25G Ethernet IP

IP Version: v2.3.0

User Guide

FPGA-IPUG-02249-1.5

December 2025

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults, and all associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. LATTICE PRODUCTS AND SERVICES ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS, OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING ANY APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, BUYER MUST TAKE PRUDENT STEPS TO PROTECT AGAINST PRODUCT AND SERVICE FAILURES, INCLUDING PROVIDING APPROPRIATE REDUNDANCIES, FAIL-SAFE FEATURES, AND/OR SHUT-DOWN MECHANISMS. LATTICE EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Inclusive Language

This document was created consistent with Lattice Semiconductor’s inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice’s inclusive language FAQ 6878 for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

Contents

Figures

Tables

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviations	Definition
25GMII	25-Gigabit Media Independent Interface
AXI4-Lite	Advanced eXtensible Interface 4 Lite
AXI4-Stream	Advanced eXtensible Interface 4 Stream
CRC	Cyclic Redundancy Check
DA	Destination Address
DIC	Deficit Idle Count
EBR	Embedded Block RAM
FCS	Frame Check Sequence
FIFO	First-In First-Out
GPLL	Generic Phase-Locked Loop
IFG	Inter-Frame Gap
IPG	Inter-Packet Gap
LUT	Look-Up Table
L/T	Length/Type
MAC	Media Access Controller
MII	Media Independent Interface
MTU	Maximum Transmission Unit
PCS	Physical Coding Sublayer
PHY	Physical Layer Device
PMA	Physical Medium Attachment
SA	Source Address
SFD	Start Frame Delimiter
SOF	Start of Frame

1. Introduction

1.1. Overview of the IP

The Lattice Semiconductor 25G Ethernet (GbE) IP core supports the ability to transmit and receive data between a host processor and an Ethernet network. The 25GbE IP core consists of the 25-Gigabit Media Independent Interface (25GMII), which connects media access controllers (MACs) and Physical Layer devices (PHYs).

The main function of the 25GbE MAC is to ensure that the media access rules specified in the 802.3 IEEE standards are met while transmitting a frame of data over an Ethernet. On the receiver side, the Ethernet MAC extracts the different components of a frame and transfers them to higher applications through an AXI4-Stream interface. The PHY implements the physical coding sublayer (PCS) and physical medium attachment (PMA) functionality based on the IEEE 802.3 25GBASE-R specification.

The following figure shows an example of a 25GBASE-R application. The 25GbE MAC is connected to the 25GbE PHY within the 25G Ethernet IP core through the 25GMII. The clock source to the Ethernet MAC is from xg_tx_gclk_o. For example, the clock output from the Ethernet PHY.

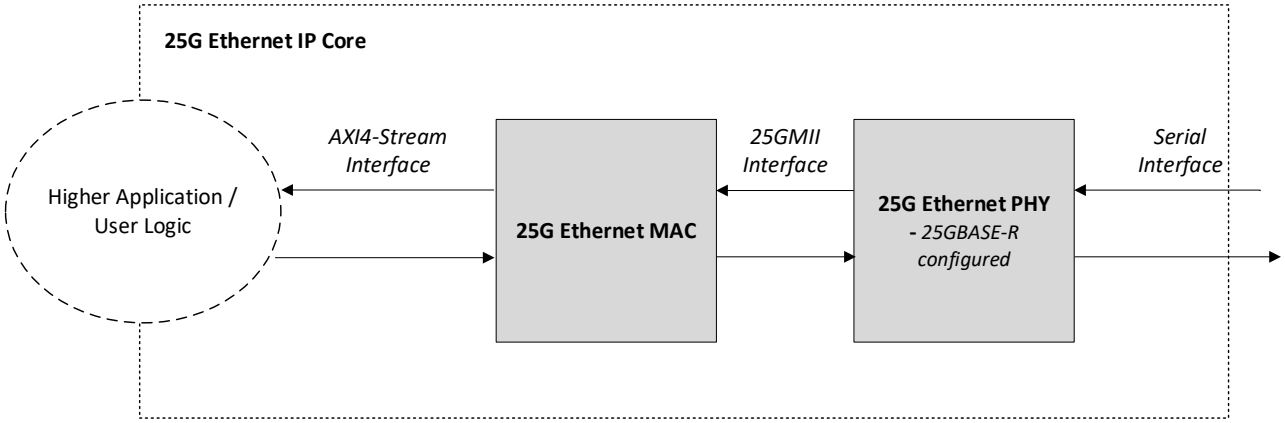

Figure 1.1. 25GBASE-R Application

1.2. Quick Facts

The following table provides quick facts about the 25G Ethernet IP core.

Table 1.1. Summary of the 25G Ethernet IP

IP Requirements	Supported Devices	Avant™-AT-X (Speed grade 3 only).
	IP Changes[1]	For a list of changes to the IP, refer to the 25G Ethernet IP Release Notes (FPGA-RN-02034).
Resource Utilization	Supported User Interface	AXI4-Stream or AXI4-Lite.
	Resources	See Table A.1.
Design Tool Support	Lattice Implementation	IP core v2.3.0—Lattice Radiant software 2025.2 or later.
	Synthesis	Synopsys® Synplify Pro® software, O-2018.09LR-SP1.
	Simulation	QuestaSim™ Lattice FPGA Edition software.
Driver Support	API Reference	Refer to the 2.5G, 10G, and 25G Ethernet Driver API Reference (FPGA-TN-02375).

Note:

1. In some instances, the IP may be updated without changes to the user guide. This user guide may reflect an earlier IP version but remains fully compatible with the later IP version. Refer to the IP Release Notes for the latest updates.

1.3. IP Support Summary

The following table provides IP support information on the 25G Ethernet IP core.

Table 1.2. 25G Ethernet IP Core Support Readiness

Device Family	Mode	Radiant Timing Model	Hardware Validated
Avant-AT-X	PHY only, MAC only, MAC + PHY	Preliminary	Yes

1.4. Features

The 25G Ethernet IP core offers the following key features:

MAC

- Compliant to the IEEE 802.3-2018 standard
- Supports standard 25 Gbps Ethernet link layer data rate
- 128-bit wide internal datapath operating at 195.3125 MHz
- AXI4-Stream interface on the client's transmit and receive interfaces
- 25GMII interface to physical (PHY) layer
- Supports deficit idle count (DIC)
- Supports VLAN, SVLAN, and up to 9,600 bytes of Jumbo frames
- Custom preamble mode
- Independent TX and RX Maximum Transmission Unit (MTU) frame length
- Comprehensive statistics support
- Optional frame check sequence (FCS) generation on transmission
- Optional FCS stripping during reception
- Optional multicast address filtering
- Programmable Inter-Frame Gap (IFG)
- Supports flow control using pause frames
- Supports Priority Flow Control (PFC)
- Automatic padding of short frames
- Inter-Frame Stretch Mode during transmission
- Supports full-duplex operation
- Programmable promiscuous (transparent) mode

PHY

- Designed to the IEEE 802.3-2018 25GBASE-R specification
- 64b/66b encoding and decoding
- 25GMII interface: 128-bit, 195.3125 MHz
- Supports AXI4-Lite interface for PHY and MAC register access

1.5. Licensing and Ordering Information

An IP core-specific license string is required to enable full use of the 25G Ethernet IP core in a complete, top-level design. The IP core can be fully evaluated through functional simulation and implementation (synthesis, map, place and route) without an IP license string. This IP core supports Lattice's IP hardware evaluation capability, which enables you to create versions of the IP core that operate in hardware for a limited time (approximately four hours) without requiring an IP license string. For details, refer to the Hardware Evaluation section. However, a license string is required to enable timing simulation and to generate a bitstream file that does not include the hardware evaluation timeout limitation.

For more information about pricing and availability of the 25G Ethernet IP core, contact your local Lattice Sales Office.

1.5.1. Hardware Evaluation

The 25G Ethernet IP core supports Lattice's IP hardware evaluation capability when used with Avant-AT-X devices. The hardware evaluation capability enables you to create versions of the IP core that operate in hardware for a limited period (approximately four hours) without requiring the purchase of an IP license. It may also be used to evaluate the core in hardware in user-defined designs. The hardware evaluation capability may be enabled or disabled in the **Strategy** dialog box. It is enabled by default. To change this setting, go to **Project > Active Strategy > Synplify Pro Settings**.

1.5.2. Ordering Part Number

Table 1.3. Ordering Part Number

Device Family	Part Number	
	Single Seat Annual	Single Seat Perpetual
Avant-AT-X	ETHER-25G-AVX-US	ETHER-25G-AVX-UT

1.6. Hardware Support

For Avant device support, refer to the Example Design section for more information on the boards used.

1.7. Minimum Device Requirements

The minimum device requirements for the 25G Ethernet IP core are as follows:

Table 1.4. Minimum Device Requirements for the 25G Ethernet IP Core

Devices	Speed Grades
LAV-AT-X70	Speed grade 3 only
LAV-AT-X50	Speed grade 3 only
LAV-AT-X30	Speed grade 3 only

For more information, refer to the Appendix A. Resource Utilization section.

1.8. Naming Conventions

1.8.1. Nomenclature

The nomenclature used in this document is based on Verilog HDL.

1.8.2. Signal Names

Signal names that end with:

- *_n* are active low signals (asserted when value is logic 0)
- *_i* are input signals
- *_o* are output signals
- *_io* are bi-directional input/output signals

2. Functional Description

2.1. MAC + PHY

2.1.1. IP Architecture Overview

The 25G Ethernet IP core transmits and receives data between a host processor and an Ethernet network. With the *MAC + PHY* option selected, the IP consists of a MAC and a PHY that is connected through 25GMII internally.

2.1.2. Block Diagram

The following figure shows the 25G Ethernet IP core block diagram.

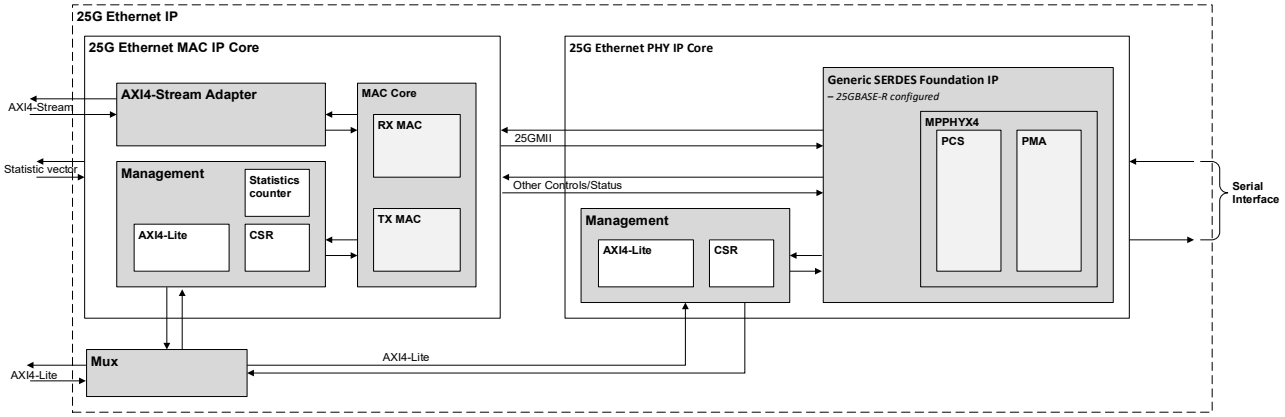

Figure 2.1. 25GbE IP Core Block Diagram

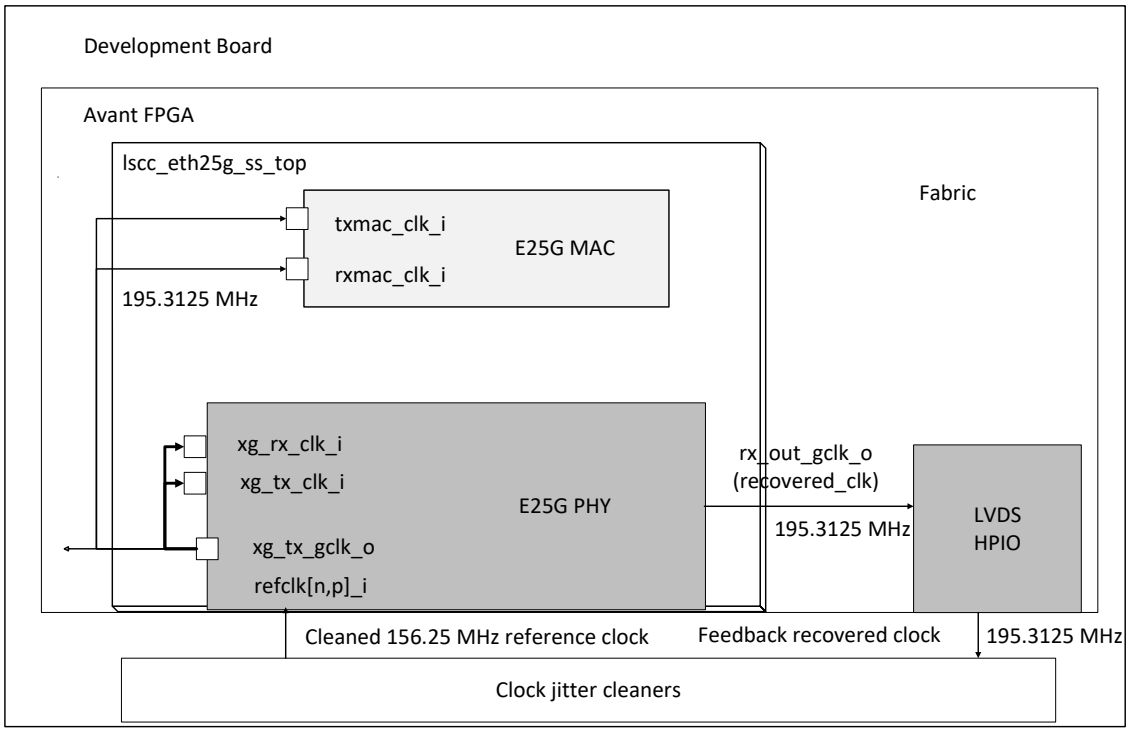

Figure 2.2. 25GbE IP Core SyncE Block Diagram

2.1.3. Management Block

The Management block can be accessed through the AXI4-Lite interface when the *MAC + PHY* option is selected.

For more details about the management blocks in MAC and PHY respectively, refer to the MAC Management Block and PHY Management Block sections.

For AXI4-Lite protocol timing details, refer to AMBA 4 AXI and ACE Protocol Specification issue H.

2.2. MAC

2.2.1. IP Architecture Overview

The main function of the 25G Ethernet MAC IP is to ensure that the media access rules specified in the IEEE 802.3 standards are met while transmitting a frame of data over an Ethernet. On the receiver side, the Ethernet MAC extracts the different components of a frame and transfers them to higher applications through an AXI4-Stream interface.

2.2.2. Block Diagram

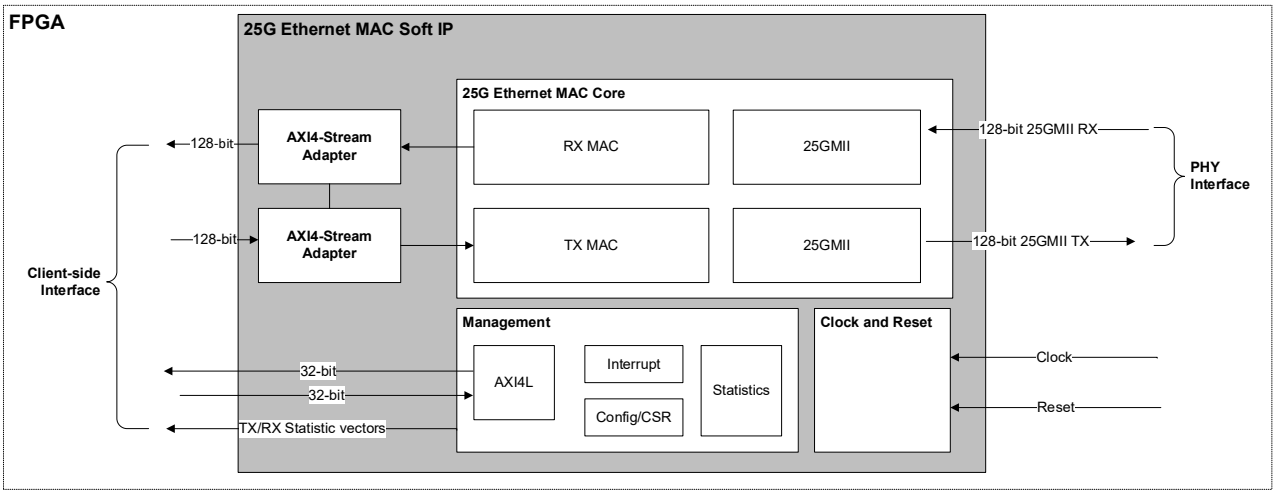

Figure 2.3. 25G Ethernet MAC IP Core Block Diagram

2.2.3. Ethernet Data Format

Figure 2.4 shows an untagged Ethernet frame format. Figure 2.5 and Figure 2.6 show the single-tagged and stacked Ethernet frame formats, respectively.

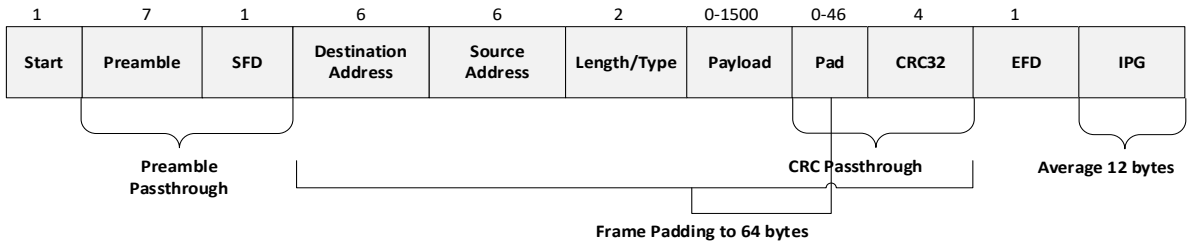

Figure 2.4. Untagged Ethernet Frame Format

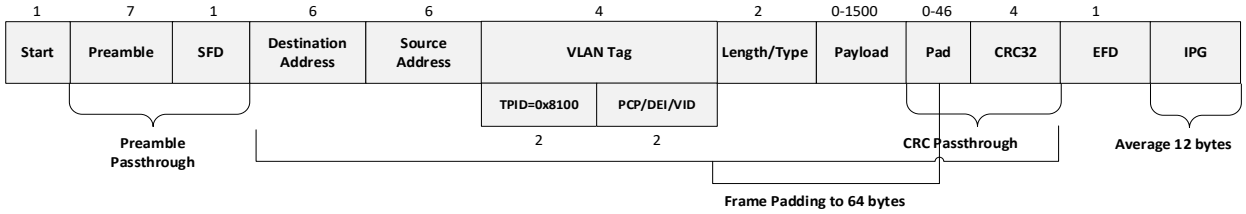

Figure 2.5. VLAN Tagged Ethernet Frame Format

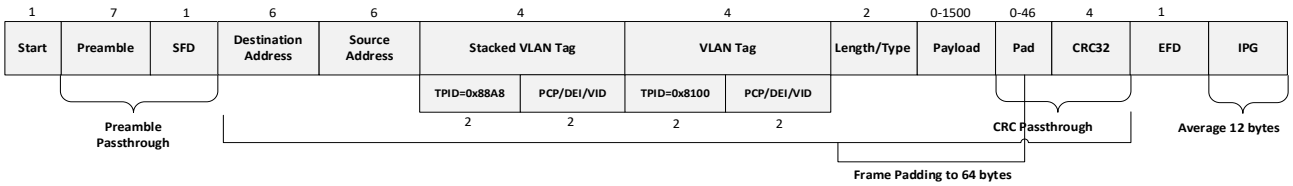

Figure 2.6. Stacked VLAN Tagged Ethernet Frame Format

The MAC is responsible for constructing a valid frame from the data received from the client before transmitting it.

The TX MAC module receives the client payload data with the destination and source addresses. It then adds, appends, or updates various header fields in accordance with the configuration specified. The MAC does not modify the destination address, the source address, or the payload received from the client. However, the TX MAC module adds a preamble, if the IP core is not configured to receive the preamble from user logic. It pads the payload of frames greater than eight bytes to satisfy the minimum Ethernet frame payload of 46 bytes. By default, the MAC inserts the CRC bytes. The TX MAC module inserts IDLE bytes to maintain an average IPG of 12.

The fields that are expected from the client-side interface can be one of the following:

- The frame contains the FCS along with the destination address (DA), source address (SA), length/type (L/T), and data. The TX MAC adds the preamble and start frame delimiter (SFD) before transmitting the frame. This mode can be set by enabling tx_pass_fcs bit in the TX_CTL register. For timing details, see the In-Band FCS Passing section.
- A 4-byte VLAN tag is inserted between the source address and the length/type field of the original untagged frame when a frame enters the VLAN-aware portion of the network. The Tag Protocol Identified (TPID) being a 16-bit wide field in a VLAN tag, occupies the same position as the length/type field of the untagged frame. For a single-tagged VLAN (IEEE 802.1Q) Ethernet frame, its TPID is assigned a value of 0x8100. As for a stacked VLAN (IEEE 802.1ad) Ethernet frame, the TPID of its outer tag (next to SA) is assigned a value of 0x88A8.
- The TX MAC calculates the number of bytes to be padded as well (if required) in addition to the FCS for the entire frame and adds the preamble and SFD before transmitting the frame. For timing details, see the Default Normal Frame section.
- When tx_pass_pream bit is set in the TX_CTL register, the client supplies a custom data in the preamble and SFD field. The TX MAC preserves the preamble field and passed it directly to physical layer. This applies to both cases mentioned above. For timing details, see the Custom Preamble Passing section.

On the receiver path, it receives frames from the network through 25GMII interface and passes the parameters of the frame to the client via the AXI4-Stream interface. The RX MAC receives Ethernet frames and forwards the payload with relevant header bytes to the client after performing some MAC functions on header bytes. The RX MAC processes all incoming valid frames.

On the receiver path, the MAC can be programmed to transfer frame in one of the following cases:

- Transfer the frame after stripping off the FCS and any pad fields. This is the default settings of the RX MAC.
- Transfer the frame with the FCS field and any pad fields by setting the rx_pass_fc bit of RX_CTL register.

- Transfer the frame in promiscuous mode by setting the prms bit of RX_CTL register. This mode transfers all frames it receives to the client rather than passing only the frames that is specifically programmed to receive.

If preamble passthrough is enabled, the RX MAC will forward the preamble bytes over the AXI4-Stream interface. If preamble passthrough is disabled, the start of packet is aligned to the MSB.

2.2.4. Receive MAC

The Receive MAC receives the incoming frames and transfers them to the client via the AXI4-Stream interface. In the process, it performs the following operations:

- Checks the frame for a valid start of frame (SOF) and SFD symbol. This checking is disabled when RX is configured to custom preamble mode.
- Determines whether the frame should be received by analyzing the DA.
- Determines the type of frame by analyzing the length/type field.
- Checks for any errors in the frame by recalculating the CRC and comparing it with the expected value.

The RX MAC operation is determined by programming the MODE and RX_CTL registers.

You can specify whether the FCS field should be transferred on to the AXI4-Stream interface by programming the rx_pass_fcs bit of RX_CTL register. If the FCS field is to be stripped off the frame, any padding bytes within the frame are stripped off as well.

When a valid SOF is detected, the DA field of the incoming frame is analyzed as follows:

- If the DA field is a unicast address, it is compared with the programmed MAC address. Unless the prms bit of RX_CTL register is set, the incoming frame is discarded if the DA field and the programmed MAC address (MAC_ADDR_{0,1} registers) do not match.
- If the frame had a multicast address and if receive_all_mc signal is not asserted, all such frames are dropped (except PAUSE frames).
- If the frame had a multicast address and if receive_all_mc signal is asserted, the multicast frames are subject to the following address filtering rules:
  - For all frames with multicast address, the CRC of the destination address is computed and the mid-six bits of the least significant byte of the CRC is chosen as the address to a hash table.
  - The 64-bit hash table is programmed in the MC_TABLE_{0,1} registers.
  - The MAC implements an eight-row table with eight bits in each row.
  - The lower three bits of the selected CRC are used to select one of these eight rows and the next three bits are used to select one of the bits in the selected table.
  - The incoming multicast frame is accepted if the bit selected from the hash table is set to one. It is discarded if the bit selected is zero.
- If the incoming frame has a broadcast address, it is accepted if either the prms or the receive_bc bit of RX_CTL register is set. A broadcast frame is discarded if none of these bits are set.

2.2.4.1. RX Filtering

The IP core can operate in cut-through mode or in store-and-forward mode. In cut-through mode, the IP core does not buffer incoming Ethernet packets for filtering. It can filter out incoming runt packets but cannot filter on any other criteria. The value in bit 0 of the RX_FILTER_CTRL register at offset 0x103 determines the mode, and the value in bit 3 determines whether the IP core filters runt packets.

When the IP core is in cut-through mode, it does not filter incoming Ethernet packets based on destination address. Therefore, when in cut-through mode, the IP core is in promiscuous receive mode. The Ethernet standard definition of promiscuous receive mode requires that the IP core accept all valid frames, regardless of destination address. The MAC core accepts or rejects invalid frames based on the filtering criteria that are turned on.

In store-and-forward mode, you can enable oversized-frame handling. When the maximum frame size is set to 9,600 bytes, the IP core passes some of the frames between 9,601-9,644 bytes in size and drops frames of 9,645 bytes or more.

The IP core supports the following filtering options:

- Destination address mismatch—refer to the descriptions on the RX_FILTER_CTRL register and the MADDR_CTRL register as well as the Address Checking section.
- Runt frame—refer to the description on the dout_runt_last_data signal.
- Oversized frame—refer to the description on Jumbo frames.
- Pause frame
- Control frame
- FCS error

2.2.4.2. Preamble Processing

The preamble sequence is Start, six preamble bytes, and SFD. If this sequence is incorrect the frame is ignored. The Start byte must be on receive lane 0 (most significant byte). The IP core uses the SFD byte (0xD5) to identify the last byte of the preamble. The MAC RX looks for the Start, six preamble bytes, and SFD.

By default, the MAC RX removes all Start, SFD, preamble, and IPG bytes from accepted frames. However, if you turn on the RX preamble pass-through feature, by setting bit 0 of the Preamble Pass-Through Configuration register at offset 0x125, the MAC RX does not remove the eight-byte preamble sequence.

2.2.4.3. CRC Checking and FCS (CRC32) Forwarding

The RX MAC checks the incoming CRC32 for errors. It asserts rx_error in the same cycle as l1_rx_endofpacket when it detects an error. CRC takes several cycles. The packet frame is delayed to align the CRC output with the end of the frame. By default, the RX MAC strips off the CRC bytes before forwarding the packet to the MAC client. You can configure the core to retain the RX CRC and forward it to the client by updating the CRC_CONFIG register. In the user interface, the TLAST indicates the end of CRC bytes if CRC is not removed. When CRC is removed, the TLAST signal indicates the final byte of payload.

By default, the IP core asserts the FCS error signal (l<n>_rx_fcs_error or dout_fcs_error) and the EOP signal on the same clock cycle if the current frame has an FCS error. However, if the IP core is in RX automatic pad removal mode, the signals might not be asserted in the same clock cycle.

2.2.4.4. Automatic PAD Removal Control

In the MAC configurations, you can enable and disable RX automatic pad removal with a configuration register bit in run-time. In these figures, normal packet data, highlighted in gray, lasts until the end of the payload section. However, the IP core might pass along additional padding, marked with PAD, to ensure that the frame length is at least 64 bytes. The Ethernet standard requires padding insertion when the payload length is less than 46 bytes. EOP at the RX interface is normally marked after padding, but you can disable CRC removal to place the EOP at the end of the CRC block.

When enabled, RX automatic pad removal moves the EOP marker to the end of the payload as indicated in the length field, whether padding has been inserted or not. If the length is greater than 46 bytes, no padding has been inserted and this feature has no effect. When you enable RX automatic pad removal, the CRC is excluded from the EOP marker on packets that have stripped padding, and enabling CRC retention has no effect on padded packets.

Note: Signals ending in *_fcs_valid and *_fcs_error are not shifted along with the new EOP marker. Instead, they function as if pad removal were disabled. Do not rely on these signals when operating in RX automatic pad removal mode.

The PAD_CONFIG register controls RX automatic pad removal. By default, pad removal is disabled. Statistics counting is not affected by RX automatic pad removal; data reports as default, as if the padding were not removed.

2.2.4.5. Address Checking

The RX MAC supports all three types of addresses:

- Unicast—Specifies a destination address as a unicast (individual) address. Bit 0 is 0.
- Multicast—Specifies a destination address as a multicast or group address. Bit 0 is 1.
- Broadcast—Specifies a broadcast address when all 48 bits in the destination address are all 1s, 48'hFFFF_FFFF_FFFF.

If destination address matching is enabled, IP core address checking compares the address to the address programmed in the destination address register and accepts only the frames with a matching address. You must enable filtering to discard mismatched destination addresses.

To enable address checking, you must ensure the IP core has the following values in the specified register fields:

- Bit 0 of the RX_FILTER_CTRL register at offset 0x103 has the value of 0.
- Bit 0 of the MADDR_CTRL register at offset 0x140 has the value of 1.
- Bit 30 of the MADDR_CTRL register has the value of 1.

The MADDR_CTRL fields allow you to turn off destination address checking but still enable the IP core to filter RX traffic based on other criteria.

If bit 0 of the RX_FILTER_CTRL register has the value of 1, the IP core is in promiscuous receive mode. In this mode, the IP core omits address checking and accepts all the Ethernet frames it receives, except possibly runt frames.

2.2.4.6. Inter-Packet Gap

The MAC RX removes all IPG octets received and does not forward them to the client interface.

2.2.4.7. Pause and PFC Ignore

When the pause frame receive enable bits are not set, the IP core does not process incoming pause frames. In this case, the MAC TX traffic is not affected by the valid pause frames.

Similarly, when the PFC frame receive enable bits are not set, the IP core does not process incoming PFC frames. In this case, the MAC RX will not assert rx_pfc_tuser_o[7:0].

When drop_mac_ctrl bit of RX_CTL is set, the RX MAC will not pass the pause frame or PFC frame received to the client interface. The signal axis_rx_tvalid_o will be '0' to indicate that any values at axis_rx_tdata_o are invalid, hence representing the pause frame or PFC frame is dropped.

You can enable unicast or multicast pause receive by setting the appropriate bits of the pause registers.

2.2.5. Flow Control

Flow control reduces congestion at the local or remote link partner. When either link partner experiences congestion, the respective transmit control sends pause frames. XOFF pause frames stop the remote transmitter. XON pause frames let the remote transmitter resume data transmission. The IP core supports both standard and Priority-based Flow Control (PFC) control frames.

Standard Flow Control (Pause Frame Flow Control):

- Inhibits the next client frame transmission on the reception of a valid pause frame.

Priority-based Flow Control (PFC):

- PFC frame transmission follows a priority-based arbitration scheme, where the Frame Type indication is provided for the usage of external downstream logic.
- Inhibits the per-queue client frame transmission on the reception of a valid PFC frame from the client. Includes per-queue PFC pause quanta duration indicator.

Standard Flow Control and PFC must strictly not be enabled at the same time.

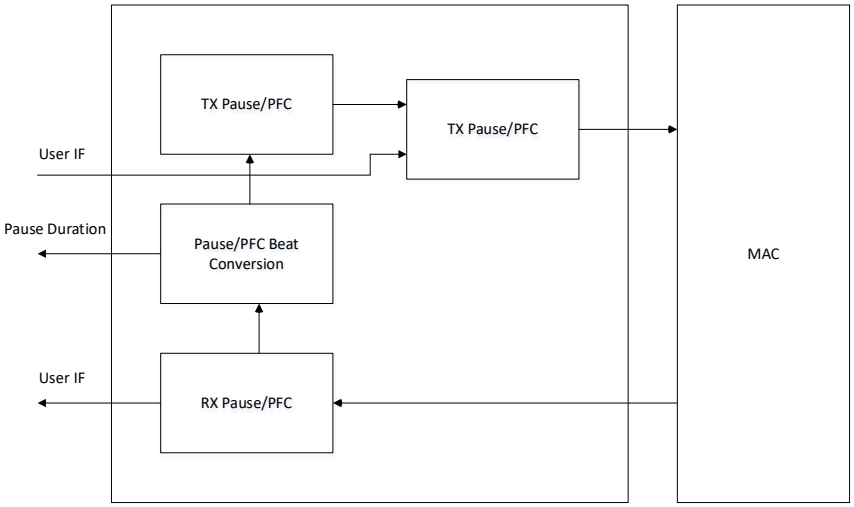

Figure 2.7. Priority-based Flow Control

2.2.5.1. Congestion drop_mac_ctrl and Flow Control Using Pause Frames

The flow control block is designed to clause 31 of the IEEE 802.3-2012 standard. The core can be configured to transmit pause requests and to act on their reception; these modes of operation can be independently enabled or disabled. The IP core provides flow control to reduce congestion at the local or remote link partner. When either link partner experiences congestion, the respective transmit control sends pause frames. The pause frame instructs the remote transmitter to stop sending data for the duration that the congested receiver specified in an incoming XOFF frame.

When the IP core receives the XOFF pause control frame with the following conditions, the IP core stops transmitting frames for a period equal to the pause quanta of the incoming pause frame:

- The appropriate bit of the RECEIVE_PAUSE_CONTROL register has the value of 1
- Address matching is positive

While paused, the IP core does not transmit data but can transmit pause frames.

The pause quanta can be configured in the pause quanta register of the device sending XOFF frames. If the pause frame is received in the middle of a frame transmission, the transmitter finishes sending the current frame and then suspends transmission for a period specified by the pause quanta. Data transmission resumes when a pause frame with quanta of zero is received or when the timer has expired. The pause quanta received overrides any counter currently stored. When more than one pause quanta is sent, the value of the pause is set to the last quanta received.

XOFF pause frames stop the remote transmitter. XON pause frames let the remote transmitter resume data transmission.

One pause quanta fraction is equivalent to 512 bit times, which equates to 512/128 (the width of the MAC data bus), or four system clock cycles.

XOFF Frame
START[7:0]
PREAMBLE[47:0]
SFD[7:0]
DESTINATION ADDRESS[47:0] = 0x010000C28001[10]
SOURCE ADDRESS[47:0]
TYPE[15:0] = 0x8808
OPCODE[15:0] = 0x001
PAUSE QUANTA[15:0] = 0xP1, 0xP2[11]
PAD[335:0]
CRC[31:0]

XON Frame
START[7:0]
PREAMBLE[47:0]
SFD[7:0]
DESTINATION ADDRESS[47:0] = 0x010000C28001
SOURCE ADDRESS[47:0]
TYPE[15:0] = 0x8808
OPCODE[15:0] = 0x001
PAUSE QUANTA[15:0] = 0x0000
PAD[335:0]
CRC[31:0]

Figure 2.8. XOFF and XON Pause Frames

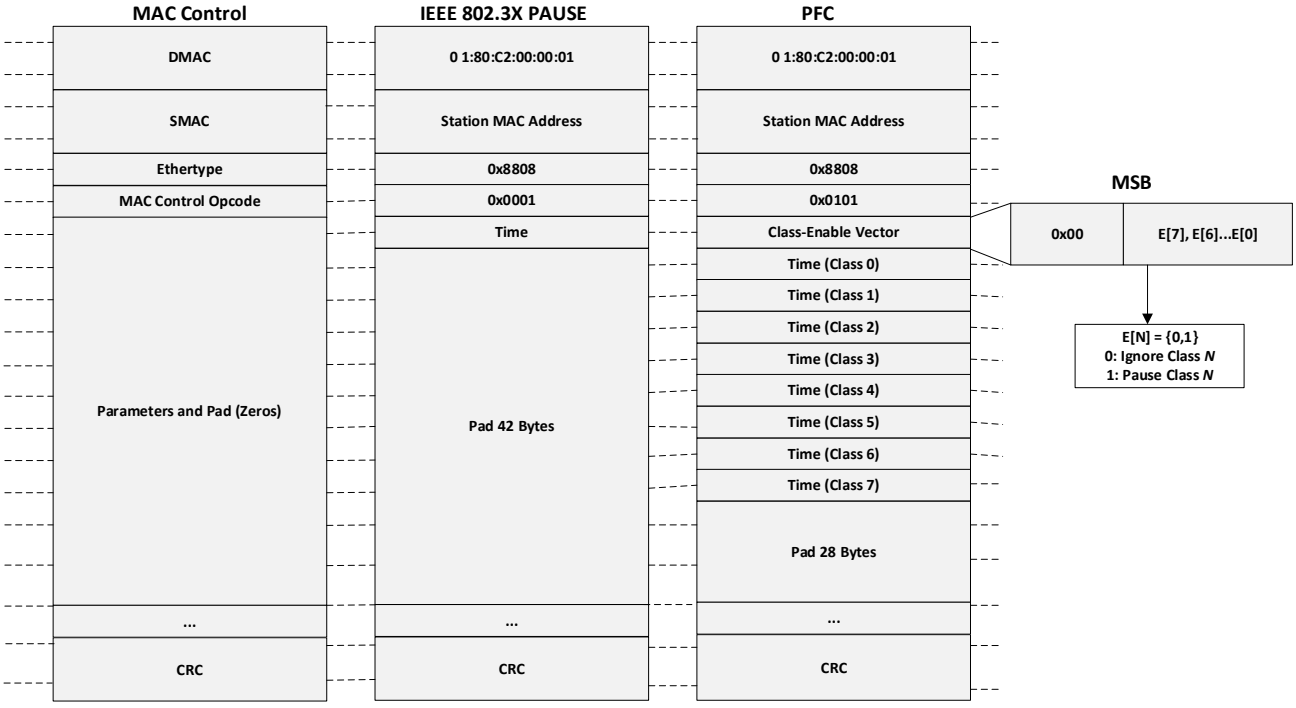

Figure 2.9. Ethernet MAC Control Frame Format

2.2.5.2. Conditions Triggering XOFF Frame Transmission

The TX MAC transmits XOFF frames when one of the following conditions occurs:

- Client requests XOFF transmission—A client can explicitly request that XOFF frames be sent using the pause control interface signals. When pause_insert_tx is asserted and pause_insert_time is not zero, an XOFF frame is sent to the Ethernet network when the current frame transmission completes.
- Host (software) requests XOFF transmission—Setting the pause control register triggers a request that an XOFF frame be sent.

2.2.5.3. Conditions Triggering XON Frame Transmission

The TX MAC transmits XON frames when one of the following conditions occurs:

- Client requests XON transmission—A client can explicitly request that XON frames be sent using the pause control interface signals. If pause_insert_tx is asserted and pause_insert_time is zero, an XON frame is sent to the Ethernet network when the current frame transmission completes.
- Host (software) requests XON transmission—Setting the pause control register triggers a request that an XON frame be sent.

2.2.5.4. Pause Transmission Logic

An XON/XOFF request triggers the IP core to transmit a pause or PFC flow control frame on the Ethernet link. You can control XON/XOFF requests using the TX flow control registers or the pause_insert_tx0 and pause_insert_tx1 input signals.

You can specify whether the IP core accepts XON/XOFF requests in 1-bit or 2-bit format by updating the TX Flow Control Request Mode register field. By default, the IP core assumes 1-bit requests.

2.2.5.5. Pause Control and Generation Interface

The pause control interface implements flow control as specified by the *IEEE 802.3ba 2010 100G Ethernet Standard*. Upon receiving a pause packet, the pause logic temporarily stops packet transmission, and can pass the pause packets through as normal traffic or drop the pause control frames in the RX direction.

2.2.5.6. Pause Control Frame and Non-Pause Control Frame Filtering and Forwarding

The MAC IP cores can pass the pause packets through as normal traffic or drop the pause control frames in the RX direction. By default, pass-through is disabled.

The following rules define pause control frames filtering control:

- The RX_FILTER_CTRL register contains options to filter different packet types, such as runt packets, FCS error packets, address mismatch packets, and so on, from the RX MAC. The RX_FILTER_CTRL register contains one bit to enable pause packet filtering and one bit to enable non-pause control packet filtering. The reset state for both bits is 1, where filtering is enabled. The bits are gated by RX_FILTER_CTRL bit [0], which enables and disables all filtering.
- If you have enabled pause packet filtering, the IP core drops packets that enter the RX MAC and match the length and type of 0x8808 with an opcode of 0x1 (pause packets) and does not process them or forward them to the client interface.
- If you have enabled non-pause control packet filtering, the IP core drops packets that enter the RX MAC and match the length and type of 0x8808 with an opcode other than 0x1 (pause packets) and does not forward them to the client interface.
- If you have disabled pause packet filtering, the RX MAC forwards pause packets to the client interface depending on their destination address. If destination address filtering is not enabled, you are forwarded all pause packets. If destination address filtering is enabled, you are only forwarded pause packets with a valid packet multicast address or a destination address matching the 40-100GbE IP core address.

Pause and control packet pass-through do not affect the pause functionality in the TX or RX MAC.

2.2.6. Statistics Counter Interface

The statistics counters module is a synthesis option that you select in the parameter editor. However, the statistics status bit output vectors are provided whether you select the statistics counters module option or not.

The increment vectors are brought to the top level as output ports. If you configure the statistics counters module, the increment vectors also function internally as input ports to the control and status registers (CSR).

2.2.7. Transmit MAC

The Transmit MAC controls access to the physical medium. Its main functions are as follows:

- Data padding for short frames when FCS generation is enabled.
- Adds preamble.
- Generation of a pause frame when the tx_pausreq bit of MAC_CTL register is asserted. The bit tx_fc_en of TX_CTL register should be set to 1 to enable this feature.
- To stop frame transmission when a pause frame is received by the Receive MAC.
- Generation of an XOFF frame when tx_pfc_tuser_i[7:0] signals or tx_p[0-7]_pfcreq bits of TX_PFC_CTL register are asserted. Before this, the tx_pfc_p[7:0]_en and tx_pfc_en bits of TX_CTL register must be set to 1 to enable this feature.
- Implement link fault signaling logic and transmit appropriate sequences based on the remote link status of the TX_RX_STS register.

