



5G Ethernet IP

IP Version: v1.0.0

User Guide

FPGA-IPUG-02313-1.0

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

Contents	3
Abbreviations in This Document.....	8
1. Introduction	9
1.1. Overview of the IP	9
1.2. Quick Facts	9
1.3. IP Support Summary	10
1.4. Features	10
1.5. Licensing and Ordering Information	10
1.5.1. Hardware Evaluation	11
1.6. Hardware Support	11
1.7. Minimum Device Requirements	11
1.8. Naming Conventions	11
1.8.1. Nomenclature	11
1.8.2. Signal Names	11
2. Functional Description.....	12
2.1. MAC + PHY	12
2.1.1. IP Architecture Overview	12
2.1.2. Block Diagram.....	12
2.1.3. Management Block	12
2.2. MAC.....	12
2.2.1. IP Architecture Overview	12
2.2.2. Block Diagram.....	13
2.2.3. Ethernet Data Format	13
2.2.4. Receive MAC.....	14
2.2.5. Transmit MAC.....	14
2.2.6. Receive AXI4-Stream Interface.....	15
2.2.7. Transmit AXI4-Stream Interface	19
2.2.8. Data Unpacking	22
2.2.9. MAC Management Block.....	23
2.3. PHY	23
2.3.1. IP Architecture Overview	23
2.3.2. Block Diagram.....	24
2.3.3. Lane Merging.....	24
2.3.4. PHY Management Block	26
2.3.5. Loopback Modes	27
2.4. Clocking	29
2.4.1. Clocking Overview	29
2.5. Reset	30
2.5.1. MAC Reset Sequence	30
2.5.2. PHY Reset Sequence.....	30
2.6. Latency	31
3. IP Parameter Description.....	32
3.1. MAC + PHY	32
3.2. MAC Only	33
3.3. PHY Only.....	34
4. Signal Description	36
4.1. MAC + PHY Signals	36
4.2. MAC Only Signals	39
4.3. PHY Only Signals.....	43
5. Register Description	45
5.1. MAC + PHY Registers.....	45
5.2. MAC Registers	45

5.2.1.	Configuration Registers for MAC.....	45
5.2.2.	Interrupt Registers	49
5.2.3.	Statistics Counters	50
5.3.	PHY Registers	55
5.3.1.	Configuration Registers for PHY	55
6.	Example Design.....	56
6.1.	Example Design Configuration	56
6.2.	Overview of Example Design and Features	56
6.3.	Example Design Components.....	57
6.3.1.	Versa Example Design Components (Avant Devices)	57
6.4.	Generating Example Design	58
6.4.1.	Create New Radiant Project	58
6.4.2.	IP Installation and Generation.....	62
6.4.3.	Importing Example Design Files to a Project	63
6.5.	Example Design Simulation	67
6.5.1.	5 Gb Ethernet MAC Example Design Testbench Flow	67
6.5.2.	Create a Simulation Project.....	67
6.6.	Hardware Testing	73
7.	Designing with the IP	78
7.1.	Generating and Instantiating the IP	78
7.1.1.	Generated Files and File Structure	81
7.2.	Design Implementation	81
7.3.	Timing Constraints	82
7.3.1.	Create Clock Constraints for MAC-only Option	82
7.3.2.	Create Clock Constraints for MAC + PHY Option	82
7.3.3.	Create Clock Constraints for PHY-only Option	83
7.4.	Specifying the Strategy.....	83
7.5.	Running Functional Simulation	83
7.5.1.	Simulation Results	85
Appendix A.	Resource Utilization	86
References	87
Technical Support Assistance	88
Revision History	89

Figures

Figure 1.1. 5GBASE-R Application	9
Figure 2.1. 5GbE MAC + PHY IP Core Block Diagram	12
Figure 2.2. 5G Ethernet MAC IP Core Block Diagram	13
Figure 2.3. Untagged Ethernet Frame Format	13
Figure 2.4. Tagged Ethernet Frame Format	13
Figure 2.5. AXI4-Stream RX Adapter Interface Diagram	15
Figure 2.6. Normal Frame Reception	16
Figure 2.7. Back-to-back Frames Reception	17
Figure 2.8. Frame Reception with In-Band FCS Passing	18
Figure 2.9. Reception with Custom Preamble	19
Figure 2.10. AXI4-Stream TX Adapter Interface Diagram	19
Figure 2.11. Default Normal Frame Transmission	20
Figure 2.12. Transmission with In-Band FCS Passing	21
Figure 2.13. Transmission with Custom Preamble Passing	22
Figure 2.14. Receive Physical Interface for 8/6-bit GMII or MII	22
Figure 2.15. Transmit Physical Interface for 8/16-bit GMII or MII	23
Figure 2.16. 5G Ethernet PHY IP Core Block Diagram	24
Figure 2.17. Set the Lane ID Configuration of the 5G Ethernet PHY IP Core	25
Figure 2.18. Lane ID Numbering	25
Figure 2.19. Lane Merging Report File	26
Figure 2.20. Remapped Addresses for PCS Register Address through the AXI4-Lite Interface	27
Figure 2.21. Far End Parallel Loopback	28
Figure 2.22. Near End Parallel Loopback	29
Figure 2.23. 5G Clock Network Diagram for Avant Devices	29
Figure 2.24. Sequence to Configure RX MAC In-Band FCS Passing	30
Figure 2.25. PHY Initialization Sequence	31
Figure 6.1 5 Gb Ethernet MAC IP Versa Example Design Block Diagram	57
Figure 6.2. Start Page of the Radiant Software	58
Figure 6.3. Create a New Project	59
Figure 6.4. Add Source to Project	59
Figure 6.5. Select an Avant Device for the Project	60
Figure 6.6. Select a Synthesis Tool	60
Figure 6.7. Project Information	61
Figure 6.8. A Radiant Software Project is Created	61
Figure 6.9. Generate Component	62
Figure 6.10. Configure Component	62
Figure 6.11. Device Setup for Avant Devices	63
Figure 6.12. Add Related Files into the Radiant Software Project	64
Figure 6.13. CONTINUOUS_TRAFFIC Parameter for Continuous Mode in versa_top.sv File	64
Figure 6.14. Add the tb_top_versa.sv File into the Radiant Software Project	65
Figure 6.15. CONTINUOUS_TRAFFIC Parameter for Continuous Mode in the tb_versa_top.sv File	65
Figure 6.16. CONTINUOUS_TRAFFIC Parameter for Non-Continuous Mode in the tb_versa_top.sv File	65
Figure 6.17. Set the tb_top_versa.sv File to be Included for Simulation Only	66
Figure 6.18. Add the .pdc File into the Radiant Software Project	66
Figure 6.19. Timing Constraint File (versa.pdc Example from an Avant Device)	66
Figure 6.20. 5G Ethernet MAC Example Design Testbench Flowchart	67
Figure 6.21. Create a Simulation Project	68
Figure 6.22. Include Source Files	68
Figure 6.23. Set tb_top* as Top Module	69
Figure 6.24. Change the Time Setting for Default Run	69
Figure 6.25. Example Design Simulation Waveform for Continuous Mode	70
Figure 6.26. Example Design Simulation Output for Continuous Mode	70

Figure 6.27. Example Design Simulation Waveform for Non-Continuous Mode	71
Figure 6.28. Example Design Simulation Output for Non-Continuous Mode	72
Figure 6.29. Avant-X Versa Board Setup	73
Figure 6.30. Loopback Module Setup	73
Figure 6.31. Successful Design Compilation	74
Figure 6.32. Cable Selection for Avant-AT-X Device	74
Figure 6.33. Programmer GUI	74
Figure 6.34. Push Buttons on Avant-AT-X Device	75
Figure 6.35. Passing Scenario on Avant-AT-X Device	76
Figure 6.36. Failing Scenario on Avant-AT-X Device	76
Figure 6.37. LED 7-Segment	77
Figure 6.38. Parameters for Frame Size in the Top File	77
Figure 7.1. Module/IP Block Wizard	78
Figure 7.2. Configure the User Interface of 5G Ethernet IP Core	79
Figure 7.3. Check Generated Result	80
Figure 7.4. Synthesize Design	80
Figure 7.5. Timing Constraint File (.pdc) for the 5G MAC IP	81
Figure 7.6. Simulation Wizard	83
Figure 7.7. Add and Reorder Source	84
Figure 7.8. Summary Window	84
Figure 7.9. Simulation Results	85

Tables

Table 1.1. Summary of the 5G Ethernet IP	9
Table 1.2. 5G Ethernet IP Support Readiness	10
Table 1.3. Minimum Device Requirements for 5G Ethernet IP Core	11
Table 2.1. Shared Signal Mapping of 5G Ethernet IP to MPPHY for Lane Merging	26
Table 2.2. AXI4-Lite to PCS Address and Data Conversion	27
Table 2.3. MAC + PHY Attributes	31
Table 3.1. MAC + PHY Attributes	32
Table 3.2. MAC Only Attributes	33
Table 3.3. PHY Only Attributes	34
Table 4.1. Signal Description—MAC + PHY	36
Table 4.2. Signal Description—MAC Only	39
Table 4.3. Signal Description—PHY Only	43
Table 5.1. Access Types for MAC Only	45
Table 5.2. Configuration Registers for MAC	45
Table 5.3. MODE Register	46
Table 5.4. TX_CTL Register	46
Table 5.5. RX_CTL Register	46
Table 5.6. MAX_PKT_LENGTH Register	47
Table 5.7. IPG_VAL Register	47
Table 5.8. MAC_ADDR_0 Register	47
Table 5.9. MAC_ADDR_1 Register	48
Table 5.10. TX_RX_STS Register	48
Table 5.11. VLAN_TAG Register	48
Table 5.12. MC_TABLE_0 Register	48
Table 5.13. MC_TABLE_1 Register	48
Table 5.14. PAUSE_OPCODE Register	49
Table 5.15. MAC_CTL Register	49
Table 5.16. PAUSE_TM Register	49
Table 5.17. Summary of Interrupt Registers	49

Table 5.18. INT_STATUS Register	50
Table 5.19. INT_ENABLE Register	50
Table 5.20. INT_SET Register	50
Table 5.21. Summary of Statistics Counters	50
Table 5.22. Access Types for PHY	55
Table 5.23. Register Address Map for PHY	55
Table 6.1. IP Configuration for the 5G Ethernet MAC IP Example Design	56
Table 6.2. AXI4-Lite Module Configuration Registers	58
Table 6.3. Description of Generated Files in the Eval Folder	63
Table 6.4. Top-Level Files	64
Table 6.5. LED 7-Segment Description for DIG 1	77
Table 6.6. LED 7-Segment Description for DIG 3	77
Table 7.1. Generated File List	81
Table A.1. Resource Utilization for Avant Devices	86

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
APB	Advanced Peripheral Bus
AXI4-Lite	Advanced eXtensible Interface 4 Lite
AXI4-Stream/AXI4S	Advanced eXtensible Interface 4 Stream
CRC	Cyclic Redundancy Check
DA	Destination Address
DFE	Decision Feedback Equalizer
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
LINTR	Lattice Interrupt Interface
LUT	Look-Up Table
L/T	Length/Type
MAC	Media Access Controller
MDIO	Management Data Input/Output
MMD	MDIO Manageable Device
MTU	Maximum Transmission Unit
PCS	Physical Coding Sublayer
PHY	Physical Layer Device
PMA	Physical Medium Attachment
PTP	Precision Time Protocol
SA	Source Address
SFD	Start Frame Delimiter
SOF	Start of Frame
ToD	Time of Day
TSU	Timestamp Unit
XGMII	10-Gigabit Media Independent Interface

1. Introduction

1.1. Overview of the IP

The Lattice Semiconductor 5G Ethernet (GbE) IP core supports the ability to transmit and receive data between a host processor and an Ethernet network. The 5GbE IP core consists of the 10-Gigabit Media Independent Interface (XGMII), which connects media access controllers (MACs) and Physical Layer devices (PHYs). The main function of the 5GbE MAC 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. The PHY implements the physical coding sublayer (PCS) and physical medium attachment (PMA) functionality based on the IEEE 802.3 5GBASE-R specification.

The following figure shows an example of a 5GBASE-R application. The 5GbE MAC is connected to the 5GbE PHY within the 5G Ethernet IP core through the XGMII interface. The clock source to the Ethernet MAC is from `xg_tx_gclk_o[0]` (clock output from the Ethernet PHY).

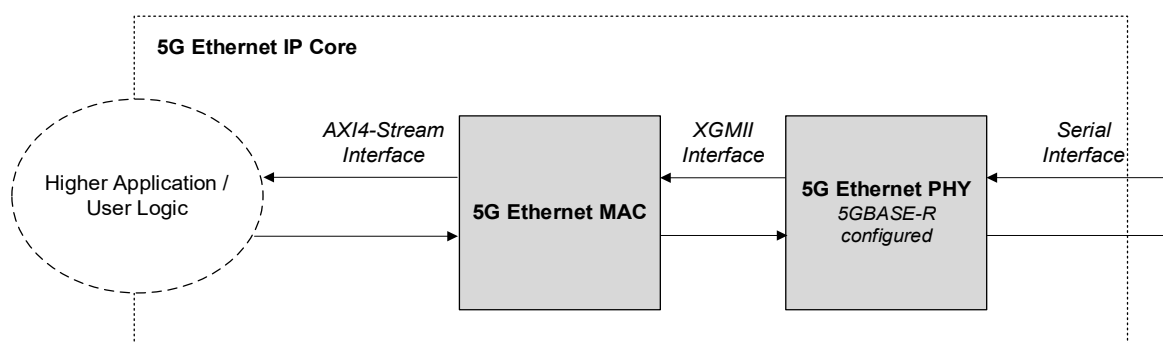


Figure 1.1. 5GBASE-R Application

1.2. Quick Facts

The following table provides quick facts about the 5G Ethernet IP core.

Table 1.1. Summary of the 5G Ethernet IP

IP Requirements	Supported Devices	Avant™-AT-G and Avant-AT-X
	IP Option	<ul style="list-style-type: none"> MAC only PHY only MAC + PHY
	IP Changes ¹	Refer to the 5G Ethernet IP Release Notes (FPGA-RN-02102) .
Resource Utilization	Supported User Interface	AXI4-Stream/APB/AXI4-Lite interface.
	Resources	See Appendix A. Resource Utilization .
Design Tool Support	Lattice Implementation	IP core v1.0.0—Lattice Radiant™ software 2025.2 or later.
	Synthesis	Synopsys® Synplify Pro® software, O-2018.09LR-SP1.
	Simulation	For the list of supported simulators, see the Lattice Radiant Software User Guide.

Note:

- 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 5G Ethernet IP core.

Table 1.2. 5G Ethernet IP Support Readiness

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

1.4. Features

The 5G Ethernet IP core offers the following key features:

MAC

- Compliant to the IEEE 802.3-2022 standard
- Supports standard 5 Gbps Ethernet link layer data rate
- 32-bit AXI4-Stream interface on the client's transmit and receive interfaces operating at 156.25 MHz
- 64-bit XGMII interface operating at 78.125 MHz
- Supports deficit idle count (DIC)
- Supports VLAN and Jumbo frames of 9,600 bytes
- 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
- Supports flow control using pause frames
- Automatic padding of short frames
- Inter-Frame Stretch Mode during transmission
- Supports full-duplex operation
- Advanced Peripheral Bus (APB) interface or AXI4-Lite interface for register accessProgrammable promiscuous (transparent) mode

PHY

- Designed to the IEEE 802.3-2022 5GBASE-R specification
- 64b/66b encoding and decoding
- XGMII interface: 64-bit, 78.125 MHz
- Supports AXI4-Lite and management data input/output for PCS/
- TX phase compensation FIFO and RX clock compensation FIFO

1.5. Licensing and Ordering Information

The 5G Ethernet IP is available with Lattice Radiant Subscription Software. To purchase the Lattice Radiant Subscription license, [contact sales](#) or buy through our [Online Store](#).

1.5.1. Hardware Evaluation

The 5G Ethernet IP core supports Lattice's IP hardware evaluation capability when used with Avant-AT-G, and 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**.

1.6. Hardware 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 5G Ethernet IP core are as follows:

Table 1.3. Minimum Device Requirements for 5G Ethernet IP Core

Devices	Speed Grades
Avant-AT-G/X	All speed grades

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 5G 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 the XGMII internally.

This IP option is supported on Avant-AT-G and Avant-AT-X devices.

For Lane Merging details, refer to the [Lane Merging](#) section.

2.1.2. Block Diagram

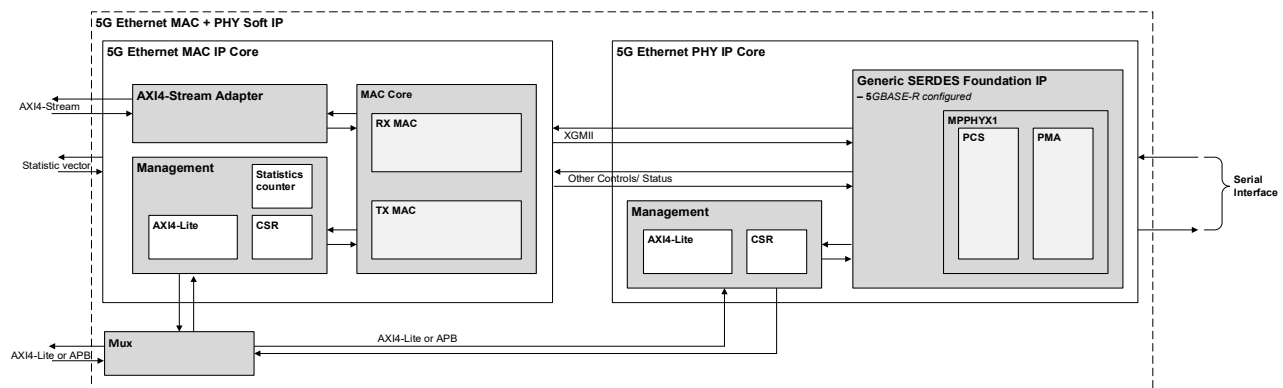


Figure 2.1. 5GbE MAC + PHY IP Core Block Diagram

2.1.3. Management Block

The Management block can be accessed through the AXI4-Lite or APB 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 protocol timing details, refer to AMBA 4 AXI and ACE Protocol Specification issue H or AMBA 3 APB Protocol version 1.0 Specification.

2.2. MAC

2.2.1. IP Architecture Overview

The main function of the 5G 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

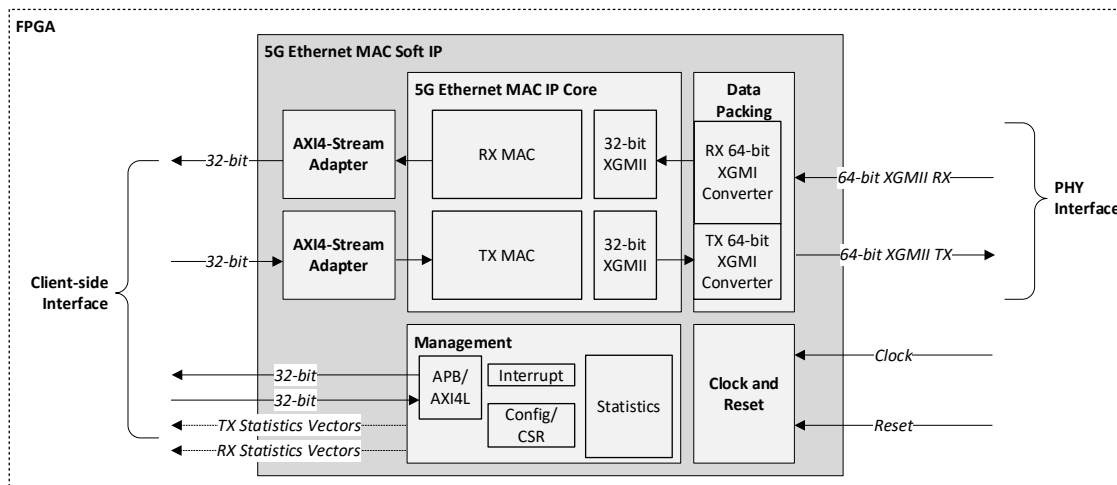


Figure 2.2. 5G Ethernet MAC IP Core Block Diagram

2.2.3. Ethernet Data Format

The following figure shows an untagged Ethernet frame format.

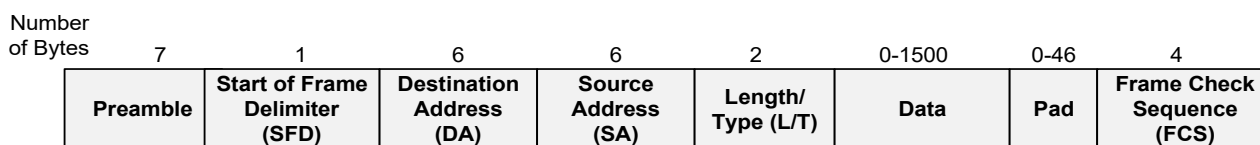


Figure 2.3. Untagged Ethernet Frame Format

The following figure shows a tagged Ethernet frame format.

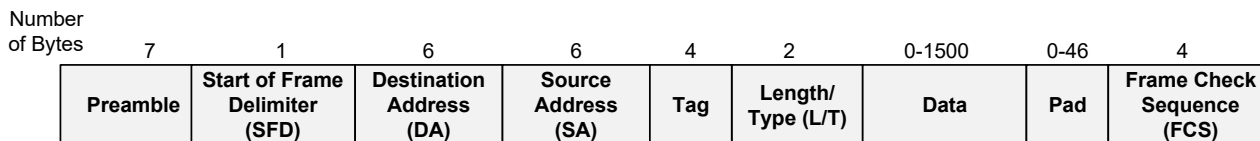


Figure 2.4. Tagged Ethernet Frame Format

The MAC is responsible for constructing a valid frame from the data received from the client before transmitting it. On the receiver path, it receives frames from the network through XGMII interface and passes the parameters of the frame to the client through the AXI4-Stream interface.

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.
- 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, 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.

In all cases, the preamble and SFD bytes are always stripped off the frame before it is transferred on the AXI4-Stream interface unless the rx_pass_pream bit is set in the RX_CTL register. When rx_pass_pream bit is set, the RX MAC does not check for the SFD.

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 the 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.

Once a valid SOF is detected, the DA field of the incoming frame is analyzed. If the DA field is a unicast address, it is compared with the programmed MAC address. Unless the prms bit of RX_CTL register was 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 had 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.

2.2.5. Transmit MAC

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

- Data padding for short frames when FCS generation is enabled.
- 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.
- 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.

The frame is not sent over the network until the network has been idle for a minimum of 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 802.3-2022 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 return 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 transmit a pause frame with the pause time value supplied through the tx_paustim bits of the PAUSE_TMR register.

2.2.6. Receive AXI4-Stream Interface

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

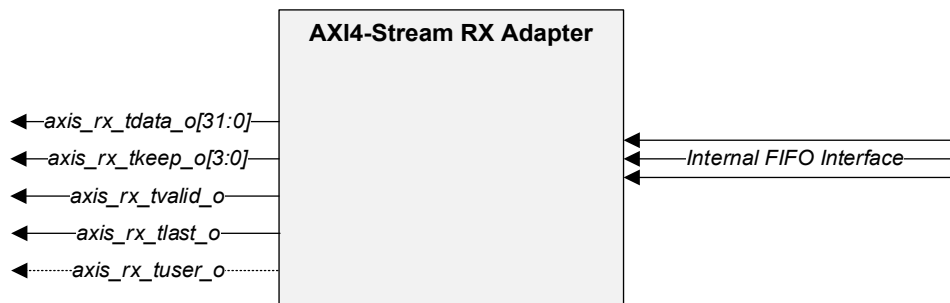


Figure 2.5. AXI4-Stream RX Adapter Interface Diagram

2.2.6.1. 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.

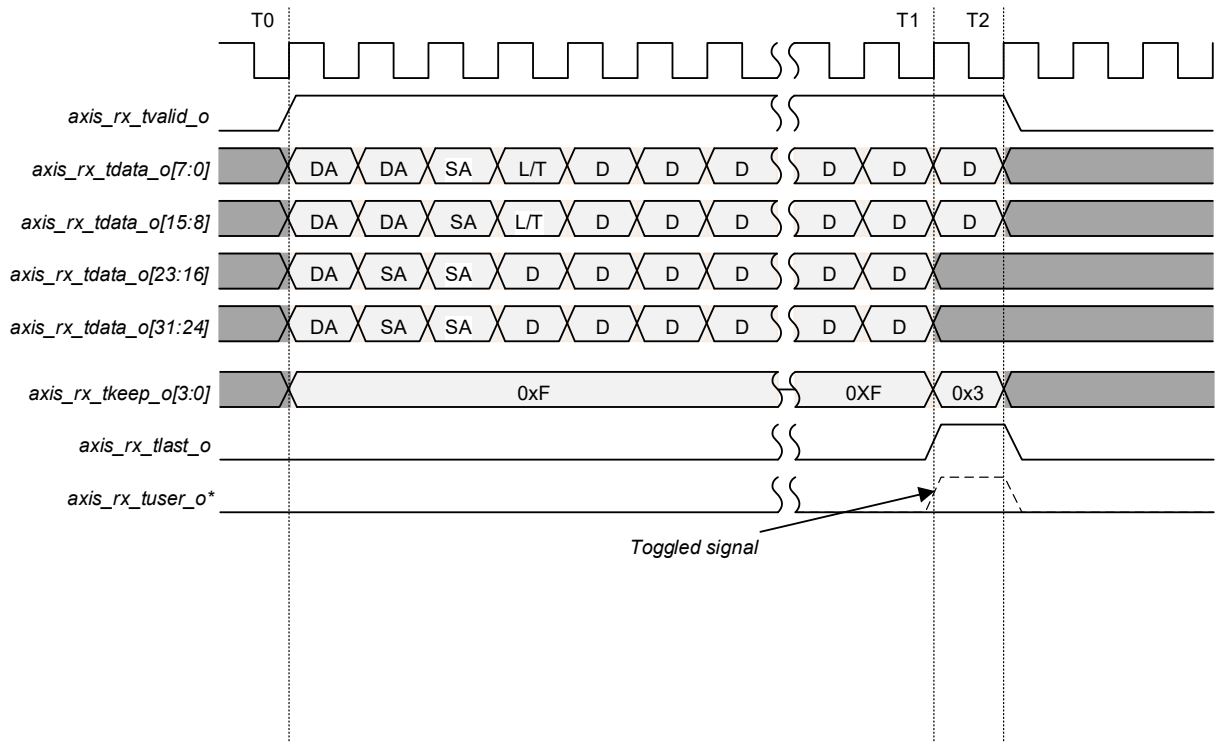


Figure 2.6. Normal Frame Reception

*** Note:** Under normal circumstances, the `axis_rx_tuser_o` signal must not be toggled when there is no error. However, if an error occurs and the `axis_rx_tuser_o` signal is toggled, you must check on the status signal—`rx_statvec_o` signal.

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.

