Introduction

The LogiCORE™ IP AMBA® AXI Ethernet Lite MAC (Media Access Controller) is designed to incorporate the applicable features described in the IEEE Std. 802.3 Media Independent Interface (MII) specification, which should be used as the definitive specification.

The Ethernet Lite MAC supports the IEEE Std. 802.3 Media Independent Interface (MII) to industry standard Physical Layer (PHY) devices and communicates with a processor using the AXI4 or AXI4-Lite interface. The design provides a 10 Mb/s and 100 Mb/s (also known as Fast Ethernet) interface. The goal is to provide the minimal functions necessary to provide an Ethernet interface with the least resources used.

Features

- Parameterized AXI4 slave interface based on the AXI4 or AXI4-Lite specification
- Memory mapped direct I/O interface to the transmit and receive data dual port memory
- Media Independent Interface (MII) for connection to external 10/100 Mb/s PHY transceivers
- Independent internal 2K byte TX and RX dual port memory for holding data for one packet
- Optional dual buffer memories, 4K byte ping-pong, for TX and RX
- Receive and Transmit Interrupts
- Optional MDIO interface for PHY access
- Internal loopback support

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family [(1)]	Virtex-7, Kintex™-7, Artix ™-7, Zynq™-7000, Virtex[(2)]-6, Spartan[(3)]-6
Supported User Interfaces	AXI4/AXI4-Lite
Configuration	See Table 17 and Table 18
Provided with Core	
Documentation	Product Specification
Design Files	VHDL
Example Design	Not Provided
Test Bench	Not Provided
Constraints File	Not Provided
Simulation Model	Not Provided
Tested Design Tools	
Design Entry Tools	XPS 13.3
Simulation[(4)]	Mentor Graphics ModelSim
Synthesis Tools	XST 13.3
Support	
Provided by Xilinx, Inc.	

Notes:

1. For a complete listing of supported devices, see the release notes for this core.
2. For more information, see [Ref 1].
3. For more information, see[Ref 2].
4. For a listing of the supported tool versions, see the ISE Design Suite 13: Release Note Guide.

AXI4 Interface Support

The Ethernet Lite MAC core is compliant to the AMBA AXI4 interface specifications listed in the Reference Documents section. The Ethernet Lite MAC core includes the following features and exceptions when the AXI4 interface is selected.

Features

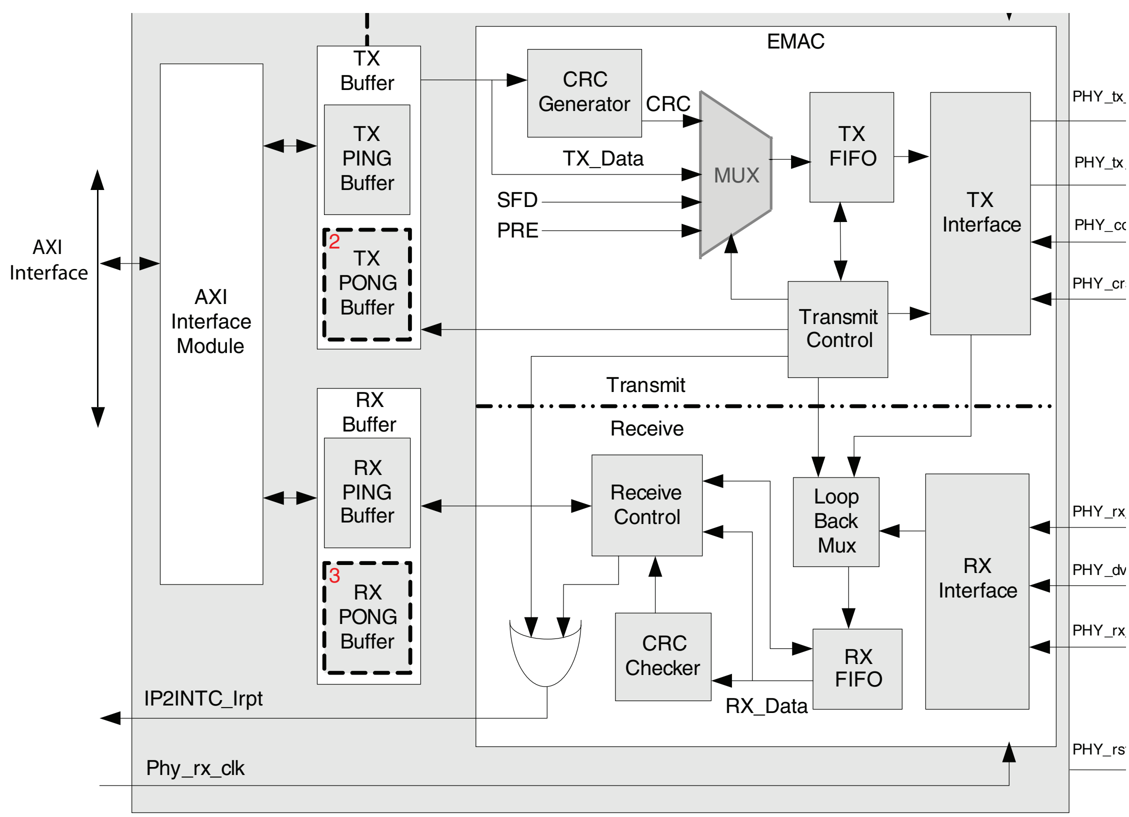
- Supports 32-bit data width
- Supports burst size of 4 bytes (word transfers)
- Supports INCR burst length of 1-256 beats

AXI4-Lite Interface Support

For systems where burst is not supported by the AXI4 master, this core can be configured for an AXI4-Lite interface. This configuration reduces the FPGA resource utilization. The Ethernet Lite MAC supports all requests from an AXI4 master as per the AXI4-Lite specification. The AXI4-Lite interface is selected by configuring the parameter C_S_AXI_PROTOCOL as "AXI4LITE".

Functional Description

The top level block diagram of the Ethernet Lite MAC is shown in Figure 1.



Notes : -
1. MDIO master is included in the design if the parameter C_INCLUDE_MDIO = 1
2. TX PONG Buffer is included in the design if the parameter C_TX_PING_PONG = 1
3. RX PONG Buffer is included in the design if the parameter C_RX_PING_PONG = 1

DS787_01

Figure 1: **Block Diagram of Ethernet Lite MAC**

AXI4 Interface Module

This module provides the interface to the AXI4 and implements AXI4 protocol logic. The AXI4 interface module is a bidirectional interface between a Ethernet Lite MAC core and the AXI4/AXI4-Lite interface standard.

TX Buffer

The TX Buffer module consists of 2K byte dual port memory to hold transmit data for one complete frame and the transmit interface control registers. It also includes optional 2K byte dual port memory for the pong buffer based on the parameter C_TX_PING_PONG.

RX Buffer

The RX Buffer module consists of 2K byte dual port memory to hold receive data for one complete frame and the receive interface control register. It also includes optional 2K byte dual port memory for the pong buffer based on the parameter C_RX_PING_PONG.

Transmit

This module consists of transmit logic, CRC generator module, transmit data mux, TX FIFO and the transmit interface module. The CRC generator module calculates the CRC for the frame to be transmitted. The transmit control mux arranges this frame and sends the preamble, SFD, frame data, padding and CRC to the transmit FIFO in the required order. When the frame is transmitted to the PHY, this module generates a transmit interrupt and updates the transmit control register.

Receive

This module consists of the RX interface, loopback control mux, RX FIFO, CRC checker and Receive Control module. Receive data signals from the PHY are passed through the loopback control mux and stored in the RX FIFO. If loopback is enabled, data on the TX lines is passed to the RX FIFO. The CRC checker module calculates the CRC of the received frame and if the correct CRC is found, receive control logic generates the frame receive interrupt.

MDIO Master Interface

The MDIO Master Interface module is included in the design if the parameter C_INCLUDE_MDIO is set to '1'. This module provides access to the PHY register for PHY management. The MDIO interface is described in Management Data Input/Output (MDIO) Master Interface Module.

Ethernet Protocol

Ethernet data is encapsulated in frames (Figure 2). The fields and bits in the frame are transmitted from left to right (from the least significant bit to the most significant bit), unless specified otherwise.

Preamble

The preamble field is used for synchronization and must contain seven bytes with the pattern "10101010". If a collision is detected during the transmission of the preamble or start of frame delimiter fields, the transmission of both fields is completed.

For transmission, this field is always automatically inserted by the Ethernet Lite MAC core and should never appear in the packet data provided to the Ethernet Lite MAC core. For reception, this field is always stripped from the packet data. The Ethernet Lite MAC design does not support the Ethernet 8-byte preamble frame type.

Start Frame Delimiter

The start frame delimiter field marks the start of the frame and must contain the pattern 10101011. If a collision is detected during the transmission of the preamble or start of frame delimiter fields, the transmission of both fields is completed.

The receive data valid signal from the PHY (PHY_dv) can go active during the preamble but is active prior to the start frame delimiter field. For transmission, this field is always automatically inserted by the Ethernet Lite MAC core and should never appear in the packet data provided to the Ethernet Lite MAC core. For reception, this field is always stripped from the packet data.

Destination Address

The destination address field is 6 bytes in length. The least significant bit of the destination address is used to determine if the address is an individual/unicast (0) or group/multicast (1) address. Multicast addresses are used to group logically related stations.

The broadcast address (destination address field is all 1's) is a multicast address that addresses all stations on the LAN. The Ethernet Lite MAC supports transmission and reception of unicast and broadcast packets. The Ethernet Lite MAC core does not support multicast packets. This field is always provided in the packet data for transmissions and is always retained in the receive packet data.

Note: The Ethernet Lite MAC design does not support 16-bit destination addresses as defined in the IEEE 802 standard.

Source Address

The source address field is 6 bytes in length. This field is always provided in the packet data for transmissions and is always retained in the receive packet data.

Note: The Ethernet Lite MAC design does not support 16-bit source addresses as defined in the IEEE 802 standard.

Type/Length

The type/length field is 2 bytes in length. When used as a length field, the value in this field represents the number of bytes in the subsequent data field. This value does not include any bytes that might have been inserted in the padding field following the data field. The value of this field determines if it should be interpreted as a length as defined by the IEEE 802.3 standard or a type field as defined by the Ethernet protocol.

The maximum length of a data field is 1,500 bytes. Therefore, a value in this field that exceeds 1,500 (0x05DC) indicates that a frame type rather than a length value is provided in this field. The IEEE 802.3 standard uses the value 1536 (0x0600) or greater to signal a type field. The Ethernet Lite MAC does not perform any processing of the type/length field. This field is transmitted with the least significant bit first but with the high order byte first. This field is always provided in the packet data for transmissions and is always retained in the receive packet data.

Data

The data field can vary from 0 to 1500 bytes in length. This field is always provided in the packet data for transmissions and is always retained in the receive packet data.

Pad

The pad field can vary from 0 to 46 bytes in length. This field is used to ensure that the frame length is at least 64 bytes in length (the preamble and SFD fields are not considered part of the frame for this calculation) which is required for successful CSMA/CD operation. The values in this field are used in the frame check sequence calculation but are not included in the length field value if it is used. The length of this field and the data field combined must be at least 46 bytes. If the data field contains 0 bytes, the pad field is 46 bytes. If the data field is 46 bytes or more, the pad field has 0 bytes. For transmission, this field is inserted automatically by the Ethernet Lite MAC if required to meet the minimum length requirement. If present in the receive packet, this field is always retained in the receive packet data.

FCS

The FCS field is 4 bytes in length. The value of the FCS field is calculated over the source address, destination address, length/type, data, and pad fields using a 32-bit Cyclic Redundancy Check (CRC) defined in paragraph 3.2.8 of [Ref 5]:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + x^{0}$$

The CRC bits are placed in the FCS field with the x^{31} term in the left most bit of the first byte and the x^0 term is the right most bit of the last byte (that is, the bits of the CRC are transmitted in the order x^{31}, x^{30},..., x^1, x^0).

The Ethernet Lite MAC implementation of the CRC algorithm calculates the CRC value a nibble at a time to coincide with the data size exchanged with the external PHY interface for each transmit and receive clock period.

For transmission, this field is always inserted automatically by the Ethernet Lite MAC core and is always retained in the receive packet data.

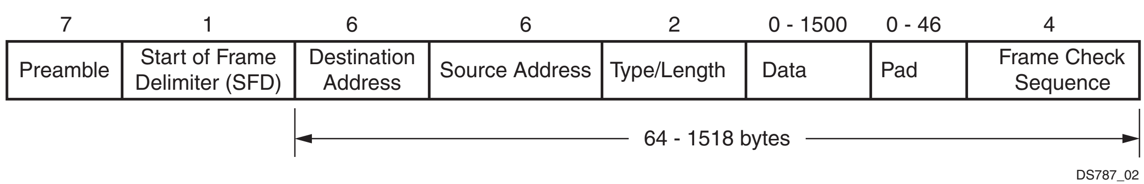


Figure 2: **Ethernet Data Frame**

Interframe Gap and Deferring

Note: Interframe Gap and interframe spacing are used interchangeably and are equivalent.

