Ethernet Basics and DWC MAC

# Ethernet PHY and OSI layer

A diagram of a computer network

Description automatically generated

URL: <https://docs.amd.com/r/en-US/pg051-tri-mode-eth-mac/Ethernet-Data-Format>

# Half-Duplex Frame Transmission

In a half-duplex system, the CSMA/CD media access method defines how two or more stations share a common medium.

1. Even when it has nothing to transmit, the Ethernet MAC monitors the Ethernet medium for traffic by watching the carrier sense signal (CRS) from the external PHY. Whenever the medium is busy (CRS = 1 ), the Ethernet MAC defers to the passing frame by delaying any pending transmission of its own.
2. After the last bit of the passing frame (when the carrier sense signal changes from TRUE to FALSE), the Ethernet MAC starts the timing of the interframe gap.
3. The Ethernet MAC resets the interframe gap timer if the carrier sense becomes TRUE during the period defined by “interframe gap part 1 (IFG1).” IEEE Std 802.3-2008 states that this should be the first 2/3 of the interframe gap timing interval (64-bit times) but it can be shorter and as small as zero. The purpose of this option is to support a possible brief failure of the carrier sense signal during a collision condition and is described in paragraph 4.2.3.2.1 of the IEEE standard.
4. To ensure fair access to the bus, the Ethernet MAC does not reset the interframe gap timer if carrier sense becomes TRUE during the period defined by “interframe gap part 2 (IFG2).” IEEE Std 802.3-2008 states that this should be the last 1/3 of the interframe gap timing interval.

After initiating a transmission, if the message collides with the message of another station (COL = 1 ), each transmitting station intentionally continues to transmit (jam) for an additional predefined period (32-bit times for 10/100 Mbbs) to ensure the propagation of the collision throughout the system. The station remains silent for a random amount of time (back off) before attempting to transmit again.

A station can experience a collision during the beginning of its transmission (the collision window) before its transmission has had time to propagate to all stations on the bus. After the collision window is passed, a transmitting station acquires the bus. Subsequent collisions (late collisions) are avoided because all other (properly functioning) stations are assumed to have detected the transmission and trying to defer it.

# Full-Duplex Frame Transmission

In a full-duplex system, there is a point-to-point dedicated connection between two Ethernet devices, capable of simultaneously transmit and receive with no possibility of collisions. The Ethernet MAC does not use the carrier sense signal from the external PHY because the medium is not shared and the Ethernet MAC only needs to monitor its own transmissions. After the last bit of an Ethernet MAC frame transmission, the Ethernet MAC starts the interframe gap timer and defer transmissions until the IFG count completes. The minimum value supported for the IFG depends on the TEMAC solution options and the current mode of operation.

If the TEMAC solution has been built with half-duplex support, the IFG delay is 96-bit times. When IFG Adjustment is enabled, greater than 64-bit times, the value presented on tx\_ifg\_delay .

If the TEMAC solution has been built with only full-duplex support, the IFG delay is 96-bit times. When IFG Adjustment is enabled, greater than 32-bit times, the value presented on tx\_ifg\_delay .

# Flow Control Using IEEE 802.3

<https://docs.amd.com/r/en-US/pg051-tri-mode-eth-mac/Flow-Control-Using-IEEE-802.3>

# Bandwidth Requirements

<https://docs.amd.com/r/en-US/pg051-tri-mode-eth-mac/Low-Frequency-Statistical-Counters>

The frequency of stats\_ref\_clk is flexible but depends upon both the number of counters and the maximum frequency supported by the MAC. The low-frequency increment vectors can update at a maximum rate of one per minimum sized Ethernet frame. This translates to 584 ns when running at 1 Gbps (64 bytes of minimum Ethernet frame size, plus 1-byte of minimum received preamble, plus eight bytes of minimum received interframe gap, at a byte rate of 1-byte per 8 ns).

With stats\_ref\_clk set to 125 MHz, 36 statistical counters can be safely updated between successive Ethernet frames (584 ns divided by the 8 ns clock period of stats\_ref\_clk , divided by two because a counter update requires two accesses). As this is less than the provided 44 counters, extra decode logic is included to take advantage of the frame size counters’ one-hot status (that is, only one of the seven RX and one of the seven TX frame size counters can update on a per packet basis).

The round-robin function that controls which counter is being accessed, only accesses the required frame size counter and skips the other five. This means the 44 counters supported only require 32 counter accesses. However, this does mean that the stats\_ref\_clk should be at least as fast as the clock used for the maximum rate supported by the MAC (125 MHz at 1 Gbps or 12.5 MHz at 10/100 Mbps).

# Frame Filter

The MAC can be configured with an optional frame filter.

The frame filter performs two functions:

* Checking if any received packet matches one of the predefined Destination Address values: Pause Address, Broadcast Address, User Defined Unicast Address and the special multicast Pause Address.
* Comparing the first 64 bytes of a received packet against a user defined pattern.

In the case of the Destination Address comparisons, the results are used in other blocks within the MAC, such as flow control and in the generation of statistics vectors.

# Using the Frame Filter

<https://docs.amd.com/r/en-US/pg051-tri-mode-eth-mac/Using-the-Frame-Filter>

# Usage of frame filter to implement VLAN priority-based filtering

 In a standard VLAN Ethernet frame, the VLAN type value of 0x8100 is found in bytes [14:13], with the priority field being the upper three bits of byte 15. This requires the following register settings:

* Frame Filter Value bytes [15:12] set to 0xE0008100
* Frame Filter Mask Value bytes [15:12] set to 0xE0FFFF00
* All other Frame Filter Mask Value bytes are set to 0x0

# DWC XGMAC

A diagram of a computer hardware system

Description automatically generated

Transmit Features  
■ Transmit memory size from 1 KB to 32 KB for single queue configurations and up to 256 KB for  
multiple queues configurations, with size configurable in powers of 2.  
Programmable size with granularity of 256 bytes per queue.

