LATTICE SEMICONDUCTOR

LatticeCORE

Tri-Speed Ethernet MAC IP User Guide

April 2015
IPUG51_3.3

Table of Contents

Introduction

This document provides technical information about the Lattice 10/100/1G Tri-Speed Ethernet Media Access Controller (TSMAC) IP core.

The TSMAC IP core supports the ability to transmit and receive data between a host processor and an Ethernet network. The main function of the Ethernet MAC is to ensure that the Media Access rules specified in the 802.3 IEEE standard are met while transmitting a frame of data over Ethernet. On the receiving side, the Ethernet MAC extracts the different components of a frame and transfers them to higher applications through the FIFO interface.

The TSMAC IP core comes with the following documentation and files:

- Protected netlist/database
- Behavioral RTL simulation model
- Source files for instantiating and evaluating the core

Quick Facts

Table 1-1 gives quick facts about the TSMAC IP core.

Table 1-1. TSMAC IP Core Quick Facts

		TSMAC IP Configuration			
		Across All IP Configurations (Classic, Gigabit, SGMII, MIIM)			
Core Requirements	FPGA Families Supported	ECP5, LatticeECP3, LatticeXP2			
	Minimal Device Needed	LFE5UM-25F-6MG285C	LFE5U-25F-6MG285C	LFE3-17E-6FN256C	LFXP2-5E-5F132C
Resource Utilization	Data Path Width	8			
	LUTs	1400-1900			1500-2000
	sysMEM EBRs	1-2			
	Registers	1100-1400			
Design Tool Support	Lattice Implementation	Lattice Diamond® 3.2			
	Synthesis	Synopsys® Synplify® Pro for Lattice I-2013.09L-SP1-1			
		Mentor Graphics® Precision® RTL			
	Simulation	Aldec® Active-HDL™ 9.3 Lattice Edition			
		Mentor Graphics® ModelSim® SE (Verilog only)			

Features

- Compliant to IEEE 802.3-2005 standard
- Generic 8-bit host interface
- 8-bit wide internal data path
- Generic transmit and receive FIFO interface
- Full-duplex operation in 1G mode
- Full- and half-duplex operation in 10/100 mode
- Transmit and receive statistics vector
- Programmable Inter-Packet Gap (IPG)
- Multicast address filtering
- Selectable MAC operating options
  - Classic TSMAC with G/MII
  - Gigabit MAC with GMII
  - SGMII Easy Connect MAC with GMII, configurable option available on ECP5 and LatticeECP3 devices
- Supports
  - Full-duplex control using PAUSE frames
  - VLAN tagged frames
  - Automatic re-transmission on collision
  - Automatic padding of short frames
  - Multicast and Broadcast frames
  - Optional FCS transmission and reception
  - Optional MII management interface module
  - Jumbo frames of any length

Functional Description

The TSMAC IP core is a fully synchronous machine composed of Transmit and Receive MAC sections that operate independently to support full duplex operation.

The block diagram of the TSMAC IP core is shown in Figure 2-1. The major functional modules are:

- Host Interface
- Receive MAC
- Transmit MAC
- Internal Buffers and FIFO Interfaces
- G/MII
- Management interface module (optional)

In the 1G mode, the 125 MHz system clock is supplied to the Transmit MAC. The system clock is used to clock the GMII interface for data transmission. When receiving data, an external PHY device provides the 125 MHz clock to the GMII receive section. The 125 MHz clock is used to clock the Receive MAC.

In the 10/100 mode, an external PHY device supplies the clock to the Transmit MAC and the Receive MAC.

Configuration Options

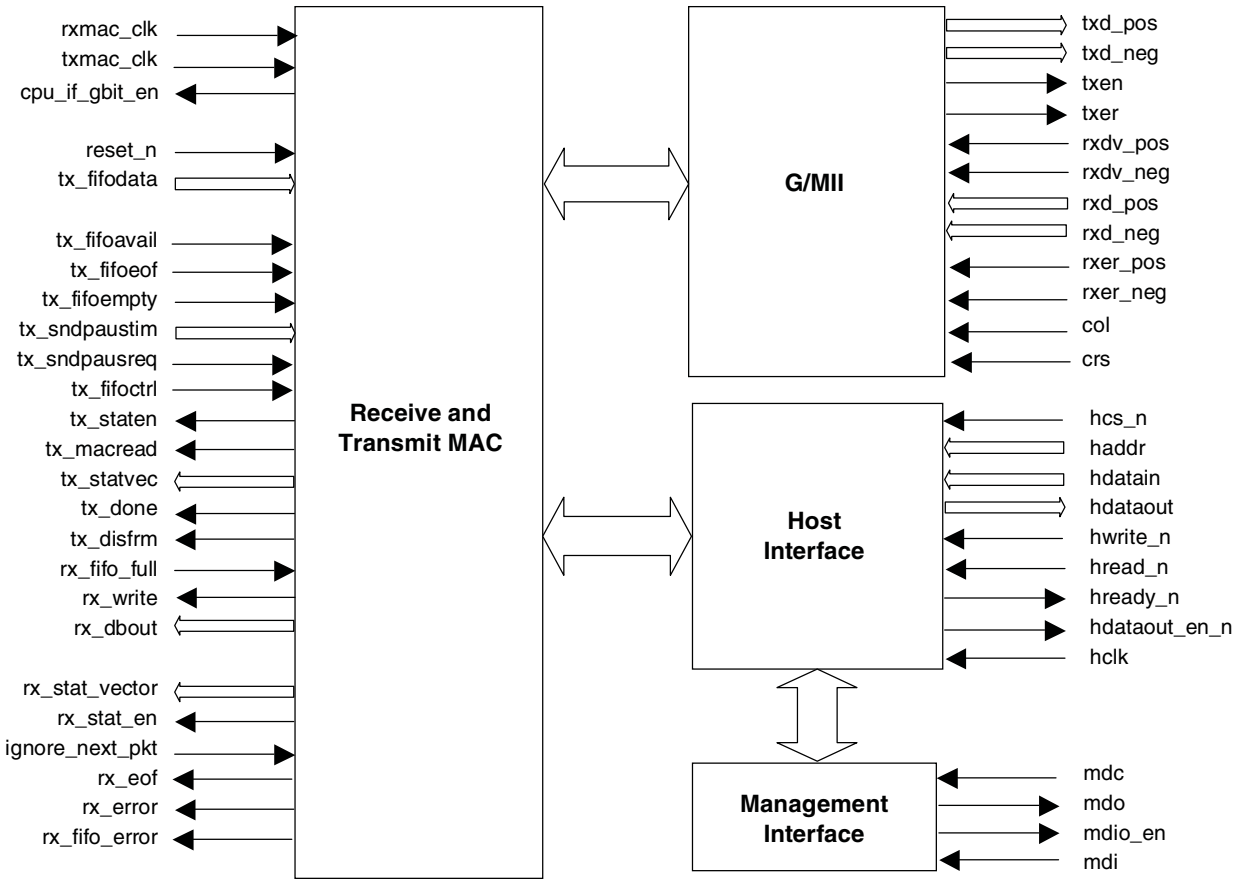
The TSMAC IP core supports three basic configuration options:

- Classic TSMAC with G/MII
- Gigabit MAC with GMII
- SGMII Easy Connect MAC with GMII.

Classic TSMAC Option

When the Classic TSMAC option is selected, the TSMAC IP core can be configured to operate in either the 1G mode (1000Mbps data rate) or the Fast Ethernet mode (10/100 Mbps data rate). A block diagram of the Classic TSMAC IP core option is shown in Figure 2-1. Operation in either 1G mode or Fast Ethernet mode is selected by setting an internal register bit.

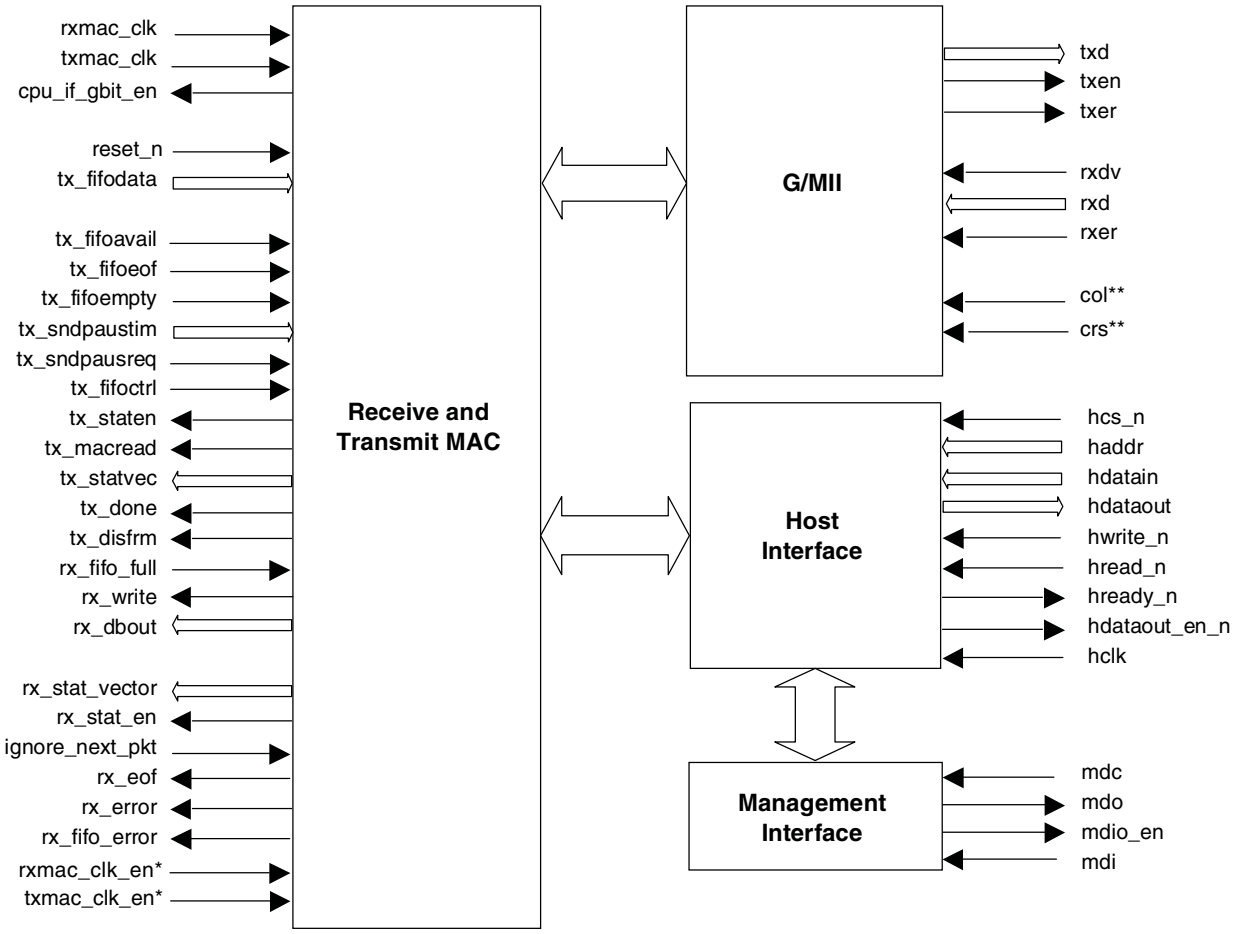
Figure 2-1. Core Block Diagram for the Classic TSMAC IP Core Option



1G or SGMII Configurable Options

For the SGMII Easy Connect configuration option, the TSMAC operates at the Gigabit data rate and uses clock enables provided by the SGMII PCS IP core to work at the three different speeds. For the Gigabit MAC configuration option, the TSMAC always operates at the Gigabit data rate and is effectively configured as a full-duplex Gigabit MAC only. A block diagram of the Gigabit MAC configuration or SGMII Easy Connect configuration is shown in Figure 2-2.

Figure 2-2. Core Block Diagram for the 1G or SGMII Configurable Options



* These inputs are only present for the SGMII Easy Connect option.
** These inputs are not present for the Gigabit MAC option.

Functional Overview

The TSMAC IP core transmits and receives data between a client application and an Ethernet network. The main function of the Ethernet MAC is to ensure that the Media Access rules specified in the 802.3 IEEE standard are met while transmitting and receiving Ethernet frames. Figure 2-3, Figure 2-4, and Figure 2-5 show some of the frame formats of data transmitted and received on the Ethernet network that the TSMAC IP core supports.

On the receiving side, the Ethernet MAC extracts the different components of a frame and transfers them to higher applications through the client FIFO interface.

The data received from the G/MII interface is first buffered until sufficient data is available to be processed by the Receive MAC (Rx MAC). The Preamble and the Start-of-Frame Delimiter (SFD) information are then extracted from the incoming frame to determine the start of a valid frame. The Receive MAC checks the address of the received packet and validates whether the frame can be received before transferring it into the FIFO. Only valid frames are transferred into the FIFO (runts and fragments are discarded). The Rx MAC also provides a statistics vector on a per packet basis that can be used by the application. The TSMAC IP core always calculates CRC to check whether the frame was received error-free.

On the transmit side, the Tx MAC is responsible for controlling access to the physical medium. The Tx MAC reads data from an external client Tx FIFO, formats this data into an Ethernet packet and passes it to the G/MII module.

The Tx MAC reads data from the Tx Client FIFO when the client indicates a packet is available, and the Tx MAC is in its appropriate state. The Tx MAC pre-fixes the Preamble and the Start-of-Frame Delimiter information to the data and appends the Frame Check Sequence at the end of the data. In half-duplex operation, the Tx MAC stores the first 64 bytes of data from the external FIFO in an internal buffer, to be used in re-transmitting data on collisions.

The SGMII Easy Connect configuration option adds pins and logic for seamless connection to the Lattice Gigabit Ethernet PCS IP core.

Figure 2-3. Un-Tagged Ethernet Frame Format

PREAMBLE	SFD	DESTINATION ADDRESS	SOURCE ADDRESS	LENGTH/ TYPE	DATA/PAD	FRAME CHECK SEQUENCE
7 bytes	1 byte	6 bytes	6 bytes	2 bytes	46-1500 bytes	4 bytes

Figure 2-4. VLAN-Tagged Ethernet Frame Format

PREAMBLE	SFD	DESTINATION ADDRESS	SOURCE ADDRESS	VLAN TAG HEADER	LENGTH/ TYPE	DATA/PAD	FRAME CHECK SEQUENCE
7 bytes	1 byte	6 bytes	6 bytes	4 bytes	2 bytes	46-1500 bytes	4 bytes

Figure 2-5. Ethernet Control Pause Frame Format

PREAMBLE	SFD	DESTINATION ADDRESS 01-80-C2-00-00-01	SOURCE ADDRESS	LENGTH/TYPE 88-08	MAC CTL OP_CODE 00-01	OP_CODE PARAMS/RSV	FRAME CHECK SEQUENCE
7 bytes	1 byte	6 bytes	6 bytes	2 bytes	2 bytes	60 bytes	4 bytes

A Tagged frame includes a 4-byte VLAN Tag field, which is located between the Source Address field and the Length/Type field. The VLAN Tag field includes the VLAN Identifier and other control information needed when operating with Virtual Bridged LANs as described in IEEE P802.1Q.

System Block Diagrams

Figure 2-6 shows an FPGA system-level block diagram of how the MAC core is instantiated and used when configured to support the Classic MAC mode of operation. Note that the MAC core needs a certain amount of support logic like PLLs, I/O buffers, and I/O flip flops to meet the IEEE 802.3 I/O specifications (see Application Support for more details).

Figure 2-6. System Block Diagram for the Classic TSMAC IP Core Option

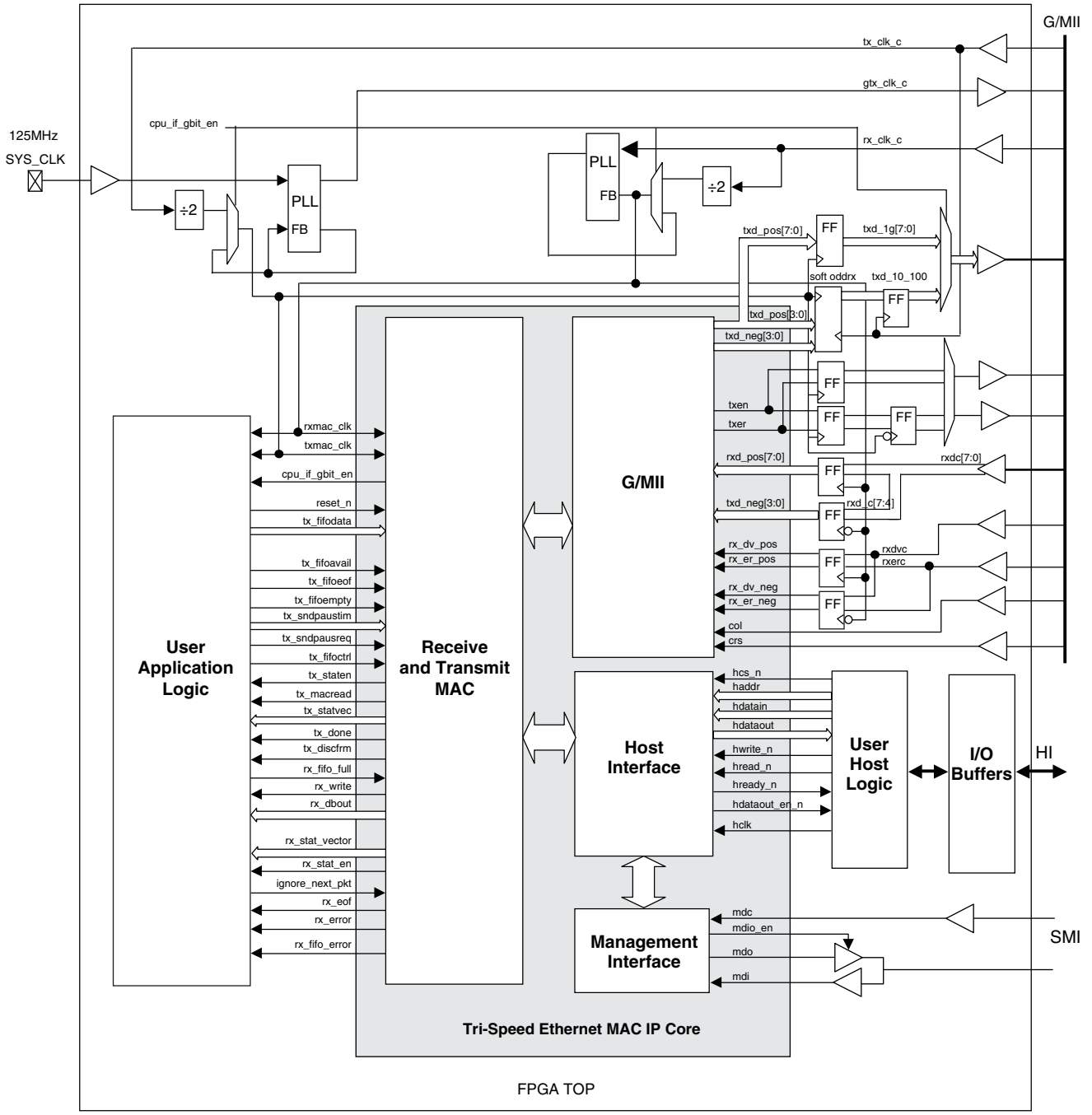


Figure 2-7 shows a FPGA system-level block diagram of how the MAC core is instantiated and used when configured to support the Gigabit MAC mode of operation.

