
# XaQti XQ11800FP
# 1000 Mbps
# Gigabit Ethernet Controller

## Data Sheet

Order Number: 11800-0998-08

Applies to XQ11800FP chip revisions C and subsequent spins

**Revision/Update History:**

| | | |
|---|---|---|
| Rev. 8 | September 29, 1998 | Production Chip release, enhanced features |
| Rev. 7 | February, 1998 | Enhanced formatting, corrections, additional timing diagrams |
| Rev. 6 | August 28, 1997 | New Document |
| Revs. 1-5 | Various | Internal Document |

XaQti Corporation
San Jose, California
http://www.xaqti.com

**September 1998**

## 1.0 Features

- Highly Integrated Gigabit Ethernet Media Access Controller (MAC) with on-chip transmit and receive FIFO's and 8B10B PCS Encoder/Decoder.
- High performance, single-chip Gigabit Ethernet solution for uplinks, gigabit switches, buffered repeaters, embedded point-to-point applications and network servers adapters.
- Network data bandwidth: 1000 Mbps in CSMA/CD (Half Duplex) and 2000 Mbps in Full Duplex.
- Capable of both Full Duplex and Half Duplex modes of operation, configurable in manual mode or automatically in response to Link Autonegotiation.
- Meets IEEE 802.3z Gigabit Ethernet and GMII (1000BASE-T) standard specifications and IEEE 802.3x specifications for frame-based flow control.
- The dual independent 32-bit 'streaming' FIFO interfaces allow single-cycle data transfers to support Full Duplex bandwidth for network data.
- Integrated, on-chip 8 KB Rx and 4 KB Tx FIFO's with programmable thresholds for minimizing overflows and underruns.
- Asynchronous FIFO interface operates between 33 MHz to 66 MHz Both 16-bit and 32-bit modes are supported and the FIFO interface can be hardware controlled without CPU intervention for the highest performance.
- Programmable *PAUSE* Frame-based flow control integrated with Rx FIFO watermarks for automatic pause and resumption. The Pause time value is configurable and all PHY *PAUSE* capabilities are supported (None, Asymmetric, Symmetric, Symmetric/Asymmetric). A Pause pin provides an integral host system pause capability.
- User configurable PHY interface supports the Gigabit Media Independent Interface (GMII) for 8-bit gigabit transceivers.
- The GMII interface is required for the emerging 1000BASE-T specifications (IEEE 802.3ab).
- 10-bit FC-0 PHY interface to: industry-standard 1.25 Gbps Gigabit Ethernet Transceiver (SERDES) devices.
- Supports fiber optic and short-haul copper media options (1000 BASE-SX, 1000BASE-CX, 1000BASE-LX)
- TrueSTATS™ SNMP and RMON management counters provide accurate and atomic statistics even when accessed under full gigabit traffic conditions without affecting performance.
- The 32-bit and 64-bit counters conform to IETF and ISO management standards and minimize counter 'wrapping'.
- Sixty-seven (67) status and configuration registers and the fifty-three (53) Etherstat counters can be addressed in 32-bit or 16-bit format through a generic node processor interface.
- Programmable options for the detection of one level and two level VLAN tag frames on the receive side.
- Unicast, multicast, broadcast and promiscuous address filtering capabilities.
- Packet Bursting and Carrier Extension in Half-duplex mode.
- Low-Voltage 0.35µ CMOS technology for 3.3V operation (5.0 V I/O tolerant).
- Packaged in 240-pin PQFP.

## 2.0 General Description

The XMACII is an advanced next-generation Gigabit Ethernet Media Access Controller for high performance 1000BASE applications such as switches, uplinks, buffered repeaters, point-to-point embedded applications and server adapters. The full-speed Ethernet Controller is both Full Duplex and Half Duplex capable, and integrates full support of PCS—8B10B Encoding/Decoding, Half Duplex and Full Duplex Link Autonegotiation. The Controller's Half Duplex mode of operation supports Carrier Extension and Packet Burst.

The Controller interfaces directly to 10-bit FC-0 SERDES 1.25 Gbps Gigabit Ethernet transceivers.

The XMACII controller includes integrated dual independent 8KB Receive and 4KB Transmit FIFO buffers with 32-bit wide buses that support 'bursting' and ensures 2 Gbps Full Duplex bandwidth in the most demanding network implementations. Programmable FIFO thresholds minimize overflows and underruns and can trigger automatic IEEE 802.3x Pause Frame-based flow control. Asymmetric or Symmetric Pause implementations can be implemented for full support of this specification.

The granularity of the internal 32-bit 'timestamp' timer can be controlled through the external clock source and reset signal and can accept clock source signals from 1 to 31.25 Mhz.

The unique TrueSTATS™ SNMP and RMON management counter sets are accessible through a generic Node Processor interface which is also used for Controller mode programming. Transmit and Receive *Utilization Statistics* are constantly computed and instantly available. The TrueSTATS *shadow registers* design ensures atomic and accurate statistics even in the midst of full burst network activity and eliminates the inherent access latency problems of other MAC counter implementations. The full set of Etherstat MIB statistics counters conform to IETF and ISO specifications and are at least 32-bits wide to eliminate frequent counter 'wrapping'.

The XMACII includes sixty-seven (67) configuration and status registers and fifty-three (53) statistical counters. The registers and counters can be accessed by either 16-bit, or 32-bit mode through the Node Processor interface.

The XMACII is rich in features, found in no other controllers for special traffic shaping and test applications. Sixteen internal address-matching registers are supplied to support Multicast applications or multiple users and a separate 64-bit hash filter is provided for less rigorous matching. If there is a match, the frame is received and signalled and the *MACAddress Match* signal is asserted. The Controller is capable of of transmitting and receiving 'Jumbo' packets for special server applications.

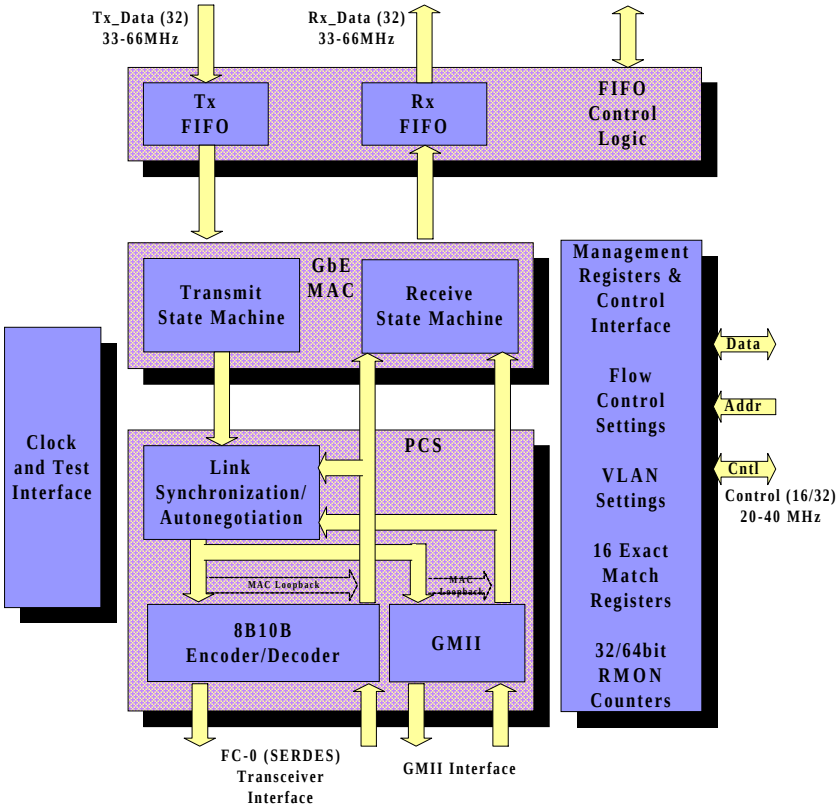
The XMACII controller is fully compatible with the latest IEEE 802.3 specifications for Gigabit Ethernet, PCS, VLAN and Pause MAC-frame flow control.

The XaQti Gigabit Ethernet Controller is implemented in a low-power 3.3V CMOS device within a 240-pin PQFP package. The pins are 5.0V tolerant.

## 3.0 XMAC II Special and Unique Features

- On-chip address CAM-like registers provides 16 exact matches on address - each of which can be individually programmed. Indication of matches is provided even in promiscuous mode. One exact Source/Destination address pair may also be configured.
- A special register provides the source address of the latest 'good' received packet.
- 64-bit hash filter for multicast frames.
- Programmable frame-by-frame CRC append or disable; CRC check on Rx packets.
- Receive frame status is appended to the end of the received frame along with optional 32-bit timestamp.
- The internal 32-bit 'timestamp' timer can be controlled with an external clock source with a input signal range of 1 to 31.25 Mhz and can be reset externally.
- Two programmable VLAN registers - Rx event counters correctly adjusted for new frame sizes - 1522 for one level VLAN, 1538 for two level VLAN.
- Interpacket Gap (IPG) can be adjusted for traffic shaping.
- Transmit and Receive Utilization continually computed.
- Capable of promiscuous mode operation for testers and special applications.
- Provides internal and external loopback options.
- Programmable options allow selective reception of all: MAC-Control frames; CRC and other error packets and transmission of MAC-Control frames and 'error' packets.
- *Transparent* mode allows transmission of characters in encoded/'unencoded' format on the transmit path with 8B10B PCS encoding but with preamble (SFD) on the receive path.
- Receive FIFO interface can also be tri-stated for multi-device interconnection.
- The Controller's Host Receive Ready signal enables the system to stop and start data transfer upon demand.
- Transmit and Receive status is automatically generated on a per-packet basis.
- Special options to *flush* the Tx and Rx FIFOs or the current Tx or Rx frames.
- Little or Big Endian byte ordering for the Host FIFO interface.

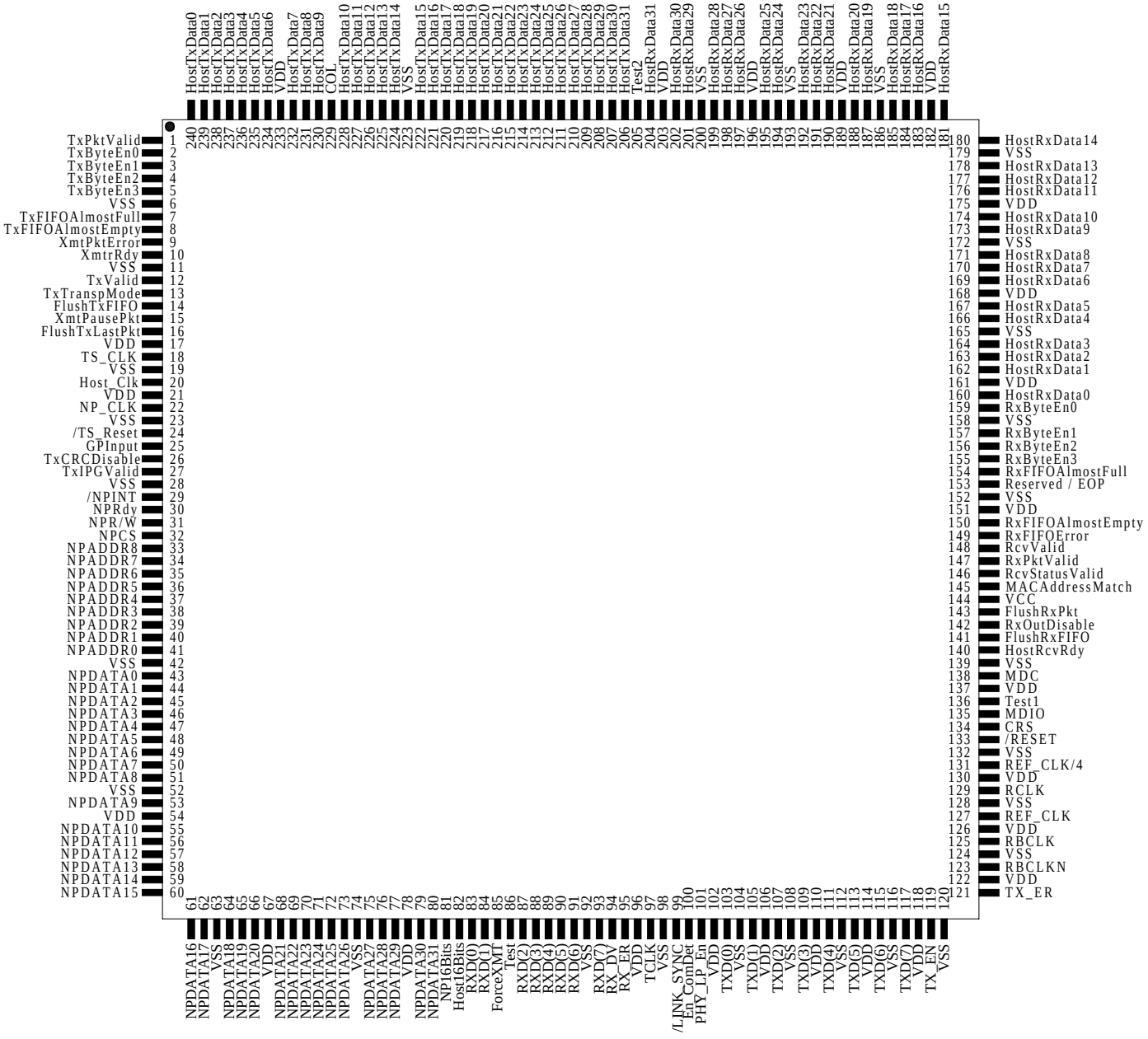
*Figure 1* - XMAC II System Diagram



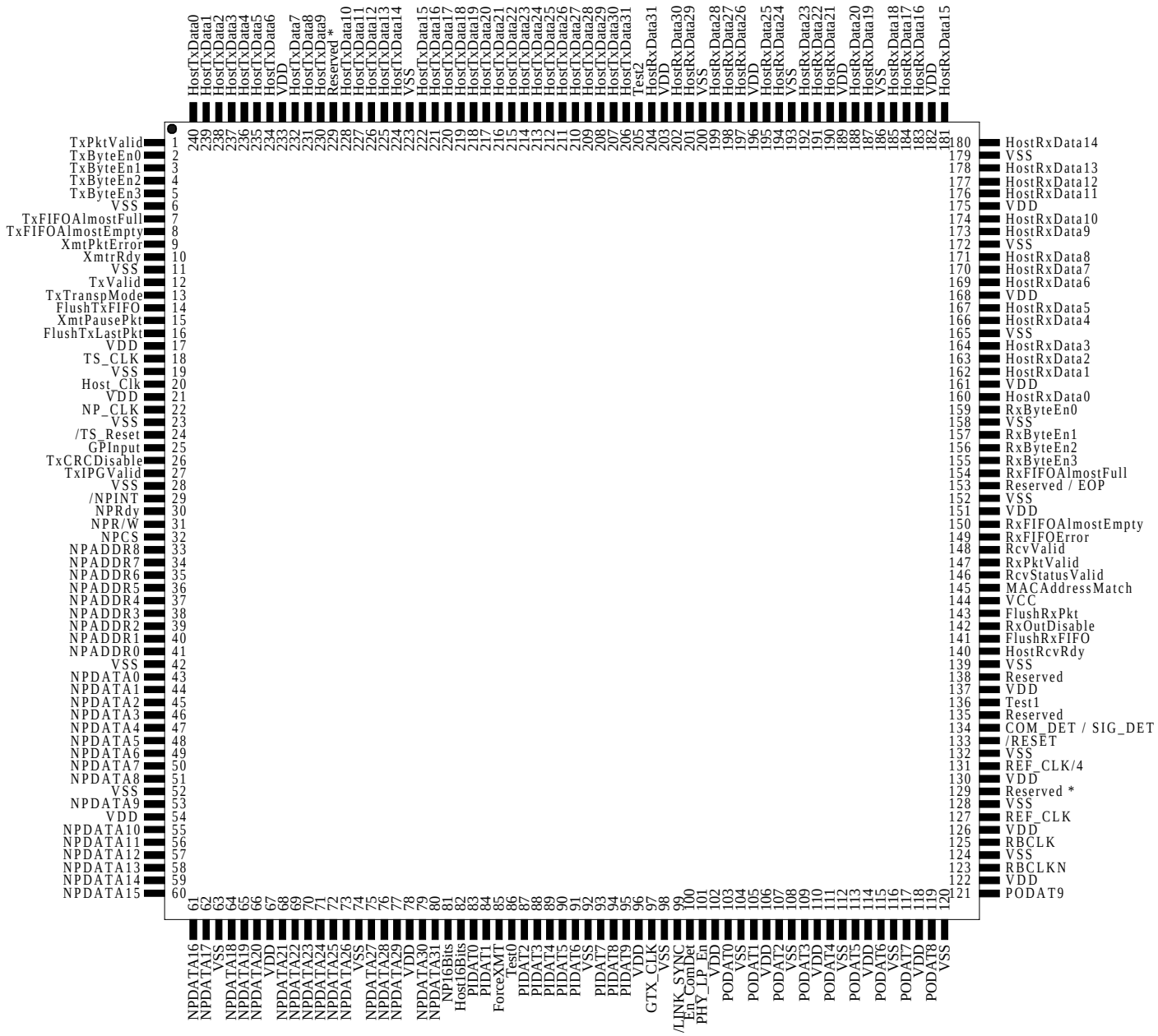
## 4.0 Pinout

This section describes the XQ11800FP signal pinouts and descriptions.

### 4.0.1 Chip Pinouts, GMII Mode PHY Interface



## 4.0.2 Chip Pinouts, FC-0 Mode PHY Interface



## 4.1 SIGNAL DESCRIPTIONS

The following abbreviations are used:

| | | |
|---|---|---|
| I | = | Input |
| O | = | Output |
| I/O | = | Input/Output |
| TTL In | = | TTL level input signal |
| TTL4 | = | TTL, 4mA output signal |
| TTL8 | = | TTL, 8mA output signal |
| CMOS2 | = | CMOS, 2mA output signal |
| CMOS4 | = | CMOS, 4mA output signal |
| CMOS8 | = | CMOS, 8mA output signal |
| Pull Up | = | Pull up to $V_{DD}$ through resistor |
| OD | = | Open Drain |
| Tri | = | Tri-stated TTL pins |
| GND | = | Connect to ground |
| No connect | = | Must not be connected |
| Power | = | Power Input, Refer to Section 9.2, Electrical and Environmental Specifications |

## 4.2 XMAC II PIN LISTINGS

The following pages describe the signals used by the XQ11800FP. The XMAC II may be configured in either of two PHY interface modes. The PHY interface mode is determined by the *gmiiMode* bit (Bit #0) in the **Hardware Configuration** register.

If this bit is set to ‘1’, then the GMII interface is selected for the XMACII PHY interface. This XMAC II Pin Listing for the GMII interface is provided in Section 4.2.1, immediately following.

If this bit is set to ‘0’ (default), then the 10-bit, 125 MHz FC-0 interface is selected for the XMAC II PHY interface. The Pin Listings for the FC-0 PHY interface mode are listed separately in Section 4.2.2.

### 4.2.1 XMAC II PIN LISTING, GMII PHY INTERFACE MODE

| Pin # | Pin Name | I/O | Type |
|---|---|---|---|
| 1 | TxPktValid | I | TTL In |
| 2 | TxByteEn0 | I | TTL In |
| 3 | TxByteEn1 | I | TTL In |
| 4 | TxByteEn2 | I | TTL In |
| 5 | TxByteEn3 | I | TTL In |
| 6 | VSS | I | Power |
| 7 | TxFIFOAlmostFull | O | CMOS4 Out |
| 8 | TxFIFOAlmostEmpty | O | CMOS4 Out |
| 9 | XmtPktError | O | CMOS4 Out |
| 10 | XmtrRdy | O | CMOS8 Out |
| 11 | VSS | I | Power |
| 12 | TxValid | I | TTL In |
| 13 | TxTranspMode | I | TTL In |
| 14 | FlushTxFIFO | I | TTL In |
| 15 | XmtPausePkt | I | TTL In |
| 16 | FlushTxLastPkt | I | TTL In |
| 17 | VDD | I | Power |
| 18 | TS_CLK | I | TTL In |
| 19 | VSS | I | Power |
| 20 | HOST_CLK | I | TTL In |
| 21 | VDD | I | Power |
| 22 | NP_CLK | I | TTL In |
| 23 | VSS | I | Power |
| 24 | /TS_RESET | I | TTL In |
| 25 | GPINPUT | I | TTL In |
| 26 | TxCRCDisable | I | TTL In |
| 27 | TxIPGValid | I | TTL In |
| 28 | VSS | I | Power |
| 29 | /NPINT | O | CMOS OD |
| 30 | NPRDY | O | CMOS4 Out |
| 31 | NPR/W | I | TTL In |
| 32 | NPCS | I | TTL In |
| 33 | NPADDR8 | I | TTL In |
| 34 | NPADDR7 | I | TTL In |
| 35 | NPADDR6 | I | TTL In |
| 36 | NPADDR5 | I | TTL In |
| 37 | NPADDR4 | I | TTL In |
| 38 | NPADDR3 | I | TTL In |
| 39 | NPADDR2 | I | TTL In |
| 40 | NPADDR1 | I | TTL In |

| Pin # | Pin Name | I/O | Type |
|---|---|---|---|
| 41 | NPADDR0 | I | TTL In |
| 42 | VSS | I | Power |
| 43 | NPDATA0 | I/O | TTL4 I/O |
| 44 | NPDATA1 | I/O | TTL4 I/O |
| 45 | NPDATA2 | I/O | TTL4 I/O |
| 46 | NPDATA3 | I/O | TTL4 I/O |
| 47 | NPDATA4 | I/O | TTL4 I/O |
| 48 | NPDATA5 | I/O | TTL4 I/O |
| 49 | NPDATA6 | I/O | TTL4 I/O |
| 50 | NPDATA7 | I/O | TTL4 I/O |
| 51 | NPDATA8 | I/O | TTL4 I/O |
| 52 | VSS | I | Power |
| 53 | NPDATA9 | I/O | TTL4 I/O |
| 54 | VDD | I | Power |
| 55 | NPDATA10 | I/O | TTL4 I/O |
| 56 | NPDATA11 | I/O | TTL4 I/O |
| 57 | NPDATA12 | I/O | TTL4 I/O |
| 58 | NPDATA13 | I/O | TTL4 I/O |
| 59 | NPDATA14 | I/O | TTL4 I/O |
| 60 | NPDATA15 | I/O | TTL4 I/O |
| 61 | NPDATA16 | I/O | TTL4 I/O |
| 62 | NPDATA17 | I/O | TTL4 I/O |
| 63 | VSS | I | Power |
| 64 | NPDATA18 | I/O | TTL4 I/O |
| 65 | NPDATA19 | I/O | TTL4 I/O |
| 66 | NPDATA20 | I/O | TTL4 I/O |
| 67 | VDD | I | Power |
| 68 | NPDATA21 | I/O | TTL4 I/O |
| 69 | NPDATA22 | I/O | TTL4 I/O |
| 70 | NPDATA23 | I/O | TTL4 I/O |
| 71 | NPDATA24 | I/O | TTL4 I/O |
| 72 | NPDATA25 | I/O | TTL4 I/O |
| 73 | NPDATA26 | I/O | TTL4 I/O |
| 74 | VSS | I | Power |
| 75 | NPDATA27 | I/O | TTL4 I/O |
| 76 | NPDATA28 | I/O | TTL4 I/O |
| 77 | NPDATA29 | I/O | TTL4 I/O |
| 78 | VDD | I | Power |
| 79 | NPDATA30 | I/O | TTL4 I/O |
| 80 | NPDATA31 | I/O | TTL4 I/O |

### 4.2.1 XMAC II PIN LISTING, GMII MODE (CONT.)

| Pin # | Pin Name | I/O | Type |
|---|---|---|---|
| 81 | NP16Bits | I | TTL In |
| 82 | Host16Bits | I | TTL In |
| 83 | RXD(0) | I | CMOS In |
| 84 | RXD(1) | I | CMOS In |
| 85 | ForceXmt | I | TTL In |
| 86 | TEST0 | I | GND |
| 87 | RXD(2) | I | CMOS In |
| 88 | RXD(3) | I | CMOS In |
| 89 | RXD(4) | I | CMOS In |
| 90 | RXD(5) | I | CMOS In |
| 91 | RXD(6) | I | CMOS In |
| 92 | VSS | I | Power |
| 93 | RXD(7) | I | CMOS In |
| 94 | RX_DV | I | CMOS In |
| 95 | RX_ER | I | CMOS In |
| 96 | VDD | I | Power |
| 97 | TCLK | O | CMOS16 Out |
| 98 | VSS | I | Power |
| 99 | /LINK_SYNC | O | CMOS4 Out |
| 100 | Reserved | - | No Connect |
| 101 | Reserved | - | No Connect |
| 102 | VDD | I | Power |
| 103 | TXD(0) | O | CMOS16 Out |
| 104 | VSS | I | Power |
| 105 | TXD(1) | O | CMOS16 Out |
| 106 | VDD | I | Power |
| 107 | TXD(2) | O | CMOS16 Out |
| 108 | VSS | I | Power |
| 109 | TXD(3) | O | CMOS16 Out |
| 110 | VDD | I | Power |
| 111 | TXD(4) | O | CMOS16 Out |
| 112 | VSS | I | Power |
| 113 | TXD(5) | O | CMOS16 Out |
| 114 | VDD | I | Power |
| 115 | TXD(6) | O | CMOS16 Out |
| 116 | VSS | I | Power |
| 117 | TXD(7) | O | CMOS16 Out |
| 118 | VDD | I | Power |
| 119 | TX_EN | O | CMOS16 Out |
| 120 | VSS | I | Power |

| Pin # | Pin Name | I/O | Type |
|---|---|---|---|
| 121 | TX_ER | O | CMOS16 Out |
| 122 | VDD | I | Power |
| 123 | Reserved * | I | GND |
| 124 | VSS | I | Power |
| 125 | Reserved* | I | GND |
| 126 | VDD | I | Power |
| 127 | REF_CLK | I | TTL In |
| 128 | VSS | I | Power |
| 129 | RCLK | I | TTL In |
| 130 | VDD | I | Power |
| 131 | REF_CLK/4 | O | CMOS4 Out |
| 132 | VSS | I | Power |
| 133 | /RESET | I | TTL In |
| 134 | CRS | I | CMOS In |
| 135 | MDIO | I/O | CMOS |
| 136 | TEST1 | I | GND |
| 137 | VDD | I | Power |
| 138 | MDC | O | CMOS2 Out |
| 139 | VSS | I | Power |
| 140 | HostRcvRdy | I | TTL In |
| 141 | FlushRxFIFO | I | TTL In |
| 142 | RxOutDisable | I | TTL In |
| 143 | FlushRxPkt | I | TTL In |
| 144 | VCC | I | Power |
| 145 | MACAddressMatch | O | CMOS8 Out |
| 146 | RcvStatusValid | O | CMOS8 Out |
| 147 | RxPktValid | O | CMOS8 Out |
| 148 | RcvValid | O | CMOS8 Out |
| 149 | RxFIFOError | O | CMOS4 Out |
| 150 | RxFIFOAlmostEmpty | O | CMOS4 Out |
| 151 | VDD | O | Power |
| 152 | VSS | I | Power |
| 153 | Reserved / EOP | O | CMOS8 Out |
| 154 | RxFIFOAlmostFull | O | CMOS4 Out |
| 155 | RxByteEn3 | O | TTL8 Tri |
| 156 | RxByteEn2 | O | TTL8 Tri |
| 157 | RxByteEn1 | O | TTL8 Tri |
| 158 | VSS | I | Power |
| 159 | RxByteEn0 | O | TTL8 Tri |
| 160 | HostRxData0 | O | TTL8 Tri |

### 4.2.1 XMAC II PIN LISTING, GMII MODE (CONT.)

| Pin # | Pin Name | I/O | Type |
|---|---|---|---|
| 161 | VDD | I | Power |
| 162 | HostRxData1 | O | TTL8 Tri |
| 163 | HostRxData2 | O | TTL8 Tri |
| 164 | HostRxData3 | O | TTL8 Tri |
| 165 | VSS | I | Power |
| 166 | HostRxData4 | O | TTL8 Tri |
| 167 | HostRxData5 | O | TTL8 Tri |
| 168 | VDD | I | Power |
| 169 | HostRxData6 | O | TTL8 Tri |
| 170 | HostRxData7 | O | TTL8 Tri |
| 171 | HostRxData8 | O | TTL8 Tri |
| 172 | VSS | I | Power |
| 173 | HostRxData9 | O | TTL8 Tri |
| 174 | HostRxData10 | O | TTL8 Tri |
| 175 | VDD | I | Power |
| 176 | HostRxData11 | O | TTL8 Tri |
| 177 | HostRxData12 | O | TTL8 Tri |
| 178 | HostRxData13 | O | TTL8 Tri |
| 179 | VSS | I | Power |
| 180 | HostRxData14 | O | TTL8 Tri |
| 181 | HostRxData15 | O | TTL8 Tri |
| 182 | VDD | I | Power |
| 183 | HostRxData16 | O | TTL8 Tri |
| 184 | HostRxData17 | O | TTL8 Tri |
| 185 | HostRxData18 | O | TTL8 Tri |
| 186 | VSS | I | Power |
| 187 | HostRxData19 | O | TTL8 Tri |
| 188 | HostRxData20 | O | TTL8 Tri |
| 189 | VDD | I | Power |
| 190 | HostRxData21 | O | TTL8 Tri |
| 191 | HostRxData22 | O | TTL8 Tri |
| 192 | HostRxData23 | O | TTL8 Tri |
| 193 | VSS | I | Power |
| 194 | HostRxData24 | O | TTL8 Tri |
| 195 | HostRxData25 | O | TTL8 Tri |
| 196 | VDD | I | Power |
| 197 | HostRxData26 | O | TTL8 Tri |
| 198 | HostRxData27 | O | TTL8 Tri |
| 199 | HostRxData28 | O | TTL8 Tri |
| 200 | VSS | I | Power |

| Pin # | Pin Name | I/O | Type |
|---|---|---|---|
| 201 | HostRxData29 | O | TTL8 Tri |
| 202 | HostRxData30 | O | TTL8 Tri |
| 203 | VDD | I | Power |
| 204 | HostRxData31 | O | TTL8 Tri |
| 205 | TEST2 | I | GND |
| 206 | HostTxData31 | I | TTL In |
| 207 | HostTxData30 | I | TTL In |
| 208 | HostTxData29 | I | TTL In |
| 209 | HostTxData28 | I | TTL In |
| 210 | HostTxData27 | I | TTL In |
| 211 | HostTxData26 | I | TTL In |
| 212 | HostTxData25 | I | TTL In |
| 213 | HostTxData24 | I | TTL In |
| 214 | HostTxData23 | I | TTL In |
| 215 | HostTxData22 | I | TTL In |
| 216 | HostTxData21 | I | TTL In |
| 217 | HostTxData20 | I | TTL In |
| 218 | HostTxData19 | I | TTL In |
| 219 | HostTxData18 | I | TTL In |
| 220 | HostTxData17 | I | TTL In |
| 221 | HostTxData16 | I | TTL In |
| 222 | HostTxData15 | I | TTL In |
| 223 | VSS | I | Power |
| 224 | HostTxData14 | I | TTL In |
| 225 | HostTxData13 | I | TTL In |
| 226 | HostTxData12 | I | TTL In |
| 227 | HostTxData11 | I | TTL In |
| 228 | HostTxData10 | I | TTL In |
| 229 | COL | I | CMOS In |
| 230 | HostTxData9 | I | TTL In |
| 231 | HostTxData8 | I | TTL In |
| 232 | HostTxData7 | I | TTL In |
| 233 | VDD | I | Power |
| 234 | HostTxData6 | I | TTL In |
| 235 | HostTxData5 | I | TTL In |
| 236 | HostTxData4 | I | TTL In |
| 237 | HostTxData3 | I | TTL In |
| 238 | HostTxData2 | I | TTL In |
| 239 | HostTxData1 | I | TTL In |
| 240 | HostTxData0 | I | TTL In |

### 4.2.2 XMAC II PIN LISTING, 10-BIT FC-0 PHY INTERFACE MODE

| Pin # | Pin Name | I/O | Type |
|---|---|---|---|
| 1 | TxPktValid | I | TTL In |
| 2 | TxByteEn0 | I | TTL In |
| 3 | TxByteEn1 | I | TTL In |
| 4 | TxByteEn2 | I | TTL In |
| 5 | TxByteEn3 | I | TTL In |
| 6 | VSS | I | Power |
| 7 | TxFIFOAlmostFull | O | CMOS4 Out |
| 8 | TxFIFOAlmostEmpty | O | CMOS4 Out |
| 9 | XmtPktError | O | CMOS4 Out |
| 10 | XmtrRdy | O | CMOS8 Out |
| 11 | VSS | I | Power |
| 12 | TxValid | I | TTL In |
| 13 | TxTranspMode | I | TTL In |
| 14 | FlushTxFIFO | I | TTL In |
| 15 | XmtPausePkt | I | TTL In |
| 16 | FlushTxLastPkt | I | TTL In |
| 17 | VDD | I | Power |
| 18 | TS_CLK | I | TTL In |
| 19 | VSS | I | Power |
| 20 | HOST_CLK | I | TTL In |
| 21 | VDD | I | Power |
| 22 | NP_CLK | I | TTL In |
| 23 | VSS | I | Power |
| 24 | /TS_RESET | I | TTL In |
| 25 | GPINPUT | I | TTL In |
| 26 | TxCRCDisable | I | TTL In |
| 27 | TxIPGValid | I | TTL In |
| 28 | VSS | I | Power |
| 29 | /NPINT | O | CMOS OD |
| 30 | NPRDY | O | CMOS4 Out |
| 31 | NPR/W | I | TTL In |
| 32 | NPCS | I | TTL In |
| 33 | NPADDR8 | I | TTL In |
| 34 | NPADDR7 | I | TTL In |
| 35 | NPADDR6 | I | TTL In |
| 36 | NPADDR5 | I | TTL In |
| 37 | NPADDR4 | I | TTL In |
| 38 | NPADDR3 | I | TTL In |
| 39 | NPADDR2 | I | TTL In |
| 40 | NPADDR1 | I | TTL In |

| Pin # | Pin Name | I/O | Type |
|---|---|---|---|
| 41 | NPADDR0 | I | TTL In |
| 42 | VSS | I | Power |
| 43 | NPDATA0 | I/O | TTL4 I/O |
| 44 | NPDATA1 | I/O | TTL4 I/O |
| 45 | NPDATA2 | I/O | TTL4 I/O |
| 46 | NPDATA3 | I/O | TTL4 I/O |
| 47 | NPDATA4 | I/O | TTL4 I/O |
| 48 | NPDATA5 | I/O | TTL4 I/O |
| 49 | NPDATA6 | I/O | TTL4 I/O |
| 50 | NPDATA7 | I/O | TTL4 I/O |
| 51 | NPDATA8 | I/O | TTL4 I/O |
| 52 | VSS | I | Power |
| 53 | NPDATA9 | I/O | TTL4 I/O |
| 54 | VDD | I | Power |
| 55 | NPDATA10 | I/O | TTL4 I/O |
| 56 | NPDATA11 | I/O | TTL4 I/O |
| 57 | NPDATA12 | I/O | TTL4 I/O |
| 58 | NPDATA13 | I/O | TTL4 I/O |
| 59 | NPDATA14 | I/O | TTL4 I/O |
| 60 | NPDATA15 | I/O | TTL4 I/O |
| 61 | NPDATA16 | I/O | TTL4 I/O |
| 62 | NPDATA17 | I/O | TTL4 I/O |
| 63 | VSS | I | Power |
| 64 | NPDATA18 | I/O | TTL4 I/O |
| 65 | NPDATA19 | I/O | TTL4 I/O |
| 66 | NPDATA20 | I/O | TTL4 I/O |
| 67 | VDD | I | Power |
| 68 | NPDATA21 | I/O | TTL4 I/O |
| 69 | NPDATA22 | I/O | TTL4 I/O |
| 70 | NPDATA23 | I/O | TTL4 I/O |
| 71 | NPDATA24 | I/O | TTL4 I/O |
| 72 | NPDATA25 | I/O | TTL4 I/O |
| 73 | NPDATA26 | I/O | TTL4 I/O |
| 74 | VSS | I | Power |
| 75 | NPDATA27 | I/O | TTL4 I/O |
| 76 | NPDATA28 | I/O | TTL4 I/O |
| 77 | NPDATA29 | I/O | TTL4 I/O |
| 78 | VDD | I | Power |
| 79 | NPDATA30 | I/O | TTL4 I/O |
| 80 | NPDATA31 | I/O | TTL4 I/O |