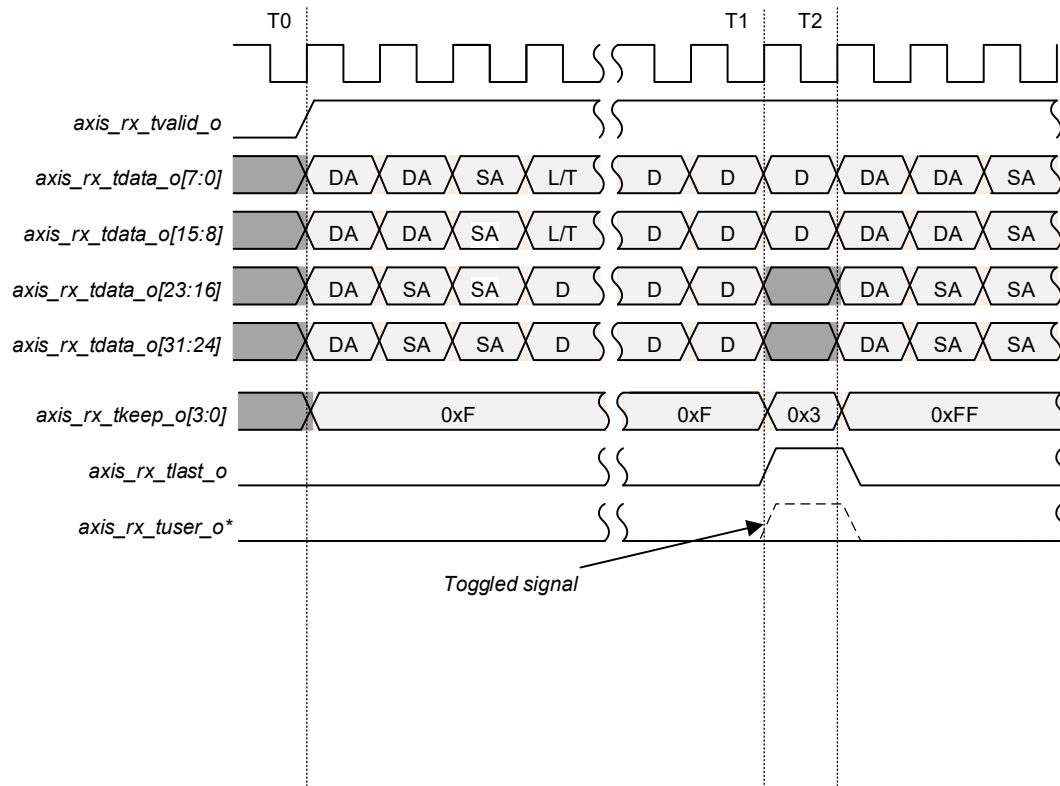


Figure 2.7. Back-to-back Frames Reception

*** Note:** Under normal circumstances, the `axis_rx_tuser_o` signal must not be toggled when there is no error. However, if an error occurs and the `axis_rx_tuser_o` signal is toggled, you must check on the status signal—`rx_statvec_o` signal.

2.2.6.2. 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.

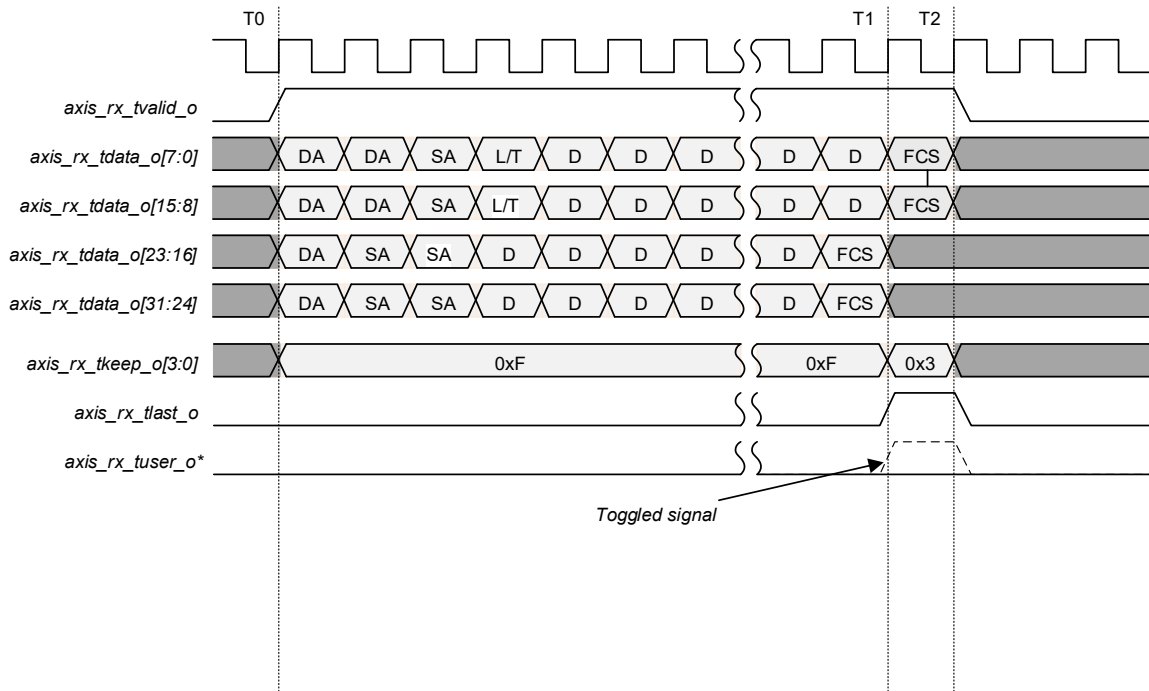


Figure 2.8. Frame Reception with In-Band FCS Passing

*** Note:** Under normal circumstances, the `axis_rx_tuser_o` signal must not be toggled when there is no error. However, if an error occurs and the `axis_rx_tuser_o` signal is toggled, you must check on the status signal—`rx_statvec_o` signal.

2.2.6.3. 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.

Note: Custom preamble mode is only supported by XGMII interface.

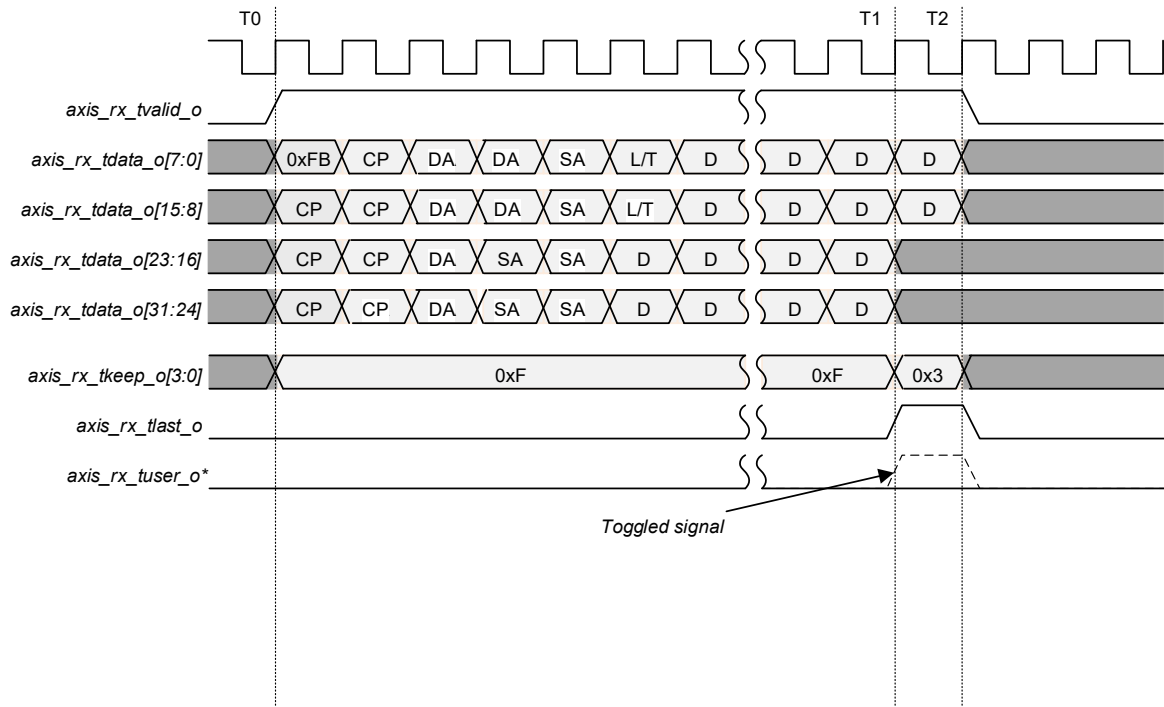


Figure 2.9. Reception with Custom Preamble

* **Note:** Under normal circumstances, the axis_rx_tuser_o signal must not be toggled when there is no error. However, if an error occurs and the axis_rx_tuser_o signal is toggled, you must check on the status signal—rx_statvec_o signal.

2.2.7. Transmit AXI4-Stream Interface

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

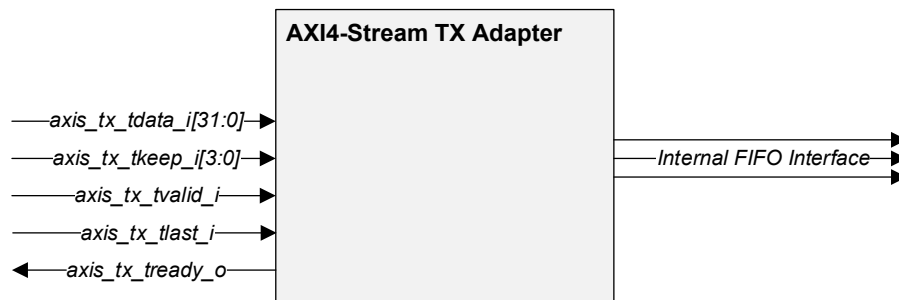


Figure 2.10. AXI4-Stream TX Adapter Interface Diagram

2.2.7.1. 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`).

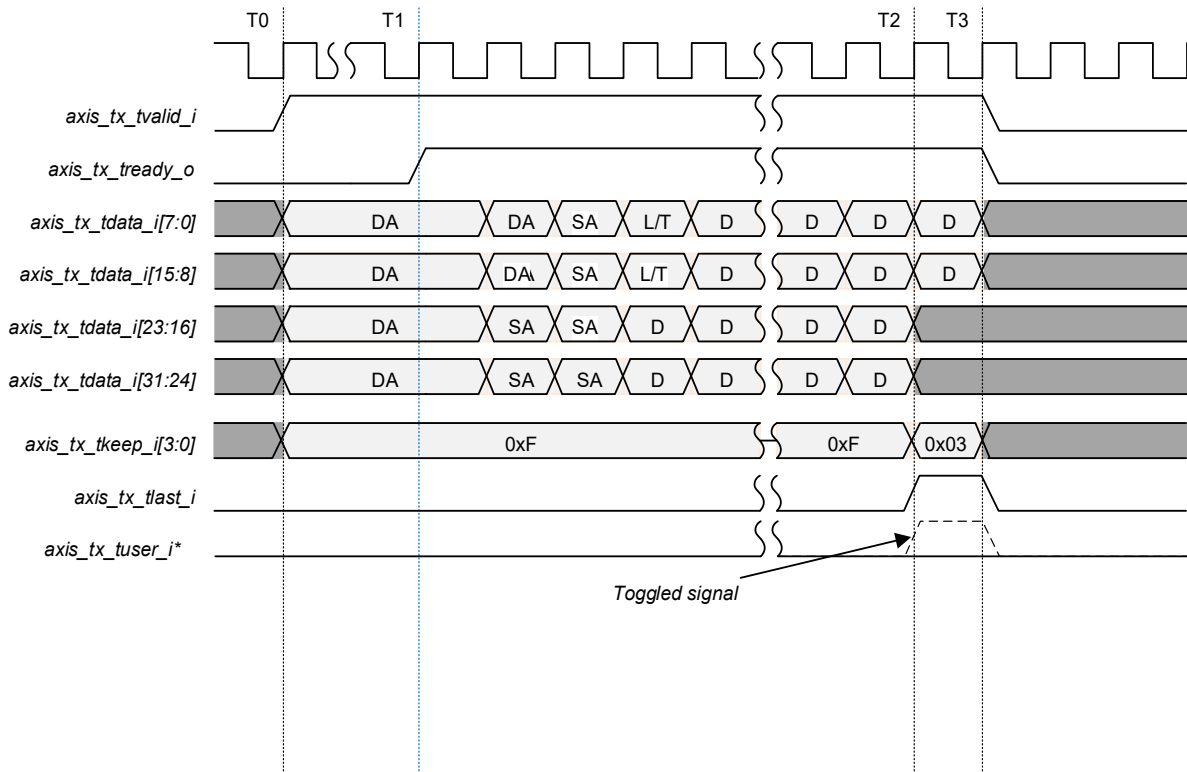


Figure 2.11. Default Normal Frame Transmission

* **Note:** You can inject the error by asserting the `axis_tx_tuser_i` signal during EOP. Otherwise, the `axis_tx_tuser_i` signal must drive to low.

2.2.7.2. In-Band FCS Passing

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

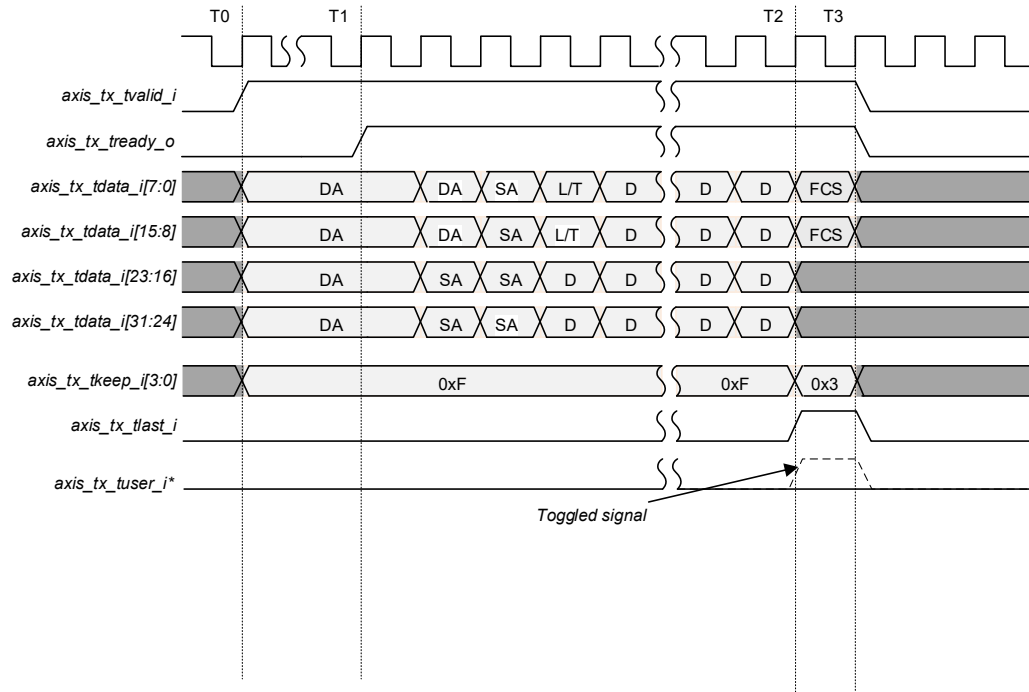


Figure 2.12. Transmission with In-Band FCS Passing

*** Note:** You can inject the error by asserting the axis_tx_tuser_i signal during EOP. Otherwise, the axis_tx_tuser_i signal must drive to low.

2.2.7.3. Custom Preamble Passing

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

Note: Custom preamble mode is only supported by XGMII interface.

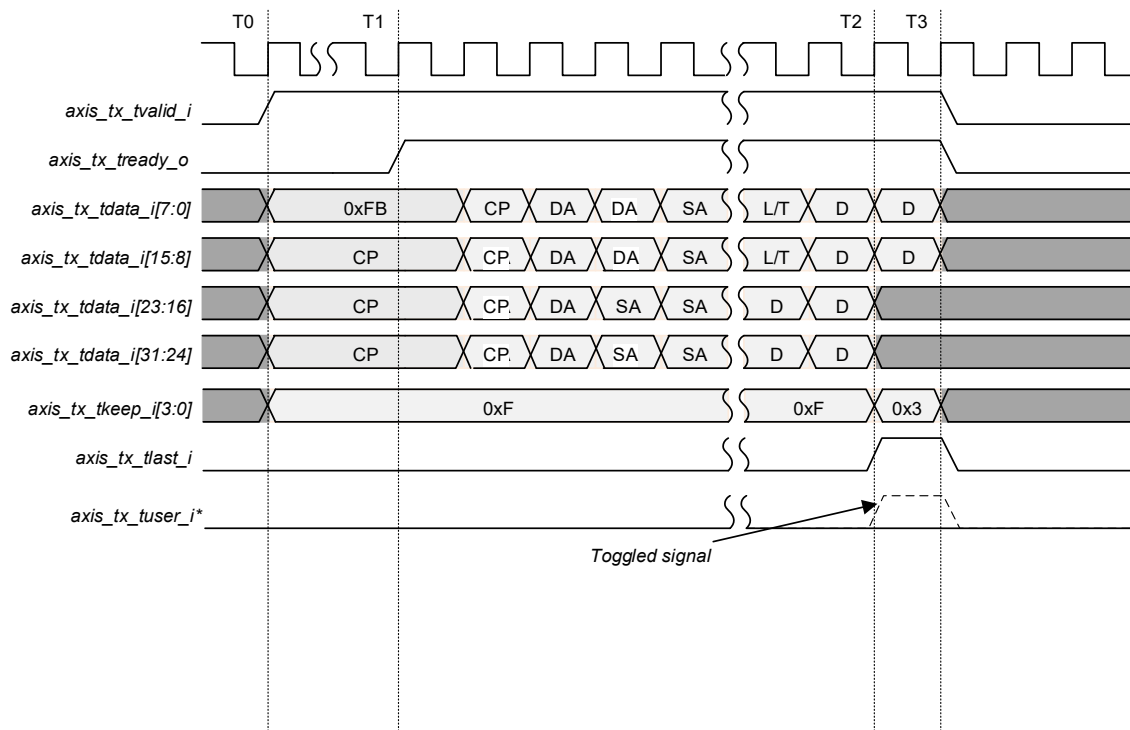


Figure 2.13. Transmission with Custom Preamble Passing

*** Note:** You can inject the error by asserting the axis_tx_tuser_i signal during EOP. Otherwise, the axis_tx_tuser_i signal must drive to low.

2.2.8. Data Unpacking

This module converts between 64-bit XGMII and 32-bit XGMII interfaces.

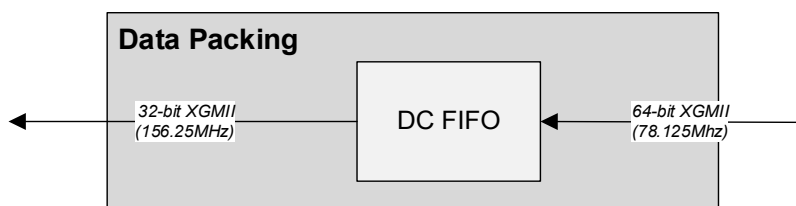


Figure 2.14. Receive Physical Interface for 8/6-bit GMII or MII

The receiver side takes the 64-bit XGMII from PHY and convert into 32-XGMII packets. The transformation process involves clock crossing from 64-bit XGMII at 78.125 MHz to 32-bit XGMII at 156.25 MHz by using a DC FIFO block.

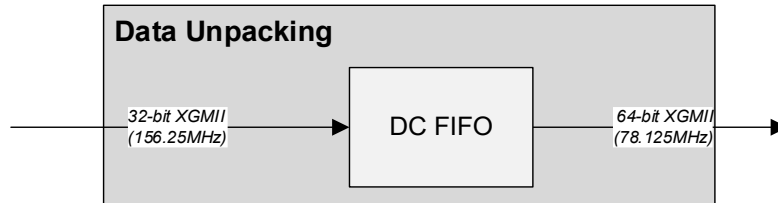


Figure 2.15. Transmit Physical Interface for 8/16-bit GMII or MII

The Data Unpacking block converts the 32-bit XGMII data to 64-bit XGMII through the DC FIFO block. This module also include clock crossing from 156.25 MHz to 78.125 MHz handling.

2.2.9. MAC Management Block

The MAC Management block is accessed through the APB 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 3 APB protocol, refer to the AMBA 3 APB Protocol version 1.0 Specification.

2.3. PHY

2.3.1. IP Architecture Overview

The 5G Ethernet PHY IP core provides the XGMII interface to the MAC and follows the IEEE 802.3 5GBASE-R standard. It supports 64 bits of data and 8 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, or APB access to PCS registers. For more information on MPPHY foundation IP, refer to the [Lattice Avant SERDES/PCS User Guide \(FPGA-TN-02313\)](#).

The functional description in this section is only applicable for 5G Ethernet PHY IP core on Avant-AT-G and Avant-AT-X devices.

2.3.2. Block Diagram

The following figure shows the top-level block diagram of the 5G Ethernet PHY IP core.

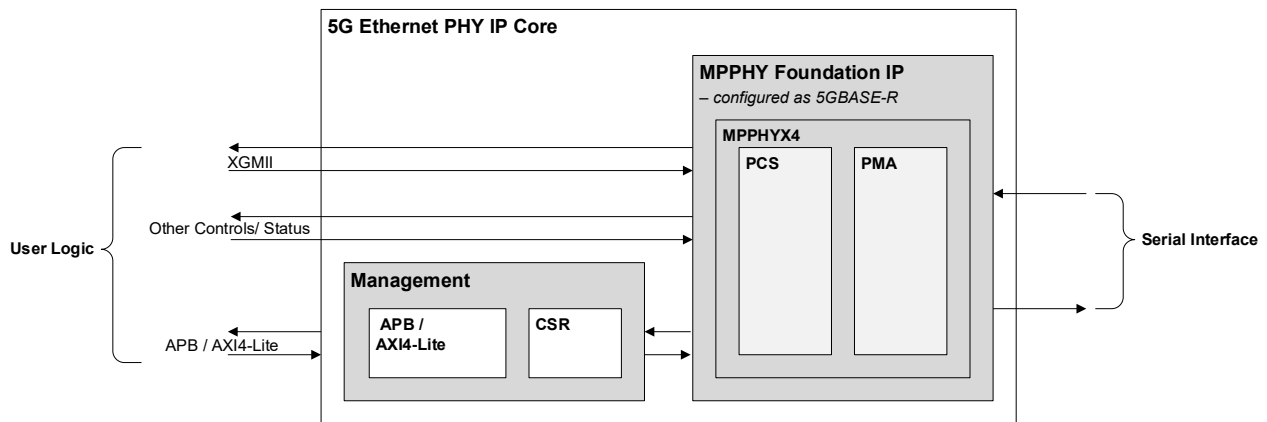


Figure 2.16. 5G Ethernet PHY IP Core Block Diagram

2.3.3. Lane Merging

2.3.3.1. Overview

The Avant-AT-G/X silicon can have up to seven usable quads, depending on the package. Each MPPHY block is a quad made up of four lanes (X4). Therefore, a single device could theoretically contain up to 28 lanes across seven 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 Avant devices. For more information, refer to the [Lattice Avant-G/X MPPHY Module User Guide \(FPGA-IPUG-02233\)](#).

2.3.3.2. Usage

The 5G 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 can set the LANE ID configuration of the IP according to the number of quads you want to use.

To set the Lane ID, 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 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.

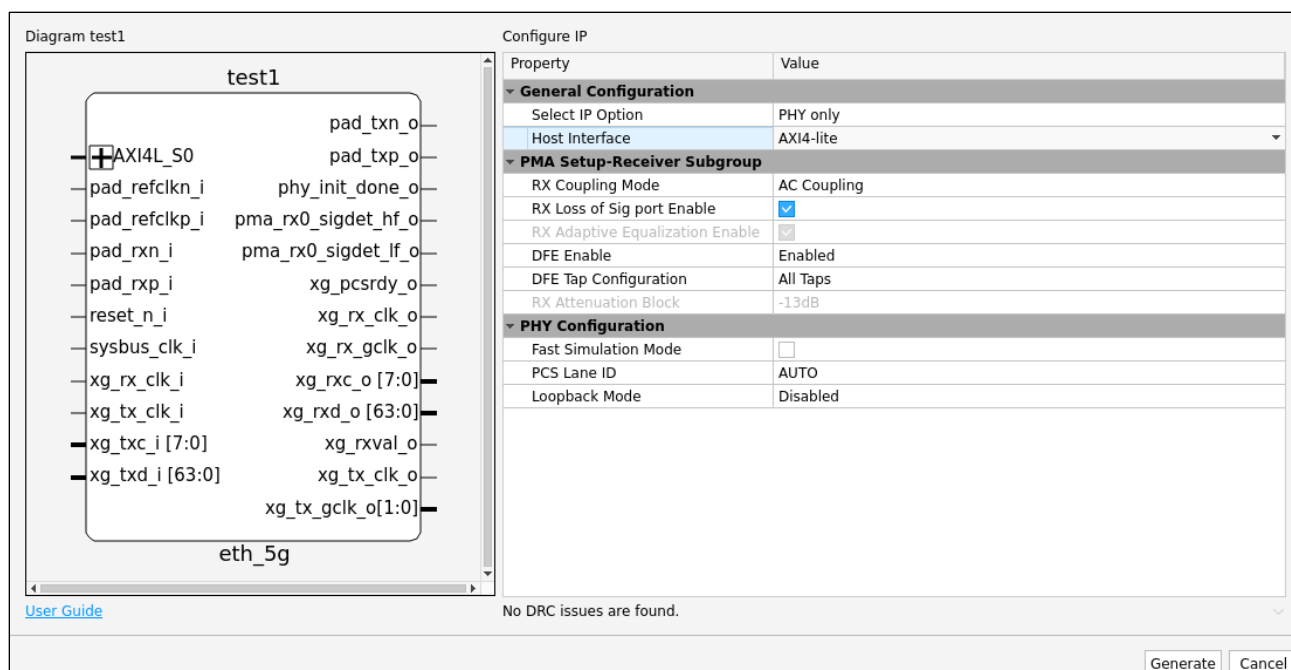


Figure 2.17. Set the Lane ID Configuration of the 5G Ethernet PHY IP Core

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-G/X70. An example of Lane ID numbering is shown in the figure below.

