10 Gb Ethernet MAC IP Core – Lattice Radiant Software

User Guide

FPGA-IPUG-02162-1.1

November 2021

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 associated risk the responsibility entirely of the Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. 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. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice's product could create a situation where personal injury, death, severe property or environmental damage may occur. 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.

Contents

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
APB	Advanced Peripheral Bus
AXI4-Stream	Advanced eXtensible Interface 4 Stream
GPLL	Generic Phase-Locked Loop
MAC	Media Access Controller
MTU	Maximum Transmission Unit
PCS	Physical Coding Sublayer
XGMII	10-Gigabit Media Independent Interface

1. Introduction

The Lattice Semiconductor 10 Gb Ethernet MAC IP Core supports the ability to transmit and receive data between a host processor and an Ethernet network. The main function of the 10 Gb Ethernet MAC is to ensure that the Media Access rules specified in the 802.3 IEEE standards are met while transmitting a frame of data over Ethernet. On the receive side, the Ethernet MAC extracts the different components of a frame and transfers them to higher applications through an AXI4-stream interface.

Figure 1.1 shows an example of a 10GBase-R application. The 10 Gb Ethernet MAC IP Core is connected to the 10G Ethernet PCS IP Core and its clock source is from the GPLL.

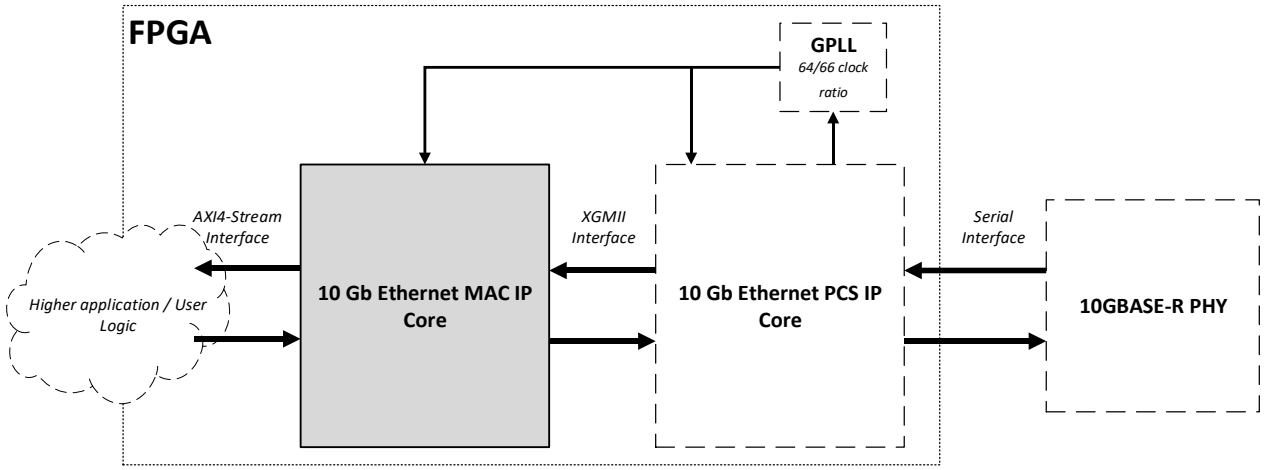

Figure 1.1. 10GBase-R Application

1.1. Quick Facts

Table 1.1 provides quick facts about the 10 Gb Ethernet MAC IP Core.

Table 1.1. 10 Gb Ethernet MAC IP Core Quick Facts

IP Requirements	Supported FPGA Family	CertusPro™-NX
Resource Utilization	Targeted Devices	LFCPNX-100
	Supported User Interface	AXI4-Stream/APB/LINTR Interface
	Resources	See Table A.1
Design Tool Support	Lattice Implementation	IP Core v1.0.x – Lattice Radiant™ software 3.0 or later
	Synthesis	Lattice Synthesis Engine
		Synopsys® Synplify Pro®, O-2018.09LR-SP1
	Simulation	For the list of supported simulators, see the Lattice Radiant Software User Guide.

1.2. Features

The key features of the 10 Gb Ethernet MAC IP Core include:

- Compliant to IEEE 802.3-2012 standard
- Supports standard 10 Gbps Ethernet link layer data rate
- 64-bit wide internal data path operating at 156.25 MHz
- AXI4-stream interface on Client transmit and receive interfaces
- Supports Deficit Idle Count
- Supports VLAN, Jumbo Frames and WAN mode
- Custom Preamble mode
- Independent TX and RX Maximum Transmission Unit (MTU) frame length
- Comprehensive Statistics Support
- Optional 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
- APB interface for register access
- Supports 8-bit/16-bit GMII or 4-bit MII to and from Physical (PHY) layer

1.3. Conventions

1.3.1. Nomenclature

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

1.3.2. Signal Names

Signal names that end with:

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

1.3.3. Attribute Names

Attribute names in this document are formatted in title case and italicized (*Attribute Name*).

2. Functional Description

2.1. Overview

The 10 Gb Ethernet MAC IP Core transmits and receives data between a host processor and an Ethernet network. The main function of the 10 Gb Ethernet MAC is to ensure that the Media Access rules specified in the 802.3 IEEE standards are met while transmitting a frame of data over Ethernet. On the receive side, the Ethernet MAC extracts the different components of a frame and transfers them to higher applications through an AXI4-stream interface.

2.2. Block Diagram

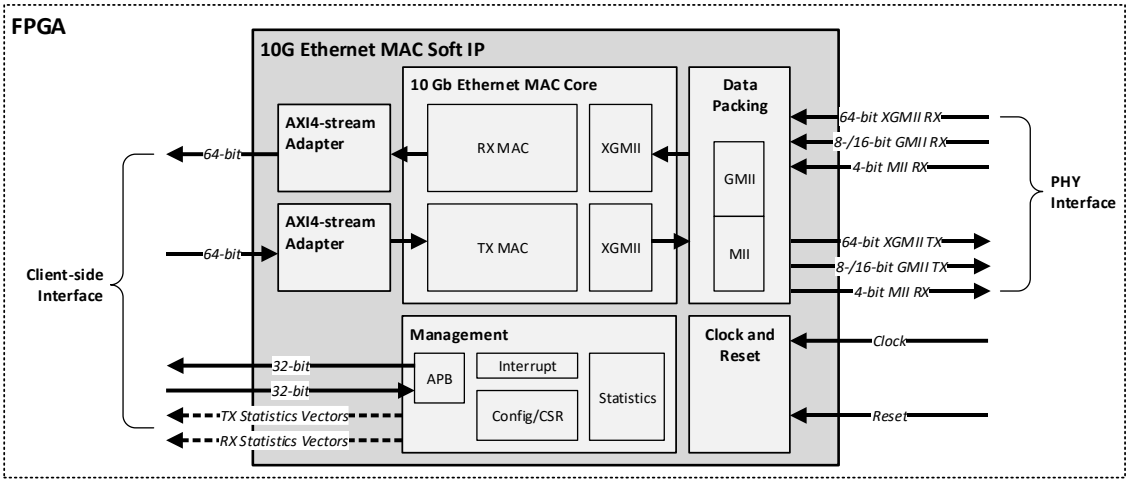

Figure 2.1. 10 Gb Ethernet MAC IP Core Block Diagram

2.3. Signal Description

Table 2.1. Signal Description

Port Name	I/O	Width	Default	Description
Clock and Reset				
reset_n_i	In	1	—	Active-low asynchronous reset.
rx_mac_clk_i	In	1	—	Clock for Receive AXI4-stream and XGMII or 8-bit/16-bit GMII or MII data path. 156.25 MHz clock for XGMII Interface. 125 MHz clock for 8-bit GMII Interface. 156.25 MHz clock for 16-bit Interface. 125 MHz clock for MII RX data.
tx_mac_clk_i	In	1	—	Clock for Transmit AXI4-stream and XGMII/GMII/MII data path. 156.25 MHz clock for XGMII Interface. 125 MHz clock for 8-bit GMII Interface. 156.25 MHz clock for 16-bit Interface. 125 MHz clock for MII Interface.
apb_clk_i	In	1	—	Clock for Management module
PHY Interface				
XGMII Interface[1]				
xgmii_rxd_i	In	64	—	8-lane Receive SDR XGMII data from PHY. • 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	In	8	—	Control bits for each lane in xgmii_rxd_i[ ]. • Lane 0: xgmii_rxc_i[0] • Lane 1: xgmii_rxc_i[1] • 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	Out	64	64'h0	8-lane Transmit SDR XGMII data to PHY. • 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]

