

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](#).

Features

- 32-bit OPB master and slave interfaces¹
- Memory mapped direct I/O interface to registers and FIFOs as well as simple DMA and Scatter/Gather DMA
- Data Realignment and Checksum offloading capabilities for higher system performance when C_DMA_PRESET=3.
- 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. 16K or 32K byte depth is required for Jumbo frames up to 9K bytes long)

LogiCORE™ Facts		
Core Specifics		
Supported Device Family	QPro™-R Virtex-II™, QPro Virtex-II, Spartan-II™, Spartan-IIE, Spartan-3, Spartan-3E , Virtex, Virtex-II, Virtex-II Pro, Virtex-4, Virtex-E	
Version of Core	opb_ethernet	v1.04a
Resources Used		
	Min	Max
Total Core I/Os	181	181
Core FPGA IOBs	13	19
LUTs	1960	6157
FFs	1538	3198
Block RAMs	5	32
Provided with Core		
Documentation	Product Specification	
Design File Formats	NGC netlist, VHDL wrapper	
Constraints File	N/A	
Verification	N/A	
Instantiation Template	N/A	
Reference Designs	None	
Design Tool Requirements		
Xilinx Tools	8.1i or later	
Verification	N/A	
Simulation	ModelSim SE/EE 5.8d or later	
Synthesis	XST	
Support		
Support provided by Xilinx, Inc.		

1. The master interface is only used if either simple or scatter gather DMA is included in the core at build time. The core always includes a slave interface.

© 2005 Xilinx, Inc. All rights reserved. All Xilinx trademarks, registered trademarks, patents, and further disclaimers are as listed at <http://www.xilinx.com/legal.htm>. All other trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.
 NOTICE OF DISCLAIMER: Xilinx is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Xilinx makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Xilinx expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

Features (contd)

- 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
- Optional support of jumbo frames up to 9K bytes in length
- Supports unicast, multicast, and broadcast transmit and receive modes as well as promiscuous and 64 entry Contents Addressable Memory (CAM) based receive modes
- Auto source address field insertion or overwrite or pass through for transmission
- Auto pad field insertion on transmit
- Auto Frame Check Sequence (FCS) field insertion or pass through on transmit
- Auto pad and FCS field stripping or pass through 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
- Unaligned source and destination support
- Checksum offloading support

Evaluation Version

The EMAC LogiCORE is delivered with a hardware evaluation license. When programmed into a Xilinx device, the core will function in hardware for about 8 hours at the typical frequency of operation. To use the EMAC LogiCORE without this timeout limitation, a full license must be purchased.

Functional Description

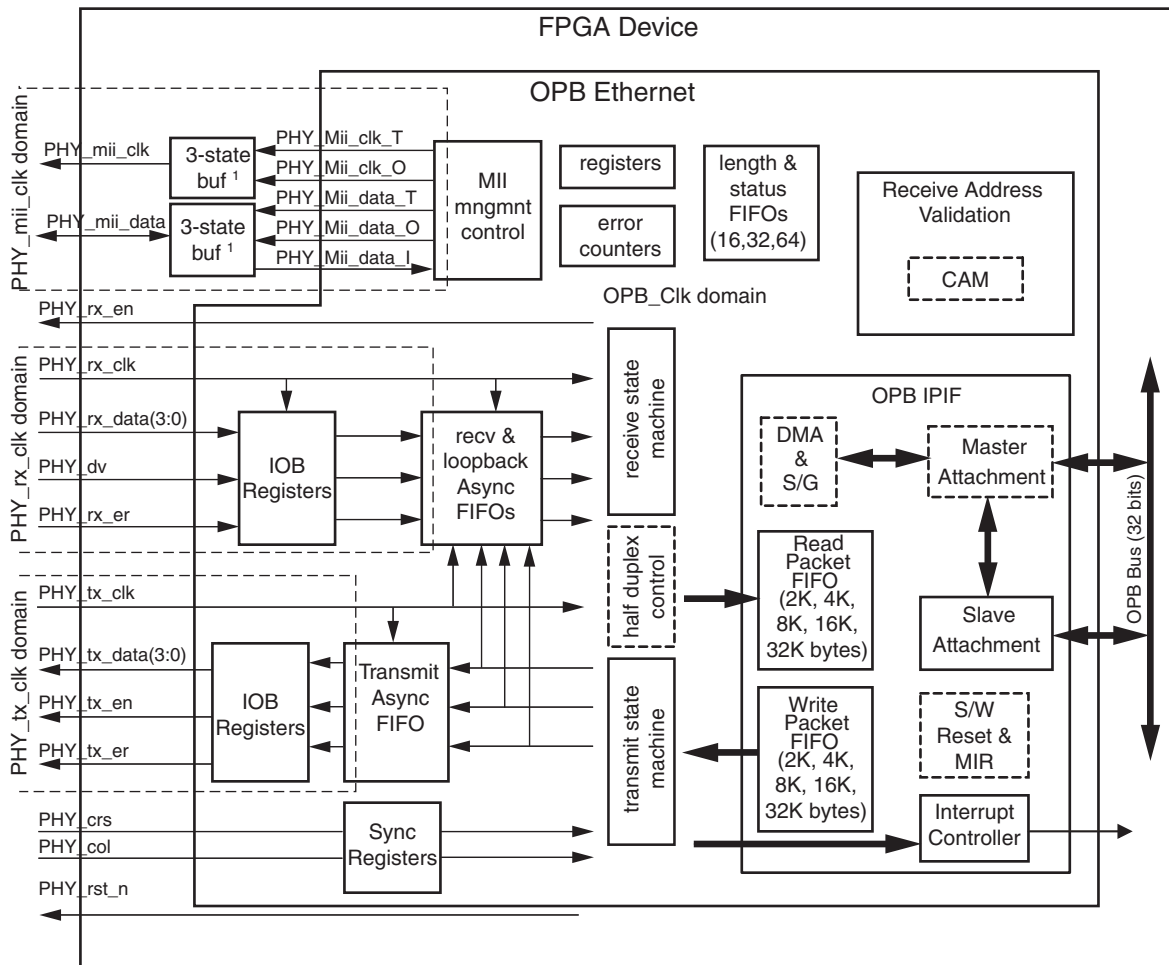
The EMAC Interface design is a soft intellectual property (IP) core designed for implementation in several Xilinx FPGAs. It supports the IEEE Std. 802.3 Media Independent Interface (MII) to industry standard Physical Layer (PHY) devices and communicates to a processor via an IBM On-Chip Peripheral Bus (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 packet¹ 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.

EMAC Endianness

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.



Notes:
 1. These buffers are automatically inserted by Platgen and are located in the opb_ethernet wrapper

ipif_emac_modules_blk_diagram.eps

Figure 1: IPIF and EMAC Modules

Ethernet Protocol

Ethernet data is encapsulated in frames as shown in Figure 2 for standard Ethernet frames, Figure 3 for VLAN frames, Figure 4 for jumbo frames, and Figure 5 for pause/flow control frames¹. The fields in the frame are

1. IEEE Std. 802.3 uses the terms Frame and Packet interchangeably when referring to the Ethernet unit of transmission; this specification does likewise
1. The EMAC design does not support the Ethernet 8-byte preamble frame type

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.

Destination Address

The destination address field is 6 bytes in length¹. 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 length². 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., no 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.

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

The maximum length of a data field is 1,500 bytes for normal (non-jumbo) frames. Therefore, a value in this field that exceeds 1,500 (05DC hex) would indicate 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. Jumbo frames can have a data field as large as 8982 bytes.

For reception, if the field is a length field and jumbo frames are disabled, 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 or jumbo frames are enabled, the EMAC will ignore the value and pass it along with the packet data with no further processing. A type/length field value of 8100 hex indicates that the frame is a VLAN frame and a value of 8808 hex indicates a pause MAC control frame.

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.

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 for a normal frame and up to 8982 bytes for a jumbo frame. 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 register¹.

If EMAC insertion of padding is enabled in the EMAC control register, the number of pad bytes to be inserted will be 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:

$$\text{PAD (bytes)} = 64 - [\text{TXLengthReg} + (\text{ENFCS} * 4) + (\text{ENSA} * 6)].$$

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 as²:

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

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

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 $x^{31}, x^{30}, \dots, x^1, x^0$). 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.

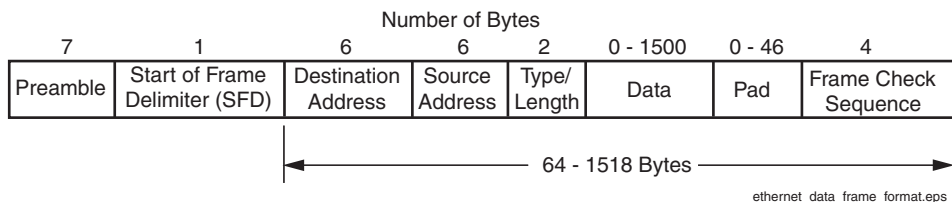


Figure 2: Ethernet Data Frame Format

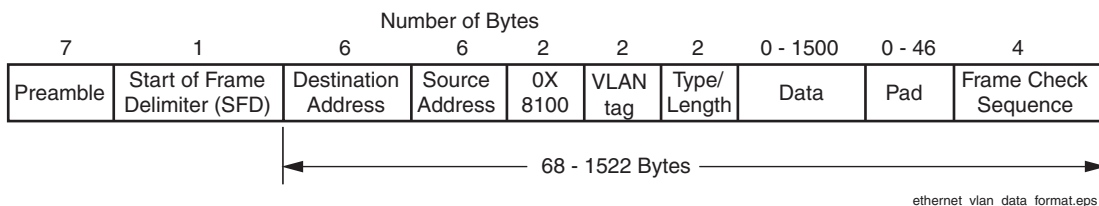


Figure 3: Ethernet VLAN Frame Format

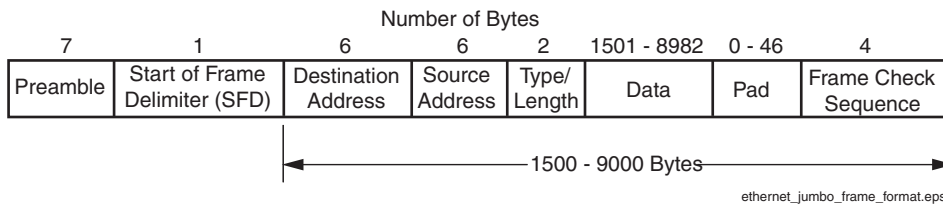


Figure 4: Ethernet Jumbo Frame Format

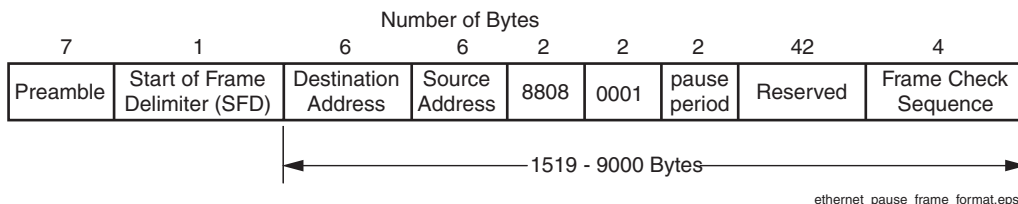


Figure 5: Ethernet Pause Frame Format

Interframe Gap¹ 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.

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

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

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 6** shows the high level flow followed for packet transmission.