Figure 2-7. System Block Diagram for the Gigabit MAC Configurable Option

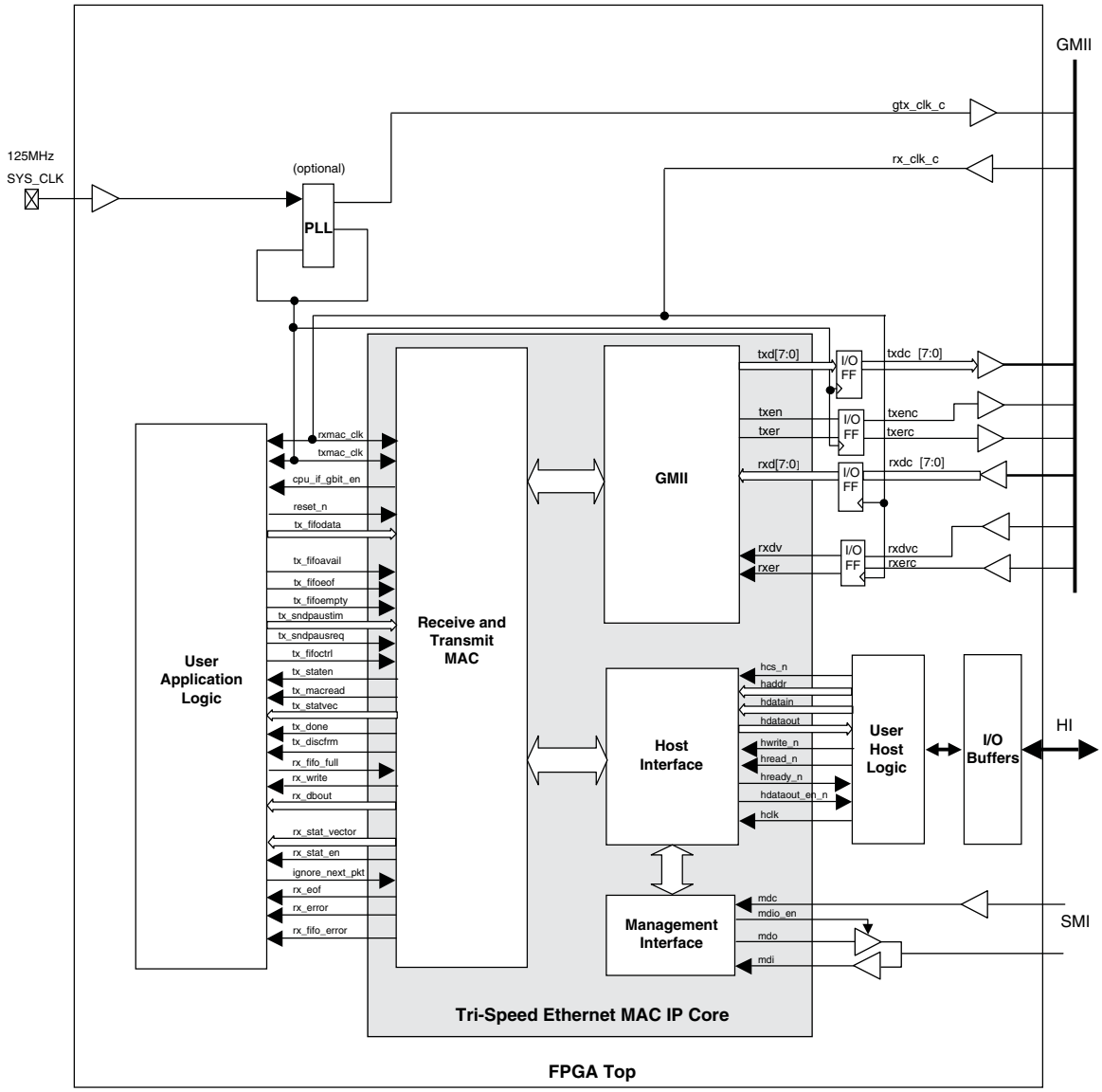
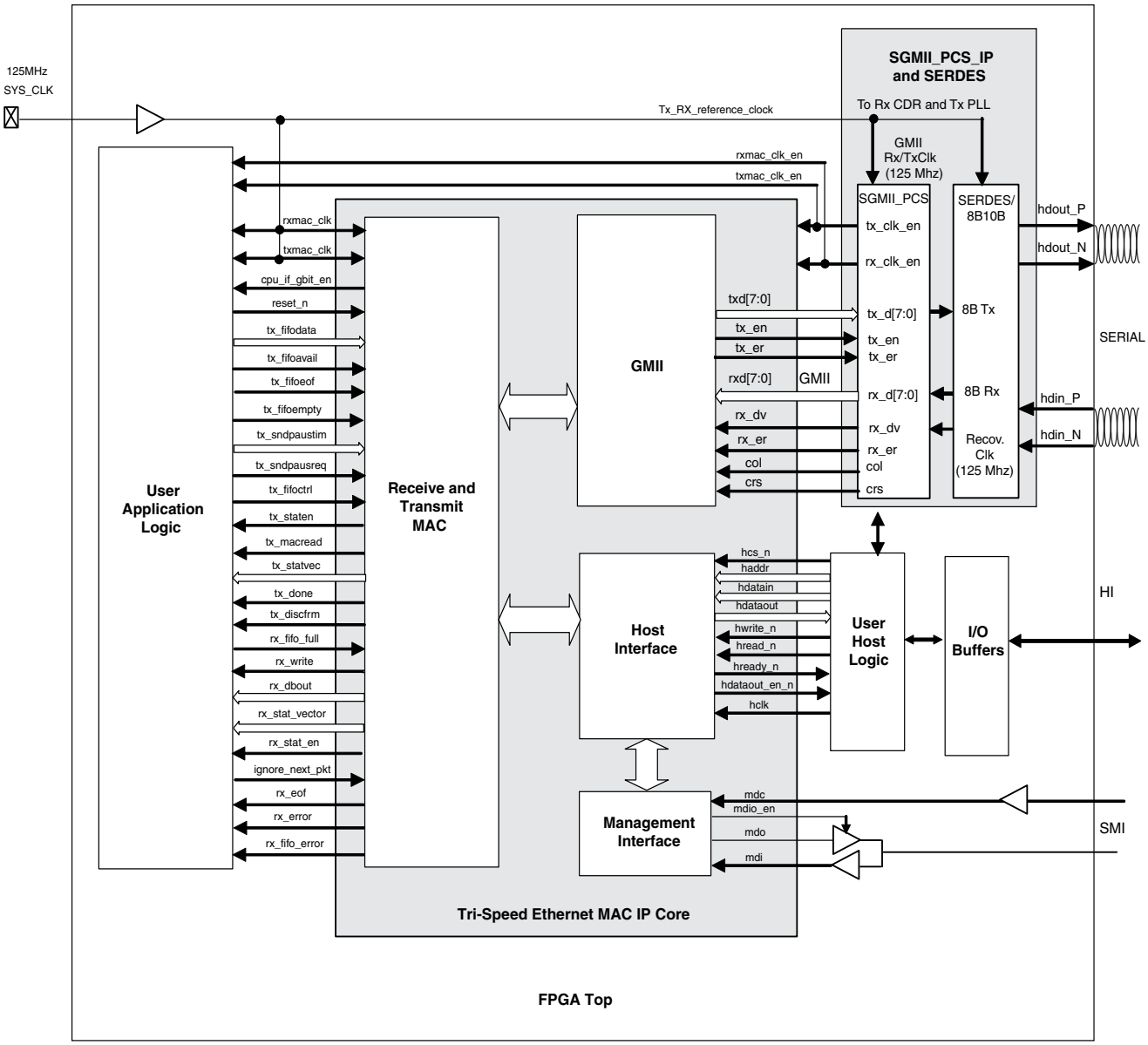


Figure 2-8 shows a FPGA system-level block diagram of how the MAC core is instantiated and used when configured to support the SGMII Easy Connect mode of operation.

Figure 2-8. System Block Diagram for the SGMII Easy Connect Option



Core Signal Descriptions

Table 2-1 lists the I/O signals for the TSMAC IP core.

Table 2-1. TSMAC IP Core Input and Output Signals

Port Name	Type	Active State	Description
Clocks and Reset/Other			
rxmac_clk	Input	N/A	**Receive MAC Application Interface Clock.** This clock is used by the client application and MAC. All outputs driven by the Rx MAC on the client side are synchronous to this clock. This clock's frequency is 125,12.5 or 1.25 MHz depending on the mode 1G/100/10 respectively. *Note: this clock can be viewed as a "byte" clock, since all Rx MAC bytes are aligned with this clock. This clock is derived from the system G/MII rx_clk.* Always 125 MHz in Easy Connect mode.
txmac_clk	Input	N/A	**Transmit MAC Application Interface Clock**. This clock is used by the client application and MAC. All inputs to the Tx MAC on the client side should be synchronous to this clock. This clock's frequency is 125,12.5 or 1.25 MHz depending on the mode 1G/100/10 respectively. *Note: this clock can be viewed as a "byte" clock, since all Tx MAC bytes should be aligned with this clock. This clock is derived from the system sys_clk or tx_clk. (1G or 10/100 respectively).* Always 125 MHz in Easy Connect mode.
hclk	Input	N/A	**Host Clock**. This is the Host Bus clock, and is used to clock the Host Bus interface.
mdc	Input	N/A	**Management Data Clock**. This clock is used only when the Management Interface module is implemented.
reset_n	Input	Low	**Reset**. This is an active low asynchronous signal that resets the internal registers and internal logic. When activated, the I/O signals are driven to their inactive levels.
txmac_clk_en	Input	N/A	**Tx Clock Enable**. This input signal is a clock enable used only in the SGMII Easy Connect option. The SGMII_PCS IP core drives this signal. The clock enable is always high for 1G operation. For 100 Mbps operation the clock enable is asserted high once every ten (125MHz) clocks, and for 10 Mbps operation the clock enable is asserted high once every hundred (125MHz) clocks.
rxmac_clk_en	Input	N/A	**Rx Clock Enable**. This input signal is a clock enable used only in the SGMII Easy Connect option. The SGMII_PCS IP core drives this signal. The clock enable is always high for 1G operation. For 100 Mbps operation the clock enable is asserted high once every ten (125MHz) clocks, and for 10 Mbps operation the clock enable is asserted high once every hundred (125MHz) clocks.
cpu_if_gbit_en	Output	High	**CPU Interface 1G Mode Enabled Indication**. This signal, when high, is an indication from the CPU interface that the 1G mode is enabled. This signal reflects the state of bit 0 of the MAC mode register.
Host Interface			
hcs_n	Input	Low	**Chip Select**. This is an active low signal used to select the core for register Read/Write operations.
haddr[7:0]	Input	N/A	**Address**. This selects one of the internal core registers.
hdatain[7:0]	Input	N/A	**Data Bus Input**. The CPU writes to the internal registers through the data bus.
hwrite_n	Input	Low	**Host Write**. This active low signal is used to write data to the selected register.
hread_n	Input	Low	**Host Read**. This active low signal is used to read data from the selected register.
hready_n	Output	Low	**Ready**. This is an active low signal used to indicate the end of transfer. For write operations, hready_n is asserted after data is accepted (written). For read operations hready_n is asserted after data on the hdataout bus is ready to be driven out.

Table 2-1. TSMAC IP Core Input and Output Signals

Port Name	Type	Active State	Description
hdataout_en_n	Output	Low	**Data Out Enable**. This signal is driven low whenever the TTSMAC IP core outputs valid data onto the hdataout bus. This signal can be used to build a bi-directional data bus.
hdataout[7:0]	Output	N/A	**Data Bus Output**. The CPU reads the internal registers through the data bus.
Transmit MAC Application Interface			
tx_fifodata[7:0]	Input	N/A	**Transmit FIFO Read Data Bus**. The data from the FIFO is presented on this bus.
tx_fifoavail	Input	High	**Transmit FIFO Data Available**. When asserted, this signal indicates that the TxFIFO has data ready for transmission on the G/MII interface. Once this signal is asserted by the client, a short delay later the frame will be transmitted. The client needs to use an appropriate threshold on the client FIFO to indicate that a frame is ready to be sent and use that threshold as the tx_fifo_avail signal.
tx_fifoeof	Input	High	**Transmit FIFO End of Frame**. This signal is asserted along with the last byte of frame data indicating the end of the frame.
tx_fifoempty	Input	High	**Transmit FIFO Empty**. This signal indicates that the TxFIFO is empty. When this signal is asserted and the TSMAC IP core is reading the FIFO, the under-run condition is transferred to the network through the txer signal.
tx_sndpaustim[15:0]	Input	N/A	**PAUSE Frame Timer**. This signal indicates the PAUSE time value that should be sent in the PAUSE frame.
tx_sndpausreq	Input	High	**PAUSE Frame Request**. When asserted, the TSMAC IP core transmits a PAUSE frame. This is also the qualifying signal for the tx_sndpausetim bus.
tx_fifoctrl	Input	N/A	**FIFO Control Frame**. This signal indicates whether the current frame in the Transmit FIFO is a control frame or a data frame. It is qualified by the tx_fifoavail signal. The following values apply: • 1 = Control frame • 0 = Normal frame
tx_staten	Output	High	**Transmit Statistics Vector Enable**. When asserted, the contents of the statistics vector bus tx_statvec are valid.
tx_macread	Output	High	**Transmit FIFO Read**. This is the TSMAC IP core Transmit FIFO read request, asserted by the TSMAC IP core when it intends to read the client FIFO. The MAC core will first assert the tx_macread signal if the client FIFO is not empty (i.e., tx_fifoempty = 0), after which the tx_macread may de-assert (based on MAC processing) or re-assert (based on MAC processing and if tx_fifoempty is still 0). The tx_macread signal should be tied to the client FIFO read pin, and the FIFO empty pin should be tied to the tx_fifoempty of the MAC core.

Table 2-1. TSMAC IP Core Input and Output Signals

Port Name	Type	Active State	Description
tx_statvec[30:0]	Output	N/A	**Transmit Statistics Vector**. This bus includes useful information about the frame that was just transmitted. The corresponding bit locations of this bus are defined as follows: • tx_statvec[0] - UNICAST frame • tx_statvec[1] - Multicast frame • tx_statvec[2] - BROACAST frame • tx_statvec[3] - Bad FCS frame • tx_statvec[4] - JUMBO frame • tx_statvec[5] - FIFO under-run • tx_statvec[6] - PAUSE frame • tx_statvec[7] - VLAN tagged frame • tx_statvec[21:8] - Number of bytes in the transmitted frame • tx_statvec[22] - Deferred transmission • tx_statvec[23] - Excessive deferred transmission • tx_statvec[24] - Late collision • tx_statvec[25] - Excessive collision • tx_statvec[29:26] - Number of early collisions • tx_statvec[30] - FCS generation is disabled and a short frame was transmitted
tx_done	Output	High	**Transmit Done**. This signal is asserted for one clock cycle after transmitting a frame if no errors were present in transmission.
tx_discfrm	Output	High	**Discard Frame**. This signal is asserted at the end of a frame transmit process if the TSMAC IP core detected an error. The possible conditions are: • A FIFO under-run • Late collision (10/100 Mode only) • Excessive Collisions (10/100 Mode only) The user application normally moves the pointer to next frame in these conditions.
Management Interface Signals			
mdi	Input	High	**Management Data Input**. Used to transfer information from the PHY to the management module.
mdo	Output	High	**Management Data Output**. Used to transmit information from the management module to the PHY.
mdio_en	Output	High	**Management Data Out Enable**. Asserted whenever `mdo` is valid. This may be used to implement a bi-directional signal for `mdi` and `mdo`.
G/MII Signals			
txd_pos[7:0][1] txd_neg[3:0][1] txd[7:0][2]	Output	High	**txd_pos[7:4] - Transmit Data Sent to the PHY Chip** - High nibble. In 1G mode, these bits are used as the GMII txd[7:4] bits after they are pipelined outside the core through some I/O flip-flops clocked at 125 MHz (txmac_clk). These bits are not used in the 10/100 mode. See Figure 2-6. **txd_pos[3:0], txd_neg[3:0] - Transmit Data Sent to the PHY Chip - Low nibble.** In both 1G mode and 10/100 mode, both the txd_pos[3:0] and txd_neg[3:0] bits are used to generate the G/MII txd[3:0] bits after they are muxed outside the core through some I/O DDR cells. Note in the 1G mode, txd_pos[3:0] and txd_neg[3:0] will always have the same value, whereas in the 10/100 mode, the txd_pos[3:0] will have the high nibble of the byte transmitted and txd_neg[3:0] will have the low nibble. In the 1G mode, the txmac_clk rate is 125 MHz and in the 10/100 mode, the clock rate is 1.25 MHz and 12.5 MHz respectively. See Figure 2-6. **txd[7:0]** - Transmit Data Sent to the PHY Interface. These GMII Tx data outputs go to the SGMII_PCS IP core (SGMII Easy Connect option) or to the 1G GMII PHY interface (Gigabit interface option). See Figure 2-7 and Figure 2-8.

Table 2-1. TSMAC IP Core Input and Output Signals

Port Name	Type	Active State	Description
txen	Output	High	**Transmit Enable**. Asserted by the TSMAC IP core to indicate the txd bus contains valid frame.
txer	Output	High	**Transmit Error**. Asserted when the TSMAC IP core generates a coding error on the byte currently being transferred.
rxdv_pos[1] rxdv_neg[1] rxdv[2]	Input	High	**Receive Data Valid**. Indicates the data on the rxd bus is valid. Rxdv is used in the SGMII Easy Connect MAC option and the Gigabit MAC option. Rxdv_pos and rxdv_neg are used only in the Classic TSMAC IP core interface option. This signal is pipelined before entering the core with the rx_clk. Note the "_pos" signals are sampled on the rising edge of the rxmac_clk and the "_neg" signals are sampled on the falling edge of the rxmac_clk, before being presented to the MAC core. See Figure 2-6.
rxd_pos[7:0][1] rxd_neg[3:0][1] rxd[7:0][2]	Input	N/A	**Receive Data Bus**. Data is driven by the PHY on these lines, and is valid whenever rxdv is asserted. These signals are pipelined before entering the core with the rxmac_clk. Note the "_pos" signals are sampled on the rising edge of the rxmac_clk and the "_neg" signals are sampled on the falling edge of the rxmac_clk, before being presented to the MAC core. See Figure 2-6. **rxd[7:0] - Receive Data from the PHY Interface**. These GMII Rx data inputs (valid whenever rxdv is asserted) come from the SGMII_PCS IP core (SGMII Easy Connect option) or from the 1G GMII PHY interface (Gigabit MAC option). See Figure 2-7 and Figure 2-8.
rxer_pos[1] rxer_neg[1] rxer[2]	Input	High	**Receive Data Error**. This signal is asserted by the external PHY device when it detects an error during frame reception. This signal is pipelined before entering the core with the rxmac_clk. Note the "_pos" signals are sampled on the rising edge of the rxmac_clk and the "_neg" signals are sampled on the falling edge of the rxmac_clk, before being presented to the MAC core. See Figure 2-6. **rxer - Receive Data Error from the PHY Interface**. This GMII Rx error input comes from the SGMII_PCS IP core (SGMII Easy Connect option) or from the 1G GMII PHY interface (Gigabit MAC option). See Figure 2-7 and Figure 2-8.
col	Input	High	**Collision**. This active-high signal indicates a collision occurred during transmission. This signal is valid for half-duplex operation in Fast Ethernet (10/100) for the Classic and SGMII Easy Connect options only. Otherwise, it is ignored.
crs	Input	High	**Carrier Sense**. This signal, when logic high, indicates the network has activity. Otherwise, it indicates the network is idle. This signal is valid for half-duplex operation in Fast Ethernet (10/100) for the Classic and SGMII Easy Connect options only.
Receive MAC Application Interface			
rx_fifo_full	Input	High	**Receive FIFO Full**. This signal indicates the Rx FIFO is full and cannot accept any more data. This is an error condition and should never happen.
rx_write	Output	High	**Receive FIFO Write**. This signal is asserted by the TSMAC IP core to request a FIFO write.
rx_dbout[7:0]	Output	N/A	**Receive FIFO Data Output**. This bus contains the data that is to be written into the Receive FIFO.

Table 2-1. TSMAC IP Core Input and Output Signals

Port Name	Type	Active State	Description
rx_stat_vector[31:0]	Output	N/A	**Receive Statistics Vector**. This bus indicates the events encountered during frame reception. This bus is qualified by the rx_stat_en signal. The definition of each signal is explained in the Receive MAC section of this user's guide. The corresponding bit locations of this bus are defined as follows: rx_statvec[15:0] - Frame Byte Count rx_statvec[16] - VLAN Tag Detected rx_statvec[17] - Pause Frame rx_statvec[18] - Control Frame tx_statvec[19] - Unsupported Opcode rx_statvec[20] - Dribble Nibble rx_statvec[21] - Broadcast Address rx_statvec[22] - Multicast Address rx_statvec[23] - Receive OK rx_statvec[24] - Length Check Error rx_statvec[25] - CRC Error rx_statvec[26] - Packet Ignored rx_statvec[27] - Carrier Event Previously Seen rx_statvec[28] – Unused rx_statvec[29] - IPG Violation rx_statvec[30] - Short Frame rx_statvec[31] - Long Frame
rx_stat_en	Output	High	**Receive Statistics Vector Enable**. When asserted, this signal indicates that the contents of the rx_stat_vector bus is valid.
ignore_next_pkt	Input	High	**Ignore Next Packet**. This signal is asserted by the host to prevent a Receive FIFO Full condition. The Receive MAC continues dropping packets as long as this signal is asserted. This is an asynchronous signal.
rx_eof	Output	High	**End Of Frame**. Indicates all the data for the current packet has passed on to the FIFO.
rx_error	Output	High	**Receive Packet Error**. When asserted, this signal indicates the packet contains error(s). This signal is qualified with the rx_eof signal. The rx_error signal will be asserted for any of the following three conditions: • The rxer signal on the GMII is asserted by the PHY during frame reception. • There are Rx FCS errors on received frames. • There is a length check error on the received frame.
rx_fifo_error	Output	High	**Receive FIFO Error**. This signal is asserted when the external Rx FIFO is full (rx_fifo_full signal asserted) and the Rx FIFO is being written to by the Rx MAC (rx_write is asserted). When this error signal is asserted the rx_write signal will be de-asserted as long as the rx_fifo_error signal is asserted. The rx_fifo_error signal will be de-asserted when the end of packet (rx_eof) exits the receive FIFO (Note: for this errored condition data will continue to be pulled out of the receive FIFO, even while the rx_write signal has been de-asserted).

1. Classic TSMAC IP core option.
2. Gigabit MAC or SGMII Easy Connect MAC options.

Table 2-2 lists TSMAC IP core system input and output signals.

Table 2-2. TSMAC IP Core System Input and Output Signals

Port Name	Type	Active State	Description
Clocks and Reset			
sys_clk	Input	N/A	**System Clock**. In the 1G mode, the Tx MAC is clocked by this signal. All the input and the output signals of the Tx MAC are synchronous to this clock in the 1G mode. The frequency is always at 125 MHz. Note in the 1G mode the core's txmac_clk is derived from this clock.
tx_clk	Input	N/A	**Transmit Clock**. This clock is used in the 10/100 Mbps mode only. The Tx MAC, Tx MAC application interface and the MII are synchronous to this signal. This clock has a frequency of 2.5/25 MHz for 10/100 Mbps operation respectively. Note in the 10/100 mode tx_clk is divided by two to provide the clock (txmac_clk) to the transmit MAC section. In the 10/100 mode the transmit signals at the GMII interface are always synchronous to tx_clk.
rx_clk	Input	N/A	**Receive Clock**. This clock is an input from the PHY device. In the 1G mode, rx_clk frequency is 125 MHz while in the 10/100 mode, the corresponding rx_clk frequency is 2.5/25 MHz respectively. In the 10/100 mode rx_clk is divided by two to provide the clock (rxmac_clk) to the Receive MAC section. In the 1G mode this clock is provided directly to the Receive MAC section. The receive signals at the GMII interface are always synchronous to rx_clk.
gtx_clk	Output	N/A	**Gigabit Transmit Clock**. This clock is used in the 1G mode only. The transmit signals that are outputs on the GMII interface are synchronous to this clock. This clock has a frequency of 125 MHz. This clock is derived from the sys_clk. See Figure 2-6.

Host Interface

The Host Interface module is a fully synchronous module that runs off the host clock. A number of registers are initialized via the Host interface to ensure that the TSMAC IP core functions as intended. The write operation to an internal register is initiated when the hcs_n and hwrite_n signals are asserted and hread_n signal is deasserted. The address of the targeted register is placed on the haddr bus, while the valid data is placed on the hdatain bus. The contents of the address and data busses should remain unchanged until the TSMAC IP core asserts the hready_n signal. The signals hcs_n, hwrite_n and hread_n must remain unchanged until hready_n is asserted.

A register read is initiated by asserting the hcs_n and hread_n signals, while keeping the hwrite_n signal deasserted. The address of the targeted register is placed on the haddr bus. The TSMAC IP core places the content of the targeted register on the hdataout bus and qualifies it with the assertion of hready_n signal. The haddr bus should not change until the hready_n signal is asserted.