Port Name	I/O	Width	Default	Description
xgmii_txc_o	Out	8	8'h0	Control bits for each lane in xgmii_txd_o[ ]. • 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]
GMII Interface[2]				
gmii_rx_d_i	In	8	-	8-bit GMII RX data.
gmii_rx_dv_i	In	1	-	Indicates the GMII RX data is valid when asserted.
gmii_rx_err_i	In	1	-	Indicates the GMII RX data contains error.
gmii_tx_d_o	Out	8	8'h0	8-bit GMII TX data.
gmii_tx_en_o	Out	1	1'b0	Indicates the GMII TX data is valid when asserted.
gmii_tx_err_o	Out	1	1'b0	Indicates the GMII RX data contains error.
gmii_16_rx_d_i	In	16	—	16-bit GMII RX data.
gmii_16_rx_dv_i	In	2	—	Indicates the GMII RX data is valid when asserted.
gmii_16_rx_err_i	In	2	—	Indicates the GMII RX data contains error.
gmii_16_tx_d_o	Out	16	16'h0	16-bit GMII TX data.
gmii_16_tx_en_o	Out	2	2'h0	Indicates the GMII TX data is valid when asserted.
gmii_16_tx_err_o	Out	2	2'h0	Indicates the GMII RX data contains error.
MII Interface[3]				
mii_rx_d_i	In	4	—	4-bit MII RX data.
mii_rx_dv_i	In	1	—	Indicates the MII RX data is valid when asserted.
mii_rx_err_i	In	1	—	Indicates the MII RX data contains error.
mii_tx_d_o	Out	4	4'h0	4-bit MII TX data.
mii_tx_en_o	Out	1	1'b0	Indicates the MII TX data is valid when asserted.
mii_tx_err_o	Out	1	1'b0	Indicates the MII RX data contains error.
Client-Side Interface				
AXI4-Stream Receive Interface				
axis_rx_tdata_o	Out	64	64'h0	AXI4-Stream data from PHY to client.
axis_rx_tkeep_o	Out	8	8'h0	AXI4-Stream control that indicates which data is valid on axis_rx_tdata_o[ ]. • axis_rx_tkeep_o[0]: axis_rx_tdata_o [7:0] • axis_rx_tkeep_o[1]: axis_rx_tdata_o [15:8] • axis_rx_tkeep_o[2]: axis_rx_tdata_o [23:16] • axis_rx_tkeep_o[3]: axis_rx_tdata_o [31:24] • axis_rx_tkeep_o[4]: axis_rx_tdata_o [39:32] • axis_rx_tkeep_o[5]: axis_rx_tdata_o [47:40] • axis_rx_tkeep_o[6]: axis_rx_tdata_o [55:48] • axis_rx_tkeep_o[7]: axis_rx_tdata_o [63:56]
axis_rx_tvalid_o	Out	1	1'b0	AXI4-Stream data valid.
axis_rx_tuser_o	Out	1	1'b0	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	Out	1	1'b0	AXI4-Stream signal indicating the end of a packet.

Port Name	I/O	Width	Default	Description
AXI4-Stream Transmit Interface				
axis_tx_tready_o	Out	64	—	AXI4-Stream signal indicating that the core can accept data transfer.
axis_tx_tdata_i	In	64	64'h0	AXI4-Stream data from client
axis_tx_tkeep_i	In	8	8'h0	AXI4-Stream control that indicates which data is valid on axis_rx_tdata_o[ ]. • axis_rx_tkeep_o[0]: axis_rx_tdata_o [7:0] • axis_rx_tkeep_o[1]: axis_rx_tdata_o [15:8] • axis_rx_tkeep_o[2]: axis_rx_tdata_o [23:16] • axis_rx_tkeep_o[3]: axis_rx_tdata_o [31:24] • axis_rx_tkeep_o[4]: axis_rx_tdata_o [39:32] • axis_rx_tkeep_o[5]: axis_rx_tdata_o [47:40] • axis_rx_tkeep_o[6]: axis_rx_tdata_o [55:48] • axis_rx_tkeep_o[7]: axis_rx_tdata_o [63:56]
axis_tx_tvalid_i	In	1	1'b0	AXI4-Stream data valid
axis_tx_tuser_i	In	1	1'b0	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	In	1	1'b0	AXI4-Stream signal indicating the end of a packet.
APB Interface				
apb_psel_i	I	1	—	Select signal. Indicates that the slave device is selected and a data transfer is required.
apb_paddr_i	I	32	—	Address signal.
apb_pwdata_i	I	32	—	Write data signal.
apb_pwrite_i	I	1	—	Direction signal. Write = 1, Read = 0
apb_penable_i	I	1	—	Enable signal. Indicates the second and subsequent cycles of an APB transfer.
apb_pready_o	O	1	0	Ready signal. Indicates transfer completion. Slave uses this signal to extend an APB transfer.
apb_prdata_o	O	32	0	Read data signal.
Statistics Vector Interface				
tx_statvec_o	O	26	26'h000 4000	Contains information on the frame transmitted. This bus is qualified by the tx_staten_o signal. • tx_statvec_o[13:0]: Frame byte count • tx_statvec_o[14]: Transmit is OK • tx_statvec_o[15]: MAC control inserted by MAC • tx_statvec_o[16]: MAC control inserted by Client • tx_statvec_o[17]: Jumbo frame • tx_statvec_o[18]: Tagged frame • tx_statvec_o[19]: Broadcast address • tx_statvec_o[20]: Multicast address • tx_statvec_o[21]: Underrun error • tx_statvec_o[22]: CRC error • tx_statvec_o[23]: Length check error • tx_statvec_o[24]: Terminate error • rx_statvec_o[25]: Long Frame error
tx_staten_o	O	1	1'b0	When asserted, indicates that the contents of the tx_statvec_o bus are valid. This signal is asserted for 3 tx_mac_clk_i periods.

Port Name	I/O	Width	Default	Description
rx_statvec_o	O	26	26'h0	Contains information on the frame received. This bus is qualified by the rx_staten_o signal. • rx_statvec_o[13:0]: Frame byte count • rx_statvec_o[14]: Frame dropped • rx_statvec_o[15]: Broadcast frame received • rx_statvec_o[16]: Multicast frame received • rx_statvec_o[17]: CRC error • rx_statvec_o[18]: VLAN Tag detected • rx_statvec_o[19]: PAUSE frame • rx_statvec_o[20]: Length check error • rx_statvec_o[21]: Frame is too long • rx_statvec_o[22]: MAC Address mismatch • rx_statvec_o[23]: Unsupported opcode. Only the opcode for PAUSE frame is supported. • rx_statvec_o[24]: Minimum IPG Violated • rx_statvec_o[25]: Receive packet ignored
rx_staten_o	O	1	1'b0	When asserted, indicates that the contents of the tx_statvec_o bus are valid. This signal is asserted for 3 rx_mac_clk_i periods.
LINTR				
int_o	O	1	0	Interrupt request. See Interrupt registers for the details of the assertion of this signal.
Miscellaneous				
speed_sel_i[4]	I	2	0	PHY speed selection. • 0x0: XGMII interface • 0x1: 8-bit GMII interface • 0x2: 16-bit GMII interface • 0x3: MII interface Make sure that the MAC are idle with no packet transmission. After modifying this signal, Client is required to trigger the reset.

Notes:

1. Available when PHY Inerface== XGMII or Dynamic Speed Selection == Enabled
2. 8-bit GMII is available when PHY Interface== 8-bit GMI or Dynamic Speed Selection == Enabled; 16-bit GMII signals is available when PHY Interface== 16-bit GMII or Dynamic Speed Selection == Enabled
3. Available when PHY Interface==MII or Dynamic Speed Selection == Enabled
4. Available when Dynamic Speed Selection == Enabled

2.4. Attribute Summary

The 10 Gb Ethernet MAC IP Core configurable attributes are as shown in Table 2.2. The values set for attributes with corresponding registers serves as the maximum values and cannot set higher than these values during dynamic reconfiguration.

Table 2.2. Attributes Table

Attribute	Selectable Values	Default	Description	Dependency on Other Attributes
General Configuration				
PHY Interface	"XGMII", "8-bit GMII", "16-bit GMII", "MII"	"XGMII"	Set the default PHY Interface.	—
Dynamic Speed Selection	Enabled, Disabled	Disabled	Enables or disables dynamic speed selection.	—
Multicast Address Filtering	Enabled, Disabled	Disabled	Enables or disables address filtering for Multicast frames	—
Statistics Counter Registers	Enabled, Disabled	Disabled	Enabled or disables statistics counter registers	—
Non-XGMII Configuration				
Frame Type	"Standard", "Jumbo", "Super Jumbo"	"Super Jumbo"	Set Ethernet frame type to determine FIFO depth for 8-bit/16-bit GMII or MII configurations or when Dynamic Selection is enabled.	Enabled when *Dynamic Speed Selection == Enabled*, or *PHY Interface != XGMII*

2.5. Registers Description

All registers are accessed through the APB interface. Access Types of each registers are defined in Table 2.3.

Table 2.3. Access Types

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

2.5.1. Configuration Registers

The 10 Gb Ethernet MAC configuration registers are summarized in Table 2.4. These registers are accessed through APB Interface.

Table 2.4. Summary of Configuration Registers

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

2.5.1.1. MODE Register

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

Table 2.5. MODE Register

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

2.5.1.2. 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 2.6. TX_CTL Register

Bit	Label	Description	Default
[31:5]	RSVD	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.	0
[3]	transmit_short	Transmit Short Frames When set to 1, enables the TX MAC to transmit frames shorter than 64 bytes without adding padding bytes. When set to 0, TX MAC adds padding bytes when frames are shorter than 64 bytes before transmitted to the PHY.	0
[2]	tx_ipg_stretch	IFG Stretch Mode When set to 1, the TX MAC operates in the IFG stretch mode, to match the data rates of OC-192. The IPG required to match OC-192 is added to the value specified in IPG_VAL register.	0
[1]	tx_fc_en	Flow-control Enable When set to 1, this enables the flow control functionality of the TX MAC. This bit should be set for the TX MAC to transmit a PAUSE frame.	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

2.5.1.3. 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 behaviour.

Table 2.7. RX_CTL Register

Bit	Label	Description	Default
[15:8]	RSVD	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.	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
[4]	receive_bc	Receive Broadcast Frames. When set to 1, 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 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 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

Bit	Label	Description	Default
[0]	prms	Promiscuous Mode. When set to 1, all filtering schemes are abandoned and the RX MAC receives frames with any address.	0

2.5.1.4. MAX_PKT_LNGTH 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 2.8. MAX_PKT_LNGTH Register

Bit	Label	Description	Default
[31:14]	RSVD	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

2.5.1.5. 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 Deficit Idle Count.

