LatticeCORE

LatticeECP3 and ECP5 10 Gb Ethernet MAC IP Core User Guide

December 2014
IPUG114_1.0

Table of Contents

Introduction

This document provides technical information about the Lattice 10 Gigabit (10 Gb) Ethernet Media Access Controller (MAC) Intellectual Property (IP) core. The 10 Gb Ethernet MAC 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 10 Gb Ethernet MAC IP core for LatticeECP3™ and ECP5™ devices.

Table 1-1. 10 Gb Ethernet MAC IP Core Quick Facts

		10 Gb Ethernet MAC IP Configuration	
Core Requirements	FPGA Families Supported	LatticeECP3, ECP5	
	Minimal Device Needed	LFE3-17EA-8FTN25	LFE5UM-25F-8MG285C
Resources Utilization	Target Device	LFE3-150EA-8FN672C	LFE5UM-85F-8BG756CES
	Data Path Width	64	
	LUTs	4600-5100	
	sysMEM EBRs	4	
	Registers	2600-2900	
Design Tool Support	Lattice Implementation	Lattice Diamond™ 3.4	
	Synthesis	Synopsys® Synplify™ Pro for H-2014.03L	
		Lattice Synthesis Engine (ECP5 only)	
	Simulation	Aldec® Active-HDL® 9.3 Lattice Edition	
		Mentor Graphics® ModelSim® SE 10.0	

Features

- Compliant to IEEE 802.3-2005 standard, successfully passed University of New Hampshire InterOperability Laboratory (UNH-IOL) 10GbE MAC hardware tests[1]
- Supports standard 10Gbps Ethernet link layer data rate
- 64-bit wide internal data path operating at 156.25MHz
- XGMII interface to the PHY layer (using IODDR external to the core)
- Simple FIFO interface with user's application
- Optional Multicast address filtering
- Transmit and receive statistics vector
- Optional statistics counters of length from 16 to 40 bits for all devices (statistic counters are external to the core)
- Programmable Inter Frame Gap

1. Successfully passed all UNH-IOL Clause 4 (MAC), Clause 31 (Flow Control) and Clause 46 (Reconciliation Sublayer) testing.

- Supports:
  - Full duplex operation
  - Flow control using PAUSE frames
  - VLAN tagged frames
  - Automatic padding of short frames
  - Optional FCS generation during transmission
  - Optional FCS stripping during reception
  - Jumbo frames up to 16k
  - Inter frame Stretch Mode during transmission
  - Deficit Idle Count

This chapter provides a functional description of the 10 Gb Ethernet MAC IP core. Figure 2-1 shows a top-level interface diagram for the 10 Gb Ethernet MAC IP.

Figure 2-1. 10 Gb Ethernet MAC Core Block Diagram

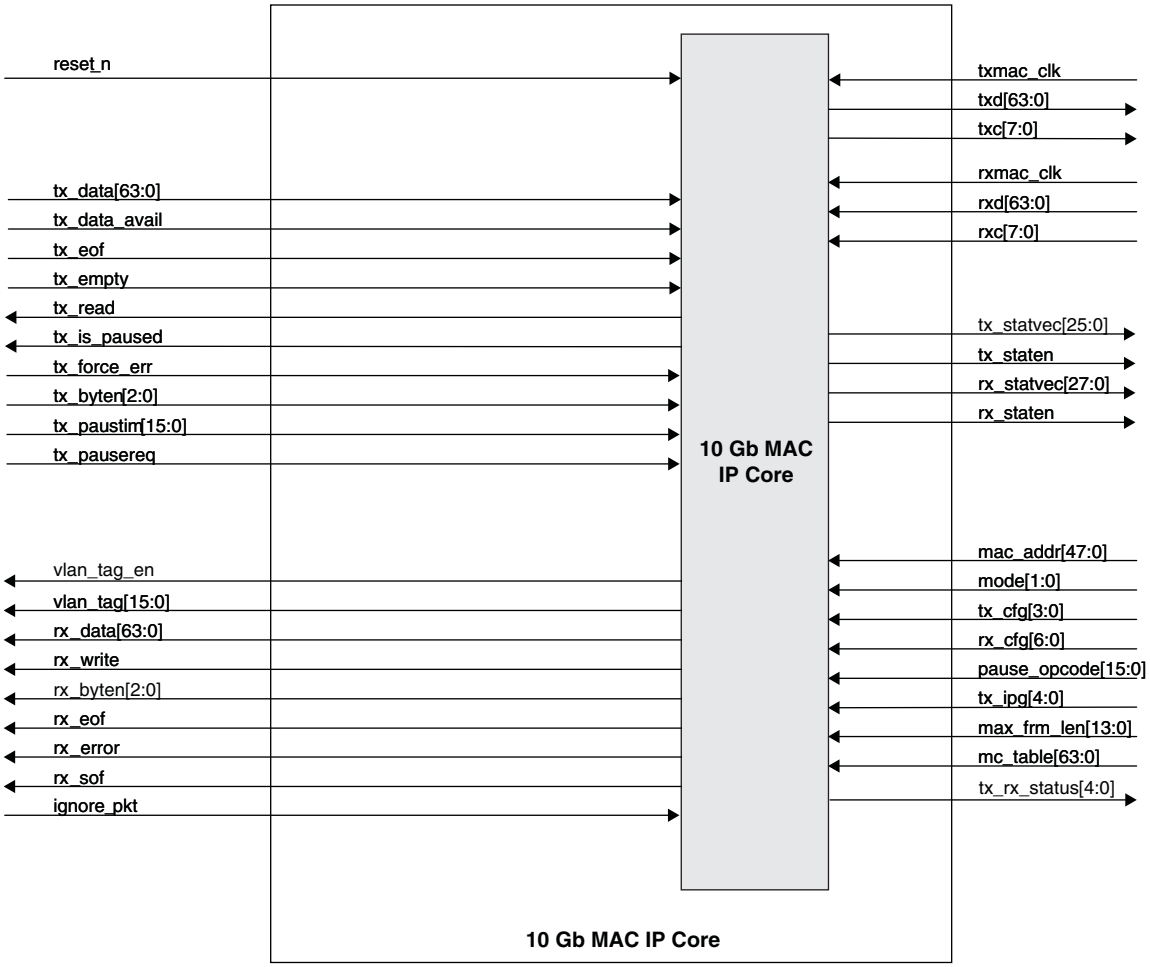
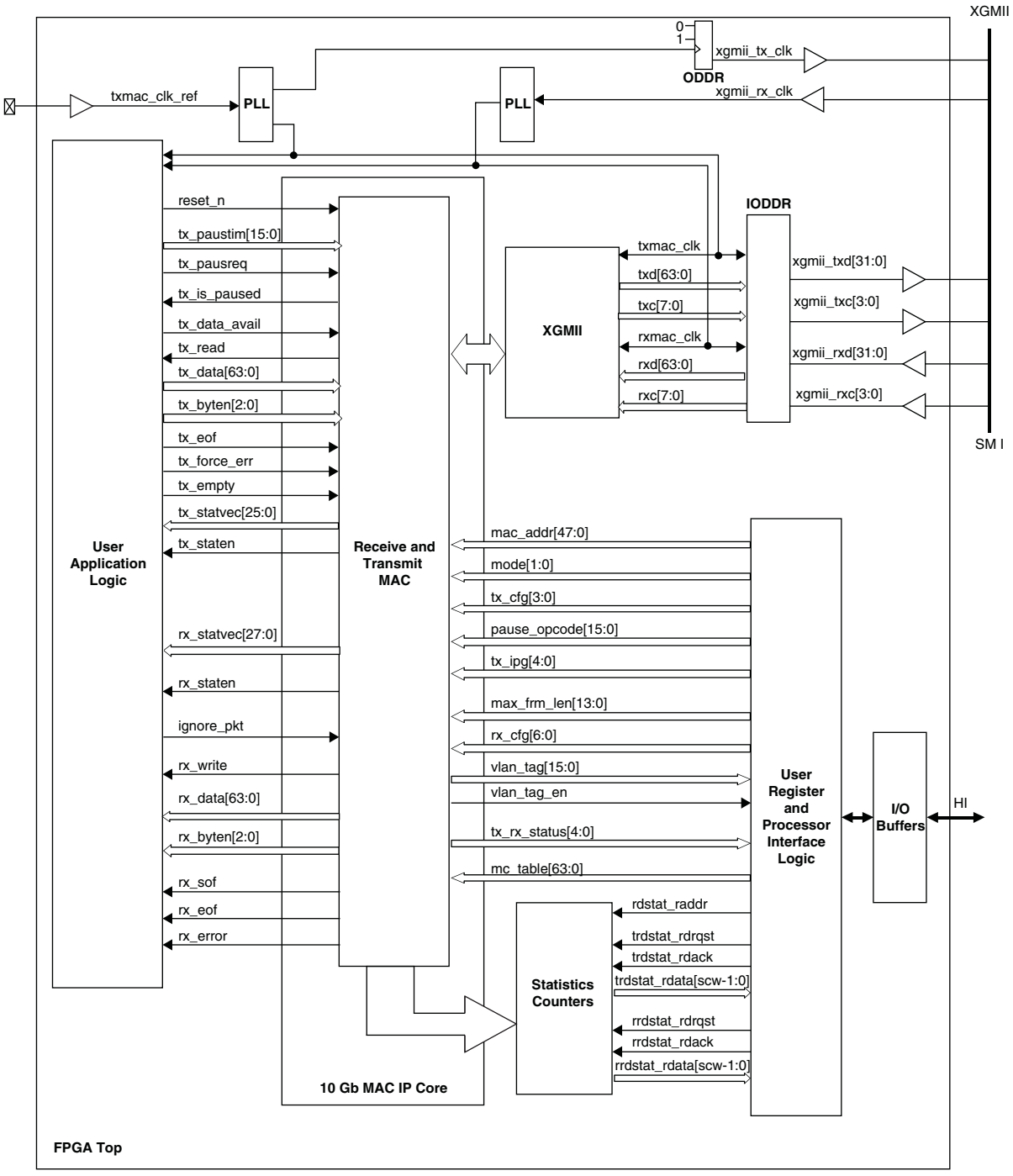


Figure 2-2 shows a system block diagram for the 10 Gb Ethernet MAC IP core.

Figure 2-2. 10 Gb Ethernet MAC Core System Block Diagram



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

The format of an untagged Ethernet frame is shown in Figure 2-3. The format of a tagged Ethernet frame is shown in Figure 2-4.

Figure 2-3. Untagged Ethernet Frame Format

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

Figure 2-4. Tagged Ethernet Frame Format

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

The MAC is responsible for constructing a valid frame from the data received from the user's application before transmitting it. On the receive path, it receives frames from the network through the XGMII interface and passes the parameters of the frame to the MAC client (FIFO interface). The transmit path logic accepts the frame from the user application through a FIFO interface. The fields of the Ethernet frame that are expected from the application interface are shown in Figure 2-5 and 2-6.

On the transmit path, the MAC can be programmed in one of two modes. In the first mode (Figure 2-5), when the frame from the user contains the FCS along with the DESTINATION ADDRESS, SOURCE ADDRESS, LENGTH / TYPE and Data, the Transmit MAC adds the SOF, preamble and SFD before transmitting the frame. This mode can be set by enabling the tx_pass_fcs bit in the TX_CTL register. In the second mode (Figure 2-6), the MAC calculates the number of bytes to be padded as well (if required) in addition to the FCS for the entire frame and adds the SOF, preamble and SFD before transmitting the frame.

Similarly, on the receive path, the MAC can be programmed to transfer the frame as it was received to the FIFO (Promiscuous mode) or after stripping off the FCS and any pad fields. In all cases the SOF, preamble and SFD bytes will always be stripped off the frame before it is transferred on the FIFO interface.

Figure 2-5. Ethernet Frame with Frame Check Sequence

DESTINATION ADDRESS	SOURCE ADDRESS	LENGTH/ TYPE	DATA/PAD 46-1500 bytes	FRAME CHECK SEQUENCE

Figure 2-6. Ethernet Frame with out Frame Check Sequence

DESTINATION ADDRESS	SOURCE ADDRESS	LENGTH/ TYPE	DATA/PAD 46-1500 bytes

The core has a number of inputs, which are used to provide control of the various modes of operation. Registers have been included at the top level of the FPGA. These registers are not part of the core. They are discussed in an appendix. The functionality of the MAC core is described in the following sections.

Receive MAC

The receive MAC is responsible for receiving the incoming frames and transferring them to the FIFO. In the process, it performs the following operations:

- Checks the frame for a valid SOF and SFD symbol.
- Determines whether the frame should be received by analyzing the Destination Address.
- Determines the type of the frame by analyzing the Length/Type field.
- Checks for any errors in the frame by recalculating the CRC and comparing it with the expected value.

The user can specify whether the FCS field should be transferred on to the FIFO interface by programming the receive MAC configuration register. If the FCS field is to be stripped off the frame, any padding bytes within the frame will be stripped off as well.

Once a valid start of frame is detected, the destination address field of the incoming frame is analyzed. If the destination address was a unicast address, it is compared with the programmed MAC address. Unless the prms bit was set in the RX_CTL register, the incoming frame will be discarded if the destination address field and the programmed MAC address do not match.

If the frame had a multicast address and if receive_all_mc signal is not asserted, all such frames are dropped (except pause frames). If the frame had a multicast address and if receive_all_mc signal is asserted, the multicast frames are subject to address filtering rules as described below.

For all frames with multicast address, the CRC of the destination address is computed and the mid six bits of the least significant byte of the CRC is chosen as the address to a hash table. The MAC implements an eight row table with eight bits in each row. The lower three bits of the selected CRC are used to select one of these eight rows and the next three bits are used to select one of the bits in the selected table. The incoming multicast frame is accepted if the bit selected from the hash table was set to one. It is discarded if the bit selected was zero.

If the incoming frame had a broadcast address it will be accepted if the either the prms or the receive_bc bit was set. A broadcast frame is discarded if none of these bits are set.

A statistics vector that provides information about the incoming frame is generated when an incoming frame is received and transferred on to the FIFO interface. A vector is not generated for all those frames that are discarded (No address match or frame length less than 64 bytes) or ignored (user asserts the ignore_pkt signal).

The vector is used to drive the management information block counters that keep track of all the happenings on the line. The composition of the statistics vector is shown in Table 2-1.

