Triple Speed Ethernet MAC Core User Guide

UG-CORE-TSEMAC-v4.9
December 2023
www.efinixinc.com

Contents

Introduction

The Triple Speed Ethernet MAC core is a configurable core that complies with the IEEE Std. 802.3-2008 specification. The core supports 1000 Mbps, 100 Mbps, and 10 Mbps Ethernet speed with full duplex transfer mode.

Use the IP Manager to select IP, customize it, and generate files. The Triple Speed Ethernet MAC core has an interactive wizard to help you set parameters. The wizard also has options to create a testbench and/or example design targeting an Efinix® development board.

Features

- Complies to the IEEE Std. 802.3-2008 specification
- Supports 1000 Mbps, 100 Mbps, and 10 Mbps with full duplex transfer mode
- Management data I/O (MDIO) for PHY device management
- Supports VLAN frame and jumbo frame
- Includes statistic counter, address filtering, TX interpacket gap (IPG), and pause frame flow control
- Supports internal FIFO and error-correction code (ECC) protection
- Programmable source and destination MAC addresses
- RGMII, RMII, GMII, and MII PHY interfaces
- 8-bit, 16-bit, and 32-bit AXI4-Stream user interface for packet transfer
- AXI4-Lite user interface for MAC register setting
- Includes an example design targeting:
  - Trion® T120 BGA324 Development Board
  - Titanium Ti60 F225 Development Board

Device Support

Table 1: Triple Speed Ethernet MAC Core Device Support

FPGA Family	Supported Device
Trion	All
Titanium	All

Resource Utilization and Performance

Note: The resources and performance values provided are based on some of the supported FPGAs. These values are just guidance and change depending on the device resource utilization, design congestion, and user design.

Table 2: Trion Resource Utilization and Performance

FPGA	Logic Utilization (LUTs)		Memory Blocks		F_{MAX} (MHz)		Efinity® Version[1]
	FIFO On	FIFO Off	FIFO On	FIFO Off	FIFO On	FIFO Off	
T13 BGA256 C4	2,508	2,404	12	3	126	130	2019.3
T20 BGA256 C4	2,508	2,404	12	3	126	130	
T85 BGA324 C4	2,508	2,404	12	3	128	134	
T120 BGA324 C4	2,508	2,404	12	3	128	134	

Table 3: Titanium Resource Utilization and Performance

FPGA	Logic Elements (Logic, Adders, Flipflops, etc.)	Memory Blocks	DSP Blocks	Efinity® Version[1]
Ti60 F225 C4	4,201/60800 (6.9 %)	6/256 (2.34 %)	0/160 (0%)	2023.2

Release Notes

You can refer to the IP Core Release Notes for more information about the IP core changes. The IP Core Release Notes is available in the Efinity Downloads page under each Efinity software release version.

Note: You must be logged in to the Support Portal to view the IP Core Release Notes.

[1] Using Verilog HDL.

Functional Description

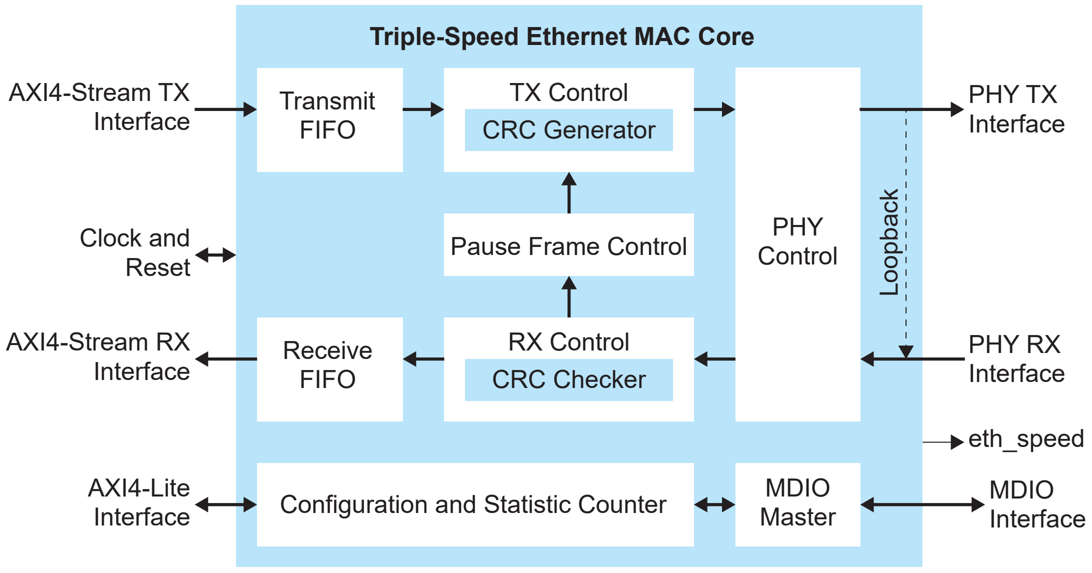
The Triple Speed Ethernet MAC core outputs data sent through the AXI4-Stream TX interface to the PHY TX interface in double data-rate form. The Triple Speed Ethernet MAC core outputs the double data-rate data sent through the PHY RX to the AXI4-Stream RX interface.

The Triple Speed Ethernet MAC core includes an AXI4-Lite interface to configure and read out the configuration registers and statistic counter registers. Refer to the Table 29: Example Design Configuration Registers on page 36 for more details.

The MDIO is a bidirectional signal that transmits control information and status between the PHY and the Triple Speed Ethernet MAC core. The Triple Speed Ethernet MAC core sends MDC to the PHY as a timing reference for transmitting information on the MDIO signal.

The Triple Speed Ethernet MAC core has an internal loop-back function at the PHY interface. When the internal loop-back is enabled, the PHY TX data is looped back to the PHY RX data and the Triple Speed Ethernet MAC core outputs the data through the AXI4-Stream RX interface.

Figure 1: Triple Speed Ethernet MAC System Block Diagram



Ports

Table 4: AXI4-Stream TX Interface Ports

Name	Direction	Description
tx_axis_clk	Input	AXI4-Stream interface TX clock.
tx_axis_mac_tdata [(AXI_WIDTH)-1:0]	Input	AXI4-Stream TX data.
tx_axis_mac_tvalid	Input	AXI4-Stream TX data valid strobe.
tx_axis_mac_tlast	Input	AXI4-Stream TX last data indicator.
tx_axis_mac_tuser	Input	Assertion indicates error in the TX packet. The MAC sends the error to the PHY.
tx_axis_mac_tready	Output	AXI4-Stream TX ready signal.
tx_axis_mac_tstrb [(AXI_WIDTH/8)-1:0]	Input	AXI4-Stream TX data strobe.

Table 5: AXI4-Stream RX Interface Ports

Name	Direction	Description
rx_axis_clk	Input	AXI4-Stream interface RX clock.
rx_axis_mac_tdata [(AXI_WIDTH)-1:0]	Output	AXI4-Stream RX data.
rx_axis_mac_tvalid	Output	AXI4-Stream RX data valid strobe.
rx_axis_mac_tlast	Output	AXI4-Stream RX last data indicator.
rx_axis_mac_tuser	Output	The core asserts this signal with rx_axis_mac_tlast to indicate one of the following error conditions: • CRC error • Error received from PHY • Filtered packet thru address filtering • Oversize/undersize packet • RXFIFO overflow
rx_axis_mac_tready	Input	AXI4-Stream RX ready signal.
rx_axis_mac_tstrb [(AXI_WIDTH/8)-1:0]	Output	AXI4-Stream RX data strobe.

Table 6: AXI4-Lite Interface Ports

Name	Direction	Description
s_axi_aclk	Input	AXI4-Lite interface clock.
s_axi_awaddr [9:0]	Input	AXI4-Lite write address bus.
s_axi_awvalid	Input	AXI4-Lite write address valid strobe.
s_axi_awready	Output	AXI4-Lite write address ready signal.
s_axi_wdata [31:0]	Input	AXI4-Lite write data.
s_axi_wvalid	Input	AXI4-Lite write data valid strobe.
s_axi_wready	Output	AXI4-Lite write ready signal.
s_axi_bresp [1:0]	Output	AXI4-Lite write response.
s_axi_bvalid	Output	AXI4-Lite write response valid strobe.
s_axi_bready	Input	AXI4-Lite write response ready signal.
s_axi_araddr [9:0]	Input	AXI4-Lite read address bus.
s_axi_arvalid	Input	AXI4-Lite read address valid strobe.
s_axi_arready	Output	AXI4-Lite read address ready signal.
s_axi_rdata [31:0]	Output	AXI4-Lite read data.
s_axi_rresp [1:0]	Output	AXI4-Lite read response.
s_axi_rvalid	Output	AXI4-Lite read data valid strobe.
s_axi_rready	Input	AXI4-Lite read data ready signal.

Table 7: RGMII Interface Ports

Name	Direction	Description
rgmii_txd_HI [3:0]	Output	4 most significant bit (MSB) of transmit data to the PHY. In the Interface Designer, if the Double Data I/O option is turned on for rgmii_txd bus, connect the rgmii_txd_HI to Pin Name (HI). Set the clock pin name reference to tx_mac_aclk.
rgmii_txd_LO [3:0]	Output	4 least significant bit (LSB) of transmit data to the PHY. In the Interface Designer, if the Double Data I/O option is turned on for rgmii_txd bus, connect the rgmii_txd_LO to Pin Name (LO). Set the clock pin name reference to tx_mac_aclk.
rgmii_tx_ctl_HI	Output	MSB of transmit control signal to the PHY. In the Interface Designer, if Double Data I/O option is turned on for rgmii_tx_ctl bus, connect the rgmii_tx_ctl_HI to Pin Name (HI). Set the clock pin name reference to tx_mac_aclk.
rgmii_tx_ctl_LO	Output	LSB of transmit control signal to the PHY. In the Interface Designer, if the Double Data I/O option is turned on for rgmii_tx_ctl bus, connect the rgmii_tx_ctl_LO to Pin Name (LO). Set the clock pin name reference to tx_mac_aclk.
rgmii_txc_HI	Output	MSB of transmit clock signal to the PHY. In the Interface Designer, if the Double Data I/O option is turned on for rgmii_txc bus, connect the rgmii_txc_HI to Pin Name (HI). Set the clock pin name reference to 90 degree phase shift of tx_mac_aclk.
rgmii_txc_LO	Output	LSB of transmit clock signal to the PHY. In the Interface Designer, if the Double Data I/O option is turned on for rgmii_txc bus, connect the rgmii_txc_HI to Pin Name (HI). Set the clock pin name reference to 90 degree phase shift of tx_mac_aclk.
rgmii_rxd_HI [3:0]	Input	4 MSB of receive data from the PHY. In the Interface Designer, if the Double Data I/O option is turned on for rgmii_rxd bus, connect the rgmii_rxd_HI to Pin Name (HI). Set the clock pin name reference to rgmii_rxc.
rgmii_rxd_LO [3:0]	Input	4 LSB of receive data from the PHY. In the Interface Designer, if the Double Data I/O option is turned on for rgmii_rxd bus, connect the rgmii_rxd_LO to Pin Name (LO). Set the clock pin name reference to rgmii_rxc.
rgmii_rx_ctl_HI	Input	MSB of receive control signal from the PHY. In the Interface Designer, if the Double Data I/O option is turned on for rgmii_rx_ctl bus, connect the rgmii_rx_ctl_HI to Pin Name (HI). Set the clock pin name reference to rgmii_rxc.(2)
rgmii_rx_ctl_LO	Input	LSB of receive control signal from the PHY. In the Interface Designer, if the Double Data I/O option is turned on for rgmii_rx_ctl bus, connect the rgmii_rx_ctl_LO to Pin Name (LO). Set the clock pin name reference to rgmii_rxc.
rgmii_rxc	Input	RX clock from the PHY.