Figure 2-17 shows the timing diagram associated with the host interface write and read operations.

Receive MAC (Rx MAC)

The main function of the Rx MAC is to accept the formatted data from the G/MII interface and pass it to the host application through an external FIFO. In this process, the Rx MAC performs the following functions:

- Detect the start of frame
- Compare the MAC address
- Re-calculate CRC
- Process the control frame and pass it to the flow control module.

The Rx MAC operation is determined by programming the MODE and TX_RX_CTL registers. These register definitions and bit descriptions can be found in Table 2-4. Note that setting the Gbit_en bit in the MODE register to high sets the TSMAC to operate in 1G mode whereas setting the Gbit_en bit to low sets the TSMAC to operate in 10/100 Mode.

Programming the MODE and TX_RX_CTL registers can control the Receive MAC operation. The various events that occur during the reception of a frame are logged into the rx_stat_vector signal and the TX_RX_STS register. At the end of reception, the rx_stat_en signal is asserted to qualify the rx_stat_vector signal. The TSMAC IP core can report a wealth of information such as

- FIFO overflow
- CRC error
- Receive error
- Short frame reception
- Long frame reception
- IPG violation

By default, the entire frame, except the preamble and SFD bytes, is sent to the FIFO via the Rx MAC application interface signals. If the user does not want to receive the FCS, the core can be programmed to strip the FCS field as well as any PAD bytes in the frame and send the rest to the FIFO.

The Rx MAC section operates on the rxmac_clk derived from the rx_clk sourced from the PHY. All the signals on the Receive MAC FIFO interface are synchronous to this clock.

The Rx MAC is disabled while Rx_en is low (Bit_2 of the MODE register) and should only be enabled after the associated registers are properly initialized.

Receiving Frames

The frames received by the Rx MAC are analyzed and the Preamble and SFD bytes are stripped off the frame before it is transferred to an external FIFO. The client data interface between the MAC and the FIFO is eight bits wide.

The default behavior of the MAC is to transfer the unmodified frame after stripping off the Preamble and SFD bytes. This behavior can be changed by setting bit [1] of the TX_RX_CTL register. When bit [1] is set, the Rx MAC strips the Preamble, SFD, FCS bytes and the PAD bytes, if any. Note that the Rx MAC assumes a received frame has PAD bytes if a 64 byte packet is received with its Length/Type field set to a value of less than 46 bytes.

Once the frame is ready to be written into the FIFO, the Rx MAC asserts the rx_write signal, then presents the data on the rx_dbout bus. The rx_write signal is asserted as long as the frame is being written. After transferring the entire frame into the FIFO, the Rx MAC asserts rx_eof indicating the end of the frame. If the frame is received with errors, rx_error is asserted along with rx_eof. If the frame is received with no errors, rx_error remains de-asserted. In either case, a rich set of statistics vectors is presented, containing information about the frame that was received. The statistics vector bus, rx_stat_vector, is qualified by the assertion of rx_stat_en.

If the RxFIFO becomes full, rx_fifo_full is asserted and the frame data is lost. Therefore, the FIFO full condition must be avoided at all times. The rx_fifo_error signal will be asserted along with rx_eof for all frames written into the FIFO while it is full.

The Rx MAC goes to the IDLE state when it is done receiving the frame. This is indicated by bit[10] of the TX_RX_STS register. If the Rx MAC is disabled while it is in the process of receiving a frame, it goes to the IDLE state after it completes the current frame reception.

Address Filtering

The Rx MAC offers several address filtering methods the user can employ to effectively block unwanted frames. It also provides a Promiscuous mode, in which all supported filtering schemes are abandoned and the Rx MAC transfers all the frames irrespective of the address they contain.

By default, the Rx MAC is configured to filter and discard Broadcast frames (i.e. all bits of the received DA == 1) and multicast frames (i.e. bit[0] of the received DA == 1). The MAC can be configured to receive broadcast frames by setting bit [7] of the TX_RX_CTL register.

Multicast frames are received only when bit [4] of the TX_RX_CTL register is set. When set, multicast frames are subject to filtering that is dependent on a 64-bit hash table lookup. The 64-bit hash table is organized as eight, 8-bit registers. The six middle bits of the most significant byte of the CRC calculated for the destination address field of the frame, are used to address one of the 64 bits of the hash table. The three most significant bits of the calculated CRC select one of the eight tables, and the three least significant bits select a bit. The frame is received only if the retrieved bit is set. The IP core registers specifying the hash tables contents are described in Internal Registers. An example of C programming language code that can be used to determine hash table contents based on the multicast addresses to be received is given in Code Listing for Multicast Bit Selection Hash Algorithm in C Language.

All other regular frames are filtered based on the Rx MAC address programmed into the MAC_ADDR_0, MAC_ADDR_1 and MAC_ADDR_2 registers.

Filtering Based on Frame Length

The default minimum Ethernet frame size is 64 bytes. Any frame smaller than 64 bytes could possibly be a collision fragment. By default, the Rx MAC is configured to ignore bytes shorter than 64 bytes. The user can configure the MAC to receive shorter frames by setting bit [8] of the TX_RX_CTL register. Whenever a short frame is received, the appropriate bit is set in the statistics vector, marking it as a Short frame.

The Rx MAC has been designed to receive frames larger than the standard specified maximum as easily as any other frame. This ensures the MAC can work in environments that can generate jumbo frames. However, for statistics purposes, the user can set the maximum length of the frame in the MAX_PKT_SIZE register. When a received frame is larger than the number in this register, bit [31] of the Receive Statistics Vector bus is set, marking it as a Long frame.

Receiving a PAUSE Frame

When the Rx MAC receives a PAUSE frame, the Tx MAC continues with the current transmission, then pauses for the duration indicated in the PAUSE time. During this time, the Tx MAC can transmit Control frames.

Although PAUSE frames may contain the Multicast Address, Multicast filtering rules do not apply to them. If bit [3] of the TX_RX_CTL register is set, the Rx MAC will signal the Tx MAC to stop transmitting for the duration specified in the frame. If this bit is reset, the Rx MAC assumes the Tx MAC does not have the PAUSE capability and/or does not wish to be paused and will not signal it to stop transmitting. If the drop control, bit[6] in the TX_RX_CTL register, is set then the PAUSE frame is received but dropped internal to the MAC and is not transferred to the client FIFO interface. Otherwise, the PAUSE frame is received and transferred to the FIFO.

Statistics Vector

By default, a Statistics Vector is generated for all received frames transferred to the external FIFO. If the user wants the Rx MAC to ignore all incoming frames, then the input signal ignore_next_pkt must be asserted. In this case, a frame that should have been received is ignored and the Rx MAC sets the Packet Ignored bit (bit 26) of the Statistics Vector.

The MAX_PKT_SIZE register is programmed by the user as a threshold for setting the Long Frame bit of the Statistics Vector. This value is used for un-tagged frames only. The Receive MAC will add “4” to the value specified in this register for all VLAN tagged frames when checking against the number of bytes received in the frame. This is because all VLAN tagged frames have an additional four bytes of data.

When a tagged frame is received, the entire VLAN tag field is stored in the VLAN_TAG register. Additionally, every time a statistics vector is generated, some of the bits are written into the corresponding bit locations [9:1] of the TX_RX_STS register. This is done so the user can get this information via the Host interface.

The description of the bits in the Statistics Vector bus is shown in Table 2-3.

Table 2-3. Receive Statistics Vector Descriptions

Bit	Description
31	**Long Frame**. This bit is set when a frame longer than specified in the **MAX_PKT_SIZE** register is received.
30	**Short Frame**. This bit is set when a frame shorter than 64 bytes is received.
29	**IPG Violation**. This bit is set when a frame is received before the IPG timer runs out (96 bit times).
28	**Not Used**. This bit always returns a zero.
27	**Carrier Event Previously Seen**. When asserted, indicates that a carrier event was detected since the last frame.
26	**Packet Ignored**. When set, this bit indicates the incoming packet is to be ignored.
25	**CRC Error**. This bit is set when a frame is received with an error in the CRC field.
24	**Length Check Error**. This bit is set if the number of data bytes in the incoming frame do not match the value in the length field of the frame.
23	**Receive OK**. This bit is set if the frame is received without any error.
22	**Multicast Address**. This bit is set to indicate the received frame contains a Multicast Address.
21	**Broadcast Address**. This bit is set to indicate the received frame contains a Broadcast Address.
20	**Dribble Nibble**. This bit is set when only four bits of the data presented on the RS interface are valid.
19	**Unsupported Opcode**. This bit is set if the received control frame has an unsupported opcode. In this version of the IP, only the opcode for PAUSE frame is supported.
18	**Control Frame**. This bit is set to indicate that a Control frame was received.
17	**PAUSE Frame**. This bit is set when the received Control frame contains a valid PAUSE opcode.
16	**VLAN Tag Detected**. This bit is set when the TSMAC IP core receives a VLAN Tagged frame.
15:0	**Frame Byte Count**. This bus contains the length of the frame that was received. The frame length includes the DA, SA, L/T, TAG, DATA, PAD and FCS fields.

Transmit MAC (Tx MAC)

The Tx MAC is responsible for controlling access to the physical medium. The Tx MAC reads data from an external Tx FIFO when the FIFO is not empty and it detects an active tx_fifoavail. The Tx MAC then formats this data into an Ethernet packet and passes it to the G/MII module.

The Tx MAC is disabled while Tx_en is low (Bit_3 of the MODE register) and should only be enabled after the associated registers are properly initialized. Once enabled, the Tx MAC will continuously monitor the FIFO interface for an indication that frame(s) are ready to be transmitted. In the 1G mode, Tx MAC and the Tx FIFO interface operations are synchronous to txmac_clk derived from sys_clk. In the 10/100 mode, the Tx MAC is clocked by txmac_clk (derived from the tx_clk supplied from the PHY device). The Tx FIFO interface signals are always synchronous to txmac_clk.

In 10/100 mode, the Tx MAC can be configured to operate in the half-duplex or full-duplex mode. This is done by writing to bit[5] of the TX_RX_CTL register. In full-duplex operation, it is possible for the receiver's buffer to fill up rapidly. In such cases, the receiver sends flow control (PAUSE) frames to the transmitter, requesting that it stop transmitting frames. When the receiver is able to free the buffers, the transmitter completes transmitting the current frame and stops for the duration specified in the PAUSE frame.

Transmitting Frames

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 bit[2] of the Tx_RX_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 the user's responsibility to ensure that short frames are properly padded before the FCS is generated. If the MAC receives a frame 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 FIFO interface and then encapsulated into an Un-tagged Ethernet frame. This frame is not sent over the network until the network has been idle for a minimum of Inter-Packet Gap (IPG) time. 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 padding is not disabled, all short frames are padded with hexadecimal 00.

The input signal tx_fifoeof is asserted along with the last set of data transfer to indicate the end of the frame. 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. If the MAC is able to transmit the frame without any errors, the tx_done signal is asserted. Once the transmission has ended, data on the tx_stat_vector bus is presented to the host, including all the statistical information collected in the process of transmitting the frame. Data on this bus is qualified by assertion of the tx_staten signal.

After the Transmit MAC is done transmitting a frame, it waits for more frames from the FIFO 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 will completely transmit 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.

External Transmit FIFO

The interface between the Tx MAC and the external, client side FIFO is eight bits wide. The bit presented on position 0 is transmitted first and the bit in position 7 is transmitted last. In other words, bit[0] will be transmitted on the txd[0] signal of GMII while the bit[7] will be transmitted on txd[7].

Logic inside the MAC signals if the frame ready for transmission at the head of the FIFO is a Control frame. This is done so the Tx MAC can continue transmission of a Control frame while it is paused.

FIFO Under-flow

If a FIFO underflow occurs, the FIFO logic must assert tx_fifoempty. If at least 64 bytes have been transmitted, the Tx MAC aborts the transmission by asserting tx_er. In addition, the Tx MAC inserts erroneous CRC bits into the packet to guarantee the receiver will detect the error in the packet. If less than 64 bytes have been transmitted when the FIFO underflow occurs, the MAC will pad the remaining bytes before ending the transmission. In either case, the MAC asserts tx_discfrm indicating an error during transmission.

Transmitting PAUSE Frame

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 FIFO. In the other method, the application layer signals the Tx MAC directly to transmit a PAUSE frame. This is accomplished by asserting tx_sndpausreq. In this case the Tx MAC will complete transmission of the current packet and then transmit a PAUSE frame with the PAUSE time value supplied through the tx_sndpaustim bus.

Retries on Collision

When operating in the half-duplex mode, the Transmit MAC has the capability to perform re-transmission of frames that have experienced in-window collision up to the specified maximum. This is possible because the MAC always buffers the first 64 bytes of the frame.

If the MAC has been disabled while it is backing off (soon after a collision), it will only return to the IDLE state after it has successfully transmitted the frame or has exceeded the retry limit.

In the 10/100 mode, the Tx MAC provides the following information:

- Whether the frame deferred before transmission
- The number of times the frame experiences collision before transmission.

This information is sent as a part of the statistics vector. For a frame transmitted without any errors, the statistics vector, qualified by the enable signal, is asserted along with the tx_done signal.

When the frame experiences excessive collision or late collision, the statistics bit for the appropriate condition is set and the tx_discfrm signal is asserted. This indicates an error condition.

Internal Data Buffer and FIFO Interfaces

In the Classic TSMAC IP core and SGMII Easy Connect option, the transmit and receive sections each contain internal FIFO buffers. In the Gigabit MAC option, only the Rx internal FIFO buffer is used. Note that External Transmit and Receive FIFOs (that interface to the MAC client side) are still required to store variable-length normal packets.

On the receive side, the internal FIFO buffer is used to support the dropping of packets less than 64 bytes and to provide additional data buffering for normal packets. The core provides a feature where the user can block all the frames that are shorter than the minimum frame length of 64 bytes in the TSMAC IP core itself (for example collision fragments, runt frames, and such). This prevents these frames from reaching the user's application.

On the transmit side, the internal FIFO buffer stores the first 64 bytes of the frame. This ensures that the TSMAC IP core can re-transmit the frame automatically without help from the application software during an in-window collision. This important feature prevents the propagation of collision information into the application software.

The TSMAC IP core provides two independent interfaces for use with external Transmit and Receive FIFOs. This feature enables the TSMAC IP core to support full duplex operation in either 10/100 or 1G modes.

G/MII Interface

The G/MII module uses the clock supplied by the external PHY. The core implements the standard G/MII interface to connect to the PCS layer.

The module implementing the interface also converts the data to a format usable by the MAC. In the 1G mode, the 8-bit data at the interface is presented to the 8-bit data path of the MAC. In the 10/100 mode, the 4-bit MII data is packed and input to the 8-bit data path of the MAC.

Although not implemented as a separate module, the Reconciliation Sub-layer is implemented as a part of the G/MII interface. This module is responsible for passing the data from one clock domain (TSMAC IP core) to the other G/MII.

(Optional) Media Independent Interface Management Module (MIIM)

The MIIM accesses management information from the PHY device and writes to or reads from the PHY registers. A single MIIM can address up to 32 PHY devices. This module runs off its own clock called mdc. The standard specifies this clock to be at 2.5MHz, but PHY devices can accept a 10MHz mdc clock. Therefore, the TSMAC IP core can have a MIIM that is capable of running at up to 10MHz.

The MIIM read or write operations are specified in the GMII_MNG_CTL register. This register also specifies the addressed PHY and the register within the PHY that needs to be accessed. The Command Finished bit in the GMII_MNG_CTL register is reset as soon as a command to read or write is given. It is set only when the MIIM module completes the operation. While the interface is busy, the GMII_MNG_CTL register cannot be overwritten, and all write operations to the register are ignored. For a write operation, the data to be written is stored in the GMII_MNG_DAT register. For a read operation, the data read from the addressed PHY is stored in this register. The ready bit in the GMII_MNG_CTL is set at the end of the read/write operation.

Internal Registers

The TSMAC IP core internal registers are initialized through the generic Host Interface. These rules apply when accessing the internal registers:

- With the 8-bit Host Interface, the individual bytes of the registers are accessed through their corresponding addresses, with the lower address pointing to the lower byte.
- The reserved bits should be programmed to 0. These bits are invalid, and should be discarded when read.
- All registers except the MODE and GMII Management registers can be written into only when the core is disabled, i.e., MAC is in the IDLE state (Tx_en and Rx_en low in the MODE register). The MODE and GMII Management registers are the only registers that can be written to after the TSMAC IP core is no longer disabled.

Table 2-4 lists the TSMAC IP core registers accessible via the Host Interface. The registers are either Read/Write (R/W) or Read Only (RO) for status reporting purposes. The values of the registers immediately after the Reset Condition is removed from the TSMAC IP core (POR Value in Hexadecimal format) are also given.

Table 2-4. TSMAC IP Core Internal Registers

Register Description	Mnemonic	I/O Address	POR Value
Mode register	MODE	00H - 01H	0000H
Transmit and Receive Control register	TX_RX_CTL	02H - 03H	0000H
Maximum Packet Size register	MAX_PKT_SIZE	04H - 05H	05EEH
Inter-Packet Gap register	IPG_VAL	08H - 09H	000CH
TSMAC IP core Address register 0	MAC_ADDR_0	0AH - 0BH	0000H
TSMAC IP core Address register 1	MAC_ADDR_1	0CH - 0DH	0000H
TSMAC IP core Address register 2	MAC_ADDR_2	0EH - 0FH	0000H
Transmit and Receive Status	TX_RX_STS	12H - 13H	0000H
GMII Management Interface Control register	GMII_MNG_CTL	14H - 15H	0000H
GMII Management Data register	GMII_MNG_DAT	16H - 17H	0000H
VLAN Tag Length/type register	VLAN_TAG	32H - 33H	0000H
Multicast_table_0	MLT_TAB_0	22H - 23H	0000H
Multicast_table_1	MLT_TAB_1	24H - 25H	0000H
Multicast_table_2	MLT_TAB_2	26H - 27H	0000H
Multicast_table_3	MLT_TAB_3	28H - 29H	0000H
Multicast_table_4	MLT_TAB_4	2AH - 2BH	0000H
Multicast_table_5	MLT_TAB_5	2CH - 2DH	0000H
Multicast_table_6	MLT_TAB_6	2EH - 2FH	0000H
Multicast_table_7	MLT_TAB_7	30H - 31H	0000H
Pause_opcode	PAUS_OP	34H - 35H	0080H

Register Descriptions

MODE (R/W)

Mnemonic: MODE
POR Value = 0000H

Name	Range	Description
Rsvd	15:4	**Reserved**.
Tx_en	3	**Transmit Enable**. When this bit is set, the Tx MAC is enabled to transmit frames. When reset, the Tx MAC completes transmission of the packet currently being processed, then stops.
Rx_en	2	**Receive Enable**. When this bit is set, the Rx MAC is enabled to receive frames. When reset, the Rx MAC completes reception of the packet currently being processed, then stops.
FC_en	1	**Flow-control Enable**. When set, this bit enables the flow control functionality of the Tx MAC. This bit should be set to enable the Tx MAC to transmit a PAUSE frame via the tx_sndpausreq and tx_sndpaustim[15:0] MAC input ports.
Gbit_en	0	**Gigabit Enable**. For the Classic Tri-speed MAC option, in order to operate in GbE mode, this bit must be set high. For 10/100 mode, this bit must be set low. For the SGMII Easy Connect MAC option, this bit does not control anything (note the MAC operation speed is determined by the clock enables provided by the SGMII IP core). This bit will echo back what is written to it. For the Gigabit MAC option, this bit is a read-only bit and will always read back as logic high. This bit does not control anything in the core. Note, the state of this bit is useful for system use, since the cpu_if_gbit_en output signal, from the core, always reflects the state of this register bit.

Transmit and Receive Control (R/W)

Mnemonic: TX_RX_CTL
POR Value = 0000H

This register can be overwritten only when the Rx MAC and the Tx MAC are disabled. This register controls the various features of the MAC.

Name	Range	Description
Rsvd	15:9	**Reserved**.
Receive_short	8	**Receive Short Frames**. When high, enables the Rx MAC to receive frames shorter than 64 bytes.
Receive_brdcst	7	**Receive Broadcast**. When high, enables the Rx MAC to receive broadcast frames
Drop_control	6	**Drop control**. When high, received pause control frames are dropped internal to the MAC and not transferred to the external Rx client FIFO.
Hden	5	**Half-duplex Enable** (10/100 mode only). When high, configures the Tx MAC to operate in half-duplex mode.
Receive_mltcst	4	**Receive Multicast**. When high, the multicast frames will be received per the filtering rules for such frames. When low, no Multicast (except PAUSE) frames will be received.
Receive_pause	3	**Receive PAUSE**. When set, the Rx MAC will indicate the Rx PAUSE frame reception to the Tx MAC and thereby cause the Tx MAC to pause sending data frames for the period specified within the Rx PAUSE frame. Note this indication is independent of the Drop_control bit setting.
Tx_dis_fcs	2	**Transmit Disable FCS**. When set, the FCS field generation is disabled in the Tx MAC.
Discard_fcs	1	**Rx Discard FCS and Pad**. When set, the FCS and any of the padding bytes of an IEEE 802.3 frame are stripped off the frame before it is transferred to the Rx FIFO. When low, the entire frame is transferred into the Rx FIFO. Note: Discarding padding bytes is only applicable to pure IEEE 802.3 frames (such as in backplane applications) and will not function on Ethernet frames (IP, UDP, ICMP, etc.) where the length field is now interpreted as a protocol type field.
Prms	0	**Promiscuous Mode**. When asserted, all filtering schemes are abandoned and the Rx MAC receives frames with any address.

Maximum Packet Size (R/W)

Mnemonic: MAX_PKT_SIZE
POR Value = 05EEH (1518 decimal)

This register can be overwritten only when the MAC is disabled. All frames longer than the value (number of bytes) in this register will be tagged as long frames.

Name	Range	Description
Max_frame	15:0	Maximum size of the packet than can be handled by the core.

IPG (Inter-Packet Gap) (R/W)

Mnemonic: IPG_VAL
POR Value = 000CH

Name	Range	Description
Rsvd	15:5	Reserved.
IPG	4:0	Inter-packet gap value in units of (byte time).