Table 2-1. Rx MAC Statistics Vector

Bit	Description
25	Receive Packet Ignored
24	Minimum IPG Violated
23	Unsupported Opcode
21	Frame Too Long
20	In Range Length Error
19	PAUSE Frame
18	VLAN Tag Detected
17	CRC Error
16	Multicast Frame Received
15	Broadcast Frame Received
13:0	Frame Byte Count

The full condition of the FIFO can be avoided by asserting the ignore_pkt signal. At the time a new frame is received, the receive logic samples this signal to determine whether the frame should be received.

Latency

Since the frame is buffered before being sent on the FIFO interface, there will be an initial latency. The first 64 bytes of a frame are buffered before they are sent out. Since the internal data path is 64 bits wide, the latency from the time a frame appears at the XGMII input to the time it begins to transfer on to the FIFO interface will be eight clock cycles.

Transmit MAC

The main functions of the Transmit MAC are:

- Data padding for short frames when FCS generation is enabled.
- Generation of a pause frame when the tx_pausreq signal is asserted.
- To stop frame transmission when a Pause frame is received by the Receive MAC.
- Implement link fault signaling logic and transmit appropriate sequences based on the remote link status.

The TX_CTL register controls the operation of this module. For every frame transmitted, a statistics vector is generated. The composition of the vector is shown in Table 2-2.

Table 2-2. Transmit MAC Statistics Vector

Bit	Description
25	Long Frame Error
24	Terminate Error
23	Length Check Error
22	CRC Error
21	Underrun Error
20	Multicast Address
19	Broadcast Address
18	Tagged Frame
17	Jumbo Frame
16	MAC Control Inserted by Client
15	MAC Control Inserted by MAC
14	Transmit OK
13-0	Transmitted Bytes

Statistics Counters

This module provides counters for all of the bits in the statistics vectors. It also implements statistic counters from RFC2819 and RMON. The width of the counters is specified through a parameter and can have a value from 16 to 40bits for all devices. The counters are read through an interface that returns the complete counter value in response to a read request. An acknowledgement is returned to indicate that the counter value on the read data bus is valid. There are separate request, acknowledge, and data buses for transmit counters and receive counters. The address bus is shared for transmit and receive counters. The Statistics Counter is a part of the reference design. A list of statistic counters is provided in Table 5-1 on page 31.

PHY Interface

The Reconciliation sublayer interface is implemented as a 64-bit wide single edge data bus and an 8-bit wide single edge control bus. To allow this interface to conform to the XGMII definition DDR I/O cells are added at the top level of the FPGA when this core is implemented.

Register Description

There are no registers in this core. All control and status information is passed between this core and the top level of the device through individual I/O ports on the core. Registers must be added to the top level to control and monitor these ports. A reference description of a set of registers to do this is included as an appendix to this user's guide.

Signal Descriptions

Table 2-3. 10 Gb Ethernet MAC IP Core Input and Output Signals

Port Name	Active State	I/O Type	Description
reset_n	Low	Input	Asynchronous reset signal – Resets the entire core when asserted.
tx_paustim[15:0]	N/A	Input	Contains parameters to be transmitted in a pause frame. Valid when tx_pausreq is asserted.
tx_pausreq	Positive Edge	Input	Asserted to initiate the transmitting of a pause frame.
tx_is_paused	High	Output	Active when the transmitter has been placed in the pause state by a received pause frame.
tx_data_avail	High	Input	Asserted to alert the MAC transmitter that the transmit FIFO has data ready for transmission.
tx_read	High	Output	Transmit FIFO read request, asserted by the MAC transmitter in response to signal tx_data_avail.
tx_data[63:0]	N/A	Input	Data read from transmit FIFO. The least significant byte (bits 7:0) is sent out on the link first.
tx_byten[2:0]	N/A	Input	Indicates the valid bytes on the tx_data bus. Must be 7 except when tx_eof is active. 0 - tx_data[7:0] valid 1 - tx_data[15:0] valid 2 - tx_data[23:0] valid 3 - tx_data[31:0] valid 4 - tx_data[39:0] valid 5 - tx_data[47:0] valid 6 - tx_data[55:0] valid 7 - tx_data[63:0] valid
tx_eof	High	Input	End of frame signal asserted with the last segment of the frame.
tx_force_err	High	Input	Indicates that the current frame has errors. This input is qualified with input signal tx_eof.
tx_empty	High	Input	When asserted, indicates that the transmit FIFO is empty.
tx_statvec[25:0]	N/A	Output	Contains information on the frame transmitted (details given in the Functional Description section of this document). This bus is qualified by the tx_staten signal.
tx_staten	High	Output	When asserted, indicates that the contents of the tx_statvec bus are valid. This signal is asserted for 3 txmac_clk periods.
rx_statvec[27:0]	N/A	Output	Contains information on the frame received (details given in the Functional Description section of this document). This bus is qualified by the rx_staten signal.
rx_staten	High	Output	When asserted, indicates that the contents of the rx_statvec bus are valid. This signal is asserted for 3 rxmac_clk periods.
ignore_pkt	High	Input	Asserted to prevent a receive FIFO full condition. The Receive MAC will continue dropping packets as long as this signal is asserted.
rx_write	High	Output	Driven by the MAC core to request a receive FIFO write.
rx_data[63:0]	N/A	Output	Contains data that is to be written into the receive FIFO.
rx_byten[2:0]	N/A	Output	Indicates the valid bytes on the rx_data bus. Must be 7 except when rx_eof is active. 0 - rx_data[7:0] valid 1 - rx_data[15:0] valid 2 - rx_data[23:0] valid 3 - rx_data[31:0] valid 4 - rx_data[39:0] valid 5 - rx_data[47:0] valid 6 - rx_data[55:0] valid 7 - rx_data[63:0] valid
rx_sof	High	Output	Start of frame signal asserted with the first segment of the frame.

Table 2-3. 10 Gb Ethernet MAC IP Core Input and Output Signals (Continued)

Port Name	Active State	I/O Type	Description
rx_eof	High	Output	End of frame signal asserted with the last segment of the frame.
rx_error	High	Output	When asserted, indicates that the frame had a length error, a termination error, or a CRC error. This signal is qualified with the rx_eof signal.
txmac_clk	N/A	Input	156.25 MHz MAC transmit clock. Ethernet frame transmit data is output on the rising edge of this clock.
txd[63:0]	N/A	Output	Single data rate Ethernet frame transmit data. Converted to double data rate XGMII transmit data signals with FPGA ODDR circuit elements external to the IP core.
txc[7:0]	High=control Low=data	Output	Asserted by the MAC to indicate on a per byte basis that the txd bus contains data or control. Converted to double data rate XGMII transmit control signals with FPGA ODDR circuit elements external to the IP core.
rxmac_clk	N/A	Input	156.25 MHz MAC receive clock. Receive data is presented to the receive MAC coincident with the rising edge of this clock.
rxd[63:0]	N/A	Input	Single data rate Ethernet frame receive data. Converted from double data rate XGMII receive data signals with FPGA IDDR circuit elements external to the IP core.
rxc[7:0]	High=control Low=data	Input	Indicates on a per byte basis that the rxd bus contains data or control.
mac_addr[47:0]	N/A	Input	Unicast address for the MAC. This address is received from or sent to the link from most significant byte to least significant byte. For MAC address: AC-DE-48-00-00-80, AC is sent first and 80 is sent last.
mode[1:0]	High	Input	Bit 0 (rx_en) - Enables the receive side of the MAC. Bit 1 (tx_en) - Enables the transmit side of the MAC.
tx_cfg[3:0]	High	Input	Bit 0 (tx_pass_fcs) - When zero, the MAC generates and inserts FCS in outgoing packets. Bit 1 (transmit_pause_en) - Enables the transmit side of the MAC to support flow control. Bit 2 (tx_ipg_stretch) - Enables the transmit side of the MAC to insert the proper amount of inter-frame gap to support matching rate of OC192. Bit 3 (transmit_short) - Enables the transmit side of the MAC to transmit short frames.
pause_opcode[15:0]	N/A	Input	Supplies opcode for transmit pause frames.
tx_ipg[4:0]	N/A	Input	Specifies the amount of inter-frame gap in increments of 4 bytes. 0 - indicates the minimum value of 8 bytes.
max_frm_len[13:0]	N/A	Input	Specifies the maximum frame length. A frame longer than this will be marked as long.
rx_cfg[6:0]	High	Input	Bit 0 (prms) - Enables the receive side of the MAC to receive frames without doing any address filtering. Bit 1 (rx_pass_fcs) - When zero, the MAC discards the FCS of incoming packets after checking it. Bit 2 (rx_pause_en) - Enables the receive side of the MAC to support flow control. Bit 3 (receive_all_mc) - Enables the receive side of the MAC to receive multicast frames as per address filtering rules. Bit 4 (receive_bc) - Enables the receive side of the MAC to receive broadcast frames. Bit 5 (receive_short) - Enables the receive side of the MAC to receive short frames. Bit 6 (drop_mac_ctrl) - Enables the receive side of the MAC to drop MAC Control packets before passing them on to client.

Table 2-3. 10 Gb Ethernet MAC IP Core Input and Output Signals (Continued)

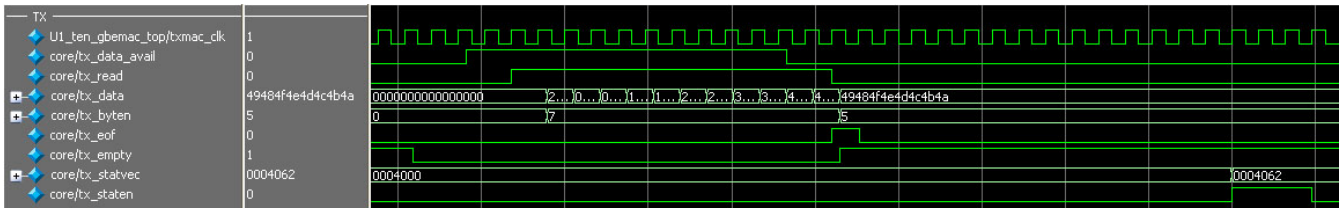
Port Name	Active State	I/O Type	Description
vlan_tag[15:0]	N/A	Output	The most recently received VLAN tag.
vlan_tag_en	High	Output	When asserted, indicates that the contents of the vlan_tag bus are valid. This signal is asserted for 3 rxmac_clk periods.
tx_rx_status[4:0]	High	Output	Bit 0 (tx_idle) - When asserted indicates that the transmitter is idle. Bit 1 (rx_idle) - When asserted indicates that the receiver is idle. Bits[4:2] (link_sts[2:0]) - Contains the RS layer status of the link. (1=local fault, 2= remote fault, 4=link ok)
Multicast Address Filtering Optional Signals			
mc_table[63:0]	N/A	Input	Multicast table

Timing Specifications

Transmit Interface

When there is a packet ready for transmission, the tx_data_avail is asserted. This signal should stay active until the MAC asserts the tx_read, after which MAC will ignore the status of tx_data_avail and can be deactivated. The data is available one clock cycle after the MAC asserts the tx_read. The MAC continues reading the packet until it gets tx_eof active. The tx_eof signal is asserted when the last byte of data is transmitted. Internally the MAC buffers the data before it starts transmission of a packet. The MAC may stop/continue reading to maintain the level of the buffer due to factors like IPG insertion and Pause reception. Hence, the tx_read signal can be deactivated anytime during the transfer. Once the MAC receives tx_eof, the MAC will again start monitoring the tx_data_avail signal. The tx_staten signal will be asserted once the entire frame is received from the user's interface.

Figure 2-7. Transmission of a 64 Data Byte Frame

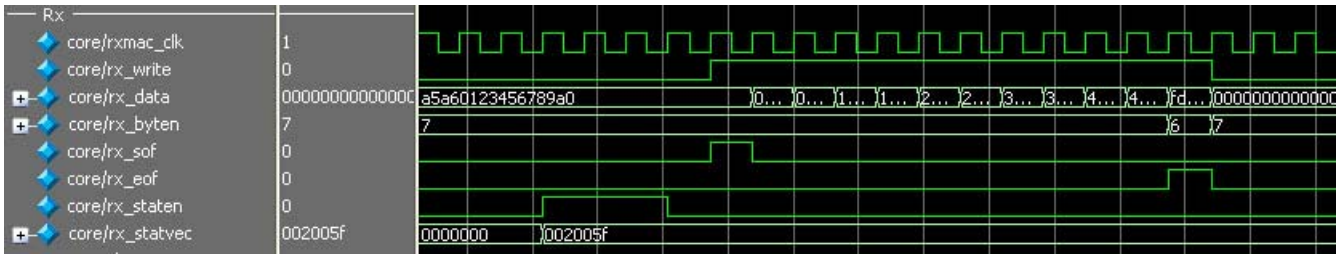


Note: In the above figure, a scenario is demonstrated wherein the tx_read signal is deactivated before the assertion of the tx_eof signal so that the buffer level is maintained due to factors like IPG insertion and Pause reception.

Receive Interface

When the MAC receives the data, it asserts rx_write. The MAC asserts rx_sof when it writes the first byte of data. The rx_write signal may be de-asserted any time before the end of a packet. In this case the value on rx_data signal is not valid. The end of a packet is indicated with the assertion of rx_eof signal.

Figure 2-8. Reception of a 64 Data Byte Frame



The IPexpress™/Clarity Designer tool is used to create IP and architectural modules in the Diamond software. Refer to "IP Core Generation" on page 14 for a description on how to generate the IP.

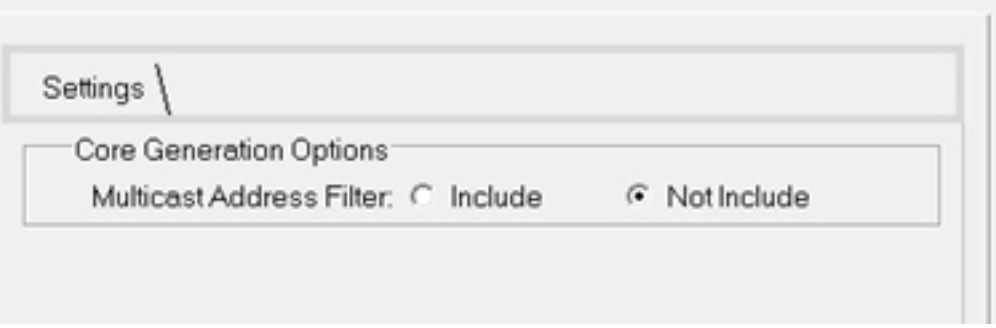
Table 3-1 provides the list of user configurable parameters for the 10 Gb Ethernet MAC IP core. The parameter settings are specified using the 10 Gb Ethernet MAC IP core Configuration GUI in IPexpress.