(2) The Efinity software displays a warnings for rgmii_rx_ctl_HI during compilation when using 10/100 Mbps Ethernet. This warning is related to the Rx packet error feature that is not supported and you can safely ignore them.

Table 8: GMII Interface Ports

The core only supports 1000 Mb transfer in GMII mode. The transmit clock is tx_mac_aclk and runs at 125 MHz.

Name	Direction	Description
gm_tx_d [7:0]	Output	Transmit data to the PHY.
gm_tx_en	Output	Assertion indicated that the data on the transmit data bus is valid.
gm_tx_err	Output	Assertion indicates the PHY that the frame sent is invalid.
gm_rx_d [7:0]	Input	Receive data from the PHY.
gm_rx_dv	Input	Assert this signal to indicate that the data on the receive data bus is valid. Keep this signal asserted during frame reception, from the first preamble byte until the last byte of the CRC field is received.
gm_rx_err	Input	Assert this signal to indicate that the receive frame contains errors.
gm_rx_c	Input	RX clock from the PHY. 125 MHz for 1000 Mb date rate.
gm_tx_c	Output	TX clock to PHY. 125 MHz for 1000 Mb date rate.

Table 9: RMII Interface Ports

Name	Direction	Description
rmii_txd [1:0]	Output	Transmit data to the PHY.
rmii_tx_en	Output	Assertion indicated that the data on the transmit data bus is valid.
rmii_clk_ref	Input	Continuous 50 MHz RMII clock from the PHY.
rmii_rxd [1:0]	Input	Receive data from the PHY.
rmii_crs_dv	Input	Valid receive data strobe.
rmii_rx_err	Input	Assert this signal to indicate that the receive frame contains errors.

Table 10: MII Interface Ports

Name	Direction	Description
mii_txd [3:0]	Output	Transmit data to the PHY.
mii_tx_dv	Output	Transmit enable.
mii_tx_err	Output	Assertion indicates the PHY that the frame sent is invalid.
mii_txc	Input	TX clock from the PHY. 25 MHz for 100 Mb data rate. 2.5 MHz for 10 Mb data rate.
mii_rxc	Input	RX clock from the PHY. 25 MHz for 100 Mb data rate. 2.5 MHz for 10 Mb data rate.
mii_rxd [3:0]	Input	Receive data from the PHY.
mii_rx_dv	Input	Assert this signal indicate that the data on `mii_rxd[3:0]` is valid. The signal stays asserted during frame reception, from the first preamble byte until the last byte of the CRC field is received.
mii_rx_err	Input	Assert this signal to indicate that the receive frame contains errors.

Table 11: Conduit Ports

Name	Direction	Description
eth_speed [2:0]	Output	Ethernet MAC transfer speed. 3'b100: 1000 Mbps 3'b010: 100 Mbps 3'b001: 10 Mbps

Table 12: MDIO Interface Ports

Name	Direction	Description
Mdi	Input	Input data signal for communication with the PHY's configuration and status registers. Always set to high when not in use.
Mdo	Output	Output data signal for communication with the PHY's configuration and status registers.
MdoEn	Output	Output data enable signal for MDIO bidirectional bus. In the Interface Designer, you have to create an INOUT tristate port, set Mdi as input, Mdo as output and MdoEn as output enable.
Mdc	Output	MDIO Management Clock. Derived from s_axi_aclk on the basis of supplied configuration data.

Table 13: Clock and Reset Ports

Name	Direction	Description
mac_reset	Input	Active high system global reset.
proto_reset	Input	Active high protocol reset on TX and RX path.
tx_mac_aclk	Input	TX path transfer clock signal at 125 MHz. Provides timing reference for RGMII/GMII TX path interface.
rx_mac_aclk	Output	RX path transfer clock signal. Provides timing reference for RGMII/GMII/RMII/MII RX path interface. 125 MHz for 1 Gbps 25 MHz for 100 Mbps 2.5 MHz for 10 Mbps

Registers

Table 14: MAC Configuration Registers

Dword Offset	Name	R/W	Description
0x00	VERSION	R	Indicates the version of the IP core. Default: As per VERSION parameter
0x08	Command_Config	R/W	Command configuration settings. Refer to **Table 15: Command_Config Register Field Descriptions** on page 12 for more details. Default: 0x40003
0x0C	mac_addr [31:0]	R/W	Source MAC address.
0x10	mac_addr [47:32]	R/W	Example: If the MAC address is set to 01-1B-43-17-7B-CD: Write 0x43177BCD to register 0x0C Write 0x0000011B to register 0x10 Default: 0x0
0x14	frm_length [15:0]	R/W	Indicates the maximum packet length. Default: 0x5ee
0x18	pause_quant [15:0]	R/W	Pause quanta value. A pause frame includes the period of pause time being requested. This pause time is measured in units of pause quanta, where each unit is equal to 512 bit times. Example: If pause_quant is set to 8, the pause time is 8 x 512 bit times. Default: 0x0
0x5c	tx_ipg_length [5:0]	R/W	Transmit interpacket gap value. Default: 0xC

Table 15: Command_Config Register Field Descriptions

Bit	Name	R/W	Description
0	tx_ena	R/W	Enables transmit path. Default: 1
1	rx_ena	R/W	Enables receive path. Default: 1
2	xon_gen	R/W	Pause frame generation. Assert high to trigger a pause frame with 0 pause quanta. Default: 0
3	Reserved		
4	promis_en	R/W	Assert high to disable MAC address filtering. Default: 0
5	Reserved		
6	crc_fwd	R/W	Assert high to forward the CRC value to the AXI4 Stream RX data rx_axis_mac_tdata. Default: 0
7	Reserved		
8	pause_ignore	R/W	Assert high to disable pause frame control. Default: 0
9	tx_addr_ins	R/W	Assert high to insert the transmit packet's source MAC address with the value in register 0xC and 0x10. Default: 0
10-14	Reserved		
15	rgmii_loop_ena	R/W	Assert high to loop back the RGMII TX path to RGMII RX path. Default: 0
18-16	eth_speed [2:0]	R/W	Ethernet MAC transfer speed. 3'b100: 1000 Mbps 3'b010: 100 Mbps 3'b001: 10 Mbps Default: 3'b100
19-21	Reserved		
22	xoff_gen	R/W	Pause frame generator with pause quanta. Assert high to trigger a pause frame which the pause quanta field of the pause packet is set according to pause quanta value in register 0x18. Default: 0
23-30	Reserved		
31	cnt_reset	R/W	Assert high to reset the statistic counter registers value. Default: 0

Table 16: MDIO Configuration Registers

Dword Offset	Name	R/W	Description
0x100	Bit 7:0 – Divider	R/W	To derive the Mdc clock frequency. The Mdc frequency = s_axi_aclk frequency/Divider. Default: 0x64
	Bit 8 - NoPre		0: Send the Mdio with preamble data. 1: Send the Mdio without preamble data. Default: 0x0
0x104	Bit 0 - RdEn	System Controlled	Assert high to read the Mdio bus. Default: 0x0
	Bit 1 - WrEn		Assert high to write to the Mdio bus. Default: 0x0
0x108	Bit [4:0] – RegAddr	R/W	MDIO configuration register address. Default: 0x0
	Bit [7:5] – Reserved		–
	Bit [12:8] - PhyAddr		5-bit PHY address. Default: 0x0
0x10c	WrData [15:0]	R/W	MDIO write data. Default: 0x0
0x110	RdData [15:0]	R	MDIO read data. Default: 0x0
0x114	Bit 0 - LinkFailStatus	R	Fail to link with the PHY MDIO Default: 0x0
	Bit 1 - BusyStatus		PHY MDIO busy Default: 0x0
	Bit 2 - NvalidStatus		Invalid Status (qualifier for the valid scan result) Default: 0x0

Table 17: Receive Supplementary Registers

Dword Offset	Name	R/W	Description
0x140	broadcast_filter_en	R/W	Assert high to filter out the broadcast address receive packet. Default: 0x0
0x144	mac_addr_mask [31:0]	R/W	MAC address mask bits. Set the address mask bits to turn on the address filtering. When promis_en = 0, Triple Speed Ethernet MAC core compares the destination MAC address data bytes of the received packet with the MAC address sets in register 0xC and 0x10. The receive packet is filtered if the address does not match. Default: 0x0
0x148	mac_addr_mask [47:32]	R/W	

Table 18: Transmit Supplementary Registers

Dword Offset	Name	R/W	Description
0x180	tx_dst_addr_ins	R/W	Assert high to insert the destination MAC address value in 0x184 and 0x188 to TX packet destination MAC address data bytes. Default: 0x0
0x184	dst_mac_addr [31:0]	R/W	Destination MAC address. Default: 0x0
0x188	dst_mac_addr [47:32]	R/W	