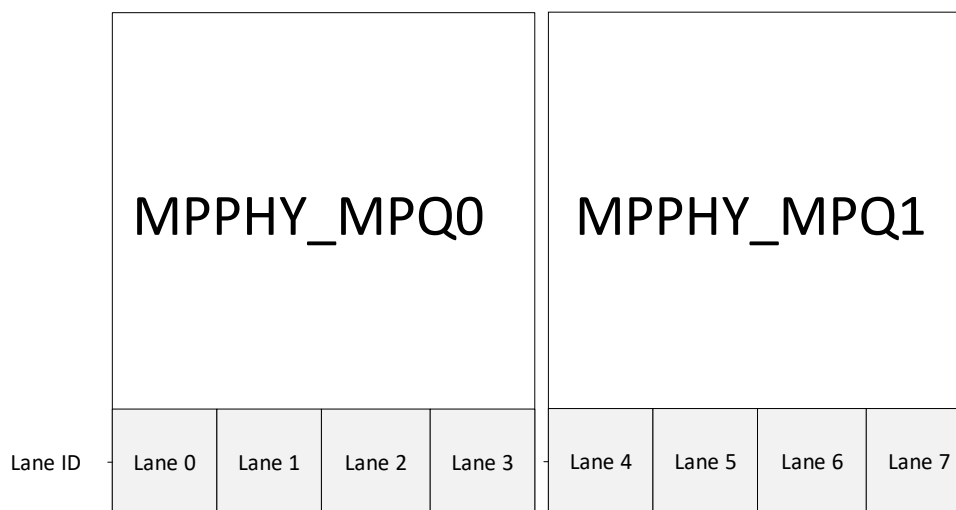


Figure 2.18. Lane ID Numbering

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.

```

mpphy_lane_assignment.mrf - Notepad
File Edit Format View Help
MPLL: -- un-assigned --

MPPHYX4_MPQ0: new instance 'MPPHYX4_MPQ0_merge'
  ~ Lane Assignment Within Quad ~
    Lane 3: 'inst2/lsc_ten_gbe_phy_mac_inst/genblk1.genblk1.u_ten_gbe_phy/mpphy_03A_inst/lsc_mpphy_inst/mpp_nonpipe_x1.u_nonpipe_x1' (
      Assignment is due to:
        - LANE_ID=3 parameter assignment
    Lane 2: -- un-assigned --
    Lane 1: -- un-assigned --
    Lane 0: 'inst1/lsc_ten_gbe_phy_mac_inst/genblk1.genblk1.u_ten_gbe_phy/mpphy_03A_inst/lsc_mpphy_inst/mpp_nonpipe_x1.u_nonpipe_x1' (
      Assignment is due to:
        - auto-assigned

  ~ PLL Assignment Within Quad (determined by protocol/data rate) ~
    MPLLA: inst2/lsc_ten_gbe_phy_mac_inst/genblk1.genblk1.u_ten_gbe_phy/mpphy_03A_inst/lsc_mpphy_inst/mpp_nonpipe_x1.u_nonpipe_x1, in
    MPLLB: -- un-assigned --

```

Figure 2.19. Lane Merging Report File

2.3.3.4. Restriction 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/data rate dependent, and will be 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.

The table below shows the signal mapping of the 10G Ethernet IP to MPPHY that must be shared for lane merging to work. All input must be driven by the same source.

For output clock, use only 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 the clock to drive any logic. Refer to the table below.

Table 2.1. Shared Signal Mapping of 5G Ethernet IP to MPPHY for Lane Merging

MPPHY	2.5G Ethernet IP	Direction
Immickl_q0_i	sysbus_clk_i	Input
refclk_p_q0_i	pad_refclkp_i	Input
refclk_n_q0_i	pad_refclk_n_i	
Immireset_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 you select AUTO lane ID, your pin location assignment must be included in sdc or ldc instead of pdc.

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, or APB interface. This block is responsible for register access to the PCS registers. While all the two interfaces (AXI4-Lite, or APB) are available for PCS Module 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 will need to 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 will be mapped 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 2.20](#).

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.

With APB interfaces, no address remapping is required. The actual PCS Module Registers addresses (0x5000 – 0x507F) are used.

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
0xA004	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]

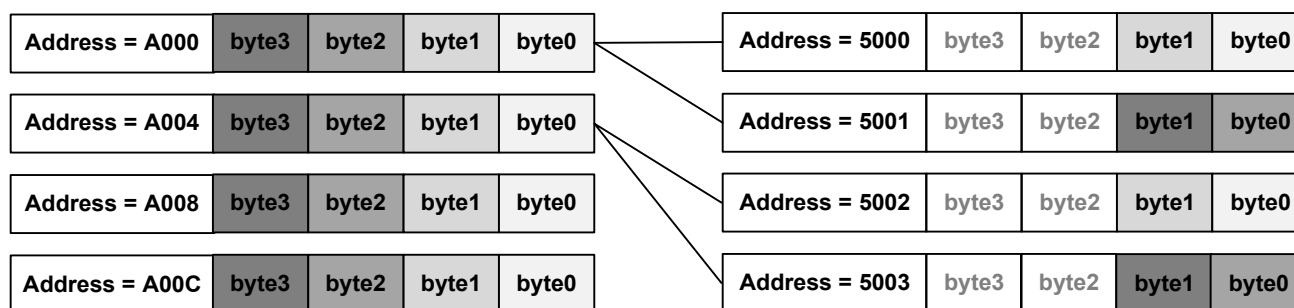


Figure 2.20. Remapped Addresses for PCS Register Address through the AXI4-Lite Interface

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.

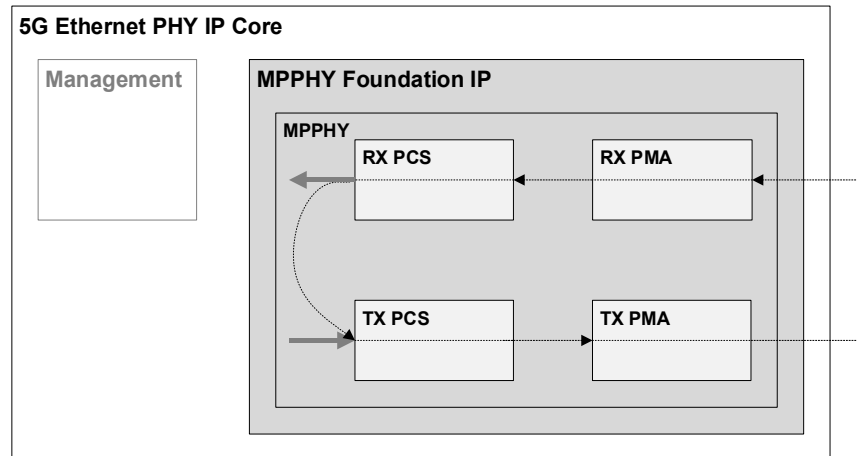


Figure 2.21. Far End Parallel Loopback

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.

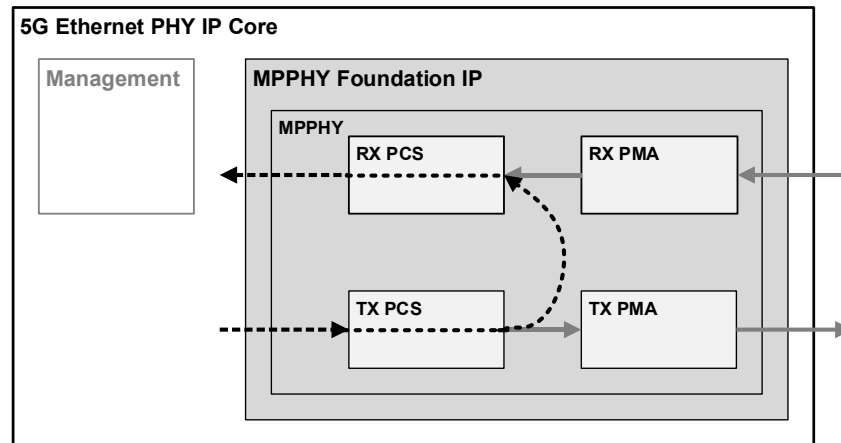


Figure 2.22. Near End Parallel Loopback

2.4. Clocking

2.4.1. Clocking Overview

The following figure shows the clock network of the 5G Ethernet IP core for Avant devices. The clock frequency requirements are described in the [Signal Description](#) section according to the configuration selected—MAC only, PHY only, or MAC+PHY. In the MAC + PHY configuration, the txmac_clk_i and rxmac_clk_i are connected internally to the output clock from the PHY block.

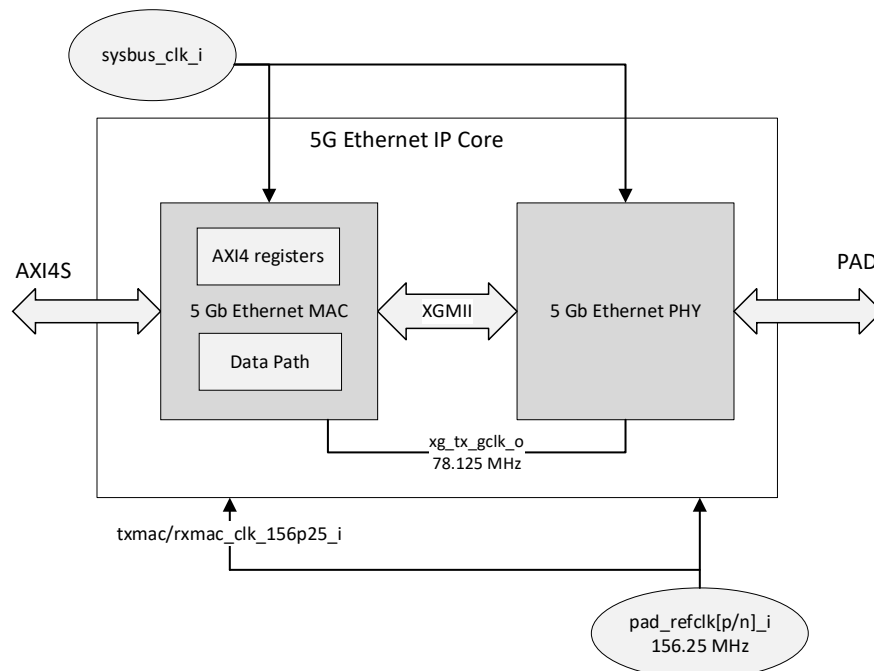


Figure 2.23. 5G Clock Network Diagram for Avant Devices

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 5G 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*.

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.
2. Wait for the 5G 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. The MAC is ready to receive transfers.

* **Note:** The fastest clock refers to the sysbus_clk_i with a frequency of 100 MHz.

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

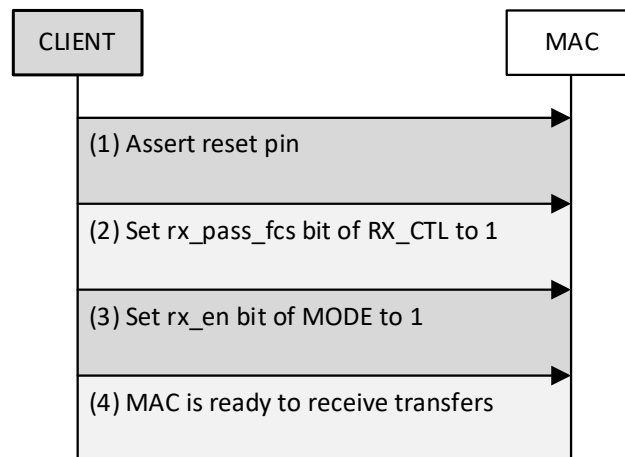


Figure 2.24. Sequence to Configure RX MAC In-Band FCS Passing

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 5GbE 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. Sequence

The following lists 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.
2. Wait for phy_init_done_o to assert. It is expected during this time (before T0), xg_tx_out_clk_o and xg_tx_out_clk_o are not toggling yet.
3. After T0, user drives continuous IDLE patterns until xg_pcsrdy_o asserts.

4. After T1, the PHY is ready to transmit data packets.

* **Note:** The fastest clock refers to the sysbus_clk_i with a frequency of 100 MHz.

The following figure shows the PHY initialization sequence.

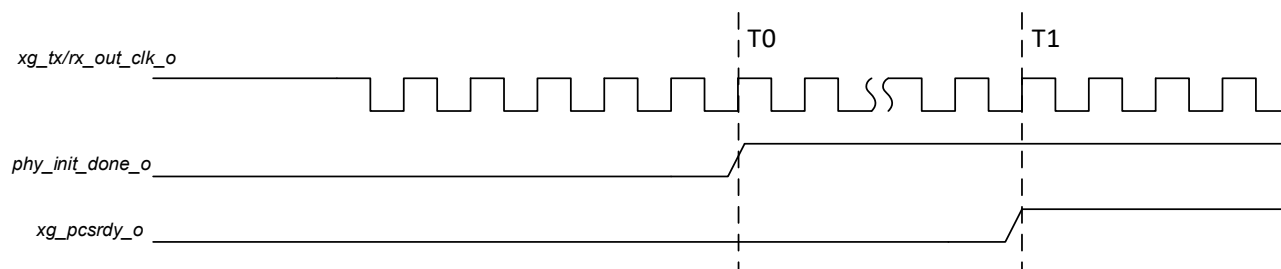


Figure 2.25. PHY Initialization Sequence

2.6. Latency

The following table provides the measured latency information for 5G Ethernet IP.

Table 2.3. MAC + PHY Attributes

Core	Core Configuration	Latency (ns)	User Bus Width (bits)	Core Clock Frequency (MHz)
MAC + PHY	5G BASE-R TX	252	32	156.25
MAC + PHY	5G BASE-R RX	204	32	156.25

3. IP Parameter Description

The configurable attributes of the 5G Ethernet IP core are shown in the following tables. You can configure the IP core by setting the attributes accordingly in the IP Catalog's Module/IP wizard of the Lattice Radiant software.

3.1. MAC + PHY

The following table lists the 5G Ethernet IP core configurable attributes for the *MAC + PHY* option. The values set for attributes with corresponding registers serve as the maximum values and cannot be set higher during dynamic reconfiguration. Select IP Option—*MAC + PHY* option.

Table 3.1. MAC + PHY Attributes

Attribute	Selectable Values	Default	Description	Dependency on Other Attributes
General Configuration				
Select IP Option	<ul style="list-style-type: none"> MAC + PHY PHY Only MAC Only 	MAC + PHY	IP option.	—
Host Interface	<ul style="list-style-type: none"> AXI4-Lite APB 	AXI4-Lite	Set the register interface.	—
MAC Configuration				
Multicast Address Filtering	<ul style="list-style-type: none"> Enabled Disabled 	Disabled	Enables or disables address filtering for multicast frames.	Enabled when <i>Select IP Option == MAC Only</i> or <i>MAC + PHY</i>
TX Pause Frame Generation via Ports	<ul style="list-style-type: none"> Enabled Disabled 	Disabled	Enables or disables TX pause frame generation via ports.	Enabled when <i>Select IP Option == MAC Only</i> or <i>MAC + PHY</i>
Statistics Counter Configuration				
Statistic Counter Registers	<ul style="list-style-type: none"> Enabled Disabled 	Disabled	Enables or disables statistics counter registers	Enabled when <i>Select IP Option == MAC Only</i> or <i>MAC + PHY</i>
Counter Width	<ul style="list-style-type: none"> 32 64 	32	Statistics counters register size.	Enabled when <i>Select IP Option == MAC Only</i> or <i>MAC + PHY</i>
TX Statistics	<ul style="list-style-type: none"> Enabled Disabled 	Disabled	TX statistics.	Enabled when <i>Select IP Option == MAC Only</i> or <i>MAC + PHY</i>
RX Statistics	<ul style="list-style-type: none"> Enabled Disabled 	Disabled	RX statistics.	Enabled when <i>Select IP Option == MAC Only</i> or <i>MAC + PHY</i>
PMA Setup-Transmitter Subgroup (default values are recommended)				
TX Amplitude Control [0-63]	0-63	24	Transmitter amplitude adjustment control.	Enabled when <i>Select IP Option == PHY Only</i> or <i>MAC + PHY</i>
TX Pre Cursor [0-31]	0-31	0	Transmitter pre-emphasis level adjustment control.	Enabled when <i>Select IP Option == PHY Only</i> or <i>MAC + PHY</i>
TX Post Cursor [0-31]	0-31	0	Transmitter post-emphasis level adjustment control.	Enabled when <i>Select IP Option == PHY Only</i> or <i>MAC + PHY</i>

Attribute	Selectable Values	Default	Description	Dependency on Other Attributes
PMA Setup-Receiver Subgroup (default values are recommended)				
RX Coupling Mode	<ul style="list-style-type: none"> AC Coupling DC Coupling 	AC Coupling	PMA Coupling mode.	Enabled when <i>Select IP Option == PHY Only</i> or <i>MAC + PHY</i>
RX Loss of Sig port Enable	<ul style="list-style-type: none"> Enabled Disabled 	Enabled	RX Loss of Sig capability.	Enabled when <i>Select IP Option == PHY Only</i> or <i>MAC + PHY</i>
DFE Enable	<ul style="list-style-type: none"> Enabled Disabled 	Enabled	DFE capability.	Enabled when <i>Select IP Option == PHY Only</i> or <i>MAC + PHY</i>
RX Adaptive Equalization Enable	<ul style="list-style-type: none"> Enabled Disabled 	Enabled	RX adaptive EQ capability.	Enabled when <i>Select IP Option == PHY Only</i> or <i>MAC + PHY</i> Selectable only when <i>DFE Enable == Disabled</i>
RX Attenuation Block	<ul style="list-style-type: none"> -13dB -2dB 	-13dB	RX Attenuation setting. Higher dB loss reflects higher capability of the RX to handle a lossy signal.	Enabled when <i>Select IP Option == PHY Only</i> or <i>MAC + PHY</i> Selectable only when <i>RX Adaptive Equalization Enable == Disabled</i>
PHY Configuration				
Fast Simulation Mode	<ul style="list-style-type: none"> Enabled Disabled 	Disabled	Enables or disables the fast sim mode.	Enabled when <i>Select IP Option == PHY Only</i> or <i>MAC + PHY</i>
PCS Lane ID	<ul style="list-style-type: none"> AUTO 0-27 	AUTO	Specifies the Lane ID.	Enabled when <i>Select IP Option == PHY Only</i> or <i>MAC + PHY</i>
Loopback Mode	<ul style="list-style-type: none"> Far End Parallel Loopback Near End Parallel Loopback No Loopback 	No Loopback	Enables the Far End Parallel Loopback or Near End Parallel Loopback or No Loopback.	Enabled when <i>Select IP Option == PHY Only</i> or <i>MAC + PHY</i>

3.2. MAC Only

The following table lists the 5Gb Ethernet IP core configurable attributes for the *MAC Only* option. The values set for attributes with corresponding registers serves as the maximum values and cannot set higher than these values during dynamic reconfiguration. Select IP Option—*MAC Only* option.

Table 3.2. MAC Only Attributes

Attribute	Selectable Values	Default	Description	Dependency on Other Attributes
General Configuration				
Select IP Option	<ul style="list-style-type: none"> MAC + PHY PHY Only MAC Only 	MAC + PHY	IP option.	—
Host Interface	<ul style="list-style-type: none"> AXI4-Lite APB 	AXI4-Lite	Set the register interface.	—

Attribute	Selectable Values	Default	Description	Dependency on Other Attributes
MAC Configuration				
Multicast Address Filtering	<ul style="list-style-type: none"> Enabled Disabled 	Disabled	Enables or disables address filtering for Multicast frames.	Enabled when <i>Select IP Option == MAC Only</i> or <i>MAC + PHY</i>
TX Pause Frame Generation via Ports	<ul style="list-style-type: none"> Enabled Disabled 	Disabled	Enables or disables TX pause frame generation via ports.	Enabled when <i>Select IP Option == MAC Only</i> or <i>MAC + PHY</i>
Statistics Counter Configuration				
Statistics Counter Registers	<ul style="list-style-type: none"> Enabled Disabled 	Disabled	Enables or disables statistics counter registers.	Enabled when <i>Select IP Option == MAC Only</i> or <i>MAC + PHY</i>
Counter Width	<ul style="list-style-type: none"> 32 64 	32	Statistics counters register size.	Enabled when <i>Select IP Option == MAC Only</i> or <i>MAC + PHY</i>
TX Statistics	<ul style="list-style-type: none"> Enabled Disabled 	Disabled	TX statistics.	Enabled when <i>Select IP Option == MAC Only</i> or <i>MAC + PHY</i>
RX Statistics	<ul style="list-style-type: none"> Enabled Disabled 	Disabled	RX statistics.	Enabled when <i>Select IP Option == MAC Only</i> or <i>MAC + PHY</i>

3.3. PHY Only

The following table lists the 5G Ethernet IP core configurable attributes for the *PHY Only* option in Avant devices. Select IP Option—*PHY Only* option.

Table 3.3. PHY Only Attributes

Attribute	Selectable Values	Default	Description	Dependency on Other Attributes
General Configuration				
Select IP Option	<ul style="list-style-type: none"> MAC + PHY PHY Only MAC Only 	MAC + PHY	IP option.	—
Host Interface	<ul style="list-style-type: none"> AXI4-Lite APB 	AXI4-Lite	Set the register interface.	—
PMA Setup-Receiver Subgroup (default values are recommended)				
RX Coupling Mode	<ul style="list-style-type: none"> AC Coupling DC Coupling 	AC Coupling	PMA Coupling mode.	Enabled when <i>Select IP Option == PHY Only</i> or <i>MAC + PHY</i>
RX Loss of Sig port Enable	<ul style="list-style-type: none"> Enabled Disabled 	Enabled	RX Loss of Sig capability.	Enabled when <i>Select IP Option == PHY Only</i> or <i>MAC + PHY</i>
DFE Enable	<ul style="list-style-type: none"> Enabled Disabled 	Enabled	DFE capability.	Enabled when <i>Select IP Option == PHY Only</i> or <i>MAC + PHY</i>
RX Adaptive Equalization Enable	<ul style="list-style-type: none"> Enabled Disabled 	Enabled	RX adaptive EQ capability.	Enabled when <i>Select IP Option == PHY Only</i> or <i>MAC + PHY</i> Selectable only when <i>DFE Enable ==</i>

Attribute	Selectable Values	Default	Description	Dependency on Other Attributes
				<i>Disabled</i>
RX Attenuation Block	<ul style="list-style-type: none"> -13dB -2dB 	-13dB	RX Attenuation setting. Higher dB loss reflects higher capability of the RX to handle a lossy signal	Enabled when <i>Select IP Option == PHY Only</i> or <i>MAC + PHY</i> Selectable only when <i>RX Adaptive Equalization Enable == Disabled</i>
PHY Configuration				
Fast Simulation Mode	<ul style="list-style-type: none"> Enabled Disabled 	Disabled	Enables or disables the fast sim mode	Enabled when <i>Select IP Option == PHY Only</i> or <i>MAC + PHY</i>
PCS Lane ID	For Avant-AT-G/X devices: <ul style="list-style-type: none"> AUTO 0-27 	AUTO	Specifies the Lane ID.	Enabled when <i>Select IP Option == PHY Only</i> or <i>MAC + PHY</i>
Loopback Mode	<ul style="list-style-type: none"> Far End Parallel Loopback Near End Parallel Loopback No Loopback 	No Loopback	Enables the Far End Parallel Loopback or Near End Parallel Loopback or No Loopback.	Enabled when <i>Select IP Option == PHY Only</i> or <i>MAC + PHY</i>

4. Signal Description

This section describes the 5G Ethernet IP core ports.

4.1. MAC + PHY Signals

Table 4.1. Signal Description—MAC + PHY

Port Name	Clock Domain	I/O	Width	Description
Clock and Reset				
reset_n_i	Asynchronous	In	1	Active-low asynchronous reset.
txmac_clk_156p25_i	—	In	1	TX MAC clock input (156.25 MHz). Ensure that this clock is at 0 ppm.
rxmac_clk_156p25_i	—	In	1	RX MAC clock input (156.25 MHz). Ensure that this clock is at 0 ppm.
rxmac_clk_78p125_i	—	In	1	Clock for 64-bits XGMII from PHY. 78.125 MHz clock for XGMII interface. For XGMII configuration, use xg_tx_gclk_o. For lane merging, use the same xg_tx_gclk.
txmac_clk_78p125_i	—	In	1	Clock for 64-bits XGMII data path. 78.125 MHz clock for XGMII interface. For XGMII configuration, use xg_tx_gclk_o. For lane merging, use the same xg_tx_gclk.
xg_tx_clk_o	—	Out	1	TX clock output (78.125 MHz).
xg_rx_clk_o	—	Out	1	RX clock output (78.125 MHz).
xg_tx_gclk_o	—	Out	2	TX clock forwarded to global clock distribution (78.125 MHz).
xg_rx_gclk_o	—	Out	1	RX clock forwarded to global clock distribution (78.125 MHz).
sysbus_clk_i	—	In	1	Clock for the Management module (AXI4-Lite/APB interface). The fastest frequency is 100 MHz for XGMII interface.
Serial I/O				
pad_refclk_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_refclk_i	In	1	RX- differential signal.
pad_rxp_i	pad_refclkp_i	In	1	RX+ differential signal.
pad_txn_o	pad_refclk_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	txmac_clk_156p25_i ⁴	Out	32	AXI4-Stream data from PHY to the client.
axis_rx_tkeep_o	txmac_clk_156p25_i ⁴	Out	4	AXI4-Stream control that indicates which data is valid on axis_rx_tdata_o[]. <ul style="list-style-type: none"> 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_tvalid_o	txmac_clk_156p25_i ⁴	Out	1	AXI4-Stream data valid.
axis_rx_tuser_o	txmac_clk_156p25_i ⁴	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	txmac_clk_156p25_i ⁴	Out	1	AXI4-Stream signal indicating the end of a packet.

Port Name	Clock Domain	I/O	Width	Description
AXI4-Stream Transmit Interface				
axis_tx_tready_o	txmac_clk_156p25_i ⁴	Out	1	AXI4-Stream signal indicating that the core can accept data transfer.
axis_tx_tdata_i	txmac_clk_156p25_i ⁴	In	32	AXI4-Stream data from the client.
axis_tx_tkeep_i	txmac_clk_156p25_i ⁴	In	4	AXI4-Stream control that indicates which data is valid on axis_tx_tdata_i []. <ul style="list-style-type: none"> 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_tvalid_i	txmac_clk_156p25_i ⁴	In	1	AXI4-Stream data valid.
axis_tx_tuser_i	txmac_clk_156p25_i ⁴	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	txmac_clk_156p25_i ⁴	In	1	AXI4-Stream signal indicating the end of a packet.
Non-Standard RX/TX Signals				
xg_rxval_o	txmac_clk_78p125_i ⁵	Out	1	RX valid signal. When high, indicates that the corresponding XGMII signals from PHY are valid.
xg_pcsrdy_o	txmac_clk_78p125_i ⁵	Out	1	The high level of this signal indicates alignment is achieved and RX PCS is ready for transaction.
phy_init_done_o	txmac_clk_78p125_i ⁵	Out	1	The high level of this signal indicates alignment is PHY initialization or reset is done.
pma_rx0_sigdet_hf_o	txmac_clk_78p125_i ⁵	Out	1	Receive high-frequency signal detection output. When asserted, indicates that the receiver is receiving high-frequency signals.
pma_rx0_sigdet_lf_o	txmac_clk_78p125_i ⁵	Out	1	Receive low-frequency signal detection output. When asserted, indicates that the receiver is receiving low-frequency signals.
APB Interface¹				
apb_psel_i	sysbus_clk_i	In	1	Select signal. Indicates that the completer device is selected, and a data transfer is required.
apb_paddr_i	sysbus_clk_i	In	32	Address signal.
apb_pwdata_i	sysbus_clk_i	In	32	Write data signal.
apb_pwrite_i	sysbus_clk_i	In	1	Direction signal. Write = 1, Read = 0.
apb_penable_i	sysbus_clk_i	In	1	Enable signal. Indicates the second and subsequent cycles of an APB transfer.
apb_pready_o	sysbus_clk_i	Out	1	Ready signal. Indicates transfer completion. The completer uses this signal to extend an APB transfer.
apb_prdata_o	sysbus_clk_i	Out	32	Read data signal.
apb_pslverr_o	sysbus_clk_i	Out	1	Transfer success/failure signal. Tie to low.
apb_psel_i	sysbus_clk_i	In	1	Select signal. Indicates that the completer device is selected, and a data transfer is required.
apb_paddr_i	sysbus_clk_i	In	32	Address signal.
apb_pwdata_i	sysbus_clk_i	In	32	Write data signal.
apb_pwrite_i	sysbus_clk_i	In	1	Direction signal. Write = 1, Read = 0.
AXI4-Lite Interface²				
axi4l_awaddr_i	sysbus_clk_i	In	32	Write address bus.
axi4l_awvalid_i	sysbus_clk_i	In	1	Write address valid.
axi4l_awready_o	sysbus_clk_i	Out	1	Write address acknowledge.
axi4l_wdata_i	sysbus_clk_i	In	32	Write data bus.

