OPB Ethernet Media Access Controller (EMAC) (v1.00m)

DS435 (v1.23.1) October 8, 2003

Product Overview

Introduction

This document provides the design specification for the 10/100 Mbs Ethernet Media Access Controller (EMAC). The EMAC incorporates the applicable features described in IEEE Std. 802.3 MII interface specification. The IEEE Std. 802.3 MII interface specification is referenced throughout this document and should be used as the authoritative specification. Differences between IEEE Std. 802.3 MII interface specification and the Xilinx EMAC implementation are highlighted and explained in Specification Exceptions.

The EMAC Interface design is a soft intellectual property (IP) core designed for implementation in a Virtex-E, Virtex-II, Spartan-II, Spartan-IIE, or Virtex-II Pro FPGA. It supports the IEEE Std. 802.3 Media Independent Interface (MII) to industry standard Physical Layer (PHY) devices and communicates to a processor via an IBM On-Chip Peripheral Bus (OPB) interface. The design provides a 10 Megabits per second (Mbps) and 100 Mbps (also known as Fast Ethernet) EMAC Interface. This design includes many of the functions and the flexibility found in dedicated Ethernet controller devices currently on the market.

The Xilinx EMAC design allows the customer to tailor the EMAC to suit their application by setting certain parameters to enable/disable features. The parameterizable features of the design are discussed in EMAC Design Parameters.

The EMAC is comprised of two IP blocks as shown in Figure 1: The IP Interface (IPIF) block is a subset of OPB bus interface features chosen from the full set of IPIF features to most efficiently couple the second block, the EMAC core, to the OPB processor bus for this packet[1] based interface (this combined entity is referred to as a device). Although there are separate specifications for the IPIF design, this specification addresses the specific implementation required for the EMAC design.

LogiCORE™ Facts		
Core Specifics		
Supported Device Family	Virtex™II Pro, Virtex™II, Virtex™, Virtex™E, Spartan™II, Spartan™IIE, Spartan™III	
Version of Core	opb_ethernet	v1.00m
Resources Used		
	Min	Max
I/O	179	179
LUTs	2018	3688
FFs	1557	2228
Block RAMs	2	16
Provided with Core		
Documentation	Product Specification	
Design File Formats	VHDL	
Constraints File	N/A	
Verification	N/A	
Instantiation Template	N/A	
Reference Designs	None	
Design Tool Requirements		
Xilinx Implementation Tools	5.1i or later	
Verification	N/A	
Simulation	ModelSim SE/EE 5.6e or later	
Synthesis	XST	
Support		
Support provided by Xilinx, Inc.		

1. IEEE Std. 802.3 uses the terms Frame and Packet interchangeably when referring to the Ethernet unit of transmission; this specification does likewise

EMAC Endianess

Please note that the EMAC is designed as a big endian device (bit 0 is the most significant bit and is shown on the left of a group of bits).The 4-bit transmit and receive data interface to the external PHY is little endian (bit 3 is the most significant bit and appears on the left of the bus). The MII management interface to the PHY is serial with the most significant bit of a field being transmitted first.

Features

The EMAC is a soft IP core designed for Xilinx FPGAs and contains the following features:

- 32-bit OPB master and slave interfaces
- Memory mapped direct I/O interface to registers and FIFOs as well as simple DMA and Scatter/Gather DMA capabilities for low processor and bus utilization
- Media Independent Interface (MII) for connection to external 10/100 Mbps PHY transceivers
  - IEEE 802.3-compliant MII
  - Supports auto-negotiable and non auto-negotiable PHYs
  - Supports 10BASE-T and 100BASE-TX/FX IEEE 802.3 compliant MII PHYs at full or half duplex
- Independent internal 2K, 4K, 8K, 16K, or 32K byte TX and RX FIFOs for holding data for more than one packet. 2K byte depth is sufficient for normal 1518 maximum byte packets but 4K byte depth provides better throughput.
- 16, 32, or 64 entry deep FIFOs for the Transmit Length, Receive Length, and Transmit Status registers to support multiple packet operation.
- CSMA/CD compliant operation at 10 Mbps and 100 Mbps in half duplex mode
- Programmable PHY reset signal
- Internal loop-back capability
- Supports unicast, multicast, and broadcast transmit and receive modes as well as promiscuous address receive mode
- Supports a "Freeze" (graceful halt) mode based on input signal assertion to assist with emulator based software development
- Provides auto or manual source address field insertion or overwrite for transmission
- Provides auto or manual pad and Frame Check Sequence (FCS) field insertion
- Provides auto pad and FCS field stripping on receive
- Processes received pause packets
- Supports reception of longer VLAN type frames
- Supports MII management control writes and reads with MII PHYs
- Programmable interframe gap
- Provides counters and interrupts for many error conditions

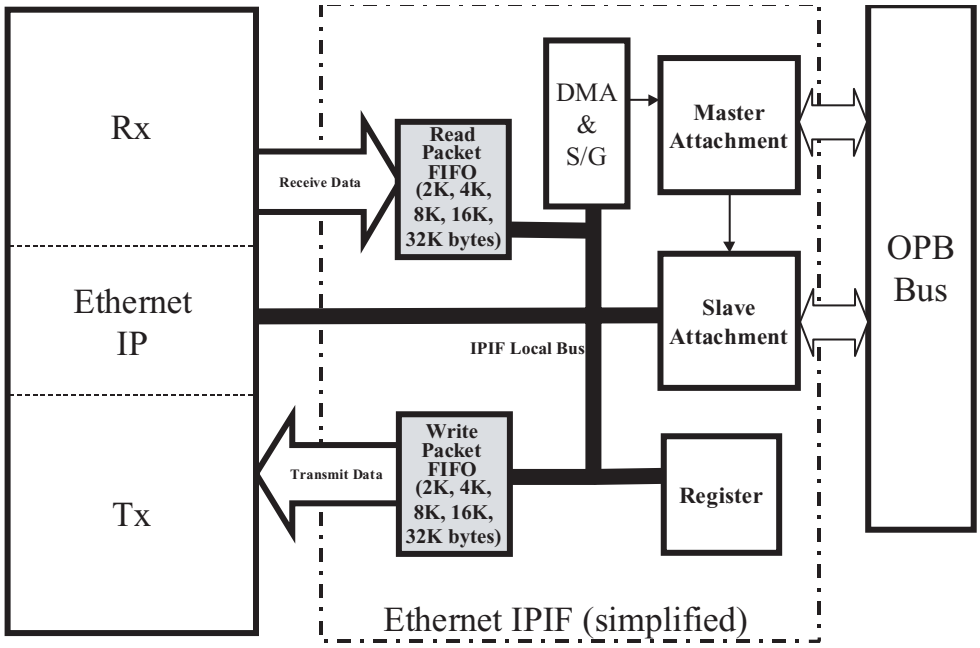

Figure 1: **IPIF and EMAC Modules**

Ethernet Protocol

Ethernet data is encapsulated in frames as shown in Figure 2 for standard Ethernet and Figure 3 for VLANEthernet[1]. The fields in the frame are transmitted from left to right. The bits within the frame are transmitted from left to right (from least significant bit to most significant bit unless specified otherwise).

Preamble

The preamble field is used for synchronization and must contain seven bytes with the pattern 10101010. The pattern is transmitted from left to right. If a collision is detected during the transmission of the preamble or start of frame delimiter fields, the transmission of both fields will be completed. For transmission, this field is always automatically inserted by the EMAC and should never appear in the packet data provided to the EMAC. For reception, this field is always stripped from the packet data.

Start Frame Delimiter

The start frame delimiter field marks the start of the frame and must contain the pattern 10101011. The pattern is transmitted from left to right. If a collision is detected during the transmission of the preamble or start of frame delimiter fields, the transmission of both fields will be completed. The receive data valid signal from the PHY (RX_DV) may go active during the preamble but will be active prior to the start frame delimiter field. For transmission, this field is always automatically inserted by the EMAC and should never appear in the packet data provided to the EMAC. For reception, this field is always stripped from the packet data.

1. The EMAC design does not support the Ethernet 8-byte preamble frame type

Destination Address

The destination address field is 6 bytes in length[1]. 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 EMAC supports transmission and reception of unicast, multicast, and broadcast packets.

Bits in the EMAC control register can be used to independently enable reception of unicast (destination address matches the station address in Station Address High (SAH) and Station Address Low (SAL) registers), multicast, and broadcast frames. An additional bit in the control register can be used to enable promiscuous mode which accepts all frames regardless of destination address. This field is transmitted with the least significant bit first. This field is always provided in the packet data for transmissions and is always retained in the receive packet data.

Source Address

The source address field is 6 bytes in length[2]. This field is transmitted with the least significant bit first. For transmission, this field may be inserted automatically by the EMAC with information provided in the SAH and SAL registers or may be supplied as part of the packet data provided to the EMAC as indicated by a bit in the EMAC control register.

When the source address is provided automatically by the EMAC, a bit in the EMAC control register determines if the data in the SAH and SAL registers is inserted into the packet data in the transmit packet FIFO (i.e., not source address field exists in the transmit packet FIFO data) or if it overwrites a source address field provided in the transmit packet FIFO. This field is always retained in the receive packet data.

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 following data field. This value does not include any bytes that may 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 (05DC hex) would indicates that a frame type rather than a length value is provided in this field. The IEEE 802.3 standard uses the value 1536 (0600 hex) or greater to signal a type field and that is what is used in the EMAC design.

For reception, if the field is a length field, the EMAC will compare the length against the actual data field length and will flag an error if they are different. If the field is a type field, the EMAC will ignore the value and pass it along with the packet data with no further processing unless the value is 8100 hex which indicates that the frame is a VLAN frame or 8808 hex which indicates a pause MAC control frame (refer to Carrier sense multiple access with collision detection (CSMA/CD) access method).

If the frame is a VLAN type frame, the EMAC must accept 4 additional bytes which are provided with the received packet data. No additional processing is performed by the EMAC other than to process the additional bytes.

The EMAC does not perform any processing of the type/length field on transmissions. The data provided in the transmit packet is transmitted without any interpretation or validation.

1. The EMAC design does not support 16-bit destination addresses as defined in the IEEE 802 standard
2. The EMAC design does not support 16-bit source addresses as defined in the IEEE 802 standard

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 may vary from 0 to 1500 bytes in length. This field is transmitted with the least significant bit first. This field is always provided in the packet data for transmissions and is always retained in the receive packet data.

Pad

The pad field may vary from 0 to 46 bytes in length. This field is used to insure 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 will be 46 bytes. If the data field is 46 bytes or more, the pad field will have 0 bytes.

For transmission, this field may be inserted automatically by the EMAC or may be supplied as part of the packet data provided to the EMAC as indicated by a bit in the EMAC control register[1].

If EMAC insertion of padding is enabled in the EMAC control register, the number of pad bytes to be inserted will determined by the transmit data length register and the FCS and Source address insertion enable bits in the EMAC control register resulting in the following formula:

PAD (bytes) = 64 - [TXLengthReg + (ENFCS * 4) + (ENSA * 6)].

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 as[2]:

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

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 (i.e., the bits of the CRC are transmitted in the order x^{31}, x^{30},..., x^1, x^0). The EMAC 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 may be inserted automatically by the EMAC or may be supplied as part of the packet data provided to the EMAC as indicated by a bit in the EMAC control register.

1. If the pad field is inserted by the EMAC, the FCS field will also be calculated and inserted by the EMAC. This is necessary to insure proper FCS calculation over the pad field. If the pad field is supplied as part of the transmit packet, the FCS may be inserted by the EMAC or provided as part of the packet to the EMAC.
2. Reference IEEE Std. 802.3 para. 3.2.8

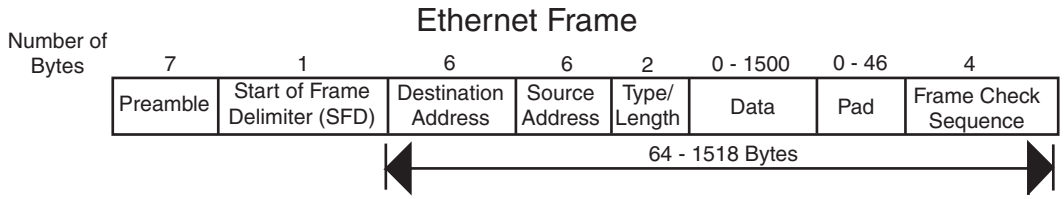

Figure 2: **Ethernet Data Format**

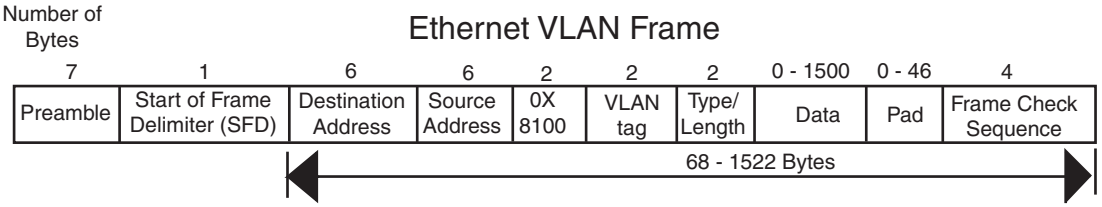

Figure 3: **Ethernet VLAN Data Format**

Interframe Gap[1] and Deferring

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). This is a minimum value and may be increased with a resulting decrease in throughput (results in a less aggressive approach to gaining access to a shared Ethernet bus). 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 EMAC monitors the bus for traffic by watching the carrier sense signal (CRS) from the external PHY. Whenever the bus is busy (CRS ='1'), the EMAC 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 EMAC starts the timing of the interframe gap.
3. The EMAC will reset the interframe gap timer if carrier sense becomes true during the period defined by the "interframe gap part 1 (IFG1)" field of the IFGP register. The IEEE std. 802.3 states that this should be the first 2/3 of the interframe gap timing interval (64 bit times) but may be shorter and as small as zero. The purpose of this option is to support a possible brief failure of the carrier sense signal during a collision condition and is described in paragraph 4.2.3.2.1 of the IEEE standard.
4. The EMAC will not reset the interframe gap timer if carrier sense becomes true during the period defined by the "interframe gap part 2 (IFG2)" field of the IFGP register to ensure fair access to the bus. The IEEE std. 802.3 states that this should be the last 1/3 of the interframe gap timing interval (32 bit times) but may be longer and as large as the whole interframe gap time.

Full-Duplex

1. The EMAC does not use the carrier sense signal from the external PHY when in full duplex mode since the bus is not shared and only needs to monitor its own transmissions. After the last bit of an EMAC transmission, the EMAC starts the interframe gap timer and defers transmissions until it has reached the value represented by the combination of the IFG1 and IFG2 fields of the IFGP register.