Table 19: Statistic Counter Registers

Dword Offset	Name	R/W	Description
0x68	aFramesTransmittedOK [15:0]	R	Indicates the number of frames successfully transmitted, including the pause frames. Default: 0x0
0x6c	aFramesReceivedOK [15:0]	R	Indicates the number of frames successfully received, including the pause frames. Frames with error are not included. Default: 0x0
0x70	aFrameCheckSequenceErrors [15:0]	R	Indicates the number of received frames with CRC error. Default: 0x0
0x80	aTxPAUSEMACCtrlFrames [15:0]	R	Indicates the number of pause frame transmitted. Default: 0x0
0x84	aRxPAUSEMACCtrlFrames [15:0]	R	Indicates the number of pause frame received. Default: 0x0
0x88	ifInErrors [15:0]	R	Indicates the number of frames received with either one of the following errors: • CRC error • Error received from PHY • Filtered packet thru address filtering • Oversize/undersize packet • RXFIFO overflow Default: 0x0
0x8c	ifOutErrors [15:0]	R	Indicates the number of frames transmitted with either one of the following errors: • Incorrect packet length • Error defined by the user application thru tx_axis_mac_tuser signal • TXFIFO overflow • TX FIFO underflow Default: 0x0
0x9c	aRxFilterFramesErrors [15:0]	R	Indicates the number of filtered packets. Default: 0x0

Dword Offset	Name	R/W	Description
0xb4	etherStatsPkts [15:0]	R	Indicates the total number of frames received, including frames with errors. Default: 0x0
0xb8	etherStatsUndersizePkts [15:0]	R	The number of frames received with a length less than 64 bytes, including pause frames. This count does not include frames with errors. Default: 0x0
0xbc	etherStatsOversizePkts [15:0]	R	The number of frames received that are longer than the value configured in the frm_length register. This count does not include frames with errors. Default: 0x0
0xc0	aFramesReceivedLen[15:0]	R	Indicates the number of frames length. Default: 0x0
0xC4	aTxFifoOverflowFramesErrors[15:0]	R	Indicates the number of transmitted frames with TXFIFO overflow error. Default: 0x0
0xC8	aTxIncontinuityFramesErrors[15:0]	R	Indicates the number of transmitted frames with incontinuity error. Default: 0x0

IP Manager

The Efinity® IP Manager is an interactive wizard that helps you customize and generate Efinix® IP cores. The IP Manager performs validation checks on the parameters you set to ensure that your selections are valid. When you generate the IP core, you can optionally generate an example design targeting an Efinix development board and/or a testbench. This wizard is helpful in situations in which you use several IP cores, multiple instances of an IP core with different parameters, or the same IP core for different projects.

Note: Not all Efinix IP cores include an example design or a testbench.

Generating the Triple Speed Ethernet MAC Core with the IP Manager

The following steps explain how to customize an IP core with the IP Configuration wizard.

1. Open the IP Catalog.
2. Choose **Ethernet** > **Triple Speed Ethernet MAC** core and click **Next**. The **IP Configuration** wizard opens.
3. Enter the module name in the **Module Name** box.

   Note: You cannot generate the core without a module name.

4. Customize the IP core using the options shown in the wizard. For detailed information on the options, refer to the *Customizing the Triple Speed Ethernet MAC* section.
5. (Optional) In the **Deliverables** tab, specify whether to generate an IP core example design targeting an Efinix® development board and/or testbench. These options are turned on by default.
6. (Optional) In the **Summary** tab, review your selections.
7. Click **Generate** to generate the IP core and other selected deliverables.
8. In the **Review configuration generation** dialog box, click **Generate**. The Console in the **Summary** tab shows the generation status.

   Note: You can disable the **Review configuration generation** dialog box by turning off the **Show Confirmation Box** option in the wizard.

9. When generation finishes, the wizard displays the **Generation Success** dialog box. Click **OK** to close the wizard.

The wizard adds the IP to your project and displays it under **IP** in the Project pane.

Generated Files

The IP Manager generates these files and directories:

- **<module name>_define.vh**—Contains the customized parameters.
- **<module name>_tmpl.v**—Verilog HDL instantiation template.
- **<module name>_tmpl.vhd**—VHDL instantiation template.
- **<module name>.v**—IP source code.
- **settings.json**—Configuration file.
- **<kit name>_devkit**—Has generated RTL, example design, and Efinity® project targeting a specific development board.
- **Testbench**—Contains generated RTL and testbench files.

Customizing the Triple Speed Ethernet MAC

The core has parameters so you can customize its function. You set the parameters in the General tab of the core's IP Configuration window.

Table 20: Triple Speed Ethernet MAC Core Parameters

Name	Options	Description
TSE MAC Version	Any	Triple Speed Ethernet MAC version. Default: 16
TX FIFO(3)	Enable, Disable	Enables the TX FIFO in the core. Must be enabled if the tx_axis_clk clock frequency is not the same as the PHY clock frequency. The FIFO stores the AXI data to avoid overflow/underflow occurring in the core. Default: Enable
RX FIFO(3)	Enable, Disable	Enables the RX FIFO in the core. Must be enabled if the rx_axis_clk clock frequency is not the same as the PHY clock frequency. The FIFO stores the AXI data to avoid overflow/underflow occurring in the core. Default: Enable
TX FIFO Depth(4)	16 – 16384	TX FIFO depth. The TXFIFO_DTH must be greater than the length of a packet. When sending a jumbo packet with length is greater than the maximum TX FIFO Depth, 16384, you must disable the TXFIFO_EN. Default: 2048
RX FIFO Depth(4)	16 – 16384	RX FIFO depth. Default: 2048
PHY Interface Mode	RGMII, GMII, RMII, MII	Defines PHY interface mode. Default: RGMII
AXI4-Stream Data Width	8, 16, 32	Defines AXI4-Stream data width. Default: 16

(3) TX FIFO and RX FIFO are not supported in I4L speedgrade FPGAs.
(4) Deeper FIFO depth uses more RAM.

Double Data I/O Settings

The Triple Speed Ethernet MAC can run at 10, 100, and 1000 Mbps Ethernet speed. You must have the double data I/O (DDIO) option enabled on certain signals for the Triple Speed Ethernet MAC core to run at 1000 Mbps.

You must enable the DDIO option for the following signals:

- `rgmii_rx_ctl`
- `rgmii_txc`
- `rgmii_tx_ctl`

In the Efinity® Interface Designer, use the Create Block menu to add the signals and set the Block Editor settings as shown in the following table.

Table 21: Block Editor Settings for rgmii_rx_ctl, rgmii_txc and rgmii_tx_ctl

Parameter	**Setting**		
	rgmii_rx_ctl	**rgmii_txc**	**rgmii_tx_ctl**
Instance Name	User defined	User defined	User defined
Mode	input	output	output
Register Option	register	register	register
Double Data I/O Option	resync	resync	resync
Clock Pin Name	rgmii_rxc	-	-
Pin Name (HI)	rgmii_rx_ctl_HI	rgmii_txc_HI	rgmii_tx_ctl_HI
Pin Name (LO)	rgmii_rx_ctl_LO	rgmii_txc_LO	rgmii_tx_ctl_LO
Enable invert clock	No	Yes (Trion) No (Titanium)	No
Output Clock Pin Name	-	clk_125m_90deg	clk_125m

You must enable the DDIO option for the following buses:

- `rgmii_rxd`
- `rgmii_txd`

Use the Create GPIO Bus wizard to add the buses and set the Bus Property settings as shown in the following table.

Table 22: Bus Property Settings for rgmii_rxd and rgmii_txd

Parameter	**Setting**	
	rgmii_rxd	**rgmii_txd**
Mode	Input	Output
Pin Name (HI)	rgmii_rxd_HI	rgmii_txd_HI
Pin Name (LO)	rgmii_rxd_LO	rgmii_txd_LO
Register Option	register	register
Double Data I/O Option	resync	resync
Clock Pin Name	rgmii_rxc	-
Output Clock Pin Name	-	clk_125m
Enable invert clock	No	No

Triple Speed Ethernet MAC Example Design

You can choose to generate the example design when generating the core in the IP Manager Configuration window. Compile the example design project and download the **.hex** or **.bit** file to your board.

Important: Efinix tested the example design generated with the default parameter options only.

Efinix provides a Triple Speed Ethernet MAC (TSEMAC) core example design that targets the Trion® T120 BGA324 Development Board and Titanium Ti60 F225 Development Board. The example design demonstrates the functionality of the Triple Speed Ethernet MAC core running at 1000 Mbps. The example design includes two test modes:

- *Normal mode test*—The MAC/UDP pattern generator constructs and initiates TX packets for the TX path of the Triple Speed Ethernet MAC core.
- *Linked-Partner mode test*—The TX packets are constructed by the other end of the network, for example a computer.

The example design consists of:

- *MAC/UDP pattern generator*—Generates the MAC/UDP TX packet to the Triple Speed Ethernet MAC core thru AXI4-ST TX interface.
- *RISC-V Sapphire SOC*—Allows you to configure the Triple Speed Ethernet MAC core configuration register and example design configuration register. Refer to the Table 29: Example Design Configuration Registers on page 36 for more details.
- *Triple Speed Ethernet MAC core*