■ Optional module to calculate and insert IPv4 (L3) header checksum and TCP, UDP, or ICMP (L4)  
checksum in each queue.

■ The TX queues can be mapped into 8 Traffic Classes for scheduling and bandwidth allocation.  
■ Automatic retransmission of collision packets in half-duplex mode.  
■ Discard packets on late collision, excessive collision, excessive deferral, and under-run conditions with appropriate status.

■ Multiple (up to eight) traffic classes are supported, with the following scheduling algorithms. The number of traffic classes and transmit queues can be independently selected in such a way that the number of transmit queues is always greater than or equal to the number of traffic classes.  
1. Weighted Round Robin (WRR)  
2. Deficit Weighted Round Robin (DWRR) (when DCB enabled)  
3. Weighted Fair Queuing (WFQ) (when DCB enabled)  
4. Strict Priority (SP)

■ Optional packet level control for  
1. VLAN tag insertion or replacement  
2. Ethernet source address insertion  
3. Checksum insertion

1. One-step timestamp correction  
   5. Timestamp control  
   6. CRC and pad control

■ When audio-video bridging is enabled, supports Credit-based Shaper scheduling algorithms in  
configurations with multiple queues.

Receive Features  
■ Rx FIFO size from 4 KB to 32 KB for single queue configurations and up to 256 KB for multiple queues configurations, with size configurable in powers of 2. Programmable size with granularity of 256 bytes per queue.

■ Insertion of Rx Status vectors into the Rx queue after the EOP transfer (in Threshold mode) and before SOP (in Store-and-Forward mode) in XGMAC-MTL configuration. In store-and-forward mode, the status is available before the packet data. Therefore, the application can take the packet processing decision even before the packet is read from the queue. This enables multiple-frame storage in the Rx Queue without requiring additional memories blocks to store the Rx status of packets.

■ Option to filter all error packets on reception and not forward them to the application in the  
store-and-forward mode.

■ Option to forward the undersized good packets.

■ Automatic generation of Pause packet control to the MAC based on the Rx Queue fill level with programmable Flow control thresholds in each Rx queue.

■ Received packets are mapped into the multiple queues based on the Priority field of the VLAN tag or on Packet Type (DCB control packet, and so on).

■ Option to select Audio-Video Bridging feature on Rx side only (AVB Listener applications) by de-selecting this feature on Tx side.

DMAThe DMA block exchanges the data between the MTL block and the host memory. The host can use a set of registers (DMA CSR) to control the DMA operations. The DMA block supports the following features:

■ 64-bit, 128-bit data transfers  
■ Multi-channel Transmit and Receive engines (up to 16 Transmit and 16 Receive channels)  
■ Separate DMA channel in the Transmit path for each queue in MTL. Single or multiple DMA channels for any number of queues in MTL Receive path  
■ Descriptor structure to support the following:  
❑ Byte-aligned addressing for data buffer  
❑ Dual-buffer descriptor ring and support for 64-bit addressing in descriptor structure.   
■ Individual programmable burst size for Tx DMA and Rx DMA engines for optimal host bus utilization  
■ Local memory option for storing TSO headers and better performance  
■ Selectable number of Tx DMA channels with TCP Segmentation Offload (TSO) feature enabled. TSO engine to support TCP over IPv4/IPv6 segmentation  
■ Routing of received packets to the DMA channels based on RSS hash or the Ethernet DA or VLAN Priority in multi-channel DMA configurations  
■ Register programmable RX buffer sizes for each DMA channel  
■ Additional per-DMA channel interrupt pins for normal packet transfers  
■ Per-packet Transmit and Receive Interrupt control in descriptors  
■ Enhanced DMA architecture for better performance and throughput  
■ Supports pre-fetching and caching up to 32 descriptors for each DMA channel  
■ Descriptor fetches, Data transfers and Descriptor writes are de-linked and performed independently so as to reduce gaps between data transfers of each DMA  
■ TxDMA engines have capability to issue up to 16 outstanding requests for packet/burst transfer  
■ DMA descriptor writes can be “posted”, so that next descriptor can be processed immediately for next packet  
■ Descriptor memory cache can be either shared with TSO Header memory or can be a separate instance

Reference:

[ Document name – DesignWare Core XGMAC ]

The DMA supports up to 16 Tx and 16 Rx Descriptor lists (or DMA channels). The base address of each list is written to the respective Tx Descriptor List Address register and Rx Descriptor List Address register. The descriptor list is forward linked, and the next descriptor is always considered at a fixed offset to the current one.

The offset is controlled by the programming of the DSL field of DMA\_Ch(#i)\_Control register. The number of descriptors in the list is programmed in the respective Tx (or Rx) Descriptor Ring Length register.  
After the DMA processes the last descriptor in the list, it jumps back to the descriptor in the List Address register to create a descriptor ring.

## MAC Layer: -

The MAC layer complies fully with the IEEE 802.3-2018/industry standard, implementing the  
XGMII/GMII/MII/RGMII/RMII full duplex interface for communication with the physical coding  
sub-layer. The MAC layer communicates with the application side through the MAC Transmit Interface (MTI), MAC Receive Interface (MRI) and the MAC Control Interface (MCI).

## Multiple channels and queue support: -

A diagram of a computer hardware

Description automatically generated