cādence®

Gigabit Ethernet MAC (GEM)

Technical Data Sheet

Part Number: T-CS-ET-0005-100

Document Number: I-IPA01-0099-USR Rev 26

January 2012

Gigabit Ethernet MAC (GEM)

Gigabit Ethernet MAC (GEM)

MAC Features

- UNH tested
- Compatible with IEEE Standard 802.3
- 10, 100 and 1000 Mbps operation
- Statistics counter registers for RMON/MIB
- GMII/MII/TBI interfaces supported
- Support for SGMII
- Support for RGMII included by means of the Cadence T-CS-ET-0019-100 “Reduced Gigabit Media Independent Interface block”
- Integrated 1000BASE-X physical coding sub-layer (PCS) with auto-negotiation
- Either external FIFO interface, AMBA AHB or AMBA AXI bus master direct memory access (DMA) interface to external memory
- Support for up to eight priority queues on transmit and receive
- Support for 802.1AS and 1588 precision clock synchronization protocol
- Optional IEEE 1588 time stamp unit
- Support for 802.3az Energy Efficient Ethernet
- Full duplex flow control with recognition of incoming pause frames and hardware generation of transmitted pause frames
- Support for 802.1Qbb priority-based flow control
- Half duplex flow control by forcing collisions on incoming frames
- Receive and transmit IP, TCP and UDP checksum offload. Both IPv4 (with IP options) and IPv6 (with extension headers) packet types supported
- Automatic pad and cyclic redundancy check (CRC) generation on transmitted frames
- Frame extension and frame bursting at 1000 Mbps in half duplex mode
- Address checking logic for four or thirty two specific 48 bit addresses, four type Ids, promiscuous mode, external address checking, hash matching of unicast and multicast destination addresses and Wake-on-LAN
- MDIO interface for physical layer management
- Support for jumbo frames up to 10,240 bytes
- Programmable IPG stretch
- Configuration interface compatible with AMBA APB interface, version 2.0
- Interrupt generation to signal receive and transmit completion, errors and other events
- Support for 802.1Q VLAN tagging with recognition of incoming VLAN and priority tagged frames

Optional DMA Features

- AHB or AXI4 compliant master interface
- Selectable low latency FIFOs or packet buffer SRAM on transmit and receive DMA
- Support for full store and forward or flexible low latency partial store and forward configurations available
- Configurable for 32 bit, 64 bit and 128 bit data paths (128 bit AXI is not supported)
- Support for Priority Queue Management to stream data to and from up to 8 independent external traffic queues
    - Highly flexible and configurable screening algorithm based on VLAN,IP,TCP,UDP and other indexed packet fields to map receive frames to priority queues.
    - High priority traffic given precedence over lower priority traffic on TX path.
    - Ability for high priority transmit traffic to be internally reordered to reduce traffic latency.
- Programmable burst length and endianism
- Automatic self completing retries for half duplex operation
- Optional automatic discard of frames received with MAC related errors
- Optional support for manual or automatic flushing of frames received when external memory resource has been exhausted or AHB/AXI access errors occur
- Options to help maximize burst efficiency

Description

The Gigabit Ethernet MAC (GEM) module implements a 10/100/1000 Mbps Ethernet MAC compatible with the IEEE 802.3 standard. The GEM can operate in either half or full duplex at all three speeds. The network configuration register is used to select the speed, duplex mode and interface type (MII, GMII, RGMII, TBI or SGMII).

The GEM comprises five constituent components:

- GEM_MAC controlling transmit, receive, address checking and loopback
- GEM_REG_TOP providing control and status registers, statistics registers and synchronization logic
- GEM_PCS controlling the 8B/10B encoder, 8B/10B decoder, PCS transmit, PCS receive and PCS auto-negotiation
- GEM_DMA_TOP controlling DMA transmit, DMA receive and AHB/AXI arbitration logic
- GEM_TSU calculates the IEEE 1588 timer values

In system applications where no DMA operation is required, the GEM_DMA_TOP module can be removed using a compile option and an external FIFO interface used to incorporate the GEM into a SoC environment. Similarly, the GEM_PCS module may be removed using a compile option.

GEM_MAC

The MAC transmit block takes data from the external FIFO interface, adds preamble and, if necessary, pad and frame check sequence (FCS). Both half duplex and full duplex Ethernet modes of operation are supported. When operating in half duplex mode, the MAC transmit block generates data according to the carrier sense multiple access with collision detect (CSMA/CD) protocol. The start of transmission is deferred if carrier sense (`crs`) is active. If collision (`col`) becomes active during transmission, a jam sequence is asserted and the transmission is retried after a random back off. The `crs` and `col` signals have no effect in full duplex mode. When operating in gigabit mode half duplex, both carrier extension and frame bursting are performed in accordance with the IEEE 802.3 standard.

The MAC receive block checks for valid preamble, FCS, alignment and length, and presents received frames to the MAC address checking block and external FIFO interface. Software can configure the GEM to receive jumbo frames up to 10,240 bytes. It can optionally strip CRC from the received frame prior to transfer to the external FIFO interface.

The address checker recognizes four specific 48 bit addresses, can recognize four different type ID values, and contains a 64 bit hash register for matching multicast and unicast addresses as required. It can recognize the broadcast address of all ones, copy all frames and act on external address matching signals. The MAC can also reject all frames that are not VLAN tagged and recognise Wake on LAN events. Address comparison against individual bits of Specific Address Register 1 can be masked by means of the Specific Address Mask Register.

The MAC receive block supports offloading of IP, TCP and UDP checksum calculations (both IPv4 and IPv6 packet types supported), and can automatically discard bad checksum frames.

GEM_PCS

A PCS sub-layer is incorporated for 1000 Mbps operation and provides a TBI to the PMA layer. A PCS transmit module and 8B/10B encoder form the transmit path, and a PCS receiver and two 8B/10B decoders form the receive path, which is expanded into 16 bits to simplify clocking. Auto-negotiation is also provided for network configuration.

The PCS can be configured through the network configuration register to support an SGMII interface between the MAC and an external PHY. For a complete SGMII solution a SerDes hard macro is also required.

GEM_REG_TOP

Control registers drive the management data input/output (MDIO) interface, set-up DMA activity, start frame transmission and select modes of operation such as full duplex, half duplex and 10/100/1000 Mbps operation. The register interface is compatible with the AMBA APB bus standard. Where possible, register set compatibility has been maintained with the Ethernet MAC 10/100 (Enhanced) and the 10 Gigabit Ethernet MAC (XGM) from Cadence.

The statistics register block contains registers for counting various types of event associated with transmit and receive operation. These registers, along with the status words stored in the receive buffer list, enable software to generate network management statistics compatible with IEEE 802.3 clause 30.

GEM_DMA_TOP

The DMA may be configured in either a very low latency buffering mode using internal transmit and receive FIFOs or a packet buffering mode using external memories.

Data path bus widths of 32 bit, 64 bit and 128 bit are supported at all data rates (128 bit AXI is not supported), although system designers should carefully consider the bandwidth requirements of the external data path being used — either external FIFO interface or AMBA AHB/AXI interface. The DMA block connects to external memory through its AMBA AHB or AXI bus interface. The DMA loads the transmit buffer and empties the receive buffer using bus master operations. Receive data is not sent to memory until the address checking logic has determined that the frame should be copied.