Figure 2: Example Design Block Diagram

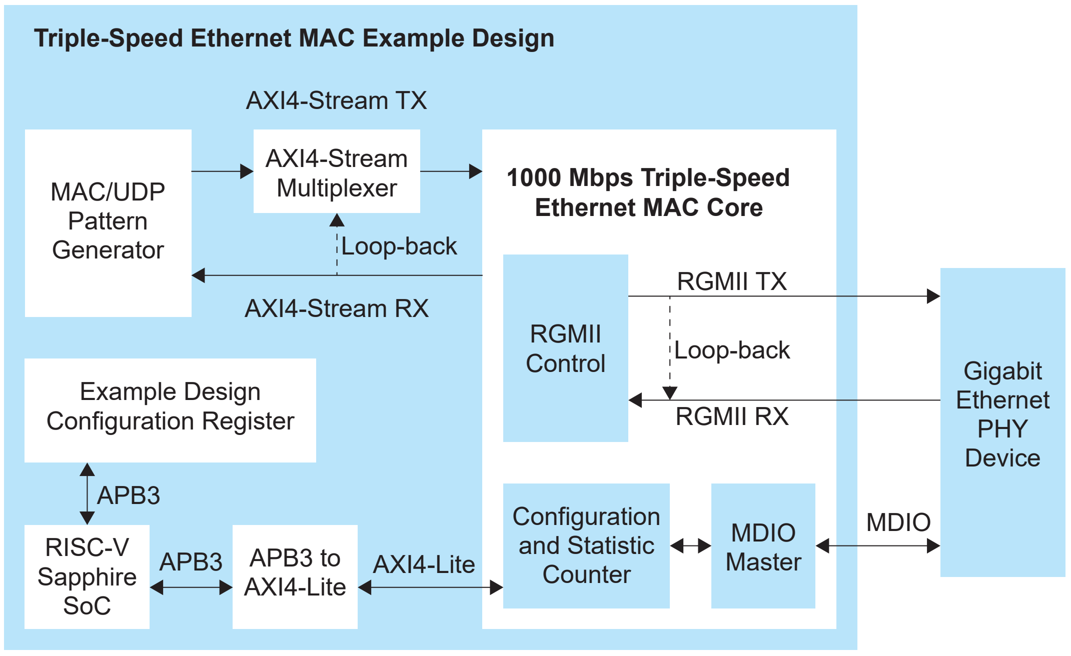

Triple-Speed Ethernet MAC Example Design
AXI4-Stream TX
MAC/UDP Pattern Generator
AXI4-Stream Multiplexer
Loop-back
AXI4-Stream RX
1000 Mbps Triple-Speed Ethernet MAC Core
RGMII TX
RGMII Control
Loop-back
RGMII RX
Gigabit Ethernet PHY Device
Example Design Configuration Register
APB3
RISC-V Sapphire SoC
APB3
APB3 to AXI4-Lite
AXI4-Lite
Configuration and Statistic Counter
MDIO Master
MDIO

Table 23: Trion® T120 BGA324 Development Board Example Design Implementation

FPGA	LUTs	Memory Blocks	f_{MAX} (MHz)[5]			Efinity® Version
			clk	clk_125m	rgmii_rxc	
T120 BGA324 C4	7,536	109	56.598	133.491	138.939	2020.2

Table 24: Titanium Ti60 F225 Development Board Example Design Implementation

FPGA	Logic Elements (Logic, Adders, Flipflops, etc.)	Memory Blocks	DSP Blocks	Efinity® Version
Ti60 F225 C4	12,127/60800 (19.96 %)	60/256 (23.43 %)	4/160 (2.5%)	2022.2

Required Hardware

Trion® T120 BGA324 Development Kit Example

The example design uses the following hardware from the Trion® T120 BGA324 Development Kit:

- Trion® T120 BGA324 Development Board
- Micro-USB cable
- 12 V power adapter

You also need the following hardware, which you provide yourself:

- USB-to-UART converter module
- Ethernet cable
- Computer

Titanium Ti60 F225 Development Board Example

The example design uses the following hardware from the Titanium Ti60 F225 Development Kit:

- Titanium Ti60 F225 Development Board
- USB-C data cable
- 12 V power adapter
- Ethernet Connector Daughter Card

You also need the following hardware, which you provide yourself:

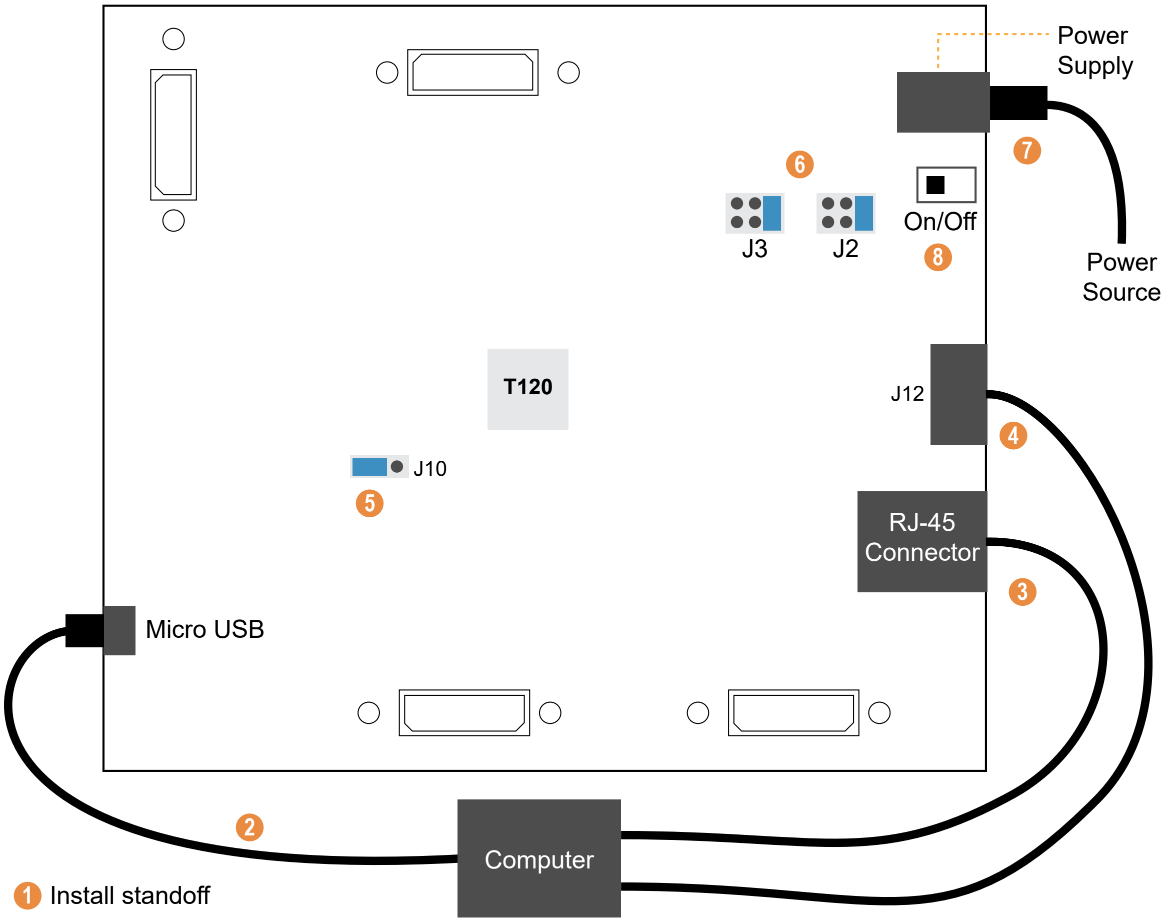
- Ethernet cable
- Computer

[5] Using default parameter settings.

Hardware Set Up for Trion® T120 BGA324 Development Board

The following figure shows the hardware setup steps for Trion® T120 BGA324 Development Board example design. If you have not already done so, attach standoffs to the board.

Figure 3: Trion® T120 BGA324 Development Board Hardware Setup



Important: Make sure that the development board is turned off before connecting any cables.

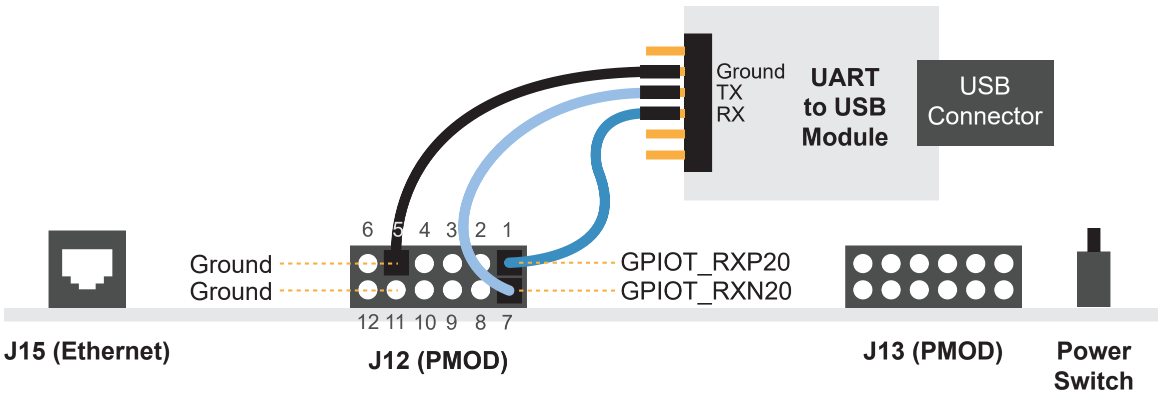
1. Connect a USB cable to the board and to your computer.
2. Connect an Ethernet cable to the RJ-45 receptacle on the board and to your computer.
3. Connect a USB-to-UART module to the J12 header. See Set Up a USB-to-UART Module (Trion) for more details.
4. On header J10, connect pins 2 - 3 with a jumper (default) to use the on-board oscillator.
5. Leave the jumper on J2 at the defaults (connecting pins 1 - 2). On J3, connect pins 1 - 2.
6. Connect the 12 V power supply to the board connector and to a power source.
7. Turn on the board.

Note: Ensure that you install the driver for the correct interface. Refer to the development kit user guide for information about installing drivers.

Set Up a USB-to-UART Module

The Trion® T120 BGA324 Development Board does not have a USB-to-UART converter, therefore, you need to use a separate USB-to-UART converter module. A number of modules are available from various vendors; any USB-to-UART module should work.

Figure 4: Connect the UART Module to I/O Header J12



1. Connect the UART's RX, TX, and ground pins to the Trion® T120 BGA324 Development Board using:
   - *RX*—GPIOT_RXP20, which is labeled as 1 on header J12
   - *TX*—GPIOT_RXN20, which is labeled as 7 on header J12
   - *Ground*—Ground, connect to one of the GND pins on header J12
2. Plug the UART module into a USB port on your computer. The driver should install automatically if needed.

Finding the COM Port (Windows)

1. Type Device Manager in the Windows search box.
2. Expand **Ports (COM & LPT)** to find out which COM port Windows assigned to the UART module; it is listed as USB Serial Port (COM*n*) where *n* is the assigned port number. Note the COM number.