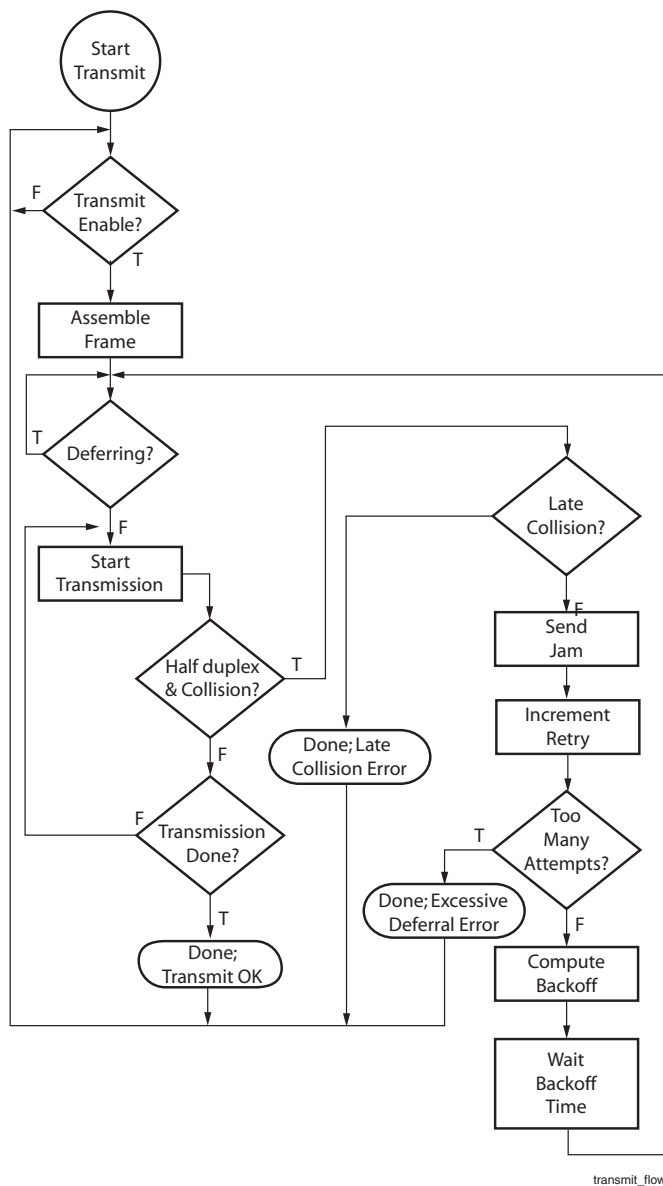


Figure 6: Transmit Flow

Receive Flow

The flow chart in **Figure 7** shows the high level flow followed for packet reception.

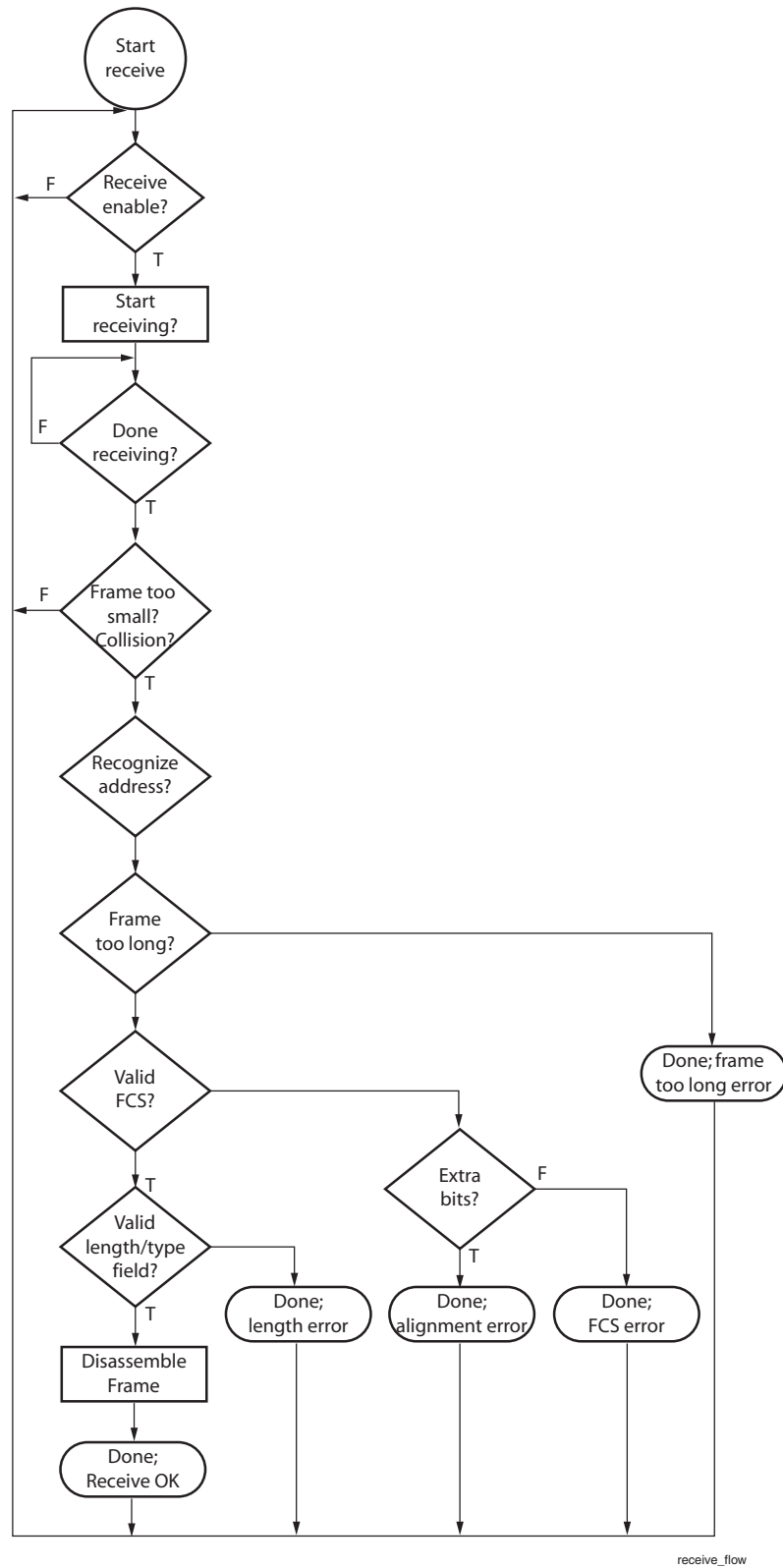


Figure 7: Receive Flow

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

Generic	Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
Top Level					
G1	Device Block Id	C_DEV_BLK_ID	0 - 255	1	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, spartan3, qvirtex2, qvirtex2, virtex2, virtex2p, virtex4	virtex2	string
G4	Device base address	C_BASEADDR	See Note 2	FFFFFFFF	std logic vector
G5	Device maximum address	C_HIGHADDR	See Note 2	00000000	std logic vector
G6	MAC length and status FIFO depth	C_MAC_FIFO_DEPTH	16, 32, 64	64	integer
G7	MAC length and status FIFO style	C_MAC_FIFO_BRAM_1_SRL_0	1 = use BRAMs 0 = use SRL logic	0	integer
G8	Include or exclude half duplex logic	C_HALF_DUPLEX_EXIST	1 = include half duplex logic 0 = exclude half duplex logic	1	integer
G9	Include or exclude jumbo frame support	C_JUMBO_EXIST	1 = support jumbo frames 0 = do not support jumbo frames	0	integer
G10	Include or exclude CAM receive address validation logic	C_CAM_EXIST	1 = include CAM logic 0 = exclude CAM logic	0	integer
G11	CAM style	C_CAM_BRAM_0_SRL_1	1 = use SRL logic 0 = use BRAMs	1	integer
OPB/IPIF Interface					
G12	Module Identification Read	C_DEV_MIR_ENABLE	1 = MIR reads Exists 0 = MIR reads Non-existent	1	integer

Table 1: EMAC Design Parameters (Contd)

Generic	Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G13	IPIF receive/read Packet FIFO depth in bits	C_IPIF_RDFIFO_DEPTH	262144 ⁽⁴⁾ , 131072, 65536, 32768 or 16384	32768	integer
G14	IPIF transmit/write Packet FIFO depth in bits	C_IPIF_WRFIFO_DEPTH	262144 ⁽⁴⁾ , 131072, 65536, 32768 or 16384	32768	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 ⁽³⁾	1 = no DMA function is required 2 = simple 2 ch DMA is required 3 = Scatter Gather DMA for packets is required	3	integer
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	32	integer
G20	OPB data bus width (in bits)	C_OPB_DWIDTH	32	32	integer
G21	Rx side DRE ⁽⁵⁾	C_RX_DRE_TYPE	0 = no DRE included 1 = DRE with Logic Mux 2 = DRE with DSP48 Mux	0	integer
G22	Tx side DRE ⁽⁵⁾	C_TX_DRE_TYPE	0 = no DRE included 1 = DRE with Logic Mux 2 = DRE with DSP48 Mux	0	integer

Table 1: EMAC Design Parameters (Contd)

Generic	Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G23	Rx side CSUM ⁽⁵⁾	C_RX_INCLUDE_CS UM	0 = no CSUM support 1 = CSUM support	0	integer
G24	Tx side CSUM ⁽⁵⁾	C_TX_INCLUDE_CS UM	0 = no CSUM support 1 = CSUM support	0	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. The default value will 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 0x4000 (for example, 0x10000000 and 0x10003FFF).
3. When C_DMA_PRESENT is '2' or '3' an OPB master interface is included in the core. When C_DMA_PRESENT is '1', no OPB master interface is used. The OPB slave interface is always present.
4. The largest size of C_IPIF_FIFO_DEPTH is available for Virtex-II and Virtex-II Pro devices only.
5. DRE and CSUM are only available when C_DMA_PRESENT=3.

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.

Detailed Parameter Descriptions

C_DEV_BLK_ID

The block ID is reflected as a field in the Module Identification Registers (MIR). This may be used for identification and verification of correct register access capability.

C_OPB_CLK_PERIOD_PS

This clock period information is used by the IPIF sub-module for calculating the interrupt coalescing period. Interrupt coalescing is optional and is only used when using Scatter Gather DMA.

C_FAMILY

The family parameter is required to implement the core using family specific architecture features. This parameter is automatically updated by the EDK tools based on the project target device information.

C_BASEADDR and C_HIGHADDR

These values are used to generate the read and write enables for the FIFOs and registers and are byte addresses. The address range defined by these parameters must be at least 0x4000. For example, if the C_BASEADDR is set to 0x10000000, then C_HIGHADDR must be set to at least 0x10003FFF. These parameters must be initialized since the default values have been selected so that they will generate an error during build if left unchanged.

C_MAC_FIFO_DEPTH

This parameter is used to select a depth for the transmit status, transmit length, and receive length registers. These registers are actually FIFOs and the depth represents the maximum number of entries that may be queued up before an overflow condition occurs.

Selecting a larger depth will use more resources but offers the potential for higher bandwidth traffic on the Ethernet bus while reducing processor overhead. The number of transmit and receive packets you will be able to

queue up will be limited by the number of entries in these FIFOs or by the size of the packet FIFOs whichever fills up first.

If the packet sizes will be small, these FIFOs will fill up more quickly and should be as large as possible and conversely, if the packet sizes are large, the packet FIFOs will fill up first and should be as large as possible. If the packet size is unknown or is distributed over all sizes than these FIFOs and the packet FIFOs should both be made as large as possible.

C_MAC_FIFO_BRAM_1_SRL_0

This parameter is used to implement the status and length register FIFOs in either BRAM or in logic using SRL FIFOs. This is strictly a trade off between logic and BRAM resource utilization and doesn't affect performance appreciably although it may have an impact on place and route times depending on the fullness of the target device.