Receive or transmit frames are stored in one or more DMA buffers. The receive buffer size is programmable between 64 bytes and 16 Kbytes. Transmit buffers range in length between 1 and 2047 bytes, and up to 128 buffers are permitted per frame. The DMA block manages the transmit and receive frame buffer queues.

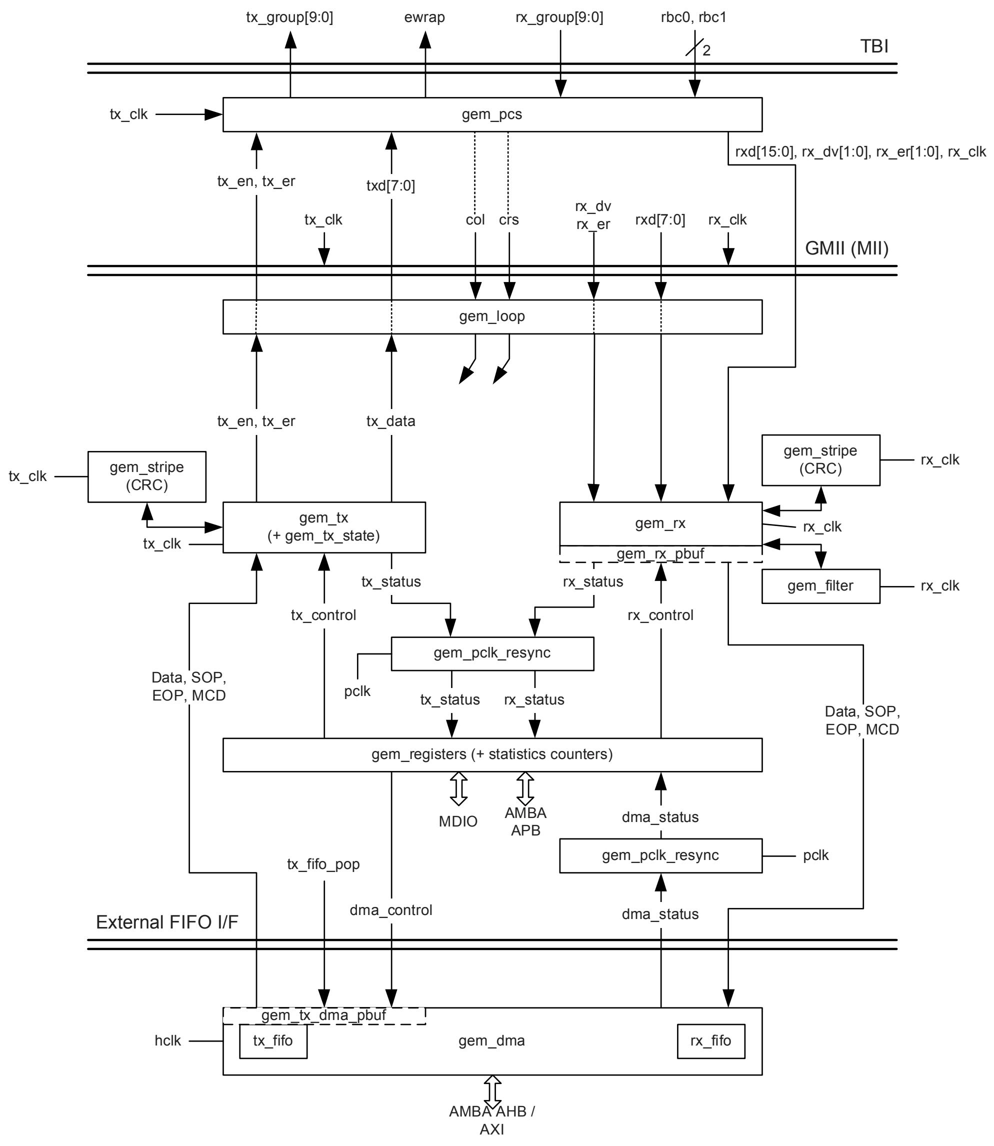
For chip architecture where DMA is not required, the GEM is supplied with an external transmit and receive external FIFO interface, thus simplifying design complexity and reducing gate count.

Note the DMA block does not use AMBA AHB split and retry operations.

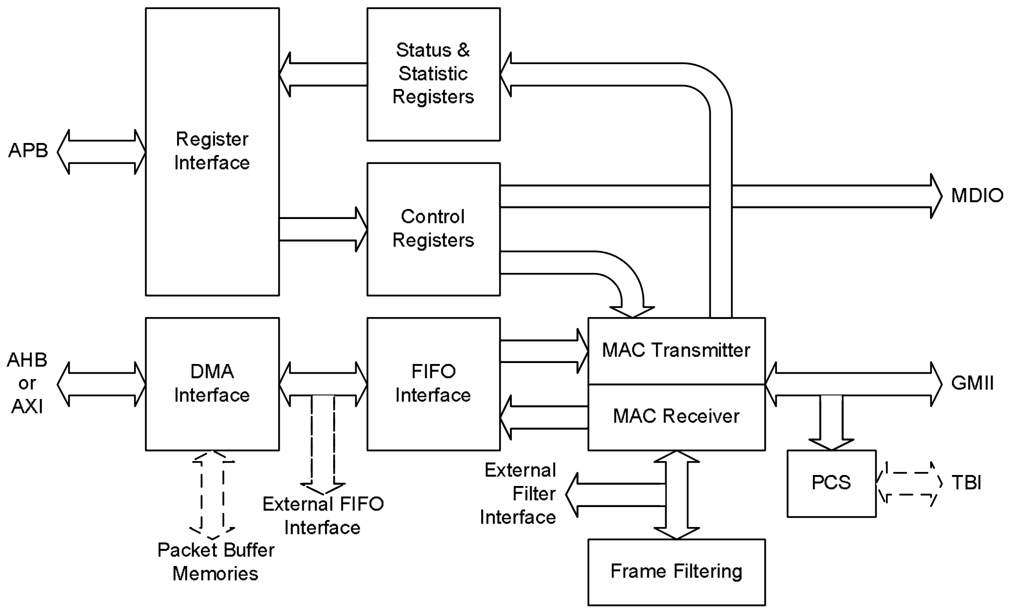
GEM_TSU

The 1588 time stamp unit (TSU) is a timer implemented as a 62 bit register. The 32 upper bits count seconds and the 30 lower bits count nanoseconds. The 30 lower bits roll over when they have counted to one second. The timer increments by a programmable number of nanoseconds with each PCLK period and can be adjusted (incremented or decremented) through APB register accesses.

Block Diagram



Signal Interfaces



The GEM includes the following signal interfaces:

- GMII, MII, and TBI to an external PHY
- MDIO interface for external PHY management
- AMBA Advanced Peripheral Bus (APB) slave interface for accessing the GEM's registers
- AMBA Advanced High Speed Bus (AHB or AXI4) master interface for memory access
- An optional FIFO interface in applications where DMA functionality is not required
- An optional packet buffer memory interface for the DMA

The AMBA interfaces fully conform to the ARM AMBA Revision 2.0 specification.