Table 2.9. IPG_VAL Register

Bit	Label	Description	Default
[15:5]	RSVD	Reserved bits.	
[4:0]	tx_ipg	Transmit Inter Packet Gap Specifies the amount of inter-frame gap in increments of 4 bytes. Refer to Section 2.8 for details. A value of 0 of this register is prohibited.	5'h1

2.5.1.6. MAC_ADDR_0 and MAC_ADDR_1 Register

The MAC address is stored in the registers in Hexadecimal form, so for example to set the MAC Address to: AC-DE-48-00-00-80 would require writing 0x00_48_DE_AC to address 0x014 (MAC_ADDR_0). 0x80_00 to address 0x018 (MAC_ADDR_1).

Table 2.10. 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 supported by the MAC.	0

Table 2.11. MAC_ADDR_1 Register

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

2.5.1.7. TX_RX_STS Register

Table 2.12. TX_RX_STS Register

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

2.5.1.8. VLAN_TAG Register

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

Table 2.13. VLAN_TAG Register

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

2.5.1.9. 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 Receive MAC and Appendix B.

Table 2.14. 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 2.15. 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

2.5.1.10. PAUSE_OPCODE Register

This register contains the PAUSE Opcode. This is compared against the Opcode in the received PAUSE frame. This value is also included in any PAUSE frame transmitted by the MAC.

Table 2.16. PAUSE_OPCODE Register

Bit	Label	Description	Default
[31:16]	RSVD	Reserved bits.	0
[15:0]	pause_opcode	PAUSE Opcode	16'h01

2.5.1.11. MAC_CTL Register

Table 2.17. MAC_CTL Register

Bit	Label	Description	Default
[31:5]	RSVD	Reserved bits.	
[4]	ignore_pkt	Ignore Packet. When set to 1, the RX MAC ignores/drops incoming packets.	0
[3:1]	RSVD	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.	0

2.5.1.12. PAUSE_TM Register

This register has the pause time for a flow control packet sourced by the 10 Gb MAC Transmitter.

Table 2.18. PAUSE_TM Register

Bit	Label	Description	Default
[31:16]	reserved		—
[15:0]	tx_paustim	PAUSE duration.	0

2.5.2. Interrupt Registers

Refer to the Lattice Interrupt Interface (LINTR) User Guide for details of these registers.

Table 2.19. 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

2.5.2.1. INT_STATUS Register

Table 2.20. INT_STATUS Register

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

2.5.2.2. INT_ENABLE Register

Table 2.21. INT_ENABLE Register

Bit	Label	Description	Default
[31:2]	RSVD	Reserved bits	0
[1]	remote_fault_en	Enable remote_fault_int	0
[0]	local_fault_en	Enable local_fault_int	0

2.5.2.3. INT_SET Register

Table 2.22. INT_SET Register

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

2.5.3. Statistics Counters

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

Register name with "_0" means the least significant word of the counter and "_1" is the most significant word.

Table 2.23. Summary of Statistics Counters

Offset	Register Name	Access	Description
0x044	TX_STAT_PKT_LNGTH_0	RO	Transmit Packet Length Statistics Counter. Indicates the total number of octet transmitted in a particular frame. tx_statvec_o[13:0] is used to implement this counter.
0x048	TX_STAT_PKT_LNGTH_1	RO	
0x04C	TX_STAT_ERR_0	RO	Transmit TX Error Statistics Counter. Counts the total number of PHY terminated packet. The tx_statvec_o[24] is used to implement this counter.
0x050	TX_STAT_ERR_1	RO	
0x054	TX_STAT_UNDER_RUN_0	RO	Transmit Underrun Error Statistics Counter. Counts the total number of underrun packets transmitted. tx_statvec_o[21] is used to implement this counter.
0x058	TX_STAT_UNDER_RUN_1	RO	
0x05C	TX_STAT_CRC_ERR_0	RO	Transmit CRC Error Statistics Counter. Counts the total number of packets transmitted with crc error. tx_statvec_o[22] is used to implement this counter.
0x060	TX_STAT_CRC_ERR_1	RO	
0x064	TX_STAT_LNGTH_ERR_0	RO	Transmit Length Error Statistics Counter. Counts the total number of packets transmitted with length of the packet and length in the Length/Type field mismatch. tx_statvec_o[23] is used to implement this counter.
0x068	TX_STAT_LNGTH_ERR_1	RO	
0x06C	TX_STAT_LNG_PKT_0	RO	Transmit Long packet Statistics Counter. Counts the total number of packets transmitted with length of the packet longer than the max_frm_size. tx_statvec_o[25] is used to implement this counter.
0x070	TX_STAT_LNG_PKT_1	RO	
0x074	TX_STAT_MULTCST_0	RO	Transmit Multicast Packet Statistics Counter. Counts the total number of multicast packets transmitted. tx_statvec_o[20] is used to implement this counter.
0x078	TX_STAT_MULTCST_1	RO	
0x07C	TX_STAT_BRDCST_0	RO	Transmit Broadcast Packet Statistics Counter. Counts the total number of broadcast packets transmitted. tx_statvec_o[19] is used to implement this counter.
0x080	TX_STAT_BRDCST_1	RO	

Offset	Register Name	Access	Description
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 length 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 length 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 length between 256 and 511.
0x0C8	TX_STAT_PKT_256_511_1	RO	
0x0CC	TX_STAT_PKT_512_1023_0	RO	Transmit Packet 512-1023 Statistics Counter. Counts the total number of packets transmitted with length between 512 and 1023.
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 length between 1024 and 1518.
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 1518.
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 length between 1024 and 2047.
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 length between 2048 and 4095.
0x0F8	TX_STAT_PKT_2048_4095_1	RO	
0x0FC	TX_STAT_PKT_4096_9216_0	RO	

Offset	Register Name	Access	Description
0x100	TX_STAT_PKT_4096_9216_1	RO	Transmit Packet 4096-9216 Statistics Counter. Counts the total number of packets transmitted with length between 4096 and 9216.
0x104	TX_STAT_PKT_9217_16383_0	RO	Transmit Packet 9217-16383 Statistics Counter. Counts the total number of packets transmitted with length between 9217 and 16383.
0x108	TX_STAT_PKT_9217_16383_1	RO	
0x10C	RX_STAT_PKT_LNGTH_0	RO	Receive Packet Length Statistics Counter. Indicated the length of the packet received. rx_statvec_o[13:0] is used to implement this counter.
0x110	RX_STAT_PKT_LNGTH_1	RO	
0x114	RX_STAT_VLN_TG_0	RO	Receive VLAN Tag Statistics Counter. Counts the total number of tagged packets received. rx_statvec_o[18] is used to implement this counter.
0x118	RX_STAT_VLN_TG_1	RO	
0x11C	RX_STAT_PAUSE_0	RO	Receive Pause Packet Statistics Counter. Counts the total number of pause packets received. rx_statvec_o[19] is used to implement this counter.
0x120	RX_STAT_PAUSE_1	RO	
0x124	RX_STAT_FLT_0	RO	Receive Filtered Packet Statistics Counter. rx_statvec_o[22] is used to implement this counter.
0x128	RX_STAT_FLT_1	RO	
0x12C	RX_STAT_UNSP_OPCODE_0	RO	Receive Unsupported Opcode Statistics Counter. Counts the number of packets received with unsupported Opcode. rx_statvec_o[23] is used to implement this counter.
0x130	RX_STAT_UNSP_OPCODE_1	RO	
0x134	RX_STAT_BRDCST_0	RO	Receive Broadcast Packet Statistics Counter. Counts the number of packets received that were directed to the broadcast address. This does not include multicast packets. rx_statvec_o[15] is used to implement this counter.
0x138	RX_STAT_BRDCST_1	RO	
0x13C	RX_STAT_MULTCST_0	RO	Receive Multicast Packet Statistics Counter. Counts the number of packets received that were directed to the 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	

Offset	Register Name	Access	Description
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 a 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 1518 octets and has either a 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 imple-ment this counter.
0x180	RX_STAT_PKT_64_1	RO	
0x184	RX_STAT_PKT_65_127_0	RO	Receive Packet 65-127 Statistics Counter. Counts the number of packets received that were between 65-127 octets in length (including bad packets).
0x188	RX_STAT_PKT_65_127_1	RO	
0x18C	RX_STAT_PKT_128_255_0	RO	Receive Packet 128-255 Statistics Counter. Counts the number of packets received that were between 128-255 octets in length (including bad packets).
0x190	RX_STAT_PKT_128_255_1	RO	
0x194	RX_STAT_PKT_256_511_0	RO	Receive Packet 256-511 Statistics Counter. Counts the number of packets received that were between 256-511 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 512-1023 octets in length (including bad packets).
0x1A0	RX_STAT_PKT_512_1023_1	RO	
0x1A4	RX_STAT_PKT_1024_1518_0	RO	Receive Packet 1024-1518 Statistics Counter. Counts the number of packets received that were between 1024- 1518 octets in length (including bad packets).
0x1A8	RX_STAT_PKT_1024_1518_1	RO	
0x1AC	RX_STAT_PKT_UNDERSIZE_0	RO	Receive Packet Undersize Statistics Counter. Counts the number of packets received that were less than 64 octets long and were otherwise well formed.
0x1B0	RX_STAT_PKT_UNDERSIZE_1	RO	
0x1B4	RX_STAT_PKT_UNICAST_0	RO	Receive Packet Unicast Statistics Counter. Counts the number of good packets received that were directed to a single address.
0x1B8	RX_STAT_PKT_UNICAST_1	RO	
0x1BC	RX_STAT_PKT_RCVD_0	RO	Packets Received Statistics Counter.