Table 3-1. 10 Gb Ethernet MAC IP Core Configuration Parameters

Parameter	Value Range	Default
Core Generation Options		
Multicast Address Filter	Include/Not Include	Not Include

Figure 3-1 shows the 10 Gb Ethernet MAC IP Core configuration dialog box.

Figure 3-1. 10 Gb Ethernet MAC IP Core Configuration Dialog Box



Parameter Descriptions

This section describes the available parameters for the 10 Gb Ethernet MAC IP core.

Multicast Address Filter

This parameter determines whether the optional Multicast Address Filtering will be included in the core's implementation.

IP Core Generation

This chapter provides information on how to generate the 10 Gb Ethernet MAC IP core using the Diamond software IPexpress/Clarity Designer 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 10 Gb Ethernet MAC 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 10 Gb Ethernet MAC IP core and fully evaluate the core through functional simulation and implementation (synthesis, map, place and route) without an IP license. The 10 Gb Ethernet MAC 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 30 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.

IP Core Generation in IPexpress

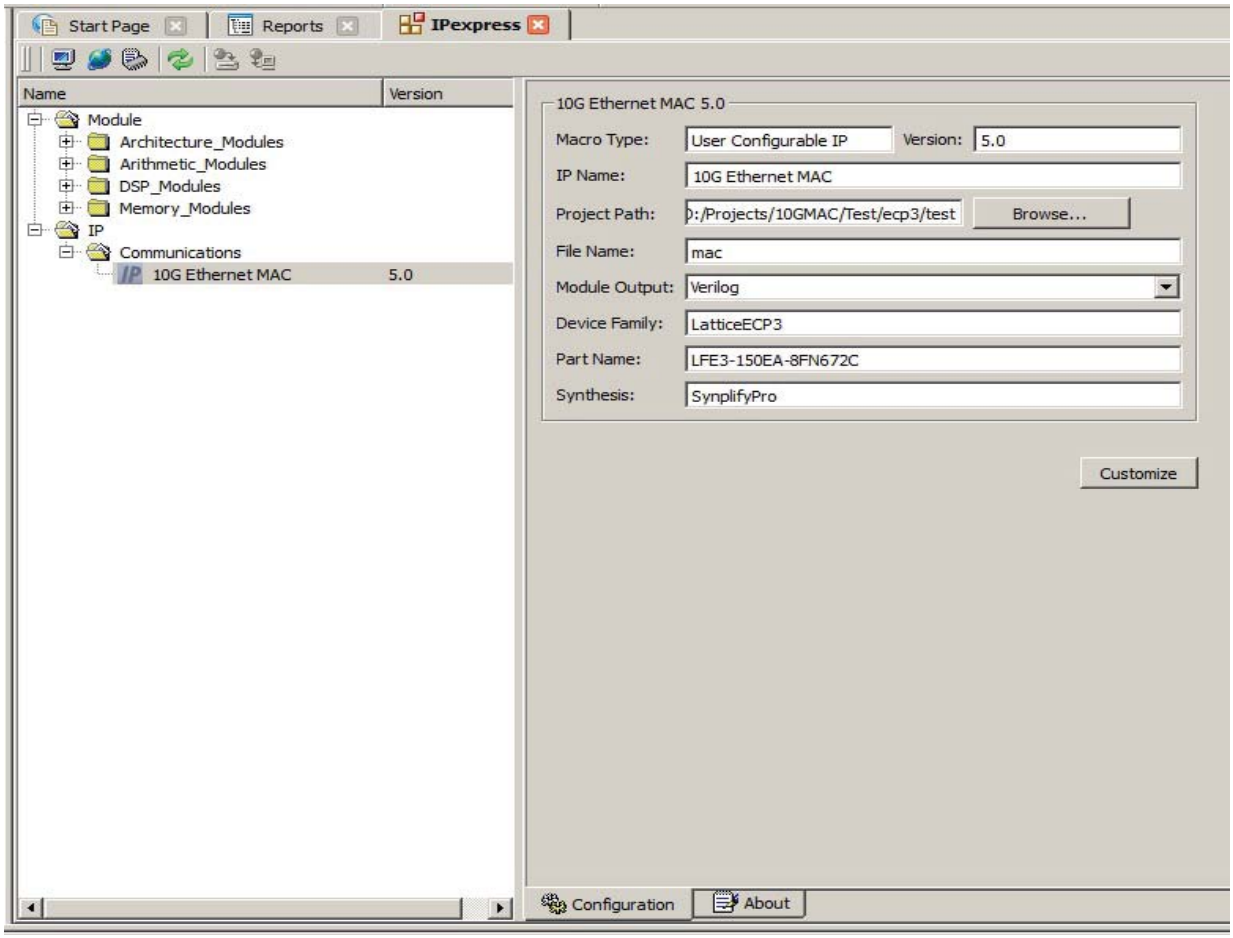
Getting Started

The 10 Gb Ethernet MAC 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 10 Gb Ethernet MAC 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.
- **Diamond Module Output** – Verilog or VHDL.
- **Device Family** – Device family to which IP is to be targeted (e.g. LatticeECP3). Only families that support the particular IP core are listed.
- **Part Name** – Specific targeted part within the selected device family.

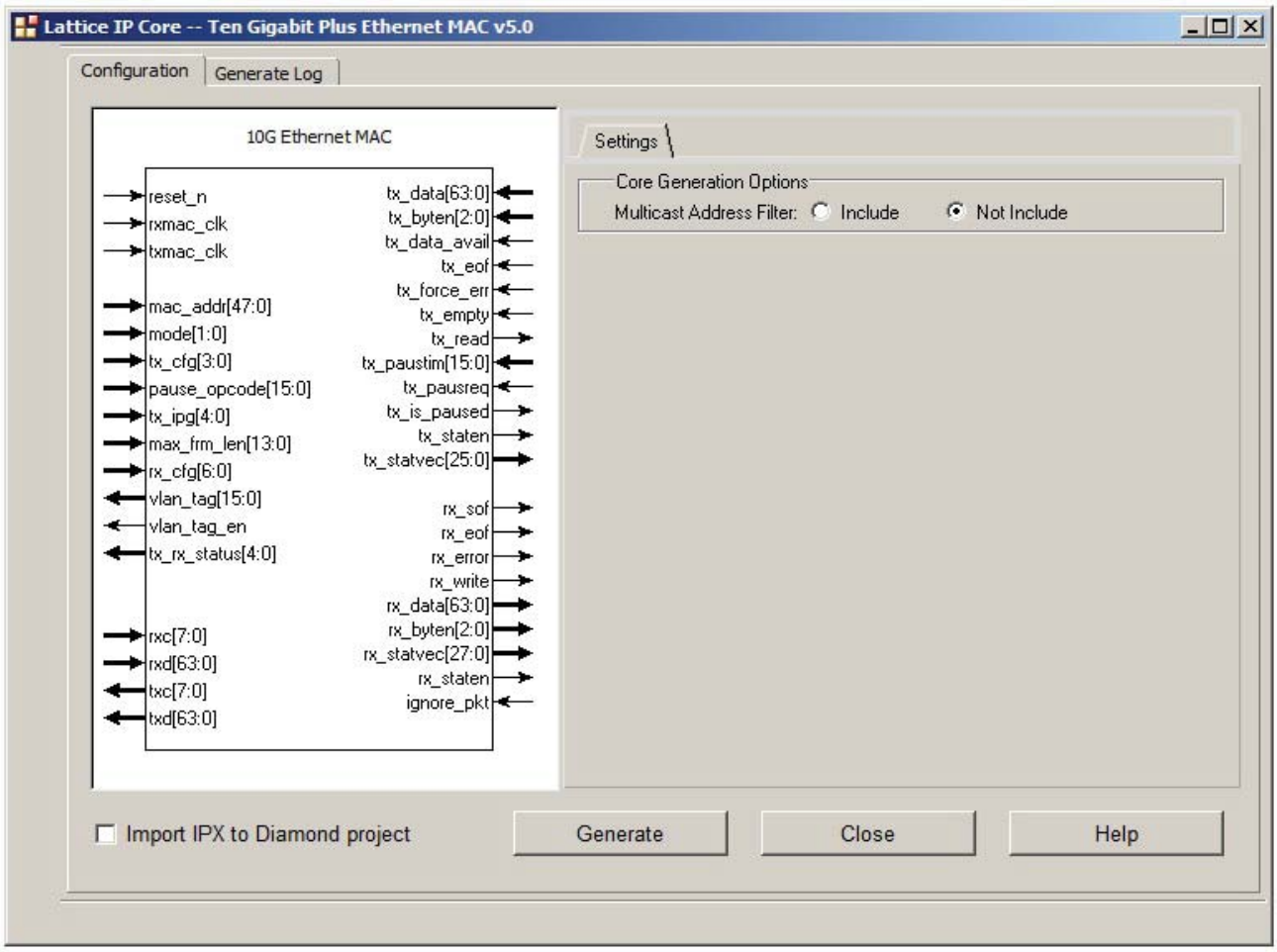
Figure 4-1. IPexpress Dialog Box



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 10 Gb Ethernet MAC 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" on page 13 for more information on the 10 Gb Ethernet MAC IP core parameter settings.

Figure 4-2. 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. IPexpress 10Gb Ethernet MAC IP Core Directory Structure

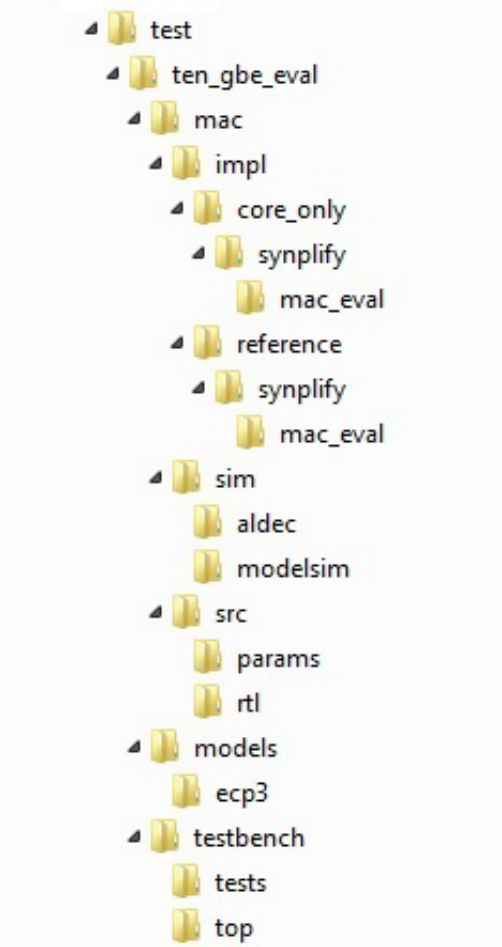


Table 4-1 provides a list of key files and directories created by the IPexpress tool and how they are used. The IPexpress tool creates several files that are used throughout the design cycle. The names of most of the created files are customized to the user's module name specified in the IPexpress tool.

Table 4-1. File List

File	Description
<*username*>_inst.v	This file provides an instance template for the IP.
<*username*>.v	This file provides the 10 Gb Ethernet MAC core for simulation.
<*username*>_beh.v	This file provides a behavioral simulation model for the 10 Gb Ethernet MAC core.
<*username*>_params.v	This file provides parameters necessary for the simulation.
<*username*>_bb.v	This file provides the synthesis black box for the user's synthesis.
<*username*>.ngo	This file provides the synthesized IP core.

Table 4-1. File List (Continued)

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.

These are all of the files necessary to implement and verify the 10 Gb Ethernet MAC IP core in your own top-level design. The following additional files providing IP core generation status information are also generated in the "Project Path" directory:

- <*username*>_generate.log – Synthesis and map log file.
- <*username*>_gen.log – IPexpress IP generation log file.

The \<**ten_gbe_eval**> and subtending directories provide files supporting 10 Gb Ethernet MAC core evaluation. The \<**ten_gbe_eval**> directory shown in Figure 4-3 contains files/folders with content that is constant for all configurations of the 10 Gb Ethernet MAC. The \<*username*> subfolder (**ten_gbe_core0** in this example) contains files/folders with content specific to the username configuration.

The **ten_gbe_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. \<**ten_gbe_core0**>, \<**ten_gbe_core1**>, etc.

Instantiating the Core

The generated 10 Gb Ethernet MAC IP core package includes black-box (<*username*>_bb.v) and instance (<*username*>_inst.v) templates that can be used to instantiate the core in a top-level design. Two example RTL top-level reference source files are provided in\<*project_dir*>**ten_gbe_eval**\<*username*>**src\rtl**.

The top-level file <username>_eval_top.v is the same top-level that is used in the simulation model described in the next section. Designers may use this top-level reference as the starting template for the top-level for their complete design. Included in <username>_eval_top.v is logic, memory and clock modules supporting an XGMII interface loop back capability, a register module supporting programmable control of the 10 Gb Ethernet MAC core parameters and system processor interface. Verilog source RTL for these modules is provided in \<*project_dir*>**ten_gbe_eval**\<*username*>. The top-level configuration is specified via the parameters defined in the ten_gbemac_defines.v file in \<*project_dir*>**ten_gbe_eval**\<*username*>**src\params**. A description of the 10 Gb Ethernet MAC parameters is in the parameters section of this document. A description of the 10 Gb Ethernet MAC register layout for this reference design is provided in an appendix to this document.

The top-level file <username>_core_only_top.v supports the ability to implement just the 10 Gb Ethernet MAC core itself. This design is intended only to provide an accurate indication of the device utilization associated with the 10 Gb Ethernet MAC core and should not be used as an actual implementation example.

Running Functional Simulation

The functional simulation includes a configuration-specific behavioral model of the 10 Gb Ethernet MAC IP Core (<username>_beh.v) that is instantiated in an FPGA top level along with a client-side interface loop back capability module and register implementation module.

The top-level file supporting ModelSim eval simulation is provided in \<*project_dir*>**ten_gbe_eval**\<*username*>**sim\modelsim**. This FPGA top is instantiated in an eval testbench provided in \<*project_dir*>**ten_gbe_eval\testbench\top** that configures FPGA test logic registers and 10 Gb Ethernet MAC IP core control and status registers via an included test file stimulus_file.v provided

in \<*project_dir*>\ten_gbe_eval\testbench\tests. Note that the user can edit the stimulus_file.v file to configure and monitor whatever registers they desire.