### 4.2.2 XMAC II PIN LISTING, FC-0 PHY MODE (CONT.)

| Pin # | Pin Name | I/O | Type |
|---|---|---|---|
| 81 | NP16Bits | I | TTL In |
| 82 | Host16Bits | I | TTL In |
| 83 | PIDAT0 | I | CMOS In |
| 84 | PIDAT1 | I | CMOS In |
| 85 | ForceXmt | I | TTL In |
| 86 | TEST0 | I | GND |
| 87 | PIDAT2 | I | CMOS In |
| 88 | PIDAT3 | I | CMOS In |
| 89 | PIDAT4 | I | CMOS In |
| 90 | PIDAT5 | I | CMOS In |
| 91 | PIDAT6 | I | CMOS In |
| 92 | VSS | I | Power |
| 93 | PIDAT7 | I | CMOS In |
| 94 | PIDAT8 | I | CMOS In |
| 95 | PIDAT9 | I | CMOS In |
| 96 | VDD | I | Power |
| 97 | GTX_CLK | O | CMOS16 Out |
| 98 | VSS | I | Power |
| 99 | /LINK_SYNC | O | CMOS4 Out |
| 100 | EN_COMDET | O | CMOS4 Out |
| 101 | PHY_LP_EN | O | CMOS2 Out |
| 102 | VDD | I | Power |
| 103 | PODAT0 | O | CMOS16 Out |
| 104 | VSS | I | Power |
| 105 | PODAT1 | O | CMOS16 Out |
| 106 | VDD | I | Power |
| 107 | PODAT2 | O | CMOS16 Out |
| 108 | VSS | I | Power |
| 109 | PODAT3 | O | CMOS16 Out |
| 110 | VDD | I | Power |
| 111 | PODAT4 | O | CMOS16 Out |
| 112 | VSS | I | Power |
| 113 | PODAT5 | O | CMOS16 Out |
| 114 | VDD | I | Power |
| 115 | PODAT6 | O | CMOS16 Out |
| 116 | VSS | I | Power |
| 117 | PODAT7 | O | CMOS16 Out |
| 118 | VDD | I | Power |
| 119 | PODAT8 | O | CMOS16 Out |
| 120 | VSS | I | Power |

| Pin # | Pin Name | I/O | Type |
|---|---|---|---|
| 121 | PODAT9 | O | CMOS16 Out |
| 122 | VDD | I | Power |
| 123 | RBCLKN | I | TTL In |
| 124 | VSS | I | Power |
| 125 | RBCLK | I | TTL In |
| 126 | VDD | I | Power |
| 127 | REF_CLK | I | TTL In |
| 128 | VSS | I | Power |
| 129 | Reserved (*) | I | GND |
| 130 | VDD | I | Power |
| 131 | REF_CLK/4 | O | CMOS4 Out |
| 132 | VSS | I | Power |
| 133 | /RESET | I | TTL In |
| 134 | COM_DET / SIG_DET | I | CMOS In |
| 135 | Reserved | - | Pull Up |
| 136 | TEST1 | I | GND |
| 137 | VDD | I | Power |
| 138 | Reserved | - | No Connect |
| 139 | VSS | I | Power |
| 140 | HostRcvRdy | I | TTL In |
| 141 | FlushRxFIFO | I | TTL In |
| 142 | RxOutDisable | I | TTL In |
| 143 | FlushRxPkt | I | TTL In |
| 144 | VCC | I | Power |
| 145 | MACAddressMatch | O | CMOS8 Out |
| 146 | RcvStatusValid | O | CMOS8 Out |
| 147 | RxPktValid | O | CMOS8 Out |
| 148 | RcvValid | O | CMOS8 Out |
| 149 | RxFIFOError | O | CMOS4 Out |
| 150 | RxFIFOAlmostEmpty | O | CMOS4 Out |
| 151 | VDD | O | Power |
| 152 | VSS | I | Power |
| 153 | Reserved / EOP | O | CMOS8 Out |
| 154 | RxFIFOAlmostFull | O | CMOS4 Out |
| 155 | RxByteEn3 | O | TTL8 Tri |
| 156 | RxByteEn2 | O | TTL8 Tri |
| 157 | RxByteEn1 | O | TTL8 Tri |
| 158 | VSS | I | Power |
| 159 | RxByteEn0 | O | TTL8 Tri |
| 160 | HostRxData0 | O | TTL8 Tri |

### 4.2.2 XMAC II PIN LISTING , FC-0 PHY MODE(CONT.)

| Pin # | Pin Name | I/O | Type |
|---|---|---|---|
| 161 | VDD | I | Power |
| 162 | HostRxData1 | O | TTL8 Tri |
| 163 | HostRxData2 | O | TTL8 Tri |
| 164 | HostRxData3 | O | TTL8 Tri |
| 165 | VSS | I | Power |
| 166 | HostRxData4 | O | TTL8 Tri |
| 167 | HostRxData5 | O | TTL8 Tri |
| 168 | VDD | I | Power |
| 169 | HostRxData6 | O | TTL8 Tri |
| 170 | HostRxData7 | O | TTL8 Tri |
| 171 | HostRxData8 | O | TTL8 Tri |
| 172 | VSS | I | Power |
| 173 | HostRxData9 | O | TTL8 Tri |
| 174 | HostRxData10 | O | TTL8 Tri |
| 175 | VDD | I | Power |
| 176 | HostRxData11 | O | TTL8 Tri |
| 177 | HostRxData12 | O | TTL8 Tri |
| 178 | HostRxData13 | O | TTL8 Tri |
| 179 | VSS | I | Power |
| 180 | HostRxData14 | O | TTL8 Tri |
| 181 | HostRxData15 | O | TTL8 Tri |
| 182 | VDD | I | Power |
| 183 | HostRxData16 | O | TTL8 Tri |
| 184 | HostRxData17 | O | TTL8 Tri |
| 185 | HostRxData18 | O | TTL8 Tri |
| 186 | VSS | I | Power |
| 187 | HostRxData19 | O | TTL8 Tri |
| 188 | HostRxData20 | O | TTL8 Tri |
| 189 | VDD | I | Power |
| 190 | HostRxData21 | O | TTL8 Tri |
| 191 | HostRxData22 | O | TTL8 Tri |
| 192 | HostRxData23 | O | TTL8 Tri |
| 193 | VSS | I | Power |
| 194 | HostRxData24 | O | TTL8 Tri |
| 195 | HostRxData25 | O | TTL8 Tri |
| 196 | VDD | I | Power |
| 197 | HostRxData26 | O | TTL8 Tri |
| 198 | HostRxData27 | O | TTL8 Tri |
| 199 | HostRxData28 | O | TTL8 Tri |
| 200 | VSS | I | Power |

| Pin # | Pin Name | I/O | Type |
|---|---|---|---|
| 201 | HostRxData29 | O | TTL8 Tri |
| 202 | HostRxData30 | O | TTL8 Tri |
| 203 | VDD | I | Power |
| 204 | HostRxData31 | O | TTL8 Tri |
| 205 | TEST2 | I | GND |
| 206 | HostTxData31 | I | TTL In |
| 207 | HostTxData30 | I | TTL In |
| 208 | HostTxData29 | I | TTL In |
| 209 | HostTxData28 | I | TTL In |
| 210 | HostTxData27 | I | TTL In |
| 211 | HostTxData26 | I | TTL In |
| 212 | HostTxData25 | I | TTL In |
| 213 | HostTxData24 | I | TTL In |
| 214 | HostTxData23 | I | TTL In |
| 215 | HostTxData22 | I | TTL In |
| 216 | HostTxData21 | I | TTL In |
| 217 | HostTxData20 | I | TTL In |
| 218 | HostTxData19 | I | TTL In |
| 219 | HostTxData18 | I | TTL In |
| 220 | HostTxData17 | I | TTL In |
| 221 | HostTxData16 | I | TTL In |
| 222 | HostTxData15 | I | TTL In |
| 223 | VSS | I | Power |
| 224 | HostTxData14 | I | TTL In |
| 225 | HostTxData13 | I | TTL In |
| 226 | HostTxData12 | I | TTL In |
| 227 | HostTxData11 | I | TTL In |
| 228 | HostTxData10 | I | TTL In |
| 229 | Reserved (*) | I | GND |
| 230 | HostTxData9 | I | TTL In |
| 231 | HostTxData8 | I | TTL In |
| 232 | HostTxData7 | I | TTL In |
| 233 | VDD | I | Power |
| 234 | HostTxData6 | I | TTL In |
| 235 | HostTxData5 | I | TTL In |
| 236 | HostTxData4 | I | TTL In |
| 237 | HostTxData3 | I | TTL In |
| 238 | HostTxData2 | I | TTL In |
| 239 | HostTxData1 | I | TTL In |
| 240 | HostTxData0 | I | TTL In |

## 4.3 PIN COUNT

Pin Count, GMII Mode PHY Interface

| Type | Total |
|---|---|
| Host Processor I/F (Tx & Rx FIFOs) | 98 |
| Node Processor I/F | 45 |
| GMII PHY | 26 |
| Clock/Control/Test | 12 |
| Total I/O | 181 |
| $V_{SS}$ (Ground) | 29 |
| $V_{DD}$ | 24 |
| $V_{CC}$ | 1 |
| Reserved | 5 |
| Total | 240 |

Pin Count, 10-bit FC-0 Transceiver Mode PHY Interface

| Type | Total |
|---|---|
| Host Processor I/F (Tx & Rx FIFOs) | 98 |
| Node Processor I/F | 45 |
| 10-bit PHY (SerDes) | 27 |
| Clock/Control/Test | 12 |
| Total I/O | 182 |
| $V_{SS}$ (Ground) | 29 |
| $V_{DD}$ | 24 |
| $V_{CC}$ | 1 |
| Reserved | 4 |
| Total | 240 |

## 4.4 CONNECTION RULES:

- All reserved pins must remain unconnected (except as noted with the asterik *).
- Test pins must be connected to Ground.

## 4.5 XMAC II PIN DESCRIPTIONS

| Interface | Signal | I/O | Signal Description |
|---|---|---|---|
| Host (Tx) | HostTxData(31:0) | I | These input pins provide the 32-bit data that will be clocked into the transmit FIFO on the rising edge of the *HOST_CLK*. *TxValid*, *TxPktValid* and *XmtrRdy* signals should be asserted for this 32-bit data to be clocked into the transmit FIFO. In 16-bit mode, *HostTxData (31:16)* needs to be connected to $V_{CC}$ through a resistor. |
| | TxByteEn(3:0) | I | These input signals determine which of the 4 bytes of the current 32-bit data on HostTxData(31:0) lines are valid and are latched into the device on the rising edge of the *HOST_CLK*. *See Section 4.11.1, Byte Alignment* |
| | TxPktValid | I | This input signal is asserted by the host to indicate that it has a frame for transmission. This signal must be asserted for the entire duration of a frame. |
| | TxCRCDisable | I | If this input signal is asserted, then the XMAC II will not append CRC to the current frame that is being written into the transmit FIFO. This signal can be asserted anytime during TxPktValid. |
| | TxFIFOAlmostEmpty | O | This output signal is asserted by the XMAC II to indicate that the number of bytes (to be transmitted) in the transmit FIFO is less than the value indicated by the **Transmit Low Water Mark Register**. |
| | TxFIFOAlmostFull | O | This output signal is asserted by the XMAC II to indicate that the number of bytes (to be transmitted) in the transmit FIFO is more than the value indicated by the **Transmit High Water Mark Register**. |
| | FlushTxFIFO | I | The host can assert this input signal if the entire transmit FIFO needs to be flushed. *TxPktValid* should be deasserted when this signal is asserted. Also, this signal should be asserted when *XmtPktError* is asserted. |
| | FlushTxLastPkt | I | The host can assert this input signal (during a frame transfer) to flush the frame. If this signal is asserted after the frame transfer is complete, then there is no effect. This signal should be asserted only after transferring 8 32-bit words into the transmit FIFO. |
| | TxTranspMode | I | The host asserts this input signal whenever the XMAC II transmitter needs to be configured for transparent mode operation. *See Section 4.18, Transparent Mode*. |
| | XmtPausePkt | I | The host asserts this input signal to generate a pre-programmed PAUSE frame. Any frame in the transmission process at the time of asserting this signal will be completely transmitted before the PAUSE frame is generated. The destination address field of the PAUSE frame will be same as the content of the **Pause Destination Address Register**. The pause timer field of the PAUSE frame will be same as the content of the **Pause Timer Register**. The source address field of the PAUSE frame will be same as the contents of the **Station Address Register**. This signal should not be asserted in half-duplex mode. Assert this signal for one *NP_CLK* clock cycle to transmit one PAUSE frame.<br>If the *ExtPauseGen* (Bit #25) in the **Mode** register is set, then during the rising edge of this signal a non-zero PAUSE frame will be sent and during the falling edge of this signal a PAUSE frame with a zero timer value will be sent out. *For more information, refer to Section 4.9.2 Flow Control.* |

## 4.5 XMAC II PIN DESCRIPTIONS (CONT.)

| Interface | Signal | I/O | Signal Description |
|---|---|---|---|
| Host (Tx) | XmtPktError | O | This output signal is asserted by the XMAC II to indicate that an error condition occurred. The entire transmit FIFO must be flushed before transmission can continue. This signal stays asserted until the transmit FIFO flush is complete. Typically, this output signal is connected to *FlushTxFIFO* input signal. The potential error conditions are:<br>- Excessive number of collisions<br>- Late collision<br>- Excessive deferral<br>- Tx FIFO underrun |
| | TxIPGValid | I | If the host asserts this signal, then the least significant 8 bits of the first 32-bit word on *HostTxData(31:0)* lines represents the IPG value. This can be used to control the IPG value between frames. *See Section 4.16, Programmable IPG Control*. |
| | XmtrRdy | O | This signal indicates that the transmit FIFO is ready to accept data from the host. The host should check this signal before transferring any data into the transmit FIFO. *See also Section 4.9.3 Transmit FIFO operation.* |
| | TxValid | I | This input signal is asserted by the host to indicate to the XMAC II that there is valid data on *HostTxData(31:0)* lines. |
| | ForceXmt | I | This pin can be used to perform flow control in half duplex mode. ***Do <u>not</u> assert in Full Duplex mode.*** *See Section 4.17, Force Transmit Function.* |
| Host (Rx) | HostRxData(31:0) | O | These pins provide the 32 bit data that is clocked out by the receive FIFO on the rising edge of the *HOST_CLK*. *RxPktValid*, *RcvValid* & *HostRcvRdy* signals should also be active for data transfer. In 16-bit mode *HostRxData(31:16)* should be connected to $V_{CC}$ through individual resistors. |
| | RxByteEn(3:0) | O | These output signals indicate which of the 4 bytes on *HostRxData(31:0)* pins are valid data. *See Section 4.11.2, Byte Alignment.* |
| | RcvValid | O | This output signal, when asserted indicates that the data on *HostRxData(31:0)* pins are valid. This signal is asserted (for the first word) at the same time as *RxPktValid*. Note that in between this signal may be asserted and deasserted. |
| | RxPktValid | O | This output signal from the XMAC II defines the boundary (the beginning-to-the-end-of-frame) of the incoming frame. This signal will be asserted when the XMAC II is ready to transfer the first word of an incoming packet over the *HostRxData(31:0)* lines and will remain asserted until the last word of the incoming packet is transferred. Note that this signal will be asserted even if the *HostRcvRdy* |
| | HostRcvRdy | I | This signal is asserted by the host to indicate that it is ready to accept data from the receive FIFO. Refer to description for *HostRxData(31:0)*. |
| | RxFIFOAlmostEmpty | O | This output signal is asserted by the XMAC II to indicate that the number of valid data words in the receive FIFO is less than the value indicated by the **Receive Low Water Mark Register**. This signal will be de-asserted when the number of valid data words in the receive FIFO is more than the value indicated by the **Receive Low Water Mark Register**. |

## 4.5 XMAC II PIN DESCRIPTIONS (CONT.)

| Interface | Signal | I/O | Signal Description |
|---|---|---|---|
| Host (Rx) | RxFIFOAlmostFull | O | This output signal is asserted by the XMAC II to indicate that the number of valid data words in the receive FIFO is more than the value indicated by the **Receive High Water Mark Register**. This signal will be de-asserted when the number of valid data words in the receive FIFO is less than the value indicated by the **Receive High Water Mark Register**. |
| | RxFIFOError | O | This output signal will be asserted by the XMAC II whenever the receive FIFO overflows. This signal stays asserted as long as RxFIFO is not read or flushed. |
| | RxOutDisable | I | If this input signal is asserted by the host, then the XMAC II will tristate the *HostRxData(31:0)* & *RxByteEn(3:0)* lines. |
| | FlushRxPkt | I | The host can assert this signal to request the XMAC II to flush the current packet that is being transferred from the receive FIFO to the host. Note that the FIFO will still output the data on the *HostRxData(31:0)* lines but *RcvValid*, *RxPktValid* signals will not be asserted. This pin can be used for external filtering. This signal should be asserted for at least one *HOST_CLK* cycle. |
| | FlushRxFIFO | I | The host asserts this signal to request the XMAC II to flush the entire receive FIFO. This signal should be asserted for at least one *HOST_CLK* cycle. |
| | MACAddressMatch | O | The XMAC II asserts this signal whenever the Destination address field of a packet that is being transferred from XMAC II to the Host matches the contents of the **Station Address Registers** or any of **Exact Match Registers**. This signal is asserted and deasserted with the *RxPktValid* signal. |
| | RcvStatusValid | O | When this output signal is asserted by the XMAC II (after the last word of the current frame is clocked out of the receive FIFO), a valid status for the current frame is presented on the *HostRxData(31:0)* lines. If the *AppendTimeStamp bit* is set in the **Mode Register**, then this signal will be asserted for two clock cycles and the time stamp value is presented on *HostRxData(31:0)* lines during the second clock. |
| Node Processor Interface | NPDATA(31:0) | I/O | These input/output data signals are used for accessing counters and registers. In 16-bit mode, unused pins *NPDATA(31:16)* must be connected to $V_{CC}$ through a resistor. |
| | NPADDR(8:0) | I | These input signals are used during registers/counters accesses. The address specifies which counter/register is being accessed for read/write operation. The input on these pins is a 16-bit word (not a byte address). In 32-bit mode, *NPADDR(0)* is not used. |
| | NPR/W | I | This input signal is used for selecting a read (or) write operation. This signal should be LOW for write operation and HIGH for read operation. |
| | NPCS | I | This input signal must be asserted HIGH for read/write operations of counters and registers. |
| | NPRDY | O | This output signal from the XMAC II (when asserted) indicates that the XMAC II has completed the current read/write operation. |

## 4.5 XMAC II PIN DESCRIPTIONS (CONT.)

| Interface | Signal | I/O | Signal Description |
|---|---|---|---|
| Node Processor Interface | /NPINT | O | This output signal is asserted (LOW) whenever one or more bits are set in the **Interrupt Status Register** and the corresponding bit(s) in the **Interrupt Mask Register** are cleared. This signal is deasserted when the **Interrupt Status** & the **Interrupt Event Registers** (which caused this signal to be asserted) are read. This is an Open Drain output signal and should be pulled up to $V_{CC}$ through a resistor. |
| Control/ Clock Interface | HOST_CLK | I | This input clock is used for clocking the data in and out of the transmit and receive FIFO. The rising edge is used for clocking. Valid *HOST_CLK* frequency is 33-66 MHz. |
| | REF_CLK | I | This input clock is used to derive other clocks required for the internal state machine operations. *REF_CLK* frequency is 125 MHz. This clock is also used to generate *GTX_CLK*. |
| | NP_CLK | I | This input clock is used for register access. Valid *NP_CLK* frequency is 33-40MHz. |
| | TS_CLK | I | This input clock is used for time stamping the receive packets. The time stamp data (32-bits) will be available on the *HostRxData(31:0)* lines. Note that the *RcvStatusValid* signal will be asserted by the XMAC II when the time stamp data is valid. The valid range is 1 – 31.25 MHz. External input (other than 31.25 MHz) to this pin should be syncronized to *REF/4_CLK*. |
| | /TS_RESET | I | When the host asserts this signal, the XMAC II will reset the internal time stamp value to zero. The signal should be asserted for a minimum of one *TS_CLK* period. |
| | /RESET | I | This async input signal is used to initialize all XMAC II internal state machines, transmit & receive FIFOs, counters and registers. This should be asserted for a minimum of two *NP_CLK* periods. |
| | TEST (2:0) | I/O | These input/output signals are used for testing purposes. These pins should be connected to GND. |
| | REF/4_CLK | O | This is a 31.25 MHz output clock provided by XMAC II. This could be used for *TS_CLK*. |
| | Host16bits | I | If this input signal is asserted, then the least significant 16 pins of the *HostTxData(31:0)* & *HostRxData(31:0)* pins will be used for host data transfer. (16-bit operation will be selected). |
| | NP16bits | I | If this input signal is asserted, the least significant 16 pins of the *NPDATA(31:0)* 'pins will be used for register/counter access. (16-bit operation will be selected) |
| | GPInput | I | If this input pin is asserted, then XMAC II will set *Bit 0* of **General Purpose Port Register**. This bit will remain set as long as this pin stays asserted. *This is an async. Input signal and must be valid for at least 40 nsec.* |
| PHY Interface **GMII Mode** | TXD(7:0) | O | These signals are the GMII Transmit Data signals. These signals are driven by the XMAC II and the data is synchronized with TCLK. |
| | TCLK | O | This clock (125 MHz) is provided by XMAC II |
| | TX_EN | O | This signal is asserted by XMACII when data is present on the the TXD(7:0) lines. This signal is synchronized with the TCLK. |

## 4.5 XMAC II PIN DESCRIPTIONS (CONT.)

| Interface | Signal | I/O | Signal Description |
|---|---|---|---|
| PHY Interface **GMII Mode** | TX_ER | O | This signal is asserted by XMACII if it detects any error. This signal is synchronized with the TCLK. |
| | RXD(7:0) | I | These signals are the GMII Receive Data signals. These signals are driven by the PHY device and the data is synchronized with RCLK. |
| | RCLK | I | This clock (125 MHz) is provided by the PHY device. |
| | RX_DV | I | This signal is asserted by the PHY device when data is present on the RXD(7:0) lines. This signal is synchronized with the RCLK. |
| | RX_ER | I | This signal is asserted by the PHY device if it detects any error. This signal is synchronized with the RCLK. |
| | COL | I | This signal is asserted by the PHY device when it detects a collision event. *This is an async input signal and must be valid for at least 40 nsec.* |
| | CRS | I | This signal is asserted by the PHY device when the media is active. *This is an async input signal and must be valid for at least 40 nsec.* |
| | MDC | O | GMII Management data clock is sourced by XMAC II to the PHY device as a timing reference for the transfer of information on MDIO signal. |
| | MDIO | I/O | GMII Management data input/output transfers control and status information between the PHY and XMAC II. |
| PHY Interface **FC-O Mode** | PODAT(9:0) | O | These 10-bit signals provide the encoded input for the SERDES chip for transmission. These signals are clocked out on the rising edge of *GTX_CLK*. |
| | GTX_CLK | O | This output clock is used by the SERDES chip to clock in the *PODAT(9:0)* signals. This clock is derived from *REF_CLK*. |
| | EN_COM_DET | O | This output signal is used for enabling the COMMA detection logic of SERDES chip. This signal is activated when the PCS state machine detects loss of sync and stays asserted as long as the link is not in sync. |
| | PHY_LP_EN | O | This output signal is used to configure the external SERDES chip in loopback configuration. This signal is asserted when *phyLoop bit* in the **PHY CONTROL Register** is set. |
| | PIDAT(9:0) | I | These 10-bit signals contains receive data from the external SERDES chip and are clocked in by *RBCLK* & *RBCLK_N*. |
| | RBCLK, RBCLK_N | I | These input clocks are the recovered clocks provided by the external SERDES chip. The rising edge of these clocks are used by XMAC II to Clock in PIDAT(9:0). The XMAC II's RBCLK should be the clock input that is aligned to the SERDES COMMA character (17Ch). For additional information, please check the XMAC II Reference Design. |
| | COM_DET | I | This input signal is provided by the external SERDES chip to indicate the detection of a COMMA character. *This is an async. input signal and must be valid for at least 40 nsec.* |
| | SIG_DET | | If the *SigStatChk* bit (Bit # 2) in the **Hardware Configuration** register is set to '1', then XMAC II will treat this input as SIG_DET. Note that SerDes chips do not provide SIG_DET but Fiber Optic transceivers do. *This is an async. input signal and must be valid for at least 40 nsec.* |

## 4.5 XMAC II PIN DESCRIPTIONS (CONT.)

| Interface | Signal | I/O | Signal Description |
|---|---|---|---|
| | /LINK_SYNC | O | This output signal is asserted low by the XMAC II to indicate the receive link synchronized condition. |
| Power | $V_{DD}$ | I | 3.3V Power |
| | $V_{CC}$ | I | 5.0V Reference voltage. This input voltage is as a reference voltage for 5V tolerance only.<br>*If 5.0V is not available, tie $V_{CC}$ to $V_{DD}$ (3.3V). In this configuration, XMAC II's 5.0V I/O tolerance is disabled and only $V_{DD}$ (3.3V) should be applied to I/O pins.* |
| | $V_{SS}$ | I | Ground |

## 4.6 FUNCTIONAL DESCRIPTION

The XMAC II Controller integrates the following block functions:

1. Programmable Ethernet Media Access Controller supporting both Half Duplex (CSMA/CD) with full collision support (detection, jamming, backoff and retransmission) and Full Duplex operation. Other programmatic features support special CRC generation and removal; individual interrupt masks, automatic discard (or programmable reception) of Error, CRC and MAC-control frames; big and little endian byte ordering; and programmable IPG generation.
2. Host Bus (FIFO) Interface that implements a high performance 'streaming' bus interface to the dual independent Transmit and Receive FIFOs.
3. Integral Tx and Rx buffers with programmable High and Low watermarks. The Rx FIFO implements an integrated Pause frame-based flow control mechanism that generates a Pause and Resume frame as appropriate. The buffers can be configured to 'optimize' traffic flow.
4. A generic Node Processor interface to access the XMAC II configuration, command and status registers and to access the TrueSTATS traffic management counters.
5. On-chip Address CAM and Multicast support which includes sixteen (16) exact match address registers and a 64-bit hash register for non-exact multicast address recognition.
6. The unique TrueSTATS network statistics architecture that implements a dependable, atomic and very low-overhead method of capturing accurate network statistics. Upon host system command, a snapshot of all counters is instantly copied to a set of 'shadow counters' for non-intrusive access without sacrificing any network performance.
7. PCS 8B10B Encoder/Decoder Media Interface to a 10-bit FC-0 interface of a industry-standard 1.25 Gbps Gigabit Ethernet transceivers (Serializer/DeSerializer, or SerDes).
8. GMII PHY Media Interface is provided as an XMAC II mode-selectable PHY interface configuration. The Gigabit Media Independent Interface (GMII) is an 8-bit media-independent specification to external transceivers and is required for 1000BASE-T implementations..

## 4.7 POWER-UP AND INITIALIZATION

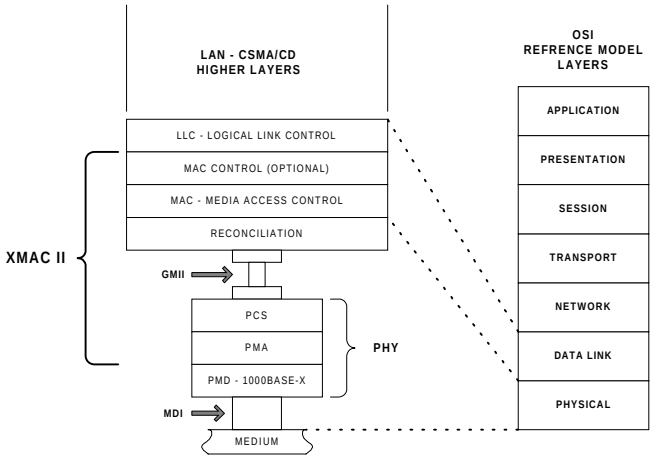
The XMAC II Controller configures itself after Power-On, or when a /RESET signal is applied. To be recognized as a valid Reset the /RESET signal must be active for at least two (2) NP_CLK periods.

Power-up or Reset causes the Controller to initialize to the default configuration values. The Controller will not be functional or accessible during this period. After Reset or Power-On the transmitter and receiver state machines are disabled and must be enabled by setting the appropriate bits in the **MMU Command register** and the desired interrupt processing mode needs to be set in the **Interrupt Mask** and **Interrupt Status registers**. *Note: sample chip initialization code is available from the XaQti Website for chip customers.*

## 4.8 MEDIA ACCESS CONTROLLER

The XMAC II is a full function CSMA/CD (Half Duplex) and Full Duplex Media Access Controller for Gigabit Ethernet, as defined by the IEEE 802.3z standard. The following diagram shows the ISO Open Systems Interconnection (OSI) reference model and the XMAC II implementation.

***Figure 2 -*** OSI Reference Model




The XMAC II includes the optional MAC control , MAC, Reconciliation, PCS and PMA layers. The MAC control layer is needed for 802.3x based full duplex and flow control support. The GMII interface will be available as an option in future versions of the XMAC II in order to support external category 5 UTP copper cable PHY implementations

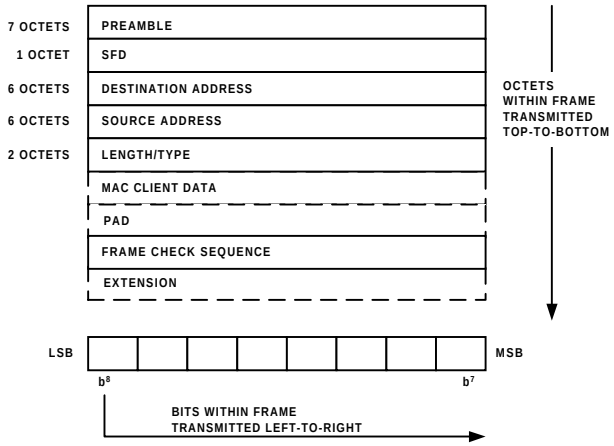
### 4.8.1 MAC Frame Format

Traffic on a Gigabit Ethernet network is transmitted as frames (sometimes referred to as packets). The Media Access Controller (MAC) processes Ethernet frames and enforces traffic, collision and link information.

The Gigabit Ethernet frame is defined by the IEEE 802.3z specification and consists of nine fields within the frame: the

Preamble: Start Frame Delimiter (SFD): the frame's Destination Address (DA); the frame's Source Address (SA); a length or type field to indicate the length of protocol type of the following field; MAC Client, or frame payload; a PAD field that contains padding if required; a the Frame Check Sequence (FCS) field containing a 32-bit cyclic redundancy check value to detect errors in a received frame; and an Extension Field, if required (for 1000 Mbps Half Duplex operation only). Of these nine fields, all are of fixed size except for the data, and pad, and Extension fields. The Extension Field bits are readily distinguishable from data bits and are not calculated in the FCS. At Gigabit Ethernet data rates, the slotTime employed by Ethernet may be inadequate to detect all collision conditions and the Extension Field is employed to circumvent this problem and ensure proper operation of the CSMA/CD protocol. This field is not used in Full Duplex and can be ignored.

***Figure 3 -*** MAC Frame Format




### 4.8.1.1 Preamble and SFD

The Preamble and SFD are prepended by the Controller on Transmit frames and are stripped by the Controller on Receive. The Transmit Command Register allows the optional transmission of packets without Preamble.

### 4.8.1.2 Destination Address

The Destination Address (DA) is a 48-bit value that represents the IEEE-MAC Address, of the station to which the Frame is addressed. The address can be a Unicast address that specifies one, and only one receiving station; a Multicast, or group, address that specifies a number of addresses; or a Broadcast address which is addressed to all of the stations on the media.

The Station Address Registers define the station's own unique address. When not in Promiscuous Mode, the destination address of incoming frames are compared with the contents of these registers and if there is a match, then the incoming frame will be received into the Rx FIFO. Typically, these registers are loaded with the IEEE-MAC address.

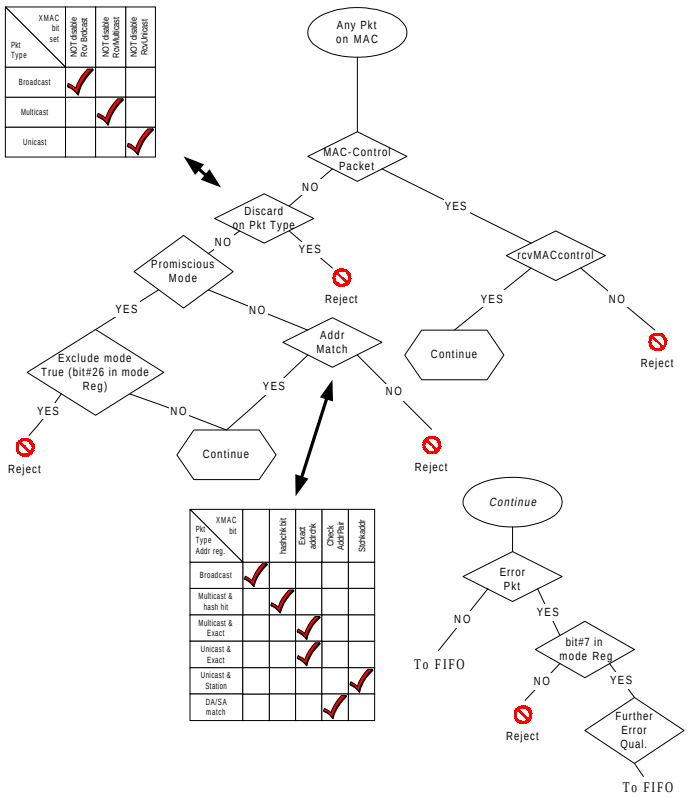
### 4.8.1.3 Address Match Filtering Options

The XMAC II also has 16 exact match address registers and an additional source address match register. When enabled, the XMAC II compares the DA of an incoming frame with the contents of the 16 **Exact Match Address registers**. If there is a match, then the corresponding frame will be passed onto the Rx FIFO and the *MACAddrMatch* signal will be asserted. Typically, these registers are loaded with Unicast/Multicast addresses that may be required by the device.

The source address match register may be paired with **Exact Match Address Register 15** to allow filtering of both Destination and Source address pair match frames. This feature is useful for capturing receive-side conversations between stations.

#### 4.8.1.3.1 Address Match Decision Process

**Figure 4 -** XMAC II Address Match Process



#### 4.8.1.3.2 48-bit Non-exact Hashing Function

Finally, the XMAC II provides a 64-bit hashing function on the DA field of incoming Multicast frames. If the hashing algorithm matches the index, the frame is passed onto the Rx FIFO. Note that this is an imperfect method of recognizing incoming Multicast packets.

The following C-code example models the CRC-32 Hashing function employed by the XMAC II. It generates a hash index from a multicast packet address. The bit position in the hash table is derived from checksum on the 48-bit address.

```
#define CRC32_POLY              0xEDB88320UL/* CRC-32 Poly — Little Endian*/
#define HASH_BITS               7                           /* Number of bits in hash
    */
unsigned
crc32_mchash (
    unsigned char *mca)
{
    u_int idx, bit, data, crc = 0xFFFFFFFFUL;
    for (idx = 0; idx < 6; idx++)
                for (data = *mca++, bit = 0; bit < 8; bit++, data >>=1)
                        crc = (crc >> 1) ^ (((crc ^ data) & 1) ? CRC32_POLY : 0);
    return crc & ((1 << HASH_BITS) - 1)             /* return low bits for hash */
}
```

#### 4.8.1.3.3 Match Notification Signal

In switch/router designs, the MAC typically is set to run in promiscuous receive mode. The XMAC II provides a signal in this mode to indicate address match. Any receive frames that match the station, or exact address match registers, are flagged using the *MACAddrMatch* signal.

### 4.8.1.4 Source Address

The Source Address is a 48-bit value that represents the IEEE Address of the Sending station.

The **LastSrcAddress register** contains the SA of the last 'good' received frame.

### 4.8.1.5 Length/Type Fields

The 16-bit field is a value of the frame length, in bytes (minus padding), for Ethernet (DIX) frames, and the value of this field is the Frame Type if the frame is a IEEE 802.3 frame.

### 4.8.1.6 MAC Client Data (Frame Payload) and PAD Field

This is a variable size field containing the data, or layered protocol information, that is exchanged between the two stations. The length of this field is variable and must be between 46 and 1500 bytes.

If the actual data is less than 46 bytes in length, the PAD field is filled with extra '0's are added to increase the combined length of the Length/Type and MAC Client data fields to the 46 byte minimum size.

The Controller can be configured to automatically strip PAD bytes from incoming frames and to automatically add PAD bytes on outgoing frames.

### 4.8.1.7 Frame Check Sequence (FCS)

A 32-bit Cyclical Redundancy Check (CRC) value is computed and appended to the frame. It is typically used by the receiving station to immediately detect most frame transmission errors.

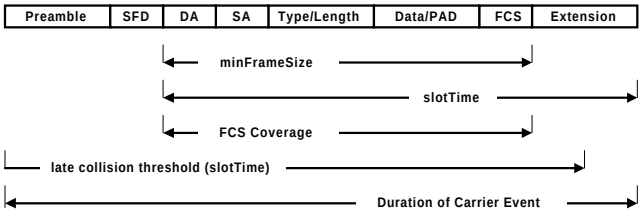
The Controller can automatically strip the FCS from incoming frames before passing them to the Rx FIFO. On outgoing packets, the Controller automatically computes and appends the CRS to outgoing frames. This function can be inhibited for special situation by setting the appropriate field in the Transmit Command Register.