1. Interframe Gap and interframe spacing are used interchangeably and are equivalent.

Carrier sense multiple access with collision detection (CSMA/CD) access 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 (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 (COL -'1'), then each transmitting station intentionally continues to transmit (jam) for an additional predefined period (32 bit times for 10/100 Mbs) to ensure propagation of the collision throughout the system.

The station remains silent for a random amount of time (backoff) 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. Once the collision window has passed, a transmitting station has acquired the bus. Subsequent collisions (late collisions) are avoided since 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 Mbs). In order to minimize processor bus transactions, the EMAC design operating in half duplex mode will retain the first 64 bytes of a transmission until the collision window has successfully passed. If a collision does occur in the collision window, the EMAC will retry the transmission without the need to re-acquire the packet data over the processor bus. This is accomplished by using special FIFOs in the IPIF interface.

Transmit Flow

The flow chart in Figure 4 shows the high level flow followed for packet transmission.

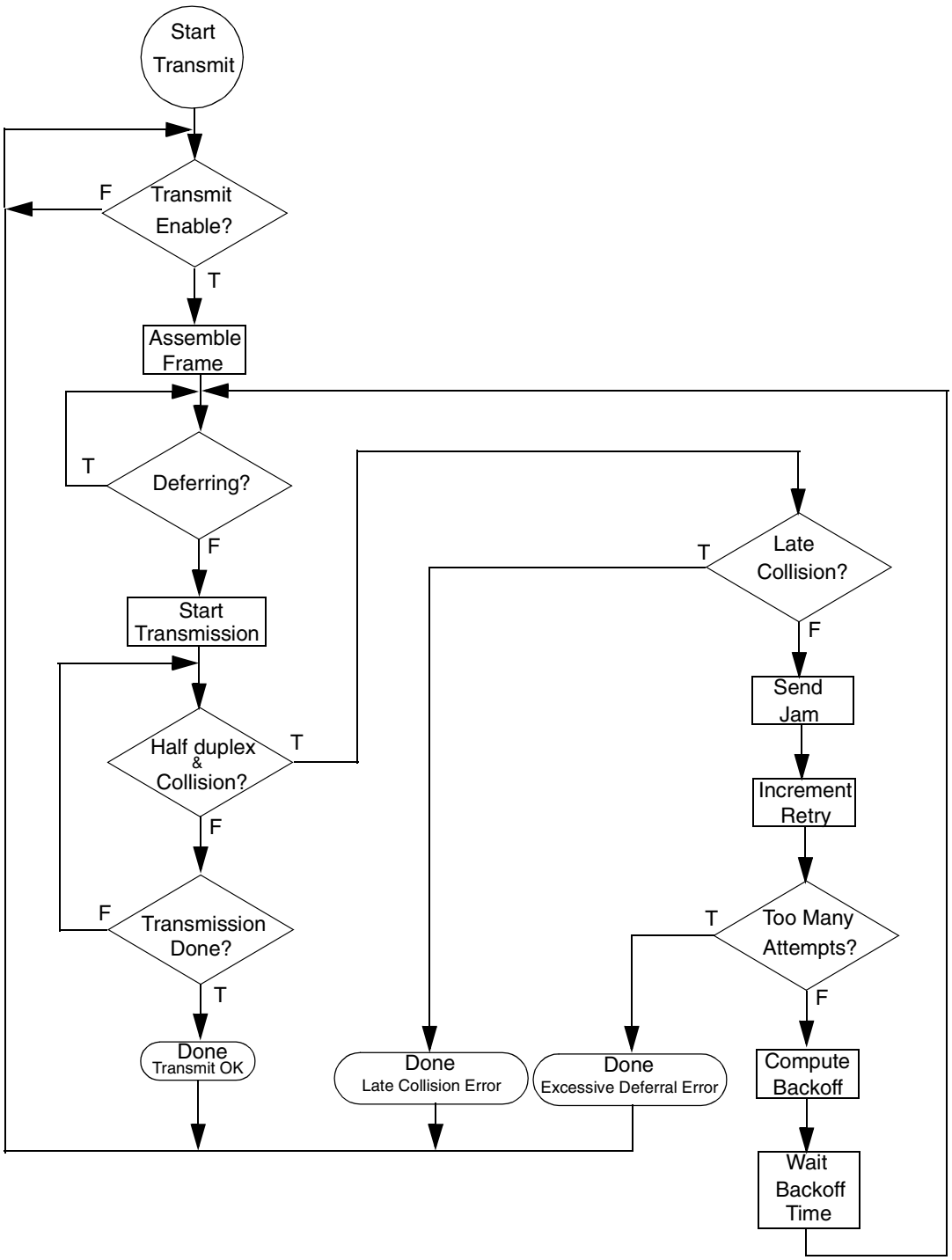

Figure 4: **Transmit Flow**

Receive Flow

The flow chart in Figure 5 shows the high level flow followed for packet reception.

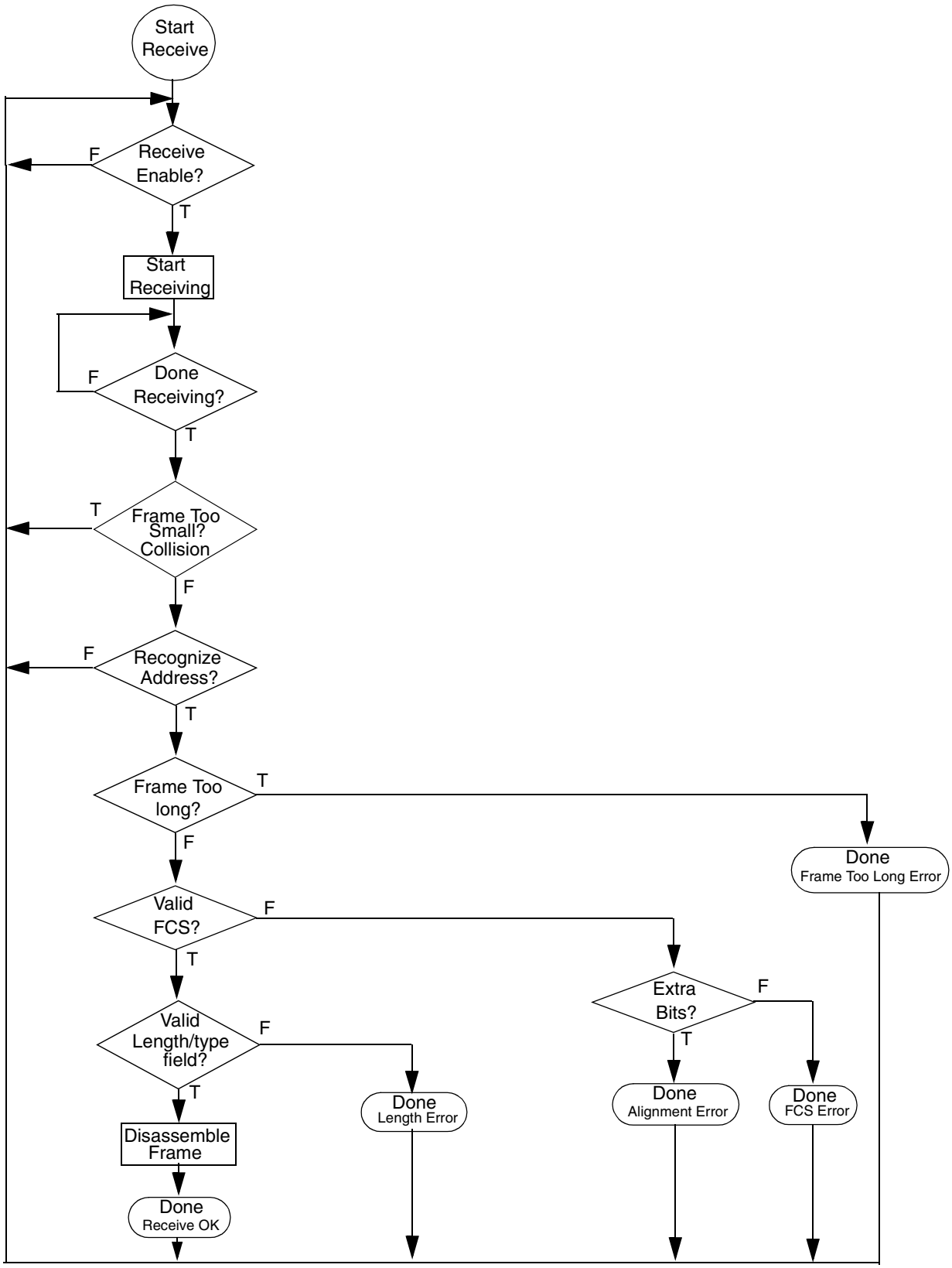

Figure 5: **Receive Flow**

EMAC Design Parameters

To allow the user to generate an EMAC that is tailored for their system, certain features are parameterizable in the EMAC design. This allows the user to have a design that only utilizes the resources required by their system and runs at the best possible performance. The features that are parameterizable in the Xilinx EMAC design are shown in Table 1.

Table 1: **EMAC Design Parameters**

Grouping / Number		Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
Top Level	G1	Device Block Id	C__DEV_BLK_ID		0	integer
	G2	BUS clock period in pS	C__OPB_CLK_PERIOD_PS	Requirements as stated in note 1	10000	integer
	G3	Device family	C_FAMILY	virtex, virtexe, spartan2, spartan2e, virtex2, virtex2p	virtex2p	string
	G4	IPIF Packet FIFO depth in bits	C_IPIF_FIFO_DEPTH	262144, 131072, 65536, 32768 or 16384	32768	integer
	G5	Device base address	C_BASEADDR	See Note 2	None	std logic vector
	G6	Device maximum address	C_HIGHADDR	See Note 2	None	std logic vector
	G7	MAC length and status FIFO depth	C_MAC_FIFO_DEPTH	16, 32, 64	32	integer
OPB/IPIF Interface	G14	Module Identification Read	C_DEV_MIR_ENABLE	1 = MIR reads Exists 0 = MIR reads Non-existent	1	integer
	G15	Software Reset Function	C_RESET_PRESENT	1 = software reset Exists 0 = software reset Non-existent	1	integer
	G16	Interrupt device ID encoder	C_INCLUDE_DEV_PENCODER	1 = interrupt device ID encoder Exists 0 = interrupt device ID encoder Non-existent	1	integer
	G17	DMA Present	C_DMA_PRESENT	1 = no DMA function is required 2 = simple 2 ch DMA is required 3 = Scatter Gather DMA for packets is required	1	integer

Notes:

1. The OPB BUS clock frequency must be greater than or equal to 65 MHz for 100 Mbs Ethernet operation and greater than or equal to 6.5 Mhz for 10 Mbs Ethernet operation.
2. No default value will be specified for values to insure that the actual value is set, i.e if the value is not set, a compiler error will be generated. The address range must be at least 3FFF.

Table 1: **EMAC Design Parameters *(Continued)***

Grouping / Number		Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
	G18	DMA interrupt coalescing functionality	C_DMA_INTR_COALESCE	1 = DMA interrupt coalescing Exists 0 = DMA interrupt coalescing Non-existent	1	integer
	G19	OPB address bus width (in bits)	C_OPB_AWIDTH		32	integer
	G20	OPB data bus width (in bits)	C_OPB_DWIDTH		32	integer

Notes:
1. The OPB BUS clock frequency must be greater than or equal to 65 MHz for 100 Mbs Ethernet operation and greater than or equal to 6.5 Mhz for 10 Mbs Ethernet operation.
2. No default value will be specified for values to insure that the actual value is set, i.e if the value is not set, a compiler error will be generated. The address range must be at least 3FFF.

Allowable Parameter Combinations

The EMAC is a synchronous design. Due to the state machine control architecture of receive and transmit operations, the OPB Clock must be greater than or equal to 65 MHz to allow Ethernet operation at 100 Mbs and greater than or equal to 6.5 Mhz for Ethernet operation at 10 Mbs.

EMAC I/O Signals

The external I/O signals for the EMAC are listed in Table 2.

Table 2: **EMAC I/O Signals**

Grouping		Signal Name	Interface	I/O	Initial State	Description
EMAC Signals	P1	PHY_rx_data(3:0)	System	I		Ethernet receive data. Input from I/O block registers
	P2	PHY_tx_data(3:0)	System	O	0000	Ethernet transmit data. Output to I/O block registers
	P3	PHY_dv	System	I		Ethernet receive data valid. Input from I/O block register
	P4	PHY_rx_er	System	I		Ethernet receive error. Input from I/O block register
	P5	PHY_tx_en	System	O	0	Ethernet transmit enable. Output to I/O block register

Table 2: **EMAC I/O Signals *(Continued)***

Grouping		Signal Name	Interface	I/O	Initial State	Description
	P6	PHY_rx_en	System	0	0	Ethernet receive enable controlled byu control register bit 4
	P7	PHY_tx_er	System	O	0	Ethernet transmit error. Output to I/O block register
	P8	PHY_tx_clk	System	I		Ethernet transmit clock input from input buffer
	P9	PHY_rx_clk	System	I		Ethernet receive clock input from input buffer
	P10	PHY_crs	System	I		Ethernet carrier sense input from input buffer
	P11	PHY_col	System	I		Ethernet collision input from input buffer
	P12	PHY_rst_n	System	O	1	Ethernet PHY reset output to output buffer
	P13	PHY_mii_clk_O	System	O	0	MII management interface clock output to 3-state output buffer
	P14	PHY_mii_clk_T	System	O	0	MII management interface clock enable output to 3-state output buffer
	P15	PHY_mii_data_I	System	I		MII management interface data input from 3-state I/O buffer
	P16	PHY_mii_data_O	System	O	0	MII management interface data output to 3-state I/O buffer
	P17	PHY_mii_data_T	System	O	0	MII management interface data enable output to 3-state I/O buffer
OPB Signals	P18	Sln_DBus(0:C_OPB_DWIDTH-1)	IPIF	O	0	EMAC slave output data bus
	P19	Sln_xferAck	IPIF	O	0	EMAC slave transfer acknowledge

Table 2: **EMAC I/O Signals *(Continued)***

Grouping		Signal Name	Interface	I/O	Initial State	Description
	P20	Sln_Retry	IPIF	O	0	EMAC slave retry
	P21	Sln_ToutSup	IPIF	O	0	EMAC slave timeout suppress
	P22	Sln_ErrAck	IPIF	O	0	EMAC slave error acknowledge
	P23	OPB_ABus(0:C_OPB_AWIDTH-1)	IPIF	I		OPB address bus
	P24	OPB_BE(0:3)	IPIF	I		OPB byte enables
	P25	OPB_DBus(0:C_OPB_DWIDTH-1)	IPIF	I		OPB data bus
	P26	OPB_RNW	IPIF	I		Read not Write (OR of all master RNW signals)
	P27	OPB_select	IPIF	I		Master has taken control of the bus (OR of all master selects)
	P28	OPB_seqAddr	IPIF	I		OPB sequential address
	P29	OPB_errAck	IPIF	I		OPB error acknowledge
	P30	OPB_MnGrant	IPIF	I		OPB master bus grant
	P31	OPB_retry	IPIF	I		OPB retry
	P32	OPB_timeout	IPIF	I		OPB timeout error
	P33	OPB_xferAck	IPIF	I		OPB transfer acknowledge
	P34	Mn_request	IPIF	O	0	EMAC master bus request
	P35	Mn_busLock	IPIF	O	0	EMAC master bus arbitration lock
	P36	Mn_select	IPIF	O	0	EMAC master select
	P37	Mn_RNW	IPIF	O	0	EMAC master Read not Write
	P38	Mn_BE(0:3)	IPIF	O	0	EMAC master byte enables
	P39	Mn_seqAddr	IPIF	O	0	EMAC master sequential address
	P40	Mn_ABus(0:C_OPB_AWIDTH-1)	IPIF	O	0	EMAC master address bus

Table 2: **EMAC I/O Signals *(Continued)***

Grouping		Signal Name	Interface	I/O	Initial State	Description
System	P41	OPB_Clk	System	I		System clock
	P42	OPB_Rst	System	I		System Reset (active high)
	P43	IP2INTC_Irpt	System	O	0	System Interrupt
	P44	Freeze	System	I		System Freeze Input

EMAC Port Dependencies

The width of some of the EMAC signals depend on parameters selected in the design. The dependencies between the EMAC design parameters and I/O signals are shown in Table 3.