The TX MAC operation is determined by programming the MODE and TX_CTL registers.

By default, the Transmit MAC is configured to generate the FCS pattern for the frame to be transmitted. However, this can be prevented by setting tx_pass_fcs bit of the TX_CTL register. This feature is useful if the frames being presented for transmission already contain the FCS field. When FCS field generation by the MAC is disabled, ensure that short frames are properly padded before the FCS is generated. If the MAC receives a frame to transmit that is shorter than 64 bytes when FCS generation is disabled, the frame is sent as is and a statistic vector for the condition is generated.

The DA, SA, L/T, and data fields are derived from higher applications through the AXI4-Stream interface and then encapsulated into an un-tagged Ethernet frame. The frame encapsulation consists of adding the preamble bits, the SFD bits, and the CRC check sum to the end of the frame. If transmit_short bit of TX_CTL register is not set, all short frames are padded. However, there's a limitation that you must not transmit a 16-byte or shorter frame in application due to 128-bit data width design requirement.

The frame is not sent over the network until the network has been idle for a minimum Inter-Packet Gap (IPG) of 12 bytes. The IPG is adjustable through the tx_ipg field of the IPG_VAL register. The Transmit MAC uses DIC to reach an average IPG of 8 bytes + (tx_ipg x 4 bytes). With the default tx_ipg value of 1, the Transmit MAC aims for an average IPG of 12 bytes. For more information on DIC, refer to the IEEE 802.3-2012 Section 46.

The TX MAC requires a continuous stream of data for the entire frame. There cannot be any bubbles of no data transfer within a frame. After the TX MAC is done transmitting a frame, it waits for more frames from the AXI4-Stream interface. During this time, it goes to an idle state that can be detected by reading the TX_RX_STS register. Because the MODE register can be written at any time, the TX MAC can be disabled while it is actively transmitting a frame. In such cases, the MAC completely transmits the current frame and then returns to the idle state. The control registers should be programmed only after the MAC has returned to the IDLE state.

There are two different methods for transmitting a pause frame. In the first method, the application layer forms a pause frame and submits it for transmission via the AXI4-Stream interface. In the other method, the application layer signals the TX MAC directly to transmit a pause frame. This is accomplished by asserting tx_pausreq bit of MAC_CTL register. In this case, the TX MAC completes the transmission of the current packet and then transmits a pause frame with the pause time value supplied through the tx_paustim bits of the PAUSE_TMR register.

As for the transmission of a Priority Flow Control XOFF frame, it can be initiated through two different methods - the assertion of tx_pfc_tuser_i[7:0] signals or tx_p[0-7]_pfcreq bits of TX_PFC_CTL register. This must only be done at least clock cycle after both tx_pfc_en and tx_pfc_p[0-7]_en bits of TX_CTL register have been set. If the second method is used, for example through the assertion of TX_PFC_CTL tx_p[0-7]_pfcreq bits, tx_pfc_tuser_i signals should be tied to '0'. Similar to a PAUSE frame, a PFC frame with the per priority pause quanta duration supplied through bit [15:0] of PFC_P[0-7]_TM registers will only be sent out after the TX MAC has completed the transmission of the current packet.

In case there are multiple assertions of tx_pfc_tuser_i[7:0] signals or tx_p[0-7]_pfcreq bits during an existing transmission, these triggers will be accumulated - only a single XOFF frame will be sent out after the current transmission is complete. The internal quanta counter will start counting after an XOFF frame has been sent out. If tx_pfc_tuser_i[7:0] signals or tx_p[0-7]_pfcreq bits remain asserted when the quanta counter has reached the respective refresh value of enabled priorities (supplied through bit [31:16] of PFC_P[0-7]_TM registers), a new XOFF frame will be sent out. No XON frame will be sent out explicitly when the internal quanta counter has expired.

Note: The pause frame feature (IEEE 802.3) and Priority Flow Control (IEEE 802.1Qbb) are mutually exclusive. Only one feature can be enabled at a time because simultaneous enabling of both features is not supported.

The TX MAC module receives the client payload data with the destination and source addresses and then adds, appends, or updates various header fields in accordance with the configuration specified. The MAC does not modify the destination address or the payload received from the client. However, the TX MAC module adds a preamble (if the IP core is not programmed to receive the preamble from user logic), pads the payload of frames greater than eight bytes to satisfy the minimum Ethernet frame payload of 46 bytes, and calculates the CRC over the entire MAC frame. (If padding is added, it is also included in the CRC calculation). The TX MAC module can also modify the source address, and always inserts IDLE bytes to maintain an average IPG.

The following figure shows the changes that TX MAC makes to the client frame and uses the following notational conventions:

- *<p>* = payload size = 0–1,500 bytes, or 9,600 bytes for jumbo frames.
- *<s>* = padding bytes = 0–46 bytes.
- *<g>* = number of IPG bytes

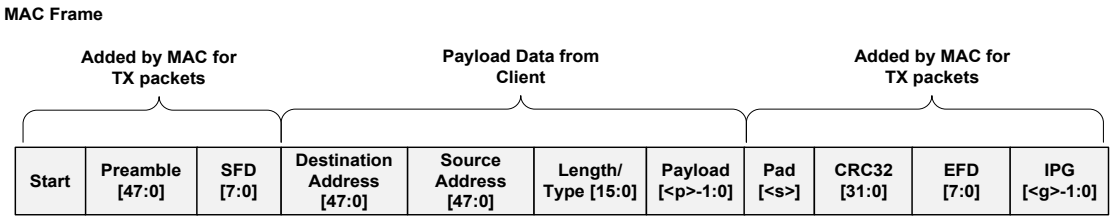

Figure 2.10. Typical Client Frame at the Transmit Interface

2.2.7.1. Frame Padding

When the length of client frame is less than 64 bytes (meaning the payload is less than 46 bytes) and greater than eight bytes, the TX MAC module inserts pad bytes (0x00) after the payload to create a frame length equal to the minimum size of 64 bytes.

2.2.7.2. Preamble Insertion and SFD Insertion

In the TX datapath the MAC appends an eight-byte preamble that begins with a Start byte (0xFB) to the client frame.

The source of the 6-byte preamble and 1-byte SFD depends on whether you turn on the TX preamble pass-through feature by setting bit 1 of the Preamble Pass-Through Configuration register at offset 0x125 .

If the TX preamble pass-through feature is turned on, the client provides the eight-byte preamble (including Start byte and final 1-byte SFD) on the data bus. However, the IP core overwrites the Start byte the client provides, and only passes on the intermediate six bytes and the SFD. The client is responsible for providing an appropriate SFD byte.

2.2.7.3. Address Insertion

The client provides the destination MAC address and the source address of the local MAC. However, if enabled by bit [31] of the MADDR_CTRL register at offset 0xC2, the source MAC address can be replaced by the source address contained in two, 32-bit MAC registers: SRC_AD_LO and SRC_AD_HI.

2.2.7.4. Length/Type Field Processing

This two-byte header represents either the length of the payload or the type of MAC frame. When the value of this field is equal to or greater than 1,536 (0x600) it indicates a type field. Otherwise, this field provides the length of the payload data that ranges from 0–1,500 bytes. The TX MAC does not modify this field before forwarding it to the network.

2.2.7.5. Frame Check Sequence (CRC-32) Insertion

The TX MAC computes and inserts a CRC32 checksum in the transmitted MAC frame. The frame check sequence (FCS) field contains a 32-bit CRC value. The MAC computes the CRC32 over the frame bytes that include the source address, destination address, length, data, and pad (if applicable). The CRC checksum computation excludes the preamble, SFD, and FCS. The encoding is defined by the following generating polynomial:

$FCS(X) = X^{32} +X^{26} +X^{23} +X^{22} +X^{16} +X^{12} +X^{11} +X^{10} +X^{8} +X^{7} +X^{5} +X^{4} +X^{2} +X +1$

CRC bits are transmitted with MSB (X^{32}) first.

Independent user configuration register bits control FCS CRC insertion at runtime. Bit [0] of the CRC_CONFIG register enables and disables CRC insertion. By default, the CRC insertion feature is enabled.

2.2.7.6. Inter-Packet Gap and Insertion

The TX MAC maintains the minimum inter-packet gap (IPG) between transmitted frames required by the IEEE 802.3 Ethernet standard. The standard requires an average minimum IPG of 96-bit time (or 12-byte time). The deficit idle counter maintains the average IPG of 12 bytes.

The MAC adjusts the IPG to compensate for Alignment Marker insertion by PHY. You can program this adjustment using the IPG_DEL_PERIOD and IPG_DEL_ENABLE registers at offsets 0x126 and 0x127, respectively. By default, the adjustment removes one idle byte for every 16,384 bytes. This removal rate corresponds to the bandwidth used by the Alignment Marker that the PHY inserts in the outgoing Ethernet communication. You can modify the value in the IPG_DEL_PERIOD register to specify more or less frequent removal of idle bytes from the sequence.

2.2.7.7. Reconciliation Layer

This MAC module also incorporates the functions of the reconciliation sublayer.

2.2.8. Receive AXI4-Stream Interface

The receive client-side interface supports the AXI4-Stream interface.

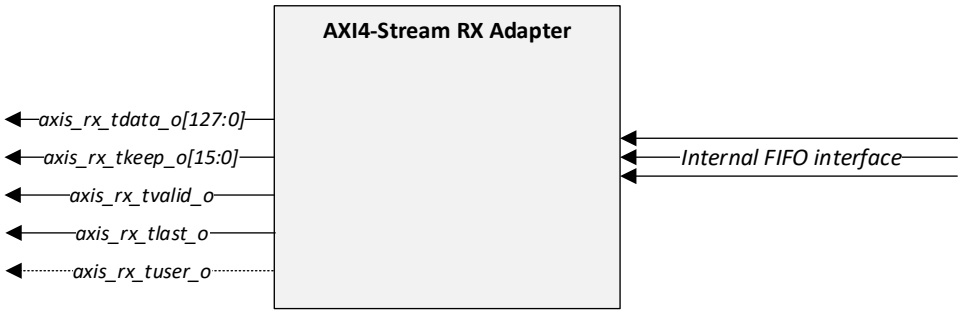

Figure 2.11. AXI4-Stream RX Adapter Interface Diagram

2.2.8.1. Default Normal Frame

The following figure shows the timing diagram of a default normal frame at the Receive AXI4-Stream interface:

- T0: MAC asserts axis_rx_tvalid_o to indicate start of frame. The valid packet includes from the DA field up to the end of the data field.
- T1: MAC asserts axis_rx_tlast_o to indicate last packet transfer.
- T2: MAC deasserts axis_rx_tvalid_o when there is no next packet transfer.

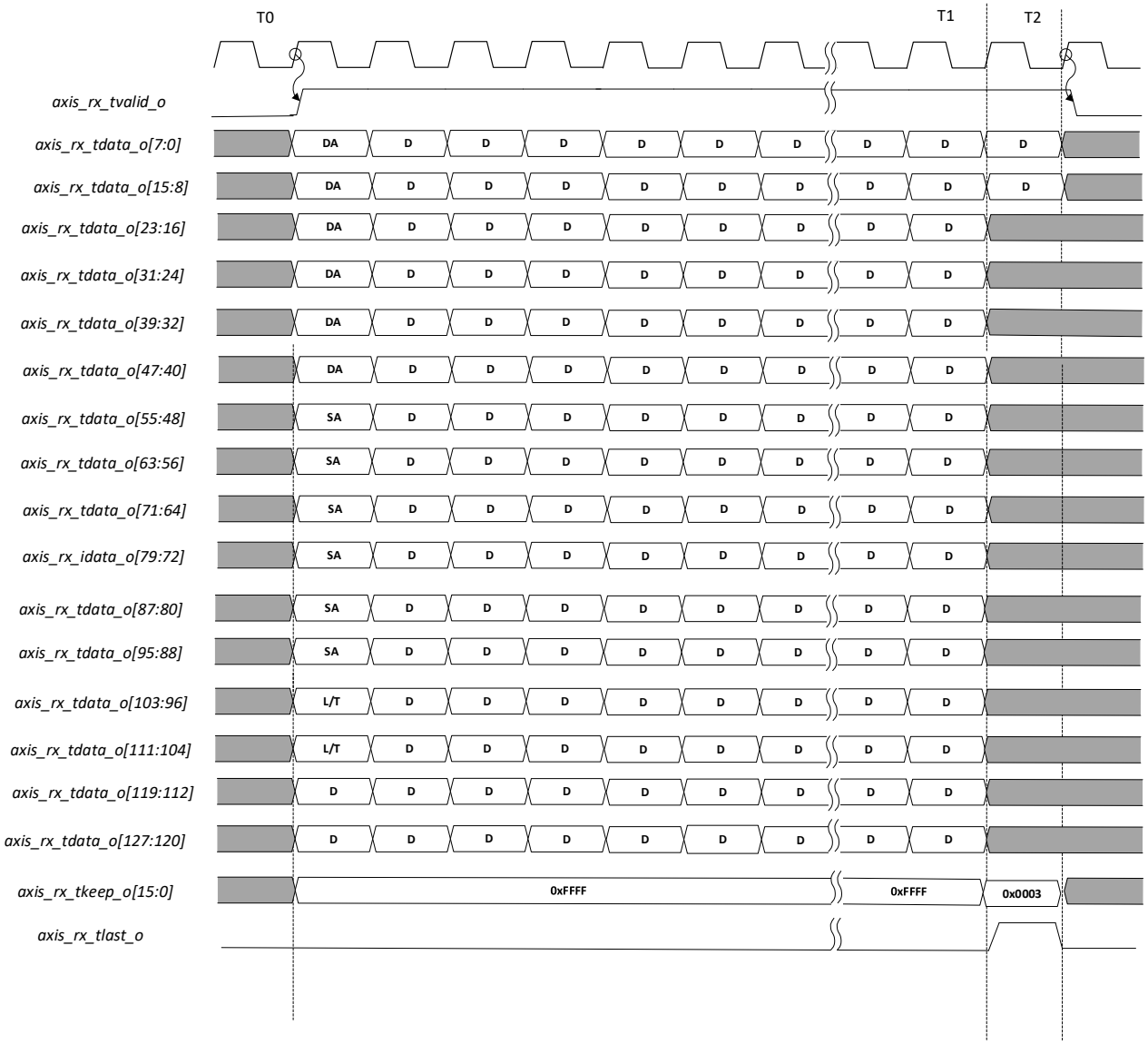

Figure 2.12. Normal Frame Reception

The following figure shows the timing diagram of a back-to-back frame at the Receive AXI4-Stream interface:

- T0: MAC asserts axis_rx_tvalid_o to indicate start of frame. The valid packet starts from the DA field up to the end of the data field.
- T1: MAC asserts axis_rx_tlast_o to indicate last packet transfer.
- T2: MAC still asserts the axis_rx_tvalid_o to indicate another packet transfer.

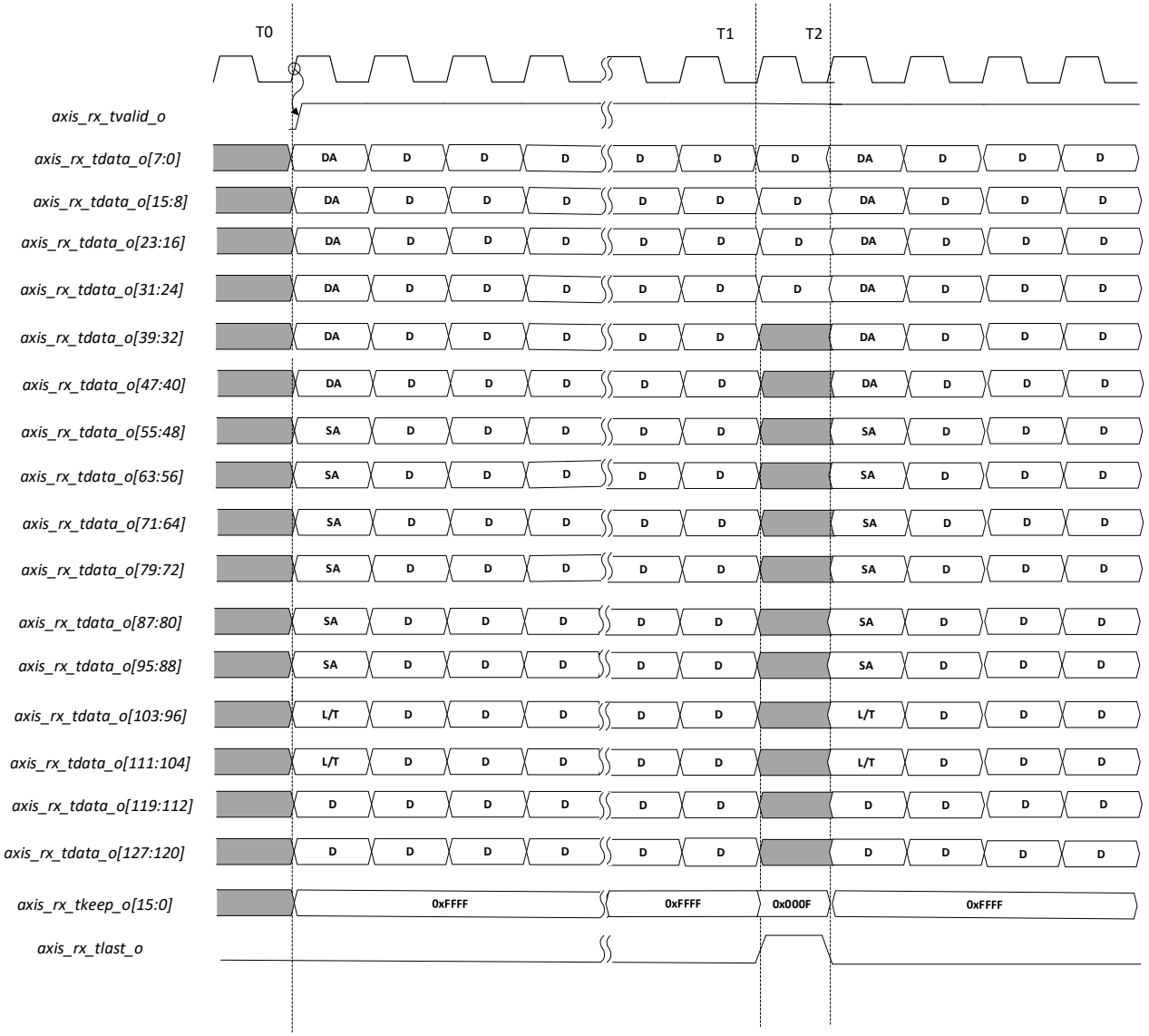

Figure 2.13. Back-to-back Frames Reception

2.2.8.2. In-Band FCS Passing

The following figure shows the timing diagram of a frame with in-band FCS. The FCS field is included inside the valid frame.

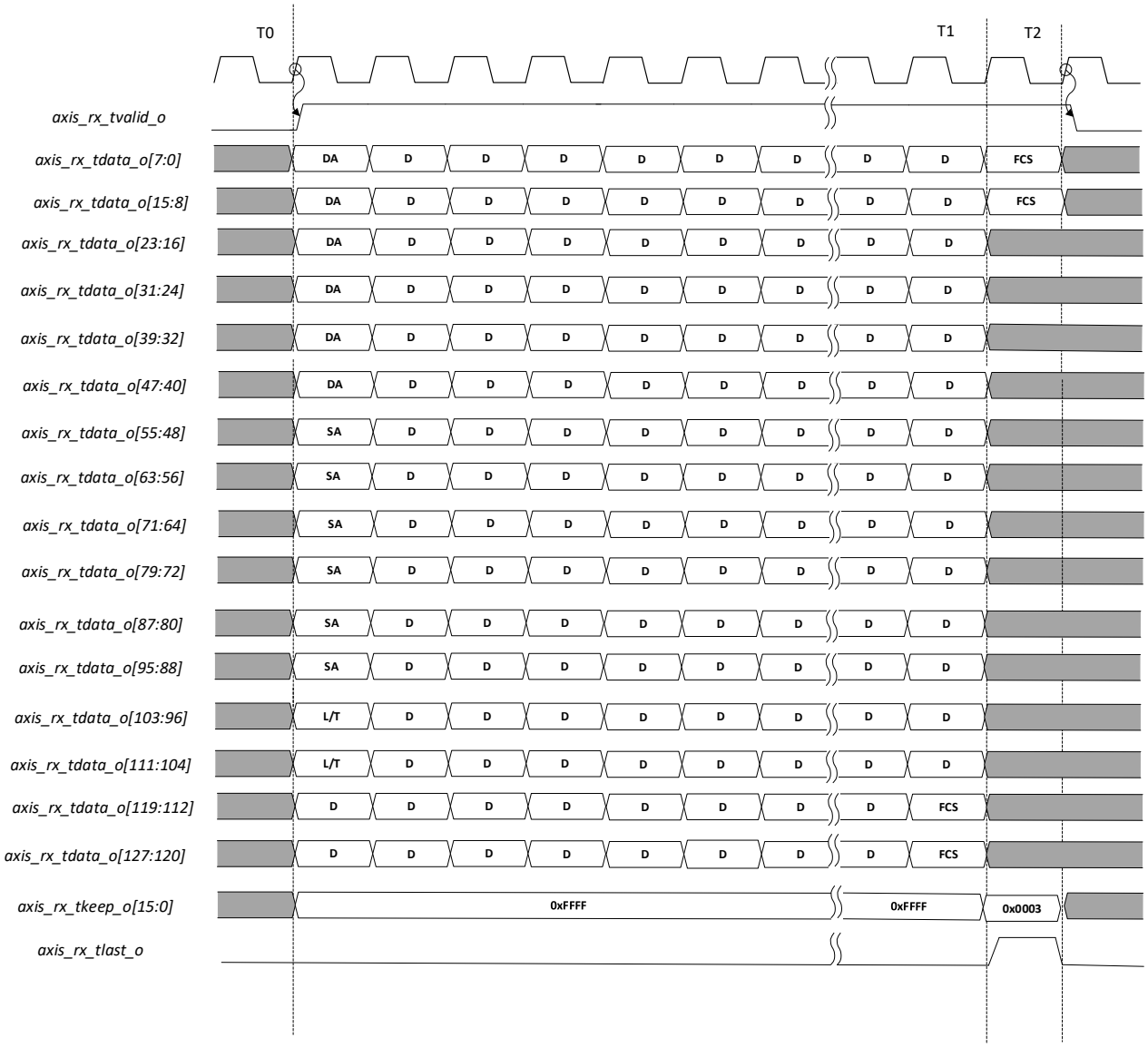

Figure 2.14. Frame Reception with In-Band FCS Passing

2.2.8.3. Custom Preamble Passing

The following figure shows the timing diagram of a frame with custom preamble mode enabled. A custom preamble (CP) field is included inside the valid frame.

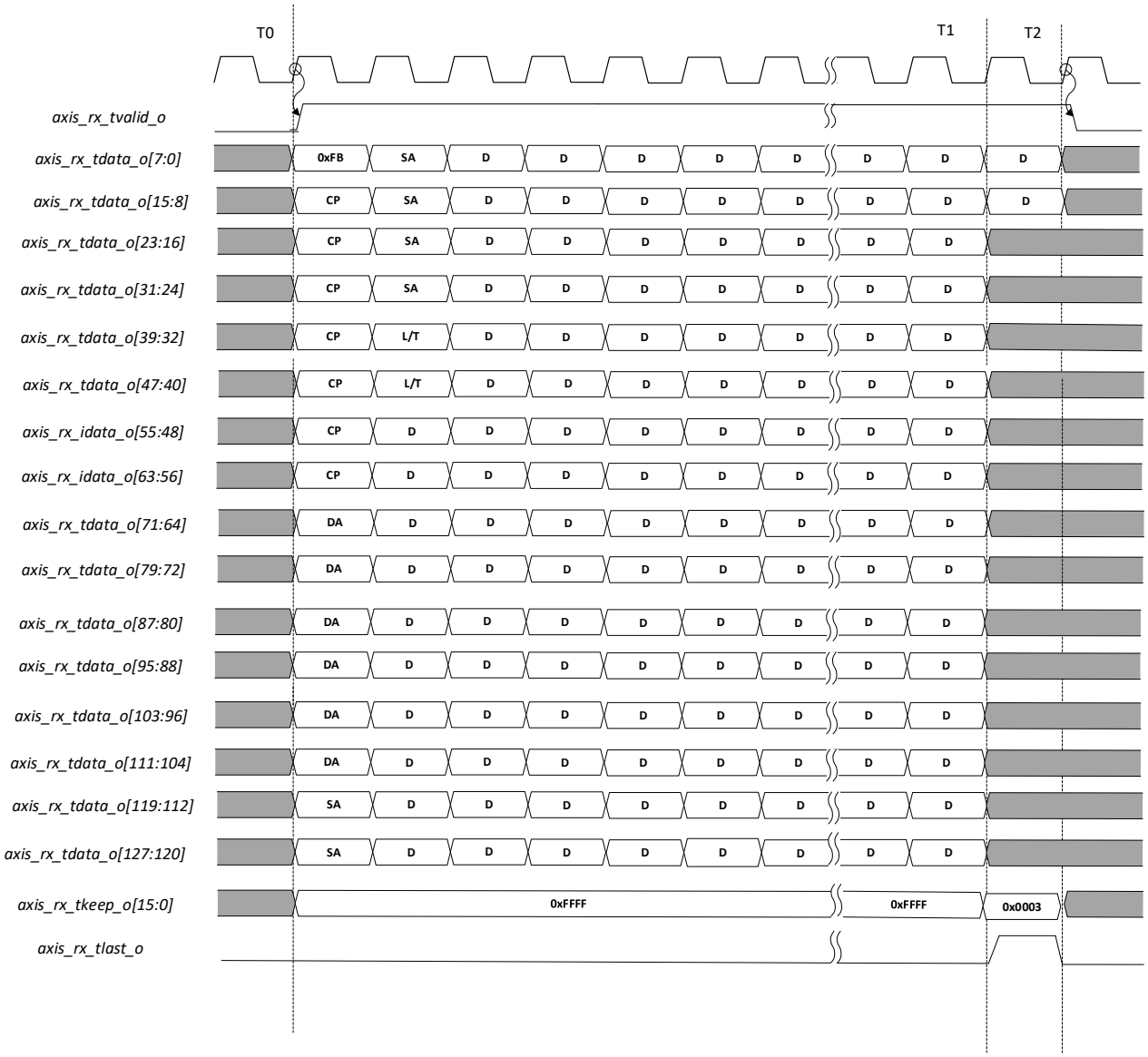

Figure 2.15. Reception with Custom Preamble

2.2.9. Transmit AXI4-Stream Interface

The transmit client-side interface supports the AXI4-Stream interface.

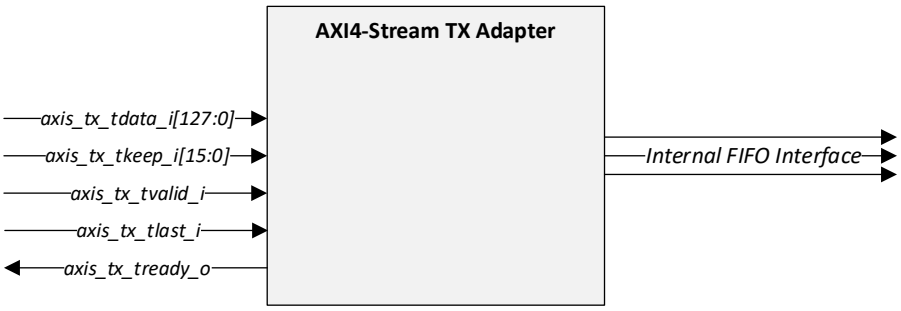

Figure 2.16. AXI4-Stream TX Adapter Interface Diagram

2.2.9.1. Default Normal Frame

The following figure shows the timing diagram of a default normal frame at the Transmit AXI4-Stream interface:

- T0: Client asserts axis_tx_tvalid_i to indicate the start of packet transfer. It must hold the data and axis_tx_tvalid_i until axis_tx_tready_o asserts.
- T1: Client continues to send data once it sees axis_tx_tready_o asserted.
- T2: Client asserts axis_tx_tlast_i to indicate last packet transfer.
- T3: Client deasserts axis_tx_tvalid_i when there is no packet transfer. Transmit MAC also deasserts axis_tx_tready_o once it sees the last packet transfer (axis_tx_tlast_i = 1).

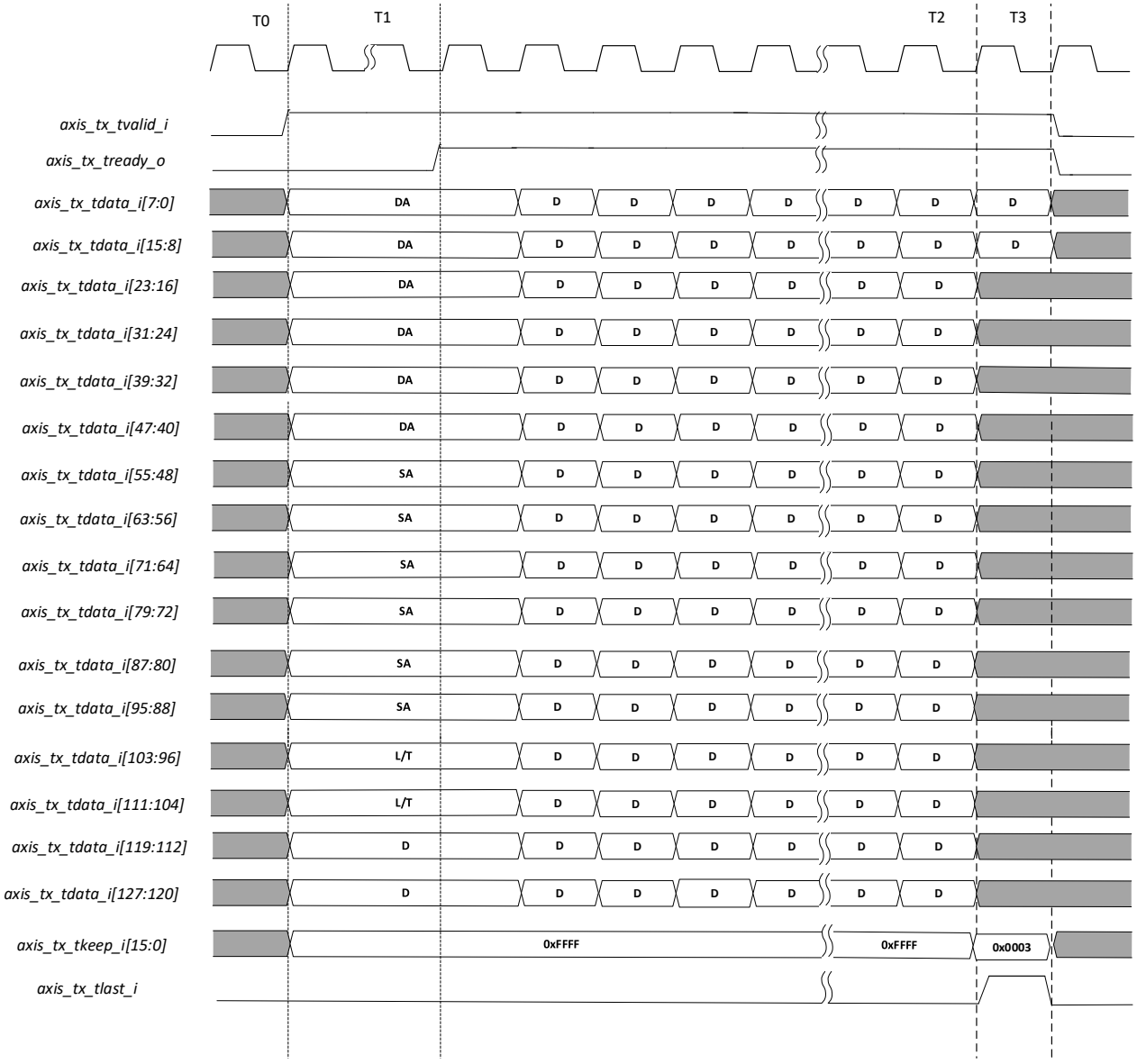

Figure 2.17. Default Normal Frame Transmission

2.2.9.2. In-Band FCS Passing

The following figure shows the timing diagram of a frame when In-Band FCS passing is enabled.

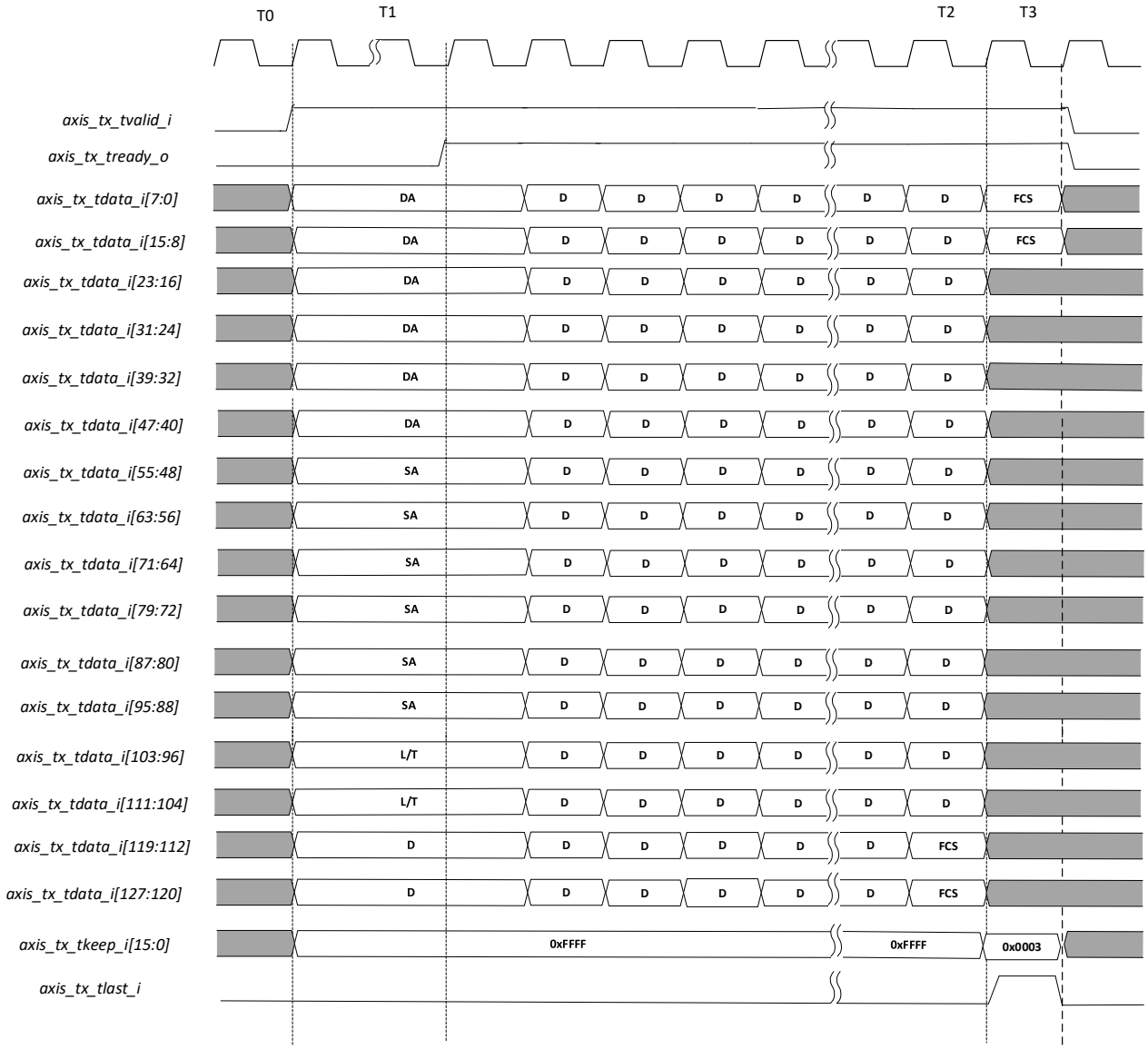

Figure 2.18. Transmission with In-Band FCS Passing

2.2.9.3. Custom Preamble Passing

The following figure shows the timing diagram of a frame when custom preamble passing is enabled.

Figure 2.19. Transmission with Custom Preamble Passing

2.2.10. MAC Management Block

The MAC Management block is accessed through the AXI4-Lite interface. This block is responsible for the following:

- Configuration of the core
- Access to interrupt block
- Access to statistics counter

The various events that occur during the reception of a frame are also logged into the statistics vector signals (rx_statvec_o) and the TX_RX_STS register. At the end of reception, the rx_staten_o signal is asserted to qualify the rx_statvec_o signal. A vector is not generated for all those frames that are discarded (no address match or frame length is less than 64 bytes) or ignored (you assert the ignore_pkt of MAC_CTL register). For every frame transmitted, a statistics vector signals (tx_statvec_o) is generated, including all the statistical information collected in the process of transmitting the frame. Data on tx_statvec_o is qualified by assertion of the tx_staten_o signal. These statistics can also be accessed in the statistics counter when the *Statistics Counter Register* attribute is enabled.

For the timing details of the AMBA AXI protocol, refer to the AMBA AXI Protocol version 1.0 Specification.

2.3. PHY

2.3.1. IP Architecture Overview

The 25G Ethernet PHY IP core provides the 25GMII interface to the MAC and follows the IEEE 802.3 25GBASE-R standard. It supports 128 bits of data and 16 bits of control signals for both the transmit and receive paths.

This IP core instantiates the MPPHY foundation IP configured as 1-Lane 64b/66b PCS and a Management block that supports AXI4-Lite access to PCS and MMD registers.

The following figure shows the top-level block diagram of the 25G Ethernet PHY IP core.

2.3.2. Block Diagram

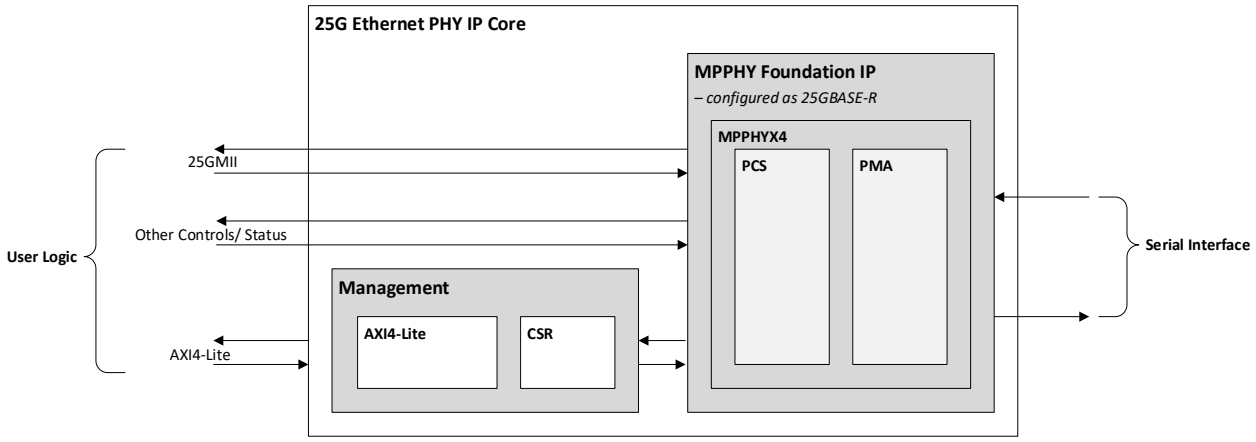

Figure 2.20. 25G Ethernet PHY IP Core Block Diagram

2.3.3. Lane Merging

2.3.3.1. Overview

The LAV-AT-G/X70 silicon can have up to seven usable quads while the LAV-AT-G/X30 silicon have up to three usable quads depending on the package. Therefore, a single device could theoretically contain up to 28 of these quads.

You can configure the quad to use each lane individually, enabling you to merge multiple MPPHY instances into a single physical quad. This maximizes the usage of silicon resources in your design.

Note: Lane merging is only supported by LAV-AT-G70 and LAV-AT-X70 devices. For more information, refer to the Lattice Avant-G/X MPPHY Module User Guide (FPGA-IPUG-02233).

2.3.3.2. Usage

The 25G Ethernet PHY IP core instantiates the MPPHY foundation IP configured with a 1X1 link width, which occupies a single lane of a quad. By default, the Radiant software attempts to merge MPPHY instances to minimize the device power consumption.

If you want to override this behavior and select specific locations for the MPPHY instances, you have two options:

- Use top-level port constraints (ldc_set_location) to control the pin or pad assignments of the top-level design ports such as reference clock, TX/RX data, and so on.
  For example, given two 25G Ethernet PHY IP core instances inst0 and inst1, there could be two top-level design ports:
  - input pad_rxp_i_inst0
  - input pad_rxp_i_inst1

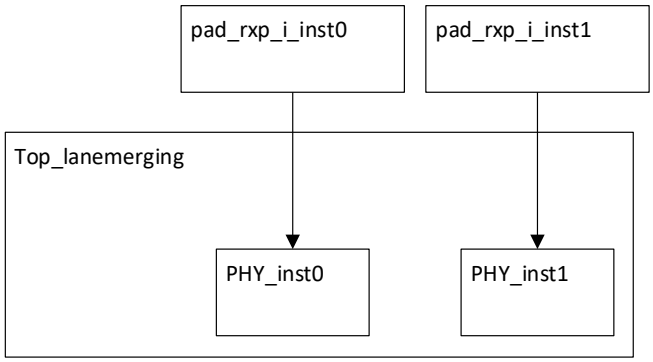

Figure 2.21. MPPHY Instances Before Lane Merging

If you want to constraint inst0 to Lane 0 and inst1 to Lane 1 of MPPHY Quad 0, use the following constraints:

- ldc_set_location -site MPQ0_RX0P [get_ports pad_rxp_i_inst0]
- ldc_set_location -site MPQ0_RX1P [get_ports pad_rxp_i_inst1]

As a result of these constraints, inst0 and inst1 are merged into MPPHY Quad 0.