### 4.8.1.8 Extension Field

At operating speeds above 100 Mbps, the slotTime employed at slower speeds is inadequate to accommodate network topologies of the desired physical extent. Carrier Extension provides a means by which the slotTime can be increased to a sufficient value for the desired topologies, without increasing the minFrameSize parameter, as this would have deleterious effects. Non-data bits, referred to as extension bits, are appended to frames which are less than slotTime bits in length so that the resulting transmission is at least one slotTime in duration. Carrier Extension can be performed only if the underlying physical layer is capable of sending and receiving symbols which are readily distinguished from data symbols, as is the case in most physical layers which use a block encoding/decoding scheme. The maximum length of the extension is equal to the quantity (slotTime – minFrameSize). The Figure below depicts a frame with carrier extension.

The MAC continues to monitor the medium for collisions while it is transmitting extension bits, and it will treat any collision which occurs after the threshold (slotTime) as a late collision.

***Figure 5 -*** MAC Frame With Carrier Extension



#### 4.8.1.9 Interpacket Gap (IPG)

The IPG is the spacing interval between frames. The minimum IPG value is 96 bit times, where 1 bit time = 1 nanosecond at 1000 Mbps. There is no maximum IPG value.

In Half Duplex Gigabit implementations the interframe gap may be filled with Extension bits in an optional transmission condition (Carrier Extension) that allows the station to send multiple frames without relinquishing control of the transmission medium. This is call Frame Bursting and is supported by the XMAC II Controller and is further described in the CSMA/CD section. For test purposes, Carrier Extension may be disabled.

The Controller can transmit and receive frames continuous at the minimum IPG of 96 bit-times. For advanced 'traffic shaping', the IPG gap on frame transmission can be adjusted, on a frame-by-frame basis. The Interpacket Gap between frames can be controlled using the **Transmit IPG register** or the *TxIPGValid* pin. IPG can be controlled in 32 nano second increments with a minimum of 32 nano second IPG value.

### 4.8.2 CSMA/CD

The XMAC II Controller is fully CSMA/CD (Half Duplex) capable and implements the standard CSMA/CD functions, including carrier sense, collision detection, jamming and the standard backoff algorithm. As a Gigabit Ethernet MAC, the XMAC II also fully implements the new functions of Carrier Extension and Frame Bursting, Framing and Collision Filtering. (Note: For special situations, the maximum number of retries (attemptLimit) and the slotTime can be programmed to nonstandard values.)

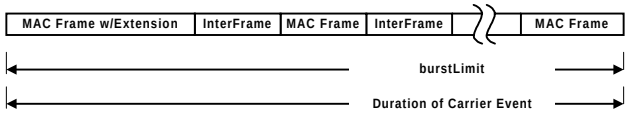
#### 4.8.2.1 Carrier Extension

Carrier Extension is a technique to ensure that Collision Detect works at Gigabit Ethernet speeds. It is an optional field that is appended to the frame after the CRC and the Extension bits can be immediately recognized as non-data. This function is fully described in the MAC section above.

#### 4.8.2.2 Frame Bursting

At operating speeds above 100 Mbps, an implementation may optionally transmit a series of frames without relinquishing control of the transmission medium. This mode of operation is referred to as burst mode. The XMAC II supports optional "burst" mode data transmission. Once the XMAC II successfully completes transmission of a frame, it begins transmission of subsequent frames without allowing the medium to assume an idle condition between frames. The XMAC II fills the interframe spacing interval with Carrier Extension bits that are easily distinguishable from data bits. The XMAC II will continue to transmit all available frames until the burst limit value of 65,536 bits, or as the value programmed in the **Burst register,** is reached. The XMAC II performs Carrier Extension for the first frame in a burst if it is less than 512 bytes. Subsequent frames within the burst are transmitted without any extension. In a properly configured network, collisions cannot occur any time after the first frame of a burst has been transmitted. The XMAC II treats all collisions that occur after the first frame of a burst as a late collision.

***Figure 6 -*** MAC Frame Bursting




#### 4.8.2.3 Framing

When a burst of frames is received while operating in half-duplex mode at speeds above 100 Mbps, the individual frames within the burst are delimited by sequences of interframe fill symbols, which are conveyed to the receiving MAC sublayer as extension bits. Once the collision filtering requirements for a given frame have been satisfied, the receipt of an extension bit can be used as an indication that all of the data bits of the frame have been received.

#### 4.8.2.4 Collision Filtering

In the absence of a collision, the shortest valid transmission in half duplex mode must be at least one slotTime in length. Within a burst of frames, the first frame of a burst must be at least slotTime bits in length in order to be accepted by the receiver, while subsequent frames within a burst must be at least minFrameSize in length. Anything less is presumed to be a fragment resulting from a collision, and is discarded by the receiver.

In half-duplex mode, occasional collisions are a normal part of the Media Access management procedure. The discarding of such a fragment by a MAC is not reported as an error.

The shortest valid transmission in full duplex mode must be at least minFrameSize in length. While collisions do not occur in full-duplex mode, a full-duplex MAC nevertheless discards received frames containing less than minFrameSize bits unless the *RcvRuntPkt* and *RcvError* bits in the **Mode Register** are set. The discarding of such a frame by a MAC is not reported as an error.

**Figure 7 -** XMAC II Ethernet Support

| Ethernet Parameter | Values | Supported |
|---|---|---|
| Slot Time | 4096 bit times *# | Yes |
| InterFrame Gap | 96 bit times (96 nsec) *# | Yes [1] |
| Attempt Limit | 16 | Yes [2] |
| BackOff Limit | 10 | Yes |
| Jam Size | 32 bits | Yes |
| Maximum Frame Size | 1518 octets | Yes [3] |
| Minimum Frame Size | 512 bits (64 octets) | Yes [4] |
| Burst Limit | 65,536 bits (8192 octets) * | Yes [5] |
| CRC Size | 32 bits (4 octets) | Yes [6] |
| Source/Destination Address Size | 48 bits (6 octets) | Yes |
| Start Frame Delimiter (SFD) | 8 bits (1octet) | Yes |
| Preamble | 56 bits (7octets) | Yes [7] |
| | | |
| * Specific to 1000 Mbps 802.3 | | |
| # At 1000 Mbps 1 bit time = 1 nanosecond | | |
| [1] May be adjusted in the *Transmit IPG* Register, or *TxIPGValid* signal | | |
| [2] May be adjusted in the *Transmit Retry Limit* Register | | |
| [3] The XMAC II supports 2 levels of VLAN tag IDs and can be programmed | | |
| to accept Jumbo packets | | |
| - With one level VLAN, the Max. frame size is adjusted to 1522 octets. | | |
| - With two level VLAN, the Max. frame size is adjusted to 1538 octets | | |
| [4] Frame size automatically adjusted to Min. if *xauto* bit is set | | |
| [5] Adjustable through the **Burst register** | | |
| [6] Disabled with the *xnoCRC* bit in the **Transmit Command register** | | |
| [7] Disabled with the *xnoPreamble* bit in the **Transmit Command register** | | |

### 4.8.3 Bit Ordering

The byte ordering of the Rx and Tx data bits defaults to Little Endian format. The Controller may be programmed to operate in Big Endian format mode by setting the *Enable Big Endian* bit in the **Mode register**. This applies to the Host (FIFO) Interface only. The Node Processor (NP) interface supports Little Endian format only.

### 4.8.4 Receive Frame Status and Optional Timestamp

A Receive Frame Status word is appended to the end of the received frame with an optional 32-bit timestamp.

### 4.8.5 Promiscuous Mode

By setting the *EnablePromiscuous* bit in the **Mode register**, the Controller will operate in Promiscuous mode.

### 4.8.6 Unicast, Multicast and Broadcast Modes

The reception of Unicast, Multicast and Broadcast frames can be enabled or disabled by setting the appropriate bit in the **Mode register**.

### 4.8.7 MAC Loopback

The Controller can loopback the data from the MAC Tx block to the MAC Rx block before entering the PCS 8B10B Encoder/Decoder. This mode is set by setting the *EnableMACLoopback* bit in the **MMU Command register**.

### 4.8.8 MAC Control and Pause Frames

The XMAC II generates and transmits MAC Control frames. The MAC Control Pause frame is used for flow control. The Pause function is programmable and is integrated with the XMAC II FIFO watermarks and is fully described in the Host Interface section.

If the *RcvMACCtrlPkt* bit is set in the **Mode register,** then XMAC II will transfer all MAC Control Frames to the Rx Host Interface.

### 4.8.9 Error Monitoring and Frame Discard

One of the functions of the XMAC II Media Access Controller is to check and detect common frame errors. The Controller checks the validates the FCS of incoming frames by recomputing the CRC and comparing it to the FCS value. Any frame with a checksum error is assumed to be invalid and is typically discarded and not passed to the Rx FIFO.

Frames whose length exceeds the 802.3 maximum packet size are called Long or Oversized frames. They can either be malformed frames, or “Jumbo” frames with expanded payload. The maximum frame size can be either 1518, 1522 or 1538 bytes, (excluding the Preamble and SFD) depending upon the level of VLAN support. To receive ‘Jumbo’ frames the *RcvLongPkts* and the *RcvError* bits in the **Mode register** must be set.

Likewise, frames of less than 64 bytes are typically defined as Runt, or undersized frames.

These error frames are typically discarded, but the Controller can be configured to pass these packets onto the Rx FIFO (rather than discarding them) by setting the appropriate bits in the **Mode register**.

These errors are also captured in the Controller management statistics counters.

### 4.8.10 Special Features and Test/Diagnostics Support

The XMAC II has unique features that facilitate test and diagnostic functions. The test and diagnostics issues require extensive hardware support; particularly during the design/development phase and later in an installed network environment. The XMAC II addresses these needs by providing vari-

ous functions that aid link and PHY testing. The more creative system designs can enable performance self tuning, self diagnostics and self healing network equipment.

#### 4.8.10.1 Traffic Generation

The XMAC II has a special feature to transmit a frame continuously. Using this feature any frame, up to a maximum size of 4 Kbytes, can be loaded into the transmit FIFO and transmitted continuously. This feature can be accessed using the *Mode* register on the XMAC II.

#### 4.8.10.2 Transparent Mode

The XMAC II also has a unique feature called 'transparent mode' which is controlled using an external hardware pin. Using this feature any frame pattern including Random jitter test patterns, can be transmitted. This feature is particularly useful for PHY testing. *Additional information on this feature may be found in Section 4.18, XMAC II Transparent Mode.*

#### 4.8.10.3 Generating Collisions

The XMAC II can also force collisions on demand for test and diagnostics purposes. Using the packet offset register, POFF, the XMAC II can transmit a "jam" with a programmed offset into the received frames to generate collisions. This feature can also be used to generate collisions and late-collisions. The feature can be used with or without another node.

#### 4.8.10.4 Generating "Phantom" Pause

This feature can be used in both full-duplex and half-duplex modes to implement a "coarse" form of rate control, or for diagnostic purposes. A local "Phantom Pause" can be affected at any time by writing to the *RateCtr* bit of the ***MMU* register** on the XMAC II. The pause duration value is provided by the **POFF register** in units of 512 bit times. *Note: No PAUSE MAC-Control frame is created, received or transmitted.*

## 4.9 FIFO GENERIC BUS INTERFACE CONTROL

The XMAC II use a generic bus interface for data transfer to and from its FIFOs. There are independent FIFO interfaces to the Transmit (Tx) and Receive (Rx) FIFOs. The data bus for either of these FIFOs is 32-bits wide and can be programmed as a 16-bit wide interface that uses only the lower 16-bit data lines. Big and little endian byte ordering is set according to the *BigEndian* bit in the *Mode* Register. Data synchronized to the is Host Clock (HOST_CLK) and new data is transferred on each clock cycle.

The Transmit and Receive FIFOs on the XMAC II are system design friendly. Both FIFOs have programmable water marks and threshold controls. There are 'Almost Full' and 'Almost Empty' hardware signals based upon High and Low watermarks.

The Receive request threshold register on the XMAC II can be used to control the receive FIFO and tune the number of bytes in the FIFO before a request for service is issued to the host. If a complete frame is in the Receive FIFO, the Host will be notified regardless of the watermark values.

Similarly, the Transmit request threshold register controls transmit FIFO. Using this feature, the XMAC II can be programmed to transmit immediately after a transmit FIFO write operation and maximize back to back frame transmission. Optionally, it can be programmed to initiate transmit after a programmed threshold is reached. If a full frame has been loaded onto the FIFO, transmission will be commenced regardless of the watermark.

The receive FIFO interface supports current Frame Flush and the transmit FIFO supports Last Frame Flush with hardware signals. Both FIFOs can also be completely flushed individually.

### 4.9.1 Rx FIFO

The Receive Host Interface consists of two FIFOs:

- 144 Byte Sync FIFO
- 8K Byte Main RxFIFO

The input to the Main RxFIFO comes from MAC Receive State Machine. The input clock for this operation is 31.25 MHz. Then received data is moved from the Main RxFIFO to the Sync FIFO at 31.25 MHz. The host receives the data (via HostRxData(31:0) pins) at frequency ranging from 33 MHz to 66 MHz. There are two conditions which must be valid together for RcvValid to be asserted:

- There is data in SyncFIFO
- The receive request threshold is met. If Receive Request Threshold Register is programmed with a value of (say) N, then XMAC II will assert the RxValid only after N*4 bytes of data (or a complete packet) are received into the RxFIFO. Once this condition is satisfied, it is not checked again till FIFO goes empty (which can happen only during inter packet gap).

#### 4.9.1.1 Normal Frame Reception

To receive a frame, the host needs to assert the **HostRcvRdy** signal. When a frame is available, the XMAC II will assert **RxPktValid** to indicate the starting boundary of a frame that is being transferred to the host. Once this signal is asserted, the host can latch the data on

the **HostRxData(31:0)** lines using the rising edge of **HOST_CLK,** provided that the **RcvValid** signal also is asserted. The **RcvValid** signal indicates that the data on **HostRxData(31:0)** lines are valid. The **RxPktValid** signal will stay asserted until the last word of a received packet is transferred to the host. After the last word transfer, **RxPktValid** will be de-asserted by the XMAC II and **RcvStatusValid** will be asserted for one cycle. When this signal is valid, the **HostRxData(31:0)** lines will contain the status of the current packet. If the *AppendTimeStamp* bit in the **Mode register** is set, then **RcvStatusValid** will be asserted for one more cycle. During this cycle, the **HostRxData(31:0)** lines will contain the timestamp value for that packet. In this case, *RcvStatusValid* is asserted for two clock cycles.

The Receive Status consists of one (or two) 32-bit words. If the *AppendTimeStamp* bit in the **Mode register** is not set, then the Receive Status is just one 32-bit word and this word gives information relevant to the current packet that is transferred from the receive FIFO to the host. The format for the status word(s) can be seen in the following Figures.

**Figure 9 -** Format of Receive Status, Little Endian

**Format of Receive StatusIn Little Endian Mode**

| Pin # | Description | Status Bit |
|---|---|---|
| 204 | HostRxData31 | Pins 31-18 gives the packet length |
| 202 | HostRxData30 | |
| 201 | HostRxData29 | |
| 199 | HostRxData28 | |
| 198 | HostRxData27 | |
| 197 | HostRxData26 | |
| 195 | HostRxData25 | |
| 194 | HostRxData24 | |
| 192 | HostRxData23 | |
| 191 | HostRxData22 | |
| 190 | HostRxData21 | |
| 188 | HostRxData20 | |
| 187 | HostRxData19 | |
| 185 | HostRxData18 | |
| 184 | HostRxData17 | Frame tagged with two-level VLAN ID |
| 183 | HostRxData16 | Frame tagged with one-level VLAN ID |
| 181 | HostRxData15 | Broadcast Frame |
| 180 | HostRxData14 | Multicast Frame |
| 178 | HostRxData13 | Unicast Frame |
| 177 | HostRxData12 | Reserved |
| 176 | HostRxData11 | Burst Mode (i.e.,this frame was received in a burst) |
| 174 | HostRxData10 | Carrier Extension Error |
| 173 | HostRxData9 | 802.3 Frame |
| 171 | HostRxData8 | Collision Error (i.e., collision occurred while receiving packet) Valid in Half Duplex |
| 170 | HostRxData7 | Carrier Event Error |
| 169 | HostRxData6 | In-Range Length Error |
| 167 | HostRxData5 | Framing Error |
| 166 | HostRxData4 | Runt Frame |
| 164 | HostRxData3 | Giant Frame |
| 163 | HostRxData2 | CRC Error Frame |
| 162 | HostRxData1 | Error Frame |
| 160 | HostRxData0 | MAC Control Packet |

**Figure 10** - Format of Receive Status, Big Endian Mode

**Format of Receive Status In Big Endian Mode**

| Pin # | Description | Status bit |
|---|---|---|
| 204 | HostRxData31 | Carrier Event Error |
| 202 | HostRxData30 | In-Range Length Error |
| 201 | HostRxData29 | Framing Error |
| 199 | HostRxData28 | Runt Frame |
| 198 | HostRxData27 | Giant Frame |
| 197 | HostRxData26 | CRC Error Frame |
| 195 | HostRxData25 | Error Frame |
| 194 | HostRxData24 | MAC Control Packet |
| 192 | HostRxData23 | Broadcast Frame |
| 191 | HostRxData22 | Multicast Frame |
| 190 | HostRxData21 | Unicast Frame |
| 188 | HostRxData20 | Reserved |
| 187 | HostRxData19 | Burst Mode (i.e., this frame was received in a burst) |
| 185 | HostRxData18 | Carrier Extension Error |
| 184 | HostRxData17 | 802.3 Frame |
| 183 | HostRxData16 | Collision Error (collision occurred while receiving this packet) |
| 181 | HostRxData15 | HostRxData15 to HostRxData10 gives the least significant 6 bits of the length field |
| 180 | HostRxData14 | |
| 178 | HostRxData13 | |
| 177 | HostRxData12 | |
| 176 | HostRxData11 | |
| 174 | HostRxData10 | |
| 173 | HostRxData9 | Frame tagged with two-level VLAN ID |
| 171 | HostRxData8 | Frame tagged with one-level VLAN ID |
| 170 | HostRxData7 | HostRxData7 to HostRxData0 gives the most significant 7-bits of the length field |
| 169 | HostRxData6 | |
| 167 | HostRxData5 | |
| 166 | HostRxData4 | |
| 164 | HostRxData3 | |
| 163 | HostRxData2 | |
| 162 | HostRxData1 | |
| 160 | HostRxData0 | |

#### 4.9.1.2 Receive Watermarks

There are two Receive watermarks:

A receive Low Watermark, **Receive Low Water Mark register**;
A receive High Watermark, **Receive High Water Mark register**.

The *RxFIFOAlmost Empty* output signal is asserted by the XMAC II to indicate that the number of valid data words in the receive FIFO is less than the value indicated by the **Receive Low Water Mark register**. This signal will be de-asserted when the number of valid data words in the receive FIFO is more than the value indicated by the **Receive Low Water Mark register**. The *RxFIFOAlmostFull* signal is asserted by the XMAC II to indicate that the number of valid data words in the receive FIFO is more than the value indicated by the **Receive High Water Mark register**. This signal will be de-asserted when the number of valid data words in the receive FIFO is less than the value indicated by the **Receive High Water Mark register**.

#### 4.9.1.3 Rx Underrun

If the host frequency is faster than 31.25 MHz, then it is possible that Receive underrun will occur. In this condition, the XMAC II will deassert the *RcvValid* signal.

#### 4.9.1.4 Receive Overflow Condition

Whenever a Receive Overflow condition occurs, the packet that caused the overflow condition will be flushed out and the **Missed Frames** and **RxFIFOOverflow** counters will be incremented. So it is possible that after flushing the packet that caused the RxFIFO overflow, a new packet may be received into the Receive FIFO (if the new packet can fit into the free space in the RxFIFO).

Whenever the Rx FIFO overflows, *RxFIFOError* signal will be asserted and this signal stays asserted until a 32-bit word is transferred from the RxFIFO or *FlushRxFIFO* signal is asserted.

### 4.9.2 Integrated Flow Control

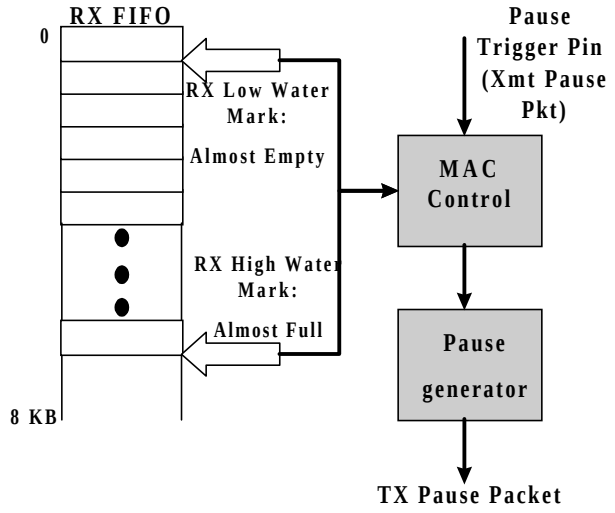
Integrated flow control is built into the Host Interface of the XMAC II. Automatic flow control can be enabled using the **Mode** register. When Rx FIFO reaches almost full condition, a “Pause” frame is automatically transmitted. When Rx FIFO reaches almost empty condition, a “Pause” frame, with a pause timer value of ‘0’, is transmitted automatically to start the link back up again. The *XmtPausePkt* pin on the XMAC II can also be used by the system to send out “Pause” control frames.

#### 4.9.2.1 Internally controlled Flow Control

To enable this mode of operation, user needs to set bits #17 & #18 in the **Mode** register. Once these two bits are set, whenever XMAC II detects an RxFIFOAlmostFull condition it will send out a Pause frame with a non-zero timer value (as programmed in the **PauseTimer** Register). *Note that XMAC II will send only one Pause frame (i.e., XMAC II will not keep sending Pause frames as long as RxFIFOAlmostFull condition is active).* XMAC II will send a Pause frame with a zero timer value when it first detects an RxFIFOAlmostEmpty condition. If the RxFIFO goes into the following sequences of states, then it is possible that more than one Pause frame with a non-zero timer value will be sent out before a Pause frame with a zero timer value is sent out:

Not(RxFIFOAlmostFull) -> RxFIFOAlmostFull -> Not(RxFIFOAlmostFull) -> RxFIFOAlmostFull -> RxFIFOAlmostEmpty

**Figure 11 -** XMAC II FIFO with integrated Pause control




#### 4.9.2.2 Externally controlled Flow Control

Option 1: Whenever the device on the other end of connection is to be inhibited from transmission, user can pulse the *XmtPausePkt* (active HIGH) signal (after initializing **PauseTimer** register with proper value). This will result in XMAC II sending out a Pause frame and the device connected to other end (assuming it supports flow control function) will be prevented from transmitting packets for a duration defined by the **PauseTimer** register. Under this option, if the local device wants the remote device to start transmission again before the timer runs out (in the remote device), then user can program the **Pause Timer** register with a zero value and then pulse the *XmtPausePkt* signal.

Option 2: User can perform the same function as in Option 1 but without having to program the **PauseTimer** register with a zero value. Under this option, when user asserts *XmtPausePkt* signal (i.e., HIGH), XMAC II will send out a Pause frame with a non-zero timer value (as programmed in the **Pause Timer** register). When user deasserts *XmtPausePkt* signal (i.e., LOW), XMAC II will send out a Pause frame with a zero timer value. So duration between these two pause frames is equal to the time duration for which *XmtPausePkt* signal is asserted.

### 4.9.3 Tx FIFO

#### 4.9.3.1 Normal Frame Transmission

To transmit a packet, the host needs to check the *XmtrRdy* signal, which will be asserted only if certain number of free locations (determined by *HostReqThreshold* setting in the **XmtRequestThreshold register**) are available in the Tx FIFO. The host asserts the *TxPktValid* signal to indicate the start of packet event to the XMAC II. *(Note that this signal should stay asserted until the end of the packet.)* After asserting the *TxPktValid* [or along with the *TxPktValid*], the host drives the *HostTxData(31:0)* lines**,** and the appropriate *TxByteEn(3:0)* and *TxValid* signals. When *TxValid* is asserted, the XMAC II will latch the *HostTxData* into the Tx FIFO on the rising edge of the *HOST_CLK* as long as the *XmtrRdy* signal is still asserted. After presenting the last word of a packet, the host deasserts the *TxPktValid* signal to indicate the end of packet event to the XMAC II. The host can assert *TxCRCDisable* to request the XMAC II not to append CRC to the current packet.

#### 4.9.3.2 Continuous Transmission

It is possible to send a given frame continuously. To initiate continuous transmission, the host needs to perform the following sequence of operations:

- Clear the transmit FIFO
- Set the *SendContinuous* bit (#20) to '1' in the **Mode register**
- Load one frame in the transmit FIFO

To terminate the continuous transmission, the host needs to set the *SendContinuous* bit (#20) to '0' in the Mode Register.

#### 4.9.3.3 Transmit Watermark

There are two Receive watermarks:

A transmit Low Watermark, **Transmit Low Water Mark register**;
A transmit High Watermark, **Transmit High Water Mark register**.

The *TxFIFOAlmostEmpty* signal is asserted by the XMAC II to indicate that the number of bytes (to be transmitted) in the transmit FIFO is less than the value indicated by the **Transmit Low Water Mark register**. The *TxFIFOAlmostFull* signal is asserted by the XMAC II to indicate that the number of bytes (to be transmitted) in the transmit FIFO is more than the value indicated by the **Transmit High Water Mark register**.

#### 4.9.3.4 Tx Underflow

If the Host provides to XMAC II at less than gigabit rate, then it is possible that the TxFIFO will underrun. When this happens, XMAC II will assert *XmtPktErrorOvr* signal and the system should flush the TxFIFO when this underrun condition occurs. Otherwise, the transmit process will not be initiated.

#### 4.9.3.5 Tx Overflow

If the Host interface provides data to the XMAC II at more than gigabit rate, then TxFIFO overflow condition may occur and the *XmtrRdy* signal will be deasserted by XMAC II**.**

#### 4.9.3.6 Transmit Pause Control Registers

There are two available registers on the XMAC II to control the contents and parameters of the Pause frames automatically generated and transmitted by the XMAC II. The **Pause Destination Address register** controls the destination address field. The **Opcode/Pause Timer register** allows control of Opcode and pause timer value.

#### 4.9.3.7 PAUSE Frame Transmission

To transmit a Pause frame (i.e., MACCtrlFrame) using the XMAC II either the *XmtPausePkt* signal is asserted, or by setting bit 15 (*SendPauseFrame*) of the **Mode register**. In the second method, the XMAC II builds a Pause frame internally and transmits it. The various fields of the pause frame are as follows:

Destination Address = Pause Destination Address Registers (the default value is 0180C2000001).
Source Address = Station Address Registers.
Opcode Field = Least Significant 16-bits of the Opcode/ Pause Timer Register (the default value is 0001)
Timer Value Field = The Most Significant 16-bits of the Opcode/Pause Timer Register (the default value is 0000)

The user needs to program these registers with the appropriate values before issuing a command to the XMAC II. When a Pause frame is successfully transmitted, the XMAC II will increment the following counters:

PAUSEMACCtrlFramesTransmitted Counter
MACCtrlFramesTransmitted Counter (since a PAUSE frame is also a MAC Control frame)

#### 4.9.3.8 'Jumbo' Packet Handling

##### 4.9.3.8.1 Transmit Operation

XMAC II is capable of transmitting packets of any length. To do this, user must make sure that the transmit FIFO never underruns. It is important to note that XMAC II TxFIFO is 4K bytes in length and the moment user transfer (4*contents of TransmitRequestThreshold) bytes into the TxFIFO, XMAC II will start removing the packet from the TxFIFO and starts transmitting that packet to the PHY device.

##### 4.9.3.8.2 Receive Operation

XMAC II RxFIFO is 8K bytes in length and operates in two different modes:

- Filter Error packets
- Receive Error packets

Filter Error packet mode of operation:

In this mode, XMAC II goes into store and forward mode of operation. Since the Receive FIFO is only 8K bytes, XMAC II cannot receive Jumbo packets.

Receive Error packet mode of operation:

In this mode, XMAC II goes into streaming mode. In this mode, user can specify a Receive Request Threshold. Once this is done, XMAC II will start transferring the received packet (either full or partial packet depending upon the Receive Request Threshold value). So if a Jumbo packet is received from the media, XMAC II will start transferring the jumbo packet out of the RxFIFO the moment certain number of bytes of the Jumbo packet is received and thereafter it will continuously transfer the packet till the end of the packet. So the user should be capable of removing the data out of the RxFIFO at wire speed.

## 4.10 GENERIC NODE PROCESSOR INTERFACE

The XMAC II contains a rich set of registers and statistical counters to simplify the implementation of network management. These registers and counters are accessed through the

Node Processor (NP) interface. The NP interface supports both 16 and 32 bit modes, however, most of the counters and few registers are 32 bits wide. In 32-bit mode, the addresses are at 4 byte boundaries (i.e. 0,4,8,C etc.), while in 16-bit mode, the addresses are at 2 byte boundaries (i.e. 0,2,4,6,8 etc.).

The following example shows how to access registers in 16 & 32 bit mode operations.

**To Select a value of 1K bytes for Transmit Low Water Mark and a value of 3K bytes for Transmit High Water Mark:**

The **Transmit Low** & **High Water Mark** Registers are implemented internally as a 32-bit register. In 32-bit mode operation, programming the registers can be done in a single write operation, at the register address of 60h. The low and high water mark values are defined by 11 bits each. Bits 0-10 define the low water mark and bits 16-26 define the high water mark. Note that the value defined in these registers indicate the number of long words (32 bits) not bytes. In 16 bit mode, the same operation is done in two steps. First, the transmit low water mark value is written to a 16 bit register at address 60h. Then transmit high water mark value is written to a 16 bit register at address 62h.

### 4.10.1 Reading and Writing Address Registers

The XMAC II contains numerous address registers such as the Station Address Registers, the Last Source Address Registers, and the Exact Match Address Registers. Each of these addresses are 48-bits in length and are implemented by three 16-bit registers. These 16 bit registers are named Address 1, Address 2 & Address 3 registers. The following shows how different bytes of the 48 bits of a address are stored in these registers.

Low Byte of Address1 Register = Bits[47:40]
High Byte of Address1 Register = Bits[39:32]

Low Byte of Address2 Register = Bits[31:24]
High Byte of Address2 Register = Bits[23:16]

Low Byte of Address3 Register = Bits[15:8]
High Byte of Address3 Register = Bits[7:0]

### 4.10.2 Burst Read on the NP Interface

For reading statistics area, it may be desirable to do a burst read. The address increment is still responsibility of host logic. The address increment can start after first ready (*NPRDY*) has been given out by XMAC II. The data for any given address (during burst) becomes available in the third clock with respect to corresponding address (the first clock being the one during which the address becomes valid on address bus). The XMAC II remains in burst mode as long as *NPCS* signal remains asserted

## 4.11 BYTE ALIGNMENT FUNCTION IN XMAC II

### 4.11.1 Transmit Side

The Transmit interface consists of 32 data pins (**HostTxData31:0**) and four Byte Enable pins (**TxByteEn3:0**). **TxByteEn3:0** pins are used to identify which of the four bytes in **HostTxData 31:0** are valid in a given 32-bit word transfer from the host into the TxFIFO.

**TxByteEn 0** indicates if the data byte in **HostTxData 7:0** is valid

**TxByteEn 1** indicates if the data byte in **HostTxData 15:8** is valid
**TxByteEn 2** indicates if the data byte in **HostTxData 23:16** is valid
**TxByteEn 3** indicates if the data byte in **HostTxData 31:24** is valid

The following rules apply for use of byte enable signals (little endian mode):

1. One or more bytes need to be valid in a 32-bit (4-byte) word.
2. If only one byte is valid, then that byte needs to be present in Byte 0 (i.e. **HostTxData 7:0**).
3. If more than one byte are valid, then those bytes needs to start from Byte 0 (i.e. **HostTxData 7:0**) and be contiguous.

The following rules apply for use of byte enable signals (big endian mode):

1. One or more bytes need to be valid in a 32-bit (4-byte) word.
2. If only one byte is valid, then that byte needs to be present in Byte 3 (i.e. **HostTxData 31:24**).
3. If more than one byte are valid, then those bytes needs to start from Byte 3 (i.**e. HostTxData 31:24**) and be contiguous.

The following are the only valid combinations for **TxByteEn 3:0** for Little Endian Mode:

| TxByteEn3 | TxByteEn2 | TxByteEn1 | TxByteEn0 |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 |

The following are the only valid combinations for **TxByteEn 3:0** for Big Endian Mode:

| TxByteEn3 | TxByteEn2 | TxByteEn1 | TxByteEn0 |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 |

### 4.11.2 *Receive Side*

The Receive interface consists of 32 data pins (**HostRxData31:0**) and four Byte Enable pins (**RxByteEn3:0**). **RxByteEn3:0** pins are used by XMAC II to identify which of the four bytes it outputs on **HostRxData 31:0** pins are valid in a given 32-bit word transfer from the RxFIFO to the host.

**RxByteEn 0** indicates if the data byte in **HostRxData 7:0** is valid

**RxByteEn 1** indicates if the data byte in **HostRxData 15:8** is valid

**RxByteEn 2** indicates if the data byte in **HostRxData 23:16** is valid

**RxByteEn 3** indicates if the data byte in **HostRxData 31:24** is valid

The following are the only valid combinations for **RxByteEn 3:0** for Little Endian Mode:

| RxByteEn3 | RxByteEn2 | RxByteEn1 | RxByteEn0 |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 |

The following are the only valid combinations for **RxByteEn 3:0** for Big Endian Mode:

| RxByteEn3 | RxByteEn2 | RxByteEn1 | RxByteEn0 |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 |

*Note: When XMAC II outputs the status (i.e. data provided on **HostRxData 31:0** with **RcvStatusValid** signal asserted), **RxByteEn** signals have no meaning and user should ignore the **RxByteEn** signals when **RcvStatusValid** signal is asserted.*

## 4.12 NETWORK MANAGEMENT STATISTICS

The XMAC II supports all the management register and counter extensions for Gigabit Ethernet. The chip also provides the necessary registers and counters to support IEEE 802.3x standard for full duplex and flow control.

To accommodate the gigabit rate of packets, all of the counters on the XMAC II are 32 bits wide, with the exception of transmit and receive byte counters which are 64 bits wide. The XMAC II has counters to track Gigabit Ethernet and 802.3x flow control application specific issues such as Burst Mode counters for transmit and receive, Carrier Event Errors, Carrier Extension Errors, Symbol Errors, MAC Control frame and Pause control frame counters.

The XMAC II chip architecture provides extensive traffic and event statistics using XaQti's TrueSTATS™ SNMP and RMON management counters. The XMAC II's internal counters can be copied to the *Snapshot Registers* at any moment. Because

the counters are copied as a set, the transfer is instantaneous and the set of statistics is 'atomic' and will remain intact until the network management agent accesses all of the necessary information. This eliminates the potential for interleaved and non-atomic statistics and allows the use of slower, more cost-effective embedded network management processors.

Through the **StatisticsCommandRegister**, transmit and receive statistics can either be constantly updated through the *Copy* command or viewed at an instant through the *Snap* command. The XMAC II also has two unique new counters to facilitate traffic monitoring: Transmit Utilization and Receive Utilization Counters. These counters are dynamically updated and maintained by the Controller. TrueSTATS counters are accessible, without interrupting the network traffic flow though the generic 32/16-bit Node Processor interface.

**Figure 12 -** XMAC II Receive Counters

| XMAC II Receive Counters List |
|---|
| Total Frames Received OK |
| Total Octets Received OK |
| Total Broadcast Frames Received OK |
| Total Unicast Frames Received OK |
| Total Multicast Frames Received OK |
| Total Long (Oversize) Frames Received |
| Total 64 Bytes Frames Received OK |
| Total 65-127 Bytes Frames Received OK |
| Total 128-255 Bytes Frames Received OK |
| Total 256-511 Bytes Frames Received OK |
| Total 512-1023 Bytes Frames Received OK |
| Total 1024+ Maximum Size Bytes Frames Received OK |
| (Adjusted automatically for VLAN) |
| Total Burst Mode Frames Received OK |
| Total MAC "PAUSE" control Frames Received |
| Total MAC control Frames Received |
| Total Invalid PAUSE Frames Received |
| Total MAC Control Frames Rec'd w/Unsupported Opcode |
| Total Receive FIFO Overrun errors |
| Total Receive Missed Frames |
| Total Frames received with Framing Error |
| Total Receive Frames with Carrier Event Error |
| Total Receive Frames with Carrier Extension Errors |
| Total Receive Frames with Symbol Errors |
| Total Receive "jabber" frames |
| Total Receive Frames with InRange Length errors |
| Total Receive Frames with FCS errors |
| Total Short frames received |
| Total Runt Frames received |
| Receive Utilization Counter |

**Figure 13 -** XMAC II Transmit Counters

| XMAC II Transmit Counters List |
|---|
| Total Frames Transmitted OK |
| Total Octets Transmitted OK |
| Total Broadcast Frames Transmitted OK |
| Total Unicast Frames Transmitted OK |
| Total Multicast Frames Transmitted OK |
| Total Long (Oversize) Frames Transmitted |
| Total 64 Bytes Frames Transmitted OK |
| Total 65-127 Bytes Frames Transmitted OK |
| Total 128-255 Bytes Frames Transmitted OK |
| Total 256-511 Bytes Frames Transmitted OK |
| Total 512-1023 Bytes Frames Transmitted OK |
| Total 1024+ Maximum Size Bytes Frames Transmitted OK (Adjusted automatically for VLAN frames) |
| Total Burst Mode Frames Transmitted OK |
| Total MAC "PAUSE" control Frames Transmitted |
| Total MAC control Frames Transmitted |
| Total Frames Transmitted with Single Collision |
| Total Frames Transmitted with Multiple Collisions |
| Total Frames Aborted due to Excessive Collisions |
| Total Frames with Excessive deferral |
| Total Frames with Late Collisions |
| Total Frames with deferred Transmissions |
| Total Transmit FIFO underrun errors |
| Total Transmit Carrier Sense errors |
| Transmit Utilization Counter |

## 4.13 INTERRUPT PROCESSING:

The XMAC II will generate an interrupt when certain events occur. These events are defined in the **Interrupt Status register**. Interrupts due to certain events can be prevented through the use of **Interrupt Status Mask register**. For a particular event to generate an interrupt, the corresponding bit(s) in the **Interrupt Status register** need to be set and the appropriate bit in the **Interrupt Status Mask register** need to be cleared. At reset, the **Interrupt Status Mask register** is set to all 1's and hence no interrupt will occur. When a hardware interrupt occurs, the driver first reads the **Interrupt Status register** to identify the primary source of interrupt. To identify the source of an interrupt due to some of the events, the driver may have to read an additional register to determine the root cause (for example: transmit/receive counter overflows). Ideally, when interrupt occurs the interrupt service routine reads the **Interrupt Status register** to identify the source of the interrupt. At that time, the **Interrupt Status register** bits get cleared. When the necessary registers are read, the hardware interrupt will be cleared.