Table 3: **EMAC Parameter Port Dependencies**

		Name	Affects	Depends	Relationship Description
Design Parameters	G20	C_OPB_DWIDTH	P18, P25		Specifies the Data Bus width
	G19	C_OPB_AWIDTH	P23, P40		Specifies the Address Bus width
	G17	C_DMA_PRESENT	G18		Specifies if DMA is present and which type
	G18	C_DMA_INTR_COASLESCE		G17	Not used if scatter gather DMA not present (G17 is 0, 1, 2)
I/O Signals	P18	Sln_DBus(0:C_OPB_DWIDTH-1)		G20	Width varies with the size of the Data bus.
	P23	OPB_ABus(0:C_OPB_AWIDTH-1)		G19	Width varies with the size of the Address bus.
	P25	OPB_DBus(0:C_OPB_DWIDTH-1)		G20	Width varies with the size of the Data bus.
	P40	M_ABus(0:C_OPB_AWIDTH-1)		G19	Width varies with the size of the Address bus.

EMAC Interrupt Interface

The interrupt signals generated by the EMAC are managed by the Interrupt Source Controller in the EMAC IPIF module. This interface provides many of the features commonly provided for interrupt handling. Please refer to the OPB Device Interrupt Architecture specification listed in *Reference Documents*.

Interrupt (data bus bit 31) -- Transmit complete interrupt

Indicates that at least one transmit has completed and that the transmit status word is available.

Interrupt (data bus bit 30) -- Receive complete interrupt

Indicates that at least one successful receive has completed and that the receive status word packet data and packet data length is available. This signal is not set for unsuccessful receives.

Interrupt (data bus bit 29) -- Transmit error interrupt

Indicates that at least one failed transmit has completed and that the transmit status word is available. This active high signal is one bus clock in width.

Interrupt (data bus bit 28) -- Receive Error interrupt

Indicates that at least one failed receive has completed. No receive status word, packet data, or packet data length is available since it is not retained for failed receives.

Interrupt (data bus bit 27) -- Transmit Status FIFO Empty interrupt

This reflects the status of the transmit status FIFO empty flag. It may be used to indicate that the status words for all completed transmissions have been processed. Any other transmit packets already provided to the EMAC are either queued for transmit or are currently being transmitted but have not yet completed. This active high signal remains active as long as the condition persists.

Interrupt (data bus bit 26) --Receive Length FIFO Empty interrupt

This reflects the status of the receive length FIFO empty flag. It may be used to indicate that the packet lengths for all successfully completed receives have been processed. The status of this FIFO should always track the status of the receive status FIFO. This active high signal remains active as long as the condition persists.

Interrupt (data bus bit 25) -- Transmit Length FIFO Full interrupt

This reflects the status of the transmit length FIFO full flag. It may be used to pause queueing of transmit packets until some of the queued packets have been processed by the EMAC. This active high signal remains active as long as the condition persists.

Interrupt (data bus bit 24) -- Receive Length FIFO Overrun interrupt

Indicates that the receive length FIFO became full during the reception of a packet and data was lost. The EMAC will remove the corresponding packet from the receive data FIFO and no receive status will be stored but to insure that more data is not lost, receive packets stored in the FIFO should be processed to free up more locations. Once set, this bit can only be cleared with a reset.

Interrupt (data bus bit 23) -- Receive Length FIFO Underrun interrupt

Indicates that an attempt was made to read the receive length FIFO when it was empty and that the data received is not valid. Once set, this bit can only be cleared with a reset.

Interrupt (data bus bit 22) -- Transmit Status FIFO Overrun interrupt

Indicates that the Transmit status FIFO became full following the transmission of a packet and data was lost. Care must be taken under these conditions to ensure that the transmit status words do not become out of sync with the originating packet information. To insure that more data is not lost, transmit status words stored in the FIFO should be processed to free up more locations. Once set, this bit can only be cleared with a reset.

Interrupt (data bus bit 21) -- Transmit Status FIFO underrun interrupt

Indicates that an attempt was made to read the transmit status FIFO when it was empty and that the data received is not valid. Once set, this bit can only be cleared with a reset.

Interrupt (data bus bit 20) -- Transmit Length FIFO Overrun interrupt

Indicates that more transmit packets were written to the EMAC transmit queue than the transmit length FIFO could store and data was lost. This is non-recoverable condition since some or all of the packet data may have been stored in the transmit data FIFO and it can not be removed.

Since there is not a transmit length entry for that packet, the transmit length and data FIFOs are no longer synchronized. This condition should be corrected by forcing an immediate reset of the transmit data FIFO and the transmit path in the EMAC. Once set, this bit can only be cleared with a reset.

Interrupt (data bus bit 19) -- Transmit Length FIFO Underrun interrupt

Indicates that the EMAC attempted to remove an entry from the transmit length FIFO following the completion of a transmission and there were no entries in the FIFO. This should never be possible and represents a serious error. This condition should be corrected by forcing an immediate reset of the transmit data FIFO and the transmit path in the EMAC. condition. Once set, this bit can only be cleared with a reset.

Interrupt (data bus bit 18) -- Transmit Pause Packet Received interrupt

Indicates that transmissions has paused as requested by a received pause packet.

Interrupt (data bus bit 17) -- Receive Data FIFO Overrun interrupt

Indicates that the receive data FIFO became full during the reception of a packet and data was lost. The EMAC will remove the partial packet from the receive data FIFO and no receive status or length will be stored but to insure that more data is not lost, receive packets stored in the FIFO should be processed to free up more locations.

Interrupt (data bus bit 16) -- Receive Missed Frame Error interrupt

Indicates that at least one frame addressed to the EMAC under the current address validation modes, could not be received and the corresponding data was lost.

Interrupt (data bus bit 15) -- Receive Collision Error interrupt

Indicates that at least one frame could not be received due to a collision and the corresponding data was lost.

Interrupt (data bus bit 14) -- Receive FCS Error interrupt

Indicates that at least one frame addressed to the EMAC under the current address validation modes, contained an FCS error and the corresponding data was discarded.

Interrupt (data bus bit 13) -- Receive Length Field Error interrupt

Indicates that at least one frame addressed to the EMAC under the current address validation modes, contained a length field which did not match the actual frame length and the corresponding data was discarded.

Interrupt (data bus bit 12) -- Receive Short Error interrupt

Indicates that at least one frame addressed to the EMAC under the current address validation modes, was shorter than allowed and the corresponding data was discarded.

Interrupt (data bus bit 11) -- Receive Long Error interrupt

Indicates that at least one frame addressed to the EMAC under the current address validation modes, was longer than allowed and the corresponding data was discarded.

Interrupt (data bus bit 10) -- Receive Alignment Error interrupt

Indicates that at least one frame addressed to the EMAC under the current address validation modes, was not integral number of bytes in length corresponding data was truncated to the last full byte

.

EMAC Register Definition

EMAC IPIF Registers

The EMAC design contains registers in each of the two modules (IPIF and EMAC core). The registers in Table 4 are contained in the IPIF module and are included for completeness of this specification. Detailed descriptions of these registers are provided in the IPIF specifications listed in *Reference Documents*.

The registers in Table 5 are contained in the EMAC core module and are described in detail in this specification. The addresses for all registers are based on a parameter which is the base address for the entire EMAC module. The address of each register is then calculated by an offset to the base address.

Table 4: **EMAC IPIF Registers**

Register Name	OPB ADDRESS	Access
Transmit DMA & Scatter Gather Reset Register	C_DEV_BASEADDR + 0x2300	Write
Transmit DMA & Scatter Gather Module Identification Register	C_DEV_BASEADDR + 0x2300	Read
Transmit DMA & Scatter Gather Control Register	C_DEV_BASEADDR + 0x2304	Read/Write
Transmit DMA & Scatter Gather source address	C_DEV_BASEADDR + 0x2308	Read/Write
Transmit DMA & Scatter Gather destination address	C_DEV_BASEADDR + 0x230C	Read/Write
Transmit DMA & Scatter Gather start/length	C_DEV_BASEADDR + 0x2310	Read/Write
Transmit DMA & Scatter Gather Status Register	C_DEV_BASEADDR + 0x2314	Read
Transmit DMA & Scatter Gather Buffer Descriptor Address	C_DEV_BASEADDR + 0x2318	Read/Write
Transmit DMA Software Control Register	C_DEV_BASEADDR + 0x231C	Read/Write
Transmit DMA & Scatter Gather Unserviced Packet Count	C_DEV_BASEADDR + 0x2320	Read/Write
Transmit DMA & Scatter Gather Packet Count Threshold	C_DEV_BASEADDR + 0x2324	Read/Write
Transmit DMA & Scatter Gather Packet Wait Bound	C_DEV_BASEADDR + 0x2328	Read/Write
Transmit DMA & Scatter Gather Interrupt Status Register	C_DEV_BASEADDR + 0x232C	Read/toggle on Write

Table 4: **EMAC IPIF Registers *(Continued)***

Register Name	OPB ADDRESS	Access
Transmit DMA & Scatter Gather Interrupt Enable Register	C_DEV_BASEADDR + 0x2330	Read/Write
Receive DMA & Scatter Gather Reset Register	C_DEV_BASEADDR + 0x2340	Write
Receive DMA & Scatter Gather Module Identification Register	C_DEV_BASEADDR + 0x2340	Read
Receive DMA & Scatter Gather Control Register	C_DEV_BASEADDR + 0x2344	Read/Write
Receive DMA & Scatter Gather source address	C_DEV_BASEADDR + 0x2348	Read/Write
Receive DMA & Scatter Gather destination address	C_DEV_BASEADDR + 0x234C	Read/Write
Receive DMA & Scatter Gather start/length	C_DEV_BASEADDR + 0x2350	Read/Write
Receive DMA & Scatter Gather Status Register	C_DEV_BASEADDR + 0x2354	Read
Receive DMA & Scatter Gather Buffer Descriptor Address	C_DEV_BASEADDR + 0x2358	Read/Write
Receive DMA Software Control Register	C_DEV_BASEADDR + 0x235C	Read/Write
Receive DMA & Scatter Gather Unservice Packet Count	C_DEV_BASEADDR + 0x2360	Read/Write
Receive DMA & Scatter Gather Packet Count Threshold	C_DEV_BASEADDR + 0x2364	Read/Write
Receive DMA & Scatter Gather Packet Wait Bound	C_DEV_BASEADDR + 0x2368	Read/Write
Receive DMA & Scatter Gather Interrupt Status Register	C_DEV_BASEADDR + 0x236C	Read/toggle on Write
Receive DMA & Scatter Gather Interrupt Enable Register	C_DEV_BASEADDR + 0x2370	Read/Write
Write Packet FIFO reset (write) Module Identification (read)	C_DEV_BASEADDR + 0x2000	Read/Write
Write Packet FIFO Vacancy	C_DEV_BASEADDR + 0x2004	Read
Write Packet FIFO data write port	C_DEV_BASEADDR + 0x2100 thru 0x28FF	Write
Read Packet FIFO reset (write) Module Identification (read)	C_DEV_BASEADDR + 0x2010	Read/Write
Read Packet FIFO Occupancy	C_DEV_BASEADDR + 0x2014	Read
Read Packet FIFO data read port	C_DEV_BASEADDR + 0x2200 thru 0x29FF	Read
Device Interrupt Status Register	C_DEV_BASEADDR + 0x0000	Read/Write
Device Interrupt Pending Register	C_DEV_BASEADDR + 0x0004	Read/Write
Device Interrupt Enable Register	C_DEV_BASEADDR + 0x0008	Read/Write
Device Interrupt Identification Register	C_DEV_BASEADDR + 0x0018	Read/Write
Device Global Interrupt Enable	C_DEV_BASEADDR + 0x001C	Read/Write
IP Interrupt Status Register	C_DEV_BASEADDR + 0x0020	Read/Write
IP Interrupt Enable Register	C_DEV_BASEADDR + 0x0028	Read/Write
Device Software Reset (write) Module Identification (read) Register	C_DEV_BASEADDR + 0x0040	Read/Write

EMAC Core Registers

The EMAC core registers are listed in Table 5.