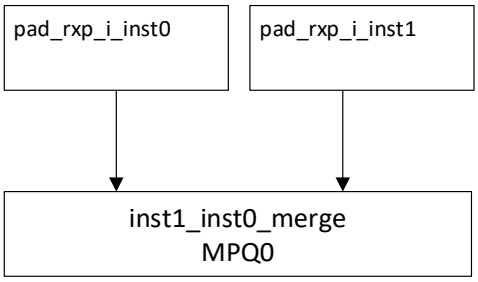

Figure 2.22. MPPHY Instances After Lane Merging

You can create constraints using the Device Constraint editor or by manually entering them into the constraint file. The package pin name can also be used, depending on the selected package.
Note that an MPPHY quad shares a single reference clock for all four lanes. Therefore, a reference clock constraint will constrain an instance to a particular quad, rather than to a specific lane.

- Set the LANE ID configuration of the IP according to the number of quad you want to use. Follow these steps:

1. In the Select IP Option field, select **PHY only** or **MAC + PHY.**
2. In the PCS Lane ID field, set the lane ID configuration of the IP as shown in the figure below. By default the ID is set to **AUTO**. If the **AUTO** Lane ID is selected, the pin location assignment must be included in sdc or ldc constraint file . If the ID is not set to **AUTO**, in the case of a conflict between the Lane ID configuration and a top-level design port constraint, the top-level design port constraint takes precedence, which means the Lane ID setting is ignored and a warning message is shown.

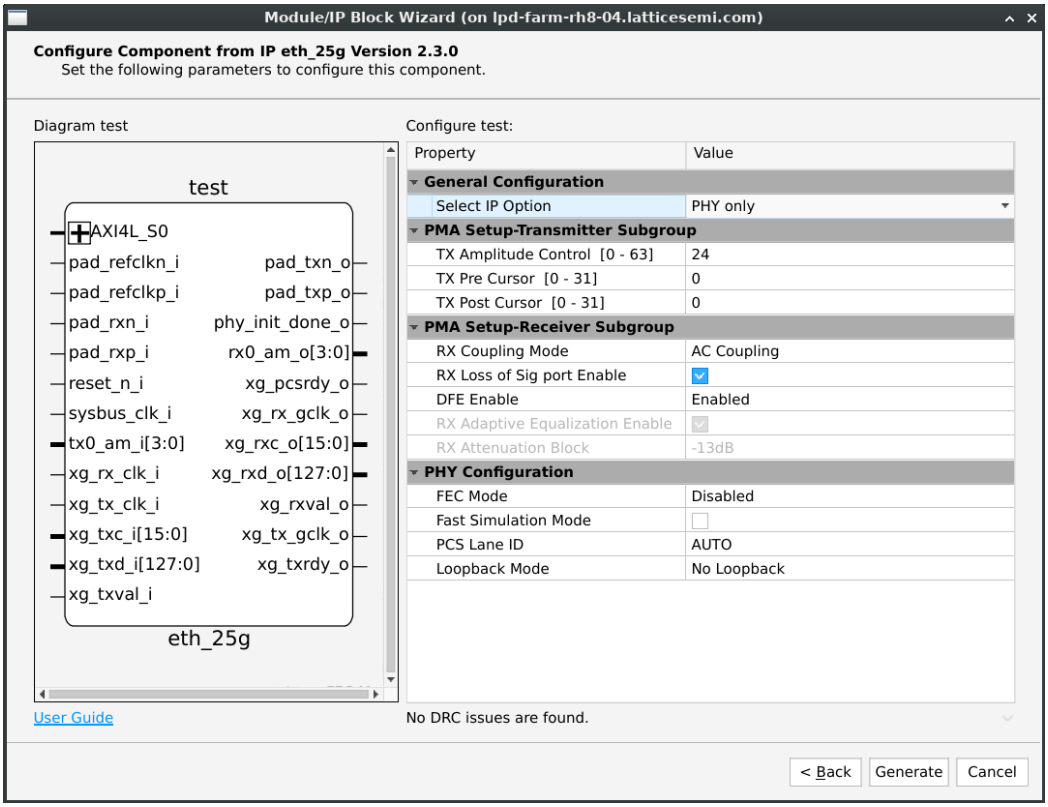

Figure 2.23. Set the Lane ID Configuration of the 25G Ethernet PHY IP Core

The Lane ID value is global for the entire device, starting at 0 from Lane 0 of the left-most quad, then incrementing up to 27 for Lane 3 of the right-most quad for the largest package of LAV-AT-X70. An example of Lane ID numbering is shown in the figure below.

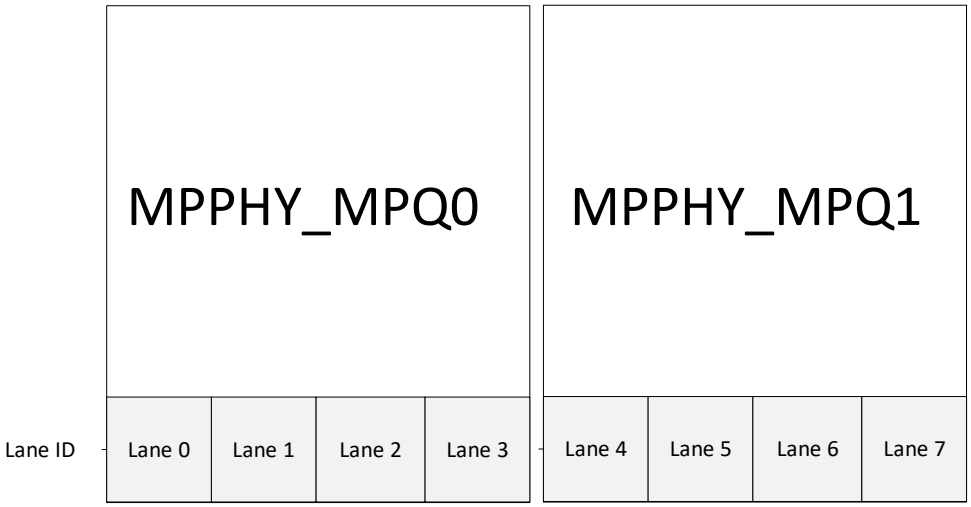

Figure 2.24. Lane ID Numbering

2.3.3.3. Lane Merging Report

During lane merging, the Post-Synthesis process produces a report file named *mpphy_lane_assignment.mrf* in the project's active implementation folder that shows how the MPPHY design instances were merged into the device MPPHY quads. You can open the report file with any text editor.

```
MPPHYX4_MPQ0: new instance 'MPPHYX4_MPQ0_merge'
    ~~ Lane Assignment Within Quad ~~
        Lane 3: -- un-assigned --
        Lane 2: -- un-assigned --
        Lane 1: 'inst_1/lscc_2p5_gbe_top_inst/genblk1.genblk1.lscc_2p5_gbephy_inst/mpphy_03A_inst/lscc_mpphy_inst/mpp_nonpipe_x1.u_nonpipe_x1' (MPPHYX1A)
            Assignment is due to:
                - auto-assigned
        Lane 0: 'inst_0/lscc_2p5_gbe_top_inst/genblk1.genblk1.lscc_2p5_gbephy_inst/mpphy_03A_inst/lscc_mpphy_inst/mpp_nonpipe_x1.u_nonpipe_x1' (MPPHYX1A)
            Assignment is due to:
                - auto-assigned

    ~~ PLL Assignment Within Quad (determined by protocol/data rate) ~~
        MPLLA:  -- un-assigned --
        MPLLB:  inst_1/lscc_2p5_gbe_top_inst/genblk1.genblk1.lscc_2p5_gbephy_inst/mpphy_03A_inst/lscc_mpphy_inst/mpp_nonpipe_x1.u_nonpipe_x1, inst_0/lscc
_2p5_gbe_top_inst/genblk1.genblk1.lscc_2p5_gbephy_inst/mpphy_03A_inst/lscc_mpphy_inst/mpp_nonpipe_x1.u_nonpipe_x1
```

Figure 2.25. Lane Merging Report File

2.3.3.4. Restrictions and Limitations

To merge instances into the same quad, instances must abide by the following restrictions:

- Shared reference clock connection.
- Shared LMMI clock and reset connections.
- Compatible PLL settings—This is protocol or data rate dependent, and is configured by the IP Catalog tool.
- Compatible "per-quad" connections—a limited number of ports exposed on the MPPHY IP physically have only a single instance on the silicon. These must be connected to the same net if it is an input, or have a maximum of one connected per-quad if it is an output.
- All input must be driven by the same source. For output clock, use only the output clock from one instance of the quad to drive the logic. For output clocks from other IP instances that are merged into the same quad, do not use these clocks to drive any logic. Refer to the table below.

Table 2.1. Shared Signal Mapping of 25G Ethernet IP to MPPHY for Lane Merging

MPPHY	25G Ethernet IP	Direction
lmmiclk_q0_i	sysbus_clk_i	Input
refclk_p_q0_i refclk_n_q0_i	pad_refclkp_i pad_refclkn_i	Input
lmmireset_n_q0_i	reset_n_i	Input
txoutgclk_pll0_q0_o	xg_tx_gclk_o[0]	Output
txoutgclk_pll1_q0_o	xg_tx_gclk_o[1]	Output

If any of these restrictions are violated, the Radiant software will not automatically merge the MPPHY instances into a single quad. If user constraints or lane assignment forces incompatible MPPHY instances into the same quad, an error message is issued, and the Radiant software flow will not continue past the Post-Synthesis stage.

2.3.4. PHY Management Block

The PHY Management block is accessed through the AXI4-Lite interface. This block is responsible for register access of the PCS registers.

For PCS register access through AXI4-Lite, this soft IP requires the use of remapped addresses (0xA000 – 0xA0FC) as AXI4-Lite addresses must be DWORD-aligned (0xA000, 0xA004, 0xA008, 0xA00C, and so on). To get the corresponding AXI4-Lite addresses, you must left shift even-numbered PCS address once (or multiply by two) – the lower 2 bytes of the AXI4-Lite read/write data will be mapped to the even-numbered PCS register; while the upper 2 bytes of the AXI4-Lite read/write data maps to the subsequent odd-numbered PCS register.
For details and illustrations on how the AXI4-Lite address remapping works, refer to Table 2.2 and Figure 2.26.

Note: The read/write data values used in the following illustrations are just examples and are not the suggested register values to be written. Because PCS registers are half-duplex, simultaneous write to/read from PCS registers through AXI4-Lite channels are not supported.

Table 2.2. AXI4-Lite to PCS Address and Data Conversion

axi4l_awaddr_i[31:0] / axi4l_araddr_i[31:0]	axi4l_wdata_i[31:0] / axi4l_rdata_o[31:0]	PCS Register Address (16-bit)	Access Types	PCS Register Values (16-bit)
0xA000	0x22334455	0x5000	RW	0x4455 = axi4l_wdata_i[15:0]
		0x5001	—	Not available
0xA0004	0xaabbccdd	0x5002	RW	0xccdd = axi4l_wdata_i[15:0]
		0x5003	RW	0xaabb = axi4l_wdata_i[31:16]
0xA008	0x11335577	0x5004	RW	0x5577 = axi4l_wdata_i[15:0]
		0x5005	RW	0x1133 = axi4l_wdata_i[31:16]
0xA0FC	0x8899eeff	0x507E	RO	Read-only register
		0x507F	RW	0x8899 = axi4l_wdata_i[31:16]

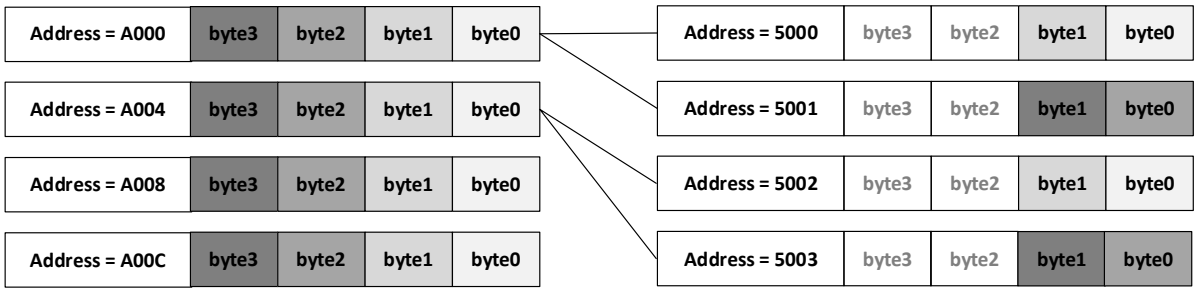

Figure 2.26. Remapped Addresses for PCS Register Address through the AXI4-Lite Interface

2.3.5. Loopback Modes

2.3.5.1. Far End Parallel Loopback

The input signal is driven at the serial RX port (RX PMA input) and the output is observed at the serial TX port (TX PMA output). In this loopback, user logic is not involved.

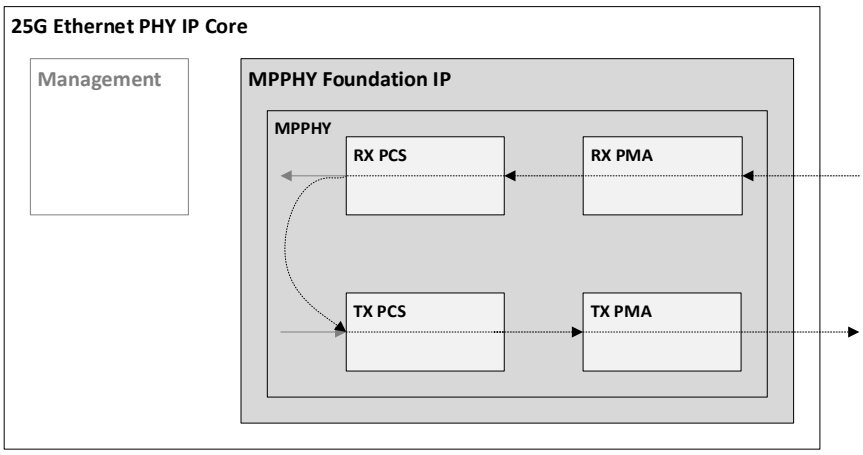

Figure 2.27. Far End Parallel Loopback

2.3.5.2. Near End Parallel Loopback

The input signal is driven at the parallel TX port (user logic side) and the output signal is observed at the parallel RX port (user logic side). In this loopback, PMA is not involved.

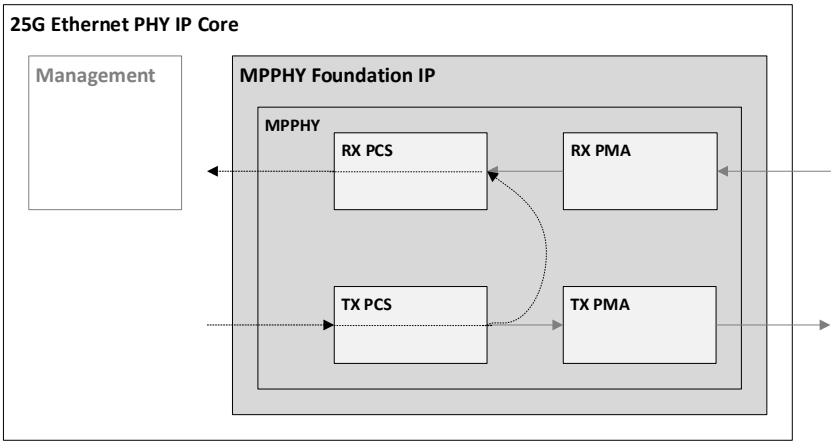

Figure 2.28. Near End Parallel Loopback

2.4. Clocking

2.4.1. Clocking Overview

The following figure shows the clock network of the 25G Ethernet IP core. The clock frequency requirements depends on the configuration selected—be it MAC only, PHY only, or MAC+PHY. The configuration details are described in the Signal Description section. In MAC + PHY configuration, the txmac_clk_i and rxmac_clk_i are connected internally to the output clock from the PHY block.

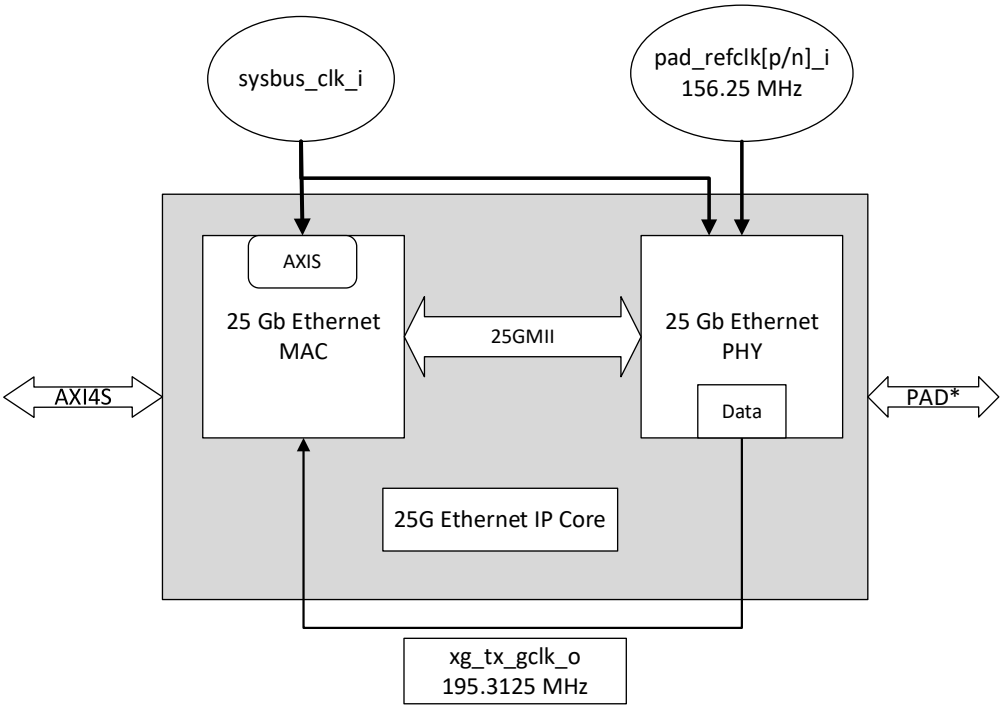

Figure 2.29. Clock Network Diagram

2.4.2. RSFEC Clocking Overview

The following figure shows the RSFEC clock network of the 25G Ethernet IP core. The clock frequency requirements depends on the configuration selected—be it PHY only or MAC+PHY. The configuration details are described in the Signal Description section. In MAC + PHY configuration, the txmac_clk_i and rxmac_clk_i are connected internally to the output clock from the PHY block. To close timing comfortably when RSFEC is enabled, use statistic counter of 32 bit instead of 64 bit.

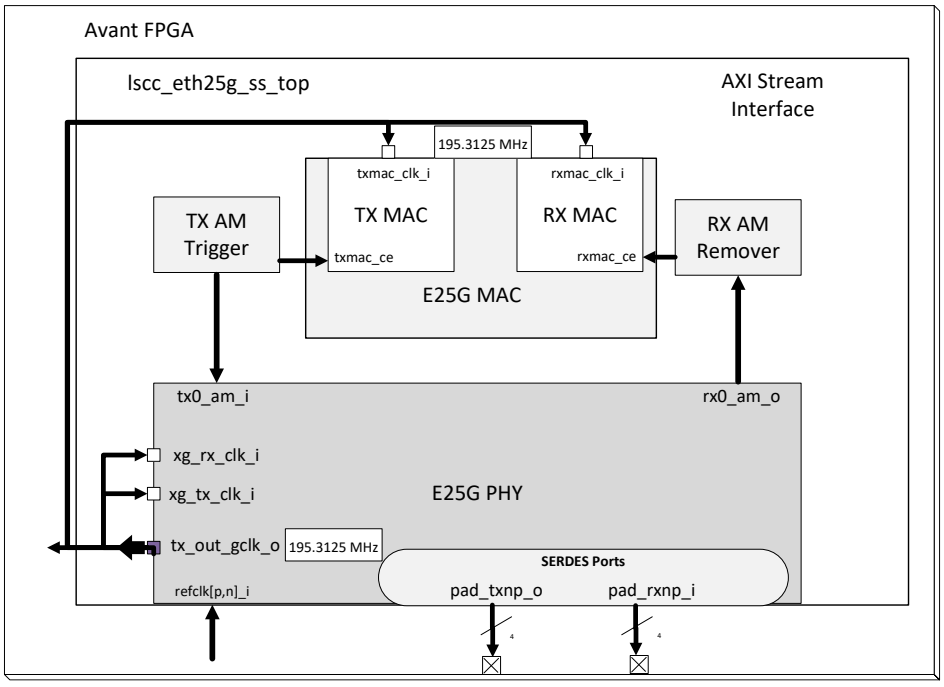

Figure 2.30. RSFEC Clock Network Diagram

2.5. Reset

2.5.1. MAC Reset Sequence

2.5.1.1. Reset

An asynchronous reset pin (active low) as system reset is used for resetting the 25G Ethernet IP core. Internal reset logic is implemented to guarantee synchronous deassertion all throughout the different clock domain among soft logic blocks. The minimum assertion of reset is five clock cycles of the slowest clock (for example, 20 MHz).

2.5.1.2. Power Up Sequence

The following lists the sequence after power-up of the chip or system:

1. Assert reset pin for five clock cycles of the slowest clock of the system (for example, 20 MHz).
2. Wait for the 25G PHY to be in a ready state. For more information on the sequence, see the PHY section.
3. Send settings through the AXI4-Lite interface to configure the system. For a list of MAC registers, refer to the Configuration Registers for MAC section.
4. Transmit or receive frames.

The following figure shows a sample sequence when RX MAC is configured to pass the FCS field to the client.

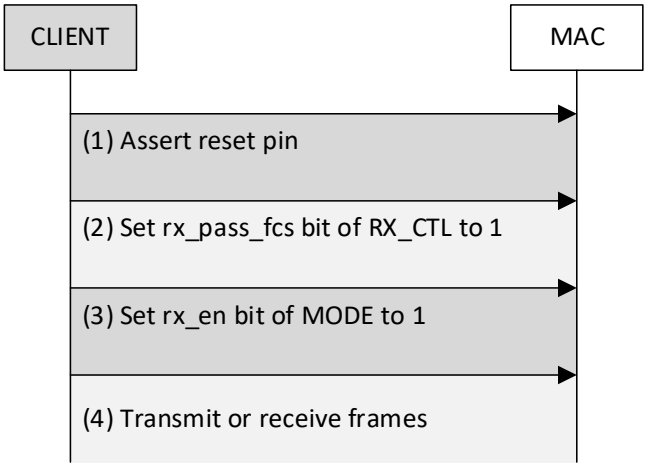

Figure 2.31. Sequence to Configure RX MAC In-Band FCS Passing

2.5.2. PHY Reset Sequence

2.5.2.1. Reset

An asynchronous reset pin (active low) as system reset is used for resetting the 25GbE PHY IP core. Internal reset logic is implemented to guarantee synchronous deassertion all throughout the different clock domain among soft logic blocks. The minimum assertion of reset is five clock cycles of the slowest clock.

2.5.2.2. Init Sequence

The following is the sequence after power-up of the chip or system:

1. Assert reset pin, reset_n_i, for five clock cycles of the slowest clock of the system (20 MHz).
2. Wait for phy_init_done_o to assert. It is expected during this time (before T0), xg_tx_out is not toggling yet.
3. After T0, user drives continuous IDLE patterns until xg_pcsrdy_o asserts.
4. After T1, transmit or receive frames.

The following figure shows the PHY initialization sequence.

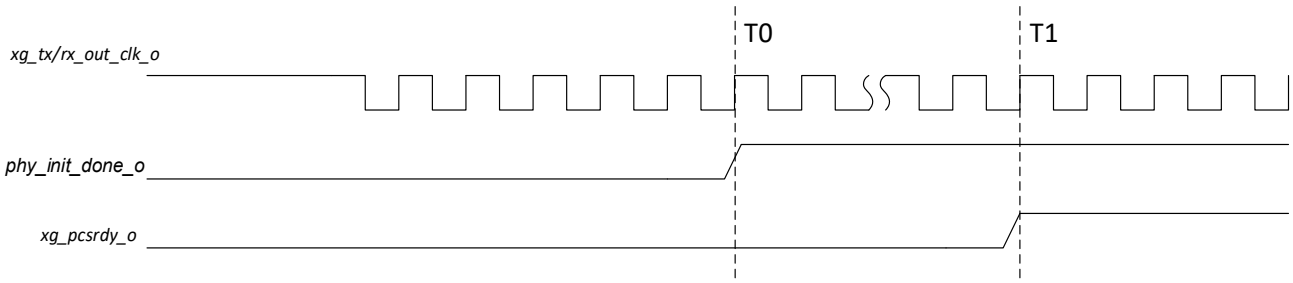

Figure 2.32. PHY Initialization Sequence

2.6. Latency

The following table provides the measured latency information for 25G Ethernet IP.

Table 2.3. MAC + PHY Attributes (Avant Devices)

Core	Core Configuration	Latency (ns)	User Bus Width (bits)	Core Clock Frequency (MHz)
MAC + PHY	25G BASE-R TX	194	128	195.3125
MAC + PHY	25G RS-FEC TX	251	128	195.3125
MAC + PHY	25G FC-FEC TX	211	128	195.3125
MAC + PHY	25G BASE-R RX	195	128	195.3125
MAC + PHY	25G RS-FEC RX	552	128	195.3125
MAC + PHY	25G FC-FEC RX	305	128	195.3125

3. IP Parameter Description

3.1. MAC + PHY

The following table lists the 25G Ethernet IP core configurable attributes for the *MAC + PHY* option.

Table 3.1. Configurable Attributes for MAC + PHY Option

Attribute	Selectable Values	Default	Description	Dependency on Other Attributes
General Configuration				
Select IP Option	• MAC + PHY • PHY only • MAC only	MAC + PHY	IP option	—
MAC Configuration				
Multicast Address Filtering	• Enabled • Disabled	Disabled	Enables or disables address filtering for multicast frames.	Enabled when *Select IP Option == MAC Only* or *MAC + PHY*
TX Pause Frame Generation via Ports	• Enabled • Disabled	Disabled	Enables or disables TX pause frame generation via ports.	Enabled when *Select IP Option == MAC only* or *MAC + PHY*
Statistic Counter Configuration				
Statistics Counter Registers	• Enabled • Disabled	Disabled	Enables or disables statistics counter registers	Can be enabled when *Select IP Option == MAC only* or *MAC + PHY*
Counter Width	• 32 • 64	32	Width size of the statistic counter.	Enabled when *Statistic Counter Register == Enabled*
TX Statistics	• Enabled • Disabled	Disabled	Enables or disables the TX path statistic counters.	Enabled when *Statistic Counter Register == Enabled*
RX Statistics	• Enabled • Disabled	Disabled	Enables or disables the RX path statistic counters.	Enabled when *Statistic Counter Register == Enabled*
PMA Setup-Transmitter Subgroup (default values are recommended)				
TX Amplitude Control [0-63]	• 0-63	24	Transmitter amplitude adjustment control	Enabled when *Select IP Option == PHY Only* or *MAC + PHY*
TX Pre Cursor [0-31]	• 0-31	0	Transmitter pre-emphasis level adjustment control	Enabled when *Select IP Option == PHY Only* or *MAC + PHY*
TX Post Cursor [0-31]	• 0-31	0	Transmitter post-emphasis level adjustment control	Enabled when *Select IP Option == PHY Only* or *MAC + PHY*
PMA Setup-Receiver Subgroup (default values are recommended)				
RX Coupling Mode	• AC Coupling • DC Coupling	AC Coupling	PMA Coupling mode	Enabled when *Select IP Option == PHY Only* or *MAC + PHY*
RX Loss of Sig port Enable	• Enabled • Disabled	Enabled	RX Loss of Sig capability.	Enabled when *Select IP Option == PHY Only* or *MAC + PHY*
DFE Enable	• Enabled • Disabled	Enabled	DFE capability.	Enabled when *Select IP Option == PHY Only* or *MAC + PHY*

Attribute	Selectable Values	Default	Description	Dependency on Other Attributes
RX Adaptive Equalization Enable	• Enabled • Disabled	Enabled	RX adaptive EQ capability.	Enabled when *Select IP Option == PHY Only* or *MAC + PHY* Selectable only when *DFE Enable == Disabled*
RX Attenuation Block	• -13dB • -2dB	-13dB	RX Attenuation setting. Higher dB loss reflects higher capability of the RX to handle a lossy signal.	Enabled when *Select IP Option == PHY Only* or *MAC + PHY* Selectable only when *RX Adaptive Equalization Enable == Disabled*
PHY Configuration				
FEC Mode	• RSFEC • FCFEC • Disabled	Disabled	Enables the RSFEC, FCFEC, or disables the FEC mode.	Enabled when *Select IP Option == PHY only* or *MAC + PHY*
Fast Simulation Mode	• Enabled • Disabled	Disabled	Enables or disables the fast simulation mode.	Enabled when *Select IP Option == PHY only* or *MAC + PHY*
PCS Lane ID	AUTO, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27	AUTO	Specifies the location of the first lane of the PCS instance.	Enabled when *Select IP Option == PHY only* or *MAC + PHY*
Loopback Mode	• Far End Parallel Loopback • Near End Parallel Loopback • No Loopback	No Loopback	Enables the far end parallel loopback, near end parallel loopback, or disables the loopback.	Enabled when *Select IP Option == PHY only or MAC + PHY*

3.2. MAC Only

The following table lists the 25G Ethernet IP core configurable attributes for the *MAC only* option.

Table 3.2. Configurable Attributes for MAC Only Option

Attribute	Selectable Values	Default	Description	Dependency on Other Attributes
General Configuration				
Select IP Option	• MAC + PHY • PHY only • MAC only	MAC + PHY	IP option.	—
MAC Configuration				
TX Pause Frame Generation via Ports	• Enabled • Disabled	Disabled	Enables or disables TX pause frame generation via ports.	Enabled when *Select IP Option == MAC only* or *MAC + PHY*
Multicast Address Filtering	• Enabled • Disabled	Disabled	Enables or disables multicast address filtering	Enabled when *Select IP Option == MAC only* or *MAC + PHY*
Statistic Counter Configuration				
Statistics Counter Registers	• Enabled • Disabled	Disabled	Enables or disables statistics counter registers.	Can be enabled when *Select IP Option == MAC only* or *MAC + PHY*
Counter Width	• 32 • 64	32	Width size of the statistic counter.	Enabled when *Statistic Counter Register == Enabled*
TX Statistics	• Enabled • Disabled	Disabled	Enables or disables the TX path statistic counters.	Enabled when *Statistic Counter Register == Enabled*
RX Statistics	• Enabled • Disabled	Disabled	Enables or disables the RX path statistic counters.	Enabled when *Statistic Counter Register == Enabled*

3.3. PHY Only

The following table lists the attributes when the 25G Ethernet IP core is configured as *PHY only* option.

Table 3.3. Configurable Attributes for PHY Only Option

Attribute	Selectable Values	Default	Description	Dependency on Other Attributes
General Configuration				
Select IP Option	• MAC + PHY • PHY only • MAC only	MAC + PHY	Selects the IP connection. This section discusses the *PHY only* option.	—
PMA Setup-Transmitter Subgroup (default values are recommended)				
TX Amplitude Control [0-63]	• 0-63	24	Transmitter amplitude adjustment control	Enabled when *Select IP Option == PHY Only* or *MAC + PHY*
TX Pre Cursor [0-31]	• 0-31	0	Transmitter pre-emphasis level adjustment control	Enabled when *Select IP Option == PHY Only* or *MAC + PHY*
TX Post Cursor [0-31]	• 0-31	0	Transmitter post-emphasis level adjustment control	Enabled when *Select IP Option == PHY Only* or *MAC + PHY*
PMA Setup-Receiver Subgroup (default values are recommended)				

Attribute	Selectable Values	Default	Description	Dependency on Other Attributes
RX Coupling Mode	• AC Coupling • DC Coupling	AC Coupling	PMA Coupling mode	Enabled when *Select IP Option == PHY Only* or *MAC + PHY*
RX Loss of Sig port Enable	• Enabled • Disabled	Enabled	RX Loss of Sig capability.	Enabled when *Select IP Option == PHY Only* or *MAC + PHY*
DFE Enable	• Enabled • Disabled	Enabled	DFE capability.	Enabled when *Select IP Option == PHY Only* or *MAC + PHY*
RX Adaptive Equalization Enable	• Enabled • Disabled	Enabled	RX adaptive EQ capability.	Enabled when *Select IP Option == PHY Only* or *MAC + PHY* Selectable only when *DFE Enable == Disabled*
RX Attenuation Block	• -13dB • -2dB	-13dB	RX Attenuation setting. Higher dB loss reflects higher capability of the RX to handle a lossy signal.	Enabled when *Select IP Option == PHY Only* or *MAC + PHY* Selectable only when *RX Adaptive Equalization Enable == Disabled*
PHY Configuration				
FEC Mode	• RSFEC • FCFEC • Disabled	Disabled	Enables the RSFEC, RCFEC, or disables the FEC mode.	Enabled when *Select IP Option == PHY only* or *MAC + PHY*
Fast Simulation Mode	• Enabled • Disabled	Disabled	Enables or disables the fast simulation mode.	Enabled when *Select IP Option == PHY only* or *MAC + PHY*
PCS Lane ID	AUTO,0,1,2,3,4,5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27	AUTO	Specifies the location of the first lane of the PCS instance.	Enabled when *Select IP Option == PHY only* or *MAC + PHY*
Loopback Mode	• Far End Parallel Loopback • Near End Parallel Loopback • No Loopback	No Loopback	Enables the far end parallel loopback, near end parallel loopback, or disables the loopback.	Enabled when *Select IP Option == PHY only or MAC + PHY*

4. Signal Description

4.1. MAC + PHY

Table 4.1. Signal Description for MAC + PHY

Port Name	Clock Domain	I/O	Width	Description
Clock and Reset				
reset_n_i	Asynchronous	In	1	Active-low asynchronous reset.
xg_tx_gclk_o	—	Out	1	TX clock (195.3125 MHz) forwarded to global clock distribution.
xg_rx_gclk_o	—	Out	1	RX clock (195.3125 MHz) forwarded to global clock distribution.
txmac_clk_i	-	In	1	195.3125 MHz clock for MAC interface. It is recommended to use xg_tx_gclk_o[0] as input.
rxmac_clk_i	-	In	1	195.3125 MHz clock for MAC interface. It is recommended to use xg_tx_gclk_o[0] as input.
sysbus_clk_i	—	In	1	Clock for the Management module (AXI4-Lite interface). It is recommended to use between 20 MHz to 195.3125 MHz clock for 25GMII interface.
Serial I/O				
pad_refclkn_i	—	In	1	156.25 MHz SERDES PLL reference clock signal (-).
pad_refclkp_i	—	In	1	156.25 MHz SERDES PLL reference clock signal (+).
pad_rxn_i	pad_refclkn_i	In	1	RX- differential signal.
pad_rxp_i	pad_refclkp_i	In	1	RX+ differential signal.
pad_txn_o	pad_refclkn_i	Out	1	TX- differential signal.
pad_txp_o	pad_refclkp_i	Out	1	TX+ differential signal.
AXI4-Stream Receive Interface				
axis_rx_tdata_o	xg_tx_gclk_o	Out	128	AXI4-Stream data from PHY to the client.
axis_rx_tkeep_o	xg_tx_gclk_o	Out	16	AXI4-Stream control that indicates which data is valid on axis_rx_tdata_o[ ]. • axis_rx_tkeep_o[0]: axis_rx_tdata_o [7:0] • axis_rx_tkeep_o[1]: axis_rx_tdata_o [15:8] • axis_rx_tkeep_o[2]: axis_rx_tdata_o [23:16] • axis_rx_tkeep_o[3]: axis_rx_tdata_o [31:24] • axis_rx_tkeep_o[4]: axis_rx_tdata_o [39:32] • axis_rx_tkeep_o[5]: axis_rx_tdata_o [47:40] • axis_rx_tkeep_o[6]: axis_rx_tdata_o [55:48] • axis_rx_tkeep_o[7]: axis_rx_tdata_o [63:56] • axis_rx_tkeep_o[8]: axis_rx_tdata_o [71:64] • axis_rx_tkeep_o[9]: axis_rx_tdata_o [79:72] • axis_rx_tkeep_o[10]: axis_rx_tdata_o [87:80] • axis_rx_tkeep_o[11]: axis_rx_tdata_o [95:88] • axis_rx_tkeep_o[12]: axis_rx_tdata_o [103:96] • axis_rx_tkeep_o[13]: axis_rx_tdata_o [111:104] • axis_rx_tkeep_o[14]: axis_rx_tdata_o [119:112] • axis_rx_tkeep_o[15]: axis_rx_tdata_o [127:120]
axis_rx_tvalid_o	xg_tx_gclk_o	Out	1	AXI4-Stream data valid.
axis_rx_tuser_o	xg_tx_gclk_o	Out	1	AXI4-Stream user signal used to indicate that the frame had a length error, termination error, or a cyclic redundancy check (CRC) error. This signal is qualified with the axis_rx_tlast_o signal.
axis_rx_tlast_o	xg_tx_gclk_o	Out	1	AXI4-Stream signal indicating the end of a packet.
AXI4-Stream Transmit Interface				
axis_tx_tready_o	xg_tx_gclk_o	Out	1	AXI4-Stream signal indicating that the core can accept data transfer.

Port Name	Clock Domain	I/O	Width	Description
axis_tx_tdata_i	xg_tx_gclk_o	In	128	AXI4-Stream data from the client
axis_tx_tkeep_i	xg_tx_gclk_o	In	16	AXI4-Stream control that indicates which data is valid on axis_tx_tdata_i [ ]. • axis_tx_tkeep_i[0]: axis_tx_tdata_i[7:0] • axis_tx_tkeep_i[1]: axis_tx_tdata_i[15:8] • axis_tx_tkeep_i[2]: axis_tx_tdata_i[23:16] • axis_tx_tkeep_i[3]: axis_tx_tdata_i[31:24] • axis_tx_tkeep_i[4]: axis_tx_tdata_i[39:32] • axis_tx_tkeep_i[5]: axis_tx_tdata_i[47:40] • axis_tx_tkeep_i[6]: axis_tx_tdata_i[55:48] • axis_tx_tkeep_i[7]: axis_tx_tdata_i[63:56] • axis_tx_tkeep_i[8]: axis_tx_tdata_i[71:64] • axis_tx_tkeep_i[9]: axis_tx_tdata_i[79:72] • axis_tx_tkeep_i[10]: axis_tx_tdata_i[87:80] • axis_tx_tkeep_i[11]: axis_tx_tdata_i[95:88] • axis_tx_tkeep_i[12]: axis_tx_tdata_i[103:96] • axis_tx_tkeep_i[13]: axis_tx_tdata_i[111:104] • axis_tx_tkeep_i[14]: axis_tx_tdata_i[119:112] • axis_tx_tkeep_i[15]: axis_tx_tdata_i[127:120]
axis_tx_tvalid_i	xg_tx_gclk_o	In	1	AXI4-Stream data valid.
axis_tx_tuser_i	xg_tx_gclk_o	In	1	AXI4-Stream user signal to indicate an error in the frame. This signal is qualified with the axis_tx_tlast_i signal.
axis_tx_tlast_i	xg_tx_gclk_o	In	1	AXI4-Stream signal indicating the end of a packet.
Non-Standard RX/TX Signals				
xg_pcsrdy_o	xg_tx_gclk_o	Out	1	The high level of this signal indicates alignment is achieved and RX PCS is ready for transaction.
phy_init_done_o	xg_tx_gclk_o	Out	1	The high level of this signal indicates alignment is PHY initialization or reset is done.
AXI4-Lite Interface				
axi4l_awaddr_i	sysbus_clk_i	In	32	AXI4-Lite write address bus.
axi4l_awvalid_i	sysbus_clk_i	In	1	AXI4-Lite write address valid.
axi4l_awready_o	sysbus_clk_i	Out	1	AXI4-Lite write address acknowledge.
axi4l_wdata_i	sysbus_clk_i	In	32	AXI4-Lite write data bus.
axi4l_wvalid_i	sysbus_clk_i	In	1	AXI4-Lite write data valid.
axi4l_wready_o	sysbus_clk_i	Out	1	AXI4-Lite write data acknowledge.
axi4l_wstrb_i	sysbus_clk_i	In	4	AXI4-Lite write strobe. This feature is not supported so it is tied to 0.
axi4l_bresp_o	sysbus_clk_i	Out	2	AXI4-Lite write transaction response.
axi4l_bvalid_o	sysbus_clk_i	Out	1	AXI4-Lite write response valid.
axi4l_bready_i	sysbus_clk_i	In	1	AXI4-Lite write response acknowledge.
axi4l_araddr_i	sysbus_clk_i	In	32	AXI4-Lite read address bus.
axi4l_arvalid_i	sysbus_clk_i	In	1	AXI4-Lite read address valid.
axi4l_arready_o	sysbus_clk_i	Out	1	AXI4-Lite read address acknowledge.
axi4l_rdata_o	sysbus_clk_i	Out	32	AXI4-Lite read data output.
axi4l_rresp_o	sysbus_clk_i	Out	2	AXI4-Lite read data response.
axi4l_rvalid_o	sysbus_clk_i	Out	1	AXI4-Lite read data/response valid.
axi4l_rready_i	sysbus_clk_i	In	1	AXI4-Lite read data acknowledge.
Priority Flow Control Interface				
tx_pfc_tuser_i	xg_tx_gclk_o	In	8	Priority FIFO (RX) asserts this input to request for pause, according to the defined priority ‘n’ where n = 0 to 7. Upon the assertion of this input, a PFC frame will be constructed and transmitted at the next

Port Name	Clock Domain	I/O	Width	Description
				available transmission slot. To ensure the successful transmission of an XOFF frame, this input can only be asserted at least one clock cycle after tx_pfc_en and tx_pfc_p[0-7]_en bits of TX_CTL register have been set. This input should be tied to '0' when left unused. For example, when using CSR-based PFC triggers through TX_PFC_CTL registers.
rx_pfc_tuser_o	xg_tx_gclk_o	Out	8	Denotes the pause request to priority FIFO (TX), according to the defined priority 'n' where n = 0 to 7. • Asserted when priority 'n' is enabled and its pause quanta > 0 (to pause TX). • De-asserted when priority 'n' is disabled or its pause quanta = 0 (to resume TX).
rx_pfc_tready_i	xg_tx_gclk_o	In	8	Priority FIFO (TX) asserts this input to acknowledge pause request based on the respective priority 'n', where n = 0 to 7. Hence, the quanta counter will only start to expire when this input is asserted. This input should be tied to '1' if it is left unused.
Statistics Vector Interface				
tx_statvec_o	xg_tx_gclk_o	Out	29	Contains information on the frame transmitted. This bus is qualified by the tx_staten_o signal. • tx_statvec_o[13:0]: Frame byte count • tx_statvec_o[14]: Transmit is OK • tx_statvec_o[15]: MAC control inserted by MAC • tx_statvec_o[16]: MAC control inserted by client • tx_statvec_o[17]: Jumbo frame • tx_statvec_o[18]: Tagged frame • tx_statvec_o[19]: Broadcast address • tx_statvec_o[20]: Multicast address • tx_statvec_o[21]: Underrun error • tx_statvec_o[22]: CRC error • tx_statvec_o[23]: Length check error • tx_statvec_o[24]: Terminate error • tx_statvec_o[25]: Long frame error • tx_statvec_o[26]: PTP1588 frame transmitted • tx_statvec_o[27]: PFC frame transmitted • tx_statvec_o[28]: Stacked VLAN frame transmitted
tx_staten_o	xg_tx_gclk_o	Out	1	When asserted, it indicates that the contents of the tx_statvec_o bus are valid. This signal is asserted for three tx_mac_clk_i periods.
rx_statvec_o	xg_tx_gclk_o	Out	29	Contains information on the frame received. This bus is qualified by the rx_staten_o signal. • rx_statvec_o[13:0]: Frame byte count • rx_statvec_o[14]: Frame dropped • rx_statvec_o[15]: Broadcast frame received • rx_statvec_o[16]: Multicast frame received • rx_statvec_o[17]: CRC error • rx_statvec_o[18]: VLAN tag detected • rx_statvec_o[19]: Pause frame • rx_statvec_o[20]: Length check error • rx_statvec_o[21]: Frame is too long • rx_statvec_o[22]: MAC address mismatch • rx_statvec_o[23]: Unsupported opcode. Only the opcodes for pause frame and PFC frame are supported. • rx_statvec_o[24]: Minimum IPG violated • rx_statvec_o[25]: Receive packet ignored

Port Name	Clock Domain	I/O	Width	Description
				• rx_statvec_o[26]: PTP1588 frame received • rx_statvec_o[27]: PFC frame received • tx_statvec_o[28]: Stacked VLAN frame transmitted
rx_staten_o	xg_tx_gclk_o	Out	1	When asserted, indicates that the contents of the rx_statvec_o bus are valid. This signal is asserted for three rx_mac_clk_i periods.
TX Pause Frame[1]				
tx_pausreq_i	xg_tx_gclk_o	In	1	Transmit pause frame. 1 = send request, 0 = do not send request. This is a positive edge-triggered bit. The tx_fc_en bit of TX_CTL register should be set to 1 to enable this feature. When this port is enabled, the tx_pausreq bit of MAC_CTL register is not accessible. You must wait for TX_STAT_PAUSE increment before issuing the next pause req. Otherwise, it may result in indeterminate behavior. This signal must be asserted the earliest 1 clock cycle after the tx_paustm_i values are updated.
tx_paustm_i	xg_tx_gclk_o	In	16	Pause Quanta time. This is equivalent to the tx_paustim bit of PAUSE_TM register. These signals must be ready for 1 cycle before the tx_pausereq_i is asserted.