Users may run the eval simulation by doing the following:

1. Open ModelSim.

2. Under the File tab, select **Change Directory** and choose folder \<*project_dir*>\ten_gbe_eval\<*username*>\sim\modelsim.

3. Under the Tools tab, select **Execute Macro** and execute one of the ModelSim "do" scripts shown.

The top-level file supporting Aldec Active-HDL® simulation is provided in \<*project_dir*>\ten_gbe_eval\<*username*>\sim\aldec. This FPGA top is instantiated in an eval testbench provided in \<*project_dir*>\ten_gbe_eval\testbench that configures FPGA test logic registers and 10 Gb Ethernet MAC IP core control and status registers via an included test file stimulus_file.v provided in \<*project_dir*>\ten_gbe_eval\testbench\tests. Note the user can edit the stimulus_file.v file to configure and monitor whatever registers they desire.

Users may run the eval simulation by doing the following:

1. Open Active-HDL.

2. Under the Console tab, change the directory to: <*project_dir*>\ten_gbe_eval\<*username*>\sim\aldec

3. Execute the Active-HDL "do" scripts shown.

The simulation waveform results will be displayed in the Aldec Active-HDL Wave window.

Synthesizing and Implementing the Core in a Top-Level Design

The 10 Gb Ethernet MAC IP core itself is synthesized and is provided in NGO format when the core is generated. 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 Synplify Pro.

Two example RTL top-level configurations supporting 10 Gb Ethernet MAC core top-level synthesis and implementation are provided with the 10 Gb Ethernet MAC IP core in \<*project_dir*>\ten_gbemac_eval\<*username*>\impl.

The top-level file ten_<username>_core_only_top.v provided in \<*project_dir*>\ten_gbemac_eval\<*username*>\src\rtl supports the ability to implement just the 10 Gb Ethernet MAC core. This design is intended only to provide an accurate indication of the device utilization associated with the core itself and should not be used as an actual implementation example.

The top-level file <username>_eval_top.v provided in \<*project_dir*>\ten_gbemac_eval\<*username*>\src\rtl supports the ability to instantiate, simulate, map, place and route the Lattice 10 Gb Ethernet MAC IP core in a complete example design. This reference design basically provides a loopback path for packets on the MAC Rx/Tx client interface, through a FIFO and associated logic. Ethernet packets are sourced to the Rx XGMII and looped back on the MAC Rx/Tx Client FIFO inter- face. Source and destination addresses in the Ethernet frame can be swapped so the looped back packets on the Tx XGMII have the correct source and destination addresses. This is the same configuration that is used in the evaluation simulation capability described previously. Note that implementation of the reference evaluation configuration is targeted to a specific device and package type for each device family.

Push-button implementation of both top-level configurations is supported via the Diamond project files, <username>_reference_eval.ldf and <username>_core_only_eval.ldf. These files are located in \<*project_dir*>\ten_gbemac_eval\<*username*>\impl\<configuration>\synplify.

To use this project file in Diamond:

1. Choose **File > Open > Project**.
2. Browse to **\<***project_dir***>\ten_gbemac_eval\<***username***>\impl\<configuration>\synplify** in the Open Project dialog box.
3. Select and open <*username*>.ldf. At this point, all of the files needed to support top-level synthesis and implementation will be imported to the project.
4. Select the **Process** tab in the left-hand GUI window.
5. Implement the complete design via the standard Diamond GUI flow.

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 IPexpress

To regenerate an IP core in IPexpress:

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**.

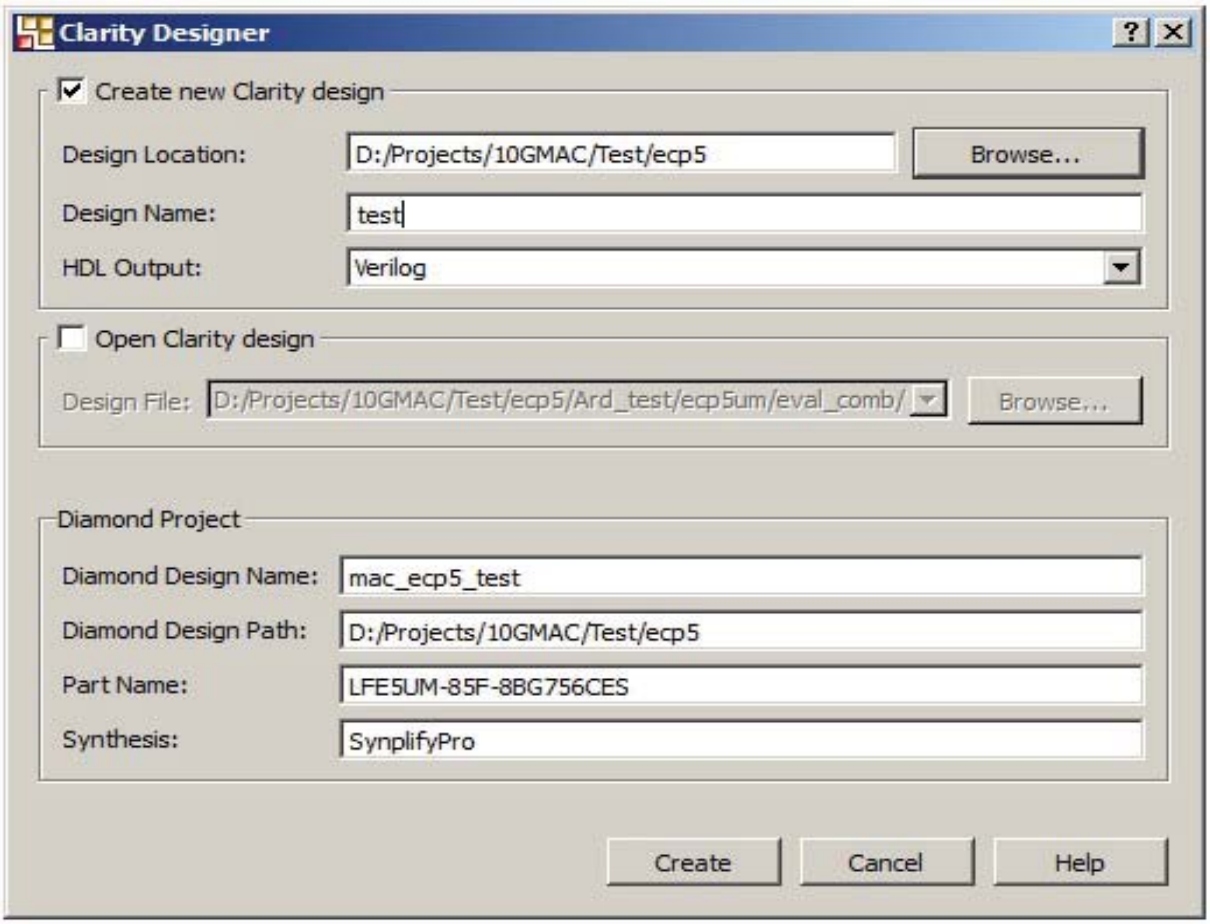
The IPexpress package file (.ipx) supported by Diamond holds references to all of the elements of the generated IP core required to support simulation, synthesis and implementation. The IP core may be included in a user's design by importing the .ipx file to the associated Diamond project. To change 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.

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.

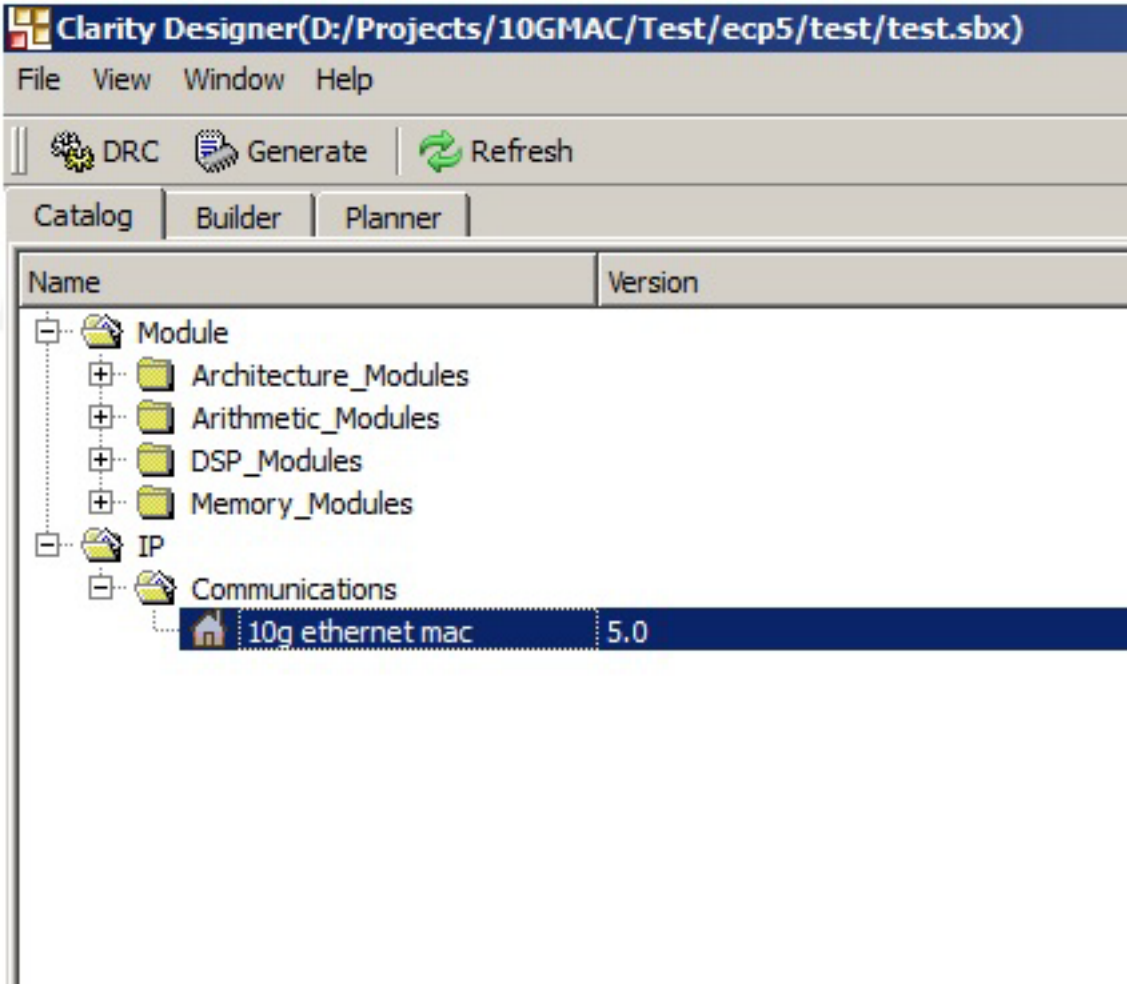
Figure 4-4. Starting a Project in Clarity Designer



To create a custom configuration, the user clicks the **Create** button to open the Clarity Designer main window as shown in Figure 4-4.

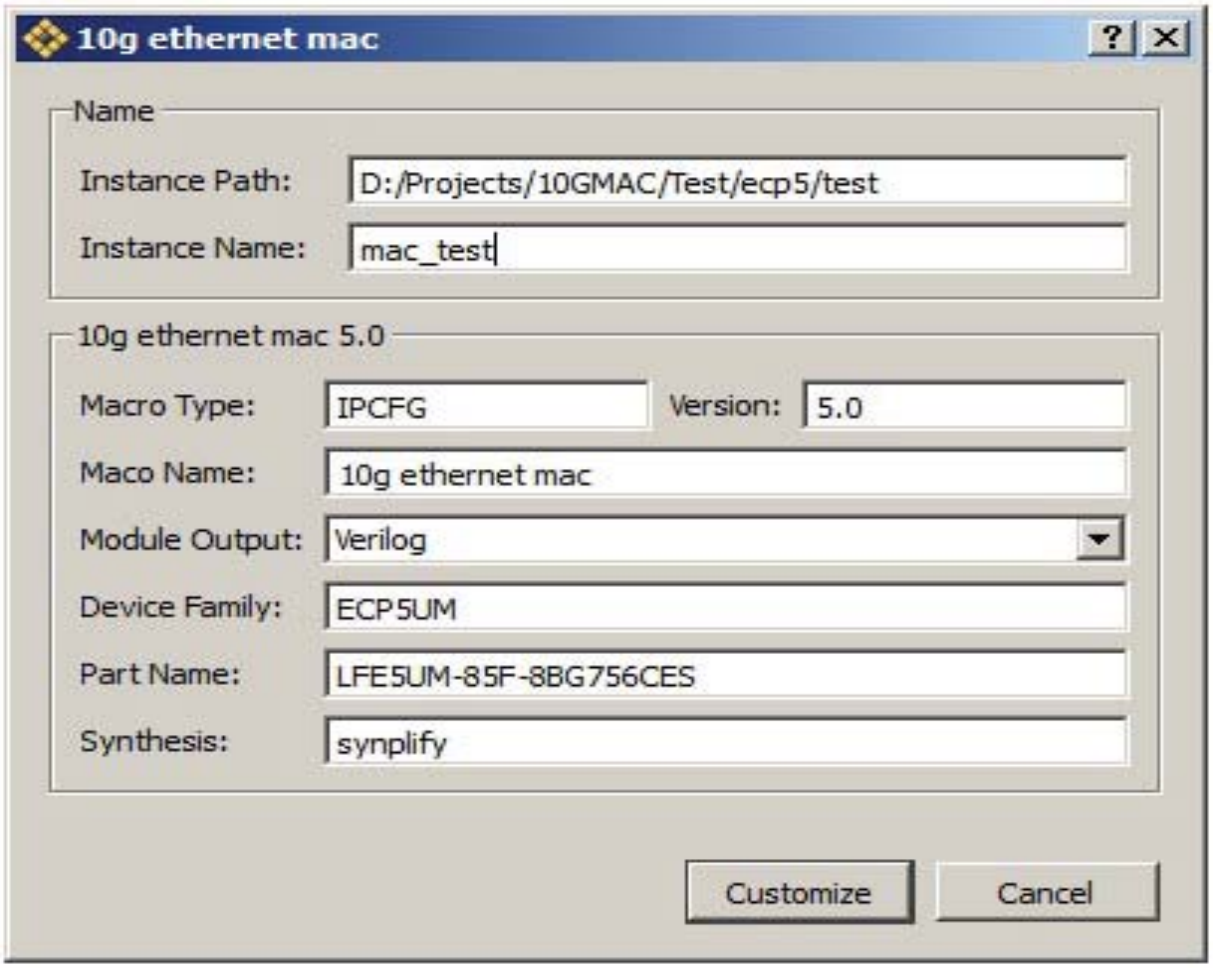
The 10 Gb Ethernet MAC IP core is available for download from the Lattice IP Server using the Clarity Designer tool. The IP files are automatically installed using ispUPDATE technology in any customer-specified directory. After the IP core has been installed, it will available in the Clarity Designer GUI Catalog box shown in Figure 4-5.