Port Name	Clock Domain	I/O	Width	Description
axi4l_wvalid_i	sysbus_clk_i	In	1	Write data valid.
axi4l_wready_o	sysbus_clk_i	Out	1	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	Write transaction response.
axi4l_bvalid_o	sysbus_clk_i	Out	1	Write response valid.
axi4l_bready_i	sysbus_clk_i	In	1	Write response acknowledge.
axi4l_araddr_i	sysbus_clk_i	In	32	Read address bus.
axi4l_arvalid_i	sysbus_clk_i	In	1	Read address valid.
axi4l_arready_o	sysbus_clk_i	Out	1	Read address acknowledge.
axi4l_rdata_o	sysbus_clk_i	Out	32	Read data output.
axi4l_rresp_o	sysbus_clk_i	Out	2	Read data response.
axi4l_rvalid_o	sysbus_clk_i	Out	1	Read data/response valid.
axi4l_rready_i	sysbus_clk_i	In	1	Read data acknowledge.
Statistics Vector Interface				
tx_statvec_o	txmac_clk_156p25_i ⁴	Out	28	<p>Contains information on the frame transmitted. This bus is qualified by the tx_staten_o signal.</p> <ul style="list-style-type: none"> 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. tx_statvec_o[27]: Reserved for future use.
tx_staten_o	txmac_clk_156p25_i ⁴	Out	1	When asserted, indicates that the contents of the tx_statvec_o bus are valid. This signal is asserted for three txmac_clk_i periods.
rx_statvec_o	txmac_clk_156p25_i ⁴	Out	28	<p>Contains information on the frame received. This bus is qualified by the rx_staten_o signal.</p> <ul style="list-style-type: none"> 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 opcode for pause frame is 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]: Reserved for future use.

Port Name	Clock Domain	I/O	Width	Description
rx_staten_o	txmac_clk_156p25_i ⁴	Out	1	When asserted, indicates that the contents of the rx_statvec_o bus are valid. This signal is asserted for three rxmac_clk_i periods.
TX Pause Frame³				
tx_pausreq_i	txmac_clk_156p25_i ⁴	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. Subsequent pause request will not be serviced if the pause frame of the first request is not transmitted yet.
tx_paustm_i	txmac_clk_156p25_i ⁴	In	16	Pause Quanta time. This is equivalent to the tx_paustim bit of PAUSE_TM register.

Notes:

1. Available when APB is selected as the Host Interface.
2. Available when AXI4-Lite is selected as the Host Interface.
3. Available when TX Pause Frame Generation via Ports is enabled.
4. It is recommended to use txmac_clk_156p25_i.
5. It is recommended to use txmac_clk_78p125_i.

4.2. MAC Only Signals

Table 4.2. Signal Description—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_156p25_i	—	In	1	Clock for Receive 32-bits AXI4-Stream. 156.25 MHz clock for XGMII interface.
rxmac_clk_78p125_i	—	In	1	Clock for 64-bits XGMII from PHY. 78.125 MHz clock for XGMII interface. For XGMII configuration, it is recommended to use xg_tx_gclk_o.
txmac_clk_156p25_i	—	In	1	Clock for Transmit 32-bits AXI4-Stream.
txmac_clk_78p125_i	—	In	1	Clock for 64-bits XGMII data path. 78.125 MHz clock for XGMII interface. For XGMII configuration, it is recommended to use xg_tx_gclk_o.
sysbus_clk_i	—	In	1	Clock for Management module (AXI4L/APB interface). The fastest clock frequency is 100 MHz clock .
XGMII Interface				
xgmii_rxd_i	txmac_clk_78p125_i/ rxmac_clk_78p125_i ⁴	In	64	8-lane Receive SDR XGMII data from PHY. <ul style="list-style-type: none"> • Lane 0: xgmii_rxd_i[7:0] • Lane 1: xgmii_rxd_i[15:8] • Lane 2: xgmii_rxd_i[23:16] • Lane 3: xgmii_rxd_i[31:24] • Lane 4: xgmii_rxd_i[39:32] • Lane 5: xgmii_rxd_i[47:40] • Lane 6: xgmii_rxd_i[55:48] • Lane 7: xgmii_rxd_i[63:56]
xgmii_rxc_i	txmac_clk_78p125_i/ rxmac_clk_78p125_i	In	8	Control bits for each lane in xgmii_rxd_i[]. <ul style="list-style-type: none"> • Lane 0: xgmii_rxc_i[0] • Lane 1: xgmii_rxc_i[1]

Port Name	Clock Domain	I/O	Width	Description
	5_i ⁴			<ul style="list-style-type: none"> Lane 2: xgmii_rxc_i[2] Lane 3: xgmii_rxc_i[3] Lane 4: xgmii_rxc_i[4] Lane 5: xgmii_rxc_i[5] Lane 6: xgmii_rxc_i[6] Lane 7: xgmii_rxc_i[7]
xgmii_txd_o	txmac_clk_78p12 5_i/ rxmac_clk_78p12 5_i ⁴	Out	64	8-lane Transmit SDR XGMII data to PHY. <ul style="list-style-type: none"> Lane 0: xgmii_txd_o[7:0] Lane 1: xgmii_txd_o[15:8] Lane 2: xgmii_txd_o[23:16] Lane 3: xgmii_txd_o[31:24] Lane 4: xgmii_txd_o[39:32] Lane 5: xgmii_txd_o[47:40] Lane 6: xgmii_txd_o[55:48] Lane 7: xgmii_txd_o[63:56]
xgmii_txc_o	txmac_clk_78p12 5_i/ rxmac_clk_78p12 5_i ⁴	Out	8	Control bits for each lane in xgmii_txd_o[]. <ul style="list-style-type: none"> Lane 0: xgmii_txc_o[0] Lane 1: xgmii_txc_o[1] Lane 2: xgmii_txc_o[2] Lane 3: xgmii_txc_o[3] Lane 4: xgmii_txc_o[4] Lane 5: xgmii_txc_o[5] Lane 6: xgmii_txc_o[6] Lane 7: xgmii_txc_o[7]
AXI4-Stream Receive Interface				
axis_rx_tdata_o	txmac_clk_156p2 5_i ³	Out	32	AXI4-Stream data from PHY to client.
axis_rx_tkeep_o	txmac_clk_156p2 5_i ³	Out	4	AXI4-Stream control that indicates which data is valid on axis_rx_tdata_o[]. <ul style="list-style-type: none"> 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_tvalid_o	txmac_clk_156p2 5_i ³	Out	1	AXI4-Stream data valid.
axis_rx_tuser_o	txmac_clk_156p2 5_i ³	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	txmac_clk_156p2 5_i ³	Out	1	AXI4-Stream signal indicating the end of a packet.
AXI4-Stream Transmit Interface				
axis_tx_tready_o	txmac_clk_156p2 5_i ³	Out	1	AXI4-Stream signal indicating that the core can accept data transfer.
axis_tx_tdata_i	txmac_clk_156p2 5_i ³	In	32	AXI4-Stream data from client.
axis_tx_tkeep_i	txmac_clk_156p2 5_i ³	In	4	AXI4-Stream control that indicates which data is valid on axis_tx_tdata_i []. <ul style="list-style-type: none"> 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_tvalid_i	txmac_clk_156p2	In	1	AXI4-Stream data valid.

Port Name	Clock Domain	I/O	Width	Description
	5_i ³			
axis_tx_tuser_i	txmac_clk_156p2 5_i ³	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	txmac_clk_156p2 5_i ³	In	1	AXI4-Stream signal indicating the end of a packet.
APB Interface¹				
apb_psel_i	sysbus_clk_i	In	1	Select signal. Indicates that the completer device is selected and a data transfer is required.
apb_paddr_i	sysbus_clk_i	In	32	Address signal.
apb_pwdata_i	sysbus_clk_i	In	32	Write data signal.
apb_pwrite_i	sysbus_clk_i	In	1	Direction signal. Write = 1, Read = 0
apb_penable_i	sysbus_clk_i	In	1	Enable signal. Indicates the second and subsequent cycles of an APB transfer.
apb_pready_o	sysbus_clk_i	Out	1	Ready signal. Indicates transfer completion. The completer uses this signal to extend an APB transfer.
apb_prdata_o	sysbus_clk_i	Out	32	Read data signal.
apb_pslverr_o	sysbus_clk_i	Out	1	Transfer success or failure signal. Tie to low.
AXI4-Lite Interface				
axi4l_awaddr_i	sysbus_clk_i	In	32	Write address bus.
axi4l_awvalid_i	sysbus_clk_i	In	1	Write address valid.
axi4l_awready_o	sysbus_clk_i	Out	1	Write address acknowledge.
axi4l_wdata_i	sysbus_clk_i	In	32	Write data bus.
axi4l_wvalid_i	sysbus_clk_i	In	1	Write data valid.
axi4l_wready_o	sysbus_clk_i	Out	1	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	Write transaction response.
axi4l_bvalid_o	sysbus_clk_i	Out	1	Write response valid.
axi4l_bready_i	sysbus_clk_i	In	1	Write response acknowledge.
axi4l_araddr_i	sysbus_clk_i	In	32	Read address bus.
axi4l_arvalid_i	sysbus_clk_i	In	1	Read address valid.
axi4l_arready_o	sysbus_clk_i	Out	1	Read address acknowledge.
axi4l_rdata_o	sysbus_clk_i	Out	32	Read data output.
axi4l_rresp_o	sysbus_clk_i	Out	2	Read data response.
axi4l_rvalid_o	sysbus_clk_i	Out	1	Read data/response valid.
axi4l_rready_i	sysbus_clk_i	In	1	Read data acknowledge.
Statistics Vector Interface				
tx_statvec_o	txmac_clk_156p2 5_i ³	Out	28	<p>Contains information on the frame transmitted. This bus is qualified by the tx_staten_o signal.</p> <ul style="list-style-type: none"> 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

Port Name	Clock Domain	I/O	Width	Description
				<ul style="list-style-type: none"> tx_statvec_o[25]: Long Frame error tx_statvec_o[26]: PTP1588 frame tx_statvec_o[27]: Reserved for future use
tx_statn_o	txmac_clk_156p25_i ³	Out	1	When asserted, indicates that the contents of the tx_statvec_o bus are valid. This signal is asserted for three txmac_clk_i periods.
rx_statvec_o	txmac_clk_156p25_i ³	Out	28	<p>Contains information on the frame received. This bus is qualified by the rx_statn_o signal.</p> <ul style="list-style-type: none"> 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 opcode for PAUSE frame is 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]: Reserved for future use
rx_statn_o	txmac_clk_156p25_i ³	Out	1	When asserted, indicates that the contents of the rx_statvec_o bus are valid. This signal is asserted for three rxmac_clk_i periods.
xg_pcsrdy_i	txmac_clk_156p25_i ³	In	1	This is an active-high level status signal from external PHY. Asserted when the PHY link status is ready. In cases where the local fault/remote fault is sending an earlier signal such as during the reset state, then this signal is used by MAC to determine the PHY link status when out of reset. As a result, the RX_STAT_LINK_OK counter is increased.
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 Frame²				
tx_pausreq_i	txmac_clk_156p25_i ³	In	1	<p>Transmit PAUSE frame.</p> <p>1 = send request, 0 = do not send request. This is a positive edge-triggered bit.</p> <p>The tx_fc_en bit of TX_CTL register should be set to 1 to enable this feature.</p> <p>When this port is enabled, the tx_pausreq bit of MAC_CTL register is not accessible.</p> <p>Subsequent pause request will not be serviced if the pause frame of the first request is not transmitted yet.</p>
tx_paustm_i	txmac_clk_156p25_i ³	In	16	<p>Pause Quanta time.</p> <p>This is equivalent to the tx_paustim bit of the PAUSE_TM register.</p>
rx_pause_cntr_o	txmac_clk_156p25_i ³	Out	19	<p>Received Pause counter.</p> <p>This signal is only available in IP option = <i>MAC only</i>. Each Pause Quanta corresponds to 512 bit times. In 5G Ethernet IP, the 512 bit times is 16 valid clock cycles. Hence, the counter value is equivalent to quanta time multiplied by 16.</p>
rx_pause_req_o	txmac_clk_156p25_i ³	Out	1	<p>Received pause request.</p> <p>This signal is only available in IP option = <i>MAC only</i>. Used to</p>

Port Name	Clock Domain	I/O	Width	Description
				indicate there is a pause frame detected.
rx_pause_time_o	txmac_clk_156p25_i ³	Out	16	Received Pause Quanta time. This signal is only available in IP option = <i>MAC only</i> . This value represents the requested pause time from the pause frame. Each pause quanta corresponds to 512 bit times.

Notes:

1. Available when APB is selected as Host Interface.
2. Available when TX Pause Frame Generation via Ports is enabled.
3. It is recommended to use txmac_clk_156p25_i.
4. This clock source must drive from PHY.

4.3. PHY Only Signals

Table 4.3. Signal Description—PHY Only

Port Name	Clock Domain	I/O	Width	Description
Serial I/O				
pad_refclk_i	—	In	1	78.125 MHz SERDES PLL reference clock signal (-).
pad_refclkp_i	—	In	1	78.125 MHz SERDES PLL reference clock signal (+).
pad_rxn_i	pad_refclk_i	In	1	RX- differential signal.
pad_rxp_i	pad_refclkp_i	In	1	RX+ differential signal.
pad_txn_o	pad_refclk_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 – 78.125 MHz clock for transmit data path. It is recommended to use the xg_tx_gclk_o.
xg_rx_clk_i	—	In	1	RX clock input – 78.125 MHz clock for receive data path. It is recommended to use the xg_tx_gclk_o.
xg_tx_clk_o	—	Out	1	TX clock output (78.125 MHz).
xg_rx_clk_o	—	Out	1	RX clock output (78.125 MHz).
xg_tx_gclk_o	—	Out	2	TX clock forwarded to global clock distribution (78.125 MHz).
xg_rx_gclk_o	—	Out	1	RX clock forwarded to global clock distribution (78.125 MHz).
reset_n_i	Asynchronous	In	1	Active-low asynchronous reset.
sysbus_clk_i	—	In	1	Clock for Management module. The fastest clock frequency is 100 MHz.
XGMII				
xg_txc_i	xg_tx_clk_i	In	8	8-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[7] is the control signal for xg_txd_i [63:56] When control bit is high, indicates a control byte, otherwise it is a data byte.
xg_txd_i	xg_tx_clk_i	In	64	64-bit TX data signal.
xg_rxc_o	xg_tx_clk_i	Out	8	8-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[7] is the control signal for xg_rxd_o [63:56] When control bit is high, indicates a control byte, otherwise it is a data byte.
xg_rxd_o	xg_tx_clk_i	Out	64	64-bit RX data signal.

Port Name	Clock Domain	I/O	Width	Description
Non-Standard RX/TX Signals				
xg_rxval_o	xg_tx_clk_i	Out	1	RX valid signal. When high, indicates that the corresponding values on signals xg_rxc_o and xg_rxd_o are valid.
xg_pcsrdy_o	xg_tx_clk_i	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_clk_i	Out	1	The high level of this signal indicates alignment is PHY initialization or reset is done.
pma_rx0_sigdet_hf_o	xg_tx_clk_i	Out	1	Receive high-frequency signal detection output. When asserted, indicates that the receiver is receiving high-frequency signals.
pma_rx0_sigdet_lf_o	xg_tx_clk_i	Out	1	Receive low-frequency signal detection output. When asserted, indicates that the receiver is receiving low-frequency signals.
AXI4-Lite Interface¹				
axi4l_awaddr_i	sysbus_clk_i	In	32	Write address bus.
axi4l_awvalid_i	sysbus_clk_i	In	1	Write address valid.
axi4l_awready_o	sysbus_clk_i	Out	1	Write address acknowledge.
axi4l_wdata_i	sysbus_clk_i	In	32	Write data bus.
axi4l_wvalid_i	sysbus_clk_i	In	1	Write data valid.
axi4l_wready_o	sysbus_clk_i	Out	1	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	Write transaction response.
axi4l_bvalid_o	sysbus_clk_i	Out	1	Write response valid.
axi4l_bready_i	sysbus_clk_i	In	1	Write response acknowledge.
axi4l_araddr_i	sysbus_clk_i	In	32	Read address bus.
axi4l_arvalid_i	sysbus_clk_i	In	1	Read address valid.
axi4l_arready_o	sysbus_clk_i	Out	1	Read address acknowledge.
axi4l_rdata_o	sysbus_clk_i	Out	32	Read data output.
axi4l_rresp_o	sysbus_clk_i	Out	2	Read data response.
axi4l_rvalid_o	sysbus_clk_i	Out	1	Read data/response valid.
axi4l_rready_i	sysbus_clk_i	In	1	Read data acknowledge.
APB Interface¹				
apb_psel_i	sysbus_clk_i	In	1	Select signal. Indicates that the completer device is selected and a data transfer is required.
apb_paddr_i	sysbus_clk_i	In	32	Address signal.
apb_pwdata_i	sysbus_clk_i	In	32	Write data signal.
apb_pwrite_i	sysbus_clk_i	In	1	Direction signal. Write = 1, Read = 0.
apb_penable_i	sysbus_clk_i	In	1	Enable signal. Indicates the second and subsequent cycles of an APB transfer.
apb_pready_o	sysbus_clk_i	Out	1	Ready signal. Indicates transfer completion. The completer uses this signal to extend an APB transfer.
apb_prdata_o	sysbus_clk_i	Out	32	Read data signal.
apb_pslverr_o	sysbus_clk_i	O	1	Transfer success/failure signal. Tie to low.

Note:

1. Only one of the two interfaces is available as selected by the *Host Interface*.

5. Register Description

This section describes the registers for the 5G Ethernet IP core.

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 5GbE 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 or APB interface is available for access to both PCS and MAC registers.

5.2. MAC Registers

All registers are accessed through the APB interface when the IP is configured as *MAC Only*.

Table 5.1. Access Types for MAC Only

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 5GbE MAC configuration registers.

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_LENGTH	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. When set to 0, the received AXIS TX data will be dropped internally.	0
[0]	rx_en	RX MAC Enable When set to 1, the RX MAC is enabled to receive frames. When set to 0, the received XGMII data packet will be dropped internally. However, remote fault packets are still able to be received.	0

5.2.1.2. TX_CTL Register

This register should be 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
[31:5]	Reserved	Reserved bits.	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. Note: Custom preamble mode is only supported by XGMII 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]	Reserved	Reserved bits.	0
[1]	tx_fc_en	Flow-control Enable. When set to 1, this enables the flow control functionality of the TX MAC. This bit must be set for the TX MAC to transmit a pause frame.	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 should be 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
[15:8]	Reserved	Reserved bits.	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. Note: Custom preamble mode is only supported by XGMII interface.	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, enables the RX MAC to receive frames shorter than 64 bytes.	0*

Bit	Label	Description	Default
[4]	receive_bc	Receive Broadcast Frames. When set to 1, it enables the 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 frames with any address.	0

* **Note:** If the L/T value is less than 46 bytes, it detects as short frame, and it will not be dropped. If the L/T value is less than 46 bytes but the payload is more than the defined value, then the extra payload will be treated as the padded byte.

5.2.1.4. MAX_PKT_LENTH Register

This register should be 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.

Table 5.6. MAX_PKT_LENTH Register

Bit	Label	Description	Default
[31:14]	Reserved	Reserved bits.	0
[13:0]	max_pkt_len	Maximum Frame Length. Used only for statistical purposes, all frames longer than the value given here are marked as long. This value does not affect the frame's reception.	0

5.2.1.5. IPG_VAL Register

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

Table 5.7. 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. A 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 5.8. MAC_ADDR_0 Register

Bit	Label	Description	Default
[31:0]	mac_addr_0	First four bytes of the MAC address. Ethernet address assigned to the port that is supported by the MAC.	0

Table 5.9. 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 5.10. 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 5.11. 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 5.12. 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 5.13. MC_TABLE_1 Register

Bit	Label	Description	Default
[31:0]	mc_table_1	Multicast Table Word 1. 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 5.14. 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 5.15. 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 should be set to 1 to enable this feature. 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 1,214.4 ns. (1518 bytes divided by 8 bytes per clock cycle = 1,214.4 ns) Super jumbo frame size of= 9,600 bytes requires a hold time of 7,680 ns. (9,600 bytes divided by 8 bytes per clock cycle = 7,680 ns) Clock cycle of 6.4 ns (per 8 bytes)	0

5.2.1.12. PAUSE_TM Register

This register has the pause time for a flow control packet sourced by the 5 Gb MAC transmitter.

Table 5.16. 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 5.17. 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 5.18. INT_STATUS Register

Bit	Label	Description	Default
[31:2]	Reserved	Reserved bits.	0
[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 5.19. INT_ENABLE Register

Bit	Label	Description	Default
[31:2]	Reserved	Reserved bits.	0
[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 5.20. INT_SET Register

Bit	Label	Description	Default
[31:2]	Reserved	Reserved bits.	0
[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. For full statistics, it is recommended to read from the least significant word then followed by the most significant word.

Table 5.21. Summary of Statistics Counters

Offset	Register Name	Access	Description
0x044	TX_STAT_PKT_LENGTH_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_LENGTH_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 CRC error. tx_statvec_o[22] is used to implement this counter.
0x060	TX_STAT_CRC_ERR_1	RO	
0x064	TX_STAT_LENGTH_ERR_0	RO	Transmit Length Error Statistics Counter. Counts the total number of packets transmitted with

Offset	Register Name	Access	Description
0x068	TX_STAT_LNGTH_ERR_1	RO	length of the packet and length in the Length/Type field mismatch. tx_statvec_o[23] is used to implement this counter.
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	

Offset	Register Name	Access	Description
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 multicast address. This does not include broadcast packets. rx_statvec_o[16] is used to implement this counter.
0x140	RX_STAT_MULTCST_1	RO	
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	

Offset	Register Name	Access	Description
0x194	RX_STAT_PKT_256_511_0	RO	Receive Packet 256-511 Statistics Counter. Counts the number of packets received that were between 256-511 octets in length (including bad packets).
0x198	RX_STAT_PKT_256_511_1	RO	
0x19C	RX_STAT_PKT_512_1023_0	RO	Receive Packet 512-1023 Statistics Counter. Counts the number of packets received that were between 512-1,023 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 1,024-1,518 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 1,518 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	
0X204	TX_STAT_PTP1588_PKT_0	RO	Transmit PTP1588 Statistics Counter. Counts the number of PTP1588 packets transmitted. The

Offset	Register Name	Access	Description
0x208	TX_STAT_PTP1588_PKT_1	RO	tx_statvec_o[26] is used to implement this counter.
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	
0x21C	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 statistics counter, you cannot reset the MAC signals, such as xg_rxval_o and xg_rx_blk_lock_o signals when the RX PHY is down.
0x224	TX_STAT_PKT_LNGTH_ACCU_0	RO	Transmit Accumulation Byte Statistic Counter. Count and accumulate the total packet length that will receive from AXIS interface and transmits to XGMII interface. The counter will roll over to 0 if full.
0x228	TX_STAT_PKT_LNGTH_ACCU_1	RO	
0x22C	RX_STAT_PKT_LNGTH_ACCU_0	RO	Receive Accumulation Byte Statistic Counter. Count and accumulate the total packet length that will receive from XGMII interface and transmit to AXIS interface. The counter will not increase if the packet received is discarded by MAC. The counter will roll over to 0 if full.
0x230	RX_STAT_PKT_LNGTH_ACCU_1	RO	

5.3. PHY Registers

All registers are accessed through the APB interface when the IP is configured as *PHY Only*. Access Types of each register are defined in the following table.

Table 5.22. Access Types for PHY

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 shows the register address map for the 5GbE PHY IP core.

Table 5.23. Register Address Map for PHY

AXI4-Lite Offset	APB Offset		Register Name	Bit Width	Description
PCS Registers					
0xA000 – 0xA0FC	0x5000 – 0x507F		PCS Module Registers	16	Refer to the Lattice Avant SERDES/PCS User Guide (FPGA-TN-02313) .

For more information on the address mapping, refer to the [PHY Management](#) section.

6. Example Design

The Versa example design is generated along with the 5G Ethernet IP core in the <Component Name>/eval folder.

The Versa example design integrates the 5G Ethernet MAC and Avant PHY IP core with the AXI4-Lite and data checker into an Ethernet sub-system. The 5G Ethernet IP Versa example design allows you to compile, simulate, and test the 5G Ethernet IP on the Avant G/X Versa board. For more information, refer to the [Versa Example Design Components \(Avant Devices\)](#) section.

For more information on testing the IP with the evaluation board, refer to the [Hardware Testing](#) section.

6.1. Example Design Configuration

The following table shows the IP configuration for the 5G Ethernet MAC IP example design.