Frames are transmitted over the serial interface with an interframe gap which is specified by the IEEE Std. 802.3 to be 96 bit times (9.6 uS for 10 MHz and 0.96 uS for 100 MHz). The process for deferring is different for half-duplex and full-duplex systems and is as follows:

Half-Duplex

1. Even when it has nothing to transmit, the Ethernet Lite MAC monitors the bus for traffic by watching the carrier sense signal (PHY_crs) from the external PHY. Whenever the bus is busy (PHY_crs = '1'), the Ethernet Lite MAC defers to the passing frame by delaying any pending transmission of its own.
2. After the last bit of the passing frame (when carrier sense signal changes from true to false), the Ethernet Lite MAC starts the timing of the interframe gap.
3. The Ethernet Lite MAC resets the interframe gap timer if the carrier sense becomes true.

Full-Duplex

The Ethernet Lite MAC does not use the carrier sense signal from the external PHY when in full duplex mode because the bus is not shared and only needs to monitor its own transmissions. After the last bit of an Ethernet Lite MAC transmission, the Ethernet Lite MAC starts the timing of the interframe gap.

Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Method

A full-duplex Ethernet bus is, by definition, a point-to-point dedicated connection between two Ethernet devices capable of simultaneous transmit and receive with no possibility of collisions.

For a half-duplex Ethernet bus, the CSMA/CD media access method defines how two or more stations share a common bus. To transmit, a station waits (defers) for a quiet period on the bus (no other station is transmitting (PHY_crs = ’0’)) and then starts transmission of its message after the interframe gap period. If, after initiating a transmission, the message collides with the message of another station (PHY_col - ’1’), then each transmitting station intentionally continues to transmit (jam) for an additional predefined period (32 bits for 10/100 Mb/s) to ensure propagation of the collision throughout the system. The station remains silent for a random amount of time (back off) before attempting to transmit again. A station can experience a collision during the beginning of its transmission (the collision window) before its transmission has had time to propagate to all stations on the bus. When the collision window has passed, a transmitting station has acquired the bus. Subsequent collisions (late collisions) are avoided because all other (properly functioning) stations are assumed to have detected the transmission and are deferring to it. The time to acquire the bus is based on the round-trip propagation time of the bus (64 byte times for 10/100 Mb/s).

Transmit Flow

The flowchart in Figure 3 shows the high level flow followed for packet transmission.

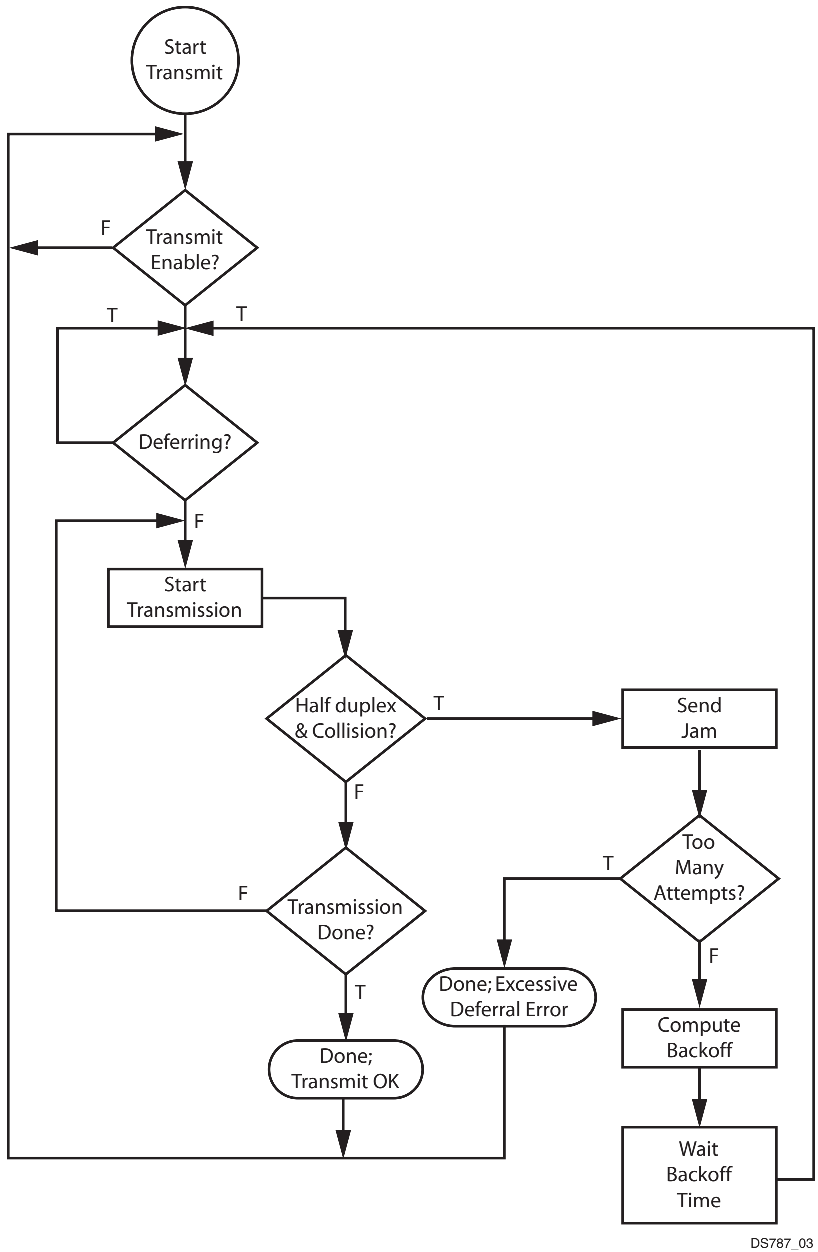


Figure 3: **Transmit Flow**

Receive Flow

The flowchart in Figure 4 shows the high level flow followed for packet reception.

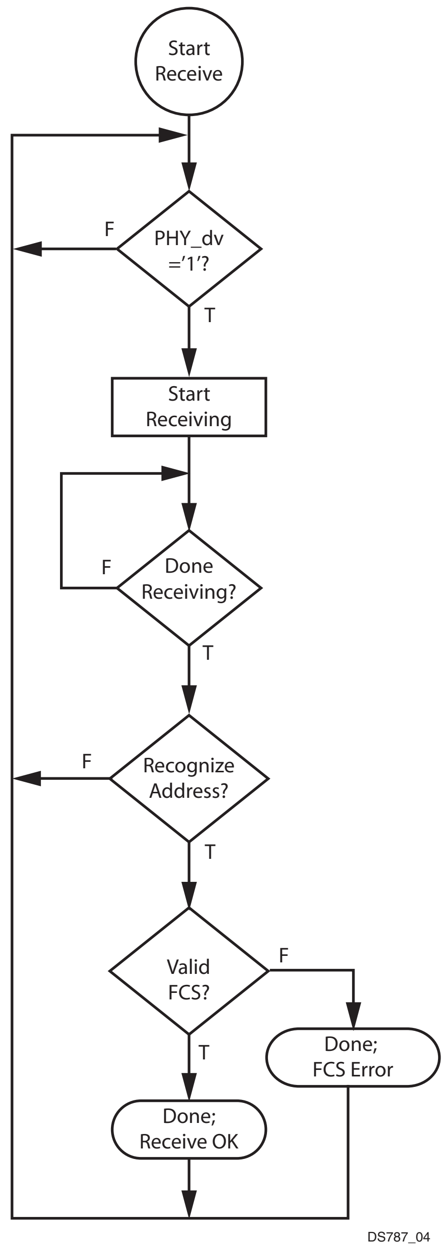


Figure 4: **Receive Flow**

I/O Signals

The Ethernet Lite MAC I/O signals are listed and described in Table 1.

Table 1: **I/O Signal Descriptions**

Port	Signal Name	Interface	I/O	Initial State	Description
System Signals					
P1	S_AXI_ACLK	System	I	-	AXI4 clock
P2	S_AXI_ARESETN	System	I	-	AXI4 reset, active low
P3	IP2INTC_Irpt	System	O	0x0	Edge rising interrupt
AXI4 Write Address Channel Signals					
P4	S_AXI_AWID[C_S_AXI_ID_WIDTH-1:0] (1)	AXI4/ AXI4-Lite	I	-	Write address ID. This signal is the identification tag for the write address group of signals.
P5	S_AXI_AWADDR[C_S_AXI_ADDR_WIDTH-1:0]	AXI4/ AXI4-Lite	I	-	AXI4 Write address. The write address bus gives the address of the first transfer in a write burst transaction.
P6	S_AXI_AWLEN[7:0] (1)	AXI4	I	-	Burst length. This signal gives the exact number of transfers in a burst “00000000“ - “11111111” indicates Burst Length 1 - 256.
P7	S_AXI_AWSIZE[2:0] (1)	AXI4	I	-	Burst size. This signal indicates the size of each transfer in the burst. “000“ - 1 Byte “001“ - 2 byte (Half word) “010“ - 4 byte (Word)
P8	S_AXI_AWBURST[1:0] (1)	AXI4	I	-	Burst type. This signal coupled with the size information, details how the address for each transfer within the burst is calculated. “00“ - FIXED “01“ - INCR “10“ - WRAP “11“ - Reserved
P9	S_AXI_AWCACHE[3:0] (1)	AXI4	I	-	Cache type. This signal provides additional information about the cacheable characteristics of the transfer.
P10	S_AXI_AWVALID	AXI4/ AXI4-Lite	I	-	Write address valid. This signal indicates that valid write address and control information are available.
P11	S_AXI_AWREADY	AXI4/ AXI4-Lite	O	0	Write address ready. This signal indicates that the slave is ready to accept an address and associated control signals.
AXI4 Write Channel Signals					
P12	S_AXI_WDATA[C_S_AXI_DATA_WIDTH	AXI4	I	-	Write data
P13	S_AXI_WSTB[C_S_AXI_DATA_WIDTH/8-1:0]	AXI4	I	-	Write strobes. This signal indicates which byte lanes in S_AXI_WDATA are/is valid.
P14	S_AXI_WLAST (1)	AXI4	I	-	Write last. This signal indicates the last transfer in a write burst.
P15	S_AXI_WVALID	AXI4/ AXI4-Lite	I	-	Write valid. This signal indicates that valid write data and strobes are available.
P16	S_AXI_WREADY	AXI4/ AXI4-Lite	O	0	Write ready. This signal indicates that the slave can accept the write data.

Table 1: **I/O Signal Descriptions *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
AXI4 Write Response Channel Signals					
P17	S_AXI_BID[C_AXI_ID_WIDTH-1:0] (1)	AXI4	O	0	Write response ID. This signal is the identification tag of the write response. The S_AXI_BID value must match the S_AXI_AWID value of the write transaction to which the slave is responding.
P18	S_AXI_BRESP[1:0]	AXI4	O	0	Write response. This signal indicates the status of the write transaction. "00" - OKAY "01" - EXOKAY - NA "10" - SLVERR - NA "11" - DECERR - NA
P19	S_AXI_BVALID	AXI4/ AXI4-Lite	O	0	Write response valid. This signal indicates that a valid write response is available.
P20	S_AXI_BREADY	AXI4/ AXI4-Lite	I	-	Response ready. This signal indicates that the master can accept the response information.
AXI4 Read Address Channel Signals					
P21	S_AXI_ARID[C_S_AXI_ID_ WIDTH-1:0] (1)	AXI4	I	-	Read address ID. This signal is the identification tag for the read address group of signals.
P22	S_AXI_ARADDR[C_S_AXI_ADDR_WIDTH -1:0]	AXI4/ AXI4-Lite	I	-	Read address. The read address bus gives the initial address of a read burst transaction.
P23	S_AXI_ARLEN[7:0] (1)	AXI4	I	-	Burst length. The burst length gives the exact number of transfers in a burst.
P24	S_AXI_ARSIZE[2:0] (1)	AXI4	I	-	Burst size. This signal indicates the size of each transfer in the burst.
P25	S_AXI_ARBURST[1:0] (1)	AXI4	I	-	Burst type. The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated.
P26	S_AXI_ARCACHE[3:0] (1)	AXI4	I	-	Cache type. This signal provides additional information about the cacheable characteristics of the transfer.
P27	S_AXI_ARVALID	AXI4/ AXI4-Lite	I	-	Read address valid. This signal indicates, when high, that the read address and control information is valid and remains stable until the address acknowledgement signal, S_AXI_ARREADY, is high.
P28	S_AXI_ARREADY	AXI4/ AXI4-Lite	O	0	Read address ready. This signal indicates that the slave is ready to accept an address and associated control signals.
AXI4 Read Data Channel Signals					
P29	S_AXI_RID[C_S_AXI_ID_WIDTH-1:0] (1)	AXI4	O	0	Read ID tag. This signal is the ID tag of the read data group of signals. The RID value is generated by the core and matches to the S_AXI_ARID value of the read transaction to which it is responding.
P30	S_AXI_RDATA[C_S_AXI_DATA_WIDTH -1:0]	AXI4/ AXI4-Lite	O	0	Read data