Figure 4-5. Clarity Designer Catalog Window



Double-click the IP name to open a dialog box as shown in Figure 4-6.

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 (e.g. ECP5). Only families that support the particular IP core are listed.
- **Part Name** – Specific targeted part within the selected device family.

Note that as 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 the Clarity Designer tool online help for further information.

To create a custom configuration:

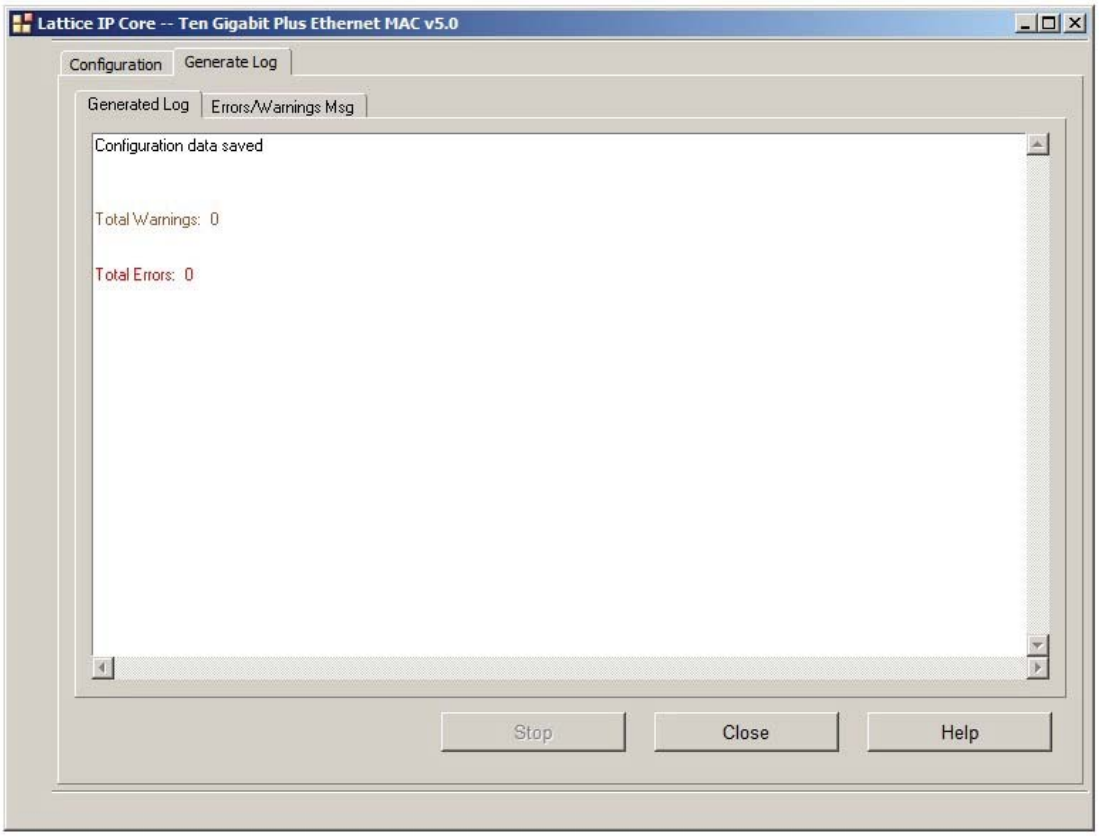
1. Click the **Customize** button in the Clarity Designer dialog box to display the 10 Gb Ethernet MAC IP core Configuration GUI as shown in Figure 4-2.

2. Select the IP parameter options specific to your application. Refer to Parameter Settings for more information on the 10 Gb Ethernet MAC IP core parameter settings.

3. After setting the parameters, click **Generate**.

4. A dialog box, shown in Figure 4-7, displays logs, errors and warnings. Click **Close**.

Figure 4-7. Clarity Designer Generate Log Tab

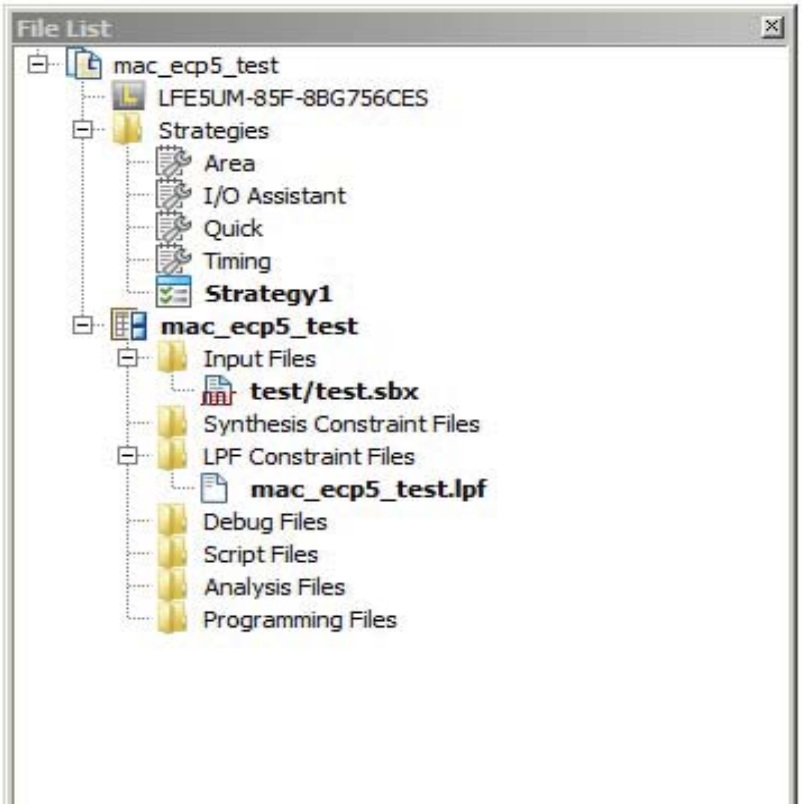


5. Click the **Generate** button in the Clarity Designer window to create the Clarity Designer (.sbx) file.

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 the current Diamond Project Inputs Files list as shown in Figure 4-8.

Figure 4-8. 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.

Clarity Designer-Created Files and Top Level Directory Structure

When the user clicks the Generate button in the Clarity Designer window, 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 4-9.

Figure 4-9. Clarity Designer 10Gb Ethernet MAC IP Core Directory Structure

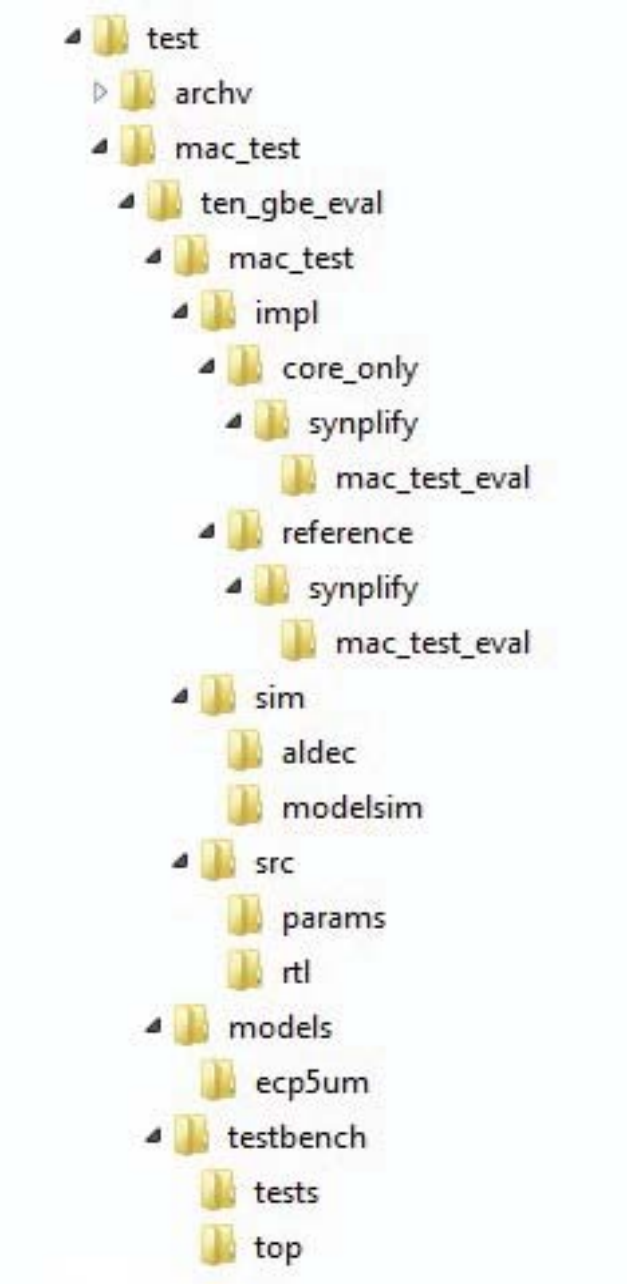


Table 4-2 provides a list of key files and directories created by the Clarity Designer tool and how they are used. The IPexpress tool creates several files that are used throughout the design cycle. The names of most of the created files are customized to the user’s module name specified in the Clarity Designer tool.

Table 4-2. File List

File	Description
<username>_inst.v	This file provides an instance template for the IP.
<username>.v	This file provides the 10 Gb Ethernet MAC core for simulation.
<username>_beh.v	This file provides a behavioral simulation model for the 10 Gb Ethernet MAC core.
<username>_params.v	This file provides parameters necessary for the simulation.
<username>_bb.v	This file provides the synthesis black box for the user's synthesis.
<username>.ngo	This file provides the synthesized IP core.
<username>.lpc	This file contains the Clarity Designer tool options used to recreate or modify the core in the Clarity Designer tool.
<username>.sbx	Clarity Designer project file. This is a container that 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 this file to the associated Diamond project.

These are all of the files necessary to implement and verify the 10 Gb Ethernet MAC IP core in your own top-level design. The following additional files providing IP core generation status information are also generated in the "Project Path" directory:

- <username>_generate.log – Clarity Synthesis and EDIF2NGD log file.
- <username>_gen.log – Clarity Designer IP generation log file.

The **\<ten_gbe_eval>** and subtending directories provide files supporting 10 Gb Ethernet MAC core evaluation. The **\<ten_gbe_eval>** directory shown in Figure 4-9 contains files/folders with content that is constant for all con- figurations of the 10 Gb Ethernet MAC. The *\<username>* subfolder (**\ten_gbe_core0** in this example) contains files/folders with content specific to the username configuration.

The **\ten_gbe_eval** directory is created by Clarity Designer the first time the core is generated and updated each time the core is regenerated. A *\<username>* directory is created by Clarity Designer 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. **\<ten_gbe_core0>**, **\<ten_gbe_core1>**, etc.

Instantiating the Core

The generated 10 Gb Ethernet MAC IP core package includes black-box (<*username*>_bb.v) and instance (<*username*>_inst.v) templates that can be used to instantiate the core in a top-level design. Two example RTL top-level reference source files are provided in *\<project_dir>**\<username>***ten_gbe_eval***\<username>***src\rtl**.

The top-level file <username>_eval_top.v is the same top-level that is used in the simulation model described in the next section. Designers may use this top-level reference as the starting template for the top-level for their complete design. Included in <username>_eval_top.v is logic, memory and clock modules supporting an XGMII interface loop back capability, a register module supporting programmable control of the 10 Gb Ethernet MAC core parameters and system processor interface. Verilog source RTL for these modules is provided in \<project_dir>\<username>\ten_gbe_eval\<username>. The top-level configuration is specified via the parameters defined in the ten_gbemac_defines.v file in *\<project_dir>**\<username>***ten_gbe_eval***\<username>***src\params**. A description of the 10 Gb Ethernet MAC parameters is in the parameters section of this document. A description of the 10 Gb Ethernet MAC register lay- out for this reference design is provided in an appendix to this document.

The top-level file <username>_core_only_top.v supports the ability to implement just the 10 Gb Ethernet MAC core itself. This design is intended only to provide an accurate indication of the device utilization associated with the 10 Gb Ethernet MAC core and should not be used as an actual implementation example.

Running Functional Simulation

The functional simulation includes a configuration-specific behavioral model of the 10 Gb Ethernet MAC IP Core (<username>_beh.v) that is instantiated in an FPGA top level along with a client-side interface loop back capability module and register implementation module.

The top-level file supporting ModelSim eval simulation is provided in **\<*project_dir*>\<*username*>\ten_gbe_eval\<*username*>\sim\modelsim**. This FPGA top is instantiated in an eval tes- tbench provided in **\<project_dir>\<*username*>\ten_gbe_eval\testbench\top** that configures FPGA test logic registers and 10 Gb Ethernet MAC IP core control and status registers via an included test file stimulus_file.v provided in **\<project_dir>\<*username*>\ten_gbe_eval\testbench\tests**. Note that the user can edit the stimulus_file.v file to configure and monitor whatever registers they desire.

To run the evaluation simulation using ModelSim:

1. Open ModelSim.
2. Under the File tab, select **Change Directory** and choose the folder **\<*project_dir*>\<*username*>\ten_gbe_eval\<*username*>\sim\modelsim**.
3. Under the Tools tab, select **Execute Macro** and execute one of the ModelSim “do” scripts shown. The top-level file supporting Aldec Active-HDL simulation is provided in **\<*project_dir*>\<*username*>\ten_gbe_eval\<*username*>\sim\aldec**. This FPGA top is instantiated in an eval testbench provided in **\<*project_dir*>\<*username*>\ten_gbe_eval\testbench** that configures FPGA test logic registers and 10 Gb Ethernet MAC IP core control and status registers via an included test file stimulus_file.v provided in **<*project_dir*>\<username>\ten_gbe_eval\testbench\tests**. Note that the user can edit the stimulus_file.v file to configure and monitor whatever registers they desire.

