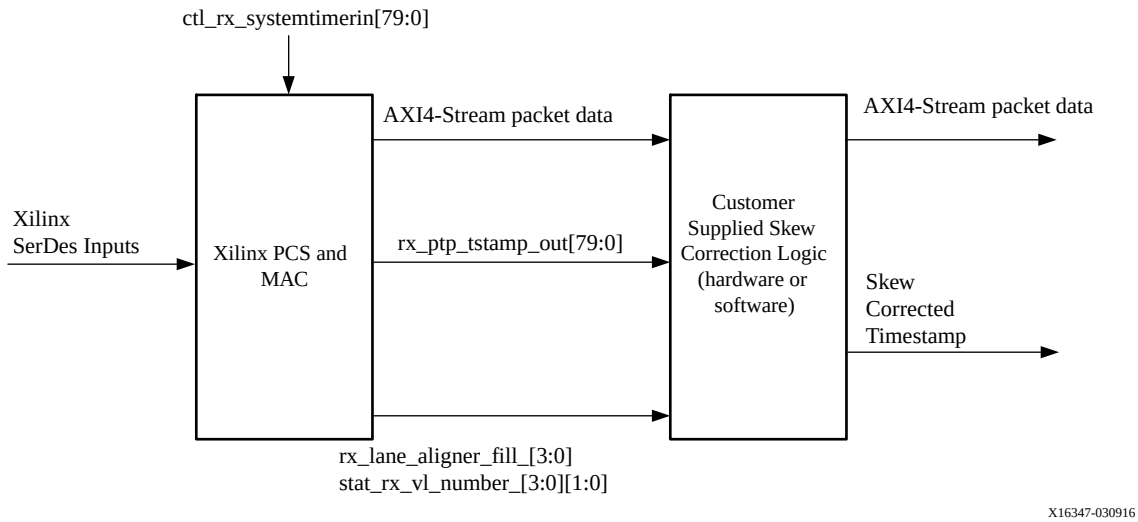


Figure 39: **Timestamp Skew Correction Logic**



The first step is to take a time average of the alignment buffer fill levels because the granularity of these signals is one SerDes clock cycle. While the skew remains relatively constant over time (for example, minutes or hours), the alignment buffer levels have short-term fluctuations of SerDes clock cycles due to sampling quantization. Therefore the actual skew can be obtained to a high degree of accuracy (for example, sub nanoseconds) by taking a time average of each of the fill levels.

Assuming the fill levels have been accurately determined as above, the following formula is used to correct for skew:

```
correction = ((RX_LANE_ALIGNER_FILL_n) - (RX_LANE_ALIGNER_FILL_0)) * SerDes
clock period
corrected timestamp = RX_PTP_TSTAMP_OUT + correction
```

where,

corrected timestamp is the skew-corrected timestamp, which is required to be kept in step with the corresponding packet data

RX_PTP_TSTAMP_OUT is the captured timestamp.

RX_LANE_ALIGNER_FILL_0 is the alignment buffer fill level for the lane on which the timestamp was taken, usually lane 0 (check with Xilinx technical sales support for updates).

RX_LANE_ALIGNER_FILL_n is the alignment buffer fill level for the lane containing the SOF.

The units of all the calculations need to be consistent. Because fill levels are provided in terms of clock cycles, they might have to be converted to nanoseconds or whatever units are consistent with the calculation.

For additional information, see the IEEE Standard 1588-2008, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems" (standards.ieee.org/findstds/standard/1588-2008.html)

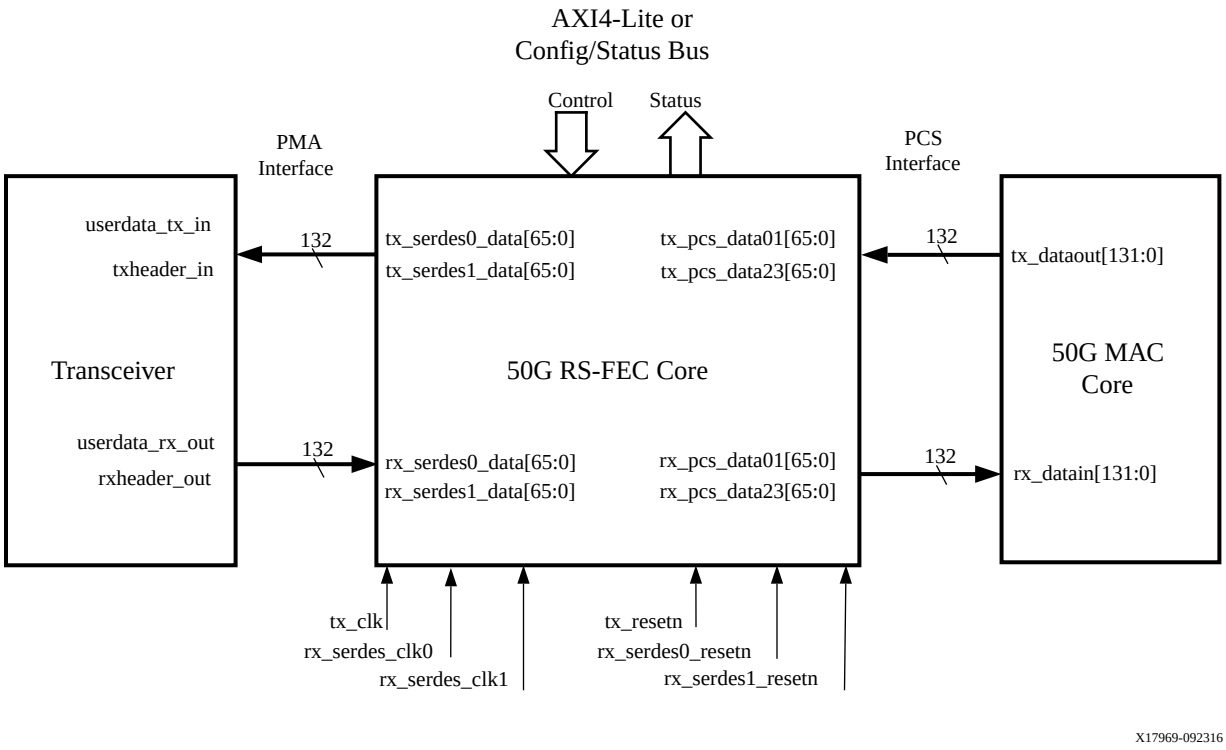
RS-FEC Support

This section describes the optional RS-FEC function of the 50G 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.

The RS-FEC block is positioned between the PCS and PMA as illustrated in 40G/50G MAC with PCS/PMA Clocking.

With reference to the following diagram, the RS-FEC core clocks and resets are equivalent to the transceiver signals, with the transceiver resets being active-High.

Figure 40: **RFC Block Position**



RS-FEC Functional Description

The 50G subsystem RS-FEC feature provides error correction capability according to 50G FEC (Ethernet Consortium Schedule 3 specification, based on IEEE 802.3 Clause 91).

The feature requires the insertion of PCS alignment markers as defined in IEEE 802.3 Table 82-3. The number of words between the PCS alignment markers is 20,480.

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 the 50G IEEE 802.3 Reed-Solomon Forward Error Correction *50G IEEE 802.3 Reed-Solomon Forward Error Correction LogiCORE IP Product Guide* (PG234) (registration required) for the latest latency performance data in the various bypass modes.

The following feature bypass modes are selectable.

- FEC Bypass Correction: The decoder performs error detection without correction, (see IEEE Std 802.3 section 91.5.3.3 [Ref 1]). The latency is reduced in this mode.
- 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 codewords (see IEEE Std 802.3 section 91.5.3.3 [Ref 1]). 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 are in bypass mode.

RS-FEC Configuration and Status Port List and Descriptions

The following table describes the configuration and status ports when the RS-FEC feature is included.

Table 275: **RS-FEC Configuration and Status Port List and Descriptions**

Name	I/O	Description	Clock Domain
ctl_rsfec_enable	I	Changes only take effect after the reset. A new value is sampled on the first cycle after reset. Enable RS-FEC function. 1= Enable RS-FEC 0= Bypass RS-FEC	rx_serdes_clk
ctl_rx_rsfec_enable_correction	I	Changes only take effect after the reset. A new value is sampled on the first cycle after reset. Equivalent to MDIO register 1.200.0 • 0: Decoder performs error detection without error correction (see IEEE 802.3802.3 by section 91.5.3.3). • 1: the decoder also performs error correction.	rx_serdes_clk

Table 275: **RS-FEC Configuration and Status Port List and Descriptions** *(cont'd)*

Name	I/O	Description	Clock Domain
ctl_rx_rsfec_enable_indication	I	Changes only take effect after the reset. A new value is sampled on the first cycle after reset. Equivalent to MDIO register 1.200.1 • 0: Bypass the error indication function (see IEEE Std 802.3 by section 91.5.3.3). • 1: Decoder indicates errors to the PCS sublayer	rx_serdes_clk
ctl_rsfec_ieee_error_indication_mode	I	Changes only take effect after the reset. A new value is sampled on the first cycle after reset. • 1: Core conforms to the IEEE RS-FEC specification • 0: If ctl_rx.rsfec_enable_correction and ctl_rx_rsfec_enable_indication are 0, the RS decoder is bypassed.	rx_serdes_clk
stat_tx_rsfec_pcs_block_lock	O	TX PCS block lock status 0=unlocked 1=locked	tx_clk
stat_tx_rsfec_lane_alignment_ status	O	TX PCS frame alignment status 0=unaligned 1-aligned	tx_clk
stat_rx_rsfec_am_lock0	O	RX lane 1 lock status 0= unlocked 1= locked	rx_serdes_clk
stat_rx_rsfec_am_lock1	O	RX lane 1 lock status 0= unlocked 1= locked	rx_serdes_clk
stat_rx_rsfec_lane_alignment_ status	O	RX alignment status 0=unaligned 1=aligned	rx_serdes_clk
stat_rx_rsfec_lane_fill_0[13:0]	O	RX lane 0 additional delay. The top seven bits [13:7] of each delay bus is the number of additional clock cycles delay being added due to deskew. The bottom seven bits [6:0] of each delay bus is the fractional clock cycle delay being added due to deskew, in units of 1/66th of a clock cycle.	rx_serdes_clk
stat_rx_rsfec_lane_fill_1[13:0]	O	RX lane 1 additional delay. The top seven bits [13:7] of each delay bus is the number of additional clock cycles delay being added due to de-skew. The bottom seven bits [6:0] of each delay bus is the fractional clock cycle delay being added due to de-skew, in units of 1/66th of a clock cycle.	rx_serdes_clk
stat_rx_rsfec_lane_mapping [1:0]	O	RX lane mapping bit 0= index of FEC lane carried on PMA lane 0 bit 1= index of FEC lane carried on PMA lane 1	rx_serdes_clk
stat_rx_rsfec_hi_ser	O	This output is only active when the core is in bypass indication mode. It indicates when High that the number of FEC symbol errors in a window of 8192 codewords has exceeded the threshold K (417)	rx_serdes_clk
stat_rx_rsfec_corrected_cw_inc	O	Corrected codeword count increment	rx_serdes_clk
stat_rx_rsfec_uncorrected_cw_ inc	O	Uncorrected codeword count increment	rx_serdes_clk
stat_rx_rsfec_symbol_error_ count0_inc[2:0]	O	Symbol error count increment for lane 0.	rx_serdes_clk

Table 275: **RS-FEC Configuration and Status Port List and Descriptions** *(cont'd)*

Name	I/O	Description	Clock Domain
stat_rx_rsfec_symbol_error_count1_inc[2:0]	O	Symbol error count increment for lane 1.	rx_serdes_clk

Status/Control Interface

The Status/Control interface allows you to set up the 40G/50G High Speed Ethernet Subsystem configuration and to monitor the status of the 40G/50G High Speed Ethernet Subsystem. The following subsections describe in more detail some of the various Status and Control signals.

RX and TX PCS Lane Marker Values

The IEEE Std 802.3 defines the PCS Lane marker values, shown in the following table.

Table 276: **40G/50G Marker Definitions**

Input Signal Name	Value
ctl_rx_vl_marker_id[0][63:0] ctl_tx_vl_marker_id[0][63:0]	64'h90_76_47_00_6f_89_b8_00
ctl_rx_vl_marker_id[1][63:0] ctl_tx_vl_marker_id[1][63:0]	64'hf0_c4_e6_00_0f_3b_19_00
ctl_rx_vl_marker_id[2][63:0] ctl_tx_vl_marker_id[2][63:0]	64'hc5_65_9b_00_3a_9a_64_00
ctl_rx_vl_marker_id[3][63:0] ctl_tx_vl_marker_id[3][63:0]	64'ha2_79_3d_00_5d_86_c2_00

RX PCS Lane Alignment Status

The 40G/50G High Speed Ethernet Subsystem provides status bits to indicate the state of word boundary synchronization and PCS lane alignment. All signals are synchronous with the rising-edge of `clk` and a detailed description of each signal follows.

stat_rx_synced[3:0]

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

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

stat_rx_synced_err[3:0]

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

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

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

stat_rx_mf_len_err[3:0]

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

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

stat_rx_mf_repeat_err[3:0]

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

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

stat_rx_mf_err[3:0]

When `stat_rx_aligned` is a value of 1, all of the lanes are aligned/deskewed and the receiver is ready to receive packet data.

stat_rx_aligned_err

When stat_rx_aligned_err is a value of 1, one of two things occurred:

- Lane alignment failed after several attempts.
- Lane alignment was lost (stat_rx_aligned was asserted and then it was negated).

stat_rx_misaligned

When `stat_rx_misaligned` is a value of 1, a valid PCS Lane Marker Word was not received on all PCS lanes simultaneously.

This output is asserted for one clock period each time this error condition is detected.

stat_rx_framing_err_[3:0][3:0] and stat_rx_framing_err_valid_[3:0]

This set of buses is intended to be used to keep track of sync header errors. There is a pair of outputs for each PCS Lane. The `stat_rx_framing_err_[PCSL_LANES-3:0]` output bus indicates how many sync header errors were received and it is qualified (that is, the value is only valid) when the corresponding `stat_rx_framing_err_valid_[PCSL_LANES-3:0]` is sampled as a 1.

stat_rx_vl_number[3:0][1:0]

Each bus indicates which PCS lane has its status reflected on specific status pins. For example, `stat_rx_vlane_number_0` indicates which PCS lane has its status reflected on pin 0 of the other status signals.

These buses can be used to detect if a PCS lane has not been found or if one has been mapped to multiple status pins.

stat_rx_vl_demuxed[3:0]

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

stat_rx_block_lock[3:0]

Each bit indicates that the corresponding PCS lane has achieved sync header lock as defined by the IEEE Std 802.3-2015. A value of 1 indicates block lock is achieved.

stat_rx_status

This output is set to a 1 when stat_rx_aligned is a 1 and stat_rx_hi_ber is a 0. This is as defined by the IEEE Std 802.3-2015.

stat_rx_local_fault

This output is set to a 1 when `stat_rx_received_local_fault` or `stat_rx_internal_local_fault` is asserted. This output is level sensitive.

RX Error Status

The 40G/50G High Speed Ethernet Subsystem 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 and a detailed description of each signal follows.

stat_rx_bad_fcs

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

When a CRC32 error is detected, the received packet is marked as containing an error and is sent with `rx_errout` asserted during the last transfer (the cycle with `rx_eopout` asserted) unless `ctl_rx_ignore_fcs` is asserted.

This signal is asserted for one clock period each time a CRC32 error is detected.

stat_rx_bad_code[1:0]

This signal indicates how many cycles the RX PCS receive state machine is in the RX_E state as defined by the IEEE Std 802.3-2015.

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 subsystem 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 IP subsystem 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 the *Vivado Design Suite User Guide: Getting Started* (UG910).

Figures in this chapter are illustrations of the Vivado 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 tabs.

Figure 41: **Configuration Tab**

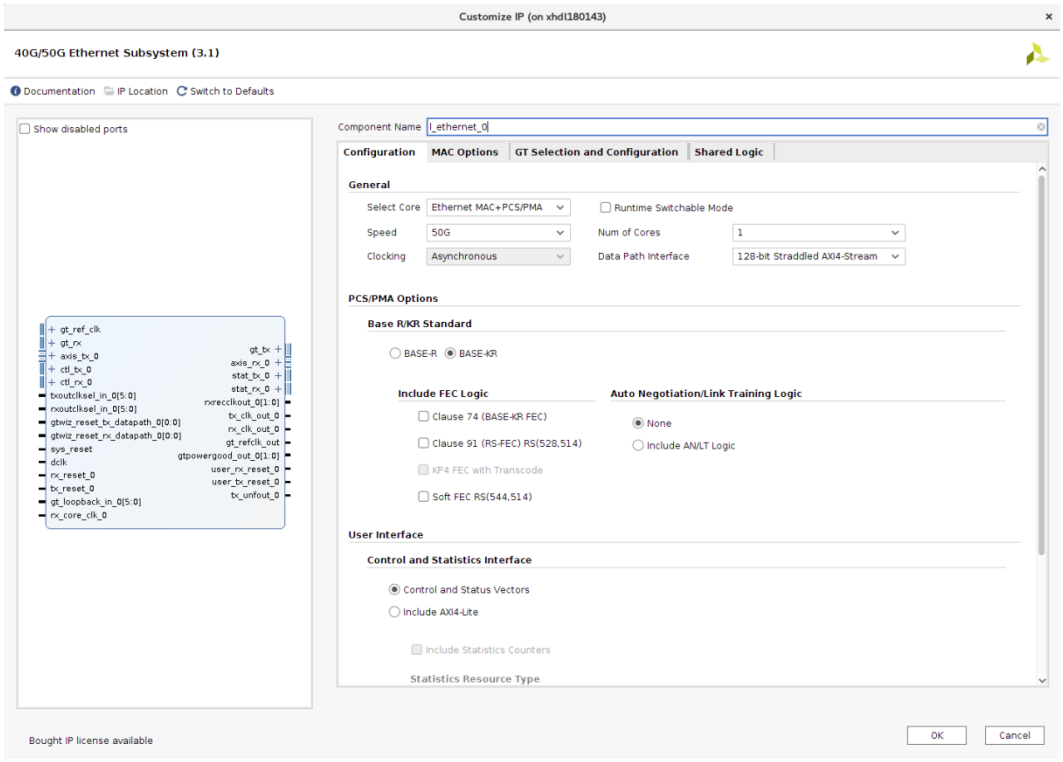


Table 277: **Configuration Options**

Option	Values	Default
General		
Select Core	Ethernet MAC+PCS/PMA Ethernet PCS/PMA	Ethernet MAC+PCS/PMA
Speed	50G 40G	50G
Number of Cores	1 2	1
Clocking	Synchronous Asynchronous	Asynchronous
Runtime Switchable Mode	0,1	0
Data Path Interface	128-bit Straddled AXI4-Stream[1] 256-bit Regular AXI4-Stream[2] MII[3]	128-bit Straddled AXI4-Stream

Table 277: **Configuration Options** *(cont'd)*

Option	Values	Default
PCS/PMA Options		
Base-R Base-KR	Base-R Base-KR	Base-KR
Include FEC Logic		
Clause 74 (BASE-KR FEC) [46]	0,12 mi	0
Clause 91 (RS-FEC)[56]	0,1	0
KP4 FEC with Transcode[7]	0,1	0
Soft RS-FEC (544,514)	0,1	0
Auto Negotiation/Link Training Logic		
Auto Negotiation/Link Training Logic	None Include AN/LT Logic	None
Control and Statistics Interface		
Control and Statistics interface	Control and Status Vectors Include AXI4-Lite	Control and Status Vectors
Include Statistics Counters	0,1[8]	1
Statistics Resource Type	Register, block RAM[9]	block RAM

Notes:

1. The 128-bit Straddled AXI4-Stream is visible and option for Ethernet MAC+PCS/PMA with 40G and 50G line rate options.
2. The 256-bit Regular AXI4-Stream is visible and option for Ethernet MAC+PCS/PMA with 40G Line rate only.
3. The MII interface is visible and is the only option for the Ethernet PCS/PMA core.
4. Clause 74 (BASE-KR FEC) logic is not supported for Base-R.
5. Clause 91 (RS-FEC) is not supported for Base-R, 40G speed.
6. Clause 74 (BASE-KR FEC) and Clause 91 (RS-FEC) both can be selected in Vivado® IDE but during functional operation only one can be enabled at a time using the respective control signals.
7. GTM integrated hard FEC. Only applicable for Devices with GTM.
8. The Statistics Counters are available in the register map only when you enable the Include Statistics Counters option. Otherwise, the Statistics Counters are not available.
9. Statistics Resource Type block RAM option will be provided in the future release.

MAC Options Tab

The MAC Options tab provides additional core configuration options.