Table 1: **I/O Signal Descriptions *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
P31	S_AXI_RRESP[1:0]	AXI4/ AXI4-Lite	O	0	Read response. This signal indicates the status of the read transfer.
P32	S_AXI_RLAST (1)	AXI4	O	0	Read last. This signal indicates the last transfer in a read burst.
P33	S_AXI_RVALID	AXI4/ AXI4-Lite	O	0	Read valid. This signal indicates that the required read data is available and the read transfer can complete.
P34	S_AXI_RREADY	AXI4/ AXI4-Lite	I	-	Read ready. This signal indicates that the master can accept the read data and response information.
Ethernet Lite MAC Interface Signals					
P35	PHY_tx_clk	PHY	I	-	Ethernet transmit clock input from PHY
P36	PHY_rx_clk	PHY	I	-	Ethernet receive clock input from PHY
P37	PHY_rx_data[3:0]	PHY	I	-	Ethernet receive data. Input from Ethernet PHY.
P38	PHY_tx_data[3:0]	PHY	O	0	Ethernet transmit data. Output to Ethernet PHY.
P39	PHY_dv	PHY	I	-	Ethernet receive data valid. Input from Ethernet PHY.
P40	PHY_rx_er	PHY	I	-	Ethernet receive error. Input from Ethernet PHY.
P41	PHY_tx_en	PHY	O	0	Ethernet transmit enable. Output to Ethernet PHY.
P42	PHY_crs	PHY	I	-	Ethernet carrier sense input from Ethernet PHY
P43	PHY_col	PHY	I	-	Ethernet collision input from Ethernet PHY
P44	PHY_rst_n	PHY	O	-	PHY reset, active low
P45	PHY_MDC (2)	PHY	O	0	Ethernet to PHY MII Management clock
P46	PHY_MDIO_I (2)	PHY	I	-	PHY MDIO data input from 3-state buffer
P47	PHY_MDIO_O (2)	PHY	O	0	PHY MDIO data output to 3-state buffer
P48	PHY_MDIO_T (2)	PHY	O	0	PHY MDIO data output enable to 3-state buffer

Notes:

1. This port is unused when C_S_AXI_PROTOCOL='AXI4LITE'. Output has default assignment.
2. This port is unused when C_INCLUDE_MDIO='1'. Output has default assignment

Design Parameters

The Ethernet Lite MAC has certain features that can be parameterized in the AXI Ethernet Lite design. This allows a design that only uses the resources required by the system and that operates at the best possible performance. The Ethernet Lite MAC design parameters are shown in Table 2.

Inferred Parameters

In addition to the parameters listed in Table 2, additional parameters are inferred for each AXI4 interface in the EDK tools. Through the design, these EDK-inferred parameters control the behavior of the AXI4 Interconnect. For a complete list of the interconnect settings related to the AXI4 interface, see [Ref 4].

Table 2: **Design Parameters**

Generic	Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
System Parameters					
G1	Target FPGA family	C_FAMILY	virtex6, spartan6	virtex6	string
AXI4 Interface Parameters					
G2	AXI4 Base Address	C_BASEADDR	Valid Address (1)	0xFFFFFFFF (3)	std_logic_vector
G3	AXI4 High Address	C_HIGHADDR	Valid Address (2)	0x00000000 (3)	std_logic_vector
G4	AXI4 Identification tag width	C_S_AXI_ID_WIDTH	1-16	4	integer
G5	AXI4 most significant address bus width	C_S_AXI_ADDR_WIDTH	32	32	integer
G6	AXI4 data bus width	C_S_AXI_DATA_WIDTH	32	32	integer
G8	AXI4 protocol	C_S_AXI_PROTOCOL	AXI4, AXI4LITE	AXI4	string
G9	AXI4 clock period (in ps)	C_S_AXI_ACLK_PERIOD_PS	Requirement as stated in note (4)	10000	integer
Ethernet Lite MAC Parameters					
G10	Half duplex transmit	C_DUPLEX	1 = Only full duplex operation available 0 = Only half duplex operation available	1	integer
G11	AXI4 most significant address bus width	C_TX_PING_PONG	1 = Two transmit buffers 0 = Single memory transmit buffer	0	integer
G12	Include second receive buffer	C_RX_PING_PONG	1 = Two receive buffers 0 = Single memory receive buffer	0	integer
G13	Include MII Management module	C_INCLUDE_MDIO (5)	1 = Include MDIO module 0 = No MDIO module	1	integer
G14	Include Internal Loopback	C_INCLUDE_INTERNAL_LOOPBACK (6)	1 = Include internal loopback support 0 = No internal loopback support	0	integer
G15	Include global buffers for PHY clocks	C_INCLUDE_GLOBAL_BUFFERS (7)	1 = Include global buffers for PHY clocks 0 = Use normal input buffers for PHY clocks	0	integer

Table 2: **Design Parameters *(Cont'd)***

Generic	Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G16	Include I/O constraints on the PHY ports through TCL file	C_INCLUDE_PHY_ CONSTRAINTS (8)	1 = Include PHY signal I/O constraints through TCL 0 = Exclude PHY signal I/O constraints	1	integer

Notes:

1. User-set values.
2. The range specified by C_HIGHADDR - C_BASEADDR must be a power of 2 and greater than equal to C_BASEADDR + 0x1FFF.
3. An invalid default value is specified to ensure that the actual value is set, that is, if the value is not set, a compiler error is generated.
4. The AXI4 clock frequency must be $\geq$ 50 MHz for 100 Mb/s Ethernet operation and $\geq$ 5 MHz for 10 Mb/s Ethernet operation
5. Including the MDIO interface allows PHY register access from Ethernet Lite MAC core.
6. Enabling this parameter includes BUFG for PHY clock switching when loopback is enabled.
7. Enabling this parameter includes global buffers for PHY clocks which can be used to minimize the clock skew on the PHY clocks.
8. Enabling this parameter includes I/O constraints on the PHY ports through TCL. If internal PHY is used, this parameter has to be disabled.

Allowable Parameter Combinations

The Ethernet Lite MAC is a synchronous design. Due to the state machine control architecture of receive and transmit operations, the AXI4 clock must be greater than or equal to 50 MHz to allow Ethernet operation at 100 Mb/s and greater than or equal to 5 MHz for Ethernet operation at 10 Mb/s.

The address range specified by C_BASEADDR and C_HIGHADDR must be a power of 2, and C_HIGHADDR range must be at least 0x2000. For example, if C_BASEADDR = 0xE0000000, C_HIGHADDR must be at least = 0xE0001FFF.

For the Spartan®-6 family, the parameter C_INCLUDE_GLOBAL_BUFFERS must be set to 0 because of the architecture limitation.

Dependencies between Parameters and I/O Signals

The dependencies between the Ethernet Lite MAC design parameters and I/O signals are described in Table 3. In addition, when certain features are parameterized out of the design, the related logic is no longer a part of the design. The unused input signals and related output signals are set to a specified value.

Table 3: **Parameter-I/O Signal Dependencies**

Generic or Port	Name	Affects	Depends	Relationship Description
Design Parameters				
G4	C_S_AXI_ID_WIDTH	P4, P17, P21, P29	-	Defines width of the ports
G5	C_S_AXI_ADDR_WIDTH	P5, P22	-	Defines width of the ports
G6	C_S_AXI_DATA_WIDTH	P12, P13, P30	-	Defines width of the ports
G8	C_S_AXI_PROTOCOL	P4, P6-P9, P14, P17, P21, P23-P26, P29, P32	-	Ports are unused when C_S_AXI_PROTOCOL = "AXI4LITE"

Table 3: **Parameter-I/O Signal Dependencies *(Cont'd)***

Generic or Port	Name	Affects	Depends	Relationship Description
G13	C_INCLUDE_MDIO	P45-P48	-	PHY_MDC and PHY_MDIO are included in the core only if C_INCLUDE_MDIO = 1
I/O Signals				
P4	S_AXI_AWID[C_S_AXI_ID_WIDTH-1:0]	-	G4, G8	Port width depends on C_S_AXI_ID_WIDTH. Port is unused when C_S_AXI_PROTOCOL = "AXI4LITE".
P5	S_AXI_AWADDR[C_S_AXI_ADDR_WIDTH-1:0]	-	G5	Port width depends on C_S_AXI_ADDDR_WIDTH
P6	S_AXI_AWLEN[7:0]	-	G8	Port is unused when C_S_AXI_PROTOCOL = "AXI4LITE"
P7	S_AXI_AWSIZE[2:0]	-	G8	Port is unused when C_S_AXI_PROTOCOL = "AXI4LITE"
P8	S_AXI_AWBURST[1:0]	-	G8	Port is unused when C_S_AXI_PROTOCOL = "AXI4LITE"
P9	S_AXI_AWCACHE[4:0]	-	G8	Port is unused when C_S_AXI_PROTOCOL = "AXI4LITE"
P12	S_AXI_WDATA[C_S_AXI_DATA_WIDTH	-	G6	Port width depends on C_S_AXI_DATA_WIDTH
P13	S_AXI_WSTB[C_S_AXI_DATA_WIDTH/8-1:0]	-	G6	Port width depends on C_S_AXI_DATA_WIDTH
P14	S_AXI_WLAST	-	G8	Port is unused when C_S_AXI_PROTOCOL = "AXI4LITE"
P17	S_AXI_BID[C_S_AXI_ID_WIDTH-1:0]	-	G4, G8	Port width depends on C_S_AXI_ID_WIDTH. Port is unused when C_S_AXI_PROTOCOL = "AXI4LITE".
P21	S_AXI_ARID[C_S_AXI_ID_WIDTH-1:0]	-	G4, G8	Port width depends on C_S_AXI_ID_WIDTH. Port is unused when C_S_AXI_PROTOCOL = "AXI4LITE".
P22	S_AXI_ARADDR[C_S_AXI_ADDR_WIDTH -1:0]	-	G5	Port width depends on C_S_AXI_ADDR_WIDTH
P23	S_AXI_ARLEN[7:0]	-	G8	Port is unused when C_S_AXI_PROTOCOL = "AXI4LITE"
P24	S_AXI_ARSIZE[2:0]	-	G8	Port is unused when C_S_AXI_PROTOCOL = "AXI4LITE"
P25	S_AXI_ARBURST[1:0]	-	G8	Port is unused when C_S_AXI_PROTOCOL = "AXI4LITE"
P26	S_AXI_ARCACHE[4:0]	-	G8	Port is unused when C_S_AXI_PROTOCOL = "AXI4LITE"
P29	S_AXI_RID[C_S_AXI_ID_WIDTH-1:0]	-	G4, G8	Port width depends on C_S_AXI_ID_WIDTH. Port is unused when C_S_AXI_PROTOCOL = "AXI4LITE".
P30	S_AXI_RDATA[C_S_AXI_DATA_WIDTH -1:0]	-	G6	Port width depends on C_S_AXI_DATA_WIDTH

Table 3: **Parameter-I/O Signal Dependencies *(Cont'd)***

Generic or Port	Name	Affects	Depends	Relationship Description
P32	S_AXI_RLAST	-	G8	Port is unused when C_S_AXI_PROTOCOL = "AXI4LITE"
P45	PHY_MDC	-	G13	This port is included in the core only if C_INCLUDE_MDIO = 1
P46	PHY_MDIO_I	-	G13	This port is included in the core only if C_INCLUDE_MDIO = 1
P47	PHY_MDIO_O	-	G13	This port is included in the core only if C_INCLUDE_MDIO = 1
P48	PHY_MDIO_T	-	G13	This port is included in the core only if C_INCLUDE_MDIO = 1

Ethernet Lite MAC Memory Map

The Ethernet Lite MAC memory map is shown in Table 4. The Ethernet frame should be stored in the TX buffer in byte increasing order. The Ethernet Lite MAC core receives the frame and stores it in RX buffer in byte increasing order.