For certain events the source of the interrupt resides in some other register and so before the user reads the source register of the interrupt event, it is possible that the same bit in **Interrupt Status register** will be set again. To ensure atomic processing of *all* interrupts, read the interrupt status register one more time before exiting the interrupt service routine.

## 4.14 VLAN SUPPORT

The XMAC II is able to handle VLAN tagged frames. Both one level VLAN with a 4 byte tag and two level VLAN with 20 byte tags are supported. There are two VLAN tag registers on the XMAC II which can be programmed for the match criteria. The XMAC II compares the 13$^{th}$ and the 14$^{th}$ byte of received frames for a match. Maximum frame size of 1522 and 1538 bytes are automatically accounted for by the XMAC II.

## 4.15 MAC-CONTROL FRAME GENERATION

The **Control Parameter register** allows the definition of the 4-byte data field so that any MAC-control frame can be generated in addition to the standard "Pause" frame.

## 4.16 PROGRAMMABLE IPG CONTROL

The XMAC II also provides a unique feature for rate control. The XMAC II allows frame-by-frame IPG control. The TXIPGValid hardware signal pin can be used to vary the IPG parameter along with frame data on a frame by frame basis. Using this feature frame transmission rate can be controlled. When this signal is not used the XMAC II uses the IPG value based upon the IPG register content.

With the XMAC II, there are two ways to control the IPG between packets. The simplest way to do this is to program the **TransmitIPG** register with the desired IPG value. This register is 32-bits wide, but only the least significant 8-bits are used (the remaining bits are ignored). The following expression determines the actual IPG that will be transmitted by XMAC II between packets:

[INT (reg_value/2)] * 32 + 64

The only exception to this formula, is the *reg_value* = 0. When *reg_value* = 0, then the actual IPG transmitted is 32 bit times (rather than the computed value of 64 bit times). Once the **TransmitIPG** register is programmed with a value (the default value, at reset, being 3 corresponding to an IPG of 96 bit times), XMAC II will transmit all packets with the same IPG (assuming that the host providing data to XMAC II at a speed equal to or more than the wire speed). However, this method is not useful, if the system needs to change the IPG between packets (rather than for a burst of packets).

The second method allows the system to change the IPG between consecutive packets with a constraint. In this mode of operation, the system provides an encoded version of the required IPG value through the *HostTxData(31:0)* pins along with a *TxIPGValid* signal at the end of each packet as it is transferred into the TxFIFO. The constraint is as follows:

The transmitter consists of two parts:

- TxFIFO logic
- MAC logic

A packet that is fed to the *HostTxData(31:0)* pins goes through the TxFIFO logic before entering the MAC logic. However, the IPG information of a given packet is fed directly to the MAC logic and so care should be taken to make sure that the second IPG value (which is used at the end of the second packet) does not overwrite the IPG value desired for the first packet (which is kept in the XMAC II until the end of the transmission of the first packet).

## 4.17 FORCE TRANSMIT FUNCTION

The Force Transmit function generates a preamble pattern that will create a collision in Half-Duplex mode. This function can be used to implement a form of flow control, that in other network technologies is known as 'back pressure'.

When the *ForceXmt* signal is asserted:

- The controller waits for a packet to be received
- It then waits for the programmed duration of the backoff value, *POFF_REG_VAL[15:0]* * 32 nanoseconds
- The XMAC II will then transmit a preamble pattern, causing a collision event on the shared network domain.

## 4.18 XMAC II TRANSPARENT MODE

### 4.18.1 Transmit Side

To enable transparent mode of operation for the transmit side, assert **TxTranspMode** pin and set the *xEncBypass* bit (bit #5) in the **Transmit Command Register**.

When this is done, the 20 least significant bits are used and XMAC II ignores the 12 most significant bits.

The 20 least significant bits are treated as two 10-bit characters (say AB). Then XMACII will transmit them as ABAB.

So if the data presented on the **HostTxData(31:0)** pins are as follows:

xxAB
xxCD
xxEF

then the following pattern will go out on the wire:

AB AB CD CD EF EF

It is not possible just to set the *xEncBypass* bit alone and send out a well-formed Ethernet packet in unencoded format.

### 4.18.2 Receive Side

To enable transparent mode of operation for the receive side, set *transparentMode* bit (bit #5) in the **ReceiveCommand register**.

When this is done, XMAC II will not disable 8B10B decoding function. However it will give out decoded control and data characters.

## 4.19 MEDIA INTERFACE

### 4.19.1 8B10B PCS Encoder/Decoder

The XMAC II Controller includes an integral 8B10B Encoder and Decoder (ENDEC). The 8B Data Bytes and the 10B Symbols are defined by the IEEE 802.3z specification and the 8B10B encoding technique is used for fiber optic transmission and short haul copper links.

#### *4.19.1.1 Transmit Encoding*

The Controllers 8B10B Encoder accepts MAC format frames on the Tx interface from the MAC

- Encodes the frame according 8B10B encoding rules
- Appends a Start-of-Frame Delimiter (SOF)
- Prepends a End-of-Frame Delimiter (EOF)
- Adds Idle code, as necessary
- Formats and outputs the frame in 10B PHY format which is passed to the 10-bit Gigabit Ethernet PHY transceiver interface.

Link control and configuration messages also pass through the 8B10B Encoder during the Link Autonegotiation process.

#### *4.19.1.2 Receive Decoding*

The Controller's PCS block also decodes the 8B10B encoded data from the 10-bit PHY interface

- Uses the SOF to synchronize on the data
- Strips the SOF and EOF delimiters
- Removes Idle code characters
- Decodes the data according to 8B10B decoding rules
- Passes the MAC frames to the Rx interface of the MAC

Link control and configuration messages also pass through the 8B10B Decoder during the Link Autonegotiation process.

## 4.20 PHY INTERFACE

### 4.20.1 8-bit GMII Mode

The 8-bit Gigabit Media Independent Interface mode is also programmable between the PCS block and the Physical Media. This interface also conforms to IEEE 802.3z specifications for GMII.

The PHY GMII mode consists of twenty-six (26) signals, eight transmit bits (*TXD(7:0)*); an output clock (*TCLK*); an output management data clock (*MDC*); transmit enable output (*TX_EN*); transmit error output (*TX_ER*); management data control and status input/output (*MDIO*); eight receive bits (*RXD(7:0)*); a receive clock signal (*RCLK*); receive data valid input (*RX_DV*); receive error input (*RX_ER*); and two async input signals for collision detection and carrier sense (*COL, CRS*).

The transceiver IC latches *TXD (7:0)* using the rising edge of *TCLK*, which is provided by the XMAC II. The XMAC II latches *TXD (7:0)* using *RCLK*, which is provided by the transceiver IC

### 4.20.2 10-bit FC-0 Mode

The 10-bit FC-0 interface is established specification between the PCS block and the Physical Media. The Controller's 10-bit FC-0 interface conforms to the IEEE 802.3z specifications for 1.25 Gbps Gigabit Ethernet transceivers (SERDES) and has been demonstrated to work with devices such as the: Vitesse 7135; AMCC S2052 and the HP HDMP-1536 or 1546. These components can be connected directly to the Controller, without any external logic.

The FC-0 interface consists of twenty-seven (27) signals, ten transmit bits (*PODAT[9:0]*); an output clock (*GTX_CLK*); enable comma detect output (*EN_COM_DET*); ten receive signals (*PIDAT[9:0]*); two receive clock signals (*RBCLK, RBCLK_N*); a SERDES comma detection input signal (*COM_DET*); a PHY loopback signal (*PHY_LP_EN*); and a receive link reference signal (*/LINK_SYNC*).

The transceiver IC latches *PODAT(9:0)* using the rising edge of *GTX_CLK*, which is provided by the XMAC II. The XMAC II latches *PIDAT(9:0)* using *RBCLK & RBCLK_N*, which is provided by the transceiver IC.

**Figure 14 - XMACII PHY Interface Mode Pin Comparison**

| Pin # | FC-0 Signal Name | GMII Signal Name |
|---|---|---|
| 83 | PIDAT0 | RXD(0) |
| 84 | PIDAT1 | RXD(1) |
| 87 | PIDAT2 | RXD(2) |
| 88 | PIDAT3 | RXD(3) |
| 89 | PIDAT4 | RXD(4) |
| 90 | PIDAT5 | RXD(5) |
| 91 | PIDAT6 | RXD(6) |
| 93 | PIDAT7 | RXD(7) |
| 94 | PIDAT8 | RX_DV |
| 95 | PIDAT9 | RX_ER |
| 97 | GTX_CLK | TCLK |
| 99 | /LINK_SYNC | /LINK_SYNC |
| 100 | EN_COMDET | Reserved |
| 101 | PHY_LP_EN | Reserved |
| 103 | PODAT0 | TXD(0) |
| 105 | PODAT1 | TXD(1) |
| 107 | PODAT2 | TXD(2) |
| 109 | PODAT3 | TXD(3) |
| 111 | PODAT4 | TXD(4) |
| 113 | PODAT5 | TXD(5) |
| 115 | PODAT6 | TXD(6) |
| 117 | PODAT7 | TXD(7) |
| 119 | PODAT8 | TX_EN |
| 121 | PODAT9 | TX_ER |
| 123 | RBCLKN | Reserved |
| 125 | RBCLK | Reserved |
| 129 | Reserved | RCLK |
| 134 | COM_DET/SIG_DET | CRS |
| 135 | Reserved | MDIO |
| 138 | Reserved | MDC |
| 229 | Reserved | COL |

## 4.21 ACCESSING PHY REGISTERS

All accesses to registers in the PHY are done through the PHY address and PHY data registers. The PHY address register is used to select a specific PHY register for read/write operations.

### 4.21.1 PHY READ Operation

To read a PHY register, first write the address of the PHY register into PHY Address register and then do a read operation on the PHY Data register.

### 4.21.2 PHY WRITE Operation

To write to a PHY register, first write the address of the PHY register into the PHY Address register and then do a write operation on the PHY Data register with the data to be written.

## 4.22 LINK CONFIGURATION - LINK AUTONEGOTIATION

The following sequence of events is required to setup autonegotiaiton. This sequence does not include optional *NextPage* configuration.

1. Wait for an interrupt.
2. Read the *Interrupt Status* register.
3. If the *AutoNegDone* bit is set, then read the **Link Partner Ability** and **PHY Resolved Ability registers** and proceed to Step 4).
4. If any of the remote fault bits are set in the **Link Partner Ability register** then proceed accordingly (see description of remote fault bits for more information)..
5. If both *hdplx* and *fdplx* resolved abilities in the **PHY Resolved Ability Register** are zero, do not enable either the transmitter or the receiver. Likewise, if *receivePause* abilities (available in the **PHY Resolved Ability register** as well as **Link Partner Ability register**) are not consistent, do not enable the transmitter.
6. If the autonegDone interrupt came while the link is operational (may be because reconfigure command on the link partner's side), process the link configuration, as described above.

## 4.23 CONFIGURATION ERROR DURING AUTONEGOTIATION

Whenever interrupt due to *AutoNegDone* (bit #7) happens, user should read the contents of **Autonegotiation Advertisement register** and the **PHY ResolvedAbility register** and do the following check to see if Duplex (or) Pause mismatch occurred using the following conditions:

Duplex Mismatch occurs when the following condition happens:

1. *HDModeSelected (Bit #6)* and *FDModeSelected* (Bit #5) bits are not set in ***PHY ResolvedAbility register.***

PAUSE Mismatch occurs when one of the following four conditions happen:

1. *PauseBit2* and *PauseBit1* (Bit #8 & Bit #7) in ***Autonegotiation Advertisement register*** are reset and *PauseBit2* (Bit #8) or *PauseBit1* (Bit #7) in **PHY ResolvedAbility register** is set.
2. *PauseBit1* is reset and *PauseBit2* is set in **Autonegotiation Advertisement register** and *PauseBit2* (Bit #8) and *PauseBit1* (Bit #7) in **PHY ResolvedAbility register** is reset.
3. *PauseBit1* is set and *PauseBit2* is reset in **Autonegotiation Advertisement register** and *PauseBit1* (Bit #7) in **PHY ResolvedAbility register** is reset.
4. *PauseBit2* and *PauseBit1* in the **Autonegotiation Advertisement register** are set and *PauseBit1* (Bit #7) in **PHY ResolvedAbility register** is reset.

# 5.0 Register Descriptions

This section describes the XQ11800FP registers. The registers are divided into two groups:

- Control and Status Registers
- RMON/SNMP network statistics counters

## 5.0.1 Register Conventions

In the register description tables that follow, the following abbreviations are employed to indicate register access modes:

| | | |
|---|---|---|
| R | = | Readable only |
| W | = | Writable only |
| R/W | = | Readable and Writable |

## 5.0.2 Register Access Rules

- Unlisted addresses are reserved and must not be accessed.
- Reserved addresses must not be accessed.
- Reserved bits on registers must be written as “0” and are ignored on read.
- Multibyte registers are ordered from low to high (the lower address points to the lower byte) [i.e., Little Endian format].
- Certain registers and register bits are cleared upon read access.
- Certain register bits are self-clearing upon write operation (self-clearing).
- For ‘self-clearing’ bits check the bit to be “0” before doing subsequent write operation (s).

## 5.1 XMAC II CONTROL AND STATUS REGISTER LISTING

| Name | Bits | Address 32bit Mode | Address 16bit Mode | Operation | Section |
|---|---|---|---|---|---|
| MMU Command Register | 16 | 0 | 0 | Read/Write | 6.0.1 |
| Reserved | -- | 4 | 4 | Read/Write | |
| POFF Register | 16 | 8 | 8 | Read/Write | 6.0.2 |
| BURST Register | 16 | C | C | Read/Write | 6.0.3 |
| OneLevelVLANTag Register | 16 | 10 | 10 | Read/Write | 6.0.4 |
| TwoLevelVLANTag Register | 16 | 14 | 14 | Read/Write | 6.0.5 |
| Reserved | -- | 18-1C | 18-1C | N/A | |
| Transmit Command Register | 16 | 20 | 20 | Read/Write | 6.0.6 |
| Transmit Retry Limit Register | 16 | 24 | 24 | Read/Write | 6.0.7 |
| Transmit Slottime Register | 16 | 28 | 28 | Read/Write | 6.0.8 |
| Transmit IPG Register | 16 | 2C | 2C | Read/Write | 6.0.9 |
| Receive Command Register | 16 | 30 | 30 | Read/Write | 6.0.10 |
| PHY Address Register | 16 | 34 | 34 | Read/Write | 6.0.11 |
| PHY Data Register | 16 | 38 | 38 | Read/Write | 6.0.12 |
| Reserved | -- | 3C | 3C | N/A | |
| GeneralPurposePort Register | 16 | 40 | 40 | Read/Write | 6.0.13 |
| Interrupt Mask Register | 16 | 44 | 44 | Read/Write | 6.0.14 |
| Interrupt Status Register | 16 | 48 | 48 | Read Only | 6.0.15 |
| Hardware Configuration Register | 16 | 4C | 4C | Read/Write | 6.0.16 |
| Reserved | -- | 50-5C | | N/A | |
| Transmit Low Water Mark Register | 16/32 | 60 | 60 | Read/Write | 6.0.17 |
| Transmit High Water Mark Register | 16 | - | 62 | | |
| TransmitRequestThreshold Register | 16/32 | 64 | 64 | Read/Write | 6.0.18 |
| TransmitRequestThreshold Register | 16 | - | 66 | | |
| PauseDestinationAddress1 Register | 16/32 | 68 | 68 | Read/Write | 6.0.19 |
| PauseDestinationAddress2 Register | 16 | - | 6A | | |
| PauseDestinationAddress3 Register | 16/32 | 6C | 6C | | |
| Reserved | -- | 6E | 6E | N/A | |
| ControlParameterLow Register | 16/32 | 70 | 70 | Read/Write | 6.0.20 |
| ControlParameterHigh Register | 16 | - | 72 | | |
| Opcode/PauseTimer Register | 16 | 74 | 74 | Read/Write | 6.0.21 |
| TransmitStatusLIFO Register | 16/32 | 78 | 78 | Read Only | 6.0.22 |
| ExactMatch0Address1/2 Registers | 32 | 80 | 80, 82 | Read/Write | 6.0.23 |
| ExactMatch0Address3 Register | 16 | 84 | 84 | | |
| ExactMatch1/2Address1 Registers | 32 | 88 | 88, 8A | Read/Write | 6.0.23 |
| ExactMatch1Address3 Register | 16 | 8C | 8C | | |
| ExactMatch2Address1/2 Registers | 32 | 90 | 90,92 | Read/Write | 6.0.23 |
| ExactMatch2Address3 Register | 16 | 94 | 94 | | |

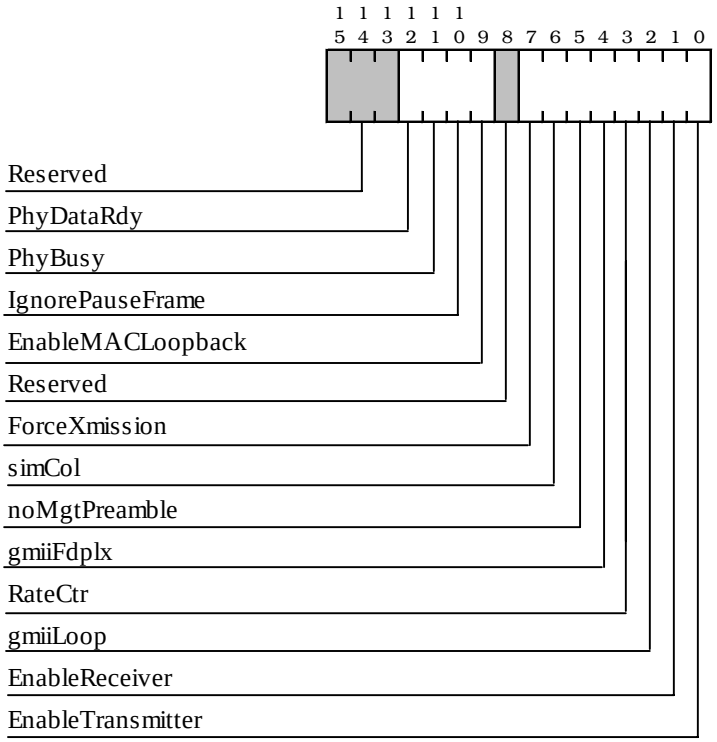
| Name | Bits | Address 32bit Mode | Address 16bit Mode | Operation | Section |
|---|---|---|---|---|---|
| ExactMatch3Address1/2 Registers | 32 | 98 | 98, 9A | Read/Write | 6.0.23 |
| ExactMatch3Address3 Registers | 16 | 9C | 9C | | |
| ExactMatch4Address1/2 Registers | 32 | A0 | A0, A2 | Read/Write | 6.0.23 |
| ExactMatch4Address3 Registers | 16 | A4 | A4 | | |
| ExactMatch5Address1/2 Registers | 32 | A8 | A8, AA | Read/Write | 6.0.23 |
| ExactMatch5Address3 Registers | 16 | AC | AC | | |
| ExactMatch6Address1/2 Registers | 32 | B0 | B0, B2 | Read/Write | 6.0.23 |
| ExactMatch6Address3 Registers | 16 | B4 | B4 | | |
| ExactMatch7Address1/2 Registers | 32 | B8 | B8, BA | Read/Write | 6.0.23 |
| ExactMatch7Address3 Registers | 16 | BC | BC | | |
| ExactMatch8Address1/2 Registers | 32 | C0 | C0, C2 | Read/Write | 6.0.23 |
| ExactMatch8Address3 Registers | 16 | C4 | C4 | | |
| ExactMatch9Address1/2 Registers | 32 | C8 | C8, CA | Read/Write | 6.0.23 |
| ExactMatch9Address3 Registers | 16 | CC | CC | | |
| ExactMatch10Address1/2 Registers | 32 | D0 | D0, D2 | Read/Write | 6.0.23 |
| ExactMatch10Address3 Registers | 16 | D4 | D4 | | |
| ExactMatch11Address1/2 Registers | 32 | D8 | D4, D6 | Read/Write | 6.0.23 |
| ExactMatch11Address3 Registers | 16 | DC | D8 | | |
| ExactMatch12Address1/2 Registers | 32 | E0 | E0, E2 | Read/Write | 6.0.23 |
| ExactMatch12Address3 Registers | 16 | E4 | E4 | | |
| ExactMatch13Address1/2 Registers | 32 | E8 | E8, EA | Read/Write | 6.0.23 |
| ExactMatch13Address3 Registers | 16 | EC | EC | | |
| ExactMatch14Address1/2 Registers | 32 | F0 | F0, F2 | Read/Write | 6.0.23 |
| ExactMatch14Address3 Registers | 16 | F4 | F4 | | |
| ExactMatch15Address1/2 Registers | 32 | F8 | F8, FA | Read/Write | 6.0.23 |
| ExactMatch15Address3 Registers | 16 | FC | FC | | |
| SourceCheckAddress1/2 Register | 32 | 100 | 100, 102 | Read/Write | 6.0.24 |
| SourceCheckAddress3 Register | 16 | 104 | 104 | | |
| StationAddress1/2 Register | 32 | 108 | 108, 10A | Read/Write | 6.0.25 |
| StationAddress3 Register | 16 | 10C | 10C | | |
| HashMatchAddress1/2 Register | 32 | 110 | 110, 112 | Read/Write | 6.0.26 |
| HashMatchAddress3/4 Register | 32 | 114 | 114, 116 | | |
| Receive Low/High Water Mark Register | 32 | 118 | 118, 11A | Read/Write | 6.0.27 |
| ReceiveRequestThreshold Register | 16/32 | 11C | 11C | Read/Write | 6.0.28 |
| DeviceID Low/High Register | 32 | 120 | 120, 122 | Read Only | 6.0.29 |
| Mode Low/High Register | 32 | 124 | 124, 126 | Read/Write | 6.0.30 |
| LastSrcAddress1/2 Register | 32 | 128 | 128, 12A | Read Only | 6.0.31 |
| LastSrcAddress3 Register | 16 | 12C | 12C | | |

## 5.1 XMAC II CONTROL AND STATUS REGISTER LISTING (CONT.)

| Name | Bits | Address | | Operation | Section |
|---|---|---|---|---|---|
| | | 32bit Mode | 16bit Mode | | |
| Time Stamp Read Register | 16 | 130 | 130 | Read Only | 6.0.32 |
| Time Stamp Load Register | 16 | 134 | 134 | Read/Write | 6.0.33 |
| StatisticsCommand Register | 16 | 200 | 200 | Read/Write | 6.0.34 |
| ReceiveCounterEventLow/High Register | 32 | 204 | 204, 206 | Read Only | 6.0.35 |
| TransmitCounterEventLow/High Register | 32 | 208 | 208, 20A | Read Only | 6.0.36 |
| ReceiveCntrEventMaskLow/High Register | 32 | 20C | 20C, 20E | Read/Write | 6.0.37 |
| TransmitCntrEventMaskLow/High Register | 32 | 210 | 210, 212 | Read/Write | 6.0.38 |
| PHY Control Register | 16 | Indirect 0 | | Read/Write | 6.0.39 |
| PHY Status Register | 16 | Indirect 1 | | Read Only | 6.0.40 |
| PHY ID0 Register | 16 | Indirect 2 | | Read Only | 6.0.41 |
| PHY ID1 Register | 16 | Indirect 3 | | Read Only | 6.0.42 |
| Autonegotiation Advertisement Register | 16 | Indirect 4 | | Read/Write | 6.0.43 |
| Link Partner Ability Register | 16 | Indirect 5 | | Read Only | 6.0.44 |
| Autonegotiation Expansion Register | 16 | Indirect 6 | | Read Only | 6.0.45 |
| Next Page Register | 16 | Indirect 7 | | Read/Write | 6.0.46 |
| Link Partner Next Page Register | 16 | Indirect 8 | | Read Only | 6.0.47 |
| Extended Status Register | 16 | Indirect 15 | | Read/Write | 6.0.48 |
| PHY Resolved Ability Register | 16 | Indirect 16 | | Read Only | 6.0.49 |

# 6.0 XMAC II Detailed Register Descriptions

## 6.0.1 MMU Command Register



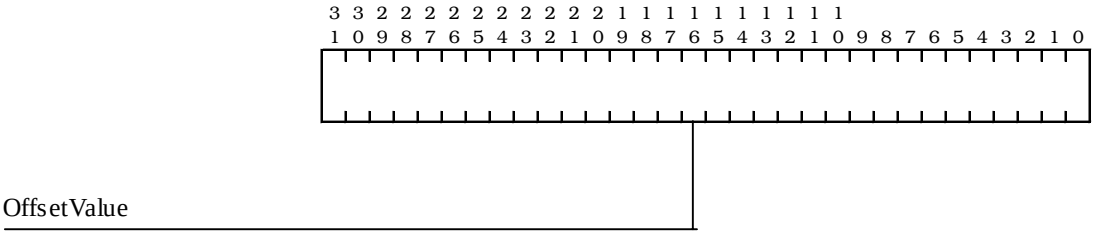

*Default Register Value =0x00000000H* *Register Access = Read/Write*
*Register Address = 0H*

| Bit | Field | Description |
|---|---|---|
| 15-13 | Reserved | |
| 12 | PhyDataRdy | This bit (when set) indicates that the earlier read operation of the external PHY is complete and CPU can read the data. |
| 11 | PhyBusy | This bit (when set) indicates that the external PHY is being read from (or) being written into. |
| 10 | IgnorePauseFrame | If this bit is set, then upon receiving a Pause frame the XMAC II will not inhibit the transmitter from sending frames. At reset, this bit is cleared. |
| 9 | EnableMACLoopback | If this bit is set, then the XMAC II will loopback the data from the MAC transmit block to the MAC receive block. Note that the encoder/decoder is not part of the loopback path. |
| 8 | Reserved | |
| 7 | ForceXmission | If this bit is '0', when *ForceXmit* pin is asserted, then XMAC II will enforce collisions. If this bit is '1', when *ForceXmit* pin is asserted, then XMAC II will enforce carrier pressure (i.e., back pressure). *This bit is useful only in Half Duplex.* |

## 6.0.1 MMU Command Register, ont'd.

| | | |
|---|---|---|
| 6 | simCol | This bit can be used to simulate local collisions for diagnostics. The offset (from the beginning of a packet) for collision = **POFF** (register value) * 32 bit times (nsecs). *This bit is useful in Half Duplex only.* |
| 5 | noMgtPreamble | If this bit is set, the XMAC II will not send preamble (for transmit) on MDC/MDIO interface. Also, it is assumed that PHY does not require preamble. |
| 4 | gmiiFdplx | User needs to set this bit only if XMAC II needs to operate in Full Duplex operation under GMII Mode. If cleared, XMAC II will operate in Half Duplex operation and GMII Mode. |
| 3 | RateCtr | This bit must be set to enable the generation of 'Phantom' Pauses as a 'coarse' form of rate control. The duration of the pause is determined by the offset value in the POFF register. (Full-duplex only) |
| 2 | gmiiLoop | If this bit is set to indicate to XMAC II that the external PHY is being placed in loopback mode of operation. This will be useful for sending and receiving packets at the same time while configured in half-duplex and PHY loopback mode (i.e., XMAC II will ignore the collision signal). This bit performs the same function as *rselfReceive* (Bit # 3) in the **Receive Command** register. |
| 1 | EnableReceiver | When this bit is set, the receive state machine is enabled |
| 0 | EnableTransmitter | When this bit is set, the transmit state machine is enabled |

### 6.0.2 POFF Register

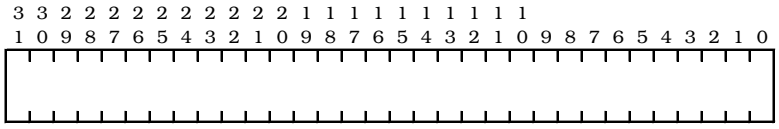



*Default Register Value = 0x00000FD1H* *Register Access = Read/Write*
*Register Address = 8H*

The POFF (Packet Offset) Register is used by the XMAC II to provide an offset (in 32ns increments) for one of several mutually exclusive functions:

1. To enforce a collision in half duplex mode. After receiving the number of 16-bit words indicated by the value of the register, the transmitter is forced to begin transmitting. The information that is sent is the Preamble followed by SFD. The collision is generated by the *ForceCol* bit of the **MMU Command register**.
2. To enforce a "phantom" pause for 'coarse' rate control in full-duplex mode. When the *RateCtr* bit of the **MMU Command register** is set, a Phantom Pause can be enforced. The duration of the Pause is given by the contents of the **POFF register.**

### 6.0.3 Burst Register

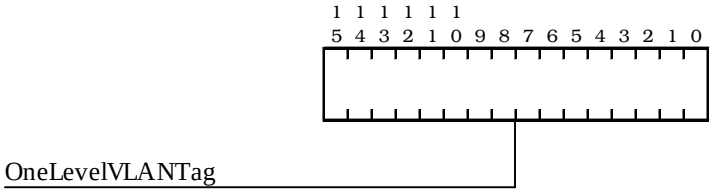



*Default Register Value = 0x00000FD1H* *Register Access = Read/Write*
*Register Address = CH*

.

The Burst Register is used in half duplex mode to store the burst limit of the XMAC II. This register is used to determine whether another packet can be burst in the current sequence. The value programmed in this register is the burst limit minus 1 in units of 16-bits. It is checked by XMAC II before beginning the burst. The default value is equivalent to an 8000-byte period.

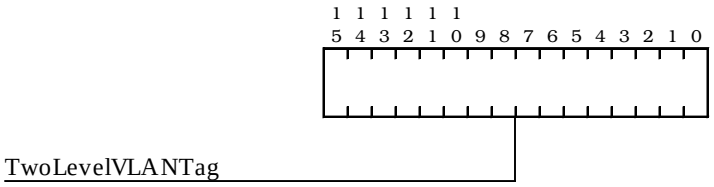
### 6.0.4 OneLevelVLANTag Register




*Default Register Value = 0x00008100H* *Register Access = Read/Write*
*Register Address = 10H*

This register can be programmed with a 2-byte one level VLAN Tag ID. When frames are received, the 13th and 14th bytes in the frame are compared to this register to check if the frame is tagged with a one-level VLAN ID. If there is a match, then the XMAC II will receive this frame if the length is up to 1522 bytes without generating a LongPkt error.

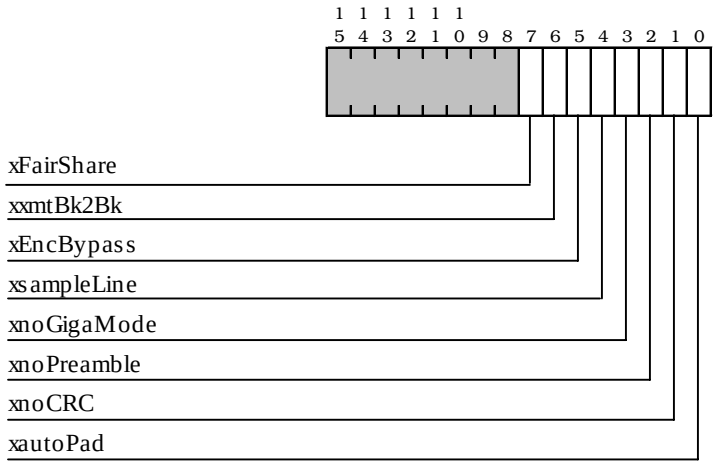
### 6.0.5 TwoLevelVLANTag Register




*Default Register Value = 0x0000FFFFH* *Register Access = Read/Write*
*Register Address = 14H*

This register can be programmed with a 2-byte two level VLAN Tag ID. When frames are received, the 13th and 14th bytes in the frame are compared to this register to check if the frame is tagged with a two-level VLAN ID. If there is a match, then the XMAC II will receive this frame if the length is up to 1538 bytes without generating a LongPkt error.

### 6.0.6 Transmit Command Register

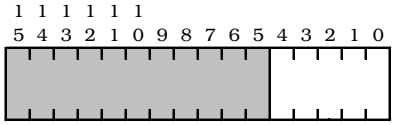



*Default Register Value = 0x00000000H* *Register Access = Read/Write*
*Register Address = 20H*

| Bit | Field | Description |
|---|---|---|
| 15-7 | Reserved | |
| 7 | xFairShare | When set in Half Duplex Mode, this bit allows the fair sharing of the medium. Without this mechanism, the regular 802.3 backoff mechanism shows serious capture effect in simulation. |
| 6 | xxmtBk2Bk | If this bit is set, then the Controller ignores Carrier Sense for the purpose of calculating IPG while sending packets back-to-back. Applicable to Half Duplex only. Generally, per the standard, a MAC Client is expected to defer to the CarrierSense signal before transmitting. In 1000 Mbps implementations it is impossible to Tx with the minimum IPG of 96 bit times, if the controller defers to Carrier Sense. Setting this bit causes the transmitter to check IPG with respect to its own transmission only. |
| 5 | xEncBypass | If this bit is set, then the 8B/10B Encoder [but not the Decoder] will be bypassed. This bit has no affect if *TxTranspMode* signal is not asserted. |
| 4 | xsampleLine | The XMAC II controller computes transmit and receive utilization on a continuous basis . *See Management Counter Section* . To start the computation of average utilization over a certain period, this bit would be set [transmit utilization computation is reset to zero]. This bit is self-clearing. |
| 3 | xnoGigaMode | When this bit is set to 1, the XMAC II disables carrier extension. This bit determines the Gigabit mode - if it is set, then the slot time is adjusted to 512 bits instead of 512 bytes. Applicable to Half-Duplex only. |
| 2 | xnoPreamble | When this bit is set to ‘1’, the transmitter does not prepend preamble (or SFD) to the packet. Note that the first byte of the stream is always converted to the "start of packet" symbol. Thus if this bit is set the first octet of the datastream from the host is transformed into SOF. CRC generation should also be disabled when preamble is disabled and the system should compute and append CRC. |
| 1 | xnoCRC | When this bit is set, XMAC II does not append CRC to the end of the packet. |
| 0 | xautoPad | When this bit is set, XMAC II does automatic padding of short frames. When CRC is disabled, autopadding will generate only 60-byte packets. |

Transmit Command Register

### 6.0.7 Transmit Retry Limit Register

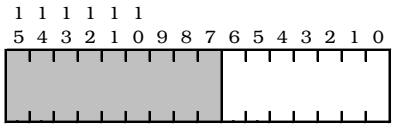



*Default Register Value = 0x0000000FH* *Register Access = Read/Write*
*Register Address = 24H*

| Bit | Field | Description |
|---|---|---|
| 4-0 | TxRetry Limit | Maximum number of times to attempt retransmission. For 1000 Mbps implementations the standard is fifteen (15) retransmissions following the initial transmission attempt. |

The TxRetryLimit register is used in Half Duplex mode to store the number of maximum retries in an Ethernet collision condition. It is automatically initialized to fifteen (15) retries (as required by the standard), but may be set to different values to make the device behavior more or less aggressive in collision conditions and may be used to evaluate shared network performance under varying collision retry limits.

### 6.0.8 Transmit Slottime Register

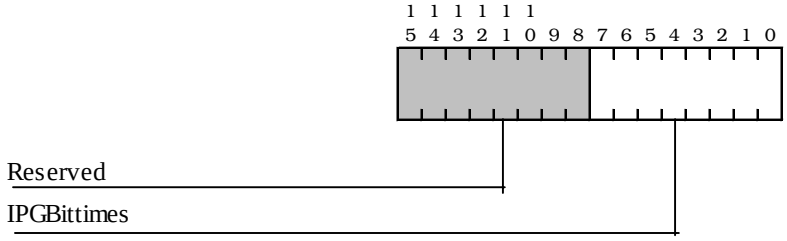



*Default Register Value = 0x000000FFH* *Register Access = Read/Write*
*Register Address = 28H*

| Bit | Field | Description |
|---|---|---|
| 7-0 | TxSlottime | For Half Duplex operations, the value of this field determines the period of time for collision handling ((value+1)*16), in bit times. For 1000 Mbps implementations the standard specifies a value of 4096 that is the default value of this register. |

The TxSlottime Register is used to define the duration of the Half Duplex (CSMA/CD) slotTime period, in bit times. For non-standard Collision behavior which may be more or less aggressive in collision conditions, different values may be programmed through this register.