Offset	Register Name	Access	Description
0x1C0	RX_STAT_PKT_RCVD_1	RO	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.
0x1C4	RX_STAT_PKT_64_GOOD_CRC_0	RO	Receive Packet 64 With Good CRC Statistics Counter. Counts the number of packets received with length less than 64 and with a good crc. rx_statvec_o[13:6] and rx_statvec_o[17] are used to implement this counter.
0x1C8	RX_STAT_PKT_64_GOOD_CRC_1	RO	
0x1CC	RX_STAT_PKT_1518_GOOD_CRC_0	RO	Receive Packet 1518 With Good CRC Statistics Counter. Counts the number of packets received with length more than 1518 and with a good crc. rx_statvec_o[13:0] and rx_statvec_o[17] are used to implement this counter.
0x1D0	RX_STAT_PKT_1518_GOOD_CRC_1	RO	
0x1D4	RX_STAT_PKT_1519_2047_0	RO	Receive Packet 1519-2047 Statistics Counter. Counts the number of packets received that were between 1519- 2047 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 2048- 4095 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 4096- 9216 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 9217- 16383 octets in length (including bad packets).
0x1F0	RX_STAT_PKT_9217_16383_1	RO	

2.6. Ethernet Data Format

The format of an untagged Ethernet frame is shown in Figure 2.2. The format of a tagged Ethernet frame is shown in Figure 2.3.

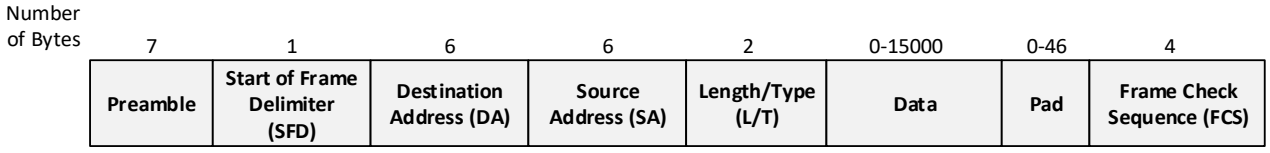

Figure 2.2. Untagged Ethernet Frame Format

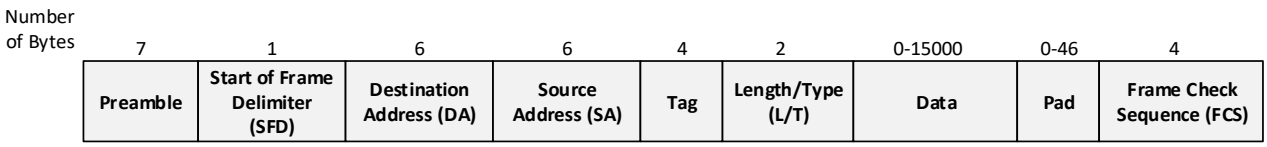

Figure 2.3. Tagged Ethernet Frame Format

The MAC is responsible for constructing a valid frame from the data received from the client before transmitting it. On the receive 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 DA, SA, L/T and Data. The TX MAC adds the Preamble and SFD before transmitting the frame. This mode can be set by enabling tx_pass_fcs bit in the TX_CTL register. See In-Band FCS Passing for timing details.
- 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. See Default Normal Frame for timing details.
- 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. See Custom Preamble Passing for the timing details.

On the receive 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.

2.7. Receive MAC

The Receive MAC is responsible for receiving the incoming frames and transferring them to the Client via AXI4-Stream interface. In the process, it performs the following operations:

- Checks the frame for a valid 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 Destination Address (DA).
- Determines the type of the frame by analyzing the Length/Type field (L/T).
- 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 start of frame (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 address filtering rules as described below:

- 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 was set to one. It is discarded if the bit selected was 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 was set. A broadcast frame is discarded if none of these bits are set.

2.8. Transmit MAC

The Transmit MAC is responsible for controlling access to the physical medium. Its main functions are:

- 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 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, it is your responsibility to 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 Start of Frame Data (SFD) bits and the CRC check sum to the end of the frame (FCS). 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 Deficit Idle Count 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. Refer to IEEE 802.3-2012 Section 46 for detailed discussion of Deficit Idle Count.

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. Since 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.

Two different methods are used 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 PAUSE_TMR register.

2.9. Receive AXI4-Stream Interface

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

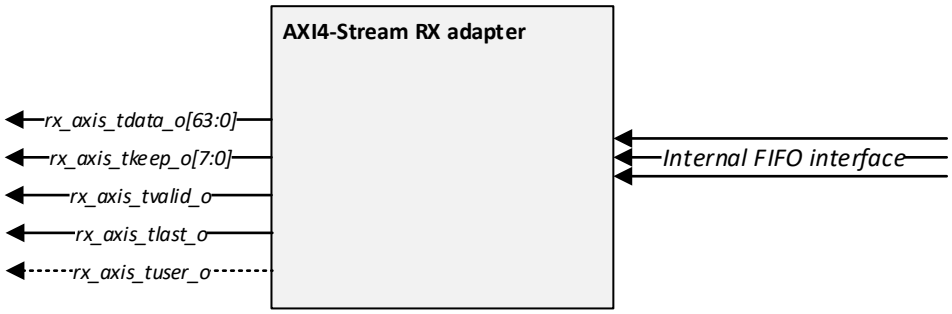

Figure 2.4. AXI4-Stream RX Adapter Interface Diagram

2.9.1. Default Normal Frame

Figure 2.5 shows the timing diagram of a default normal frame at the Receive AXI4-Stream interface:

- T0: MAC asserts rx_axis_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 rx_tlast_o to indicate last packet transfer.
- T2: MAC deasserts rx_axis_tvalid_o to indicate end of active packet.

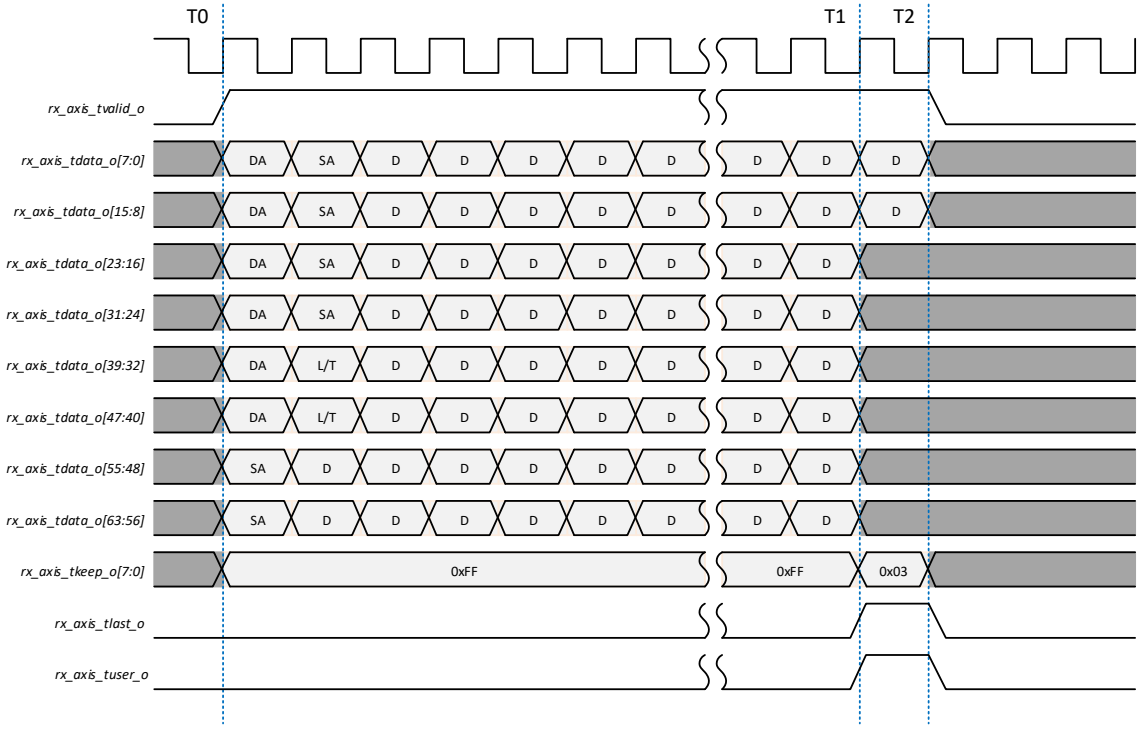

Figure 2.5. Normal Frame Reception

2.9.2. In-Band FCS Passing

Figure 2.6 shows the timing diagram of a frame with in-band FCS. The FCS field is included inside the valid frame.