Ethernet Interface (GMII/MII)

Signal Name	I/O	Description
`col`	I	Collision detect from the PHY.
`crs`	I	Carrier sense from the PHY.
`tx_er`	O	Transmit error signal to the PHY. Asserted if the DMA block fails to fetch data from memory during frame transmission.
`txd[7:0]`	O	Transmit data to the PHY. 10/100 mode: `txd[3:0]` used, `txd[7:4]` tied to logic 0. 1000 mode: `txd[7:0]` used.
`tx_en`	O	Transmit enable to the PHY.
`tx_clk`	I	Transmit clock from the system clock generator.
`rxd[7:0]`	I	Receive data from the PHY: 10/100 mode: `rxd[3:0]` used, `rxd[7:4]` not used. 1000 mode: `rxd[7:0]` used.

`rx_er`	I	Receive error signal from the PHY.
`rx_clk`	I	Receive clock from the system clock generator.
`rx_dv`	I	Receive data valid signal from the PHY.
`mdc`	O	Management data clock to pin.
`mdio_in`	I	Management data input from MDIO pin.
`mdio_out`	O	Management data output to MDIO pin.
`mdio_en`	O	Management data output enable to MDIO pin. At the top-level the three mdio pins are all used to drive a single tri-sate pin. Alternatively MDIO may be implemented as a pull-down. In this case the enable to the pull-down should be driven with ~`mdio_out & mdio_en`.
`ext_interrupt_in`	I	External interrupt input pin.

Control/Status Interface

Signal Name	I/O	Description
`loopback`	O	Selects external loopback — not used by most PHYs. This signal does not need to be bonded out, but is used by the testbench.
`loopback_local`	O	Internal loopback indication to the system clock generator.
`half_duplex`	O	Selects half duplex — this signal does not need to be bonded out, but is used by the testbench.
`speed_mode[2:0]`	O	Indicates speed and external interface that the GEM is currently configured to use to the system clock generator: 000 - 10 Mbps Ethernet operation using MII interface 001 - 100 Mbps operation using MII interface 01x - 1000 Mbps operation using GMII interface 100 - 10 Mbps operation using SGMII interface 101 - 100 Mbps operation using SGMII interface 11x - 1000 Mbps operation using TBI or SGMII interface
`tx_pause`	I	Toggle input for pause frame transmission. When `tx_pfc_sel` is LOW, a classic 802.3 pause frame will be transitted. The pause quantum value to be transmitted depends on the state of the `tx_pause_zero pin.` When `tx_pfc_sel` is HIGH, an 802.1Qbb, or PFC priority based pause frame will be transmitted. The pause quantum value to be transmitted depends on the state of the tx_pfc_priority bus. Tie this input low if hardware initiated flow control is not being used. This asynchronous input is synchronized into the tx_clk domain.
`tx_pfc_sel`	I	Indicates whether the pause frame to be transmitted should be classic 802.3 (when set to logic 0) or PFC priority based pause frame (when set to logic 1). This signal should be valid when tx_pause toggles and

		remain valid for at least two tx_clk cycles afterwards. Tie this low if hardware initiated flow control is not being used.
`tx_pause_zero`	I	Indicates whether the classic 802.3 pause frame to be transmitted is to have zero pause quantum (when set to logic 1) or the value of the transmit pause quantum register (when set to logic 0). This signal should be valid when tx_pause toggles and remain valid for at least two tx_clk cycles afterwards. Tie this low if hardware initiated flow control is not being used.
`tx_pfc_pause[7:0]`	I	Indicates the value to be used in the 8bit priority enable vector of the PFC priority based pause frame. This signal should be valid when tx_pause toggles and remain valid for at least two tx_clk cycles afterwards. Tie this low if hardware initiated flow control is not being used.
`tx_pfc_pause_zero [7:0]`	I	Indicates whether each priority entry in the PFC priority based pause frame is to have zero pause quantum (when set to logic 1) or the value of the transmit pause quantum register (when set to logic 0). This signal should be valid when tx_pause toggles and remain valid for at least two tx_clk cycles afterwards. Tie this low if hardware initiated flow control is not being used.
`trigger_dma_tx_start`	I	Toggle input for starting transmit DMA. Only present when using the DMA configured for internal packet buffering. This signal is used as a hardware alternative to setting the TX START bit of the network control register. If the user only wishes to use software to initiate transmission, then this input can be tied low. This asynchronous input is synchronized into the hclk domain
`rx_pfc_paused[7:0]`	O	Each bit corresponds to a priority indicated within the PFC priority based pause frame. Each bit is set when an PFC priority based pause frame has been received, and the associated priority pause time quantum is non-zero. Each bit is cleared when the associated pause time identified by the received pause time quantum has elapsed.
`pfc_negotiate`	O	Identifies that PFC priority based pause flow control has been negotiated. 0 - No PFC priority based pause frames have yet been received, flow control is being handled using classic 802.3 pause frames. 1 – At least one PFC priority based pause frames has been received. All subsequent 802.3 pause frames will be dropped.
`rx_databuf_wr_q0`	O	When priority queueing is disabled, this output is toggled on a DMA write to the first word of each DMA data buffer. When priority queueing is enabled, this

		output is toggled on a DMA write to the first word of each DMA data buffer associated with queue 0.
`rx_databuf_wr_q1`	O	Only present when priority queueing support is enabled. This output is toggled on a DMA write to the first word of each DMA data buffer associated with queue 1.
`rx_databuf_wr_q2`	O	Only present when priority queueing support is enabled. This output is toggled on a DMA write to the first word of each DMA data buffer associated with queue 2.
`halfduplex_flow_co ntrol_en`	I	Asynchronous Input. This input is ignored when the GEM is configured for full duplex or gigabit modes. Setting this bit high will enable the half duplex flow control mechanism. The transmit block will transmit 64 bits of data, whenever it sees an incoming frame in order to force a collision.
`dma_bus_width[1:0]`	O	Indicates width of DMA or external FIFO data bus as follows— this signal does not need to be bonded out, but is used by the testbench: 00 - 32 bit data bus 01 - 64 bit data bus 10 or 11 - 128 bit data bus.

External Filter Interface

Signal Name	I/O	Description

`ext_match1`
`ext_match2`
`ext_match3`
`ext_match4`
	I	Optional external match 1, 2, 3 and 4 — any one of these input signals from the external filter logic can be asserted indicating a match was found. Filter matches can take place on either `ext_sa`, `ext_da`, `ext_type` or `ext_vid`.

`ext_sa[47:0]`	O	Source address for comparison, extracted from packet being received.
`ext_sa_stb`	O	Validates source address.
`ext_da[47:0]`	O	Destination address for comparison, extracted from packet being received.
`ext_da_stb`	O	Validates destination address.
`ext_type[15:0]`	O	Length/type field for comparison, extracted from packet being received.
`ext_type_stb`	O	Validates length/type field.
`ext_vlan_tag1[31: 0]`	O	VLAN identifier and VLAN ID field for comparison, extracted from packet being received. For packets incorporating 2 VLAN tags (stacked VLAN), this represents the first VLAN tag
`ext_vlan_tag1_stb`	O	Validates VLAN ID field.