Table 4: **Ethernet Lite MAC Memory Map**

<table>
<tr><th>Address Offset</th><th>Parameter Dependency</th><th>Memory Location Function</th></tr>
<tr><td>0x0000</td><td rowspan="5">TX PING Buffer
C_TX_PING_PONG = '0' or '1'</td><td>Destination Address Bytes 3 - 0
or
MAC Address Bytes 3 - 0</td></tr>
<tr><td>0x0004</td><td>Source Address Bytes 1 - 0
or
MAC Address Bytes 5 - 4
Destination Address Bytes 5 - 4</td></tr>
<tr><td>0x0008</td><td>Source Address Bytes 5 - 2</td></tr>
<tr><td>0x000C</td><td>Data Field Bytes 1 - 0
Type/Length Field</td></tr>
<tr><td>0x0010 - 0x07DC</td><td>Remaining Data Field Bytes</td></tr>
<tr><td>0x7E4</td><td rowspan="4">MDIO Registers [1]</td><td>MDIO Address</td></tr>
<tr><td>0x7E8</td><td>MDIO Write Data</td></tr>
<tr><td>0x7EC</td><td>MDIO Read Data</td></tr>
<tr><td>0x07F0</td><td>MDIO Control</td></tr>
<tr><td>0x07F4</td><td rowspan="3">Transmit Register</td><td>Packet Length</td></tr>
<tr><td>0x07F8</td><td>Global Interrupt Enable</td></tr>
<tr><td>0x07FC</td><td>Control</td></tr>
</table>

Table 4: **Ethernet Lite MAC Memory Map *(Cont'd)***

Address Offset	Parameter Dependency	Memory Location Function
0x0800	TX PONG Buffer C_TX_PING_PONG = '1' else unused	Destination Address Bytes 3 - 0 or MAC Address Bytes 3 - 0
0x0804		Source Address Bytes 1 - 0 or MAC Address Bytes 5 - 4 Destination Address Bytes 5 - 4
0x0808		Source Address Bytes 5 - 2
0x080C		Data Field Bytes 1 - 0 Type/Length Field
0x0810 - 0x0FE0		Remaining Data Field Bytes
0x0FE0 - 0x0FF0		Reserved
0x0FF4		Packet Length
0x0FF8		Reserved
0x0FFC		Control
0x1000	RX PING Buffer C_RX_PING_PONG = '0' or '1'	Destination Address Bytes 3 - 0
0x1004		Source Address Bytes 1 - 0 Destination Address Bytes 5 - 4
0x1008		Source Address Bytes 5 - 2
0x100C		Data Field Bytes 1 - 0 Type/Length Field
0x1010 - 0x17DC		Remaining Data and CRC Field Bytes
0x17E0 - 0x17F8		Reserved
0x17FC		Control
0x1800	RX PONG Buffer C_RX_PING_PONG = '1' else unused	Destination Address Bytes 3 - 0
0x1804		Source Address Bytes 1 - 0 Destination Address Bytes 5 - 4
0x1808		Source Address Bytes 5 - 2
0x180C		Data Field Bytes 1 - 0 Type/Length Field
0x1810 - 0x1FDC		Remaining Data and CRC Field Bytes
0x1FE0 - 0x1FF8		Reserved
0x1FFC		Control

1. The MDIO registers are included in the memory map only if C_INCLUDE_MDIO = 1. If the MDIO interface is not enabled, this register space is treated as reserved.

Register Descriptions

Table 5 shows all the Ethernet Lite MAC core registers and their addresses. Tables 6 to 15 show the bit allocation and reset values of the registers.

Table 5: **Registers**

Base Address + Offset (hex)	Register Name	Access Type	Default Value (hex)	Description
C_BASEADDR + 0x07E4	MDIOADDR (1)	Read/Write	0x0	MDIO address register
C_BASEADDR + 0x07E8	MDIOWR (1)	Read/Write	0x0	MDIO write data register
C_BASEADDR + 0x07EC	MDIORD (1)	Read (2)	0x0	MDIO read data register
C_BASEADDR + 0x07F0	MDIOCTRL (1)	Read/Write	0x0	MDIO control register
C_BASEADDR + 0x07F4	TX Ping Length	Read/Write	0x0	Transmit length register for ping buffer
C_BASEADDR + 0x07F8	GIE	Read/Write	0x0	Global interrupt register
C_BASEADDR + 0x07FC	TX Ping Control	Read/Write	0x0	Transmit control register for ping buffer
C_BASEADDR + 0x0FF4	TX Pong Length (3)	Read/Write	0x0	Transmit length register for pong buffer
C_BASEADDR + 0x0FFC	TX Pong Control (3)	Read/Write	0x0	Transmit control register for pong buffer
C_BASEADDR + 0x17FC	RX Ping Control	Read/Write	0x0	Receive control register for ping buffer
C_BASEADDR + 0x1FFC	RX Pong Control (4)	Read/Write	0x0	Receive control register for pong buffer

Notes:
1. These registers are included only if C_INCLUDE_MDIO=1.
2. Writing of a read only register has no effect.
3. These registers are included only if C_TX_PING_PONG=1.
4. These registers are included only if C_RX_PING_PONG=1.

Transmit Length Register

The Transmit Length register is a 32-bit read/write register (Figure 5). This register is used to store the length (in bytes) of the transmit data stored in dual port memory. The higher 8 bits of the length value should be stored in data bits 15 to 8, while the lower 8 bits should be stored in data bits 7 to 0. The bit definition of this register for the ping and pong buffer interface is shown in Table 6.

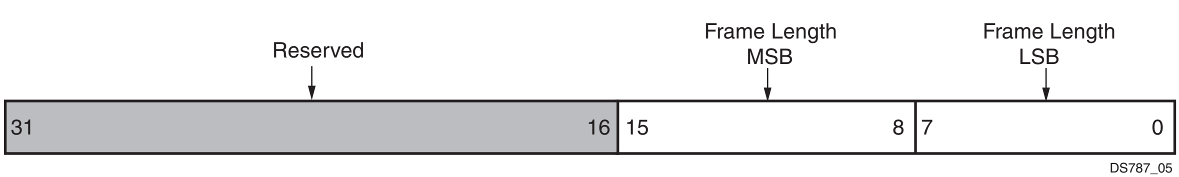


Figure 5: **Transmit Length Register**

Table 6: **Transmit Length Register Bit Definitions (C_BASEADDR + 0x07F4),(C_BASEADDR + 0x0FF4)**

Bit	Name	Access	Reset value	Description
31-16	Reserved	N/A	N/A	Reserved
15-8	MSB	Read/Write	“0x00”	The higher 8-bits of the frame length
7-0	LSB	Read/Write	“0x00”	The lower 8-bits of the frame length

Global Interrupt Enable Register (GIE)

The Global Interrupt Enable register is a 32-bit read/write register (Figure 6). The GIE register is used to enable transmit complete interrupt events. This event is a pulse and occurs when the memory is ready to accept new data. This includes the completion of programing the MAC address. The transmit complete interrupt occurs only if the GIE and transmit/receive interrupt enable bit are both set to '1'. The bit definition of this register is shown in Table 7.

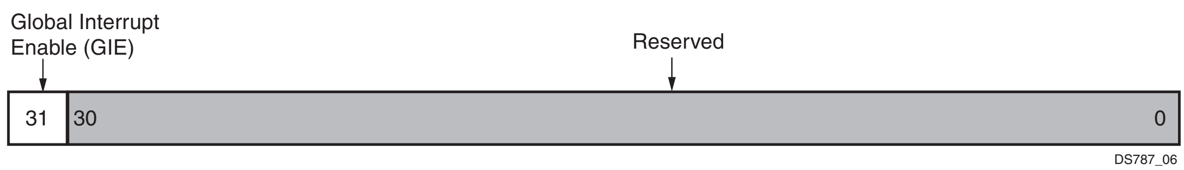


Figure 6: **Global Interrupt Enable**

Table 7: **Global Interrupt Enable Register Bit Definitions (C_BASEADDR + 0x07F8)**

Bit	Name	Access	Reset value	Description
31	GIE	Read/Write	'0'	Global Interrupt Enable bit
30-0	Reserved	N/A	N/A	Reserved

Transmit Control Register (Ping)

The Transmit Control register for the ping buffer is a 32-bit read/write register (Figure 7). This register is used to enable the global interrupt, internal loopback and to initiate transmit transactions. The bit definition of this register is shown in Table 8.

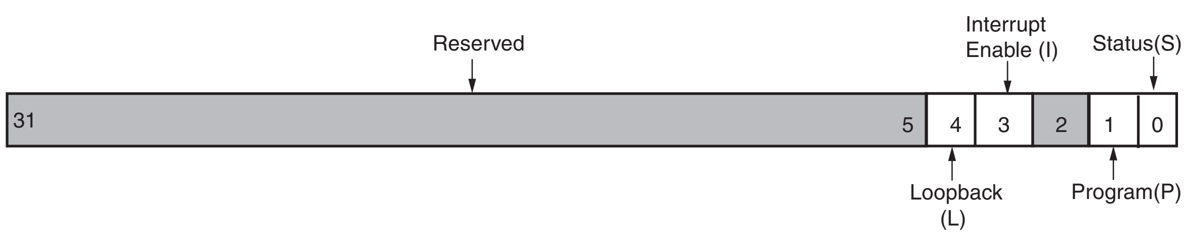


Figure 7: **Transmit Control Register (Ping)**

Table 8: **Transmit Control Register Bit Definitions (C_BASEADDR + 0x07FC)**

Bit	Name	Access	Reset value	Description
31-5	Reserved	N/A	N/A	Reserved
4	Loopback (1)	Read/Write	'0'	Internal loopback enable bit 0 - No internal loopback 1 - Internal loopback enable
3	Interrupt Enable	Read/Write	'0'	Transmit Interrupt Enable bit 0 - Disable transmit interrupt 1 - Enable transmit interrupt
2	Reserved	N/A	N/A	Reserved

Table 8: **Transmit Control Register Bit Definitions (C_BASEADDR + 0x07FC)** ***(Cont'd)***

Bit	Name	Access	Reset value	Description
1	Program	Read/Write	'0'	Ethernet Lite MAC address program bit. Setting this bit and status bit configures the new MAC address for the core as described in MAC Address.
0	Status	Read/Write	'0'	Transmit ping buffer status indicator 0 - Transmit ping buffer is ready to accept new frame 1 - Frame transfer is in progress. Setting this bit initiates transmit transaction. When transmit is complete, the Ethernet Lite MAC core clears this bit.

Notes:

1. Internal Loopback is supported only in full duplex operation mode.

Transmit Control Register (Pong)

The Transmit Control register for the pong buffer is a 32-bit read/write register (Figure 8). This register is used for MAC address programming and to initiate transmit transaction from the pong buffer. The bit definition of this register is shown in Table 9.

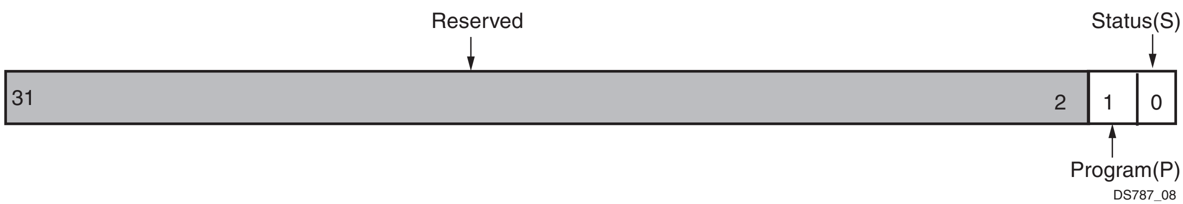


Figure 8: **Transmit Control Register (Pong)**

Table 9: **Transmit Control Register Bit Definitions (C_BASEADDR + 0x0FFC)**

Bit	Name	Access	Reset value	Description
31-2	Reserved	N/A	N/A	Reserved
1	Program	Read/Write	'0'	Ethernet Lite MAC address program bit. Setting this bit and status bit configures the new MAC address for the core as described in MAC Address.
0	Status	Read/Write	'0'	Transmit pong buffer status indicator 0 - Transmit pong buffer is ready to accept a new frame 1 - Frame transfer is in progress. Setting this bit initiates transmit transaction. When transmit is complete, the Ethernet Lite MAC core clears this bit.

Receive Control Register (Ping)

The Receive Control register for the ping buffer is a 32-bit read/write register (Figure 9). This register indicates whether there is a new packet in the ping buffer. The bit definition of this register is shown in Table 10.

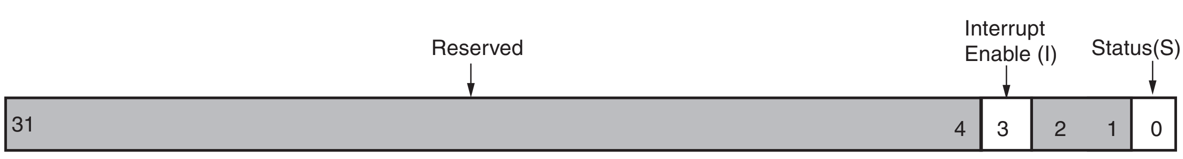


Figure 9: **Receive Control Register (Ping)**

Table 10: **Receive Control Register Bit Definitions (C_BASEADDR + 0x17FC)**

Bit	Name	Access	Reset value	Description
31-4	Reserved	N/A	N/A	Reserved
3	Interrupt Enable	Read/Write	'0'	Receive Interrupt Enable bit 0 - Disable receive interrupt 1 - Enable receive interrupt
2-1	Reserved	N/A	N/A	Reserved
0	Status	Read/Write	'0'	Receive status indicator 0 - Receive ping buffer is empty. Ethernet Lite MAC can accept new available valid packet. 1 - Indicates presence of receive packet ready for software processing. When the software reads the packet from the receive ping buffer, the software must clear this bit.

Receive Control Register (Pong)

The Receive Control register for the pong buffer is a 32-bit read/write register (Figure 10). This register indicates whether there is a new packet in the pong buffer. The bit definition of this register is shown in Table 11.