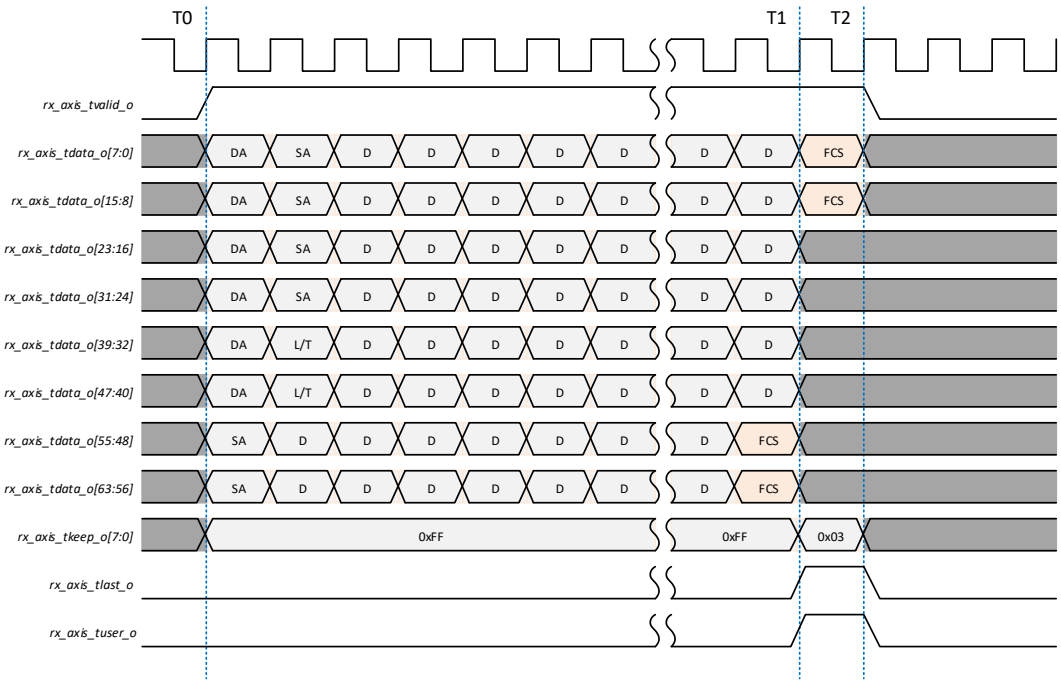

Figure 2.6. Frame Reception with In-Band FCS Passing

2.9.3. Custom Preamble Passing

Figure 2.7 shows the timing diagram of a frame with custom preamble mode enabled. A custom preamble (CP) field is included inside the valid frame.

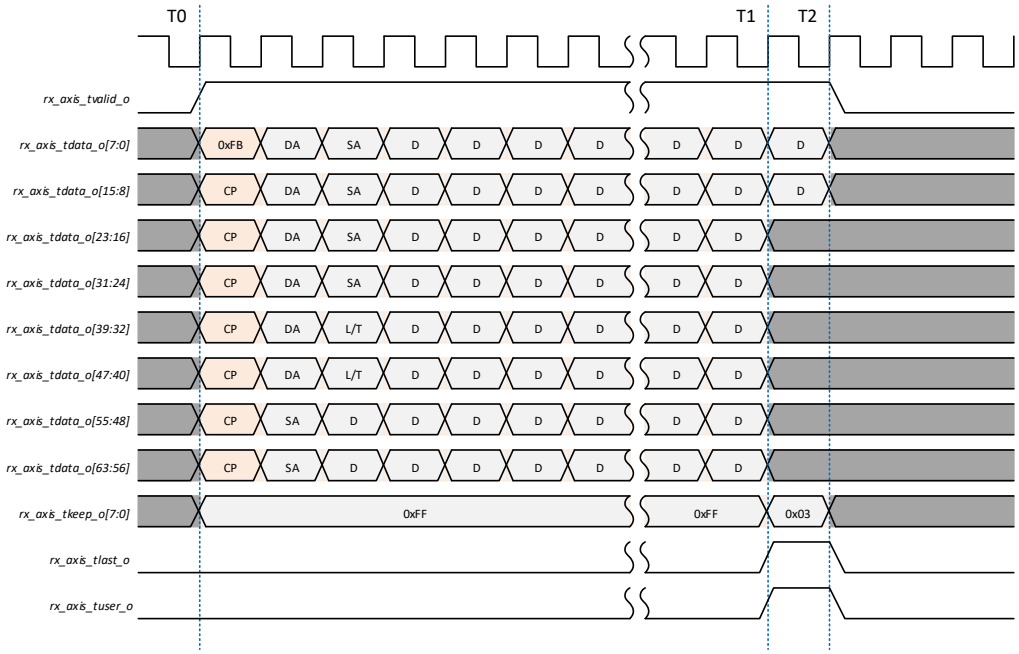

Figure 2.7. Reception with Custom Preamble

2.10. Transmit AXI4-Stream Interface

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

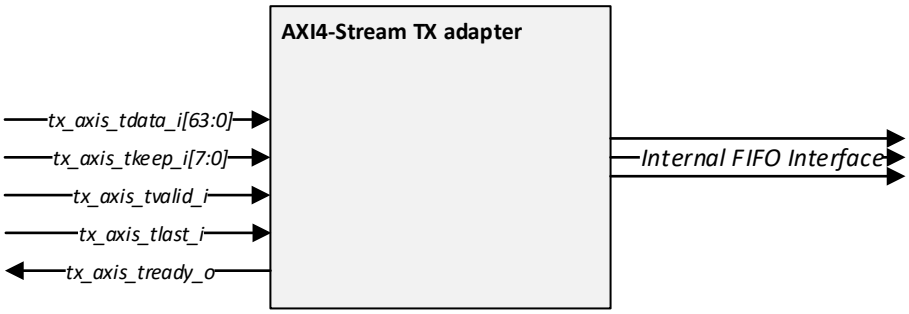

Figure 2.8. AXI4-Stream TX Adapter Interface Diagram

2.10.1. Default Normal Frame

Figure 2.9 shows the timing diagram of a default normal frame at the Transmit AXI4-Stream interface:

- T0: Client asserts tx_axis_tvalid_i to indicate start of packet transfer. It must hold the data and tx_axis_tvalid_i until tx_axis_tready_o asserts.
- T1: Client continues to send data once it sees tx_axis_tready_o asserted.
- T2: Client asserts tx_axis_tlast_i to indicate last packet transfer.
- T3: Client deasserts tx_axis_tvalid_i to indicate end of active frame. Transmit MAC also deasserts tx_axis_tready_o once it sees the last packet transfer (tx_axis_tlast_i = 1).

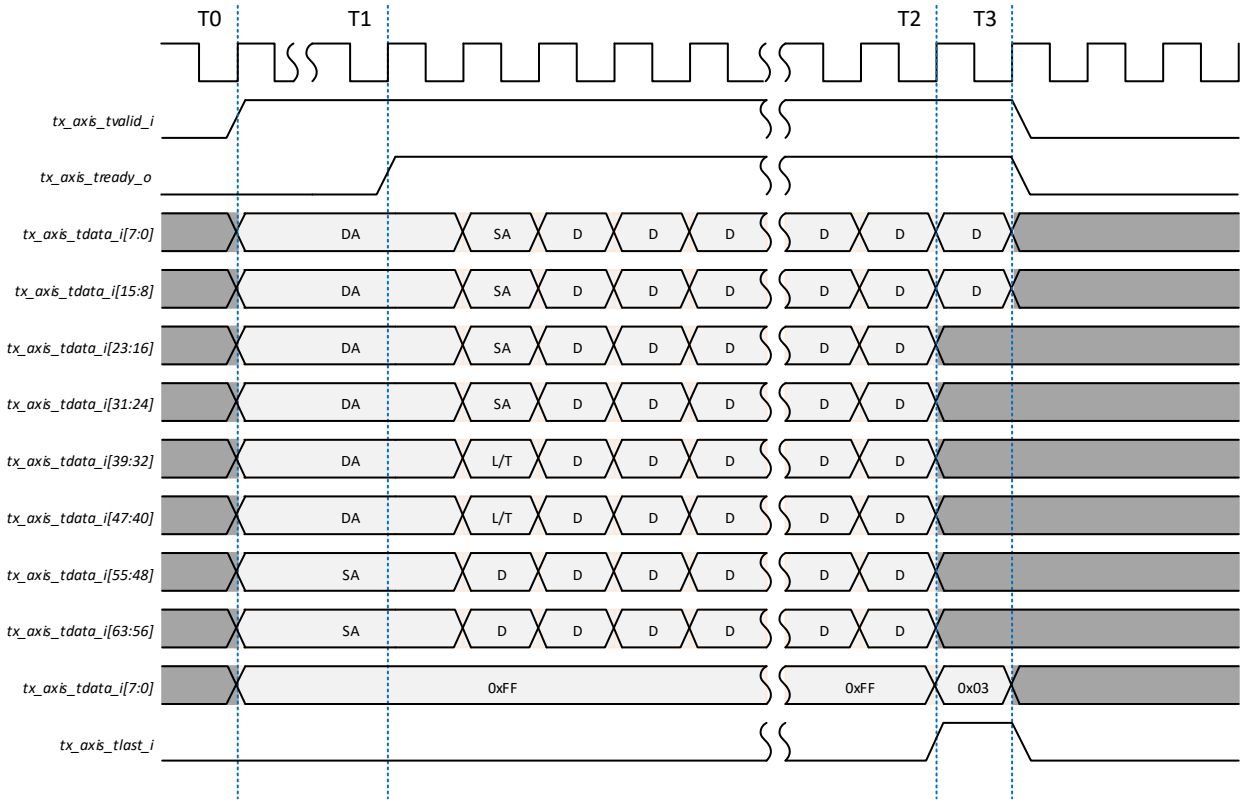

Figure 2.9. Default Normal Frame Transmission

2.10.2. In-Band FCS Passing

Figure 2.10 shows the timing diagram of a frame when In-Band FCS passing is enabled.