`ext_vlan_tag2[31:0]`	O	VLAN identifier and VLAN ID field for comparison, extracted from packet being received. For packets incorporating 2 VLAN tags (stacked VLAN), this represents the second VLAN tag. These bits are only valid for packets incorporating the stacked VLAN processing feature.
`ext_vlan_tag2_stb`	O	Validates VLAN ID field.
`ext_ip_sa[127:0]`	O	IP source address for comparison, extracted from the packet being received. For IPv4 packets, only bits [31:0] are valid. For Ipv6 packets, bits [127:0] are vaid.
`ext_ip_sa_stb`	O	Validates IP source address field.
`ext_ip_da[127:0]`	O	IP destination address for comparison, extracted from the packet being received. For IPv4 packets, only bits [31:0] are valid. For Ipv6 packets, bits [127:0] are vaid.
`ext_ip_da_stb`	O	Validates IP destination address field.
`wol`	O	Asserted for 64 `rx_clk` cycles if a magic packet, ARP request, specific address 1 or multicast hash event is decoded and is enabled through the Wake on LAN register. Only bond this pin out if Wake on LAN functionality is required.

IEEE 1588 PTP frame recognition and Time Stamp Unit

Signal Name	I/O	Description
`sof_tx`	O	Asserted high synchronous to tx_clk when the SFD is detected on a transmit frame, deasserted at end of frame
`sync_frame_tx`	O	Asserted high synchronous to tx_clk if PTP sync frame is detected on transmit.
`delay_req_tx`	O	Asserted high synchronous to tx_clk if PTP delay request frame is detected on transmit.
`pdelay_req_tx`	O	Asserted high synchronous to tx_clk if PTP peer delay request frame is detected on transmit.
`pdelay_resp_tx`	O	Asserted high synchronous to tx_clk if PTP peer delay response frame is detected on transmit.
`sof_rx`	O	Asserted high synchronous to rx_clk when the SFD is detected on a receive frame
`sync_frame_rx`	O	Asserted high synchronous to rx_clk if PTP sync frame is detected on receive.
`delay_req_rx`	O	Asserted high synchronous to rx_clk if PTP delay request frame is detected on receive.
`pdelay_req_rx`	O	Asserted high synchronous to rx_clk if PTP peer delay request frame is detected on receive.
`pdelay_resp_rx`	O	Asserted high synchronous to rx_clk if PTP peer delay response frame is detected on receive.

`tsu_clk`	I	Alternative clock source for the time stamp unit. If gem_tsu_clk is defined in the gem_defs.v file then the TSU is clocked by tsu_clk rather than pclk. Must run slower than pclk.
`gem_tsu_ms`	I	TSU master/slave. Used with gem_tsu_inc_ctrl to control incrementing of the TSU and loading the sync strobe register.
`gem_tsu_inc_ctrl[ 1:0]`	I	Used to control incrementing of the TSU and synchronous to tsu_clk or pclk. Drive high when not being used.
`tsu_timer_cnt[61: 0]`	O	TSU timer count value, synchronized to tsu_clk or pclk. Upper 32 bits are seconds value and lower 30 bits are nanoseconds.
`tsu_timer_cmp_val`	O	TSU timer comparison valid, synchronized to tsu_clk or pclk. Asserted high when upper 54 bits of TSU timer count value are equal to programmed comparison value.

Ethernet Interface (TBI)

Signal Name	I/O	Description
`tx_group[9:0]`	O	8B/10B encoded transmit data to PHY transceiver, synchronized to `tx_clk`.
`gtx_clk`	I	125 MHz PCS transmit clock. In non-SGMII application this will have the same source as tx_clk. In SGMII applications gtx_clk is supplied to the PCS at 125MHZ but divided down to drive tx_clk at 10 and 100Mb/s speeds
`rx_group[9:0]`	I	8B/10B encoded receive data from PHY transceiver.
`rbc0` `rbc1`	I	Receive clocks from PHY transceiver, `rbc1` will be 180 degrees phase shifted with respect to `rbc0`.
`ewrap`	O	Control for loopback operation in the PHY transceiver.
`en_cdet`	O	Control for comma alignment in the PHY transceiver (see IEEE 802.3 subclause 36.3.2.4). Leave unconnected if an on-chip SerDes is being used.
`signal_detect`	I	Valid signal detected from PMA. This signal must be driven high for PCS synchronization to occur. Users are advised to check that this signal is driven high by their SerDes or to provide a control signal to over-ride the signal from the SerDes.

Signals for optional on chip SerDes

These signals will only be present if the GEM has been configured to support comma alignment.

Signal Name	I/O	Description
`pma_rx_clk`	I	125MHz recovered clock from the SerDes. This clock is used by the comma alignment module.
`n_prxreset`	I	Active low pma_rx_`clk` domain reset. This signal should be asserted low asynchronously, and deasserted high synchronously with pma_rx_`clk`.
`rbc_align`	O	Used in clock control block to align the rbc clocks correctly to even and odd code groups. This signal is asserted for a single pma_rx_clk cycle when a comma is detected.

Interrupt Controller Interface

Signal Name	I/O	Description
`ethernet_int`	O	Ethernet interrupt signal synchronous to `pclk`.

External Transmit FIFO Interface

Signal Name	I/O	Description
`tx_r_data_rdy`	I	When set to logic 1, indicates enough data is present in the external FIFO for Ethernet frame transmission to commence on the current packet.
`tx_r_rd`	O	Single `tx_clk` clock cycle wide active high output requesting a word (either 32-bits, 64-bits or 128-bits) of information from the external FIFO interface. Synchronous to the `tx_clk` clock domain.
`tx_r_valid`	I	Single `tx_clk` clock cycle wide active high input indicating requested FIFO data is now valid. Validates the following inputs: `tx_r_data[127:0]`, `tx_r_sop`, `tx_r_eop`, `tx_r_err` and `tx_r_mod[3:0]`.
`tx_r_data[127:0]`	I	FIFO data for transmission, either 32-bit, 64-bit or 128-bit depending on `dma_bus_width[2:0]` which is configured through the register interface. This input is only valid whilst `tx_r_valid` is high.
`tx_r_sop`	I	Start of packet, indicates the word received from the external FIFO interface is the first in a packet. This input is only valid whilst `tx_r_valid` is high.
`tx_r_eop`	I	End of packet, indicates the word received from the external FIFO interface is the last in a packet. This input is only valid whilst `tx_r_valid` is high.

`tx_r_mod[3:0]`	I	Read module, indicates how many bytes are valid on `tx_r_data` during the last transfer of the packet: 0000- all bytes are valid 0001 - `tx_r_data[7:0]` is valid 0010 - `tx_r_data[15:0]` is valid 0011 - `tx_r_data[23:0]` is valid, and so on until 1111 - `tx_r_data[119:0]` is valid. This input is only valid when both `tx_r_eop` and `tx_r_valid` are high.
`tx_r_err`	I	Error, active high input indicating the current packet contains an error. This signal is only valid whilst `tx_r_valid` is high and may be set at any time during the packet transfer.
`tx_r_underflow`	I	FIFO under flow, indicating the transmit FIFO was empty when a read was attempted. This signal is only valid when the GEM has attempted a read by asserting tx_r_rd and the `tx_r_valid` signal has not yet been indicated. tx_r_flushed should be asserted following this event to indicate to the GEM when it is safe to resume reading.
`tx_r_flushed`	I	This signal must be driven high and then low after a major error event to indicate to GEM that the external FIFO has been flushed. This will enable the GEM to resume reading data. Events that require this to be set are indicated by any bit of `tx_r_status` being asserted
`tx_r_control`	I	`tx_no_crc`, set active high at start of packet to indicate current frame is to be transmitted without `crc` being appended. This input is only valid whilst both `tx_r_valid` and `tx_r_sop` are high.