C_HALF_DUPLEX_EXIST

This parameter controls the inclusion or exclusion of the special circuitry required for the core to operate in a half duplex environment. The amount of resources used by this circuitry is significant and half duplex systems are not very common and becoming less so.

When the half duplex circuitry is included, selection of half duplex or full duplex mode is controlled by a bit in the EMAC control register. When excluded, the bit in the control register will be fixed to full duplex only regardless of the value written to that location.

C_JUMBO_EXIST

This parameter modifies the core to allow the use of jumbo sized frames. "Normal frames" are limited to a maximum of 1518 bytes. "Jumbo frames" increases the frame size to a maximum of 9000 bytes. Why 9000? Ethernet uses a 32 bit CRC that loses its effectiveness above about 12000 bytes. Additionally, 9000 is large enough to carry an 8 KB datagram, a common size used for Network Files Storage (NFS) plus packet header overhead. Since the opb_ethernet design is driven by powers of 2, the design is actually capable of processing packets up to 16383 bytes when jumbo frames are enabled.

Jumbo frames is not part of the Ethernet standard and therefor is not supported by all Ethernet devices equally.

In general, increasing the size of the frame increases efficiency on the Ethernet bus and increases the theoretical maximum bandwidth of the Ethernet bus. In an opb_ethernet system it also tends to reduce processor overhead.

Enabling Jumbo frames with this parameter increases the size of some counters and the receive and transmit length register FIFOs. This does increase resource usage and as a result should not be selected if jumbo frames will not be used.

When jumbo frames are enabled with this parameter, a bit in the EMAC control register further enables or disables the processing of receive jumbo frames. Transmitting jumbo frames is always possible when this parameter is enabled.

C_CAM_EXIST

This parameter include or excludes a Contents Addressable Memory (CAM) which can be used to store up to 64 MAC addresses for receive packet address validation. Including this feature can significantly increase resource utilization. Please refer to the section [Receive Address Validation](#) for more information on the functionality of this feature.

C_CAM_BRAM_0_SRL_1

This parameter is used to implement the CAM in either BRAM or in logic using SRLs. This is strictly a trade off between logic and BRAM resource utilization and doesn't affect performance appreciably although it may have an impact on place and route times depending on the fullness of the target device.

C_DEV_MIR_ENABLE

This parameter includes or excludes the Module Identification Registers in the IPIF sub module. This includes the device MIR, packet FIFO MIRs, and the DMA MIR if the DMA module is included in the design. The EMAC MIR is always included. This will reduce resource utilization in those cases where the MIR functionality is not required.

The software reset function (which shares an address with the Device MIR register) is not dependent on this parameter and may be present even when the MIR registers are not. However, the Device MIR register is dependent on the C_RESET_PRESENT parameter and will not be available if the software reset function is excluded from the design.

C_IPIF_RDFIFO_DEPTH and C_IPIF_WRFIFO_DEPTH

These parameters set the depth (in bits) of the packet FIFOs. The depth of the FIFOs impacts the number of BRAMS used in the system and only has a minor impact on logic resources used.

Like the C_MAC_FIFO_DEPTH parameter, a deeper packet FIFO will use more resources but offers the potential for higher bandwidth traffic on the Ethernet bus while reducing processor overhead.

The minimum FIFO size (16384 bits = 2048 bytes) is large enough to hold one maximum "normal" frame of 1518 bytes. However, increasing the packet FIFO to hold at least two frames can significantly increase performance. A minimum packet FIFO size of 131,072 bits = 16,384 bytes is required to hold one maximum size "jumbo" frame.

In cases where packets are primarily small it may not be useful to have a large packet FIFO. For example, if all of the packets are 64 bytes, a maximum depth (64 entry) length and status FIFO will limit queued packets to 4096 total bytes in the packet FIFO.

Note that the largest size packet FIFOs (262,144 bits = 32,768 bytes) are only available in Virtex-II and newer devices (those devices with 18 Kb block RAMs).

C_RESET_PRESENT

The parameter includes or excludes the software reset function. This may allow some reduction in resources if the software reset function is not required.

The software reset function (which shares an address with the Device MIR register) is not dependent on the C_DEV_MIR_ENABLE parameter and may be present even when the MIR registers are not. However, the Device MIR register is dependent on the C_RESET_PRESENT parameter and will not be available if the software reset function is excluded from the design.

C_INCLUDE_DEV_PENCODER

This parameter includes or excludes the Device Interrupt Identification Register. This may allow some reduction in resources if the Device Interrupt Identification Register is not required.

For more information about interrupts please read the following sections and refer to the *Processor IP Reference Guide* under Part 1: Embedded Processor IP, under IPIF, under OPB IPIF Architecture, under OPB IPIF Interrupt for a complete description of the IPIF interrupt processing.

C_DMA_PRESENT

This parameter controls the inclusion of simple DMA (2), scatter gather DMA (3) or no DMA (1) functionality. Choosing to include either DMA also automatically includes an OPB master bus interface.

Including DMA can significantly increase resources but offers the potential for higher bandwidth traffic on the Ethernet bus while reducing processor overhead and increasing bandwidth on the processor bus.

For more information about DMA please refer to the *Processor IP Reference Guide* under Part 1: Embedded Processor IP, under IPIF, under OPB IPIF Architecture, under Direct Memory Access and Scatter Gather.

C_DMA_INTR_COALESCE

This parameter includes or excludes the interrupt coalesce functionality. This may allow some reduction in resources if the interrupt coalesce function is not required.

Interrupt coalescing is only available when scatter gather DMA is being used and allows multiple interrupt events to queue up before being processed while guaranteeing a maximum wait bound. This can reduce processor overhead by reducing the number of interrupts processed.

For more information about interrupt coalescing and scatter gather DMA please read the following sections and refer to the *Processor IP Reference Guide* under Part 1: Embedded Processor IP, under IPIF, under OPB IPIF Architecture, under Direct Memory Access and Scatter Gather.

C_TX_DRE_TYPE and C_RX_DRE_TYPE

This parameter includes or excludes the Data Realignment Engine which allows unaligned source and destination addresses originating from the DMA controller. When this value is set to a (0) then there is no DRE functionality included. When set to (1) then the DRE block is built entirely out of logic. If the user wishes to save resources in exchange for DSP48s set this parameter to (2) and the muxing will be done via DSP48 blocks (2 DSP48s are consumed per TX/RX). This feature is only available when C_DMA_PRESENT=3.

C_TX_INCLUDE_CSUM and C_RX_INCLUDE_CSUM

This parameter includes or excludes the Checksum offloading blocks which assist software through hardware acceleration. When this value is set to false then there is no CSUM capabilities included in the core and when this value is true and C_DMA_PRESENT=3 then the CSUM logic is included.

C_OPB_AWIDTH and C_OPB_DWIDTH

These parameters should always be set to 32.

C_MIIM_CLKDVD

C_SOURCE_ADDR_INSERT_EXIST

C_PAD_INSERT_EXIST

C_FCS_INSERT_EXIST

C_ERR_COUNT_EXIST

C_MII_EXIST

While these parameters are present in the design they should not be modified from their default values. These features have not been fully tested with their secondary values and are also not supported by the software drivers.

EMAC I/O Signals

The external I/O signals for the EMAC are listed in [Table 2](#).

Table 2: EMAC I/O Signals

Generic	Signal Name	Interface	I/O	Initial State	Description
EMAC Signals					
P1	PHY_rx_data(3:0)	Ethernet IOB	I		Ethernet receive data. Input from FPGA I/O
P2	PHY_tx_data(3:0)	Ethernet IOB	O	0000	Ethernet transmit data. Output to FPGA I/O
P3	PHY_dv	Ethernet IOB	I		Ethernet receive data valid. Input from FPGA I/O
P4	PHY_rx_er	Ethernet IOB	I		Ethernet receive error. Input from FPGA I/O
P5	PHY_tx_en	Ethernet IOB	O	0	Ethernet transmit enable. Output to FPGA I/O
P6	PHY_rx_en	Ethernet IOB	O	0	Ethernet receive enable controlled by control register bit 4. Output to FPGA I/O
P7	PHY_tx_er	Ethernet IOB	O	0	Ethernet transmit error. Output to FPGA I/O
P8	PHY_tx_clk	Ethernet IOB	I		Ethernet transmit clock input from FPGA I/O
P9	PHY_rx_clk	Ethernet IOB	I		Ethernet receive clock input from FPGA I/O
P10	PHY_crs	Ethernet IOB	I		Ethernet carrier sense input from FPGA I/O
P11	PHY_col	Ethernet IOB	I		Ethernet collision input from FPGA I/O
P12	PHY_rst_n	Ethernet IOB	O	0	Ethernet PHY reset output to FPGA I/O
P13	PHY_mii_clk_I	Ethernet IOB	I		MII management interface clock input to 3-state input buffer (not used)
P14	PHY_mii_clk_O	Ethernet Buffer	O	0	MII management interface clock output to 3-state output buffer
P15	PHY_mii_clk_T	Ethernet Buffer	O	0	MII management interface clock enable output to 3-state output buffer
P16	PHY_mii_data_I	Ethernet Buffer	I		MII management interface data input from 3-state I/O buffer
P17	PHY_mii_data_O	Ethernet Buffer	O	0	MII management interface data output to 3-state I/O buffer

Table 2: EMAC I/O Signals (Contd)

Generic	Signal Name	Interface	I/O	Initial State	Description
P18	PHY_mii_data_T	Ethernet Buffer	O	0	MII management interface data enable output to 3-state I/O buffer
OPB Signals					
P19	Sln_DBus(0:C_OPB_DWIDTH-1)	OPB bus	O	0	EMAC slave output data bus
P20	Sln_xferAck	OPB bus	O	0	EMAC slave transfer acknowledge
P21	Sln_Retry	OPB bus	O	0	EMAC slave retry
P22	Sln_ToutSup	OPB bus	O	0	EMAC slave timeout suppress
P23	Sln_ErrAck	OPB bus	O	0	EMAC slave error acknowledge
P24	OPB_ABus(0:C_OPB_AWIDTH-1)	OPB bus	I		OPB address bus
P25	OPB_BE(0:3)	OPB bus	I		OPB byte enables
P26	OPB_DBus(0:C_OPB_DWIDTH-1)	OPB bus	I		OPB data bus
P27	OPB_RNW	OPB bus	I		Read not Write (OR of all master RNW signals)
P28	OPB_select	OPB bus	I		Master has taken control of the bus (OR of all master selects)
P29	OPB_seqAddr	OPB bus	I		OPB sequential address
P30	OPB_errAck	OPB bus	I		OPB error acknowledge
P31	OPB_MnGrant	OPB bus	I		OPB master bus grant
P32	OPB_retry	OPB bus	I		OPB retry
P33	OPB_timeout	OPB bus	I		OPB timeout error
P34	OPB_xferAck	OPB bus	I		OPB transfer acknowledge
P35	Mn_request	OPB bus	O	0	EMAC master bus request
P36	Mn_busLock	OPB bus	O	0	EMAC master bus arbitration lock
P37	Mn_select	OPB bus	O	0	EMAC master select
P38	Mn_RNW	OPB bus	O	0	EMAC master Read not Write
P39	Mn_BE(0:3)	OPB bus	O	0	EMAC master byte enables
P40	Mn_seqAddr	OPB bus	O	0	EMAC master sequential address

Table 2: EMAC I/O Signals (Contd)

Generic	Signal Name	Interface	I/O	Initial State	Description
P41	Mn_ABus(0:C_OPB_AWIDTH-1)	OPB bus	O	0	EMAC master address bus
System					
P42	OPB_Clk	System	I		System clock
P43	OPB_Rst	System	I		System Reset (active high)
P44	IP2INTC_Irpt	System	O	0	System Interrupt
P45	Freeze	System	I		System Freeze Input
P46	emac_intrpts(0:21)	Genera	O'	0x0C0000	EMAC interrupts

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

Generic	Name	Affects	Depends	Relationship Description
	Design Parameters			
G20	C_OPB_DWIDTH	P18, P26		Specifies the Data Bus width
G19	C_OPB_AWIDTH	P24, P41		Specifies the Address Bus width
G17	C_DMA_PRESENT	G18		Specifies if DMA is present and which type
G18	C_DMA_INTR_COASL ESCE		G17	Not used if scatter gather DMA not present (G17 is 0, 1, 2)
	I/O Signals			
P19	SIn_DBus(0:C_OPB_DWIDTH-1)		G20	Width varies with the size of the Data bus.
P24	OPB_ABus(0:C_OPB_AWIDTH-1)		G19	Width varies with the size of the Address bus.
P26	OPB_DBus(0:C_OPB_DWIDTH-1)		G20	Width varies with the size of the Data bus.
P41	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 provided as outputs as well as being 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 Processor IP Reference Guide under Part 1: Embedded Processor IP, under IPIF, under OPB IPIF Architecture, under OPB IPIF Interrupt for a complete description of the IPIF interrupt processing. An overview diagram of the interrupt control structure is in [Figure 8](#).