Table 5: **EMAC Core Registers**

Register Name	OPB ADDRESS	Access
EMAC Module Identification Register (EMIR)	C_DEV_BASEADDR + 0x1100	Read
EMAC Control Register (ECR)	C_DEV_BASEADDR + 0x1104	Read/Write
Interframe Gap Register (IFGP)	C_DEV_BASEADDR + 0x1108	Read/Write
Station Address High (SAH)	C_DEV_BASEADDR + 0x110C	Read/Write
Station Address Low (SAL)	C_DEV_BASEADDR + 0x1110	Read/Write
MII Management Control Register (MGTCR)	C_DEV_BASEADDR + 0x1114	Read/Write
MII Management Data Register (MGTDR)	C_DEV_BASEADDR + 0x1118	Read/Write
Receive Packet Length Register (RPLR)	C_DEV_BASEADDR + 0x111C	Read
Transmit Packet Length Register (TPLR)	C_DEV_BASEADDR + 0x1120	Read/Write
Transmit Status Register (TSR)	C_DEV_BASEADDR + 0x1124	Read
Receive Missed Frame Count (RMFC)	C_DEV_BASEADDR + 0x1128	Read
Receive Collision Count (RCC)	C_DEV_BASEADDR + 0x112C	Read
Receive FCS Error Count (RFCSEC)	C_DEV_BASEADDR + 0x1130	Read
Receive Alignment Error Count (RAEC)	C_DEV_BASEADDR + 0x1134	Read
Transmit Excess Deferral Count (TEDC)	C_DEV_BASEADDR + 0x1138	Read

EMAC Module Identification Register (EMIR)

The EMAC Version Register provides the software with a convenient method of verifying the Ethernet IP version and type.

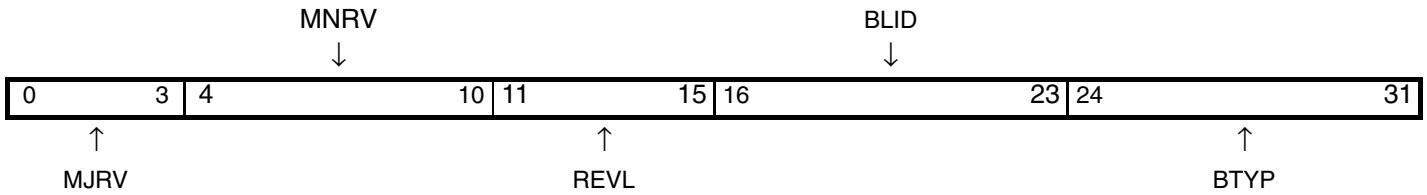

Figure 6: **EMIR**

Table 6: **EMAC Module Identification Register Bit Definitions**

Bit Location	Name	Core Access	Reset Value	Description
0 - 3	Major Version Number (MJRV)	Read	Version ID "0001" for this major version of 1	Module Major Version Number.
4 - 10	Minor Version Number (MNRV)	Read	Version ID "0000000" for this minor version of 0	**Module Minor Version Number.**

Table 6: **EMAC Module Identification Register Bit Definitions *(Continued)***

Bit Location	Name	Core Access	Reset Value	Description
11 -15	Rev. Letter (REVL)	Read	Version ID "01010" for this revision of "k"	Module Minor Version Letter. This is a binary encoding of small case letters a through z (00000 - 11001).
16 - 23	Block ID (BLID)	Read	Assigned by Platform Generator defaults to "00000001"	**Block ID Number.** Distinct number for each EMAC instantiated by Platform Generator.
24 - 31	Block Type (BTYP)	Read	"00000001"	**Block Type.** This is an 8 bit identifier unique to each IP type. For EMAC this type is hex 01.

EMAC Control Register (ECR)

The EMAC Control Register controls the operation of the EMAC. Please note that some of these bits should not be changed while transmit and receive are enabled (ECR.ENTX and/or ECR.ENRX =’1’).

.

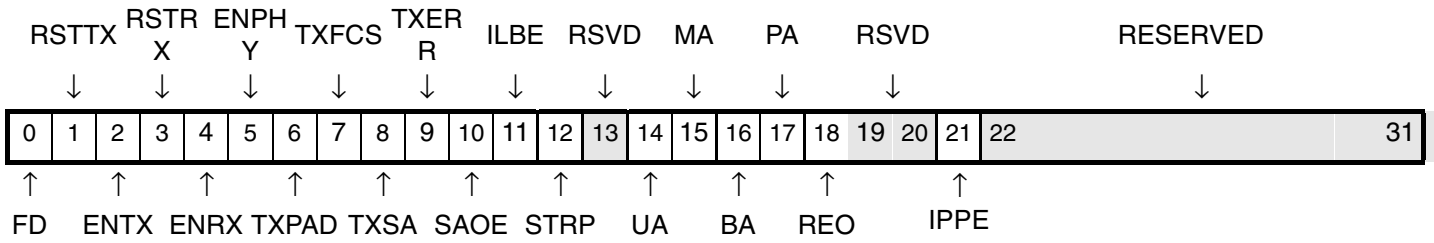

Figure 7: **ECR**

Table 7: **EMAC Control Register Bit Definitions**

Bit Location	Name	Core Access	Reset Value	Description
0	FD	Read/Write	'0'	Full Duplex. Selects either full duplex mode (i.e., EMAC can receive and transmit simultaneously on a dedicated Ethernet bus segment) or half duplex mode. Choosing half duplex enables CSMA/CD mode. Choosing full duplex mode disables CCSMA/CD mode. It is the responsibility of the software to ensure that this mode matches the PHY if the PHY is operating in auto-negotiation mode. This bit should not be modified while transmit and receive are enabled ECR.ENTX and/or ECR.ENRX ='1'. • '0' -Half Duplex • '1' - Full Duplex
1	RSTTX	Read/Write	'1'	Reset Transmitter. Immediately resets the transmitter circuitry regardless of its current state. The transmitter circuitry will remain in reset until this bit is set to '0'. • '0' - Normal Operation • '1' - Reset
2	ENTX	Read/Write	'0'	Enable Transmitter. The transmitter circuitry will leave the idle state and begin transmission of a packet only when this bit is '1' and the transmit length register is not empty. Setting this bit to '0' will cause the transmitter to enter the idle state after completion of any packet transmission in progress (graceful halt). • '0' - Disable Transmitter • '1'- Enable Transmitter
3	RSTRX	Read/Write	'1'	Reset Receiver. Immediately resets the receiver circuitry regardless of its current state. The receiver circuitry will remain in reset until this bit is set to '0'. • '0' - Normal Operation • '1' - Reset
4	ENRX	Read/Write	'0'	Enable Receiver. The receiver circuitry will leave the idle state and begin monitoring the Ethernet bus only when this bit is '1'. Setting this bit to '0' will cause the receiver to enter the idle state after completion of any packet reception in progress (graceful halt). This bit also controlls the output signal PHY_rx_en. • '0' - Disable Receiver • '1'- Enable Receiver

Table 7: **EMAC Control Register Bit Definitions *(Continued)***

Bit Location	Name	Core Access	Reset Value	Description
5	ENPHY	Read/Write	'1'	Enable PHY. This value of this bit is driven to the PHY interface reset_n signal. If the external PHY supports this signal and this bit is '0', the PHY will reset and remain in reset until this bit is set to '1'. • '0' - Disable / Reset PHY • '1'- Enable PHY
6	TXPAD	Read/Write	'1'	Enable Transmit Auto Pad Insertion. Enables automatic pad field insertion by the EMAC circuitry if it is necessary. When this is enabled, the transmit packet data provided to the EMAC should not contain pad data. When this is enabled, auto FCS insertion must also be selected to insure correct FCS calculation over the pad field. When this is disabled, the transmit packet data provided to the EMAC should contain pad data if required. This bit should not be modified while transmit and receive are enabled ECR.ENTX ='1' and/or ECR.ENRX ='1'. • '0' - Disable Auto Pad Insertion • '1'- Enable Auto Pad Insertion
7	TXFCS	Read/Write	'1'	Enable Transmit Auto FCS Insertion. Enables automatic FCS field insertion by the EMAC circuitry. When this is enabled, the transmit packet data provided to the EMAC should not contain FCS data. When this is disabled, the transmit packet data provided to the EMAC should contain FCS data. This bit should not be modified while transmit and receive are enabled ECR.ENTX ='1' and/or ECR.ENRX ='1'. • '0' - Disable Auto FCS Insertion • '1'- Enable Auto FCS Insertion
8	TXSA	Read/Write	'1'	Enable Transmit Auto Source Address Insertion. Enables automatic source address field insertion from the Station Address Registers by the EMAC circuitry. When this is enabled, the transmit packet data provided to the EMAC should not contain source address data. When this is disabled, the transmit packet data provided to the EMAC should contain source address data. This bit should not be modified while transmit and receive are enabled ECR.ENTX ='1' and/or ECR.ENRX ='1'. • '0' - Disable Auto Source Address Insertion • '1'- Enable Auto Source Address Insertion

Table 7: **EMAC Control Register Bit Definitions *(Continued)***

Bit Location	Name	Core Access	Reset Value	Description
9	TXERR	Read/Write	'0'	Transmit Error Insertion. The value of this bit is driven to the PHY interface TX_ER signal. If the external PHY supports this mode, it will inject an error encoded byte into the transmit data when operating in 100 Base-T mode. The PHY will ignore this input when operating in 10Base-T mode. This bit should not be modified while transmit and receive are enabled ECR.ENTX ='1' and/or ECR.ENRX ='1'. • '0' - Disable Error Insertion • '1' - Enable Error Insertion
10	SAOE	Read/Write	'1'	Source Address Overwrite Enable. When set to '1', it enables overwriting of the source address field provided in the packet data to be transmitted. The source address field is overwritten with the value contained in the SAH and SAL registers. When set to '0', the source address field is not included in the packet data to be transmitted and the value contained in the SAH and SAL registers is inserted into the packet data stream. This bit is only used when auto source address insertion is enabled ECR.TXSA ='1'.
11	ILBE	Read/Write	'0'	**Internal Loop-Back Enable.** Enables looping of the transmit data directly to the receive data path internally to the EMAC. The transmit and receive paths are isolated from the external PHY.
12	STRP	Read/Write	'0'	Pad & FCS Strip Enable. Enables stripping of receive pad and FCS fields when type/length field is a length. • '0' - Disable Strip • '1' - Enable Strip
13	Reserved	Read	'0'	• **Reserved.** This bit is reserved for future use.
14	UA	Read/Write	'1'	Enable Unicast Address. Enables the EMAC to accept valid frames that have a destination address field that matches the value in the station address registers. This bit should not be modified while transmit and receive are enabled ECR.ENTX ='1' and/or ECR.ENRX ='1". • '0' - Disable Unicast Address • '1' - Enable Unicast Address
15	MA	Read/Write	'0'	Enable Multicast Address. Enables the EMAC to accept valid frames that have a multicast (excluding broadcase) destination address field. This bit should not be modified while transmit and receive are enabled ECR.ENTX ='1' and/or ECR.ENRX ='1". • '0' - Disable Multicast Address • '1' - Enable Multicast Address

Table 7: **EMAC Control Register Bit Definitions *(Continued)***

Bit Location	Name	Core Access	Reset Value	Description
16	BA	Read/Write	'1'	Enable Broadcast Address. Enables the EMAC to accept valid frames that have a broadcast destination address field. This bit should not be modified while transmit and receive are enabled ECR.ENTX ='1' and/or ECR.ENRX ='1'. • '0' - Disable Broadcast Address • '1' - Enable Broadcast Address
17	PA	Read/Write	'0'	Enable Promiscuous Address Mode. Enables the EMAC to accept valid frames. This bit should not be modified while transmit and receive are enabled ECR.ENTX ='1' and/or ECR.ENRX ='1'. • '0' - Disable Promiscuous Address Mode • '1' - Enable Promiscuous Address Mode
18	REO	Read/Write	'0'	Receive Error Override. Enables the EMAC to attempt to receive and store frames even if they contain errors • '0' - Disable Error Override • '1' - Enable Error Override
19-20	Reserved	Read	"00"	**Reserved.** These bits are reserved for future use.
21	IPPE	Read/Write	'0'	Interpret Pause Packets. Enables the EMAC to proccess valid received pause packets. • '0' - Disable Pause Packets • '1' - Enable Pause Packets
22-31	Reserved	Read	0x000	**Reserved.** These bits are reserved for future use.

Interframe Gap Register (IFGP)

The Interframe Gap Register controls the duration of the interframe Gap. The Interframe Gap is the sum of IFGP1 and IFGP2, measuring in units of the bit time multiplied by four. Please refer to the paragraph Interframe Gap and Deferring for information about how the Interframe Gap is used by the EMAC. Please note that these settings should not be changed while transmit and receive are enabled (ECR.ENTX and/or ECR.ENRX ='1').

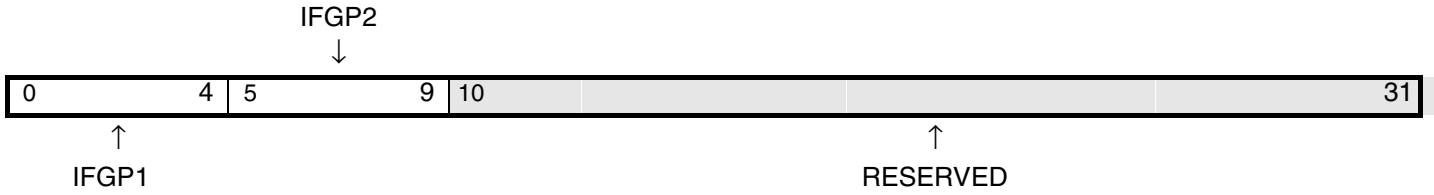

Figure 8: **IFGP**

Table 8: **Interframe Gap Register Bit Definitions**

Bit Location	Name	Core Access	Reset Value	Description
0-4	IFGP1	Read/Write	"10000"	Interframe Gap Part 1. A value of 1 in this field would provide a 4 bit time interframe part 1 gap to be combined with the interframe part 2 gap for the total interframe gap time. The reset value for this field is value recommended for most operation by IEEE Std. 802.3. This field should not be modified while transmit and receive are enabled ECR.ENTX ='1' and/or ECR.ENRX ='1'.
5-9	IFGP2	Read/Write	"01000"	**Interframe Gap Part 2.** A value of 1 in this field would provide a 4 bit time interframe part 2 gap to be combined with the interframe part 1 gap for the total interframe gap time. The reset value for this field is value recommended for most operation by IEEE Std. 802.3. This field should not be modified while transmit and receive are enabled ECR.ENTX ='1' and/or ECR.ENRX ='1'.
10-31	Reserved	Read	0x000000	**Reserved.** These bits are reserved for future use.

Receive Packet Length Register (RPLR)

The receive packet length register is actually a FIFO of register values each corresponding to a valid frame received. The data for the frame is stored in the receive data FIFO and the status word is stored in the receive status register FIFO.

The data is written by the EMAC when the frame's destination address passes the current address validation modes and when the frame has been determined to be valid and the receive data FIFO had enough locations that all of the frame data has been saved. The existence of data in the receive packet length FIFO (FIFO empty flag is '0') may be used to initiate the processing of received packets until this FIFO is empty. Reading this register causes the current value to be removed from the FIFO.