Note: The transmit FIFO interface inputs must be pre-synchronized to the `tx_clk` clock domain.

External Transmit FIFO Interface

Signal Name	I/O	Description
`dma_tx_end_tog`	O	Toggled to indicate that a frame has been completed and status is now valid on the `tx_r_status` output. Note that this signal is not activated when a frame is being retried due to a collision.
`dma_tx_status_tog`	I	This signal must be toggled each time either `dma_tx_end_tog` or `collision_occured` are activated, to indicate that the status has been acknowledged. This asynchronous input is synchronized into the tx_clk domain.

`tx_r_status[3:0]`	O	`[3]: fifo_underrun` - status output indicating that the MAC transmitter has under run due to one of the following conditions. Data under run indicated by `tx_r_underflow` input from the external FIFO interface, status write back error due to status write back not completing when another status write back is attempted or a `tx_r_err` was driven on the external FIFO interface during the last frame transfer. Reset once dma_tx_status_tog changes logic state. `[2]: collision_occured` – HALF DUPLEX status output indicating that the frame in progress has suffered a collision and that re-transmission of the frame should take place. Set when the collision occurs and reset once `dma_tx_status_tog` changes logic state. Note that `dma_tx_end_tog` is not activated. `[1]: late_coll_occured` – HALF DUPLEX status output indicating that the frame in progress suffered a late collision and was aborted. Valid when `dma_tx_end_tog` changes logic state. Cleared once `dma_tx_status_to`g changes logic state. `[0]: too_many_retries` – HALF DUPLEX status output indicating that the frame in progress experienced excess collisions and was aborted. Valid when `dma_tx_end_tog` changes logic state. Cleared once `dma_tx_status_tog` changes logic state.

External Receive FIFO Interface

Note: Receive FIFO interface outputs are synchronized to the `rx_clk` clock domain.

Signal Name	I/O	Description
`rx_w_wr`	O	Single `rx_clk` clock cycle wide active high output indicating a write to the external FIFO interface.
`rx_w_data[127:0]`	O	Received data for output to the external FIFO interface, either 32-bit, 64-bit or 128-bit depending on `dma_bus_width[2:0]` which is configured through the register interface. This output is only valid when `rx_w_wr` is high.
`rx_w_sop`	O	Start of packet, indicates the word output to the external FIFO interface is the first in a packet. This output is only valid when `rx_w_wr` is high.
`rx_w_eop`	O	End of packet, indicates the word output to the external FIFO interface is the last in a packet. This output is only valid when `rx_w_wr` is high. (Note this signal will not be asserted if the receive path is disabled during frame reception however `rx_w_flush` will be asserted.)

`rx_w_status[44:0]`	O	Status signals, valid with `rx_w_eop (with the exception of bits 20:15, which are valid on both rx_w_sop and rx_w_eop)`, definitions as follows: `[44]` - rx_w_code_error indicates a code error. `[43]` - rx_w_too_long indicates the frame was too long. `[42]` - rx_w_too_short indicates the frame was too short. `[41]` - rx_w_crc_error indicates the frame had a bad crc. `[40]` - rx_w_length_error indicates the length field was checked and was incorrect. `[39]` - `rx_w_snap_match` indicates the frame was SNAP encoded and had either no VLAN tag or a VLAN tag with the CFI bit not set. `[38]` - rx_w_checksumu indicates the UDP checksum was checked and was correct. `[37]` - `rx_w_checksumt` indicates the TCP checksum was checked and was correct. `[36]` - `rx_w_checksumi` indicates the IP checksum was checked and was correct. `[35]` - `rx_w_type_match4`, indicates the received frame was matched on type ID register 4. `[34]` - `rx_w_type_match3`, indicates the received frame was matched on type ID register 3. `[33]` - `rx_w_type_match2`, indicates the received frame was matched on type ID register 2. `[32]` - `rx_w_type_match1`, indicates the received frame was matched on type ID register 1. `[31]` - `rx_w_add_match4`, indicates the received frame was matched on specific address register 4. `[30]` - `rx_w_add_match3`, indicates the received frame was matched on specific address register 3. `[29]` - `rx_w_add_match2`, indicates the received frame was matched on specific address register 2. `[28]` - `rx_w_add_match1`, indicates the received frame was matched on specific address register 1. `[27]` - `rx_w_ext_match4`, indicates the received frame was matched externally by the ext_match4 input pin. `[26]` - `rx_w_ext_match3`, indicates the received frame was matched externally by the ext_match3 input pin. `[25]` - `rx_w_ext_match2`, indicates the received frame was matched externally by the `ext_match2` input pin. `[24]` - `rx_w_ext_match1`, indicates the received frame was matched externally by the ext_match1 input pin. `[23]` - `rx_w_uni_hash_match`, indicates the received frame was matched as a unicast hash frame. `[22]` - `rx_w_mult_hash_match`, indicates the received frame was matched as a multicast hash frame. `[21]` - `rx_w_broadcast_frame`, indicates the

		received frame is a broadcast frame. [20] - rx_w_prty_tagged, indicates a VLAN priority tag detected with received packet. [19:16] - rx_w_tci[3:0], indicates VLAN priority of received packet. [15] - rx_w_vlan_tagged, indicates VLAN tag detected with received packet. [14] - rx_w_bad_frame, indicates received packet is bad. [13:0] - rx_w_frame_length, indicates number of bytes in received packet.
add_match_vec[32:5]	O	If the GEM is configured to support 32 specific address match registers then these outputs indicate matches for specific address registers 5 to 32. Matches for registers 1 to 4 are indicated in the rx_w_status vector described above.
rx_w_mod[3:0]	O	Write module, indicates how many bytes are valid on rx_w_data during the last transfer of the packet: 0000- all bytes are valid 0001 - rx_r_data[7:0] is valid 0010 - rx_r_data[15:0] is valid 0011 - rx_r_data[23:0] is valid, and so on until 1111 - rx_r_data[119:0] is valid This output is only valid when coincident with both rx_w_wr and rx_w_eop being high.
rx_w_err	O	Error, active high output indicating the current packet contains an error. This signal is only valid when rx_w_wr is active high and may be set at any time during packet transfer.
rx_w_overflow	I	FIFO overflow, indicates to the MAC that the external RX FIFO has overflowed. The MAC uses this signal for status reporting at the end of frame.
rx_w_flush	O	FIFO flush, active high output indicating that the external RX FIFO should be cleared. This signal is set when the receive path is disabled.

AMBA (APB) Interface

Signal Name	I/O	Description
n_preset	I	Active low AMBA reset (nPRESET). This signal must be asserted low asynchronously, and deasserted high synchronously with pclk. Resets all APB registers and the pclk_syncs block.
pclk	I	Peripheral bus clock (PCLK).