MAC Address Register {0,1,2} (R/W), Set of Three

Mnemonic: MAC_ADDR
POR Value = 0000H

The MAC Address Registers 0-2 contain the Ethernet address of the port. The MAC Address Register [0] has the two bytes that are transmitted first and the MAC Address Register [2] has the two bytes that are transmitted last. Bit[8] through Bit[15] are transmitted first while bit[0] through bit[7] are transmitted last.

Note that the MAC address is stored in the registers in Hexadecimal form. For example, setting the MAC Address to AC-DE-48-00-00-80 would require writing 0xAC (octet 0) to address 0x0B (high byte of Mac_addr[15:0]), 0xDE (octet 1) to address 0x0A (Low byte of Mac_addr[15:0]), 0x48 (octet 2) to address 0x0D (high byte of Mac_addr[15:0]), 0x00 (octet 3) to address 0x0C (Low byte of Mac_addr[15:0]), 0x00 (octet 4) to address 0x0F (high byte of Mac_addr[15:0]), and 0x80 (octet 5) to address 0x0E (Low byte of Mac_addr[15:0]). Note Octet 0 is transmitted first and Octet 5 is transmitted last.

Name	Range	Description
Mac_addr	15:0	Ethernet address assigned to the port supported by the TSMAC IP core.

Transmit and Receive Status (RO)

Mnemonic: TX_RX_STS
POR Value = 0000H

This register reports events that have occurred during packet reception and transmission.

Name	Range	Description
Rsvd	15:11	**Reserved**.
Rx_idle	10	**Receive MAC Idle**. Receive MAC in idle condition used to reset configurations by CPU interface.
Tagged_frame	9	**Tagged Frame**. Tagged frame received.
Brdcst_frame	8	**Broadcast Frame**. Indicates that a Broadcast packet was received.
Multcst_frame	7	**Multicast Frame**. Indicates that a Multicast packet was received.
IPG_shrink	6	**IPG Shrink**. Received frame with shrunk IPG (IPG < 96 bit time).
Short_frame	5	**Short Packet**. Indicates that a packet shorter than 64 bytes has been received.
Long_frame	4	**Too Long Packet**. Indicates receipt of a packet longer than the maximum allowable packet size specified in the **MAX_PKT_SIZE** register.
Error frame	3	**Rx_er Asserted**. Indicates the frame was received with the rx_er signal asserted.
CRC	2	**CRC Error**. Indicates a packet was received with a CRC error.
Pause_frame	1	**PAUSE Frame**. Indicates a PAUSE frame was received.
Tx_idle	0	**Transmit MAC Idle**. Transmit MAC in idle condition, used to reset configurations by CPU interface.

VLAN Tag (RO)

Mnemonic: VLAN_TAG
POR Value = 0000H.

The VLAN tag register has the VLAN TAG field of the most recent tagged frame that was received. This is a read only register.

Name	Range	Description
VLAN	15:0	This field defines length/type of field of the VLAN tag when inserted into transmitted frames.

GMII Management Register Access Control (R/W)

Mnemonic: GMII_MNG_CTL
POR Value = 0000H.

The GMII Management Access register controls the Management Interface Module. This register can be overwritten only when the interface is not busy. A write operation will be ignored when the interface is busy.

Name	Range	Description
Rsvd	15	**Reserved**.
Cmd_fin	14	**Command Finished**. When high, it means the interface has completed the intended operation. This bit is set to 0 when the interface is busy.
RW_phyreg	13	**Read/Write PHY Registers** • When '1' -> write operation • When '0' -> read operation
Phy_add	12:8	**GMII PHY Address**. The address of the accessed PHY Bit 12 is the most significant bit, and it is the first PHY address bit to be transmitted and received.
Rsvd	7:5	**Reserved**.
Reg_add	4:0	**GMII Register Address**. The address of the register accessed. Bit 4 is the most significant bit and is the first register address bit to be transmitted or received.

GMII Management Access Data (R/W)

Mnemonic: GMII_MNG_DAT
POR Value = 0000H.

The contents of this register will be transmitted when a write operation is to be performed. When a read operation is performed, this register will contain the value that was read from a PHY register. This register should be read only after the cmd_fin bit in the control register is set.

Name	Range	Description
GMII_dat	15:0	**GMII Data**. Bit 15 is the most significant bit, corresponding to bit 15 of the accessed register.

Multicast Tables (R/W), Set of Eight

Mnemonic: MLT_TAB_[0-7]
POR Value = 0000H.

When the core is programmed to receive multicast frames, a filtering scheme is used to decide whether the frame should be received or not. The six middle bits of the most significant byte of the CRC value, calculated for the destination address, are used as a key to the 64-bit hash table. The three most significant bits select one of the eight tables, and the three least significant bits select a bit. The frame is received only if this bit is set.

Name	Range	Description
Multicast_table_[0-7]	7:0	**Multicast Table**. Eight tables that make a 64-bit hash.

Pause Opcode (R/W)

Mnemonic: PAUS_OP
POR Value = 0001H

This register contains the PAUSE Opcode, This will be compared against the Opcode in the received PAUSE frame. This value will also be included in any PAUSE frame transmitted by the TSMAC IP core. Bit 15 is transmitted first and bit 0 is transmitted last.

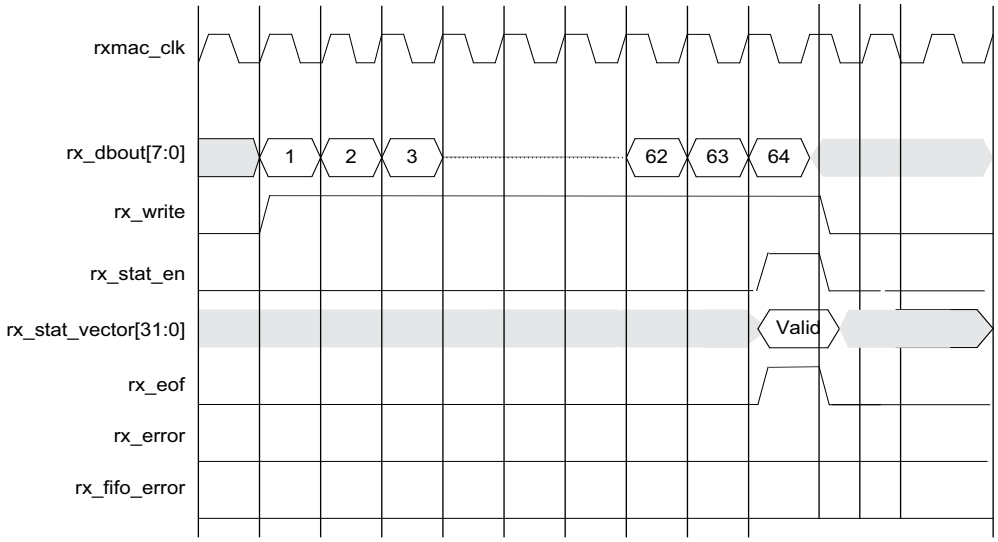
Name	Range	Description
Pause_OpCode	15:0	**PAUSE Opcode**.

Timing Specifications

This section contains operational timing diagrams applicable to the TSMAC IP core interfaces.

Reception of a 64-Byte Frame Without Error -Rx MAC Application Interface

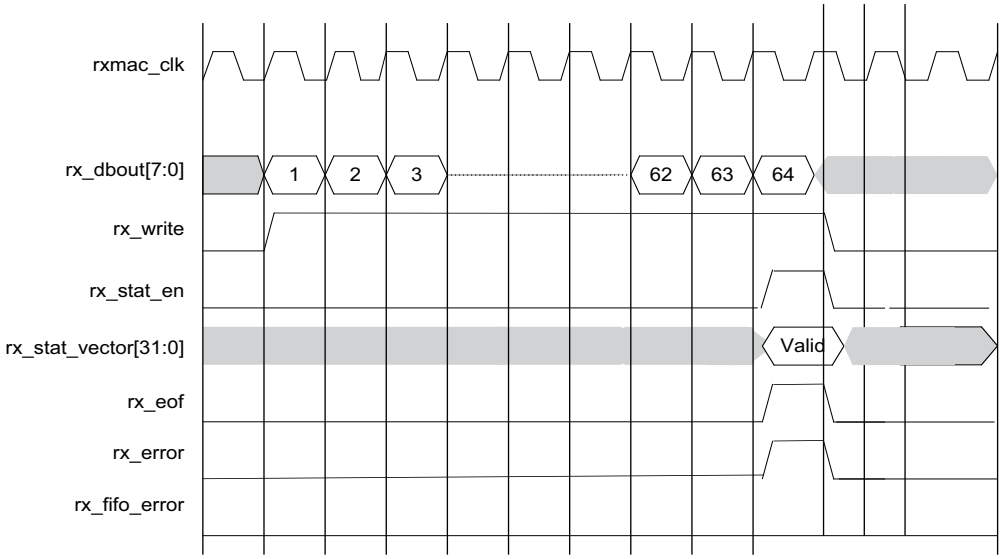
Figure 2-9. Reception of a 64-byte Frame Without Error



Reception of a 64-byte Frame with Error(s) - Rx MAC Application Interface

The signal rx_error is asserted to indicate that the 64-byte frame was received with error(s).

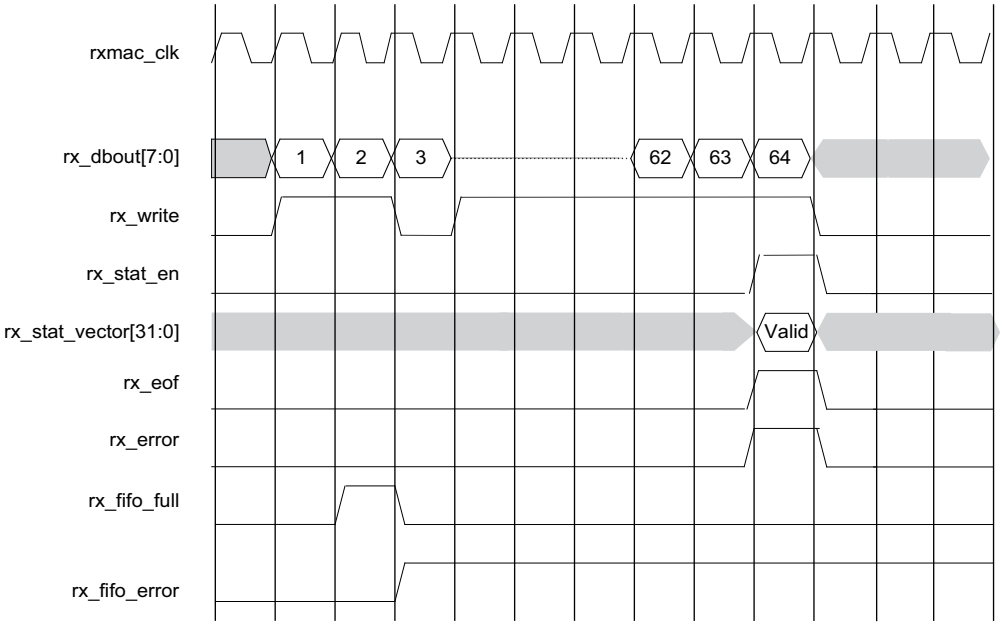
Figure 2-10. Reception of a 64-byte Frame with Error



Reception of a 64-Byte Frame with FIFO Overflow - Rx MAC Application Interface

The FIFO writing operation is suspended whenever an overflow condition occurs. When this condition occurs, the TSMAC IP core asserts rx_fifo_error. This signal should be sampled along with rx_eof in order to process the error condition.

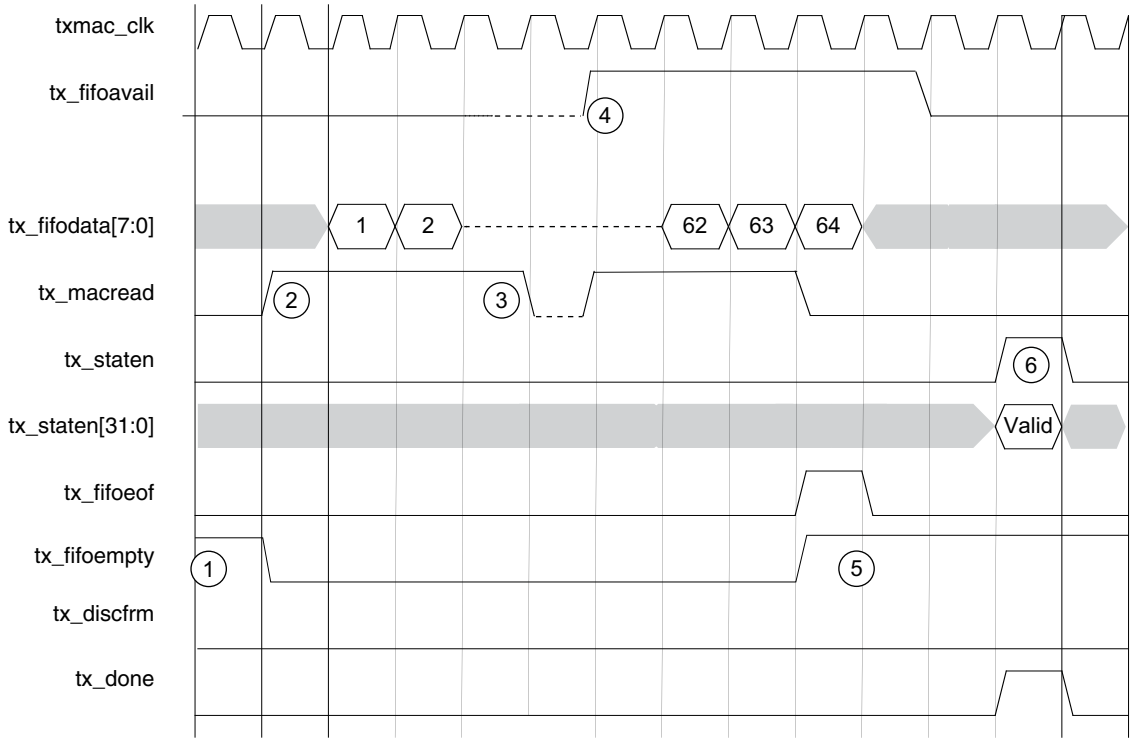
Figure 2-11. Reception of a 64-byte Frame with FIFO Overflow



Successful Transmission of a 64-Byte Frame -Tx MAC Application Interface

The assertion of tx_fifoavail indicates a frame is ready to be transmitted. It does not trigger the MAC transmit process. The MAC actually pre-reads the FIFO as soon as tx_fifoempty goes low (indicating data is present in the Tx FIFO). The tx_fifoempty signal, the tx_fifoavail signal and the MAC Tx state machine all determine when to read the Tx FIFO. The TSMAC IP core reads the FIFO and the data is transmitted until tx_fifoeof is asserted. Once the frame is transmitted, tx_staten is asserted to qualify the statistic vector, tx_statvec. The signal tx_done is asserted to indicate a successful transmission. This is shown in Figure 2-12, and described as follows in detail.

Figure 2-12. Transmission of a 64-Byte Frame without Error



1. The Tx FIFO is initially empty (tx_fifoempty is asserted) and the MAC Tx state machine is in an idle state.
2. The client interface begins loading the Tx FIFO. Once the tx_fifoempty signal is de-asserted the tx_macread is asserted and the MAC Tx state machine goes into a transmit state. The MAC performs a “pre-read” and continues to assert the tx_macread as long as the Tx FIFO is not empty (tx_fifoempty stays low). The pre-read typically is 7 to 8 bytes.
3. If tx_fifoavail is low after the pre-read, the Tx MAC will de-assert the tx_macread until the tx_fifoavail is asserted.
4. tx_fifoavail is set high by the client interface indicating all bytes of the packet are now available for transmission. The Tx MAC re-asserts tx_macread and reads all bytes until tx_fifoeof indicates the last byte.
5. Once the complete packet is read out by the Tx MAC, it checks the status of tx_fifoempty and tx_fifoavail again. If either tx_fifoavail is low or tx_fifoempty is high at the end of the packet, the MAC de-asserts tx_macread. If tx_fifoavail is high and tx_fifoempty is low, (FIFO is not empty) the Tx MAC will do another pre-read (not shown in this figure) and start the transmit process over.
6. Once the frame is transmitted, tx_staten is asserted to qualify the statistic vector, tx_statvec. The signal tx_done is asserted to indicate a successful transmission.

Additional Notes:

- The MAC Tx state machine may do other processing, such as inserting IPG after a previous packet is transmitted, so even if tx_fifoempty is low and tx_fifoavail is high the MAC may not assert the tx_macread right away.
- There are many ways to interface client logic and FIFOs to the MAC - here are two examples:

1.) An appropriate threshold level for the Tx FIFO is chosen which indicates that a full frame is ready to be sent. This is used as the tx_fifoavail signal. If there are variable size packets (some small) one may want to OR this threshold with a frames_present signal. frames_present is a frames counter that keeps track of EOPs into and out of the FIFO. Whenever the FIFO is Not Almost Empty and there is at least one full packet in the FIFO tx_fifoavail is set high to indicate it is safe to transmit a packet.

2.) Decouple the tx_fifoempty and tx_fifoavail signals from the actual Tx FIFO and generate tx_fifoavail and tx_fifoempty signals from a simple state machine:

I) Keep tx_fifoavail low and tx_fifoempty high - load Tx FIFO

II) When packet is fully loaded into Tx FIFO assert tx_fifoavail high and de-assert tx_fifoempty (set to not empty)

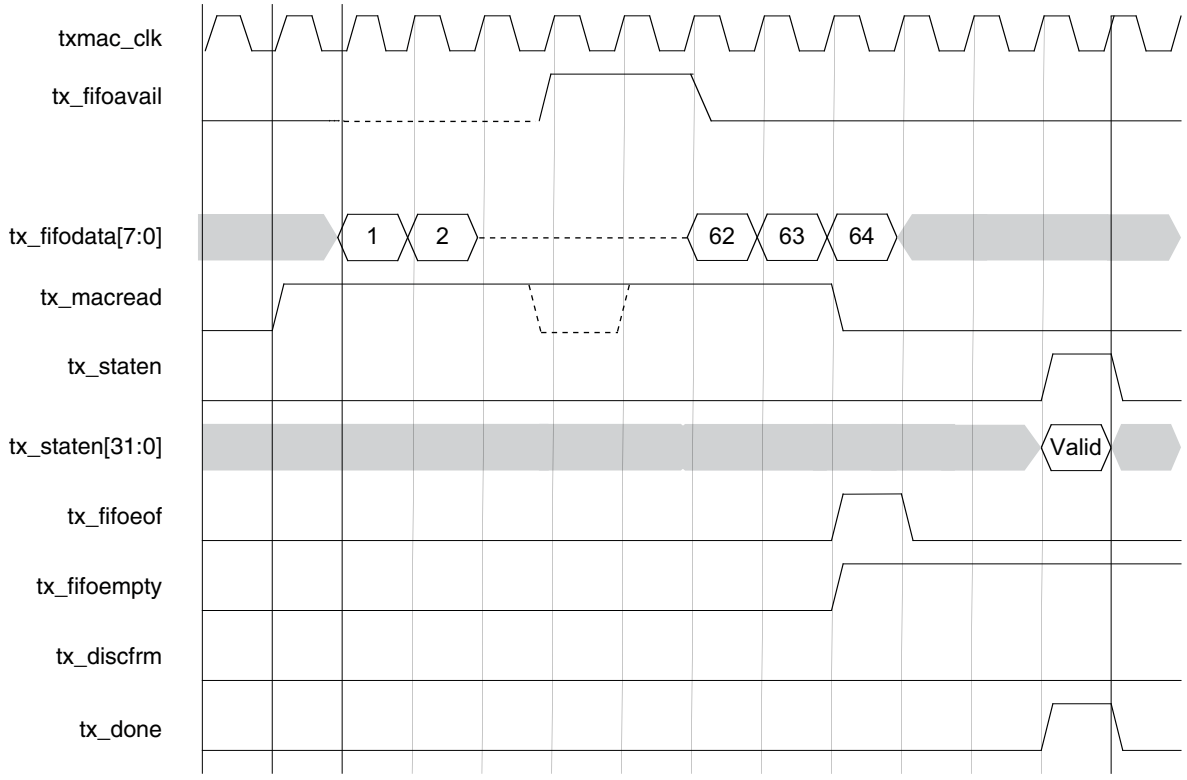
III) When packet is completely pulled out of FIFO, de-assert tx_fifoavail and force tx_fifoempty high

Successful Transmission of a 64-byte Frame with FIFO Empty - Tx MAC Application Interface

tx_fifoempty is asserted along with tx_fifoeof to indicate that the complete 64-byte frame has been read.

The frame is transmitted as a valid frame and tx_done is asserted at the end of transmission.