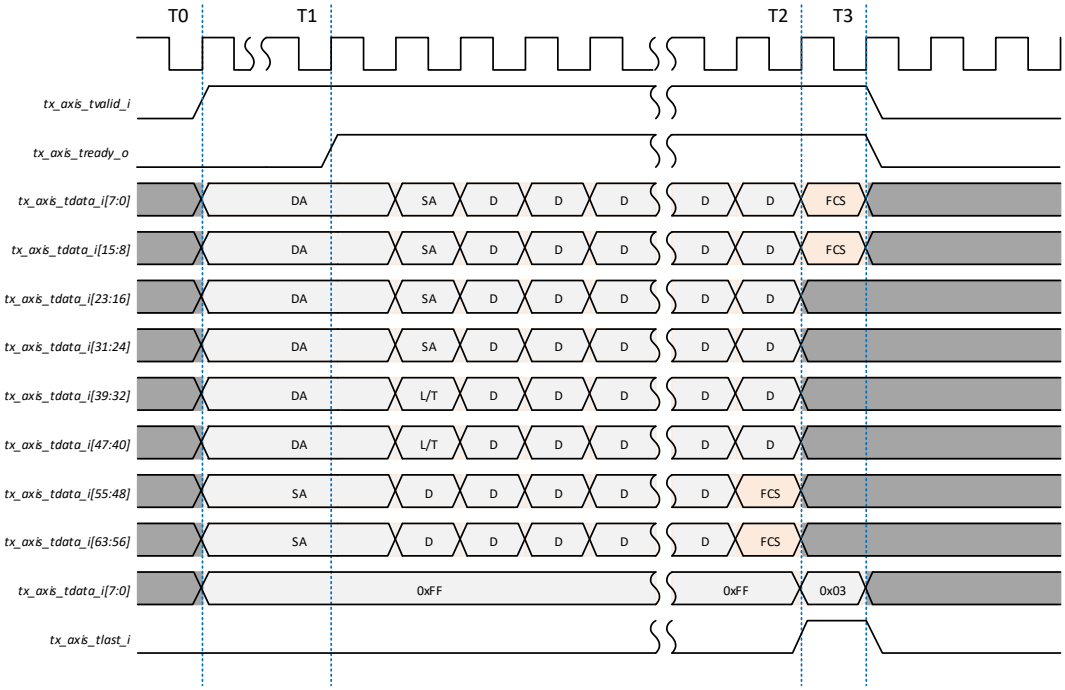

Figure 2.10. Transmission with In-Band FCS Passing

2.10.3. Custom Preamble Passing

Figure 2.11 shows the timing diagram of a frame when custom preamble passing is enabled.

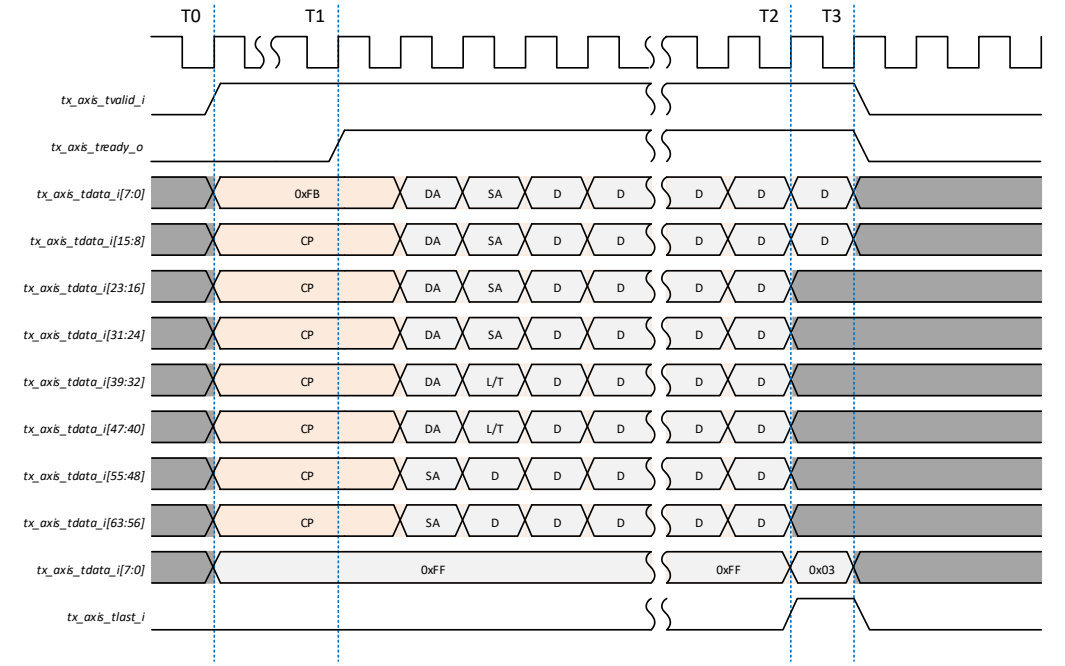

Figure 2.11. Transmission with Custom Preamble Passing

2.11. Data Packing

This module converts between XGMII, GMII and MII interfaces. This is enabled when *Dynamic Speed Selection* attribute is enabled or if *PHY interface* is set to any of the following: 8-bit GMII, 16-bit GMII or MII.

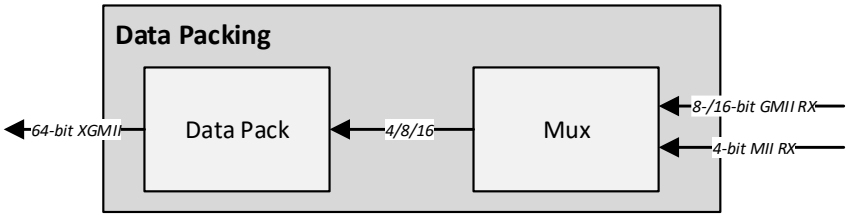

Figure 2.12. Receive Physical Interface for 8/6-bit GMII or MII

The receive side takes the 8-bit/16-bit GMII or 4-bit MII and transforms it into XGMII packets. The transformation process involves detection of the assertion of the RX data valid signal (gmii_rx_dv_i or gmii_16_rx_dv_i or mii_rx_dv_i). It then replaces the first preamble symbol (0x55) to a start character (0xFB), pack the 4/8/16 bit data to a 64-bit data and store the reconstructed 64-bit data to a FIFO before it is sent out to the MAC layer. The module also inserts an end-of-frame character (0xFD) when the data valid signal has deasserted.

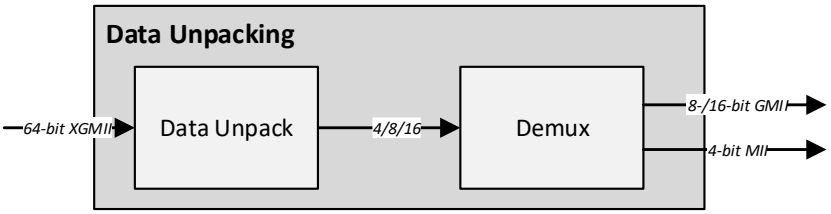

Figure 2.13. Transmit Physical Interface for 8/16-bit GMII or MII

The transmit side transfers the 64-bit XGMII data to 8-bit/16-bit GMII or 4-bit MII data. When the module detects the start character (0xFB), it asserts the TX data valid signal (gmii_tx_en_o or gmii_16_tx_en_o or mii_tx_en_o) and replaces the start character (0xFB) to a preamble symbol (0x55). It then unpacks and stores the reconstructed 4/8/16-bit data to a FIFO before it is sent out the PHY layer. When it detects the end-of-frame character (0xFD), the module deasserts gmii_tx_en_o or gmii_16_tx_en_o or mii_tx_en_o signal and remove 0xFD character before transmitting the data.

The IFG (inter-frame gap) is longer than expected for 8-bit/16-bit GMII or 4-bit MII mode since we are packing/unpacking from 64-bit data to 4/8/16-bit data and vice versa.

2.12. Management

The 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 *Statistics Counter Register* attribute is enabled.

Refer to AMBA 3 APB Protocol version 1.0 Specification for the timing details of this protocol.

2.13. Reset and Sequence

2.13.1. Reset

An asynchronous reset pin (active low) as system reset is used for resetting the 10 Gb Ethernet MAC 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.

2.13.2. Sequence

The following is 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. Send settings through the APB interface to configure the system. Refer to Registers Description for list of registers.
3. Perform desired operation.

Figure 2.14 shows a sample sequence when RX MAC is configured to pass the FCS field to the Client.

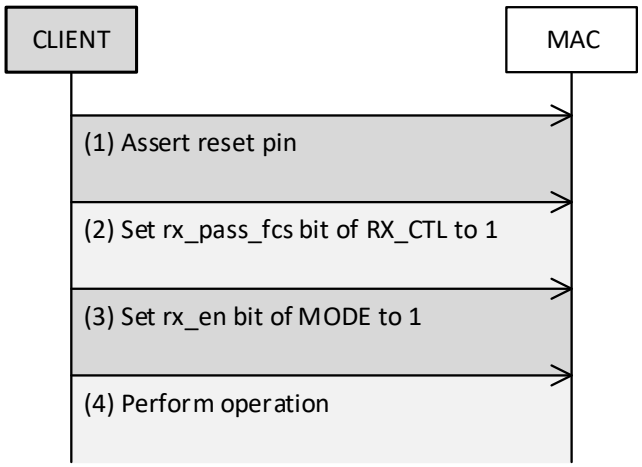

Figure 2.14. Sequence to Configure RX MAC In-Band FCS Passing

3. Core Generation and Simulation

This section provides information on how to generate and synthesize the 10 Gb Ethernet MAC IP Core using the Lattice Radiant software, as well as on how to run simulation. For more details on the Lattice Radiant software, please refer to the Lattice Radiant software user guide and relevant Lattice tutorials.

3.1. Generation and Synthesis

The Lattice Radiant software allows you to customize and generate modules and IPs and integrate them into the device's architecture.

To generate the 10 Gb Ethernet MAC IP Core in Lattice Radiant Software:

1. In the **IP Catalog** tab, double-click on **10 Gb Ethernet MAC** under **IP, Connectivity** category. The **Module/IP Block Wizard** opens as shown in Figure 3.1. Enter values in the **Instance name** and the **Create in** fields and click **Next**.