Table 6.1. IP Configuration for the 5G Ethernet MAC IP Example Design

GUI Parameter	IP Configuration
Select IP Option	MAC + PHY
Host Interface	AXI4-Lite
Multicast Address Filtering	Don't Care
TX Pause Frame Generation via Ports	Don't Care
Statistics Counters Registers	Don't Care
PMA Setup-Transmitter Subgroup	Don't Care
PMA Setup-Receiver Subgroup	Don't Care
Fase Simulation Mode	Uncheck
PCS Lane ID	24
Loopback Mode	No Loopback

6.2. Overview of Example Design and Features

Key features of the 5G Ethernet MAC IP example design include:

- Ethernet packet generator to generate and compare packets
- Integrated 5G Ethernet MAC with 5G Ethernet PCS and GPLL to demonstrate the expected clocking scheme
- Loopback transmitted packet at the serial interface
- Supports continuous and non-continuous mode packet transmission

6.3. Example Design Components

6.3.1. Versa Example Design Components (Avant Devices)

The 5G Ethernet IP versa example design includes the following blocks:

- 5 Gb Ethernet MAC + PHY IP core
- AXI4-Lite module
- AXI-Stream driver
- Data checker

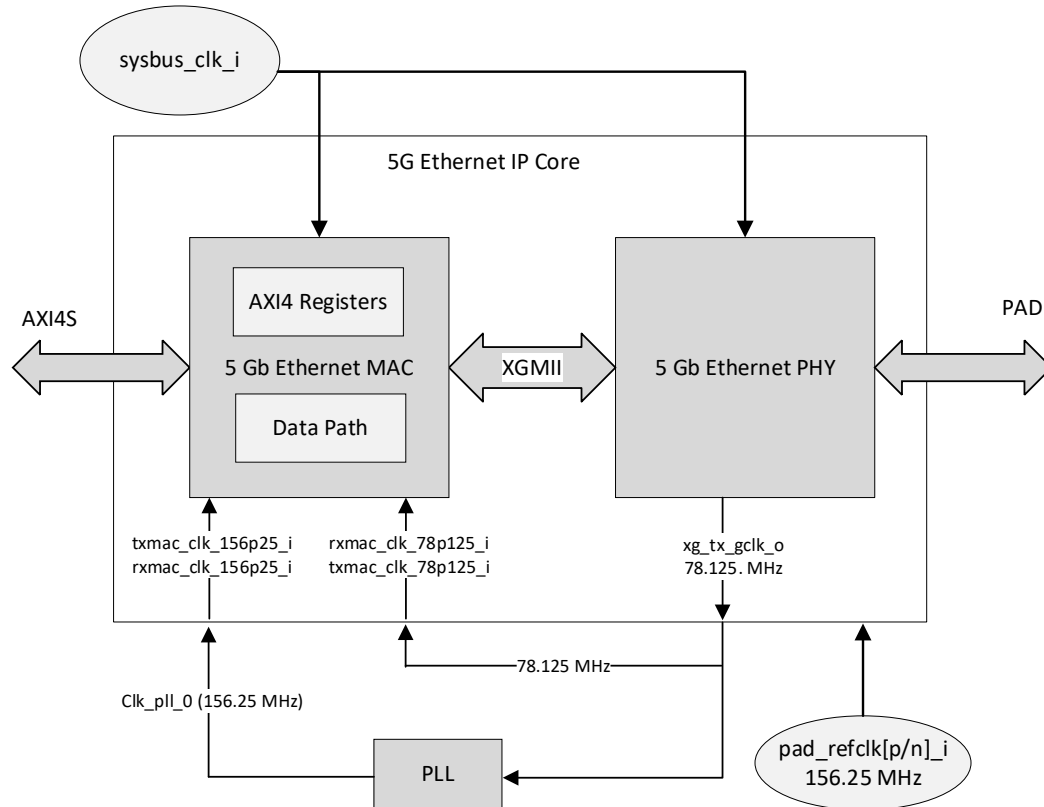


Figure 6.1 5 Gb Ethernet MAC IP Versa Example Design Block Diagram

6.3.1.1. 5 Gb Ethernet MAC + PHY IP Core

The 5 Gb Ethernet MAC instantiates the 5G Ethernet IP core based on the configuration value described in [Table 6.2](#). In the transmit data path, the 5 Gb Ethernet MAC receives packets from AXI-Stream Packet Generator and sends the packets to the 5 Gb Ethernet PHY. In the receive data path, the 5 Gb Ethernet MAC forwards packets from the 5 Gb Ethernet PHY core to the data checker. The 5 Gb Ethernet MAC is connected to the 5 Gb Ethernet PHY within the 5G Ethernet IP core through the XGMII interface. The clock source to the Ethernet MAC is from xg_tx_gclk_o, which is the clock output from the Ethernet PHY. In the transmit data path, the 5 Gb Ethernet PHY sends packets from 5 Gb Ethernet MAC core to serial interface signals. In the receive data path, it forwards the received packet at serial interface signals to the 5 Gb Ethernet MAC core.

6.3.1.2. AXI4-Lite Module

The AXI4-Lite module (Avant devices) is the example design component that configures the register of the 5 Gb Ethernet MAC. The example design component demonstrates the usage of the AXI4-Lite interface or APB interface of the 5 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

5 Gb Ethernet MAC Register Address Offset	5 Gb Ethernet MAC Register Name	Configuration Value
0x000	MODE	0x3
0x004	TX_CTL	0x0
0x008	RX_CTL	0x1
0x010	IPG_VAL	0x10

6.3.1.3. AXI-Stream Packet Generator

The AXI-Stream Packet Generator generates packets for the AXI-Stream Transmit Interface of the 5G Ethernet IP. The contents of the packets are randomized.

6.3.1.4. 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 5 Gb Ethernet MAC core. An error signal is flagged if the packets are mismatched.

6.4. Generating Example Design

6.4.1. Create New Radiant Project

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

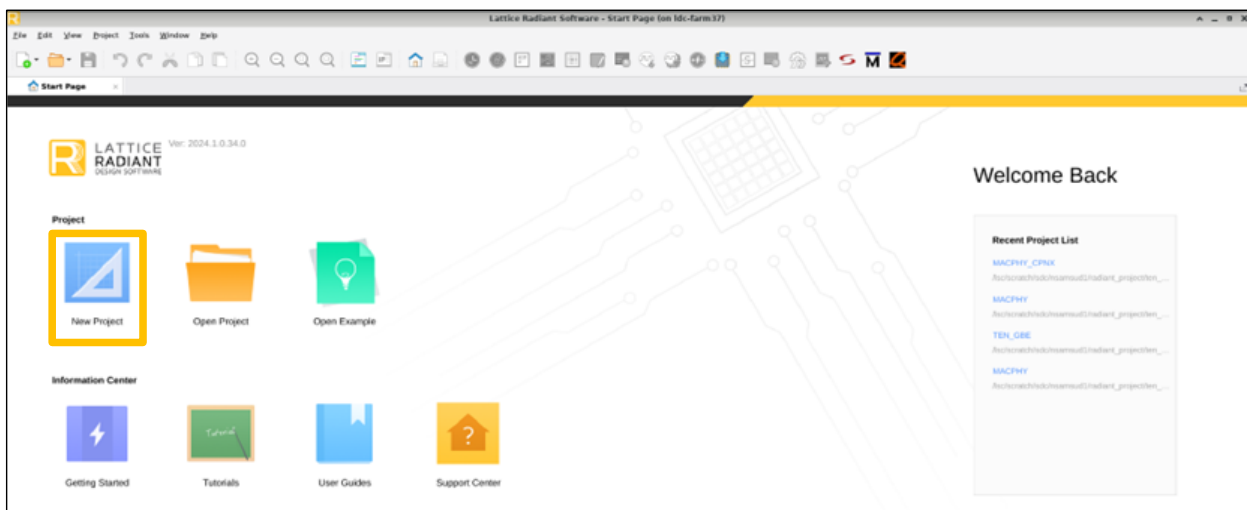


Figure 6.2. Start Page of the Radiant Software

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

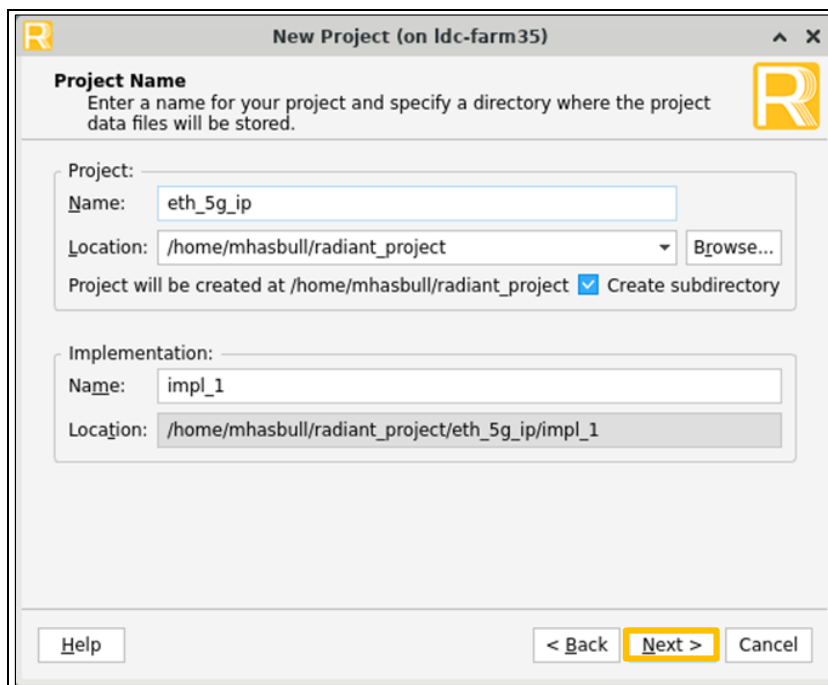


Figure 6.3. Create a New Project

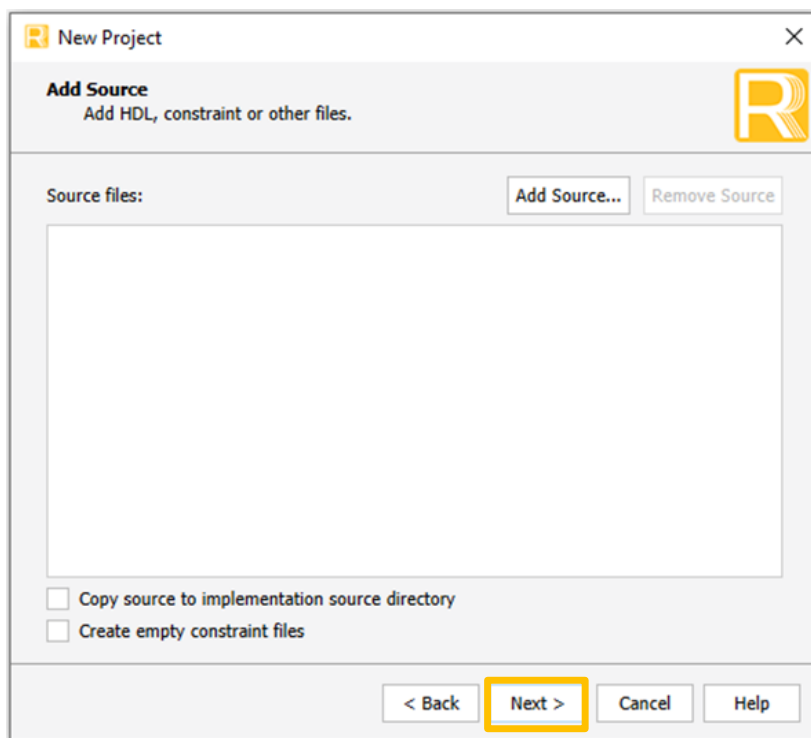


Figure 6.4. Add Source to Project

3. In the **Select Device** window, follow these steps:
 - a. Select **LAV-AT (Avant)** family → **LAV-AT-X70** or **LAV-AT-G70** device, then click **Next**.

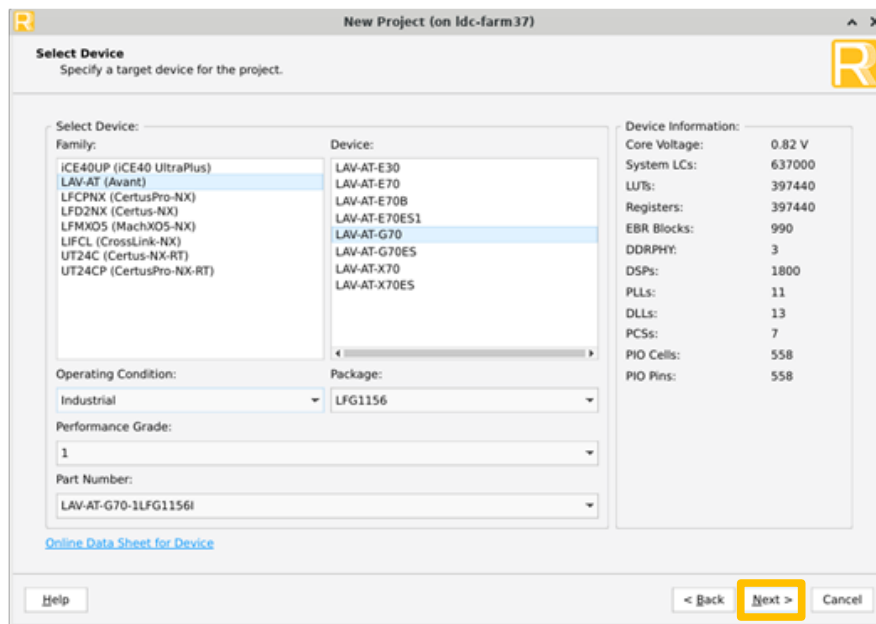


Figure 6.5. Select an Avant Device for the Project

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

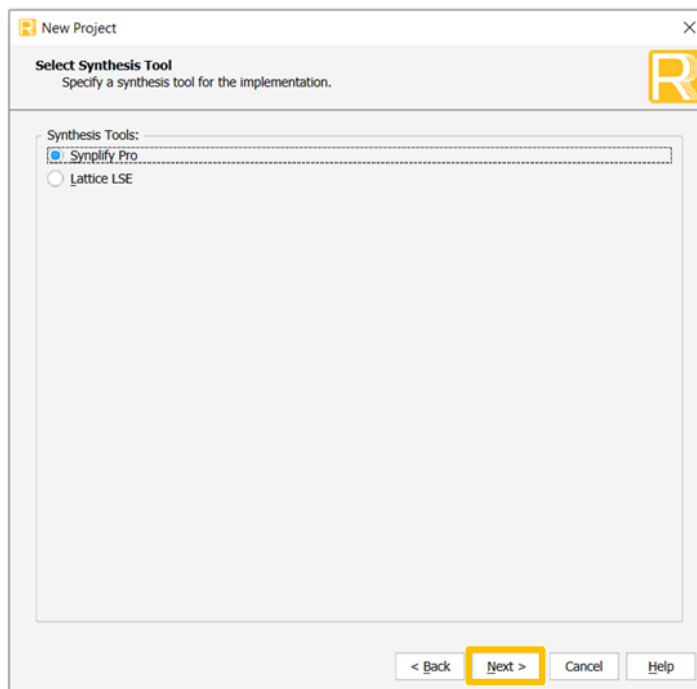


Figure 6.6. Select a Synthesis Tool

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

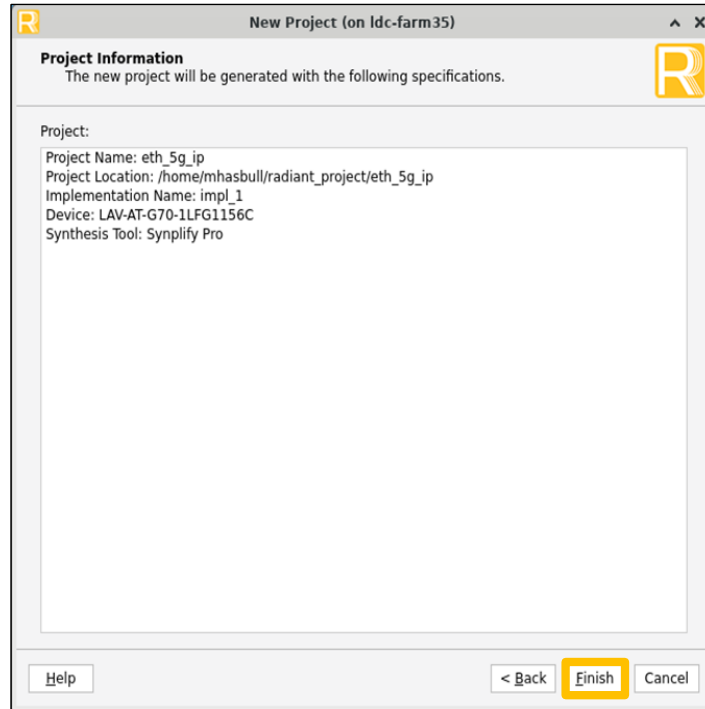


Figure 6.7. Project Information

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

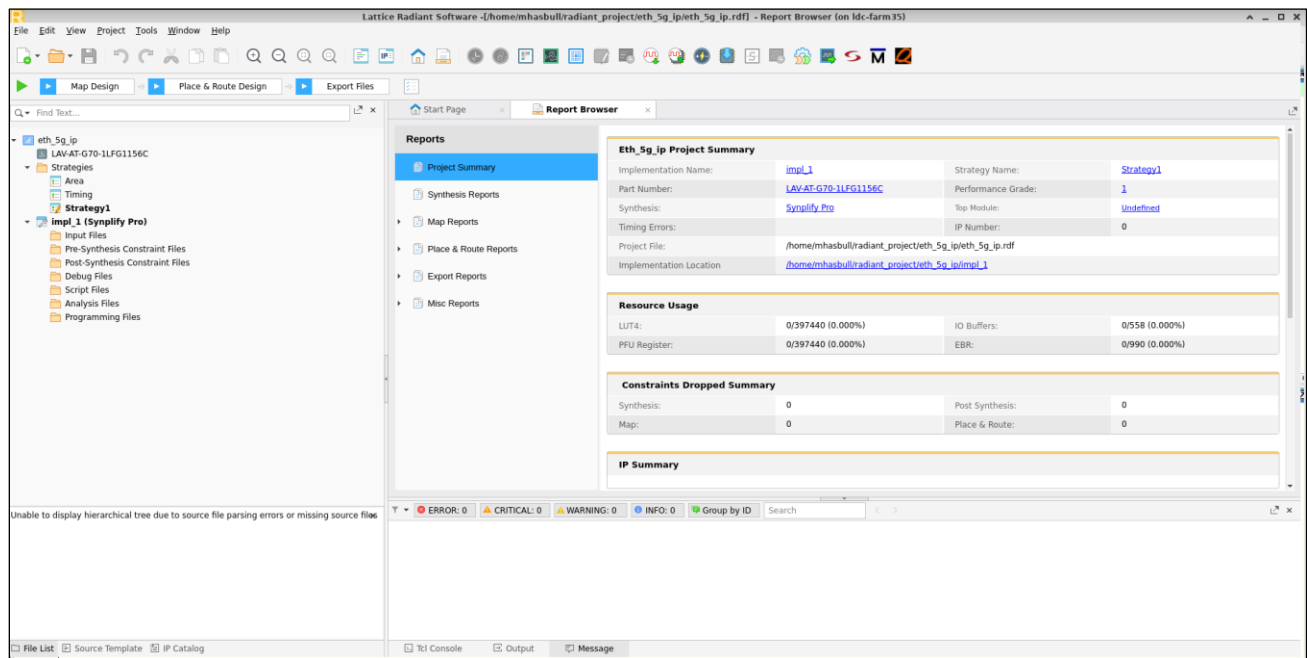



Figure 6.8. A Radiant Software Project is Created

6.4.2. IP Installation and Generation

1. Go to **IP Catalog** → **IP on Server** tab. Under the **Connectivity** category, right-click on the **5G 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 5G Ethernet IP installed, proceed to step 2.
2. Go to the **IP on Local** tab. Right-click on the 5 Gb Ethernet IP and select **Generate...** to launch the IP GUI.
3. Enter **Component Name** and click **Next**.

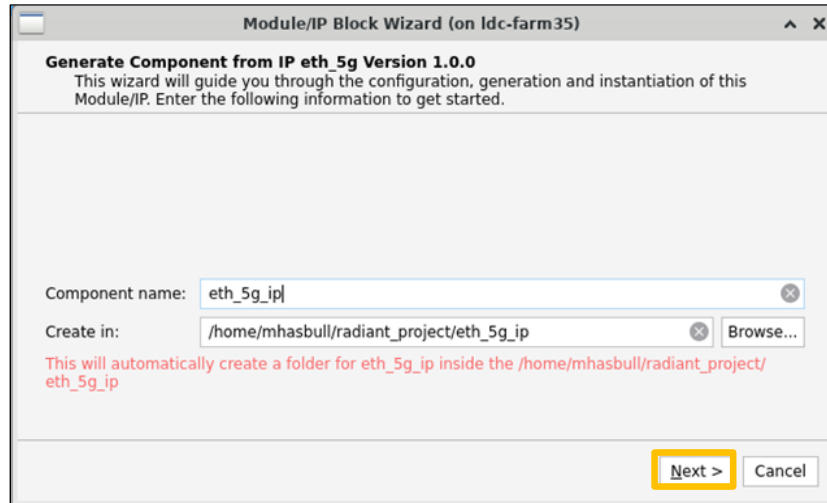


Figure 6.9. Generate Component

4. The following window allows you to change the IP configuration. You may use the default settings* and click **Generate**.

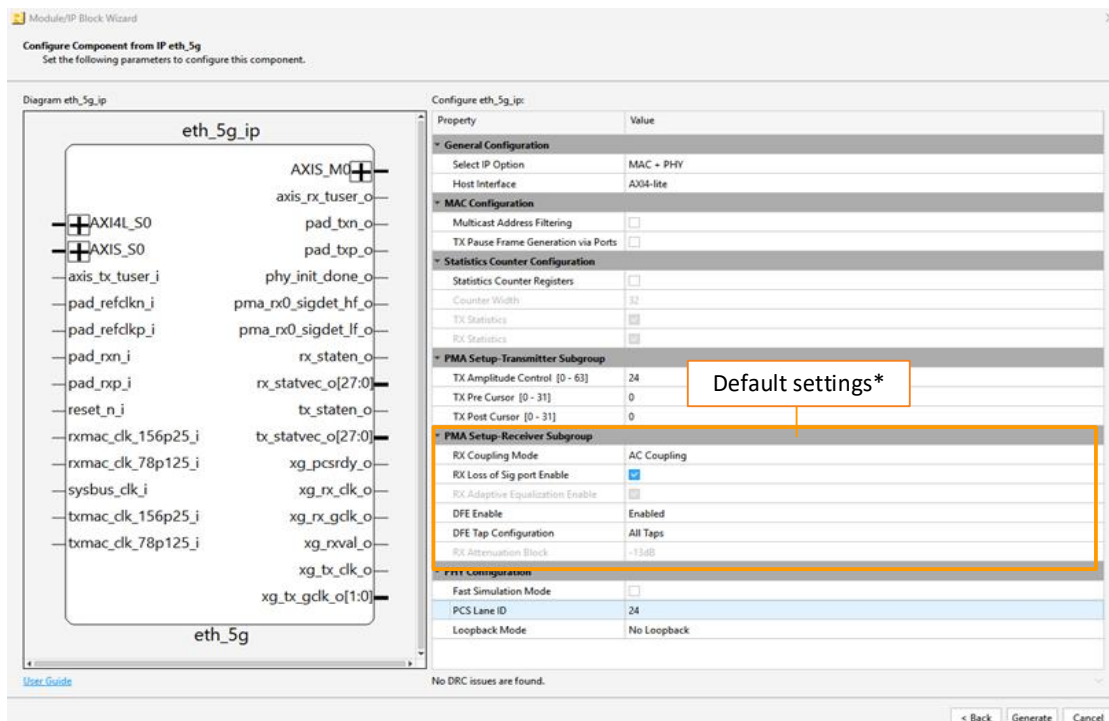


Figure 6.10. Configure Component

Note:

* Applicable for LAV-AT-G70 and LAV-AT-X70.

For PHY configuration, verify that the selected PCS Lane ID corresponds to the actual serial connection on board. For onboard connectivity details, refer to the Avant-G/X Versa Board User Guide.

6.4.3. Importing Example Design Files to a Project

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

Table 6.3. Description of Generated Files in the Eval Folder

Files	Description
<i>README.txt</i>	Contain instructions to use Eval and Versa example design files of this directory.
<i>constraint.pdc</i>	Post-synthesis constraints for Eval example design.
<i>dut_inst.v</i>	Contain instantiation of IP based on selected configuration of the 5G Ethernet IP. Included by <i>eval_top.sv</i> and <i>versa_top.sv</i> .
<i>dut_params.sv</i>	Contain IP parameters based on selected configuration of the 5G Ethernet IP. Included by <i>tb_top_eval.sv</i> and <i>tb_top_versa.sv</i> .
Files supported by Avant devices only: LAV-AT-G70 and LAV-AT-X70 (eval/versa_top/avant_x70_5g)	
<i>versa.pdc</i>	Post-synthesis constraints for the Versa example design.
<i>versa_top.sv</i>	Top-level design file for the Versa example design.
<i>tb_top_versa.sv</i>	Testbench for the Versa example design.
<i>debounce.v</i>	Debounce module for hardware mechanical input.
<i>osc_ip.v</i>	OSC module from foundation IP.
<i>psc_apb_w_r.v</i> <i>pcs_apb_w.v</i>	APB module for register configuration.
<i>traffic_genchk.v</i>	Traffic generator module to generate random traffic for transmission and perform checking in loopback.
<i>tb_top_versa.do</i>	Script to group simulation signals.

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

1. For Avant devices, select part number **LAV-AT-X70-3LFG1156** and set the **Package** option to **LFG1156**.

Note that the current Versa example design is tested on LAV-AT-X70.

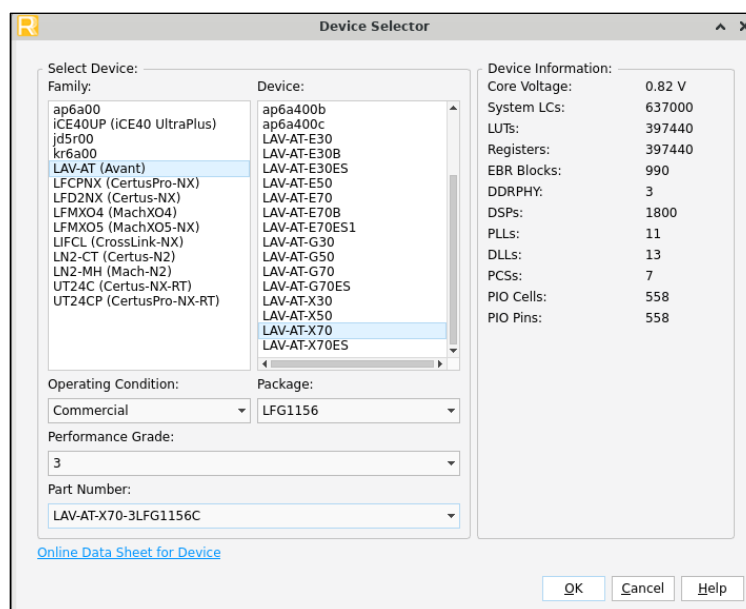


Figure 6.11. Device Setup for Avant Devices

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

Alternatively, you may set up the example design using TCL scripts by entering the command below:

TCL command: `source {<IP_INSTANCE_NAME>\eval\versa_top\avant_x70_5g\ed_setup.tcl}`

The top-level files can be found in *IP_NAME/eval/versa_top/DEVICE_INTERFACE*.