### 6.0.9 Transmit IPG Register

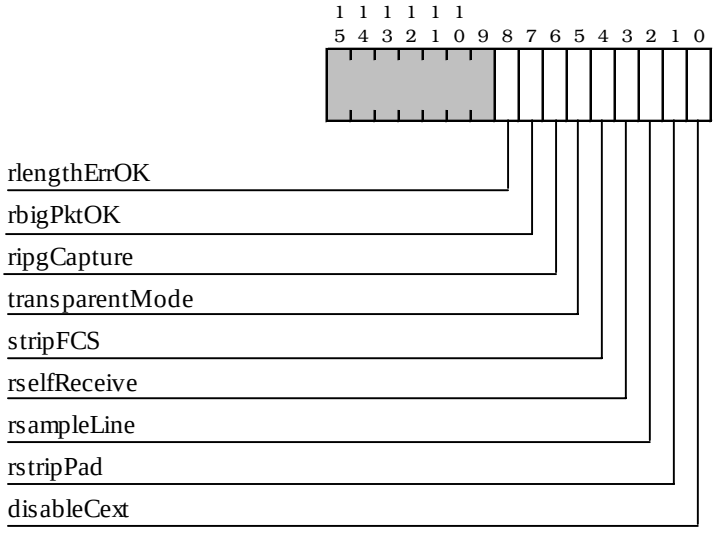



*Default Register Value = 0x00000003H* *Register Access = Read/Write*
[96-bit times]
*Register Address = 2CH*

.

| **Bit** | **Field** | **Description** |
|---|---|---|
| 15-8 | Reserved | |
| 7-0 | IPGBittimes | This register is used to store the value of IPG for the next packet to be transmitted. The following expression gives the relationship between the register value and the actual IPG.<br>Actual IPG (in nsecs) = [INT (IPGBittimes/2) ] * 32 + 64<br>*Note: If the register is programmed with a value of 0, then the IPG value is 32 bit times [32 nsecs], rather than the computed value of 64 bit times [64 nsecs].* |

### 6.0.10 Receive Command Register



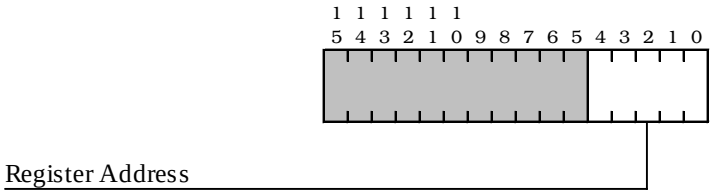

*Default Register Value = 0x00000008H* *Register Access = Read/Write*
*Register Address = 30H*

| **Bit** | **Field** | **Description** |
|---|---|---|
| 15-9 | Reserved | |
| 8 | rlengthErrOK | If this bit is set, then the XMAC II will not indicate rcvError for inrange length error packets for that reason alone. The individual status bit will still indicate this error condition. These packets will get counted as 'good' received packets (unless some other error is present). |
| 7 | rbigPktOK | If this bit is set, then the XMAC II will not indicate rcvError for big packets for size reasons alone. The individual status bit will still indicate that this is a big packet. These packets will get counted as 'good' received packets (unless some other error is present). |
| 6 | ripgCapture | If this bit is set, then the XMAC II will replace the byte count with the IPG value for each packet. This setting affects **both** the value in the status word and in the receive statistics. The IPG values is in units of 8 bit times with a resolution of 16 bits. Using this setting, the host can use the statistical counter to compute average IPG value. *NOTE1: If the IPG values is beyond 2**14**8 bit times, the values make not provide usable information. NOTE2: In Half Duplex mode, in the first packet of the burst (or the single non-burst packet), the IPG is measured after the end of the packet, if the packet is equal to or greater than 512 Bytes or at the end of a 512 Byte limit, if the packet is smaller. In the later case, the IPG measured is 2 (16 ns) more than the real IPG.* |
| 5 | transparentMode | If this bit is set, the transparent mode is enabled for receive operations. *See Section 4.8.10.2, Transparent Mode.* |
| 4 | stripFCS | If this bit is set, then the XMAC II will strip the FCS from incoming frames before passing them to the host. |

## 6.0.10 Receive Command Register, con'd

| | | |
|---|---|---|
| 3 | rselfReceive | Applicable in Half Duplex only. If this bit is set, reception of own packets is enabled. |
| 2 | rsampleLine | The XMAC II controller computes transmit and receive utilization on a continuous basis . See Management Counter Section. To start the computation of average utilization over a certain period, this bit would be set [receive utilization computation is reset to zero]. This bit is self-clearing. |
| 1 | rstripPad | When this bit is set, the XMAC II strips the pad bytes from incoming frames. |
| 0 | disableCext | When this bit is set, the XMAC II disables the check for carrier extension. (Half-duplex only) |

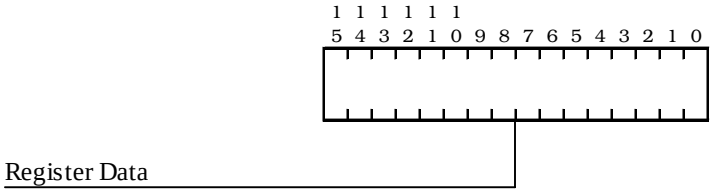
### 6.0.11 PHY Address Register




*Default Register Value = 0x00000000* *Register Access = Read/Write*
*Register Address = 34H*

| Bit | Field | Description |
|---|---|---|
| 15-5 | Reserved | |
| 4-0 | Register Address | These bits denote the indirect address of a PHY Register<br>00000 = PHY Control Register<br>00001 = PHY Status Register<br>00010 = PHY ID0 Register<br>00011 = PHY ID1 Register<br>00100 = AutoNegotiation Advertisement Register<br>00101 = Link Partner Ability Register<br>00110 = AutoNegotiation Expansion Register<br>00111 = Next Page Register<br>01000 = Next Page Link Partner Register<br>01001-01101 = Reserved<br>01111 = Extended Status Register<br>10000 = Resolved Ability Register |

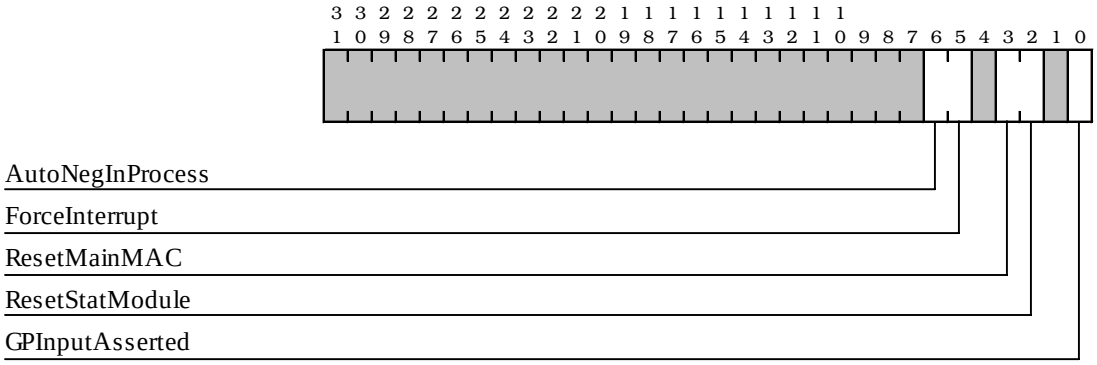
### 6.0.12 PHY Data Register




*Default Register Value = 0x00000000H* *Register Access = Read/Write*
*Register Address = 38H*

| **Bit** | **Field** | **Description** |
|---|---|---|
| 15-0 | Register Data | These bits contain the data that is to be written to or read from a given PHY registers |

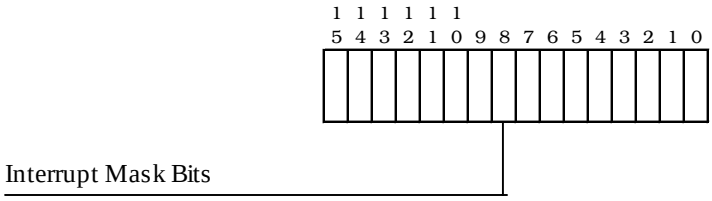
### 6.0.13 General Purpose Port Register




*Default Register Value = 00000040H* *Register Access = Read/Write*
*Register Address = 40H*

| **Bit** | **Field** | **Description** |
|---|---|---|
| 31-7 | Reserved | |
| 6 | AutoNegInProcess | If this bit is set, it indicates that the autonegotiation process is in progress. This is a read only bit. |
| 5 | ForceInterrupt | If this bit is set, then the ForceInterrupt bit in the Interrupt Status Register will be set. If the ForceInterrupt bit in the Interrupt Status Mask Register is also set, then a hardware interrupt will be generated. This bit is self-clearing. |
| 4 | Reserved | |
| 3 | ResetMainMAC | If this bit is set, then the MAC and FIFO modules will be reset. This bit is self-clearing. |
| 2 | ResetStatModule | If this bit is set, then the statistical module will be reset. This bit is self-clearing. |
| 1 | Reserved | |
| 0 | GPInputAsserted | This bit will be set if the GPInput pin is asserted by an external source and it stays set as long as GP Input stays asserted. If enabled, this bit will generate a hardware interrupt. This is a read only bit. |

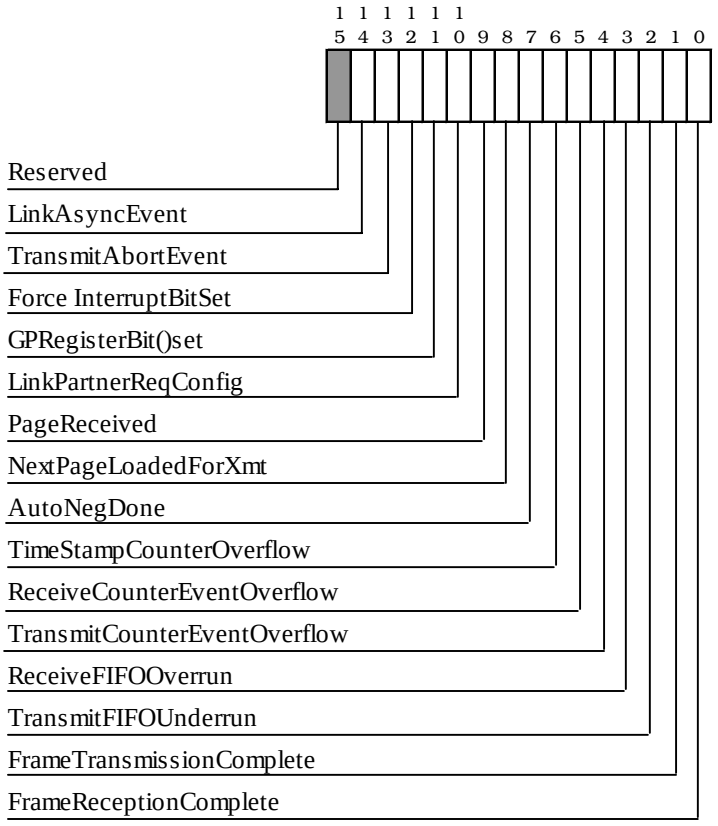
### 6.0.14 Interrupt Status Mask Register




*Default Register Value (at reset) = 0x0000FFFFH* *Register Access = Read/Write*
*Register Address = 44H*

This register can be used to enable /disable the interrupts due to the assertion of the corresponding bits in the Interrupt Status Register. A zero indicates that the interrupt is enabled for the corresponding transmit/receive event.

### 6.0.15 Interrupt Status Register




*Default Register Value = 0x4800H* *Register Access = Read Only*
*Register Address = 48H*

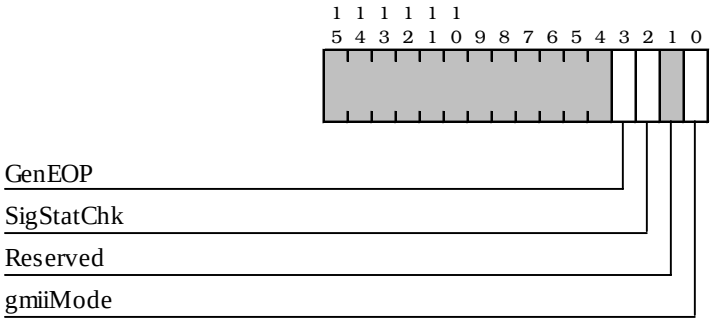
| **Bit** | **Field** | **Description** |
|---|---|---|
| 15 | Reserved | |
| 14 | LinkAsyncEvent | This bit is set, whenever the link loses synchronization |
| 13 | TransmitAbortEvent | This bit is set, whenever a Late Collision, Excessive Deferrals, or Excessive Collisions event occurs. (OR combination of LateCollision, ExcessiveCollisions and Excessive Deferral events) |
| 12 | ForceInterruptBitSet | When this bit is set, an interrupt was generated due to the ForceInterrupt bit being set in the General Purpose Register |
| 11 | GPRegisterBit0set | When this bit is set, an interrupt was generated due to bit 0 of the General Purpose Register being set. |
| 10 | LinkPartnerReqConfig | When this bit is set, an interrupt was generated due to the link partner requesting link configuration. |
| 9 | PageReceived | When this bit is set, a base or next page has been received. |
| 8 | NextPageLoadedForXmt | When this bit is set, the NextPage has been loaded for transmission |
| 7 | AutoNegDone | When this bit is set, autonegotiation process has been completed |
| 6 | TimeStampCounter Overflow | When this bit is set, the Timestamp Counter has overflowed |

Interrupt Status Register

## 6.0.15 Interrupt Status Register, cont'd.

| | | |
|---|---|---|
| 5 | ReceiveCounterEvent Overflow | When this bit is set, one (or more) bit(s) are set in the Receive Counter Event Register. |
| 4 | TransmitCounterEvent Overflow | When this bit is set, one (or more) bit(s) are set in the Transmit Counter Event |
| 3 | Receive FIFO Overrun | When this bit is set, an interrupt was generated due to an overflow condition in the Rx FIFO. |
| 2 | Transmit FIFO Underrun | When this bit is set, an interrupt was generated due to an underrun condition in the Tx FIFO |
| 1 | Frame Transmission Complete | When this bit is set, a packet was transmitted successfully |
| 0 | Frame Reception Complete | When this bit is set, a frame is copied into the RxFIFO successfully. |

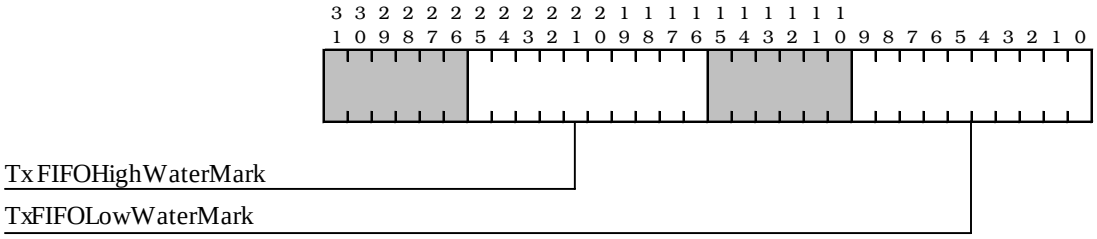
### 6.0.16 Hardware Configuration Register




*Default Register Value = 0x00000000H* *Register Access = Read/Write*
*Register Address = 4CH*

| **Bit** | **Field** | **Description** |
|---|---|---|
| 15-4 | Reserved | |
| 3 | GenEOP | If this bit is set to '1' (default value = 0), then XMAC II will put out a pulse on pin #153 (this pin is reserved in Rev. B XMAC II) a clock before *RxPktValid* goes off, indicating the EndOfPacket (EOP). The duration of pulse can be more than one HOST_CLK clock. |
| 2 | SigStatChk | If this bit is set to '1' (default value =0), then "Comma_Detect" input pin will be used as "Signal_Detect" input pin. |
| 1 | Reserved | |
| 0 | gmiiMode | If this bit is set to '1', then GMII Interface is selected for PHY Interface. If this bit is set to '0' (default), then FC-0 interface is selected for PHYinterface. |

### 6.0.17 Transmit Low/High Water Mark Register

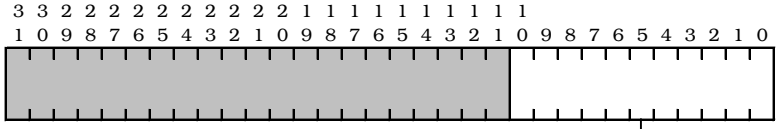



*Default Register Value = 03000100H* *Register Access = Read/Write*
*Register Address = 60H, 62H*

| **Bits** | **Field** | **Description** |
|---|---|---|
| 31-26 | Reserved | |
| 25-16 | TxFIFOHighWaterMark | These bits define the high water mark value, in 32-bit words. |
| 15-10 | Reserved | |
| 9-0 | TxFIFOLowWaterMark | These bits define the low water mark value in 32-bit words. |

The XMAC II will assert the *TxFIFOAlmostEmpty* signal when the number of bytes in the transmit FIFO is less than or equal to 4 times the TxFIFOLowWaterMark value. The *TxFIFOAlmostFull* signal will be asserted when the number of bytes in the transmit FIFO is more than 4 times the TXFIFOHighWaterMark value. In 16-bit mode, TxFIFOLowWaterMark is accessed at address 60h & *TxFIFOHighWaterMark* is accessed at address 62h.

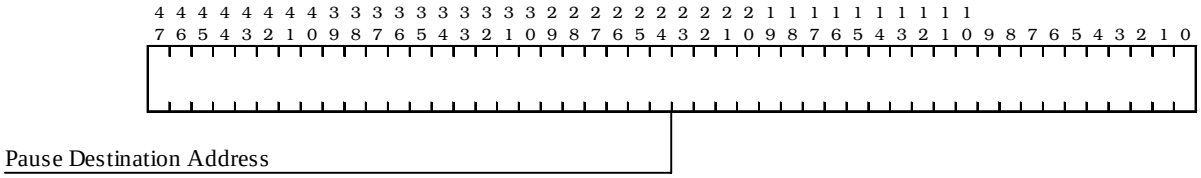
### 6.0.18 TransmitRequestThreshold Register




*Default Register Value = 0x03FB0004H* *Register Access = Read/Write*
*Register Address = 64H, 66H*

| Bit | Field | Description |
|---|---|---|
| 31-11 | Reserved | |
| 10-0 | TxReqThreshold | This determines the 'N' double-words (32-bits) that need to be transferred into the TxFIFO before the XMAC II initiates transmission. If this value larger than the size of a packet loaded into the TxFIFO, transmission will start, irrespective of this value. |

### 6.0.19 PauseDestinationAddress Register

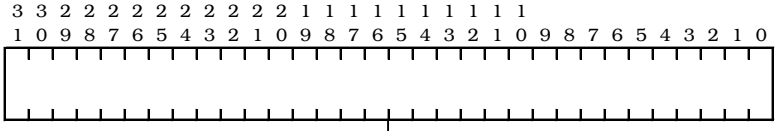



*Default Register Value = 010000C28001H  Register Access = Read/Write*
*Register Address = 68H, 6CH [In 32-bit mode]*
*Register Address = 68H, 6AH, 6CH [In 16-bit mode]*

These 16-bit registers define the destination address field of a MAC control frame that the XMAC II can generate upon command from the user (see bit #19 of Mode Register). This register set contains (3) registers and the default values (at reset) are as shown above. Note: this default value corresponds to the multicast address (01 80 C2 00 00 01) that is assigned to MACCtrlPauseFrames.

*Note: See general rules for Address Register Access, Reading and Writing Address Registers*

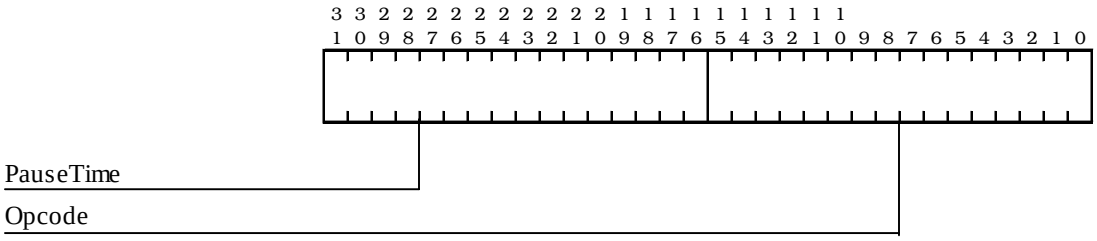
### 6.0.20 Control Parameter Register




*Default Register Value = 00000000H* *Register Access = Read/Write*
*Register Address = 70H, 72H*

This 32-bit register can be used to define the 4-byte field (Byte 22:19) of a MAC Control frame.

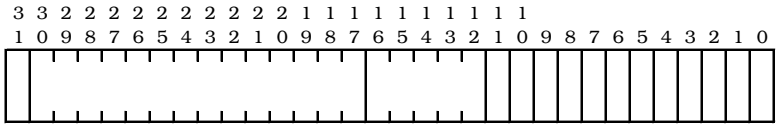
### 6.0.21 Opcode/Pause Timer Register




*Default Register Value = 00040100H* *Register Access = Read/Write*
*Register Address = 74H, 76H*

| Bit | Field | Description |
|---|---|---|
| 31-16 | PauseTime | These 16 bits define the time period [in the units of 512 ns] for which the Pause function is enabled (i.e. how long the transmitter is disabled for traffic other than MAC Control frames). At reset, this register is initialized to '00 04', which corresponds to a 512 ns period. |
| 15-0 | Opcode | These bits contain the 16-bit Opcode for MAC control frames. |

### 6.0.22 TransmitStatusLIFO Register

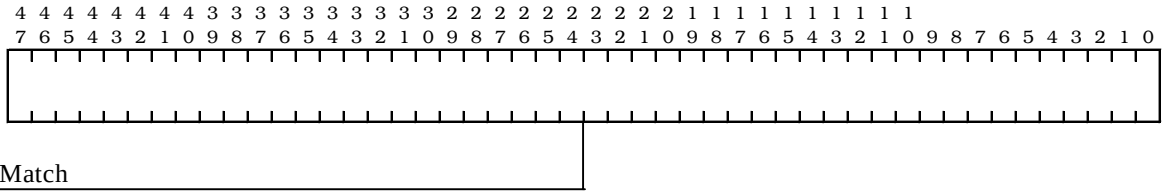



*Default Register Value = 00000000H* *Register Access = Read Only*
*Register Address = 78H*

This register is used to store that status of the most recently transmitted packets. The status is stored in Last-In First-out order. When a read operation is performed on this register, the most recent packet status is presented. Successive reads will produce the preceding packet status words (up to 3 packets). *Note: If there is no status the Status Valid will be "0".* The status words are represented as:

| Bit | Description |
|---|---|
| 31 | Status Valid |
| 30:17 | The byte count for the packet |
| 16:12 | Retry Count |
| 11 | Excessive Collisions (Abort Condition) |
| 10 | Excessive Deferral (Abort Condition) |
| 9 | Packet was transmitted in burst mode |
| 8 | Packet had to defer |
| 7 | Broadcast packet |
| 6 | Multicast packet |
| 5 | Unicast packet |
| 4 | Underflow in data supply (Abort Condition) |
| 3 | Carrier Sense Error |
| 2 | Late Collision Error (Abort Condition) |
| 1 | Multiple Collisions |
| 0 | Single Collision |

### 6.0.23 ExactMatch Registers




*Default Register Value = 000000000000H* *Register Access = Read/Write*
*Register Address = 80H - FCH*

If the user enables the exact address matching function (by setting bit #12 in the Mode Register), then the XMAC II compares the destination address field of an incoming frame with the contents of this register set. If there is a match, then the corresponding frame will be passed onto the Receive FIFO. and *MACAddrMatch* signal will be asserted for the corresponding packet.

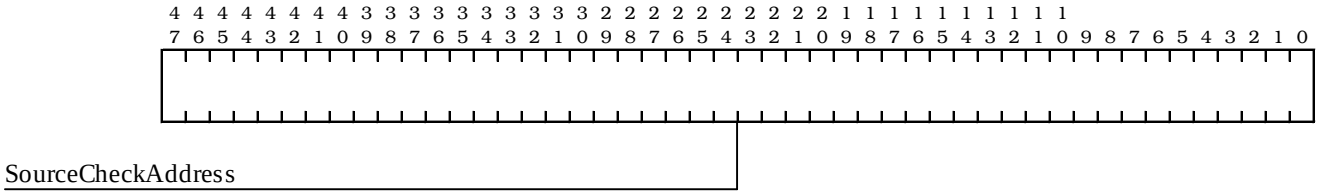
Otherwise the frame will be rejected. *Note that there are 16 different ExactAddressMatch register sets and these 16 different addresses can be matched perfectly (unlike hashing function which is imperfect).*

*Note 1: Writing a "0" to this register effectively disables the Exact Match Register function.*

*Note 2: See general rules for Address Register Access, Reading and Writing Address Registers*

Exact Address Match Register

### 6.0.24 SourceCheckAddress Register

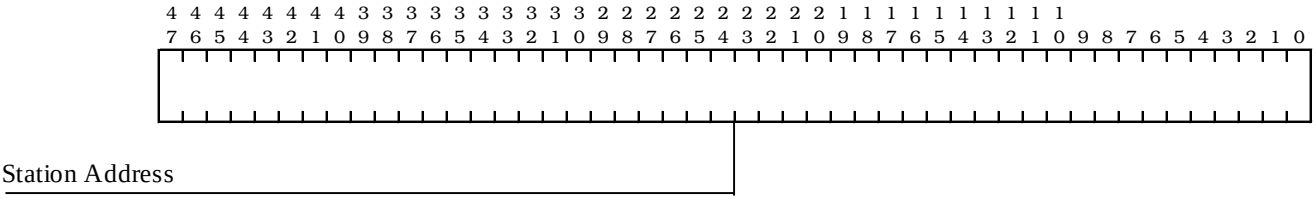



*Default Register Value = 000000000000H* *Register Access = Read/Write*
*Register Address = 100H - 104H*

This register is used in conjunction with the ExactMatch15Address Register. If the CheckAddrPair bit (#16) in the Mode Register is set, then the XMAC II will compare both the DA and SA of incoming packets with the contents of the ExactMatch15Address & SourceCheckAddress Registers. If there is a match, then that packet will be copied into the RxFIFO and the *MACAddrMatch* signal for the corresponding frame will be asserted.

*Note: See general rules for Address Register Access, Reading and Writing Address Registers*

### 6.0.25 StationAddress Register

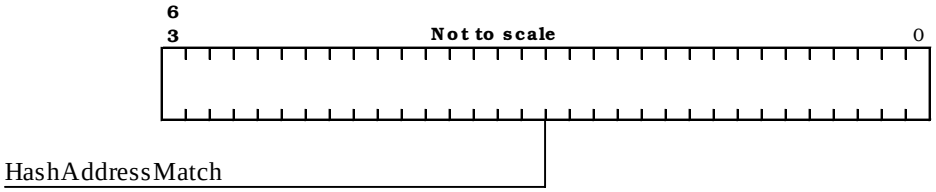



*Default Register Value = 000000000000H* *Register Access = Read/Write*
*Register Address = 108H, 10CH*

These (3) 16-bit registers define the station's unique individual address. If bit 14 of the **Mode register** is set, the destination address of incoming frames are compared with the contents of this register set. If there is a match, then the incoming frame will be received into theReceive FIFO. Typically, this register set is loaded with the IEEE address assigned to the user of this device.

*Note: See general rules for Address Register Access, Reading and Writing Address Registers*

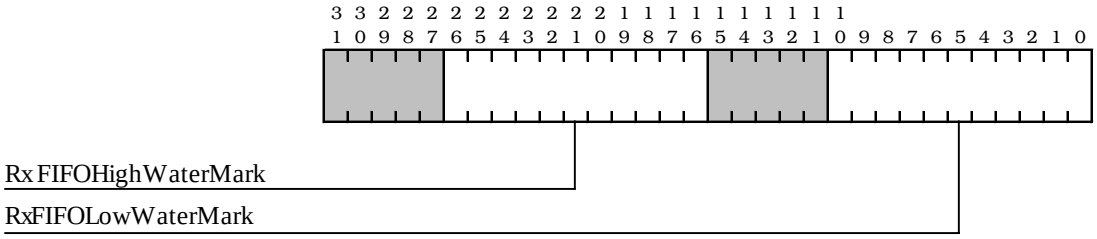
### 6.0.26 HashMatchAddress Register




*Default Register Value = 00000000H* *Register Access = Read/Write*
*Register Address = 110H - 114H*

If user enables the hashing function (by setting bit #15 of Mode Register), then the XMAC II applies a hashing algorithm function (CRC-32) to the Destination Address field of an incoming multicast frame. The least significant 6 bits of the result are used as a bit index into the content of this register set. If the indexed bit is set, then the frame is passed onto the receive FIFO. Otherwise, the frame is rejected. Note that this mode of filtering is imperfect (i.e., hashing two different DA may result in the same index bit). This register set consists of 4 16 bit registers. *For more information, see Section 4.8.1.3 Hashing Function.*

### 6.0.27 Receive Low/HighWaterMark Register

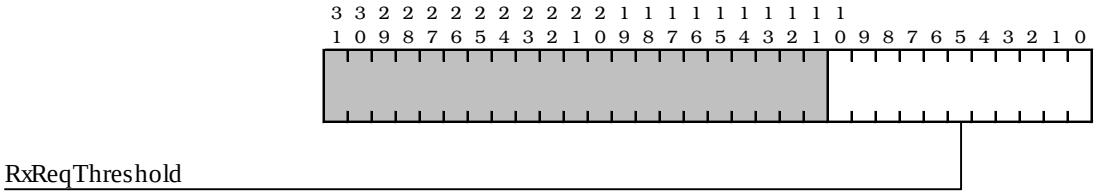



*Default Register Value = 06000200H* *Register Access = Read/Write*
*Register Address = 118H, 11AH*

| **Bit** | **Field** | **Description** |
|---|---|---|
| 31-27 | Reserved | |
| 26-16 | RxFIFOHighWaterMark | These bits define the high water mark value, in 32-bit words. |
| 15-11 | Reserved | |
| 10-0 | RxFIFOLowWaterMark | These bits define the low water mark value, in 32-bit words. |

The XMAC II will assert the *RxFIFOAlmostEmpty* signal when the number of bytes in the receive FIFO is less than or equal to 4 times the *RxFIFOLowWaterMark* value. The *RxFIFOAlmostFull* signal will be asserted when the number of bytes in the receive FIFO is more than 4 times the *RxFIFOHighWaterMark* value. In 16-bit mode, RxFIFOLowWaterMark is accessed at address 118H & *RxFIFOHighWaterMark* is accessed at address 11AH.

### 6.0.28 ReceiveRequestThreshold Register

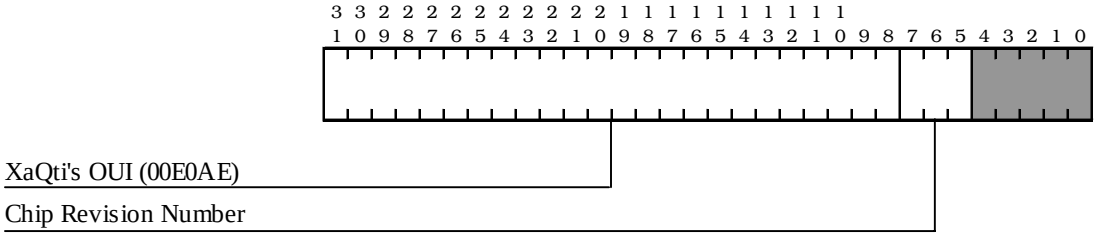



*Default Register Value = 00000004H* *Register Access = Read/Write*
*Register Address* = 011C H

| **Bit** | **Field** | **Description** |
|---|---|---|
| 31-11 | Reserved | |
| 10-0 | RxReqThreshold | The value of this field determines how many bytes (value * 4) should be received into the receive FIFO before the XMAC II asserts the *RxPktValid* signal. If the size of a incoming frame is less than (value *4), then the XMAC II will assert *RxPktValid* after a complete packet is received. Refer to Section X.x, Host (FIFO) Interface. |

Receive Request Threshold Register

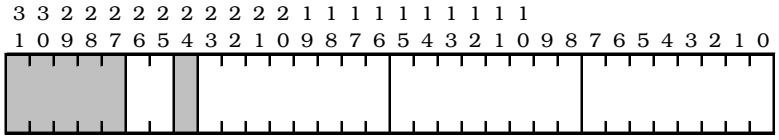
### 6.0.29 Device ID Register




*Default Register Value =0x00E0AE20H* *Register Access = Read Only*
*Register Address = 120H*

| **Bit** | **Field** | **Description** |
|---|---|---|
| 31-8 | XaQti's OUI | Value of 00E0AE00H |
| 7-5 | Chip Revision Number | XMAC Chip Revision Number<br>XMAC II, Rev C = '001' |
| 4-0 | Reserved | |

### 6.0.30 Mode Register




*Default Register Value = 0x000000000000H  Register Access = Read/Write*
*Register Address = 124H, 126H*

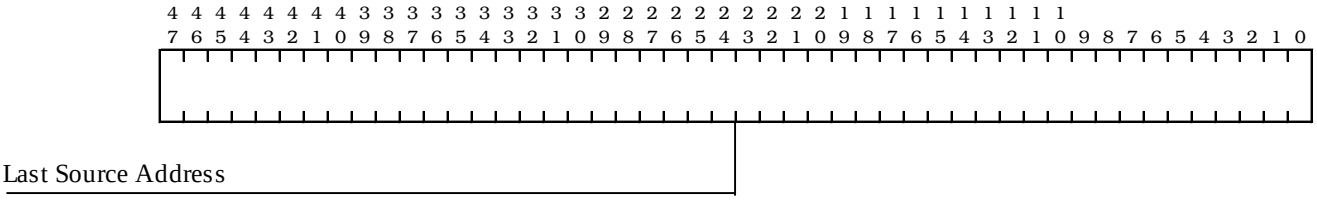
| **Bit** | **Field** | **Description** |
|---|---|---|
| 31-27 | Reserved | |
| 26 | AddrFilter | If this bit is set to '1' (at reset this bit is '0') and EnablePromiscuousMode bit (Bit #3) is also set, then received packets that matches any one of the following checks, will be rejected.<br>a) CheckAddressArray<br>b) CheckAddrPair<br>c) EnableHashing |
| 25 | ExtPauseGen | If this bit is set to '1' (at reset this bit is '0'), a PAUSE frame (with timer field = contents of **Pause Timer** register) is sent when *Xmt_PausePkt* pin is asserted (i.e., during the rising edge) and a PAUSE frame (with timer field = 0) is sent when *Xmt_PausePkt* pin is deasserted (i.e., during the falling edge). |
| 24 | Reserved | |
| 23 | AutoFIFOPause | If this bit is set to '1' (at reset this bit is '0'), then XMAC II will automatically send a PAUSE frame (with timer field = contents of **Pause Timer** register) whenever a packet is missed because of FIFO is full. |
| 22 | LEstatusWord | If this bit is set to '1' (at reset this bit is '0'), then XMAC II will present the Status word (of a received packet) on the HostRxData(31:0) lines in Little Endian format (even if the host interface is programmed to operate in Big Endian format). |
| 21 | Send Continuous | If this bit is set to '1', then XMAC II will transmit a given frame continuously. See Transmit Operations at the end of this document. |
| 20 | SendPauseFrame | If this bit is set to '1', then XMAC II will transmit a pause frame. This bit is self clearing. |
| 19 | AppendTimeStamp | If this bit is set to '1', then XMAC II will append an extra word (32 bits) of status. This extra word contains the time stamp value. The resolution of the time stamp is determined by the TS_CLK. |
| 18 | SendPauseOnLow | If this bit is set, then XMAC II will automatically send a Pause Frame (with PauseTimer field = 0) when an AlmostEmpty condition is reached in the receive FIFO. This bit should not be set in half-duplex mode. |

## Mode Register (cont)

| | | |
|---|---|---|
| 17 | SendPauseOnHigh | If this bit is set, then the XMAC II will automatically send a Pause Frame when an AlmostFull condition is reached in the receive FIFO. The PauseTimer field of the Opcode/PauseTimer register will be used for the pause timer field of the PauseFrame. This bit should not be set in half-duplex mode. |
| 16 | CheckAddrPair | If this bit is set, then the checking of a destination/source address combination is enabled. |
| 15 | EnableHashing | If this bit is set to '1', then XMAC II will apply a hashing algorithm to determine if a given multicast packet is to be received or rejected. |
| 14 | CheckStationAddress | If this bit is set, then the XMAC II will check the **Station Address** register contents with the DA field of the incoming packet. |
| 13 | CheckAddressArray | If this bit is set, then the XMAC II will check the **ExactMatchAddress** register(s) contents with the DA field of the incoming packet. |
| 12 | RcvMACCtrlPkts | If this bit is set, then the XMAC II will receive all MAC control packets into the receive FIFO. |
| 11 | RcvRuntPkts | If this bit and the RcvError bit are set, then the XMAC II will receive all runt packets into the receive FIFO. |
| 10 | RcvInRangeLenErrPkts | If this bit and the *RcvError* bit are set, then the XMAC II will receive all *InRangeLength* error packets into the receive FIFO. |
| 9 | RcvLongPkts | If this bit and the *RcvError* bit are set, then the XMAC II will receive all *Long packets* into the receive FIFO. |
| 8 | RcvCRCErrorPkts | If this bit and the *RcvError* bit are set, then the XMAC II will receive all *CRC error* packets into the receive FIFO. |
| 7 | RcvError | If this bit is set, the XMAC II will receive error frames. The error frames include: *Runt Frames*, *Too Long* Frames, *Check Sequence Error* frames, or *In-range* length error frames. To receive only (1) type of these (4) error frames, bit 7 should be set and one of the bits from the error types (8-11) should also be set. |
| 6 | DisableRcvUnicast | If set, XMAC II will not receive any unicast frames. This will override other address matching logic selections. |
| 5 | DisableRcvMulticast | If this bit is set, the XMAC II will not receive any multicast frames. This bit will override other address matching selections. |
| 4 | DisableRcvBrdcast | If this bit is set, the XMAC II will not receive any broadcast frames. This bit will override other address matching selections. |
| 3 | EnablePromiscuous | When this bit is set to '1', the XMAC II will receive all packets regardless of any address matching criteria, except for bits 6, 5 and 4. |
| 2 | Enable Big Endian | If this bit is set to '1', then Big Endian mode is selected. Otherwise, Little Endian is selected. This feature is applicable only to the host interface. *NOTE: the status (and timestamp, if enabled) is provided in the endian mode selected. The Node Processor interface supports Little Endian only.* |
| 1 | FlushTransmitFIFO | If this bit is set to '1', then the entire transmit FIFO will be flushed. This bit is a self clearing bit. |
| 0 | Flush Receive FIFO | If this bit is set to '1', then the entire receive FIFO will be flushed. This bit is a self clearing bit. |

*Note: For details on address filtering, See Section 4.8.1.3 Address Match Filtering Options.*

### 6.0.31 LastSrcAddress Register

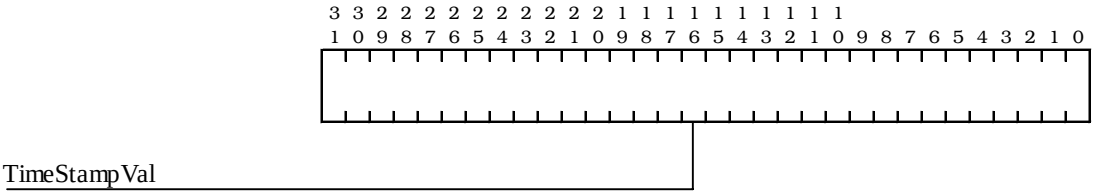



*Default Register Value = 000000000000H* *Register Access = Read Only*
*Register Address = 128H - 12CH*

These (3) 16-bit registers define the Source Address (SA) of the last 'good' packet that was received by the XMAC II.