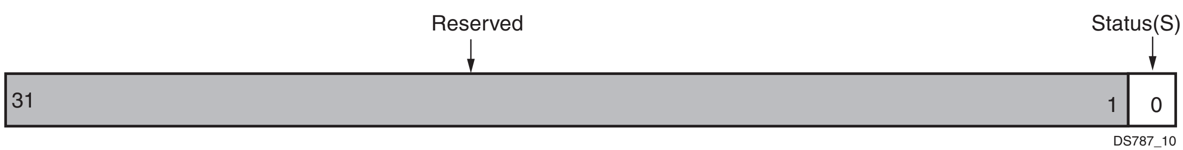


Figure 10: **Receive Control Register (Pong)**

Table 11: **Receive Control Register Bit Definitions (C_BASEADDR + 0x1FFC)**

Bit	Name	Access	Reset value	Description
31-1	Reserved	N/A	N/A	Reserved
0	Status	Read/Write	'0'	Receive status indicator 0 - Receive pong buffer is empty. Ethernet Lite MAC can accept new available valid packet. 1 - Indicates presence of receive packet ready for software processing. When the software reads the packet from the receive pong buffer, the software must clear this bit.

MDIO Address Register (MDIOADDR)

The MDIOADDR is a 32-bit read/write register (Figure 11). This register is used to configure the PHY device address, PHY register address and type of MDIO transaction. The bit definition of this register is shown in Table 12.

Figure 11: **MDIO Address Register**

Table 12: **MDIO Address Register Bit Definition (C_BASEADDR + 0x07E4)**

Bit	Name	Access	Reset Value	Description
31-11	Reserved	N/A	N/A	Reserved
10	OP	Read/Write	'0'	Operation Access Type 0 - Write Access 1 - Read Access
9-5	PHYADDR	Read/Write	"00000"	PHY device address
4-0	REGADDR	Read/Write	"00000"	PHY register address

MDIO Write Data Register (MDIOWR)

The MDIOWR is a 32-bit read/write register (Figure 12). This register contains 16-bit data to be written in to the PHY register. The bit definition of this register is shown in Table 13.

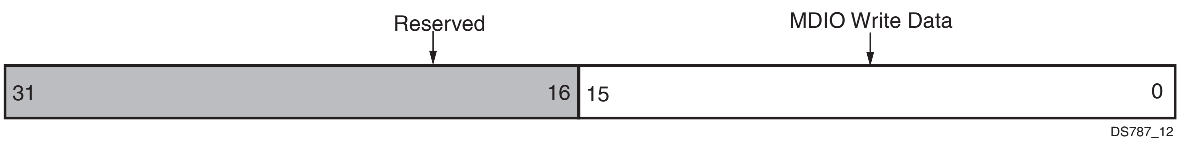


Figure 12: **MDIO Write Data Register**

Table 13: **MDIO Write Data Register Bit Definition (C_BASEADDR + 0x07E8)**

Bit	Name	Access	Reset Value	Description
31-16	Reserved	N/A	N/A	Reserved
15-0	Write Data	Read/Write	0x0000	MDIO write data to be written to PHY register

MDIO Read Data Register (MDIORD)

The MDIORD is a 32-bit read/write register (Figure 13). This register contains 16-bit read data from the PHY register. The bit definition of this register is shown in Table 14.

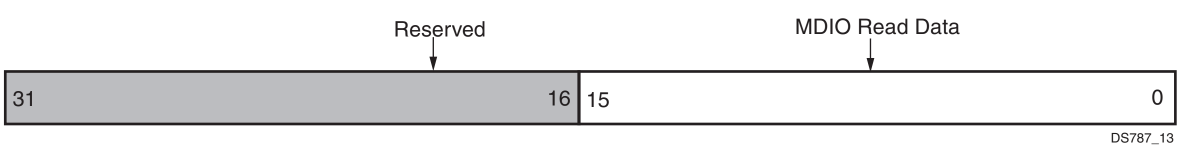


Figure 13: **MDIO Read Data Register**

Table 14: **MDIO Read Data Register Bit Definition (C_BASEADDR + 0x07EC)**

Bit	Name	Access	Reset Value	Description
31-16	Reserved	N/A	N/A	Reserved
15-0	Read Data	Read	0x0000	MDIO read data from the PHY register

MDIO Control Register (MDIOCTRL)

The MDIOCTRL is a 32-bit read/write register (Figure 14). This register contains status and control information of the MDIO interface. The MDIO Enable (bit-3) of this register is used to enable the MDIO interface. The bit definition of this register is shown in Table 15.

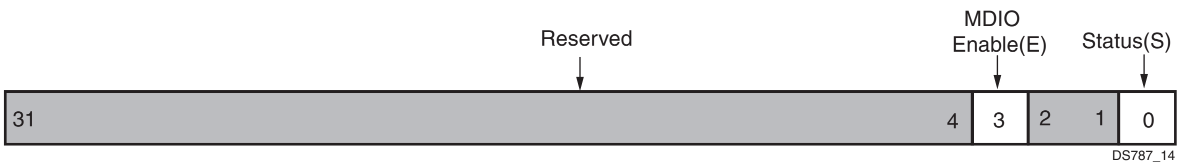


Figure 14: **MDIO Control Register**

Table 15: **MDIO Control Register Bit Definition (C_BASEADDR + 0x07F0)**

Bit	Name	Access	Reset Value	Description
31-4	Reserved	N/A	N/A	Reserved
3	MDIO Enable	Read/Write	'0'	MDIO enable bit 0 - Disable MDIO interface 1 - Enable MDIO interface
2-1	Reserved	N/A	N/A	Reserved
0	Status	Read/Write	'0'	MDIO status bit 0 - MDIO transfer is complete and core is ready to accept new MDIO request 1 - MDIO transfer is in progress. Setting this bit initiates MDIO transaction. When the MDIO transaction is complete, the Ethernet Lite MAC core clears this bit

Processor Interface

The Ethernet Lite MAC core has a very simple interface to the processor. The interface is implemented with a 32-bit wide data interface to a 4K byte block of dual port memory. The registers are implemented in the dual port memory.

The dual port memory is allocated so that 2K bytes are dedicated to the transmit function and 2K bytes are dedicated to the receive function. This memory is capable of holding one maximum length Ethernet packet in the receive and transmit memory areas simultaneously. The Ethernet Lite MAC core also includes optional 2K byte dual port memory for the pong buffer for the Transmit and Receive interface based on the parameter C_TX_PING_PONG and C_RX_PING_PONG.

Transmit Interface

The transmit data should be stored in the dual port memory starting at address C_BASEADDR + 0x0. Because of the word aligned addressing, the second 4 bytes are located at C_BASEADDR + 0x4. The 32-bit interface requires that all 4 bytes be written at once; there are no individual byte enables within one 32- bit word.

The transmit data must include the destination address (6 bytes), the source address (6 bytes), the type/length field (2 bytes), and the data field (0 - 1500 bytes). The preamble, start of frame, and CRC should not be included in the dual port memory. The destination, source, type/length, and data must be packed together in contiguous memory.

Dual port memory address C_BASEADDR + 0x07F8 is used to set the global interrupt enable (GIE) bit. Setting the GIE = '0' prevents the IP2INTC_Irpt from going active during an interrupt event. Setting GIE = '1' allows the IP2INTC_Irpt to go active when an interrupt event occurs.

Dual port memory addresses C_BASEADDR + 0x07F4 is used to store the length (in bytes) of the transmit data stored in dual port memory. The higher 8 bits of the length value should be stored in data bits 15 to 8, while the lower 8 bits should be stored in data bits 7 to 0.

The least two significant bits of dual port memory address C_BASEADDR + 0x07FC are control bits (Program or "P" and Status or "S"). The fourth bit (bit 3 on the data bus) (Transmit Interrupt Enable or "I") is used to enable transmit complete interrupt events. This event is a pulse and occurs when the memory is ready to accept new data. This includes the completion of programing the MAC address. The transmit complete interrupt occurs only if GIE and this bit are both set to '1'.

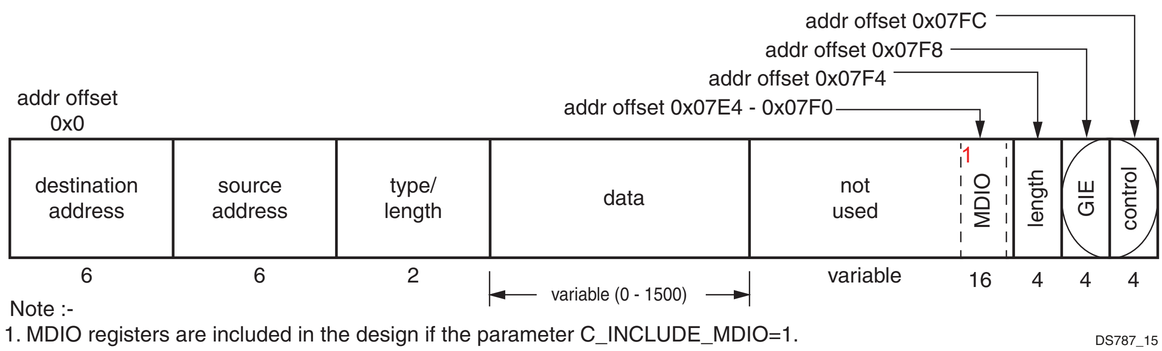


Figure 15: **Transmit Dual Port Memory**

Software Sequence for Transmit with Ping Buffer

The Ethernet Lite MAC core requires that the length of the transmit data to be stored in address offset 0x07F4 before the software sets the status bit at offset 0x07FC.

The software sequence for initiating a transmit is:

- The software stores the transmit data in the dual port memory starting at address offset 0x0
- The software writes the length data in the dual port memory at address offset 0x07F4
- The software writes a '1' to the status bit at address offset 0x07FC (bit 0 on the data bus)
- The software monitors the status bit and waits until it is set to '0' by the Ethernet Lite MAC core before initiating another transmit
- If the transmit interrupt and the global interrupt are both enabled, an interrupt occurs when the Ethernet Lite MAC core clears the status bit
- The transmit interrupt, if enabled, also occurs with the completion of writing the MAC address

Setting the status bit to a '1' initiates the Ethernet Lite MAC core transmit to perform the following functions:

- Generates the preamble and start of frame fields
- Reads the length and the specified amount of data out of the dual port memory according to the length value, adding padding if required
- Detects any collision and performs any jamming, backs off and retries, if necessary
- Calculates the CRC and appends it to the end of the data
- Clears the status bit at the completion of the transmission
- Clearing the status bit causes a transmit complete interrupt, if enabled

Software Sequence for Transmit with Ping-Pong Buffer

If C_TX_PING_PONG is set to 1 then two memory buffers exist for the transmit data. The original (ping transmit buffer) remains at the same memory address and controls the global interrupt enable. The second (pong buffer) is mapped at C_BASEADDR + 0x0800 through 0x0FFC. The length and status must be used in the pong buffer the same as in the ping buffer. The I bit and GIE bit are not used from the pong buffer (that is, the I bit and GIE bit of the ping buffer alone control the I bit and GIE bit settings for both buffers). The MAC address can be set from the pong buffer. The transmitter always empties the ping buffer first after a reset. Then, if data is ready to be transmitted from the pong buffer, that transmission takes place. However, if the pong buffer is not ready to transmit data the Ethernet Lite MAC core begins to monitor both the ping and pong buffers and transmits the buffer that is ready first.

The software sequence for initiating a transmit with both a ping and pong buffer is:

- The software stores the transmit data in the dual port memory starting at address offset 0x0
- The software writes the length data in the dual port memory at address offset 0x07F4
- The software writes a '1' to the status bit at address offset 0x07FC (bit 0 on the data bus)
- The software can write to the pong buffer (0x0800 - 0x0FFC) at any time
- The software monitors the status bit in the ping buffer and waits until it is set to '0', or waits for a transmit complete interrupt, before filling the ping buffer again
- If the transmit interrupt and the global interrupt are both enabled, an interrupt occurs when the Ethernet Lite MAC core clears the status bit
- The transmit interrupt, if enabled, also occurs with the completion of writing the MAC address

Setting the status bit to a '1' initiates the Ethernet Lite MAC core transmit which performs the following functions:

- Generates the preamble and start of frame fields
- Reads the length and the specified amount of data out of the dual port memory according to the length value, adding padding if required
- Detects any collision and performs any jamming, backs off, and retries if necessary
- Calculates the CRC and appends it to the end of the data
- Clears the status bit at the completion of the transmission
- Clearing the status bit causes a transmit complete interrupt if enabled
- The hardware then transmits the pong buffer if it is available, or begins monitoring both ping and pong buffers until data is available

MAC Address

The 48-bit MAC address defaults at reset to 00-00-5E-00-FA-CE. This value can be changed by performing an address program operation using the transmit dual port memory.

The software sequence for programming a new MAC address is:

- The software loads the new MAC address in the transmit dual port memory, starting at address offset 0x0. The most significant four bytes are stored at address offset 0x0 and the least significant two bytes are stored at address offset 0x4. The MAC address can also be programmed from the pong buffer starting at 0x0800.
- The software writes a '1' to both the program bit (bit 1 on the data bus) and the status bit (bit 0 on the data bus) at address offset 0x07FC. The pong buffer address is 0x0FFC.
- The software monitors the status and program bits and waits until they are set to '0's before performing any additional Ethernet operations.