Finding the COM Port (Linux)

In a terminal, type the command:

```
dmesg | grep ttyUSB
```

The terminal displays a series of messages about the attached devices.

```
usb <number>: <adapter> now attached to ttyUSB<number>
```

There are many USB-to-UART converter modules on the market. Some use an FTDI chip which displays a message similar to:

```
usb 3-3: FTDI USB Serial Device converter now attached to ttyUSB0
```

However, the Trion® T120 BGA324 Development Board also has an FTDI chip and gives the same message. So if you have both the UART module and the board attached at the same time, you may receive three messages similar to:

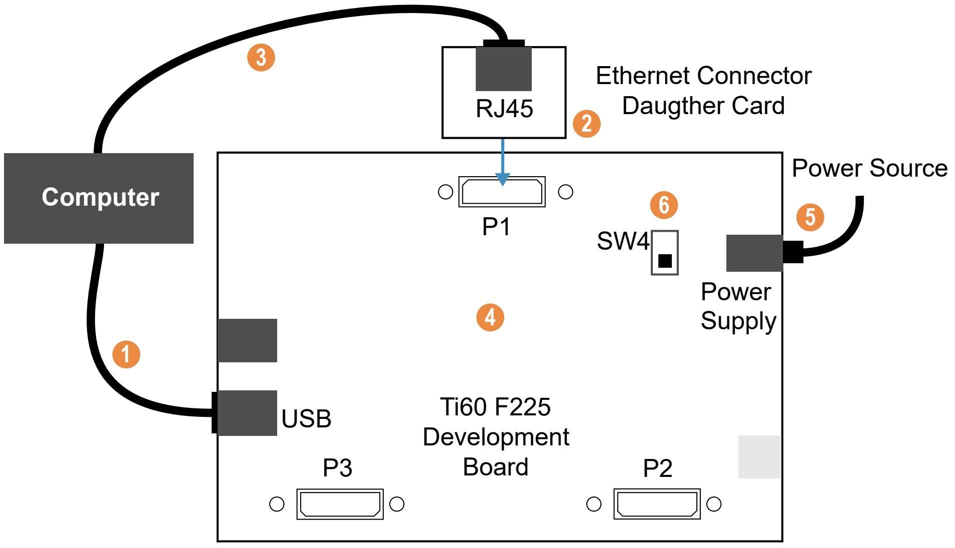
```
usb 3-3: FTDI USB Serial Device converter now attached to ttyUSB0
usb 3-2: FTDI USB Serial Device converter now attached to ttyUSB1
usb 3-2: FTDI USB Serial Device converter now attached to ttyUSB2
```

In this case, the second 2 lines (marked by `usb 3-2`) are the development board and the first line (`usb 3-3`) is the UART module.

Hardware Set Up for Titanium Ti60 F225 Development Board

The following figure shows the hardware setup steps for Titanium Ti60 F225 Development Board example design. If you have not already done so, attach standoffs to the board.

Figure 5: Titanium Ti60 F225 Development Board Hardware Setup



Important: Make sure that the development board is turned off before connecting any cables.

1. Connect a USB-C cable to the board and to your computer.
2. Attach the Ethernet Connector Daughter Card to header P1 of the development board.
3. Connect an Ethernet cable to the RJ-45 receptacle on the Ethernet Connector Daughter Card and to your computer.
4. Leave the board jumper settings to default. Refer to the development kit user guide for information about the default jumper settings.
5. Connect the 12 V power supply to the board connector and to a power source.
6. Turn on the board.

Note: Ensure that you install the driver for the correct interface. Refer to the development kit user guide for information about installing drivers.

Set Up a USB-to-UART Module

The Titanium Ti60 F225 Development Board has a USB-to-UART converter connected to the Ti60's GPIOL_01 and GPIOL_02 pins. The Titanium Ti180 J484 Development Board has a USB-to-UART converter connected to the Ti180's GPIOR_67 and GPIOR_68 pins. To use the UART, simply connect a USB cable to the FTDI USB connector on the targeted development board and to your computer.

Note: The board has an FTDI chip to bridge communication from the USB connector. FTDI interface 2 on Ti60 communicates with the on-board UART. You do not need to install a driver for this interface because when you connect the Titanium Ti60 F225 Development Board to your computer, Windows automatically installs a driver for it.

Finding the COM Port (Windows)

1. Type Device Manager in the Windows search box.
2. Expand **Ports (COM & LPT)** to find out which COM port Windows assigned to the UART module. You should see 2 devices listed as USB Serial Port (COM*n*) where *n* is the assigned port number. Note the COM number for the first device; that is the UART.

Finding the COM Port (Linux)

In a terminal, type the command:

```
ls /dev/ttyUSB*
```

The terminal displays a list of attached devices.

```
/dev/ttyUSB0 /dev/ttyUSB1 /dev/ttyUSB2 /dev/ttyUSB3
```

The UART is `/dev/ttyUSB2`.

Using this Example with the RISC-V SDK

Before working with the software included with this example design, you should already be familiar with using the Sapphire SoC and RISC-V SDK. Specifically, you should know how to:

- Launch Eclipse using the **run_eclipse.bat** file (Windows) or **run_eclipse.sh** file (Linux) scripts.
- Set up global environment variables.
- Create and build a software project.
- Debug using the OpenOCD debugger.
- Open a UART terminal.

Learn more: Refer to the RISC-V SDK chapter of the **Sapphire RISC-V SoC Hardware and Software User Guide** for detailed instructions on how to perform these tasks.

Example Design Double Data I/O Settings

Note: These sections described the DDIO and Interface Designer settings specifically for the example design. For a generic Interface Designer settings, refer to Double Data I/O Settings on page 18.

Because the Trion® T120 BGA324 Development Board has a limited number of available GPIO with capability, the example design does not assign the Triple Speed Ethernet MAC RGMII control signals, `rgmii_tx_ctl` and `rgmii_rx_ctl` to a DDIO block. Instead, the HI and LO TX control signals are OR together, while the HI and LO RX control signal are assigned with the same input control signal. You do not need this logic if you are not using a Trion® T120 BGA324 Development Board or a T120 BGA576 Development Board where the hard DDIO block is available for `rgmii_tx_ctl_HI/rgmii_tx_ctl_LO` and `rgmii_rx_ctl_HI/rgmii_rx_ctl_LO`.

To disable the logic, comment out the following lines in the **tsemac_ex.v** file.

```
assign rgmii_tx_ctl = rgmii_tx_ctl_HI | rgmii_tx_ctl_LO;
assign rgmii_rx_ctl_HI = rgmii_rx_ctl;
assign rgmii_rx_ctl_LO = rgmii_rx_ctl;
```

Note: The project file included in the example design has predefined all necessary Interface Designer settings. You do not have to set the Interface Designer described in this section again. You can refer to the following settings when designing your own Triple Speed Ethernet MAC core.

Interface Designer Settings

You must enable the DDIO option for the following signals:

- `rgmii_txc`

In the Efinity® Interface Designer, use the Create Block menu to add the signals and set the Block Editor settings as shown in the following table.

Table 25: Block Editor Settings for rgmii_txc

Parameter	rgmii_txc
Instance Name	User defined
Mode	output
Register Option	register
Double Data I/O Option	resync
Clock Pin Name	-
Pin Name (HI)	rgmii_txc_HI
Pin Name (LO)	rgmii_txc_LO
Enable invert clock	Yes (Trion) No (Titanium)
Output Clock Pin Name	clk_125m_90deg

You must enable the DDIO option for the following buses:
- `rgmii_rxd`
- `rgmii_txd`

Use the Create GPIO Bus wizard to add the buses and set the Bus Property settings as shown in the following table.

Table 26: Bus Property Settings for rgmii_rxd and rgmii_txd

Parameter	Setting	
	rgmii_rxd	**rgmii_txd**
Mode	Input	Output
Pin Name (HI)	rgmii_rxd_HI	rgmii_txd_HI
Pin Name (LO)	rgmii_rxd_LO	rgmii_txd_LO
Register Option	register	register
Double Data I/O Option	resync	resync
Clock Pin Name	rgmii_rxc	-
Output Clock Pin Name	-	clk_125m
Enable invert clock	No	No

Double Data I/O Waveforms

The following figures show the input and output waveforms when the DDIO option are enabled in the Triple Speed Ethernet MAC core signals.

Figure 6: Output Signals Waveform with DDIO Enabled

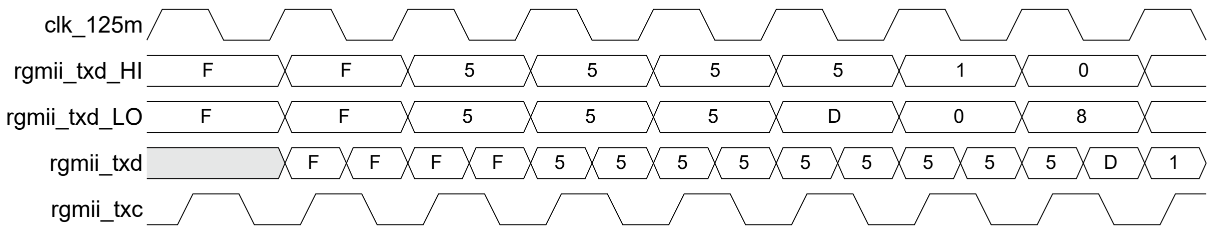
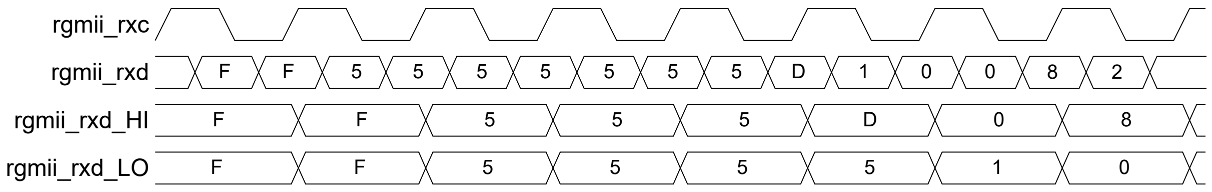

clk_125m
rgmii_txd_HI F F 5 5 5 5 1 0
rgmii_txd_LO F F 5 5 5 D 0 8
rgmii_txd F F F F 5 5 5 5 5 5 5 5 5 D 1
rgmii_txc

Figure 7: Input Signals Waveform with DDIO Enabled


rgmii_rxc
rgmii_rxd F F 5 5 5 5 5 5 5 D 1 0 0 8 2
rgmii_rxd_HI F F 5 5 5 D 0 8
rgmii_rxd_LO F F 5 5 5 5 1 0

Example Design Test Modes

The example design includes software for the Sapphire SoC in the **tseDemo** directory. In addition to the source code, Efinix provides compiled binary files using predefined settings so you can easily try out two test modes:

- *Normal Test Mode*—Use the **tseDemo.elf** file in the **build\test_mode_0** directory.
- *Linked-partner Test Mode*—Use the **tseDemo.elf** file in the **build\test_mode_1** directory.