`psel`	I	Peripheral select (PSEL). Active high select signal to indicate that a valid access is being made to one of the GEM's registers.
`penable`	I	Peripheral enable (PENABLE). This indicates the second clock cycle of an access and indicates that the write data may be strobed into a register on the next rising edge of `pclk`, or that the read data is expected to be valid at the next rising edge of `pclk`.
`pwrite`	I	Peripheral write strobe (PWRITE). This indicates that a write access is taking place (if `psel` is active).
`paddr[9:2]`	I	Address bus from selected master (PADDR). Indicates which register is being accessed. This address is word-aligned within the GEM and therefore should be connected to bits `[09:02]` of the APB address bus.
`pwdata[31:0]`	I	Write data. Data to be written into the addressed register.
`prdata[31:0]`	O	Read data. Data read from the addressed register. For APB Rev 2.0, it is driven to logic 0 when not addressed.
`perr`	O	Indicates an APB access to an invalid address. Should be sampled when `penable` is high. This signal is optional and not a standard AMBA signal.

AMBA (AHB) Interface

Signal Name	I/O	Description
`n_hreset`	I	Active low asynchronous reset for the DMA. This signal must be asserted low asynchronously and deasserted high synchronously with `hclk`.
`hclk`	I	AHB bus clock.
`hready`	I	Slave device drives this low to extend data phase.
`hresp[1:0]`	I	Slave response, 00 for OK, any other response is not OK and triggers an interrupt. AMBA split and retry operations are not supported.
`hgrant`	I	AHB bus grant.
`haddr[31:0]`	O	AHB address.
`htrans[1:0]`	O	AHB transfer type as follows: 00 - Idle 01 - Busy (not used by the GEM) 10 - Non-sequential 11 - Sequential.
`hwrite`	O	High for AHB write, low for AHB read.
`hrdata[127:0]`	I	Data read from memory, configurable for 32 bit, 64 bit or 128 bit bus widths. Unused bits should be tied to logic 0 or logic 1.

`hsize[2:0]`	O	Transfer size, configured according to AMBA AHB data bus width as follows: 010 - 32 bit transfers 011 - 64 bit transfers 100 - 128 bit transfers.
`hburst[2:0]`	O	Burst type, 000 for single transfer, 001 for incrementing burst and 011 for four beat incrementing burst.
`hprot[3:0]`	O	Protection type. Parameterizable via `` `gem_hprot_value ``
`hwdata[127:0]`	O	Data written to memory, configurable for 32 bit, 64 bit or 128 bit bus widths.
`hbusreq`	O	AHB bus request.
`hlock`	O	Always asserted with `hbusreq` to lock the grant. This signal may be left unconnected if required (refer to user guide section 2)

AMBA (AXI4) Interface

Signal Name	I/O	Description
`n_areset`	I	Active low asynchronous reset for the DMA. This signal must be asserted low asynchronously and deasserted high synchronously with `aclk`.
`aclk`	I	AXI bus clock.
`awaddr [31:0]`	O	AXI write channel address.
`awlen[7:0]`	O	AXI write channel burst length. Maximum value is 15. Bits 7:4 are tied low.
`awsize[2:0]`	O	Transfer size, configured according to AXI data bus width as follows: 010 - 32 bit transfers 011 - 64 bit transfers 100 - 128 bit transfers.
`awburst[1:0]`	O	All bursts generated by the AXI DMA are incrementing, so this output is tied to 2'b01.
`awvalid`	O	AXI write request.
`awready`	I	AXI write request accepted by slave.
`awid[3:0]`	O	Not currently used by the GEM DMA. This output is tied low.
`awcache[3:0]`	O	Cache support. Parameterizable via ` `` `gem_axi_cache_value ``
`awlock[1:0]`	O	Not currently used by the GEM DMA. This output is tied low.
`awprot[2:0]`	O	Protection unit support. Parameterizable via ` `` `gem_axi_prot_value ``

`wdata[63:0]`	O	Data write to memory, configurable for 32 bit,or 64 bit bus widths. Unused bits are tied to logic 0.
`wlast`	I	Identifies the last data of a burst.
`wvalid`	O	AXI write data valid.
`wready`	I	AXI write data accepted by slave.
`bresp[1:0]`	I	Slave response, 00 for OK, any other response is not OK and triggers an interrupt.
`bid[3:0]`	I	Not currently used by the GEM DMA.
`bvalid`	I	AXI write response bus valid.
`bready`	O	AXI write response accepted.
`araddr[31:0]`	O	AXI read channel address.
`arlen[7:0]`	O	AXI read channel burst length. Maximum value is 15. Bits 7:4 are tied low.
`arsize[2:0]`	O	Transfer size, configured according to AXI data bus width as follows: 010 - 32 bit transfers 011 - 64 bit transfers 100 - 128 bit transfers.
`arburst[1:0]`	O	All bursts generated by the AXI DMA are incrementing, so this output is tied to 2'b01.
`arvalid`	O	AXI read request.
`arready`	I	AXI read request accepted by slave.
`arid[3:0]`	O	Not currently used by the GEM DMA. This output is tied low.
`arcache[3:0]`	O	Cache support. Parameterizable via \` \``gem_axi_cache_value`
`arlock[1:0]`	O	Not currently used by the GEM DMA. This output is tied low.
`arprot[2:0]`	O	Protection unit support. Parameterizable via \` \``gem_axi_prot_value`
`rdata[127:0]`	I	Data read from memory, configurable for 32 bit, 64 bit or 128 bit bus widths. Unused bits should be tied to logic 0 or logic 1.
`rlast`	I	Identifies the last data of a read burst.
`rresp[1:0]`	I	Slave response, 00 for OK, any other response is not OK and triggers an interrupt.

`rid[3:0]`	I	Not currently used by the GEM DMA.
`rvalid`	I	AXI read data valid
`rready`	O	AXI read data accepted

Transmit Packet Buffer Memory Interface

Port A of the transmit packet memory should be clocked on `hclk/aclk` and port B should be clocked on `tx_clk`.

Signal Name	I/O	Description
`txdpram_ena`	O	Transmit packet buffer memory Port A chip select
`txdpram_wea`	O	Transmit packet buffer memory Port A write enable (always high)
`txdpram_addra[N:0]`	O	Transmit packet buffer memory Port A word address bus. Address width configurable.
`txdpram_dia[63:0]`	O	Transmit packet buffer memory Port A write data bus. Either 32-bit or 64-bit depending on `dma_bus_width[2:0]`
`txdpram_doa[63:0]`	I	Transmit packet buffer memory Port A read data bus (not used by design – now removed from top level)
`txdpram_enb`	O	Transmit packet buffer memory Port B chip select
`txdpram_web`	O	Transmit packet buffer memory Port B write enable (always low)
`txdpram_addrb[N:0]`	O	Transmit packet buffer memory Port B word address bus. Address width configurable.
`txdpram_dib[63:0]`	O	Transmit packet buffer memory Port B write data bus (not used by design – now removed from top level)
`txdpram_dob[63:0]`	I	Transmit packet buffer memory Port B read data bus. Either 32-bit or 64-bit depending on `dma_bus_width[2:0]`

Receive Packet Buffer Memory Interface

Port A of the receive packet memory should be clocked on `rx_clk` and port B should be clocked on `hclk/aclk`.