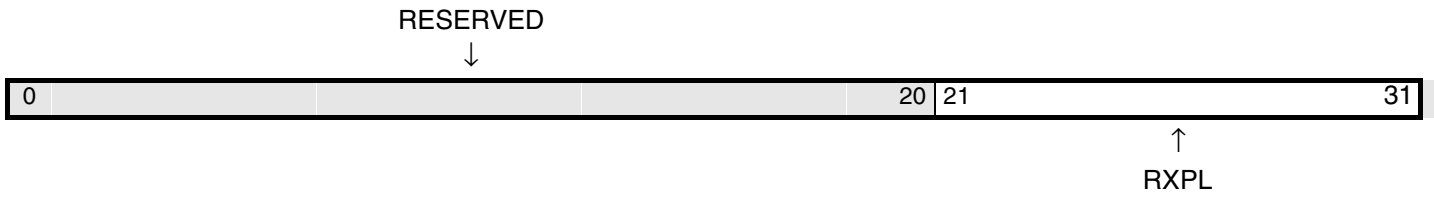

Figure 9: **RPLR**

Table 9: **Receive Packet Length Register Bit Definitions**

Bit Location	Name	Core Access	Reset Value	Description
0-20	Reserved	Read	0x00000	**Reserved.** These bits are reserved for future use.
21-31	RXPL	Read	0x000	Receive Packet Length. The number of bytes of the corresponding receive packet stored in the receive data FIFO.

Transmit Packet Length Register (TPLR)

The transmit packet length register is actually a FIFO of register values each corresponding to a valid frame ready for transmit. The data for the frame is stored in the transmit data FIFO.

The data is written to the EMAC over the external processor bus interface either by simple DMA, Scatter/Gather DMA, or by direct memory mapped access.

When presenting a transmit packet to the EMAC, the packet data should first be written to the transmit data FIFO. The existence of data in the transmit packet length FIFO (FIFO empty flag is '0') is used by the EMAC to initiate the processing of transmit packets until this FIFO is empty.

This register can be read over the processor interface but only the EMAC can remove a value from the FIFO. The EMAC will remove the current length from the FIFO when it completes the corresponding transmission. If multiple reads are performed prior to that completion, the same value will be returned for each read operation.

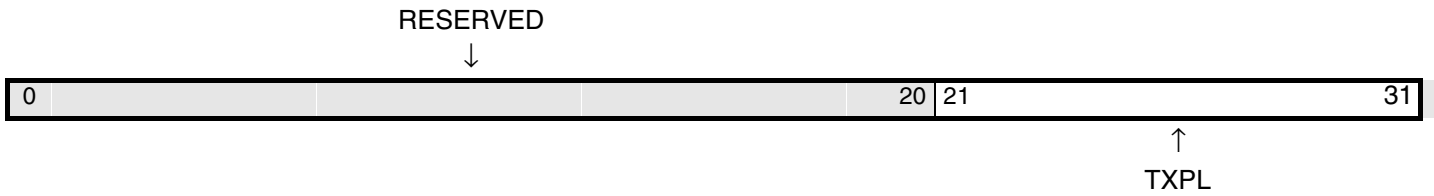

Figure 10: **TPLR**

Table 10: **Transmit Packet Length Register Bit Definitions**

Bit Location	Name	Core Access	Reset Value	Description
0-20	Reserved	Read	0x00000	**Reserved.** These bits are reserved for future use.
21-31	TXPL	Read/Write	0x000	Transmit Packet Length. The number of bytes of the corresponding transmit packet stored in the transmit data FIFO.

Receive Status Register (RSR)

The receive status register is a place holder for the receive status register that is used by the Scatter Gather DMA interface. The EMAC does not need a receive status register but is required to provide the correct value in bit 31 to the generalized Scatter Gather DMA circuitry as part of a standard receive packet operation.

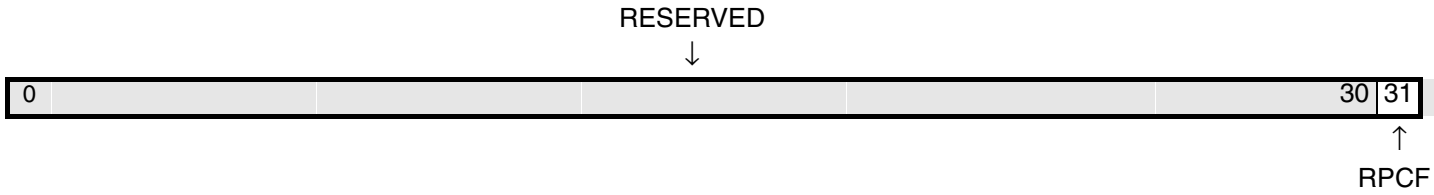

Figure 11: **RSR**

Table 11: **Receive Status Register Bit Definitions**

Bit Location	Name	Core Access	Reset Value	Description
0 - 30	Reserved	N/A	0x00000000	Reserved. These bits are unused and will always return all zeros.
31	RPCF	Read	'1'	Receive Packet Complete Flag. This bit is always '1' and is used to indicate to the software that a buffer descriptor associated with this packet has been completely processed by the DMA circuitry and is available for software use.

Transmit Status Register (TSR)

The transmit status register is actually a FIFO of register values each corresponding to a frame transmission attempt. The bits in this register reflect the specific status of the corresponding transmit operation including the EMAC settings which were applied to the transmit operation. Reading this register causes the current value to be removed from the FIFO.

Figure 12: **TSR**

Table 12: **Transmit Status Register Bit Definitions**

Bit Location	Name	Core Access	Reset Value	Description
0	TXED	Read	’0’	Transmit Excess Deferral Error. This bit is only applicable in half-duplex mode. It indicates that at least one transmit frame was not able to complete transmission due to collisions that exceed the maximum number of retries (16). This bit is cleared to ’0’ when read. • ’0’ - No excess deferrals occurred since the last read • ’1’ - At least one excess deferral has occurred
1	PFIFOU	Read	’0’	Packet Fifo Underrun. This bit indicates that at least one transmit frame experienced a packet FIFO underrun condition during transmission. This bit is cleared to ’0’ when read. • ’0’ - No packet FIFO underruns occurred since the last read ’1’ - At least one packet FIFO underrun has occurred
2- 6	TXA	Read	0x00	**Transmission Attempts.** The number of transmission attempts made. There will be a maximum of 16 attempts.
7	TXLC	Read	’0’	Transmit Late Collision Error. This bit is only applicable in half-duplex mode. It indicates a non-recoverable collision occurred more than 64-bit times after the start of the transmission. No automatic retransmission can be attempted by the EMAC. A late collision should never occur on a compliant Ethernet network. • ’0’ - No late collisions occurred • ’1’ - Late collision occurred
8 - 30	Reserved	N/A	0x000000	Reserved. These bits are unused and will always return all zeros.
31	TPCF	Read	’1’	Transmit Packet Complete Flag. This bit is always ’1’ and is used to indicate to the software that a buffer descriptor associated with this packet has been completely processed by the DMA circuitry and is available for software use.

Station Address High Register (SAH)

This register contains the high-order 16 bits of the 48 bit station address.

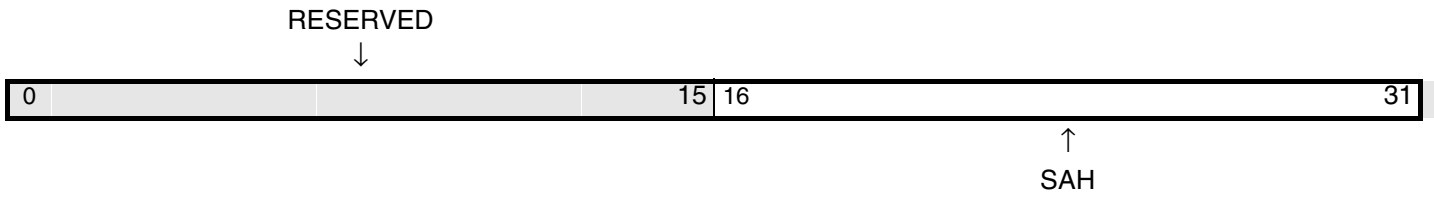

Figure 13: **SAH**

Table 13: **Station Address High Register Bit Definitions**

Bit Location	Name	Core Access	Reset Value	Description
0-15	Reserved	Read	0x0000	**Reserved.** These bits are reserved for future use.
16-31	SAH	Read/Write	0x0000	Station Address High. This register should not be modified while transmit and receive are enabled ECR.ENTX =’1’ and/or ECR.ENRX =’1’

Station Address Low Register (SAL)

This register contains the low-order 32 bits of the 48 bit station address.

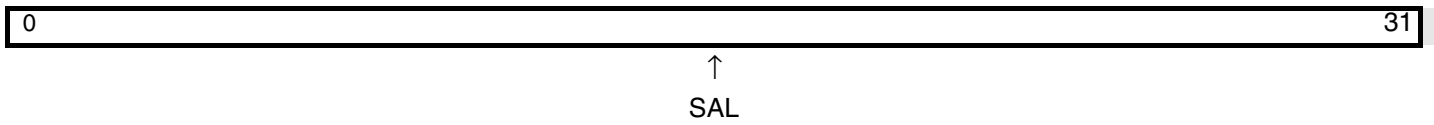

Figure 14: **SAL**

Table 14: **Station Address Low Register Bit Definitions**

Bit Location	Name	Core Access	Reset Value	Description
0-31	D0 - D31	Read/Write	0x00000000	Station Address Low. This register should not be modified while transmit and receive are enabled ECR.ENTX =’1’ and/or ECR.ENRX =’1”

MII Management Control Register (MGTCR)

The MII management control register is used with the MII management data register to perform read and writes between the EMAC and the external PHY device via the MII management interface.

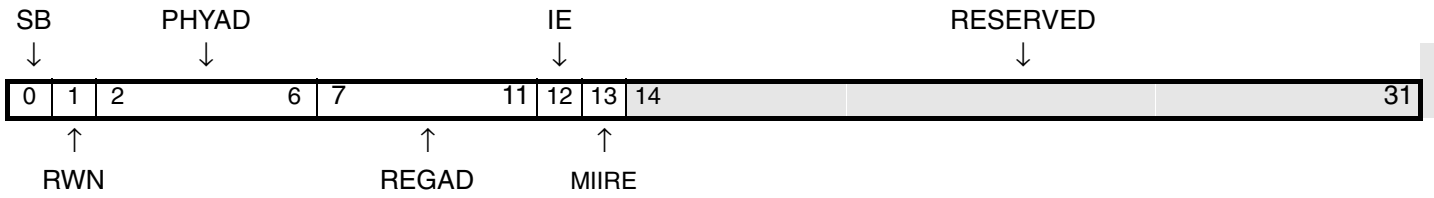

Figure 15: **MGTCR**

Table 15: **MII Management Control Register Bit Definitions**

Bit Location	Name	Core Access	Reset Value	Description
0	SB	Read/Write	'0'	Start / Busy. writing a '1' to this bit initiates an MII read or write operation. The EMAC will clear this bit to '0' when the operation has been completed. • '0' - No MII Operation in Progress • '1'- MII Read or Write in Progress
1	RWN	Read/Write	'1'	Read Write Not. This bit indicates the direction of the MII operation. • '0' - Write to PHY register • '1'- Read from PHY register
2-6	PHYAD	Read/Write	0x00	PHY Address. This field is used to specify the address of the PHY to be accessed.
7-11	REGAD	Read/Write	0x00	**Register Address.** This field is used to specify the register in the PHY to be accessed.
12	IE	Read/Write	'0'	MII Management Interface Enable. This bit controls the 3-state drivers for the MII management signal interface to the PHY. • '0' - The MII management signals to the PHY are 3-stated. • '1' - The MII management signals to the PHY are driven and controlled by the EMAC management interface.
13	MIIRE	Read	'0'	MII Management Read Error. Indicates that a read from a PHY register is invalid and the operation should be retried. This is indicated during a read turn-around cycle when the PHY does not drive the MDIO signal to the low state. This bit is cleared to '0' when read. • '0' - No read errors occurred since the last read • '1' - At least one read error has occurred
14-31	Reserved	Read	0x00000	**Reserved.** These bits are reserved for future use.

MII Management Data Register (MGTDR)

The MII management data register is used with the MII management control register to perform read and writes between the EMAC and the external PHY device via the MII management interface. For a PHY register write operation, data should be written to the data register prior to the write to the control register.

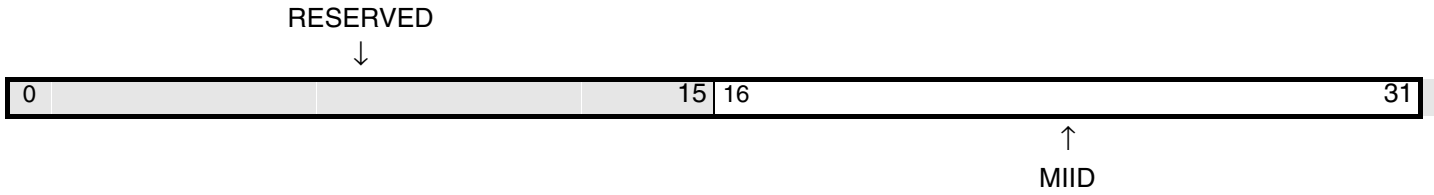

Figure 16: **MGTDR**

Table 16: **MII Management Data Register Bit Definitions**

Bit Location	Name	Core Access	Reset Value	Description
0 - 15	Reserved	N/A	0x0000	Reserved. These bits are unused and will always return all zeros.
16-31	MIID	Read/Write	0x0000	MII Management Data Register.

Receive Missed Frame Count (RMFC)

This register value represents the number of missed valid frames since the last reset with destination addresses that pass the current address validation modes.

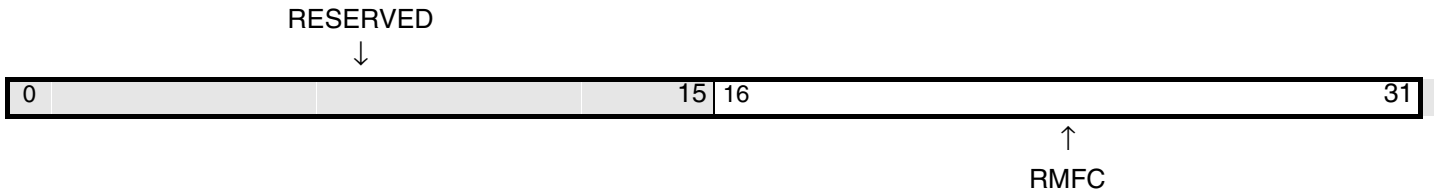

Figure 17: **RMFC**

Table 17: **Receive Missed Frame Count Bit Definitions**

Bit Location	Name	Core Access	Reset Value	Description
0 - 15	Reserved	N/A	0x0000	Reserved. These bits are unused and will always return all zeros.
16-31	RMFC	Read	0x0000	Receive Missed Frame Count.