Figure 42: **MAC Options Tab**

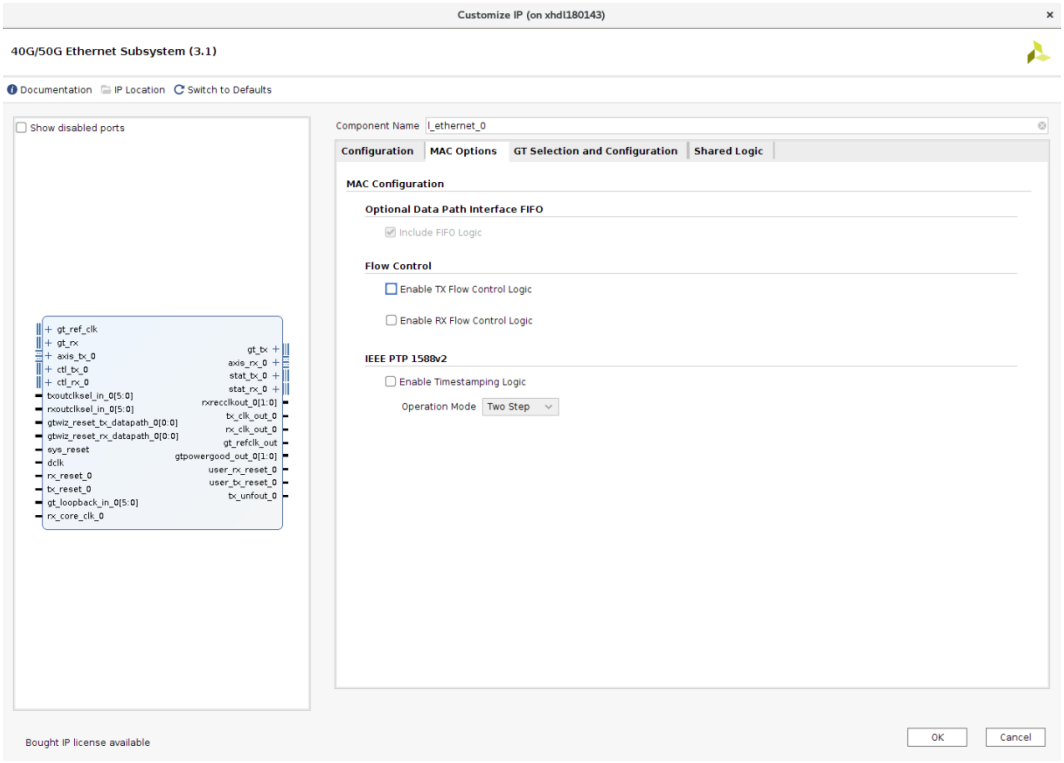


Table 278: **MAC Configuration**

Option	Values	Default
Optional Data Path Interface FIFO		
Include FIFO Logic	0, 1	1
Flow Control		
Enable TX Flow Control Logic	0, 1	0
Enable RX Flow Control Logic	0, 1	0
IEEE PTP 1588v2		
Enable Timestamping Logic	0, 1	0
Operation Mode	Two Step	Two Step

GT Selection and Configuration Tab

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

Figure 43: **GT Selection and Configuration Tab**

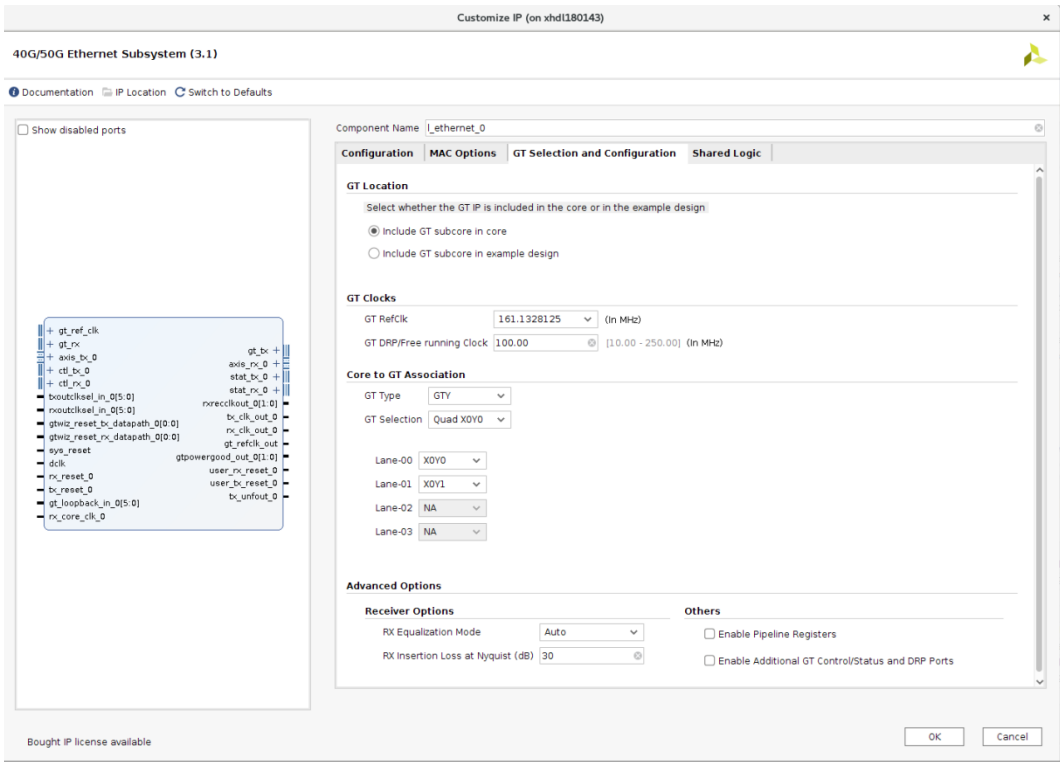


Table 279: **GT Clocks Options**

Option	Values	Default
GT Location		
Select whether the GT IP is included in the core or in the example design	Include GT subcore in core Include GT subcore in example design	Include GT subcore in core
GT Clocks		
GT RefClk (In MHz) [1]	161.1328125	161.1328125
	195.3125	
	201.4160156	
	257.8125	
	322.265625	
GT DRP Clock (In MHz)	50.00 – 175.00 MHz	100.00
Core to Transceiver Association		
GT Type	GTH GTY GTM	GTY

Table 279: **GT Clocks Options** *(cont'd)*

Option	Values	Default
GT Selection	Options based on device/package Quad groups.For example: Quad X0Y1 Quad X0Y2 Quad X0Y3 GTM_DUAL_X0Y0[2] If the GT type is selected as GTM, then the GTM Dual List is available for the selection. For example: GTM_DUAL_X0Y0 GTM_DUAL_X0Y1 ...	Quad X0Y0
Lane-00 to Lane-03	Auto filled based on device/package. For example, if Speed = 50G and Num of Cores = 2 (or Speed = 40G and Num of Cores = 1) and GT selection = Quad X0Y1, then the four GT lanes are: X0Y4 X0Y5 X0Y6 X0Y7	
RX Equalization Mode	Auto LPM DFE	Auto
RX Insertion Loss at Nyquist (dB)	Depends on the GT Wizard	30
Others		
Enable Pipeline Register	0, 1	0
Enable Additional GT Control and Status Ports	0, 1	0

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. Only applicable for GTM. For 40G GTM, two consecutive duals need to be selected.

Shared Logic Tab

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

Figure 44: **Shared Logic Tab**

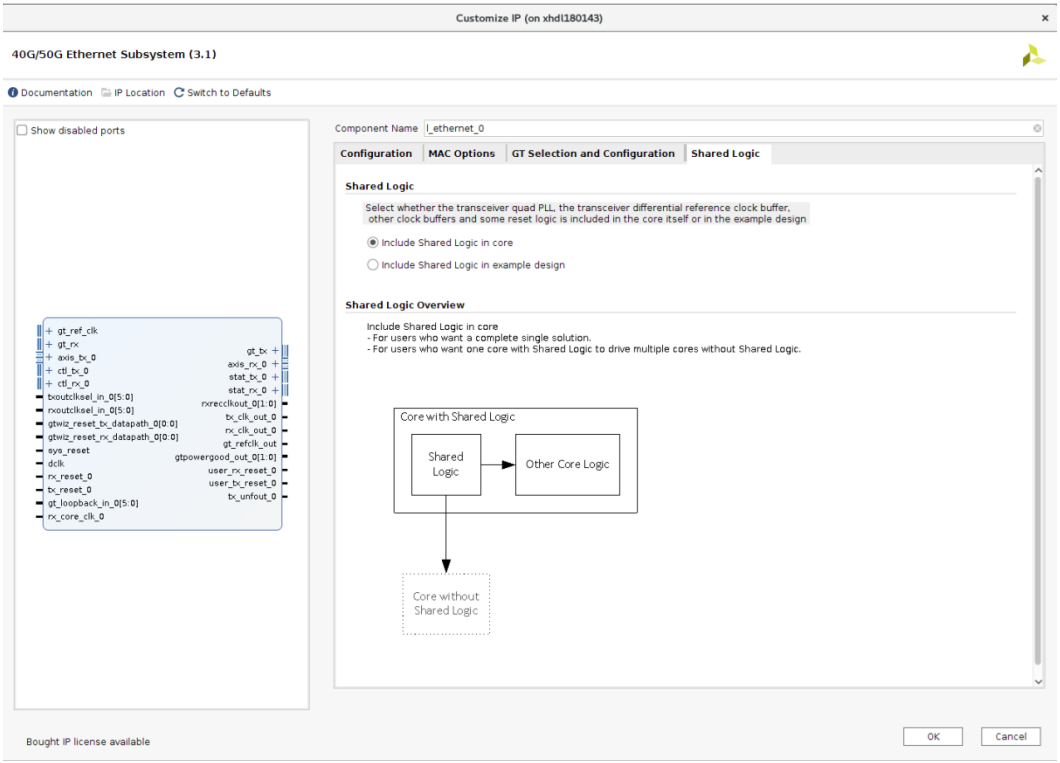


Table 280: **Shared Logic Options**

Options	Default
Include Shared Logic in Core Include Shared Logic in example design	Include Shared Logic in Core

Configuration Spreadsheet

In addition to the IP catalog in the Vivado design tools, the 40G/50G Ethernet Subsystem can be requested for a specific configuration using a customer-provided spreadsheet. You must submit the completed spreadsheet to Xilinx Technical Support to obtain the target netlist. See Synthesis and Implementation for a description of what is delivered and how to proceed with the deliverables.

For a complete understanding of the core, consult the document corresponding to the core and the device selected, such as the *UltraScale Architecture GTH Transceivers User Guide* (UG576).

Output Generation

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

Constraining the Subsystem

Each release includes one or more XDC files specific to your configuration and its clocking. Additional constraints might be required when you incorporate the IP into your design. If more information is required, contact Xilinx Technical Support.

Required Constraints

This section is not applicable for this IP subsystem.

Device, Package, and Speed Grade Selections

This section is not applicable for this IP subsystem.

Clock Frequencies

This section is not applicable for this IP subsystem.

Clock Management

This section is not applicable for this IP subsystem.

Clock Placement

This section is not applicable for this IP 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).

A demonstration simulation test bench is part of each release. Simulation is performed on the included encrypted RTL. The test bench consists of a loopback from the TX side of the user interface, through the TX circuit, looping back to the RX circuit, and checking the received packets at the RX side of the user interface.

The loopback simulation includes a path through the transceiver. The simulation is run using the provided Linux scripts for several common industry-standard simulators.

For more information, see Chapter 7: Test Bench.

Simulation can take a long time to complete due to the time required to complete alignment. A `define SIM_SPEED_UP is available to improve simulation time by reducing the PCS lane Alignment Marker (AM) spacing in order to speed up the time the IP will take to achieve alignment. Setting `define SIM_SPEED_UP will reduce CTL_TX_VL_LENGTH_MINUS1 and CTL_RX_VL_LENGTH_MINUS1.

The SIM_SPEED_UP option can be used for simulation when in serial loopback or if the Alignment Marker spacing can be reduced at both endpoints. This option is compatible with the example design simulation which uses serial loopback.

***Note*:**

- SIM_SPEED_UP is only available when running RTL simulation. It is not available when running simulation with post synthesis or post implementation netlist.
- Altering the value of CTL_TX_VL_LENGTH_MINUS1 and CTL_RX_VL_LENGTH_MINUS1 from the default value will violate the IEEE 802.3 specification.
- Decreasing the AM spacing will result in less bandwidth being available on the link. When using the PCS only core, the TX core does not provide back pressure to the user logic and the reduced bandwidth can cause stat_tx_fifo_error when transmitting larger back-to-back packets.
- This change can be made only in simulation. For a design to work in hardware, the default IEEE value must be used.
- Full rate simulation without the SIM_SPEED_UP option should still be run.

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.

RS-FEC Enabled Configuration Simulation

For faster simulation, apply SIM_SPEED_UP and deselect the **Use Precompiled IP simulation libraries** checkbox in the Settings window, as shown in the following figures. If this is not done, the simulation can run for a long time, timing out with an error.

Figure 45: **Use Precompiled IP Simulation Libraries Disabled**

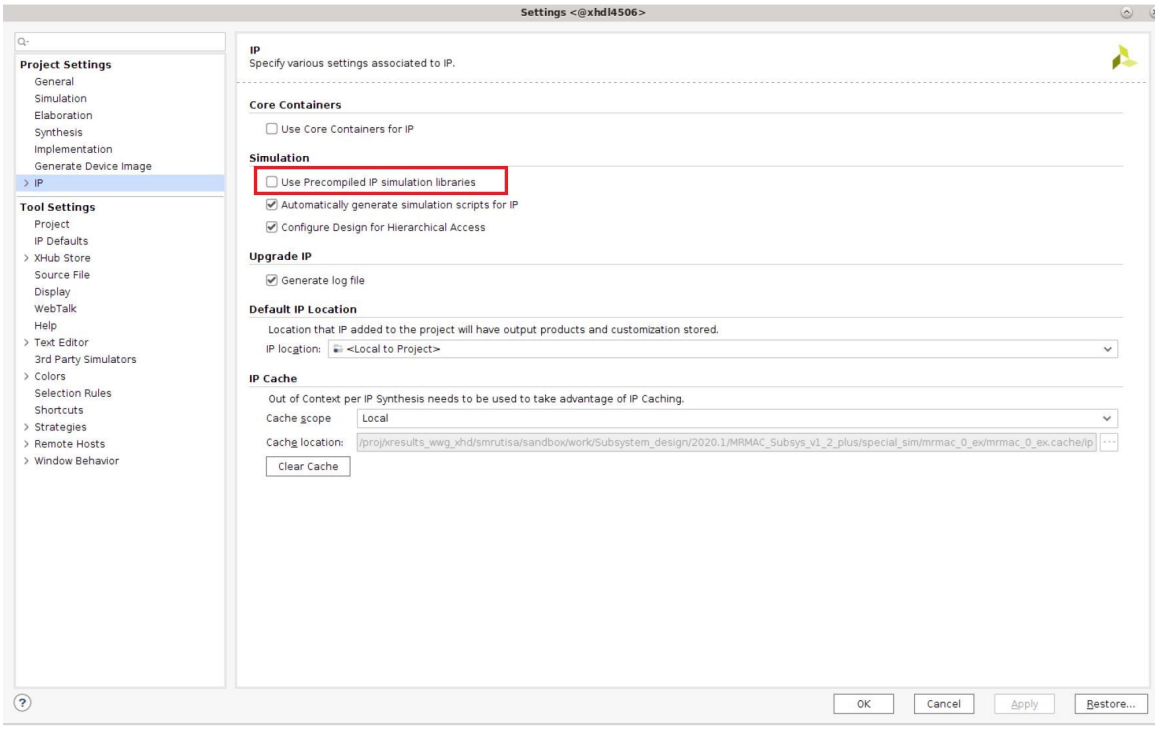
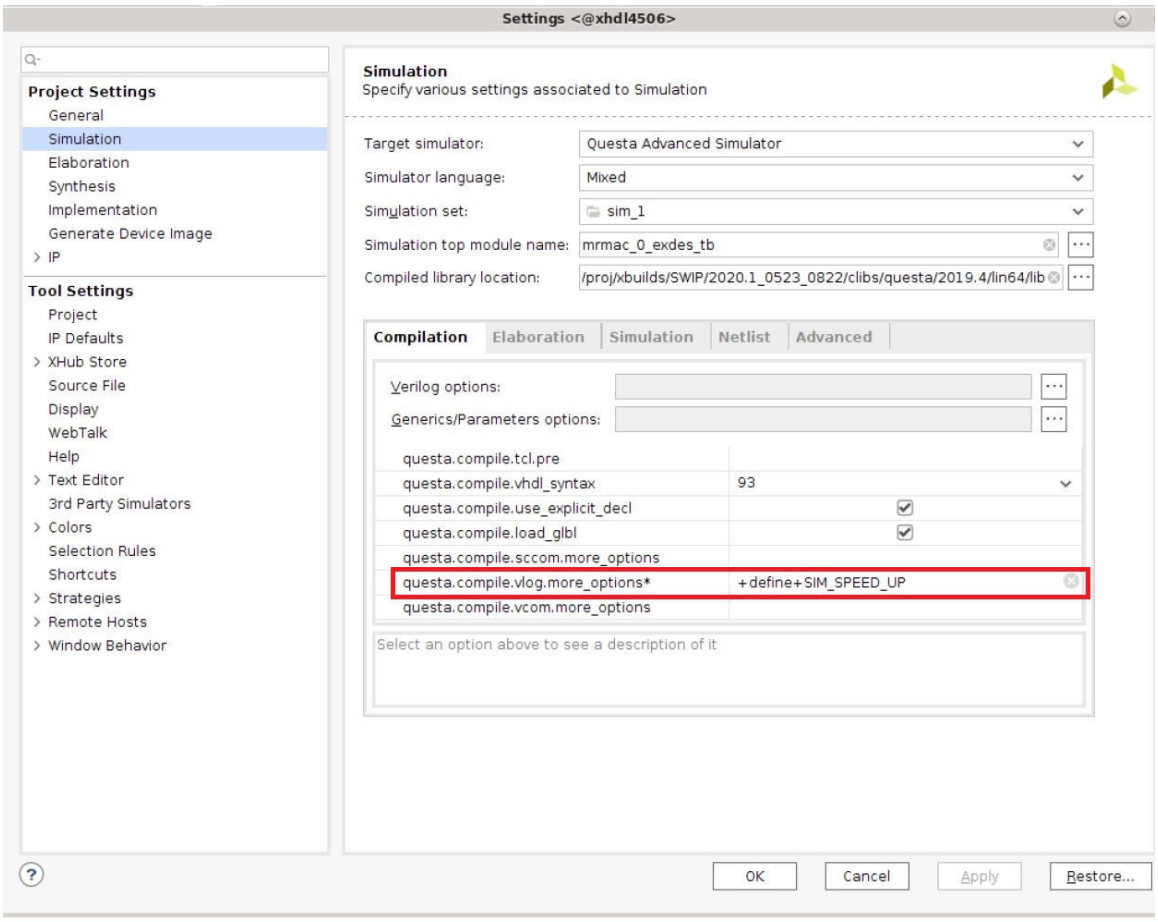


Figure 46: **SIM_SPEED_UP Enabled**



Synthesis and Implementation

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

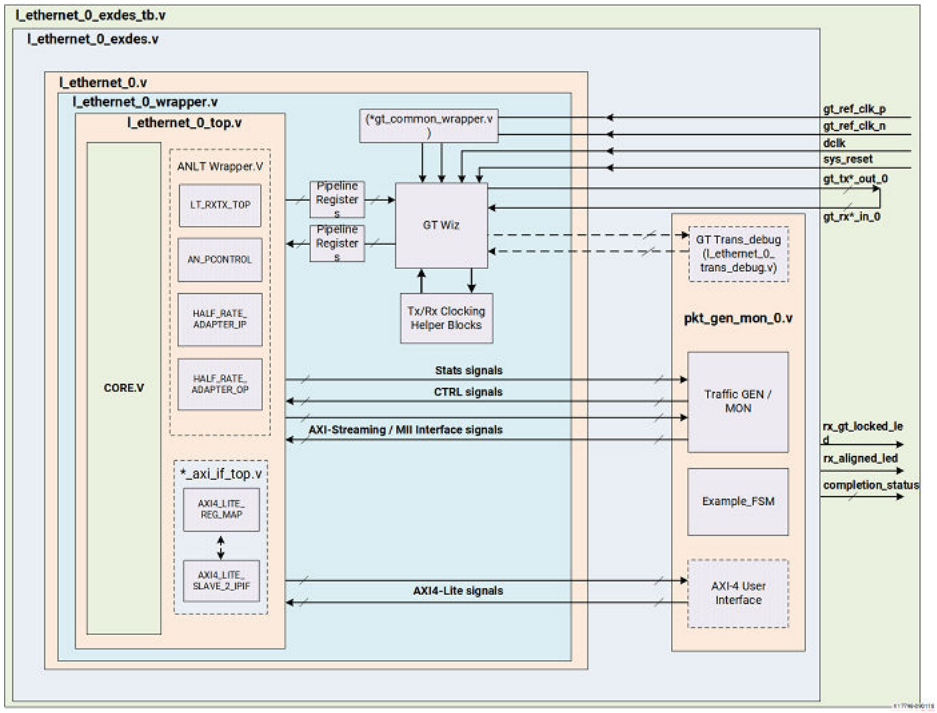
Example Design

This chapter provides a brief explanation of the 40G/50G Ethernet Subsystem example design.

Example Design Hierarchy

The following figure shows the instantiation of various modules and their hierarchy for a single core configuration of the l_ethernet_0 example design when the GT (serial transceiver) is inside the IP core. Retiming registers are used for the synchronization of data between the core and the GT. Clocking helper blocks are used to generate the required clock frequency for the core.

Figure 47: **Single Core Example Design Hierarchy**



Following are the user interfaces available for different configurations:

- MAC/PCS configuration
  - AXI4-Stream for data path interface

    - AXI4-Lite for control and statistics interface
- PCS configuration
    - MII for data path interface
    - AXI4-Lite for control and statistics interface

The l_ethernet_0_pkt_gen_mon module is used to generate the data packets for sanity testing. The packet generation and checking is controlled by a Finite State Machine (FSM) module.

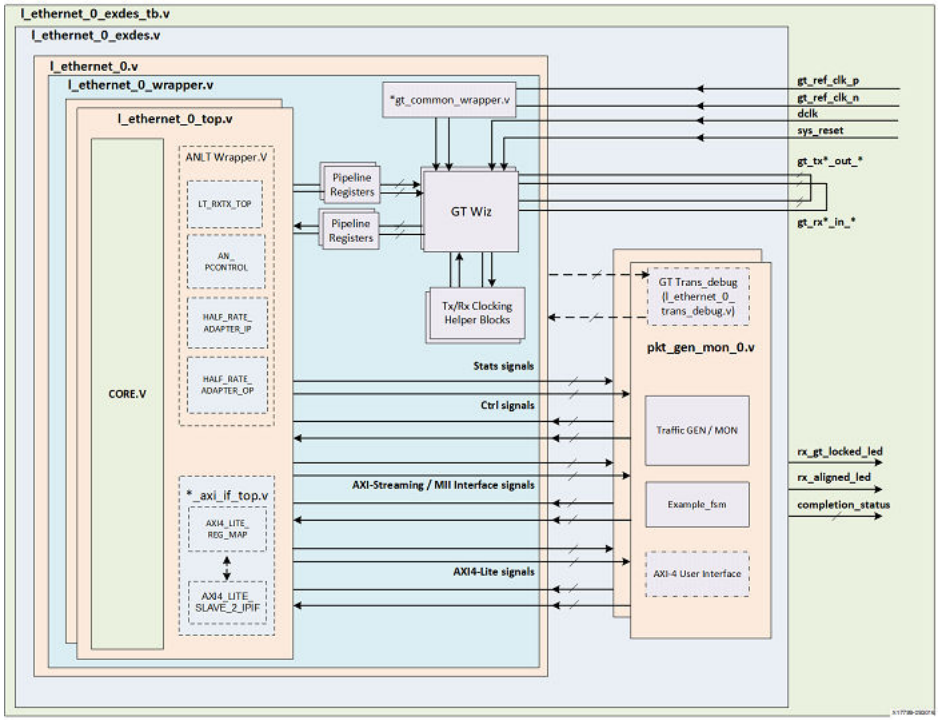
Descriptions of optional modules are as follows:

- **l_ethernet _0_trans_debug:** This module is present in the example design when you enable the **Additional GT Control and Status Ports** check box from the GT Selection and Configuration Tab in the Vivado® Integrated Design Environment (IDE) or the **Include GT subcore in example design** option in the GT Selection and Configuration tab or the **Runtime Switchable mode** option in the in the Configuration Tab. This module brings out all the GT channel DRP ports, and some control and status ports of the transceiver module out of the l_ethernet core.
- **Retiming Registers:** When you select the **Enable Retiming Register** option from the GT Selection and Configuration Tab, it includes a single stage pipeline register between the core and the GT to ease timing, using the gt_txusrclk2 and gt_rxusrclk2 for TX and RX paths respectively. However, by default two-stage registering is done for the signals between the GT and the core.