*Note: See general rules for Address Register Access, Reading and Writing Address Registers*

### 6.0.32 TimeStamp Read Register

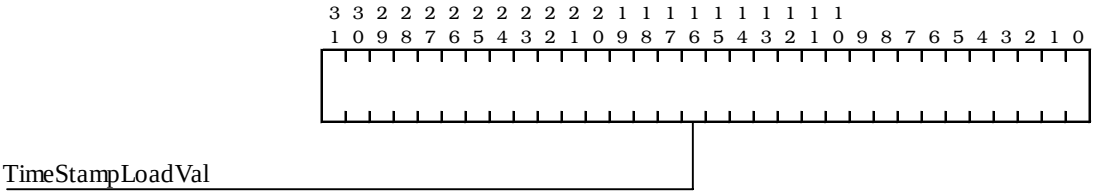



*Register Access = Read*

*Register Address = 130H*

| Bit | Field | Description |
|---|---|---|
| 15-0 | TimeStampVal | When the user reads this register, the current contents of the Time Stamp timer is placed on the NPDATA bus. This is a 32 bit register. If the NP Interface is 16 bits, then address for the lower 16 bits is 0x130 and the address for the higher 16 bits is 0x132. |

### 6.0.33 Time Stamp Load Register

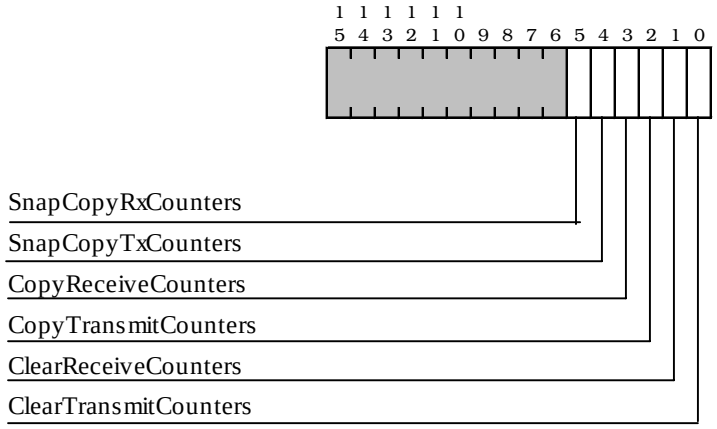



*Register Access = Read/Write*

*Register Address = 134H*

| **Bit** | **Field** | **Description** |
|---|---|---|
| 15-0 | TimeStampLoadVal | This is a read/write register. User can use this register to load the starting value for the TimeStamp timer. The contents of this register will be loaded to the TimeStamp timer when<br>a) /TS_RESET signal is activated (or)<br>b) When TimeStamp timer overflows.<br>The value is loaded synchronously to TS_CLK. This is a 32 bit register. If the NP Interface is 16 bits, then address for the lower 16 bits is 0x134 and the address for the higher 16 bits is 0x136. Using this register one can implement programmable timer interrupt. |

### 6.0.34 StatisticsCommand Register



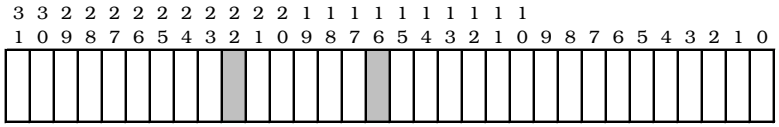

*Default Register Value = 0x000000000000H Register Access = Read/Write*
*Register Address = 200H*

| Bit | Field | Description |
|---|---|---|
| 15-6 | Reserved | |
| 5 | SnapRxCounters | If this bit is set, the **Receive** counters are copied to the shadow counters (self clearing). |
| 4 | SnapTxCounters | If this bit is set, the **Transmit** counters are copied to the shadow counters (self clearing). |
| 3 | CopyReceiveCounters | If this bit is set, the **Receive** counters are continuously copied to the shadow counters.(sticky bit) |
| 2 | CopyTransmitCounters | If this bit is set, the **Transmit** counters are continuously copied to the shadow counters.(sticky bit) |
| 1 | ClearReceiveCounters | If this bit is set, the **Receive** counters are cleared (self clearing). |
| 0 | ClearTransmitCounters | If this bit is set, the **Transmit** counters are cleared (self clearing). |

*Note 1: If Copy and Clear bits (or Snap and Clear) are set at the same time, then the counters will be copied before getting cleared.*
*Note 2: To change mode from Copy to Snap, the Copy bit must be cleared and the Snap bit set.*
*Note 3: It is possible to clear and capture the counters at the same time. This could be useful in cases where software keeps on adding the values.*

To clear the counters:
Set bits #1 & #0 (These bits are self clearing)
To get the TrueSTATS:
a) First set bits #5 & #4 (This will allow capture of all counters at that instance. These bits are self clearing);
b) Next perform the read operation of required counter(s).

### 6.0.35 ReceiveCounterEvent Register

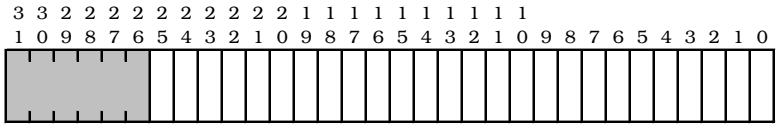



*Default Register Value = 00000000H* *Register Access = Read Only (clear on read)*
*Register Address = 204H, 206H*

| Bit | Field | Description |
|---|---|---|
| 31 | 1024-MaxSizeRxByteOverflow | When this bit is set, the 1024-MaxSizeRxByte counter has overflowed |
| 30 | 512-1023RxByteOvrflw | When this bit is set, the 512-1023RxByte counter has overflowed |
| 29 | 256-511 RxByteOvrflw | When this bit is set, the 256-511RxByte counter has overflowed |
| 28 | 128-255 RxByteOvrflw | When this bit is set, the 128-255RxByte counter has overflowed |
| 27 | 65-127 RxByteOvrflw | When this bit is set, the 65-127RxByte counter has overflowed |
| 26 | 64 RxByteOvrflw | When this bit is set, the 64RxByte counter has overflowed |
| 25 | RxOverflowed | When this bit is set, the Rx Utilization counter has overflowed |
| 24 | RxUnderflowed | When this bit is set, the Rx Utilization counter has underflowed |
| 23 | CextErrorCounter Overflow | When this bit is set, the CextError counter has overflowed |
| 22 | Reserved | Reserved |
| 21 | FrameCheckSeqErrors Ovr | When this bit is set, the FrameCheckSequenceErrors counter has overflowed |
| 20 | FrameTooLongErrors Ovr | When this bit is set, the FrameTooLongErrors counter has overflowed |
| 19 | Runt Overflow | When this bit is set, the Runt counter has overflowed |
| 18 | ShortEvent Overflow | When this bit is set, the ShortEvent counter has overflowed |
| 17 | SymbolErrorCounter Ovrflw | When this bit is set, the SymbolError counter has overflowed |
| 16 | Reserved | |
| 15 | CarrierEvent Overflow | When this bit is set, the CarrierEventErrors counter has overflowed |
| 14 | Jabber Pkt Overflow | When this bit is set, the JabberPkt counter has overflowed |
| 13 | RxFIFOOverflow | When this bit is set, the RxFIFOOverflow counter has overflowed |
| 12 | FramingErrors Overflow | When this bit is set, the FramingErrors counter has overflowed |
| 11 | Missed Frames Overflow | When this bit is set, the MissingFrames counter has overflowed |
| 10 | ReceiveBurst Overflow | When this bit is set, the ReceiveBurst counter has overflowed |
| 9 | MACCtrlFramesOpcode Ovr | When this bit is set, the MACCtrlFramesWithUnsupportedOpcode counter has overflowed |
| 8 | InvalidPAUSEFramesRcv Ovr | When this bit is set, the InvalidPAUSEFramesReceived counter has overflowed |
| 7 | MACCtrlFramesRcv Overflow | When this bit is set, the MACCtrlFramesReceived counter has overflowed |
| 6 | PAUSEMACtrlFramesRcvOvr | When this bit is set, the PAUSEMACCtrlFramesreceived counter has overflowed |
| 5 | UnicastFramesRcvdOKOvr | When this bit is set, the UnicastFramesReceivedOK counter has overflowed |
| 4 | MulticastFramesRcvdOKOvr | When this bit is set, the MulticastFramesReceivedOK counter has overflowed |
| 3 | BroadcastFramesRcvdOK Ovr | When this bit is set, the BroadcastFramesReceivedOK counter has overflowed |
| 2 | OctetsRcvdOKLow Ovrflw | When this bit is set, the OctetsReceivedOKLow counter has overflowed |
| 1 | OctetsRcvdOKHigh Ovrflw | When this bit is set, the OctetsReceivedOKHigh counter has overflowed |
| 0 | FramesReceivedOK Overflow | When this bit is set, the FramesReceivedOK counter has overflowed |

Receive Counter Event Register

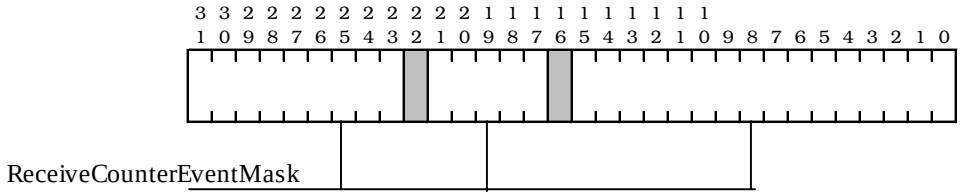
### 6.0.36 TransmitCounterEvent Register




*Default Register Value = 00000000H* *Register Access = Read Only (clear on read)*
*Register Address = 208H, 20AH*

| Bit | Field | Description |
|---|---|---|
| 31-26 | Reserved | |
| 25 | 1024-MaxSizeTxByteOvrflw | When this bit is set, the 1024-Max Size Byte counter has overflowed |
| 24 | 512-1023TxByteOvrflw | When this bit is set, the 512-1023 Byte counter has overflowed |
| 23 | 256-511TxByteOvrflw | When this bit is set, the 256-511 Byte counter has overflowed |
| 22 | 128-255TxByteOvrflw | When this bit is set, the 128-255 Byte counter has overflowed |
| 21 | 65-127TxByteOvrflw | When this bit is set, the 65-127 Byte counter has overflowed |
| 20 | 64TxByteOvrflw | When this bit is set, the 64 Byte counter has overflowed |
| 19 | TxOverflow | When this bit is set, the Tx Utilization counter has overflowed |
| 18 | RxUnerflow | When this bit is set, the Tx Utilization counter has underflow |
| 17 | CarrierSenseError | When this bit is set, the carrier sense error counter has overflowed |
| 16 | TxFIFOUnderrunOvrflw | When this bit is set, the transmit FIFO underrun counter has overflowed |
| 15 | FramesW/ExcessDeferralOvr | When this bit is set, the FramesWithExcessiveDeferral counter has overflowed |
| 14 | FramesW/DeferredXmtOvr | When this bit is set, the FramesWithDeferredXmissions counter has overflowed |
| 13 | LateCollisionsOverflow | When this bit is set, the LateCollisions counter has overflowed |
| 12 | FramesAbortedXSCollsOvr | When this bit is set, the FramesAbortedDueToXSColls counter has overflowed |
| 11 | MultipleCollisionFramesOvr | When this bit is set, the MultipleCollisionFrames counter has overflowed |
| 10 | SingleCollisionFrames Ovrflw | When this bit is set, the SingleCollisionFrames counter has overflowed |
| 9 | MACCtrlFramesTx Ovrflw | When this bit is set, the MACCtrlFramesTransmitted counter has overflowed |
| 8 | PAUSEMACtrlFramesTx Ovr | When this bit is set, the PAUSEMACCtrlFramesTransmitted counter has overflowed |
| 7 | TransmitBurst Overflow | When this bit is set, the TransmitBurst Overflow counter has overflowed |
| 6 | TransmitLongFramesOvrflw | When this bit is set, the TransmitLongFrames counter has overflowed |
| 5 | UnicastFramesXmtOK Ovrflw | When this bit is set, the UnicastFramesXmittedOK counter has overflowed |
| 4 | MulticastFramesXmtOK Ovr | When this bit is set, the MulticastFramesXmittedOK counter has overflowed |
| 3 | BroadCastFramesXmtOK Ovr | When this bit is set, the BroadCastFramesXmittedOK counter has overflowed |
| 2 | OctetsTxOKLow Overflow | When this bit is set, the OctetsTransmittedOKLow counter has overflowed |
| 1 | OctetsTxOKHigh Overflow | When this bit is set, the OctetsTransmittedOKHigh counter has overflowed |
| 0 | FramesTransmittedOK Ovrflw | When this bit is set, the FramesTransmittedOK counter has overflowed |

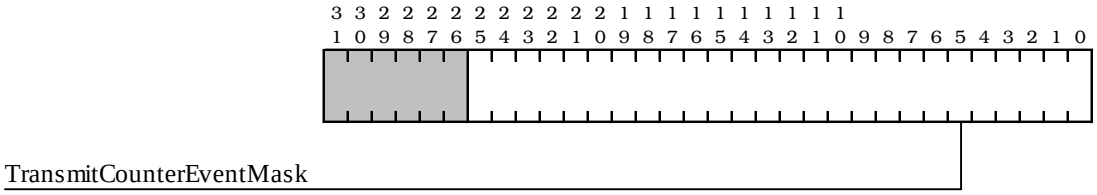
### 6.0.37 ReceiveCounterEventMask Register




*Default Register Value = FFFFFFFFH* *Register Access = Read/Write*
*Register Address = 20CH & 20EH*

This register can be used to enable/disable the interrupts due to the assertion of the corresponding bit in the Receive Counters Event register. A zero indicates that the interrupt is enabled for the corresponding event.

### 6.0.38 TransmitCounterEventMask Register

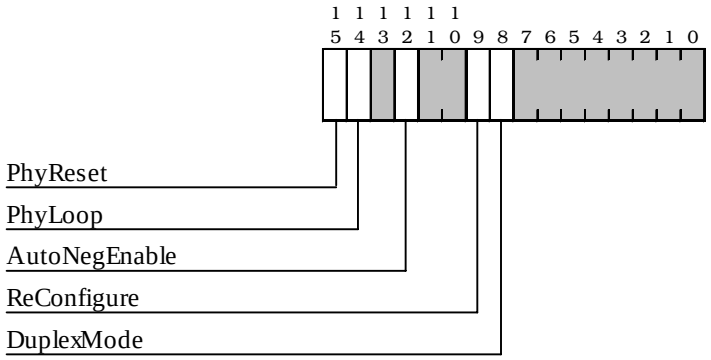



*Default Register Value = 00000000H* *Register Access = Read/Write*
*Register Address = 210H & 212H*

This register can be used to mask out one (or more) transmit counter event(s) from generating an interrupt.

Transmit Count Event Mask Register

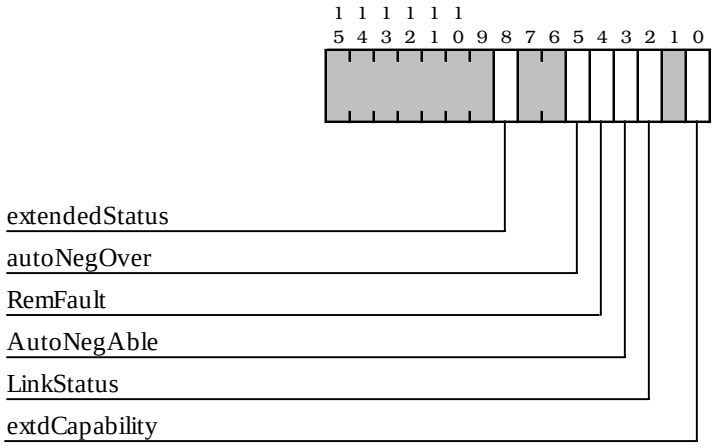
### 6.0.39 PHY Control Register




*Default Register Value = 0x00001140H* *Register Access = Read/Write*
*Register Address = Indirect (00H)*

| Bit | Field | Description |
|---|---|---|
| 15 | PhyReset | If set, clears all PHY related registers. This bit is self-clearing. |
| 14 | PhyLoop | If set, *PHY_LP_EN* signal will be asserted. |
| 13 | Reserved | |
| 12 | AutoNegEnable | If set, autonegotiation is enabled. |
| 11-10 | Reserved | |
| 9 | ReConfigure | If set, autonegotiation is restarted. This bit is self-clearing. |
| 8 | DuplexMode | If autonegotiation is disabled (AutoNegEnable bit is cleared), then this bit can be used to manually configure the XMAC II for half or full duplex operation. If this bit is set, then full duplex operation is selected. If this bit is cleared, then half duplex mode is enabled. |
| 7-0 | Reserved | |

### 6.0.40 PHY Status Register

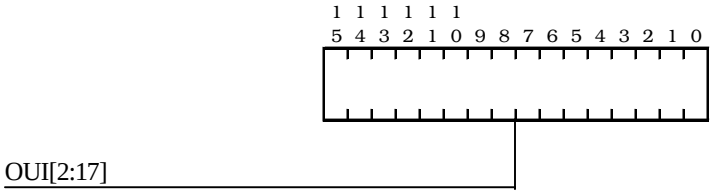



*Default Register Value = 0x0109H* *Register Access = Read Only*
*Register Address = Indirect (01H)*

| **Bit** | **Field** | **Description** |
|---|---|---|
| 15-9 | Reserved | |
| 8 | extendStatus | This bit is set when the Extended Status register is present |
| 7-6 | Reserved | |
| 5 | AutoNegOver | This bit is set when the autonegotiation process is complete |
| 4 | RemFault | This bit is set when a Remote fault condition has occurred |
| 3 | AutoNegAble | This bit is set to indicate that the device is capable of autonegotiation |
| 2 | LinkStatus | This bit is set to indicate that the link is synchronized<br>If this bit is reset, the link is not synchronized |
| 1 | Reserved | |
| 0 | extdCapability | This bit is set when the Extended Register set is available |

. . . .

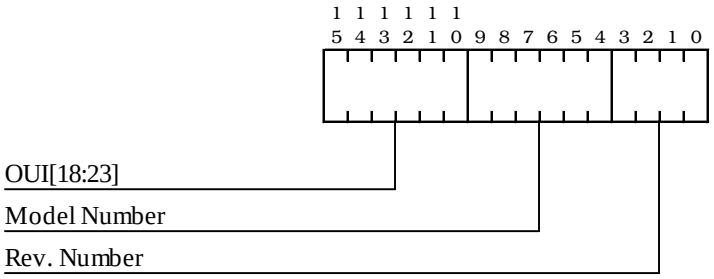
### 6.0.41 PHY ID0 Register




*Default Register Value = 0xD41C H* *Register Access = Read Only*
*Register Address = Indirect (02H)*

| Bit | Field | Description |
|---|---|---|
| 15-0 | OUI[2:17] | These bits map to 16-bits of the OUI. Bit 15 of this register maps to bit 2 of OUI. |

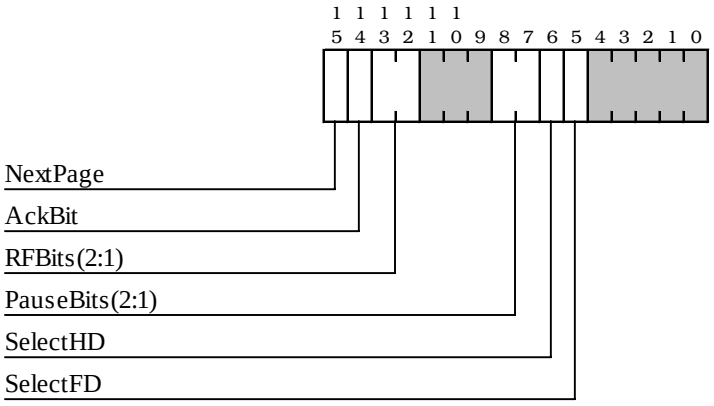
### 6.0.42 PHY ID1 Register




*Default Register Value = 0x0002H* *Register Access = Read Only*
*Register Address = Indirect (03H)*

| Bit | Field | Description |
|---|---|---|
| 15-10 | OUI[18:23] | These bits map to the six (6) most significant bits of the OUI. Bit 15 of this register maps to bit 18 of OUI. |
| 9-4 | Model Number | These bits represent the XMACII Model number.<br>XMAC II, Rev. C1 = 0 |
| 3-0 | RevisionNumber | These bits represent the XaQti Revision number.<br>XMAC II, Rev. C1 = '0010' |

### 6.0.43 Autonegotiation Advertisement Register

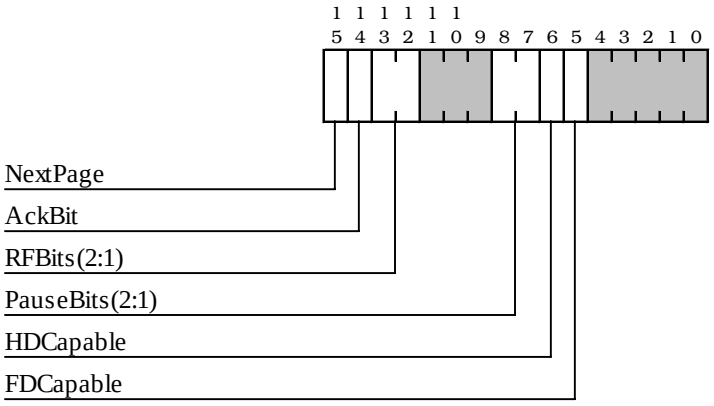



*Default Register Value = 0x01E0H* *Register Access = Read/Write*
*Register Address = Indirect (04H)*

| Bit | Field | Description |
|---|---|---|
| 15 | NextPage | When this bit is set to '1', the next page transmission is requested. |
| 14 | AckBit | This bit is set to indicate the succesful reception of link partner's base or next page. This is a read only bit. |
| 13-12 | RFBits(2:1) | These bits denote the remote fault bits. The encoding is:<br>00 = No error, link OK 10 = Offline<br>01 = Link Failure 11 = Autonegotiation Error |
| 11-9 | Reserved | |
| 8-7 | PauseBits(2:1) | These bits indicate the Pause request. The encoding is:<br>00 = No Pause possible 01 = Asymetric Pause toward link partner<br>10 = Symetric Pause 11 = Both symmetric Pause and assymetric pause towards the local device. |
| 6 | SelectHD | If this bit is set, then the half duplex mode is requested. |
| 5 | SelectFD | If this bit is set, then the full duplex mode is requested. |
| 4-0 | Reserved | |

*Note: For additional information, see: Section 4.23 Configuration Error During Autonegotiation*

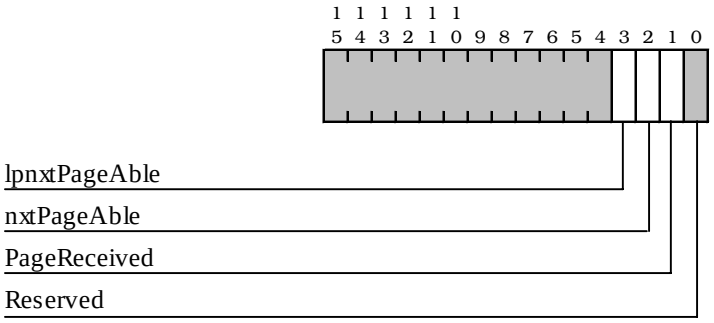
### 6.0.44 Link Partner Ability Register




*Default Register Value = 0x0000H* *Register Access = Read Only*
*Register Address = Indirect (05H)*

| Bit | Field | Description |
|---|---|---|
| 15 | NextPage | When this bit is set to '1', the next page transmission is requested. |
| 14 | AckBit | This bit indicates the succesful reception of link partner's base or next page. |
| 13-12 | RFBits(2:1) | These bits denote the remote fault bits. The encoding is:<br>00 = No error, link OK 10 = Offline<br>01 = Link Failure 11 = Autonegotiation Error |
| 11-9 | Reserved | |
| 8-7 | PauseBits(2:1) | These bits indicate the Pause capability. The encoding is:<br>00 = No Pause possible 01 = Asymetric Pause toward link partner<br>10 = Symetric Pause 11 = Both symmetric Pause and assymetric pause towards the local device. |
| 6 | HDCapable | When this bit is set, the device is half duplex capable |
| 5 | FDCapable | When this bit is set, the device is full duplex capable |
| 4-0 | Reserved | |

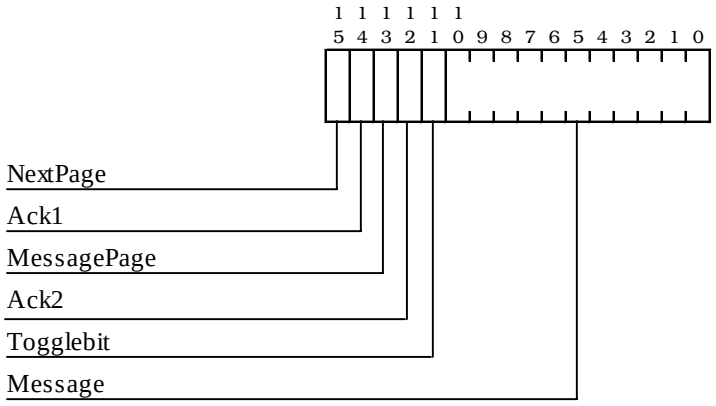
### 6.0.45 Autonegotiation Expansion Register




*Default Register Value = 0x0000H* *Register Access = Read Only (clear on read)*
*Register Address = Indirect (06H)*

| **Bit** | **Field** | **Description** |
|---|---|---|
| 15-4 | Reserved | |
| 3 | lpnxtPageAble | When this bit is set, it indicates that link partner is capable of Next Page transaction. |
| 2 | nxtPageAble | When this bit is set, it indicates that the XMAC II is capable of Next Page transaction. |
| 1 | PageReceived | When this bit is set, it indicates that a page (base or next) has been received during Autonegotiation. When read, this bit is cleared automatically. When set, this bit will generate a hardware interrupt, if not masked in the **Interrupt Mask** register. |
| 0 | Reserved | |

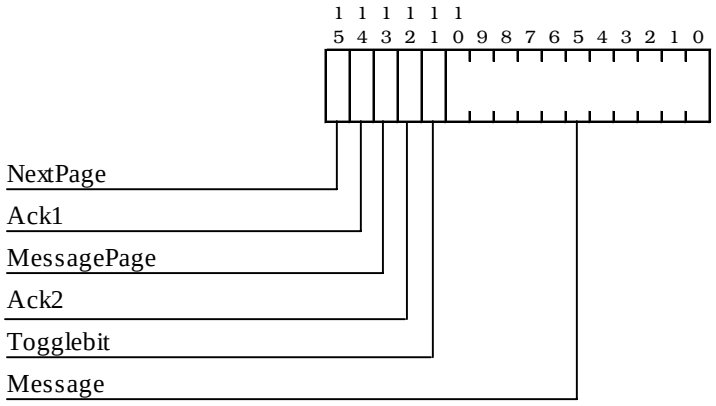
### 6.0.46 Next Page Register




*Default Register Value = 0x0000H* *Register Access = Read Only*
*Register Address = Indirect (07H)*

| Bit | Field | Description |
|---|---|---|
| 15 | NextPage | More Next Pages to follow |
| 14 | Ack1 | Acknowledgement bit received OK |
| 13 | MessagePage | When this bit is set, it indicates that it is a formatted Message Page |
| 12 | Ack2 | This bit is set by XMAC II to indicate its ability to comply with the message. Read-only |
| 11 | Togglebit | Used by the Arbitration function to synchronize with the Link Partner. Read-onl Read-only |
| 10-0 | Message | Message to the Link Partner |

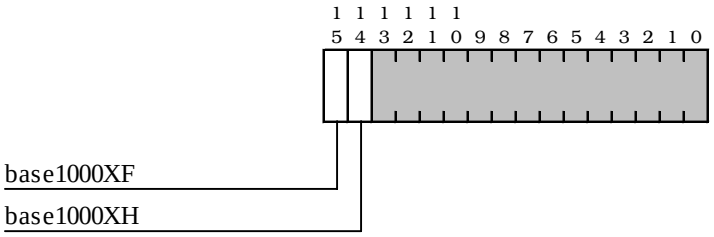
### 6.0.47 Next Page Link Partner Register




*Default Register Value = 0x0000H* *Register Access = Read Only*
*Register Address = Indirect (08H)*

| **Bit** | **Field** | **Description** |
|---|---|---|
| 15 | NextPage | More Next Pages to follow |
| 14 | Ack1 | This is an acknowledgement bit. |
| 13 | MessagePage | When this bit is set, it indicates that it is a formatted message page |
| 12 | Ack2 | This bit is set by the Link Partner to indicate its ability to comply with the message from XMAC II. |
| 11 | Togglebit | Used by the Arbitration function to synchronize with the XMAC II. |
| 10-0 | Message | Message from the Link Partner |

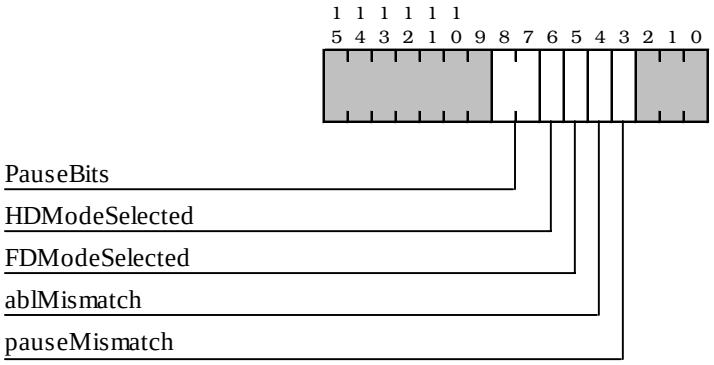
### 6.0.48 PHY Extended Status Register




*Default Register Value = 0xC000 H* *Register Access = Read Only*
*Register Address = Indirect (0F H)*

| Bit | Field | Description |
|---|---|---|
| 15 | base1000XF | This device supports Full Duplex Mode |
| 14 | base1000XH | This device supports Half Duplex Mode |
| 13-0 | Reserved | . . . |

### 6.0.49 PHY Resolved Ability Register




*Default Register Value = 41A0H* *Register Access = Read Only*
*Register Address = Indirect (10H)*

| **Bit** | **Field** | **Description** |
|---|---|---|
| 15-9 | Reserved | |
| 8-7 | PauseBits | These bits reflect the corresponding bits in the link partner ability register |
| 6 | HDModeSelected | If this bit is set, then the local device will operate in half duplex mode |
| 5 | FDModeSelected | If this bit is set, then the local device will operate in full duplex mode |
| 4 | ablMismatch | If this bit is set, it indicates that either duplex abilities do not match or pause capabilities are incompatible |
| 3 | pauseMismatch | If this bit is set, it indicates that the pause abilities are not identical. The devices may still work if ablMismatch is false |
| 2-0 | Reserved | |

*Note1: If bit 6 and 5 are both cleared, then the system should not enable XMAC II's transmitter or receiver.*
*Note2: Likewise, Pause bits should be checked against 'own' Pause bit to see if there is a possible mismatch. For more information see: Section 4.23, Configuration Error During Autonegotiation.*

## 6.1 XMAC II TRANSMIT STATISTICSCOUNTER LISTING

| Name | Bits | Address 32bit Mode | Address 16bit Mode | O.I.D. |
|---|---|---|---|---|
| FramesTransmittedOK Counter | 32 | 280 | 280,282 | |
| OctetsTransmittedOKHigh Counter | 32 | 284 | 284,286 | |
| OctetsTransmittedOKLow Counter | 32 | 288 | 288, 28A | |
| BroadCastFramesXmittedOK Counter | 32 | 28C | 28C, 28E | |
| MulticastFramesXmittedOK Counter | 32 | 290 | 290, 292 | |
| UnicastFramesXmittedOK Counter | 32 | 294 | 294, 296 | |
| TransmitLongFrames Counter | 32 | 298 | 298, 29A | |
| TransmitBurst Counter | 32 | 29C | 29C, 29E | |
| PAUSEMACCtrlFramesTransmitted Counter | 32 | 2A0 | 2A0, 2A2 | |
| MACCtrlFramesTransmitted Counter | 32 | 2A4 | 2A4, 2A6 | |
| SingleCollisionFrames Counter | 32 | 2A8 | 2A8, 2AA | |
| MultipleCollisionFrames Counter | 32 | 2AC | 2AC, 2AE | |
| FramesAbortedDueToXSColls Counter | 32 | 2B0 | 2B0, 2B2 | |
| LateCollisions Counter | 32 | 2B4 | 2B4, 2B6 | |
| FramesWithDeferredXmissions Counter | 32 | 2B8 | 2B8, 2BA | |
| FramesWithExcessiveDeferral Counter | 32 | 2BC | 2BC, 2BE | |
| TransmitFIFOUnderrun Counter | 32 | 2C0 | 2C0, 2C2 | |
| CarrierSenseError Counter | 32 | 2C4 | 2C4, 2C6 | |
| TransmitUtilization Counter | 32 | 2C8 | 2C8, 2CA | |
| 64 TxByteCounter | 32 | 2D0 | 2D0, 2D2 | |
| 65-127 TxByteCounter | 32 | 2D4 | 2D4, 2D6 | |
| 128-255 TxByteCounter | 32 | 2D8 | 2D8, 2DA | |
| 256-511 TxByteCounter | 32 | 2DC | 2DC, 2DE | |
| 512-1023 TxByteCounter | 32 | 2E0 | 2E0, 2E2 | |
| 1024-MaxSize TxByteCounter | 32 | 2E4 | 2E4, 2E6 | |

## 6.2 XMAC II RECEIVE STATISTICSCOUNTER LISTING

| Name | Bits | Address 32bit Mode | Address 16bit Mode | O.I.D. |
|---|---|---|---|---|
| FramesReceivedOK Counter | 32 | 300 | 300, 302 | |
| OctetsReceivedOKHigh Counter | 32 | 304 | 304, 306 | |
| OctetsReceivedOKLow Counter | 32 | 308 | 308, 30A | |
| BroadcastFramesReceivedOK Counter | 32 | 30C | 30C, 30E | |
| MulticastFramesReceivedOK Counter | 32 | 310 | 310, 312 | |
| UnicastFramesReceivedOK Counter | 32 | 314 | 314, 316 | |
| PAUSEMACCtrlFramesReceived Counter | 32 | 318 | 318, 31A | |
| MACCtrlFramesReceived Counter | 32 | 31C | 31C, 31E | |
| InvalidPAUSEFramesReceived Counter | 32 | 320 | 320, 322 | |
| MACCtrlFramesWithUnsupportedOpcode Counter | 32 | 324 | 324, 326 | |
| ReceiveBurst Counter | 32 | 328 | 328, 32A | |
| Missed Frames Counter | 32 | 32C | 32C, 32E | |
| FramingErrors Counter | 32 | 330 | 330, 332 | |
| RxFIFOOverflow Counter | 32 | 334 | 334, 336 | |
| Jabber Pkt Counter | 32 | 338 | 338, 33A | |
| CarrierEventErrors Counter | 32 | 33C | 33C, 33E | |
| InRangeLengthErrors Counter | 32 | 340 | 340, 342 | |
| SymbolError Counter | 32 | 344 | 344, 346 | |
| ShortEvent Counter | 32 | 348 | 348, 34A | |
| Runt Counter | 32 | 34C | 34C, 34E | |
| FrameTooLongErrors Counter | 32 | 350 | 350, 352 | |
| FrameCheckSequenceErrors Counter | 32 | 354 | 354, 356 | |
| CextError Counter | 32 | 35C | 35C, 35E | |
| Receive Utilization Counter | 32 | 360 | 360, 362 | |
| 64 RxByteCounter | 32 | 368 | 368, 36A | |
| 65-127 RxByteCounter | 32 | 36C | 36C, 36E | |
| 128-255 RxByteCounter | 32 | 370 | 370, 372 | |
| 256-511 RxByteCounter | 32 | 374 | 374, 376 | |
| 512-1023 RxByteCounter | 32 | 378 | 378, 37A | |
| 1024-MaxSize RxByteCounter | 32 | 37C | 37C, 37E | |

## 6.3 DESCRIPTION OF XMACII MANAGEMENT COUNTERS

### 6.3.1 Transmit Counter Functions

#### 6.3.1.1 Frames Transmitted OK

A count of frames (including short frames, broadcast frames, and multicast frames) that are successfully transmitted. *Conforms to IEEE 802.3z D4 Layer Management (Chapter 30) 30.3.1.1.2.*

#### 6.3.1.2 Octets Transmitted OK

A count of data and padding octets of frames that are successfully transmitted. *Conforms to IEEE 802.3z D4 Layer Management (Chapter 30) 30.3.1.1.8.*

#### 6.3.1.3 Broadcast Frames Transmitted OK

A count of the frames that were successfully transmitted to the broadcast address. Frames transmitted to multicast addresses are not broadcast frames and are excluded. *Conforms to IEEE 802.3z D4 Layer Management (Chapter 30) 30.3.1.1.19.*

#### 6.3.1.4 Multicast Frames Transmitted OK

A count of frames that are successfully transmitted to a group destination address other than broadcast. *Conforms to IEEE 802.3z D4 Layer Management (Chapter 30) 30.3.1.1.18.*

#### 6.3.1.5 Unicast Frames Transmitted OK

The total number of frames transmitted that were directed to an unicast address. This number does not include frames directed to broadcast or multicast addresses.