Receive Collision Count (RCC)

This register value represents the number of received frames that were interrupt by a collision since the last reset. These frames may or may not have satisfied the current address validation modes. This counter is not used in full duplex mode.

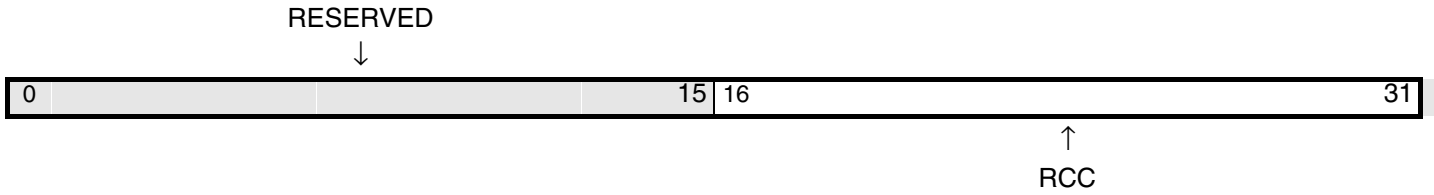

Figure 18: **RCC**

Table 18: **Receive Collision Count Bit Definitions**

Bit Location	Name	Core Access	Reset Value	Description
0 - 15	Reserved	N/A	0x0000	Reserved. These bits are unused and will always return all zeros.
16-31	RCC	Read	0x0000	Receive Collision Count.

Receive FCS Error Count (RFCSEC)

This register value represents the number of received frames since the last reset with destination addresses that pass the current address validation modes but with FCS validation failures.

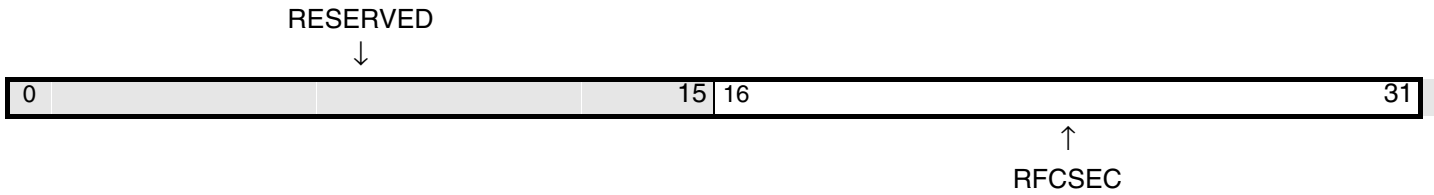

Figure 19: **CSEC**

Table 19: **Receive FCS Error Count Bit Definitions**

Bit Location	Name	Core Access	Reset Value	Description
0 - 15	Reserved	N/A	0x0000	Reserved. These bits are unused and will always return all zeros.
16-31	RFCSEC	Read	0x0000	Receive FCS Error Count.

Receive Alignment Error Count (RAEC)

This register value represents the number of received frames since the last reset with an odd number of nibbles that pass the current address validation modes and FCS validation with the extra nibbles truncated.

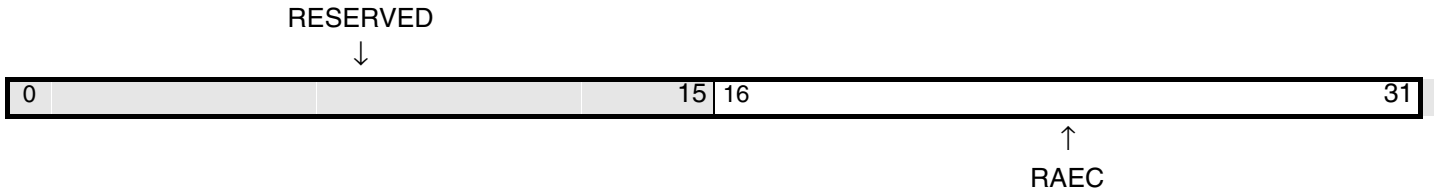

Figure 20: **RAEC**

Table 20: **Receive Alignment Error Count Bit Definitions**

Bit Location	Name	Core Access	Reset Value	Description
0 - 15	Reserved	N/A	0x0000	Reserved. These bits are unused and will always return all zeros.
16-31	RAEC	Read	0x0000	Receive Alignment Error Count.

Transmit Excess Deferral Count (TEDC)

This register value represents the number of transmitted frames since the last reset that could not be successful transmitted in 16 attempts. This counter is not used in full duplex mode.

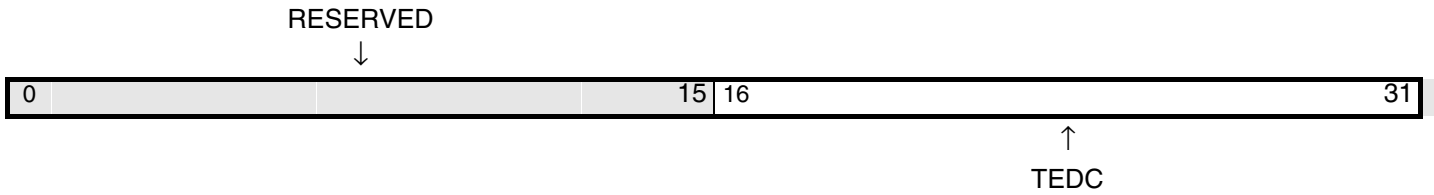

Figure 21: **TEDC**

.

Table 21: **Transmit Excess Deferral Count Bit Definitions**

Bit Location	Name	Core Access	Reset Value	Description
0 - 15	Reserved	N/A	0x0000	Reserved. These bits are unused and will always return all zeros.
16-31	TEDC	Read	0x0000	Transmit Excess Deferral Count.

EMAC Block Diagram

The top-level block diagram for the EMAC is shown in Figure 22.

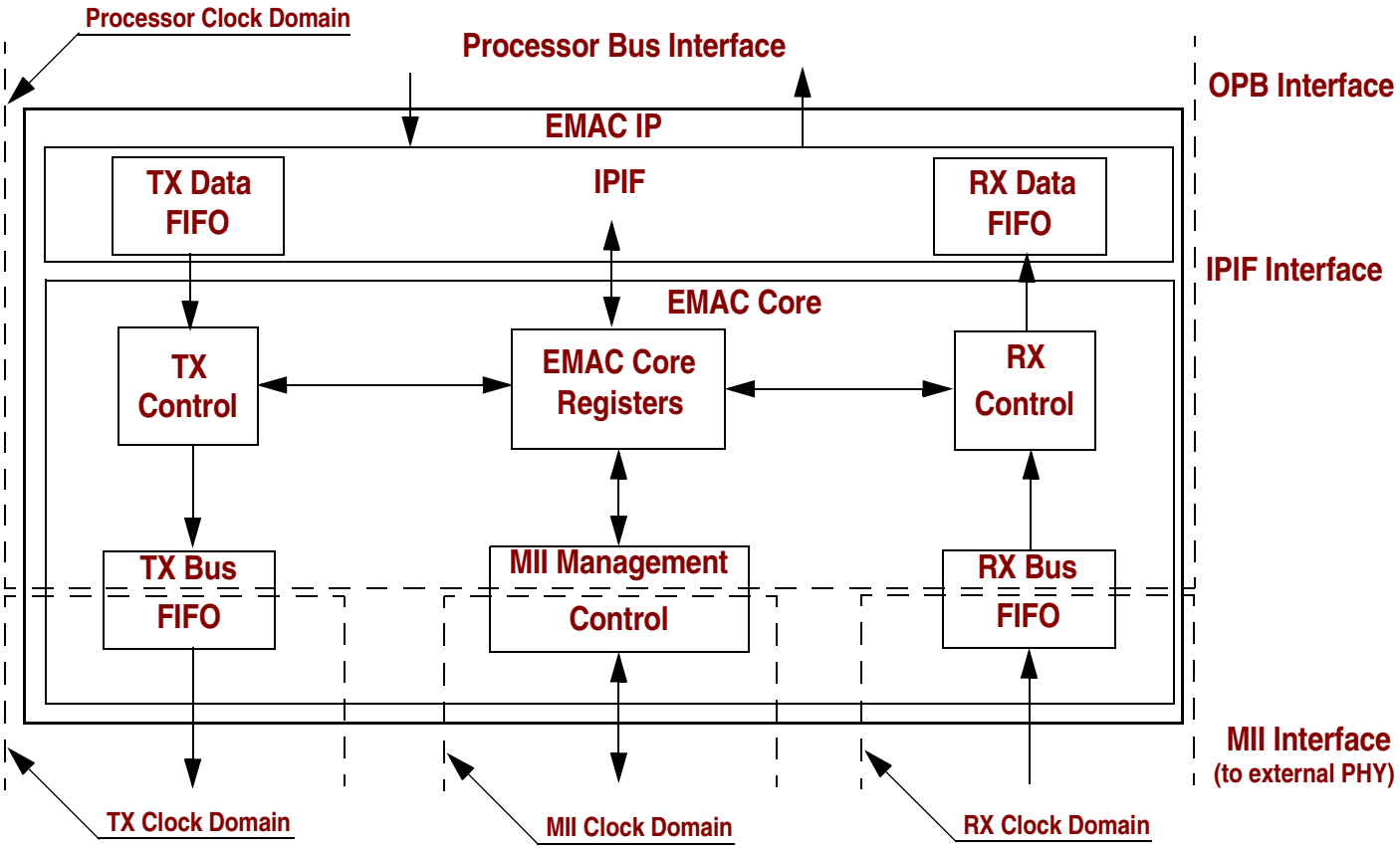

Figure 22: **EMAC Top-level Block Diagram**

Block Diagram TX Control

The block diagram for the **TX Control** is shown in Figure 23.

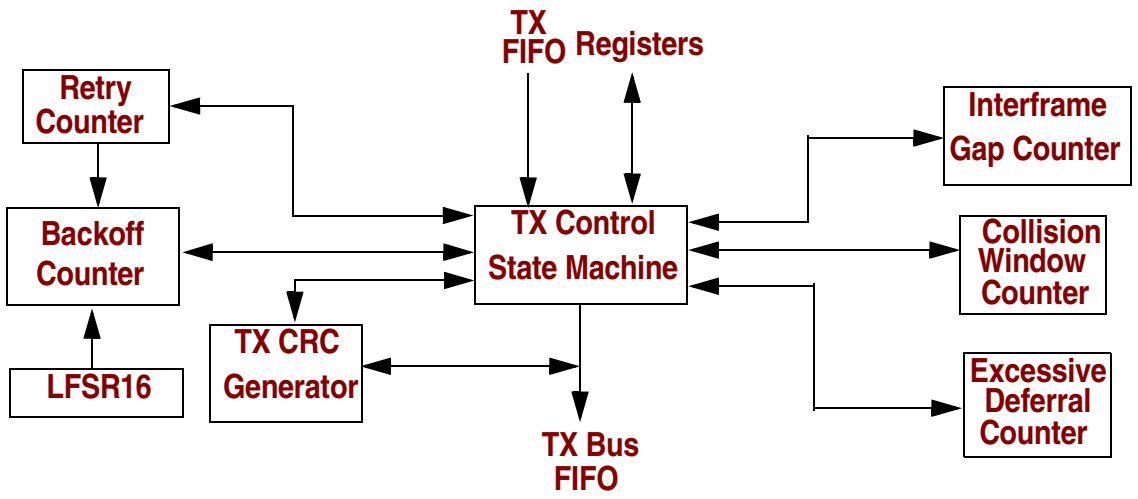

Figure 23: **TX Control Block Diagram**

Block Diagram RX Control

The block diagram for the **RX Control** is shown in Figure 24.

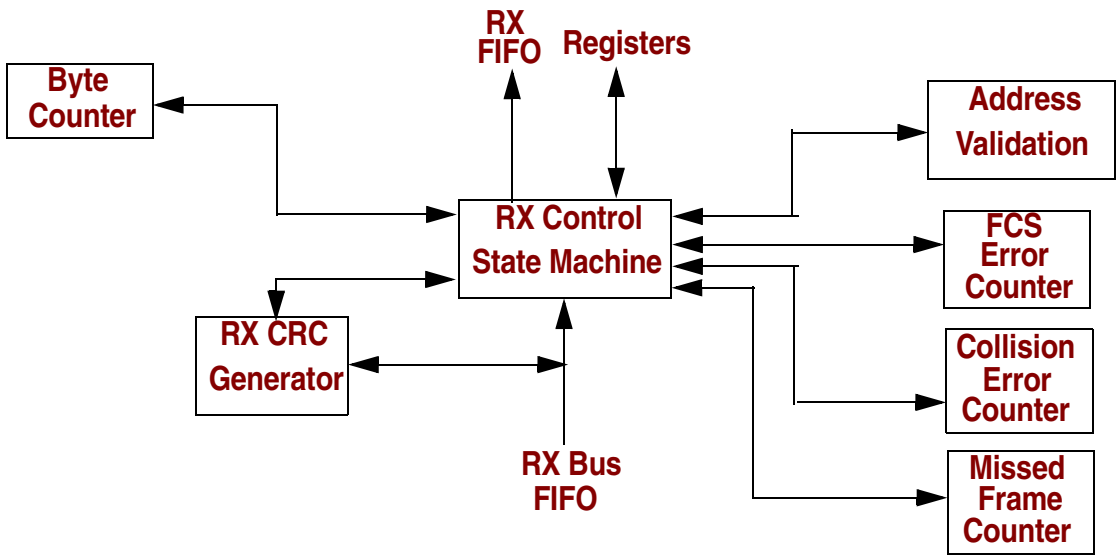

Figure 24: **RX Control Block Diagram**

EMAC Clocks

The EMAC design has four clock domains that are all asynchronous to each other. These clock domains and any special requirements regarding them are discussed below.

Transmit Clock

The transmit clock [TX_CLK] is generated by the external PHY and must be used by the EMAC to provide transmit data [TXD (3:0)] and control signals [TX_EN and TX_ER] 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 [RX_CLK] is also generated by the external PHY but is derived from the incoming Ethernet traffic. Like 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 [RX_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 EMAC to sample the receive data [RXD(3:0)] and control signals [RX_DV and RX_ER] from the PHY.