  Note: For Runtime Switchable, if **Auto Negotiation/Link training** is selected in Vivado IDE, then AN operation will be performed only with the 40G data rate during switchings and LT will be performed in the mission mode.

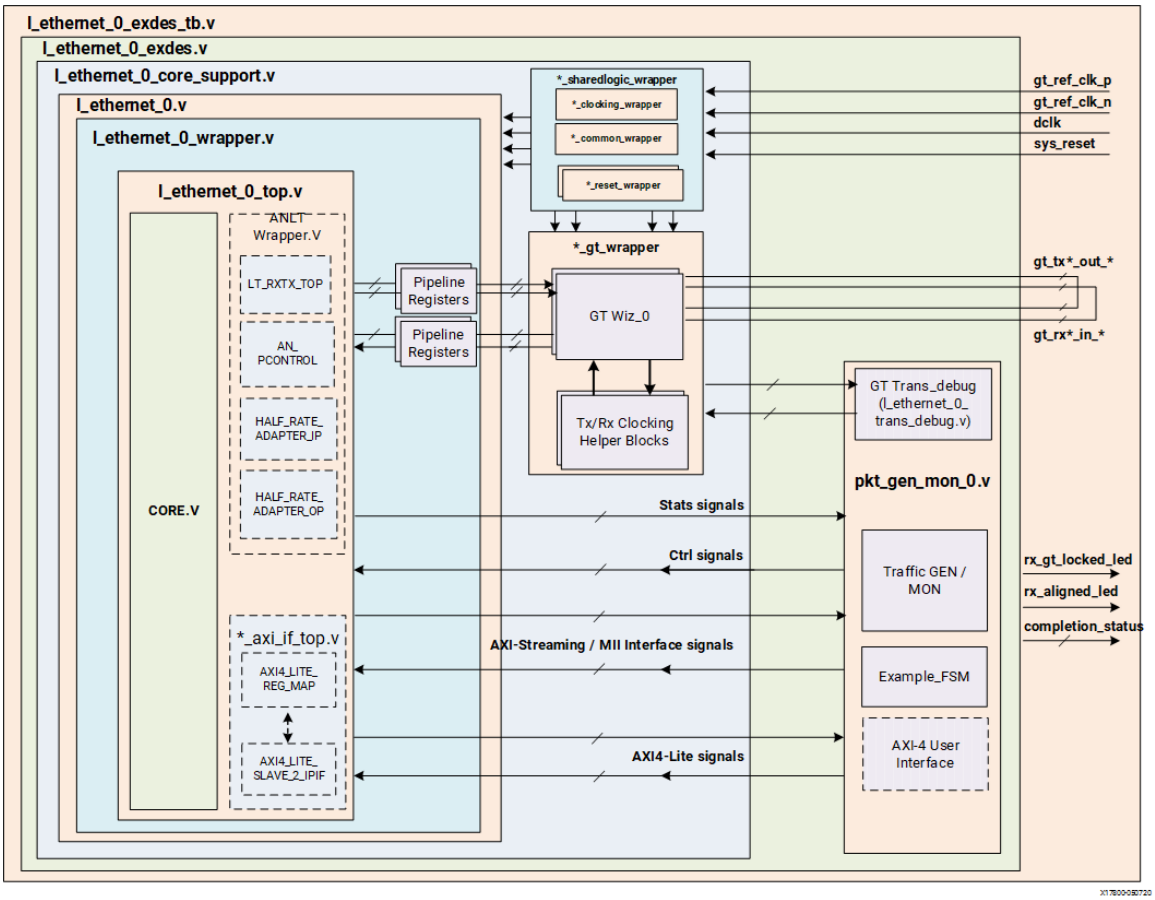
The following figure shows the instantiation of various modules and their hierarchy for the multiple core configuration of the l_ethernet_0 example design.

Figure 48: **Multiple Core Example Design Hierarchy**



Example Design Hierarchy (GT in Example Design)

Figure 49: **Single Core with GT in Example Design Hierarchy**



The previous figure shows the instantiation of various modules and their hierarchy for a single core configuration of the `l_ethernet_0` example design when the GT (serial transceiver) is outside the IP core, that is, in the example design. This hierarchical example design is delivered when you select the Include GT subcore in the example design option from the GT Selection and Configuration tab.

The `l_ethernet_0_core_support.v` is present in the hierarchy when you select the Include GT subcore in example design option from the GT Selection and Configuration tab or the Include Shared Logic in example design option from the Shared Logic tab. This instantiates the `l_ethernet_0_sharedlogic_wrapper.v` module and the `l_ethernet_0.v` module for the Include Shared Logic in example design option. The `l_ethernet_0_gt_wrapper.v` module will be present when you select the GT subcore in example design option.

The `l_ethernet_0.v` module instantiates the necessary the sync registers/retiming pipeline registers for the synchronization of data between the core and the GT.

The `l_ethernet_0_pkt_gen_mon` module is used to generate the data packets for sanity testing. The packet generation and checking is controlled by a Finite State Machine (FSM) module.

The optional modules are described as follows:

- **l_ethernet _0_sharedlogic_wrapper:** This module is present in the example design when you select the **Include GT subcore in example design** option from the GT Selection and Configuration tab or **Include Shared Logic** in the Example Design from the Shared Logic tab. This module brings all modules that can be shared between multiple IP cores and designs outside the IP core.
- **l_ethernet _0_gt_wrapper:** This module is present in the example design when you select the **Include GT subcore in example design** option from the GT Selection and Configuration tab. This module instantiates the GT along with various helper blocks. The clocking helper blocks are used to generate the required clock frequency for the Core.

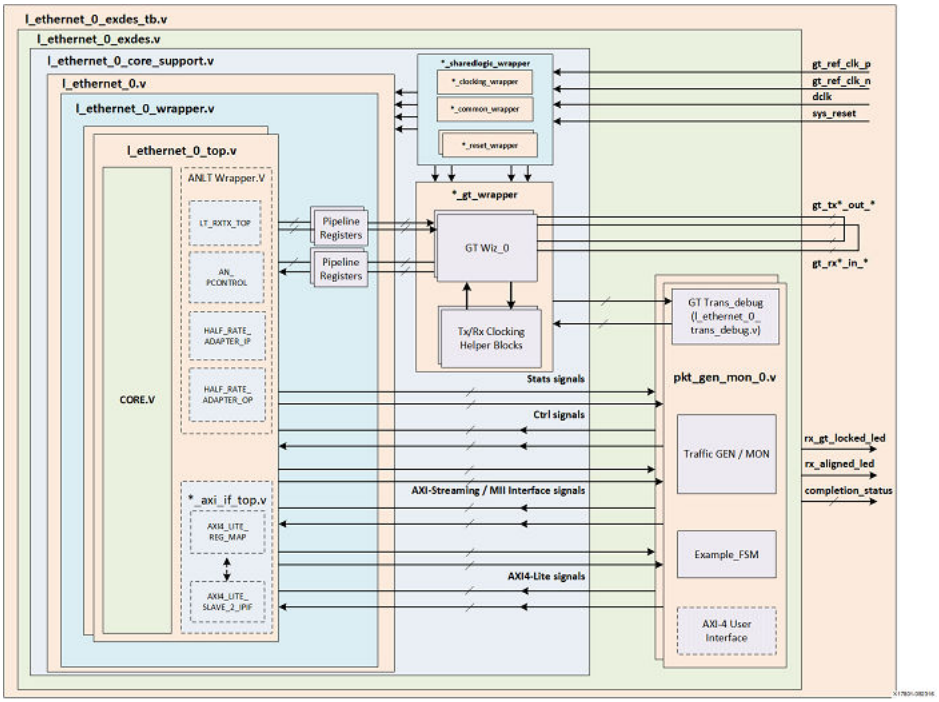
The following figure shows the instantiation of various modules and their hierarchy for the multiple core configuration of the `l_ethernet_0` example design when the GT is in the example design.

The following figure is a block design, where the 40/50G Ethernet example design connected in the IP integrator. See the *Vivado Design Suite Tutorial: Designing IP Subsystems Using IP Integrator* (UG995) for more information on IP integrator.

***Note*:** Whenever there is a change in the 40G/50G High Speed Ethernet subsystem core configuration, run the validate design and ensure that is passes. This confirms that all the changes are applied/propagated to GT in IP integrator.

User Interface

Figure 50: **Multiple Core with GT in Example Design Hierarchy**



The GPIOs in the following table have been provided to control the example design.

Table 281: **User Input / Output Ports**

Name	Size	I/O	Description
sys_reset	1	I	Reset for l_ethernet core
gt_refclk_p	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.
gt_refclk_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	1	O	Indicates that GT has been locked.
rx_aligned_led	1	O	Indicates RX aligned has been achieved

Table 281: **User Input / Output Ports** *(cont'd)*

Name	Size	I/O	Description
completion_status			This signal represents the test status/result. • 5'd0: Test did not run. • 5'd1: PASSED 50GE/40GE 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: An protocol error was detected. • 5'd15: Bit errors were detected in the received packets. • 5'd31: Test is stuck in reset.
restart_tx_rx_*	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.
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.
mode_change_0	1	I	This port is available only when Runtime Switchable is selected in Vivado IDE and this is used to switch the core speed.
core_speed_0	1	O	This signal indicates the speed with which the core is working: 1’b1 = 40G and 1’b0 = 50G.

Core xci Top Level Port List

In the following tables an asterisk (*) represents the core number, having a value of 0 and 1.

Example: Port_NAME_*

- Port_NAME_0: for first core
- Port_NAME_1: for second core (is present when you select number of cores 2)

Common Clock/Reset Signals

Table 282: **Common Clock/Reset Signals**

Name	Size	I/O	Description
sys_reset	1	I	Async reset for core This port is available when the Include GT subcore in core option is selected in the GT Selection and Configuration tab.
dclk	1	I	Stable/free running input clk to the GT This port is available when the Include GT subcore in core option is selected in the GT Selection and Configuration tab.
gt_refclk_p	1	I	Differential input clk to the GT. This port is available when the Include GT subcore in core option is selected in the GT Selection and Configuration tab and the Include Shared Logic in core is selected in the Shared Logic tab.
gt_refclk_n	1	I	Differential input clk to GT. This port is available when the Include GT subcore in core option is selected in the GT Selection and Configuration tab and the Include Shared Logic in core is selected in the Shared Logic tab.
qpll0_clk_in_*	2/4	I	QPLL0 clock input.(QPPL is quad phase-locked loop) 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 is selected in the Shared Logic tab. Port width: 2 bits for the 50G single coreand 4 bit for 40G one core and 50G two cores.
qpll0_refclk_in_*	2/4	I	QPLL0 ref clock input. 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 is selected in the Shared Logic tab. Port width: 2 bits for the 50G single coreand 4 bit for 40G one core and 50G two cores.
qpll1_clk_in_*	2/4	I	QPLL1 clock input. This port is available when the Include GT subcore in core option is selected in the GTSelection and Configuration tab and Include Shared Logic in core is selected in the Shared Logic tab. Port width: 2 bits for the 50G single core and 4 bits for 40G one core and 50G twocores.
qpll1_refclk_in_*	2/4	I	QPLL1 ref clock input. This port is available when the Include GT subcore in core option is selected in the GT Selection and Configuration tab and Include Shared Logic in example design is selected in the Shared Logic tab. Port width: 2 bits for the 50G single core and 4 bits for 40G one core and 50G two cores.
gtwiz_reset_qpll0_lock_in_*	1	I	QPLL0 lock reset input to the GT. This port is available when the Include GT subcore in core option is selected in the GT Selection and Configuration tab and Include Shared Logic in example design is selected in the Shared Logic tab.
gtwiz_reset_qpll0_reset_out_*	1	O	QPLL0 lock reset output from the GT. This port is available when the Include GT subcore in core option is selected in the GT Selection and Configuration tab and Include Shared Logic in example design is selected in the Shared Logic tab.
tx_clk_out_*	1	O	TX user clock output from GT. This port is available when the Include GT subcore in core option is selected in the GT Selection and Configuration tab.

Table 282: **Common Clock/Reset Signals** *(cont'd)*

Name	Size	I/O	Description
rx_serdes_clk_*	1	I	RX SerDes clock input to core This port is available when the Include GT subcore in core option is selected in the GT Selection and Configuration tab.
rxrecclkout_*	1	O	RX recovered clock output from GT.
tx_core_clk_*	1	I	TX Core clock input from GT wrapper. This port is available when the Include GT subcore in example design option is selected in the GT Selection and Configuration tab.
rx_core_clk _*	1	I	RX Core clock input to the core.
tx_mii_clk_*	1	O	TX user clock output from GT. This port is available when the core type is Ethernet MAC +PCS/PMA and the Include GT subcore in core option is selected in the GT Selection and Configuration tab.
rx_clk_out_*	1	O	RX user clock output from the GT.
tx_reset_*	1	I	TX reset input to the core.
user_tx_reset_*	1	O	TX reset output for the user logic. 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 is selected in the Shared Logic tab.
gt_reset_tx_done_out_*	1	O	TX reset done signal from the GT. This port is available when the Include GT subcore in core option is selected in the GT Selection and Configuration tab and the Include Shared Logic in core is selected in the Shared Logic tab.
rx_reset_*	1	I	RX reset input to the core.
user_rx_reset_*	1	O	RX reset output for the user logic. 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 is selected in the Shared Logic tab
gt_reset_rx_done_out_*	1	O	RX reset done signal from the GT. This port is available when the Include GT subcore in core option is selected in the GT Selection and Configuration tab and the Include Shared Logic in core is selected in the Shared Logic tab.
rx_serdes_reset_*	1	I	RX SerDes reset signal. This port is available when the Include Shared Logic in example design option is selected in the Shared Logic tab.
ctl_gt_reset_all_*	1	O	gt_reset_all signal from the AXI4-Lite register map. This port is available when Include AXI4-Lite is selected from the Configuration tab and the Include Shared Logic in example design is selected in the Shared Logic tab.
ctl_gt_tx_reset_*	1	O	gt_tx_reset signal from the AXI4-Lite register map. This port is available when Include AXI4-Lite is selected from the Configuration tab and the Include Shared Logic in example design is selected the Shared Logic tab.
ctl_gt_rx_reset_*	1	O	gt_rx_reset signal from the AXI4-Lite register map. This port is available when Include AXI4-Lite is selected from the Configurationtab and the Include Shared Logic in example design is selected in the Shared Logic tab.

Table 282: **Common Clock/Reset Signals** *(cont'd)*

Name	Size	I/O	Description
gt_reset_all_in_*	1	I	gt_reset_all signal from the reset_wrapper of shared logic wrapper. This port is available when the Include GT subcore in core option is selected in the GT Selection and Configuration tab and Include Shared Logic in example design is selected in the Shared Logic tab.
gt_tx_reset_in_*	1	I	gt_tx_reset_insignal from reset_wrapper of shared logic wrapper. This port is available when the Include GT subcore in core option is selected in the GT Selection and Configuration tab and Include Shared Logic in example design is selected in the Shared Logic tab.
gt_rx_reset_in_*	1	I	gt_rx_reset_in signal from reset_wrapper of shared logic wrapper. This port is available when the Include GT subcore in core option is selected in the GT Selection and Configuration tab and Include Shared Logic in example design is selected in the Shared Logic tab.
gt_refclk_out	1	O	gt_refclk which is same as gt_ref_clk to drive user fabric logic.
gtpowergood_out_*	2/4	O	Refer to the *UltraScale Architecture GTH Transceivers User Guide* (UG576) or the *UltraScale Architecture GTY Transceivers User Guide* (UG578) for the port description.
TXOUTCLKSEL_IN_*	6/12	I	This port is used to select the clock source for the gtwizard TX output clock. This port is driven with 6'b101101/12'b101101101101 as per preset.
RXOUTCLKSEL_IN_*	6/12	I	This port is used to select the clock source for the gtwizard RX output clock. This port is driven with 6'b101101/12'b101101101101 as per preset.

Common Transceiver Ports

Table 283: **Common Transceiver Ports**

Name	Size	I/O	Description
gt_loopback_in	6/12	I	GT loopback input signal. Refer to the GT user guide. 6-bit width for the 50G single core, 12-bit width for 40G single core/ 50G two core and Include GT subcore in core option selected from the GT Selection and Configuration tab)
gt_rxp_in_0	1	I	Differential serial GT RX input for lane 0. 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. This port is available when the Include GT subcore in core option is selected in the GT Selection and Configuration tab.
gt_rxp_in_1	1	I	Differential serial GT RX input for lane 1. This port is available when Include GT subcore in core option selected in the GT Selection and Configuration tab.

Table 283: **Common Transceiver Ports** *(cont'd)*

Name	Size	I/O	Description
gt_rxn_in_1	1	I	Differential serial GT RX input for lane 1. This port is available when Include GT subcore in core option selected in the GT Selection and Configuration tab.
gt_rxp_in_2	1	I	Differential serial GT RX input for lane 2. This port is available for 40G and 50G two core and the Include GT subcore in core option is selected from the GT Selection and Configuration tab.
gt_rxn_in_2	1	I	Differential serial GT RX input for lane 2. This port is available for 40G and 50G two cores and the Include GT subcore in core option selected from the GT Selection and Configuration tab.
gt_rxp_in_3	1	I	Differential serial GT RX input for lane 3. This port is available for 40G and 50G two cores and the Include GT subcore in core option is selected from the GT Selection and Configuration tab.
gt_rxn_in_3	1	I	Differential serial GT RX input for lane 3. This port is available for 40G and 50G two cores and the Include GT subcore in core option is selected from the GT Selection and Configuration tab.
gt_txp_out_0	1	O	Differential serial GT TX output for lane 0. 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. This port is available when the Include GT subcore in core option is selected in the GT Selection and Configuration tab.
gt_txp_out_1	1	O	Differential serial GT TX output for lane 1. This port is available when the Include GT subcore in core option is selected in the GT Selection and Configuration tab.
gt_txn_out_1	1	O	Differential serial GT TX output for lane 1. This port is available when the Include GT subcore in core option is selected in the GT Selection and Configuration tab.
gt_txp_out_2	1	O	Differential serial GT TX output for lane 2. This port is available for 40G and 50G two cores and the Include GT subcore in core option is selected from the GT Selection and Configuration tab.
gt_txn_out_2	1	O	Differential serial GT TX output for lane 2. This port is available for 40G and 50G two cores and the Include GT subcore in core option is selected from the GT Selection and Configuration tab.
gt_txp_out_3	1	O	Differential serial GT TX output for lane 3. This port is available for 40G and 50G two cores and the Include GT subcore in core option is selected from the GT Selection and Configuration tab.
gt_txn_out_3	1	O	Differential serial GT TX output for lane 3. This port is available for 40G and 50G two cores and the Include GT subcore in core option is selected from the GT Selection and Configuration tab.

Table 283: **Common Transceiver Ports** *(cont'd)*

Name	Size	I/O	Description
gt_txp_out	1/2	O	Differential serial GT TX output This port is available for Board Support.
gt_txn_out	1/2	O	Differential serial GT TX output This port is available for Board Support.
gt_txp_in	1/2	O	Differential serial GT TX input This port is available for Board Support.
gt_txn_in	1/2	I	Differential serial GT TX input This port is available for Board Support.
gt_loopback_out_*	1	O	GT loopback output signal from AXI4-Lite register map. See the appropriate GT user guide. 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.
rxgearboxslip_out_*	1	O	Rxgearboxslip signal from core to GT. This port is available when the Include GT subcore in example design option is selected in the GT Selection and Configuration tab.
rxdatavalid_in_*	2/4	I	RX data valid signal from GT to core. This port is available when the Include GT subcore in example design option is selected in the GT Selection and Configuration tab.
rxheader_in_*	6/12	I	RX header signal from GT to core. This port is available when the Include GT subcore in example design option is selected in the GT Selection and Configuration tab.
rxheadervalid_in_*	2/4	I	RX header valid signal from GT to core. This port is available when the Include GT subcore in example design option is selected in the GT Selection and Configuration tab.
rx_serdes_data_in_*	255	I	TX data signal from core to GT. This port is available when the Include GT subcore in example design option is selected in the GT Selection and Configuration tab.
txheader_out_*	6/12	O	TX header signal from core to GT. This port is available when the Include GT subcore in example design option is selected in the GT Selection and Configuration tab.
tx_serdes_data_out_*	255	O	TX data signal from core to GT. This port is available when the Include GT subcore in example design option is selected in the GT Selection and Configuration tab.
txsequence_out_*	7	O	TX sequence signal from the core to the GT.

Transceiver Control and Status Debug Ports

Ports under this section are available when Enable Additional GT Control/Status and DRP Ports is selected from the GT Selection and Configuration tab. Refer to the GT user guide for the port description.