You can also customize the test modes by changing the variables in the Sapphire SoC software driver code, **embedded_sw\sapphire\bsp\efinix\EfxSapphireSoc\app \tseDemo.h** file. However, you must rebuild the software if you change any variable.

Figure 8: Defining Variables in tseDemo.h File

```
/************************* Project Header File ***************************/
#define PRINTF_EN   1
#define TEST_MODE   1//0:Normal Mode; 1:Linked-partner Test Mode;

#define PAT_NUM      0
#define PAT_DLEN     8
#define PAT_IPG      4095//4095//255
#define PAT_TYPE     0//0:UDP Pattern; //1:MAC Pattern;
#define DST_MAC_H    0xffff
#define DST_MAC_L    0xffffffff
#define SRC_MAC_H    0xeae8
#define SRC_MAC_L    0x5e0060c8
#define SRC_IP       0xc0a80164
#define DST_IP       0xc0a80165
#define SRC_PORT     0x0521
#define DST_PORT     0x2715
```

Realigning the PHY Clock

There are two ways to re-align the PHY clock with respect to the data to allow the core to receive or transmit data correctly:

- Using the PHY RGMII clock-to-data delay in the Sapphire SoC software driver function code.
- Using the core top module `rgmii_rxc_edge` and `rgmii_txc_dly` signals

Using the PHY RGMII Clock-to-Data Delay (Trion Only)

To use the PHY RGMII clock-to-data delay, `TX_delay` and `RX_delay`change the PhyDlySetRXTX() value in the **main.c** function file. The function file is located in the **\embedded_sw\sapphire\software\standalone\tseDemo\src** directory.

Figure 9: Defining Delay in main.c Function File

```
/************************* Function File ***************************/
void main(){
    u32 prompt=0;
    u32 speed;

    PhyNego(1);
    bsp_init();
        bsp_print("tse demo!);

    PhyDlySetRXTX(15, 15); //change PHY delay settings here

    if(TEST_MODE==0){
       speed = PhyNormalInit();
       MacNormalInit(Speed); //MAC speed set, MAC IPG set
    }
    StartRunning();
}
```

Table 27: RX_delay and TX_delay Delay Settings

Delays	Values	Description
RX_delay	0 - 15	Clock to data setup delay for RGMII RX path with 1000 Mbps. Recommended values: 0: Smallest delay of around 0.81 ns 2: Delay of around 1.1 ns 8: Delay of around 2.17 ns 15: Largest delay of around 2.92 ns
TX_delay	0 - 15	Clock to data setup delay for RGMII TX path with 1000 Mbps. Recommended values: 0: Smallest delay of around 0.021 ns 2: Delay of around 0.4 ns 8: Delay of around 1.7 ns 15: Largest delay of around 3.3 ns

Using the RGMII `rgmii_rxc_edge` and `rgmii_txc_dly` Signals

To use the top-module `rgmii_rxc_edge` and `rgmii_txc_dly` signals, set the following signals accordingly.

Top-Module Signal	Description
rgmii_rxc_edge	Select the first rgmii_rxd valid window sampling: `'b0`: Sampled using rising-edge of the rgmii_rxc signal. `'b1`: Sampled using falling-edge of the rgmii_rxc signal. Default: `'b1`
rgmii_txc_dly	Select rgmii_txc rising alignment with the rgmii_txd signal: `'b0`: The rgmii_txc rising edge is edge-aligned with the rgmii_txd signal. `'b1`: The rgmii_txc rising edge is center-aligned with the rgmii_txd signal. Default: `'b1`

Normal Mode Test

The MAC/UDP pattern generator constructs and initiates TX packets for the TX path of the core. The receiving end can be either at other RX (disable RGMII interface loop back) or looped-back at the internal RGMII control block (enable RGMII interface loop back).If the RGMII interface loop back is disabled, the RX path receives the RX packet from the other TX at the other end or from Ethernet PHY device loop-back. The simulation testbench included in this example uses the loop-back method.

Figure 10: Normal Mode Data Flow

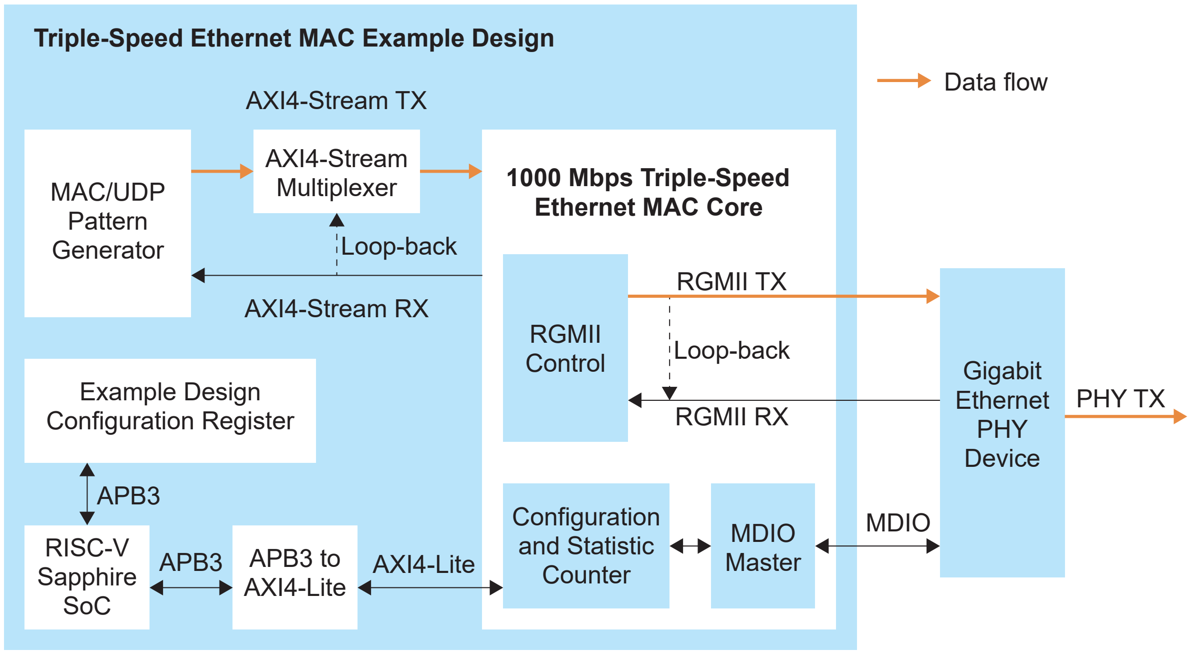

Triple-Speed Ethernet MAC Example Design
Data flow
AXI4-Stream TX
MAC/UDP Pattern Generator
AXI4-Stream Multiplexer
1000 Mbps Triple-Speed Ethernet MAC Core
Loop-back
AXI4-Stream RX
RGMII TX
RGMII Control
Loop-back
RGMII RX
Gigabit Ethernet PHY Device
PHY TX
Example Design Configuration Register
APB3
RISC-V Sapphire SoC
APB3
APB3 to AXI4-Lite
AXI4-Lite
Configuration and Statistic Counter
MDIO Master
MDIO

Table 28: Normal Mode Macros

Macro Name	Description
PAT_NUM	Number of transmit packet. Unlimited packet transmission is denoted by 0.
PAT_DLEN	Length of the UDP/MAC pattern transmit packet.
PAT_IPG	Number of interpacket gap. One interpacket gap is 8 ns.
PAT_TYPE	Type of transmit packet pattern. 1: MAC pattern 0: UDP pattern

The test passes when:

- The value of `PAT_NUM` is 0, the value of `aFramesTransmittedOK` continues to increase until 0xffff. If the value of `PAT_NUM` is non-zero, the value of `aFramesTransmittedOK` and `PAT_NUM` must be the same.
- The value of `aFramesReceivedOK` is the same with the number of frames sent by the opposite end of the network. The `ifInErrors` is 0.
- The captured network packet is consistent with the transmitted packet pattern. See Using Wireshark on page 35 for more information.

Note: You can use any terminal program, such as Putty, termite, or the built-in Eclipse terminal to monitor the `aFramesTransmittedOK`, `aFramesReceivedOK`, and `ifInErrors` statistic counter register values.