A transmit complete interrupt, if enabled, occurs when the status and program bits are cleared

Receive Interface

The entire receive frame data from destination address to the end of the CRC is stored in the receive dual port memory area which starts at address C_BASEADDR + 0x1000. The preamble and start of frame fields are not stored in dual port memory. Dual port memory address offset 0x17FC (bit 0 on the data bus) is used as a status to indicate the presence of a receive packet that is ready for processing by the software.

Dual port memory address offset 0x17FC (bit 3 on the data bus) is the Receive Interrupt enable. This event is a pulse and occurs when the memory has data available. The receive complete interrupt occurs only if this bit and GIE are both set to '1'.

When the status bit is '0', the Ethernet Lite MAC monitors the Ethernet for packets with a destination address that matches its MAC address or the broadcast address. If a packet satisfies either of these conditions, the packet is received and stored in dual port memory starting at address offset 0x1000. When the packet has been received, the Ethernet Lite MAC core verifies the CRC. If the CRC value is correct, the status bit is set. If the CRC bit is incorrect, the status bit is not set and the Ethernet Lite MAC core resumes monitoring the Ethernet bus. Also, if the Ethernet Lite MAC core receive Runt Frame (frame length less than the 60 Bytes) with a valid CRC, the core does not set the status bit and the interrupt is not generated. When the status bit is set, the Ethernet Lite MAC does not perform any receive operations until the bit has been cleared to '0' by the software, indicating that all of the receive data has been retrieved from the dual port memory.

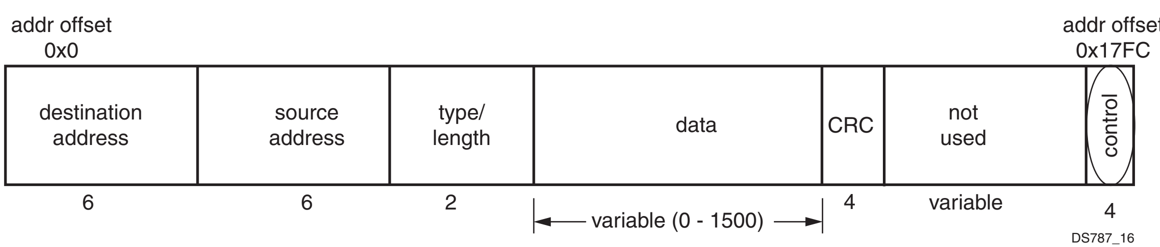


Figure 16: **Receive Dual Port Memory**

Software Sequence for Receive with Ping Buffer

The software sequence for processing a receive is:

- The software monitors the receive status bit until it is set to '1' by the Ethernet Lite MAC core and waits for a receive complete interrupt, if enabled
- When the status is set to '1', or a receive complete interrupt has occurred, the software reads the entire receive data out of the dual port memory
- The software writes a '0' to the receive status bit enabling the Ethernet Lite MAC core to resume receive processing

Software Sequence for Receive Ping-Pong

If C_RX_PING_PONG is set to '1' then two memory buffers exist for the receive data. The original (ping receive buffer) remains at the same memory location. The second (pong receiver buffer) is mapped to C_BASEADDR + 0x1800 through 0x1FFC. Data is stored the same way in the pong buffer as it is in the ping buffer.

The software sequence for processing a receive packet(s) with C_RX_PING_PONG = 1 is:

- The software monitors the ping receive status bit until it is set to '1' by the Ethernet Lite MAC, or waits for a receive complete interrupt, if enabled
- When the ping status is set to '1', or a receive complete interrupt has occurred, the software reads the entire receive data out of the ping dual port memory
- The Ethernet Lite MAC receives the next packet and stores it in the pong receive buffer

- The software writes a ’0’ to the ping receive status bit, enabling the Ethernet Lite MAC core to receive another packet in the ping receive buffer
- The software monitors the pong receive status bit until it is set to ’1’ by the Ethernet Lite MAC core, or waits for a receive complete interrupt, if enabled
- When the pong status is set to ’1’, or a receive complete interrupt has occurred, the software reads the entire receive data out of the ping dual port memory
- The hardware always writes the first received packet, after a reset, to the ping buffer; the second received packet is written to the pong buffer and the third received packet is written to the ping buffer

Management Data Input/Output (MDIO) Master Interface Module

The Management Data Input/Output Master Interface module is included in the design if the parameter C_INCLUDE_MDIO = 1. Including this logic allows Ethernet Lite MAC core to access PHY configuration registers. The MDIO Master Interface module is designed to incorporate the features described in IEEE 802.3 Media Independent Interface (MII) specification.

The MDIO module generates management data clock to the PHY(PHY_MDC) with minimum period of 400 ns. PHY_MDC is sourced to PHY as timing reference for transfer of information on the PHY_MDIO (Management Data Input/Output) data signal.

PHY_MDIO is a bidirectional signal between the PHY and MDIO module. It is used to transfer control and status information between the PHY and the MDIO module. The control information is driven by the MDIO module synchronously with respect to PHY_MDC and is sampled synchronously by the PHY. The status information is driven by the PHY synchronously with respect to PHY_MDC and is sampled synchronously by the MDIO module. PHY_MDIO is driven through a 3-state circuit that enables either the MDIO module or the PHY to drive the circuit.

The MDIO interface uses a standard method to access PHY management registers. The MDIO module supports up to 32 PHY devices. To access each PHY device, the PHY device address must be written into the MIDO Address (MDIOADDR) register followed by PHY register address (Figure 11). This module supports access to up to 32 PHY management registers. The write transaction data for the PHY must be written into MDIO Write Data (MDIOWR) register and the status data from the PHY register can be read from the MDIO Read Data (MDIORD) register. The MDIO Control (MDIOCTRL) register is used to initiate to management transaction on the MDIO lines.

MDIO Transactions

The Ethernet Lite MAC requires that the PHY device address and PHY register address be stored in the MDIO Address Register at address offset 0x07E4 before the software sets the status bit in the MDIO Control Register at offset 0x07F0.

The software sequence for initiating a PHY register write transaction is:

- The software reads the MDIOCTRL register to verify if the MDIO master is busy executing a previous request. If the status bit is ‘0’, the MDIO master can accept a new request.
- The software stores the PHY device address and PHY register address and writes ‘0’ to bit 10 in the MDIOADDR register at address offset 0x07E4.
- The software stores the PHY register write data in the MDIOWR register at address offset 0x07E8.
- The software writes ‘1’ in the MDIO enable bit in the MDIOCTRL register at address offset 0x07F0.
- The software writes a ’1’ to the status bit at address offset 0x07F0 (bit 0 on the data bus) to start the MDIO transaction.
- After completing the MDIO write transaction, the Ethernet Lite MAC core clears the status bit.

- The software monitors the status bit and waits until it is set to '0' by the Ethernet Lite MAC before initiating new transaction on the MDIO lines.

The software sequence for initiating a PHY register read transaction is:

- The software reads the MDIOCTRL register to verify if the MDIO master is busy executing a previous request. If the status bit is '0', the MDIO master can accept a new request.
- The software stores the PHY device address and PHY register address and writes '1' to bit 10 in the MDIOADDR register at address offset 0x07E4.
- The software writes '1' in the MDIO enable bit in the MDIOCTRL register at address offset 0x07F0.
- The software writes a '1' to the status bit at address offset 0x07F0 (bit 0 on the data bus) to start the MDIO transaction.
- After completing the MDIO Read transaction, the Ethernet Lite MAC core clears the status bit.

The software monitors the status bit and waits until it is set to '0' by the Ethernet Lite MAC core before initiating a new transaction on the MDIO lines.

Internal Loopback Mode

The Ethernet Lite MAC core can be configured in internal loopback mode by setting the parameter C_INCLUDE_INTERNAL_LOOPBACK to '1' and by setting bit 4 of the Transmit Control Register (Ping). Including the loopback, the logic uses BUFG for PHY clock switching. In this mode, the Ethernet Lite MAC core routes back data on the TX lines to the RX lines. The loopback mode can be tested only in full duplex mode. In this mode, the core does not accept any data from PHY and PHY_tx_clk and PHY_tx_en are used as PHY_rx_clk and PHY_dv internally (Figure 17).

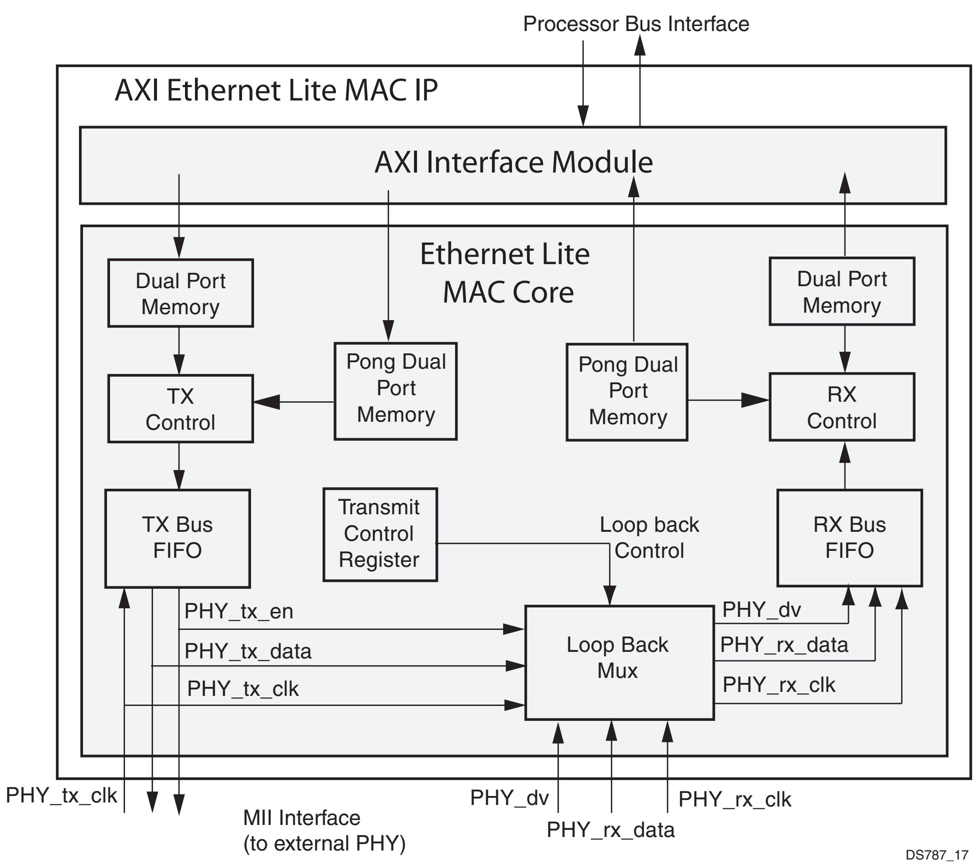


Figure 17: **Internal Loopback Mode**

Clocks

The Ethernet Lite MAC design has three clock domains that are all asynchronous to each other. The clock domain diagram for the Ethernet Lite MAC is shown in Figure 18. These clock domains and any special requirements regarding them are discussed in the subsequent sections. Control signals crossing a clock domain are synchronized to the destination clock domain.

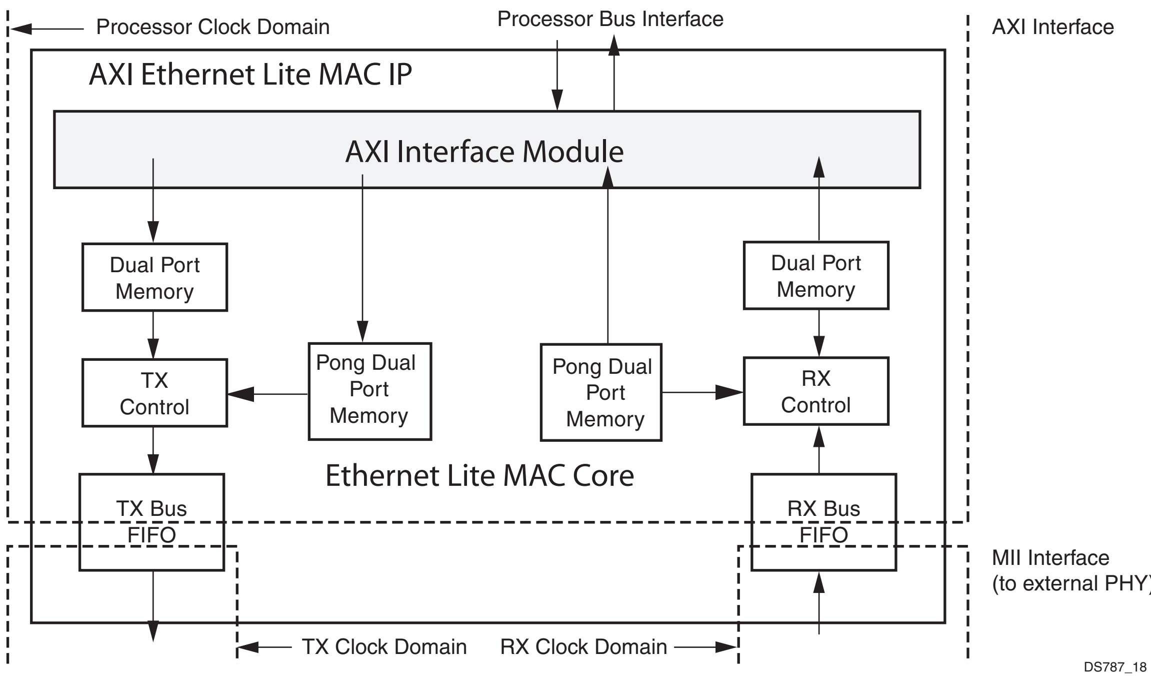


Figure 18: **Ethernet Lite MAC Clock Domain Diagram**

Transmit Clock

The transmit clock [PHY_tx_clk] is generated by the external PHY and must be used by the Ethernet Lite MAC core to provide transmit data [PHY_tx_data (3:0)] and to control signals [PHY_tx_en] to the PHY.