Note:

2. Available when TX Pause Frame Generation via Ports is enabled.

4.2. MAC Only

Table 4.2. Signal Description for MAC Only

Port Name	Clock Domain	I/O	Width	Description
Clock and Reset				
reset_n_i	Asynchronous	In	1	Active-low asynchronous reset.
rxmac_clk_i	—	In	1	Clock for Receive AXI4-Stream and 25GMII datapath. 195.3125 MHz clock for 25GMII interface. It is recommended to use xg_tx_gclk_o (clock output from PHY).
txmac_clk_i	—	In	1	Clock for Transmit AXI4-Stream and 25MII datapath. 195.3125 MHz clock for 25GMII interface. It is recommended to use xg_tx_gclk_o (clock output from PHY).
sysbus_clk_i	—	In	1	Clock for the Management module (AXI4-Lite interface). It is recommended to use between 20 MHz to 195.3125 MHz clock for 25GMII interface.
25MII Interface				
xgmii_rxd_i	xg_tx_gclk_o	In	128	16-byte Receive SDR 25GMII data from PHY. • Byte 0: xgmii_rxd_i[7:0] • Byte 1: xgmii_rxd_i[15:8] • Byte 2: xgmii_rxd_i[23:16] • Byte 3: xgmii_rxd_i[31:24] • Byte 4: xgmii_rxd_i[39:32] • Byte 5: xgmii_rxd_i[47:40] • Byte 6: xgmii_rxd_i[55:48] • Byte 7: xgmii_rxd_i[63:56] • Byte 8: xgmii_rxd_i[71:64] • Byte 9: xgmii_rxd_i[79:72] • Byte 10: xgmii_rxd_i[87:80]

Port Name	Clock Domain	I/O	Width	Description
				• Byte 11: xgmii_rxd_i[95:88] • Byte 12: xgmii_rxd_i[103:96] • Byte 13: xgmii_rxd_i[111:104] • Byte 14: xgmii_rxd_i[119:112] • Byte 15: xgmii_rxd_i[127:120]
xgmii_rxc_i	xg_tx_gclk_o	In	16	Control bits for each lane in xgmii_rxd_i[ ]. • Byte 0: xgmii_rxc_i[0] • Byte 1: xgmii_rxc_i[1] • Byte 2: xgmii_rxc_i[2] • Byte 3: xgmii_rxc_i[3] • Byte 4: xgmii_rxc_i[4] • Byte 5: xgmii_rxc_i[5] • Byte 6: xgmii_rxc_i[6] • Byte 7: xgmii_rxc_i[7] • Byte 8: xgmii_rxc_i[8] • Byte 9: xgmii_rxc_i[9] • Byte 10: xgmii_rxc_i[10] • Byte 11: xgmii_rxc_i[11] • Byte 12: xgmii_rxc_i[12] • Byte 13: xgmii_rxc_i[13] • Byte 14: xgmii_rxc_i[14] • Byte 15: xgmii_rxc_i[15]
xgmii_txd_o	xg_tx_gclk_o	Out	128	16-byte Transmit SDR 25GMII data to PHY. • Byte 0: xgmii_txd_o[7:0] • Byte 1: xgmii_txd_o[15:8] • Byte 2: xgmii_txd_o[23:16] • Byte 3: xgmii_txd_o[31:24] • Byte 4: xgmii_txd_o[39:32] • Byte 5: xgmii_txd_o[47:40] • Byte 6: xgmii_txd_o[55:48] • Byte 7: xgmii_txd_o[63:56] • Byte 8: xgmii_txd_o[71:64] • Byte 9: xgmii_txd_o[79:72] • Byte 10: xgmii_txd_o[87:80] • Byte 11: xgmii_txd_o[95:88] • Byte 12: xgmii_txd_o[103:96] • Byte 13: xgmii_txd_o[111:104] • Byte 14: xgmii_txd_o[119:112] • Byte 15: xgmii_txd_o[127:120]

Port Name	Clock Domain	I/O	Width	Description
xgmii_txc_o	xg_tx_gclk_o	Out	16	Control bits for each lane in xgmii_txd_o[ ]. • Byte 0: xgmii_txc_o[0] • Byte 1: xgmii_txc_o[1] • Byte 2: xgmii_txc_o[2] • Byte 3: xgmii_txc_o[3] • Byte 4: xgmii_txc_o[4] • Byte 5: xgmii_txc_o[5] • Byte 6: xgmii_txc_o[6] • Byte 7: xgmii_txc_o[7] • Byte 8: xgmii_txc_o[8] • Byte 9: xgmii_txc_o[9] • Byte 10: xgmii_txc_o[10] • Byte 11: xgmii_txc_o[11] • Byte 12: xgmii_txc_o[12] • Byte 13: xgmii_txc_o[13] • Byte 14: xgmii_txc_o[14] • Byte 15: xgmii_txc_o[15]
AXI4-Stream Receive Interface				
axis_rx_tdata_o	xg_tx_gclk_o	Out	128	AXI4-Stream data from PHY to the client.
axis_rx_tkeep_o	xg_tx_gclk_o	Out	16	AXI4-Stream control that indicates which data is valid on axis_rx_tdata_o[ ]. • axis_rx_tkeep_o[0]: axis_rx_tdata_o [7:0] • axis_rx_tkeep_o[1]: axis_rx_tdata_o [15:8] • axis_rx_tkeep_o[2]: axis_rx_tdata_o [23:16] • axis_rx_tkeep_o[3]: axis_rx_tdata_o [31:24] • axis_rx_tkeep_o[4]: axis_rx_tdata_o [39:32] • axis_rx_tkeep_o[5]: axis_rx_tdata_o [47:40] • axis_rx_tkeep_o[6]: axis_rx_tdata_o [55:48] • axis_rx_tkeep_o[7]: axis_rx_tdata_o [63:56] • axis_rx_tkeep_o[8]: axis_rx_tdata_o [71:64] • axis_rx_tkeep_o[9]: axis_rx_tdata_o [79:72] • axis_rx_tkeep_o[10]: axis_rx_tdata_o [87:80] • axis_rx_tkeep_o[11]: axis_rx_tdata_o [95:88] • axis_rx_tkeep_o[12]: axis_rx_tdata_o [103:96] • axis_rx_tkeep_o[13]: axis_rx_tdata_o [111:104] • axis_rx_tkeep_o[14]: axis_rx_tdata_o [119:112] axis_rx_tkeep_o[15]: axis_rx_tdata_o [127:120]
axis_rx_tvalid_o	xg_tx_gclk_o	Out	1	AXI4-Stream data valid.
axis_rx_tuser_o	xg_tx_gclk_o	Out	1	AXI4-Stream user signal used to indicate that the frame had a length error, termination error, or a CRC error. This signal is qualified with the axis_rx_tlast_o signal.
axis_rx_tlast_o	xg_tx_gclk_o	Out	1	AXI4-Stream signal indicating the end of a packet.
AXI4-Stream Transmit Interface				
axis_tx_tready_o	xg_tx_gclk_o	Out	1	AXI4-Stream signal indicating that the core can accept data transfer.
axis_tx_tdata_i	xg_tx_gclk_o	In	128	AXI4-Stream data from the client.

Port Name	Clock Domain	I/O	Width	Description
axis_tx_tkeep_i	xg_tx_gclk_o	In	16	AXI4-Stream control that indicates which data is valid on axis_tx_tdata_i [ ]. • axis_tx_tkeep_i[0]: axis_tx_tdata_i[7:0] • axis_tx_tkeep_i[1]: axis_tx_tdata_i[15:8] • axis_tx_tkeep_i[2]: axis_tx_tdata_i[23:16] • axis_tx_tkeep_i[3]: axis_tx_tdata_i[31:24] • axis_tx_tkeep_i[4]: axis_tx_tdata_i[39:32] • axis_tx_tkeep_i[5]: axis_tx_tdata_i[47:40] • axis_tx_tkeep_i[6]: axis_tx_tdata_i[55:48] • axis_tx_tkeep_i[7]: axis_tx_tdata_i[63:56] • axis_tx_tkeep_i[8]: axis_tx_tdata_i[71:64] • axis_tx_tkeep_i[9]: axis_tx_tdata_i[79:72] • axis_tx_tkeep_i[10]: axis_tx_tdata_i[87:80] • axis_tx_tkeep_i[11]: axis_tx_tdata_i[95:88] • axis_tx_tkeep_i[12]: axis_tx_tdata_i[103:96] • axis_tx_tkeep_i[13]: axis_tx_tdata_i[111:104] • axis_tx_tkeep_i[14]: axis_tx_tdata_i[119:112] axis_tx_tkeep_i[15]: axis_tx_tdata_i[127:120]
axis_tx_tvalid_i	xg_tx_gclk_o	In	1	AXI4-Stream data valid.
axis_tx_tuser_i	xg_tx_gclk_o	In	1	AXI4-Stream user signal used to indicate an error in the frame. This signal is qualified with the axis_tx_tlast_i signal.
axis_tx_tlast_i	xg_tx_gclk_o	In	1	AXI4-Stream signal indicating the end of a packet.
AXI4-Lite Interface				
axi4l_awaddr_i	sysbus_clk_i	In	32	AXI4-Lite Write address bus.
axi4l_awvalid_i	sysbus_clk_i	In	1	AXI4-Lite Write address valid.
axi4l_awready_o	sysbus_clk_i	Out	1	AXI4-Lite Write address acknowledge.
axi4l_wdata_i	sysbus_clk_i	In	32	AXI4-Lite Write data bus.
axi4l_wvalid_i	sysbus_clk_i	In	1	AXI4-Lite Write data valid.
axi4l_wready_o	sysbus_clk_i	Out	1	AXI4-Lite Write data acknowledge.
axi4l_wstrb_i	sysbus_clk_i	In	4	AXI4-Lite write strobe. This feature is not supported so it is tied to 0.
axi4l_bresp_o	sysbus_clk_i	Out	2	AXI4-Lite Write transaction response.
axi4l_bvalid_o	sysbus_clk_i	Out	1	AXI4-Lite Write response valid.
axi4l_bready_i	sysbus_clk_i	In	1	AXI4-Lite Write response acknowledge.
axi4l_araddr_i	sysbus_clk_i	In	32	AXI4-Lite Read address bus.
axi4l_arvalid_i	sysbus_clk_i	In	1	AXI4-Lite Read address valid.
axi4l_arready_o	sysbus_clk_i	Out	1	AXI4-Lite Read address acknowledge.
axi4l_rdata_o	sysbus_clk_i	Out	32	AXI4-Lite Read data output.
axi4l_rresp_o	sysbus_clk_i	Out	2	AXI4-Lite Read data response.
axi4l_rvalid_o	sysbus_clk_i	Out	1	AXI4-Lite Read data/response valid.
axi4l_rready_i	sysbus_clk_i	In	1	AXI4-Lite Read data acknowledge.
Priority Flow Control Interface				
tx_pfc_tuser_i	xg_tx_gclk_o	In	8	Priority FIFO (RX) asserts this input to request for pause, according to the defined priority ‘n’ where n = 0 to 7. Upon the assertion of this input, a PFC frame will be constructed and transmitted at the next available transmission slot. To ensure the successful transmission of an XOFF frame, this input can only be asserted at least one clock cycle after tx_pfc_en and tx_pfc_p[0-7]_en bits of TX_CTL register have been set. This input should be tied to ‘0’ when left unused. For example, when using CSR-based PFC triggers through TX_PFC_CTL registers.

Port Name	Clock Domain	I/O	Width	Description
rx_pfc_tuser_o	xg_tx_gclk_o	In	8	Denotes the pause request to priority FIFO (TX), according to the defined priority 'n' where n = 0 to 7. • Asserted when priority 'n' is enabled and its pause quanta > 0 (to pause TX). De-asserted when priority 'n' is disabled or its pause quanta = 0 (to resume TX).
rx_pfc_tready_i	xg_tx_gclk_o	In	1	Priority FIFO (TX) asserts this input to acknowledge pause request based on the respective priority 'n', where n = 0 to 7. Hence, the quanta counter will only start to expire when this input is asserted. This input should be tied to '1' if it is left unused.
Statistics Vector Interface				
tx_statvec_o	xg_tx_gclk_o	Out	29	Contains information on the frame transmitted. This bus is qualified by the tx_staten_o signal. • tx_statvec_o[13:0]: Frame byte count • tx_statvec_o[14]: Transmit is OK • tx_statvec_o[15]: MAC control inserted by MAC • tx_statvec_o[16]: MAC control inserted by client • tx_statvec_o[17]: Jumbo frame • tx_statvec_o[18]: Tagged frame • tx_statvec_o[19]: Broadcast address • tx_statvec_o[20]: Multicast address • tx_statvec_o[21]: Underrun error • tx_statvec_o[22]: CRC error • tx_statvec_o[23]: Length check error • tx_statvec_o[24]: Terminate error • tx_statvec_o[25]: Long frame error • tx_statvec_o[26]: PTP1588 frame transmitted • tx_statvec_o[27]: PFC frame transmitted • tx_statvec_o[28]: Stacked VLAN frame transmitted
tx_staten_o	xg_tx_gclk_o	Out	1	When asserted, it indicates that the contents of the tx_statvec_o bus are valid. This signal is asserted for three tx_mac_clk_i periods.
rx_statvec_o	xg_tx_gclk_o	Out	29	Contains information on the frame received. This bus is qualified by the rx_staten_o signal. • rx_statvec_o[13:0]: Frame byte count • rx_statvec_o[14]: Frame dropped • rx_statvec_o[15]: Broadcast frame received • rx_statvec_o[16]: Multicast frame received • rx_statvec_o[17]: CRC error • rx_statvec_o[18]: VLAN tag detected • rx_statvec_o[19]: Pause frame • rx_statvec_o[20]: Length check error • rx_statvec_o[21]: Frame is too long • rx_statvec_o[22]: MAC address mismatch • rx_statvec_o[23]: Unsupported opcode. Only the opcodes for pause frame and PFC frame are supported. • rx_statvec_o[24]: Minimum IPG violated • rx_statvec_o[25]: Receive packet ignored • rx_statvec_o[26]: PTP1588 frame received • rx_statvec_o[27]: PFC frame received • rx_statvec_o[28]: Stacked VLAN frame received

Port Name	Clock Domain	I/O	Width	Description
rx_staten_o	xg_tx_gclk_o	Out	1	When asserted, indicates that the contents of the rx_statvec_o bus are valid. This signal is asserted for three rx_mac_clk_i periods.
LINTR				
int_o	sysbus_clk_i	Out	1	Interrupt request. For more information on the assertion of this signal, refer to the Interrupt Registers section.
TX Pause Frame[1]				
tx_pausreq_i	xg_tx_gclk_o	In	1	Transmit PAUSE Frame. 1 = send request, 0 = do not send request. This is a positive edge-triggered bit. The tx_fc_en bit of TX_CTL register should be set to 1 to enable this feature. When this port is enabled, the tx_pausreq bit of MAC_CTL register is not accessible. You must wait for TX_STAT_PAUSE increment before issuing the next pause req. Otherwise, it may result in indeterminate behavior. This signal must be asserted the earliest 1 clock cycle after the tx_paustm_i values are updated.
tx_paustm_i	xg_tx_gclk_o	In	16	Pause Quanta time. This is equivalent to the tx_paustim bit of PAUSE_TM register. These signals must be ready for 1 cycle before the tx_pausereq_i is asserted.

Note:

1. Available when TX Pause Frame Generation via Ports is enabled.

4.3. PHY Only

Table 4.3. Signal Description for PHY Only

Port Name	Clock Domain	I/O	Width	Description
Serial I/O				
pad_refclkn_i	—	In	1	156.25 MHz SERDES PLL reference clock signal (-).
pad_refclkp_i	—	In	1	156.25 MHz SERDES PLL reference clock signal (+).
pad_rxn_i	pad_refclkn_i	In	1	RX- differential signal.
pad_rxp_i	pad_refclkp_i	In	1	RX+ differential signal.
pad_txn_o	pad_refclkn_i	Out	1	TX- differential signal.
pad_txp_o	pad_refclkp_i	Out	1	TX+ differential signal.
Clock and Reset				
xg_tx_clk_i	—	In	1	TX clock input – 195.3125 MHz clock for transmit datapath. It is recommended to use xg_tx_gclk_o.
xg_rx_clk_i	—	In	1	RX clock input – 195.3125 MHz clock for receive datapath. It is recommended to use xg_tx_gclk_o.
xg_tx_clk_o	—	Out	1	TX clock output.
xg_rx_clk_o	—	Out	1	RX clock output.
xg_tx_gclk_o	—	Out	2	TX clock forwarded to global clock distribution (156.25 MHz).
xg_rx_gclk_o	—	Out	1	RX clock forwarded to global clock distribution (156.25 MHz).
reset_n_i	Asynchronous	In	1	Active-low asynchronous reset.
sysbus_clk_i	—	In	1	Clock for the Management module.
25GMII				
xg_txc_i [15:0]	xg_tx_gclk_o	In	16	16-bit TX control signal. bit[0] is the control signal for xg_txd_i [7:0] bit[1] is the control signal for xg_txd_i [15:8] bit[2] is the control signal for xg_txd_i [23:16]

Port Name	Clock Domain	I/O	Width	Description
				bit[3] is the control signal for xg_txd_i [31:24] bit[4] is the control signal for xg_txd_i [39:32] bit[5] is the control signal for xg_txd_i [47:40] bit[6] is the control signal for xg_txd_i [55:48] bit[7] is the control signal for xg_txd_i [63:56] bit[8] is the control signal for xg_txd_i [71:64] bit[9] is the control signal for xg_txd_i [79:72] bit[10] is the control signal for xg_txd_i [87:80] bit[11] is the control signal for xg_txd_i [95:88] bit[12] is the control signal for xg_txd_i [103:96] bit[13] is the control signal for xg_txd_i [111:104] bit[14] is the control signal for xg_txd_i [119:112] bit[15] is the control signal for xg_txd_i [127:120] When control bit is high, indicates a control byte, otherwise it is a data byte.
xg_txd_i [127:0]	xg_tx_gclk_o	In	128	128-bit TX data signal.
xg_rxc_o [15:0]	xg_tx_gclk_o	Out	16	16-bit RX control signal. bit[0] is the control signal for xg_rxd_o [7:0] bit[1] is the control signal for xg_rxd_o [15:8] bit[2] is the control signal for xg_rxd_o [23:16] bit[3] is the control signal for xg_rxd_o [31:24] bit[4] is the control signal for xg_rxd_o [39:32] bit[5] is the control signal for xg_rxd_o [47:40] bit[6] is the control signal for xg_rxd_o [55:48] bit[7] is the control signal for xg_rxd_o [63:56] bit[8] is the control signal for xg_rxd_o [71:64] bit[9] is the control signal for xg_rxd_o [79:72] bit[10] is the control signal for xg_rxd_o [87:80] bit[11] is the control signal for xg_rxd_o [95:88] bit[12] is the control signal for xg_rxd_o [103:96] bit[13] is the control signal for xg_rxd_o [111:104] bit[14] is the control signal for xg_rxd_o [119:112] bit[15] is the control signal for xg_rxd_o [127:120] When control bit is high, indicates a control byte, otherwise it is a data byte.
xg_rxd_o [127:0]	xg_tx_gclk_o	Out	128	128-bit RX data signal.
Non-Standard RX/TX Signals				
xg_rxval_o	xg_tx_gclk_o	Out	1	RX valid signal. When high, indicates that the corresponding values on signals. xg_rxc_o and xg_rxd_o are valid.
xg_txval_i	xg_tx_gclk_o	In	1	TX valid signal. When high, indicates that the corresponding values on signals xg_txc_i and xg_txd_i are valid.
xg_txrdy_o	xg_tx_gclk_o	Out	1	TX ready signal. When high, indicates that TX PCS is ready to accept the user data and control signal (xg_txc_i and xg_txd_i).
xg_rxrdy_i	xg_tx_gclk_o	In	1	RX ready signal. This is a flow-control feedback signal.
xg_pcsrdy_o	xg_tx_gclk_o	Out	1	The high level of this signal indicates alignment is achieved and RX PCS is ready for transaction.
phy_init_done_o	xg_tx_gclk_o	Out	1	The high level of this signal indicates alignment is PHY initialization or reset is done.
tx0_am_i	xg_tx_gclk_o	In	4	TX alignment marker needed for latency measurement when RSFEC is enabled. When RSFEC is enabled, this input must be asserted for 2 cycles in every 160 periodic cycles of clock signal xg_tx_out_gclk_i. Both xg_txc_i and xg_txd_i must be held for 2 cycles to sync with alignment marker assertion.

Port Name	Clock Domain	I/O	Width	Description
				When RSFEC is disabled, you must provide 4'b1111 as input.
rx0_am_o	xg_tx_gclk_o	Out	4	RX alignment marker needed for latency measurement when RSFEC is enabled. When RSFEC is enabled, both xg_rxc_o and xg_rxd_o must not be consumed when this output is asserted. When RSFEC is disabled, this output is unused.
AXI4-Lite Interface				
axi4l_awaddr_i	sysbus_clk_i	In	32	AXI4-Lite write address bus.
axi4l_awvalid_i	sysbus_clk_i	In	1	AXI4-Lite write address valid.
axi4l_awready_o	sysbus_clk_i	Out	1	AXI4-Lite write address acknowledge.
axi4l_wdata_i	sysbus_clk_i	In	32	AXI4-Lite write data bus.
axi4l_wvalid_i	sysbus_clk_i	In	1	AXI4-Lite write data valid.
axi4l_wready_o	sysbus_clk_i	Out	1	AXI4-Lite write data acknowledge.
axi4l_wstrb_i	sysbus_clk_i	In	4	AXI4-Lite write strobe. This feature is not supported so tied to 0.
axi4l_bresp_o	sysbus_clk_i	Out	2	AXI4-Lite write transaction response.
axi4l_bvalid_o	sysbus_clk_i	Out	1	AXI4-Lite write response valid.
axi4l_bready_i	sysbus_clk_i	In	1	AXI4-Lite write response acknowledge.
axi4l_araddr_i	sysbus_clk_i	In	32	AXI4-Lite read address bus.
axi4l_arvalid_i	sysbus_clk_i	In	1	AXI4-Lite read address valid.
axi4l_arready_o	sysbus_clk_i	Out	1	AXI4-Lite read address acknowledge.
axi4l_rdata_o	sysbus_clk_i	Out	32	AXI4-Lite read data output.
axi4l_rresp_o	sysbus_clk_i	Out	2	AXI4-Lite read data response.
axi4l_rvalid_o	sysbus_clk_i	Out	1	AXI4-Lite read data/response valid.
axi4l_rready_i	sysbus_clk_i	In	1	AXI4-Lite read data acknowledge.

5. Register Description

5.1. MAC + PHY Registers

The registers available in the Ethernet MAC + PHY option include all registers from the MAC and PHY.

For register descriptions in the 25GbE MAC and PHY, refer to the MAC Registers and PHY Registers sections.

Note: With the MAC + PHY selected as the IP option, only the AXI4-Lite interface is available for access to both PCS and MAC registers.

5.2. MAC Registers

The following table lists the access type of each register.

Table 5.1. Access Types

Access Type	Behavior on Read Access	Behavior on Write Access
RO	Returns register value	Ignores write access.
WO	Returns 0	Updates register value.
RW	Returns register value	Updates register value.
RW1C	Returns register value	Writing 1'b1 on register bit clears the bit to 1'b0. Writing 1'b0 on register bit is ignored.
Reserved	Returns 0	Ignores write access.

5.2.1. Configuration Registers for MAC

The following table lists the 25GbE MAC configuration registers. These registers are accessed through the AXI4-Lite interface.

Table 5.2. Configuration Registers for MAC

Offset	Register Name	Access	Description
0x000	MODE	RW	Mode of Operation Register
0x004	TX_CTL	RW	Transmit MAC Control Register
0x008	RX_CTL	RW	Receive MAC Control Register
0x00C	MAX_PKT_LNGTH	RW	Maximum Packet Size Register
0x010	IPG_VAL	RW	IPG Value Register
0x014	MAC_ADDR_0	RW	MAC Address Register Word 0
0x018	MAC_ADDR_1	RW	MAC Address Register Word 1
0x01C	TX_RX_STS	RO	Transmit and Receive Status Register
0x020	VLAN_TAG	RO	VLAN Tag Register
0x024	MC_TABLE_0	RW	Multicast Tables Register Word 0
0x028	MC_TABLE_1	RW	Multicast Tables Register Word 1
0x02C	PAUSE_OPCODE	RW	Pause Opcode
0x030	MAC_CTL	RW	MAC Control Register
0x034	PAUSE_TM	RW	Pause Time Register

5.2.1.1. MODE Register

This register enables the operation of the MAC. It can be written at any time.

Table 5.3. MODE Register

Bit	Label	Description	Default
[31:2]	Reserved	Reserved bits.	0
[1]	tx_en	TX MAC Enable When set to 1, the TX MAC is enabled to transmit frames.	0
[0]	rx_en	RX MAC Enable When set to 1, the RX MAC is enabled to receive frames.	0

5.2.1.2. TX_CTL Register

This register is overwritten only when the TX MAC is disabled. Writing this register while TX MAC is active results in unpredictable behavior.

Table 5.4. TX_CTL Register

Bit	Label	Description	Default
[31:14]	Reserved	Reserved bits.	0
[13]	tx_pfc_p7_en	TX Priority Flow Control Enable (Priority 7). When set to 1 with tx_pfc_en set, the assertion of tx_pfc_tuser_i[7] input port or TX_PFC_CTL[7] bit will trigger the transmission of a PFC frame. When set to 0, the assertion of tx_pfc_tuser_i[7] input or TX_PFC_CTL[7] bit is ignored.	0
[12]	tx_pfc_p6_en	TX Priority Flow Control Enable (Priority 6). When set to 1 with tx_pfc_en set, the assertion of tx_pfc_tuser_i[6] input port or TX_PFC_CTL[6] bit trigger the transmission of a PFC frame. When set to 0, the assertion of tx_pfc_tuser_i[6] input or TX_PFC_CTL[6] bit is ignored.	0
[11]	tx_pfc_p5_en	TX Priority Flow Control Enable (Priority 5). When set to 1 with tx_pfc_en set, the assertion of tx_pfc_tuser_i[5] input port or TX_PFC_CTL[5] bit will trigger the transmission of a PFC frame. When set to 0, the assertion of tx_pfc_tuser_i[5] input or TX_PFC_CTL[5] bit is ignored.	0
[10]	tx_pfc_p4_en	TX Priority Flow Control Enable (Priority 4). When set to 1 with tx_pfc_en set, the assertion of tx_pfc_tuser_i[4] input port or TX_PFC_CTL[4] bit will trigger the transmission of a PFC frame. When set to 0, the assertion of tx_pfc_tuser_i[4] input or TX_PFC_CTL[4] bit is ignored.	0
[9]	tx_pfc_p3_en	TX Priority Flow Control Enable (Priority 3). When set to 1 with tx_pfc_en set, the assertion of tx_pfc_tuser_i[3] input port or TX_PFC_CTL[3] bit trigger the transmission of a PFC frame. When set to 0, the assertion of tx_pfc_tuser_i[3] input or TX_PFC_CTL[3] bit is ignored.	0
[8]	tx_pfc_p2_en	TX Priority Flow Control Enable (Priority 2). When set to 1 with tx_pfc_en set, the assertion of tx_pfc_tuser_i[2] input port or TX_PFC_CTL[2] bit will trigger the transmission of a PFC frame. When set to 0, the assertion of tx_pfc_tuser_i[2] input or TX_PFC_CTL[2] bit is ignored.	0
[7]	tx_pfc_p1_en	TX Priority Flow Control Enable (Priority 1). When set to 1 with tx_pfc_en set, the assertion of tx_pfc_tuser_i[1] input port or TX_PFC_CTL[1] bit trigger the transmission of a PFC frame. When set to 0, the assertion of tx_pfc_tuser_i[1] input or TX_PFC_CTL[1] bit is ignored.	0

Bit	Label	Description	Default
[6]	tx_pfc_p0_en	TX Priority Flow Control Enable (Priority 0). When set to 1 with tx_pfc_en set, the assertion of tx_pfc_tuser_i[0] input port or TX_PFC_CTL[0] bit will trigger the transmission of a PFC frame. When set to 0, the assertion of tx_pfc_tuser_i[0] input or TX_PFC_CTL[0] bit is ignored.	0
[5]	tx_pfc_en	TX Priority Flow Control Enable. When set to 1, the assertion of any TX PFC tuser input ports or TX_PFC_CTL will trigger transmission of a PFC frame. When set to 0, TX PFC tuser signals will be ignored. This bit should not be enabled together with TX_CTL[1]. For example, pause frame feature in TX.	0
[4]	tx_pass_pream	TX Custom Preamble Mode When set to 1, the TX MAC operates in custom preamble mode where it preserves the preamble field presented on the client interface.	0
[3]	transmit_short	Transmit Short Frames When set to 1, the TX MAC transmits frames shorter than 64 bytes without adding padding bytes. When set to 0, TX MAC adds padding bytes when frames are shorter than 64 bytes before transmitted to the PHY.	0
[2]	tx_ipg_stretch	IFG Stretch Mode When set to 1, the TX MAC operates in the IFG stretch mode to match the data rates of OC-192. The IPG required to match OC-192 is added to the value specified in the IPG_VAL register.	0
[1]	tx_fc_en	Flow-control Enable When set to 1, this enables the flow control functionality of the TX MAC. This bit should be set for the TX MAC to transmit a pause frame. This bit should not be enabled together with tx_pfc_en/TX_CTL[5].	0
[0]	tx_pass_fcs	In-band FCS Enable When set to 1, the FCS field generation is disabled in the TX MAC, and the client is responsible to generate the appropriate FCS field.	0

5.2.1.3. RX_CTL Register

This register is overwritten only when the RX MAC is disabled. Writing this register while RX MAC is active results in unpredictable behavior.

Table 5.5. RX_CTL Register

Bit	Label	Description	Default
[31:17]	Reserved	Reserved bits.	0
[16]	rx_pfc_p7_en	RX Priority Flow Control Enable (Priority 7). When set to 1 with rx_pfc_en set, rx_pfc_tuser_o[7] output port is asserted according to the received PFC frame. When set to 0, rx_pfc_tuser_o[7] output port is ignored. You must avoid setting this bit to 0 after receiving a PFC frame, if this is not the intention.	0
[15]	rx_pfc_p6_en	RX Priority Flow Control Enable (Priority 6). When set to 1 with rx_pfc_en set, rx_pfc_tuser_o[6] output port is asserted according to the received PFC frame. When set to 0, rx_pfc_tuser_o[6] output port is ignored. You must avoid setting this bit to 0 after receiving a PFC frame, if this is not the intention.	0
[14]	rx_pfc_p5_en	RX Priority Flow Control Enable (Priority 5). When set to 1 with rx_pfc_en set, rx_pfc_tuser_o[5] output port is asserted according to the received PFC frame. When set to 0, rx_pfc_tuser_o[5] output port is ignored. You must avoid setting this bit to 0 after receiving a PFC frame, if this is not the intention.	0

Bit	Label	Description	Default
[13]	rx_pfc_p4_en	RX Priority Flow Control Enable (Priority 4). When set to 1 with rx_pfc_en set, rx_pfc_tuser_o[4] output port is asserted according to the received PFC frame. When set to 0, rx_pfc_tuser_o[4] output port is ignored. You must avoid setting this bit to 0 after receiving a PFC frame, if this is not the intention.	0
[12]	rx_pfc_p3_en	RX Priority Flow Control Enable (Priority 3). When set to 1 with rx_pfc_en set, rx_pfc_tuser_o[3] output port is asserted according to the received PFC frame. When set to 0, rx_pfc_tuser_o[3] output port is ignored. You must avoid setting this bit to 0 after receiving a PFC frame, if this is not the intention.	0
[11]	rx_pfc_p2_en	RX Priority Flow Control Enable (Priority 2). When set to 1 with rx_pfc_en set, rx_pfc_tuser_o[2] output port is asserted according to the received PFC frame. When set to 0, rx_pfc_tuser_o[2] output port is ignored. You must avoid setting this bit to 0 after receiving a PFC frame, if this is not the intention.	0
[10]	rx_pfc_p1_en	RX Priority Flow Control Enable (Priority 1). When set to 1 with rx_pfc_en set, rx_pfc_tuser_o[1] output port is asserted according to the received PFC frame. When set to 0, rx_pfc_tuser_o[1] output port is ignored. You must avoid setting this bit to 0 after receiving a PFC frame, if this is not the intention.	0
[9]	rx_pfc_p0_en	RX Priority Flow Control Enable (Priority 0). When set to 1 with rx_pfc_en set, rx_pfc_tuser_o[0] output port is asserted according to the received PFC frame. When set to 0, rx_pfc_tuser_o[0] output port is ignored. You must avoid setting this bit to 0 after receiving a PFC frame, if this is not the intention.	0
[8]	rx_pfc_en	RX Priority Flow Control Enable. When set to 1, RX PFC tuser output ports are asserted according to the received PFC frame. When set to 0, RX PFC tuser output ports are ignored. You must avoid setting this bit to 0 after receiving a PFC frame, if this is not the intention. This bit should not be enabled together with RX_CTL[2]. For example, pause frame feature in RX.	0
[7]	rx_pass_pream	RX Custom Preamble Mode. When set to 1, the RX MAC operates in custom preamble mode where it preserves the preamble field of the received frame.	0
[6]	drop_mac_ctrl	Drop MAC Control Frames. When set to 1, all MAC control frames are not passed on to the client interface.	0
[5]	receive_short	Receive Short Frames. When set to 1, it enables the RX MAC to receive frames shorter than 64 bytes.	0
[4]	receive_bc	Receive Broadcast Frames. When set to 1, it enables RX MAC to receive broadcast frames.	0
[3]	receive_all_mc	Receive Multicast Frames. When set to 1, the multicast frames are received per the filtering rules for such frames. When set to 0, no multicast (except pause frames) frames are received.	0
[2]	rx_pause_en	Receive Pause Frames. When set to 1, the RX MAC indicates the pause frame reception to the TX MAC. Pause frames are received and transferred to the AXI4-Stream interface only when the drop_mac_ctrl bit is NOT set.	0
[1]	rx_pass_fcs	In-band FCS Passing. When set to 1, the FCS and any of the padding bytes are passed to the AXI4-Stream interface. When set to 0, the MAC strips off the FCS and any padding bytes before transferring it to the AXI4-Stream interface.	0
[0]	prms	Promiscuous Mode. When set to 1, all filtering schemes are abandoned, and the RX MAC receives	0

Bit	Label	Description	Default
		frames with any address.	

5.2.1.4. MAX_PKT_LNGTH Register

This register is overwritten only when the MAC is disabled. All frames longer than the value (number of bytes) in this register is tagged as long frames. Writing this register while the MAC is active results in unpredictable behavior.

5.2.1.5. IPG_VAL Register

This register contains the IPG value to be used by TX MAC. Back-to-back packets in the transmit buffer is sent out with the IPG setting programmed in this register with DIC.

Table 5.6. IPG_VAL Register

Bit	Label	Description	Default
[15:5]	Reserved	Reserved bits.	—
[4:0]	tx_ipg	Transmit Inter-Packet Gap Specifies the amount of inter-frame gap in increments of 4 bytes. For details, refer to the Transmit MAC section. The value of 0 of this register is prohibited.	5'h1

5.2.1.6. MAC_ADDR_0 and MAC_ADDR_1 Register

The MAC address is stored in the registers in hexadecimal form. For example, to set the MAC address to: AC-DE-48-00-00-80 would require writing 0x48_00_00_80 to address 0x014 (MAC_ADDR_0). 0xAC_DE to address 0x018 (MAC_ADDR_1).

Table 5.7. MAC_ADDR_0 Register

Bit	Label	Description	Default
[31:16]	Reserved	Reserved bits.	—
[15:0]	mac_addr_0	Last two bytes of the MAC address. Ethernet address assigned to the port that is supported by the MAC.	0

Table 5.8. MAC_ADDR_1 Register

Bit	Label	Description	Default
[31:16]	Reserved	Reserved bits.	0
[15:0]	mac_addr_1	Last two bytes of the MAC address. Ethernet address assigned to the port that is supported by the MAC.	0

5.2.1.7. TX_RX_STS Register

Table 5.9. TX_RX_STS Register

Bit	Label	Description	Default
[31:5]	Reserved	Reserved bits.	0
[4]	link_ok	Link is OK. When set to 1, this indicates that no fault symbols were received on the link.	0
[3]	remote_fault	Remote fault. When set to 1, this indicates that remote fault symbols were received on the link.	0
[2]	local_fault	Local fault. When set to 1, this indicates that local fault symbols were received on the link.	0
[1]	rx_idle	Receive MAC idle. When set to 1, this indicates that the RX MAC is inactive.	0
[0]	tx_idle	Transmit MAC idle. When set to 1, this indicates that the TX MAC is inactive.	0

5.2.1.8. VLAN_TAG Register

This register has the VLAN tag field of the most recent tagged frame that was received.

Table 5.10. VLAN_TAG Register

Bit	Label	Description	Default
[31:16]	Reserved	Reserved bits.	0
[15:0]	vlan_tag	VLAN tag ID.	0

5.2.1.9. MC_TABLE_0 and MC_TABLE_1 Register

When the core is programmed to receive multicast frames, a filtering scheme is used to decide whether the frame should be received or not. This 64-bit matrix forms the hash table that is used to filter out the incoming multicast frames. For details, refer to the Receive MAC section.

Table 5.11. MC_TABLE_0 Register

Bit	Label	Description	Default
[31:0]	mc_table_0	Multicast Table Word 0 First 4-bytes of the 64-bit hash.	0

Table 5.12. MC_TABLE_1 Register

Bit	Label	Description	Default
[31:0]	mc_table_1	Multicast Table Word 1 The last 4-bytes of the 64-bit hash.	0

5.2.1.10. PAUSE_OPCODE Register

This register contains the pause opcode. This is compared to the opcode in the received pause frame. This value is also included in any pause frame transmitted by the MAC.

Table 5.13. PAUSE_OPCODE Register

Bit	Label	Description	Default
[31:16]	Reserved	Reserved bits.	0
[15:0]	pause_opcode	Pause opcode.	16'h01

5.2.1.11. MAC_CTL Register

Table 5.14. MAC_CTL Register

Bit	Label	Description	Default
[31:5]	Reserved	Reserved bits.	—
[4]	ignore_pkt	Ignore packet. When set to 1, the RX MAC ignores or drops incoming packets.	0
[3:1]	Reserved	Reserved bits.	—
[0]	tx_pausreq	Transmit pause frame. 1 = send request, 0 = do not send request. This is a positive edge-triggered bit. The tx_fc_en bit of TX_CTL register must be set to 1 to enable this feature. Subsequent pause request will not be serviced if the pause frame of the first request is not transmitted yet. Set this register to 1 during send request, maintaining it for at least the hold time specified below before resetting it to 0. Standard maximum frame size of 1,518 bytes requires a hold time of 4,857.6 ns. Super jumbo frame size of 9,600 bytes requires a hold time of 3,072 ns. *Example of calculation for 1518 bytes: 1518 standard eth packet / 16 bytes per clock cycle = 94.875 clock cycle 94.875 clock cycle * 5.12ns per clock cycle (from 195.3125mhz) = 4857.6ns	0

5.2.1.12. PAUSE_TM Register

This register has the pause time for a flow control packet sourced by the 25 Gb MAC transmitter. To ensure the pause time is properly included in the pause frame, you must wait for at least ten clock cycles to trigger a pause frame transmission after writing pause time to this register.