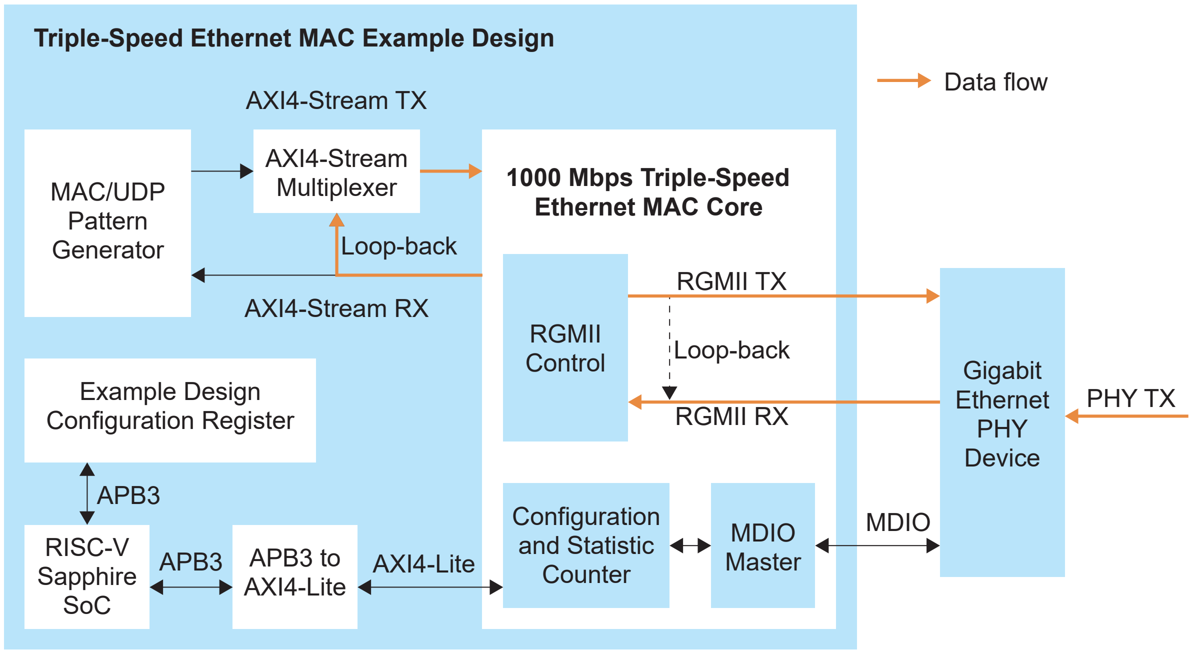
Figure 11: Normal Mode Telnet Session Output Example

```
---EFX-RISCV Command Line Menu Demo---
Wait Ethernet Link up...
Rd Phy Addr 0 : 1140
Rd Phy Addr 2 : 6e
Rd Phy Addr 3 : 3212
Wr Phy Addr 1f : 168
Wr Phy Addr 1e : 8040
Start Info : Set Phy Delay.
Wr Phy Addr 1e : 401e
Rd Phy Addr 1f : 5988
Setup New Value =
 : 0
Wr Phy Addr 1f : 5900
Wr Phy Addr 1e : 801e
Wr Phy Addr 1e : 401e
Rd Phy Addr 1f : 5900
Read New Value =
 : 5900
Rd Phy Addr 11 : ac00
Info : Phy Link up on 1000Mbps.
Info : Set Mac Speed.
Info : Set Mac IPG.
Info : Assert mac reset
Info : Deassert mac reset
Info : Mac Reset Statistics Counters.
Info : Set Pattern Generator.
--------------------
aFramesTransmittedOK : 15c3e
aFramesReceivedOK : 3
ifInErrors : 0
ifOutErrors : 0
etherStatsPkts : 3
etherStatsUndersizePkts : 0
etherStatsOversizePkts : 0
aRxFilterFramesErrors : 0
aFrameCheckSequenceErrors : 0
aTxPAUSEMACCtrlFrames : 0
aRxPAUSEMACCtrlFrames : 0
--------------------
--------------------
aFramesTransmittedOK : 2bb29
aFramesReceivedOK : 5
ifInErrors : 0
ifOutErrors : 0
etherStatsPkts : 5
etherStatsUndersizePkts : 0
etherStatsOversizePkts : 0
aRxFilterFramesErrors : 0
aFrameCheckSequenceErrors : 0
aTxPAUSEMACCtrlFrames : 0
aRxPAUSEMACCtrlFrames : 0
--------------------
```

Linked-Partner Test

The TX packets are constructs by the other end of the network, for example, a computer. The AXI4-ST loop back is enabled in this test mode, and the MAC/UDP pattern generator is not used. The Triple Speed Ethernet MAC core receives the TX packet from the PC and transmits the received packet back to the PC at the AXI4-ST interface.

Figure 12: Linked-Partner Data Flow


Triple-Speed Ethernet MAC Example Design
Data flow
AXI4-Stream TX
MAC/UDP Pattern Generator
AXI4-Stream Multiplexer
Loop-back
AXI4-Stream RX
1000 Mbps Triple-Speed Ethernet MAC Core
RGMII Control
RGMII TX
Loop-back
RGMII RX
Gigabit Ethernet PHY Device
PHY TX
Example Design Configuration Register
APB3
RISC-V Sapphire SoC
APB3
APB3 to AXI4-Lite
AXI4-Lite
Configuration and Statistic Counter
MDIO Master
MDIO

The test passes when:

- The value of `aFramesTransmittedOK` and `aFramesReceivedOK` is consistent.
- The `ifInErrors`is `0`.
- The captured network packet is consistent with the transmitted packet pattern. See Using Wireshark on page 35 for more information.

Note: You can use any terminal program, such as Putty, termite, or the built-in Eclipse terminal to monitor the `aFramesTransmittedOK`, `aFramesReceivedOK`, and `ifInErrors` statistic counter register values.

Figure 13: Linked-Partner Mode Telnet Session Output Example

```
---EFX-RISCV Command Line Menu Demo---
Wait Ethernet Link up...
Rd Phy Addr 0 : 1140
Rd Phy Addr 2 : 6e
Rd Phy Addr 3 : 3212
Wr Phy Addr 1f : 168
Wr Phy Addr 1e : 8040
Start Info : Set Phy Delay.
Wr Phy Addr 1e : 401e
Rd Phy Addr 1f : 5988
Setup New Value =
 : 0
Wr Phy Addr 1f : 5900
Wr Phy Addr 1e : 801e
Wr Phy Addr 1e : 401e
Rd Phy Addr 1f : 5900
Read New Value =
 : 5900
Rd Phy Addr 11 : ac00
Info : Phy Link up on 1000Mbps.
Info : Set Mac Speed.
Info : Set Mac IPG.
Info : Assert mac reset
Info : Deassert mac reset
Info : Set Mac Address.
Info : Mac Reset Statistics Counters.
--------------------
aFramesTransmittedOK : 3
aFramesReceivedOK : 3
ifInErrors : 0
ifOutErrors : 1
etherStatsPkts : 3
etherStatsUndersizePkts : 0
etherStatsOversizePkts : 0
aRxFilterFramesErrors : 0
aFrameCheckSequenceErrors : 0
aTxPAUSEMACCtrlFrames : 0
aRxPAUSEMACCtrlFrames : 0
--------------------
--------------------
aFramesTransmittedOK : e
aFramesReceivedOK : e
ifInErrors : 0
ifOutErrors : 1
etherStatsPkts : e
etherStatsUndersizePkts : 0
etherStatsOversizePkts : 0
aRxFilterFramesErrors : 0
aFrameCheckSequenceErrors : 0
aTxPAUSEMACCtrlFrames : 0
aRxPAUSEMACCtrlFrames : 0
--------------------
--------------------
aFramesTransmittedOK : 14
aFramesReceivedOK : 14
ifInErrors : 0
ifOutErrors : 1
etherStatsPkts : 14
etherStatsUndersizePkts : 0
etherStatsOversizePkts : 0
aRxFilterFramesErrors : 0
aFrameCheckSequenceErrors : 0
aTxPAUSEMACCtrlFrames : 0
aRxPAUSEMACCtrlFrames : 0
--------------------
```

Using Wireshark

Wireshark is a free, open-source network packet analyzer software. Efinix® recommends that you use Wireshark to check for captured and transmitted packet consistency when the opposite end of the network is connected to a PC.

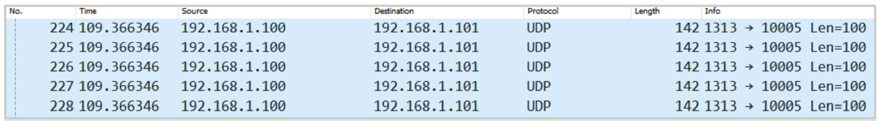
To use Wireshark:

1. Download Wireshark software from www.wireshark.org and install.
2. Open Wireshark and in the **Capture** category, you will see a list of available networks that are currently available on your computer.
3. Select **Ethernet** and Wireshark starts capturing any transaction occurring at the port.

Normal Mode

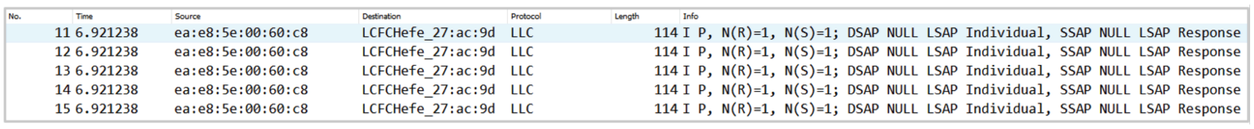
Once you start the debugging process in Open OCD debugger, you can expect the data packets to start showing up in Wireshark.

Figure 14: TEST_MODE=0 and PAT_TYPE=0



No.	Time	Source	Destination	Protocol	Length	Info
224	109.366346	192.168.1.100	192.168.1.101	UDP	142	1313 → 10005 Len=100
225	109.366346	192.168.1.100	192.168.1.101	UDP	142	1313 → 10005 Len=100
226	109.366346	192.168.1.100	192.168.1.101	UDP	142	1313 → 10005 Len=100
227	109.366346	192.168.1.100	192.168.1.101	UDP	142	1313 → 10005 Len=100
228	109.366346	192.168.1.100	192.168.1.101	UDP	142	1313 → 10005 Len=100

Figure 15: TEST_MODE=0 and PAT_TYPE=1



No.	Time	Source	Destination	Protocol	Length	Info
11	6.921238	ea:e8:5e:00:60:c8	LCFCHefe_27:ac:9d	LLC	114	I P, N(R)=1, N(S)=1; DSAP NULL LSAP Individual, SSAP NULL LSAP Response
12	6.921238	ea:e8:5e:00:60:c8	LCFCHefe_27:ac:9d	LLC	114	I P, N(R)=1, N(S)=1; DSAP NULL LSAP Individual, SSAP NULL LSAP Response
13	6.921238	ea:e8:5e:00:60:c8	LCFCHefe_27:ac:9d	LLC	114	I P, N(R)=1, N(S)=1; DSAP NULL LSAP Individual, SSAP NULL LSAP Response
14	6.921238	ea:e8:5e:00:60:c8	LCFCHefe_27:ac:9d	LLC	114	I P, N(R)=1, N(S)=1; DSAP NULL LSAP Individual, SSAP NULL LSAP Response
15	6.921238	ea:e8:5e:00:60:c8	LCFCHefe_27:ac:9d	LLC	114	I P, N(R)=1, N(S)=1; DSAP NULL LSAP Individual, SSAP NULL LSAP Response

Linked-Partner Mode

The data packets displayed in Wireshark for Linked-Partner mode will vary among users because the data packets are generated from different connected devices. For reference, see Figure 13: Linked-Partner Mode Telnet Session Output Example on page 34.

Example Design Configuration Registers

Table 29: Example Design Configuration Registers

Default value for all DWORD offsets are 0x0.