Signal Name	I/O	Description
`rxdpram_ena`	O	Receive packet buffer memory Port A chip select
`rxdpram_wea`	O	Receive packet buffer memory Port A write enable (always high)
`rxdpram_addra[N:0]`	O	Receive packet buffer memory Port A word address bus. Address width configurable.

`rxdpram_dia[63:0]`	O	Receive packet buffer memory Port A write data bus. Either 32-bit or 64-bit depending on `dma_bus_width[2:0]`
`rxdpram_doa[63:0]`	I	Receive packet buffer memory Port A read data bus (not used by design – now removed from top level)
`rxdpram_enb`	O	Receive packet buffer memory Port B chip select
`rxdpram_web`	O	Receive packet buffer memory Port B write enable (always low)
`rxdpram_addrb[N:0]`	O	Receive packet buffer memory Port B word address bus. Address width configurable.
`rxdpram_dib[63:0]`	O	Receive packet buffer memory Port B write data bus (not used by design – now removed from top level)
`rxdpram_dob[63:0]`	I	Receive packet buffer memory Port B read data bus. Either 32-bit or 64-bit depending on `dma_bus_width[2:0]`

System Interface

Signal Name	I/O	Description
`scan_test_mode`	I	Asserted when the chip is in test mode (not connected in Soft IP).
`scan_en`	I	Selects scan mode (not connected in Soft IP).
`scan_in[8:0]`	I	Scan inputs to each chain.
`scan_out[8:0]`	O	Scan outputs from each chain.
`n_txreset`	I	Active low `tx_clk` domain reset. This signal should be asserted low asynchronously, and deasserted high synchronously with `tx_clk`.
`n_rxreset`	I	Active low `rx_clk` domain reset. This signal should be asserted low asynchronously, and deasserted high synchronously with `rx_clk`.
`n_gtxreset`	I	Active low `gtx_clk` domain reset. This signal should be asserted low asynchronously, and deasserted high synchronously with `gtx_clk`.
`n_rbc0reset`	I	Active low `rbc0` domain reset. This signal should be asserted low asynchronously, and deasserted high synchronously with `rbc0`.
`n_rbc1reset`	I	Active low `rbc1` domain reset. This signal should be asserted low asynchronously, and deasserted high synchronously with `rbc1`.
`n_ntxreset`	I	Active low `n_tx_clk` domain reset. This signal should be asserted low asynchronously, and deasserted high synchronously with `n_tx_clk`.
`n_tx_clk`	I	Inverted `tx_clk` used to buffer `tx_clk` domain signals to be fed into `rx_clk` domain within the loopback module.
`n_tsureset`	I	Active low `tsu_clk` domain reset. This signal should be asserted low asynchronously, and deasserted high synchronously with `tsu_clk`.

Timing Requirements

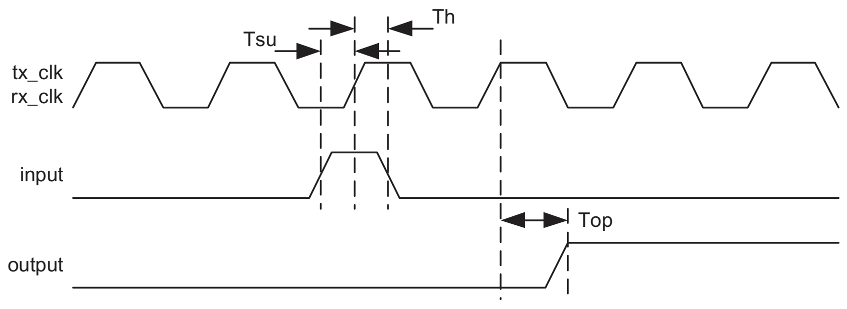
For all internal interfaces (including AHB, AXI, APB, FIFO), all inputs must be setup a minimum of 40% of the clock period before the rising edge of the related clock and all outputs will be valid at a maximum of 40% of the clock period after the rising edge of the related clock. These characteristics have been used as the default values for the trial synthesis of the module using a typical 90nm technology.

The tables below show the external interface timing requirements. These have been achieved using a typical 90nm technology, and are fully compatible with IEEE 802.3 requirements.

The absolute values in the tables below have been achieved with a typical 0.18 µm technology, and are fully compatible with IEEE 802.3 requirements.

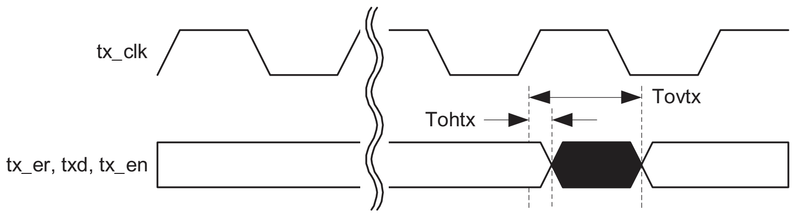
FIFO Interface Timing

Parameter	Description	Min	Max	Unit
Tsu	FIFO setup time	4	—	ns
Th	FIFO hold time	0.5	—	ns
Top	FIFO output time	—	4	ns



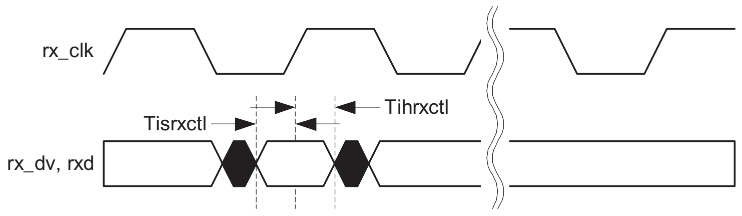
Transmit Timing (MII/GMII)

Parameter	Description	Min	Max	Unit
Tovtx	Transmit data valid after `tx_clk`	-	5.5	ns
Tohtx	Transmit data hold after `tx_clk`	0.5	-	ns



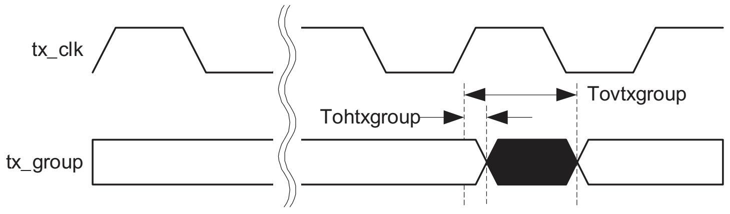
Receive Timing (MII/GMII)

Parameter	Description	Min	Max	Unit
Tisrxctl	Receive data set-up prior to `rx_clk`	2.0	-	ns
Tihrxctl	Receive data hold after `rx_clk`	-	0	ns



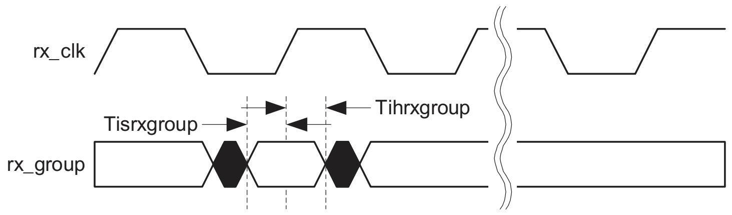
Transmit Timing (TBI)