Table 5.15. PAUSE_TM Register

Bit	Label	Description	Default
[31:16]	Reserved	Reserved bits.	—
[15:0]	tx_paustim	Pause duration.	0

5.2.2. Interrupt Registers

For details of these registers, refer to the *Lattice Interrupt Interface* section of the Lattice Memory Mapped Interface and Lattice Interrupt Interface User Guide (FPGA-UG-02039).

Table 5.16. Summary of Interrupt Registers

Offset	Register Name	Access	Description

0x038	INT_STATUS	RW1C	Interrupt Status Register
0x03C	INT_ENABLE	RW	Interrupt Enable Register
0x040	INT_SET	WO	Interrupt Set Register

5.2.2.1. INT_STATUS Register

Table 5.17. INT_STATUS Register

Bit	Label	Description	Default
[31:2]	Reserved	Reserved bits.	—
[1]	remote_fault_int	Remote fault symbols received.	0
[0]	local_fault_int	Local fault symbols received.	0

5.2.2.2. INT_ENABLE Register

Table 5.18. INT_ENABLE Register

Bit	Label	Description	Default
[31:2]	Reserved	Reserved bits.	—
[1]	remote_fault_en	Enables remote_fault_int.	0
[0]	local_fault_en	Enables local_fault_int.	0

5.2.2.3. INT_SET Register

Table 5.19. INT_SET Register

Bit	Label	Description	Default
[31:2]	Reserved	Reserved bits.	—
[1]	remote_fault_set	Sets remote_fault_int.	0
[0]	local_fault_set	Sets local_fault_int.	0

5.2.3. Statistics Counters

These statistic counters are wraparound counters and can only be reset when the system reset is asserted. The default value of these counters is 0.

The register name with "_0" refers to the least significant word of the counter and "_1" refers to the most significant word.

Table 5.20. Summary of Statistics Counters

Offset	Register Name	Access	Description
0x044	TX_STAT_PKT_LNGTH_0	RO	Transmit Packet Length Statistics Counter. Indicates the total number of octets transmitted in a particular frame. tx_statvec_o[13:0] is used to implement this counter.
0x048	TX_STAT_PKT_LNGTH_1	RO	
0x04C	TX_STAT_ERR_0	RO	Transmit TX Error Statistics Counter. Counts the total number of PHY terminated packet. The tx_statvec_o[24] is used to implement this counter.
0x050	TX_STAT_ERR_1	RO	
0x054	TX_STAT_UNDER_RUN_0	RO	Transmit Underrun Error Statistics Counter. Counts the total number of underrun packets transmitted. tx_statvec_o[21] is used to implement this counter.
0x058	TX_STAT_UNDER_RUN_1	RO	
0x05C	TX_STAT_CRC_ERR_0	RO	Transmit CRC Error Statistics Counter. Counts the total number of packets transmitted with

Offset	Register Name	Access	Description
0x060	TX_STAT_CRC_ERR_1	RO	CRC error. tx_statvec_o[22] is used to implement this counter.
0x064	TX_STAT_LNGTH_ERR_0	RO	Transmit Length Error Statistics Counter. Counts the total number of packets transmitted with length of the packet and length in the Length/Type field mismatch. tx_statvec_o[23] is used to implement this counter.
0x068	TX_STAT_LNGTH_ERR_1	RO	
0x06C	TX_STAT_LNG_PKT_0	RO	Transmit Long packet Statistics Counter. Counts the total number of packets transmitted with length of the packet longer than the max_frm_size. tx_statvec_o[25] is used to implement this counter.
0x070	TX_STAT_LNG_PKT_1	RO	
0x074	TX_STAT_MULTCST_0	RO	Transmit Multicast Packet Statistics Counter. Counts the total number of multicast packets transmitted. tx_statvec_o[20] is used to implement this counter.
0x078	TX_STAT_MULTCST_1	RO	
0x07C	TX_STAT_BRDCST_0	RO	Transmit Broadcast Packet Statistics Counter. Counts the total number of broadcast packets transmitted. tx_statvec_o[19] is used to implement this counter.
0x080	TX_STAT_BRDCST_1	RO	
0x084	TX_STAT_CNT_0	RO	Transmit Control Packet Statistics Counter. Counts the total number of control packets (pause frame) transmitted by the AXI4-Stream interface. tx_statvec_o[16] is used to implement this counter.
0x088	TX_STAT_CNT_1	RO	
0x08C	TX_STAT_JMBO_0	RO	Transmit Jumbo Packet Statistics Counter. Counts the total number of jumbo packets transmitted. tx_statvec_o[17] is used to implement this counter.
0x090	TX_STAT_JMBO_1	RO	
0x094	TX_STAT_PAUSE_0	RO	Transmit Pause Packet Statistics Counter. Counts the total number of pause packets inserted by the MAC core (enabled through MAC_CTL register). tx_statvec_o[15] is used to implement this counter.
0x098	TX_STAT_PAUSE_1	RO	
0x09C	TX_STAT_VLN_TG_0	RO	Transmit VLAN Tag Statistics Counter. Counts the total number of tagged packets transmitted. tx_statvec_o[18] is used to implement this counter.
0x0A0	TX_STAT_VLN_TG_1	RO	
0x0A4	TX_STAT_PKT_OK_0	RO	Transmit Packet OK Statistics Counter. Counts the total number of packets transmitted without any errors. tx_statvec_o[14] is used to implement this counter.
0x0A8	TX_STAT_PKT_OK_1	RO	
0x0AC	TX_STAT_PKT_64_0	RO	Transmit Packet 64 Statistics Counter. Counts the total number of packets transmitted with length equal to 64.
0x0B0	TX_STAT_PKT_64_1	RO	
0x0B4	TX_STAT_PKT_65_127_0	RO	Transmit Packet 65 - 127 Statistics Counter. Counts the total number of packets transmitted with lengths between 65 and 127.
0x0B8	TX_STAT_PKT_65_127_1	RO	
0x0BC	TX_STAT_PKT_128_255_0	RO	Transmit Packet 128-255 Statistics Counter. Counts the total number of packets transmitted with lengths between 128 and 255.
0x0C0	TX_STAT_PKT_128_255_1	RO	
0x0C4	TX_STAT_PKT_256_511_0	RO	Transmit Packet 256-511 Statistics Counter. Counts the total number of packets transmitted with lengths between 256 and 511.
0x0C8	TX_STAT_PKT_256_511_1	RO	

Offset	Register Name	Access	Description
0x0CC	TX_STAT_PKT_512_1023_0	RO	Transmit Packet 512-1023 Statistics Counter. Counts the total number of packets transmitted with lengths between 512 and 1,023.
0x0D0	TX_STAT_PKT_512_1023_1	RO	
0x0D4	TX_STAT_PKT_1024_1518_0	RO	Transmit Packet 1024-1518 Statistics Counter. Counts the total number of packets transmitted with lengths between 1,024 and 1,518.
0x0D8	TX_STAT_PKT_1024_1518_1	RO	
0x0DC	TX_STAT_PKT_1518_0	RO	Transmit Packet 1518 Statistics Counter. Counts the total number of packets transmitted with length greater than 1,518.
0x0E0	TX_STAT_PKT_1518_1	RO	
0x0E4	TX_STAT_FRM_ERR_0	RO	Transmit Frame Error Statistics Counter. Counts the total number of packets transmitted with error. tx_statvec_o[14] is used to implement this counter.
0x0E8	TX_STAT_FRM_ERR_1	RO	
0x0EC	TX_STAT_PKT_1519_2047_0	RO	Transmit Packet 1519-2047 Statistics Counter. Counts the total number of packets transmitted with lengths between 1,024 and 2,047.
0x0F0	TX_STAT_PKT_1519_2047_1	RO	
0x0F4	TX_STAT_PKT_2048_4095_0	RO	Transmit Packet 2048-4095 Statistics Counter. Counts the total number of packets transmitted with lengths between 2,048 and 4,095.
0x0F8	TX_STAT_PKT_2048_4095_1	RO	
0x0FC	TX_STAT_PKT_4096_9216_0	RO	Transmit Packet 4096-9216 Statistics Counter. Counts the total number of packets transmitted with lengths between 4,096 and 9,216.
0x100	TX_STAT_PKT_4096_9216_1	RO	
0x104	TX_STAT_PKT_9217_16383_0	RO	Transmit Packet 9217-16383 Statistics Counter. Counts the total number of packets transmitted with lengths between 9,217 and 16,383.
0x108	TX_STAT_PKT_9217_16383_1	RO	
0x10C	RX_STAT_PKT_LNGTH_0	RO	Receive Packet Length Statistics Counter. Indicated the length of the packet received. rx_statvec_o[13:0] is used to implement this counter.
0x110	RX_STAT_PKT_LNGTH_1	RO	
0x114	RX_STAT_VLN_TG_0	RO	Receive VLAN Tag Statistics Counter. Counts the total number of tagged packets received. rx_statvec_o[18] is used to implement this counter.
0x118	RX_STAT_VLN_TG_1	RO	
0x11C	RX_STAT_PAUSE_0	RO	Receive Pause Packet Statistics Counter. Counts the total number of pause packets received. rx_statvec_o[19] is used to implement this counter.
0x120	RX_STAT_PAUSE_1	RO	
0x124	RX_STAT_FLT_0	RO	Receive Filtered Packet Statistics Counter. rx_statvec_o[22] is used to implement this counter.
0x128	RX_STAT_FLT_1	RO	
0x12C	RX_STAT_UNSP_OPCODE_0	RO	Receive Unsupported Opcode Statistics Counter. Counts the number of packets received with unsupported Opcode. rx_statvec_o[23] is used to implement this counter.
0x130	RX_STAT_UNSP_OPCODE_1	RO	
0x134	RX_STAT_BRDCST_0	RO	Receive Broadcast Packet Statistics Counter. Counts the number of packets received that were directed to the broadcast address. This does not include multicast packets. rx_statvec_o[15] is used to implement this counter.
0x138	RX_STAT_BRDCST_1	RO	
0x13C	RX_STAT_MULTCST_0	RO	Receive Multicast Packet Statistics Counter. Counts the number of packets received that were directed to the

Offset	Register Name	Access	Description
0x140	RX_STAT_MULTCST_1	RO	multicast address. This does not include broadcast packets. rx_statvec_o[16] is used to implement this counter.
0x144	RX_STAT_LNGTH_ERR_0	RO	Receive Length Error Statistics Counter. Counts the total number of packets received with length of the packet and length in the Length/Type field mismatch. rx_statvec_o[20] is used to implement this counter.
0x148	RX_STAT_LNGTH_ERR_1	RO	
0x14C	RX_STAT_LNG_PKT_0	RO	Receive Long Packet Statistics Counter. Counts the number of packets received longer than the max_pkt_len. rx_statvec_o[21] is used to implement this counter.
0x150	RX_STAT_LNG_PKT_1	RO	
0x154	RX_STAT_CRC_ERR_0	RO	Receive CRC Error Statistics Counter. Counts the number of packets received with CRC error. rx_statvec_o[17] is used to implement this counter.
0x158	RX_STAT_CRC_ERR_1	RO	
0x15C	RX_STAT_PKT_DISCARD_0	RO	Receive Packet Discard Statistics Counter. Counts the number of packets discarded at the receive end. rx_statvec_o[14] is used to implement this counter.
0x160	RX_STAT_PKT_DISCARD_1	RO	
0x164	RX_STAT_PKT_IGNORE_0	RO	Receive Packet Ignored Statistics Counter. Counts the number of packets ignored when you request using the ignore_pkt. rx_statvec_o[25] is used to implement this counter.
0x168	RX_STAT_PKT_IGNORE_1	RO	
0x16C	RX_STAT_PKT_FRAGMENTS_0	RO	Receive Packet Fragments Statistics Counter. Counts the number of packets received with less than 64 octets in length and has either an FCS error or an alignment error. rx_statvec_o[13:6] along with rx_statvec[17] are used to implement this counter.
0x170	RX_STAT_PKT_FRAGMENTS_1	RO	
0x174	RX_STAT_PKT_JABBERS_0	RO	Receive Packet Jabbers Statistics Counter. Counts the number of packets received with length longer than 1,518 octets and has either an FCS error or an alignment error. rx_statvec_o[13:0] along with rx_statvec_o[17] are used to implement this counter.
0x178	RX_STAT_PKT_JABBERS_1	RO	
0x17C	RX_STAT_PKT_64_0	RO	Receive Packet 64 Statistics Counter. Counts the number of packets received that were 64 octets in length (including bad packets). rx_statvec_o[13:0] is used to implement this counter.
0x180	RX_STAT_PKT_64_1	RO	
0x184	RX_STAT_PKT_65_127_0	RO	Receive Packet 65-127 Statistics Counter. Counts the number of packets received that were between 65-127 octets in length (including bad packets).
0x188	RX_STAT_PKT_65_127_1	RO	
0x18C	RX_STAT_PKT_128_255_0	RO	Receive Packet 128-255 Statistics Counter. Counts the number of packets received that were between 128-255 octets in length (including bad packets).
0x190	RX_STAT_PKT_128_255_1	RO	
0x194	RX_STAT_PKT_256_511_0	RO	Receive Packet 256-511 Statistics Counter. Counts the number of packets received that were between 256-

Offset	Register Name	Access	Description
0x198	RX_STAT_PKT_256_511_1	RO	511 octets in length (including bad packets).
0x19C	RX_STAT_PKT_512_1023_0	RO	Receive Packet 512-1023 Statistics Counter. Counts the number of packets received that were between 512-1023 octets in length (including bad packets).
0x1A0	RX_STAT_PKT_512_1023_1	RO	
0x1A4	RX_STAT_PKT_1024_1518_0	RO	Receive Packet 1024-1518 Statistics Counter. Counts the number of packets received that were between 1024- 1518 octets in length (including bad packets).
0x1A8	RX_STAT_PKT_1024_1518_1	RO	
0x1AC	RX_STAT_PKT_UNDERSIZE_0	RO	Receive Packet Undersize Statistics Counter. Counts the number of packets received that were less than 64 octets long and were otherwise well formed.
0x1B0	RX_STAT_PKT_UNDERSIZE_1	RO	
0x1B4	RX_STAT_PKT_UNICAST_0	RO	Receive Packet Unicast Statistics Counter. Counts the number of good packets received that were directed to a single address.
0x1B8	RX_STAT_PKT_UNICAST_1	RO	
0x1BC	RX_STAT_PKT_RCVD_0	RO	Packets Received Statistics Counter. Counts the number of packets received (including bad packet, broadcast, and multicast packets). rx_statvec[15] and rx_statvec_o[16] are used to implement this counter.
0x1C0	RX_STAT_PKT_RCVD_1	RO	
0x1C4	RX_STAT_PKT_64_GOOD_CRC_0	RO	Receive Packet 64 with Good CRC Statistics Counter. Counts the number of packets received with length less than 64 and with a good CRC. rx_statvec_o[13:6] and rx_statvec_o[17] are used to implement this counter.
0x1C8	RX_STAT_PKT_64_GOOD_CRC_1	RO	
0x1CC	RX_STAT_PKT_1518_GOOD_CRC_0	RO	Receive Packet 1518 with Good CRC Statistics Counter. Counts the number of packets received with length more than 1518 and with a good CRC. rx_statvec_o[13:0] and rx_statvec_o[17] are used to implement this counter.
0x1D0	RX_STAT_PKT_1518_GOOD_CRC_1	RO	
0x1D4	RX_STAT_PKT_1519_2047_0	RO	Receive Packet 1519-2047 Statistics Counter. Counts the number of packets received that were between 1,519 - 2,047 octets in length (including bad packets).
0x1D8	RX_STAT_PKT_1519_2047_1	RO	
0x1DC	RX_STAT_PKT_2048_4095_0	RO	Receive Packet 2048-4095 Statistics Counter. Counts the number of packets received that were between 2,048 – 4,095 octets in length (including bad packets).
0x1E0	RX_STAT_PKT_2048_4095_1	RO	
0x1E4	RX_STAT_PKT_4096_9216_0	RO	Receive Packet 4096-9216 Statistics Counter. Counts the number of packets received that were between 4,096 – 9,216 octets in length (including bad packets).
0x1E8	RX_STAT_PKT_4096_9216_1	RO	
0x1EC	RX_STAT_PKT_9217_16383_0	RO	Receive Packet 9217-16383 Statistics Counter. Counts the number of packets received that were between 9,217 – 16,383 octets in length (including bad packets).
0x1F0	RX_STAT_PKT_9217_16383_1	RO	
0x1F4	TX_STAT_PFC_PKT_0	RO	Transmit PFC Frame Statistics Counter. Counts the total number of transmitted error-free PFC

Offset	Register Name	Access	Description
0x1F8	TX_STAT_PFC_PKT_1	RO	frames. tx_statvec_o[27] is used to implement this counter.
0x1FC	RX_STAT_PFC_PKT_0	RO	Receive PFC Frame Statistics Counter. Counts the total number of received error-free PFC frames. rx_statvec_o[27] is used to implement this counter.
0x200	RX_STAT_PFC_PKT_1	RO	
0X204	TX_STAT_PTP1588_PKT_0	RO	Transmit PTP1588 Frame Statistics Counter. Counts the total number of PTP1588 frames transmitted. tx_statvec_o[26] is used to implement this counter.
0x208	TX_STAT_PTP1588_PKT_1	RO	
0x20C	RX_STAT_PTP1588_PKT_0	RO	Receive PTP1588 Statistics Counter. Counts the number of PTP1588 packets received. The rx_statvec_o[26] is used to implement this counter.
0x210	RX_STAT_PTP1588_PKT_1	RO	
0x214	TX_ STAT_SVLN_TG_0	RO	Transmit Stacked VLAN Tag Statistics Counter. Counts the total number of stacked VLAN packets transmitted. tx_statvec_o[28] is used to implement this counter.
0x218	TX_STAT_SVLN_TG_1	RO	
0x21C	RX_STAT_SVLN_TG_0	RO	Receive Stacked VLAN Tag Statistics Counter. Counts the total number of stacked VLAN packets received. rx_statvec_o[28] is used to implement this counter.
0x220	RX_STAT_SVLN_TG_1	RO	
0x22C	RX_STAT_LINK_OK	RO	Receive link_ok Statistics Counter. Counts the number of link_ok bit received. Note: If you wish to use the RX link_ok statistic counter, you cannot reset the MAC signal, such as the xg_pcsrdy_o signal when the RX PHY is down.
0x234	TX_STAT_PKT_LNGTH_ACCU_0	RO	Transmit Stacked Packet Length Accumulated Statistic Counter. Accumulate total number of bytes transmitted for the frames (only available in IP v2.1.0 release).
0x238	TX_STAT_PKT_LNGTH_ACCU_1	RO	
0x23C	RX_STAT_PKT_LNGTH_ACCU_0	RO	Receive Stacked Packet Length Accumulated Statistic Counter. Accumulate total number of bytes received for the frames (only available in IP v2.1.0 release).
0x240	RX_STAT_PKT_LNGTH_ACCU_1	RO	

5.2.4. Stacked VLAN Register

Table 5.21. Summary of Stacked VLAN Registers

Offset	Register Name	Access	Description
0x27C	SVLAN_TAG	RO	VLAN Tag Register

5.2.4.1. SVLAN_TAG Register

This register has the stacked VLAN tag field of the most recent double tagged frame that was received.

Table 5.22. Summary of SVLAN TAG Register

Bit	Label	Description	Default
[31:16]	Reserved	Reserved bits	0
[15:0]	svlan_tag	SVLAN Tag ID.	0

5.2.5. Priority Flow Control Registers

The registers responsible for priority flow control operations are summarized in the following table.

Table 5.23. Summary of Priority Flow Control (PFC) Registers

Offset	Register Name	Access	Description
0x280	PFC_P0_TM	RW	Priority 0 Quanta Register
0x284	PFC_P1_TM	RW	Priority 1 Quanta Register
0x288	PFC_P2_TM	RW	Priority 2 Quanta Register
0x28C	PFC_P3_TM	RW	Priority 3 Quanta Register
0x290	PFC_P4_TM	RW	Priority 4 Quanta Register
0x294	PFC_P5_TM	RW	Priority 5 Quanta Register
0x298	PFC_P6_TM	RW	Priority 6 Quanta Register
0x29C	PFC_P7_TM	RW	Priority 7 Quanta Register
0x2A0	TX_PFC_CTL	RW	Transmit PFC Frame Request Register
0x2A4	PFC_OPCODE	RW	Priority Flow Control Opcode

5.2.5.1. PFC_P0_TM Register

This register contains the pause time for a priority 0 flow control frame and can be configured during run-time. The new values of the registers will only take effect when tx_pfc_tuser_i[0] or TX_PFC_CTL[0] is re-asserted or still asserted when quanta counter refreshes.

To ensure the pause time is properly included in the PFC frame, you must wait for at least ten clock cycles to trigger a PFC frame transmission after writing pause time to this register.

Table 5.24. Summary of PFC_PO_TM Registers

Bit	Label	Description	Default
[31:16]	tx_pfc_p0_refresh	Priority 0 Refresh Period. This field defines the refresh period in quanta (each quanta is equivalent to 512-bit time), in which a new PFC XOFF frame will be transmitted again according to current tx_pfc_tuser_i[0] input or TX_PFC_CTL[0] bit assertion. After refresh, the refresh counter will be restarted. Valid when PFC for priority 0 is enabled.	16'h7F00
[15:0]	tx_pfc_p0_quanta	Priority 0 Pause Quanta. This field defines the pause duration in quanta (each quanta is equivalent to 512-bit time) for priority 0 when generating a PFC frame. Valid when PFC for priority 0 is enabled.	16'h7FFF

5.2.5.2. PFC_P1_TM Register

This register contains the pause time for a priority 1 flow control frame and can be configured during run-time. The new values of the registers will only take effect when tx_pfc_tuser_i[1] or TX_PFC_CTL[1] is re-asserted or still asserted when quanta counter refreshes.

To ensure the pause time is properly included in the PFC frame, you must wait for at least ten clock cycles to trigger a PFC frame transmission after writing pause time to this register.

Table 5.25. Summary of PFC_P1_TM Registers

Bit	Label	Description	Default
[31:16]	tx_pfc_p1_refresh	Priority 1 Refresh Period. This field defines the refresh period in quanta (each quanta is equivalent to 512-bit time), in which a new PFC XOFF frame will be transmitted again according to current tx_pfc_tuser_i[1] input or TX_PFC_CTL[1] bit assertion. After refresh, the refresh counter will be restarted. Valid when PFC for priority 1 is enabled.	16'h7F00
[15:0]	tx_pfc_p1_quanta	Priority 1 Pause Quanta. This field defines the pause duration in quanta (each quanta is equivalent to 512-bit time) for priority 1 when generating a PFC frame. Valid when PFC for priority 1 is enabled.	16'h7FFF

5.2.5.3. PFC_P2_TM Register

This register contains the pause time for a priority 2 flow control frame and can be configured during run-time. The new values of the registers will only take effect when tx_pfc_tuser_i[2] or TX_PFC_CTL[2] is re-asserted or still asserted when quanta counter refreshes.

To ensure the pause time is properly included in the PFC frame, you must wait for at least ten clock cycles to trigger a PFC frame transmission after writing pause time to this register.

Table 5.26. Summary of PFC_P2_TM Registers

Bit	Label	Description	Default
[31:16]	tx_pfc_p2_refresh	Priority 2 Refresh Period. This field defines the refresh period in quanta (each quanta is equivalent to 512-bit time), in which a new PFC XOFF frame will be transmitted again according to current tx_pfc_tuser_i[2] input or TX_PFC_CTL[2] bit assertion. After refresh, the refresh counter will be restarted. Valid when PFC for priority 2 is enabled.	16'h7F00
[15:0]	tx_pfc_p2_quanta	Priority 2 Pause Quanta. This field defines the pause duration in quanta (each quanta is equivalent to 512-bit time) for priority 2 when generating a PFC frame. Valid when PFC for priority 2 is enabled.	16'h7FFF

5.2.5.4. PFC_P3_TM Register

This register contains the pause time for a priority 3 flow control frame and can be configured during run-time. The new values of the registers will only take effect when tx_pfc_tuser_i[3] or TX_PFC_CTL[3] is re-asserted or still asserted when quanta counter refreshes.

To ensure the pause time is properly included in the PFC frame, you must wait for at least ten clock cycles to trigger a PFC frame transmission after writing pause time to this register.

Table 5.27. Summary of PFC_P3_TM Registers

Bit	Label	Description	Default
[31:16]	tx_pfc_p3_refresh	Priority 3 Refresh Period. This field defines the refresh period in quanta (each quanta is equivalent to 512-bit time), in which a new PFC XOFF frame will be transmitted again according to current tx_pfc_tuser_i[3] input or TX_PFC_CTL[3] bit assertion. After refresh, the refresh counter will be restarted. Valid when PFC for priority 3 is enabled.	16'h7F00
[15:0]	tx_pfc_p3_quanta	Priority 3 Pause Quanta. This field defines the pause duration in quanta (each quanta is equivalent to 512-bit time) for priority 3 when generating a PFC frame. Valid when PFC for priority 3 is enabled.	16'h7FFF

5.2.5.5. PFC_P4_TM Register

This register contains the pause time for a priority 4 flow control frame and can be configured during run-time. The new values of the registers will only take effect when tx_pfc_tuser_i[4] or TX_PFC_CTL[4] is re-asserted or still asserted when quanta counter refreshes.

To ensure the pause time is properly included in the PFC frame, you need to wait for at least ten clock cycles to trigger PFC frame transmission after writing pause time to this register.

Table 5.28. Summary of PFC_P4_TM Registers

Bit	Label	Description	Default
[31:16]	tx_pfc_p4_refresh	Priority 4 Refresh Period. This field defines the refresh period in quanta (each quanta is equivalent to 512-bit time), in which a new PFC XOFF frame will be transmitted again according to current tx_pfc_tuser_i[4] input or TX_PFC_CTL[4] bit assertion. After refresh, the refresh counter will be restarted. Valid when PFC for priority 4 is enabled.	16'h7F00
[15:0]	tx_pfc_p4_quanta	Priority 4 Pause Quanta. This field defines the pause duration in quanta (each quanta is equivalent to 512-bit time) for priority 4 when generating a PFC frame. Valid when PFC for priority 4 is enabled.	16'h7FFF

5.2.5.6. PFC_P5_TM Register

This register contains the pause time for a priority 5 flow control frame and can be configured during run-time. The new values of the registers will only take effect when tx_pfc_tuser_i[5] or TX_PFC_CTL[5] is re-asserted or still asserted when quanta counter refreshes.

To ensure the pause time is properly included in the PFC frame, you must wait for at least ten clock cycles to trigger PFC frame transmission after writing pause time to this register.

Table 5.29. Summary of PFC_P5_TM Registers

Bit	Label	Description	Default
[31:16]	tx_pfc_p5_refresh	Priority 5 Refresh Period. This field defines the refresh period in quanta (each quanta is equivalent to 512-bit time), in which a new PFC XOFF frame will be transmitted again according to current tx_pfc_tuser_i[5] input or TX_PFC_CTL[5] bit assertion. After refresh, the refresh counter will be restarted. Valid when PFC for priority 5 is enabled.	16'h7F00
[15:0]	tx_pfc_p5_quanta	Priority 5 Pause Quanta. This field defines the pause duration in quanta (each quanta is equivalent to 512-bit time) for priority 5 when generating a PFC frame. Valid when PFC for priority 5 is enabled.	16'h7FFF

5.2.5.7. PFC_P6_TM Register

This register contains the pause time for a priority 6 control frame and can be configured during run-time. The new values of the registers will only take effect when tx_pfc_tuser_i[6] or TX_PFC_CTL[6] is re-asserted or still asserted when quanta counter refreshes.

To ensure the pause time is properly included in the PFC frame, you must wait for at least ten clock cycles to trigger PFC frame transmission after writing pause time to this register.

Table 5.30. Summary of PFC_P6_TM Registers

Bit	Label	Description	Default
[31:16]	tx_pfc_p6_refresh	Priority 6 Refresh Period. This field defines the refresh period in quanta (each quanta is equivalent to 512-bit time), in which a new PFC XOFF frame will be transmitted again according to current tx_pfc_tuser_i[6] input or TX_PFC_CTL[6] bit assertion. After refresh, the refresh counter will be restarted. Valid when PFC for priority 6 is enabled.	16'h7F00
[15:0]	tx_pfc_p6_quanta	Priority 6 Pause Quanta. This field defines the pause duration in quanta (each quanta is equivalent to 512-bit time) for priority 6 when generating a PFC frame. Valid when PFC for priority 6 is enabled.	16'h7FFF

5.2.5.8. PFC_P7_TM Register

This register contains the pause time for a priority 7 flow control frame and can be configured during run-time. The new values of the registers will only take effect when tx_pfc_tuser_i[7] or TX_PFC_CTL[7] is re-asserted or still asserted when quanta counter refreshes.

To ensure the pause time is properly included in the PFC frame, you must wait for at least ten clock cycles to trigger PFC frame transmission after writing pause time to this register.

Table 5.31. Summary of PFC_P7_TM Registers

Bit	Label	Description	Default
[31:16]	tx_pfc_p7_refresh	Priority 7 Refresh Period. This field defines the refresh period in quanta (each quanta is equivalent to 512-bit time), in which a new PFC XOFF frame will be transmitted again according to current tx_pfc_tuser_i[7] input or TX_PFC_CTL[7] bit assertion. After refresh, the refresh counter will be restarted. Valid when PFC for priority 7 is enabled.	16'h7F00
[15:0]	tx_pfc_p7_quanta	Priority 7 Pause Quanta. This field defines the pause duration in quanta (each quanta is equivalent to 512-bit time) for priority 7 when generating a PFC frame. Valid when PFC for priority 7 is enabled.	16'h7FFF

5.2.5.9. TX_PFC_CTL Register

This register contains the config bits that will trigger XOFF frame transmission when being set. To use these registers, tx_en bit of MODE register as well as tx_pfc_en and tx_pfc_p[0-7] bit of TX_CTL register must be set first.

Table 5.32. Summary of TX_PFC_CTL Registers

Bit	Label	Description	Default
[31:8]	Reserved	Reserved bits.	0
[7]	tx_p7_pfcreq	Request TX PFC Frame (Priority 7). Request TX PFC Frame. 1 = send request, 0 = don't send request. When this bit is set to '1', a PFC frame will be constructed and transmitted at the next available transmission slot. Setting TX_PFC_CTL[7] is equivalent to asserting tx_pfc_tuser[7]. Valid when PFC for priority 7 is enabled, where it must only be set at least one clock cycle after tx_pfc_en and tx_pfc_p7_en bits of TX_CTL register have been set.	0
[6]	tx_p6_pfcreq	Request TX PFC Frame (Priority 6). Request TX PFC Frame. 1 = send request, 0 = don't send request. When this bit is set to '1', a PFC frame will be constructed and transmitted at the next available transmission slot. Setting TX_PFC_CTL[6] is equivalent to asserting tx_pfc_tuser[6]. Valid when PFC for priority 6 is enabled, where it must only be set at least one clock cycle after tx_pfc_en and tx_pfc_p6_en bits of TX_CTL register have been set.	0
[5]	tx_p5_pfcreq	Request TX PFC Frame (Priority 5). Request TX PFC Frame. 1 = send request, 0 = don't send request. When this bit is set to '1', a PFC frame will be constructed and transmitted at the next available transmission slot. Setting TX_PFC_CTL[5] is equivalent to asserting tx_pfc_tuser[5].	0

Bit	Label	Description	Default
		Valid when PFC for priority 5 is enabled, where it must only be set at least one clock cycle after tx_pfc_en and tx_pfc_p5_en bits of TX_CTL register have been set.	
[4]	tx_p4_pfcreq	Request TX PFC Frame (Priority 4). Request TX PFC Frame. 1 = send request, 0 = don't send request. When this bit is set to '1', a PFC frame will be constructed and transmitted at the next available transmission slot. Setting TX_PFC_CTL[4] is equivalent to asserting tx_pfc_tuser[4]. Valid when PFC for priority 4 is enabled, where it must only be set at least one clock cycle after tx_pfc_en and tx_pfc_p4_en bits of TX_CTL register have been set.	0
[3]	tx_p3_pfcreq	Request TX PFC Frame (Priority 3). Request TX PFC Frame. 1 = send request, 0 = don't send request. When this bit is set to '1', a PFC frame will be constructed and transmitted at the next available transmission slot. Setting TX_PFC_CTL[3] is equivalent to asserting tx_pfc_tuser[3]. Valid when PFC for priority 3 is enabled, where it must only be set at least one clock cycle after tx_pfc_en and tx_pfc_p3_en bits of TX_CTL register have been set.	0
[2]	tx_p2_pfcreq	Request TX PFC Frame (Priority 2). Request TX PFC Frame. 1 = send request, 0 = don't send request. When this bit is set to '1', a PFC frame will be constructed and transmitted at the next available transmission slot. Setting TX_PFC_CTL[2] is equivalent to asserting tx_pfc_tuser[2]. Valid when PFC for priority 2 is enabled, where it must only be set at least one clock cycle after tx_pfc_en and tx_pfc_p2_en bits of TX_CTL register have been set.	0
[1]	tx_p1_pfcreq	Request TX PFC Frame (Priority 1). Request TX PFC Frame. 1 = send request, 0 = don't send request. When this bit is set to '1', a PFC frame will be constructed and transmitted at the next available transmission slot. Setting TX_PFC_CTL[1] is equivalent to asserting tx_pfc_tuser[1]. Valid when PFC for priority 1 is enabled, where it must only be set at least one clock cycle after tx_pfc_en and tx_pfc_p1_en bits of TX_CTL register have been set.	0
[0]	tx_p0_pfcreq	Request TX PFC Frame (Priority 0). Request TX PFC Frame. 1 = send request, 0 = don't send request. When this bit is set to '1', a PFC frame will be constructed and transmitted at the next available transmission slot. Setting TX_PFC_CTL[0] is equivalent to asserting tx_pfc_tuser[0]. Valid when PFC for priority 0 is enabled, where it must only be set at least one clock cycle after tx_pfc_en and tx_pfc_p0_en bits of TX_CTL register have been set.	0

5.2.5.10. PFC_OPCODE Register

This register contains the Priority Flow Control Opcode. This will be compared against the Opcode in the received PFC frame. This value will also be included in any PFC frame transmitted by the MAC.

Table 5.33. Summary of PFC_OPCODE Registers

Bit	Label	Description	Default
[31:16]	Reserved	Reserved bits.	0
[15:0]	pfc_opcode	PFC Opcode	16'h0101

5.3. PHY Registers

This section provides detailed descriptions of 25G Ethernet PHY IP core registers.

Table 5.34. Access Types

Access Type	Behavior on Read Access	Behavior on Write Access
RO	Returns register value	Ignores write access.
WO	Returns 0	Updates register value.
RW	Returns register value	Updates register value.
RW1C	Returns register value	Writing 1'b1 on register bit clears the bit to 1'b0. Writing 1'b0 on register bit is ignored.
Reserved	Returns 0	Ignores write access.

5.3.1. Configuration Registers for PHY

The following table lists the register address map for the 25GbE PHY IP core.

Table 5.35. Register Address Map for PHY

AXI4-Lite Offset	Register Name	Bit Width	Description
0xA000 – 0xA0FC	PCS Module Registers	16	Refer to the Lattice Avant-G/X MPPHY Module User Guide (FPGA-IPUG-02233).

For more information on the address mapping, refer to the PHY Management Block section.

6. Example Design

The example design is generated along with the 25G Ethernet IP core in *the <Component Name>/eval* folder. The example design integrates the 25G Ethernet IP core with the AXI4-Lite and data checker into an Ethernet sub-system. This section describes how to simulate the example design using the QuestaSim™ Lattice FPGA Edition software as well as how to download and program the design into the development board.

The 25G Ethernet IP example design allows you to compile, simulate, and test the 25G Ethernet IP on the following Lattice evaluation board:

- Avant X Versa board

For more information on testing the IP with the evaluation board, refer to the Hardware Testing section.

6.1. Example Design Configuration

Table 6.1. 25G Ethernet IP Configuration Supported by the Example Design

Parameter Settings	Value
MAC+PHY Interface	25GMII
Multicast Address Filtering	Don’t Care
Statistics Counters Registers	Don’t Care
Loopback Mode	No Loopback

6.2. Overview of Example Design and Features

The following lists the key features of the example design:

- Ethernet packet generator to generate and compare packets
- Integrated 25G Ethernet MAC with 25G Ethernet PHY and OSC to demonstrate the expected clocking scheme
- Loopback transmitted packet at serial interface

6.3. Example Design Components

The 25G Ethernet IP example design includes the following blocks:

- 25G Ethernet IP core
- AXI4-Lite Module
- AXI-Stream Packet Generator
- Data checker

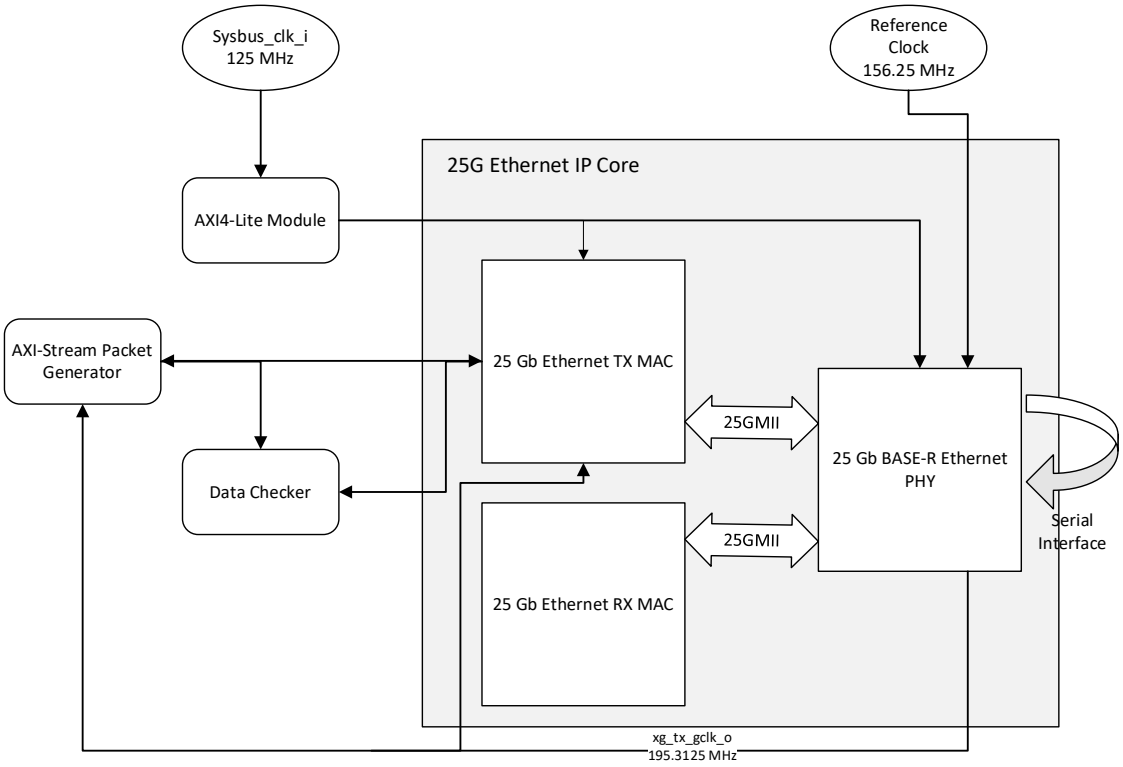

Figure 6.1. 25G Ethernet IP Example Design Block Diagram

6.3.1. 25G Ethernet IP Core

The 25G Ethernet MAC instantiates the 25G Ethernet IP core based on the configuration value described in Table 6.2. In the transmit data path, the 25 Gb Ethernet MAC receives packets from AXI-Stream Packet Generator and sends the packets to the 25G Ethernet PHY. In the receive data path, the 25G Ethernet MAC forwards packets from the 25G Ethernet PHY core to the Data Checker. The 25GbE MAC is connected to the 25GbE PHY within the 25G Ethernet IP core through the 25GMII interface. The clock source to the Ethernet MAC is from xg_tx_gclk_o, which is the clock output from the Ethernet PHY. The 25G Ethernet PHY core block instantiates the MPPHY foundation IP configured as a 1-lane 64b/66b PCS and a management block that supports AXI4-Lite access to PCS and MMD registers. In the transmit data path, the 25G Ethernet PHY sends packets from 25G Ethernet MAC core to serial interface signals. In the receive data path, it forwards the received packet at serial interface signals to the 25G Ethernet MAC core. The PHY implements the physical coding sublayer (PCS) and physical medium attachment (PMA) functionality based on the IEEE 802.3 25GBASE-R specification.

6.3.2. AXI4-Lite Module

The AXI4-Lite Module is the example design component that configures the register of the 25G Ethernet MAC. The example design component demonstrates the usage of the AXI4-Lite interface of the 25 Gb Ethernet and initializes the MAC to enable transmit and receive data path. You must modify the registers and the values to match your target application.

Table 6.2. AXI4-Lite Module Configuration Registers

25G Ethernet MAC Register Address Offset	25G Ethernet MAC Register Name	Configuration Value
0x000	MODE	0x3
0x004	TX_CTL	0x0
0x008	RX_CTL	0x1
0x010	IPG_VAL	0x10

6.3.3. AXI-Stream Packet Generator

The AXI-Stream Packet Generator generates packets for the AXI-Stream Transmit Interface of the 25G Ethernet MAC. The contents of the packets are randomized.

6.3.4. Data Checker

The Data Checker monitors the generated packets from the AXI-Stream Packet Generator and compares them with the packets from the AXI-Stream Receive Interface of the 25G Ethernet MAC core. An error signal is flagged if the packets are mismatched.