MII Management Clock

The Management Data Clock (MDC) is driven by the EMAC to the PHY as the reference for data transfer on the MDIO signal. This signal has no maximum high and low times and need not be periodic but must have a minimum high and low time of at least 160 nS with a corresponding minimum period of 400 nS (corresponds to a maximum of 2.5 MHz). The MII clock will be a divide by 64 from the OPB bus clock for OPB bus frequencies of 65 to 150 MHz range resulting in a MDC of approximately 1 to 2.3 MHz.

Processor Bus Clock

The majority of the EMAC operation functions in the processor bus clock domain. This clock must be greater than to equal to 65 MHz in order to transmit and receive Ethernet data at 100 Mbs and greater than to equal to 6.5 MHz in order to transmit and receive Ethernet data at 10 Mbs.

PHY Interface Signals

TX_EN

The EMAC uses the Transmit Enable signal (TX_EN) to indicate to the PHY that it is providing nibbles at the MII interface for transmission. It is asserted synchronously to TX_CLK with the first nibble of the preamble and remains asserted while all nibbles have been transmitted. TX_EN is negated prior to the first TX_CLK following the final nibble of a frame.

This signal is transferred between the 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 25 shows TX_EN timing during a transmission with no collisions.

RX_EN

The EMAC uses the Receive Enable signal (PHY_rx_en) to indicate to the PHY that the PHY should pass or block receive ethernet data. The value of this signal is controlled by the EMAC control register (ECR) bit 4. This signal is not used by many PHY devices.

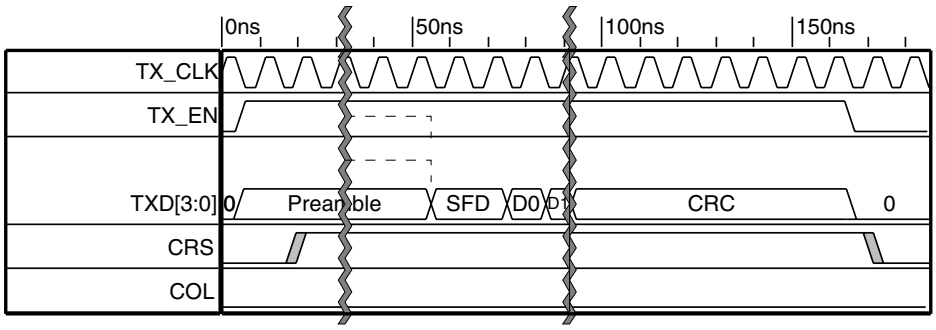

Figure 25: **Transmission with no collision**

TXD(3:0)

The EMAC drives the Transmit Data bus [TXD(3:0)] synchronously to TX_CLK. TXD(0)is the least significant bit. The PHY will transmit the value of TXD on every clock cycle that TX_EN is asserted. This bus is transferred between the 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 26.

.

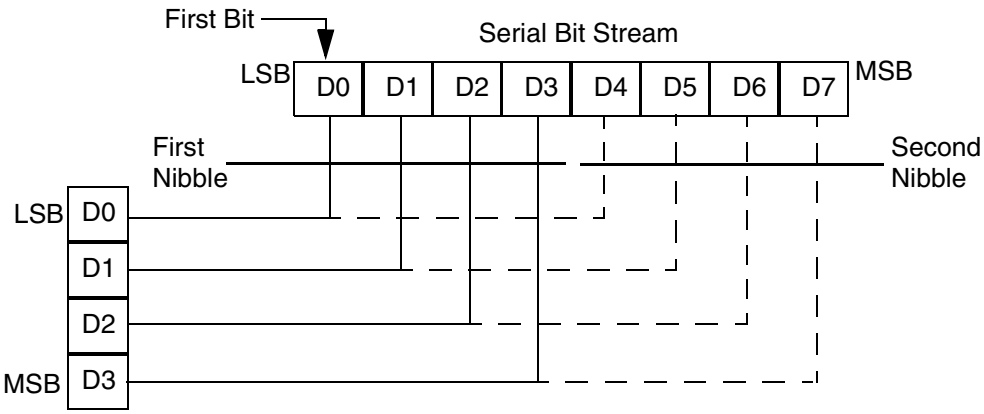

Figure 26: **Byte/Nibble Transmit and Receive Order**

TX_ER

The EMAC drives the Transmit coding Error signal (TX_ER) synchronously to TX_CLK. When TX_ER is asserted for one or more TX_CLK periods while TX_EN is also asserted and the PHY is operating in 100Mbs mode, the PHY transmits one or more symbols that are not part of the valid data or delimiter set somewhere in the frame being transmitted.

TX_ER had no effect on PHY operation in 10Mbs mode or when TX_EN is de-asserted. This signal is transferred between the 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. Table 22 shows the possible combinations for the transmit signals.

Table 22: **Possible values for TX_EN, TX_ER, and TXD(3:0)**

TX_EN	TX_ER	TXD(3:0)	Indication
0	0	0000 through 1111	Normal inter-frame
0	1	0000 through 1111	Reserved
1	0	0000 through 1111	Normal data transmission
1	1	0000 through 1111	Transmit error propagation

RX_DV

The PHY drives the Receive Data Valid (RX_DV) signal to indicate that the PHY is driving recovered and decoded nibbles on the RXD(3:0) bus and that the data on RXD(3:0) is synchronous to RX_CLK. RX_DV is driven synchronously to RX_CLK.

RX_DV remains asserted continuously from the first recovered nibble of the frame through the final recovered nibble and is negated prior to the first RX_CLK that follows the final nibble. In order for a received frame to be correctly received by the EMAC, RX_DV must encompass the frame, starting no later than the Start Frame Delimiter (SFD) and excluding any End-of-Frame delimiter.

This signal is transferred between the RX_CLK and processor clock domains at the asynchronous RX bus FIFO interface. The PHY will provide a minimum of 10 nS setup and hold time for this signal in reference to RX_CLK. Figure 27 shows the behavior of RX_DV during frame reception.

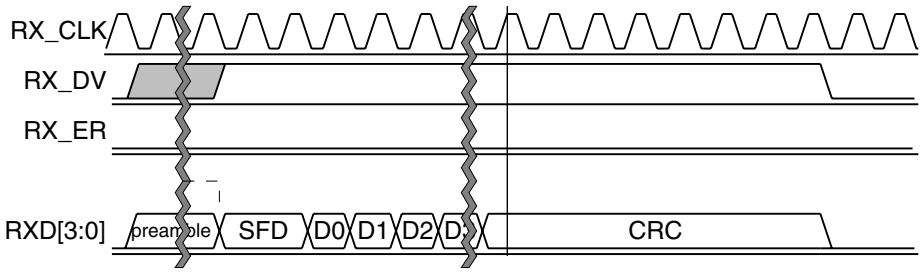

Figure 27: **Receive With No Errors**

RXD(3:0)

The PHY drives the Receive Data bus [RXD(3:0)] synchronously to RX_CLK. RXD(3:0) contains recovered data for each RX_CLK period in which RX_DV is asserted. RXD(0) is the least significant bit. The EMAC must not be affected by RXD(3:0) while RX_DV is de-asserted.

Also, the EMAC should ignore a special condition that occurs while RX_DV is de-asserted when the PHY may provide a False Carrier indication by asserting the RX_ER signal while driving the value <1110> onto RXD<3:0>. This bus is transferred between the RX_CLK and processor clock domains at the asynchronous RX bus FIFO interface. The PHY will provide a minimum of 10 nS setup and hold time for this signal in reference to RX_CLK.

RX_ER

The PHY drives the Receive Error signal (RX_ER) synchronously to RX_CLK. The PHY drives RX_ER for one or more RX_CLK periods to indicate that an error (e.g., 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 EMAC.

RX_ER should have no effect on the EMAC while RX_DV is de-asserted. This signal is transferred between the RX_CLK and processor clock domains at the asynchronous RX bus FIFO interface. The PHY will provide a minimum of 10 nS setup and hold time for this signal in reference to RX_CLK. Figure 28 shows the behavior of RX_ER during frame reception with errors.

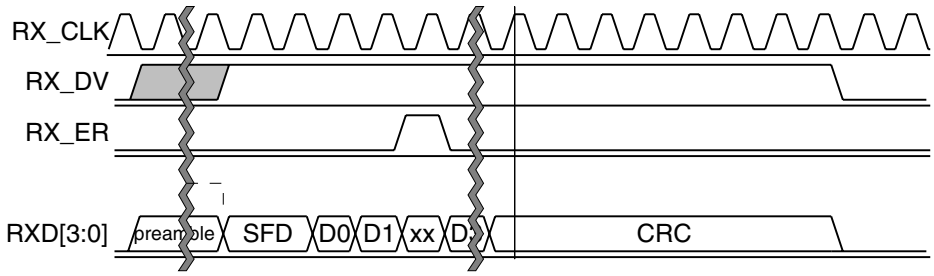

Figure 28: **Receive With Errors**

Table 23 shows the possible combinations for the receive signals.

Table 23: **Possible values for RX_DV, RX_ER, and RXD(3:0)**

TX_EN	TX_ER	TXD(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

CRS

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

The PHY drives CRS asserted throughout the duration of a collision condition. CRS is not synchronous to either the TX_CLK or the RX_CLK. The CRS signal is not used in full duplex mode. The CRS signal is used by both the EMAC transmit and receive circuitry and is double synchronized to the processor clock as it enters the EMAC.

COL

The PHY drives the Collision detected signal (COL) active to indicate the detection of a collision on the bus. The PHY drives CRS asserted while the collision condition persists. The PHY also drives COL asserted when operating at 10 Mbs for signal_quality_error (SQE) testing.

COL is not synchronous to either the TX_CLK or the RX_CLK. The COL signal is not used in full duplex mode. The COL signal is used by both the EMAC transmit and receive circuitry and is double synchronized to the processor clock as it enters the EMAC. Figure 29 shows the behavior of COL during frame transmission with a collision.

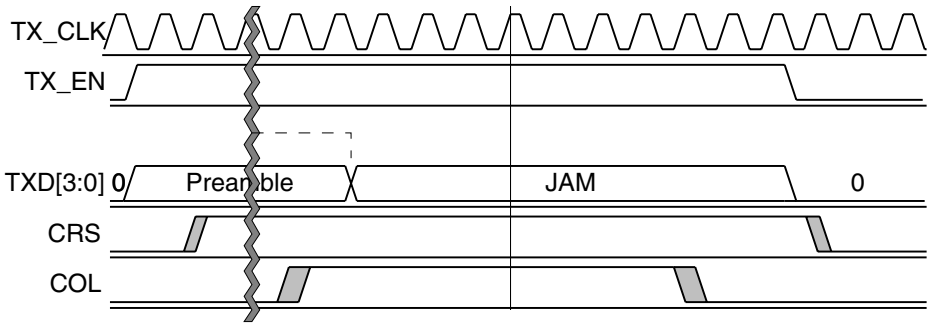

Figure 29: **Transmission With Collision**

MDIO

The Management Data Input/Output signal (MDIO) is a bidirectional signal between the PHY and the EMAC used to transfer control and status. All transfers via the MDIO are synchronous to the MDC signal. MDIO must driven through 3-state pins that enable either the EMAC or the PHY to drive the signal.

The necessary pull-up and pull-down resistors for this signal are defined in the IEEE Std. 802.3 paragraph 22.4.4.2. When the EMAC drives this signal, it must provide a minimum of 10nS setup and hold in reference to the MDC signal. When the PHY drives this signal, it will have a clock to output delay of 0 to 300nS.

Management Frame Structure

Data transferred via the MDIO is structured as frames as shown in Table 24.

Table 24: **Management Frame Format**

	Management Frame Fields							
	PRE	ST	OP	PHYAD	REGAD	TA	DATA	IDLE
Read	1...1	01	10	AAAAA	RRRRR	Z0	DDDDDDDDDDDDDDDD	Z
Write	1...1	01	01	AAAAA	RRRRR	10	DDDDDDDDDDDDDDDD	Z

Idle

The IDLE condition on MDIO is a high-impedance state. All 3-state drivers are disabled and the PHY's pull-up resistor will pull the MDIO line to a logic one.

Preamble (PRE)

At the beginning of each management transaction, the EMAC will transmit 32 contiguous logic one bits on the MDIO signal.

Start of Frame (ST)

The EMAC follows the preamble with the start of frame pattern which is "01".

Operation Code (OP)

The operation code is by the EMAC following the start of frame to indicate a read or write transaction. A read is designated by "10" and a write by "01".

PHY Address (PHYAD)

The next field transmitted by the EMAC is the 5 bit PHY address field. This identifies one of up to 32 PHYs connection to the same interface. The most significant bit of the address is transmitted first. The special PHY address "00000" will address any PHY.

Register Address (REGAD)

The EMAC follows the PHY address field with the transmission of the 5 bit register address field allowing the access of up to 32 PHY registers. The most significant bit of the address is transmitted first.

Turn Around (TA)

A turn around period of 2 bit times follows the register field and precedes the data field to prevent contention during a read cycle. For a read, both the EMAC and the PHY remain in a high-impedance state for the first bit time of the turnaround. The PHY will drive a zero bit during the second bit time of the turnaround of a read transaction.

During a write, the EMAC must drive a one bit for the first bit time of the turnaround and a zero bit for the second bit time of the turnaround. Figure 30 shows the behavior of the MDIO signal during the turnaround field of a read transaction.

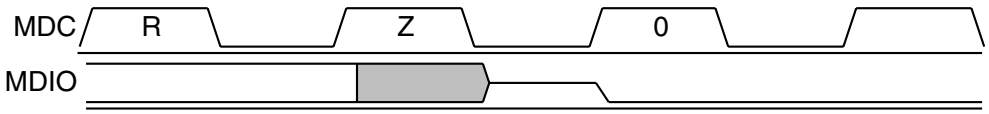

Figure 30: **MDIO Read Turn Around**

Data

The data field is 16 bits and is transmitted and received with bit 15 (MSb) first of the register being addressed.

Packet FIFO Interface

The EMAC uses the special "Mark", "Release", and "Restore" features of the packet FIFOs to assist in management of transmit and receive data under special circumstances.

The use of these features are discussed here as it applies to Ethernet operation. Also discussed are special considerations required to support a multi-cycle "dead" FIFO access time following a "Mark", "Release", and "Restore" operation.

Transmit Packet FIFO Interface

The use of the special FIFO features greatly simplifies the retransmission of a packet following a collision on a half duplex bus (the features are not needed for full duplex operation). When the EMAC is ready to transmit (transmit enable is '1' and the transmit length register is not empty) and it is in half-duplex mode and it is not deferring, it will perform a "Mark" in the transmit FIFO and begin reading transmit data (after a several cycle delay required by the FIFO) while counting down the length value.