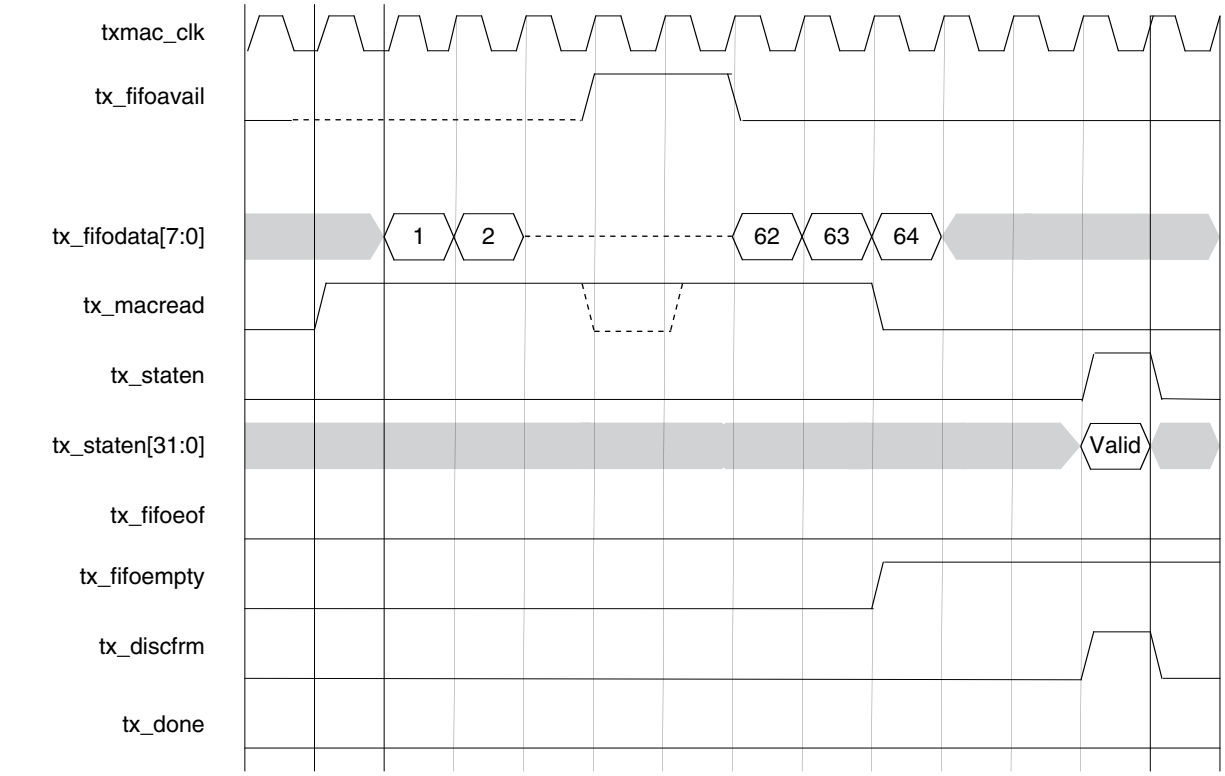
Figure 2-13. Successful Transmission of a 64-byte Frame with FIFO Empty



Aborted Transmission Due to FIFO Empty - Tx MAC Application Interface

If the tx_fifoempty is asserted while the Tx MAC is in the process of reading a frame, the MAC will stop reading the frame and assert tx_disfrm to indicate an erroneous transmission. The frame transmission is abandoned when this occurs. Note in Figure 2-14 the tx_fifoempty was asserted before the end of the frame (no tx_fifoeof asserted).

Figure 2-14. Aborted Transmission Due to FIFO Empty



Reception and Transmission of 64-byte Frames With SGMII Easy Connect Option

Figure 2-15 and Figure 2-16 illustrate how clock enables are used on the MAC client interface when the MAC is configured with SGMII Easy connect option. Note: the SGMII PCS IP sources the appropriate clock enables to the MAC. For one Gbps operation the clock enables are always high, for 100 Mbps operation the clock enables are asserted high once every 10 (125 Mhz) clock cycles, and for 10 Mbps operation the clock enable is asserted high once every 100 (125 Mhz) clock cycles.

Figure 2-15. Transmission of a 64-byte Frame at 100 Mbs Data Rate

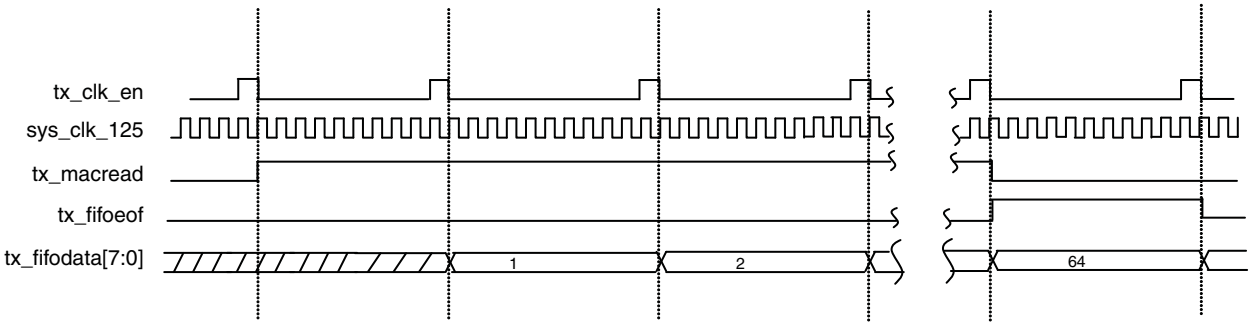
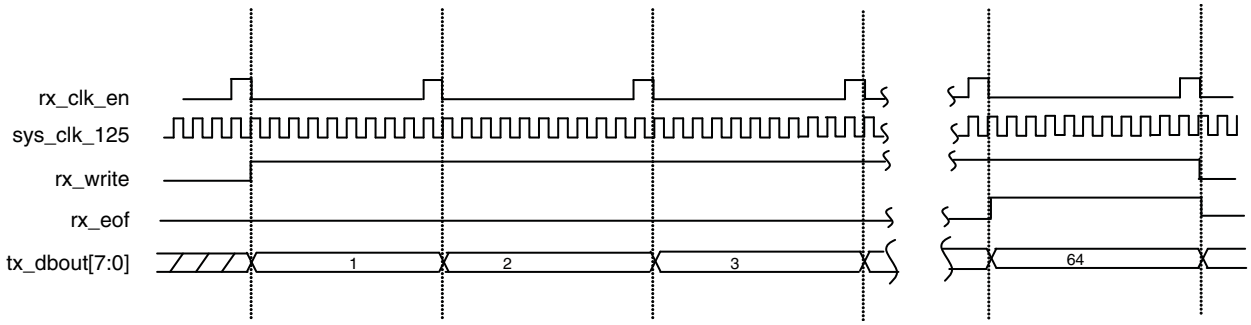


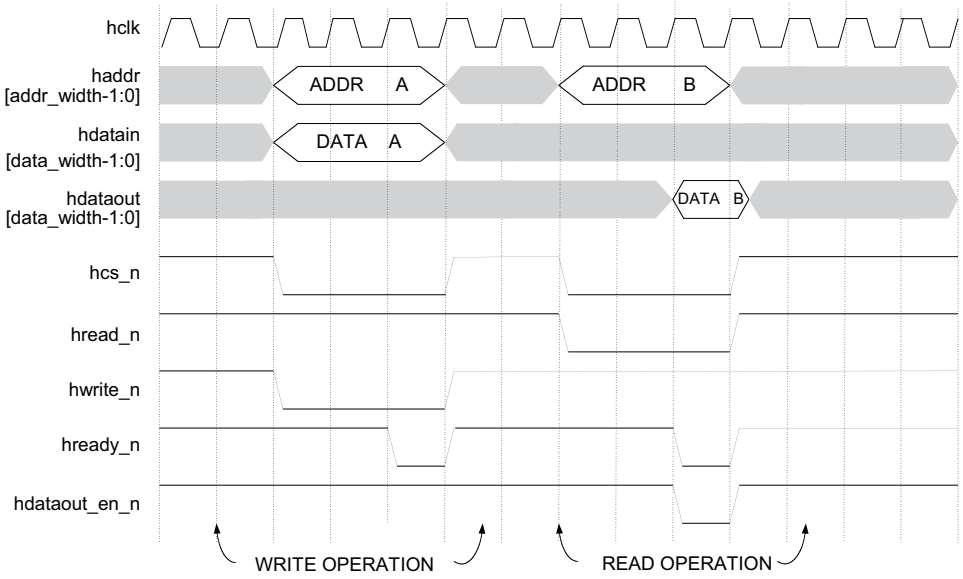
Figure 2-16. Reception of a 64-byte Frame at 100 Mbs Data Rate



Host Interface Read/Write Operation

During a write operation, haddr associated with hdatain, hcs_n and hwrite_n performs a write operation to an internal register. The end of transaction is indicated by assertion of hready_n. During a read operation, haddr associated with hcs_n and hread_n forms a write operation. The end of transaction is indicated by the assertion of hready_n and hdataout_en_n along with the valid read data on hdataout.

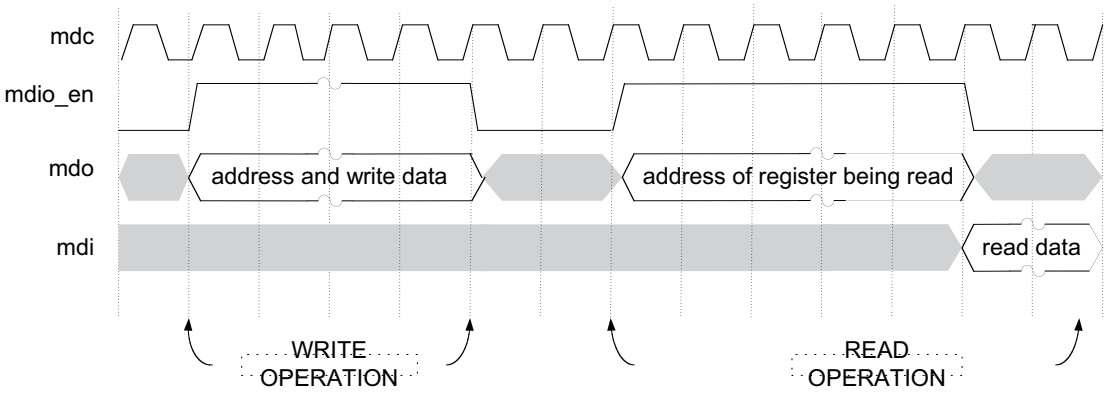
Figure 2-17. Host Interface Read/Write Operation



Management Interface Read/Write Operation

During a write operation, mdio_en is asserted and the data is transmitted on mdo. During a read operation, mdio_en is asserted while the address is being transferred. Once this is done, it is de-asserted for rest of the transfer enabling the PHY to deliver data on mdi.

Figure 2-18. Management Interface Read and Write Operations

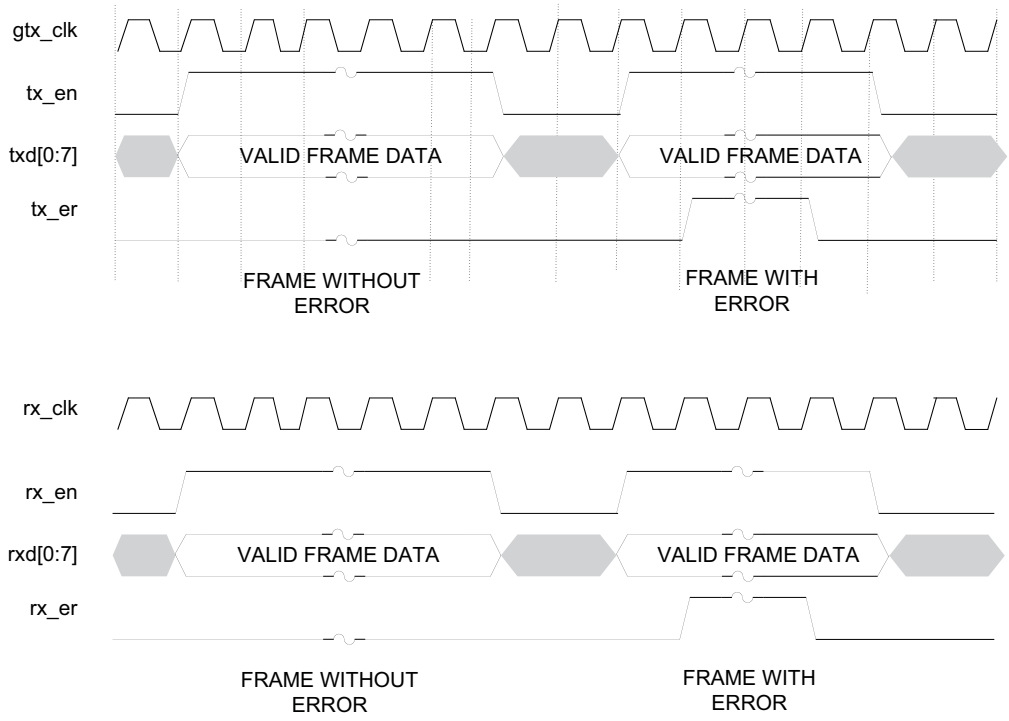


GMII Transmit and Receive Operations (1G mode and Gigabit MAC Option)

txd and tx_en are driven synchronous to the gtx_clk during transmit operations. When the frame being transmitted has an error, tx_er is asserted.

When receiving data, rxd and rx_en are sampled on the rising edge of rx_clk. An error in the frame is indicated when rx_er is asserted.

Figure 2-19. GMII Transmit and Receive Operations



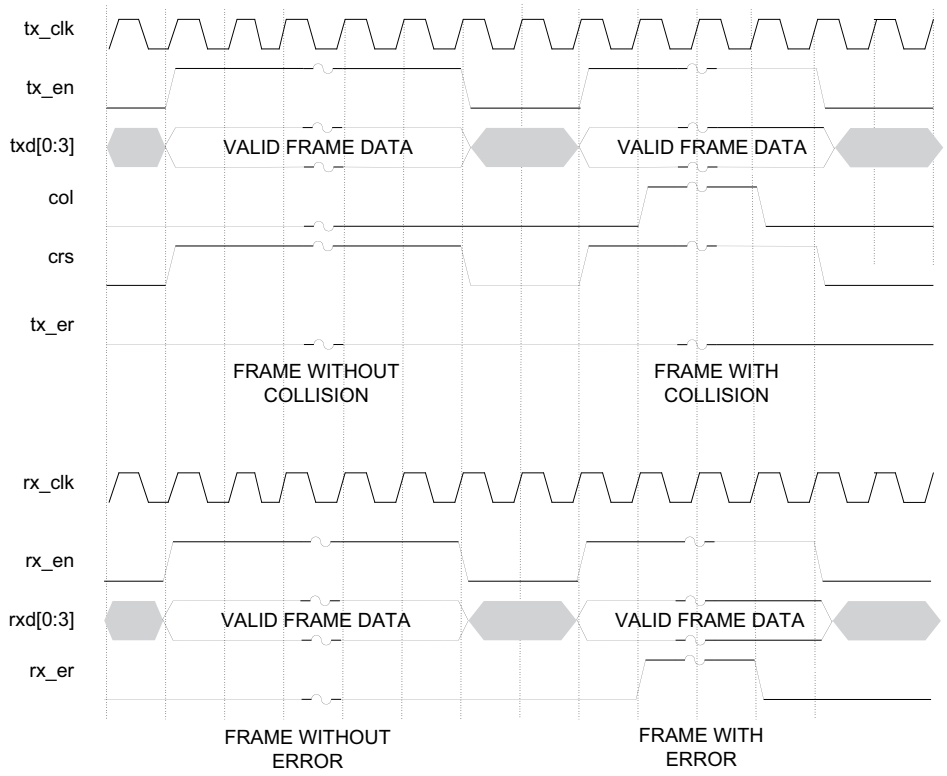
MII Transmit and Receive Operations (10/100 Mode)

On the transmit side, txd and tx_en are driven synchronous to the gtx_clk. tx_er is asserted to indicate that the frame being transmitted has an error.

On the receive interface, rxd and rx_en are sampled on the rising edge of rx_clk. An error in the frame is indicated when rx_er is high when sampled on the rising clock edge.

col and crs are asynchronous signals, useful in the half-duplex mode only.

Figure 2-20. MII Transmit and Receive Operations



The IPexpress™ tool is used to create IP and architectural modules in the Diamond software. Refer to IP Core Generation and Evaluation for a description on how to generate the IP.

Table 3-1 provides the list of user configurable parameters for the TSMAC IP core. The parameter settings are specified using the TSMAC IP core Configuration GUI in IPexpress.

Table 3-1. TSMAC IP Core Configuration Parameters

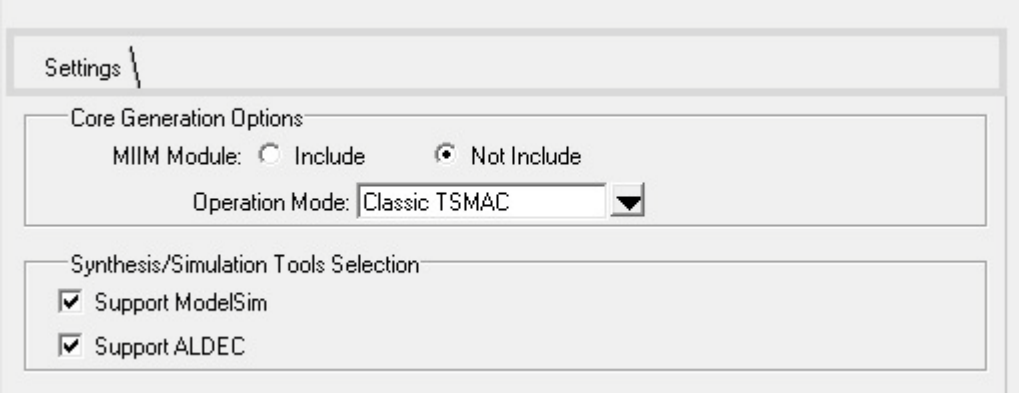
Parameter	Range/Options	Default
CLASSIC_TSMAC[1]	Selected/Not Selected	Selected
SGMII_TSMAC[1]	Selected/Not Selected	Not Selected
GBE_MAC[1]	Selected/Not Selected	Not Selected
MIIM_MODULE	Include/Do Not Include	Include

1. The first three parameters are mutually exclusive

TSMAC Configuration Dialog Box

Figure 3-1 shows the TSMAC Configuration dialog box.

Figure 3-1. TSMAC Configuration Dialog Box



Parameter Descriptions

This section describes the available parameters for the TSMAC IP core.

MIIM_MODULE

This parameter determines whether the optional MIIM Module will be included in the core's implementation.

Operation Mode

CLASSIC_TSMAC

This parameter determines whether the core will be implemented as a Classic TSMAC IP core.

SGMII_TSMAC

This parameter determines whether the core will be implemented as a TSMAC with the SGMII Easy Connect GMII interface.

GBE_MAC

This parameter determines whether the TSMAC IP core will be implemented as a Gigabit MAC core.

IP Core Generation in IPexpress

This chapter provides information on how to generate the Lattice TSMAC IP core using the Diamond software IPexpress tool, and how to include the core in a top-level design.

Licensing the IP Core

An IP core- and device-specific license is required to enable full, unrestricted use of the TSMAC IP core in a complete, top-level design. Instructions on how to obtain licenses for Lattice IP cores are given at:

http://www.latticesemi.com/Products/DesignSoftwareAndIP.aspx

Users may download and generate the TSMAC IP core and fully evaluate the core through functional simulation and implementation (synthesis, map, place and route) without an IP license. The TSMAC IP core also supports Lattice's IP hardware evaluation capability, which makes it possible to create versions of the IP core that operate in hardware for a limited time (approximately four hours) without requiring an IP license. See "Hardware Evaluation" on page 51 for further details. However, a license is required to enable timing simulation, to open the design in the Diamond EPIC tool, and to generate bitstreams that do not include the hardware evaluation timeout limitation.

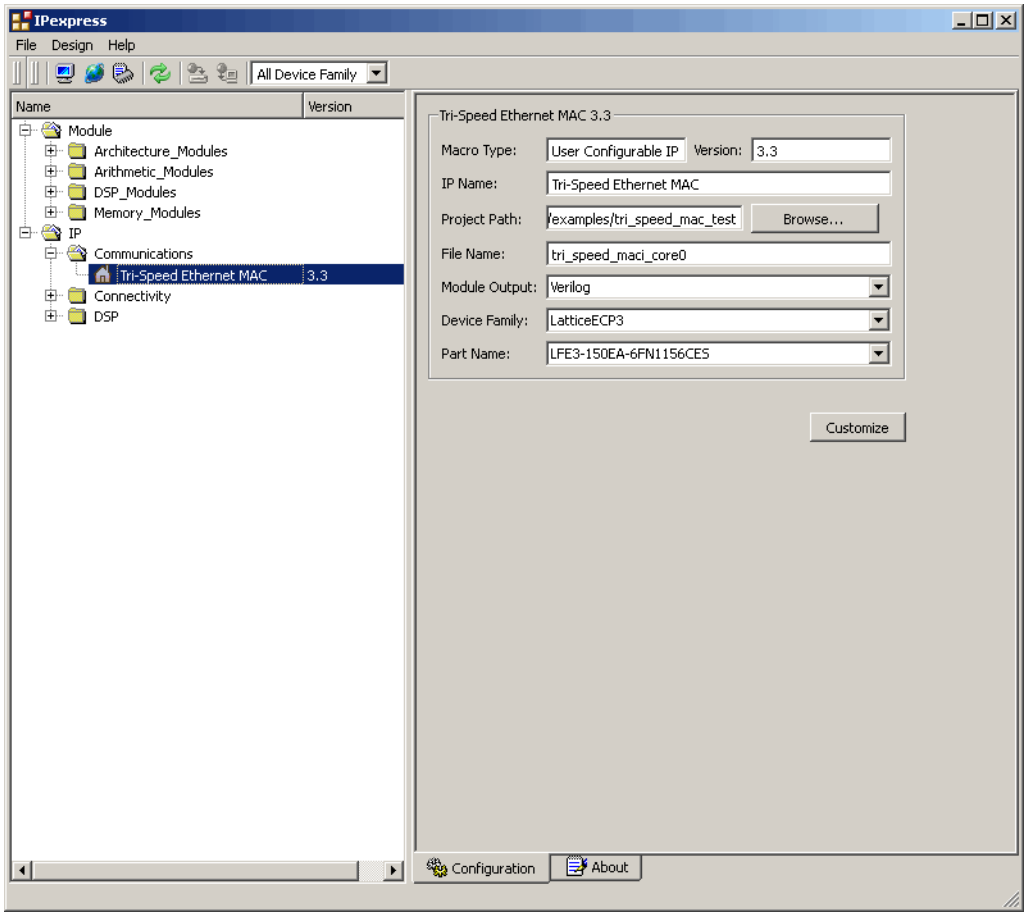
Getting Started

The TSMAC IP core is available for download from the Lattice IP Server using the IPexpress tool. The IP files are automatically installed using ispUPDATE technology in any customer-specified directory. After the IP core has been installed, the IP core will be available in the IPexpress GUI dialog box shown in Figure 4-1.

The IPexpress tool GUI dialog box for the TSMAC IP core is shown in Figure 4-1. To generate a specific IP core configuration the user specifies:

- **Project Path** – Path to the directory where the generated IP files will be located.
- **File Name** – "username" designation given to the generated IP core and corresponding folders and files.
- **Module Output** – Verilog or VHDL.
- **Device Family** – Device family to which IP is to be targeted (such as ECP5 or LatticeECP3). Only families that support the particular IP core are listed.
- **Part Name** – Specific targeted part within the selected device family.