#### 6.3.1.6 Transmit Long Frames

The total number of frames transmitted that are more than the IEEE 802.3 Maximum Frame Size. Typically, 1518 octets if neither the *OneLevelVLANTag* nor *TwoLevelVLANTag* register bits are set. If the *OneLevelVLANTag* register bit is set, then Long Frames are defined as more than 1522 octets (excluding framing bits but including FCS octets). If the *TwoLevelVLANTag* register bit is set, then Long Frames are defined as more than 1538 octets (excluding framing bits but including FCS octets).

#### 6.3.1.7 Transmit Burst Counter

The total number of Transmit burst events. A 'burst event' is a single transmit event in which more than one frame was transmitted.

#### 6.3.1.8 PAUSE MAC-Control Frames Transmitted

The total number of Pause MAC-Control frames transmitted, of Opcode = 0001. *Conforms to IEEE 802.3z D4 Layer Management (Chapter 30) 30.3.4.2.*

#### 6.3.1.9 MAC-Control Frames Transmitted

A count of MAC Control frames transmitted. *Conforms to IEEE 802.3z D4 Layer Management (Chapter 30) 30.3.3.3.*

#### 6.3.1.10 Single Collision Frames

A count of successfully transmitted frames on a particular interface for which transmission is inhibited by exactly one collision. *Conforms to IEEE 802.3z D4 Layer Management (Chapter 30) 30.3.1.1.3.*

#### 6.3.1.11 Multiple Collision Frames

A count of successfully transmitted frames on a particular interface for which transmission is inhibited by more than one collision. *Conforms to IEEE 802.3z D4 Layer Management (Chapter 30) 30.3.1.1.4.*

#### 6.3.1.12 Frames Aborted Due to Excessive Collisions

A count of frames that due to excessive collisions are not transmitted successfully. *Conforms to IEEE 802.3z D4 Layer Management (Chapter 30) 30.3.1.1.11.*

#### 6.3.1.13 Late Collisions

A count of the times that a collision has been detected later than one slotTime, from the start of the packet transmission. A late collision is counted twice, i.e., both as a collision and as a lateCollision. *Conforms to IEEE 802.3z D4 Layer Management (Chapter 30) 30.3.1.1.10.*

#### 6.3.1.14 Frames with Deferred Transmissions

A count of frames whose transmission was delayed on its first attempt because the medium was busy. *Conforms to IEEE 802.3z D4 Layer Management (Chapter 30) 30.3.1.1.9.*

#### 6.3.1.15 Frames with Excessive Deferral

A count of frames that deferred for an excessive period of time. This counter may only be incremented once per LLC transmission. *Conforms to IEEE 802.3z D4 Layer Management (Chapter 30) 30.3.1.1.20.*

#### 6.3.1.16 Carrier Sense Error Counter

A count of times that the carrier sense condition was not asserted or was deasserted during the transmission of a frame without collision. Conforms to IEEE 802.3z D4 Layer Management (Chapter 30) 30.3.1.1.13.

#### 6.3.1.17 Transmission Utilization Counter

The utilization of the transmit path, calculated as a percentage, using the following formula:

percentage utilization = (.5- Delta/MaxRange)*100

Where:

*MaxRange* = UtilWindow/256 ns

*Delta* = StartVal - CurUtilizationVal (if it is negative, do not discard, use as negative number).

*StartVal*: is the utilization value at the beginning of utilization window.

*UtilWindow*: is the time window over which, user wants the utilization to be computed.

*CurUtilizationVal*: is the current value of transmit utilization register.

**Note1:** that after giving *sampleLine* command, *UtilizationVal* becomes 80000000H. Note 2: the utilization computer can sustain about 16 minutes of worst case (worst case is defined as continuous idle or continuous traffic); thus it is necessary to either give sampleLine command at the end of 16 minutes or adjust a value 80000000H for each less than 0 or more than FFFFFFFFH values.

#### 6.3.1.18 64 Tx Byte Frames

The total number of frames transmitted that were 64 octets in length (excluding framing bits but including FCS octets)

#### 6.3.1.19 65–127 Tx Byte Frames

The total number of frames transmitted that were between 65 and 127 octets in length inclusive (excluding framing bits but including FCS octets)

#### 6.3.1.20 128–255 Tx Byte Frames

The total number of frames transmitted that were between 128 and 255 octets in length inclusive (excluding framing bits but including FCS octets)

#### 6.3.1.21 256–511 Tx Byte Frames

The total number of frames transmitted that were between 256 and 511 octets in length inclusive (excluding framing bits but including FCS octets)

#### 6.3.1.22 512–1023 Tx Byte Frames

The total number of frames transmitted that were between 512 and 1023 octets in length inclusive (excluding framing bits but including FCS octets)

#### 6.3.1.23 1024-MaxSize Tx Byte Frames

The total number of frames transmitted that were between 1024 and 1518 octets in length inclusive (excluding framing bits but including FCS octets). If the *OneLevelVLANTag* register bit is set, this statistic counts frames that are between 1024 and 1522 octets (excluding framing bits but including FCS octets). ). If the *TwoLevelVLANTag* register bit is set, this statistic counts frames that are between 1024 and 1538 octets (excluding framing bits but including FCS octets).

### 6.3.2 Receive Counter Functions

#### 6.3.2.1 Frames Received OK

A count of data and padding octets in frames that are successfully received. This does not include octets in frames received with frame-too-long, FCS, length or alignment errors, or frames lost due to internal MAC sublayer error. *Conforms to IEEE 802.3z D4 Layer Management (Chapter 30) 30.3.1.1.5.*

#### 6.3.2.2 Octets Received OK

A count of data and padding octets in frames that are successfully received. This does not include octets in frames received with frame-too-long, FCS, length or alignment errors, or frames lost due to internal MAC sublayer error. *Conforms to IEEE 802.3z D4 Layer Management (Chapter 30) 30.3.1.1.14.*

#### 6.3.2.3 Broadcast Frames Received OK

A count of frames that are successfully received and are directed to the broadcast group address. This does not include frames received with frame-too-long, FCS, length or alignment errors, or frames lost due to internal MAC sublayer error. *Conforms to IEEE 802.3z D4 Layer Management (Chapter 30) 30.3.1.1.22.*

#### 6.3.2.4 Multicast Frames Received OK

A count of frames that are successfully received and are directed to an active non-broadcast group address (as defined by the Counter Exact Address Match Criteria, .for more information, see Section X.X, Address and Counter Match Criteria). This does not include frames received with frame-too-long, FCS, length or alignment errors, or frames lost due to internal MAC sublayer error. *Conforms to IEEE 802.3z D4 Layer Management (Chapter 30) 30.3.1.1.21.*

#### 6.3.2.5 Unicast Frames Received OK

The total number of good frames received that were directed to any unicast address, as defined by the Counter Exact Address Match Criteria. *For more information, see Section X.X, Address and Counter Match Criteria.*

#### 6.3.2.6 PAUSE MAC-Control Frames Received

A count of MAC Control frames received with a lengthOrType field value equal to the reserved Type for 802.3_MAC_Control as specified in 31.4.1.3 [8808], and (2) an Opcode indicating the PAUSE operation. *Conforms to IEEE 802.3z D4 Layer Management (Chapter 30) 30.3.4.3.*

#### 6.3.2.7 MAC-Control Frames Received

A count of MAC Control frames received. MAC Control Frames are defined as valid frames with a lengthOrType field value equal to the reserved Type for 802.3_MAC_Control as specified in 31.4.1.3 [8808]. *Conforms to IEEE 802.3z D4 Layer Management (Chapter 30) 30.3.3.4.*

#### 6.3.2.8 Invalid PAUSE MAC-Control Frames Received

A count of MAC Control frames received with standard PAUSE address with a lengthOrType field value equal to the reserved Type for 802.3_MAC_Control as specified in 31.4.1.3 [8808]; but without a valid PAUSE Opcode [0001].

#### 6.3.2.9 MAC-Control Frames with Unsupported Opcode

A count of MAC Control frames received that contain an Opcode from Table 31A-1 that is not supported by the device. MAC Control Frames are defined as valid frames with a lengthOrType field value equal to the reserved Type for 802.3_MAC_Control as specified in 31.4.1.3 [8808] and with an Opcode for a function that is not supported by the device]. *Conforms to IEEE 802.3z D4 Layer Management (Chapter 30) 30.3.3.5.*

#### 6.3.2.10 Receive Burst Counter

The total number of Receive burst events. As defined in Section 4.2.2.2.7 of the IEEE 802.3z specifications. *Conforms to IEEE 802.3z D4 Layer Management (Chapter 30) 30.4.3.1.20.*

#### 6.3.2.11 Missed Frames

The total number of events in which frames were missed by the Controller due to buffer overflow. *OID = 1.3.6.1.2.1.16.3*

#### 6.3.2.12 Framing Errors

The total number of frames received with improper End-of Frame delimiter.

#### 6.3.2.13 Rx FIFO Overflow

The number of Receive FIFO overflow events.

#### 6.3.2.14 Jabber Pkt

The total number of frames received that were longer than 1518 octets (excluding framing bits, but including FCS octets), and had a bad FCS with an integral number of octets (FCS error); 1522 octets if *OneLevelVLANTag* register bit is set; or 1538 octets if the *TwoLevelVLANTag* register bit set. *Conforms to IEEE 802.3z D4 Layer Management (Chapter 30) 30.5.1.1.6.*

#### 6.3.2.15 Carrier Event Errors

An event counter for the number of times the Carrier Event is longer than the permissible duration. OID = 1.3.6.1.2.1.16.13

#### 6.3.2.16 In Range Length

A count of frames with a length/type field value between the minimum unpadded LLC MAC Client data size and the maximum allowed LLC MAC Client data size,

inclusive, that does not match the number of LLC MAC Client data octets received. The counter also contains frames with a length/type field value less than the minimum unpadded LLC MAC Client data size. Note: this error is not generated if length field is between 1 and 45 (as this becomes padding case). *Conforms to IEEE 802.3z D4 Layer Management (Chapter 30) 30.3.1.1.23.*

#### 6.3.2.17 Symbol Error

A count of the number of times the receiving media is non-idle (a carrier event) for a period of time greater than or equal to slotTime (see 4.2.4) for half-duplex operation, or greater than or equal to minFrameSize for full-duplex operation, and during which there was at least one occurrence of an event that causes the PHY to indicate "Data reception error" on the GMII (see Table 35-2). *Conforms to IEEE 802.3z D4 Layer Management (Chapter 30) 30.3.2.1.5.*

#### 6.3.2.18 Short Event

An event counter by one for each CarrierEvent with ActivityDuration less than ShortEventMaxTime. In the 1000 Mb/s case ShortEventMaxTime is 72 bits (9 octets). *Conforms to IEEE 802.3z D4 Layer Management (Chapter 30) 30.4.3.1.9.*

#### 6.3.2.19 Runt Counter

A counter for each CarrierEvent that the OctetCount is less than 64. *Conforms to IEEE 802.3z D4 Layer Management (Chapter 30) 30.4.3.1.10.*

#### 6.3.2.20 Frame Too Long

A count of frames received that exceed the maximum permitted frame size, i.e., 1518 octets (excluding framing bits, but including FCS octets) and were otherwise well formed. The maximum frame length could be 1522 octets if *OneLevelVLANTag* register bit is set; or 1538 octets if the *TwoLevelVLANTag* register bit set. *Conforms to IEEE 802.3z D4 Layer Management (Chapter 30) 30.3.1.1.25.*

#### 6.3.2.21 Frame Check Sequence Errors Counter

A count of receive frames that are an integral number of octets in length and do not pass the FCS check. This does not include frames received with frame-too-long, or frame-too-short (frame fragment) error. *Conforms to IEEE 802.3z D4 Layer Management (Chapter 30) 30.3.1.1.6.*

#### 6.3.2.22 CEXT Error

A counter of received frames with no Collision, Short, or Framing error in which the carrier extension was inadequate.

#### 6.3.2.23 Receive Utilization

The utilization of the receive path, calculated as a percentage, using the following formula:

percentage utilization = (.5- Delta/MaxRange)*100

Where:

*MaxRange = UtilWindow*/256 ns

*Delta* = StartVal - CurUtilizationVal (if it is negative, do not discard, use as negative number).

*StartVal:* is the utilization value at the beginning of utilization window.

*UtilWindow*: is the time window over which, user wants the utilization to be computed.

*CurUtilizationVal*: is the current value of receive utilization register.

**Note1:** That after giving *sampleLine* command, *UtilizationVal* becomes 80000000H. Note 2: the utilization computer can sustain about 16 minutes of worst case (worst case is defined as continuous idle or continuous traffic); thus it is necessary to either give sampleLine command at the end of 16 minutes or adjust a value 80000000H for each less than 0 or more than FFFFFFFFH values.

#### 6.3.2.24 64 Rx Byte Frames

The total number of frames (including bad frames) received that were 64 octets in length (excluding framing bits but including FCS octets). *OID = 1.3.6.1.2.1.16.14*

#### 6.3.2.25 65–127 Rx Byte Frames

The total number of frames (including bad frames) received that were between 65 and 127 octets in length inclusive (excluding framing bits but including FCS octets). *OID = 1.3.6.1.2.1.16.15*

#### 6.3.2.26 128–255 Rx Byte Frames

The total number of frames (including bad frames) received that were between 128 and 255 octets in length inclusive (excluding framing bits but including FCS octets). *OID = 1.3.6.1.2.1.16.16*

#### 6.3.2.27 256–511 RX Byte Frames

The total number of frames (including bad frames) received that were between 256 and 511 octets in length inclusive (excluding framing bits but including FCS octets). *OID = 1.3.6.1.2.1.16.17*

#### 6.3.2.28 512–1023 Rx Byte Frames

The total number of frames (including bad frames) received that were between 512 and 1023 octets in length inclusive (excluding framing bits but including FCS octets). *OID = 1.3.6.1.2.1.16.18*

#### 6.3.2.29 1024-MaxSize Rx Byte Frames

The total number of frames (including bad frames) received that were between 1024 and 1518 octets in length inclusive (excluding framing bits but including FCS octets). If the *OneLevelVLANTag* register bit is set, this statistic counts frames that are between 1024 and1522 octets (excluding framing bits but including FCS octets). ). If the *TwoLevelVLANTag* register bit is set, this statistic counts frames that are between 1024 and1538 octets (excluding framing bits but including FCS octets). *OID = 1.3.6.1.2.1.16.19*

# 7.0 Design Considerations

## 7.1 XMAC II DESIGN SUPPORT: GIGABIT DEVELOPMENT KIT

XaQti offers a Gigabit Ethernet Design Kit II as a design and evaluation tool for the XMAC II. The kit consists of an Industry-standard 32-bit, 33 MHz PCI card utilizing the XMAC II and comes with GigaBlaster, a diagnostic software package.

The GDK II has an integrated hardware Packet Generator which provides for automatic or manual packet construction, and load generation. The integrated hardware Packet Processor provides for monitoring incoming packets. The GDK II design features an Onboard independent 256 Kbyte Transmit and Receive SRAM memory. The GDK II supports Full Duplex operating mode, Supports packet sizes from 4 bytes to 2K bytes. The User defines and controls source and destination address, IPG, preamble, packet length, data pattern, CRC, and padding. The GDK II is capable of injecting errors packets such as CRC errors, short packets etc. There are On-screen graphs for visual representation of key statistics. The GDK II also allows for frame by frame IPG control, Pause frame generation and is extremely useful for traffic generation for gigabit link tests.

## 7.2 FIFO AND NODE PROCESSOR INTERFACES

The data transfer rate at Node Processor may be as high as 40 MHz and data transfer rate at the Host Interface (Rx and Tx FIFOs) may be as high as 66 MHz. Appropriate design guidelines and minimization of signal traces should be practiced.

## 7.3 10-BIT PHY INTERFACE

The XMAC II interfaces directly to 10-bit FC-0 1.25 gbps Gigabit Ethernet Transceiver (SERDES) devices without any external logic.

A list of SERDES devices that are compatible and have been tested with the XMAC II is provided below:

| Vendor | Device |
|---|---|
| AMCC | S2062 |
| Vitesse | VSC 7135 |
| HP | HDMP-1536, 1546 |

### *7.3.1 PCB Layout Considerations*

The clock and data lines to and from the 10-bit PHY interface clocks at a data rate of 125 MHz. Care should be taken to minimize signal traces and loading on the signals.

Design reference guides available upon request.

## 7.4 POWER SUPPLY DECOUPLING

Standard guidelines for decoupling the Controller's $V_{DD}$ power inputs are appropriate and should be implemented by the circuit designer.

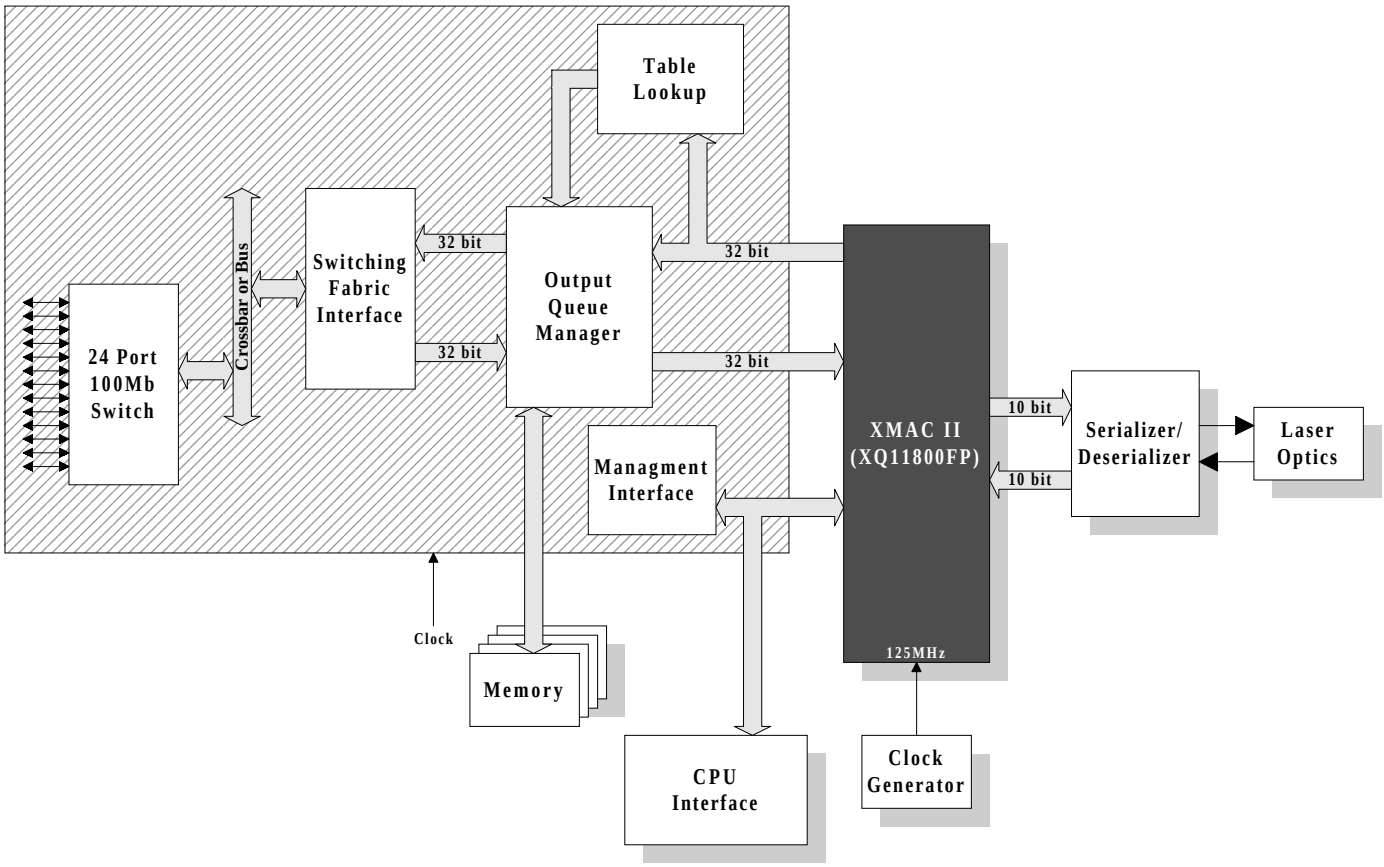
Design reference guides available upon request.

## 7.5 MODELS AND DESIGN AIDS

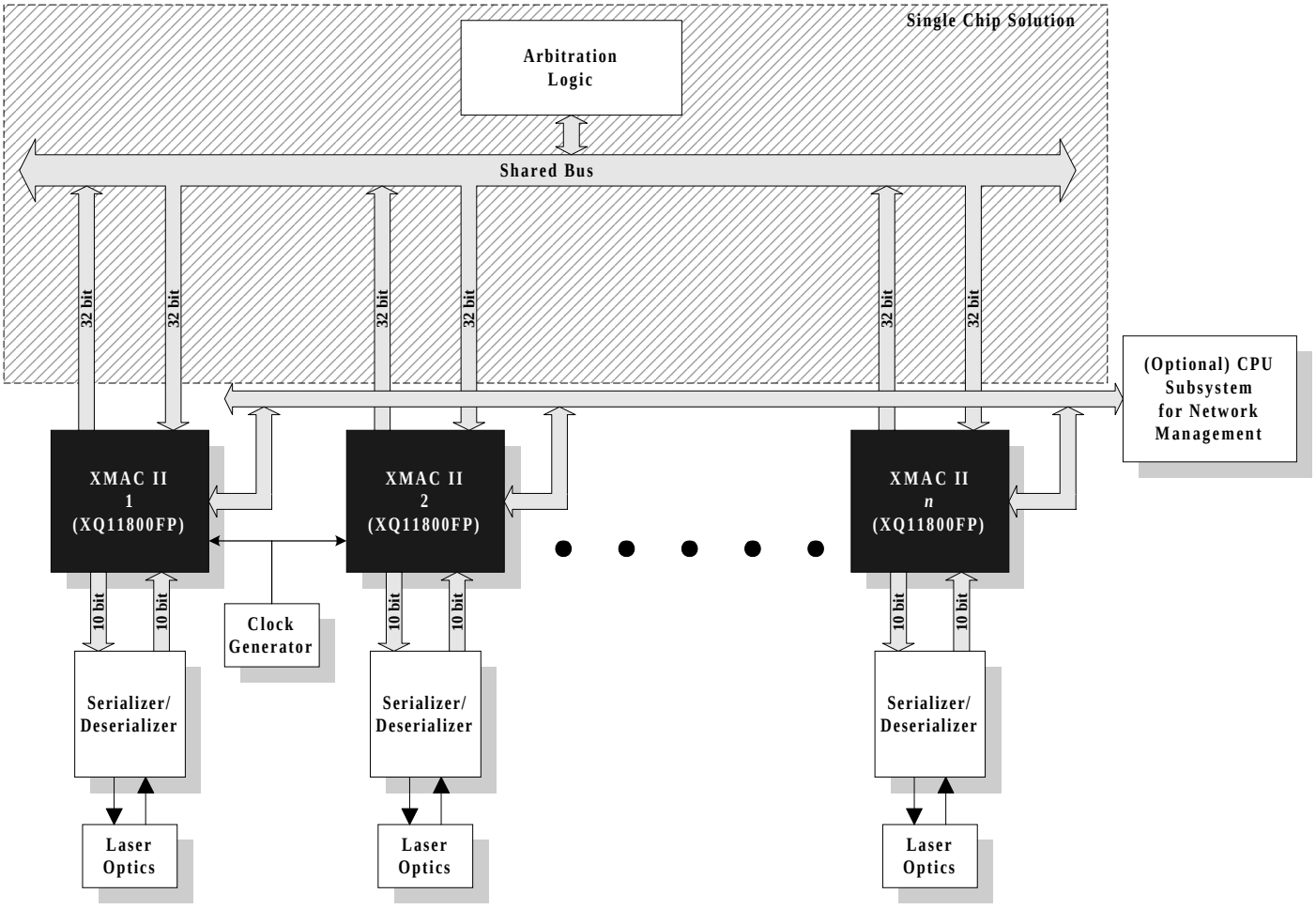
A complete design package is available for qualified systems customers including complete schematics, FPGA VHDL source, Windows 95 software source code and PCB layout information. The design supports 1000BASE-SX and 1000BASE-LX media.

## 8.0 Typical XMACII Applications

Typical Block Diagram of switch uplink design using the XMAC II



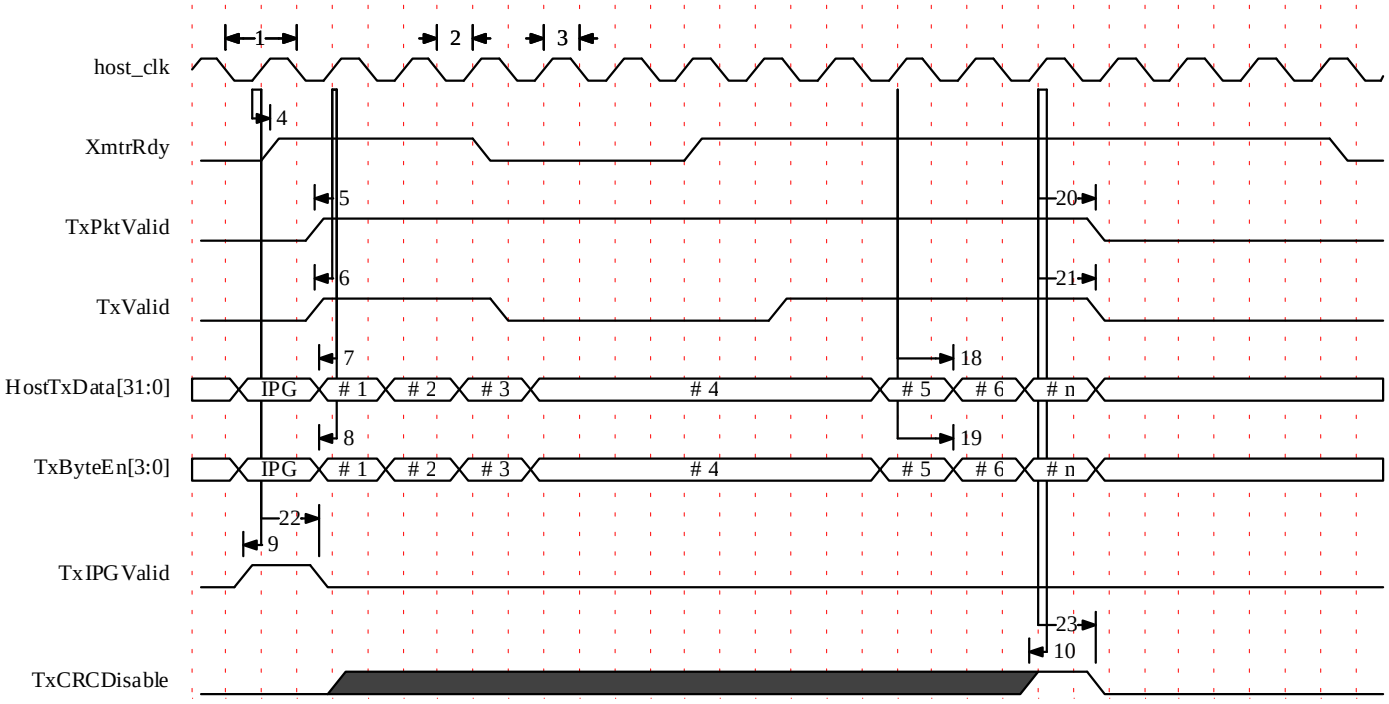
## Typical Block Diagram of buffered distributor design using the XMAC II



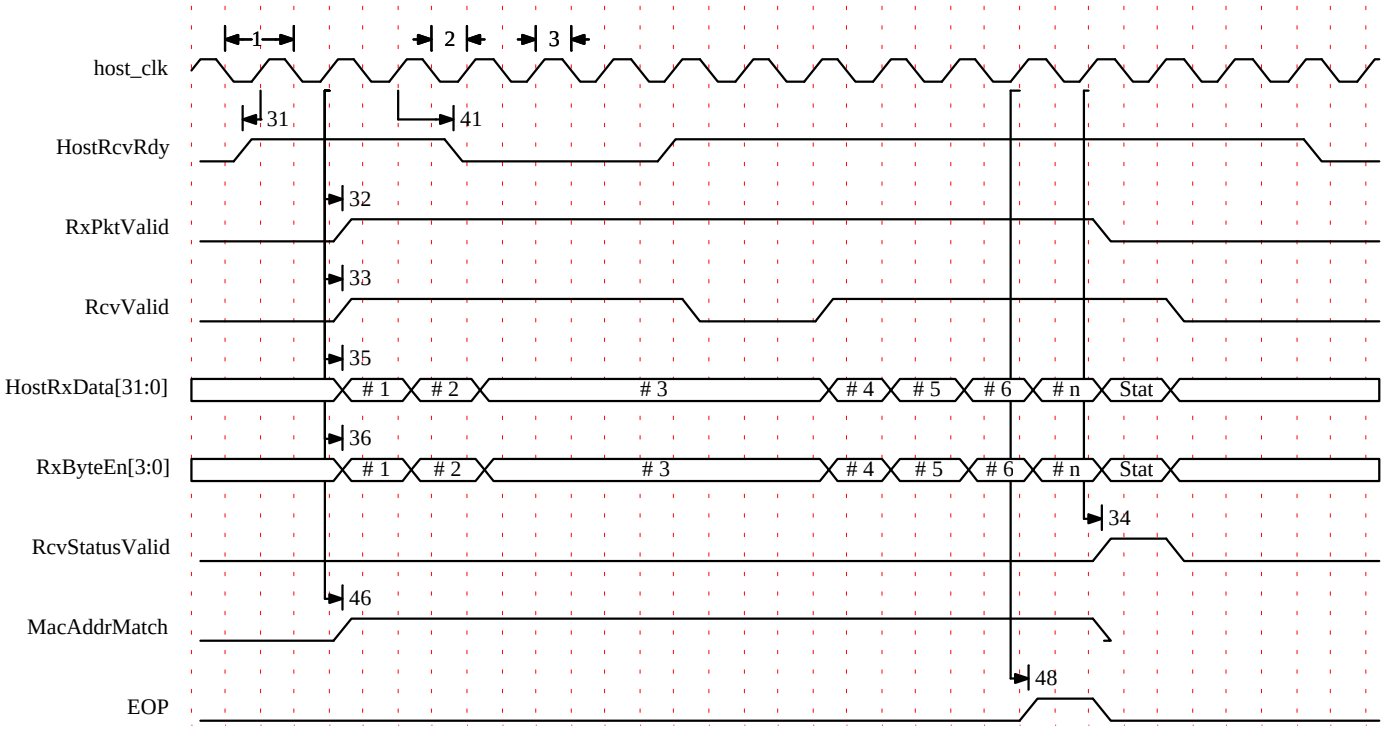
## 9.0 XQ11800FP Timing Diagrams

This section provides timing diagrams for Tx and Rx FIFOs, the Node Processor.