Table 284: **Transceiver Control and Status Debug Ports**

Name	Size	I/O	Description
gt_dmonitorout_*	34/68	O	Port width: 34 bits for the 50G single core and 68 bits for 40G.
gt_eyescandataerror_*	2/4	O	Port width: 2 bits for the 50G single core and 4 bits for 40G.
gt_eyescanreset_*	2/4	I	Port width: 2 bits for the 50G single core and 4 bits for 40G.
gt_eyescantrigger_*	2/4	I	Port width: 2 bits for the 50G single core and 4 bits for 40G.
gt_pcsrsvdin_*	32/64	I	Port width: 32 bits for the 50G single core and 64 bits for 40G.
gt_rxbufreset_*	2/4	I	Port width: 2 bits for the 50G single core and 4 bits for 40G.
gt_rxbufstatus_*	6/12	O	Port width: 6-bit for the 50G single core and 12 bits for 40G.
gt_rxcdrhold_*	2/4	I	Port width: 2 bits for the 50G single core and 4 bits for 40G.
gt_rxcommadeten_*	2/4	I	Port width: 2 bits for the 50G single core and 4 bits for 40G.
gt_rxdfeagchold_*	2/4	I	Port width: 2 bits for the 50G single core and 4 bits for 40G.
gt_rxdfelpmreset_*	2/4	I	Port width: 2 bits for the 50G single core and 4 bits for 40G.
gt_rxlatclk_*	2/4	I	Port width: 2 bits for the 50G single core and 4 bits for 40G.
gt_rxlpmen_*	2/4	I	Port width: 2 bits for the 50G single core and 4 bits for 40G.
gt_rxpcsreset_*	2/4	I	Port width: 2 bits for the 50G single core and 4 bits for 40G.
gt_rxpmareset_*	2/4	I	Port width: 2 bits for the 50G single core and 4 bits for 40G.
gt_rxpolarity_*	2/4	I	Port width: 2 bits for the 50G single core and 4 bits for 40G.
gt_rxprbscntreset_*	2/4	I	Port width: 2 bits for the 50G single core and 4 bits for 40G.
gt_rxprbserr_*	2/4	I	Port width: 2 bits for the 50G single core and 4 bits for 40G.
gt_rxprbssel_*	8/16	I	Port width: 8-bit for the 50G single core and 16 bits for 40G.
gt_rxrate_*	6/12	I	Port width: 6-bit for the 50G single core and 12 bits for 40G.
gt_rxslide_in_*	2/4	I	Port width: 2 bits for the 50G single core and 4 bits for 40G.
gt_rxstartofseq_*	4/8	O	Port width: 4-bit for the 50G single core and 8 bits for 40G.
gt_txbufstatus_*	4/8	O	Port width: 4-bit for the 50G single core and 8 bits for 40G.
gt_txdiffctrl_*	10/20	I	Port width: 10-bit for the 50G single core and 20 bits for 40G.
gt_txinhibit_*	2/4	I	Port width: 2 bits for the 50G single core and 4 bits for 40G.
gt_txlatclk_*	2/4	I	Port width: 2 bits for the 50G single core and 4 bits for 40G.
gt_txmaincursor_*	14/28	I	Port width: 14-bit for the 50G single core and 28 bits for 40G.
gt_txpcsreset_*	2/4	I	Port width: 2 bits for the 50G single core and 4 bits for 40G.
gt_txpmareset_*	2/4	I	Port width: 2 bits for the 50G single core and 4 bits for 40G.
gt_txpolarity_*	2/4	I	Port width: 2 bits for the 50G single core and 4 bits for 40G.
gt_txpostcursor_*	10/20	I	Port width: 10-bit for the 50G single core and 20 bits for 40G.
gt_txprbsforceerr_*	2/4	I	Port width: 2 bits for the 50G single core and 4 bits for 40G.
gt_txprbssel_*	8/16	I	Port width: 8-bit for the 50G single core and 16 bits for 40G.
gt_txprecursor_*	10/20	I	Port width: 10-bit for the 50G single core and 20 bits for 40G.
gtwiz_reset_tx_datapath_*	2/4	I	Port width: 2 bits for the 50G single core and 4 bits for 40G.
gtwiz_reset_rx_datapath_*	2/4	I	Port width: 2 bits for the 50G single core and 4 bits for 40G.
gt_common_drpclk	1	I	This port is available when the Include Shared Logic in core option is selected in the Shared Logic tab.
gt_common_drpdo	16	O	This port is available when the Include Shared Logic in core option is selected in the Shared Logic tab.

Table 284: **Transceiver Control and Status Debug Ports** *(cont'd)*

Name	Size	I/O	Description
gt_common_drprdy	1	O	This port is available when the Include Shared Logic in core option is selected in the Shared Logic tab.
gt_common_drpen	1	I	This port is available when the Include Shared Logic in core option is selected in the Shared Logic tab.
gt_common_drpwe	1	I	This port is available when the Include Shared Logic in core option is selected in the Shared Logic tab.
gt_common_drpaddr	10	I	This port is available when the Include Shared Logic in core option is selected in the Shared Logic tab.
gt_common_drpdi	16	I	This port is available when the Include Shared Logic in core option is selected in the Shared Logic tab.
gt_ch_drpclk_0	1	I	
gt_ch_drpdo_0	16	O	
gt_ch_drprdy_0	1	O	
gt_ch_drpen_0	1	I	
gt_ch_drpwe_0	1	I	
gt_ch_drpaddr_0	10	I	
gt_ch_drpdi_0	16	I	
gt_ch_drpclk_1	1	I	
gt_ch_drpdo_1	16	O	
gt_ch_drprdy_1	1	O	
gt_ch_drpen_1	1	I	
gt_ch_drpwe_1	1	I	
gt_ch_drpaddr_1	10	I	
gt_ch_drpdi_1	16	I	
gt_ch_drpclk_2	1	I	This port is available when core speed is 40G / speed 50G with two cores.
gt_ch_drpdo_2	16	O	This port is available when core speed is 40G / speed 50G with two cores.
gt_ch_drprdy_2	1	O	This port is available when core speed is 40G / speed 50G with two cores.
gt_ch_drpen_2	1	I	This port is available when core speed is 40G / speed 50G with two cores.
gt_ch_drpwe_2	1	I	This port is available when core speed is 40G / speed 50G with two cores.
gt_ch_drpaddr_2	10	I	This port is available when core speed is 40G / speed 50G with two cores.
gt_ch_drpdi_2	16	I	This port is available when core speed is 40G / speed 50G with two cores.
gt_ch_drpclk_3	1	I	This port is available when core speed is 40G / speed 50G with two cores.
gt_ch_drpdo_3	16	O	This port is available when core speed is 40G / speed 50G with two cores.
gt_ch_drprdy_3	1	O	This port is available when core speed is 40G / speed 50G with two cores.

Table 284: **Transceiver Control and Status Debug Ports** *(cont'd)*

Name	Size	I/O	Description
gt_ch_drpen_3	1	I	This port is available when core speed is 40G / speed 50G with two cores.
gt_ch_drpwe_3	1	I	This port is available when core speed is 40G / speed 50G with two cores.
gt_ch_drpaddr_3	10	I	This port is available when core speed is 40G / speed 50G with two cores.
gt_ch_drpdi_3	16	I	This port is available when core speed is 40G / speed 50G with two cores.

AXI4-Lite Interface Ports

Ports under this section are available when Include AXI4-Lite is selected from the Configuration tab.

Table 285: **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
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

Table 285: **AXI4-Lite Interface Ports** *(cont'd)*

Name	Size	I/O	Description
s_axi_rready_*	1	I	AXI read ready. This signal indicates the user/master can accept the read data and response information.

128-bit Straddled AXI4-Stream User Interface Signals

Ports under this section are available when Ethernet MAC+PCS/PMA with the 128-bit Straddle Packet AXI4-Stream option is selected from the Configuration tab.

Table 286: **128-bit Straddled AXI4-Stream User Interface Signals**

Name	Size	I/O	Description
tx_unfout_*	1	O	Underflow signal for TX path 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 data path interface. ***Note***: When this signal sampled as 1, you must apply tx_reset/ sys_reset to recover the core 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_*	128	I	Transmit AXI4-Stream Data bus.
tx_axis_tuser_*	70	I	TX segment and packet information signal. tx_axis_tuser_0[69:0]
			69 - tx_axis_tuser_err1
			68:66 - tx_axis_tuser_mty1[2:0]
			65 - tx_axis_tuser_eop1
			64 - tx_axis_tuser_sop1
			63 - tx_axis_tuser_ena1
			62 - tx_axis_tuser_err0
			61:59 - tx_axis_tuser_mty0[2:0]
			58 - tx_axis_tuser_eop0
			57 - tx_axis_tuser_sop0
			56- tx_axis_tuser_ena0
			55:0- tx_preamblein
rx_axis_tvalid_*	1	O	Receive AXI4-Stream Data valid.
rx_axis_tdata_*	128	O	Receive AXI4-Stream Data bus.

Table 286: **128-bit Straddled AXI4-Stream User Interface Signals** *(cont'd)*

Name	Size	I/O	Description
rx_axis_tuser	70	O	RX segment and packet information signal. rx_axis_tuser_0[69:0]
			69 - rx_axis_tuser_err1
			68:66 - rx_axis_tuser_mty1[2:0]
			65 - rx_axis_tuser_eop1
			64 - rx_axis_tuser_sop1
			63 - rx_axis_tuser_ena1
			62 - rx_axis_tuser_err0
			61:59 - rx_axis_tuser_mty0[2:0]
			58 - rx_axis_tuser_eop0
			57 - rx_axis_tuser_sop0
			56 - rx_axis_tuser_ena0
			55:0 - rx_preamblein

256-bit AXI4-Stream Ports

Ports under this section are available when Ethernet MAC + PCS/PMA with the 256-bit AXI4-Stream option is selected from the Configuration tab.

Table 287: **256-bit AXI4-Stream Ports**

Name	Size	I/O	Description
tx_axis_tready_*	1	O	AXI4-Stream acknowledge signal to indicate to start the Data transfer.
tx_axis_tvalid_*	1	I	AXI4-Stream Data Valid Input
tx_axis_tdata_*	256	I	AXI4-Stream Data
tx_axis_tuser_*	1	I	AXI4-Stream User Sideband interface. 1 indicates a bad packet has been received. 0 indicates a good packet has been received.
tx_axis_tlast_*	1	I	AXI4-Stream signal indicating End of Ethernet Packet
tx_axis_tkeep_*	32	I	AXI4-Stream Data Control
rx_axis_tdata_*	256	O	AXI4-Stream Data to user logic
rx_axis_tvalid_*	1	O	AXI4-Stream Data Valid. When this signal is 1, there is valid data on the RX AXI bus
rx_axis_tuser_*	1	O	AXI4-Stream User Sideband interface. 1 indicates a bad packet has been received. 0 indicates a good packet has been received
rx_axis_tlast_*	1	O	AXI4-Stream signal indicating an end of packet
rx_axis_tkeep_*	32	O	AXI4-Stream Data Control to upper layer.

MII User Interface Signals

Ports under this section are available when Ethernet PCS/PMA is selected from the Configuration tab.

Table 288: **MII User Interface Signals**

Name	Size	I/O	Description
tx_mii_d_*	128	I	Transmit XLGMII/50GMII Data bus.
tx_mii_c_*	16	I	XLGMII/50GMII Control bus.
rx_mii_d_*	128	O	Receive XLGMII/50GMII Data bus.
rx_mii_c_*	16	O	Receive XLGMII/50GMII Control bus.

TX Path Control/Status/Statistics Signals

Table 289: **TX Path Control/Status/Statistics Signals**

Name	Size	I/O	Description
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 82.2.10. Corresponds to MDIO register bit 3.42.3 as defined in Clause 82.3. Generates a scrambled idle pattern. This port is available when the AXI4-Lite interface is *not* selected.
ctl_tx_enable_*	1	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 port is available when the AXI4-Lite interface is *not* selected and core type is Ethernet MAC+PCS/PMA.
ctl_tx_fcs_ins_enable_*	1	I	Enable FCS insertion by the TX core. If this bit is set to 0, the 40G/50G High Speed Ethernet Subsystem does not add FCS to packet. It his bit is set to 1, the 40G/50G High Speed Ethernet Subsystem calculates and adds the FCS to the packet. This input cannot be changed dynamically between packets. This port is available when the AXI4-Lite interface is *not* selected and core type is Ethernet MAC+PCS/PMA.
ctl_tx_ipg_value_*	4	I	This signal might be optionally present. The ctl_tx_ipg_value defines the target average minimum Inter Packet Gap (IPG, in bytes) inserted between rx_serdes_clk packets. Valid values are 8 to 12. The ctl_tx_ipg_value can also be programmed to a value in the 0 to 7 range, but in that case, it is interpreted as meaning "minimal IPG", so only Terminate code word IPG is inserted; no Idles are ever added in that case and that produces an average IPG of around 4 bytes when random-size packets are transmitted. This port is available when the AXI4-Lite interface is *not* selected and core type is Ethernet MAC+PCS/PMA and Include FIFO Logic is disabled.
ctl_tx_send_lfi_*	1	I	Transmit Local Fault Indication (LFI) code word. Takes precedence over RFI. This port is available when core type is Ethernet MAC+PCS/PMA.
ctl_tx_send_rfi_*	1	I	Transmit Remote Fault Indication (RFI) code word. This port is available when core type is Ethernet MAC+PCS/PMA.

Table 289: **TX Path Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
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. This port is available when core type is Ethernet MAC+PCS/PMA.
ctl_tx_custom_preamble_enable_*	1	I	When asserted, this signal enables the use of tx_preamblein as a custom preamble instead of inserting a standard preamble. This port is available when the AXI4-Lite interface is *not* selected and core type is Ethernet MAC+PCS/PMA and Include FIFO Logic is disabled
ctl_tx_ignore_fcs_*	1	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. Statistics are reported as if there was no FCS error. This port is available when the AXI4-Lite interface is n*not*ot selected and core type is Ethernet MAC+PCS.
stat_tx_total_packets_*	1	O	Increment for the total number of packets transmitted. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_tx_total_bytes_*	5	O	Increment for the total number of bytes transmitted. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_tx_total_good_packets_*	1	O	Increment for the total number of good packets transmitted. This port is available when core type is Ethernet MAC+PCS/PMA.
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. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_tx_packet_64_bytes_*	1	O	Increment for good and bad packets transmitted that contain 64 bytes. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_tx_packet_65_127_bytes_*	1	O	Increment for good and bad packets transmitted that contain 65 to 127 bytes. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_tx_packet_128_255_bytes_*	1	O	Increment for good and bad packets transmitted that contain 128 to 255 bytes. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_tx_packet_256_511_bytes_*	1	O	Increment for good and bad packets transmitted that contain 256 to 511 bytes. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_tx_packet_512_1023_bytes_*	1	O	Increment for good and bad packets transmitted that contain 512 to 1,023 bytes. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_tx_packet_1024_1518_bytes_*	1	O	Increment for good and bad packets transmitted that contain 1,024 to 1,518 bytes. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_tx_packet_1519_1522_bytes_*	1	O	Increment for good and bad packets transmitted that contain 1,519 to 1,522 bytes. This port is available when core type is Ethernet MAC+PCS/PMA.

Table 289: **TX Path Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
stat_tx_packet_1523_1548_bytes_*	1	O	Increment for good and bad packets transmitted that contain 1,523 to 1,548 bytes. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_tx_packet_1549_2047_bytes_*	1	O	Increment for good and bad packets transmitted that contain 1,549 to 2,047 bytes. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_tx_packet_2048_4095_bytes_*	1	O	Increment for good and bad packets transmitted that contain 2,048 to 4,095 bytes. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_tx_packet_4096_8191_bytes_*	1	O	Increment for good and bad packets transmitted that contain 4,096 to 8,191 bytes. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_tx_packet_8192_9215_bytes_*	1	O	Increment for good and bad packets transmitted that contain 8,192 to 9,215 bytes. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_tx_packet_small_*	1	O	Increment for all packets that are less than 64 bytes long. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_tx_packet_large_*	1	O	Increment for all packets that are more than 9,215 bytes long. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_tx_bad_fcs_*	1	O	Increment for packets greater than 64 bytes that have FCS errors. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_tx_frame_error_*	1	O	Increment for packets with tx_errin set to indicate an EOP abort. This port is available when core type is Ethernet MAC+PCS/PMA.
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_fifo_error_*	1	O	Transmit clock compensation First In First Out (FIFO) error indicator. A value of 1 indicates the clock compensation FIFO under or overflowed. This condition only occurs if the PPM difference between the transmitter clock and the local reference clock is greater than ±200 ppm. If this output is sampled as a 1 in any clock cycle, the corresponding port must be reset to resume proper operation. This port is available when core type is Ethernet PCS/PMA.

RX Path Control/Status/Statistics Signals

Table 290: **RX Path Control/Status/Statistics Signals**

Name	Size	I/O	Description
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 82.2.17. Corresponds to MDIO register bit 3.42.2 as defined in Clause 82.3. Checks for scrambled idle pattern. This port is available when the AXI4-Lite interface is *not* selected.

Table 290: **RX Path Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
ctl_rx_enable_*	1	I	RX Enable. For normal operation, this input must be set to 1. When this input is set to 0, after the RX completes the reception of the current packet (if any), it stops receiving packets by keeping the PCS from decoding incoming data. In this mode, there are no statistics reported and the user interface is idle. This port is available when the AXI4-Lite interface is *not* selected and core type is Ethernet MAC+PCS/PMA.
ctl_rx_delete_fcs_*	1	I	Enable FCS removal by the RX core. If this bit is set to 0, the 40G/50G High Speed Ethernet Subsystem does not remove the FCS of the incoming packet. If this bit is set to 1, the 40G/50G High Speed Ethernet Subsystem deletes the FCS to the received packet. FCS is not deleted for packets that are =<8 bytes long. This input should only be changed while the corresponding reset input is asserted. This port is available when the AXI4-Lite interface is *not* selected and core type is Ethernet MAC+PCS/PMA.
ctl_rx_ignore_fcs_*	1	I	Enable FCS error checking at the user interface by the RX core. If this bit is set to 0, a packet received with an FCS error is sent with the rx_errout pin asserted during the last transfer (rx_eopout and rx_enaout sampled 1). If this bit is set to 1, the 40G/ 50G High Speed Ethernet Subsystem does not flag an FCS error at the user interface. The statistics are reported as if the packet is good. The stat_rx_bad_fcs signal, however, reports the error. This port is available when the AXI4-Lite interface is *not* selected and core type is Ethernet MAC+PCS/PMA.
ctl_rx_max_packet_len_*	15	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_errout signal is asserted along with the rx_eopout signal. Packets less than 16 bytes are dropped. The allowed value for this bus can range from 64 to 16,383. ctl_rx_max_packet_len[14] is reserved and must be set to 0. This port is available when the AXI4-Lite interface is *not* selected and core type is Ethernet MAC+PCS/PMA.
ctl_rx_min_packet_len_*	8	I	Any packet shorter than this value is considered to be undersized. If a packet has a size less than this value, the rx_errout signal is asserted during the rx_eopout asserted cycle. Packets that are less than 64 bytes are dropped. This port is available when the AXI4-Lite interface is *not* selected and core type is Ethernet MAC+PCS/PMA.
ctl_rx_custom_preamble_enable_*	1	I	When asserted, this signal causes the preamble to be presented on rx_preambleout. This port is available when the AXI4-Lite interface is *not* selected and core type is Ethernet MAC+PCS/PMA and Include FIFO Logic is disabled)
ctl_rx_check_sfd_*	1	I	When asserted, this input causes the Ethernet MAC to check the start of frame Delimiter of the received frame. This port is available when the AXI4-Lite interface is *not* selected and core type is Ethernet MAC+PCS/PMA.
ctl_rx_check_preamble_*	1	I	When asserted, this input causes the Ethernet MAC to check the preamble of the received frame. This port is available when the AXI4-Lite interface is *not* selected and core type is Ethernet MAC+PCS/PMA.

Table 290: **RX Path Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
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 SerDes. When set to 0, the RX core ignores LF control codes coming in from the SerDes. This port is available when the AXI4-Lite interface is *not* selected and core type is Ethernet MAC+PCS/PMA.
ctl_rx_force_resync_*	1	I	RX force resynchronization input. This signal is used to force the RX path to reset, re-synchronize, and realign. A value of 1 forces the reset operation. A value of 0 allows normal operation. ***Note***: This input should normally be Low and should only be pulsed (1 cycle minimum pulse) to force realignment. This port is available when the AXI4-Lite interface is *not* selected and core type is Ethernet MAC+PCS/PMA.
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_0_*	1	O	Valid indicator for stat_rx_framing_err_0. When 1 stat_rx_framing_err_0 is valid.
stat_rx_framing_err_0_*	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_0 is a 1. The values on these buses can be updated at any time and are intended to be used as increment values for sync header error counters.
stat_rx_framing_err_valid_1_*	1	O	Valid indicator for stat_rx_framing_err_1. When 1 stat_rx_framing_err_1 is valid.
stat_rx_framing_err_1_*	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_1 is a 1. The values on these buses can be updated at any time and are intended to be used as increment values for sync header error counters.
stat_rx_framing_err_valid_2_*	1	O	Valid indicator for stat_rx_framing_err_2. When 1 stat_rx_framing_err_2 is valid.
stat_rx_framing_err_2_*	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_2 is a 1. The values on these buses can be updated at any time and are intended to be used as increment values for sync header error counters.
stat_rx_framing_err_valid_3_*	1	O	Valid indicator for stat_rx_framing_err_3. When 1 stat_rx_framing_err_3 is valid.
stat_rx_framing_err_3_*	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_3 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_vl_demuxed_*	4	O	PCS Lane Marker found. If a signal of this bus is sampled as 1, it indicates that the receiver has properly de-muxed that PCS lane. This output is level sensitive.

Table 290: **RX Path Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
stat_rx_vl_number_0_*	2	O	The value of this bus indicates which physical lane appears on PCS lane 0. This bus is only valid when the corresponding bit of stat_rx_synced[PCS_LANES-1:0] is a 1. These outputs are level sensitive.
stat_rx_vl_number_1_*	2	O	The value of this bus indicates which physical lane appears on PCS lane 1.
stat_rx_vl_number_2_*	2	O	The value of this bus indicates which physical lane appears on PCS lane 2.
stat_rx_vl_number_3_*	2	O	The value of this bus indicates which physical lane appears on PCS lane 3.
stat_rx_synced_*	4	O	Word Boundary Synchronized. These signals indicate whether a PCS lane is word boundary synchronized. A value of 1 indicates the corresponding PCS lane has achieved word boundary synchronization and it has received a PCS lane marker. Corresponds to MDIO register bit 3.52.7:0 and 3.53.11:0 as defined in Clause 82.3. This output is level sensitive.
stat_rx_synced_err_*	4	O	Word Boundary Synchronization Error. These signals indicate whether an error occurred during word boundary synchronization in the respective PCS lane. A value of 1 indicates that the corresponding PCS lane lost word boundary synchronization due to sync header framing bits errors or that a PCS lane marker was never received. This output is level sensitive.
stat_rx_mf_len_err_*	4	O	PCS Lane Marker Length Error. These signals indicate whether a PCS Lane Marker length mismatch occurred in the respective lane (that is, PCS Lane Markers were received not every ctl_rx_vl_length_minus1 words apart). A value of 1 indicates that the corresponding lane is receiving PCS Lane Markers at wrong intervals. This remains High until the error condition is removed.
stat_rx_mf_repeat_err_*	4	O	PCS Lane Marker Consecutive Error. These signals indicate whether four consecutive PCS Lane Marker errors occurred in the respective lane. A value of 1 indicates an error in the corresponding lane. This output remains High until the error condition is removed.
stat_rx_mf_err_*	4	O	PCS Lane Marker Word Error. These signals indicate that an incorrectly formed PCS Lane Marker Word was detected in the respective lane. A value of 1 indicates an error occurred. This output is pulsed for one clock cycle to indicate the error condition. Pulses can occur in back-to-back cycles.
stat_rx_misaligned_*	1	O	Alignment Error. This signal indicates that the lane aligner did not receive the expected PCS lane marker across all lanes. This signal is not asserted until the PCS lane marker has been received at least once across all lanes and at least one incorrect lane marker has been received. This occurs one metaframe after the error. This signal is not asserted if the lane markers have never been received correctly. Lane marker errors are indicated by the corresponding stat_rx_mf_err signal. This output is pulsed for one clock cycle to indicate an error condition. Pulses can occur in back-to-back cycles.
stat_rx_aligned_err_*	1	O	Loss of Lane Alignment/Deskew. This signal indicates that an error occurred during PCS lane alignment or PCS lane alignment was lost. A value of 1 indicates an error occurred. This output is level sensitive.