If no collision occurs after 64 byte times, the EMAC will perform a "Release" to allow the FIFO locations containing the first parts of the transmit to be reused. Since the EMAC will not be able to read during the several cycles following the "release", it will have to insure that the transmit bus FIFO contains enough nibbles to sustain continuous transmission flow until packet FIFO reads can be resumed.

If a collision does occur in the first 64 byte times, the EMAC will transmit the JAM pattern, increment the retry count, and if retrys are less than 16, perform backoff and a "Restore" on the packet FIFO. Since the EMAC will be performing backoff and deferral it will not need to access the FIFO during the "Dead" FIFO access time following the "Restore".

If a transmission was attempted 16 retries and was unsuccessful, the EMAC will perform a "Release" and flush the remainder of the failed packet data from the transmit packet FIFO by performing the number of reads required to decrement the length counter to zero.

Receive Packet FIFO Interface

The use of the special FIFO features greatly simplifies the flushing of a packet following a receive error. When the EMAC detects that a packet is being received, the EMAC will perform a "Mark" on the receive packet FIFO and begin storing receive data (after a several cycle delay required by the FIFO) while counting up the length value.

The EMAC will continue reception and will store data into the receive FIFO until either the reception is successful, a receive error occurs, or the receive packet FIFO becomes full and overruns.

Following the completion of a successful receive, the EMAC will perform a "Release". However, if an error occurs during reception or if the receive packet FIFO becomes full and overruns, the EMAC will discontinue data storage into the receive packet FIFO and will flush all packet data stored there associated with the bad packet by performing a "Restore". There should be not be a need for special handling of the "dead" FIFO access time for "Restore" operations.

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'), and the multicast subgroup broadcast (all stations on the network).

The EMAC provides flexibility for enabling each of these modes as well as a promiscuous address mode where it accepts all addresses. The flow chart in Figure 31 shows the address validation process.

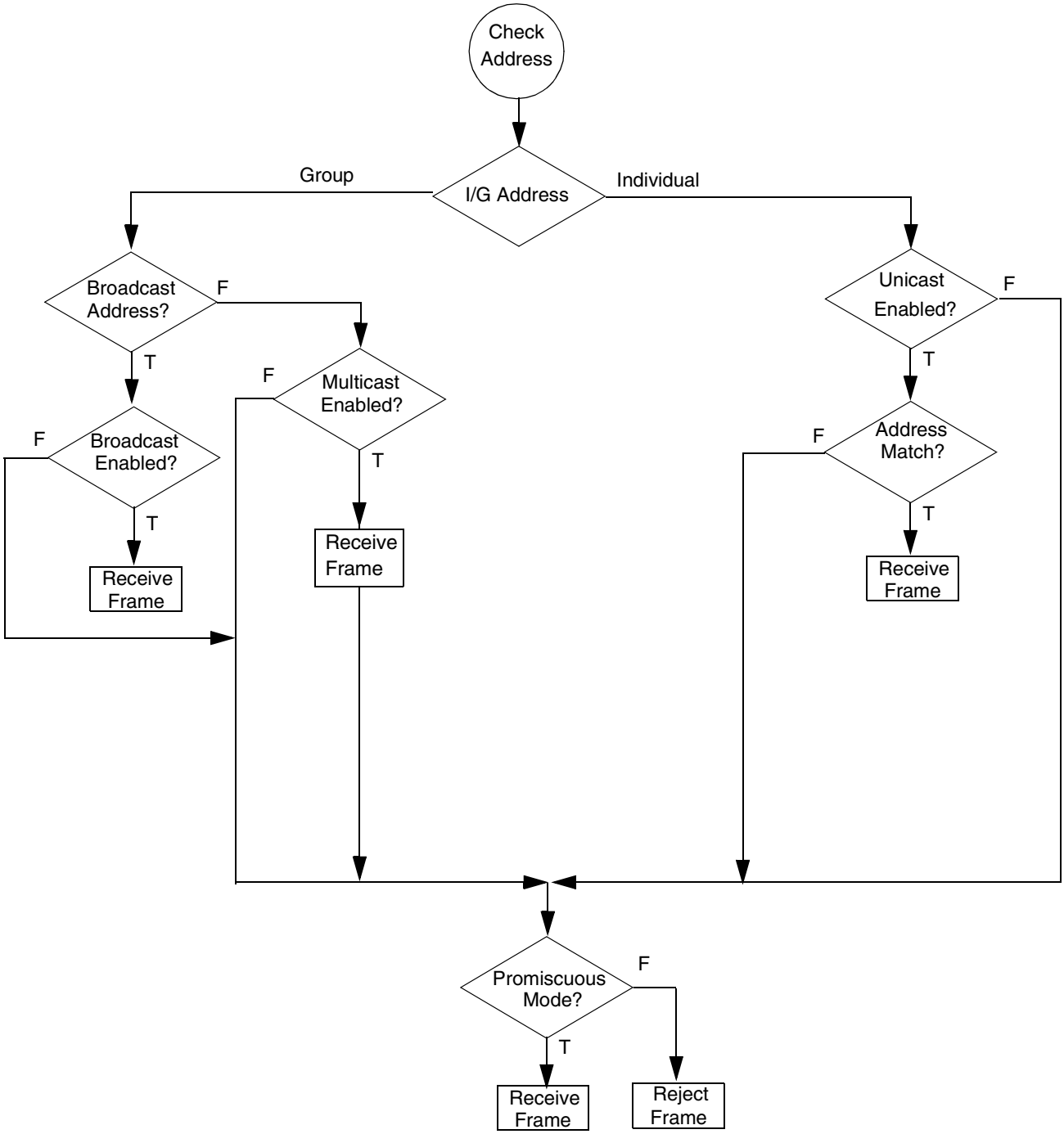

Figure 31: **Address Validation Process**

Freeze Mode

The freeze mode input signal is used for debug purposes and represents a request for a graceful halt. When this signal is activated, the transmit, receive, and MII management interface circuitry will complete any operations in progress and will go to an idle state. without modifying the state of any internal registers except for those required at the end of the completed operations.

While the EMAC is frozen, all registers normal accessible by the processor will remain accessible. When the freeze mode signal is deactivated, the EMAC will resume operation in the mode it was in prior to freezing. Note this may cause the receive circuitry to resume operation while a receive frame is already in progress creating possible error conditions.

Error Handling

Transmit Errors

Transmit errors detected by the EMAC circuitry are noted in the transmit status word and with interrupts and an error counter for maximum software implementation flexibility.

Receive Errors

The IEEE Std 802.3 prevents packets with error conditions from being passed to the software interface. The EMAC circuitry does provide interrupts, status register bits and error counters to allow the software to monitor receive error conditions. When a received frame is detected to have an error, the data is discarded by using the packet FIFO mark and restore features. No receive status word or length is stored for that frame.

MII Management Errors

The MII Management interface will detect and indicate that a read from a PHY register is invalid and the operation should be retried when the PHY does not drive the MDIO signal to the low state during a read turn-around cycle. This error condition is flagged in the MII management control register.

Design Constraints

The Ethernet Core requires design constraints to guarantee performance. These constraints should be placed in a .UCF file for the top level of the design. The following example of the constraint text is based on the port names of the Ethernet core. If these ports are mapped to FPGA pin names that are different, the FPGA pin names should be substituted for the port names in the example below.

```
NET "phy_rx_clk" TNM_NET = "RXCLK_GRP";
NET "phy_tx_clk" TNM_NET = "TXCLK_GRP";
TIMESPEC "TSTXOUT" = FROM "TXCLK_GRP" TO "PADS" 10 ns;
TIMESPEC "TSRXIN" = FROM "PADS" TO "RXCLK_GRP" 6 ns;
NET "opb_rst" TIG;
NET "phy_rx_clk" USELOWSKEWLINES;
NET "phy_tx_clk" USELOWSKEWLINES;
NET "phy_tx_clk" MAXSKEW= 2.0 ns;
NET "phy_rx_clk" MAXSKEW= 2.0 ns;
NET "phy_rx_clk" PERIOD = 40 ns HIGH 14 ns;
NET "phy_tx_clk" PERIOD = 40 ns HIGH 14 ns;
NET "phy_rx_data<3>" NODELAY;
NET "phy_rx_data<2>" NODELAY;
NET "phy_rx_data<1>" NODELAY;
```

```
NET "phy_rx_data<0>" NODELAY;
NET "phy_dv" NODELAY;
NET "phy_rx_er" NODELAY;
NET "phy_crs" NODELAY;
NET "phy_col" NODELAY;
```

Design Implementation

Device Utilization and Performance Benchmarks

Since the EMAC is a module that will be used with other design pieces in the FPGA, the utilization and timing numbers reported in this section are just estimates. As the EMAC is combined with other pieces of the FPGA design, the utilization of FPGA resources and timing of the EMAC design will vary from the results reported here.

In order to analyze the EMAC's timing within the FPGA, a design will be created that instantiates the EMAC with the following parameters set.

The EMAC benchmarks are shown in Table 25 for a Virtex™-E -8 FPGA

.

Table 25: **EMAC FPGA Performance and Resource Utilization Benchmarks (Virtex™-E -8)**

Parameter Values						Device Resources				f_{MAX} (MHz)
C_MAC_FIFO_DEPTH	C_IPIF_FIFO_DEPTH	C_DEV_MIR_ENABLE	C_RESET_PRESENT	C_INCLUDE_DEV_PENCODER	C_DMA_PRESENT	Slices	Slice Flip-Flops	BRAMS	4-input LUTs	
16	16384	0	0	0	1	1613	1587	8	2180	75
16	32768	0	0	0	1	1545	1599	16	1892	82
16	32768	0	0	1	1	1559	1599	16	1923	78
16	32768	0	1	1	1	1560	1607	16	1928	78
16	32768	1	1	1	1	1569	1608	16	1947	70
16	32768	1	1	1	2	2404	2178	16	3202	67
16	32768	1	1	1	3	2598	2263	16	3717	64
32	32768	1	1	1	3	2642	2285	16	3844	67
64	32768	1	1	1	3	2741	2321	16	4053	65

Notes:
1. F.29 with overall effort level of 5 and extra effort level of 2 into a xcv2000e-8-FG1156 device with opb_clk period constrained to 11 nS (90.9 Mhz)
2. Please refer to Table 1 for a definition of these parameters

The EMAC benchmarks are shown in Table 26 for a Virtex™-II -6 FPGA.

Table 26: **EMAC FPGA Performance and Resource Utilization Benchmarks (Virtex™-II -6)**

Parameter Values						Device Resources				f_{MAX}
C_MAC_FIFO_DEPTH	C_IPIF_FIFO_DEPTH	C_DEV_MIR_ENABLE	C_RESET_PRESENT	C_INCLUDE_DEV_PENCODER	C_DMA_PRESENT	Slices	Slice Flip-Flops	BRAMS	4-input LUTs	(MHz)
16	16384	0	0	0	1	1581	1587	2	2158	128
16	32768	0	0	0	1	1555	1597	4	2102	127
16	32768	0	0	1	1	1571	1598	4	2130	126
16	32768	0	1	1	1	1572	1605	4	2144	126
16	32768	1	1	1	1	1586	1608	4	2165	123
16	32768	1	1	1	2	2402	2172	4	3246	89
16	32768	1	1	1	3	2599	2256	4	3758	100
32	32768	1	1	1	3	2644	2277	4	3880	92
64	32768	1	1	1	3	2743	2313	4	4091	93

Notes:
1. F.29 with overall effort level of 5 and extra effort level of 2 into a xc2v6000-6-ff1152 device with opb_clk period constrained to 8 nS (125 Mhz)
2. Please refer to Table 1 for a definition of these parameters

The EMAC benchmarks are shown in Table 27 for a Virtex™-II Pro -7 FPGA

Table 27: **EMAC FPGA Performance and Resource Utilization Benchmarks (Virtex™-II Pro -7)**

Parameter Values						Device Resources				f_{MAX} (MHz)
C_MAC_FIFO_DEPTH	C_IPIF_FIFO_DEPTH	C_DEV_MIR_ENABLE	C_RESET_PRESENT	C_INCLUDE_DEV_PENCODER	C_DMA_PRESENT	Slices	Slice Flip-Flops	BRAMS	4-input LUTs	
16	16384	0	0	0	1	1589	1586	2	2010	145
16	32768	0	0	0	1	1625	1599	4	2063	145
16	32768	0	0	1	1	1647	1598	4	2106	145
16	32768	0	1	1	1	1648	1607	4	2113	131
16	32768	1	1	1	1	1662	1611	4	2129	138
16	32768	1	1	1	2	2395	2174	4	3476	109
16	32768	1	1	1	3	2572	2250	4	3457	111
32	32768	1	1	1	3	2614	2271	4	3539	109
64	32768	1	1	1	3	2723	2307	4	3689	114

Notes:
1. F.29 with overall effort level of 5 and extra effort level of 2 into a xc2vp20-7-ff1152 device with opb_clk period constrained to 7 nS (142.8 Mhz)
2. Please refer to Table 1 for a definition of these parameters

Specification Exceptions

The EMAC design currently has no exceptions to the mandatory IEEE Std. 802.3 MII interface requirements.

Reference Documents

The following document contains reference information important to understanding the EMAC design:

- IEEE Std. 802.3

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
07/31/01	1.1	Initial Xilinx release
08/09/01	1.2	Update
11/02/01	1.3	Update

Date	Version	Revision
11/06/01	1.4	Update
11/08/01	1.5	Update
11/21/01	1.6	Update
02/14/02	1.7	Release to coincide with release of first design
02/22/02	1.8	Add in parameterized features not available in previously split synthesis design.
03/22/02	1.9	Update resources and max operating frequencies.
03/28/02	1.10	Correct several register errors
05/17/02	1.11	Update for revision i
05/21/02	1.12	Update for EDK 1.0
06/27/02	1.13	Reformatting of resource information and correction of device family support
07/22/02	1.14	Update for revision j
07/22/02	1.15	Add XCO parameters for System Generator
08/02/02	1.16	Modify list of features and correct description of control register for strip enable
10/31/02	1.17	Modify list of generics to remove reference to block memory or distributed memory FIFOs and to add IPIF FIFO depth generic
01/08/03	1.18	Update for EDK SP3
01/21/03	1.19	Correct PHY mii signal names
03/13/03	1.20	Modify for support and tested simple DMA mode
03/28/03	1.21	Update for revision l
04/15/03	1.22	Update for revision m
07/09/03	1.23	Update to new template
10/08/03	1.23.1	Update trademarks. Add references to rx_en, per CR 177368