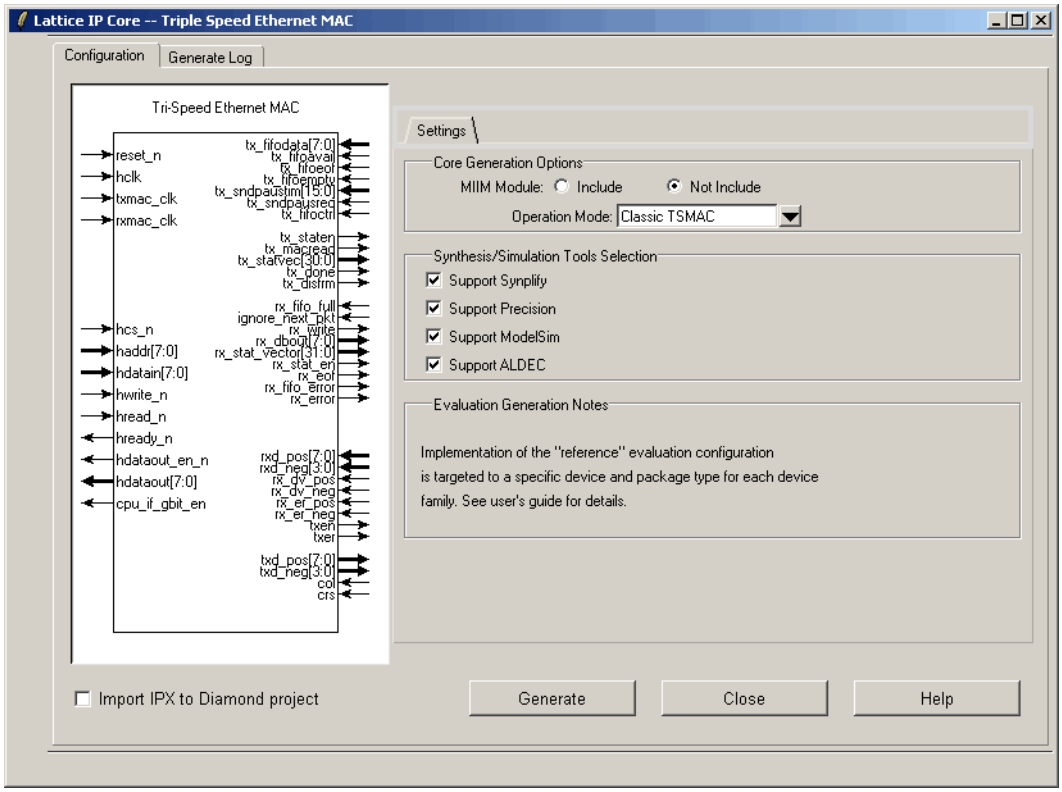
Figure 4-1. IPexpress Tool Dialog Box (Diamond Version)



Note that if the IPexpress tool is called from within an existing project, Project Path, Module Output, Device Family and Part Name default to the specified project parameters. Refer to the IPexpress tool online help for further information.

To create a custom configuration, the user clicks the **Customize** button in the IPexpress tool dialog box to display the TSMAC IP core Configuration GUI, as shown in Figure 4-2. From this dialog box, the user can select the IP parameter options specific to their application. Refer to Parameter Settings for more information on the TSMAC IP core parameter settings.

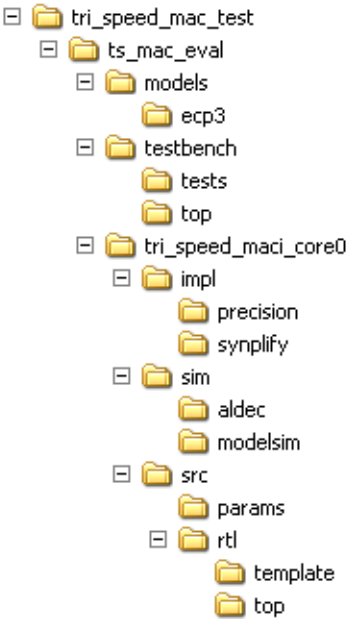
Figure 4-2. TSMAC IP Core - Configuration GUI (Diamond Version)



IPexpress-Created Files and Top Level Directory Structure

When the user clicks the **Generate** button in the IP Configuration dialog box, the IP core and supporting files are generated in the specified “Project Path” directory. The directory structure of the generated files is shown in Figure 4-3.

Figure 4-3. LatticeECP3 TSMAC IP Core Directory Structure



The design flow for IP created with the IPexpress tool uses a post-synthesized module (NGO) for synthesis and a protected model for simulation. The post-synthesized module is customized and created during the IPexpress tool generation.

Table 4-1 provides a list of key files and directories created by the IPexpress tool and how they are used. The IPexpress tool creates several files that are used throughout the design cycle. The names of most of the created files are customized to the user's module name specified in the IPexpress tool. These are all of the files needed to implement and verify the TSMAC IP core in a top-level design.

Table 4-1. File List

File	Description
<*username*>.lpc	This file contains the IPexpress tool options used to recreate or modify the core in the IPexpress tool.
<*username*>.ipx	The IPX file holds references to all of the elements of an IP or Module after it is generated from the IPexpress tool (Diamond version only). The file is used to bring in the appropriate files during the design implementation and analysis. It is also used to re-load parameter settings into the IP/Module generation GUI when an IP/Module is being re-generated.
<*username*>.ngo	This file provides the synthesized IP core.
<*username*>_bb.v	This file provides the synthesis black box for the user's synthesis.
<*username*>_inst.v	This file provides an instance template for the TSMAC IP core.
<*username*>_beh.v	This file provides the front-end simulation library for the TSMAC IP core.

Table 4-2 provides a list of key additional files providing IP core generation status information and command line generation capability are generated in the user's project directory.

Table 4-2. Additional Files

File	Description
<*username*>_generate.tcl	Created when GUI "Generate" button is pushed, invokes generation, may be run from command line.
<*username*>_generate.log	Diamond synthesis and map log file.
<*username*>_gen.log	IPexpress IP generation log file

The **\<ts_mac_eval>** and subtending directories provide files supporting TSMAC IP core evaluation. The **\<ts_mac_eval>** directory shown in Figure 4-3 contains files and folders with content that is constant for all configurations of the TSMAC. The **\<*username*>** subfolder (**\tri_speed_mac_core0** in this example) contains files and folders with content specific to the username configuration.

The **\ts_mac_eval** directory is created by IPexpress the first time the core is generated and updated each time the core is regenerated. A **\<*username*>** directory is created by IPexpress each time the core is generated and regenerated each time the core with the same file name is regenerated. A separate \<username> directory is generated for cores with different names, e.g. **\<tsmac0>, \<tsmac1>**, etc.

Instantiating the Core

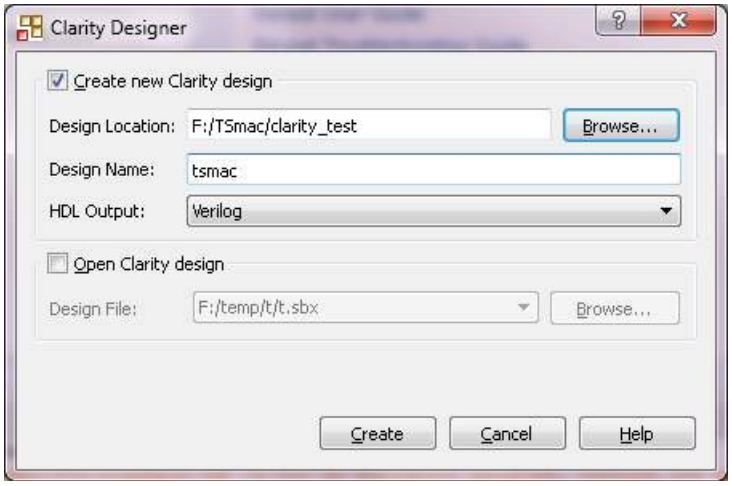
The generated TSMAC IP core package includes black-box (<*username*>_bb.v/vhd) and instance (<*username*>_inst.v/vhd) templates (Verilog or VHDL) that can be used to instantiate the core in a top-level design. An example RTL top-level reference source file that can be used as an instantiation template for the IP core is provided in **\<*project_dir*>\ts_mac_eval\<*username*>\src\rtl\top**. Users may also use this top-level reference as the starting template for the top-level for their complete design.

IP Core Generation in Clarity Designer

Getting Started

The first step in generating an IP Core in Clarity Designer is to start a project in Diamond software with the ECP5 device. Clicking the **Clarity Designer** button opens the Clarity Designer tool.

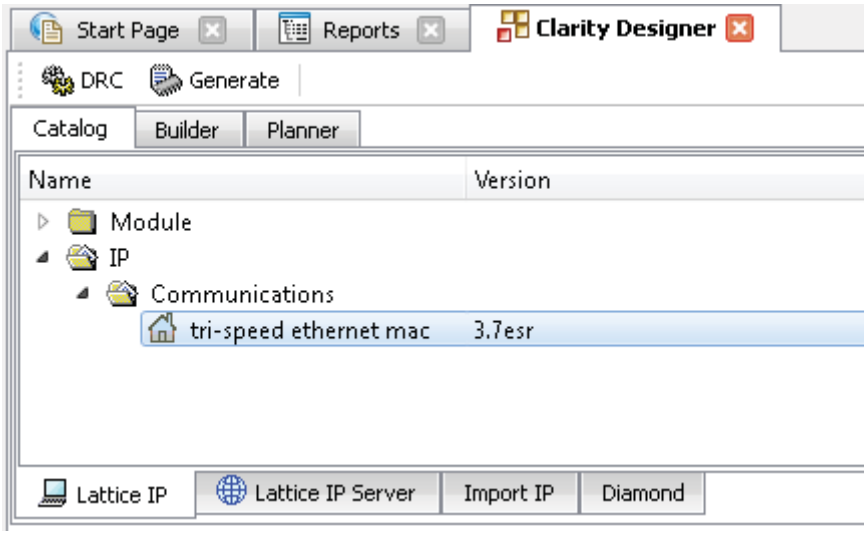
Figure 4-4. Starting a Project in Clarity Designer



As shown in Figure 4-4, you can create a new design or open an existing one. You can specify the design location, design name and HDL output format. Click **Create** to open the Clarity Designer main window.

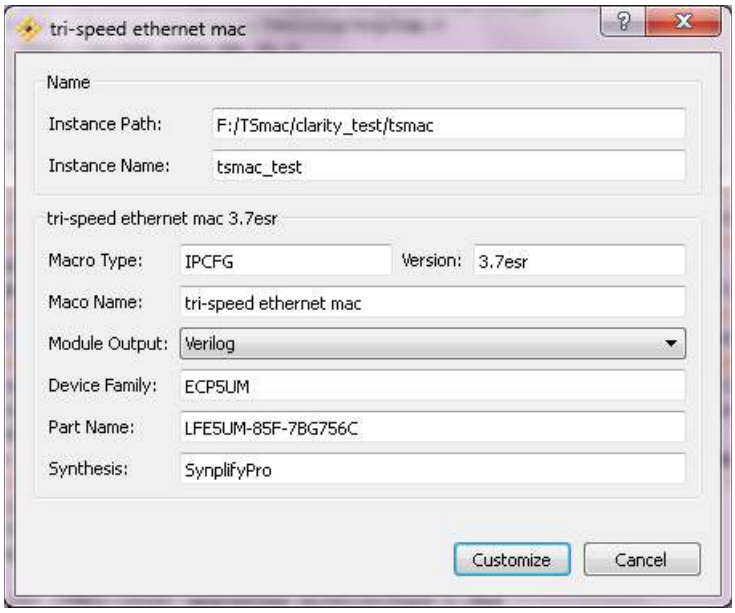
The TSMAC IP core is available for download from the Lattice IP Server using the Clarity Designer tool. The IP files are automatically installed using ispUPDATE technology in any customer-specified directory. After the IP core has been installed, the IP core will be available in the Clarity Designer GUI Catalog box shown in Figure 4-5.

Figure 4-5. Clarity Designer Catalog Box



Double-click the IP name to open a dialog box as shown in Figure 4-5.

Figure 4-6. Clarity Designer Dialog Box



To generate a specific IP core configuration the user specifies:

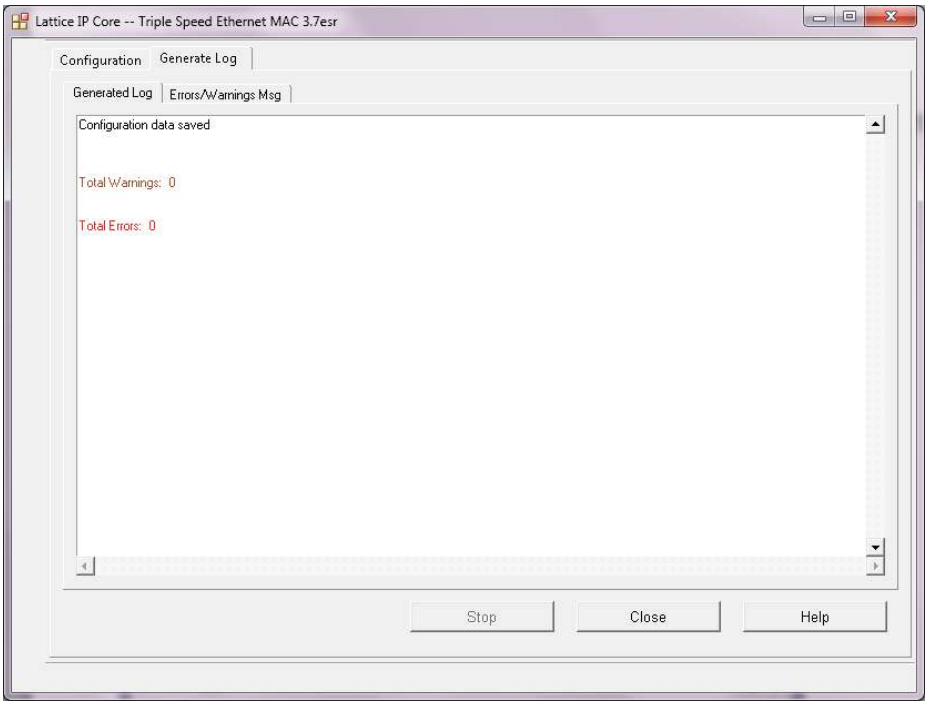
- **Instance Path** – Path to the directory where the generated IP files will be located.
- **Instance Name** – "username" designation given to the generated IP core and corresponding folders and files.
- **Module Output** – Verilog or VHDL.
- **Device Family** – Device family to which IP is to be targeted.
- **Part Name** – Specific targeted part within the selected device family.

Note that because the Clarity Designer tool must be called from within an existing project, Project Path, Module Output, Device Family and Part Name default to the specified project parameters. Refer to the IPexpress tool online help for further information.

To create a custom configuration:

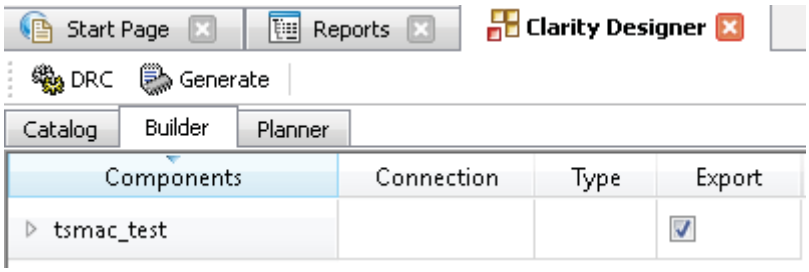
1. Click the **Customize** button in the Clarity Designer dialog box to display the TSMAC IP core Configuration GUI, as shown in Figure 4-6.
2. Select the IP parameter options specific to your application. Refer to Parameter Settings for more information on the SGMII IP core parameter settings.
3. After setting the parameters, click **Configure**.
4. A dialog box, shown in Figure 4-7, displays logs, errors and warnings. Click **Close**.

Figure 4-7. Clarity Designer Generate Log Tab



5. The Clarity Designer Builder tab, shown in Figure 4-8, opens with the defined component displayed.

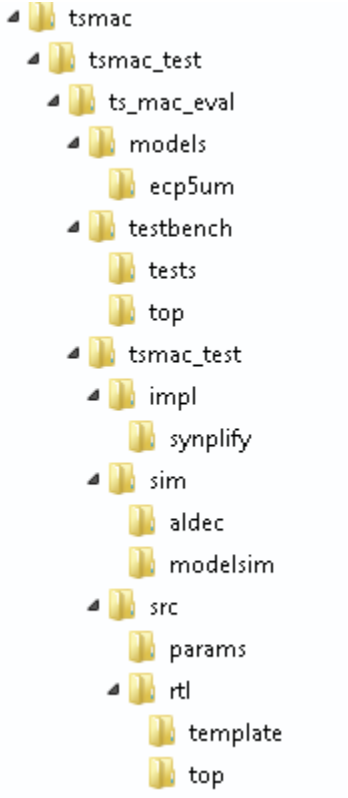
Figure 4-8. Clarity Designer Builder Tab



Clarity Designer Created Files and Top Level Directory Structure

The directory structure of the generated files is shown in Figure 4-9.

Figure 4-9. ECP5 TSMAC IP Core Directory Structure



The design flow for IP created with the Clarity Designer tool uses post-synthesized modules (NGO) for synthesis and a protected model for simulation. The post-synthesized module are customized and created during the Clarity Designer tool generation.

Table 4-3 provides a list of key files and directories created by the Clarity Designer tool and how they are used. The Clarity Designer tool creates several files that are used throughout the design cycle. The names of most of the created files are customized to the user's module name specified in the Clarity Designer tool.

Table 4-3. File List

File	Description
*<username>*_inst.v	This file provides an instance template for the TSMAC IP core.
*<username>*_beh.v	This file provides a behavioral simulation model for the TSMAC IP core.
*<username>*_bb.v	This file provides the synthesis black box for the user's synthesis.
<username>.ngo	The ngo files provide the synthesized IP core.
<username>.pc	This file contains the IPexpress tool options used to recreate or modify the core in the IPexpress tool.
<username>.ipx	The IPX file holds references to all of the elements of an IP or Module after it is generated from the IPexpress tool (Diamond version only). The file is used to bring in the appropriate files during the design implementation and analysis. It is also used to reload parameter settings into the IP/Module generation GUI when an IP/Module is being re-generated.
*<username>*_generate.tcl	This file is created when you click the **Generate** button. This file may be run from command line.
*<username>*_generate.log	This is the IPexpress scripts log file.
*<username>*_gen.log	This is the IPexpress IP generation log file.

Simulation Evaluation

Please refer to Running Functional Simulation.

IP Core Implementation

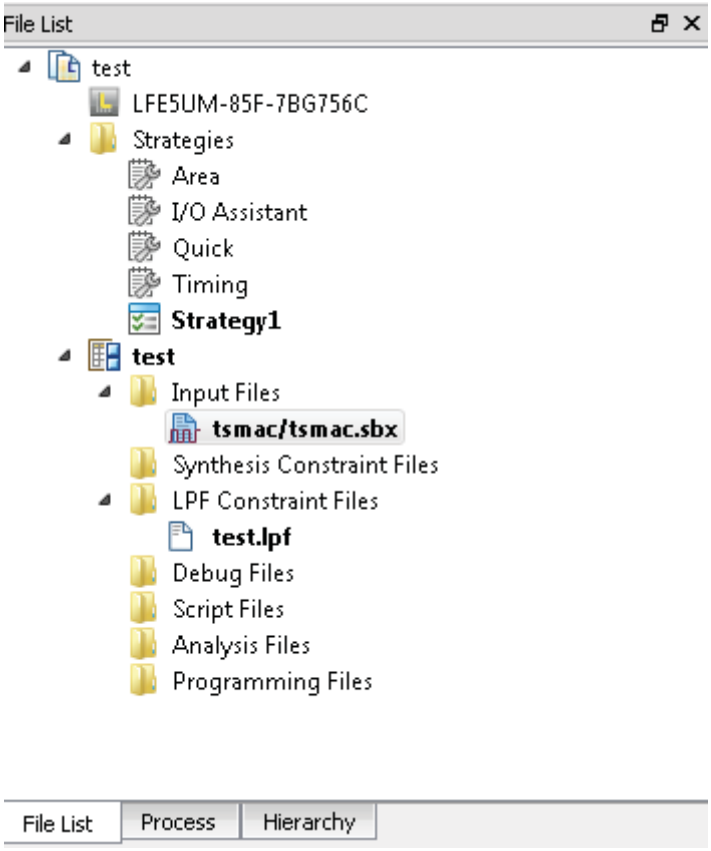
After completing the Configuration step, perform the procedure below. Please refer to Figure 4-10.

1. Click the **Planner** tab.
2. Expand the configured IP instance to Lane level.
3. Drag the lanes of the IP to the DCU channel(s).
4. Click the **Generate** button to create the Clarity Designer (.sbx) file.

Clarity Designer (.sbx) files can be used in design projects such as an HDL file or an IPexpress generated (.ipx) file. A key difference between IPexpress generated files and Clarity Designer generated files is that the latter may contain not only a single block but multiple modules or IP blocks and may represent a subsystem. In IPexpress, the process generates a single module or IP. This is a one step process since an IPexpress file can only contain one module or IP. In Clarity Designer, saving a file is a separate step. Modules or IP are configured and multiple modules or IP can optionally be added within the same file. Additionally, since building and planning can also be done, saving the file and generating the blocks may be performed later.

After the Generate step is completed, the ".sbx" file is automatically added to current Diamond Project Input Files list as shown in Figure 4-10.

Figure 4-10. File List in Report Dialog Box



After this step, click **Process** at the bottom of window, then double-click **Place & Route Design** to Start PAR. This is similar to a standard Diamond PAR flow.

Regenerating/Recreating the IP Core

By *regenerating* an IP core with the Clarity Designer tool, you can modify any of the options specific to an existing IP instance. By *recreating* an IP core with Clarity Designer tool, you can create (and modify if needed) a new IP instance with an existing LPC/IPX configuration file.

Regenerating an IP Core in Clarity Designer Tool

To regenerate an IP core in Clarity Designer:

1. In the Clarity Designer Builder tab, right-click on the existing IP instance and choose **Config**.
2. In the module dialog box, choose the desired options.

   For more information about the options, click **Help**. You may also click the **About** tab in the Clarity Designer window for links to technical notes and user guides. The IP may come with additional information.

   As the options change, the schematic diagram of the module changes to show the I/O and the device resources the module needs.
3. Click **Configure**.