Table 290: **RX Path Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
stat_rx_bip_err_0_*	1	O	BIP8 error indicator for PCS lane 0. A non-zero value indicates the BIP8 signature was in error. A non-zero value is pulsed for one clock cycle. This output is pulsed for one clock cycle to indicate an error condition.
stat_rx_bip_err_1_*	1	O	BIP8 error indicator for PCS lane 1.
stat_rx_bip_err_2_*	1	O	BIP8 error indicator for PCS lane 2.
stat_rx_bip_err_3_*	1	O	BIP8 error indicator for PCS lane 3.
stat_rx_aligned_*	1	O	All PCS Lanes Aligned/Deskewed. This signal indicates whether or not all PCS lanes are aligned and deskewed. A value of 1 indicates all PCS lanes are aligned and deskewed. When this signal is a 1, the RX path is aligned and can receive packet data. When this signal is 0, a local fault condition exists. This also corresponds to MDIO register bit 3.50.12 as defined in Clause 82.3. This output is level sensitive.
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.
stat_rx_status_*	1	O	PCS status. A value of 1 indicates that the PCS is aligned and not in hi_ber state. Corresponds to Management Data Input/ Output (MDIO) register bit 3.32.12 as defined in Clause 82.3. This output is level sensitive.
stat_rx_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_bad_code_valid_*	1	O	Indicates when stat_rx_bad_code is valid. This port is available when core type is Ethernet PCS/PMA.
stat_rx_error_valid_*	1	O	Indicates when stat_rx_error is valid. This port is available when core type is Ethernet PCS/PMA.
stat_rx_error_*	8	O	Test pattern mismatch increment. A non-zero value in any cycle indicates a mismatch occurred for the test pattern in the RX core. This output is only active when ctl_rx_test_pattern is set to a 1. This output is pulsed for one clock cycle. This port is available when core type is Ethernet PCS/PMA.
stat_rx_fifo_error_*	1	O	Indicates when RX FIFO goes into an underflow or overflow condition. If this output is sampled as a 1 in any clock cycle, the corresponding port must be reset to resume proper operation. This port is available when core type is Ethernet PCS/PMA.
stat_rx_total_packets_*	2	O	Increment for the total number of packets received. This port is available when core type is Ethernet MAC+PCS/PMA.
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. This port is available when core type is Ethernet MAC+PCS/PMA.

Table 290: **RX Path Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
stat_rx_total_bytes_*	6	O	Increment for the total number of bytes received. This port is available when core type is Ethernet MAC+PCS/PMA.
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. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_rx_packet_small_*	2	O	Increment for all packets that are less than 64 bytes long. Packets that are less than 16 bytes are dropped. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_rx_jabber_*	1	O	Increment for packets longer than ctl_rx_max_packet_len with bad FCS. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_rx_packet_large_*	1	O	Increment for all packets that are more than 9,215 bytes long. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_rx_oversize_*	1	O	Increment for packets longer than ctl_rx_max_packet_len with good FCS. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_rx_undersize_*	2	O	Increment for packets shorter than stat_rx_min_packet_len with good FCS. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_rx_toolong_*	1	O	Increment for packets longer than ctl_rx_max_packet_len with good and bad FCS. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_rx_fragment_*	2	O	Increment for packets shorter than stat_rx_min_packet_len with bad FCS. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_rx_packet_64_bytes_*	1	O	Increment for good and bad packets received that contain 64 bytes. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_rx_packet_65_127_bytes_*	1	O	Increment for good and bad packets received that contain 65 to 127 bytes. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_rx_packet_128_255_bytes_*	1	O	Increment for good and bad packets received that contain 128 to 255 bytes. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_rx_packet_256_511_bytes_*	1	O	Increment for good and bad packets received that contain 256 to 511 bytes. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_rx_packet_512_1023_bytes_*	1	O	Increment for good and bad packets received that contain 512 to 1,023 bytes. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_rx_packet_1024_1518_bytes_*	1	O	Increment for good and bad packets received that contain 1,024 to 1,518 bytes. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_rx_packet_1519_1522_bytes_*	1	O	Increment for good and bad packets received that contain 1,519 to 1,522 bytes. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_rx_packet_1523_1548_bytes_*	1	O	Increment for good and bad packets received that contain 1,523 to 1,548 bytes. This port is available when core type is Ethernet MAC+PCS/PMA.

Table 290: **RX Path Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
stat_rx_packet_1549_2047_bytes_*	1	O	Increment for good and bad packets received that contain 1,549 to 2,047 bytes. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_rx_packet_2048_4095_bytes_*	1	O	Increment for good and bad packets received that contain 2,048 to 4,095 bytes. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_rx_packet_4096_8191_bytes_*	1	O	Increment for good and bad packets received that contain 4,096 to 8,191 bytes. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_rx_packet_8192_9215_bytes_*	1	O	Increment for good and bad packets received that contain 8,192 to 9,215 bytes. This port is available when core type is Ethernet MAC+PCS/PMA.
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. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_rx_packet_bad_fcs_*	1	O	Increment for packets between 64 and ctl_rx_max_packet_len bytes that have FCS errors. This port is available when core type is Ethernet MAC+PCS/PMA.
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. This port is available when core type is Ethernet MAC+PCS/PMA.
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. ***Note***: When an invalid preamble is detected, the stat_rx_bad_preamble signal is asserted regardless of the setting of the ctl_rx_check_preamble signal. This port is available when core type is Ethernet MAC+PCS/PMA.
stat_rx_bad_preamble_*	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. ***Note***: When an invalid SFD is detected, the stat_rx_bad_preamble signal is asserted regardless of the setting of the ctl_rx_check_preamble signal. This port is available when core type is Ethernet MAC+PCS/PMA.
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***: Signal OS should not be received in an Ethernet network. This port is available when core type is Ethernet MAC+PCS/PMA.

Table 290: **RX Path Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
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. This port is available when core type is Ethernet MAC+PCS/PMA.
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. This port is available when core type is Ethernet MAC+PCS/PMA.
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. This port is available when core type is Ethernet MAC+PCS/PMA.
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. This port is available when core type is Ethernet MAC+PCS/PMA.
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. This port is available when core type is Ethernet MAC+PCS/PMA.

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 the CORE type is Ethernet MAC+PCS/PMA.

Table 291: **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. This port is available when the AXI4-Lite interface is *not* selected.

Table 291: **TX Pause Interface Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
ctl_tx_resend_pause_*	1	I	Re-transmit pending pause packets. When this input is sampled as 1, all pending pause packets are retransmitted as soon as possible (that is, after the current packet in flight is completed) and the retransmit counters are reset. This input should be pulsed to 1 for one cycle at a time.
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. This port is available when the AXI4-Lite interface is *not* selected.
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. This port is available when the AXI4-Lite interface is *not* selected.
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. This port is available when the AXI4-Lite interface is not selected.
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. This port is available when the AXI4-Lite interface is *not* selected.
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. This port is available when the AXI4-Lite interface is *not* selected.
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. This port is available when the AXI4-Lite interface is *not* selected.
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. This port is available when the AXI4-Lite interface is *not* selected.
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. This port is available when the AXI4-Lite interface is *not* selected.

Table 291: **TX Pause Interface Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
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. This port is available when the AXI4-Lite interface is *not* selected.
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. This port is available when the AXI4-Lite interface is not selected.
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. This port is available when the AXI4-Lite interface is not selected.
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. This port is available when the AXI4-Lite interface is *not* selected.
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. This port is available when the AXI4-Lite interface is *not* selected.
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. This port is available when the AXI4-Lite interface is *not* selected.
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. This port is available when the AXI4-Lite interface is *not* selected.
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. This port is available when the AXI4-Lite interface is *not* selected.
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. This port is available when the AXI4-Lite interface is *not* selected.

Table 291: **TX Pause Interface Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
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. This port is available when the AXI4-Lite interface is *not* selected.
ctl_tx_da_gpp_*	48	I	Destination address for transmitting global pause packets. This port is available when the AXI4-Lite interface is *not* selected.
ctl_tx_sa_gpp_*	48	I	Source address for transmitting global pause packets. This port is available when the AXI4-Lite interface is *not* selected.
ctl_tx_ethertype_gpp_*	16	I	Ethertype for transmitting global pause packets. This port is available when the AXI4-Lite interface is *not* selected.
ctl_tx_opcode_gpp_*	16	I	Opcode for transmitting global pause packets. This port is available when the AXI4-Lite interface is *not* selected.
ctl_tx_da_ppp_*	48	I	Destination address for transmitting priority pause packets. This port is available when the AXI4-Lite interface is *not* selected.
ctl_tx_sa_ppp_*	48	I	Source address for transmitting priority pause packets. This port is available when the AXI4-Lite interface is *not* selected.
ctl_tx_ethertype_ppp_*	16	I	Ethertype for transmitting priority pause packets. This port is available when the AXI4-Lite interface is *not* selected.
ctl_tx_opcode_ppp_*	16	I	Opcode for transmitting priority pause packets. This port is available when the AXI4-Lite interface is *not* selected.
stat_tx_pause_valid_*	9	O	If a bit of this bus is set to 1, the 40G/50G High Speed Ethernet Subsystem has transmitted a pause packet. If bit[8] is set to 1, a global pause packet is transmitted. All other bits cause a priority pause packet to be transmitted.
stat_tx_unicast_*	1	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.

RX Pause Interface Control/Status/Statistics Signals

Ports under this section are available when Enable RX Flow Control Logic is selected from the MAC Options tab and CORE type is Ethernet MAC+PCS/PMA.

Table 292: **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 the core to drop control packets. This port is available when the AXI4-Lite interface is *not* selected.

Table 292: **RX Pause Interface Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
ctl_rx_pause_ack_*	9	I	Pause acknowledge. 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. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_pause_enable_*	9	I	RX pause enable. This input is used to enable the processing of the pause quanta for the corresponding priority. ***Note*:** This signal only affects the RX user interface, not the pause processing logic. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_enable_gcp_*	1	I	A value of 1 enables global control packet processing. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_check_mcast_gcp_*	1	I	A value of 1 enables global control multicast destination address processing. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_check_ucast_gcp_*	1	I	A value of 1 enables global control unicast destination address processing. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_pause_da_ucast_*	48	I	Unicast destination address for pause processing. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_check_sa_gcp_*	1	I	A value of 1 enables global control source address processing. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_pause_sa_*	48	I	Source address for pause processing. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_check_etype_gcp_*	1	I	A value of 1 enables global control ethertype processing. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_etype_gcp_*	16	I	Ethertype field for global control processing. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_check_opcode_gcp_*	1	I	A value of 1 enables global control opcode processing. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_opcode_min_gcp_*	16	I	Minimum global control opcode value. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_opcode_max_gcp_*	16	I	Maximum global control opcode value. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_enable_pcp_*	1	I	A value of 1 enables priority control packet processing. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_check_mcast_pcp_*	1	I	A value of 1 enables priority control multicast destination address processing. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_check_ucast_pcp_*	1	I	A value of 1 enables priority control unicast destination address processing. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_pause_da_mcast_*	48	I	Multicast destination address for pause processing. This port is available when the AXI4-Lite interface is *not* selected.

Table 292: **RX Pause Interface Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
ctl_rx_check_sa_pcp_*	1	I	A value of 1 enables priority control source address processing. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_check_etype_pcp_*	1	I	A value of 1 enables priority control ethertype processing. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_etype_pcp_*	16	I	Ethertype field for priority control processing. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_check_opcode_pcp_*	1	I	A value of 1 enables priority control opcode processing. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_opcode_min_pcp_*	16	I	Minimum priority control opcode value. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_opcode_max_pcp_*	16	I	Maximum priority control opcode value. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_enable_gpp_*	1	I	A value of 1 enables global pause packet processing. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_check_mcast_gpp_*	1	I	A value of 1 enables global pause multicast destination address processing. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_check_ucast_gpp_*	1	I	A value of 1 enables global pause unicast destination address processing. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_check_sa_gpp_*	1	I	A value of 1 enables global pause source address processing. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_check_etype_gpp_*	1	I	A value of 1 enables global pause ethertype processing. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_etype_gpp_*	16	I	Ethertype field for global pause processing. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_check_opcode_gpp_*	1	I	A value of 1 enables global pause opcode processing. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_opcode_gpp_*	16	I	Global pause opcode value. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_enable_ppp_*	1	I	A value of 1 enables priority pause packet processing. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_check_mcast_ppp_*	1	I	A value of 1 enables priority pause multicast destination address processing. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_check_ucast_ppp_*	1	I	A value of 1 enables priority pause unicast destination address processing. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_check_sa_ppp_*	1	I	A value of 1 enables priority pause source address processing. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_check_etype_ppp_*	1	I	A value of 1 enables priority pause ethertype processing. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_etype_ppp_*	16	I	Ethertype field for priority pause processing. This port is available when the AXI4-Lite interface is *not* selected.
ctl_rx_check_opcode_ppp_*	1	I	A value of 1 enables priority pause opcode processing. This port is available when the AXI4-Lite interface is *not* selected.

Table 292: **RX Pause Interface Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
ctl_rx_opcode_ppp_*	16	I	Priority pause opcode value. This port is available when the AXI4-Lite interface is *not* selected.
stat_rx_unicast_*	1	O	Increment for good unicast packets.
stat_rx_multicast_*	1	O	Increment for good multicast packets.
stat_rx_broadcast_*	1	O	Increment for good broadcast packets.
stat_rx_vlan_*	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_*	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.

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

Ports under this section are available when Enable_Time_Stamping is selected from the MAC Options tab.

Table 293: **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.
ctl_tx_ptp_1step_enable_*	1	I	When set to 1, this bit enables 1-step operation. This port is available when Include AXI4-Lite is *not* selected in the Configuration tab.
ctl_tx_ptp_latency_adjust_*	11	I	This bus can be used to adjust the 1-step TX timestamp with respect to the 2-step timestamp. The units of bits [10:3] are nanoseconds and bits [2:0] are fractional nanoseconds. This port is available when Include AXI4-Lite is *not* selected in the Configuration tab.
ctl_tx_ptp_vlane_adjust_mode_*	1	I	Sets the vlan adjust mode. This port is available when Include AXI4-Lite is *not* selected in the Configuration tab.
ctl_ptp_transpclk_mode_*	1	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 will add the correction value to the TX timestamp according to the process defined in IEEE 1588v2. The sign bit of the correction value is assumed to be 0 (positive time). It is expected that the corresponding incoming PTP packet correction field has already been adjusted with the proper RX timestamp. This port is available when Include AXI4-Lite is *not* selected in the Configuration tab.
stat_tx_ptp_fifo_read_error_*	1	O	Transmit PTP FIFO write error. A 1 on this status indicates that an error occurred during the PTP Tag write. A TX Path reset is required to clear the error.
stat_tx_ptp_fifo_write_error_*	1	O	Transmit PTP FIFO read error. A 1 on this status indicates that an error occurred during the PTP Tag read. A TX Path reset is required to clear the error.

Table 293: **IEEE 1588 TX/RX Interface Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
tx_ptp_1588op_in_*	2	I	2'b00 – No operation: no timestamp is taken and the frame is not 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 is not modified. 2'b11 – Reserved: act as No operation.
tx_ptp_tag_field_in_*	16	I	The use of this field is dependent on the 1588 operation.
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	Time stamp for the transmitted packet SOP corresponding to the time at which it passed the capture plane. Used for 2-step 1588 operation. The representation of the bits contained in this bus is the same as the timer input.
rx_ptp_tstamp_out_*	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_upd_chksum_in_*	1	I	TX UPD checksum value. This port is available when IEEE PTP Operation Mode is selected as One Step in the MAC options tab.
tx_ptp_tstamp_offset_in_*	16	I	TX PTP timestamp offset. This port is available when IEEE PTP Operation Mode is selected as One Step in the MAC options tab.
tx_ptp_chksum_offset_in_*	16	I	TX PTP check sum offset. This port is available when IEEE PTP Operation Mode is selected as One Step in the MAC options tab.
tx_ptp_pcslane_out_*	2	O	This bus identifies which of the PCS lanes that the SOP was detected on for the corresponding timestamp. Note that this signal will be valid starting at the same clock cycle during which the SOP is asserted for one of the segments.
rx_ptp_pcslane_out_*	2	O	This bus identifies which of the PCS lanes that the SOP was detected on for the corresponding timestamp. Note that this signal will be valid starting at the same clock cycle during which the SOP is asserted for one of the segments.
rx_lane_aligner_fill_0_*	7	O	This output indicates the fill level of the alignment buffer for PCS lane0. This information can be used by the PTP application, together with the signal rx_ptp_pcslane_out_*, to adjust for the lane skew of the arriving SOP. The units are SerDes clock cycles.
rx_lane_aligner_fill_1_*	7	O	This output indicates the fill level of the alignment buffer for PCS lane1.
rx_lane_aligner_fill_2_*	7	O	This output indicates the fill level of the alignment buffer for PCS lane2.
rx_lane_aligner_fill_3_*	7	O	This output indicates the fill level of the alignment buffer for PCS lane3.

AN and LT Interface Control/Status/Statistics Signals

Ports under this section are available when Include AN/LT Logic is selected from the Configuration tab.

Table 294: **AN and LT Interface Control/Status/Statistics Signals**

Name	Size	I/O	Description
an_clk_*	1	I	Input Clock for the Auto-Negotiation circuit. This should be a free running clock.
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, and 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.
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.
ctl_autoneg_enable_*	1	I	Enable signal for auto-negotiation. This port is available when the AXI4-Lite interface is *not* selected.
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. This port is available when the AXI4-Lite interface is *not* selected.
ctl_an_nonce_seed_*	8	I	8-bit seed to initialize the nonce field polynomial generator. This port is available when the AXI4-Lite interface is *not* selected.
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. This port is available when the AXI4-Lite interface is *not* selected.
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. This port is available when the AXI4-Lite interface is *not* selected.
ctl_an_local_fault_*	1	I	This input signal is used to set the remote_fault bit of the transmit link codeword. This port is available when the AXI4-Lite interface is *not* selected.
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. This port is available when the AXI4-Lite interface is *not* selected.
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. This port is available when the AXI4-Lite interface is *not* selected.

Table 294: **AN and LT Interface Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
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. This port is available when the AXI4-Lite interface is *not* selected.
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. This port is available when the AXI4-Lite interface is *not* selected.
ctl_an_fec_25g_baser_request_*	1	I	This signal is used to signal the link partner that the local station is requesting clause 74 FEC on the 25Gb/s lane protocols. This port is available when the AXI4-Lite interface is *not* selected.
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, 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. This port is available when the AXI4-Lite interface is *not* selected.
ctl_an_loc_np_*	1	I	Local Next Page indicator. If this bit is a 1, the AN IP transfers the next page word at input loc_np_data to the remote link partner. If this bit is 0, 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, 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. This port is available when the AXI4-Lite interface is *not* selected.
ctl_an_cl91_fec_ability_*	1	I	This bit is used to set clause 91 FEC ability. This port is available when the AXI4-Lite interface is *not* selected.
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	

Table 294: **AN and LT Interface Control/Status/Statistics Signals** *(cont'd)*

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. This port is available when the AXI4-Lite interface is *not* selected.
ctl_lt_restart_training_*	1	I	This signal triggers a restart of link training regardless of the current state. This port is available when the AXI4-Lite interface is *not* selected.
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. This port is available when the AXI4-Lite interface is *not* selected.
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. This port is available when the AXI4-Lite interface is *not* selected.
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. This port is available when the AXI4-Lite interface is *not* selected.
ctl_lt_pseudo_seed0_*	11	I	This 11- bit signal seeds the training pattern generator. This port is available when the AXI4-Lite interface is *not* selected.
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. This port is available when the AXI4-Lite interface is *not* selected.
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. This port is available when the AXI4-Lite interface is *not* selected.
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. This port is available when the AXI4-Lite interface is *not* selected.
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. This port is available when the AXI4-Lite interface is *not* selected.
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. This port is available when the AXI4-Lite interface is *not* selected.
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. This port is available when the AXI4-Lite interface is *not* selected.
ctl_lt_pseudo_seed1_*	11	I	This 11- bit signal seeds the training pattern generator. This port is available when the AXI4-Lite interface is *not* selected.
ctl_lt_k_p1_to_tx1_*	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. This port is available when the AXI4-Lite interface is *not* selected.

Table 294: **AN and LT Interface Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
ctl_lt_k0_to_tx1_*	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. This port is available when the AXI4-Lite interface is *not* selected.
ctl_lt_k_m1_to_tx1_*	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. This port is available when the AXI4-Lite interface is *not* selected.
ctl_lt_stat_p1_to_tx1_*	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. This port is available when the AXI4-Lite interface is *not* selected.
ctl_lt_stat0_to_tx1_*	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. This port is available when the AXI4-Lite interface is *not* selected.
ctl_lt_stat_m1_to_tx1_*	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. This port is available when the AXI4-Lite interface is *not* selected.
ctl_lt_pseudo_seed2_*	11	I	This 11- bit signal seeds the training pattern generator. This port is available when the AXI4-Lite interface is *not* selected.
ctl_lt_k_p1_to_tx2_*	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. This port is available when the AXI4-Lite interface is *not* selected.
ctl_lt_k0_to_tx2_*	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. This port is available when the AXI4-Lite interface is *not* selected.
ctl_lt_k_m1_to_tx2_*	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. This port is available when the AXI4-Lite interface is *not* selected.
ctl_lt_stat_p1_to_tx2_*	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. This port is available when the AXI4-Lite interface is *not* selected.
ctl_lt_stat0_to_tx2_*	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. This port is available when the AXI4-Lite interface is *not* selected.
ctl_lt_stat_m1_to_tx2_*	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. This port is available when the AXI4-Lite interface is *not* selected.
ctl_lt_pseudo_seed3_*	11	I	This 11- bit signal seeds the training pattern generator. This port is available when the AXI4-Lite interface is *not* selected.
ctl_lt_k_p1_to_tx3_*	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. This port is available when the AXI4-Lite interface is *not* selected.