Table 6.4. Top-Level Files

File Description	Avant-AT-X70 Devices (5G Ethernet)
Top-level file to include in software project.	<i>versa_top.sv</i>
Simulation for top-level file to include.	<i>tb_top_versa.sv</i>
Post-synthesis constraint file.	<i>versa.pdc</i>

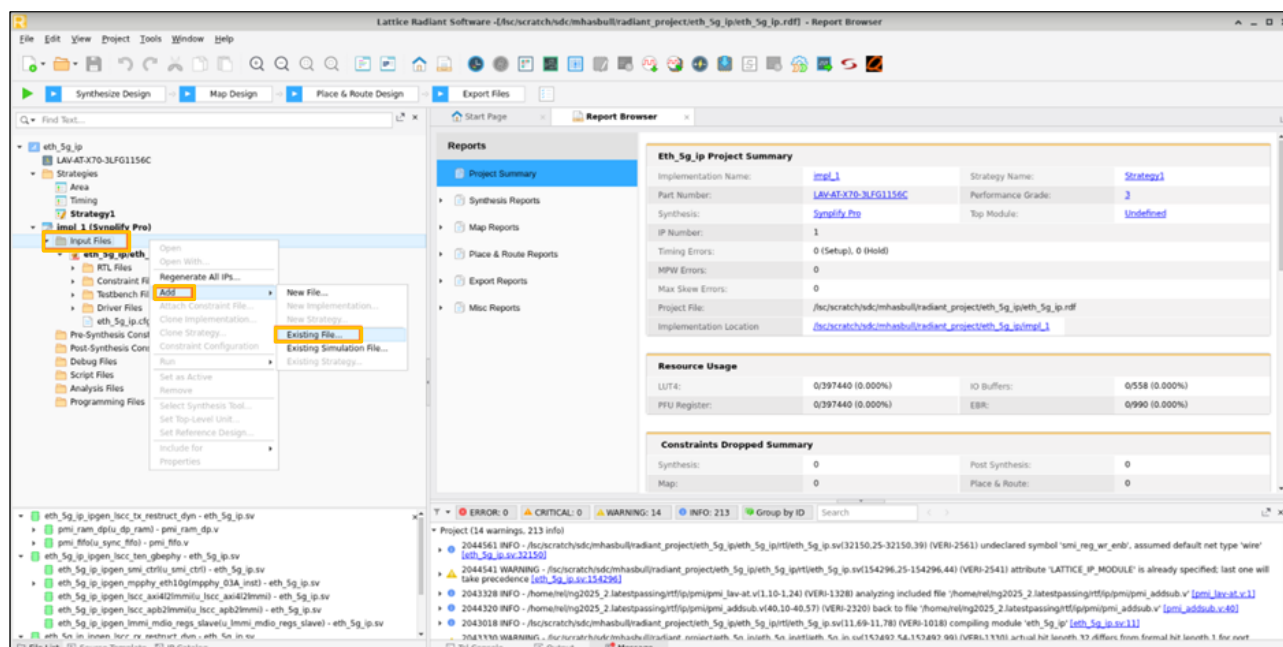


Figure 6.12. Add Related Files into the Radiant Software Project

- Versa design file: *versa_top.sv* or *top.v* file. 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 in hardware. For more information on testing the IP with the evaluation board, refer to the [Hardware Testing](#) section.

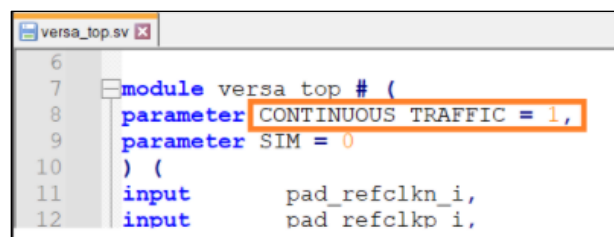


Figure 6.13. CONTINUOUS_TRAFFIC Parameter for Continuous Mode in versa_top.sv File

- Non-continuous mode
Allows you to change the CONTINUOUS_TRAFFIC parameter to 0.
4. Right-click on **Input Files**, select **Add → Existing Simulation File...**, then add the *tb_top_versa.sv* file to the Radiant software project.

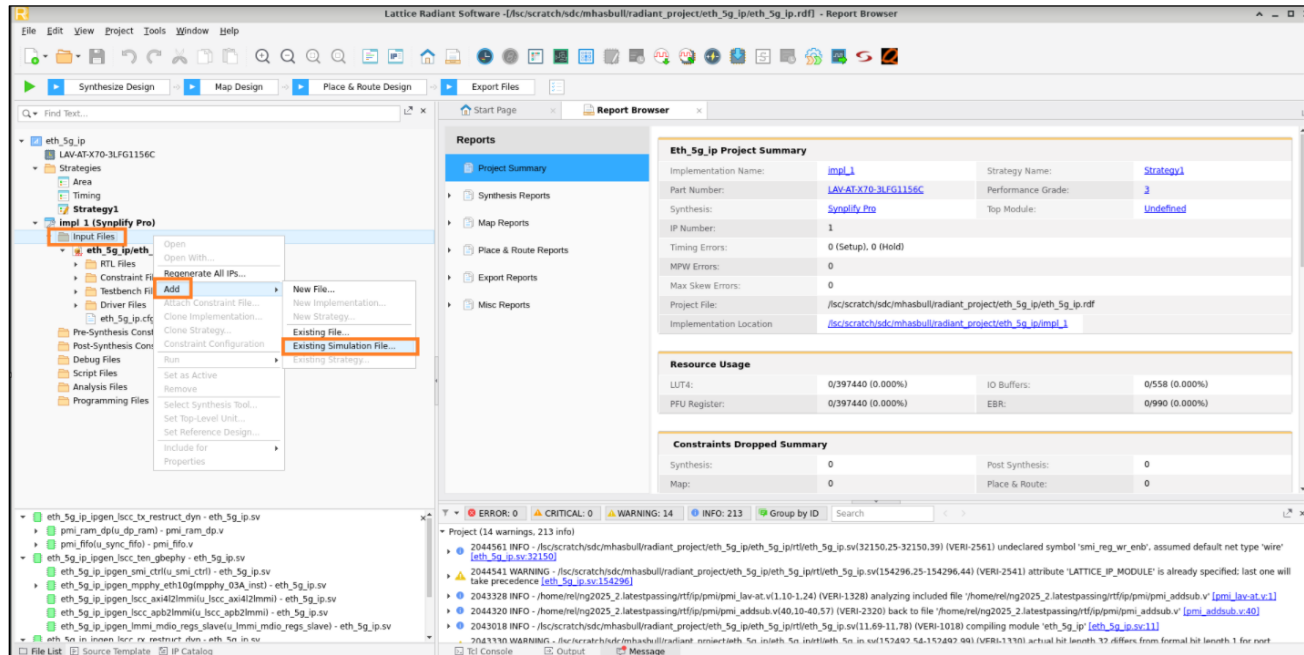


Figure 6.14. Add the *tb_top_versa.sv* File into the Radiant Software Project

5. Versa testbench: *tb_versa_top.sv* or *tb_top.v*. The SIM parameter remains unchanged for both modes.
- CONTINUOUS_TRAFFIC parameter is default to 1 for continuous mode packet transmission.

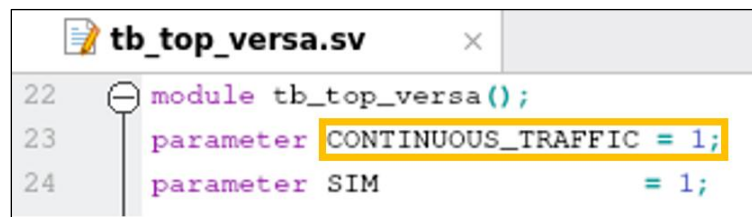


Figure 6.15. CONTINUOUS_TRAFFIC Parameter for Continuous Mode in the *tb_versa_top.sv* File

- Allows you to change the CONTINUOUS_TRAFFIC parameter to 0 for non-continuous mode.

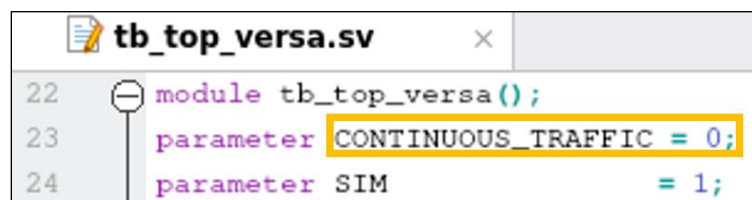


Figure 6.16. CONTINUOUS_TRAFFIC Parameter for Non-Continuous Mode in the *tb_versa_top.sv* File

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

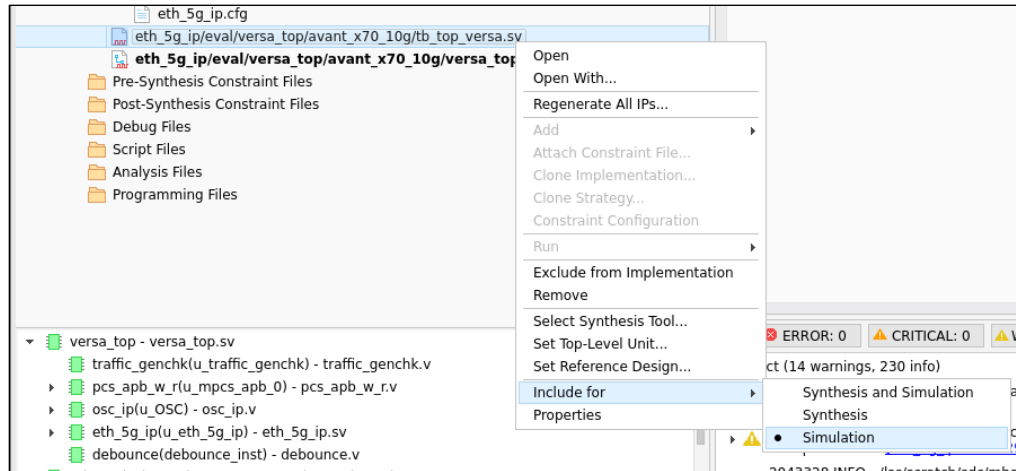


Figure 6.17. Set the tb_top_versa_top File to be Included for Simulation Only

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

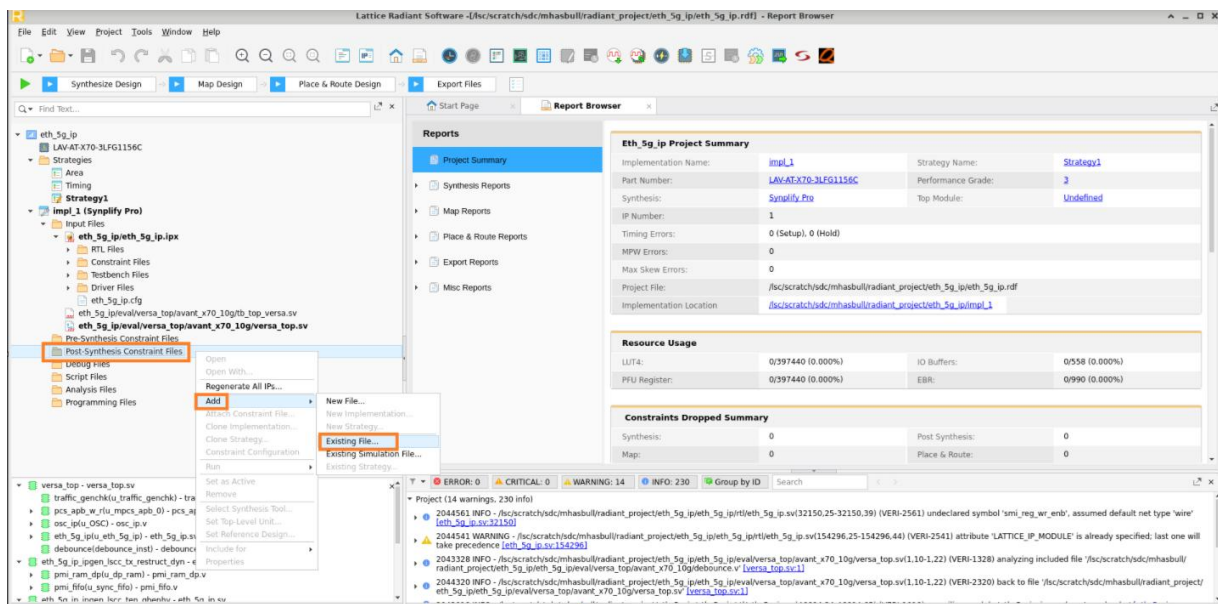


Figure 6.18. Add the .pdc File into the Radiant Software Project

- 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 or physical constraints.

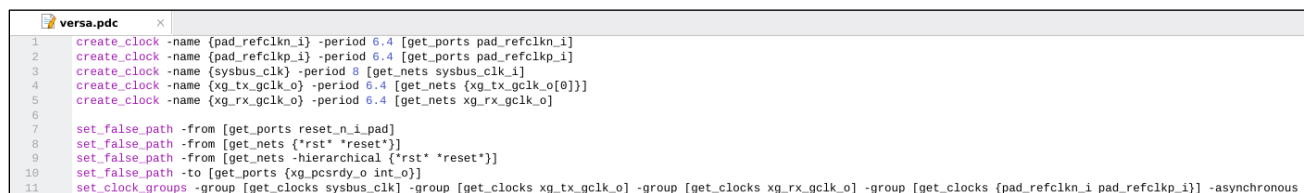


Figure 6.19. Timing Constraint File (versa.pdc Example from an Avant Device)

- The hardware example design preparation is completed.

6.5. Example Design Simulation

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

6.5.1. 5 Gb Ethernet MAC Example Design Testbench Flow

The following diagram shows the testbench simulation flow. After the simulation starts, the testbench waits for 5 Gb Ethernet PCS link up. The APB driver will configure the 5G Ethernet MAC core to enable transmit and receive data paths. After that, packets are generated by the AXI-Stream driver and loopback at serial interface. Based on the result of the content comparison between the transmitted and loopback packets, the simulation PASSED or FAILED status is displayed.

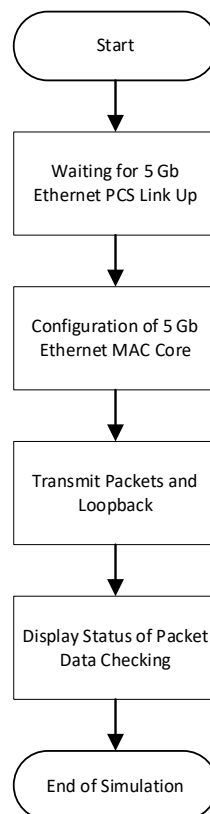



Figure 6.20. 5G Ethernet MAC Example Design Testbench Flowchart

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**.

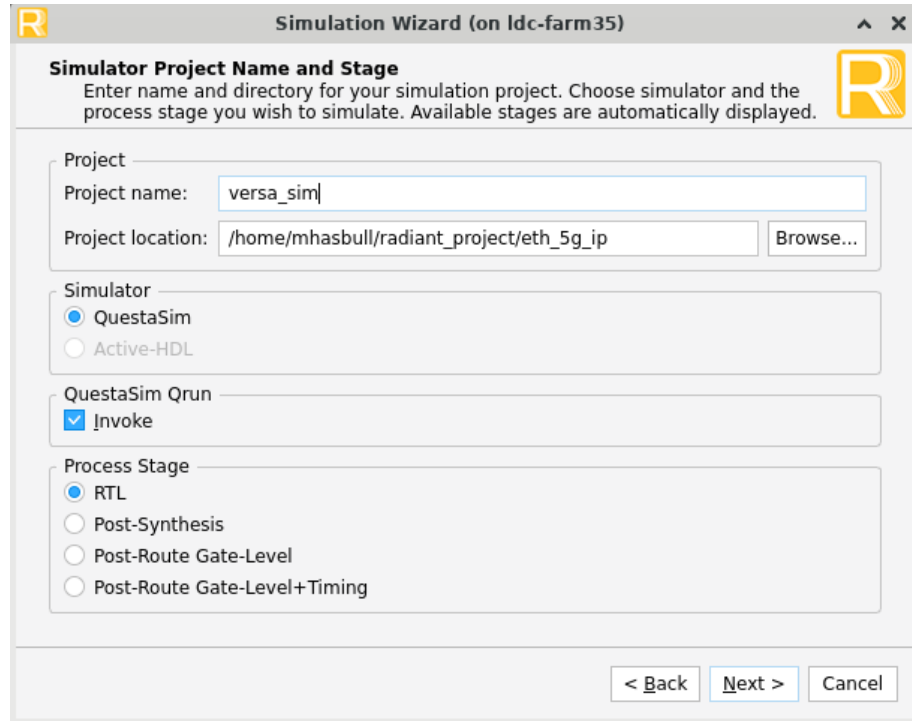


Figure 6.21. Create a Simulation Project

3. Click **Next**.



Figure 6.22. Include Source Files

4. Click **Next**.

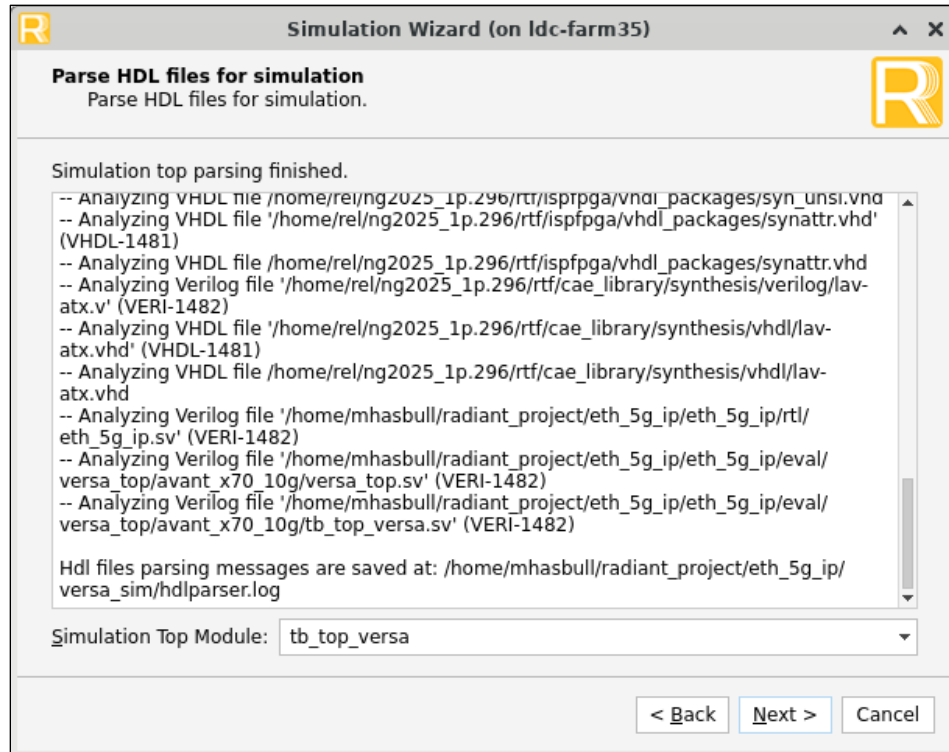


Figure 6.23. Set **tb_top*** as Top Module

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

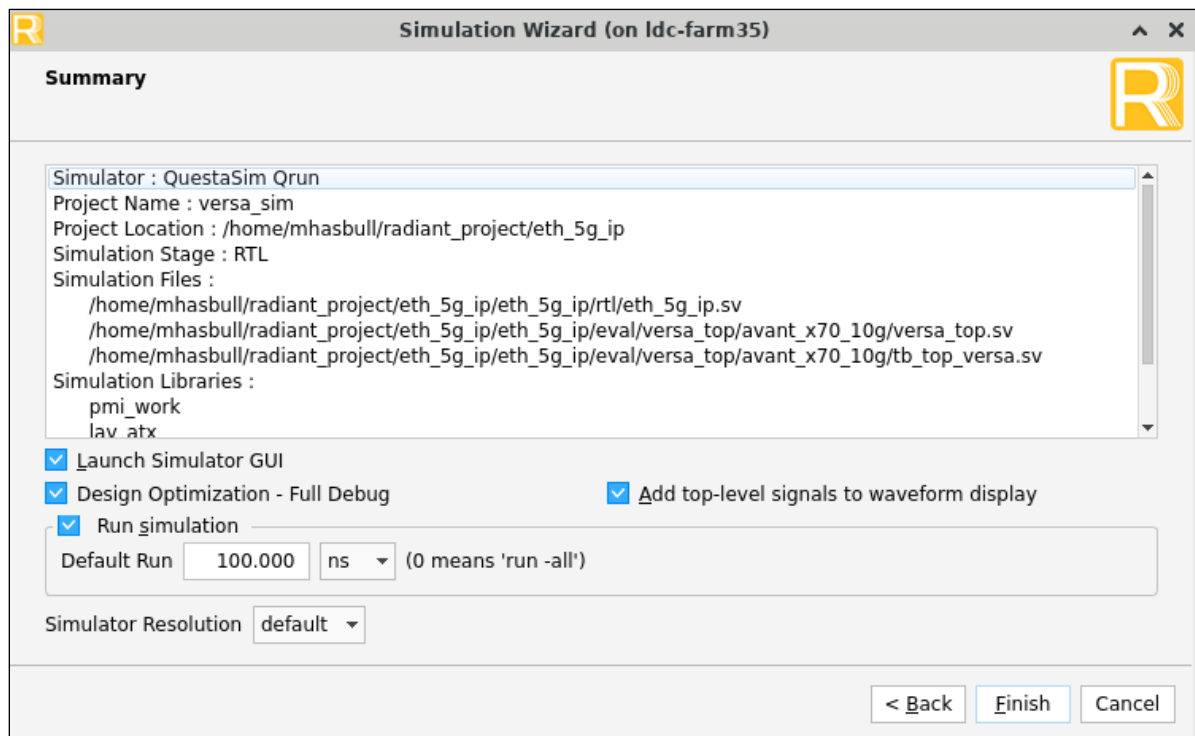


Figure 6.24. Change the Time Setting for Default Run

6.5.2.1. Continuous Mode

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

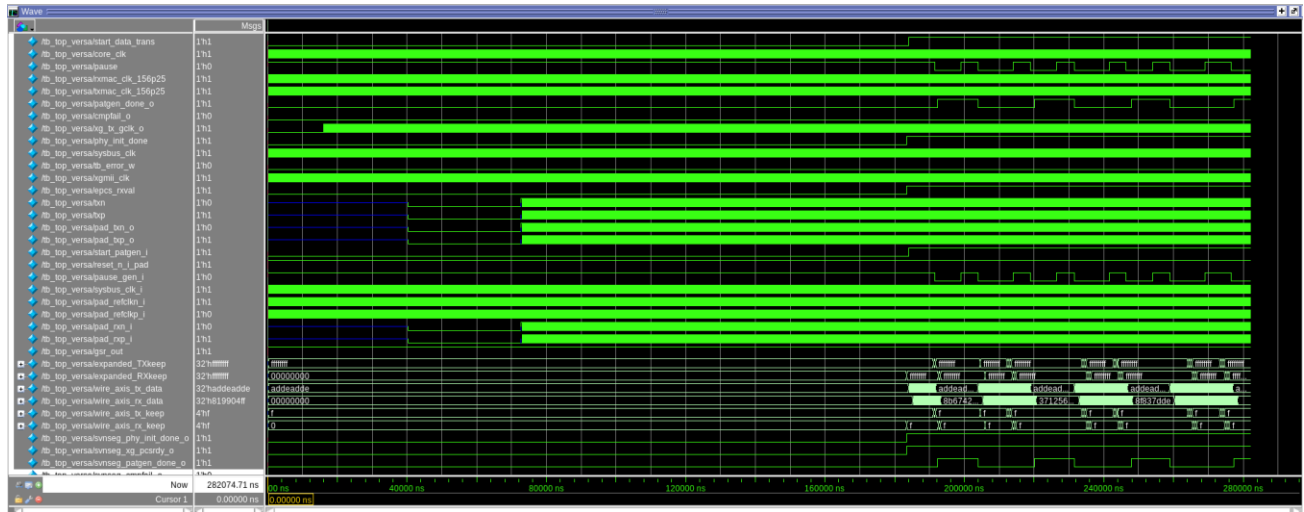


Figure 6.25. Example Design Simulation Waveform for Continuous Mode

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

```
# Data matched: exp = 7b1a3f3db55e7282, obs = 7b1a3f3db55e7282
# Data matched: exp = 945f7d688f122c48, obs = 945f7d688f122c48
# Data matched: exp = 28befad11e245890, obs = 28befad11e245890
# Data matched: exp = c7fbb8842468065a, obs = c7fbb8842468065a
# Data matched: exp = 8ff7710848d00cb5, obs = 8ff7710848d00cb5
# Data matched: exp = 60b2335d729c527f, obs = 60b2335d729c527f
# +-----+
# Pause TX traffic; 3rd attempt
# +-----+
# Data matched: exp = c16466bae538a4ff, obs = c16466bae538a4ff
# Data matched: exp = 2e2124efdf74fa35, obs = 2e2124efdf74fa35
# Data matched: exp = 5c4249dfbee9f46a, obs = 5c4249dfbee9f46a
# Data matched: exp = b3070b8a84a5aaa0, obs = b3070b8a84a5aaa0
# Data matched: exp = 660e1715094b5540, obs = 660e1715094b5540
# Data matched: exp = 894b554033070b8a, obs = 894b554033070b8a
# Data matched: exp = 1296aa80660e1715, obs = 1296aa80660e1715
# Data matched: exp = fdd3e8d55c4249df, obs = fdd3e8d55c4249df
# Data matched: exp = fba7d1aab88493bf, obs = fba7d1aab88493bf
# Data matched: exp = 14e293ff82c8cd75, obs = 14e293ff82c8cd75
# Data matched: exp = 29c527ff05919aeb, obs = 29c527ff05919aeb
# Data matched: exp = c68065aa3fddc421, obs = c68065aa3fddc421
# Data matched: exp = 579ca0af6347e7b6, obs = 579ca0af6347e7b6
# Data matched: exp = b8d9e2fa590bb97c, obs = b8d9e2fa590bb97c
# Data matched: exp = 71b3c5f4b21772f8, obs = 71b3c5f4b21772f8
# Data matched: exp = 9ef687a1885b2c32, obs = 9ef687a1885b2c32
# Data matched: exp = 3ded0f4310b65865, obs = 3ded0f4310b65865
# Data matched: exp = d2a84d162afa06af, obs = d2a84d162afa06af
# Data matched: exp = a5509a2c55f40d5f, obs = a5509a2c55f40d5f
# Data matched: exp = 4a15d8796fb85395, obs = 4a15d8796fb85395
# Data matched: exp = 942bb0f2df70a72b, obs = 942bb0f2df70a72b
# Data matched: exp = 7b6ef2a7e53cf9e1, obs = 7b6ef2a7e53cf9e1
# +-----+
# Resume TX traffic; 3rd attempt
# +-----+
# Data matched: exp = facedeaddeaddead, obs = facedeaddeaddead
# Data matched: exp = 1998a71af035ad08, obs = 1998a71af035ad08
# Data matched: exp = 33314e35e06b5a10, obs = 33314e35e06b5a10
# Data matched: exp = dc740c60da2704da, obs = dc740c60da2704da
# Data matched: exp = b8e818c1b44e09b4, obs = b8e818c1b44e09b4
# Data matched: exp = 57ad5a948e02577e, obs = 57ad5a948e02577e
# Data matched: exp = af5ab5291c04aefd, obs = af5ab5291c04aefd
# Data matched: exp = 401ff77c2648f037, obs = 401ff77c2648f037
# Data matched: exp = 803feef84c91e06e, obs = 803feef84c91e06e
# Data matched: exp = 6f7aacad76ddea4, obs = 6f7aacad76ddea4
# Data matched: exp = def5595aedbb7d48, obs = def5595aedbb7d48

# Data matched: exp = 361fafee4704b1b0, obs = 361fafee4704b1b0
# Data matched: exp = d95aebdb7d48ef7a, obs = d95aebdb7d48ef7a
# Data matched: exp = b2b5db76fa91def5, obs = b2b5db76fa91def5
# [ 15508400000]:[Normal] ]---[tb_top_versa.genbkl1]---+-----+
# [ 15508400000]:[Normal] ]---[tb_top_versa.genbkl1]---Transaction Check Done
# [ 15508400000]:[Normal] ]---[tb_top_versa.genbkl1]---+-----+
# [ 15508400000]:[Normal] ]---[tb_top_versa.genbkl1]---***** DATA TRANSACTION PASSED *****
# [ 15508400000]:[Normal] ]---[tb_top_versa.genbkl1]---+-----+
# [ 15508400000]:[Normal] ]---[tb_top_versa.genbkl1]---+-----+
# [ 15508400000]:[Normal] ]---[tb_top_versa.genbkl1]---SIMULATION PASSED
```