Recreating an IP Core in Clarity Designer Tool

To recreate an IP core in Clarity Designer:

1. In Clarity Designer click the Catalog tab.
2. Click the **Import IP** tab (at the bottom of the view).
3. Click **Browse**.
4. In the Open IPX File dialog box, browse to the .ipx or .lpc file of the module or IP to generate.
5. Click **Open**.
6. Type in a name for Target Instance. Note that this instance name should not be the same as any of the existing IP instances in the current Clarity Designer project.
7. Click **Import**. The module's dialog box opens.
8. In the dialog box, choose desired options.

   For more information about the options, click **Help**. You may also check the **About** tab in the Clarity Designer window for links to technical notes and user guides. The IP may come with additional information.

   As the options change, the schematic diagram of the module changes to show the ports and the device resources the module needs.
9. Click **Configure**.

Running Functional Simulation

Simulation support for the TSMAC IP core is provided for Aldec Active-HDL (Verilog and VHDL) simulator, Mentor Graphics ModelSim (Verilog only) simulator.

The functional simulation includes a configuration-specific behavioral model of the TSMAC IP core, which is instantiated in an FPGA top level along with some test logic (MAC client side FIFO loop back logic, PLLs, and registers with Read/Write Interface). This FPGA top, which is referred to as the Test Application Design, is instantiated in an evaluation test bench that configures FPGA test logic registers and TSMAC IP core registers. The test bench also sources Ethernet packets to the Test Application, and monitors packets from Test Application.

More information on the simulation and the Test Application Design can be found in Application Support.

The generated IP core package includes the configuration-specific behavior model (<username>_beh.v) for functional simulation in the “Project Path” root directory. Lattice does not provide a test bench for evaluating this IP core in isolation. However, a functional simulation capability is provided in which <username>_beh.v is instantiated in the Test Application Design described in Application Support section of this document.

The simulation script supporting ModelSim evaluation simulation is provided in **\<*project_dir*>\ts_mac_eval\<*username*>\sim\modelsim**.

The simulation script supporting Aldec evaluation simulation is provided in **\<*project_dir*>\ts_mac_eval\<*username*>\sim\aldec.**

The Test Application Design is instantiated in a test-bench provided in **\<*project_dir*>\ts_mac_eval\testbench**.

Both ModelSim and Aldec simulation is supported via test bench files provided in **\<*project_dir*>\ts_mac_eval\testbench**. Models required for simulation are provided in the corresponding **\models** folder.

Users may run the Aldec evaluation simulation by doing the following:

1. Open Active-HDL.
2. Under the Tools tab, select **Execute Macro**.
3. Browse to folder **\<*project_dir*>\ts_mac_eval\<*username*>\sim\aldec** and execute one of the “do” scripts shown.

Users may run the ModelSim evaluation simulation by doing the following:

1. Open ModelSim.
2. Under the File tab, select **Change Directory** and choose the folder **<*project_dir*>\ts_mac_eval\<*username*>\sim\modelsim**.
3. Under the Tools tab, select **Execute Macro** and execute the ModelSim “do” script shown.

Note: *When the simulation completes, a pop-up window will appear asking “Are you sure you want to finish?” Answer “No” to analyze the results (answering “Yes” closes ModelSim).*

Synthesizing and Implementing the Core in a Top-Level Design

Synthesis support for the TSMAC IP core is provided for Mentor Graphics Precision or Synopsys Synplify. The TSMAC IP core itself is synthesized and is provided in NGO format when the core is generated in IPexpress. Users may synthesize the core in their own top-level design by instantiating the core in their top-level as described previously and then synthesizing the entire design with either Synplify or Precision RTL Synthesis.

The following text describes the evaluation implementation flow for Windows platforms. The flow for Linux and UNIX platforms is described in the Readme file included with the IP core.

Two example top-level reference source files are provided to support TSMAC top-level synthesis and implementation.

One top file is for a TSMAC IP core only implementation in isolation. This design is intended only to provide an accurate indication of the device utilization associated with the TSMAC IP core itself and should not be used as an actual implementation example.

The other top file is for the Test Application Design, and includes both the TSMAC IP core and additional loop back test logic. This is the same top used in the functional simulation described in the previous section.

Both top-level files ts_mac_core_only_top.v (core only) and ts_mac_top.v (Application) are provided in **\<*project_dir*>\ts_mac_eval\<*username*>\src\rtl\top**.

Push-button implementation of both reference designs is supported via the project files, <*username*>_reference_eval.ldf and <*username*>_core_only_eval.ldf located in **\<*project_dir*>\ts_mac_eval\<*username*>\impl\(*synplify or precision*)**.

Using Project Files With Synplify in Diamond

1. Choose **File > Open > Project**.
2. Browse to **\<*project_dir*>\ts_mac_eval\<*username*>\impl\synplify** in the Open Project dialog box.
3. Select and open <*username*>_reference_eval.ldf or <*username*>_core_only_eval.ldf. At this point, all of the files needed to support top-level synthesis and implementation will be imported to the project.
4. In the Diamond Toolbar, choose **Project > Active Strategy >Synplify Pro Settings**.
5. Set the Frequency (MHz) to 125 in the Synplify Pro pop-up window and close the window.
6. Select the device top-level entry in the left-hand GUI window.
7. Implement the complete design via the standard Diamond GUI flow.

Hardware Evaluation

The TSMAC IP core supports Lattice's IP hardware evaluation capability, which 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.

Enabling Hardware Evaluation in Diamond:

Choose **Project > Active Strategy > Translate Design Settings**. The hardware evaluation capability may be enabled/disabled in the Strategy dialog box. It is enabled by default.

Updating/Regenerating the IP Core

By regenerating an IP core with the IPexpress tool, you can modify any of its settings including: device type, design entry method, and any of the options specific to the IP core. Regenerating can be done to modify an existing IP core or to create a new but similar one.

Regenerating an IP Core in Diamond

To regenerate an IP core in Diamond:

1. In IPexpress, click the **Regenerate** button.
2. In the Regenerate view of IPexpress, choose the IPX source file of the module or IP you wish to regenerate.
3. IPexpress shows the current settings for the module or IP in the Source box. Make your new settings in the T**arget** box.
4. If you want to generate a new set of files in a new location, set the new location in the **IPX Target File** box. The base of the file name will be the base of all the new file names. The IPX Target File must end with an .ipx extension.
5. Click **Regenerate.** The module's dialog box opens showing the current option settings.
6. In the dialog box, choose the desired options. To get information about the options, click **Help**. Also, check the About tab in IPexpress for links to technical notes and user guides. IP may come with additional information. As the options change, the schematic diagram of the module changes to show the I/O and the device resources the module will need.
7. To import the module into your project, if it's not already there, select **Import IPX to Diamond Project** (not available in stand-alone mode).
8. Click **Generate**.
9. Check the Generate Log tab to check for warnings and error messages.
10. Click **Close**.

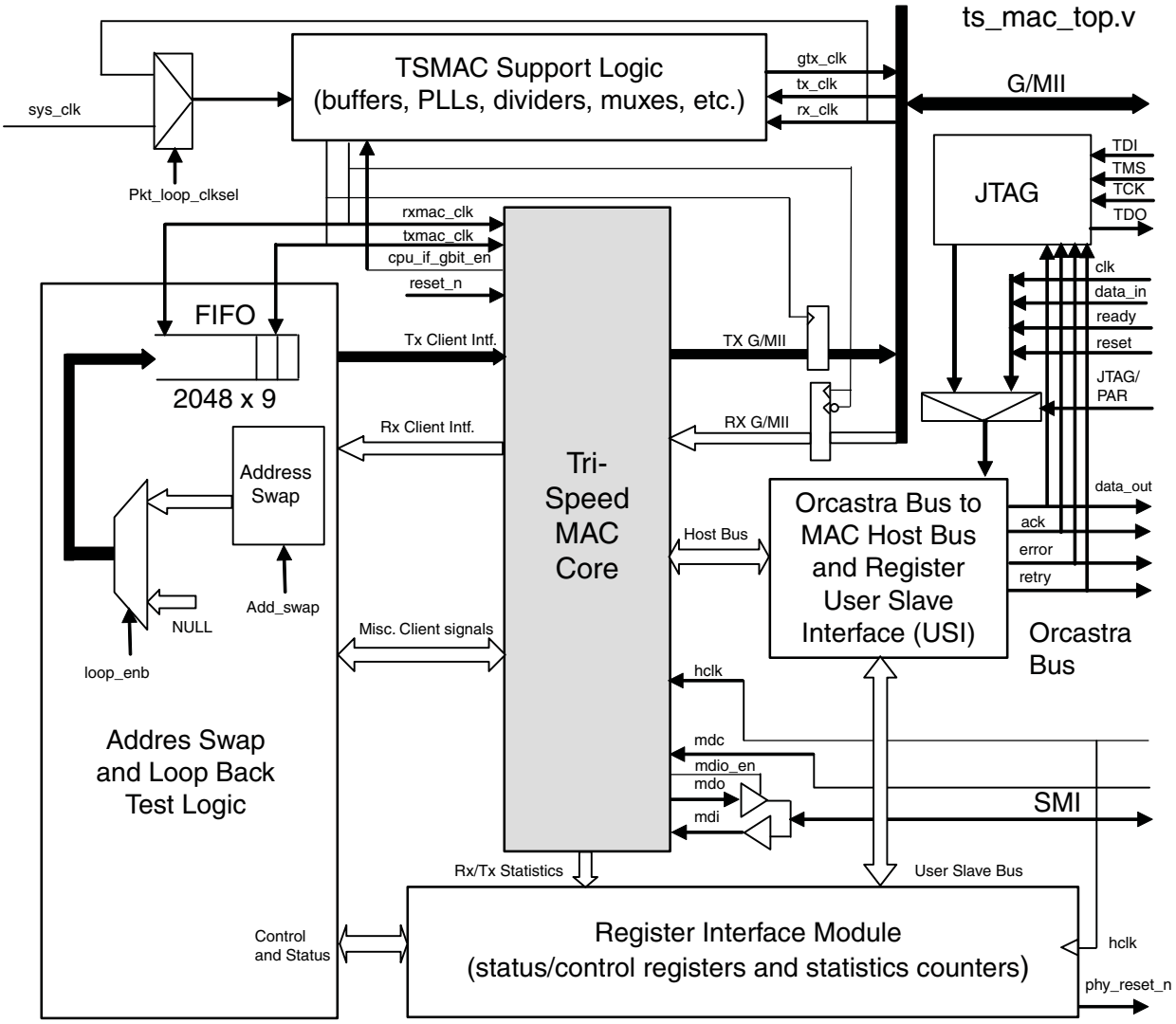
The IPexpress package file (.ipx) supported by Diamond holds references to all of the elements of the generated IP core required to support simulation, synthesis and implementation. The IP core may be included in a user's design by importing the .ipx file to the associated Diamond project. To change the option settings of a module or IP that is already in a design project, double-click the module's .ipx file in the File List view. This opens IPexpress and the module's dialog box showing the current option settings. Then go to step 6 above.

This chapter provides application support information for the TSMAC IP core.

Test Application Design

The TSMAC IP core evaluation package includes a reference design that can be used to instantiate, simulate, map, place and route the Lattice TSMAC IP core in an example working design. This reference design provides a loop back path for packets on the MAC Rx/Tx client interface, through a FIFO and associated logic. Ethernet packets are sourced to the Rx G/MII and looped back on the MAC Rx/Tx client FIFO interface. Source and destination addresses in the ethernet frame can be swapped so the looped back packets on the Tx G/MII have the correct source and destination addresses. This design also provides connections to the other interfaces of the Lattice TSMAC IP core, including the SMI, Host Bus and Rx/Tx Statistics interfaces. Figure 5-1 shows a block diagram of the test application design. This loopback design should be able to sustain a 1 Gbps throughput with minimum sized Ethernet frames.

Figure 5-1. Test Application Design



The main blocks and functions of the test application design are as follows:

The Test Logic Module

This module includes the address swap logic, loop back FIFO and associated control logic, and miscellaneous control and status glue logic between the MAC Rx/Tx Client interface and the register interface module.

The ORCAstra to Host Bus/USI module

This module converts the ORCAstra™ bus (a Lattice defined slow speed serial bus) to the host bus and a user slave interface bus. Using the ORCAstra bus, a user can access the internal TSMAC IP core registers, as well as the test application registers. Note that external PHY registers can also be accessed via the ORCAstra interface through the internal TSMAC IP core registers and the SMI interface in the TSMAC IP core. The MAC registers (accessed via the host bus) and test logic registers (accessed via the USI) are memory mapped as described in Test Application Registers. An ORCAstra Bus Test bench driver is provided to ease simulation with this interface.

The Register Interface Module

This module is accessed through the ORCAstra bus via the user slave interface (USI). The module contains registers used by the test application for control and status of the TSMAC IP core and external PHY. In addition this module contains 16 bit statistics counters fed by the TSMAC IP core's Rx/Tx statistics interfaces. These counters can be read and cleared through the ORCAstra bus. An address map and description of these registers is given in Test Application Registers.

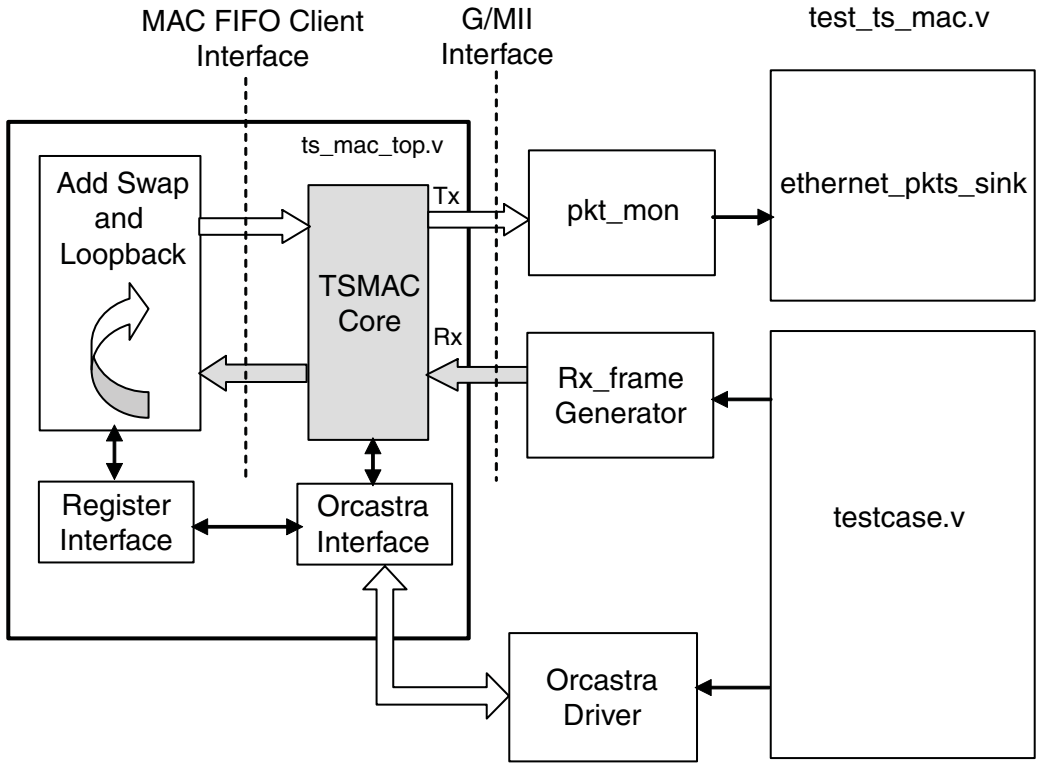
TSMAC Support Logic

This logic includes IO buffers, PLLs, clock dividers and multiplexers used by the TSMAC IP core and test application. Use of technology-specific modules (like the I/O and PLLs) and their settings are pre-defined for this application. They are based on the user configuration option selected. Information can be found in the downloaded files (e.g. the _pll.v files in the models directory) that come with the IPexpress IP core installed package.

Simulation of the Test Application

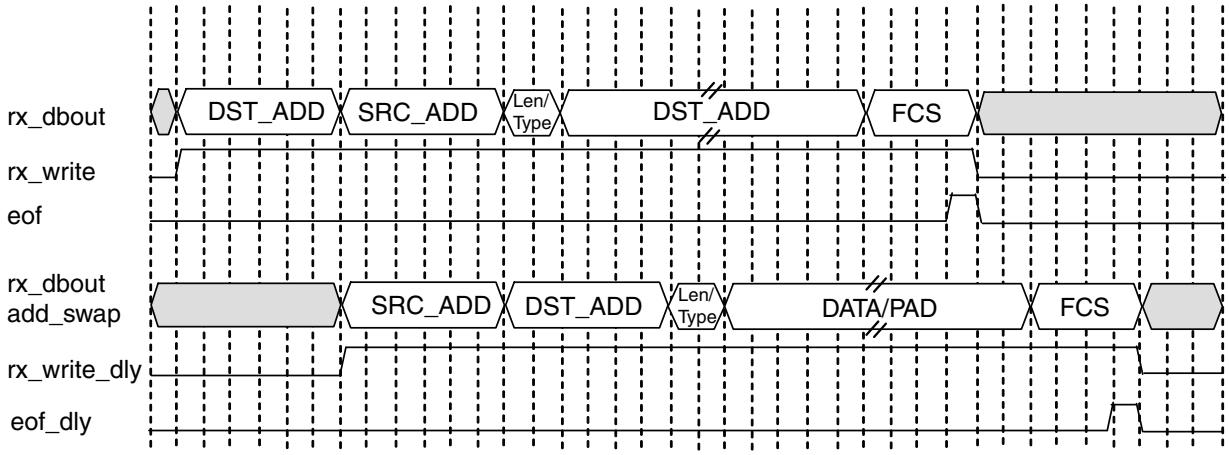
Figure 5-2 shows a block diagram of the test bench setup provided with the Test Application Design. All Accesses to the internal TSMAC registers and test application design registers can be accomplished through the testcase.v file (via an Orcastra driver). In addition, variable size and types of ethernet frames can be sourced to the TSMAC Rx G/MII port through the testcase.v file (via the Rx frame generator). These received packets get looped back inside the test application design and are monitored on the TSMAC Tx G/MII port by a packet monitor. The monitored packets are logged in a file named ethernet_pkts_sink. The testcase.v file can be found in **\<*project_dir*>\ts_mac_eval\testbench\tests**. While the ethernet_pkts_sink file will be created and placed in **\<*project_dir*>\ts_mac_eval\tsmac\sim\(modelsim or aldec)** directory once the simulation is run and completed.

Figure 5-2. Block Diagram of Testbench Setup



Note that the user can easily edit the testcase.v file to configure and monitor any registers they desire in the test application design, as well modify the type or number of packets they wish to drive to the test application design. Also note that the destination and source addresses can be swapped by enabling the swap control bit in the test control register. Figure 5-3 shows a timing waveform of data and control signals on the ingress and egress sides of the address swap module (when address swap is enabled).

Figure 5-3. Timing Waveform



Test Application Registers

There are two address spaces in the test application design. Refer to Table 2-4 for a listing of TSMAC IP Core internal registers. The first address in Table 2-4 starts at offset 0x00 and ends at 0x35. This space contains the TSMAC IP core internal registers.

The second address space shown in Table 5-1 starts at offset 0x8000 and ends at 0x8031. This space contains ID, control, status and statistics registers used by the test application.

The REGINTF logic block in the test application provides the address decoding, for the RO and R/W registers used by the test logic and statistics counters. All registers are 8 bits wide and are byte addressable. Note that the statistics counter registers are composed of two 8 bit registers, a low and a high byte register, therefore, in order to access these registers, two byte accesses must be made. For example, to access to all 16 bits of the RXOKCNT would require an access to both 0x8019 (high byte) and 0x8018 (low byte). Note that since the statistics counter registers are clear on read (COR), the high byte should be read first before reading the low byte, since a read of the low byte clears all the combined 16 bits of the low and high registers. The address map for the test application related resisters are listed in Table 5-1.

Table 5-1. Test Application Related Registers

Address	Register Description	Mnemonic	Type
0x08000	VERsion/IDentification Register	VERID	RO
0x08001	TeST CoNTroL Register	TSTCNTL	RW
0x08002	TeST CoNTroL Register 2	TSTCNTL_2	RW
0x08003	MAC CoNTroL Register	MACCNTL	RW
0x08004	PAUSe TiMeR Register - Low byte	PAUSTMRL	RW
0x08005	PAUSe TiMeR Register - High byte	PAUSTMRH	RW
0x08006	FIFO Almost Full Threshold Register - Low	FIFOAFTL	RW
0x08007	FIFO Almost Full Threshold Register - High	FIFOAFTH	RW
0x08008	FIFO Almost Empty Threshold Register - Low	FIFOAETL	RW
0x08009	FIFO Almost Empty Threshold Register - High	FIFOAETH	RW
0x0800a	RX Status Register	RXSTATUS	RO/COR
0x0800b	TX Status Register	TXSTATUS	RO/COR
0x0800c, 0x0800d	RX Packet Ignored Counter Register (L,H)	RXPICNT	RO/COR
0x0800e, 0x0800f	RX Length Check Error CouNTer (L,H)	RXLCECNT	RO/COR
0x08010, 0x08011	RX Long Frames CouNTer Register (L,H)	RXLFCNT	RO/COR
0x08012, 0x08013	RX Short Frames CouNTer Register (L,H)	RXSFCNT	RO/COR
0x08014, 0x08015	RX IPG violations CouNTer Register (L,H)	RXIPGCNT	RO/COR
0x08016, 0x08017	RX CRC errors CouNTer Register (L,H)	RXCRCCNT	RO/COR
0x08018, 0x08019	RX OK packets CouNTer Register (L,H)	RXOKCNT	RO/COR
0x0801a, 0x0801b	RX Control Frame CouNTer Register (L,H)	RXCFCNT	RO/COR
0x0801c, 0x0801d	RX Pause Frame CouNTer Register (L,H)	RXPFCNT	RO/COR
0x0801e, 0x0801f	RX Multicast Frame CouNTer Register (L,H)	RXMFCNT	RO/COR
0x08020, 0x08021	RX Broadcast Frame CouNTer Register (L,H)	RXBFCNT	RO/COR
0x08022, 0x08023	RX VLAN tagged Frame CouNTer Register (L,H)	RXVFCNT	RO/COR
0x08024, 0x08025	TX Unicast Frame CouNTer Register (L,H)	TXUFCNT	RO/COR
0x08026, 0x08027	TX Pause Frame CouNTer Register (L,H)	TXPFCNT	RO/COR
0x08028, 0x08029	TX Multicast Frame CouNTer Register (L,H)	TXMFCNT	RO/COR
0x0802a, 0x0802b	TX Broadcast Frame CouNTer Register (L,H)	TXBFCNT	RO/COR

Table 5-1. Test Application Related Registers

Address	Register Description	Mnemonic	Type
0x0802c, 0x0802d	TX VLAN tagged Frame CouNTer Register (L,H)	TXVFCNT	RO/COR
0x0802e, 0x0802f	TX BAD FCS Frame CouNTer Register (L,H)	TXBFCCNT	RO/COR
0x08030, 0x08031	TX Jumbo Frame CouNTer Register (L,H)	TXJFCNT	RO/COR
0x08032 - 0x0FFFF	UNUSED		

Register Descriptions

Version/Identification (RO)

Mnemonic: VERID

POR Value = A2H

Name	Range	Description
Version_ID [7:0]	7:0	**Version ID**: Echoes back the version and ID of the application (A2H).