Table 294: **AN and LT Interface Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
ctl_lt_k0_to_tx3_*	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. This port is available when the AXI4-Lite interface is *not* selected.
ctl_lt_k_m1_to_tx3_*	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. This port is available when the AXI4-Lite interface is *not* selected.
ctl_lt_stat_p1_to_tx3_*	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. This port is available when the AXI4-Lite interface is *not* selected.
ctl_lt_stat0_to_tx3_*	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. This port is available when the AXI4-Lite interface is *not* selected.
ctl_lt_stat_m1_to_tx3_*	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. This port is available when the AXI4-Lite interface is *not* selected.
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 10: Not Used 11: ENABLE; PCS is connected for mission mode
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_*			
stat_an_link_cntl_50gbase_kr2_*			
stat_an_link_cntl_50gbase_cr2_*			
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	Indicates a parallel detection fault during auto-negotiation.

Table 294: **AN and LT Interface Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
stat_an_start_tx_disable_*	1	O	This signal is asserted during auto-negotiation for one AN_CLK period to signal the start of the TX_DISABLE phase at the very start of auto-negotiation.
stat_an_start_an_good_check_*	1	O	This signal is asserted during auto-negotiation for one AN_CLK period to signal the start of the AN_GOOD_CHECK phase, when the selected protocol has been enabled, and the circuit is awaiting rx_pcs_status.
stat_an_lp_ability_1000base_kx_*	1	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.
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_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 10 Gb/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 10 Gb/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 25 Gb/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 25 Gb/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 294: **AN and LT 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 oc_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.
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_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.

Table 294: **AN and LT Interface Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
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.
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.
stat_lt_k_p1_from_rx1_*	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_rx1_*	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_rx1_*	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_rx1_*	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_rx1_*	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.
stat_lt_stat_m1_from_rx1_*	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_k_p1_from_rx2_*	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_rx2_*	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_rx2_*	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_rx2_*	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_rx2_*	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.
stat_lt_stat_m1_from_rx2_*	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_k_p1_from_rx3_*	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_rx3_*	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_rx3_*	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_rx3_*	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_rx3_*	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.
stat_lt_stat_m1_from_rx3_*	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.

Table 294: **AN and LT Interface Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
stat_an_rxcdrhold_*	1	O	Indicates the rx cdr hold signal.

Clause 74 FEC Interface Control/Status/Statistics Signals

Ports under this section are available when Clause 74 (BASE-KR FEC) is selected from the Configuration tab.

Table 295: **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.
stat_fec_inc_correct_count_*	4	O	This signal is asserted roughly every 32 words, while the ctl_rx_fec_enable is asserted, if the FEC decoder detected and corrected bit errors in the corresponding frame.
stat_fec_inc_cant_correct_count_*	4	O	This signal is asserted roughly every 32 words, while the ctl_rx_fec_enable is asserted, if the FEC decoder detected bit errors in the frame that it was unable to correct.
stat_fec_lock_error_*	4	O	ctl_fec_rx_enable 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.

Clause 10B RS-FEC Interface Control/Status/Statistics Signals

Ports under this section are available when Clause 91 (RS-FEC) is selected from the Configuration tab.

Table 296: **Clause 10B RS-FEC Interface Control/Status/Statistics Signals**

Name	Size	I/O	Description
ctl_rsfec_enable_*	1	I	Changes only take effect after the reset. New value is sampled on the first cycle after reset. Enable RS-FEC function. 1: Enable RS-FEC 0: Bypass RS-FEC

Table 296: **Clause 10B RS-FEC Interface Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
ctl_rx_rsfec_enable_correction_*	1	I	Changes only take effect after the reset. New value is sampled on the first cycle after reset. Equivalent to MDIO register 1.200.0 0: Decoder performs error detection without error correction (see IEEE 802.3802.3 by section 91.5.3.3) 1: the decoder also performs error correction
ctl_rx_rsfec_enable_indication_*	1	I	Changes only take effect after the reset. New value is sampled on the first cycle after reset. Equivalent to MDIO register 1.200.1 0: Bypass the error indication function (see IEEE Std 802.3 by section 91.5.3.3) 1: Decoder indicates errors to the PCS sublayer
ctl_rsfec_ieee_error_indication_mode_*	1	I	Changes only take effect after the reset. New value is sampled on the first cycle after reset. 1: Core conforms to the IEEE RS-FEC specification 0: If ctl_rx.rsfec_enable_correction and ctl_rx_rsfec_enable_indication are 0, the RS decoder is bypassed
stat_tx_rsfec_block_lock_*	1	O	TX PCS block lock status 0: unlocked 1: locked
stat_tx_rsfec_lane_alignment_status_*	1	O	TX PCS frame alignment status 0: unaligned 1: aligned
stat_rx_rsfec_am_lock0_*	1	O	RX lane 1 lock status 0: unlocked 1: locked
stat_rx_rsfec_am_lock1_*	1	O	RX lane 1 lock status 0: unlocked 1: locked
stat_rx_rsfec_lane_alignment_status_*	1	O	RX alignment status 0: unaligned 1: aligned
stat_rx_rsfec_lane_fill_0_*	14	O	RX lane 0 additional delay. The top seven bits [13:7] of each delay bus is the number of additional clock cycles delay being added due to deskew. The bottom seven bits [6:0] of each delay bus is the fractional clock cycle delay being added due to deskew, in units of 1/66th of a clock cycle.
stat_rx_rsfec_lane_fill_1_*	14	O	RX lane 1 additional delay. The top seven bits [13:7] of each delay bus is the number of additional clock cycles delay being added due to deskew. The bottom seven bits [6:0] of each delay bus is the fractional clock cycle delay being added due to de-skew, in units of 1/66th of a clock cycle.
stat_rx_rsfec_lane_mapping_*	2	O	RX lane mapping bit 0: index of FEC lane carried on PMA lane 0 bit 1: index of FEC lane carried on PMA lane 1

Table 296: **Clause 10B RS-FEC Interface Control/Status/Statistics Signals** *(cont'd)*

Name	Size	I/O	Description
stat_rx_rsfec_hi_ser_*	1	O	This output is only active when the core is in bypass indication mode. It indicates when High that the number of FEC symbol errors in a window of 8192 codewords has exceeded the threshold K (417)
stat_rx_rsfec_corrected_cw_inc_*	1	O	Corrected codeword count increment
stat_rx_rsfec_uncorrected_cw_inc_*	1	O	Uncorrected codeword count increment
stat_rx_rsfec_error_count0_inc_*	3	O	Symbol error count increment for lane 0.
stat_rx_rsfec_error_count1_inc_*	3	O	Symbol error count increment for lane 1.

Runtime Switch Signals

Ports under this section are available when the Runtime switch is selected from the Configuration tab.

Table 297: **Runtime Switch Signals**

Name	Size	I/O	Description
gt_drp_done_0	1	I	Indicates the completion of the GT DRP operation. Used to reset the GT module.
ctl_rate_mode_0	1	I	This signal causes the IP core to switch between 50G operation (0) and 40G operation (1). Note that the clock frequencies must be corrected for the mode chosen
txpllclksel_in_0	8	I	Selects the source TX PLL clock to generate TXOUTCLK from GTWIZ IP
rxpllclksel_in_0	8	I	Selects the source RX PLL clock to generate RXOUTCLK from GTWIZ IP
txsysclksel_in_0	8	I	Selects the source TX sys clock for GT Channel
rxsysclksel_in_0	8	I	Selects the source RX sys clock for GT Channel
rxafecfoken_0	4	I	Connects to RXAFECFOKEN on transceiver channel primitives
rxdfecfokfcnum_0	16	I	Connects to RXDFECFOKFCNUM on transceiver channel primitives
speed_0	1	I	This signal indicates the speed with which the core is working: 1'b1 = 40G and 1'b0 = 50G
anlt_done_0	1	O	Indicates the completion of Auto negotiation and Link Training This port is available when Include AN/LT logic is selected in the Configuration tab.
rxdata_out_0	512	I	RX data bus from GT IP to MAC This port is available when Include GT subcore in example design option is selected in the GT Selection and Configuration tab.
txdata_in_0	512	O	TX data bus from MAC to GT IP This port is available when Include GT subcore in example design option is selected in the GT Selection and Configuration tab.
axi_ctl_core_mode_switch	1	O	This signal can be used to switch Line rate from 40G to 50G and vice-verse when you select Include AXI4-Lite in the Configuration tab and write 1 to the 0x013C self clearing register to start the GT DRP operations.

Table 297: **Runtime Switch Signals** *(cont'd)*

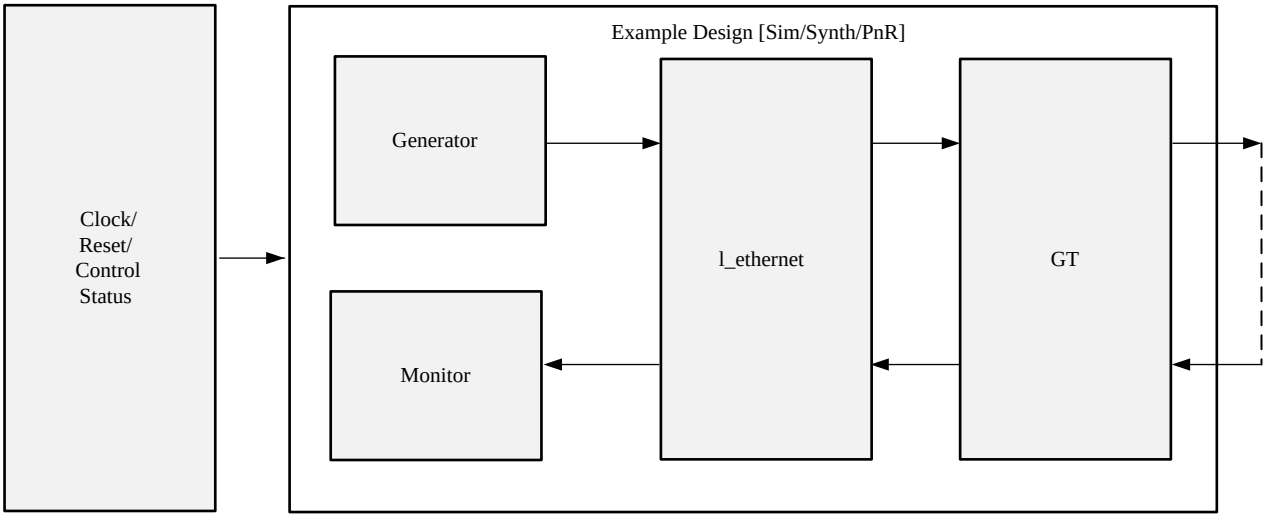
Name	Size	I/O	Description
user_reg0_*	32	O	User-defined signal from the AXI4 register map user_reg0 register. This port is available when Include AXI4-Lite is selected in the Configuration tab.

Duplex Mode of Operation

In this mode of operation, both the core transmitter and receiver are active and loop back 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 transmitting to the core using the available data interface. Monitor module checks the packets from the receiver.

The following figure shows the duplex mode of operation.

Figure 51: **Duplex Mode of Operation**



Run Time Switchable

This configuration gives the flexibility to switch the line rate between 40G to 50G and vice-versa any time. To activate this feature, select the check box **Runtime Switchable mode** option in the Configuration tab. When this option is selected the `*_trans_debug` module is present inside the `*_pkt_gen_mon.v` module of the example design. This `*_trans_debug` module is responsible for performing all the GT DRP write operations to switch the transceiver mode, that is, 40G to 50G or 50G to 40G. When you set the `mode_change_* input` signal High for two clock cycles and then make it Low, it starts the DRP write operation to the GT channel for the specific core and resets the specific core. The DRP writes are done only for the channel. The QPLL0 of the common is fixed for the line rate 50G and the QPLL1 is fixed for the line rate 40G.

The state transition occurred during this process is shown in the following figure.

Figure 52: **State Transition Diagram for Run Time Switchable DRP Operation without AN/LT**

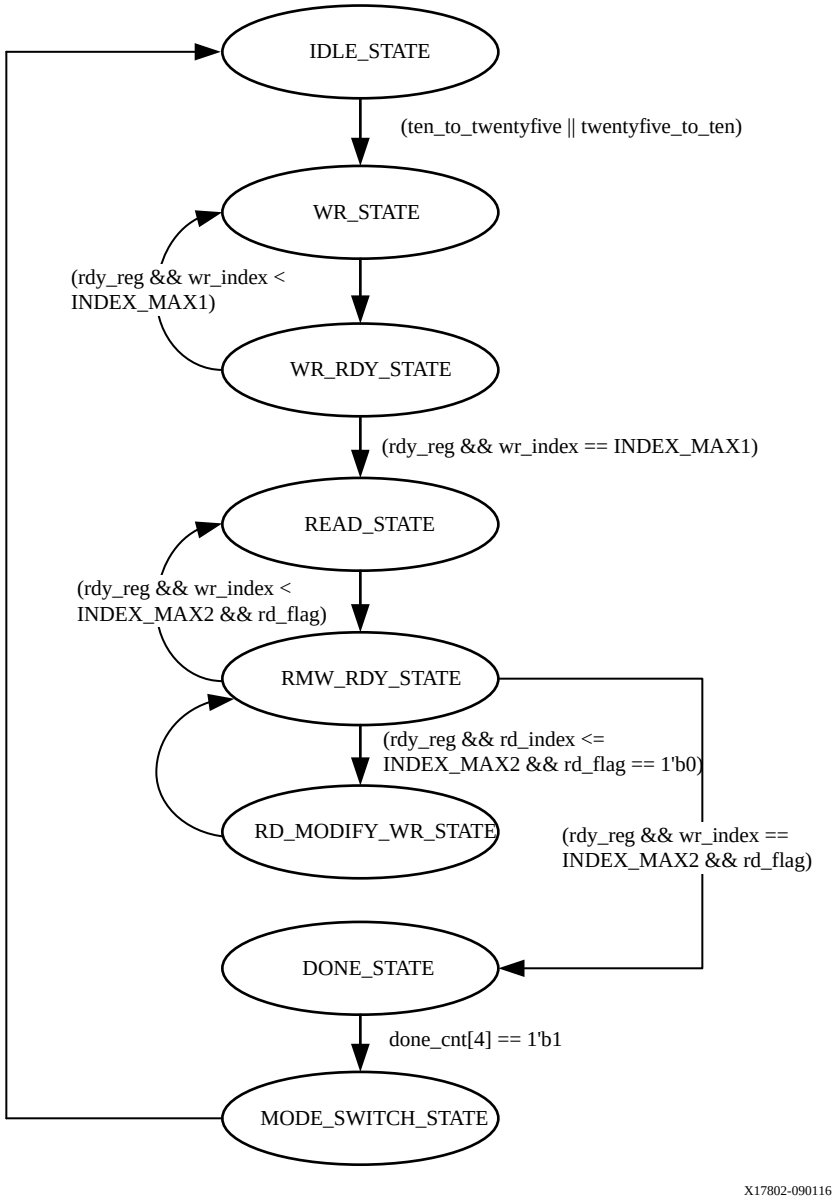
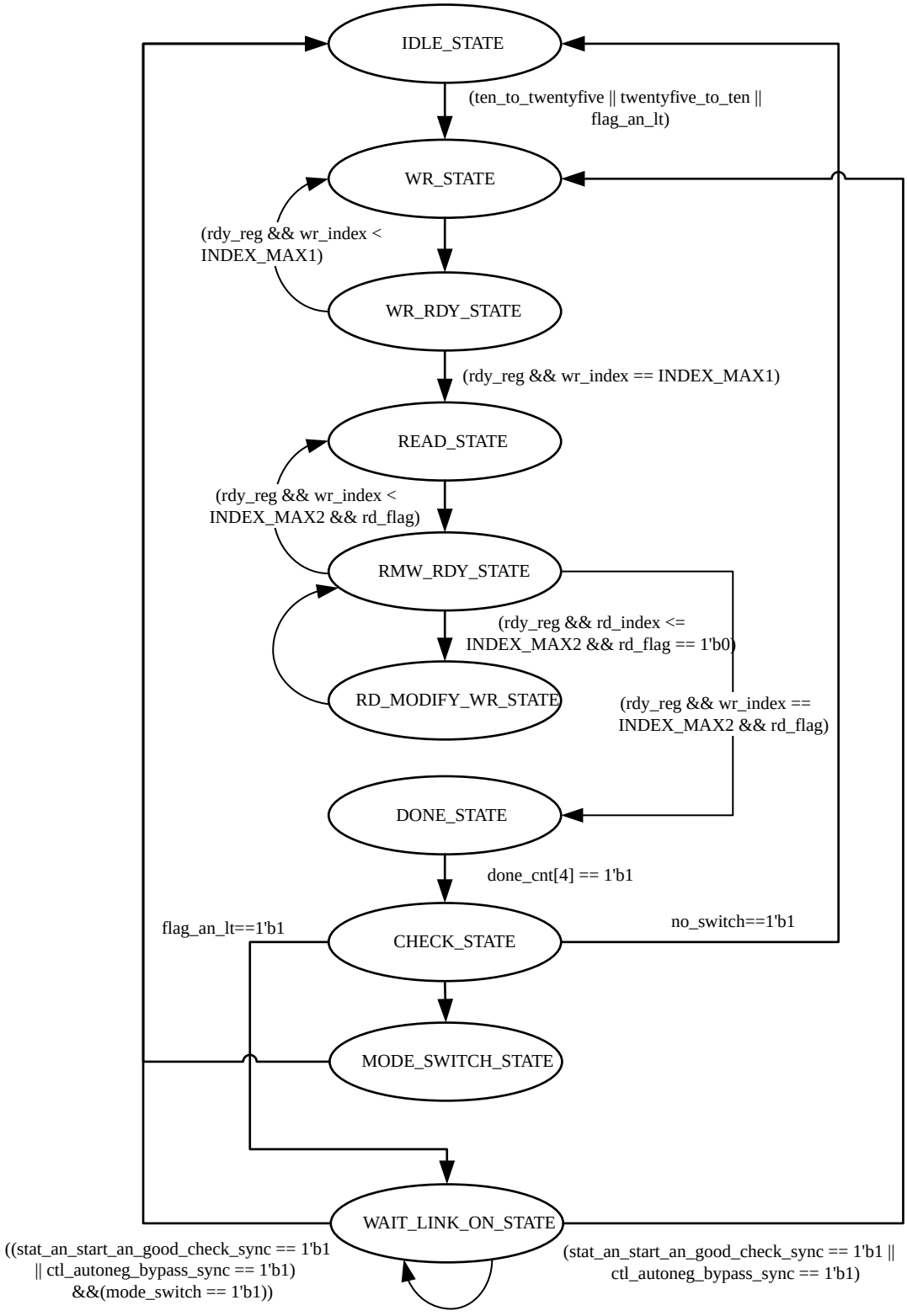


Figure 53: **State Transition Diagram for Run Time Switchable DRP Operation with AN/LT**



Shared Logic Implementation

Shared logic includes the GT common module that can be present as part of the core or in the example design.

By default GT common, reset logic and clocking modules are present inside the IP core. In case of the following conditions, these modules will be placed outside the core so that they can be shared with other designs.

- When you select the **Include GT subcore** in example design option in the GT Selection and Configuration tab.
- When you select the **Include Shared Logic in Example Design** option in the Shared Logic tab.

When the shared logic in the example design is selected, a new l_ethernet_*_core_support.v module will be instantiated between the l_ethernet_*_exdes.v and DUT (that is, l_ethernet_*.v). This module will have all the sub modules that can be shared between multiple designs.

The following figure shows the implementation when shared logic is instantiated in the example design for a single core.

Figure 54: **Single Core Example Design Hierarchy with Shared Logic Implementation**

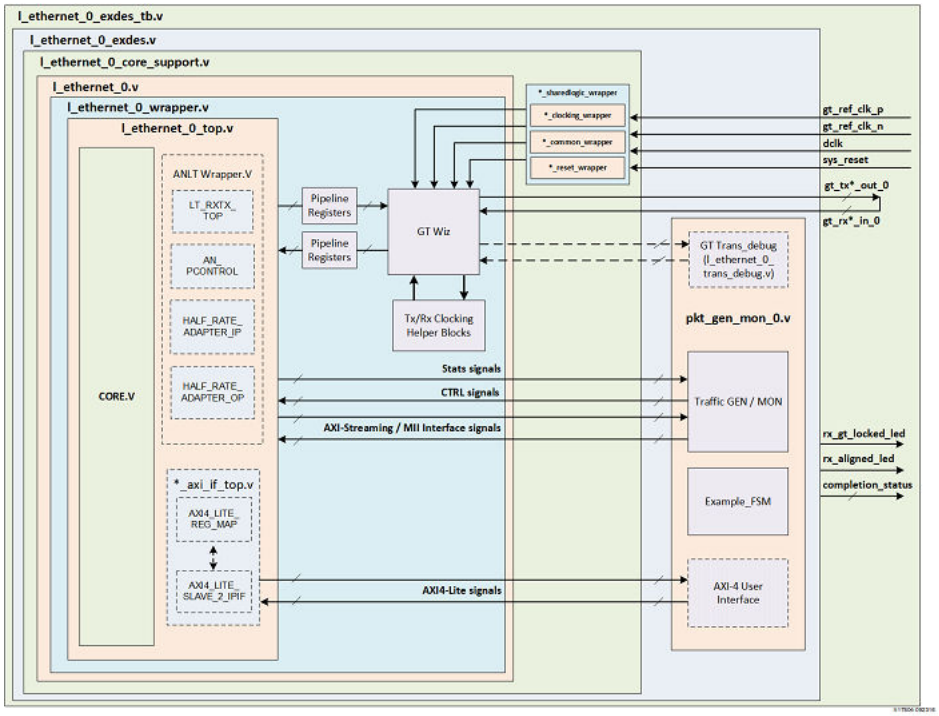
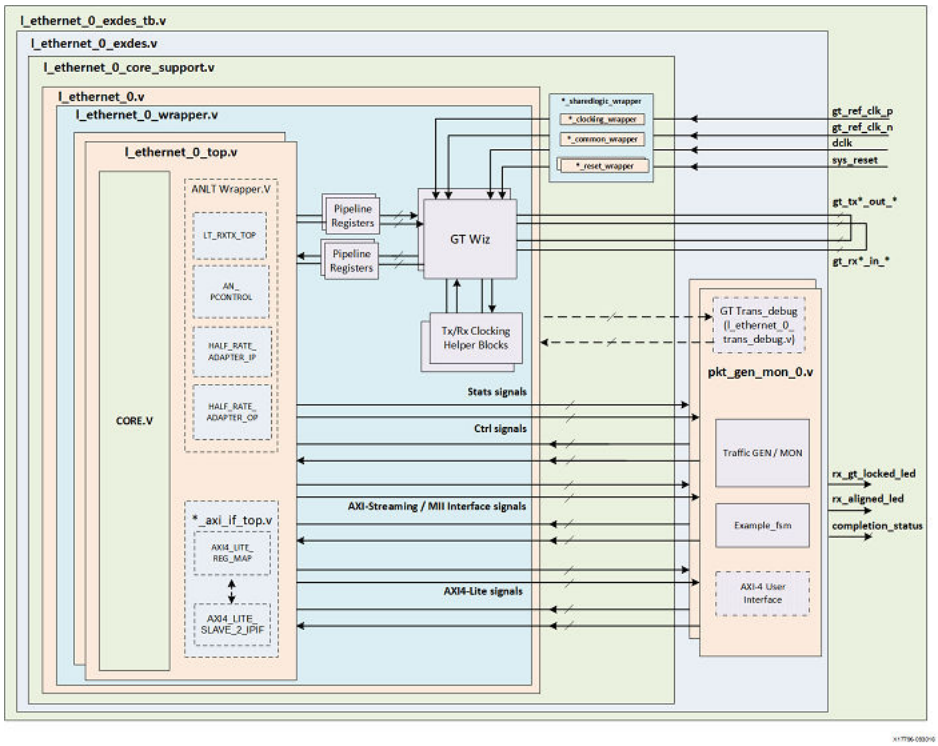


Figure 55: **Multiple Core Example Design Hierarchy with Shared Logic Implementation**



The following modules are the part of shared logic wrapper.