To run the evaluation simulation using Active-HDL:

1. Open Active-HDL.
2. Under the Console tab, change the directory to: **<*project_dir*>\<*username*>\ten_gbe_eval\<*username*>\sim\aldec**.
3. Execute the Active-HDL “do” scripts shown.

The simulation waveform results will be displayed in the Aldec Active-HDL Wave window.

Synthesizing and Implementing the Core in a Top-Level Design

The 10 Gb Ethernet MAC IP core itself is synthesized and is provided in NGO format when the core is generated. 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 LSE.

Two example RTL top-level configurations supporting 10 Gb Ethernet MAC core top-level synthesis and implementation are provided with the 10 Gb Ethernet MAC IP core in **\<*project_dir*>\<*username*>\ten_gbemac_eval\<*username*>\impl**.

The top-level file <username>_core_only_top.v provided in **\<*project_dir*>\<*username*>\ten_gbemac_eval\<*username*>\src\rtl** supports the ability to implement just the10 Gb Ethernet MAC core. This design is intended only to provide an accurate indication of the device utilization associated with the core itself and should not be used as an actual implementation example.

The top-level file <username>_eval_top.v provided in **\<*project_dir*>\<*username*>\ten_gbemac_eval\<*username*>\src\rtl** supports the ability to instantiate, simulate, map, place and route the Lattice 10 Gb Ethernet MAC IP core in a complete example design. This

reference design basically provides a loopback path for packets on the MAC Rx/Tx client interface, through a FIFO and associated logic. Ethernet packets are sourced to the Rx XGMII 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 XGMII have the correct source and destination addresses. This is the same configuration that is used in the evaluation simulation capability described previously. Note that implementation of the reference evaluation configu- ration is targeted to a specific device and package type for each device family.

Push-button implementation of both top-level configurations is supported via the Diamond project files, <username>_reference_eval.ldf and <username>_core_only_eval.ldf. These files are located in **\<*project_dir*>\<*username*>\ten_gbemac_eval\<*username*>\impl\<configuration>\synplify|lse**.

To use this project file in Diamond:

1. Choose **File > Open > Project**.
2. Browse to**\<*project_dir*>\<*username*>\ten_gbemac_eval\<*username*>\impl\<configuration>\synplify|lse** in the Open Project dialog box.
3. Select and open <*username*>.ldf. At this point, all of the files needed to support top-level synthesis and implementation will be imported to the project.
4. Select the Process tab in the left-hand GUI window.
5. Implement the complete design via the standard Diamond GUI 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 the IP instance. By recreating in 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

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

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. Use the .ipx if it is available.
5. Click **Open**.
6. Type in a name for the 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 the desired options.
9. Click **Configure**.

Hardware Evaluation

The 10 Gb Ethernet MAC 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.

This chapter gives application support information for the 10 Gb Ethernet MAC IP core.

Reference Register Descriptions

There are no registers in this IP core. All control and status information is passed between the core and the top level of the device through individual I/O ports on the core. Registers must be added to the top level to control and monitor these ports. This appendix describes all the registers needed to control and monitor the 10 Gb MAC IP core and the test logic at the top level. Unused bits of all the registers cannot be written. When read, the unused bits return a value of zero. The default values of the registers are restored when the core is reset.

All the registers except the MODE register should be written only after disabling the MAC. All of the registers can be read at any time.

Table 5-1. 10 Gb Ethernet MAC IP Core Internal Registers

Internal Registers			
Register Description	**Mnemonic**	**I/O Address**	**Reset Value**
Version Register	VERID	A00H	X1H
Mode Register	MODE	A01H	00H
Transmit Control Register	TX_CTL	A02H	00H
Receive Control Register	RX_CTL	A03H	00H
Maximum Packet Size Register	MAX_PKT_SIZE_0	A04H	05H
Maximum Packet Size Register	MAX_PKT_SIZE_1	A05H	EEH
Inter Packet Gap	IPG_VAL	A06H	01H
MAC Address 0	MAC_ADDR_0	A07H	00H
MAC Address 1	MAC_ADDR_1	A08H	00H
MAC Address 2	MAC_ADDR_2	A09H	00H
MAC Address 3	MAC_ADDR_3	A0AH	00H
MAC Address 4	MAC_ADDR_4	A0BH	00H
MAC Address 5	MAC_ADDR_5	A0CH	00H
Transmit Receive Status Register	TX_RX_STS	A0DH	00H
VLAN tag Length/Type Register	VLAN_TAG_0	A0EH	00H
VLAN tag Length/Type Register	VLAN_TAG_1	A0FH	00H
Multicast_table_0	MC_TAB_0	A10H	00H
Multicast_table_1	MC_TAB_1	A11H	00H
Multicast_table_2	MC_TAB_2	A12H	00H
Multicast_table_3	MC_TAB_3	A13H	00H
Multicast_table_4	MC_TAB_4	A14H	00H
Multicast_table_5	MC_TAB_5	A15H	00H
Multicast_table_6	MC_TAB_6	A16H	00H
Multicast_table_7	MC_TAB_7	A17H	00H
Pause Opcode	PAUS_OP_0	A18H	00H
Pause Opcode	PAUS_OP_1	A19H	01H
Test Control Register	TSTCNTL	B00H	02H
MAC Control Register	MACCNTL	B01H	00H
Pause Time Register	PAUSTMR_0	B02H	04H
Pause Time Register	PAUSTMR_1	B03H	FFH

Table 5-1. 10 Gb Ethernet MAC IP Core Internal Registers (Continued)

Internal Registers			
Register Description	**Mnemonic**	**I/O Address**	**Reset Value**
FIFO Almost Full Register	FIFOAFT_0	B04H	01H
FIFO Almost Full Register	FIFOAFT_1	B05H	00H
FIFO Almost Empty Register	FIFOAET_0	B06H	00H
FIFO Almost Empty Register	FIFOAET_1	B07H	03H
Statistic Counters			
TX Packet Length Statistics Counter	RX_STAT_PKT_LNGTH	800H	00H
TX Error Statistics Counter	TX_STAT_TX_ERR	808H	00H
TX Underrun Statistics Counter	TX_STAT_UNDER_RUN	810H	00H
TX CRC Error Statistics Counter	TX_STAT_CRC_ERR	818H	00H
TX Length Error Statistics Counter	TX_STAT_LNGTH_ERR	820H	00H
TX Long Packet Statistics Counter	TX_STAT_LNG_PKT	828H	00H
TX Multicast Packet Statistics Counter	TX_STAT_MULTCST	830H	00H
TX Broadcast Packet Statistics Counter	TX_STAT_BRDCST	838H	00H
Transmit Control Packet Statistics Counter	RX_STAT_CNT	840H	00H
TX Jumbo Packet Statistics Counter	TX_STAT_JMBO	848H	00H
TX Pause Packet Statistics Counter	TX_STAT_PAUSE	850H	00H
TX VLAN Tag Statistics Counter	TX_STAT_VLN_TG	858H	00H
TX Packet OK Statistics Counter	TX_STAT_PKT_OK	860H	00H
Transmit Packet 64 Statistics Counter	TX_STAT_PKT_64	868H	00H
Transmit Packet 65-127 Statistics Counter	TX_STAT_PKT_65_127	870H	00H
Transmit Packet 128-255 Statistics Counter	TX_STAT_PKT_128_255	878H	00H
Transmit Packet 256-511 Statistics Counter	TX_STAT_PKT_256_511	880H	00H
Transmit Packet 512-1023 Statistics Counter	TX_STAT_PKT_512_1023	888H	00H
Transmit Packet 1024-1518 Statistics Counter	TX_STAT_PKT_1024_1518	890H	00H
Transmit Packet 1518 Statistics Counter	TX_STAT_PKT_1518	898H	00H
Transmit Frame Error Statistics Counter	TX_STAT_FRM_ERR	8a0H	00H
Transmit Packet 1519-2047 Statistics Counter	TX_STAT_PKT_1519_2047	8a8H	00H
Transmit Packet 2048-4095 Statistics Counter	TX_STAT_PKT_2048_4095	8b0H	00H
Transmit Packet 4096-9216 Statistics Counter	TX_STAT_PKT_4096_9216	8b8H	00H
Transmit Packet 9217-16383 Statistics Counter	TX_STAT_PKT_9217_16383	8c0H	00H
Receive Packet Length Statistics Counter	RX_STAT_PKT_LNGTH	900H	00H
RX VLAN Tag Statistics Counter	RX_STAT_VLN_TG	908H	00H
RX Pause Packet Statistics Counter	RX_STAT_PAUSE	910H	00H
Receive Filtered Packet Statistics Counter	RX_STAT_FLT	918H	00H
RX Unsupported Opcode Statistics Counter	TX_STAT_UNSP_OPCODE	920H	00H
RX Broadcast Packet Statistics Counter	RX_STAT_BRDCST	928H	00H
RX Multicast Packet Statistics Counter	RX_STAT_MULTCST	930H	00H
RX Length Error Statistics Counter	RX_STAT_LNGTH_ERR	938H	00H
RX Long Packet Statistics Counter	RX_STAT_LNG_PKT	940H	00H
RX CRC Error Statistics Counter	RX_STAT_CRC_ERR	948H	00H
Receive Packet Discard Statistics Counter	RX_PKT_DISCARD_STAT	950H	00H
Receive Packet Ignored Statistics Counter	RX_PKT_IGNORE	958H	00H
Receive Packet Fragments Statistics Counter	RX_PKT_FRAGMENTS	960H	00H

Table 5-1. 10 Gb Ethernet MAC IP Core Internal Registers (Continued)

Internal Registers			
Register Description	**Mnemonic**	**I/O Address**	**Reset Value**
Receive Packet Jabbers Statistics Counter	RX_PKT_JABBERS	968H	00H
Receive Packet 64 Statistics Counter	RX_PKT_64	970H	00H
Receive Packet 65-127 Statistics Counter	RX_PKT_65_127	978H	00H
Receive Packet 128-255 Statistics Counter	RX_PKT_128_255	980H	00H
Receive Packet 256-511 Statistics Counter	RX_PKT_256_511	988H	00H
Receive Packet 512-1023 Statistics Counter	RX_PKT_512_1023	990H	00H
Receive Packet 1024-1518 Statistics Counter	RX_PKT_1024_1518	998H	00H
Receive Packet Undersize Statistics Counter	RX_PKT_UNDERSIZE	9a08H	00H
Receive Packet Unicast Statistics Counter	RX_PKT_UNICAST	9a8H	00H
Packets Received Statistics Counter	RX_PKT_RCVD	9b0H	00H
Receive Packet 64 With Good CRC Statistics Counter	RX_PKT_64_GOOD_CRC	9b8H	00H
Receive Packet 1518 With Good CRC Statistics Counter	RX_PKT_1518_GOOD_CRC	9c0H	00H
Receive Packet 1519-2047 Statistics Counter	RX_PKT_1519_2047	9c8H	00H
Receive Packet 2048-4095 Statistics Counter	RX_PKT_2048_4095	9d0H	00H
Receive Packet 4096-9216 Statistics Counter	RX_PKT_4096_9216	9d8H	00H
Receive Packet 9217-16383 Statistics Counter	RX_PKT_9217_16383	9e0H	00H

Version ID Register (VERID)

Identifies a specific implementation of the core and top level.

Table 5-2. Version ID Register Description

Name: VERID				Address: A00H
Bits	**Name**	**Type**	**Default**	**Description**
7:0	Version ID	RO	20H	**Version ID**. Eight-bit ID code.

MODE Register (MODE)

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

Additional details are provided in Table 3-1 on page 13.

Table 5-3. MODE Register Description

Name: Mode				Address: A01H
Bits	**Name**	**Type**	**Default**	**Description**
7:2	UNUSED	—	—	Reserved.
1	Tx_En	R/W	0	**Tx MAC Enable**. When set, the Tx MAC is enabled to receive frames.
0	Rx_En	R/W	0	**Rx MAC Enable**. When set, the Rx MAC is enabled to transmit frames.

Transmit Control (TX_CTL)

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

Table 5-4. Transmit Configuration Register Description

Name: TX_CTL				Address: A02H
Bits	**Name**	**Type**	**Default**	**Description**
7:4	UNUSED	—	—	**Reserved**
3	Transmit_short	R/W	0	Transmit Short. When high, enables the Tx MAC to transmit frames shorter than 64 bytes.
2	Tx_ifg_stretch	R/W	0	IFG Stretch Mode. When set, the Tx MAC operates in the IFG stretch mode, to match the data rates of OC-192. The IPG required to match OC192 is added to the value specified in IPG_VAL.
1	FC_en	R/W	0	Flow-control Enable. When set, this enables the flow control functionality of the Tx MAC. This bit should be set for the Tx MAC to transmit a PAUSE frame.
0	Tx_pass_fcs	R/W	0	Transmit Pass FCS. When set, the FCS field generation is disabled in the Tx MAC, and the user is responsible to generate the appropriate FCS field.

Receive Control (RX_CTL)

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

Table 5-5. Receive Control Register Description

Name: RX_CTL				Address: A03H
Bits	**Name**	**Type**	**Default**	**Description**
7	UNUSED			Reserved
6	drop_mac_ctrl	R/W	0	Drop MAC Control Frames. When set, all MAC control frames are not passed on to the client interface.
5	receive_short	R/W	0	Receive Short Frames. When high, enables the Rx MAC to receive frames shorter than 64 bytes.
4	receive_bc	R/W	0	Receive Broadcast. When high, enables the Rx MAC to receive broadcast frames
3	receive_all_mc	R/W	0	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.
2	rx_pause_en	R/W	0	Receive PAUSE. When set, the Rx MAC will indicate the PAUSE frame reception to the Tx MAC. PAUSE frames are received and transferred to the FIFO only when drop_mac_ctrl bit is NOT set.
1	rx_pass_fcs	R/W	0	Rx Pass FCS and Pad. When set, the FCS and any of the padding bytes are passed to the FIFO. When low, the MAC will strip off the FCS and any padding bytes before transferring it to the FIFO.
0	prms	R/W	0	Promiscuous Mode. When asserted, all filtering schemes are abandoned and the Rx MAC receives frames with any address.

Maximum Packet Size (MAX_PKT_LEN)

This register should 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. Writing this register while the MAC is active results in unpredictable behavior.