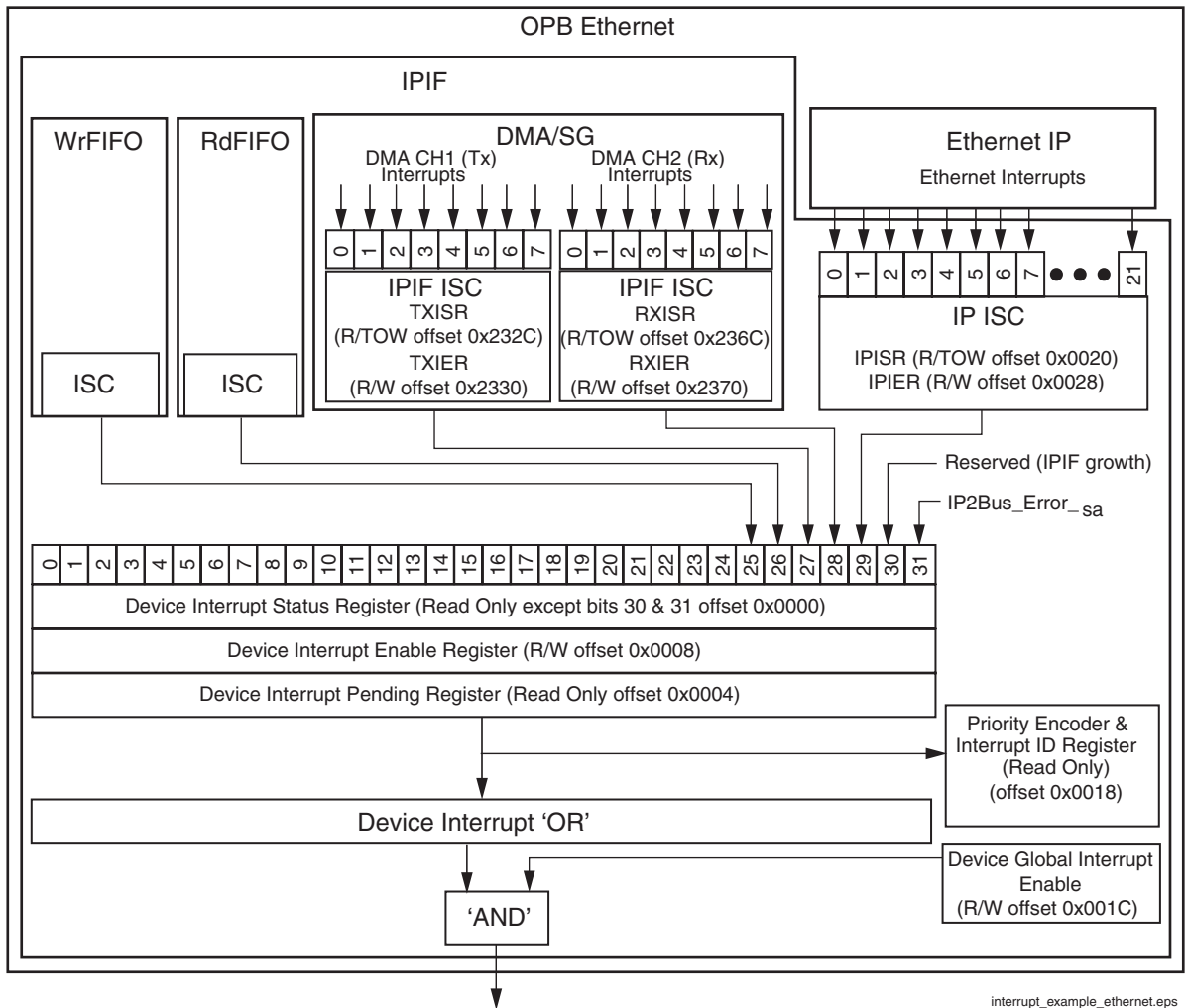


Figure 8: Interrupt Control Structure

The EMAC Interrupts values are available by reading the IP Interrupt status register (offset 0x0020); the interrupt values are shown in Figure 9 and described in Table 4.

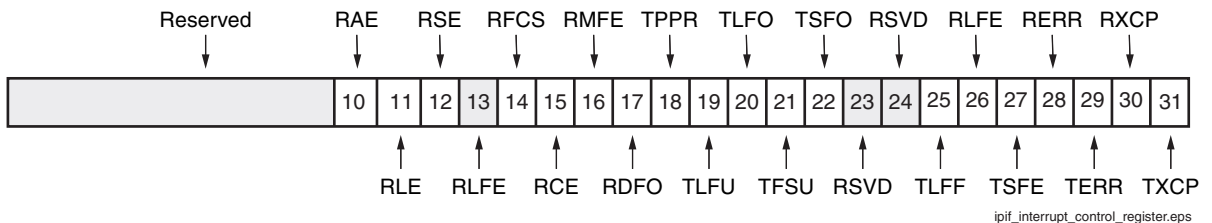


Figure 9: IPIF Interrupt Status Register (IPISR)

Table 4: EMAC IP Interrupt Status Register Bit Definitions (offset 0x0020)

Bit Location	Name	Core Access	Reset Value	Description
0 - 9	RSVD	Read	"0000000000"	Unused
10	RAE	Read/TOW	'0'	Receive Alignment Error interrupt
11	RLE	Read/TOW	'0'	Receive Long Error interrupt
12	RSE	Read/TOW	'0'	Receive Short Error interrupt
13	RLFE	Read/TOW	'0'	Receive Length Field Error interrupt
14	RFCS	Read/TOW	'0'	Receive FCS Error interrupt
15	RCE	Read/TOW	'0'	Receive Collision Error interrupt
16	RMFE	Read/TOW	'0'	Receive Missed Frame Error interrupt
17	RDFO	Read/TOW	'0'	Receive Data FIFO Overrun interrupt
18	TPPR	Read/TOW	'0'	Transmit Pause Packet Received interrupt
19	TLFU	Read/TOW	'0'	Transmit Length FIFO Underrun interrupt
20	TLFO	Read/TOW	'0'	Transmit Length FIFO Overrun interrupt
21	TSFU	Read/TOW	'0'	Transmit Status FIFO Underrun interrupt
22	TSFO	Read/TOW	'0'	Transmit Status FIFO Overrun interrupt
23	RSVD	Read	'0'	Reserved
24	RSVD	Read	'0'	Reserved
25	TLFF	Read/TOW	'0'	Transmit Length FIFO Full interrupt
26	RLFE	Read/TOW	'1'	Receive Length FIFO Empty interrupt
27	TSFE	Read/TOW	'1'	Transmit Status FIFO Empty interrupt
28	RERR	Read/TOW	'0'	Receive Error interrupt
29	TERR	Read/TOW	'0'	Transmit error interrupt
30	RXCP	Read/TOW	'0'	Receive complete interrupt
31	TXCP	Read/TOW	'0'	Transmit complete interrupt

Detailed Interrupt Descriptions

Interrupt (data bus bit 31, `emac_intrpts(0)`) -- Transmit complete interrupt

Indicates that at least one transmit has completed and that the transmit status word is available. This interrupt is actually Transmit Status Register FIFO not empty. This interrupt will stay set as long as at least one status word has not been read from the transmit status word register FIFO.

Interrupt (data bus bit 30, `emac_intrpts(1)`) -- 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. This interrupt is actually Receive Length Register FIFO not empty. This interrupt will stay set as long as at least one length value has not been read from the receive length register FIFO.

Interrupt (data bus bit 29, emac_intrpts(2)) -- 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, emac_intrpts(3)) -- 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, emac_intrpts(4)) -- 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, emac_intrpts(5)) --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, emac_intrpts(6)) -- 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, emac_intrpts(7)) -- Reserved

Interrupt (data bus bit 23, emac_intrpts(8)) -- Reserved

Interrupt (data bus bit 22, emac_intrpts(9)) -- 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, emac_intrpts(10)) -- 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, emac_intrpts(11)) -- 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, emac_intrpts(12)) -- 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, emac_intrpts(13)) -- Transmit Pause Packet Received interrupt

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

Interrupt (data bus bit 17, emac_intrpts(14)) -- 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, emac_intrpts(15)) -- 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, emac_intrpts(16)) -- 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, emac_intrpts(17)) -- 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, emac_intrpts(18)) -- 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, emac_intrpts(19)) -- 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, emac_intrpts(20)) -- 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, emac_intrpts(21)) -- 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 listed in [Table 5](#) 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 the [Reference Documents](#) section. The presence of the DMA registers varies based on the setting of the C_DMA_PRESENT parameter. The Device Interrupt Identification Register will not be present if the C_INCLUDE_DEV_PENCODER is set to '0'.

The registers listed in [Table 6](#) 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 5: 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
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 Unserviced Packet Count	C_DEV_BASEADDR + 0x2360	Read/Write
Receive DMA & Scatter Gather Packet Count Threshold	C_DEV_BASEADDR + 0x2364	Read/Write

Table 5: EMAC IPIF Registers (Contd)

Register Name	OPB ADDRESS	Access
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 Push	C_DEV_BASEADDR + 0x2008	Write
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 Pop	C_DEV_BASEADDR + 0x2018	Write
Read Packet FIFO data read port	C_DEV_BASEADDR + 0x2200 thru 0x29FF	Read
Device Interrupt Status Register	C_DEV_BASEADDR + 0x0000	Read
Device Interrupt Pending Register	C_DEV_BASEADDR + 0x0004	Read
Device Interrupt Enable Register	C_DEV_BASEADDR + 0x0008	Read/Write
Device Interrupt Identification Register	C_DEV_BASEADDR + 0x0018	Read
Device Global Interrupt Enable	C_DEV_BASEADDR + 0x001C	Read/Write
IP Interrupt Status Register	C_DEV_BASEADDR + 0x0020	Read/Toggle on 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 6.