Figure 6.26. Example Design Simulation Output for Continuous Mode

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

```
# [ 0]:[Normal ]---[tb_top_eval]--- *****
# [ 0]:[Normal ]---[tb_top_eval]--- Start of Simulation
# [ 0]:[Normal ]---[tb_top_eval]--- +-----+
# [ 0]:[Normal ]---[tb_top_eval]--- Testbench Parameters
# [ 0]:[Normal ]---[tb_top_eval]--- +-----+
# [ 0]:[Normal ]---[tb_top_eval]--- DATA WIDTH : 64
# [ 0]:[Normal ]---[tb_top_eval]--- REFERENCE CLOCK (MHz) : 161.108426
# [ 0]:[Normal ]---[tb_top_eval]--- XGMII CLOCK (MHz) : 156.250000
# [ 0]:[Normal ]---[tb_top_eval]--- +-----+
#
# Ons bist_time_out = 1'b0
# Ons latch_bist_err = 1'b0
# Ons bist_done = 1'b0
# Ons bist_ok = 1'b0
#
# 6467000 tb_top_eval.ten_gbe_subsys.u_ten_gbe_pcs.lsc.ten_gbephy_inst.<protected>.<protected>.<pr
# oted>.<protected>.<protected>.<protected>.<protected>.<protected>.<protected>.<protected>.<protect
# ed>.<protected>.<protected>.<protected> ABICDR PHY TX DRIVER : Electrical Idle III
# [ 128000000]:[Normal ]---[tb_top_eval]--- +-----+
# [ 128000000]:[Normal ]---[tb_top_eval]--- Wait for PLL to lock and PCS RX ready
# [ 128000000]:[Normal ]---[tb_top_eval]--- +-----+
#
# 16860524000 tb_top_eval.ten_gbe_subsys.u_ten_gbe_pcs.lsc.ten_gbephy_inst.<protected>.<protected>.<pr
# oted>.<protected>.<protected>.<protected>.<protected>.<protected>.<protected>.<protected>.<protect
# ed>.<protected>.<protected>.<protected> ABICDR PHY TX DRIVER : Electrical Idle I
#
# 17025036000 tb_top_eval.ten_gbe_subsys.u_ten_gbe_pcs.lsc.ten_gbephy_inst.<protected>.<protected>.<pr
# oted>.<protected>.<protected>.<protected>.<protected>.<protected>.<protected>.<protected>.<protect
# ed>.<protected>.<protected>.<protected> ABICDR PHY TX DRIVER : Transmitting Data
# [ 21203000000]:[Normal ]---[tb_top_eval]--- +-----+
# [ 21203000000]:[Normal ]---[tb_top_eval]--- PLL lock asserted
# [ 21203000000]:[Normal ]---[tb_top_eval]--- PCS RX is now ready
# [ 21203000000]:[Normal ]---[tb_top_eval]--- Driving AXI4-Stream TX random transactions
# [ 21203000000]:[Normal ]---[tb_top_eval]--- +-----+
# [ 24252069100]:[Data]--- data matched!
# [ 24252069100]:[Data]--- exp = deadbeefaaaaffff, obs = deadbeefaaaaffff, byte_en = ff
# [ 24258471100]:[Data]--- data matched!
# [ 24258471100]:[Data]--- exp = 89375212b2c28465, obs = 89375212b2c28465, byte_en = ff
# [ 24264873100]:[Data]--- data matched!
# [ 24264873100]:[Data]--- exp = 06d7cd0d00f3e301, obs = 06d7cd0d00f3e301, byte_en = ff
# [ 24271275100]:[Data]--- data matched!
# [ 24271275100]:[Data]--- exp = 1e8dcd3d3b23f176, obs = 1e8dcd3d3b23f176, byte_en = ff
# [ 24277677100]:[Data]--- data matched!
# [ 24277677100]:[Data]--- exp = 462df78c76d457ed, obs = 462df78c76d457ed, byte_en = ff
# [ 24284079100]:[Data]--- data matched!
# [ 24284079100]:[Data]--- exp = e33724c67cfde9f9, obs = e33724c67cfde9f9, byte_en = ff
# [ 24290481100]:[Data]--- data matched!
# [ 24290481100]:[Data]--- exp = d513d2aae2f784c5, obs = d513d2aae2f784c5, byte_en = ff
# [ 24296883100]:[Data]--- data matched!
# [ 24296883100]:[Data]--- exp = bbd2727772aff7e5, obs = bbd2727772aff7e5, byte_en = ff
# [ 24303285100]:[Data]--- data matched!
# [ 24303285100]:[Data]--- exp = 47ecdb8f8932d612, obs = 47ecdb8f8932d612, byte_en = ff
# [ 24309687100]:[Data]--- data matched!
# [ 24309687100]:[Data]--- exp = e77696ce793069f2, obs = e77696ce793069f2, byte_en = ff
# [ 24316089100]:[Data]--- data matched!
# [ 24316089100]:[Data]--- exp = e2ca4ec5f4007ae8, obs = e2ca4ec5f4007ae8, byte_en = ff
# [ 24322491100]:[Data]--- data matched!
# [ 24322491100]:[Data]--- exp = de8e28bd2e58495c, obs = de8e28bd2e58495c, byte_en = ff
# [ 24328893100]:[Data]--- data matched!
# [ 24328893100]:[Data]--- exp = b2a7266596ab582d, obs = b2a7266596ab582d, byte_en = ff
# [ 24335295100]:[Data]--- data matched!
# [ 24335295100]:[Data]--- exp = b2a7266596ab582d, obs = b2a7266596ab582d, byte_en = ff
# [ 24341697100]:[Data]--- data matched!
# [ 24341697100]:[Data]--- exp = 00000000b1ef6263, obs = 261b0b3fb1ef6263, byte_en = 0f
# [ 28032000000]:[Normal ]---[tb_top_eval]--- +-----+
# [ 28032000000]:[Normal ]---[tb_top_eval]--- Transaction Done
# [ 28032000000]:[Normal ]---[tb_top_eval]--- +-----+
# [ 28032000000]:[Normal ]---[tb_top_eval]--- ***** PASSED *****
# [ 28032000000]:[Normal ]---[tb_top_eval]--- +-----+
# [ 28032000000]:[Normal ]---[tb_top_eval]--- End of Simulation
# [ 28032000000]:[Normal ]---[tb_top_eval]--- *****
#
# ** Note: $finish : C:/Radiant_project/ethernet/ethernet_mac/eval/tb_top_eval.v(117)
# Time: 28672 ns Iteration: 1 Instance: /tb_top_eval
#
# Data matched: exp = b2b5db76fa91def5, obs = b2b5db76fa91def5
# Data matched: exp = 5df09923c0dd803f, obs = 5df09923c0dd803f
# Data matched: exp = bbe1324781bb007f, obs = bbe1324781bb007f
# Data matched: exp = 54a47012bbf75eb5, obs = 54a47012bbf75eb5
# Data matched: exp = a948e02577eebd6a, obs = a948e02577eebd6a
# Data matched: exp = 460da2704da2e3a0, obs = 460da2704da2e3a0
# Data matched: exp = 8c1b44e09b45c740, obs = 8c1b44e09b45c740
# Data matched: exp = 635e06b5a109998a, obs = 635e06b5a109998a
# Data matched: exp = c6bc0d6b42133314, obs = c6bc0d6b42133314
# Data matched: exp = 29f94f3e785f6dde, obs = 29f94f3e785f6dde
# Data matched: exp = 53f29e7cf0bedbbc, obs = 53f29e7cf0bedbbc
# Data matched: exp = bcb7dc29caf28576, obs = bcb7dc29caf28576
# Data matched: exp = 796fb85395e50aec, obs = 796fb85395e50aec
# Data matched: exp = 962afa06afa95426, obs = 962afa06afa95426
#
# [ 9512700000]:[Normal ]---[tb_top_versa]--- +-----+
# [ 9512700000]:[Normal ]---[tb_top_versa]--- Transaction Done
# [ 9512700000]:[Normal ]---[tb_top_versa]--- +-----+
# [ 9512700000]:[Normal ]---[tb_top_versa]--- ***** DATA TRANSACTION PASSED *****
# [ 9512700000]:[Normal ]---[tb_top_versa]--- +-----+
# [ 9512700000]:[Normal ]---[tb_top_versa]--- +-----+
# [ 9512700000]:[Normal ]---[tb_top_versa]--- SIMULATION PASSED
#
# ** Note: $finish : /home/nsamsudl/my_designs/eth_10g_demo/eth_10g_ip/eth_10g_ip/eval/tb_top_versa.sv(152)
# Time: 95752606 ps Iteration: 2 Instance: /tb_top_versa
#
```

Figure 6.28. Example Design Simulation Output for Non-Continuous Mode

6.6. Hardware Testing

The 5G Ethernet IP is hardware tested on Avant X70 Versa boards. The following setup demonstrates the steps for hardware testing on an Avant X70 Versa board:

1. Set up the board as shown in the following figure.
2. Ensure the input power is 12 V.
3. The JP17 jumper must be left open to enable the board to operate in 5 Gb. This is applicable for Avant-G/X Versa board design only.

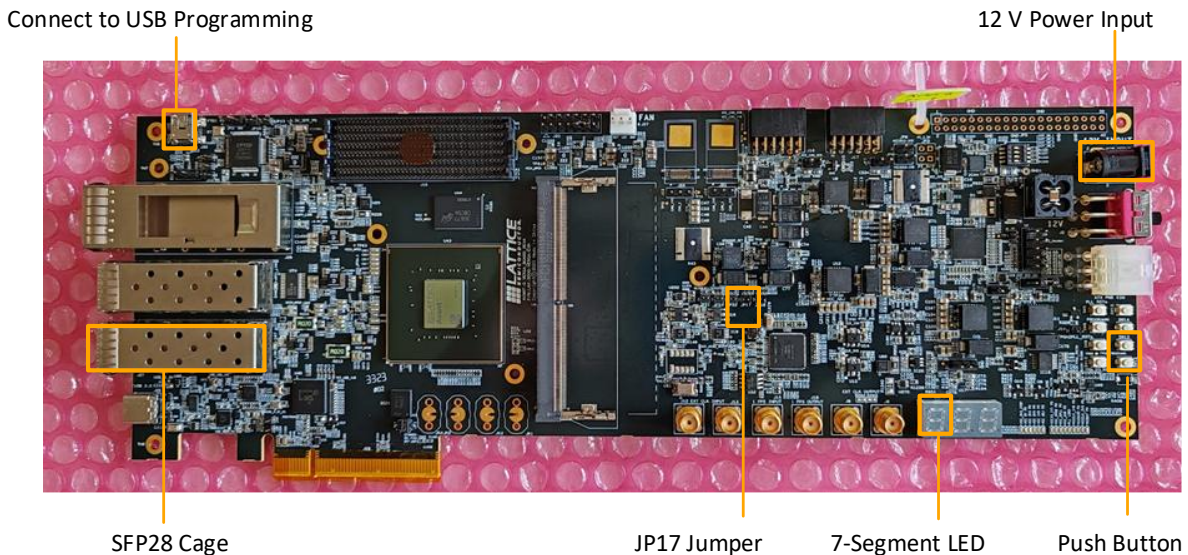


Figure 6.29. Avant-X Versa Board Setup

4. The following figure shows the 10 Gb SFP+ passive loopback module and the connection to the Avant-X Versa board SFP1.

SFP+ transceiver model: FS Cisco Compatible 10G SFP+ Passive Loopback Testing Module

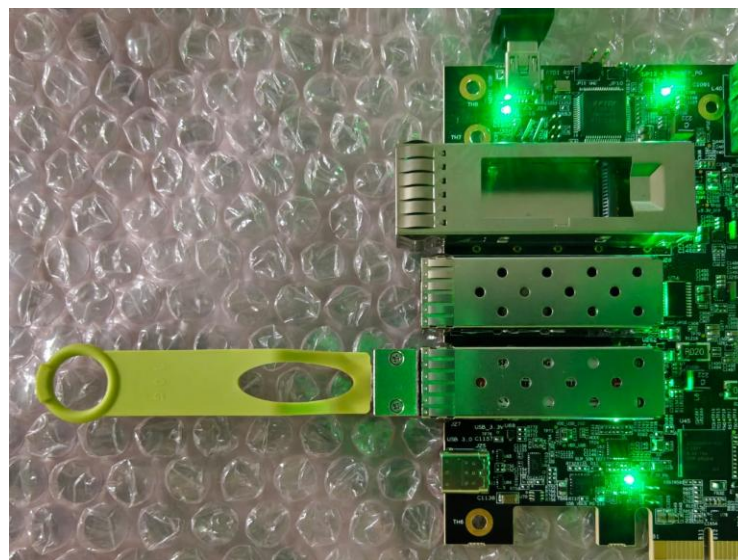


Figure 6.30. Loopback Module Setup

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

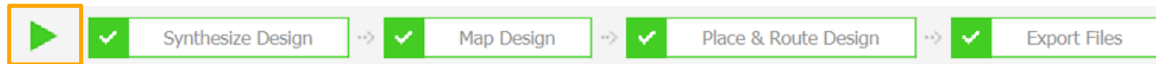




Figure 6.31. Successful Design Compilation

6. Launch the Radiant Programmer via menu **Tools** → **Programmer** or click on the Radiant Programmer icon .
7. Click on the **Detect Cable** button.
8. Select the correct cable associated with the Avant-X Versa board. The number of cables shown in your environment setup may be different. Select FTUSB-1 for Avant-X Versa board.
9. Select the correct bitstream file that is generated from the 5 Gb Ethernet project in step 5.
10. Click on the programming icon . The software displays the programming status.

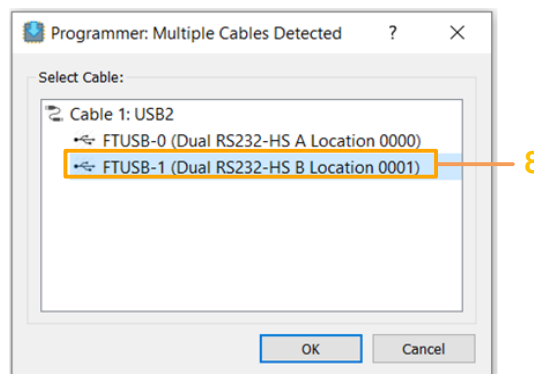


Figure 6.32. Cable Selection for Avant-AT-X Device

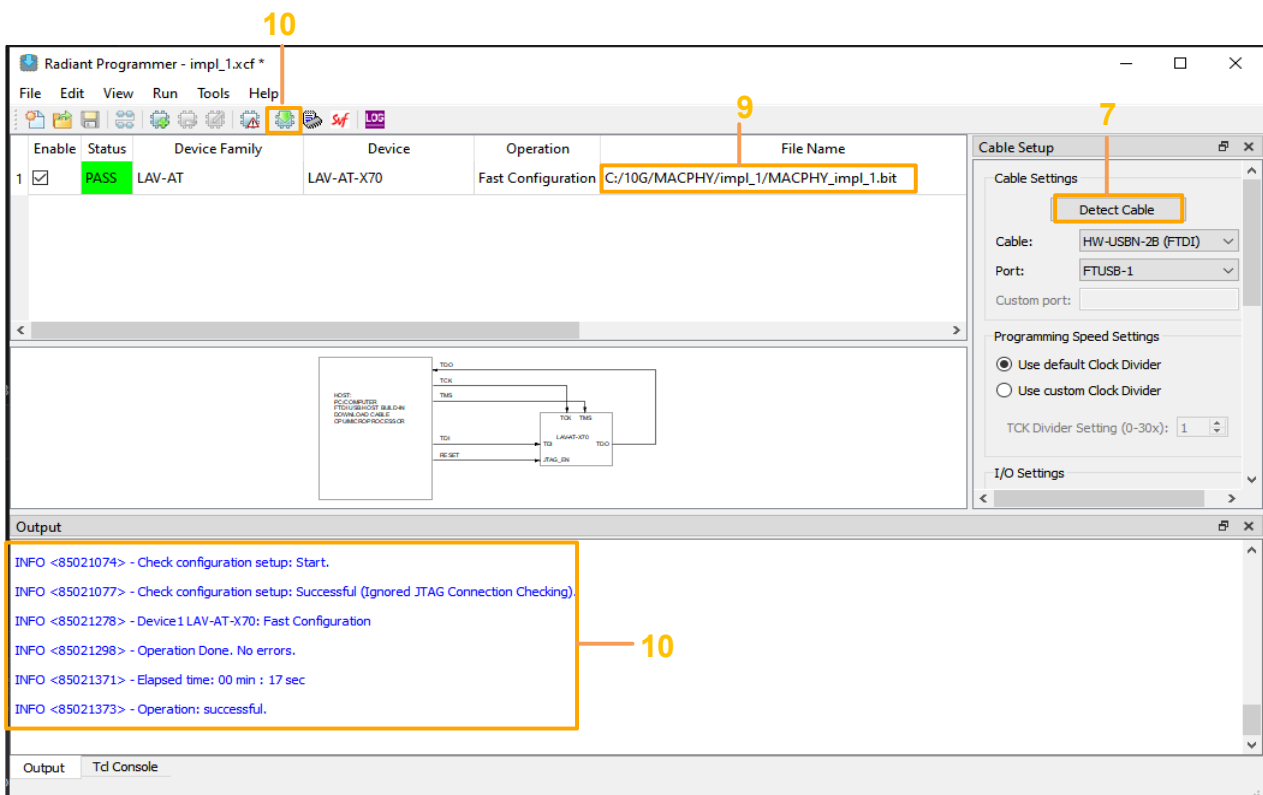


Figure 6.33. Programmer GUI

11. After the programming stage, press the pushbutton SW12 for Avant-X/G Versa board to trigger a forced reset even though the example design will automatically trigger a reset. Set the DIP12 switch for pattern generator to LOW before triggering a reset.
12. Press SW12 to RESET.
13. Press SW13 to PAUSE/ RESUME transmission.
14. Change SW14 pattern generator mode—Fix frame size, random frame size, or increment frame size.
15. After the design has linked up, toggle DIP_SW_1 for Avant-X/G Versa board 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. To change the pattern generator frame size mode, PAUSE the transmission then perform the changes.
16. You can pause and resume the Ethernet frames transmission by pressing the push button SW13* for Avant-X/G Versa board. The 7-segment LED in DIG1 glows according to the following diagram, which indicates that the hardware has been successfully set up and received a passing status. For LED definitions, refer to [Table 6.5](#).

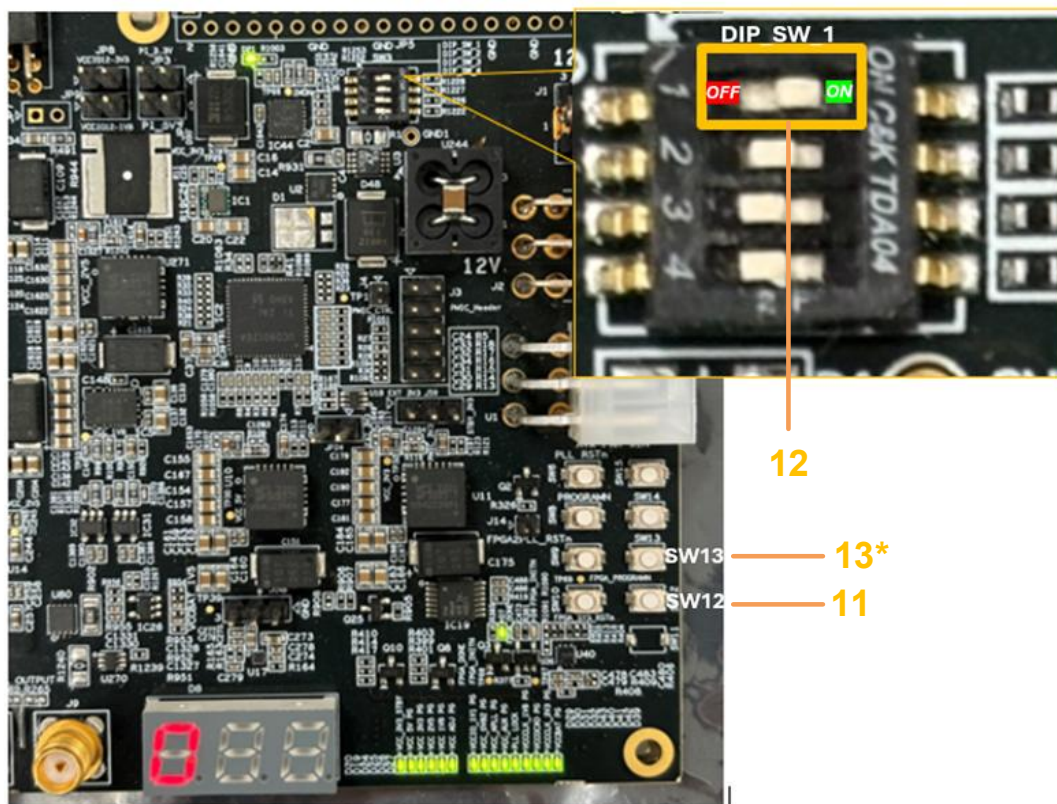


Figure 6.34. Push Buttons on Avant-AT-X Device

* **Note:** Applicable if continuous mode is enabled (CONTINUOUS_TRAFFIC=1). For 7 Segments LED definition, refer to [Table 6.5](#) and [Table 6.6](#).

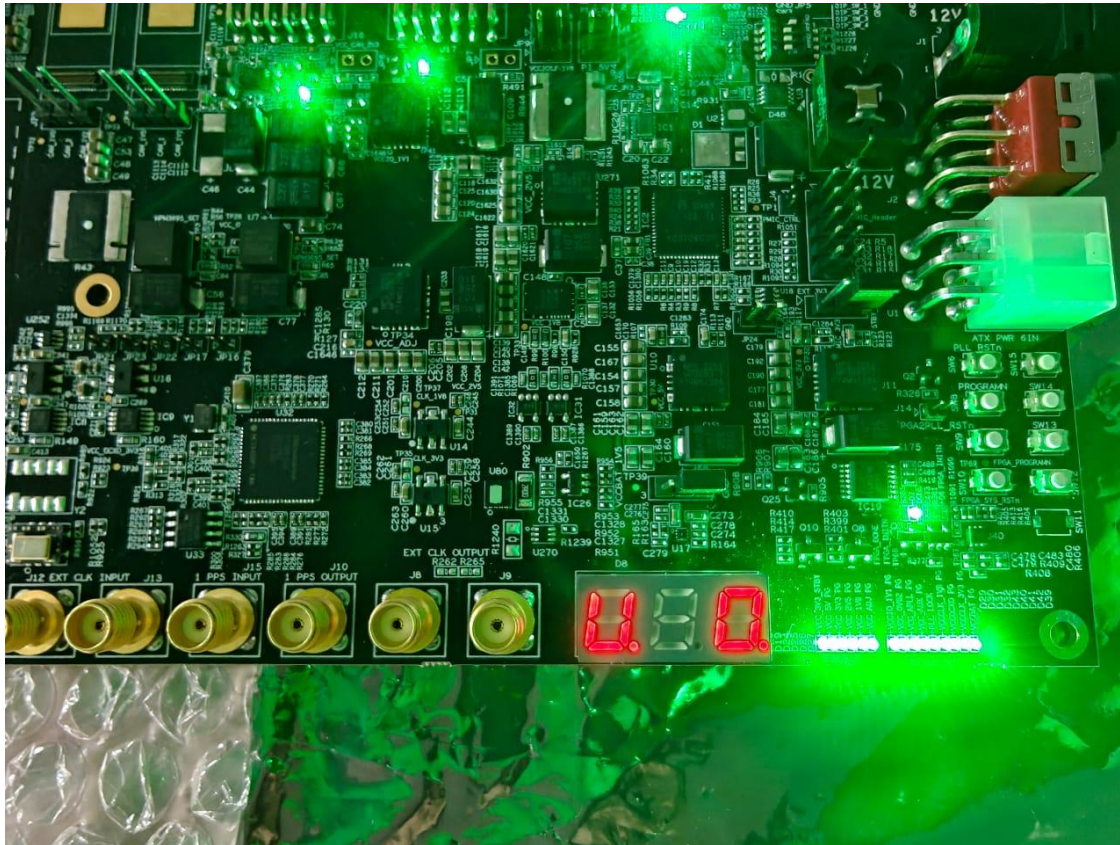


Figure 6.35. Passing Scenario on Avant-AT-X Device

17. 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 6.36. Failing Scenario on Avant-AT-X Device

18. The following diagram shows the various LED segments for Avant-X/G Versa board and the description for each segment is described in the following table.

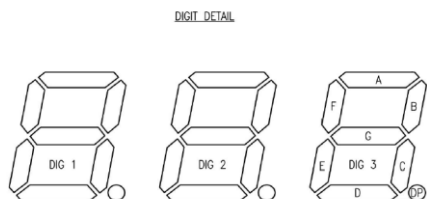


Figure 6.37. LED 7-Segment

Each segment in DIG 1 represents the status of XAUI IP. Refer to the following table for the description of each segment.