Table 5-6. Maximum Packet Size Register Description

Name: MAX_PKT_SIZE				Address: A04H
Bits	**Name**	**Type**	**Default**	**Description**
7:6	UNUSED	—	—	Reserved
5:0	Max_frame_0	R/W	05H	Maximum Frame Length. Used only for statistical purposes, all frames longer than the value given here will be marked as long. This value will in not manner affect the frame's reception. Upper byte.

Name: MAX_PKT_SIZE				Address: A05H
Bits	**Name**	**Type**	**Default**	**Description**
7:0	Max_frame_1	R/W	EEH	Maximum Frame Length. Used only for statistical purposes, all frames longer than the value given here will be marked as long. This value will in not manner affect the frame's reception. Lower byte.

IPG Value (IPG_VAL)

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

Table 5-7. IPG Value Register Description

Name: IPG_VAL				Address: A06H
Bits	**Name**	**Type**	**Default**	**Description**
7:5	UNUSED	—	—	Reserved
4:0	IPG_Value	R/W	01H	Transmit Inter Packet Gap. Specifies the amount of inter-frame gap in increments of 4 bytes. Value of 0 indicates the minimum value of 8 bytes.

MAC Address Register {0,1,2,3,4,5}, Set of six (MAC_ADDR_[5-0])

Note that the MAC address is stored in the registers in Hexadecimal form, so for example to set the MAC Address to: AC-DE-48-00-00-80 would require writing 0xAC (octet 0) to address 0x07, 0xDE (octet 1) to address 0x08, 0x48 (octet 2) to address 0x09, 0x00 (octet 3) to address 0x0A, 0x00 (octet 4) to address 0x0B, and 0x80 (octet 5) to address 0x0C.

Table 5-8. MAC Address Register Description

Name: MAC_ADDR				Address: A07H, A08H, A09H, A0AH, A0BH, A0CH
Bits	**Name**	**Type**	**Default**	**Description**
7:0	Mac_Addr[0-5]	R/W	0	MAC Address. Ethernet address assigned to the port supported by the MAC.

Transmit and Receive Status (TX_RX_STS)

Table 5-9. Transmit and Receive Register Description

Name: TX_RX_STS				Address: A0DH
Bits	**Name**	**Type**	**Default**	**Description**
7:5	UNUSED			Reserved
4	Link_ok	RO	0	Link OK. When set, indicates that no fault symbols were received on the link.
3	Remote_fault	RO	0	Remote fault. When set, indicates that remote fault symbols were received on the link.
2	Local_fault	RO	0	Local fault. When set, indicates that local fault symbols were received on the link.
1	Rx_idle	RO	0	Receive MAC Idle. When set, indicates that the RX MAC is inactive.
0	Tx_idle	RO	0	Transmit MAC Idle. When set, indicates that the TX MAC is inactive.

VLAN Tag(VLAN_TAG_[0-1])

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

Table 5-10. VLAN Tag Register Description

Name: VLAN_TAG_0				Address: A0EH
Bits	**Name**	**Type**	**Default**	**Description**
7:0	VLAN_TAG_0	RO	0	VLAN Tag ID. Upper byte

Name: VLAN_TAG_1				Address: A0FH
Bits	**Name**	**Type**	**Default**	**Description**
7:0	VLAN_TAG_1	RO	0	VLAN Tag ID. Lower byte.

Multicast Tables, set of eight (MLT_TAB_[0-7])

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.

Table 5-11. Multicast Table Register Description

Name: MLT_TAB_[0-7]				Address: A10H, A11H, A12H, A13H, A14H, A15H, A16H, A17H
Bits	**Name**	**Type**	**Default**	**Description**
7:0	Multicast Table [0-7]	R/W	0	Multicast Table. Eight tables that make a 64-bit hash.

Pause Opcode (PAUS_OP_[0-1])
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 MAC.

Table 5-12. Pause Opcode Register Description

Name: PAUS_OP_[0-1]				Address: A18H, A19H
Bits	**Name**	**Type**	**Default**	**Description**
7:0	Pause_OpCode_0	R/W	0	PAUSE Opcode. Upper byte
7:0	Pause_OpCode_1	R/W	1	PAUSE Opcode. Lower byte

Test Control Register (TSTCNTL)
Test control bits.

Table 5-13. Test Control Register Description

Name: TSTCNTL				Address: B00H
Bits	**Name**	**Type**	**Default**	**Description**
7:2	UNUSED			Reserved
1	Loop back Enable	R/W	1	1 = loop back, 0 = no loop back This bit is used to enable the loop back of packets on the client side of the 10 Gb MAC core.
0	Address Swap	R/W	0	Unused

MAC Control Register (MACCNTL)
MAC control bits.

Table 5-14. MAC Control Register Description

Name: MACCNTL				Address: B01H
Bits	**Name**	**Type**	**Default**	**Description**
7:5	UNUSED			Reserved
4	Ignore packet	R/W	0	This bit controls the ignore_pkt pin on the 10 Gb MAC core.
3	Tx fifo empty	R/W	0	This bit gets ORed with the tx_fifo empty signal. The ORed output controls the sine_tx_aem pin on the 10 Gb MAC core.
2	UNUSED	R/W	0	Reserved
1	UNUSED	R/W	0	Reserved
0	Tx Send Pause Request	R/W	0	1 = send request, 0 = don't send request This bit gets ORed with the tx_fifo almost full signal. The ORed output controls the tx_pausreq pin on the 10 Gb MAC core.

PAUSE Time Register (PAUSTMR)
This register has the pause time for a flow control packet sourced by the 10 Gb MAC transmitter.

Table 5-15. PAUSE Opcode Register Description

Name: PAUSTMR				Address: B02H, B03H
Bits	**Name**	**Type**	**Default**	**Description**
15:0	Pause_Time	R/W	04FFH	These bits control the tx_paustim[15:0] pins on the 10 Gb MAC core.

FIFO Almost Full Register (FIFOAFT)

This register contains the almost full threshold for the loop back FIFO.

Table 5-16. FIFO Almost Full Register Description

Name: FIFOAFT				Address: B04H, B05H
Bits	**Name**	**Type**	**Default**	**Description**
15:9	UNUSED			Reserved
8:0	FIFO Almost Full	R/W	0100H	These bits control the loop back FIFO almost full threshold.

FIFO Almost Empty Register (FIFOAET)

This register contains the almost empty threshold for the loop back FIFO.

Table 5-17. FIFO Almost Empty Register Description

Name: FIFOAET				Address: B06H, B07H
Bits	**Name**	**Type**	**Default**	**Description**
15:9	UNUSED			Reserved
8:0	FIFO Almost Empty	R/W	0003H	These bits control the loop back FIFO almost empty threshold.

Transmit Packet Length Statistics Counter (TX_STAT_PKT_LNGTH)

A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-18. Transmit Packet Length Statistics Counter Description

Name: TX_STAT_PKT_LNGTH				Address: 800H - 807H
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet Length	RO	—	Indicates the total number of octet transmitted in a particular frame. tx_statvec[13:0] is used to implement this counter.

Transmit TX Error Statistics Counter (TX_STAT_TX_ERR)

A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-19. Transmit TX Error Statistics Counter Description

Name: RX_STAT_TX_ERR				Address: 808H - 80FH
Bits	**Name**	**Type**	**Default**	**Description**
63:0	TX_Error	RO	—	Counts the total number of PHY terminated packet. tx_statvec[24] is used to implement this counter.

Transmit Underrun Error Statistics Counter (TX_STAT_UNDER_RUN)

A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-20. Transmit Underrun Error Statistics Counter Description

Name: RX_STAT_UNDER_RUN				Address: 810H - 817H
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Underrun	RO	—	Counts the total number of underrun packets transmitted. tx_statvec[21] is used to implement this counter.

Transmit CRC Error Statistics Counter (TX_STAT_CRC_ERR)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-21. Transmit CRC Error Statistics Counter Description

Name: TX_STAT_CRC_ERR				Address: 818H - 81FH
Bits	**Name**	**Type**	**Default**	**Description**
63:0	CRC_Error	RO	—	Counts the total number of packets transmitted with crc error. tx_statvec[22] is used to implement this counter.

Transmit Length Error Statistics Counter (TX_STAT_LNGTH_ERR)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-22. Transmit Length Error Length Statistics Counter Description

Name: TX_STAT_LNGTH_ERR				Address: 820H - 827H
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Length Error	RO	—	Counts the total number of packets transmitted with length of the packet and length in the Length/Type field mismatch. tx_statvec[23] is used to implement this counter.

Transmit Long Packet Statistics Counter (TX_STAT_LNG_PKT)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-23. Transmit Long Packet Statistics Counter Description

Name: TX_STAT_LNG_PKT				Address: 828H - 82FH
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Long Packet	RO	—	Counts the total number of packets transmitted with length of the packet longer than the max_frm_size. tx_statvec[25] is used to implement this counter.

Transmit Multicast Packet Statistics Counter (TX_STAT_MLTCST)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-24. Transmit Multicast Packet Statistics Counter Description

Name: TX_STAT_MLTCST				Address: 830H - 837H
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Multicast Packet	RO	—	Counts the total number of multicast packets transmitted. tx_statvec[20] is used to implement this counter.

Transmit Broadcast Packet Statistics Counter (TX_STAT_BRDCST)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-25. Transmit Broadcast Packet Statistics Counter Description

Name: TX_STAT_BRDCST				Address: 838H - 83FH
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Broadcast Packet	RO	—	Counts the total number of broadcast packets transmitted. tx_statvec[19] is used to implement this counter.

Transmit Control Packet Statistics Counter (RX_STAT_CNT)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-26. Transmit Control Packet Statistics Counter Description

Name: RX_STAT_CNT				Address: 840H - 847H
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Control Packet	RO	—	Counts the total number of control packets transmitted. tx_statvec[16] is used to implement this counter.

Transmit Jumbo Packet Statistics Counter (RX_STAT_JMBO)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-27. Transmit Jumbo Packet Statistics Counter Description

RX_STAT_JMBO				Address: 848H - 84FH
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Jumbo Packet	RO	—	

Transmit Pause Packet Statistics Counter (TX_STAT_PAUSE)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-28. Transmit Pause Packet Statistics Counter Description

Name: TX_STAT_PAUSE				Address: 850H - 857H
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Pause Pkt	RO	—	Counts the total number of pause packets transmitted. tx_statvec[15] is used to implement this counter.

Transmit VLAN Tag Statistics Counter (TX_STAT_VLN_TG)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-29. Transmit VLAN Tag Statistics Counter Description

Name: TX_STAT_VLN_TG				Address: 858H - 85FH
Bits	**Name**	**Type**	**Default**	**Description**
63:0	VLAN Tag Pkt	RO	—	Counts the total number of tagged packets transmitted. tx_statvec[18] is used to implement this counter.

Transmit Packet OK Statistics Counter (TX_STAT_PKT_OK)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-30. Transmit Packet OK Statistics Counter Description

Name: TX_STAT_PKT_OK				Address: 860H - 867H
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet OK	RO	—	Counts the total number of packets transmitted without any errors. tx_statvec[14] is used to implement this counter.

Transmit Packet 64 Statistics Counter (TX_STAT_PKT_64)

A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-31. Transmit Packet 64 Statistics Counter Description

Name: TX_STAT_PKT_64				Address: 868H - 86FH
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet 64	RO	—	Counts the total number of packets transmitted with length equal to 64. tx_statvec[13:0] is used to implement this counter.

Transmit Packet 65 - 127 Statistics Counter (TX_STAT_PKT_65_127)

A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-32. Transmit Packet 65-127 Statistics Counter Description

Name: TX_STAT_PKT_65_127				Address: 870H - 877H
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet 65-127	RO	—	Counts the total number of packets transmitted with length between 65 and 127. tx_statvec[13:0] is used to implement this counter.

Transmit Packet 128-255 Statistics Counter (TX_STAT_PKT_128_255)

A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-33. Transmit Packet 128-255 Statistics Counter Description

Name: TX_STAT_PKT_128_255				Address: 878H - 87FH
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet 128-255	RO	—	Counts the total number of packets transmitted with length between 128 and 255. tx_statvec[13:7] is used to implement this counter.

Transmit Packet 256-511 Statistics Counter (TX_STAT_PKT_256_511)

A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-34. Transmit Packet 256-511 Statistics Counter Description

Name: TX_STAT_PKT_256_511				Address: 880H - 887H
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet 256-511	RO	—	Counts the total number of packets transmitted with length between 256 and 511. tx_statvec[13:8] is used to implement this counter.

Transmit Packet 512-1023 Statistics Counter (TX_STAT_PKT_512_1023)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-35. Transmit Packet 512-1023 Statistics Counter Description

Name: TX_STAT_PKT_512_1023				Address: 888H - 88fH
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet 512-1023	RO	—	Counts the total number of packets transmitted with length between 512 and 1023. tx_statvec[13:9] is used to implement this counter.

Transmit Packet 1024-1518 Statistics Counter (TX_STAT_PKT_1024_1518)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-36. Transmit Packet 1024-1518 Statistics Counter Description

Name: TX_STAT_PKT_1024_1518				Address: 890H - 897H
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet 1024-1518	RO	—	Counts the total number of packets transmitted with length between 1024 and 1518. tx_statvec[13:0] is used to implement this counter.

Transmit Packet 1518 Statistics Counter (TX_STAT_PKT_1518)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-37. Transmit Packet 1518 Statistics Counter Description

Name: TX_STAT_PKT_1518				Address: 898H - 89fH
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet 1518	RO	—	Counts the total number of packets transmitted with length greater than 1518. tx_statvec[13:0] is used to implement this counter.

Transmit Frame Error Statistics Counter (TX_STAT_FRM_ERR)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-38. Transmit Frame Error Statistics Counter Description

Name: TX_STAT_FRM_ERR				Address: 8a0H - 8a7H
Bits	**Name**	**Type**	**Default**	**Description**
63:0	TX Frame Error	RO	—	Counts the total number of packets transmitted with error. tx_statvec[14] is used to implement this counter.

Transmit Packet 1519-2047 Statistics Counter (TX_STAT_PKT_1519_2047)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-39. Transmit Packet 1519-2047 Statistics Counter Description

Name: TX_STAT_PKT_1519_2047				Address: 8a8H - 8afH
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet 1519-2047	RO	—	Counts the total number of packets transmitted with length between 1024 and 2047. tx_statvec[13:0] is used to implement this counter.