Table 6: 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

Table 6: EMAC Core Registers (Contd)

Register Name	OPB ADDRESS	Access
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
Receive Status Reg (RSR)	C_DEV_BASEADDR + 0x113C	Read
Receive CAM Entry High Reg (CAMH)	C_DEV_BASEADDR + 0x1140	Read/Write
Receive CAM Entry Low Reg (CAML)	C_DEV_BASEADDR + 0x1144	Read/Write

EMAC Module Identification Register (EMIR offset 0x1100)

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

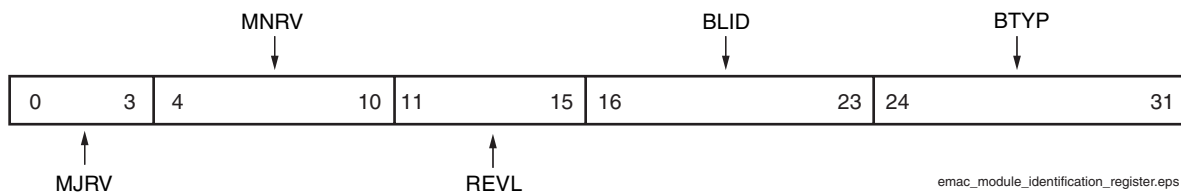


Figure 10: EMAC Module Identification Register (EMIR)

Table 7: EMAC Module Identification Register Bit Definitions

Bits	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 "0000010" for this minor version of 2	Module Minor Version Number.

Table 7: EMAC Module Identification Register Bit Definitions (Contd)

Bits	Name	Core Access	Reset Value	Description
11 - 15	Rev. Letter (REVL)	Read	Version ID "00000" for this revision of "a"	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 offset 0x1104)

The EMAC Control Register shown in Figure 11 and described in Table 8 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').

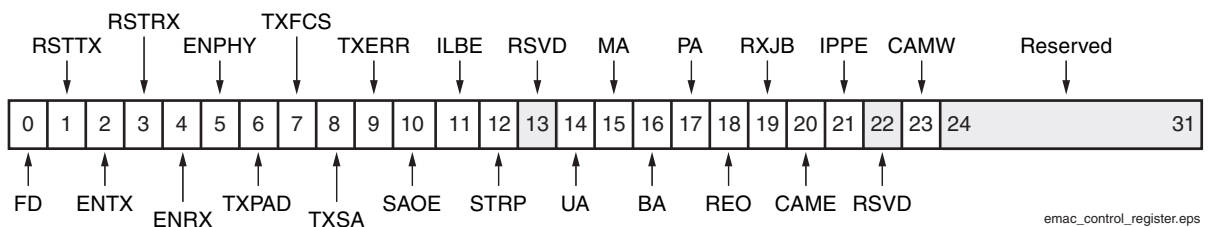


Figure 11: EMAC Control Register (ECR)

Table 8: EMAC Control Register Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
0	FD	Read/Write	'0'	<p>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'.</p> <ul style="list-style-type: none"> '0' - Half Duplex '1' - Full Duplex
1	RSTTX	Read/Write	'1'	<p>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'.</p> <ul style="list-style-type: none"> '0' - Normal Operation '1' - Reset
2	ENTX	Read/Write	'0'	<p>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).</p> <ul style="list-style-type: none"> '0' - Disable Transmitter '1' - Enable Transmitter
3	RSTRX	Read/Write	'1'	<p>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'.</p> <ul style="list-style-type: none"> '0' - Normal Operation '1' - Reset
4	ENRX	Read/Write	'0'	<p>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 controls the output signal PHY_rx_en.</p> <ul style="list-style-type: none"> '0' - Disable Receiver '1' - Enable Receiver

Table 8: EMAC Control Register Bit Definitions (Contd)

Bit(s)	Name	Core Access	Reset Value	Description
5	ENPHY	Read/Write	'1'	<p>Enable PHY. This value of this bit is driven to the PHY interface PHY_rst_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'. PHY_rst_n will also be driven active while OPB_Rst is active regardless of value of this bit. PHY_rst_n can also be driven active by writing to the Device Software Reset Register if the software reset function has been included in the build. The software reset function is not dependent on the value of this bit.</p> <ul style="list-style-type: none"> '0' - Disable / Reset PHY '1'- Enable PHY
6	TXPAD	Read/Write	'1'	<p>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'.</p> <ul style="list-style-type: none"> '0' - Disable Auto Pad Insertion '1'- Enable Auto Pad Insertion
7	TXFCS	Read/Write	'1'	<p>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'.</p> <ul style="list-style-type: none"> '0' - Disable Auto FCS Insertion '1'- Enable Auto FCS Insertion
8	TXSA	Read/Write	'1'	<p>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'.</p> <ul style="list-style-type: none"> '0' - Disable Auto Source Address Insertion '1'- Enable Auto Source Address Insertion

Table 8: EMAC Control Register Bit Definitions (Contd)

Bit(s)	Name	Core Access	Reset Value	Description
9	TXERR	Read/Write	'0'	<p>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'.</p> <ul style="list-style-type: none"> '0' - Disable Error Insertion '1' - Enable Error Insertion
10	SAOE	Read/Write	'1'	<p>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'.</p>
11	ILBE	Read/Write	'0'	<p>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.</p>
12	STRP	Read/Write	'0'	<p>Pad & FCS Strip Enable. Enables stripping of receive pad and FCS fields when type/length field is a length.</p> <ul style="list-style-type: none"> '0' - Disable Strip '1' - Enable Strip
13	Reserved	Read	'0'	<ul style="list-style-type: none"> Reserved. This bit is reserved for future use.
14	UA	Read/Write	'1'	<p>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'.</p> <ul style="list-style-type: none"> '0' - Disable Unicast Address '1' - Enable Unicast Address
15	MA	Read/Write	'0'	<p>Enable Multicast Address. Enables the EMAC to accept valid frames that have a multicast (excluding broadcast) destination address field. This bit should not be modified while transmit and receive are enabled ECR.ENTX = '1' and/or ECR.ENRX = '1'.</p> <ul style="list-style-type: none"> '0' - Disable Multicast Address '1' - Enable Multicast Address

Table 8: EMAC Control Register Bit Definitions (Contd)

Bit(s)	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'. <ul style="list-style-type: none"> '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'. <ul style="list-style-type: none"> '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 <ul style="list-style-type: none"> '0' - Disable Error Override '1' - Enable Error Override
19	RXJB	Read/Write	"0"	Receive Jumbo Frames Enable. Enables the EMAC to receive jumbo size frames. This bit can not be set if the jumbo functionality was not included using the parameter C_JUMBO_EXIST. <ul style="list-style-type: none"> '0' - Disable Rx Jumbo Frames '1' - Enable Rx Jumbo Frames
20	CAME	Read/Write	"0"	Receive CAM Enable. Enables the EMAC to use the 64 entry CAM for receive address validation. This bit can not be set if the CAM was not included using the parameter C_CAM_EXIST. <ul style="list-style-type: none"> '0' - Disable Rx CAM '1' - Enable Rx CAM
21	IPPE	Read/Write	'0'	Interpret Pause Packets. Enables the EMAC to process valid received pause packets. <ul style="list-style-type: none"> '0' - Disable Pause Packets '1' - Enable Pause Packets
22	Reserved	Read	'0'	Reserved. These bits are reserved for future use.
23	CAMW	Read/Write	'0'	Receive CAM Write. This bit Writes the contents of the CAMH and CAML registers into the CAM address also provided in CAMH. This bit will automatically reset to '0'. Setting this bit will have no effect if the CAM was not included using the parameter C_CAM_EXIST. <ul style="list-style-type: none"> '0' - CAM Write Idle '1' - Initiate a CAM Write
24-31	Reserved	Read	0x000	Reserved. These bits are reserved for future use.

Interframe Gap Register (IFGP offset 0x1108)

The Interframe Gap Register shown in Figure 12, and described in Table 9, 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').

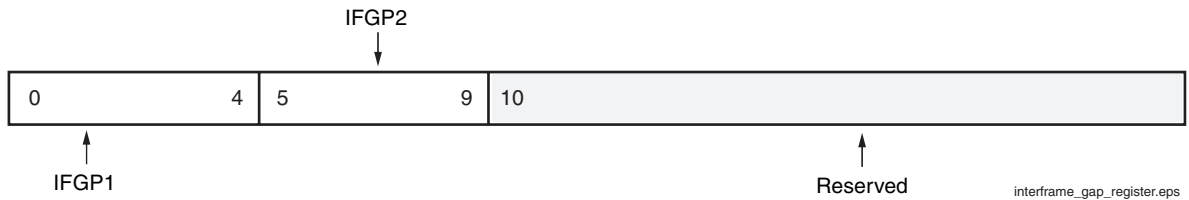


Figure 12: Interframe Gap Register (IFGP)

Table 9: Interframe Gap Register Bit Definitions

Bit(s)	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 offset 0x111C)

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 width of this register varies based on the value of the parameter C_JUMBO_EXIST. If C_JUMBO_EXIST = '1', bits 18 through 31 are used to store the length value where as bits 21 through 31 are used when C_JUMBO_EXIST = '0'.

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. The interrupt Receive Complete is set when at least one entry is stored in this FIFO from a successful packet reception. An interrupt Receive Length FIFO Empty also is available. Reading this register causes the current value to be removed from the FIFO. The Receive Packet Length register is shown in **Figure 13** and described in **Table 10**.

Reading of this register will Flush the RX_DRE of any residue data if C_RX_DRE_TYPE>0.

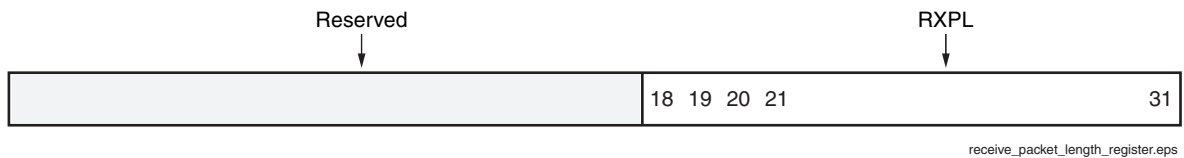


Figure 13: Receive Packet Length Register (RPLR)

Table 10: Receive Packet Length Register Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
0-17 or 0-20	Reserved	Read	0x00000	Reserved. These bits are reserved for future use.
18-31 or 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 offset 0x1120)

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 width of this register varies based on the value of the parameter C_JUMBO_EXIST. If C_JUMBO_EXIST = ‘1’, bits 18 through 31 are used to store the length value where as bits 21 through 31 are used when C_JUMBO_EXIST = ‘0’.

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.

The Transmit Packet Length register, shown in **Figure 14** and described in **Table 11**, 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.

Writing to this register will cause an automatic Write Push to happen on the Write Packet FIFO if $C_TX_DRE_TYPE > 0$.

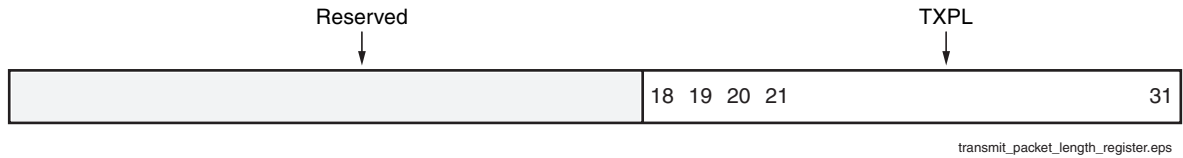


Figure 14: Transmit Packet Length Register (TPLR)

Table 11: Transmit Packet Length Register Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
0-17 or 0-20	Reserved	Read	0x00000	Reserved. These bits are reserved for future use.
18-31 or 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 offset 0x113C)

The receive status register, shown in Figure 15 and described in Table 12, 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.

Note that when $C_RX_INCLUDE_CSUM$ is true this value is written into the RDPFIFO along with the RAW checksum.

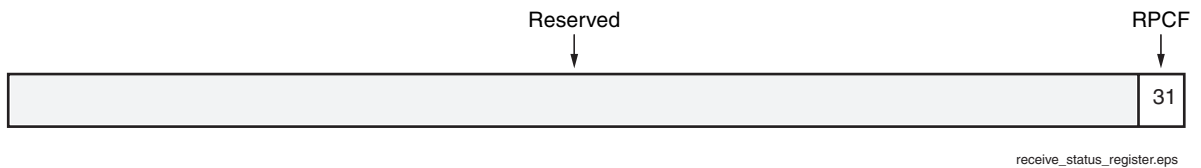


Figure 15: Receive Status Register (RSR)

Table 12: Receive Status Register Bit Definitions

Bit(s)	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 offset 0x1124)

The transmit status register, shown in Figure 16 and described in Table 13, 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.