6.4. Generating an Example Design

This section describes how to create a new Radiant project, install the IP, and generate the example design.

6.4.1. Create a New Radiant Project

1. In the Radiant software, go to **File** → **New** → **Project…** or click on the **New Project** icon.

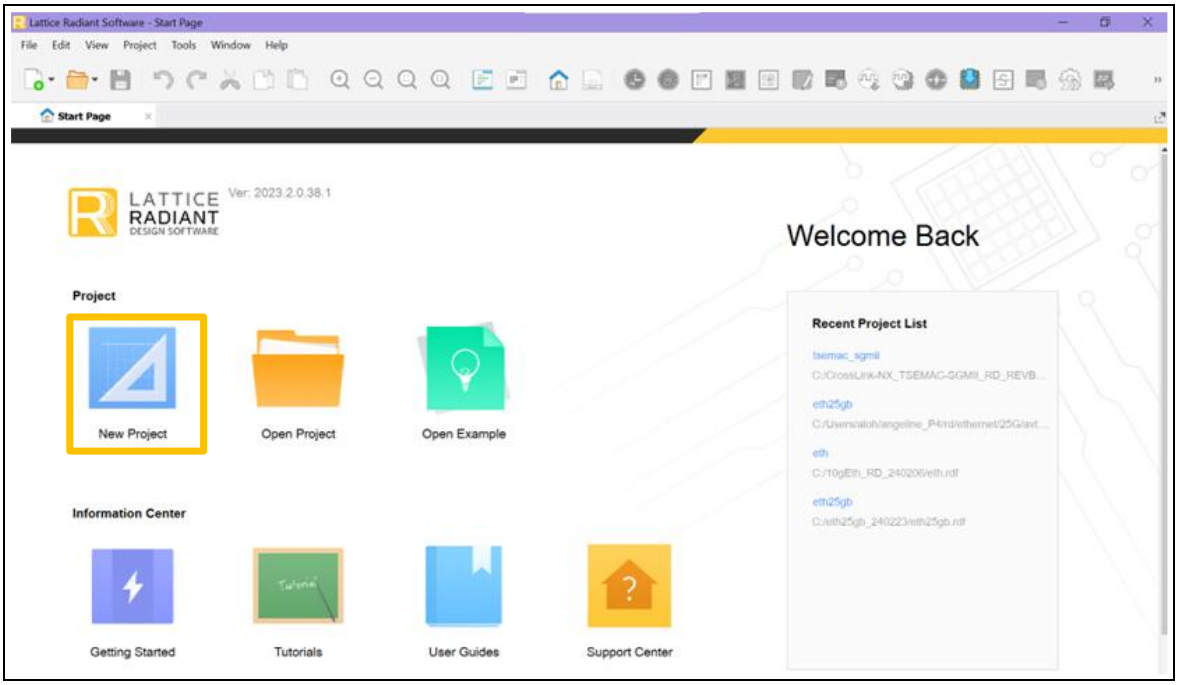

Figure 6.2. Start Page of the Radiant Software

2. Enter the project **Name** and **Location**, and click **Next**.

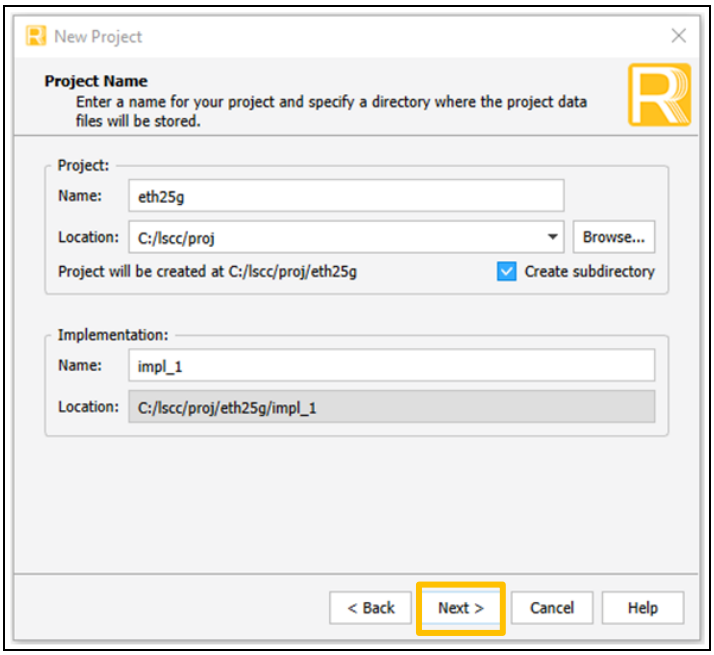

Figure 6.3. Start Page of the Radiant Software

3. In the **Add Source** window, click **Next**.

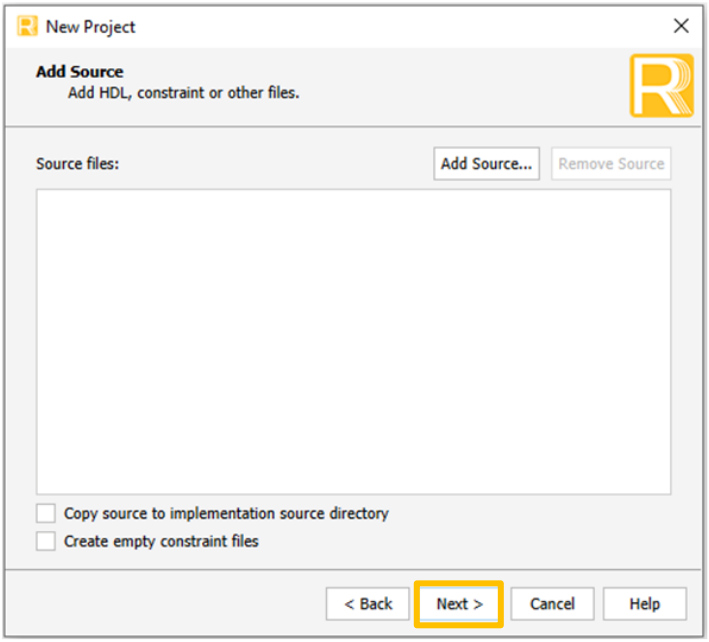

Figure 6.4. Add Source to Project

4. In the **Select Device** window, select **LAV-AT (Avant)** family, **LAV-AT-X70** device, and **LFG1156** package, then click **Next.**

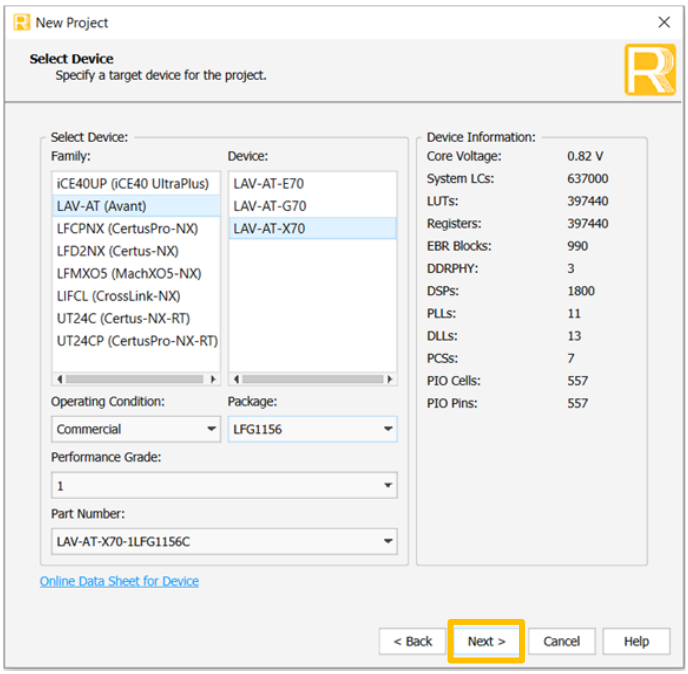

Figure 6.5. Select a Device for the Project

5. In the **Select Synthesis Tool** window, click **Next**.

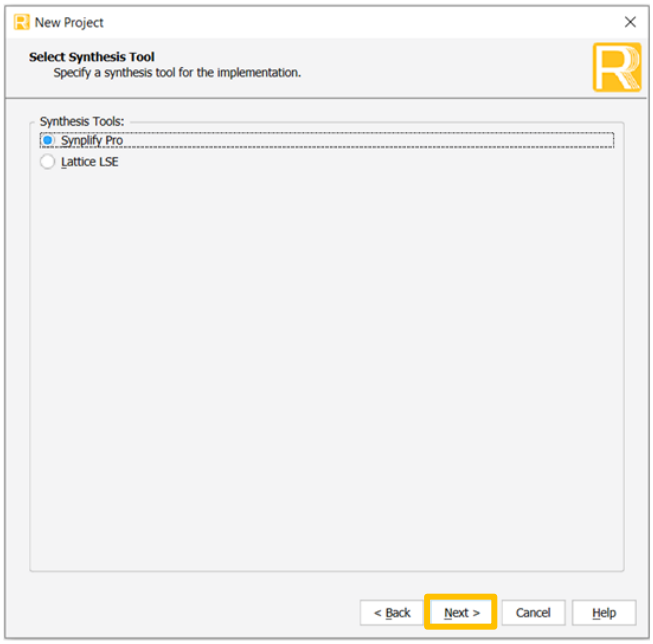

Figure 6.6. Select a Synthesis Tool

6. The Project Information window lists the specifications of the new project. Click **Finish**.

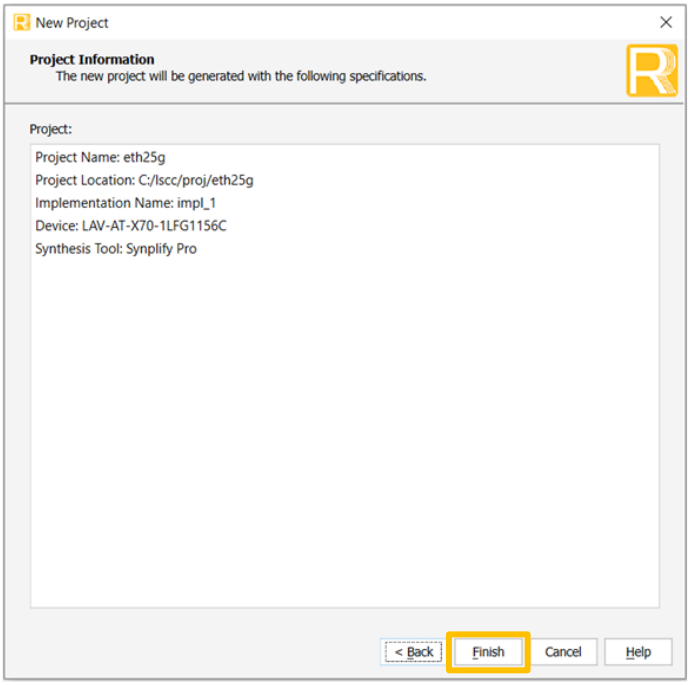

Figure 6.7. Project Information

7. The new Radiant project is generated. View the project summary under the **Project Summary** tab.

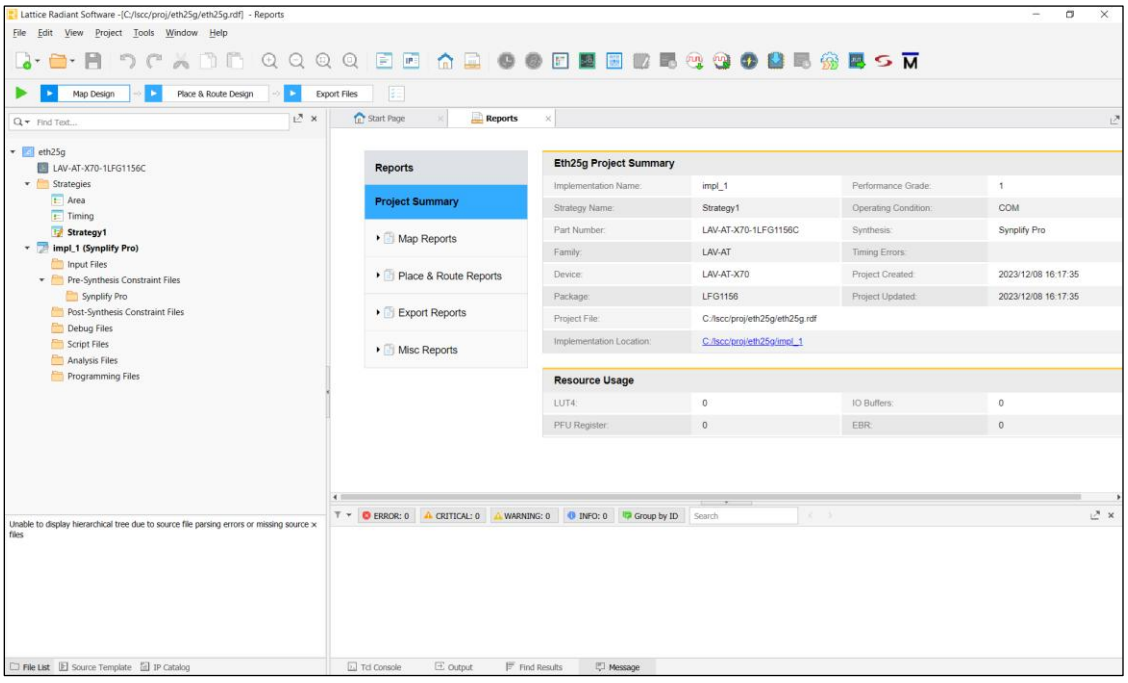

Figure 6.8. A Radiant Software Project is Created

6.4.2. IP Installation and Generation

1. Go to **IP Catalog → IP on Server** tab. Under the **Connectivity** category, right-click on the **25 Gb Ethernet** IP and select **Download** to install the IP. Alternatively, you can click on the icon to start the IP installation. If you already have the 25G Ethernet IP installed, proceed to step 2.

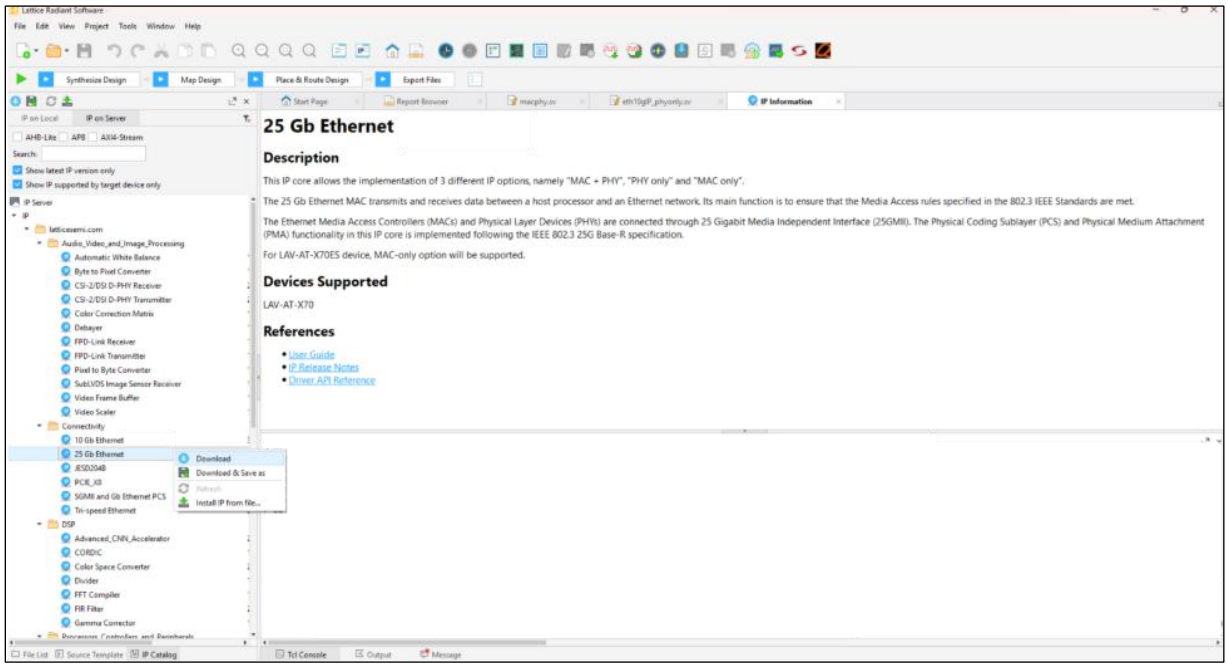

Figure 6.9. IP Catalog

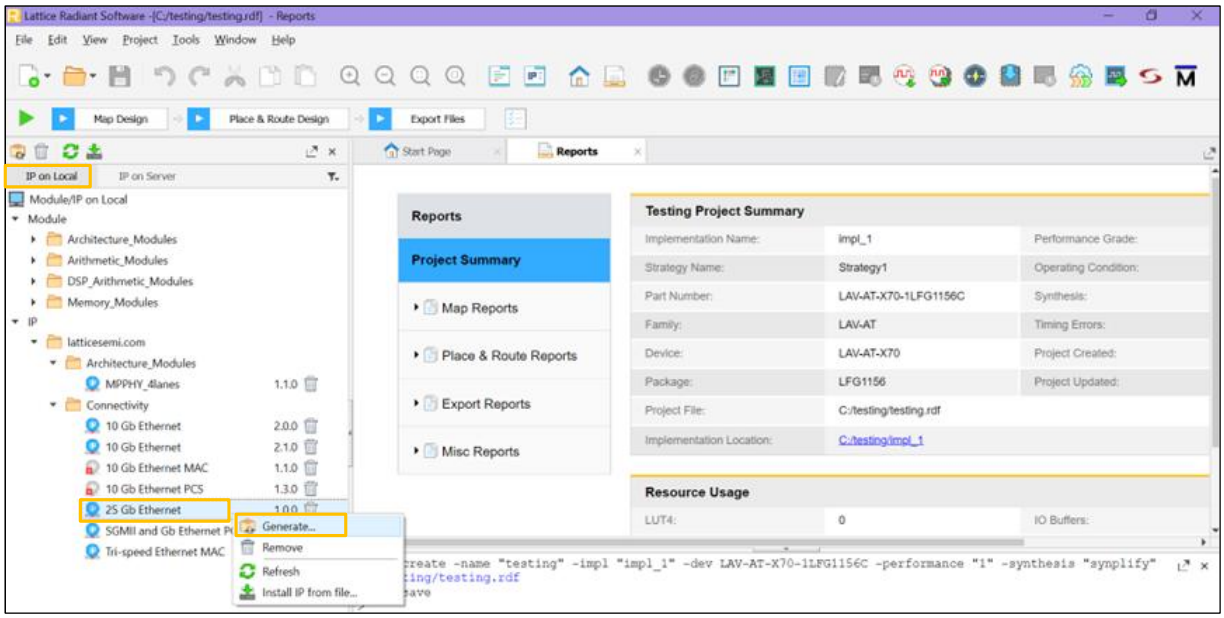

Figure 6.10. IP Generation

2. Enter **Component Name,** and click **Next**.

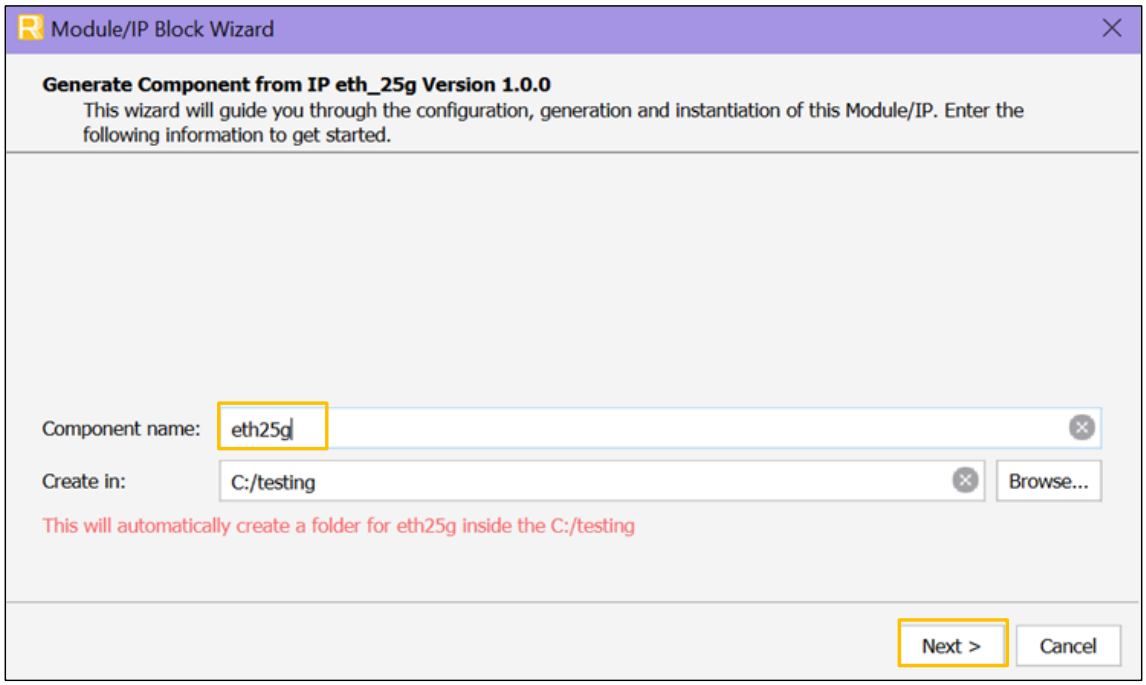

Figure 6.11. Generate Component

3. This window allows you to change the IP configuration. You may use the default settings and click **Generate.**

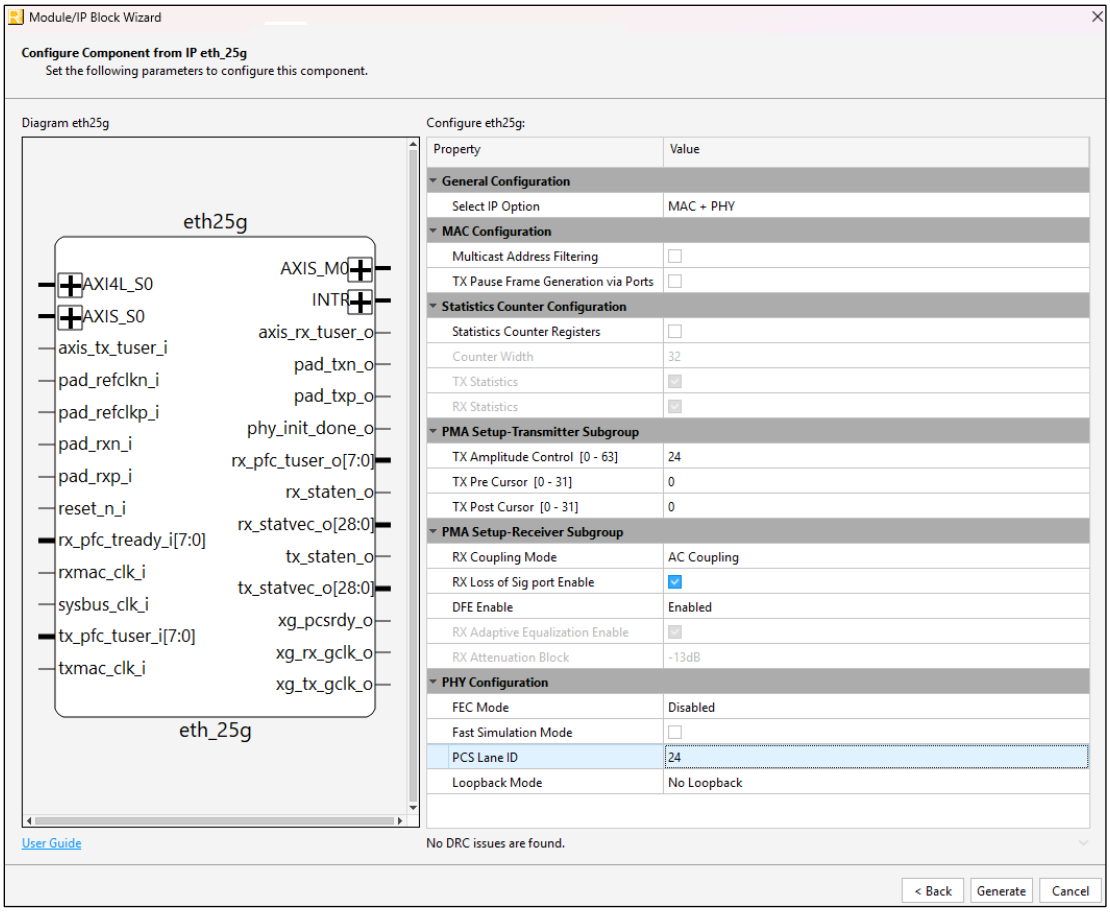

Figure 6.12. Configure Component

6.4.3. Importing Versa Files to a Project

The 25G Ethernet IP generates two sets of example design files in the eval directory of the IP.

Table 6.3. Comparison of Eval and Versa Example Design Files

Feature Settings	Example Design	
	Eval	Versa
Supported variants	MAC only PHY only MAC + PHY	MAC + PHY
Simulation	Yes	Yes
Run hardware in development board	No	Yes

The following figure shows the files generated under the Eval directory.

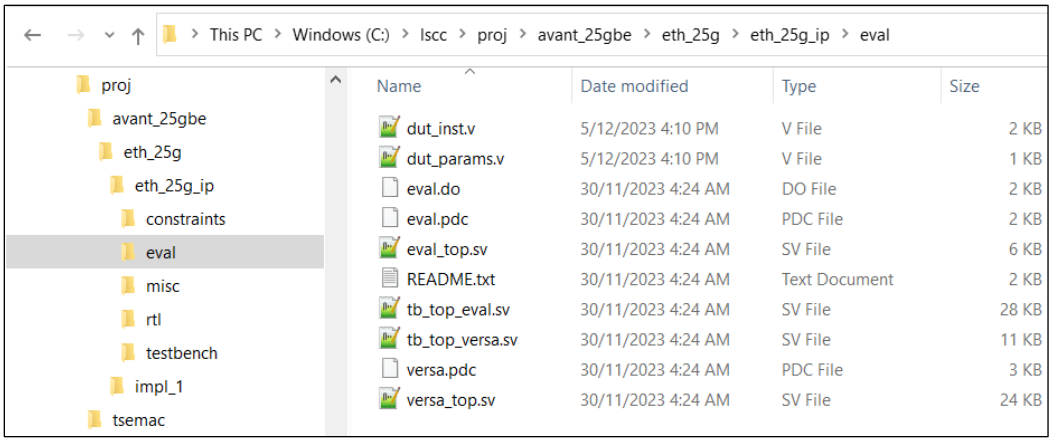

Figure 6.13. File List in the Eval Directory

Table 6.4. Description of Generated Files

Files	Description
README.txt	Contain instructions to use Eval and Versa example design files of this directory.
dut_inst.v	Contain instantiation of IP based on selected configuration of 25G Ethernet IP. Included by *eval_top.sv* and *versa_top.sv*.
dut_params.sv	Contain IP parameters based on selected configuration of 25G Ethernet IP. Included by *tb_top_eval.sv* and *tb_top_versa.sv*.
eval.pdc	Post-synthesis constraints for Eval example design.
eval.do	QuestaSim script to perform simulation for the Eval testbench.
eval_top.sv	Top-level design file for the Eval example design.
tb_top_eval.sv	Testbench for the Eval example design.
Files supported by Avant devices only: LAV-AT-X30, LAV-AT-X50, LAV-AT-X70 (eval/versa_top/avant_x70_25g)	
versa.pdc	Post-synthesis constraints for the Versa example design.
versa_top.sv	Top-level design file for the Versa example design.
tb_top_versa.sv	Testbench for the Versa example design.
debounce.v	Debounce module for hardware mechanical input.
osc_ip.v	OSC module from foundation IP.
psc_apb_w_r.v *pcs_apb_w.v*	APB module for register configuration.
traffic_genchk.v	Traffic generator module to generate random traffic for transmission and perform checking in loopback.
tb_top_versa.do	Script to group simulation signals.

To import Versa files to the Radiant software project, follow these steps:

1. Right-click on **Input Files**, select **Add → Existing File…,** then add the *versa_top.sv* file to the Radiant software project. The top level files can be found in the IP_NAME/eval/versa_top/avant_x70_25g folder.

   For Avant X70 Device

   Top level file to include in project: *versa_top.sv*

   Simulation for Top level file to include: *tb_top_versa.sv*

   Post-Synthesis Constraint File: *versa.pdc*

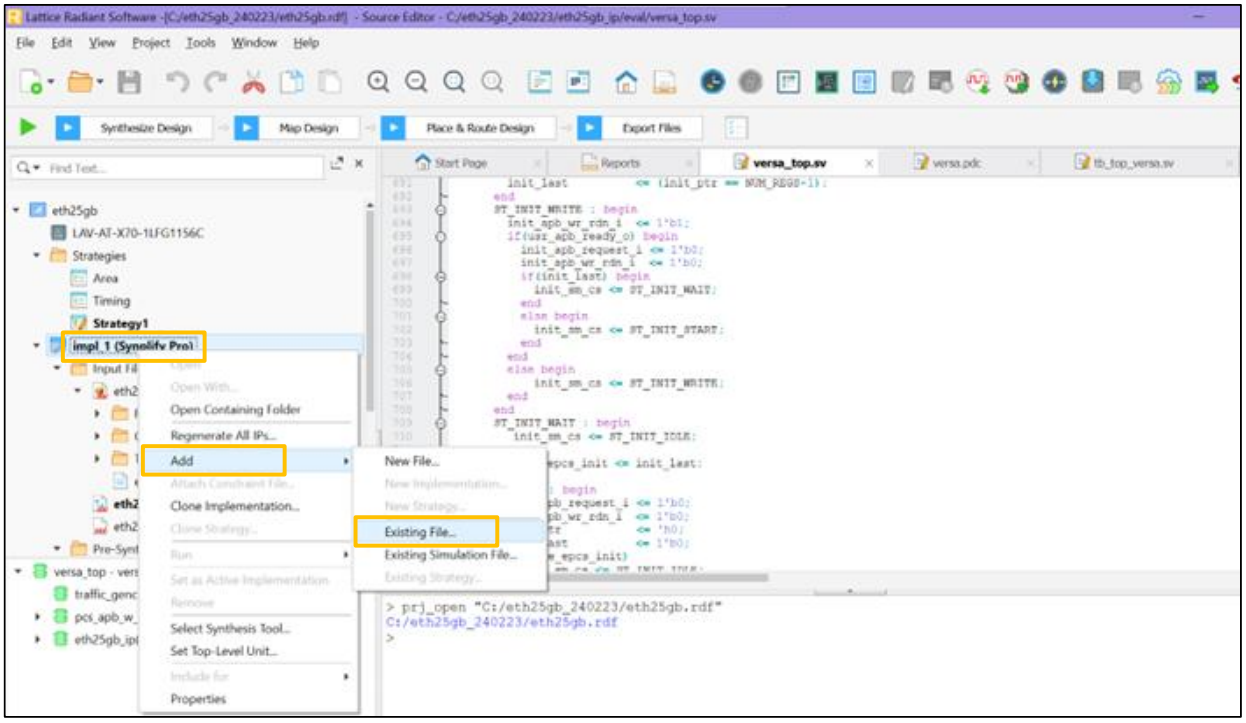

Figure 6.14. Add the versa_top.sv File into the Radiant Software Project

2. Versa design file: *versa_top.sv*. The SIM parameter remains unchanged for both modes.
   - Continuous mode:
     - CONTINUOUS_TRAFFIC parameter is default to 1.
     - Allows you to pause and resume the packet transmission via push button (SW13) in hardware.

     For more information on testing the IP with the evaluation board, refer to the Hardware Testing section.

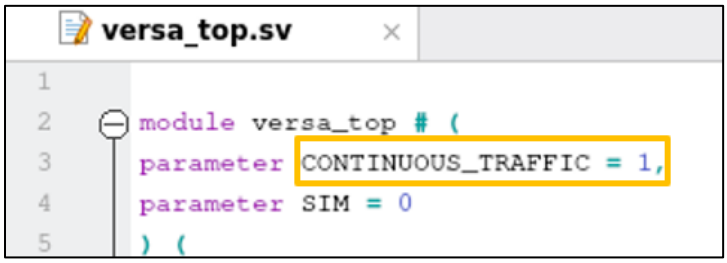

Figure 6.15. CONTINUOUS_TRAFFIC parameter for Continuous Mode in versa_top.sv File

   - Non-continuous mode:
     - Allows you to change the CONTINUOUS_TRAFFIC parameter to 0.
     - Allows you to define the number of packets transmitted through the NUM_PKT parameter.

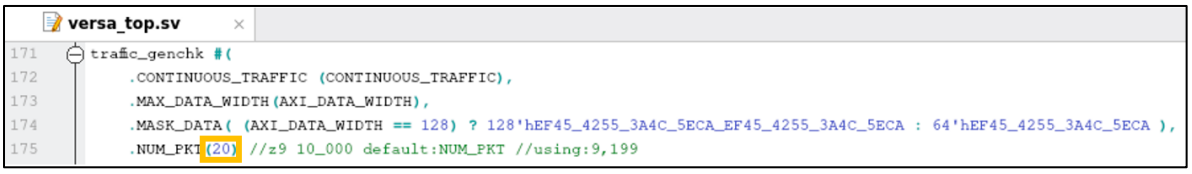

Figure 6.16. NUM_PKT for Non-Continuous Mode

3. Right-click on **Input Files**, select **Add → Existing Simulation File**…, then add the *tb_top_versa.sv* file to the Radiant software project.

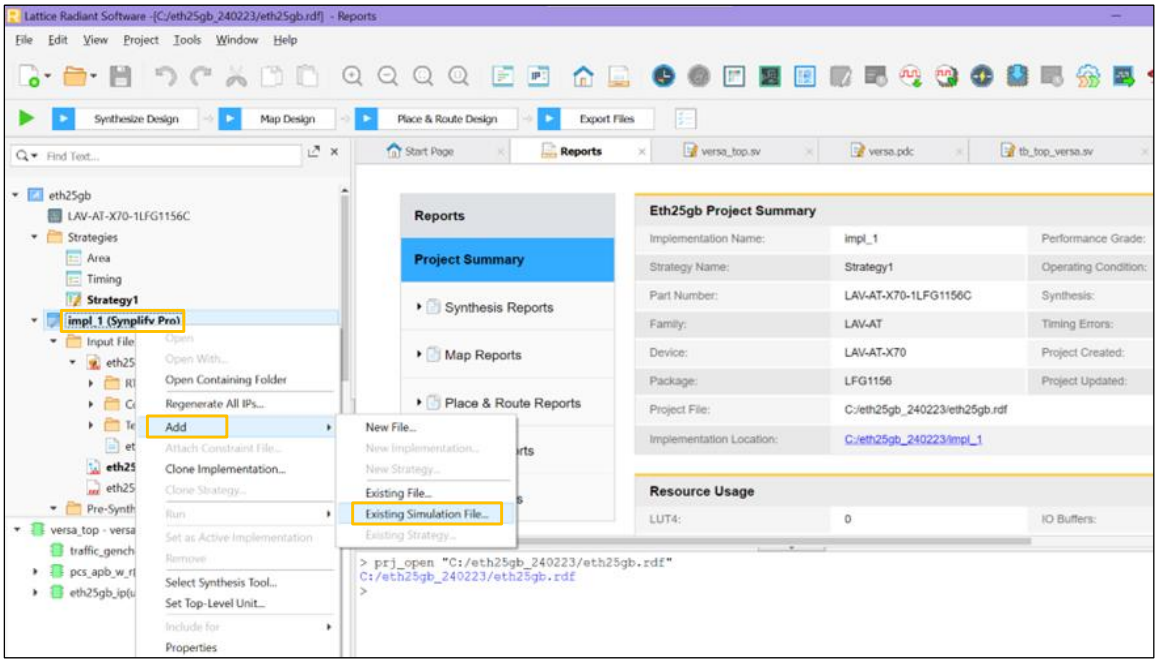

Figure 6.17. Add the tb_top_versa.sv File into the Radiant Software Project

4. Versa testbench: *tb_versa_top.sv*. The SIM parameter remains unchanged for both modes.
   - Continuous mode: CONTINUOUS_TRAFFIC parameter is default to 1.

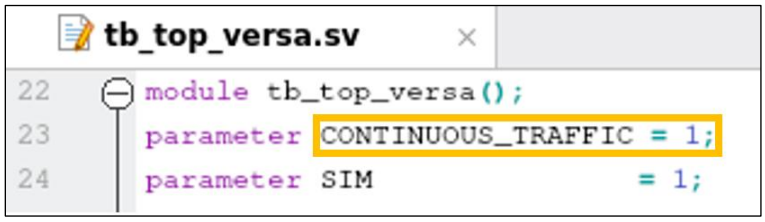

Figure 6.18. CONTINUOUS_TRAFFIC Parameter for Continuous Mode in the tb_versa_top.sv File

- Non- continuous mode: Allows you to change the CONTINUOUS_TRAFFIC parameter to 0.

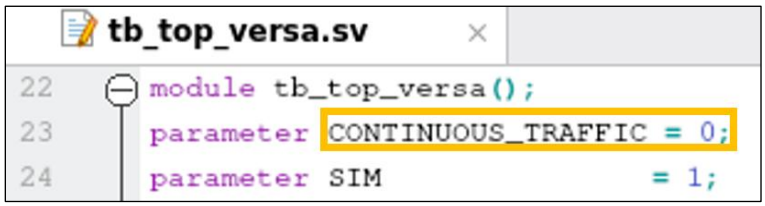

Figure 6.19. CONTINUOUS_TRAFFIC Parameter for Non-Continuous Mode in the tb_versa_top.sv File

5. Right-click on **Post-Synthesis Constraint Files**, select **Add → Existing File…,** then add the *versa.pdc* file to the Radiant software project.

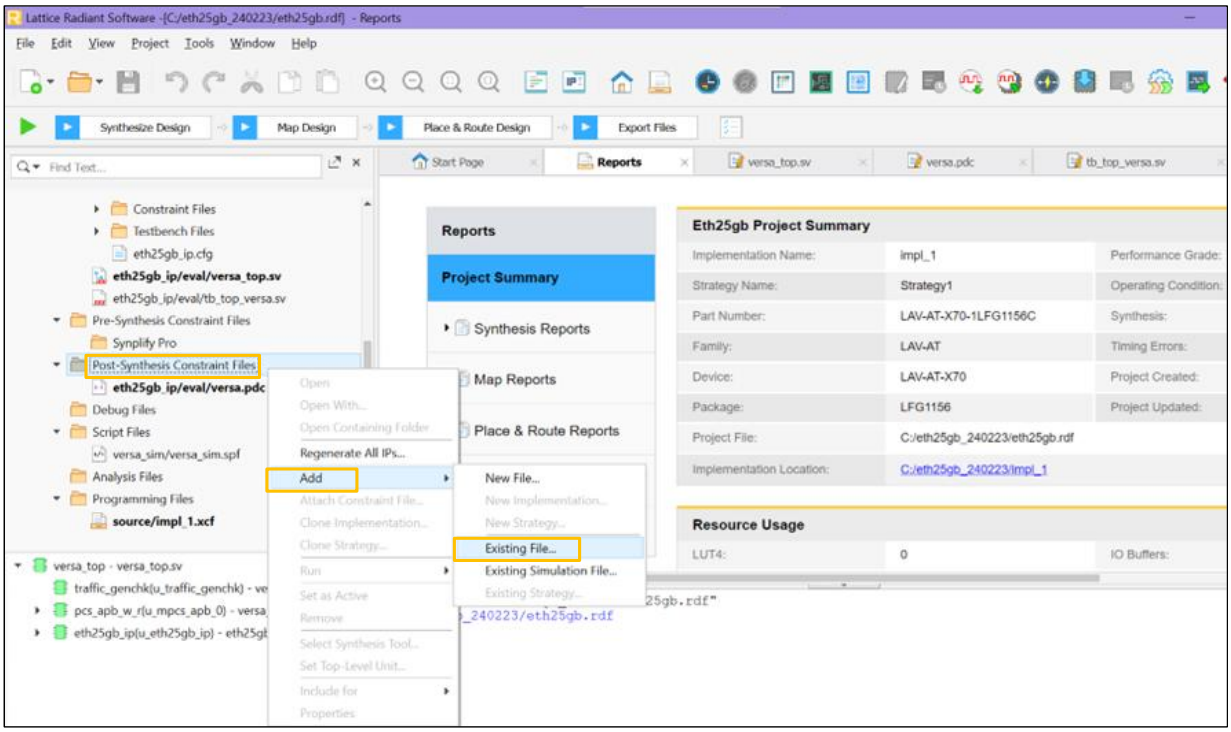

Figure 6.20. Add the versa.pdc File into the Radiant Software Project

6. Remember to turn off GSR instantiation in the *versa_top.sv* file before compilation as this may cause the board to force reset. You may skip this step if the GSR has been removed from the design.

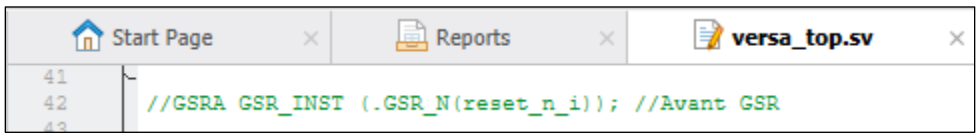

Figure 6.21. Turn Off GSR in the versa_top.sv File

7. The device must be set to Performance Grade 3 and Package of LFG1156 for versa project.

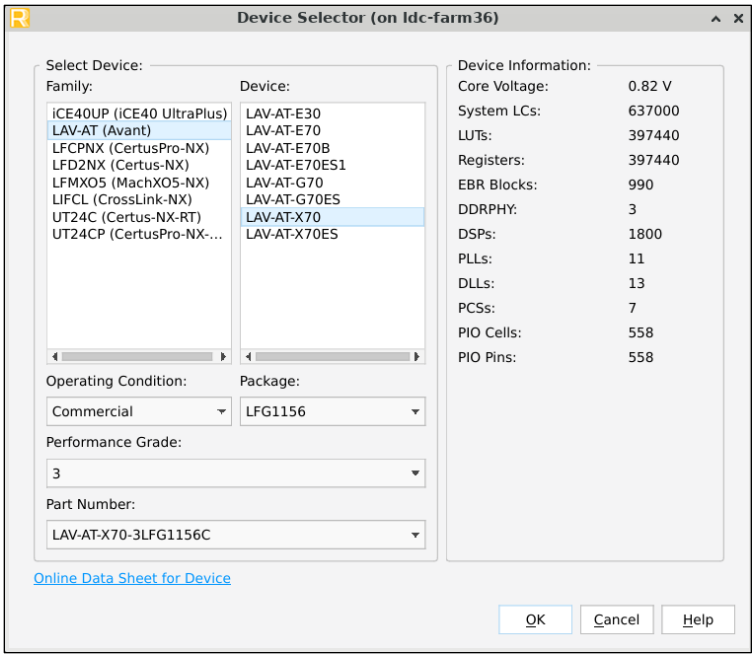

Figure 6.22. Select the Device According to the IP Version

8. The *tb_top_versa.sv* must be included for simulation only so that the *versa_top.sv* is set automatically as the top-level file for the versa project.