The PHY provides one clock cycle for each nibble of data transferred resulting in a 2.5 MHz clock for 10BASE-T operation and 25 MHz for 100BASE-T operation at +/- 100 ppm with a duty cycle of between 35% and 65%, inclusive. The PHY derives this clock from an external oscillator or crystal.

Receive Clock

The receive clock [PHY_rx_clk] is also generated by the external PHY but is derived from the incoming Ethernet traffic. Similarly to the transmit clock, the PHY provides one clock cycle for each nibble of data transferred, resulting in a 2.5 MHz clock for 10BASE-T operation and 25 MHz for 100BASE-T operation with a duty cycle of between 35% and 65%, inclusive, while incoming data is valid [PHY_dv is '1'].

The minimum high and low times of the receive clock are at least 35% of the nominal period under all conditions. The receive clock is used by the Ethernet Lite MAC core to sample the receive data [PHY_rx_data(3:0)] and control signals [PHY_dv and PHY_rx_er] from the PHY.

Processor Bus Clock

The majority of the Ethernet Lite MAC operation functions in the processor bus clock domain. This clock must be greater than or equal to 100 MHz to transmit and receive Ethernet data at 100 Mb/s and greater than or equal to 10 MHz to transmit and receive Ethernet data at 10 Mb/s.

PHY Interface Signals

PHY_rst_n

Many PHY devices require that they be held in reset for some period after the power-up sequence. The PHY_rst_n signal is an active low reset which is tied directly to the AXI reset signal (S_AXI_ARESETN). This output signal can be connected to the active low reset input of a PHY device.

PHY_tx_en

The Ethernet Lite MAC uses the Transmit Enable signal (PHY_tx_en) to indicate to the PHY that it is providing nibbles at the MII interface for transmission. It is asserted synchronously to PHY_tx_clk with the first nibble of the preamble and remains asserted while all nibbles are transmitted. PHY_tx_en is de-asserted prior to the first PHY_tx_clk following the final nibble of a frame.

This signal is transferred between the PHY_tx_clk and processor clock domains at the asynchronous TX bus FIFO interface. The clock to output delay of this signal must be 0 to 25 ns. Figure 19 shows PHY_tx_en timing during a transmission with no collisions.

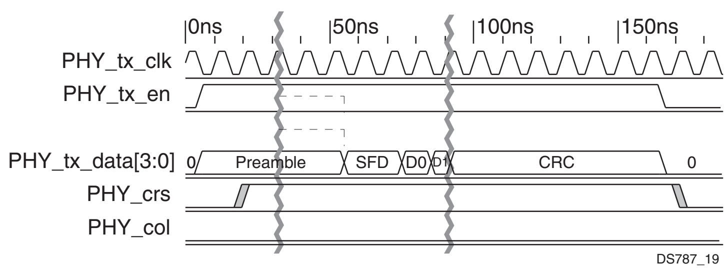


Figure 19: **Transmission with no Collision**

PHY_tx_data(3:0)

The Ethernet Lite MAC drives the Transmit Data bus PHY_tx_data(3:0) synchronously to PHY_tx_clk. PHY_tx_data(0) is the least significant bit. The PHY transmits the value of PHY_tx_data on every clock cycle that PHY_tx_en is asserted.

This bus is transferred between the PHY_tx_clk and processor clock domains at the asynchronous TX bus FIFO interface. The clock-to-output delay of this signal must be 0 to 25 ns. The order of the bits, nibbles, and bytes for transmit and receive are shown in Figure 20.

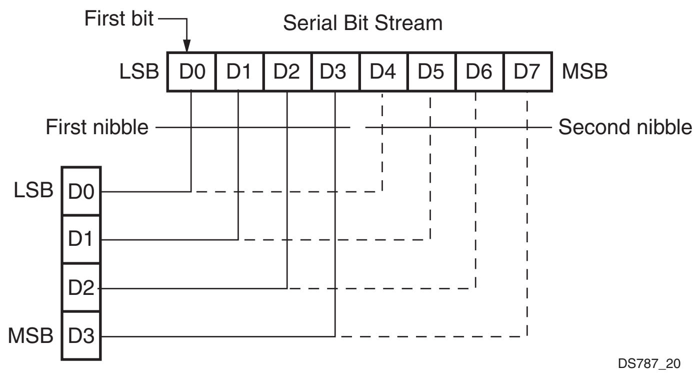


Figure 20: **Byte/Nibble Transmit and Receive Order**

PHY_dv

The PHY drives the Receive Data Valid (PHY_dv) signal to indicate that the PHY is driving recovered and decoded nibbles on the PHY_rx_data(3:0) bus and that the data on PHY_rx_data(3:0) is synchronous to PHY_rx_clk. PHY_dv is driven synchronously to PHY_rx_clk. PHY_dv remains asserted continuously from the first recovered nibble of the frame through the final recovered nibble and is negated prior to the first PHY_rx_clk that follows the final nibble.

For a received frame to be correctly received by the Ethernet Lite MAC, PHY_dv must encompass the frame, starting no later than the Start-of-Frame Delimiter (SFD) and excluding any End-of-Frame delimiter.

This signal is transferred between the PHY_rx_clk and processor clock domains at the asynchronous RX bus FIFO interface. The PHY provides a minimum of 10 ns setup and hold time for this signal in reference to PHY_rx_clk. Figure 21 shows the behavior of PHY_dv during frame reception.

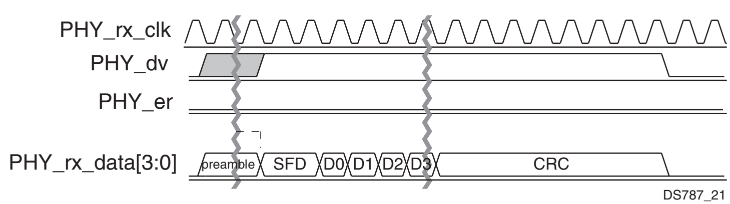


Figure 21: **Receive With No Errors**

PHY_rx_data(3:0)

The PHY drives the Receive Data bus PHY_rx_data(3:0) synchronously to PHY_rx_clk. PHY_rx_data(3:0) contains recovered data for each PHY_rx_clk period in which PHY_dv is asserted. PHY_rx_data(0) is the least significant bit. The Ethernet Lite MAC must not be affected by PHY_rx_data(3:0) while PHY_dv is de-asserted.

The Ethernet Lite MAC should ignore a special condition that occurs while PHY_dv is de-asserted; the PHY can provide a False Carrier indication by asserting the PHY_rx_er signal while driving the value 1110 onto PHY_rx_data(3:0). This bus is transferred between the PHY_rx_clk and processor clock domains at the asynchronous RX bus FIFO interface. The PHY provides a minimum of 10 ns setup and hold time for this signal in reference to PHY_rx_clk.

PHY_rx_er

The PHY drives the Receive Error signal (PHY_rx_er) synchronously to PHY_rx_clk. The PHY drives PHY_rx_er for one or more PHY_rx_clk periods to indicate that an error (such as a coding error or any error that the PHY is capable of detecting) was detected somewhere in the frame presently being transferred from the PHY to the Ethernet Lite MAC.

PHY_rx_er should have no effect on the Ethernet Lite MAC while PHY_dv is de-asserted. This signal is transferred between the PHY_rx_clk and processor clock domains at the asynchronous RX bus FIFO interface.

The PHY provides a minimum of 10 ns setup and hold time for this signal in reference to PHY_rx_clk. Figure 22 shows the behavior of PHY_rx_er during frame reception with errors.

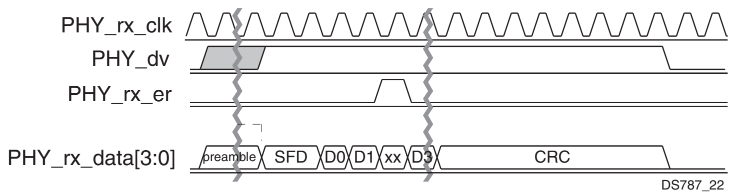


Figure 22: **Receive With Errors**

Table 16 shows the possible combinations for the receive signals.

Table 16: **Possible Values for PHY_dv, PHY_rx_er, and PHY_rx_data[3:0]**

PHY_dv	PHY_rx_er	PHY_rx_data[3:0]	Indication
0	0	0000 through 1111	Normal inter-frame
0	1	0000	Normal inter-frame
0	1	0001 through 1101	Reserved
0	1	1110	False carrier indication
0	1	1111	Reserved
1	0	0000 through 1111	Normal data reception
1	1	0000 through 1111	Data reception with errors

PHY_crs

The PHY drives the Carrier Sense signal (PHY_crs) active to indicate that either the transmit or receive is non-idle when operating in half duplex mode. PHY_crs is de-asserted when both the transmit and receive are idle.

The PHY asserts PHY_crs for the duration of a collision condition. PHY_crs is not synchronous to either the PHY_tx_clk or the PHY_rx_clk. The PHY_crs signal is not used in full duplex mode. The PHY_crs signal is used by both the Ethernet Lite MAC transmit and receive circuitry and is double-synchronized to the processor clock as it enters the Ethernet Lite MAC core.

PHY_col

The PHY asserts the Collision detected signal (PHY_col) to indicate the detection of a collision on the bus. The PHY asserts PHY_crs while the collision condition persists. The PHY also drives PHY_col asserted when operating at 10 Mb/s for signal_quality_error (SQE) testing.

PHY_col is not synchronous to either the PHY_tx_clk or the PHY_rx_clk. The PHY_col signal is not used in full duplex mode. The PHY_col signal is used by both the Ethernet Lite MAC core transmit and receive circuitry and is double-synchronized to the processor clock as it enters the Ethernet Lite MAC. Figure 23 shows the behavior of PHY_col during frame transmission with a collision.

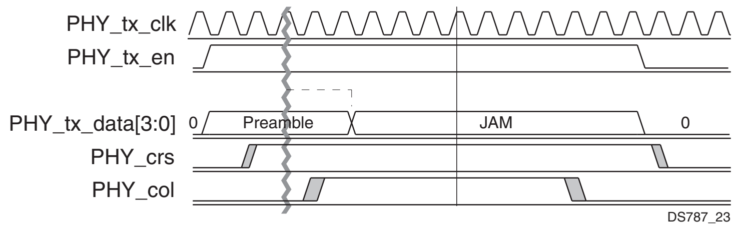


Figure 23: **Transmission With Collision**

Receive Address Validation

Destination addresses are classified as either unicast (a single station address indicated by the I/G bit = '0'), multicast (a group of stations indicated by the I/G bit = '1'), or a multicast subgroup broadcast (all stations on the network). The Ethernet Lite MAC accepts messages addressed to its unicast address and the broadcast address.

Design Constraints

The Ethernet Lite MAC core is designed to not use global buffers for the TX and RX clocks in the default configuration. Therefore, the Ethernet Lite MAC core requires design constraints (Figure 24) to guarantee performance. If the global clock buffers are used for TX/RX clocks, MAXSKEW constraints are not required. These constraints should be placed in a UCF for the top level of the design. The example in Figure 24 is for a 25 MHz PHY clock. The NET names are based on the port names of the Ethernet Lite MAC core. If these ports have different top-level port names, these NET names should be modified accordingly. The listed constraints are included automatically in the design if C_INCLUDE_PHY_CONSTRAINTS is set to '1' and need not be added to the UCF.

```
Common Constraints for all FPGA devices
NET "phy_rx_clk" PERIOD = 40 ns HIGH 14 ns;
NET "phy_tx_clk" PERIOD = 40 ns HIGH 14 ns;
OFFSET = OUT 10 ns AFTER "phy_tx_clk" ;
OFFSET = IN 6 ns BEFORE "phy_rx_clk" ;
NET "phy_rx_data<3>" IOBDELAY = NONE;
NET "phy_rx_data<2>" IOBDELAY = NONE;
NET "phy_rx_data<1>" IOBDELAY = NONE;
NET "phy_rx_data<0>" IOBDELAY = NONE;
NET "phy_dv" IOBDELAY = NONE;
NET "phy_rx_er" IOBDELAY = NONE;
NET "phy_crs" IOBDELAY = NONE;
NET "phy_col" IOBDELAY = NONE;
NET "phy_tx_clk" MAXSKEW = 6.0 ns;
NET "phy_rx_clk" MAXSKEW = 6.0 ns;
```

Figure 24: **Design Constraints**

Design Implementation

Target Technology

The intended target technology is the Virtex®-6 and Spartan-6 family FPGAs.