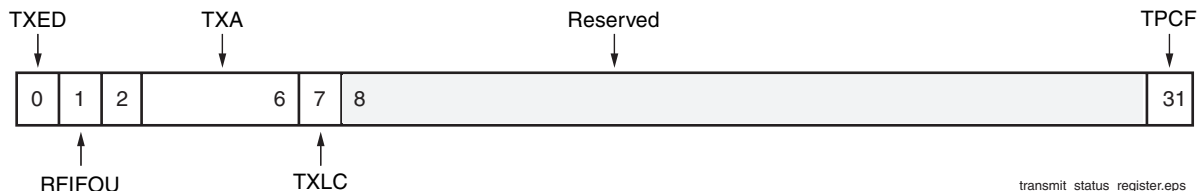


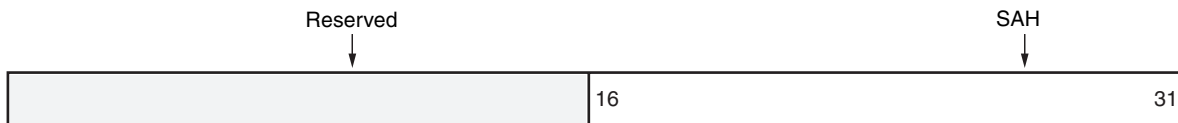
Figure 16: Transmit Status Register (TSR)

Table 13: Transmit Status Register Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
0	TXED	Read	'0'	<p>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.</p> <ul style="list-style-type: none"> '0' - No excess deferrals occurred since the last read '1' - At least one excess deferral has occurred
1	PFIFOU	Read	'0'	<p>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.</p> <ul style="list-style-type: none"> '0' - No packet FIFO underruns occurred since the last read '1' - At least one packet FIFO underrun has occurred
2- 6	TXA	Read	0x00	<p>Transmission Attempts. The number of transmission attempts made. There will be a maximum of 16 attempts.</p>
7	TXLC	Read	'0'	<p>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.</p> <ul style="list-style-type: none"> '0' - No late collisions occurred '1' - Late collision occurred
8 - 30	Reserved	N/A	0x000000	<p>Reserved. These bits are unused and will always return all zeros.</p>
31	TPCF	Read	'1'	<p>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.</p>

Station Address High Register (SAH offset 0x110C)

The station address high register, shown in Figure 17 and described in Table 14, contains the high-order 16 bits of the 48 bit station address.



station_address_high_register.eps

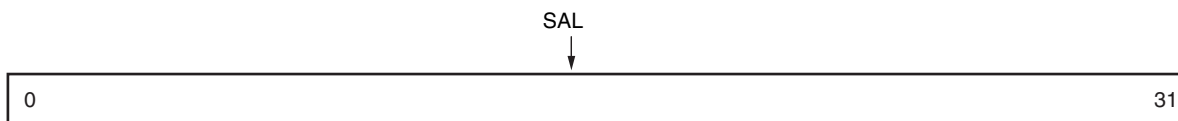
Figure 17: Station Address High (SEH) Register

Table 14: Station Address High Register Bit Definitions

Bit(s)	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 offset 0x1110)

The station address low register, shown in Figure 18 and described in Table 15, contains the low-order 32 bits of the 48 bit station address.



station_address_low_register.eps

Figure 18: Station Address Low (SAL) Register

Table 15: Station Address Low Register Bit Definitions

Bit(s)	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 offset 0x1114)

The MII management control register, shown in Figure 19 and described in Table 16, 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.

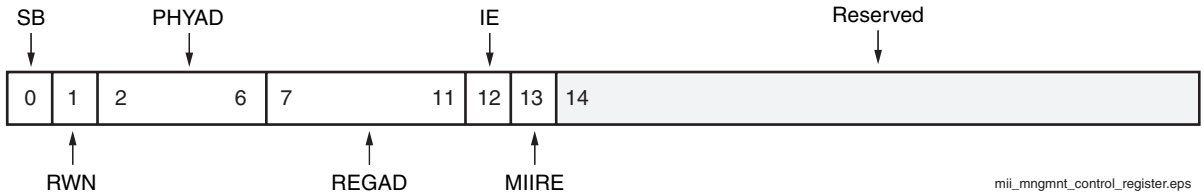


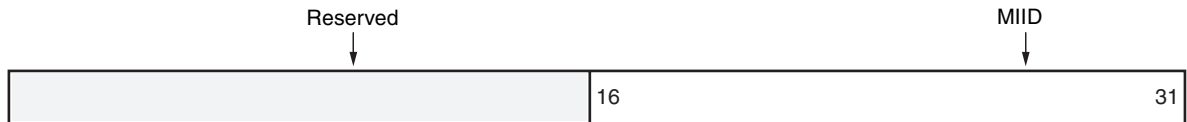
Figure 19: MII Management Control Register (MGTCR)

Table 16: MII Management Control Register Bit Definitions

Bit(s)	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. <ul style="list-style-type: none"> '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. <ul style="list-style-type: none"> '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. <ul style="list-style-type: none"> '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. <ul style="list-style-type: none"> '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 (MGTD R offset 0x1118)

The MII management data register, shown in Figure 20 and described in Table 17, 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.



mii_mngmnt_data_register.eps

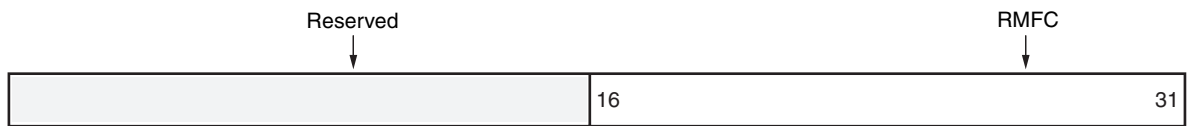
Figure 20: MII Management Data Register (MGTD R)

Table 17: MII Management Data Register Bit Definitions

Bit(s)	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 offset 0x1128)

The receive missed frame count register is shown in Figure 21 and described in Table 18. This register value represents the number of missed valid frames since the last reset with destination addresses that pass the current address validation modes.



receive_missed_frame_count_register.eps

Figure 21: Received Missed Frame Count (RMFC) Register

Table 18: Receive Missed Frame Count Bit Definitions

Bit(s)	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 offset 0x112C)

The receive collision count register value represents the number of received frames that were interrupted 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. The receive collision count register is shown in [Figure 22](#) and described in [Table 19](#).

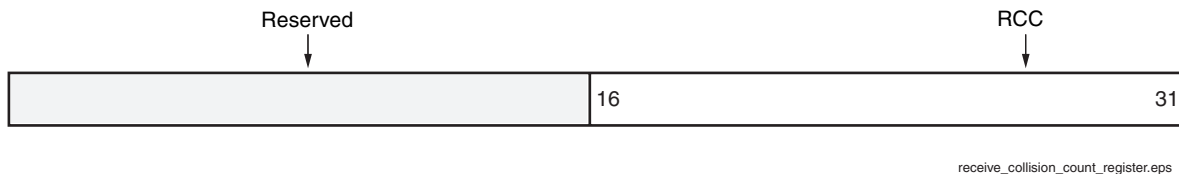


Figure 22: Receive Collision Count (RCC) Register

Table 19: Receive Collision Count Bit Definitions

Bit(s)	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 offset 0x1130)

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. The RFCSEC register is shown in [Figure 23](#) and described in [Table 20](#).

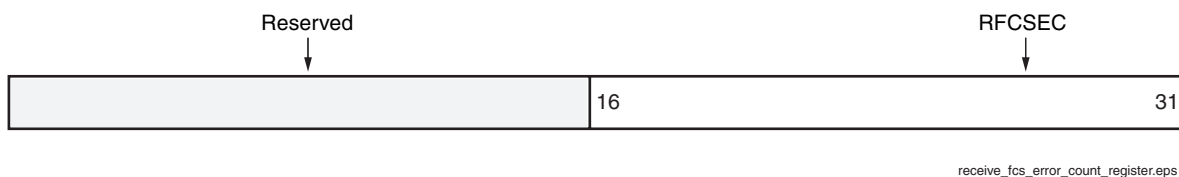


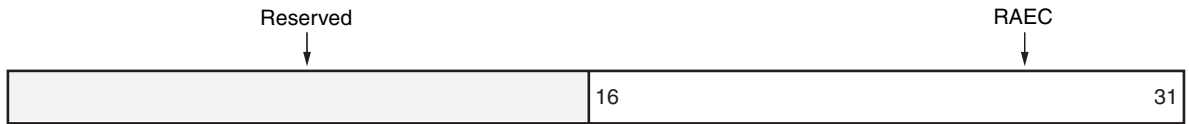
Figure 23: Receive FCS Error Count (RFCSEC) Register

Table 20: Receive FCS Error Count Bit Definitions

Bit(s)	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 offset 0x1134)

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. The RAEC register is shown in [Figure 24](#) and described in [Table 21](#).



receive_alignment_error_count_register.eps

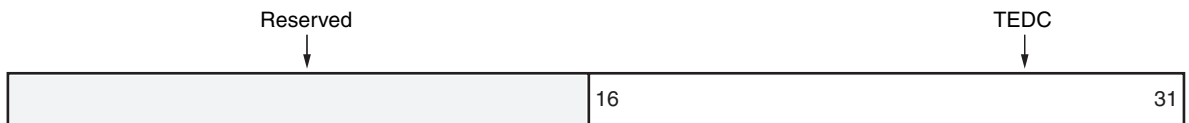
Figure 24: Receive Alignment Error Count (RAEC) Register

Table 21: Receive Alignment Error Count Bit Definitions

Bit(s)	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 offset 0x1138)

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. The TEDC register is shown in [Figure 25](#) and described in [Table 22](#).



transmit_excess_deferral_count_register.eps

Figure 25: Transmit Excess Deferral Count (TEDC) Register

Table 22: Transmit Excess Deferral Count Bit Definitions

Bit(s)	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.

Receive CAM Entry High Register (CAMH offset 0x1140)

The CAMH register, shown in [Figure 26](#) and described in [Table 23](#), exists only if the CAM is included in the design (C_CAM_EXIST = '1'). This register contains the address of the CAM location (0 to 63) to which a MAC address will be written along with the upper 16 bits of that MAC address (similar to the SAH register). The MAC address in this register and the CAML register will be written into the CAM at the address provided in this register when ECR bit 23 is set to '1'.

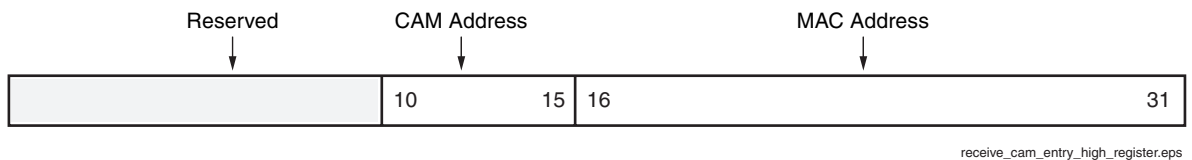


Figure 26: Receive CAM Entry High (CAMH) Register

Table 23: Receive CAM Entry High Register Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
0-9	Reserved	Read	"0000000000"	Reserved. These bits are reserved for future use.
10-15	CAMA	Read/Write	"00000"	CAM entry address (0 to 63)
16-31	MACH	Read/Write	0x0000	MAC Address High

Receive CAM Entry Low Register (SAL offset 0x1144)

The SAL register, shown in [Figure 27](#) and described in [Table 24](#), exists only if the CAM is included in the design (C_CAM_EXIST = '1'). This register contains the lower 32 bits of that MAC address (similar to the SAL register). The MAC address in this register and the CAMH register will be written into the CAM at the address provided in the CAMH register when ECR bit 23 is set to '1'.