Test Control Register (R/W)

Mnemonic: TSTCNTL

POR Value = 00H

Name	Range	Description
Rsvd	7:4	Reserved
pkt_loop_clksel	3	**Packet Loop Clock Select**: When this bit is set High, sys_clk is tied to rx_clk. When the bit is Low sys_clk is sourced from an external IO pin.
reset_phy_n	2	**Reset PHY**: When this bit is set High, the phy_reset_n pin is de-asserted (High). When this bit is Low the phy_reset_n pin is asserted (Low). This pin can be used to reset an External PHY device.
loop back enable	1	**Loop back Enable**: When this bit is set High the client side loop back is enabled. When the bit is Low the Loop back is disabled.
destination/source address swap	0	**Swap Destination and Source Addresses:** When this bit is set High the Ethernet Destination and source addresses are swapped. When the bit is Low there is no address swapping.

Test Control Register 2 (R/W)

Mnemonic: TSTCNTL 2

POR Value = 00H

Name	Range	Description
testcontrol[7:0]	7:0	Reserved.

MAC Control Register (R/W)

Mnemonic: MACCNTL

POR Value = 00H

Name	Range	Description
Rsvd [7:5]	7-5	Reserved
ignore next packet	4	**ignore next packet**: This bit asserts the ignore_next_pkt pin on the TSMAC IP core. See Table 2-1 for more information.
tx_fifo_empty	3	**tx_fifo_empty**: This bit gets ORed with the FIFO empty signal. The ORed output sets the tx_fifoempty pin on the TSMAC IP core. See Table 2-1 for more information on this TSMAC IP core signal. Note by setting this bit you can mimic the Tx FIFO being empty. Also note that the application design only has one loopback FIFO. So the Tx FIFO is the same as the Rx FIFO.
rx_fifo full	2	**rx_fifo full**: This bit gets ORed with the FIFO full signal. The ORed output sets the rx_fifo_full pin on the TSMAC IP core. See Table 2-1 for more information on this TSMAC IP core signal. Note by setting this bit you can mimic the rx fifo being full. Also note that the application design only has one loopback FIFO. So the Tx FIFO is the same as the Rx FIFO.
fifo control frame	1	**fifo control frame**: This bit sets the tx_fifoctrl pin on the TSMAC IP core. See Table 2-1 for more information on this TSMAC IP core signal.
send pause request	0	**send pause request**: This bit gets ORed with the Tx FIFO almost full signal. The ORed output sets the tx_sndpausreq pin on the TSMAC IP core. See Table 2-1 for more information on this TSMAC IP core signal. When this bit is set High OR the FIFO is almost full tx_sndpausreq is asserted, otherwise tx_sndpausreq is de-asserted.

Pause Timer Register - Low Byte (R/W)

Mnemonic: PAUSTMRL

POR Value = 00H

Name	Range	Description
pause timer Low bits [7:0]	7:0	**Pause Timer Low bits**: These reg bits set the tx_sndpaustim[7:0] pins on the TSMAC IP core. See Table 2-1 for more information on this TSMAC IP core signal.

Pause Timer Register - High Byte (R/W)

Mnemonic: PAUSTMRH

POR Value = 00H

Name	Range	Description
pause timer High bits [7:0]	7:0	**Pause Timer High bits**: These bits set the tx_sndpaustim[15:8] pins on the TSMAC IP core. See Table 2-1 for more information on this TSMAC IP core signal.

FIFO Almost Full Threshold Register - Low (R/W)

Mnemonic: FIFOAFTL

POR Value = 00H

Name	Range	Description
FIFO almost Full Low bits [7:0]	7:0	**FIFO almost Full Low bits**: These bits set the loop back FIFO Almost Full threshold [7:0] bits

FIFO Almost Full Threshold Register - High (R/W)

Mnemonic: FIFOAFTH

POR Value = 00H

Name	Range	Description
Rsvd [7:1]	7 - 1	Reserved.
FIFO almost Full High bit [0]	0	**FIFO almost Full High bit**: This bit sets the loop back FIFO Almost Full threshold [8] bit.

FIFO Almost Empty Threshold Register - Low (R/W)

Mnemonic: FIFOAETL

POR Value = 00H

Name	Range	Description
FIFO almost Empty Low bits [7:0]	7:0	**FIFO almost Empty Low bits**: These bits set the loop back FIFO Almost Empty threshold [7:0] bits

FIFO Almost Full Threshold Register - High (R/W)

Mnemonic: FIFOAETH

POR Value = 00H

Name	Range	Description
Rsvd [7:1]	7 - 1	Reserved.
FIFO almost Empty High bit [0]	0	**FIFO almost Empty High bit**: This bit sets the loop back FIFO Almost Empty threshold [8] bit.

RX Status Register (RO/COR)

Mnemonic: RXSTATUS

POR Value = 00H

Name	Range	Description
Rsvd [7:2]	7:2	Reserved.
rx_error	1	**rx_error**: This bit is set high and latched if the rx_fifo_error signal is asserted on TSMAC IP core pin. See Table 2-1 for more information on this TSMAC IP core signal.
rx_fifo_error	0	**rx_fifo_error**: This bit is set high and latched if the rx_error signal is asserted on TSMAC IP core pin. See Table 2-1 for more information on this TSMAC IP core signal.

TXSTATUS (RO/COR)

Mnemonic: TXSTATUS

POR Value = 00H

Name	Range	Description
Rsvd [7:2]	7:2	Reserved.
tx_fifo_full	1	**tx_fifo_full**: This bit is set high and latched if the tx_fifo_full signal from the Loop back FIFO is asserted.
tx_discfrm	0	**tx_discfrm**: This bit is set high and latched if the tx_discfrm signal is asserted on TSMAC IP core pin. See Table 2-1 for more information on this TSMAC IP core signal.

The counter registers listed in Table 5-2 are all 16 bits with 8 bit low and 8 bit high address locations. The counters count different Rx and Tx statistics as defined by the tx_statvec and rx_statvec statistics vectors defined in Table 2-1. All counters have a power-on reset (POR) value of 0x0000.

Table 5-2. Counter Registers

Address	Register Description	Mnemonic	Type
0x0800c	RX Packet Ignored Counter Register	RXPICNT	(RO/COR)
0x0800e	RX Length Check Error CouNTer	RXLCECNT	(RO/COR)
0x08010	RX Long Frames CouNTer Register	RXLFCNT	(RO/COR)
0x08012	RX Short Frames CouNTer Register	RXSFCNT	(RO/COR)
0x08014	RX IPG violations CouNTer Register	RXIPGCNT	(RO/COR)
0x08016	RX CRC errors CouNTer Register	RXCRCCNT	(RO/COR)
0x08018	RX OK packets CouNTer Register	RXOKCNT	(RO/COR)
0x0801a	RX Control Frame CouNTer Register	RXCFCNT	(RO/COR)
0x0801c	RX Pause Frame CouNTer Register	RXPFCNT	(RO/COR)
0x0801e	RX Multicast Frame CouNTer Register	RXMFCNT	(RO/COR)
0x08020	RX Broadcast Frame CouNTer Register	RXBFCNT	(RO/COR)
0x08022	RX VLAN tagged Frame CouNTer Register	RXVFCNT	(RO/COR)
0x08024	TX Unicast Frame CouNTer Register	TXUFCNT	(RO/COR)
0x08026	TX Pause Frame CouNTer Register	TXPFCNT	(RO/COR)
0x08028	TX Multicast Frame CouNTer Register	TXMFCNT	(RO/COR)
0x0802a	TX Broadcast Frame CouNTer Register	TXBFCNT	(RO/COR)
0x0802c	TX VLAN tagged Frame CouNTer Register	TXVFCNT	(RO/COR)
0x0802e	TX BAD FCS Frame CouNTer Register	TXBFCCNT	(RO/COR)
0x08030	TX Jumbo Frame CouNTer Register	TXJFCNT	(RO/COR)

Table 5-3 lists test application I/Os.

Table 5-3. Counter Registers

FPGA Signal Name	FPGA Pin	Notes
col	Input	Collision detect signal
crs	Input	Carrier sense signal
gtx_clk	Output	Tx GMII clock used only in 1 G mode
mdc	Input	SMI clock sourced to PHY from FPGA
mdio	Bi-directional	Bi-directional SMI data to/from PHY from/to MAC
pc_ack	Output	Orcastra parallel signals – only valid if jtag_parallel is Low
pc_clk	Input	Orcastra parallel signals – only valid if jtag_parallel is Low
pc_datain	Input	Orcastra parallel signals – only valid if jtag_parallel is Low
pc_dataout	Output	Orcastra parallel signals – only valid if jtag_parallel is Low
pc_error	Output	Orcastra parallel signals – only valid if jtag_parallel is Low
pc_ready	Input	Orcastra parallel signals – only valid if jtag_parallel is Low
pc_retry	Output	Orcastra parallel signals – only valid if jtag_parallel is Low
jtag_present	Output	JTAG is present
jtag_parallel	Input	Used to select Orcastra interface 1 = JTAG (for HW testing) 0 = Parallel (used during simulation)
reset_n	Input	Active Low
rx_clk	Input	Rx GMII clock
rx_dv	Input	Rx GMII data valid
rx_er	Input	Rx GMII error
rxd_0	Input	Rx GMII eata bit 0
rxd_1	Input	Rx GMII data bit 1
rxd_2	Input	Rx GMII data bit 2
rxd_3	Input	Rx GMII data bit 3
rxd_4	Input	Rx GMII data bit 4
rxd_5	Input	Rx GMII data bit 5
rxd_6	Input	Rx GMII data bit 6
rxd_7	Input	Rx GMII data bit 7
sys_clk	Input	Not needed if pkt_loop_clksel is set to 1
tx_clk	Input	Tx GMII clock
tx_en	Output	Tx GMII enable
tx_er	Output	Tx GMII error
txd_0	Output	Tx GMII data bit 0
txd_1	Output	Tx GMII data bit 1
txd_2	Output	Tx GMII data bit 2
txd_3	Output	Tx GMII data bit 3
txd_4	Output	Tx GMII data bit 4
txd_5	Output	Tx GMII data bit 5
txd_6	Output	Tx GMII data bit 6
txd_7	Output	Tx GMII data bit 7
rxmac_clk	Output	Rx MAC clock – used internally can be brought out
txmac_clk	Output	Tx MAC Clock – used internally can be brought out

Table 5-3. Counter Registers

FPGA Signal Name	FPGA Pin	Notes
hclk	Input	Host bus clock
phy_reset_n	Output	Reset to external PHY (optional signal)

Code Listing for Multicast Bit Selection Hash Algorithm in C Language

The code listing below is to aid software developers in programming the multicast tables in the TSMAC IP core 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 multicastcast_table register index (0 to 7) as well as the bit within the register that needs to be set (0 to 7) 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 will be passed to the MAC client interface.

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

//Hexadecimal equivalent of the CRC
//equn.
#define CRC_POLYNOMIAL 0x04c11db6

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 MSB of the 32bit CRC value,
    // this six bit value is used to index a
    // unique filter bit.
    printf(" crc  %lx \n", crc);
    crc >>= 25;  // TSMAC refers to Bit 30..25
    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;
}
```

Core Validation

The Lattice TSMAC IP core has been validated through interoperability tests with several external PHY devices. An external ethernet protocol generator/analyzer, such as the Spirent Smartbits tester, was used to drive/monitor ethernet traffic through the IP core test circuit.

The IP core was also tested in a real ethernet network environment. Refer to the following Lattice interoperability Technical Notes and documents:

- TN1196, LatticeECP3 Marvell 1 GbE (1000BASE-X) Physical/MAC Layer Interoperability
- TN1197, LatticeECP3/Marvell SGMII Physical/MAC Layer Interoperability
- LatticeMico32 Tri-Speed Ethernet MAC Demo web site (http://www.latticesemi.com/products/intellectualproperty/ipcores/mico32/mico32trispeedethernetmac.cfm)
- LatticeMico32 Ethernet Gigabit MAC Demo web site: (http://www.latticesemi.com/products/intellectualproperty/ipcores/mico32/mico32ethernetgigabitmacd.cfm)

Support Resources

This chapter contains information about Lattice Technical Support, additional references, and document revision history.

Lattice Technical Support

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

IEEE

IEEE offers publications and technology standards on its web site at http://www.ieee.org.

References

- LatticeMico32 Tri-Speed Ethernet MAC Demo web site (http://www.latticesemi.com/products/intellectualproperty/ipcores/mico32/mico32trispeedethernetmac.cfm)
- LatticeMico32 Ethernet Gigabit MAC Demo web site: (http://www.latticesemi.com/products/intellectualproperty/ipcores/mico32/mico32ethernetgigabitmacd.cfm)

LatticeECP3

- DS1021, LatticeECP3 Family Data Sheet
- TN1196, LatticeECP3 Marvell 1 GbE (1000BASE-X) Physical/MAC Layer Interoperability
- TN1197, LatticeECP3/Marvell SGMII Physical/MAC Layer Interoperability

ECP5

DS1044, ECP5 Family Data Sheet

Revision History

Date	Document Version	IP Core Version	Change Summary
April 2015	3.3	4.0	Updated TSMAC Configuration Dialog Box section. Changed Figure 3-1, TSMAC Configuration Dialog Box.
			Removed Synthesis/Simulation Tools Selection.
			Updated Lattice Technical Support section.
			Updated References section.
June 2014	03.2	3.7ea	Added support for ECP5 FPGA family.
			Added support for Diamond 3.2 software.
			Removed support for Lattice ispLEVER.
			Removed information on LatticeECP, LatticeEC, LatticeECP2, LatticeECP2M, and LatticeXP
July 2013	03.1	3.3	Updated the File List table.
			Updated document with new corporate logo.
			Updated Lattice Technical Support information.
December 2010	03.0	3.3	Updated the Receiving Frames text section.
July 2010	02.9	3.3	Added support for Diamond software.
April 2010	02.8	3.2	Divided the document into chapters and added table of contents.
			Added Quick Facts Table in Chapter 1, Introduction.
			Added Chapter 5, “Application Support.”
			Added Chapter 6, “Core Validation.”
November 2009	02.7	3.1	Updated for ispLEVER 8.0.
September 2009	02.6	3.0	Updated Address Filtering text section.
			Added Appendix H. Code Listing for Multicast Bit Selection Hash Algorithm in C Language.
April 2009	02.5	3.0	Updated to include support for LatticeECP3 family.
			Updated appendices.
September 2008	02.4	2.7	Added the Instantiating the Core text section.
			Added the Running Functional Simulation text section.
			Added the Synthesizing and Implementing the Core in a Top-Level Design text section.
			Updated results in utilization tables.
September 2008	02.3	2.7	Expanded Feature list.
			Updated General Description text section.
			Added Tri-Speed 10/100/1G Ethernet MAC Core Block Diagram (Gigabit MAC or SGMII Easy Connect options).
			Added VLAN-Tagged Ethernet Frame Format diagram.
			Added Ethernet Control Pause Frame Format diagram.
			Added Tri-Speed 10/100/1G Ethernet MAC Core System Block Diagram (Gigabit MAC option).
			Added Tri-Speed 10/100/1G Ethernet MAC Core System Block Diagram (SGMII Easy Connect option).
			Updated Signal Descriptions table.
			Updated Parameter Descriptions table.
			Updated Receiving a PAUSE Frame text section.
			Updated Receive Statistics Vector Description table.

Date	Document Version	IP Core Version	Change Summary
			Updated Internal Data Buffer and FIFO Interfaces text section.
			Updated Internal Registers text section.
			Updated MODE (R/W) Register Descriptions table.
			Updated Transmit and Receive Control (R/W) Register Descriptions table.
			Added the Core Generation text section.
June 2008	02.2	2.6	TSMAC IP core Internal Registers table - Corrected the POR value for Inter-Packet Gap Register. Value changed to 000CH.
January 2008	02.1	2.5	Updated part numbers listed in appendices. U2 changed to U3.
January 2008	02.0	2.5	Updated the Core Generation and Simulation section.
January 2008	01.9	2.5	Added Software Requirements text section.
November 2007	01.8	2.4	Updated Performance and Resource Utilization table in the LatticeXP2 appendix.
September 2007	01.7	2.4	Updated MAC Address Register mnemonic and POR.
September 2007	01.6	2.4	Updated description for Bit 30 in the Receive Statistics Vector Description table.
August 2007	01.5	2.4	Revised Transmit and Receive Control Register Description. Updated description and diagram for Successful Transmission of a 64-Byte Frame -Tx MAC Application Interface.
May 2007	01.4	2.4	Added support for LatticeXP2 family.
November 2006	01.3	2.3	Added support for LatticeECP2M family.
July 2006	01.2	2.2	Added support for LatticeSC family.
May 2006	01.1	2.1	Added support for LatticeECP2 family.
February 2006	01.0	2.0	Initial release.

Resource Utilization

This appendix gives resource utilization information for Lattice FPGAs using the TSMAC IP core. The IP configurations shown in this chapter were generated using the IPexpress software tool. IPexpress is the Lattice IP configuration utility, and is included as a standard feature of the Diamond design tools. Details regarding the usage of IPexpress can be found in the IPexpress and Diamond help systems. For more information on the Diamond design tools, visit the Lattice web site at: www.latticesemi.com/software.

LatticeECP3 FPGAs

Table 8-1. Performance and Resource Utilization[1]

Configuration		SLICEs	LUTs	Registers	EBRs	External Pins	f_{MAX} (MHz)
MIIM Module	Operational Option						
No	Classic	1232	1721	1193	2	25	125
No	Gigabit	1037	1427	1061	1	22	125
No	SGMII	1227	1718	1173	2	4	125
Yes	Classic	1355	1872	1345	2	27	125
Yes	Gigabit	1186	1596	1213	1	24	125
Yes	SGMII	1371	1881	1325	2	6	125

1. Performance and utilization data are generated targeting an LFE3-95EA-8FN484C device using Lattice Diamond 3.2 and Synplify Pro for Lattice I-2013.09L-SP1-1 software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeECP3 family.

Ordering Part Number

The Ordering Part Number (OPN) for the Tri-Speed Ethernet Media Access Controller IP core targeting LatticeECP3 devices is TS-MAC-E3-U4.

ECP5 (LFE5U) FPGAs

Table 8-2. Performance and Resource Utilization[1]

Configuration		SLICEs	LUTs	Registers	EBRs	External Pins	f_{MAX} (MHz)
MIIM Module	Operational Option						
No	Classic	1245	1694	1193	2	25	125
No	Gigabit	1080	1430	1061	1	22	125
No	SGMII	1249	1696	1173	2	4	125
Yes	Classic	1408	1849	1345	2	27	125
Yes	Gigabit	1263	1582	1213	1	24	125
Yes	SGMII	1423	1857	1325	2	6	125

1. Performance and utilization data are generated targeting an LFE5U-85F-8BG756C device using Lattice Diamond 3.2 and Synplify Pro for Lattice I-2013.09L-SP1-1 software. Performance may vary when using a different software version or targeting a different device density or speed grade within the Sapphire family.

Ordering Part Number

The Ordering Part Number (OPN) for the Tri-Speed Ethernet Media Access Controller IP core targeting LFE5U devices is TS-MAC-E5-U/TS-MAC-E5-UT.

ECP5 (LFE5UM) FPGAs

Table 8-3. Performance and Resource Utilization[1]

Configuration							
MIIM Module	Operational Option	SLICEs	LUTs	Registers	EBRs	External Pins	f_{MAX} (MHz)
No	Classic	1245	1694	1193	2	25	125
No	Gigabit	1080	1430	1061	1	22	125
No	SGMII	1249	1696	1173	2	4	125
Yes	Classic	1408	1849	1345	2	27	125
Yes	Gigabit	1263	1582	1213	1	24	125
Yes	SGMII	1423	1857	1325	2	6	125

1. Performance and utilization data are generated targeting an LFE5UM-85F-8BG756C device using Lattice Diamond 3.2 and Synplify Pro for Lattice I-2013.09L-SP1-1 software. Performance may vary when using a different software version or targeting a different device density or speed grade within the Sapphire family.

Ordering Part Number

The Ordering Part Number (OPN) for the Tri-Speed Ethernet Media Access Controller IP core targeting LFE5UM devices is TS-MAC-E5-U/TS-MAC-E5-UT.

LatticeXP2 FPGAs

Table 8-4. Performance and Resource Utilization[1]

Configuration							
MIIM Module	Operational Option[2]	SLICEs	LUTs	Registers	EBRs	External Pins	f_{MAX} (MHz)
No	Classic	1328	1823	1197	2	25	125
No	Gigabit	1110	1510	1061	1	22	125
Yes	Classic	1482	2000	1352	2	27	125
Yes	Gigabit	1285	1668	1213	1	24	125

1. Performance and utilization data are generated targeting an LFXP2-17E-6F484C device using Lattice Diamond 3.2 and Synplify Pro for Lattice I-2013.09L-SP1-1 software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeXP2 family.
2. The SGMII Easy Connect option is only available on device families with SERDES I/O.

Ordering Part Number

The Ordering Part Number (OPN) for the Tri-Speed Ethernet Media Access Controller IP core targeting LatticeXP2 devices is TS-MAC-X2-U4.