- ***_clocking_wrapper:** This module contains all the clk resource which can be shared with other designs.
- ***_common_wrapper:** This module contains the GT common which can be shared with other designs.
- ***_reset_wrapper:** This module contains all the reset logics for the user specified Vivado IDE configuration.

AXI4-Lite Interface Implementation

If you want to instantiate the AXI4-Lite interface to access the control and status registers of the l_ethernet core, you must enable the **Include AXI4-Lite** check box in the Configuration tab. It enables the l_ethernet _0_axi_if_top module (that contains l_ethernet_0_pif_registers with the l_ethernet _0_slave_2_ipif module). You can accesses the AXI4-Lite interface logic registers (control, status and statistics) from the l_ethernet _0_pkt_gen_mon module.

This mode enables the following features:

- You can configure all the control (CTL) ports of the core through the AXI4-Lite interface. This operation is performed by writing to a set of address locations with the required data to the register map interface.
- You can access all the status and statistics registers from the core through the AXI4-Lite interface. This is performed by reading the address locations for the status and statistics registers through register map.

.h Header File

AXI4 register information such as register address, register name with bit position, mask value, access type and their default values are provided in the header (.h) file format when the IP core is generated with Include AXI4-Lite enabled in the Vivado Design Suite and the header file can be found under the folder header_files of the project path.

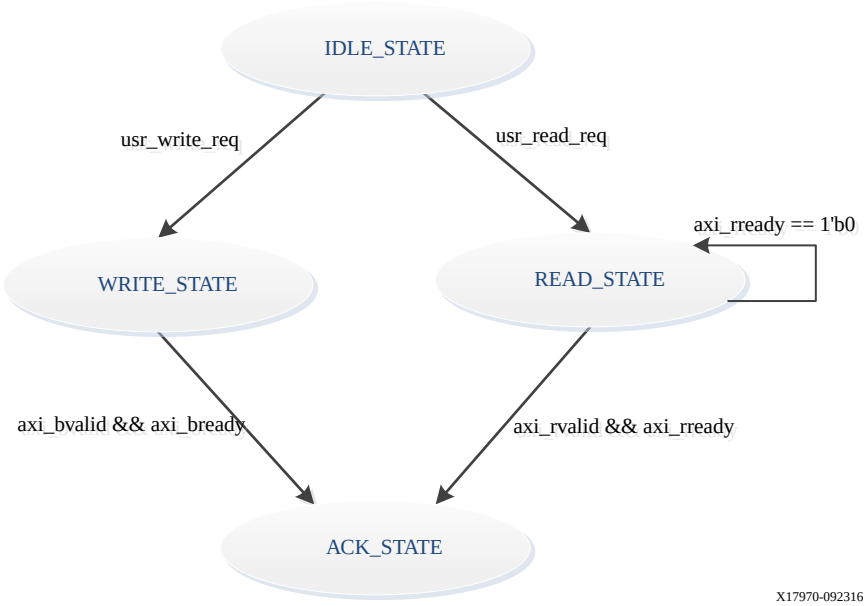
AXI4-Lite Interface User Logic

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 in the following figure:

Figure 56: **User State Machine for the 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 READ_STATE else if `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 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.

AXI4-Lite User Interface Ports

Table 298: **AXI4-Lite 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

Table 298: **AXI4-Lite User Interface Ports** *(cont'd)*

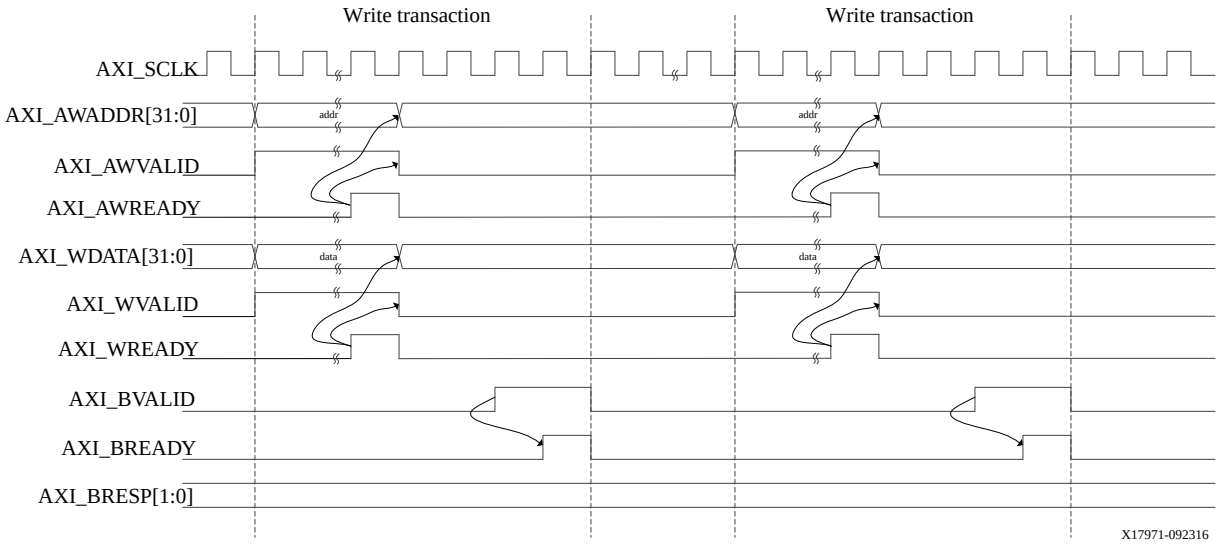
Name	Size	I/O	Description
S_AXI_RRESP	2	O	AXI read response. This signal indicates the status of the read transfer. 'b00 = OKAY 'b01 = EXOKAY 'b10 = SLVERR 'b11 = DECERR
S_AXI_RVALID	1	O	AXI read data valid
S_AXI_RREADY	1	I	AXI read ready. This signal indicates the user/master can accept the read data and response information.

User Side AXI4-Lite Write / Read Transactions

The following figures show timing diagram waveforms for the AXI4-Lite interface.

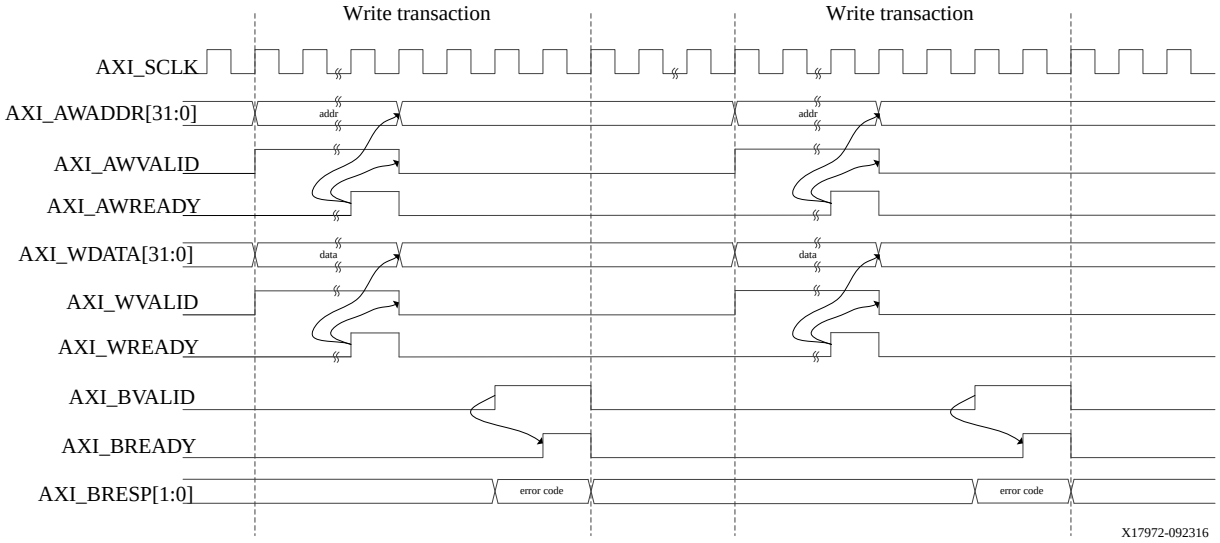
Valid Write Transactions

Figure 57: **AXI4-Lite User Side Write Transaction**



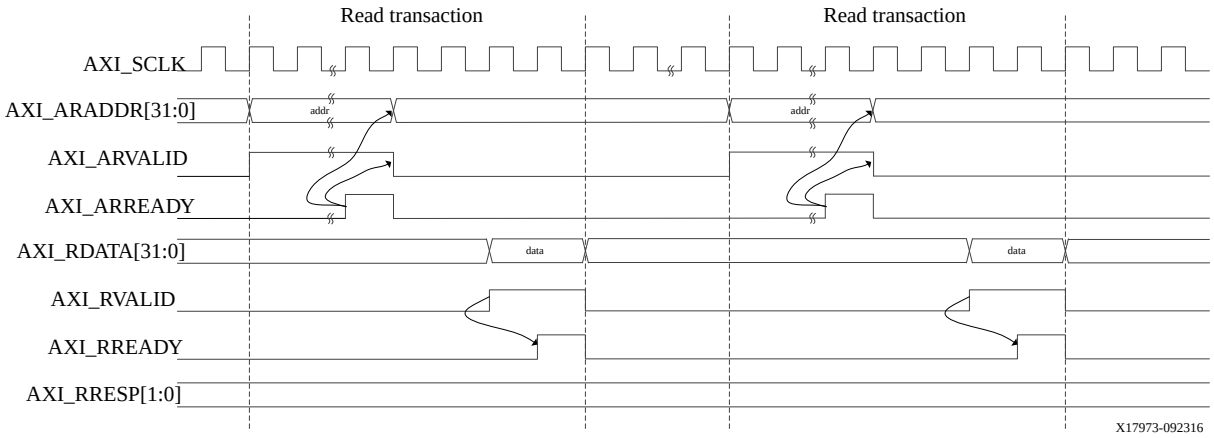
Invalid Write Transactions

Figure 58: **AXI4-Lite User Side Write Transaction with Invalid Write Address**



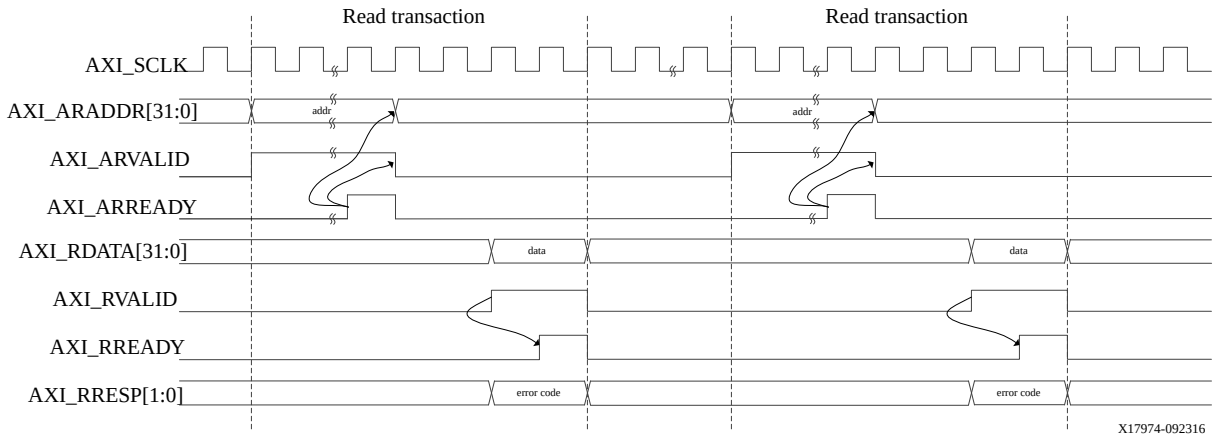
Valid Read Transactions

Figure 59: **AXI4-Lite User Side Read Transaction**



Invalid Read Transactions

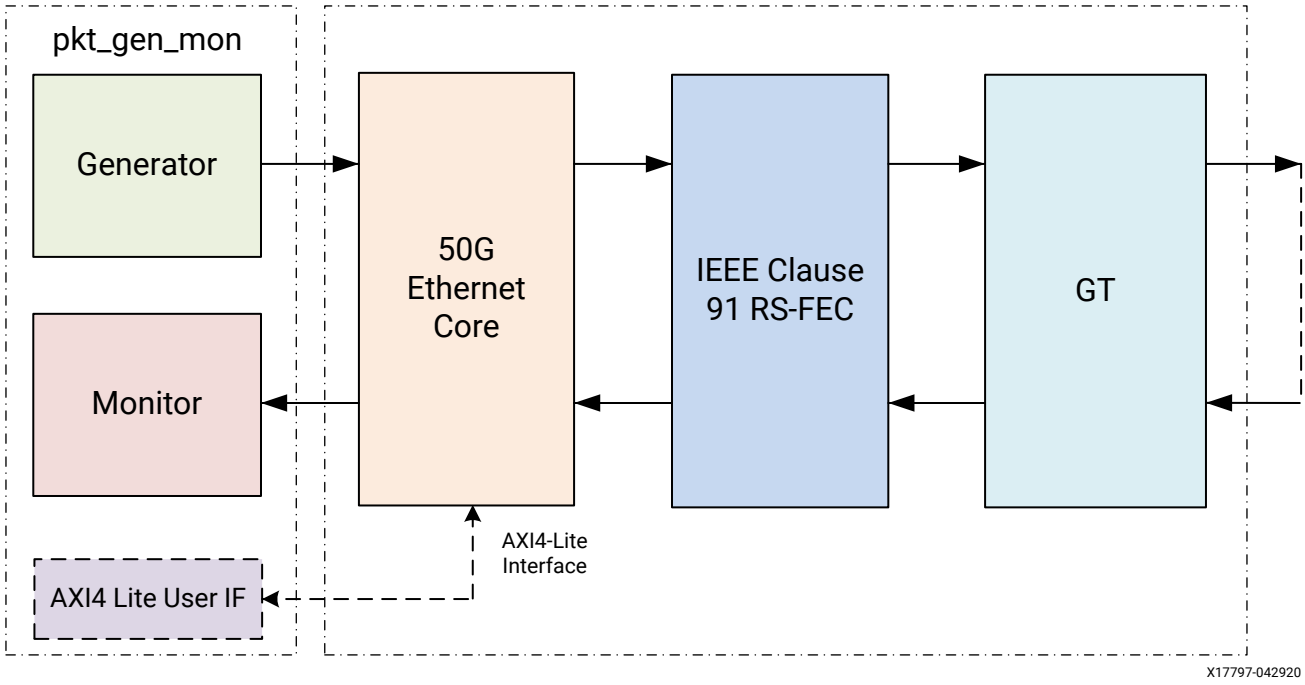
Figure 60: **AXI4-Lite User Side Read Transaction with Invalid Read Address**



IEEE Clause 91 (RS-FEC) Integration

If you want to include IEEE clause 91 RS-FEC soft IP (for error correction) in between 50G Ethernet IP and the GT, you must select the Include Clause 91 (RS-FEC) check box in the Configuration tab. This option is available only for 50G speed.

Figure 61: **RS-FEC Integration in between 50G Ethernet IP and GT**



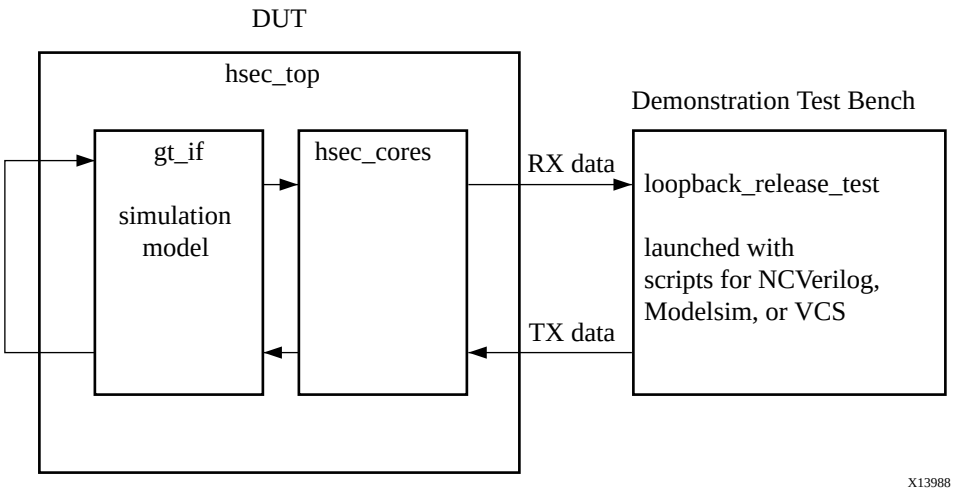
This feature enables the IEEE Clause 91 RS-FEC soft IP component instantiated in between the 50G core and the GT. The TX SerDes lines from the 50G core will be input to the RS-FEC soft IP for forward error correction encoding. The output from the RS-FEC module is then fed to the GT. Similarly, the RX SerDes lines from the GT will be fed to the RS-FEC module for error correction decoding and then to the 50G core.

Refer to the *50G IEEE 802.3 Reed-Solomon Forward Error Correction LogiCORE IP Product Guide* (PG234) (registration required) for IEEE clause 91 Reed-Solomon Forward Error Correction for the LogiCORE™ IP core and its functionality.

Test Bench

Each release includes a demonstration test bench that performs a loopback test on the complete 40G/50G High Speed Ethernet Subsystem. The test program exercises the datapath to check that the transmitted frames are received correctly. RTL simulation models for the 40G/50G High Speed Ethernet 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 62: **Test Bench**



Upgrading

The following sections describe the changes for each release.

Changes from v2.2 to v2.3

Port Changes

- Corrected port name from `rx_serdes_resetn` to `rx_serdes_reset` in figure in Port Descriptions.
- Added text about 2-step 1588 operation to `tx_ptp_tstamp_out[80-1:0]` and `tx_ptp_tstamp_out_*` descriptions.
- `ctl_tx_ptp_latency_adjust[10:0]` signal

  Deleted sentence about 802 decimal clock mode value.
- Updated `axi_ctl_core_mode_switch` to update 0x018C to 0x013C.
- Changed name from `restart_tx_rx_0` to `restart_tx_rx_*`

Ports Added

- `tx_ptp_upd_chksum_in`
- `tx_ptp_pcslane_out`
- `tx_ptp_chksum_offset_in`
- `rx_ptp_pcslane_out`
- `rx_lane_aligner_fill`
- `rx_ptp_tstamp_valid_out`
- `send_continuous_pkts_*`
- `restart_tx_rx_0`
- `mode_change_0`
- `core_speed_0`

Ports Removed

- tx_ptp_rxtstamp_in
- tx_ptp_rxtstamp_in

Registers Added

- STAT_CORE_SPEED_REG: 047C
- user_reg0_*
- RXOUTCLKSEL_IN_*
- TXOUTCLKSEL_IN_*

Register Name Changed

- Updated SWITCH_CORE_SPEED_REG: 0180 to SWITCH_CORE_SPEED_REG: 013C.
- Changed USER_REG_0: 0184 to USER_REG_0: 0138.

Registers Deleted

- USER_REG_1: 0188
- CORE_SPEED_REG:180

Changes from v2.1 to v2.2

Ports Added

Added gtpowergood_out_*.

Port Descriptions Updated

- tx_axis_tuser.
- dclk.
- rx_clk
- gt_refclk_p, gt_refclk_n, and dclk.
- an_clk_*.

Register Descriptions Updated

- ctl_gt_reset_all.
- tick_reg.

- stat_rx_rsfec_am_lock0, stat_rx_rsfec_am_lock1.

Changes from v2.0 to v2.1

Features Added

256-bit AXI4-Stream Packet interface for 40Gb/s MAC+PCS.

Features Updated

Auto-negotiation and Link Training with reduced resource utilization.