Device Utilization and Performance Benchmarks

Core Performance

Because the Ethernet Lite MAC is a module that is used with other design pieces in the FPGA, the resource utilization and timing numbers reported in this section are estimates only. When the Ethernet Lite MAC is combined with other pieces of the FPGA design, the utilization of FPGA resources and timing of the design vary from the results reported here.

The Ethernet Lite MAC resource utilization benchmarks for a variety of parameter combinations measured with Virtex6 FPGA as the target device are shown in Table 17.

Table 17: **Performance and Resource Utilization Benchmarks on the Virtex-6 FPGA (xc6vlx130t-ff1156-1)**

Parameter Values (other parameters at default value)					Device Resources				Performance
C_DUPLEX	C_RX_PING_PONG	C_TX_PING_ PONG	C_INCLUDE_MDIO	C_S_AXI_PROTOCOL	Slices	Slice Flip-Flops	LUTs	Block RAMS	F_{MAX} (MHz)
0	0	0	0	AXI4LITE	222	493	537	2	200
1	0	0	0	AXI4LITE	191	435	470	2	200
1	1	1	0	AXI4LITE	212	460	536	4	200
1	1	1	0	AXI4	215	472	586	4	200
1	0	0	0	AXI4	206	444	507	2	200
1	1	1	1	AXI4LITE	250	546	629	4	200
1	1	1	1	AXI4	295	555	673	4	200
1	0	0	1	AXI4LITE	235	520	575	2	200
1	0	0	1	AXI4	269	529	612	2	200

The Ethernet Lite MAC resource utilization for various parameter combinations measured with Spartan-6 as the target device are detailed in Table 18.

Table 18: **Performance and Resource Utilization Benchmarks on the Spartan-6 FPGA (xc6slx75t-fgg676-2)**

Parameter Values (other parameters at default value)					Device Resources				Performance
C_DUPLEX	C_RX_PING_PONG	C_TX_PING_ PONG	C_INCLUDE_MDIO	C_S_AXI_PROTOCOL	Slices	Slice Flip-Flops	LUTs	Block RAMS	F_{MAX} (MHz)
0	0	0	0	AXI4LITE	205	434	466	2	133
1	0	0	0	AXI4LITE	205	434	466	2	133
1	1	1	0	AXI4LITE	243	549	538	4	133
1	1	1	0	AXI4	243	469	568	4	133
1	0	0	0	AXI4	212	443	502	2	133
1	1	1	1	AXI4LITE	305	548	629	4	133
1	1	1	1	AXI4	329	554	671	4	133
1	0	0	1	AXI4LITE	266	519	572	2	133
1	0	0	1	AXI4	291	529	610	2	133

System Performance

To measure the system performance (F_{MAX}) of this core, this core was added to a Virtex-6 FPGA system and a Spartan-6 FPGA system as the device under test (DUT) as illustrated in Figure 25 and Figure 26.

Because the Ethernet Lite MAC core is used with other design modules in the FPGA, the utilization and timing numbers reported in this section are estimates only. When this core is combined with other designs in the system, the design's FPGA resources and timing usage will vary from the results reported here.

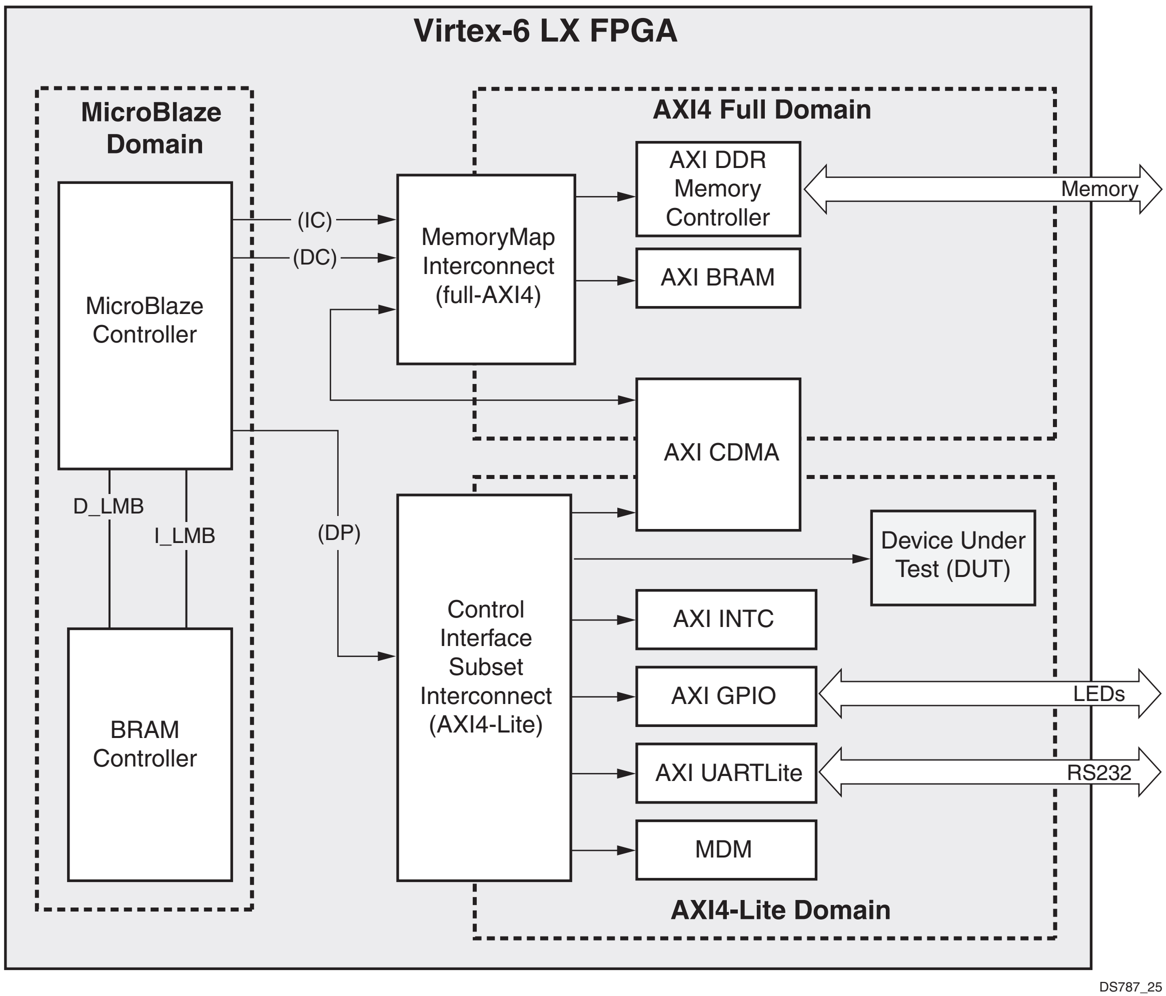


Figure 25: **Virtex-6 FPGA System with the Ethernet Lite MAC as the DUT**

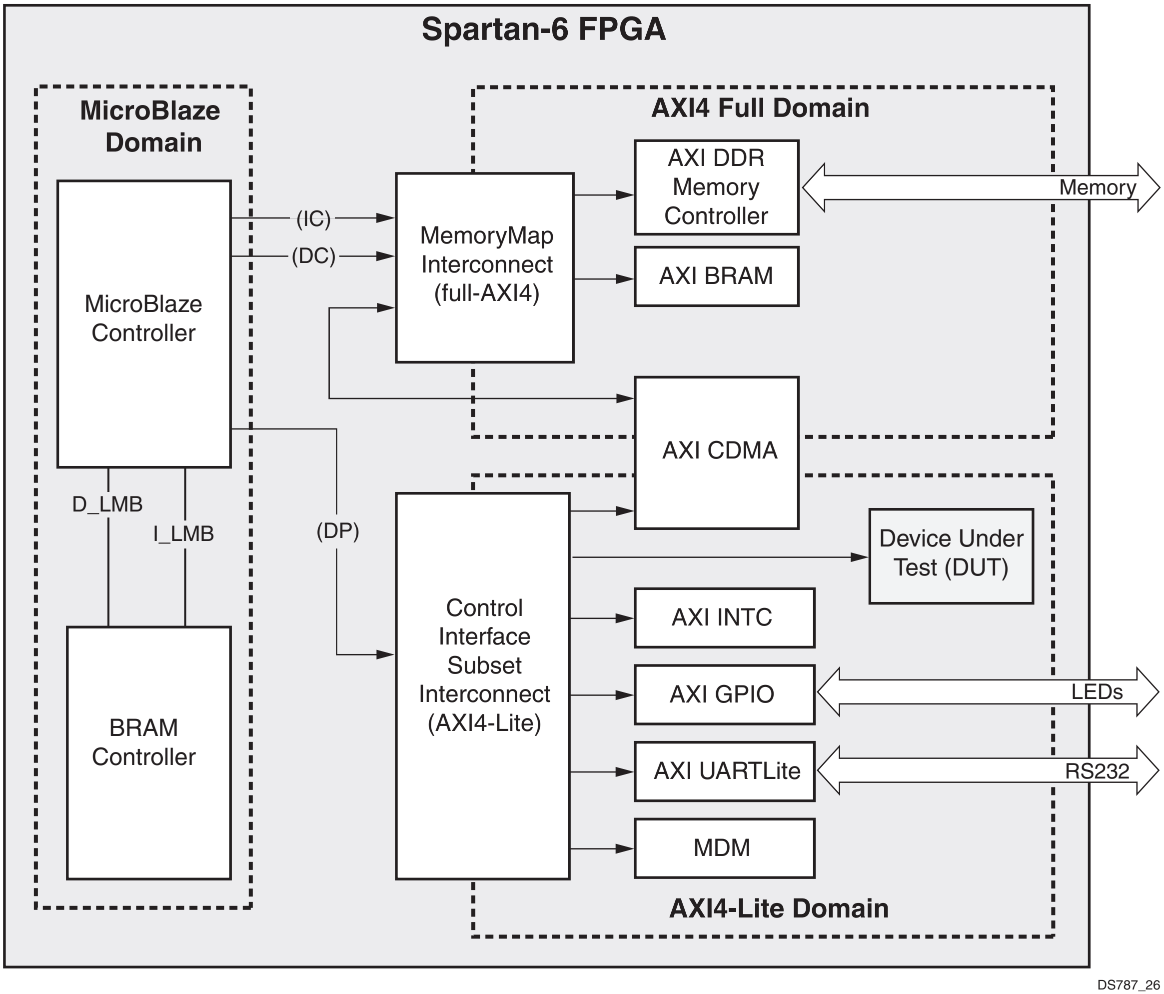


Figure 26: **Spartan-6 FPGA System with the Ethernet Lite MAC as the DUT**

The target FPGA was filled with logic to drive the LUT and block RAM utilization to approximately 70% and the I/O utilization to approximately 80%. Using the default tool options and the slowest speed grade for the target FPGA, the resulting target F_{MAX} numbers are shown in Table 19.

Table 19: **System Performance**

Target FPGA	Target F_{MAX} (MHz)		
	AXI4	AXI4-Lite	MicroBlaze
xc6slx45t [1]	90 MHz	120 MHz	80
xc6vlx240t [2]	135 MHz	180 MHz	135

Notes:
1. Spartan-6 LUT utilization: 60%; Block RAM utilization: 70%; I/O utilization: 80%; MicroBlaze™ not AXI4 interconnect; AXI4 interconnect configured with a single clock of 120 MHz.
2. Virtex-6 LUT utilization: 70%; Block RAM utilization: 70%; I/O utilization: 80%.

The target F_{MAX} is influenced by the exact system and is provided for guidance. It is not a guaranteed value across all systems.

Unsupported Features/Limitations

- AXI data bus width greater than 32 bits
- AXI address bus width other than 32 bits
- AXI Exclusive Accesses
- AXI Trustzone
- AXI Low-Power interface
- AXI Narrow transfers
- AXI FIXED, WRAP transactions
- AXI Barrier transactions
- AXI Debug transactions
- AXI user signals

Reference Documents

1. Virtex-6 Family Overview (DS150)
2. Spartan-6 Family Overview (DS160)
3. AXI4 AMBA AXI Protocol Version: 2.0 Specification
4. LogiCORE AXI Interconnect IP (DS768)
5. IEEE Std. 802.3 Media Independent Interface Specification

Support

Xilinx provides technical support for this LogiCORE product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY*.

Ordering Information

This Xilinx LogiCORE IP module is provided at no additional cost with the Xilinx® ISE® Design Suite Embedded Edition software under the terms of the Xilinx End User License. The core is generated using the Xilinx ISE Embedded Edition software (EDK).

Information about this and other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information on pricing and availability of other Xilinx LogiCORE modules and software, contact your local Xilinx sales representative.

Revision History

Date	Version	Description of Revisions
9/21/10	1.0	Initial Xilinx release
6/22/11	1.1	Updated for 13.2.
10/19/11	1.2	Updated for ISE Software Release 13.3. Added device support for Zynq-7000.

Notice of Disclaimer