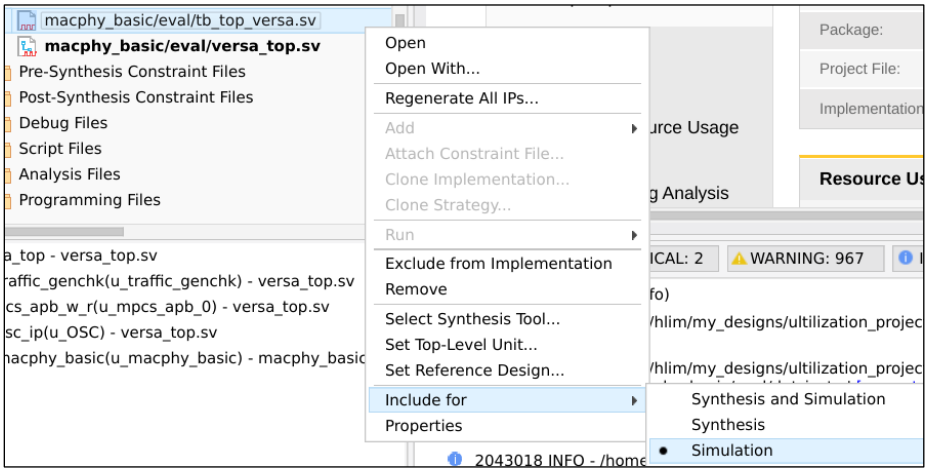

Figure 6.23. Set the tb_top_versa.sv to be include for Simulation Only

9. The Versa project preparation is completed.

6.5. Example Design Simulation

This section describes the example design testbench simulation flow, how to create a simulation project, and run the simulation.

6.5.1. 25G Ethernet IP Example Design Testbench Simulation Flow

The following figure shows the testbench simulation flow. When the simulation starts, the testbench waits for 25G Ethernet PHY link up. The AXI4-Lite Module configures the 25G Ethernet MAC to enable transmit and receive data paths. After that, the AXI-Stream Packet Generator and Loopback generates the packets at the serial interface. Based on the result of the content comparison between the transmitted and loopback packets, the simulation status—PASSED or FAILED—is displayed.

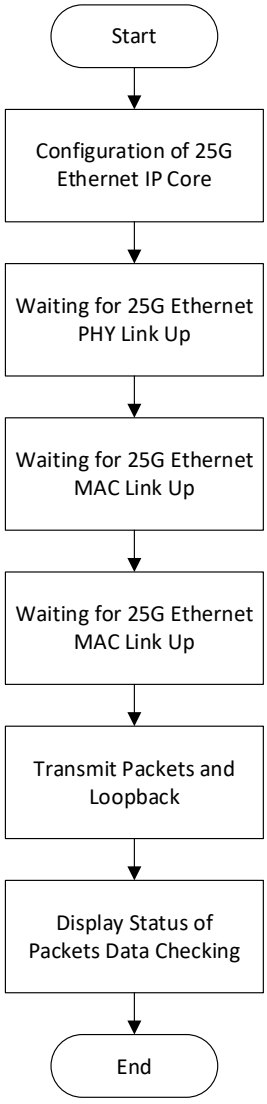

Figure 6.24. 25G Ethernet IP Example Design Testbench Flowchart

6.5.2. Create a Simulation Project

1. Go to menu **Tools → Simulation Wizard** or click on  icon to launch the Simulation Wizard GUI.
2. Enter **Project name**, and click **Next**.

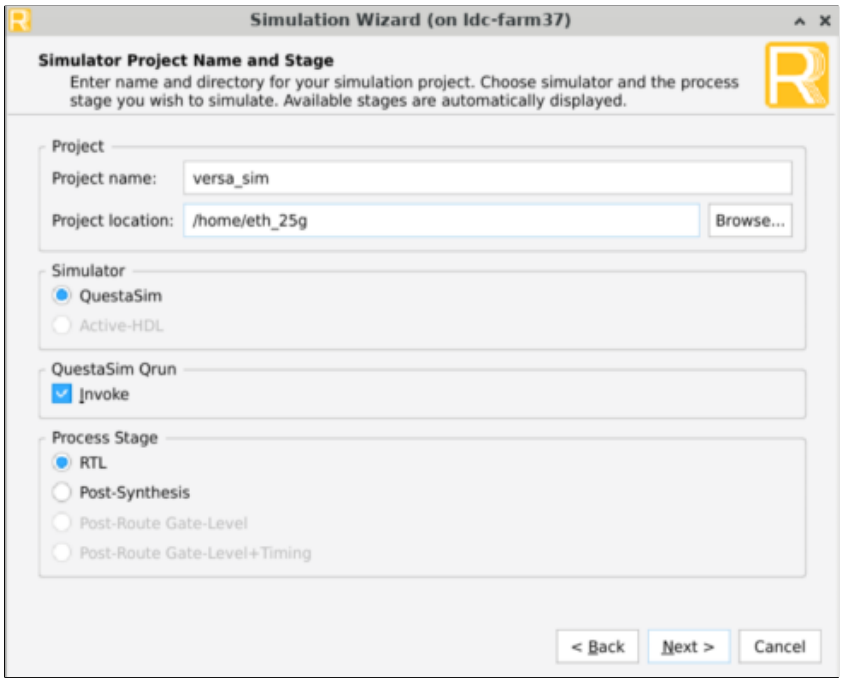

Figure 6.25. Create a Simulation Project

3. Click **Next**.

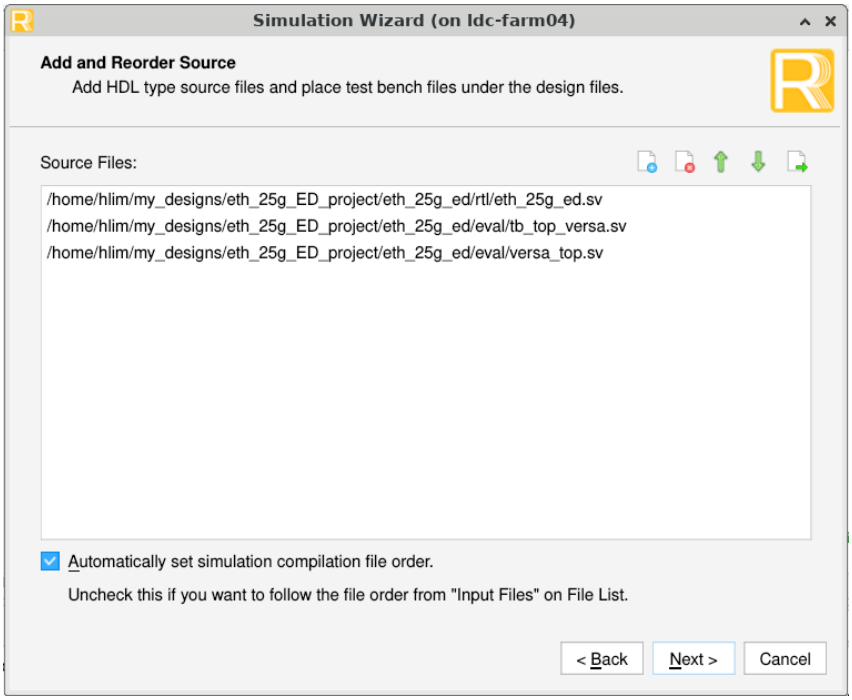

Figure 6.26. Include Source Files

4. Click **Next**.

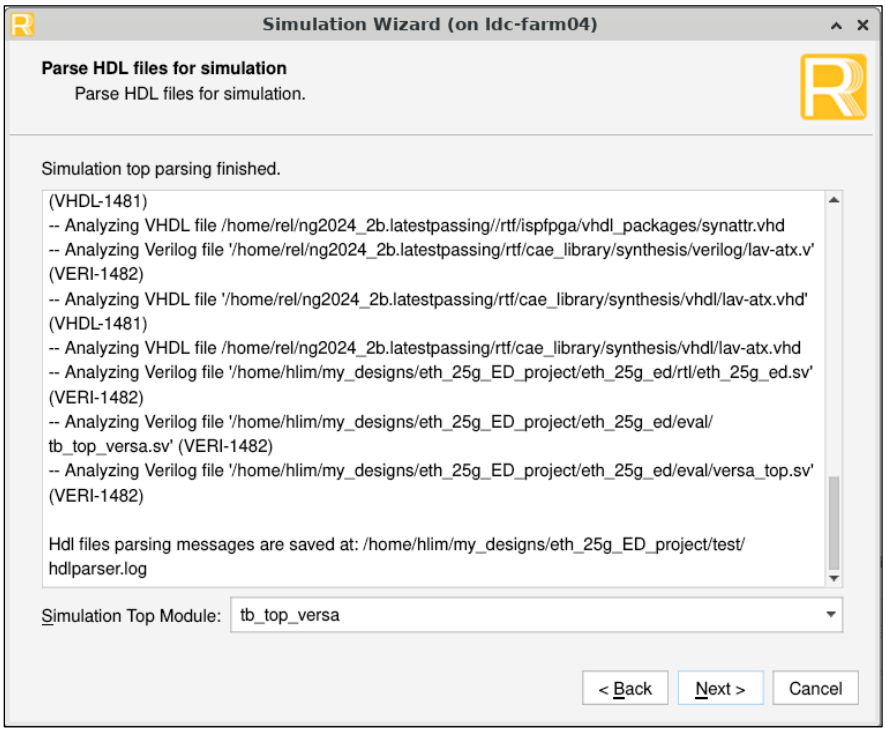

Figure 6.27. Set tb_top_versa as Top Module

5. Change the settings according to the following figure and click **Finish** to run the QuestaSim Lattice-Edition software.

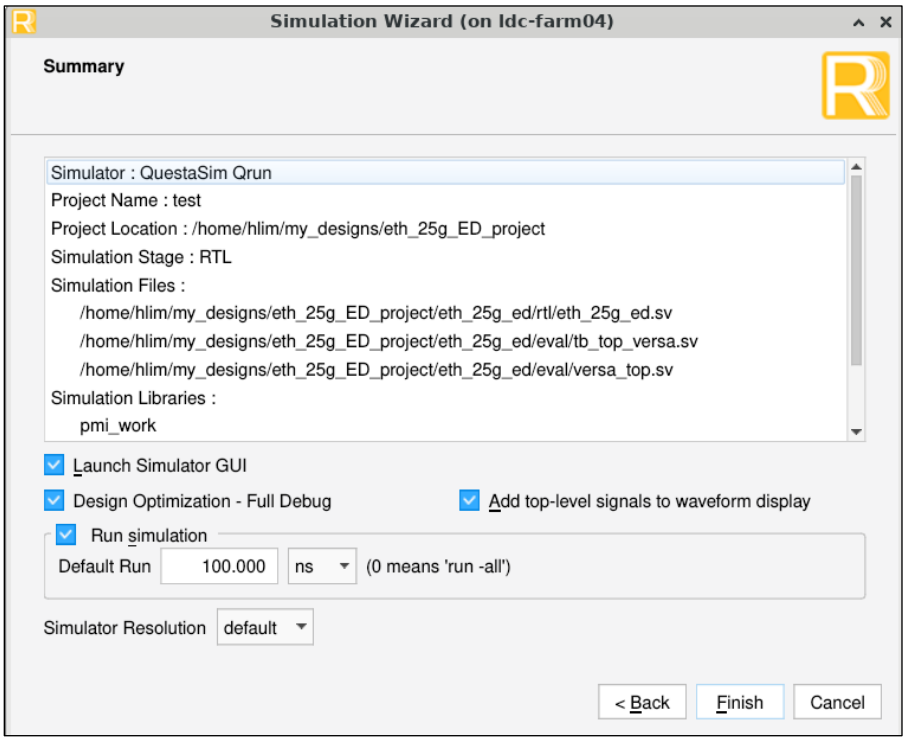

Figure 6.28. Change the Time Setting for Default Run

6.5.3. Running the Simulation

1. After you have launched the QuestaSim Lattice-Edition software and executed the simulation script setup through the Simulation Wizard, enter `log -r /*` and `radix hex` in the Transcript window.
   The `log -r /*` command allows all signals in the design to be captured so that these signals can be pulled out to the waveform window later.
   The `radix hex` command changes the default radix of the signals from binary to hex format. This enables you to view the content of the Ethernet frames.
2. Enter `run -all` to continue running the simulation until the end.

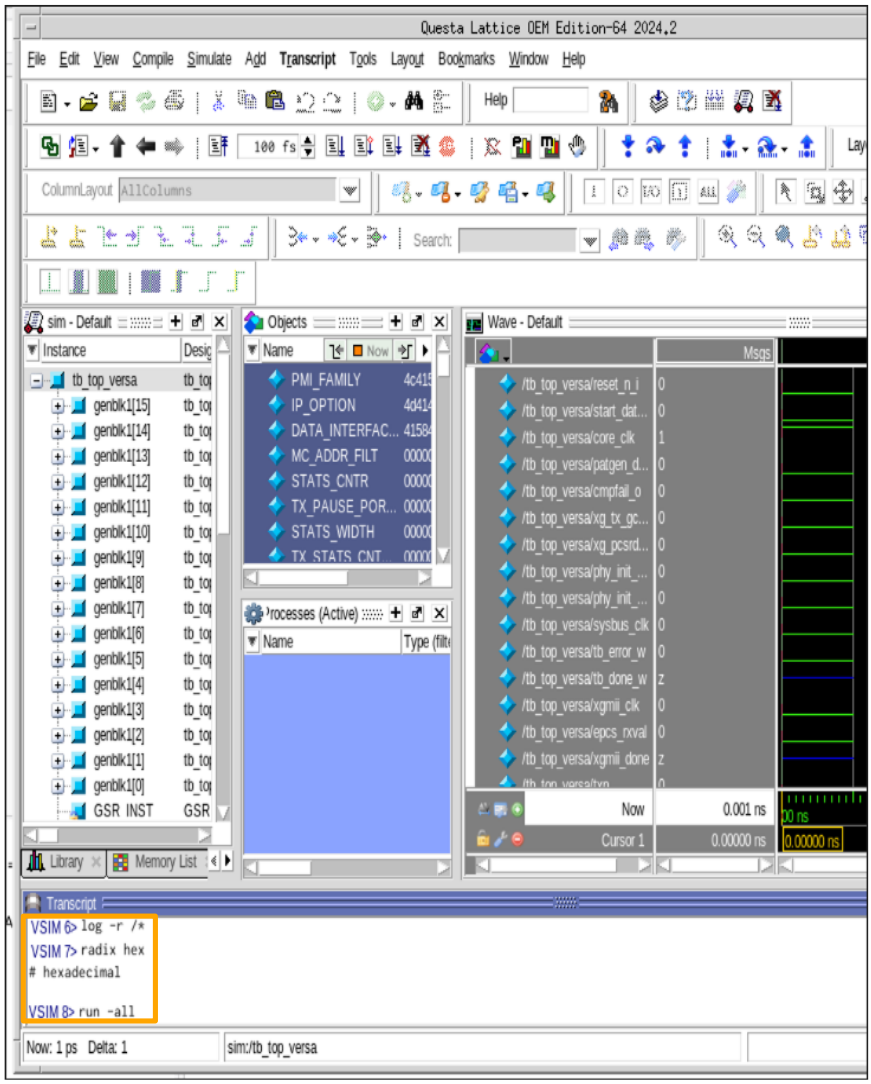

Figure 6.29. Execute Simulation Script

6.5.3.1. Continuous Mode

The following figure shows the simulation output for continuous mode with a PASSED status.

```
# Data matched: exp = 9ef687a1885b2c32b8d9e2e1885b2c32, obs = 9ef687a1885b2c32b8d9e2e1885b2c32
# Data matched: exp = 0000000000b6586571b3c5c310b65865, obs = 0000000000b6586571b3c5c310b65865
# +-----------------------------------------------
# Pause TX traffic; 3rd attempt
# +-----------------------------------------------
# Data matched: exp = d2a83e05facefacefacedeaddeaddead, obs = d2a83e05facefacefacedeaddeaddead
# Data matched: exp = a5509a2c55f40d5f3ded0f2c55f40d5f, obs = a5509a2c55f40d5f3ded0f2c55f40d5f
# Data matched: exp = 4a15d8796fb85395d2a84d796fb85395, obs = 4a15d8796fb85395d2a84d796fb85395
# Data matched: exp = 942bb0f2df70a72ba5509af2df70a72b, obs = 942bb0f2df70a72ba5509af2df70a72b
# Data matched: exp = 7b6ef2a7e53cf9e14a15d8a7e53cf9e1, obs = 7b6ef2a7e53cf9e14a15d8a7e53cf9e1
# Data matched: exp = f6dde54fca79f3c2942bb14fca79f3c2, obs = f6dde54fca79f3c2942bb14fca79f3c2
# Data matched: exp = 1998a71af035ad087b6ef31af035ad08, obs = 1998a71af035ad087b6ef31af035ad08
# Data matched: exp = 33314e35e06b5a10f6dde635e06b5a10, obs = 33314e35e06b5a10f6dde635e06b5a10
# Data matched: exp = dc740c60da2704da1998a460da2704da, obs = dc740c60da2704da1998a460da2704da
# Data matched: exp = b8e818c1b44e09b4333148c1b44e09b4, obs = b8e818c1b44e09b4333148c1b44e09b4
# Data matched: exp = 57ad5a948e02577edc740a948e02577e, obs = 57ad5a948e02577edc740a948e02577e
# Data matched: exp = af5ab5291c04aefdb8e815291c04aefd, obs = af5ab5291c04aefdb8e815291c04aefd
# Data matched: exp = 401ff77c2648f03757ad577c2648f037, obs = 401ff77c2648f03757ad577c2648f037
# Data matched: exp = 803feef84c91e06eaf5aaef84c91e06e, obs = 803feef84c91e06eaf5aaef84c91e06e
# Data matched: exp = 6f7aacad76ddbea4401fecad76ddbea4, obs = 6f7aacad76ddbea4401fecad76ddbea4
# Data matched: exp = def5595aedbb7d48803fd95aedbb7d48, obs = def5595aedbb7d48803fd95aedbb7d48
# Data matched: exp = 31b01b0fd7f723826f7a9b0fd7f72382, obs = 31b01b0fd7f723826f7a9b0fd7f72382
# Data matched: exp = 6360361fafee4704def5361fafee4704, obs = 6360361fafee4704def5361fafee4704
# Data matched: exp = 8c25744a95a219ce31b0744a95a219ce, obs = 8c25744a95a219ce31b0744a95a219ce
# Data matched: exp = 184ae8952b44339c6360e8952b44339c, obs = 184ae8952b44339c6360e8952b44339c
# Data matched: exp = f70faac011086d568c25aac011086d56, obs = f70faac011086d568c25aac011086d56
# Data matched: exp = ee1f55802210daad184b55802210daad, obs = ee1f55802210daad184b55802210daad
# Data matched: exp = 015a17d5185c8467f70e17d5185c8467, obs = 015a17d5185c8467f70e17d5185c8467
# Data matched: exp = 02b42faa30b908cfee1c2faa30b908cf, obs = 02b42faa30b908cfee1c2faa30b908cf
# Data matched: exp = edf16dff0af5560501596dff0af55605, obs = edf16dff0af5560501596dff0af55605
# Data matched: exp = dbe2dbfe15eaac0a02b2dbfe15eaac0a, obs = dbe2dbfe15eaac0a02b2dbfe15eaac0a
# Data matched: exp = 34a799ab2fa6f2c0edf799ab2fa6f2c0, obs = 34a799ab2fa6f2c0edf799ab2fa6f2c0
# Data matched: exp = 694f33565f4de581dbef33565f4de581, obs = 694f33565f4de581dbef33565f4de581
# Data matched: exp = 860a71036501bb4b34aa71036501bb4b, obs = 860a71036501bb4b34aa71036501bb4b

# Data matched: exp = 29c527fc7fbb884229c527ff05919aeb, obs = 29c527fc7fbb884229c527ff05919aeb
# Data matched: exp = c68065a945f7d688c68065aa3fddc421, obs = c68065a945f7d688c68065aa3fddc421
# +-----------------------------------------------
# Resume TX traffic; 3rd attempt
# +-----------------------------------------------
# Data matched: exp = 8d00cb528befad118d00cb547fbb8842, obs = 8d00cb528befad118d00cb547fbb8842
# Data matched: exp = 62458907b1a3f3db6245890145f7d688, obs = 62458907b1a3f3db6245890145f7d688
# Data matched: exp = c48b120f6347e7b6c48b12028befad11, obs = c48b120f6347e7b6c48b12028befad11
# Data matched: exp = 2bce505a590bb97c2bce5057b1a3f3db, obs = 2bce505a590bb97c2bce5057b1a3f3db
# Data matched: exp = 579ca0b4b21772f8579ca0af6347e7b6, obs = 579ca0b4b21772f8579ca0af6347e7b6
# Data matched: exp = b8d9e2e1885b2c32b8d9e2fa590bb97c, obs = b8d9e2e1885b2c32b8d9e2fa590bb97c
# Data matched: exp = 71b3c5c310b6586571b3c5f4b21772f8, obs = 71b3c5c310b6586571b3c5f4b21772f8
# Data matched: exp = 9ef687962afa06af9ef687a1885b2c32, obs = 9ef687962afa06af9ef687a1885b2c32
# Data matched: exp = 3ded0f2c55f40d5f3ded0f4310b65865, obs = 3ded0f2c55f40d5f3ded0f4310b65865
# Data matched: exp = d2a84d796fb85395d2a84d162afa06af, obs = d2a84d796fb85395d2a84d162afa06af
# Data matched: exp = a5509af2df70a72ba5509a2c55f40d5f, obs = a5509af2df70a72ba5509a2c55f40d5f
# Data matched: exp = 4a15d8a7e53cf9e14a15d8796fb85395, obs = 4a15d8a7e53cf9e14a15d8796fb85395
# Data matched: exp = 942bb14fca79f3c2942bb0f2df70a72b, obs = 942bb14fca79f3c2942bb0f2df70a72b
# Data matched: exp = 7b6ef31af035ad087b6ef2a7e53cf9e1, obs = 7b6ef31af035ad087b6ef2a7e53cf9e1
# Data matched: exp = f6dde635e06b5a10f6dde54fca79f3c2, obs = f6dde635e06b5a10f6dde54fca79f3c2
# Data matched: exp = 1998a460da2704da1998a71af035ad08, obs = 1998a460da2704da1998a71af035ad08
# Data matched: exp = 333148c1b44e09b433314e35e06b5a10, obs = 333148c1b44e09b433314e35e06b5a10
# Data matched: exp = dc740a948e02577edc740c60da2704da, obs = dc740a948e02577edc740c60da2704da
# Data matched: exp = b8e81529lc04aefdb8e818c1b44e09b4, obs = b8e815291c04aefdb8e818c1b44e09b4
# Data matched: exp = 57ad577c2648f03757ad5a948e02577e, obs = 57ad577c2648f03757ad5a948e02577e
# Data matched: exp = af5aaef84c91e06eaf5ab5291c04aefd, obs = af5aaef84c91e06eaf5ab5291c04aefd
# Data matched: exp = 401fecad76ddbea4401ff77c2648f037, obs = 401fecad76ddbea4401ff77c2648f037
# Data matched: exp = 803fd95aedbb7d48803feef84c91e06e, obs = 803fd95aedbb7d48803feef84c91e06e
# Data matched: exp = 6f7a9b0fd7f723826f7aacad76ddbea4, obs = 6f7a9b0fd7f723826f7aacad76ddbea4
# Data matched: exp = def5361fafee4704def5595aedbb7d48, obs = def5361fafee4704def5595aedbb7d48
# Data matched: exp = 31b0744a95a219ce31b01b0fd7f72382, obs = 31b0744a95a219ce31b01b0fd7f72382
# Data matched: exp = 6360e8952b44339c6360361fafee4704, obs = 6360e8952b44339c6360361fafee4704
# Data matched: exp = 8c25aac011086d568c25744a95a219ce, obs = 8c25aac011086d568c25744a95a219ce
# Data matched: exp = 184b55802210daad184ae8952b44339c, obs = 184b55802210daad184ae8952b44339c
# Data matched: exp = f70e17d5185c8467f70faac011086d56, obs = f70e17d5185c8467f70faac011086d56
# Data matched: exp = ee1c2faa30b908cfee1f55802210daad, obs = ee1c2faa30b908cfee1f55802210daad
# Data matched: exp = 01596dff0af55605015a17d5185c8467, obs = 01596dff0af55605015a17d5185c8467
# Data matched: exp = 02b2dbfe15eaac0a02b42faa30b908cf, obs = 02b2dbfe15eaac0a02b42faa30b908cf
# Data matched: exp = edf799ab2fa6f2c0edf16dff0af55605, obs = edf799ab2fa6f2c0edf16dff0af55605
# Data matched: exp = dbef33565f4de581dbe2dbfe15eaac0a, obs = dbef33565f4de581dbe2dbfe15eaac0a
# Data matched: exp = 34aa71036501bb4b34a799ab2fa6f2c0, obs = 34aa71036501bb4b34a799ab2fa6f2c0
# Data matched: exp = 6954e206ca037696694f33565f4de581, obs = 6954e206ca037696694f33565f4de581
# Data matched: exp = 8611a053f04f285c860a71036501bb4b, obs = 8611a053f04f285c860a71036501bb4b
# Data matched: exp = 0c2140a7e09e50b90c14e206ca017696, obs = 0c2140a7e09e50b90c14e206ca017696

# Data matched: exp = e64a60a8b034cf159c60720162563941, obs = e64a60a8b034cf159c60720162563941
# Data matched: exp = 0000000000000000000000000000008b, obs = 0000000000000000000000000000008b
# [         17336500000]:[Normal ]---[tb_top_versa.genblk1]--- +-----------------------------------------------
# [         17336500000]:[Normal ]---[tb_top_versa.genblk1]--- Transaction Check Done
# [         17336500000]:[Normal ]---[tb_top_versa.genblk1]--- +-----------------------------------------------
# [         17336500000]:[Normal ]---[tb_top_versa.genblk1]--- ********** DATA TRANSACTION PASSED **********
# [         17336500000]:[Normal ]---[tb_top_versa.genblk1]--- +-----------------------------------------------
# [         17336500000]:[Normal ]---[tb_top_versa.genblk1]--- +-----------------------------------------------
# [         17336500000]:[Normal ]---[tb_top_versa.genblk1]---             SIMULATION PASSED
```

Figure 6.30. Example Design Simulation Output for Continuous Mode

The following waveform shows an example of a simulation result for continuous mode.

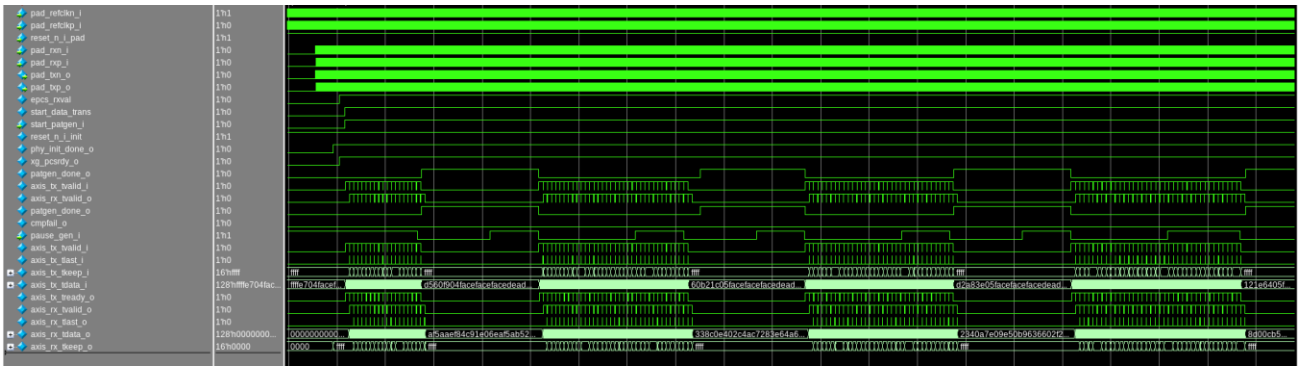

Figure 6.31. Example Design Simulation Waveform for Continuous Mode

6.5.3.2. Non-Continuous Mode

1. The Transcript window displays the testbench parameters.
2. Ethernet frame transmission takes place when the PCS is ready.
3. The Transcript window displays the expected frame content along with the comparison result.
4. The simulation ends when all transactions are completed.
5. The Transcript window shows a passing status of the simulation if there are no mismatch in the frame content.

Figure 6.32. Example Design Simulation Output for Non-Continuous Mode

6. From the waveform of the added 25G Ethernet IP signals, you can observe the following:
   - IP initialization using AXI4-Lite interface.
   - 20 Ethernet frames are transmitted and received in both the TX and RX data paths.

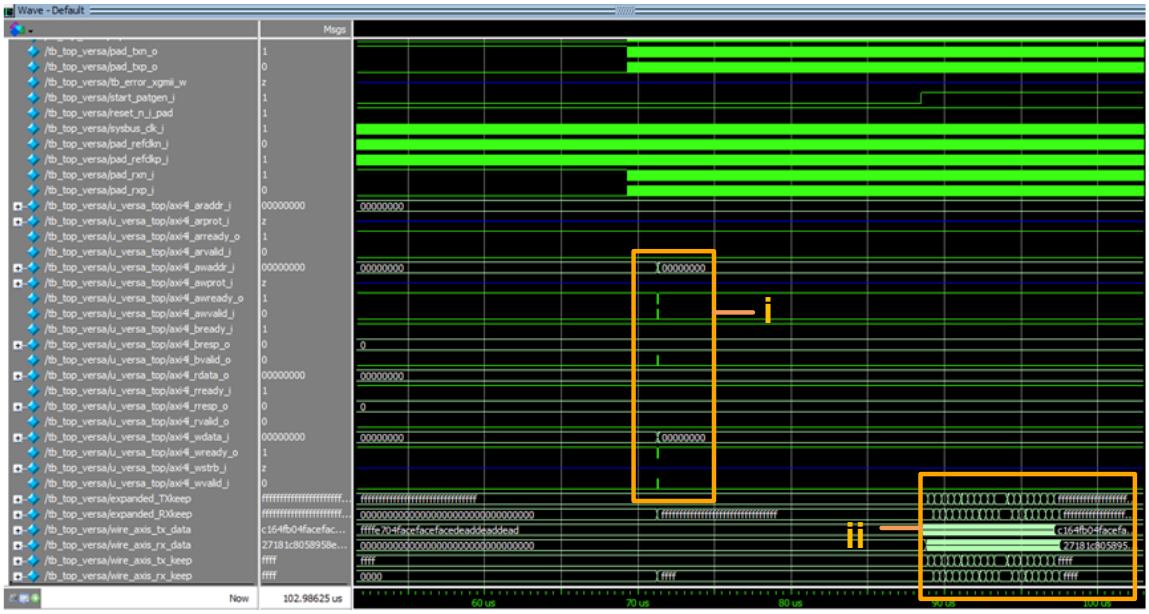

Figure 6.33. Example Design Simulation Waveform for Non-Continuous Mode

6.6. Hardware Testing

The 25G Ethernet IP is hardware tested on the Avant X versa boards.

The following steps demonstrate how to set up the board for hardware testing:

1. Set up the board as shown in the following figure.
   Ensure the input power is 12 V.
2. Connect the JP17 jumper to enable the board to operate in 25 Gb.

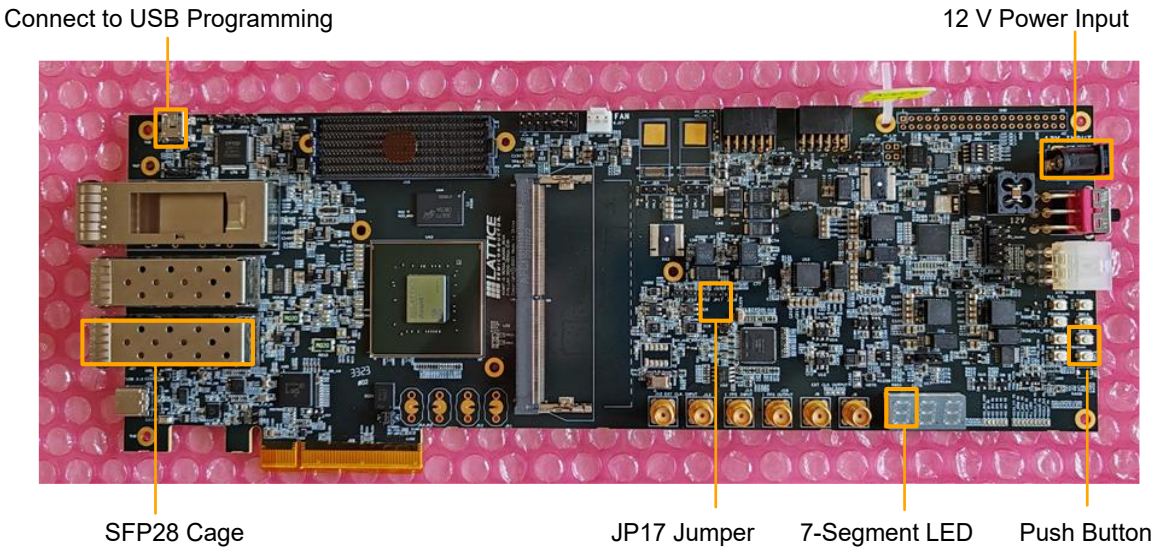

Figure 6.34. Avant-X Board Setup

3. The 25 Gb SFP28 optical module and optical loopback module and the connection to the Avant-X Versa board SFP1 are shown in the following figure.

Figure 6.35. Loopback Module Setup

4. Click on the Run button shown in the following figure to compile the design until it generates the bitstream file.

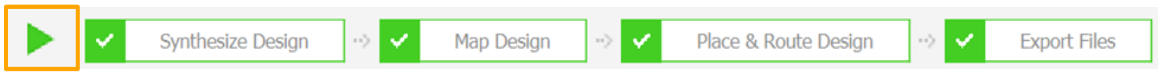

Figure 6.36. Successful Design Compilation

5. Launch the Radiant Programmer via menu **Tools→ Programmer** or click on the Radiant Programmer icon .
6. Click on the **Detect Cable** button.
7. Select the correct cable associated with the Avant-X Versa board. The number of cables shown in your environment setup may be different.
8. Select the correct bitstream file that is generated from the 25G Ethernet project in step 5.
9. Click on the programming icon . The software displays the programming status.

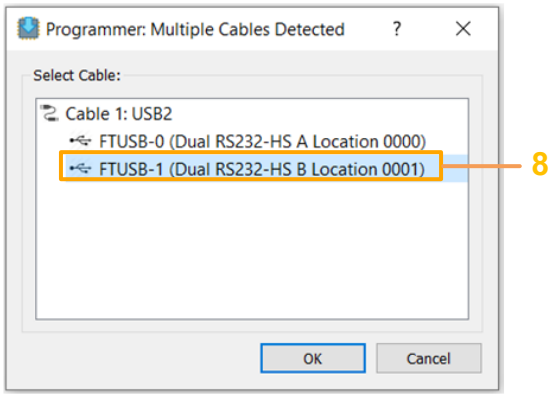

Figure 6.37. Cable Selection for Avant-AT-X Device

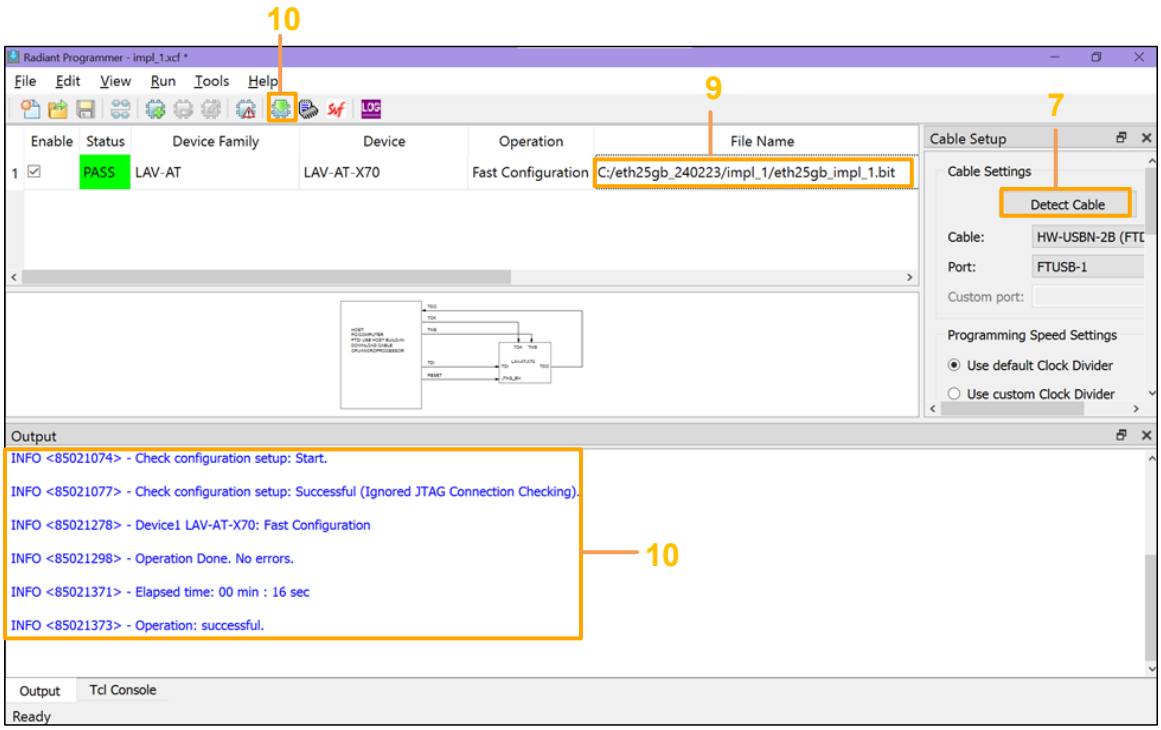

Figure 6.38. Programmer GUI

10. After the programming stage, press the pushbutton SW12 to trigger a forced reset even though the example design will automatically trigger a reset.
11. After the design has linked up, toggle DIP_SW_1 to start Ethernet frames generation and transmission. The Ethernet frames loopback via the connected SFP with the optical loopback module and the traffic generator receives and compares the frames.
12. You can pause and resume the Ethernet frames transmission by pressing the pushbutton SW13*. The 7-segment LED glows according to the following diagram, which indicates that the hardware has been successfully set up and received a passing status.

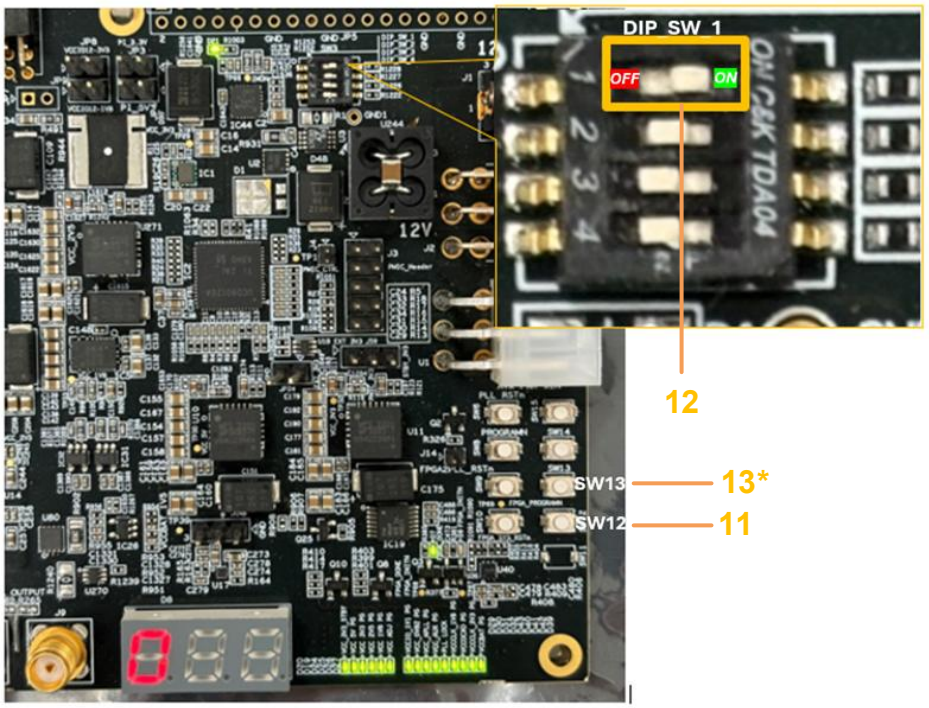

Figure 6.39. Passing Scenario on Avant-AT-X Device

* **Note:** Applicable if continuous mode is enabled (CONTINUOUS_TRAFFIC=1).

13. In the failing scenario due to various reasons (for example, link is not stable, loopback module is not connected, and so on), the 7-segment LED glows according to the following diagram. Check the connectivity, re-flash the bitstream, or power cycle to resolve it.