Table 6.5. LED 7-Segment Description for DIG 1

Segment	Description
A	Traffic transmission is completed or paused.
B	PLL lock achieved, clock from PHY output clock.
C	PHY initialization is completed.
D	Link up achieved.
E	TX transmission is up or transmitting.
F	RX transmission is up or receiving.
G	Data checker detected a mismatch.

***Note:** Default continuous mode and if 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.

DIG 3 displays the code corresponding to different types of frame length, including static or fixed frame size, randomized frame size, and incremental frame size. To change the type of frame size, use pushbutton SW14.

Table 6.6. LED 7-Segment Description for DIG 3

Code for DIG 3	Code Description
0	Fixed frame size.
1	Randomized frame size.
2	Incremental frame size.

The maximum and minimum boundary for randomized and incremental frame size can be defined in the top file, traffic_genchk module. The following values are based on the limitations of the traffic_genchk module:

- FRAME_LEN_MIN is the minimum value for randomized size, and the starting value for incremental frame size. The minimum length that is supported is 128.
- FRAME_LEN_MAX is the maximum value for randomized size, and the largest value for incremental frame size, which starts over again using the minimum value. The maximum length that is supported is 9,600.
- FRAME_LEN_INIT is used for defining the static frame size. The supported range is 128 to 9,600.

```

traffic_genchk #(
    .CONTINUOUS_TRAFFIC (CONTINUOUS_TRAFFIC),
    .MAX_DATA_WIDTH (AXI_DATA_WIDTH),
    .MASK_DATA ( (AXI_DATA_WIDTH == 128) ? 128'hEF45_4255_3A4C_5ECA_EF45_4255_3A4C_5ECA : 64'hEF45_4255_3A4C_5ECA ),
    .FRAME_LEN_MIN (16'd128),
    .FRAME_LEN_MAX (16'd9600),
    .FRAME_LEN_INIT (16'd1500)
) u_traffic_genchk (

```

Figure 6.38. Parameters for Frame Size in the Top File

7. Designing with the IP

This section provides information on 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 architecture. The following steps describe how to generate the 5G 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 **5 Gb Ethernet** under **IP, Connectivity** category. The **Module/IP Block Wizard** opens. Enter values in the **Component name** and the **Create in** fields and click **Next**.

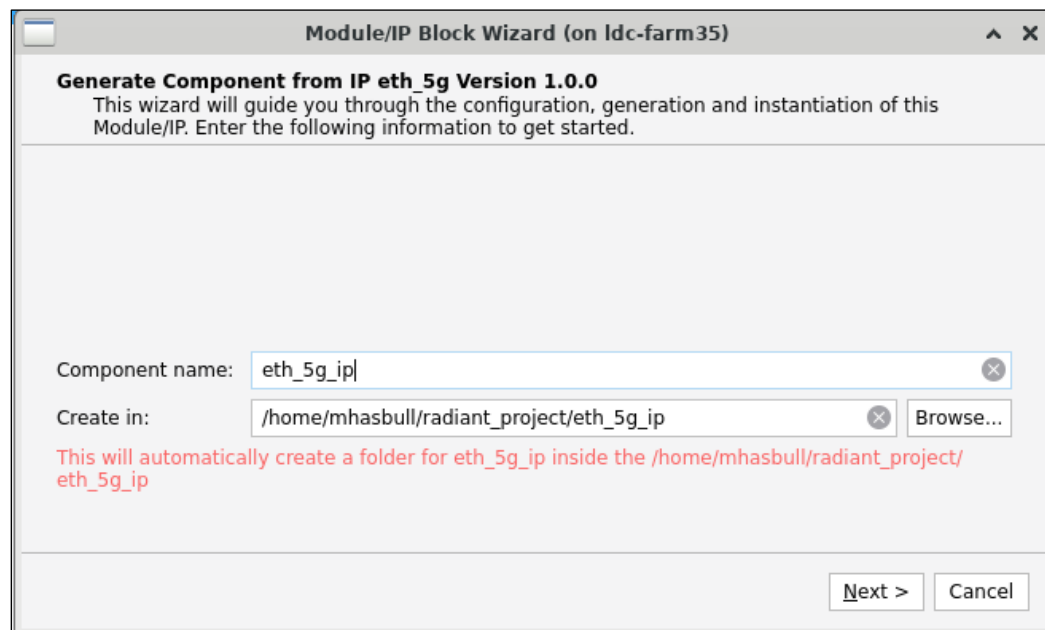


Figure 7.1. Module/IP Block Wizard

3. In the next **Module/IP Block Wizard** window, customize the selected 5G Ethernet IP core using the drop-down lists and checkboxes. The following figure shows a configuration example* of the 5G Ethernet IP core. For details on the configuration options, refer to the relevant *IP Parameter Description* section.

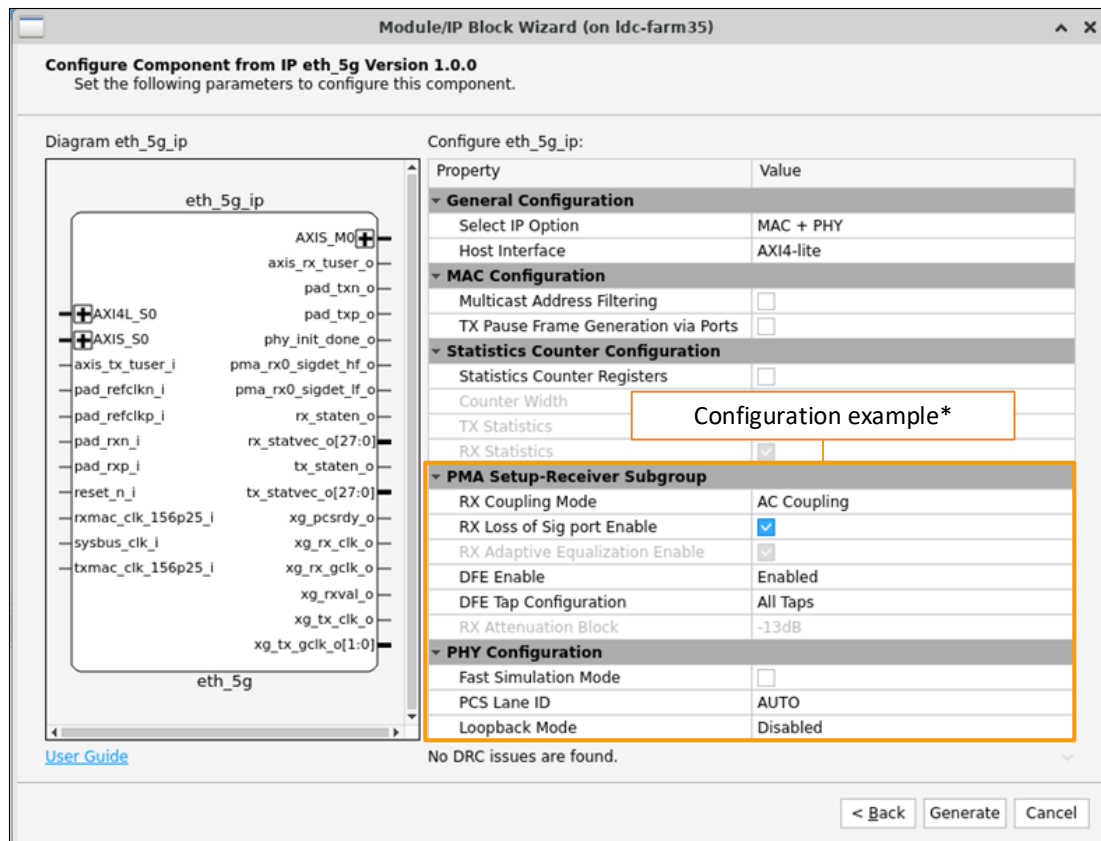


Figure 7.2. Configure the User Interface of 5G Ethernet IP Core

Note:

* Applicable for LAV-AT-G70 and LAV-AT-X70.

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

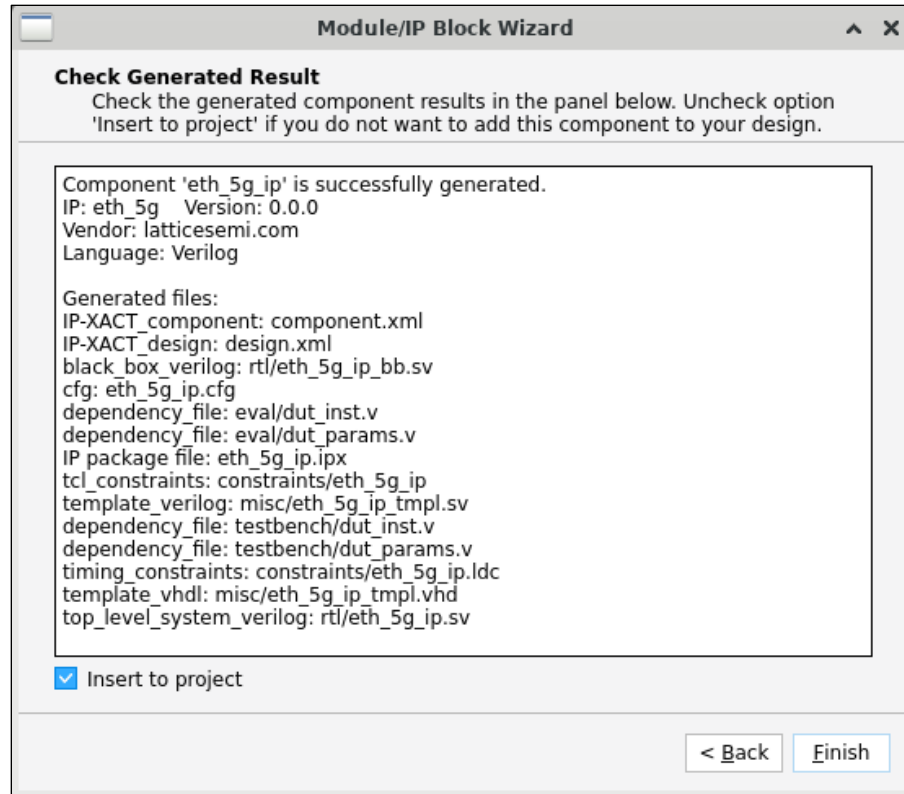


Figure 7.3. Check Generated Result

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

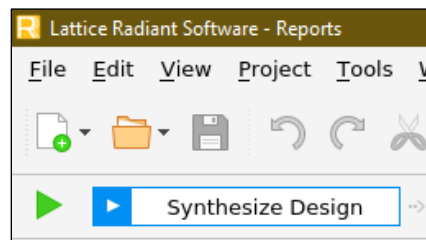


Figure 7.4. Synthesize Design

7.1.1. Generated Files and File Structure

The generated 5G Ethernet IP core package includes the closed box (*<Instance Name>_bb.v*) and instance templates (*<Instance Name>_tmpl.v/vhd*) that can be used to instantiate the core in a top-level design. 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 Files	Description
<i><Instance Name>.ipx</i>	This file contains the information on the files associated with the generated IP.
<i><Instance Name>.cfg</i>	This file contains the parameter values used in IP configuration.
<i>component.xml</i>	This file contains the ipxact:component information of the IP.
<i>design.xml</i>	This file documents the configuration parameters of the IP in IP-XACT 2014 format.
<i>rtl/<Instance Name>.v</i>	This file provides an RTL example of the top-level source file that instantiates the IP core.
<i>rtl/<Instance Name>_bb.v</i>	This file provides the synthesis closed-box.
<i>misc/<Instance Name>_tmpl.v</i> <i>misc /<Instance Name>_tmpl.vhd</i>	These files provide instance templates for the IP core.
<i>eval/versa_top/avant_x70_10g/.v</i>	This folder provides the hardware example design with simulation. Applicable to the <i>MAC + PHY</i> option. Note: For other generated files in the Eval folder, refer to Table 6.3. Description of Generated Files in the Eval Folder .
<i>testbench/tb_top.v</i>	Top testbench to run loopback test of the generated <i><Instance Name>.v</i> 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 or physical constraints.

```
#####
## For PDC file (Post-synthesis constraint file)
#####

## Use the constraints below in the PDC file if you want to run stand-alone MAC
## For STA purposes or Post-PAR netlist generation so that you
## will not encounter MAP issue (lacking IOs)

#ldc_set_attribute {VIRTUAL_IO=TRUE} [get_ports "apb_p"]
#ldc_set_attribute {VIRTUAL_IO=TRUE} [get_ports "*statvec*"]
#ldc_set_attribute {VIRTUAL_IO=TRUE} [get_ports "*staten*"]
#ldc_set_attribute {VIRTUAL_IO=TRUE} [get_ports "axis*"]
#ldc_set_attribute {VIRTUAL_IO=TRUE} [get_ports "xgmii*"]
#ldc_set_attribute {VIRTUAL_IO=TRUE} [get_ports "*mii*"]
#ldc_set_attribute {VIRTUAL_IO=TRUE} [get_ports int_o]

## Use the constraints below for stand-alone MAC; this is to exclude analysis between these clocks

##set_clock_groups -group [get_clocks sysbus_clk] -group [get_clocks rxmac_clk] -asynchronous
##set_clock_groups -group [get_clocks sysbus_clk] -group [get_clocks txmac_clk] -asynchronous

## Use the constraints below if you're using the eval/rtl design

#Management module clock
#create_clock -name {sysbus_clk} -period 50 [get_ports sysbus_clk_i]
#XGMII clock from PLL
#create_clock -name {xgmii_clk} -period 6.4 [get_nets xgmii_clk_o_c]

##Exclude in timing analysis since CDC is handled by FIFO
#set_clock_groups -group [get_clocks sysbus_clk] -group [get_clocks xgmii_clk] -asynchronous

##These paths are excluded in timing analysis
#set_false_path -from [get_nets -hierarchical {"rst" "reset"}]
#set_false_path -to [get_ports {sys_ready_o int_o}]
```

Figure 7.5. Timing Constraint File (.pdc) for the 5G MAC IP

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

Timing constraints (.sdc) is provided to ensure that the IP core meets the design timing requirement in Lattice FPGA device. The constraint file contains the necessary design constraints to ensure the proper timing closure. The timing constraints file is generated automatically during IP generation. However, clock constraints must be performed at user level design.

You must include the following clock constraints in your design per IP configuration. Note that you might need to modify the port name according to your RTL. Alternatively, you can find the following constraints in [<IP>/constraints/constraint.sdc](#).

7.3.1. Create Clock Constraints for MAC-only Option

```
#create 156.25MHz clock for serdes refclk pin
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 156.25MHz clock for rxmac_clk_156p25_i and txmac_clk_156p25_i
create_clock -name {rx_clk_1} -period 6.4 [get_ports rxmac_clk_156p25_i]
create_clock -name {tx_clk_1} -period 6.4 [get_ports txmac_clk_156p25_i]

#create 78.125MHz clock for rxmac_clk_78p125_i and txmac_clk_78p125_i
create_clock -name {rxmac_clk_2} -period 12.8 [get_ports rxmac_clk_78p125_i]
create_clock -name {txmac_clk_2} -period 12.8 [get_ports txmac_clk_78p125_i]

#create 100MHz(Max) clock for sysbus_clk_i
#The maximum frequency is 100MHz and users can change the value per design requirement.
create_clock -name {sysbus_clk} -period 10 [get_ports sysbus_clk_i]
```

7.3.2. Create Clock Constraints for MAC + PHY Option

```
#create 156.25MHz clock for serdes refclk pin
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 156.25MHz clock for rxmac_clk_156p25_i and txmac_clk_156p25_i
create_clock -name {rx_clk_1} -period 6.4 [get_ports rxmac_clk_156p25_i]
create_clock -name {tx_clk_1} -period 6.4 [get_ports txmac_clk_156p25_i]

#create 100MHz(Max) clock for sysbus_clk_i
#The maximum frequency is 100MHz and users can change the value per design requirement.
create_clock -name {sysbus_clk} -period 10 [get_ports sysbus_clk_i]
```

7.3.3. Create Clock Constraints for PHY-only Option

```
#create 156.25 MHz clock for serdes refclk pin
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 100MHz(Max) clock for sysbus_clk_i
#The maximum frequency is 100MHz and users can change the value per design requirement.
create_clock -name {sysbus_clk} -period 10 [get_ports sysbus_clk_i]
```

For more information on timing constraints, refer to the [Lattice Radiant Timing Constraints Methodology Application Note \(FPGA-AN-02059\)](#).

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.

To run the standalone QuestaSim software, the FOUNDRY environment variable must be set as follow:

```
set ::env(FOUNDRY) <Radiant installation path>/ispfpga
```

To run functional simulation, follow these steps:

1. Click the  button located on the **Toolbar** to initiate the **Simulation Wizard**, as shown in [Figure 7.6](#).

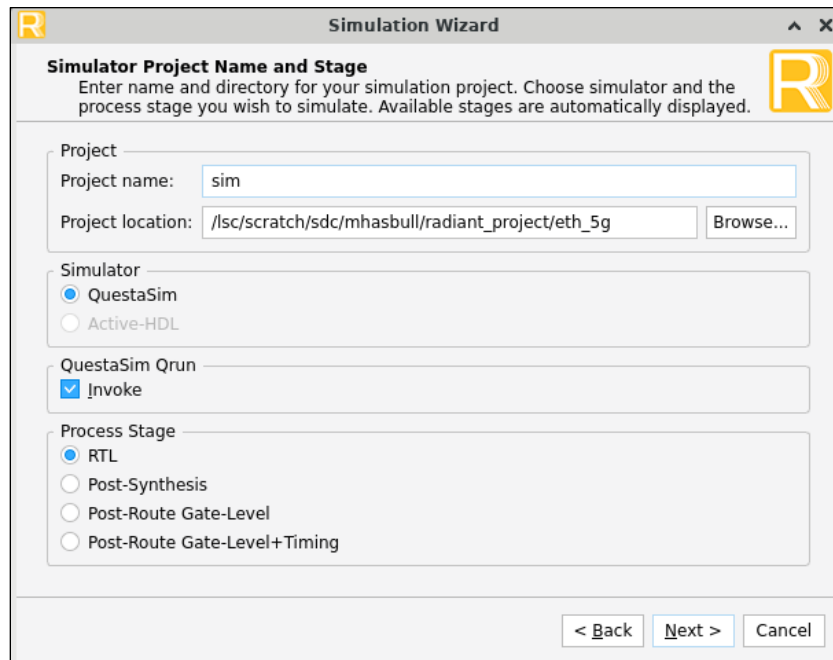


Figure 7.6. Simulation Wizard

- Click **Next** to open the **Add and Reorder Source** window, as shown in the following figure.

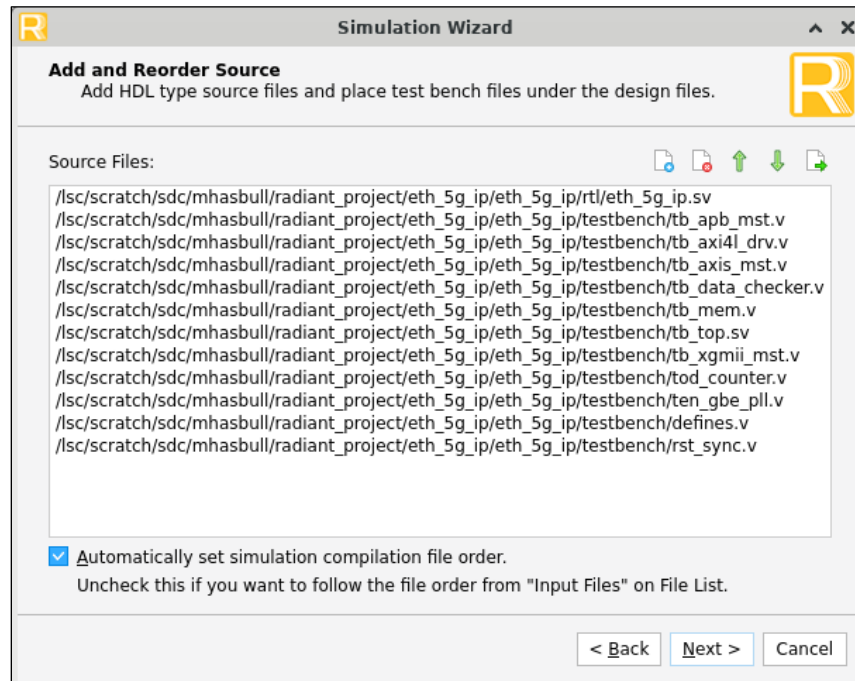


Figure 7.7. Add and Reorder Source

- Click **Next**. The **Summary** window is shown.

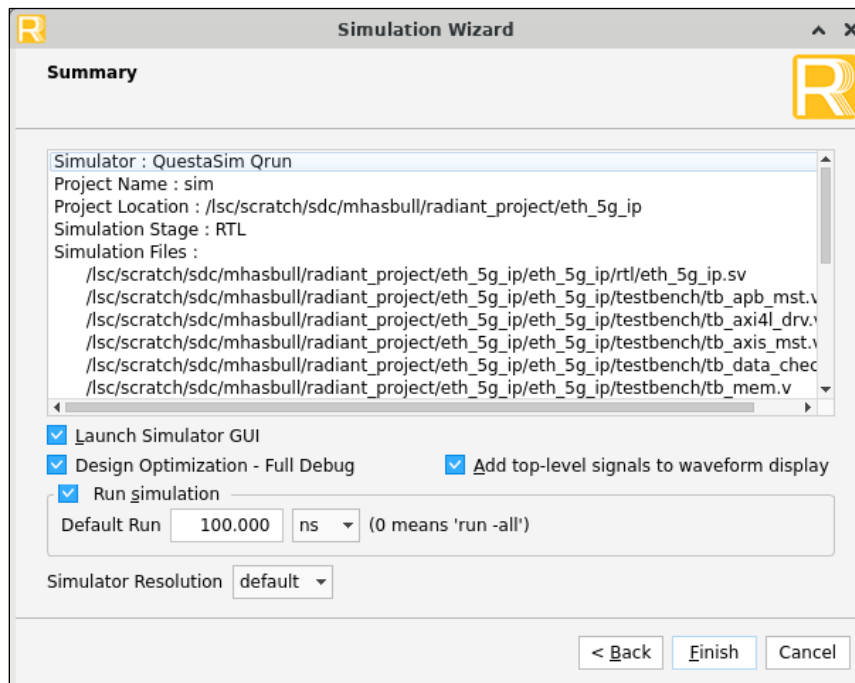


Figure 7.8. Summary Window

- 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.

7.5.1. Simulation Results

The following waveform shows an example of the simulation result.

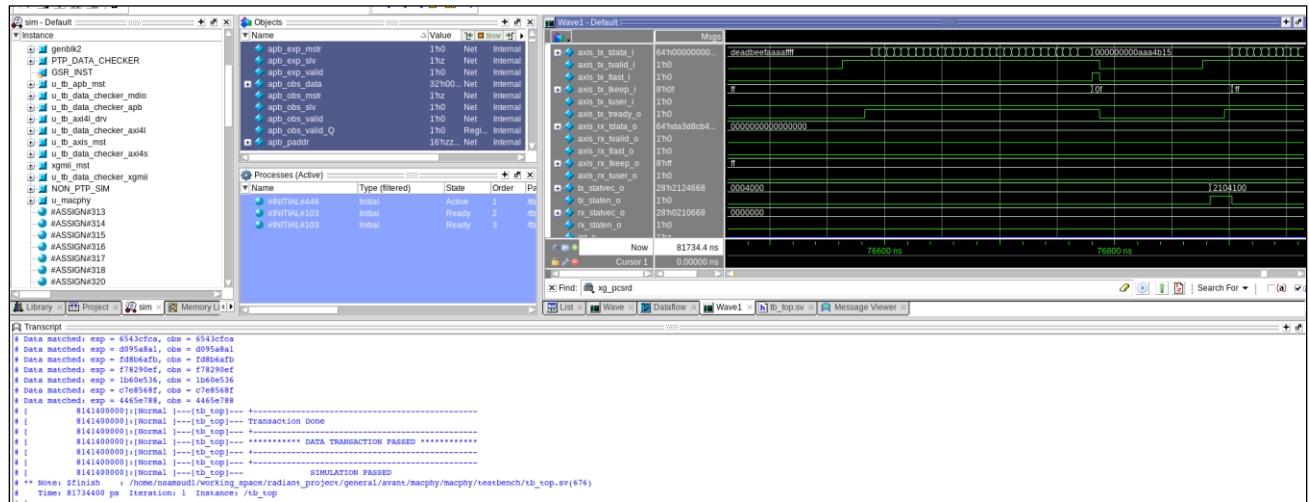


Figure 7.9. Simulation Results

Appendix A. Resource Utilization

Table A.1. Resource Utilization for Avant Devices

Configuration	Registers	LUTs	EBRs	Target Device	Synthesis Tools
PHY only Host Interface == AXI4-Lite	283 (<1%)	445 (<1%)	0 (0%)	LAV-AT-G70	Synplify Pro
PHY only Host Interface == APB	261 (<1%)	370 (<1%)	0 (0%)	LAV-AT-G70	Synplify Pro
MAC + PHY Host Interface == AXI4-Lite	2,609 (1%)	2,859 (1%)	6 (<1%)	LAV-AT-G70	Synplify Pro
MAC + PHY Multicast Address Filtering == Enabled Statistics Counter Registers == Enabled Counter Width == 32 Host Interface == AXI4-Lite	2,624 (2%)	2,914 (2%)	6 (<1%)	LAV-AT-G70	Synplify Pro
MAC + PHY Multicast Address Filtering == Enabled Statistics Counter Registers == Enabled Counter Width == 64 Host Interface == AXI4-Lite	2,624 (3%)	2,916 (3%)	6 (<1%)	LAV-AT-G70	Synplify Pro
MAC only Multicast Address Filtering == Enabled	1,213 (1%)	1,431 (1%)	4 (<1%)	LAV-AT-G70	Synplify Pro
MAC only Multicast Address Filtering == Enabled Statistics Counter Registers == Enabled Counter Width == 64	1,213 (3%)	1,324 (3%)	4 (<1%)	LAV-AT-G70	Synplify Pro

References

- [5G Ethernet IP Release Notes \(FPGA-RN-02102\)](#)
- [Lattice Avant-G/X MPPHY Module User Guide \(FPGA-IPUG-02233\)](#)
- [Lattice Memory Mapped Interface and Lattice Interrupt Interface User Guide \(FPGA-UG-02039\)](#)
- [Multi-Protocol PCS Module User Guide \(FPGA-IPUG-02118\)](#)
- [5G Ethernet IP web page](#)
- [Lattice Radiant Timing Constraints Methodology Application Note \(FPGA-AN-02059\)](#)
- [Lattice Radiant Software web page](#)
- [Avant-G web page](#)
- [Avant-X web page](#)
- [IP Cores and Reference Designs for Avant Devices web page](#)
- [Kits, Boards, and Demonstrations for Avant Devices web page](#)
- [Lattice Insights web page for Lattice Semiconductor 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 1.0, IP v1.0.0, December 2025

Section	Change Summary
All	Initial release.



www.latticesemi.com