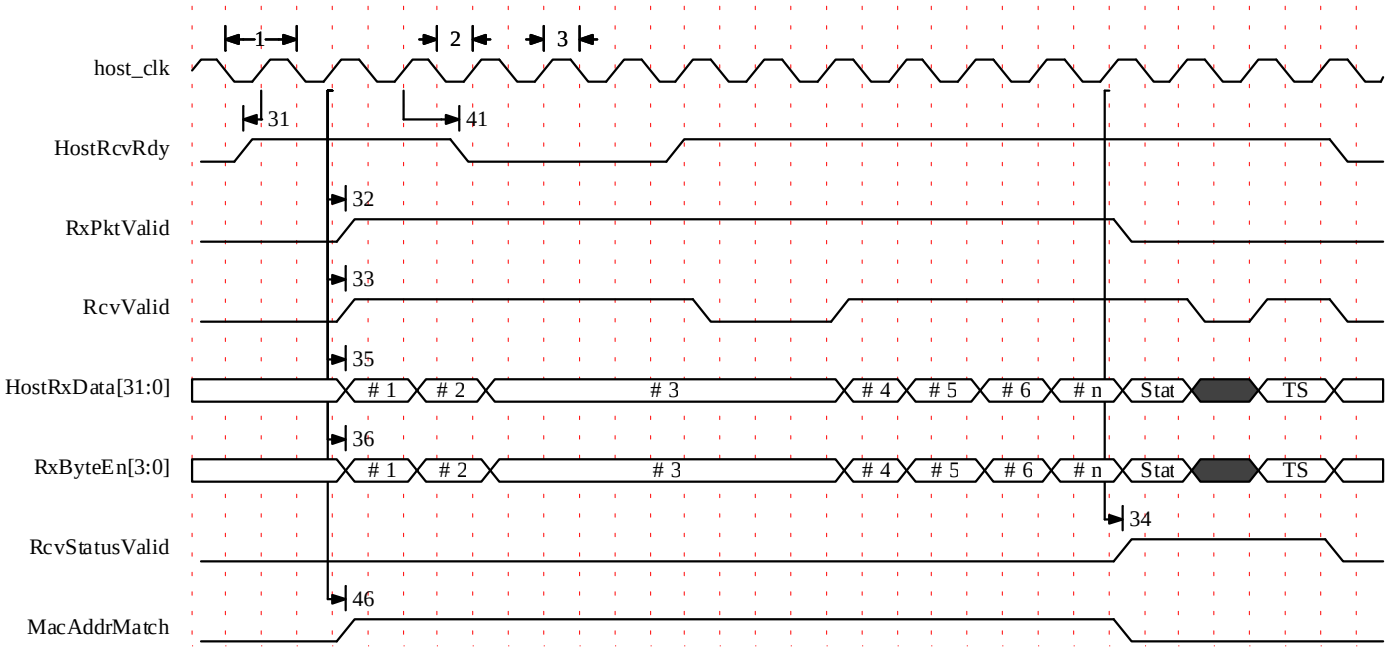
### XMAC II Transmit Timing Diagram



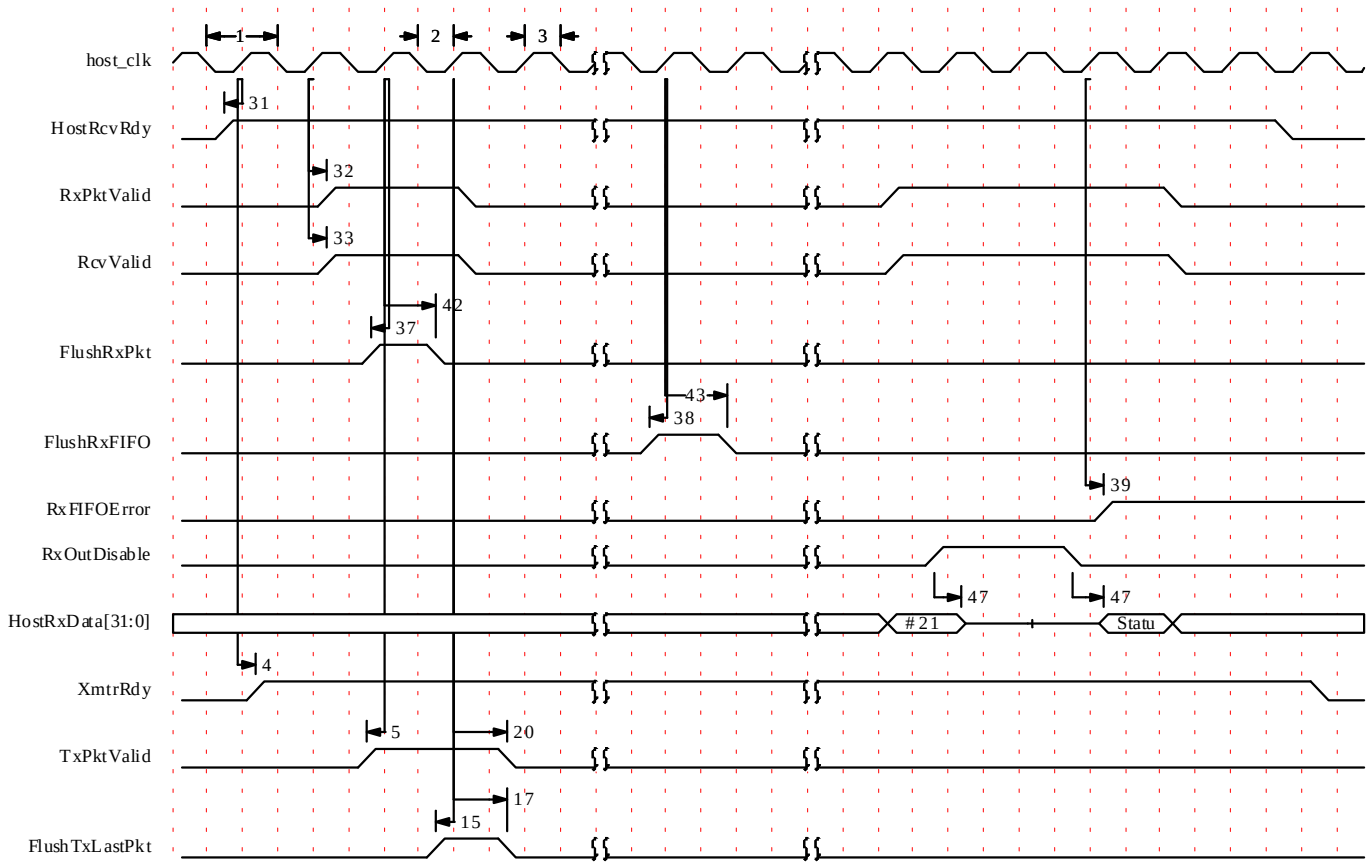
## XMAC II Receive Timing Diagram, without Timestamp



## XMAC II Receive Timing Diagram, with Timestamp



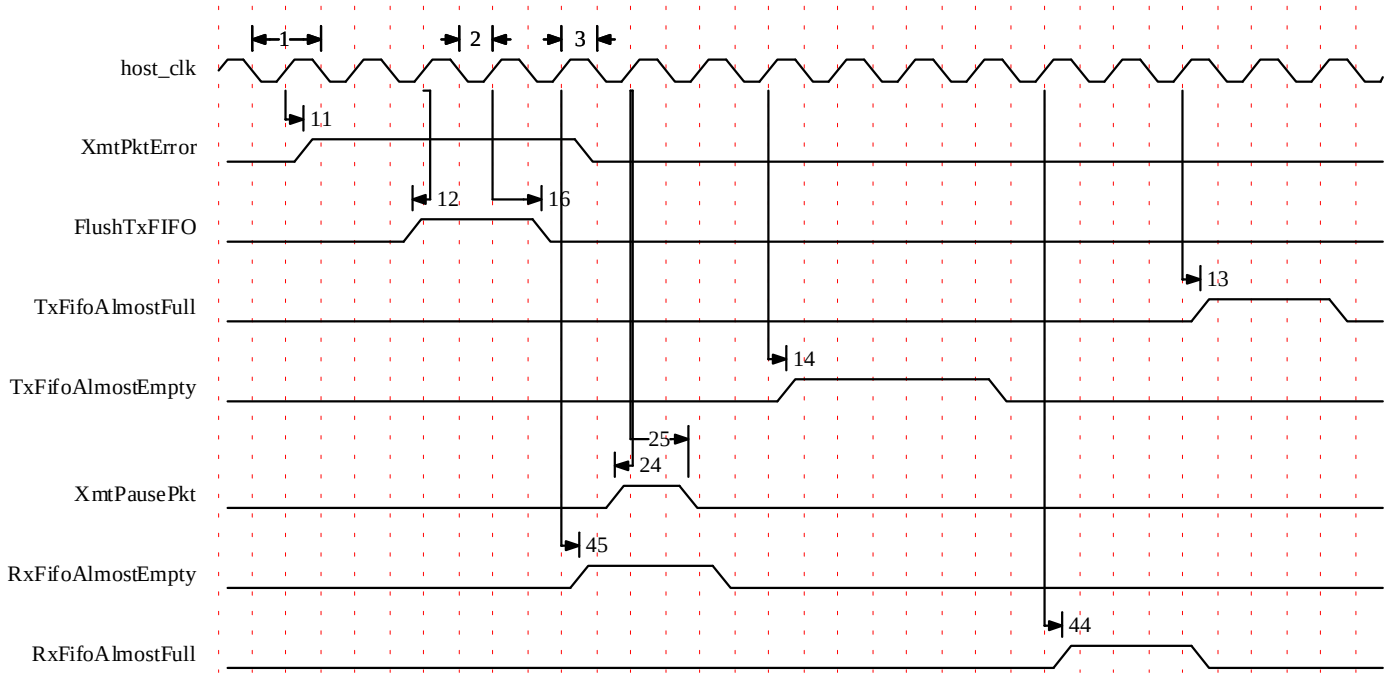
## XMAC II Host Transmit/Receive Control Status Timing, Part1




*Note 1: FlushTxLastPkt can be asserted at any time after the first word of a packet but not after the last word of the packet.*

*Note 2: There is no defined timing relationship between individual signals [except with respect to HOST_CLK] in this diagram.*

## XMAC II Host Transmit/Receive Control Status Timing

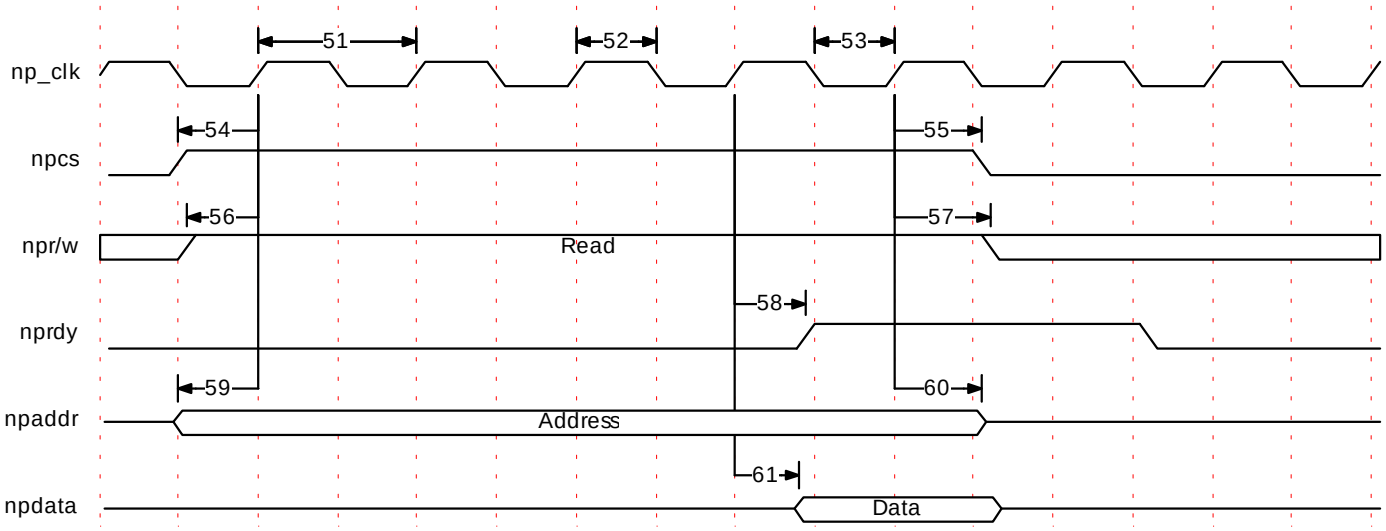



*Note : There is no defined timing relationship between individual signals [except with respect to HOST_CLK] in this diagram.*

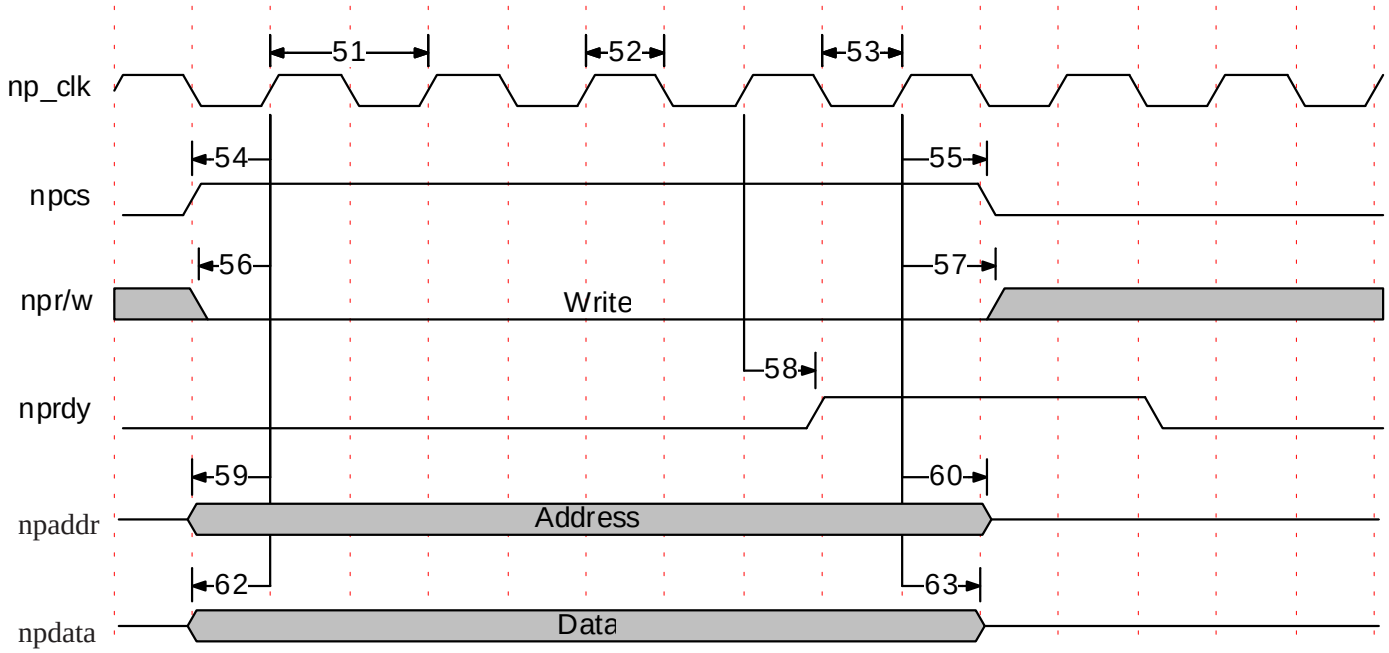
## Timing Parameters

| Parameter | Parameter Description | Min | Max | Unit |
|---|---|---|---|---|
| **Transmit Timing Parameters** | | | | |
| 1 | HOST_CLK Period | 15.15 | 30.30 | ns |
| 2 | HOST_CLK ON Period | 40.00 | 60.00 | % |
| 3 | HOST_CLK OFF Period | 40.00 | 60.00 | % |
| 4 | CLK to XmtrRdy Delay | 2.61 | 11.96 | ns |
| 5 | TxPktValid Setup Time | 2.09 | - | ns |
| 6 | TxValid Setup Time | 1.09 | - | ns |
| 7 | HostTxData Setup Time | 1.31 | - | ns |
| 8 | TxByteEn Setup Time | 1.08 | - | ns |
| 9 | TxIPGValid Setup Time | 1.79 | - | ns |
| 10 | TxCRCDisable Setup Time | 0.78 | - | ns |
| 11 | CLK to XmtPktError Delay | 3.12 | 14.40 | ns |
| 12 | FlushTxFIFO Setup Time | 5.14 | - | ns |
| 13 | CLK to TxFIFOAlmostFull Delay | 2.83 | 12.90 | ns |
| 14 | CLK to TxFIFOAlmostEmpty Delay | 2.82 | 12.89 | ns |
| 15 | FlushTxLastPkt Setup Time | 0.87 | - | ns |
| 16 | FlushTxFIFO Hold Time | 0.96 | - | ns |
| 17 | FlushTxLastPkt Hold Time | 1.94 | - | ns |
| 18 | HostTxData Hold Time | 2.14 | - | ns |
| 19 | TxByteEn Hold Time | 1.36 | - | ns |
| 20 | TxPktValid Hold Time | 1.28 | - | ns |
| 21 | TxValid Hold Time | 0.92 | - | ns |
| 22 | TxIPGValid Hold Time | 2.62 | - | ns |
| 23 | TxCRCDisable Hold Time | 2.20 | - | ns |
| 24 | XmtPausePkt Setup Time | 0.81 | - | ns |
| 25 | XmtPausePkt Hold Time | 2.27 | - | ns |
| 26 | ForceXmt Setup Time | 0.95 | - | ns |
| 27 | ForceXmt Hold Time | 1.80 | - | ns |
| 28 | TxTransp.Mode Setup Time | 0.82 | - | ns |
| 29 | TxTransp.Mode Hold Time | 2.23 | - | ns |
| **Receive Timing Parameters** | | | | |
| 31 | HostRcvRdy Setup Time | 5.86 | - | ns |
| 32 | CLK to RxPktValid Delay | 2.43 | 11.10 | ns |
| 33 | CLK to RxValid Delay | 2.43 | 11.13 | ns |
| 34 | CLK to RcvStatusValid Delay | 2.42 | 11.09 | ns |
| 35 | CLK to HostRxData Delay | 2.12 | 11.64 | ns |
| 36 | CLK to RxByteEn Delay | 2.27 | 11.72 | ns |
| 37 | FlushRxPkt Setup Time | 0.90 | - | ns |
| 38 | FlushRxFIFO Setup Time | 0.68 | - | ns |
| 39 | CLK to RxFIFOError Delay | 3.06 | 14.16 | ns |
| 40 | RxOutDisable(Data) Enable/Disable Time | 2.12 | 12.29 | ns |
| 41 | HostRcvRdy Hold Time | 2.42 | - | ns |
| 42 | FlushRxPkt Hold Time | 1.71 | - | ns |
| 43 | FlushRxFIFO Hold Time | 2.43 | - | ns |
| 44 | CLK to RxFIFOAlmostFull Delay | 3.20 | 14.77 | ns |
| 45 | CLK to RxFIFOAlmostEmpty Delay | 3.11 | 14.42 | ns |
| 46 | CLK to MACAddressMatch Delay | 2.43 | 11.10 | ns |
| 47 | RxOutDisable(ByteEn)) Enable/Disable Time | 2.08 | 10.95 | ns |
| 48 | CLK to EOP Delay | 2.27 | 10.22 | ns |

## XMAC II Node Processor Read Timing Diagram



## XMAC II Node Processor Write Timing Diagram



## Node Processor Timing Parameters

| Parameter | Parameter Description | Min | Max | Unit |
|---|---|---|---|---|
| **NP Read/Write Operation Timing Parameters** | | | | |
| 51 | NP_CLK Period | 25.00 | 50.00 | ns |
| 52 | NP_CLK ON Period | 40.00 | 60.00 | % |
| 53 | NP_CLK OFF Period | 40.00 | 60.00 | % |
| 54 | NPCS Setup Time | 5.26 | | ns |
| 55 | NPCS Hold Time | 1.41 | | ns |
| 56 | NPR/W Setup Time | 4.87 | | ns |
| 57 | NPR/W Hold Time | 1.08 | | ns |
| 58 | CLK to NPRDY Delay | 3.06 | 14.97 | ns |
| 59 | NPADDR Setup Time | 3.70 | | ns |
| 60 | NPADDR Hold Time | 1.99 | | ns |
| 61 | CLK to NPDATA Delay (for Read op.) | 2.80 | 14.18 | ns |
| 62 | NPDATA Setup Time (for Write op.) | 1.05 | | ns |
| 63 | NPDATA Hold Time (for Write op.) | 1.75 | | ns |

## PHY Interface Timing Parameters

| Parameter | Parameter Description | Min | Typical | Max | Unit |
|---|---|---|---|---|---|
| **GMII Mode PHY Transmit Timing Parameters** | | | | | |
| 70 | TXCLK Period | — | 8.00 | — | ns |
| 71 | TXCLK ON Period | 40 | — | 60 | % |
| 72 | TXCLK OFF Period | 40 | — | 60 | % |
| 73 | TXD (7:0) Setup Time w.r. to TXCLK | 3.76 | — | — | ns |
| 74 | TXD (7:0) Hold Time w.r. to TXCLK | 3.92 | — | — | ns |
| 75 | TX_EN Setup Time w.r. to TXCLK | 3.76 | — | — | ns |
| 76 | TX_EN Hold Time w.r. to TXCLK | 3.92 | — | — | ns |
| 77 | TX_ER Setup Time w.r. to TXCLK | 3.76 | — | — | ns |
| 78 | TX_ER Hold Time w.r. to TXCLK | 3.92 | — | — | ns |
| **GMII Mode PHY Receive Timing Parameters** | | | | | |
| 79 | RCLK Period | — | 8.00 | — | ns |
| 80 | RCLK ON Period | 40 | — | 60 | % |
| 81 | RCLK OFF Period | 40 | — | 60 | % |
| 82 | Setup Time for RXD(7:0) | — | — | 1.14 | ns |
| 83 | Hold Time for RXD(7:0) | 0.41 | — | — | ns |
| 84 | Setup Time for RX_DV | — | — | 1.29 | ns |
| 85 | Hold Time for RX_DV | 0.94 | — | — | ns |
| 86 | Setup Time for RX_ER | — | — | 1.32 | ns |
| 87 | Hold Time for RX_ER | 0.58 | — | — | ns |
| **FC-0 Mode PHY Transmit Timing Parameters** | | | | | |
| 90 | PODAT Setup Time to GTX_CLK | 3.76 | — | — | ns |
| 91 | PODAT Hold Time to GTX_CLK | 3.92 | — | — | ns |
| **FC-0 Mode PHY Receive Timing Parameters** | | | | | |
| 92 | PIDAT Setup Time | — | — | 1.37 | ns |
| 93 | PIDAT Hold Time | 1.44 | — | — | ns |

## 9.1 RESET AND MISC. SIGNAL SPECIFICATIONS

### XMAC II Reset

The XQ11800FP reset signal (*/RESET*) is an asynchronous signal that must be active for at least 40 nanoseconds with stable power.

### Asynchronous Input Signals

The following are asynchronous input signals and must be valid for at least 40 nanoseconds.

- /RESET
- COL
- COMDET
- SIGSTAT
- CRS
- GPINPUT

### Asynchronous Output Signals

The following are asynchronous output signals.

- /LINK-SYNC
- EN_COM_DET
- PHY_LP_EN
- /NPINT

## 9.2 ELECTRICAL AND ENVIRONMENTAL SPECIFICATIONS

This section contains the elecrical and environmental specification for the XQ11800FP:

- Absolute Maximum Ratings
- Recommended Operating Conditions
- Input/Output Capacitance
- DC Characteristics
- Reset Specifications
- Clock Requirements
- ESD Consideration
- Moisture Sensitivity and reflow processing

## 9.3 ABSOLUTE MAXIMUM RATINGS

| Parameter | Conditions | Symbol | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| Supply Voltage | | $V_{DD}$ | -0.3 | | 4.6 | V |
| Reference Voltage | | $V_{CC}$ | -0.3 | | 5.7 | V |
| Voltage on non-TTL Pins | | $V_{I3}$ | -0.3 | | $V_{DD}$ + 0.3 | V |
| Voltage on TTL Pins | | $V_{IT}$ | -0.3 | | 6.0 | V |
| ESD protection voltage | 'human body model' | HBM | -- | | 2000 | V |
| Power Dissipation | | $P_D$ | | | 1.2 | W |
| Ambient Operating Temperature | | $T_A$ | -40 | | 70 | ºC |
| Storage Temperature | | $T_{STG}$ | -55 | | 150 | ºC |
| Lead Temperature (Soldering, 10 sec) | | | | | 220 | ºC |
| Body Temperature (Soldering, 10 sec) | | | | | 220 | ºC |

**Caution:** Exposure of the device to conditions greater than the maximum rating or less than the minimum rating can cause permanent damage to the part. Operation of the device at the maximum or minimum ratings for extended periods of time can affect the reliability of the part.

## 9.4 RECOMMENDED OPERATING CONDITIONS (AT NORMAL OPERATIONS)

| Parameter | Conditions | Symbol | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| Supply Voltage (3.3v ± 5%) | | $V_{DD}$ | 3.1 | 3.3 | 3.5 | V |
| Reference Voltage (5.0v ± 5%) | | $V_{CC}$ | 4.75 | 5.00 | 5.25 | V |
| Supply Current for $V_{DD}$ | | $I_{DD}$ | | | 365 | mA |
| Supply Current for $V_{CC}$ | | $I_{CC}$ | | | 30 | µA |
| Ambient Operating Temperature | | $T_A$ | 0 | | 70 | ºC |
| Transition Rise and Fall Time | | $t_R$ & $t_F$ | 0 | | 100 | ns |
| Thermal Impedance | Junction-to-case | $Z_{\Theta JC}$ | | 6 | | ºC/W |
| Thermal Impedance | Junction-to-ambient, airflow=0 | $Z_{\Theta JA}$ | | 35 | | ºC/W |

## 9.5 INPUT/OUTPUT CAPACITANCE

| Parameter | Conditions | Symbol | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| Input Capacitance | | $C_I$ | | 7 | 15 | pF |
| Output Capacitance | | $C_O$ | | 7 | 15 | pF |

## 9.6 DC CHARACTERISTICS

| Parameter | Conditions | Symbol | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| **TTL Input** | | | | | | |
| Input High Voltage | | $V_{IHT}$ | 2.0 | | 5.25 | V |
| Input Low Voltage | | $V_{ILT}$ | 0 | | 0.8 | V |
| Input Leakage Current | $V_I$ = 5.25V or $V_{SS}$ | $I_{ILT}$ | | | ±10 | µA |
| **TTL Output** *** | | | | | | |
| Output High Voltage | $I_{OH}$ = -4mA, $V_I$ = $V_{DD}$ or $V_{SS}$ | $V_{OHT}$ | 2.4 | | | V |
| Output Low Voltage | $I_{OL}$ = 4mA, $V_I$ = $V_{DD}$ or $V_{SS}$ | $V_{OLT}$ | | | 0.4 | V |
| Output Leakage Current | $V_O$ = 5.25V or $V_{SS}$ | $I_{OLT}$ | | | ±10 | µA |
| **CMOS Input** | | | | | | |
| Input High Voltage | | $V_{IHC}$ | $V_{DD}$ * 0.7 | | $V_{DD}$ | V |
| Input Low Voltage | | $V_{ILC}$ | 0 | | $V_{DD}$ * 0.3 | V |
| Input Leakage Current | $V_I$ = $V_{DD}$ or $V_{SS}$ | $I_{ILC}$ | | | ±5 | µA |
| **CMOS Output** | | | | | | |
| Output High Voltage | $I_{OH}$ = -4mA, $V_I$ = $V_{DD}$ or $V_{SS}$ | $V_{OHC}$ | $V_{DD}$ -0.6 | | | V |
| Output Low Voltage | $I_{OL}$ = 4mA, $V_I$ = $V_{DD}$ or $V_{SS}$ | $V_{OLC}$ | | | 0.4 | V |
| **CMOS OD** | | | | | | |
| Output High Voltage | | $V_{OHO}$ | | | | V |
| Output Low Voltage | $I_{OL}$ = 4mA, $V_I$ = $V_{DD}$ or $V_{SS}$ | $V_{OLO}$ | | | 0.4 | V |
| Output Leakage Current | $V_O$ = 5.25V or $V_{SS}$ | $I_{OLO}$ | | | ±10 | µA |

***Note - For CMOS2 type outputs $I_{OH}$ = -2mA, $I_{OL}$ = 2mA

- For CMOS4 type outputs $I_{OH}$ = -4mA, $I_{OL}$ = 4mA
- For CMOS8 type outputs $I_{OH}$ = -8mA, $I_{OL}$ = 8mA
- For CMOS16 type outputs $I_{OH}$ = -16mA, $I_{OL}$ = 16mA
- For TTL4 type outputs $I_{OH}$ = -4mA, $I_{OL}$ = 4mA
- For TTL8 type outputs $I_{OH}$ = -8mA, $I_{OL}$ = 8mA
- The TTL Tri Pins can be tri-stated

## 10.0 ESD Handling Consideration

The XQ11800FP should be handled in accordance with industry-standard ESD protection procedures.

### 10.1 MOISTURE SENSITIVITY AND REFLOW PROCESSING

THESE DEVICES ARE MOISTURE SENSITIVE (LEVEL 3 PER JEDEC A112)

- Shelf life in sealed bag: 12 months minimum at <40 degrees C and < 90% relative humidity (RH).
- After the bag seal is broken, the following three options apply:

1) Use the devices within 168 hours (1 week), under conditions of < 30°C and < 60% RH.
2) Store devices in a controlled atmosphere at less than 20% RH, such as in a desiccator cabinet with controlled dry air or dry nitrogen.
3) Seal the parts completely under a partial vacuum with an impulse sealer in an approved Moisture Barrier Bag (MBB) within 12 hours, using fresh desiccant and a Humidity Indicator Card (HIC).

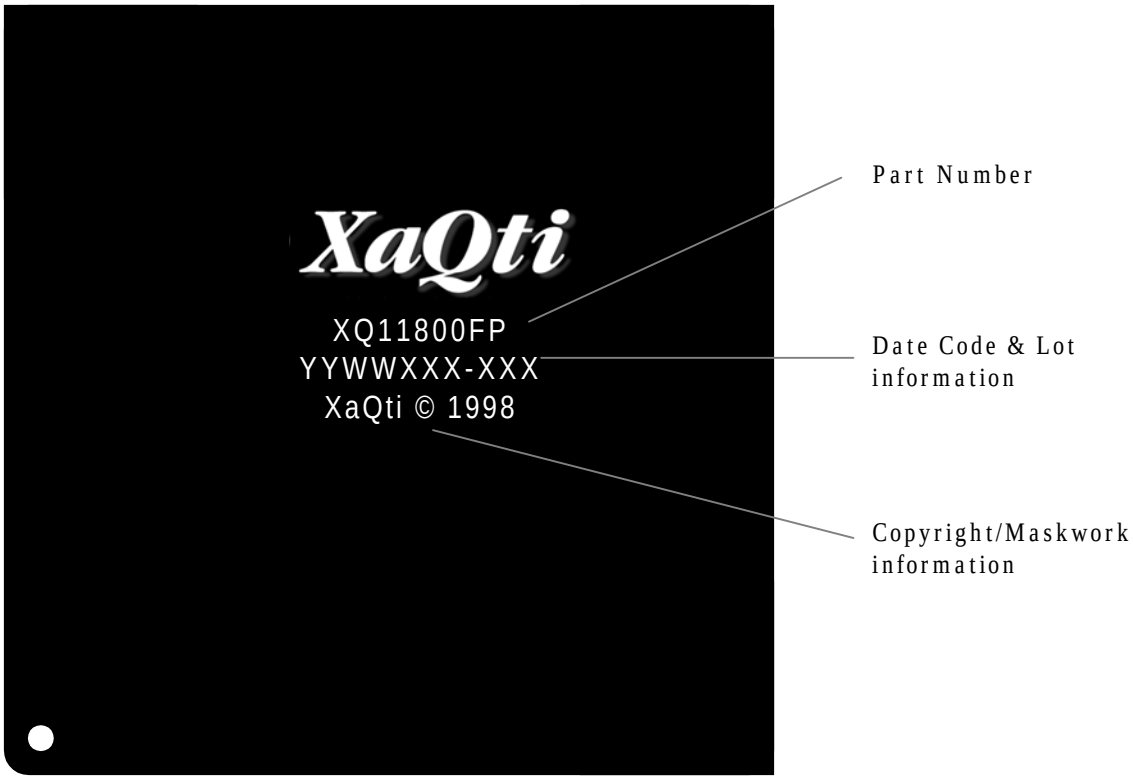
- If the above conditions have not been met, or if the Humidity Indicator Card is >20% when read at a temperature between 18°C and 28°C, the devices will require baking. Please use one of the following two bake schedules:

1) 24 hours in air at 125°C.
2) 192 hours in a controlled atmosphere of 40°C, equal to or less than 5% RH.

## 11.0 Mechanical Specifications

The XQ11800FP is a 240-pin PQFP package.
The following figure shows the part marking

Part Marking



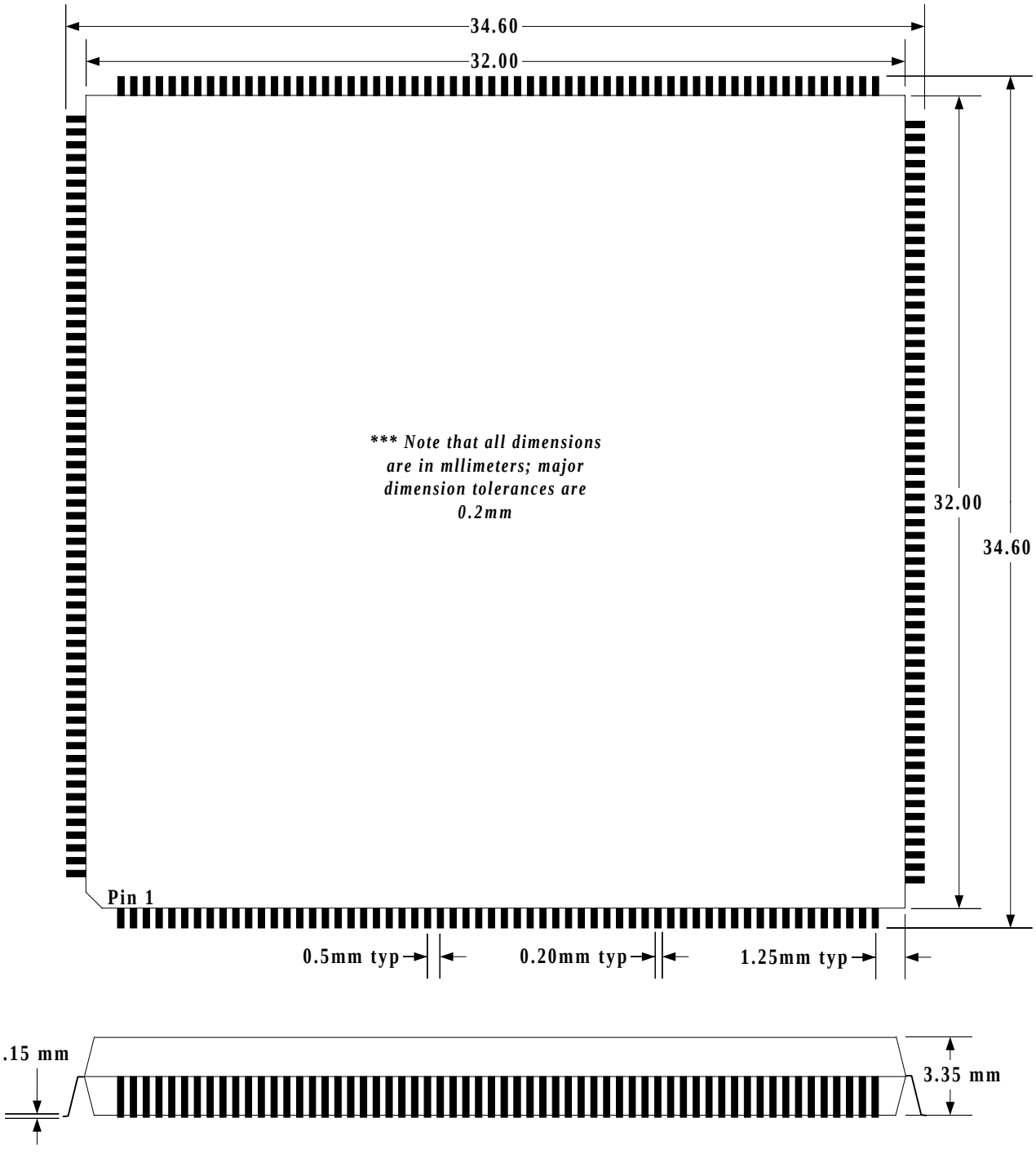

### *11.0.1 Date and Lot Code Marking*

YYWWXXX-XXX

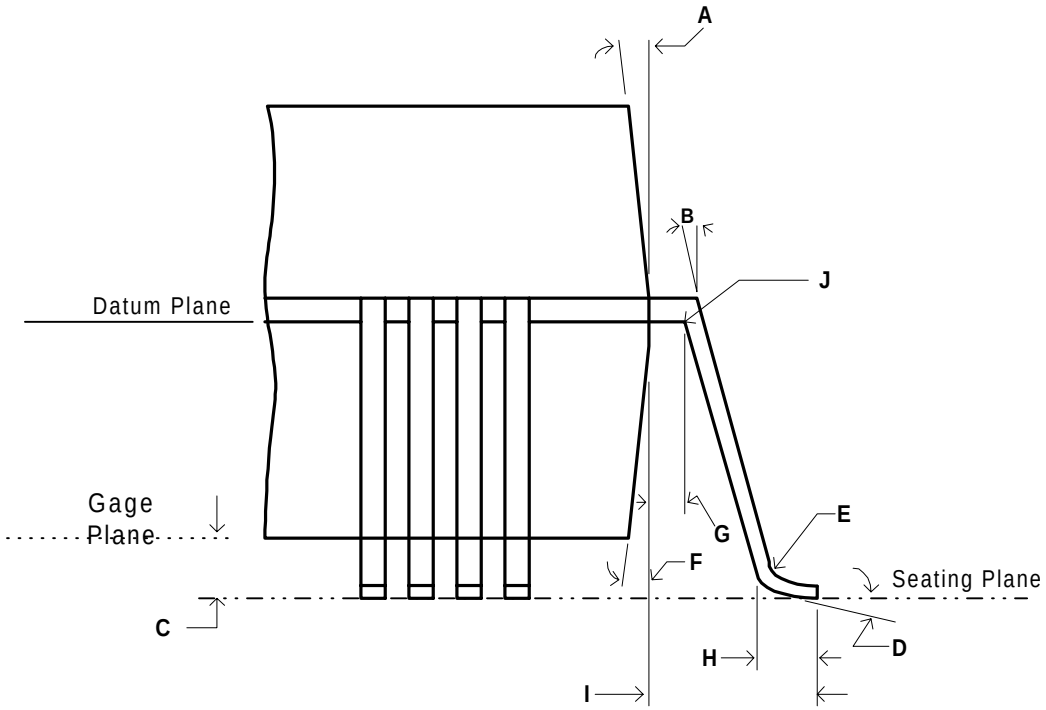
| | |
|---|---|
| YY | YEAR |
| WW | ASSEMBLY WEEK |
| XXX-XXX | LOT TRACEABILITY INFORMATION |

## 11.1 PATENT PENDING

## XMAC II Package Mechanical Diagram



## PFPQ Lead Dimensional Attributes




| Dimension | Symbol | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| Upper Package Angle (Avg top-to-bot | A | | 13 | | degrees |
| Lead Vertical Angle | B | 0 | | 10 | degrees |
| Standoff | C | | 0.4 | | mm |
| Foot Angle | D | 0 | | 7.0 | degrees |
| Lower Radius of Foot | E | | 0.2 | | mm |
| Lower Package Angle | F | | 12 | | degrees |
| Lead Protusion | G | | 0.4 | | mm |
| Foot Length | H | | 0.5 | | mm |
| Foot Protusion | I | | 1.3 | | mm |
| Upper Radius of Lead | J | | 0.2 | | mm |

## 12.0 Product Support and Documentation

If you need technical support, or additional documentation,: call the local XaQti Distributor or Representative; email **support@xaqti.com**; or visit the XaQti Corporation World Wide Web site:

**http://www.xaqti.com**