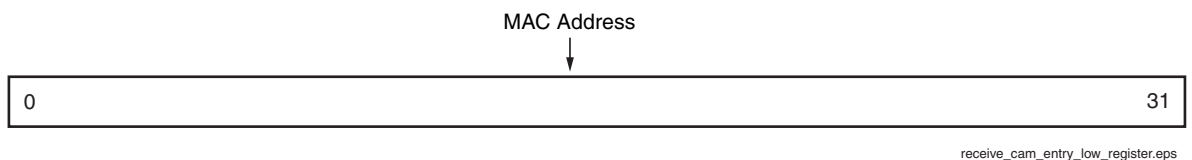


Figure 27: Receive CAM Entry Low (CAMH) Register

Table 24: Receive CAM Entry Low Register Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
0-31	D0 - D31	Read/Write	0x00000000	MAC Address Low.

EMAC Block Diagram

The top-level block diagram for the EMAC is shown in [Figure 1](#).

Block Diagram TX Control

The block diagram for the **TX Control** is shown in [Figure 28](#).

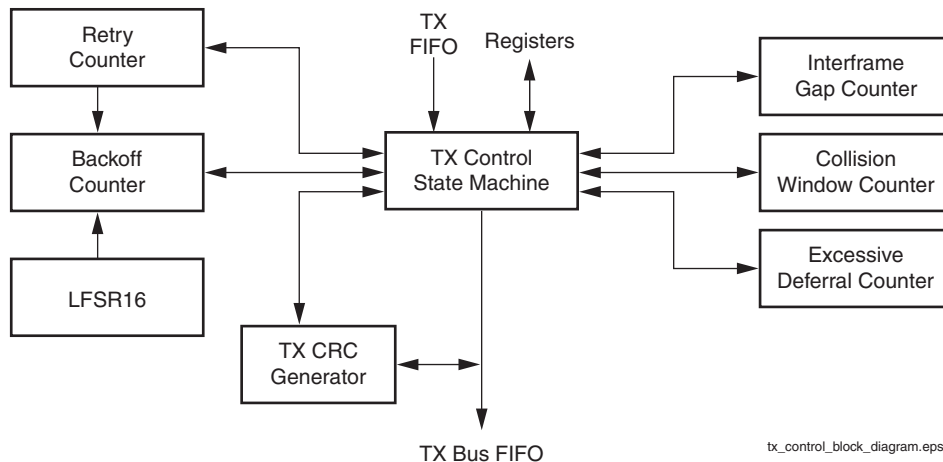


Figure 28: TX Control Block Diagram

Block Diagram RX Control

The block diagram for the **RX Control** is shown in [Figure 29](#).

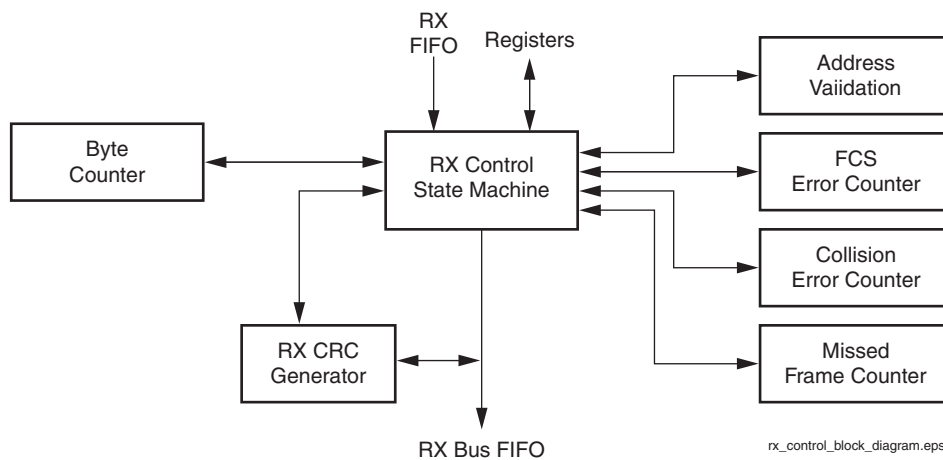


Figure 29: RX Control Block Diagram

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 (PHY_tx_clk)

The transmit clock [PHY_tx_clk] is generated by the external PHY and must be used by the EMAC to provide transmit data [PHY_tx_data (3:0)] and control signals [PHY_tx_en and PHY_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 (PHY_rx_clk)

The receive clock [PHY_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 [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 EMAC to sample the receive data [PHY_rx_data(3:0)] and control signals [PHY_dv and PHY_rx_er] from the PHY.

MII Management Clock (PHY_mii_clk)

The Management Data Clock (PHY_mii_clk) is driven by the EMAC to the PHY as the reference for data transfer on the PHY_mii_data 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 PHY_mii_clk of approximately 1 to 2.3 MHz.

Processor Bus Clock (OPB_Clk)

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

PHY_rst_n

Some PHYs use a reset input to establish or re-establish a known initial state. The EMAC provides the PHY reset signal (PHY_rst_n) for this purpose. This active low output is driven active when OPB_Rst is active, when the EMAC Control Register bit 5 is '0', or when a soft reset is performed by writing to the Device Software Reset Register (offset 0x0040) when the software reset function is included in the build (C_RESET_PRESENT = '1').

PHY_tx_en

The EMAC 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 have been transmitted. PHY_tx_en is negated 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 30](#) shows PHY_tx_en timing during a transmission with no collisions.

PHY_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.

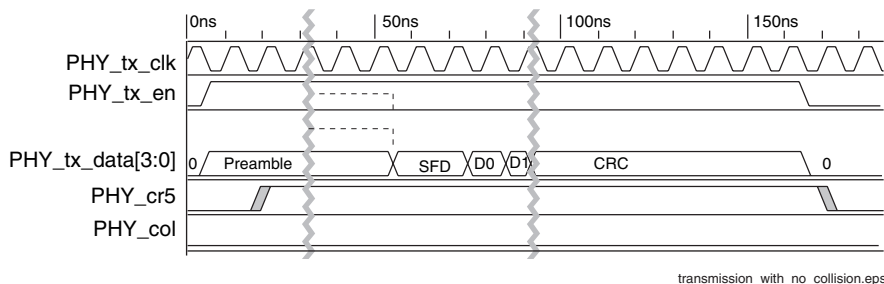


Figure 30: Transmission with no collision

PHY_tx_data(3:0)

The EMAC 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 will transmit 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 31.

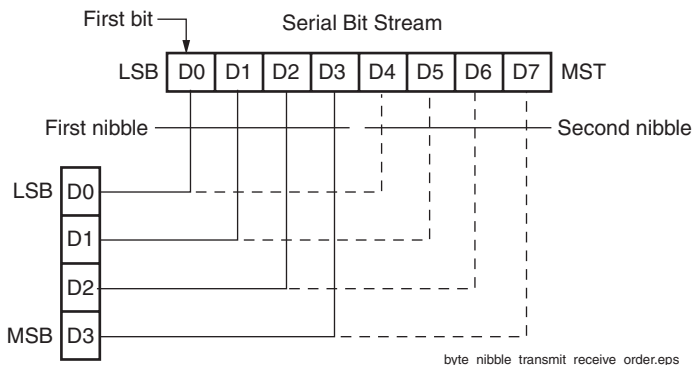


Figure 31: Byte/Nibble Transmit and Receive Order

PHY_tx_er

The EMAC drives the Transmit coding Error signal (PHY_tx_er) synchronously to PHY_tx_clk. When PHY_tx_er is asserted for one or more PHY_tx_clk periods while PHY_tx_en is also asserted and the PHY is operating in 100Mbps 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.

PHY_tx_er has no effect on PHY operation in 10Mbps mode or when PHY_tx_en is de-asserted. 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. Table 25 shows the possible combinations for the transmit signals.

Table 25: Possible values for PHY_tx_en, PHY_tx_er, PHY_tx_data[3:0]

PHY_tx_en	PHY_tx_er	PHY_tx_data[3:0]	Indication
0	0	0000 through 1111	Normal interframe
0	1	0000 through 1111	Reserved
1	0	0000 through 1111	Normal data transmission
1	1	0000 through 1111	Transmit propagation

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. In order for a received frame to be correctly received by the EMAC, PHY_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 PHY_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 PHY_rx_clk. Figure 32 shows the behavior of PHY_dv during frame reception.

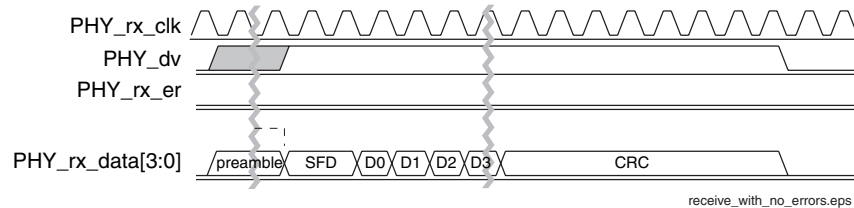


Figure 32: Receive With No Errors

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 EMAC must not be affected by PHY_rx_data(3:0) while PHY_dv is de-asserted.

Also, the EMAC should ignore a special condition that occurs while PHY_dv is de-asserted when the PHY may 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 will provide a minimum of 10 nS setup and hold time for this signal in reference to 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 (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.

PHY_rx_er should have no effect on the EMAC 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 will provide a minimum of 10 nS setup and hold time for this signal in reference to PHY_rx_clk. Figure 33 shows the behavior of PHY_rx_er during frame reception with errors.

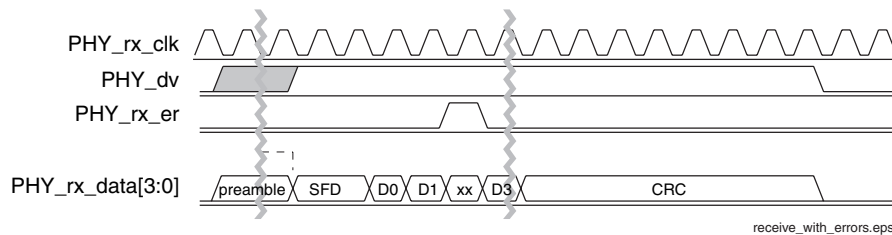


Figure 33: Receive With Errors

Table 26 shows the possible combinations for the receive signals.

Table 26: 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 interframe
0	1	0000	Normal interframe
0	1	0000 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 nonidle when operating in half duplex mode. PHY_crs is de-asserted when both the transmit and receive are idle.

The PHY drives PHY_crs asserted throughout 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 EMAC transmit and receive circuitry and is double synchronized to the processor clock as it enters the EMAC.

PHY_col

The PHY drives the Collision detected signal (PHY_col) active to indicate the detection of a collision on the bus. The PHY drives PHY_col asserted while the collision condition persists. The PHY also drives PHY_col asserted when operating at 10 Mbps 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 EMAC transmit and receive circuitry and is double synchronized to the processor clock as it enters the EMAC. Figure 34 shows the behavior of PHY_col during frame transmission with a collision.