Ports Added

- stat_tx_underflow_err
- stat_tx_overflow_err

Ports Updated

- tx_axis_tvalid
- tx_mii_reset
- rx_mii_clk
- gt_refclk_out
- tx_mii_d_*
- tx_mii_c_*
- rx_mii_d_*
- rx_mii_c_*
- axi_ctl_core_mode_switch

Register Added

STAT_STATUS_REG1: 0408

Changes from v2.0 (10/05/2016) to v2.0 (11/30/2016 version)

Port Name Changes

- `qpll0clk_in` to `qpll0_clk_in_*`
- `qpll0refclk_in` to `qpll0_refclk_in_*`
- `qpll1clk_in` to `qpll1_clk_in_*`
- `qpll1refclk_in` to `qpll1_refclk_in_*`
- `gtwiz_reset_qpll0lock_in` to `gtwiz_reset_qpll0_lock_in_*`
- `gtwiz_reset_qpll0lock_out` to `gtwiz_reset_qpll0_lock_out_*`
- `Stat_rx_rsfec_symbol_error_count 0_inc_*` to `Stat_rx_rsfec_error_count 0_inc_*`
- `Stat_rx_rsfec_symbol_error_count 1_inc_*` to `Stat_rx_rsfec_error_count 1_inc_*`
- `Stat_tx_rsfec_pcs_block_lock_*` to `Stat_tx_rsfec_block_lock_*`

Ports Added

- `ctl_tx_ptp_1step_enable_*`
- `ctl_tx_ptp_latency_adjust_*`
- `ctl_tx_ptp_vlane_adjust_mode_*`
- `ctl_ptp_transpclk_mode_*`
- `tx_ptp_upd_chksum_in_*`
- `tx_ptp_chksum_offset_in_*`
- `tx_ptp_rxtstamp_in_*`
- `stat_an_rxcdrhold_*`
- `gt_drp_done_0`
- `ctl_rate_mode_0`
- `txpllclksel_in_0`
- `rxpllclksel_in_0`
- `txsysclksel_in_0`
- `rxsysclksel_in_0`

- Rxafecfoken_0
- Rxdfecfokfcnum_0
- Speed_0
- anlt_done_0
- rxdata_out_0
- txdata_in_0

Changes from v1.1 to v2.0

Hex Addresses Added

- CONFIGURATION_1588_REG: 0038
- CONFIGURATION_1RSFEC_REG: 00D0
- CORE_SPEED_REG: 0180
- USER_REG_0: 0184
- USER_REG_1: 0188
- STAT_RX_RSFEC_STATUS_REG: 0418
- STAT_RX_RSFEC_LANE_FILL_REG1: 0448
- STAT_TX_RSFEC_STATUS_REG: 044C
- STAT_RX_RSFEC_CORRECTED_CW_INC_LSB: 0670
- STAT_RX_RSFEC_CORRECTED_CW_INC_MSB: 0674
- STAT_RX_RSFEC_UNCORRECTED_CW_INC_LSB: 0678
- STAT_RX_RSFEC_UNCORRECTED_CW_INC_MSB: 067C
- STAT_RX_RSFEC_ERR_COUNT0_INC_LSB: 0680
- STAT_RX_RSFEC_ERR_COUNT0_INC_MSB: 0684
- STAT_RX_RSFEC_ERR_COUNT1_INC_LSB: 0688
- STAT_RX_RSFEC_ERR_COUNT1_INC_MSB: 068C

Register Tables Added

- CONFIGURATION_1588_REG: 0038
- CONFIGURATION_1RSFEC_REG: 00D0
- CORE_SPEED_REG: 0180

- USER_REG_0: 0184
- USER_REG_1: 0188
- STAT_RX_RSFEC_STATUS_REG: 0418
- STAT_RX_RSFEC_LANE_FILL_REG1: 0448
- STAT_TX_RSFEC_STATUS_REG: 044C
- STAT_RX_RSFEC_CORRECTED_CW_INC_LSB: 0670
- STAT_RX_RSFEC_CORRECTED_CW_INC_MSB: 0674
- STAT_RX_RSFEC_UNCORRECTED_CW_INC_LSB: 0678
- STAT_RX_RSFEC_UNCORRECTED_CW_INC_MSB: 067C
- STAT_RX_RSFEC_ERR_COUNT0_INC_LSB: 0680
- STAT_RX_RSFEC_ERR_COUNT0_INC_MSB: 0684
- STAT_RX_RSFEC_ERR_COUNT1_INC_LSB: 0688
- STAT_RX_RSFEC_ERR_COUNT1_INC_MSB: 068C

Changes from v1.0 to v1.1

Ports Added

The following ports have been added.

- `ctl_gt_reset_all_*`
- `ctl_gt_reset_*`
- `ctl_gt_rx_reset_*`
- `gt_reset_all_in_*`
- `gt_tx_reset_in_*`
- `gt_rx_reset_in_*`
- `ctl_an_fec_25g_rs_request`
- `ctl_an_fec_25g_baser_request`
- `stat_an_lp_fec_25g_rs_request`
- `stat_an_lp_fec_25g_baser_request`
- `tx_ptp_rxtstamp_in`
- `rx_ptp_pcslane_out_*`

- ctl_an_fec_25g_baser_request_*

Ports Removed

stat_tx_underflow_err

Port Changes

- rx_serdes_reset_done_in_* to rx_serdes_reset_*.
- SERDES_WIDTH with 64 for several I/O ports.
- ctl_an_ability_25gbase_cr to ctl_an_ability_25gbase_krcr
- ctl_an_ability_25gbase_kr to ctl_an_ability_25gbase_krcr_s
- ctl_an_fec_request to ctl_an_fec_10g_request. Updated description.
- stat_rx_errored_block_increment_valid to stat_rx_bad_code_valid
- stat_rx_bad_sh_increment_0[3:0] to stat_rx_framing_err_0[3:0]
- stat_rx_bad_sh_increment_1[3:0] to stat_rx_framing_err_1[3:0]
- stat_rx_bad_sh_increment_2[3:0] to stat_rx_framing_err_2[3:0]
- stat_rx_bad_sh_increment_3[3:0] to stat_rx_framing_err_3[3:0]
- stat_rx_bad_sh_increment_valid_1 to stat_rx_valid_1
- stat_rx_bad_sh_increment_valid_2 to stat_rx_valid_2
- stat_rx_bad_sh_increment_valid_3 to stat_rx_valid_3
- rx_axis_tuser_0 to rx_axis_tuser_*

Hex Addresses Added

- CONFIGURATION_LT_SEED_REG1: 0114
- CONFIGURATION_LT_COEFFICIENT_REG1: 0134
- STAT_LT_COEFFICIENT1_REG: 0478
- STAT_RX_ERROR_LSB: 0668
- STAT_RX_ERROR_LSB: 066C

Signals Added to Register Definitions

- ctl_gt_rx_reset
- ctl_gt_tx_reset
- ctl_tx_ipg_value
- ctl_tx_custom_preamble_enable

- ctl_rx_custom_preamble_enable
- ctl_an_fec_25g_rs_request
- ctl_an_fec_25g_baser_request
- stat_an_lp_fec_25g_rs_request
- stat_an_lp_fec_25g_baser_request

Register Tables Added

- CONFIGURATION_LT_SEED_REG1: 0114
- CONFIGURATION_LT_COEFFICIENT_REG1: 0134
- STAT_LT_COEFFICIENT1_REG: 0478
- STAT_RX_ERROR_LSB: 0668
- STAT_RX_ERROR_MSB: 066C

XILINX

Appendix B

Debugging

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

Finding Help on Xilinx.com

To help in the design and debug process when using the 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.

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.

Master Answer Record for the 40G/50G High Speed Ethernet Subsystem

AR 54690.

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.

Debug Tools

There are many tools available to address 40G/50G High Speed Ethernet design issues. It is important to know which tools are useful for debugging various situations.

Example Design

The High Speed Ethernet IP core is delivered with an example design netlist complete with functional test benches. The design includes example transceivers and loopback tests for common simulator programs.

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 40G/50G High Speed Ethernet core. These boards can be used to prototype designs and establish that the core can communicate with the system.

UltraScale™ devices are recommended for optimum performance. Ensure that the board transceivers support the required Ethernet bit rate. For example, the following board is suitable for many UltraScale™ implementations: UltraScale FPGA evaluation board, VCU108.

Simulation Debug

Each High Speed Ethernet IP core release includes a sample simulation test bench. This typically consists of a loopback from the TX side of the user interface, through the TX circuit, looping back to the RX circuit, and checking the received packets at the RX side of the user interface.

Each release usually includes a sample instantiation of a Xilinx transceiver corresponding to the device selected by the customer. The loopback simulation includes a path through the transceiver.

The simulation is run using provided scripts for several common industry-standard simulators.

If the simulation does not run properly from the scripts, the following items should be checked.

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.

Library File Location

Each simulation script calls up the required Xilinx library files. These are called by the corresponding `liblist*` file in the bin directory of each release.

There can be an error message indicating that the simulator is unable to find certain library files. In this case, the path to the library files might have to be modified. Check with your IT administrator to ensure that the paths are correct.

Version Compatibility

Each release has been tested according the Xilinx tools version requested by customers. If the simulation does not complete successfully, you should first ensure that a properly up-to-date version of the Xilinx tools is used. The preferred version is indicated in the README file of the release, and is also indicated in the simulation sample log file included with the release.

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 you create your own test bench)
- Send fewer packets. This can be accomplished by modifying the appropriate parameter in the provided sample test bench.
- Specify a shorter time between alignment markers. This should result in a shorter lane alignment phase, at the expense of more overhead. However, when the High Speed Ethernet IP core is finally implemented in hardware, the distance between alignment markers should follow the specification requirements (after every 16,383 words).

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. See Simulation for more details.

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.

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.

Many of these common issues can also be applied to debugging design simulations. Details are provided in the following sections:

- General Checks
- Transceiver Specific Checks
- Ethernet Specific Checks

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 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.

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

Several issues can commonly occur during the first hardware test of an Ethernet IP core. These should be checked as indicated in the following subsections.

It is assumed that the Ethernet IP core has already passed all simulation testing which is being implemented in hardware. This is a prerequisite 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 each SerDes achieves clock data recovery (CDR) lock.
3. Check that each lane has achieved word alignment.
4. Check that lane alignment has been achieved.
5. Proceed to Interface and Protocol debug.

Signal Integrity

When bringing up a board for the first time and the High Speed Ethernet IP core does not seem to be achieving lane alignment, the most likely issue is related to signal integrity.

IMPORTANT! *Signal integrity issues must be addressed before any other debugging can take place.*

Even if lane alignment is achieved, if there are periodic bip-8 errors, signal integrity issues are indicated. Check the bip-8 signals to assist with debug.

Signal integrity should be debugged independently from the High Speed Ethernet IP core. The following procedures should be carried out.

Note: 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 Standard for Ethernet (IEEE Std 802.3-2015).

- Transceiver settings
- Checking for noise
- Bit error rate testing

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

Lane Swapping

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

N/P Swapping

If the positive and negative signals of a differential pair are swapped, data is not received correctly on that lane. You should verify that each link has the correct polarity of each differential pair.

Clocking and Resets

See Chapter 4: Designing with the Subsystem for these requirements.

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

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

If any subsequent instability is detected in a clock, the High Speed Ethernet IP core must be reset. One example of such instability is a loss of CDR lock. The user logic should determine all external conditions that would require a reset (for example, clock glitches, loss of CDR lock, 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 IP core is reset. An example of a configuration change would be setting a different maximum packet length. Check the description for the particular signal on the port list to determine if this requirement applies to the parameter that is being changed.

Interface Debug

AXI4-Stream Interfaces

The High Speed Ethernet IP core user interface is called the AXI4-Stream. There are two versions used — regular AXI4-Stream and straddled AXI4-Stream. 50Gb/s operation allows use of only straddled AXI4-Stream whereas 40Gb/s operation provides an option of 256-bit regular AXI4-Stream in addition to the 128-bit straddled interface. See the appropriate section in this guide for a detailed description of each.

TX Debug (Buffer Errors)

TX debug is assisted by using several diagnostic signals.

Data must be written to the TX AXI4-Stream such that there are no overflow or underflow conditions. The AXI4-Stream bandwidth must always be greater than the Ethernet bandwidth to guarantee that data can be sent without interruption.

When writing data to the AXI4-Stream, the `tx_rdyout` signal must always be observed. This signal indicates whether the fill level of the TX buffer is within an acceptable range or not. If this signal is ever asserted, you must stop writing to the TX AXI4-Stream until the signal is deasserted.

Because the TX AXI4-Stream has greater bandwidth than the TX Ethernet interface, it is not unusual to see this signal being frequently asserted and this is not a cause for concern. You must ensure that TX writes are stopped when `tx_rdyout` is asserted.

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

When a packet data transaction has begun in the TX direction, it must continue until completion or there can be a buffer underflow as indicated by the `tx_unfout` signal. This must not be allowed to occur; data must be written on the TX AXI4-Stream without interruption. Ethernet packets must be present on the line from start to end with no gaps or idles. If `tx_unfout` is ever asserted, debugging must stop until the condition which caused the underflow has been addressed.

***Note*:** When this signal sampled as 1, you must apply `tx_reset`/`sys_reset` to recover the core from the underflow issue. `tx_reset` resets the TX path only and `sys_reset` recovers the complete system.

RX Debug (Buffer Errors)

For a detailed description of the diagnostic signals which are available to debug the RX, see Port Descriptions.

If the Ethernet packets are being transmitted properly according to IEEE Std 802.3-2015, there should not be RX errors. However, the signal integrity of the received signals must be verified first.

The `stat_rx_bip_err` signals provide a per-lane indicator of signal quality. The `stat_rx_hi_ber` signal is asserted when the bit error rate is too high, according to IEEE Std 802.3-2015. The threshold is BER = 10–4.

To aid in debug, GT near-end PMA loopback can be performed with the `gt_loopback_in` signal. This connects the TX SerDes to the RX SerDes, effectively bypassing potential signal integrity problems. In this way, the received data can be checked against the transmitted packets to verify that the logic is operating properly.

Related Information
Port Descriptions

Protocol Debug

To achieve error-free data transfers with the Ethernet IP core, the IEEE Std 802.3-2015 should be followed. Signal integrity should always be ensured before proceeding to the protocol debug.

Alignment Marker Spacing

According to IEEE Std 802.3-2015, the alignment marker spacing should be set to 16,383 for both the TX and RX. Check that both ends of the link are programmed to this value.

Diagnostic Signals

There are many error indicators available to check for protocol violations. Carefully read the description of each one to see if it is useful for a particular debugging problem.

The following is a suggested debug sequence.

1. Ensure that Word sync has been achieved.
2. Ensure that Lane sync has been achieved (this uses the lane marker alignment words which occur after every 16,383 words).
3. Verify that the bip8 indicators are clean.
4. Make sure there are no descrambler state errors.
5. Eliminate CRC32 errors, if any.
6. Make sure the AXI4-Stream protocol is being followed correctly.

7. Ensure that there are no overflow or underflow conditions when packets are sent.

Statistics Counters

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

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 of 40G/50G High Speed Ethernet 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.

Related Information

Board Testing of the 40G/50G High Speed Ethernet Using the AXI4-Lite Interface

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

To debug auto-negotiation and link training using the AXI4-Lite interface, see Board Testing of the 40G/50G High Speed Ethernet Using the AXI4-Lite Interface.

XILINX

Appendix C

Pause Processing Interface

The 40G/50G High Speed Ethernet Subsystem provides a comprehensive mechanism for pause packet termination and generation. The TX and RX have independent interfaces for processing pause information as described in this appendix.

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.

IMPORTANT! *Requesting both global and priority pause packets at the same time results inunpredictable 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 40G/50G High Speed Ethernet Subsystem automatically calculates and adds the FCS to the packet. For priority pause packets the 40G/50G High Speed 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 a 1 and keep it at 1 for the duration of the pause request (that is, if these inputs are set to 0, all pending pause packets are canceled). The 40G/50G High Speed Ethernet Subsystem transmits the pause packet immediately after the current packet in flight is completed.

To retransmit pause packets, the 40G/50G High Speed 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 `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.

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

To reduce the number of pause packets for priority mode operation, a timer is considered "timed out" if any of the other timers time out. Additionally, while waiting for the current packet in flight to be completed, any new timer that times out or any new requests from you 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 from you, both timers are forced to be timed out and a pause packet for both priorities is sent as soon as the current in-flight packet (if any) is transmitted.

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

RX Pause Termination

The 40G/50G High Speed 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.

   Note: Packets that pass step 1 are forwarded to you only if `ctl_rx_forward_control` is set to 1.

2. If step 1 passes, the packet is checked to determine if it is a global pause packet.
3. If step 2 fails, the packet is checked to determine if it is a priority pause packet.

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

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

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

assign etype_match_gcp = !ctl_rx_check_etype_gcp || (ETYPE ==
ctl_rx_etype_gcp);
assign opcode_match_gcp = !ctl_rx_check_opcode_gcp || ((OPCODE >=
ctl_rx_opcode_min_gcp) && (OPCODE <= ctl_rx_opcode_max_gcp));

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

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

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

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

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

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

assign control_packet = global_control_packet || priority_control_packet;
```

Where DA is the destination address, SA is the source address, OPCODE is the opcode, and ETYPE is the ethertype/length field that are 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 both buses, Bit [8] corresponds to global pause packets and bits [7:0] correspond to priority pause packets.

The following steps occur when a pause packet is received:

1. If the corresponding bit of `ctl_rx_pause_enable[8:0]` is 0, the quanta is ignored and the 40G/50G High Speed Ethernet Subsystem stays in step 1. Otherwise, the corresponding bit of the `stat_rx_pause_req[8:0]` bus is set to 1, and the received quanta is loaded into a timer.

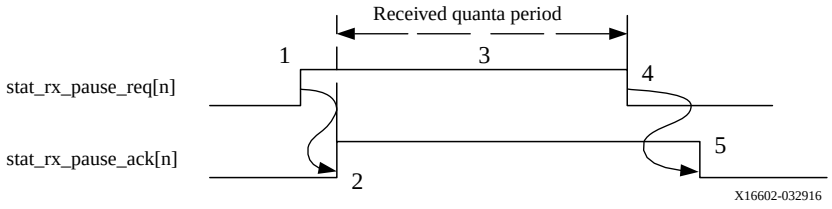
   Note: If one of the bits of `ctl_rx_pause_enable[8:0]` is set to 0 (that is, disabled) when the pause processing is in step 2 or later, the 40G/50G High Speed Ethernet Subsystem completes the steps as normal until it comes back to step 1.

2. If `ctl_rx_check_ack` input is 1, the 40G/50G High Speed Ethernet Subsystem 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 40G/50G High Speed Ethernet Subsystem starts counting down the timer.
4. When the timer times out, the 40G/50G High Speed Ethernet Subsystem 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 40G/50G High Speed Ethernet Subsystem deems the operation complete after 32 clock cycles.

The preceding steps are demonstrated in the following figure with each step shown on the waveform.

Figure 63: **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.

XILINX

Appendix D

Additional Resources and Legal Notices

Xilinx Resources

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

Documentation Navigator and Design Hubs

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

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

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

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

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

References

These documents provide supplemental material useful with this guide:

1. IEEE Standard for Ethernet (IEEE Std 802.3-2015)
2. 25G and 50G Ethernet Consortium Schedule 3 version 1.6 (August 18, 2015)(http://25gethernet.org/)
3. *UltraScale Architecture GTH Transceivers User Guide* (UG576)
4. *UltraScale Architecture GTY Transceivers User Guide* (UG578)
5. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
6. *Vivado Design Suite Tutorial: Designing IP Subsystems Using IP Integrator* (UG995)
7. *Vivado Design Suite User Guide: Designing with IP* (UG896)
8. *Vivado Design Suite User Guide: Getting Started* (UG910)
9. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
10. *ISE to Vivado Design Suite Migration Guide* (UG911)
11. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
12. 50G IEEE 802.3 Reed-Solomon Forward Error Correction LogiCORE IP Product Guide (PG234). (Registration required for access)
13. AMBA AXI4-Stream Protocol Specification V1.0 (Arm IHI 0051A)
14. IEEE Standard 1588-2008, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems" (standards.ieee.org/findstds/standard/1588-2008.html)

Revision History

The following table shows the revision history for this document.

Section	Revision Summary
06/03/2020 Version 3.1	
Board Testing of the 40G/50G High Speed Ethernet Using the AXI4-Lite Interface	Topic updated
LogiCORE Example Design Clocking and Resets	Diagrams updated
Customizing and Generating the Subsystem	Vivado graphics updated
Example Design Hierarchy (GT in Example Design)	Graphic updated

Section	Revision Summary
Debugging Auto-Negotiation and Link Training Debugging Auto-Negotiation and Link Training Using AXI4-Lite Interface	Debugging topics added
10/30/2019 Version 3.0	
Board Testing of the 40G/50G High Speed Ethernet Using the AXI4-Lite Interface	Added new section
05/22/2019 Version 2.5	
Entire document	Updated the GTM clocking and reset diagrams.
Chapter 4: Designing with the Subsystem	Added a new table
12/05/2018 Version 2.4	
Entire document	Added the GTM clocking and reset diagrams. RSFEC clause updated from 108 to 91.
04/04/2018 Version 2.3	
Latency	Updated tables Added a note for the `ctl_autoneg_bypass` signal and added description for `runtime_switchable` signal.
Chapter 5: Design Flow Steps	Updated figures Added a table note for the GT RefClk (In MHz) option Fixed typo: “udp” replaced with “upd”
AXI4-Lite Interface Implementation	Added .h Header File description
12/20/2017 Version 2.3	
Please Read: Important Legal Notices	Updated
Chapter 3: Product Specification	Updated 3.42.3 to 3.42.7 in the Description for `ctl_tx_test_pattern` Added a note about an invalid preamble in the tables Updated description for `stat_rx_bad_preamble`
Chapter 6: Example Design	Added a note about an invalid preamble in the tables Updated description for `send_continuous_pkts_*`
10/04/2017 Version 2.3	
Chapter 2: Overview	Updated tables
Chapter 3: Product Specification	Added RX and TX Latency values Updated figures Added clarification text to the first paragraph of the Back to Back Continuous Transfer section Added note about STAT_*_MSB/LSB registers to Statistics Counters section Added text below SWITCH_CORE_SPEED_REG: 018C table about Runtime Switch mode.
Egress	Added latency values
Chapter 5: Design Flow Steps	Updated screen displays Removed AN/LT Clock option from tables.
Changes from v2.2 to v2.3	Added new section
06/07/2017 Version 2.2	
Licensing and Ordering	Updated the Ordering information
Latency	Updated tables. Replaced 64 bytes with 16 bytes for several signals.

Section	Revision Summary
Chapter 4: Designing with the Subsystem	Updated screen displays
Changes from v2.1 to v2.2	Added new section
04/05/2017 Version 2.1	
Features	Added two new features
IP Facts	Updated Supported User Interfaces row with 128-bit straddle interfaces
Chapter 2: Overview	Updated
Standards	Updated
Chapter 3: Product Specification	Added new figures
Chapter 5: Design Flow Steps	Updated figures
TX Pause Generation	Added an important note
References	Updated
11/30/2016 Version 2.0	
Chapter 3: Product Specification	Changed “tx_reset and rx_reset” to “s_axi_aresetn” and changed “active-High” to “active-Low” in the second sentence of the Configuration Registers subsection of the AXI4-Lite Register Space
Chapter 5: Design Flow Steps	Updated the tables to add table notes
Chapter 6: Example Design	Added text about clearing status registers to the Status Registers section Added text about clearing statistics counters to the Statistics Counters section
10/05/2016 Version 2.0	
Entire document	Added `tick_reg_mode_sel` references
Chapter 4: Designing with the Subsystem	Updated figures
Chapter 5: Design Flow Steps	Updated figures
RS-FEC Support	Added new section
Example Design Hierarchy (GT in Example Design)	Added new section
Latency	Added new section
Run Time Switchable	Added new section
Chapter 3: Product Specification	Added new Register information
06/08/2016 Version 1.1	
Chapter 3: Product Specification	Updated figures
Chapter 4: Designing with the Subsystem	Updated figures
Chapter 5: Design Flow Steps	Updated figures
Appendix A: Upgrading	Added ports and modified port names.
Entire document	Changed “HSEC” to “40G/50G High Speed Ethernet Subsystem” Changed 50GMII to XL/50GMII throughout
TX Debug (Buffer Errors)	Updated description
04/06/2016 Version 1.0	
Initial release	N/A

Please Read: Important Legal Notices

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any 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.

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