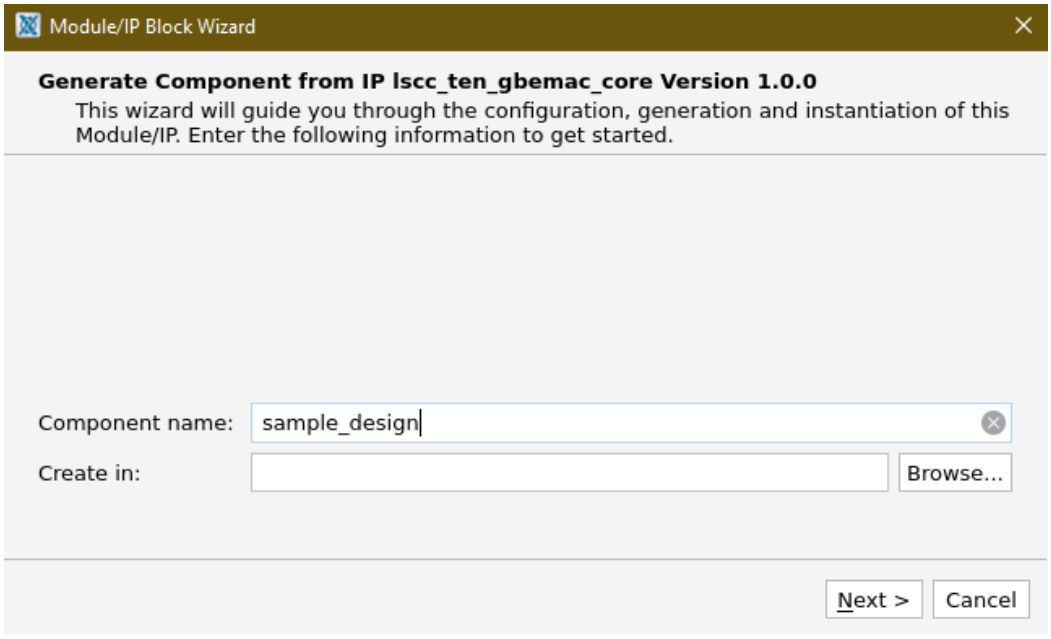

Figure 3.1. Module/IP Block Wizard

2. In the module's dialog box of the **Module/IP Block Wizard** window, customize the selected XGMII IP Core. As a sample configuration, see Figure 3.2. For configuration options, see the Attribute Summary section.

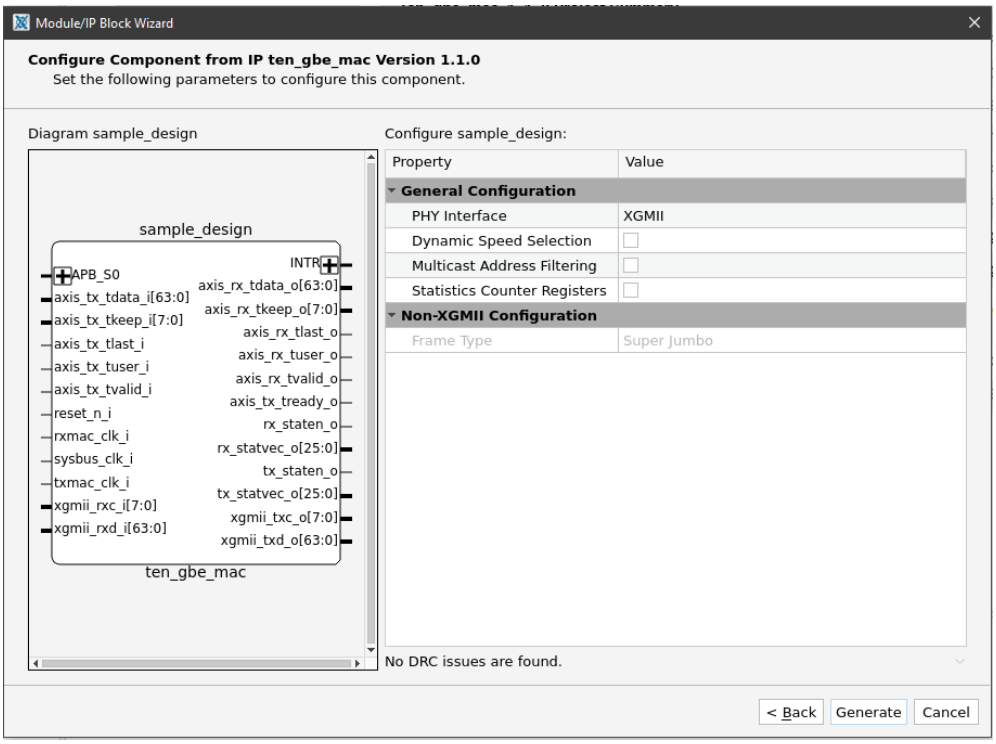

Figure 3.2. Configure User Interface of 10 Gb Ethernet MAC IP Core

3. Click **Generate**. The **Check Generating Result** dialog box opens, showing design block messages and results as shown in Figure 3.3.

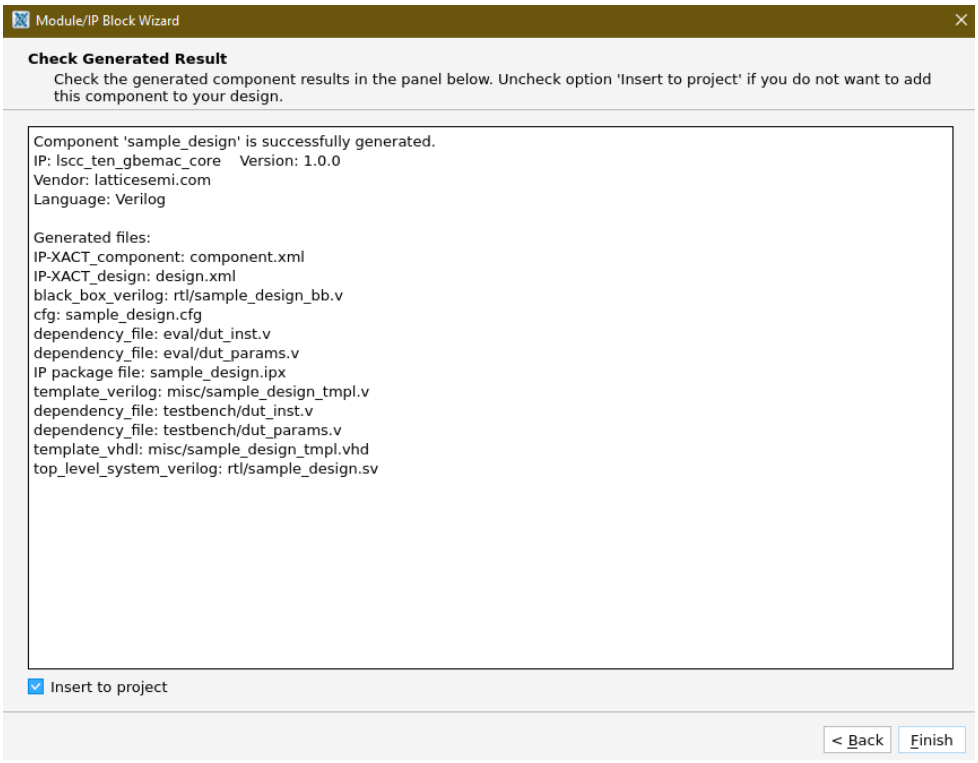

Figure 3.3. Check Generating Result

4. Click the **Finish** button. All the generated files are placed under the directory paths in the **Create in** and the **Instance name** fields shown in Figure 3-1.
5. You can synthesize your generated design by clicking Synthesize Design located in the top left corner of the screen, as shown in Figure 3-4.

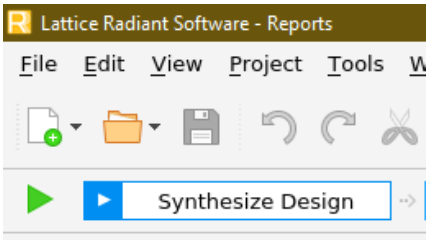

Figure 3.4. Synthesizing Design

The generated 10 Gb Ethernet MAC IP Core package includes the black 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 example RTL 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 top-level reference as the starting template for the top-level for their complete design. The generated files are listed in Table 3.1.

Table 3.1. Generated File List

Generated Files	Description
<Instance Name>.ipx	This file contains the information on the files associated to the generated IP.
<Instance Name>.cfg	This file contains the parameter values used in IP configuration.
component.xml	Contains the ipxact:component information of the IP.
design.xml	Documents the configuration parameters of the IP in IP-XACT 2014 format.
rtl/<Instance Name>.v	This file provides an example RTL top file that instantiates the IP core.
rtl/<Instance Name>_bb.v	This file provides the synthesis black box.
misc/<Instance Name>_tmpl.v misc /<Instance Name>_tmpl.vhd	These files provide instance templates for the IP core.
eval/ ten_gb_subsys.do	.do file to execute sub-system (Generated <Instance Name>.v with 10 Gb PCS IP sample design) simulation
testbench/tb_top.v	Top testbench to run loopback test of generated <Instance Name>.v

3.2. Running Functional Simulation

Running functional simulation can be performed after the IP is generated.