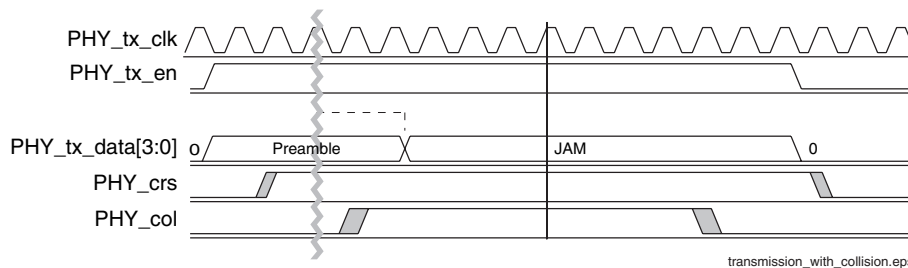


Figure 34: Transmission With Collision

PHY_mii_data

The Management Data Input/Output signal (PHY_mii_data) is a bidirectional signal between the PHY and the EMAC used to transfer control and status. All transfers via the PHY_mii_data are synchronous to the PHY_mii_clk signal. PHY_mii_data must be 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 PHY_mii_clk 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 PHY_mii_data is structured as frames as shown in Table 27.

Table 27: Management Frame Format

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

Idle

The IDLE condition on PHY_mii_data is a high-impedance state. All 3-state drivers are disabled and the PHY's pull-up resistor will pull the PHY_mii_data 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 PHY_mii_data 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 35** shows the behavior of the PHY_mii_data signal during the turnaround field of a read transaction.

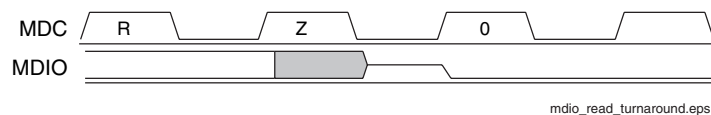


Figure 35: MDIO Read Turn Around

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 retries 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.

When RX checksum offloading is included in the design then the Release signal will also mark the end of packet and cause the RAW checksum and RX status to be written into the FIFO. This will eventually be read out of the FIFO by the DMA and put in the SR field of the RX buffer descriptor.

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 36](#) shows the address validation process.

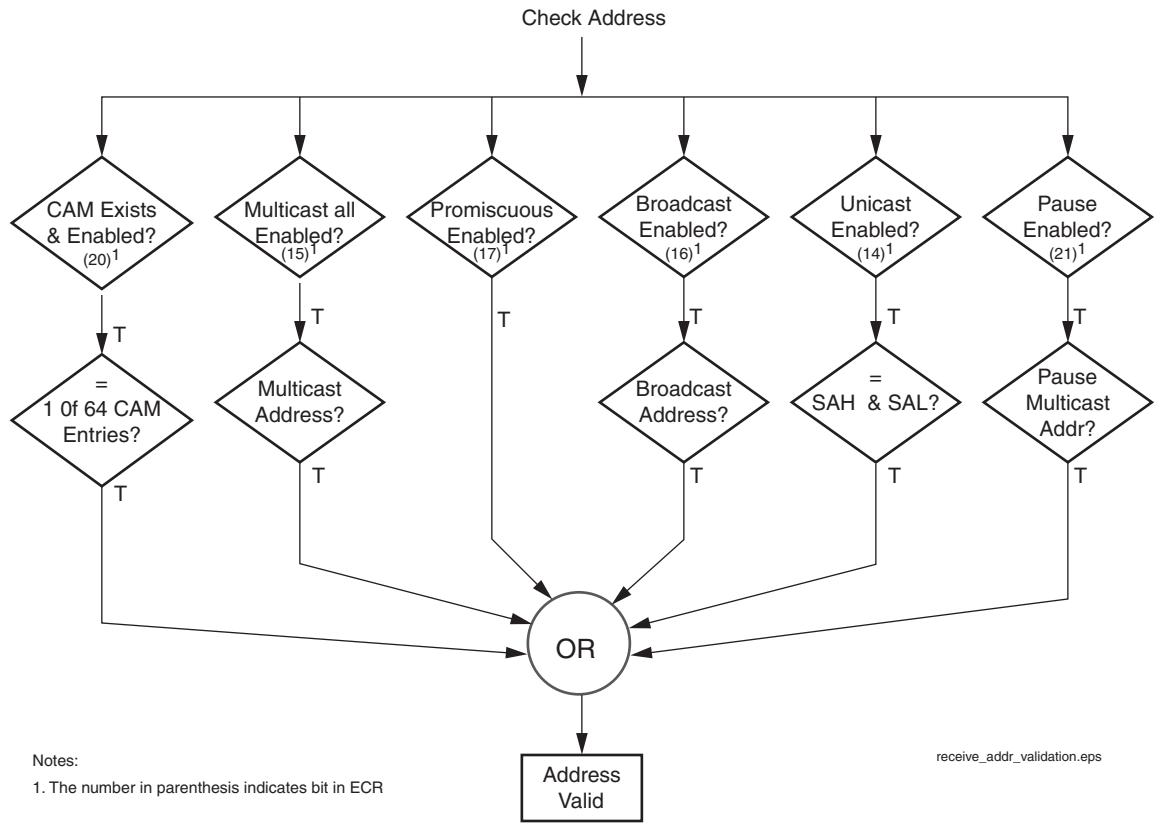


Figure 36: Address Validation Process

Using the CAM for Receive Address Validation

In order to use the CAM for receive address validation, it must be included in the design by setting the C_CAM_EXISTS parameter = '1'. The parameter C_CAM_BRAM_0_SRL_1 is set to select between BRAM or SRL logic based implementation. This is strictly a trade off between using BRAMS or FPGA logic.

The CAM is capable of holding up to 64 48 bit MAC addresses. The CAM can not be read or reset. To clear the CAM, each of the 64 entries must be written.

To write an entry to the CAM, place the upper 16 bits of the MAC address in bits 16 through 31 of the CAMH register along with the CAM entry address in bits 10 through 15. The address should be in the range 0 to 63 inclusive. The lower 32 bits of the MAC address must be placed in the CAML register.

The write to the CAM is initiated by writing a '1' to bit 23 of the ECR. This bit will automatically reset to '0'.

Once all of the entries have been placed in the CAM, the CAM must be enabled by writing a '1' to bit 20 of the ECR.

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 PHY_mii_data 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;1
NET "phy_tx_clk" USELOWSKEWLINES;1
NET "phy_tx_clk" MAXSKEW= 2.0 ns;2
NET "phy_rx_clk" MAXSKEW= 2.0 ns;2
NET "phy_tx_clk" MAXSKEW= 1.0 ns;3
NET "phy_rx_clk" MAXSKEW= 1.0 ns;3
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;2
```

1. Should be used only for Virtex, Virtex-E, Spartan-II or Spartan-III devices.
2. Should be used for all devices except Virtex-4
3. Should be used only for Virtex-4

```

NET "phy_rx_data<2>" NODELAY;2
NET "phy_rx_data<1>" NODELAY;2
NET "phy_rx_data<0>" NODELAY;2
NET "phy_dv" NODELAY;2
NET "phy_rx_er" NODELAY;2
NET "phy_crs" NODELAY;2
NET "phy_col" NODELAY;2
NET "phy_rx_data<3>" IOBDELAY=NONE;3
NET "phy_rx_data<2>" IOBDELAY=NONE;3
NET "phy_rx_data<1>" IOBDELAY=NONE;3
NET "phy_rx_data<0>" IOBDELAY=NONE;3
NET "phy_dv" IOBDELAY=NONE3
NET "phy_rx_er" IOBDELAY=NONE;3
NET "phy_crs" IOBDELAY=NONE;3
NET "phy_col" IOBDELAY=NONE;3
    
```

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 28](#) for a Virtex-II Pro -7

Table 28: EMAC FPGA Performance and Resource Utilization Benchmarks (Virtex-II Pro -7)

Parameter Values													Device Resources					
C_IPIF_WRFIFO_DEPTH	C_IPIF_RDFIFO_DEPTH	C_DEV_MIR_ENABLE	C_DMA_PRESENT	C_RESET_PRESENT	C_INCLUDE_DEV_PENCODER	C_CAM_BRAM_0_SRL_1	C_CAM_EXIST	C_JUMBO_EXIST	C_MAC_FIFO_BRAM_1_SRL_0	C_MAC_FIFO_DEPTH	C_HALF_DUPLEX_EXIST	C_TX1RX_DRE_TYPE	C_TX1RX_INCLUDE_CSUM	Slices	Slice Flip-Flops	BRAMS	4-input LUTs	f _{MAX} (MHz)
16384	16384	0	1	0	0	0	0	0	1	16	0	0		1474	1538	5	1960	143

Table 28: EMAC FPGA Performance and Resource Utilization Benchmarks (Virtex-II Pro -7) (Contd)

16384	16384	0	1	0	0	0	0	0	0	16	0	0		1499	1556	2	1992	143
16384	16384	0	1	0	0	0	0	0	1	16	1	0	0	1615	1706	5	2165	143
16384	16384	0	1	0	1	0	0	0	1	16	1	0	0	1620	1705	5	2173	143
16384	16384	0	1	1	1	0	0	0	1	16	1	0	0	1640	1715	5	2192	143
16384	16384	1	1	1	1	0	0	0	1	16	1	0	0	1653	1719	5	2211	143
16384	16384	1	1	1	1	0	0	0	1	32	1	0	0	1670	1722	5	2247	143
16384	16384	1	1	1	1	0	0	0	0	32	1	0	0	1697	1740	2	2280	143
16384	16384	1	1	1	1	0	0	0	0	64	1	0	0	1728	1743	2	2343	143
16384	16384	1	1	1	1	0	0	0	1	64	1	0	0	1702	1725	5	2310	143
16384	16384	1	1	1	1	0	0	1	1	64	1	0	0	1722	1741	5	2346	143
16384	16384	1	1	1	1	0	1	1	1	64	1	0	0	2143	1997	17	2961	142
16384	16384	1	1	1	1	1	1	1	1	64	1	0	0	2382	1934	5	3418	129
32768	16384	1	1	1	1	0	1	1	1	64	1	0	0	2382	2004	18	2980	143
65536	16384	1	1	1	1	0	1	1	1	64	1	0	0	2152	2012	20	1981	136
131072	16384	1	1	1	1	0	1	1	1	64	1	0	0	2158	2017	24	2988	113
262144	16384	1	1	1	1	0	1	1	1	64	1	0	0	2165	2023	32	3006	143
262144	262144	1	1	1	1	0	1	1	1	64	1	0	0	2186	2047	47	3033	143
262144	262144	1	2	1	1	0	1	1	1	64	1	0	0	3073	2706	47	4469	115
262144	262144	1	3	1	1	0	1	1	1	64	1	0	0	3459	2873	47	5129	123
262144	262144	1	3	1	1	1	1	1	0	64	1	0	0	3739	2832	32	5631	122
262144	262144	1	3	1	1	0	1	1	1	64	1	1	0	3576	3016	47	5337	114
262144	262144	1	3	1	1	1	1	1	0	64	1	1	0	3857	2977	32	5839	116
262144	262144	1	3	1	1	0	1	1	1	64	1	0	1	3781	3237	47	5655	119
262144	262144	1	3	1	1	1	1	1	0	64	1	0	1	4021	3198	32	6157	126
32768	32768	1	3	1	1	1	0	1	1	64	1	1	0	3125	2725	7	4640	135

Notes:

1. I.19.4 with overall effort level of "high" and synthesis resource sharing on into a xc2vp20-7-ff1152 device with opb_clk period constrained to 7 nS (142.8 Mhz). Turning resource sharing off in XST may increase f_{MAX} but also may increase resources used. Using "-Timing" based Map may also increase f_{MAX}.
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
- OPB_IPIF_V2_00_1

- DMA_SG_V1_04_b

Revision History

Date	Version	Revision
12/6/04	1.1	Initial Xilinx release
2/16/05	1.2	Converted to new DS template; updated figures to Xilinx graphic standards, reformatted tables; incorporated CR198497.