Transmit Packet 2048-4095 Statistics Counter (TX_STAT_PKT_2048_4095)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-40. Transmit Packet 2048-4095 Statistics Counter Description

Name: TX_STAT_PKT_2048_4095				Address: 8b0H - 8b7H
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet 2048-4095	RO	—	Counts the total number of packets transmitted with length between 2048 and 4095. tx_statvec[13:11] is used to implement this counter.

Transmit Packet 4096-9216 Statistics Counter (TX_STAT_PKT_4096_9216)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-41. Transmit Packet 4096-9216 Statistics Counter Description

Name: TX_STAT_PKT_4096_9216				Address: 8b8H - 8bfH
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet 4096-9216	RO	—	Counts the total number of packets transmitted with length between 4096 and 9216. tx_statvec[13:0] is used to implement this counter.

Transmit Packet 9217-16383 Statistics Counter (TX_STAT_PKT_9217_16383)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-42. Transmit Packet 9217-16383 Statistics Counter Description

Name: TX_STAT_PKT_9217_16383				Address: 8c0H - 8c7H
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet 9217-16383	RO	—	Counts the total number of packets transmitted with length between 9217 and 16383. tx_statvec[13:0] is used to implement this counter.

Receive Packet Length Statistics Counter (RX_STAT_PKT_LNGTH)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-43. Receive Packet Length Statistics Counter Description

Name: RX_STAT_PKT_LNGTH				Address: 900H - 907H
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet Length	RO	—	Indicated the length of the packet received. rx_statvec[13:0] is used to implement this counter.

Receive VLAN Tag Statistics Counter (RX_STAT_VLN_TG)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-44. Receive VLAN Tag Statistics Counter Description

Name: RX_STAT_VLN_TG				Address: 908H - 90FH
Bits	**Name**	**Type**	**Default**	**Description**
63:0	VLAN Tag Pkt	RO	—	Counts the total number of tagged packets received. rx_statvec[18] is used to implement this counter.

Receive Pause Packet Statistics Counter (RX_STAT_PAUSE)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-45. Receive Pause Packet Statistics Counter Description

Name: RX_STAT_PAUSE				Address: 910H - 917H
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Pause Pkt	RO	—	Counts the total number of pause packets received. rx_statvec[19] is used to implement this counter.

Receive Filtered Packet Statistics Counter (RX_STAT_FLT)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-46. Receive Filtered Packet Statistics Counter Description

Name: RX_STAT_FLT				Address: 918H - 91FH
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Filtered Pkt	RO	—	rx_statvec[22] is used to implement this counter.

Receive Unsupported Opcode Statistics Counter (RX_STAT_UNSP_OPCODE)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-47. Receive Unsupported Opcode Statistics Counter Description

Name: RX_STAT_UNSP_OPCODE				Address: 920H - 927H
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Unsupported Opcode	RO	—	Counts the number of packets received with unsupported Opcode. rx_statvec[23] is used to implement this counter.

Receive Broadcast Packet Statistics Counter (RX_STAT_BRDCST)

A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-48. Receive Broadcast Packet Statistics Counter Description

Name: RX_STAT_BRDCST				Address: 928H - 92FH
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Broadcast Packet	RO	—	Counts the number of packets received that were directed to the broadcast address. This does not include multicast packets. rx_statvec[15] is used to implement this counter.

Receive Multicast Packet Statistics Counter (RX_STAT_MLTCST)

A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-49. Receive Multicast Packet Statistics Counter Description

Name: RX_STAT_MLTCST				Address: 930H - 937H
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Multicast Packet	RO	—	Counts the number of packets received that were directed to the multicast address. This does not include broadcast packets. rx_statvec[16] is used to implement this counter.

Receive Length Error Statistics Counter (RX_STAT_LNGTH_ERR)

A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-50. Receive Length Error Length Statistics Counter Description

Name: RX_STAT_LNGTH_ERR				Address: 938H - 93FH
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Length Error	RO	—	Counts the total number of packets received with length of the packet and length in the Length/Type field mismatch. rx_statvec[20] is used to implement this counter.

Receive Long Packet Statistics Counter (RX_STAT_LNG_PKT)

A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-51. Receive Long Packet Statistics Counter Description

Name: RX_STAT_LNG_PKT				Address: 958H - 95FH
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Long Packet	RO	—	Counts the number of packets received longer than the max_pkt_size. rx_statvec[21] is used to implement this counter.

Receive CRC Error Statistics Counter (RX_STAT_CRC_ERR)

A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-52. Receive CRC Error Statistics Counter Description

Name: RX_STAT_CRC_ERR				Address: 960H - 967H
Bits	**Name**	**Type**	**Default**	**Description**
63:0	CRC_Error	RO	—	Counts the number of packets received with crc error. rx_statvec[17] is used to implement this counter.

Receive Packet Discard Statistics Counter (RX_PKT_DISCARD_STAT)

A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-53. Receive Packet Discard Statistics Counter Description

Name: RX_PKT_DISCARD_STAT				Address: 968H - 96FH
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet Discard	RO	—	Counts the number of packets discarded at the receive end. rx_statvec[14] is used to implement this counter.

Receive Packet Ignored Statistics Counter (RX_PKT_IGNORE)

A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-54. Receive Packet Ignored Statistics Counter Description

Name: RX_PKT_IGNORE				Address: 970H - 977H
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet Ignore	RO	—	Counts the number of packets ignored when the user requests using the ignore_pkt signal. rx_statvec[25] is used to implement this counter.

Receive Packet Fragments Statistics Counter (RX_PKT_FRAGMENTS)

A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-55. Receive Packet Fragments Statistics Counter Description

Name: RX_PKT_FRAGMENTS				Address: 978H - 97fH
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet Fragments	RO	—	Counts the number of packets received with less than 64 octets in length and has either a FCS error or an Alignment error. rx_statvec[13:6] along with rx_statvec[17] are used to implement this counter.

Receive Packet Jabbers Statistics Counter (RX_PKT_JABBERS)

A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-56. Receive Packet Jabbers Statistics Counter Description

Name: RX_PKT_JABBERS				Address: 980H - 987H
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet Jabbers	RO	—	Counts the number of packets received with length longer than 1518 octets and has either a FCS error or an Alignment error. rx_statvec[21] along with rx_statvec[17] are used to implement this counter.

Receive Packet 64 Statistics Counter (RX_PKT_64)

A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-57. Receive Packet 64 Statistics Counter Description

Name: RX_PKT_64				Address: 988H - 98fH
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet 64	RO	—	Counts the number of packets received that were 64 octets in length (including bad packets). rx_statvec[13:0] is used to implement this counter.

Receive Packet 65-127 Statistics Counter (RX_PKT_65_127)

A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-58. Receive Packet 65-127 Statistics Counter Description

Name: RX_PKT_65_127				Address: 990H - 997H
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet 65-127	RO	—	Counts the number of packets received that were between 65-127 octets in length (including bad packets). rx_statvec[13:0] is used to implement this counter.

Receive Packet 128-255 Statistics Counter (RX_PKT_128_255)

A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-59. Receive Packet 128-255 Statistics Counter Description

Name: RX_PKT_128_255				Address: 998H - 99fH
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet 128-255	RO	—	Counts the number of packets received that were between 128-255 octets in length (including bad packets). rx_statvec[13:7] is used to implement this counter.

Receive Packet 256-511 Statistics Counter (RX_PKT_256_511)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-60. Receive Packet 256-511 Statistics Counter Description

Name: RX_PKT_256_511				Address: 9a0H - 9a7H
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet 256-511	RO	—	Counts the number of packets received that were between 256-511 octets in length (including bad packets). rx_statvec[13:8] is used to implement this counter.

Receive Packet 512-1023 Statistics Counter (RX_PKT_512_1023)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-61. Receive Packet 512-1023 Statistics Counter Description

Name: RX_PKT_512_1023				Address: 9a8H - 9afH
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet 512-1023	RO	—	Counts the number of packets received that were between 512-1023 octets in length (including bad packets). rx_statvec[13:9] is used to implement this counter.

Receive Packet 1024-1518 Statistics Counter (RX_PKT_1024_1518)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-62. Receive Packet 1024-1518 Statistics Counter Description

Name: RX_PKT_1024_1518				Address: 9b0H - 9b7H
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet 1024-1518	RO	—	Counts the number of packets received that were between 1024-1518 octets in length (including bad packets). rx_statvec[13:0] is used to implement this counter.

Receive Packet Undersize Statistics Counter (RX_PKT_UNDERSIZE)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-63. Receive Packet Undersize Statistics Counter Description

Name: RX_PKT_UNDERSIZE				Address: 9b8H - 9bfH
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet Undersize	RO	—	Counts the number of packets received that were less than 64 octets long and were otherwise well formed. rx_statvec[13:6] is used to implement this counter.

Receive Packet Unicast Statistics Counter (RX_PKT_UNICAST)

A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-64. Receive Packet Unicast Statistics Counter Description

Name: RX_PKT_UNICAST				Address: 9c0H - 9c7H
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet Unicast	RO	—	Counts the number of good packets received that were directed to a single address. rx_statvec[15] and rx_statvec[16] are used to implement this counter.

Packets Received Statistics Counter (RX_PKT_RCVD)

A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-65. Packets Received Statistics Counter Description

Name: RX_PKT_RCVD				Address: 9c8H - 9cfH
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packets Received	RO	—	Counts the number of packets received (including bad packet, broadcast and multicast packets). rx_statvec[15] and rx_statvec[16] are used to implement this counter.

Receive Packet 64 With Good CRC Statistics Counter (RX_PKT_64_GOOD_CRC)

A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-66. Receive Packet 64 With Good CRC Statistics Counter Description

Name: RX_PKT_64_GOOD_CRC				Address: 9d0H - 9d7H
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet 64 with Good CRC	RO	—	Counts the number of packets received with length less than 64 and with a good crc. rx_statvec[13:6] and rx_statvec[17] are used to implement this counter.

Receive Packet 1518 With Good CRC Statistics Counter (RX_PKT_1518_GOOD_CRC)

A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-67. Receive Packet 1518 With Good CRC Statistics Counter Description

Name: RX_PKT_1518_GOOD_CRC				Address: 9d8H - 9dfH
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet 1518 with Good CRC	RO	—	Counts the number of packets received with length more than 1518 and with a good crc. rx_statvec[21] and rx_statvec[17] are used to implement this counter.

Receive Packet 1519-2047 Statistics Counter (RX_PKT_1519_2047)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-68. Receive Packet 1519-2047 Statistics Counter Description

Name: RX_PKT_1519_2047				Address: 9e0H - 9e7H
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet 1519-2047	RO	—	Counts the number of packets received that were between 1519-2047 octets in length (including bad packets). rx_statvec[13:0] is used to implement this counter.

Receive Packet 2048-4095 Statistics Counter (RX_PKT_2048_4095)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-69. Receive Packet 2048-4095 Statistics Counter Description

Name: RX_PKT_2048_4095				Address: 9e8H - 9efH
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet 2048-4095	RO	—	Counts the number of packets received that were between 2048-4095 octets in length (including bad packets). rx_statvec[13:11] is used to implement this counter.

Receive Packet 4096-9216 Statistics Counter (RX_PKT_4096_9216)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-70. Receive Packet 4096-9216 Statistics Counter Description

Name: RX_PKT_4096_9216				Address: 9f0H - 9f7H
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet 4096-9216	RO	—	Counts the number of packets received that were between 4096-9216 octets in length (including bad packets). rx_statvec[13:0] is used to implement this counter.

Receive Packet 9217-16383 Statistics Counter (RX_PKT_9217_16383)
A read of the low byte of this counter returns that byte and latches the upper bytes for later reading. A read of any of the other bytes returns the previously latched value for that byte. The low address addresses the low byte.

Table 5-71. Receive Packet 9217-16383 Statistics Counter Description

Name: RX_PKT_9217_16383				Address: 9f8H - 9ffH
Bits	**Name**	**Type**	**Default**	**Description**
63:0	Packet 9217-16383	RO	—	Counts the number of packets received that were between 9217-16383 octets in length (including bad packets). rx_statvec[13:0] is used to implement this counter.

Support Resources

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

Lattice Technical Support

There are a number of ways to receive technical support.

E-mail Support

techsupport@latticesemi.com

Local Support

Contact your nearest Lattice sales office.

Internet

www.latticesemi.com

References

DS1021, LatticeECP3 Family Data Sheet

DS1044, ECP5 Family Data Sheet

Revision History

Date	Document Version	IP Version	Change Summary
December 2014	1.0	5.0	Initial release.

Resource Utilization

This appendix gives resource utilization information for Lattice FPGAs using the 10 Gb Ethernet MAC IP core. IPexpress/Clarity Designer is the Lattice IP configuration utility, and is included as a standard feature of the Diamond design tools. Details regarding the usage of IPexpress/Clarity Designer can be found in the IPexpress/Clarity Designer and Diamond help system. For more information on the Diamond design tools, visit the Lattice web site at www.latticesemi.com/software.

LatticeECP3 FPGAs

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

Mode	SLICEs	LUTs	Registers	sysMEM™ EBRs	f_{MAX} (MHz)
Multicast Address Filtering	3190	4681	2814	4	156.25

1. Performance and utilization data are generated using an LFE3-150EA-8FN1156C device with Lattice's Diamond 3.4. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeECP3 family.
2. The 10 Gb Ethernet MAC core itself does not use any external pins. However, in an application the core is used together IODDR and I/O Buffers integrated in the LatticeECP3 series FPGA. Thus the application implementing the 10 Gb Ethernet MAC will utilize I/O pins.

Ordering Part Number

Contact Lattice sales office for ordering 10 Gb Ethernet MAC IP core targeting LatticeECP3 devices.

ECP5 FPGAs

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

Mode	SLICEs	LUTs	Registers	sysMEM™ EBRs	f_{MAX} (MHz)
Multicast Address Filtering	2796	4085	2696	4	156.25

1. Performance and utilization data are generated using an LFE5UM-85F-8FBG756CES device with Lattice's Diamond 3.4. Performance may vary when using a different software version or targeting a different device density or speed grade within the ECP5 family.
2. The 10 Gb Ethernet MAC core itself does not use any external pins. However, in an application the core is used together IODDR and I/O Buffers integrated in the ECP5 series FPGA. Thus the application implementing the 10 Gb Ethernet MAC will utilize I/O pins.

Ordering Part Number

Contact Lattice sales office for ordering 10 Gb Ethernet MAC IP core targeting ECP5 devices.