To run Verilog simulation:

1. Click the  button located on the **Toolbar** to initiate the **Simulation Wizard** shown in Figure 3.5.

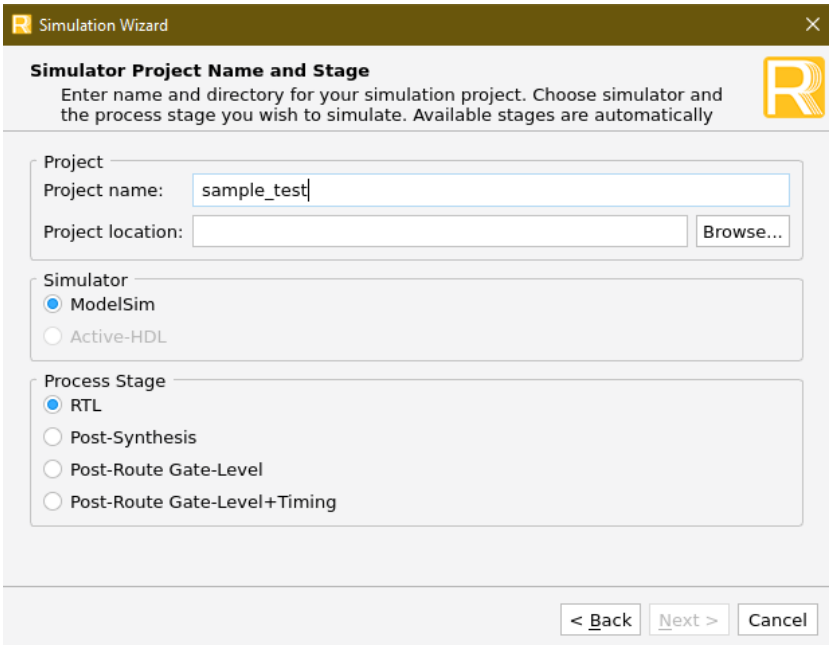

Figure 3.5. Simulation Wizard

2. Click **Next** to open the **Add and Reorder Source** window as shown in Figure 3.6.

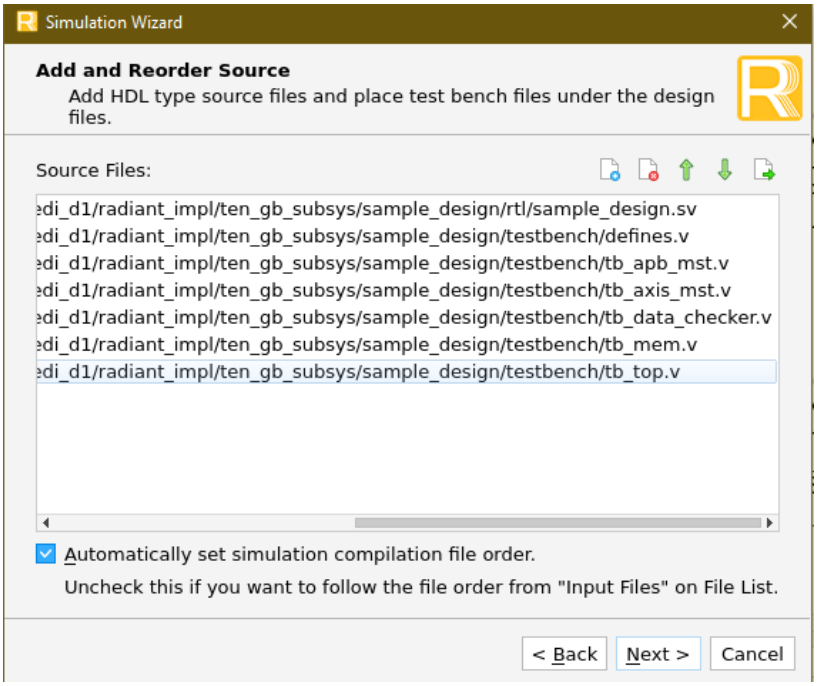

Figure 3.6. Adding and Reordering Source

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

4. Licensing and Evaluation

4.1. Licensing the IP

An IP core-specific license string is required to enable full use of the 10 Gb Ethernet MAC IP Core in a complete, top-level design. The IP Core can be fully evaluated through functional simulation and implementation (synthesis, map, place and route) without an IP license string. This IP core supports Lattice's IP hardware evaluation capability, which makes it possible to create versions of the IP core, which operate in hardware for a limited time (approximately four hours) without requiring an IP license string. See Hardware Evaluation section for further details. However, a license string is required to enable timing simulation and to generate bitstream file that does not include the hardware evaluation timeout limitation.

4.2. Hardware Evaluation

The 10 Gb Ethernet MAC IP Core supports Lattice's IP hardware evaluation capability when used with LFCPNX devices. This makes it possible to create versions of the IP core that operate in hardware for a limited period of time (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/disabled in the Strategy dialog box. It is enabled by default. To change this setting, go to Project > Active Strategy > LSE/Synplify Pro Settings.

5. Ordering Part Number

The Ordering Part Number (OPN) for this IP Core are the following:

- ETHER-10G-CPNX-U – 10G Ethernet MAC for CertusPro-NX – Single Design License
- ETHER-10G-CPNX-UT – 10G Ethernet MAC for CertusPro-NX – Site License

Appendix A. Resource Utilization

Table A.1. Resource Utilization

Configuration	Registers	LUTs	EBRs	Target Device	Synthesis Tools
Multicast Address Filtering == Enabled	3246 (4%)	4713 (6%)	4 (1%)	LFCPNX-100	Synplify Pro
Multicast Address Filtering == Enabled Statistics Counter Registers == Enabled	6612 (8%)	10491 (13%)	4 (1%)	LFCPNX-100	Synplify Pro

Appendix B. Code Listing for Multicast Bit Selection Hash Algorithm in C Language

The code listed below is to aid software developers in programming the multicast tables when the core is programmed to receive multicast frames.

When a software developer wishes to accept a specific multicast address, they should follow the hash algorithm illustrated in the C language code listing to determine which filter bit in the multicast registers to set. The C algorithm returns the multicast table register index as well as the bit within the register that needs to be set based on a given multicast destination address input to the algorithm. Several bits can be set to accept several multicast addresses. If all 64 multicast filter register bits are set to 1, then all received multicast addresses are passed to the MAC client interface.

```
#include <stdio.h>
#include <stdlib.h>

//Hexadecimal equivalent of the CRC equation
#define CRC_POLYNOMIAL 0x04c11db7

int main(int argc, unsigned char *argv[])
{
//The Multicast address is held in a 6 byte array
unsigned char multi_addr[6];
//variables
unsigned long int crc;
unsigned int val;
int i, j, bit;
int carry;
int register_no, register_bit;
crc = 0xffffffff;

//check number of arguments
if (argc != 7) {
printf("Invoke eth_crc with arguments specifying a MAC Address.\n");
printf("Use hex format and blanks.\n");
printf("Example:\n");
printf("eth_crc 01 A2 B3 C4 D5 E6\n");
system("PAUSE");
return 1;
}

printf("\n DA  ");
//Input data from command line
for (j=0; j <6; j++)
{
      sscanf(argv[j+1], "%x", &val);
multi_addr[j] = (unsigned char) val;
printf("%s", argv[j+1]);
      }
      printf("\n");

      //check for multicast destination address
      if ((multi_addr[0] & 0x01) == 0)
      {
            printf(" Not a multicast address\n");
```

```
            printf(" Bit 0 of MSB must be 1\n\n");
            system("PAUSE");
            return 1;
      }

      //The following loops create the 32-bit crc value.
//loop through each byte of the address.
for(i=0; i<6; i++)
{
      //Loop through each byte bit of that byte.
      for(bit=0; bit<8; bit++)
      {
            carry = (crc >> 31)^((multi_addr[i] & (1 << bit)) >> bit);
            crc <<= 1;
            if (carry)
            crc = (crc ^ CRC_POLYNOMIAL) | carry;
      }
}

//Extract the middle 6 bits from the LSB of the 32bit CRC value,
//this six bit value is used to index a unique filter bit.
printf(" crc %lx \n", crc);
crc >>= 1; // refers to Bit 6..1
crc &= 0x3F; // mask six bit

//Find the multicast register number and bit of that register to set.
printf(" hash %lx \n", crc);
register_no = crc >> 3;
register_bit = crc & 7;
printf (" register_no %lx\n register_bit %lx \n\n", register_no, register_bit);
system("PAUSE");
return 0;
}
```

References

For complete information on Lattice Radiant Project-Based Environment, Design Flow, Implementation Flow and Tasks, as well as on the Simulation Flow, see the Lattice Radiant software user guide.

Technical Support Assistance

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

Revision History

Document Revision 1.1, Lattice Radiant SW Version 3.0, November 2021

Section	Change Summary
All	Minor adjustments in formatting across the document.
Introduction	• Updated Design Tool Support in Table 1.1. • Added new bullet for 8-bit/16-bit GMII and corrected a typo in Features.
Functional Description	• Updated Figure 2.1. • Updated Clock and Reset description, updated Statistics Vector Interface default value, added GMII, MII Interface and Miscellaneous group, updated sysbus_clk_i, axis_rx_tuser_o, axis_tx_tuser_i, tx_statvec_o and rx_statsvec_o description, and added table notes in Table 2.1. • Updated Table 2.2 to add new row for General Configuration. • Updated TX_STAT_CNT_0/1 and TX_STAT_PAUSE_0/1 description in Table 2.23. • Added Statistics Counters and Data Packing section. • Updated Default Normal Frame content including Figure 2.9. • Updated Figure 2.10 and Figure 2.11.
Core Generation and Simulation	Updated Figure 3.2.
Appendix A. Resource Utilization	Updated registers and LUTs in Multicast Address Filtering Enabled and added Statistics Counter Registers Enabled in Table A.1.

Document Revision 1.0, Lattice Radiant SW Version 3.0, June 2021

Section	Change Summary
All	Initial release.

www.latticesemi.com