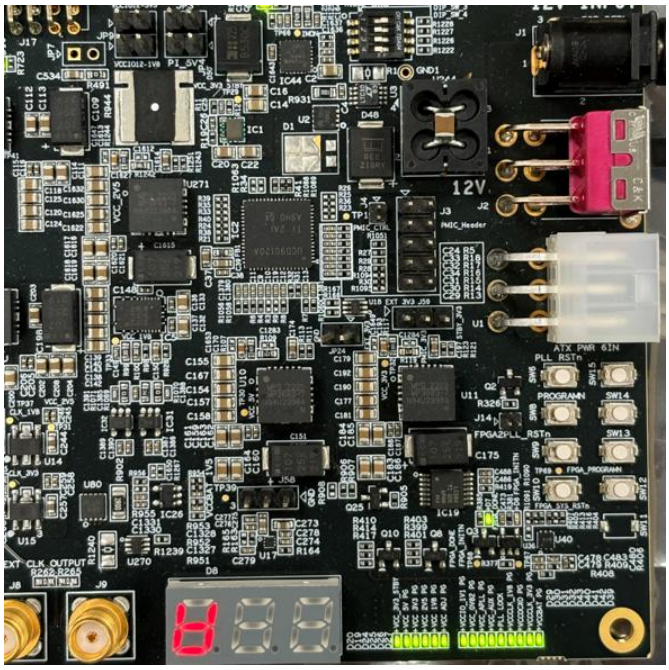

Figure 6.40. Failing Scenario on Avant-AT-X Device

14. The following diagram shows the various LED segments and the description for each segments are described in the following table.

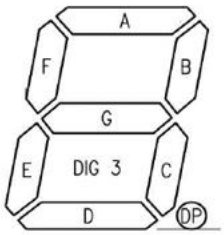

Figure 6.41. LED 7-Segment

Table 6.5. LED 7-Segment Description

Segment	Description
A	Traffic generator ended, or frames transmission paused*.
B	Reset (push button) deasserted.
C	25G Ethernet IP initialization is done.
D	25G Ethernet IP PCS is ready.
E	Frames transmitting or traffic generator ended.
F	Frames receiving or traffic generator ended.
G	Loopback patterns comparison fails.

*** Note**: For default continuous mode and if the SW13 is pressed, Segment A of the 7-segment LED indicates that the frames transmission is paused. For non-continuous mode, Segment A of the 7-segment LED indicates that the traffic generator has ended.

7. Designing with the IP

This section describes how to generate the IP core using the Lattice Radiant software and how to run simulation and synthesis. For more details on the Lattice Radiant software, refer to the Lattice Radiant Software User Guide and relevant Lattice tutorials.

Note: The screenshots provided are for reference only. Details may vary depending on the version of the IP or software being used. If there have been no significant changes to the GUI, a screenshot may reflect an earlier version of the IP.

7.1. Generating and Instantiating the IP

You can use the Lattice Radiant software to generate IP modules and integrate them into the device's architecture. The following steps describe how to generate the 25G Ethernet IP core in the Lattice Radiant software:

1. Create a new Lattice Radiant software project or open an existing project.
2. In the **IP Catalog** tab, double-click **25 Gb Ethernet** under **IP, Connectivity** category. The **Module/IP Block Wizard** opens as shown in the following figure. Enter values in the **Component name** and the **Create in** fields and click **Next**.

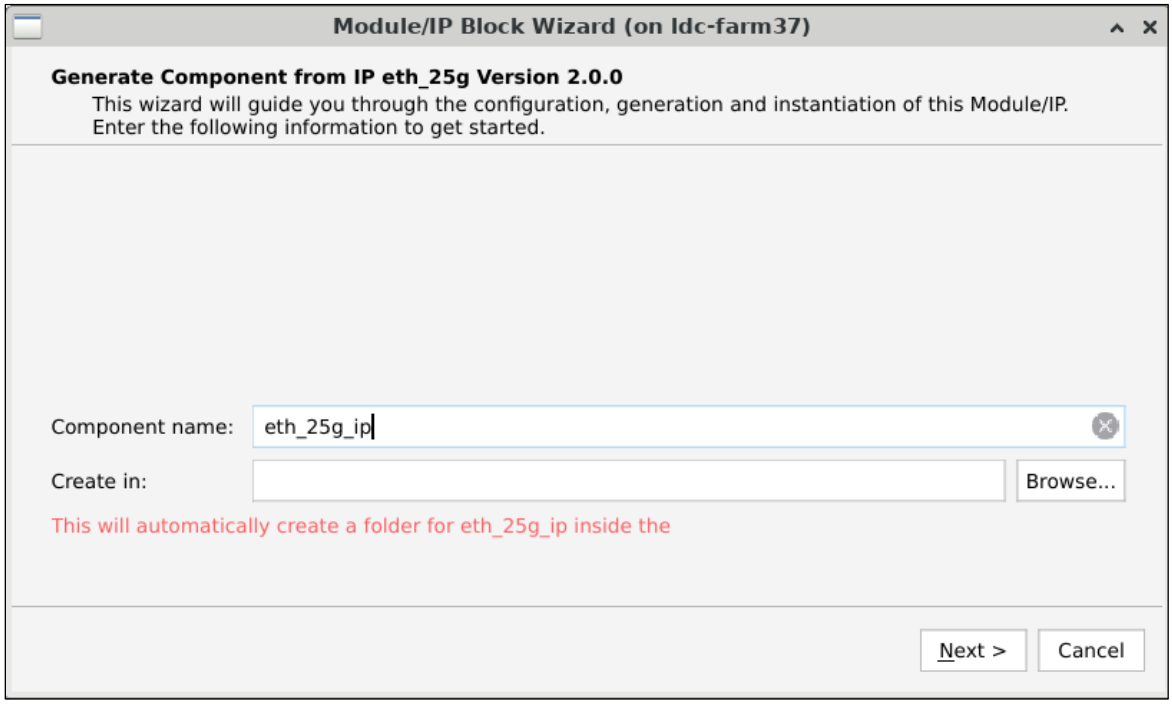

Figure 7.1. Module/IP Block Wizard

3. In the next **Module/IP Block Wizard** window, customize the selected 25G Ethernet IP core using the drop-down lists and check boxes. Figure 7.2 shows a configuration example of the 25G Ethernet IP core. For details on the configuration options, refer to the relevant *IP Parameter Description* section.

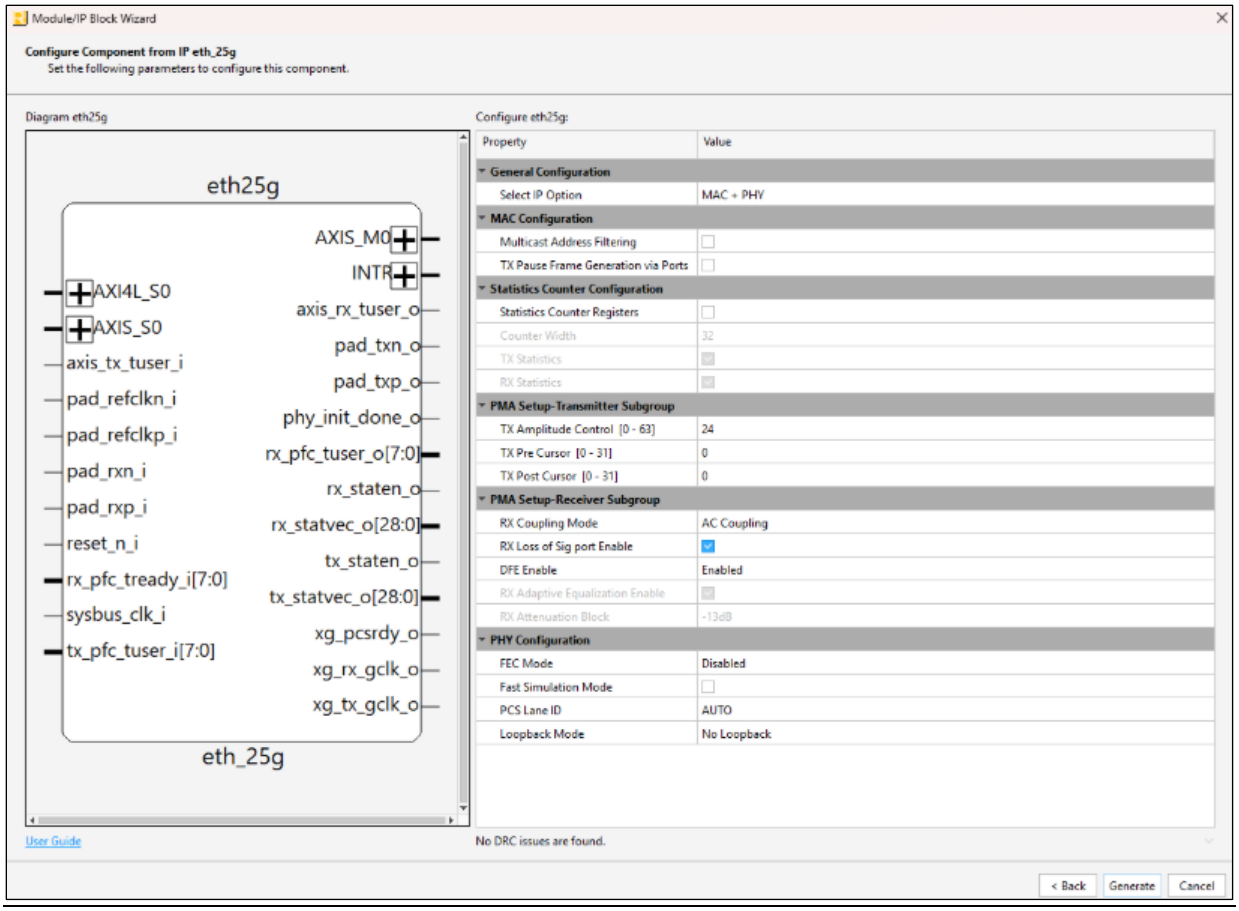

Figure 7.2. Configure the User Interface of 25G Ethernet IP Core

4. Click **Generate**. The **Check Generated Result** dialog box opens with design block messages and results as shown in Figure 7.3.

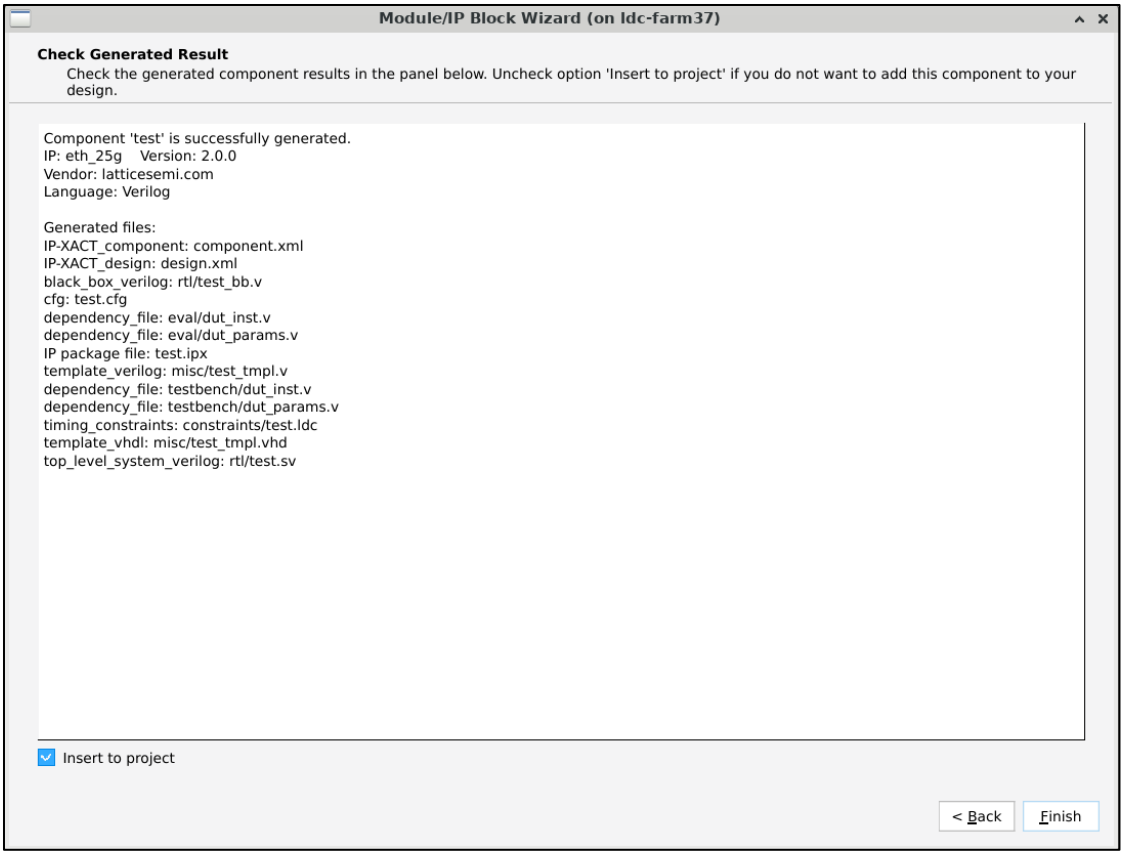

Figure 7.3. Check Generated Result

5. Click **Finish**. All the generated files are placed under the directory paths in the **Create in** and the **Component name** fields, as shown in Figure 7.1.
6. You can synthesize your generated design by clicking on **Synthesize Design** located in the top-left corner of the screen, as shown in Figure 7.4.

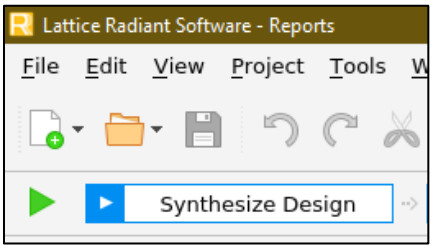

Figure 7.4. Synthesizing Design

7.1.1. Generated Files and File Structure

The generated 25G Ethernet IP core package includes the closed box (*<Instance Name>_bb.v*). An RTL example of the top-level reference source file (*<Instance Name>.v*) that can be used as an instantiation template for the IP core is also provided. You may also use this example as the starting template for your top-level design.

Table 7.1. Generated File List

Generated File	Description
<Instance Name>.ipx	This file contains the information on the files associated with the generated IP.
<Instance Name>.cfg	This file contains the parameter values used in IP configuration.
component.xml	This file contains the ipxact:component information of the IP.
design.xml	This file documents the configuration parameters of the IP in IP-XACT 2014 format.
rtl/<Instance Name>.v	This file provides an RTL example of the top-level source file that instantiates the IP core.
rtl/<Instance Name>_bb.v	This file provides the synthesis closed-box.
eval/README.txt	This file contains instructions to use Eval and Versa example design files of this directory.
eval/eval.do	This file is used to execute sub-system (Generated *<Instance Name>.v* with 25Gb PCS IP sample design) simulation. Only applicable for the *MAC + PHY* option.
eval/dut_inst.v	This file contains instantiation of IP based on selected configuration of the 25G Ethernet IP. Included by *eval_top.sv* and *versa_top.sv*.
eval/dut_params.sv	This file contains IP parameters based on selected configuration of the 25G Ethernet IP. Included by *tb_top_eval.sv* and *tb_top_versa.sv*.
eval/eval.pdc	Post-synthesis constraints for Eval example design.
eval/eval_top.sv	Top-level design file for the Eval example design.
eval/tb_top_eval.sv	Testbench for the Eval example design.
eval/tb_top_versa.sv	Testbench for the Versa example design.
eval/versa.pdc	Post-synthesis constraints for the Versa example design.
eval/versa_top.sv	Top-level design file for the Versa example design.
testbench/tb_top.v	Top testbench to run loopback test of the generated *<Instance Name>.v* file.

7.2. Design Implementation

Completing your design includes additional steps to specify analog properties, pin assignments, and timing and physical constraints. You can add and edit the constraints using the Device Constraint Editor or by manually creating a PDC file.

Post-Synthesis constraint files (*.pdc*) contain both timing and non-timing *constraint.pdc* source files for storing logical timing/physical constraints. Constraints that are added using the Device Constraint Editor are saved to the active *.pdc* file. The active post-synthesis design constraint file is then used as input for post-synthesis processes.

For more information on how to create or edit constraints and how to use the Device Constraint Editor, refer to the relevant sections in the Lattice Radiant Software User Guide.

7.3. Timing Constraints

The timing constraints are based on the clock frequency used. The timing constraints for the IP are defined in relevant constraint files. The following example shows the IP timing constraints generated for the 25G Ethernet IP core.

The content of the file should be used as reference to constrain the IP only, you must modify the constraints according to system level implementation. For example, the default clock period of sysbus_clk_i is 50 ns (20 MHz), if this clock is driven by 25 MHz clock in your project, the clock period of the constraint must be changed to 40 ns.

You may include the following clock constraints in your design per IP options configuration. Note that port names of these constraints might need to be modified according to RTL port declaration. Alternatively, you can find the following IP options based on constraints at the following path *<IP gen path>/constraints/<IP gen name>.ldc*.

7.3.1. create_clock Constraints for MAC Only Option

```
##Management module clock
create_clock -name {sysbus_clk} -period 8 [get_ports sysbus_clk_i]
##RX MAC clock
create_clock -name {rxmac_clk} -period 5.12 [get_ports rxmac_clk_i]
##TX MAC clock
create_clock -name {txmac_clk} -period 5.12 [get_ports txmac_clk_i]
```

7.3.2. create_clock Constraints for MAC + PHY Option

```
##Management module clock
create_clock -name {sysbus_clk} -period 8 [get_ports sysbus_clk_i]
##RX MAC clock
create_clock -name {rxmac_clk} -period 5.12 [get_ports rxmac_clk_i]
##TX MAC clock
create_clock -name {txmac_clk} -period 5.12 [get_ports txmac_clk_i]
create_clock -name {pad_refclkn_i} -period 6.4 [get_ports pad_refclkn_i]
create_clock -name {pad_refclkp_i} -period 6.4 [get_ports pad_refclkp_i]
create_clock -name {xgmii_clk} -period 5.12 [get_nets xg_tx_gclk_o]
create_clock -name {mpprx_clk} -period 5.12 [get_nets xg_rx_gclk_o]
```

7.3.3. create_clock Constraints for PHY Only Option

```
##Management module clock
create_clock -name {sysbus_clk} -period 8 [get_ports sysbus_clk_i]
##RX MAC clock
create_clock -name {rxmac_clk} -period 5.12 [get_ports xg_rx_clk_i]
##TX MAC clock
create_clock -name {txmac_clk} -period 5.12 [get_ports xg_tx_clk_i]

create_clock -name {pad_refclkn_i} -period 6.4 [get_ports pad_refclkn_i]
create_clock -name {pad_refclkp_i} -period 6.4 [get_ports pad_refclkp_i]
create_clock -name {xgmii_clk} -period 5.12 [get_nets xg_tx_gclk_o]
create_clock -name {mpprx_clk} -period 5.12 [get_nets xg_rx_gclk_o]
```

7.4. Specifying the Strategy

The Radiant software provides two predefined strategies—Area and Timing. It also enables you to create customized strategies. For details on how to create a new strategy, refer to the *Strategies* section of the Lattice Radiant Software User Guide.

7.5. Running Functional Simulation

You can run functional simulation after the IP is generated.

Note: To run the standalone QuestaSim simulator, the FOUNDRY environment variable must be set as follow:
set ::env(FOUNDRY) <Radiant install path>/ispfpga

To run the functional simulation, follow these steps:

1. Click the  icon located on the **Toolbar** to initiate the **Simulation Wizard**.

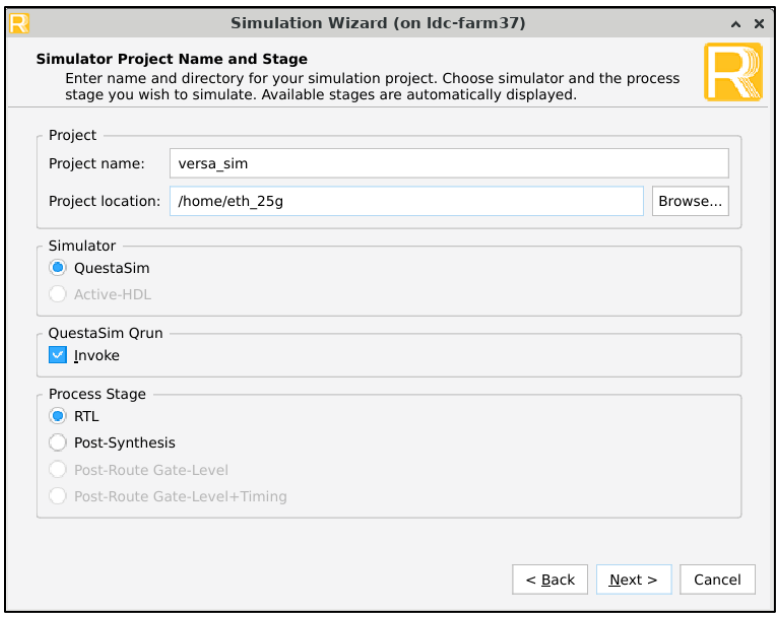

Figure 7.5. Simulation Wizard

2. Click **Next** to open the **Add and Reorder Source** window.

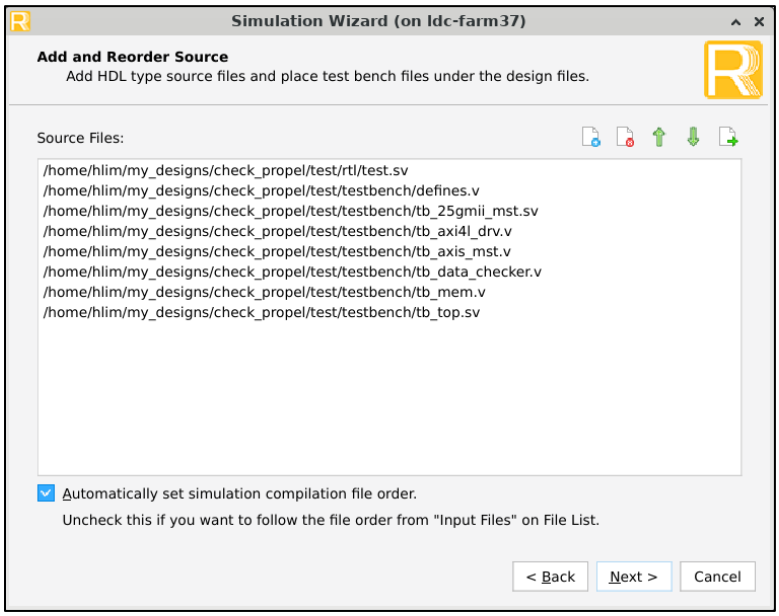

Figure 7.6. Add and Reorder Source

3. Click **Next**. The **Summary** window displays.
4. Click **Finish** to run the simulation.

   Note: It is necessary to follow the procedure above until it is fully automated in the Lattice Radiant Software Suite.

The following figure shows an example of the simulation result.

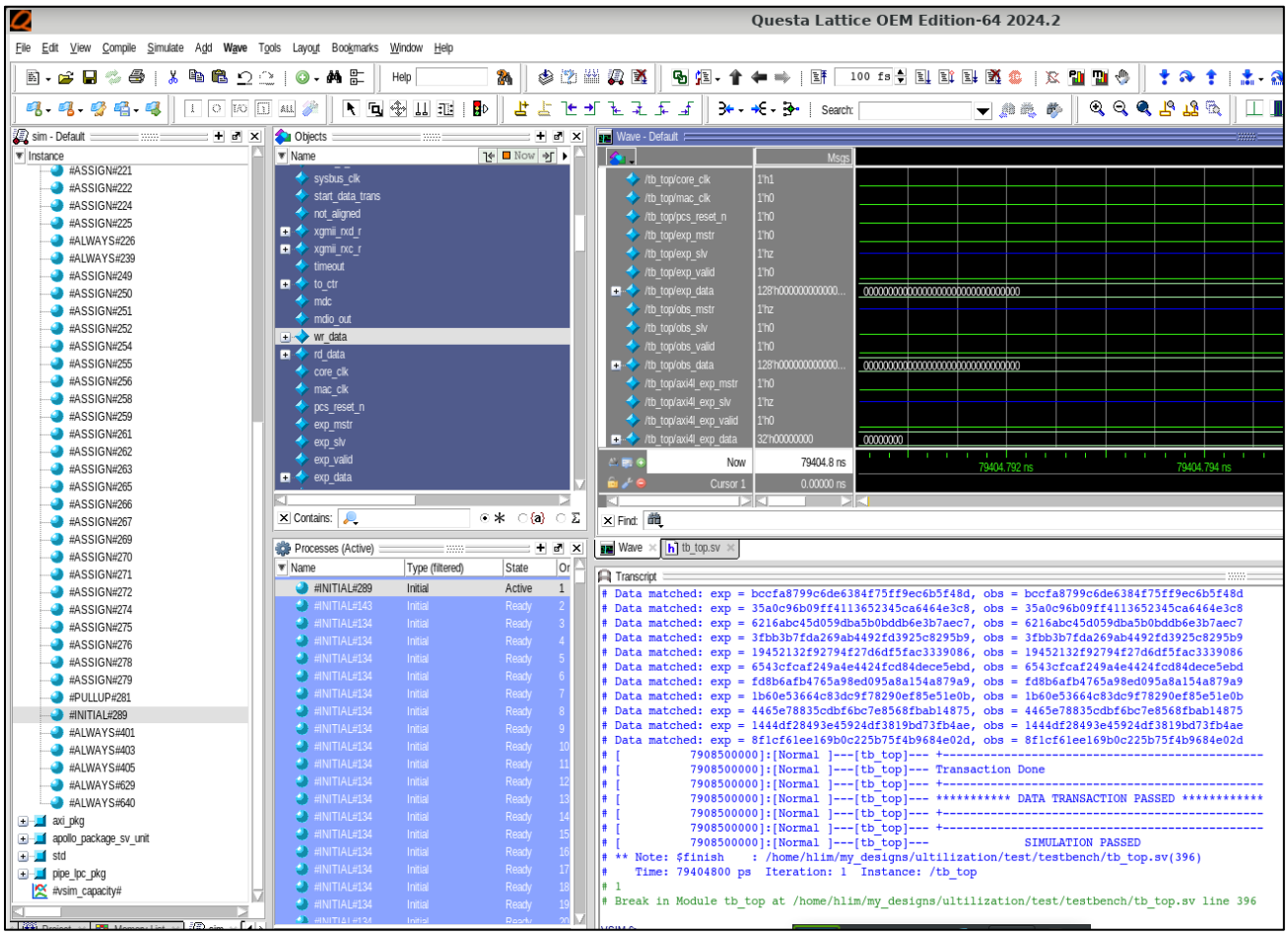

Figure 7.7. Simulation Result

Appendix A. Resource Utilization

Table A.1. Resource Utilization for LAV-AT-X50 Devices

Configuration	Registers	LUTs	EBRs	Target Device	Synthesis Tools
PHY only Host Interface == AXI4-Lite	263	824	5/630	LAV-AT-X50	Synplify Pro
MAC only (basic) Host Interface == AXI4-Lite	8,184	11,977	4/630	LAV-AT-X50	Synplify Pro
MAC only (full featured) Host Interface == AXI4-Lite	12,469	15,833	5/630	LAV-AT-X50	Synplify Pro
MAC (basic) + PHY Host Interface == AXI4-Lite	8,452	12,229	5/630	LAV-AT-X50	Synplify Pro
MAC (full featured) + PHY Host Interface == AXI4-Lite	12,737	16,106	5/630	LAV-AT-X50	Synplify Pro

Table A.2. Resource Utilization for LAV-AT-X70 Devices

Configuration	Registers	LUTs	EBRs	Target Device	Synthesis Tools
PHY only(basic) Host Interface == AXI4-Lite	130	619	5/990	LAV-AT-X70	Synplify Pro
PHY only (full_featured) Host Interface == AXI4-Lite	203	618	5/990	LAV-AT-X70	Synplify Pro
MAC only (basic) Host Interface == AXI4-Lite	8,183	10,977	5/990	LAV-AT-X70	Synplify Pro
MAC only (full featured) Host Interface == AXI4-Lite	16,435	18,129	5/990	LAV-AT-X70	Synplify Pro
MAC (basic) + PHY (basic) Host Interface == AXI4-Lite	8,385	11,015	5/990	LAV-AT-X70	Synplify Pro
MAC (full featured) + PHY (full_featured) Host Interface == AXI4-Lite	16,818	18,243	5/990	LAV-AT-X70	Synplify Pro

Table A.3. Resource Utilization for LAV-AT-X30 Devices

Configuration	Registers	LUTs	EBRs	Target Device	Synthesis Tools
PHY only (basic) Host Interface == AXI4-Lite	182	146	0/400	LAV-AT-X30	Synplify Pro
PHY only (full_featured) Host Interface == AXI4-Lite	183	154	0/400	LAV-AT-X30	Synplify Pro
MAC only (basic) Host Interface == AXI4-Lite	8,176	9,855	4/400	LAV-AT-X30	Synplify Pro
MAC only (full featured) Host Interface == AXI4-Lite	12,421	10,804	4/400	LAV-AT-X30	Synplify Pro
MAC (basic) + PHY Host Interface == AXI4-Lite	8,363	10,044	4/400	LAV-AT-X30	Synplify Pro
MAC (full featured) + PHY Host Interface == AXI4-Lite	16,551	10,912	5/400	LAV-AT-X30	Synplify Pro

References

- 25G Ethernet IP Release Notes (FPGA-RN-02034).
- 2.5G, 10G, and 25G Ethernet Driver API Reference (FPGA-TN-02375).
- Lattice Memory Mapped Interface and Lattice Interrupt Interface User Guide (FPGA-UG-02039)
- Lattice Radiant Timing Constraints Methodology Application Note (FPGA-AN-02059)
- Lattice Radiant Software web page
- Ethernet IP web page
- Avant-X web page
- Lattice Solutions IP Cores web page
- Lattice Solutions Reference Designs web page
- Lattice Solutions Boards web page
- Lattice Solutions Demonstrations web page
- Lattice Insights web page for training courses and learning plans

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

For frequently asked questions, refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

Revision History

Note: In some instances, the IP may be updated without changes to the user guide. The user guide may reflect an earlier IP version but remains fully compatible with the later IP version. Refer to the IP Release Notes for the latest updates.

Revision 1.5, IP v2.3.0, December 2025

Section	Change Summary
All	Updated the IP version on the cover page.
Introduction	• Updated Table 1.1. Summary of the 25G Ethernet IP. • Updated Table 1.4. Minimum Device Requirements for the 25G Ethernet IP Core.
Functional Description	• Updated Figure 2.2. 25GbE IP Core SyncE Block Diagram. • Updated the Lane Merging section. • Updated the figures in the Clocking section.
Signal Description	Added descriptions for the following signals in Table 4.1. Signal Description for MAC + PHY: • txmac_clk_i • rxmac_clk_i
Register Description	Updated the description for tx_pausreq register in Table 5.14. MAC_CTL Register.
Example Design	• Added LAV-AT-X30 and LAV-AT-X50 device support to Table 6.4. Description of Generated Files. • Updated Figure 6.12. Configure Component.
Designing with the IP	Updated the Timing Constraints section.
Appendix A. Resource Utilization	• Updated Table A.1. Resource Utilization for LAV-AT-X50 Devices. • Added Table A.3. Resource Utilization for LAV-AT-X30 Devices.

Revision 1.4, IP v2.2.1, October 2025

Section	Change Summary
All	• Updated the IP name to 25G Ethernet IP. • Updated the IP version on the cover page. • Removed Avant-ES related information.
Introduction	• Updated Table 1.1. Summary of the 25G Ethernet IP. • Updated Table 1.2. 25G Ethernet IP Core Support Readiness.
Functional Description	Updated the section and figure titles in this section: • Far End Parallel Loopback section. • Near End Parallel Loopback section.
IP Parameter Description	Updated the following tables: • Table 3.1. Configurable Attributes for MAC + PHY Option. • Table 3.3. Configurable Attributes for PHY Only Option.
Example Design	• Updated Table 6.1. 25G Ethernet IP Configuration Supported by the Example Design. • Updated the IP Installation and Generation section. • Updated the Hardware Testing section.
Designing with the IP	Updated Figure 7.2. Configure the User Interface of 25G Ethernet IP Core.

Revision 1.3, IP v2.2.0, June 2025

Section	Change Summary
All	Updated the IP version on the cover page.
Introduction	• Updated Table 1.1. Summary of the 25G Ethernet IP. • Updated Table 1.3. Ordering Part Number.
Functional Description	• Updated the following sentence in the IP Architecture Overview section:

Section	Change Summary
	This IP core instantiates the MPPHY foundation IP configured as 1-Lane 64b/66b128b/132b PCS and a Management block that supports AXI4-Lite access to PCS and MMD registers. • Added the Lane Merging section. • Added the Latency section.
IP Parameter Description	Updated the following tables: • Table 3.1. Configurable Attributes for MAC + PHY Option. • Table 3.3. Configurable Attributes for PHY Only Option.
Signal Description	Updated the following tables: • Table 4.1. Signal Description for MAC + PHY. • Table 4.2. Signal Description for MAC Only. • Table 4.3. Signal Description for PHY Only.

Example Design	• Updated the Importing Versa Files to a Project section.
	• Updated the following figures:
	  • Figure 6.26. Include Source Files.
	  • Figure 6.27. Set tb_top_versa as Top Module.
	  • Figure 6.28. Change the Time Setting for Default Run
	• Added the Continuous Mode section.
	• Added the Hardware Testing section.

Designing with the IP	Added the following sentence to the Running Functional Simulation section: *To run the standalone QuestaSim simulator, the FOUNDRY environment variable must be set as follow:* *set ::env(FOUNDRY) <Radiant install path>/ispfpga.*
References	Added the following reference: *2.5G, 10G, and 25G Ethernet Driver API Reference (FPGA-TN-02375).*

Revision 1.2, IP v2.1.0 (LAV-AT-X70), v1.0.3 (LAV-AT-X70ES), December 2024

Section	Change Summary
All	Added IP version on the cover page.
Introduction	• Updated the IP versions in Table 1.1. Summary of the 25G Ethernet IP. • Added the IP Changes in Table 1.1. Summary of the 25G Ethernet IP. • Added the IP Support Summary section. • Renamed the section title *IP Validation Summary* to *Hardware Support* and updated the content in the Hardware Support section.
Functional Description	• Added Figure 2.2. 25GbE MAC + PHY IP Core SyncE Block Diagram. • Added the following sentence to the Transmit MAC section: *However, there's a limitation that you must not transmit a 16-byte or shorter frame in application due to 128-bit data width design requirement.* • Updated the RSFEC Clocking Overview section.
IP Parameter Description	Updated the following tables: • Table 3.1. Configurable Attributes for MAC + PHY Option. • Table 3.2. Configurable Attributes for MAC Only Option. • Table 3.3. Configurable Attributes for PHY Only Option.

Signal Description	• Removed the following signals from Table 4.1. Signal Description for MAC + PHY:
	  • xg_tx_clk_o
	  • xg_rx_clk_o
	• Added the following signals to Table 4.3. Signal Description for PHY Only:
	  • tx0_am_i
	  • rx0_am_o
	• Updated the description for the following signals in Table 4.1. Signal Description for MAC + PHY and Table 4.2. Signal Description for MAC Only:
	  • tx_pausreq_i

Section	Change Summary
	• tx_paustm_i • Updated the description for the tx0_am_i signal in Table 4.3. Signal Description for PHY Only.
Register Description	Added the following registers to Table 5.20. Summary of Statistics Counters: • TX_STAT_PKT_LNGTH_ACCU_0 • TX_STAT_PKT_LNGTH_ACCU_1 • RX_STAT_PKT_LNGTH_ACCU_0 • RX_STAT_PKT_LNGTH_ACCU_1
Example Design	• Updated this section to list the evaluation board. • Added step 5 and step 6 to the Importing Versa Files to a Project section. • Updated the Hardware Testing section.
Appendix A. Resource Utilization	Updated Table A.2. Resource Utilization for LAV-AT-X70 Devices.
References	• Added the following references: • 25G Ethernet MAC + PHY IP Release Notes • Lattice Solutions IP Cores web page • Lattice Solutions Reference Designs web page • Lattice Solutions Boards web page • Lattice Solutions Demonstrations web page • Removed the reference to the Lattice Radiant Software User Guide

Revision 1.1, September 2024

Section	Change Summary
Introduction	Updated the following bullet point in the Features section: *Supports VLAN, SVLAN, and up to 9600 bytes of Jumbo frames.*
Functional Description	• Updated the following figures: • Figure 2.12. Normal Frame Reception. • Figure 2.13. Back-to-back Frames Reception. • Figure 2.14. Frame Reception with In-Band FCS Passing. • Figure 2.15. Reception with Custom Preamble. • Figure 2.17. Default Normal Frame Transmission. • Figure 2.18. Transmission with In-Band FCS Passing. • Figure 2.19. Transmission with Custom Preamble Passing. • Figure 2.24. Clock Network Diagram. • Figure 2.26. Sequence to Configure RX MAC In-Band FCS Passing. • Updated the MAC Reset Sequence section.
Signal Description	• Updated the description for the following signals in Table 4.1. Signal Description for MAC + PHY: • xg_tx_clk_o • xg_rx_clk_o • xg_tx_gclk_o • xg_rx_gclk_o • tx_pausreq_i • Updated the description for the tx_pausreq_i signal in Table 4.2. Signal Description for MAC Only. • Updated the value for all occurrences of the *sysbus_clk_i* signal from *100 MHz* to *20 MHz*.
Register Description	• Updated the description for the tx_pausreq signal in Table 5.14. MAC_CTL Register. • Updated a reference to the Lattice Avant-G/X MPPHY Module User Guide (FPGA IPUG 02233) in Table 5.35. Register Address Map for PHY.
Designing with the IP	• Updated Table 7.1. Generated File List. • Updated the following sentence in the Timing Constraints section:

Section	Change Summary
	For example, the default clock period of sysbus_clk_i is 50 ns (20 MHz), if this clock is driven by 25 MHz clock in your project, the clock period of the constraint must be changed to 40 ns.

Revision 1.0, June 2024

Section	Change Summary
Introduction	• Updated Table 1.1. Summary of the 25G Ethernet IP. • Added the following feature to the MAC list in the Features section: *Programmable promiscuous (transparent) mode.* • Removed LSE from the following sentence in the Hardware Evaluation section: *To change this setting, go to Project > Active Strategy > LSE/Synplify Pro Settings.* • Added the Hardware Support section. • Added the Minimum Device Requirements section.

Functional Description	• Updated Figure 2.1. 25GbE MAC + PHY IP Core Block Diagram.
	• Updated the following polynomial in the Frame Check Sequence (CRC-32) Insertion section: $FCS(X) = X^{32} +X^{26} +X^{23} +X^{22} +X^{16} +X^{12} +X^{11} +X^{10} +X^{8} +X^{7} +X^{5} +X^{4} +X^{2}+X+1$
	• Updated the following figures:
	  • Figure 2.12. Normal Frame Reception.
	  • Figure 2.13. Back-to-back Frames Reception.
	  • Figure 2.14. Frame Reception with In-Band FCS Passing.
	  • Figure 2.15. Reception with Custom Preamble.
	  • Figure 2.17. Default Normal Frame Transmission.
	  • Figure 2.18. Transmission with In-Band FCS Passing.
	  • Figure 2.19. Transmission with Custom Preamble Passing.
	• Added the following sections:
	  • Clocking section.
	  • Reset section.

IP Parameter Description	Moved the IP Parameter Description section from the Functional Description section to this section.
Signal Description	• Moved the Signal Description section from the Functional Description section to this section. • Updated the description for sysbus_clk_i in Table 4.1. Signal Description for MAC + PHY. • Added the *axi4l_wstrb_i* signal to Table 4.1. Signal Description for MAC + PHY. • Added the TX Pause Frame signals and a table note to Table 4.1. Signal Description for MAC + PHY. • Added the *axi4l_wstrb_i* signal to Table 4.2. Signal Description for MAC Only. • Added the TX Pause Frame signals and a table note to Table 4.2. Signal Description for MAC Only. • Added the *axi4l_wstrb_i* signal to Table 4.3. Signal Description for PHY Only
Register Description	Added the RX_STAT_LINK_OK register to Table 5.20. Summary of Statistics Counters.

Example Design	• Updated all references of *ModelSim Lattice Edition software* in this section to *QuestaSim Lattice Edition software*.
	• Updated Figure 6.1. Ethernet MAC + PHY IP Example Design Block Diagram.
	• Updated the content for Step 1 in the IP Installation and Generation section.
	• Updated the following figures:
	  • Figure 6.21. Create a Simulation Project.
	  • Figure 6.22. Include Source Files.
	  • Figure 6.23. Set tb_top_versa as Top Module.
	  • Figure 6.24. Change the Time Setting for Default Run.
	  • Figure 6.25. Execute Simulation Script.

Section	Change Summary

Designing with the IP	• Updated the following figures:
	  • Figure 7.1. Module/IP Block Wizard.
	  • Figure 7.2. Configure the User Interface of 25G Ethernet MAC + PHY IP Core.
	  • Figure 7.6. Simulation Wizard.
	  • Figure 7.7. Add and Reorder Source.
	• Added the following sections:
	  • Design Implementation section.
	  • Timing Constraints section.
	  • Specifying the Strategy section.
	• Added Figure 7.8. Simulation Result.

Appendix A. Resource Utilization	• Updated the resource utilization information in Table A.1. Resource Utilization for LAV-AT-X70ES Devices. • Added Table A.2. Resource Utilization for LAV-AT-X70 Devices.

Revision 0.81, April 2024

Section	Change Summary
Introduction	Removed Lattice Synthesis Engine (LSE) from Table 1.1. Summary of the 25G Ethernet IP.
Register Description	• Updated the MAC address values in MAC_ADDR_0 and MAC_ADDR_1 Register section to the following: *For example, to set the MAC address to: AC-DE-48-00-00-80 would require writing 0x48_00_00_80 to address 0x014 (MAC_ADDR_0). 0xAC_DE to address 0x018 (MAC_ADDR_1).* • Removed the reference to the Appendix B section in the MC_TABLE_0 and MC_TABLE_1 Register section.
Example Design	Added this section.
Appendix B. Code Listing for Multicast Bit Selection Hash Algorithm in C Language	Removed this section.

Revision 0.80, December 2023

Section	Change Summary
All	Preliminary release.

www.latticesemi.com