DWORD Offset	Name	R/W	Description
0x200	mac_sw_rst	R/W	Assert high to trigger proto_reset signal. The register output is tied to proto_reset signal.
0x204	Bit 0 - axi4_st_mux_select	R/W	0: Transmit AXI4-Stream TX packet from pattern generator. 1: Loopback AXI4-Stream RX path to AXI4-Stream TX path.
	Bit 1 - pat_mux_select	R/W	0: Select UDP pattern generator. 1: Select MAC pattern generator.
0x208	Bit 0 - udp_pat_gen_en	R/W	Assert high then low to trigger a UDP packet.
	Bit 1 - mac_pat_gen_en	R/W	Assert high then low to trigger MAC packet.
0x20c	Bit [15:0] - pat_gen_num	R/W	The number of transmit packet to send. Zero indicates unlimited packets.
	Bit [31:16] - pat_gen_ipg	R/W	The UDP/MAC interpacket gap.
0x210	pat_dst_mac [31:0]	R/W	The UDP/MAC pattern's destination MAC address.
0x214	pat_dst_mac [47:32]	R/W	
0x218	pat_src_mac [31:0]	R/W	The UDP/MAC pattern's source MAC address.
0x21c	pat_src_mac [47:32]	R/W	
0x220	pat_mac_dlen [15:0]	R/W	The length of a MAC pattern packet. The complete MAC pattern packet is pat_mac_dlen + 14 bytes (excluding CRC).
0x224	pat_src_ip [31:0]	R/W	The UDP pattern's source IP address.
0x228	pat_dst_ip [31:0]	R/W	The UDP pattern's destination IP address.
0x22c	Bit [15:0] pat_src_port	R/W	The UDP pattern's source port.
	Bit [31:16] pat_dst_port		The UDP pattern's destination port.
0x230	pat_udp_dlen [15:0]	R/W	The length of a UDP pattern packet. The complete UDP pattern packet is pat_udp_dlen + 45 bytes (excluding CRC).

Triple Speed Ethernet MAC Testbench

You can choose to generate the testbench when generating the core in the IP Manager Configuration window.

Note: You must include all **.v** files generated in the **/testbench** directory in your simulation.

Efinix provides a simulation script for you to run the testbench quickly using the Modelsim software. To run the Modelsim testbench script, run `vsim -do modelsim.do` in a terminal application. You must have Modelsim installed on your computer to use this script.

Table 30: Testbench Files

The IP Manager generates different encrypted source codes for you to simulate with different simulators.

Directory	Note
../Testbench	Contains the example design and testbench files.
../Testbench/modelsim	Contains the generated encrypted source code to simulate with the Modelsim simulator.
../Testbench/ncsim	Contains the generated encrypted source code to simulate with the NCSIM simulator.
../Testbench/synopsys	Contains the generated encrypted source code to simulate with the VCS simulator.

The simulation testbench environment is set in a RGMII TX and RGMII RX loop-back manner by connecting the RGMII TX interface to the RGMII RX interface at the testbench file. You can select the MAC speed, type of pattern generator, and packet length in the testbench.

The testbench has seven test cases. The default test case is test case 1. To use other available test cases, define the `TSET_CASE` parameter value in the **tb_top.v** file.

Figure 16: Testbench Data Flow

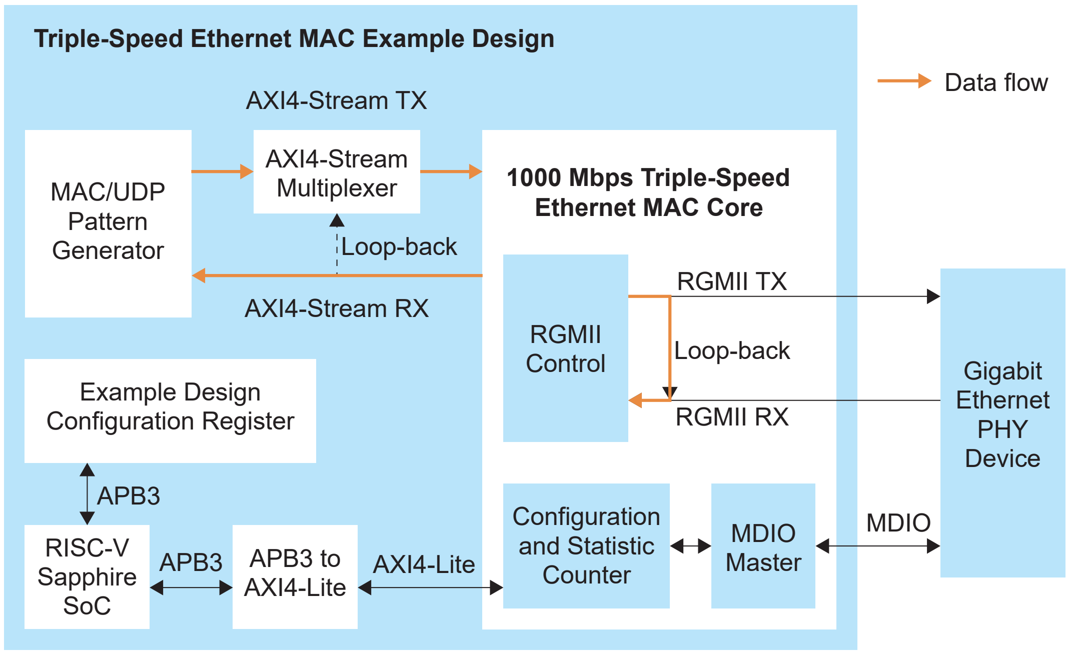

Triple-Speed Ethernet MAC Example Design
Data flow
AXI4-Stream TX
MAC/UDP Pattern Generator
AXI4-Stream Multiplexer
1000 Mbps Triple-Speed Ethernet MAC Core
Loop-back
AXI4-Stream RX
RGMII TX
RGMII Control
Loop-back
RGMII RX
Gigabit Ethernet PHY Device
Example Design Configuration Register
APB3
RISC-V Sapphire SoC
APB3
APB3 to AXI4-Lite
AXI4-Lite
Configuration and Statistic Counter
MDIO Master
MDIO

Table 31: Test Cases

TSET_CASE Value	Test Case Name	Description
1	Transmit MAC pattern packet test	Packet length is set to 100, packet number is set to 1000, and packet type is set to MAC pattern.
2	Transmit UDP pattern packet test	Packet length is set to 100, packet number is set to 1000, and packet type is set to UDP pattern.
3	Pause frame packet test	Five packets are transmitted. The first and second packets are normal packets, the third packet is a pause frame, and the fourth and fifth packets are normal packets.
4	Jumbo packet and oversize jumbo packet test	Packet length is set to 9000. Transmit the first jumbo packet with length set to 9000 and the second jumbo packet with length set to 9001. The second jumbo packet triggers an error on the ifInError register and the rx_axis_mac_tuser value turns 1.
5	Undersize packet test	The packet length is set to 0. Data byte of value 8'h00 is inserted to the packet to full fill the 64-bytes minimum packet length.
6	Error packet test	The first packet is a normal packet, the second packet has CRC error, and the third packet is an oversize packet.
7	Address filtering test	The packet's destination MAC address is not the same as the local MAC address.

The pattern generator and checker prints out `Correct MAC packet received` if the received packet is consistent with the transmitted packet. The following example shows the test case 1 output in the terminal.

```
---- Configure TSE MAC IP register setting ----
  9610000 - Correct MAC packet          0, received
 18330000 - Correct MAC packet          1, received
 27050000 - Correct MAC packet          2, received
 35770000 - Correct MAC packet          3, received
 44490000 - Correct MAC packet          4, received
 53210000 - Correct MAC packet          5, received
 61930000 - Correct MAC packet          6, received
 70650000 - Correct MAC packet          7, received
 79370000 - Correct MAC packet          8, received
 88090000 - Correct MAC packet          9, received
 96810000 - Correct MAC packet         10, received
105530000 - Correct MAC packet         11, received
114250000 - Correct MAC packet         12, received
122970000 - Correct MAC packet         13, received
```

Revision History

Table 32: Revision History

Date	Version	Description
December 2023	4.9	Updated Titanium resource utilization. (DOC-1601)
June 2023	4.8	Added Device Support and release notes sections. (DOC-1234) Updated rgmii_txc settings for example design and to run other designs at 1000 Mbps.
March 2023	4.7	Added-in Normal Mode and Linked-Partner Mode sub-topic in Using Wireshank topic. (DOC-1176) Updated program in Using the PHY RGMII Clock-to-Data Delay (Trion Only).
February 2023	4.6	Added Ti60 F225 Development Board example design support. (DOC-1152)
February 2023	4.5	Added note about the resource and performance values in the resource and utilization table are for guidance only.
December 2022	4.4	Updated example design. (DOC-1015) Added New in version section.
October 2022	4.3	Updated ports width and descriptions and added new ports. Updated registers width and added new registers. Corrected AXI4-Stream Data Width parameter option. Corrected s_axi_wdata width. Updated example design directory and file names. Added Description about realigning PHY clock with top-module signals.
September 2022	4.2	Added support for GMII, RMII, and MII PHY interfaces. Updated core IP manager parameters.
April 2022	4.1	Added note about RX packet error feature is not supported and user can ignore warnings on rgmii_rx_ctl_HI signal. (DOC-787)
December 2021	4.0	Added example design in IP manager. Corrected PHY IC description. Updated example design RISC-V SoC to Sapphire. Updated supported FIFO depth to 16384.
November 2021	3.3	Corrected Block Editor Settings for rgmii_rx_ctl and rgmii_tx_ctl signals.
October 2021	3.2	Updated Block Editor Settings for rgmii_rx_ctl and rgmii_tx_ctl signals. (DOC-586)
October 2021	3.1	Added note to state that the f_{MAX} in Resource Utilization and Performance, and Example Design Implementation tables were based on default parameter settings.

Date	Version	Description
June 2021	3.0	Added missing rgmii_txc_LO interface description in the ports table. Added parameters to set in Interface Designer DDIO settings. Corrected Output Signals Waveform with DDIO Enabled figure. Added note about including all **.v** generated in testbench folder is required for simulation.
December 2020	2.0	Updated user guide for Efinix® IP Manager which includes added IP Manager topics, updated parameters, and user guide structure.
October 2020	1.2	Corrected Bit 0 - axi4_st_mux_select of the example design configuration registers description. Updated normal and link test mode descriptions. Added aFramesReceivedLen[15:0] to statistic counter register. (DOC-318) Added information about adding delay to re-allign the PHY clock in the example design. (DOC-318) Added settings to enable the hard DDIO blocks in Interface Designer and waveforms after integrating the hard DDIO blocks into the core. (DOC-318)
August 2020	1.1	Updated the Using this Example with the RISC-V SDK topic.
July 2020	1.0	Initial release.