Parameter	Description	Min	Max	Unit
Tovtxgroup	Transmit data valid after `tx_clk`	-	6.0	ns
Tohtxgroup	Transmit data hold after `tx_clk`	1.0	-	ns



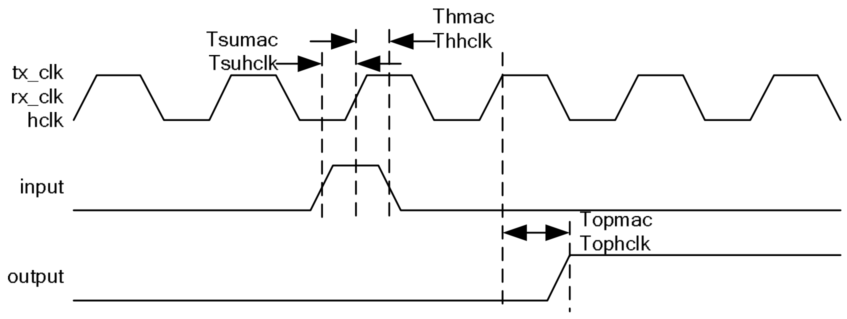
Receive Timing (TBI)

Parameter	Description	Min	Max	Unit
Tisrxgroup	Receive data set-up prior to `rbc0/rbc1`	2.5	-	ns
Tihrxgroup	Receive data hold after `rbc0/rbc1`	-	1.5	ns



Packet Buffer Memory Interface Timing

Parameter	Description	Min	Max	Unit
Tsumac	RX ports A and B or TX port B input setup times	4	—	ns
Thmac	RX ports A and B or TX port B input hold time	0.5	—	ns
Topmac	RX ports A and B or TX port B output time	—	4	ns
Tsuhclk	TX port A input setup times	4	—	ns
Thhclk	TX port A input hold time	0.5	—	ns
Tophclk	TX port A output time	—	4	ns



Clock Domains

Depending on the configuration selected the GEM has up to nine clock domains used internally within the design. These are described in the following table:

Clock	Speed	Description
`hclk`	10 to 500 MHz	AMBA AHB clock used by the DMA block.
`aclk`	10 to 500 MHz	AMBA AXI clock used by the DMA block.
`pclk`	5 to 500 MHz	AMBA APB clock used by the MAC register block and PCS register block.
`tsu_clk`	5 to 400 MHz	Alternate clock source for the time stamp unit. Must run slower than pclk.
`tx_clk`	2.5, 25, or 125 MHz	MAC transmit clock, used by the MAC transmit block. In 10/100 mode, `tx_clk` will run at either 2.5 MHz or 25 MHz as determined by the external PHY MII clock input. When using gigabit mode, the transmit clock must be sourced from a 125 MHz reference clock. Depending upon system architecture, this reference clock may be sourced from an on-chip clock multiplier, generated directly from an off-chip oscillator, or taken from the PHY `rx_clk`. In SGMII mode this clock is sourced from gtx_clk. In 100Mb/s SGMII mode gtx_clk phases need to be deleted to bring the effective frequency down to 12.5MHz and in 10Mb/s SGMII down to 1.25MHz.
`gtx_clk`	125 MHz	PCS transmit clock. In non-SGMII applications this may be sourced directly from tx_clk. In SGMII applications this is sourced from the SerDes and fixed at 125 MHz and divided down in 10 and 100 Mb/s modes to drive tx_clk.
`rx_clk`	2.5, 25, 62.5, or 125 MHz	Clock used by the MAC receive synchronization block. In 10/100 and gigabit mode using the GMII/MII interface, this clock is sourced from the `rx_clk` input of the external PHY and will be either 2.5 MHz, 25 MHz or 125 MHz. When the PCS is selected in gigabit mode, the clock must be sourced from the `rbc1` input of the ten bit interface and will be 62.5 MHz. In 100Mb/s SGMII mode rbc1 phases need to be deleted to bring the effective frequency down to 12.5MHz and in 10Mb/s SGMII down to 1.25MHz. In 10/100 SGMII 8 bits of data are transferred from the PCS to the MAC each clock cycle rather than 16.
`rbc0` `rbc1`	Two 62.5 MHz clocks 180° out of phase	Clocks used in the PCS receive channel.

`n_tx_clk`	2.5, 25, 62.5, or 125 MHz	Inverted `tx_clk` used for loopback module. `tx_clk` domain signals are re-timed to this clock before being passed to the `rx_clk` domain receiver inputs

The GEM contains handshaking logic and synchronizers that ensure reliable propagation of signals across clock boundaries.

As far as the GEM is concerned, management data clock (MDC) is not a clock, but a `pclk` timed output from the registers block.

The maximum frequency of `hclk/aclk` and `pclk` is determined by the speed of the technology library.

The `pclk` clock frequency should be kept to a minimum in the SoC, thus easing the synthesis constraints as control and status will propagate throughout the chip.

The `hclk/aclk` clock frequency must be chosen such that the bandwidth requirements of the chosen transmit and receive Ethernet data rates are met.

When using internal loopback mode, `tx_clk` and `rx_clk` must be provided using the same loopback reference clock, and `n_tx_clk` must be provided with the inverted version of the loopback reference clock.

It is important that receive and transmit are disabled when making the switch into and out of internal loopback, as the clocks provided may glitch whilst switching to the loopback reference clock. Also TBI mode must be disabled for internal loopback. This is because TBI mode configures the receive path to be 16 bits and the transmit path to 8 bits wide.

When operating at gigabit speed using the GMII interface, the system must provide a 125 MHz reference clock to the PHY, normally termed `g_tx_clk`. This clock must be the same clock used for the `tx_clk` input of the GEM to maintain correct relationship between the reference clock and data on the GMII.

The design includes a Verilog module called `gem_clk_cntrl.v`, an example of a basic system clock generator module. This module is instantiated in the testbench to demonstrate connectivity to the GEM design.

Programming Interface

The following registers may be programmed for the GEM module.

Register Map

Note: The Offset Address in the following table is byte-aligned as required by the CPU. It is therefore four times that supplied on `paddr[9:2]`.

Register Map

Offset	Function	R/W	Reset to
0x000	Network control	R/W	0x0000_0000
0x004	Network configuration	R/W	0x0008_0000
0x008	Network status	RO	0b01x0

0x00C	User input/output	R/W	0x0000_0000
0x010	DMA configuration	R/W	0x0002_0004
0x014	Transmit status	R/W	0x0000_0000
0x018	Receive buffer queue base address	R/W	0x0000_0000
0x01C	Transmit buffer queue base address	R/W	0x0000_0000
0x020	Receive status	R/W	0x0000_0000
0x024	Interrupt status	R/W	0x0000_0000
0x028	Interrupt enable	WO	N.A.
0x02C	Interrupt disable	WO	N.A.
0x030	Interrupt mask	RO	0x07FF_FFFF
0x034	PHY maintenance	R/W	0x0000_0000
0x038	Received pause quantum	RO	0x0000_0000
0x03C	Transmit pause quantum	R/W	0x0000_FFFF
0x040	TX Partial Store and Forward	R/W	0x0000_0000
0x044	RX Partial Store and Forward	R/W	0x0000_0000
0x080	Hash register bottom [31:0]	R/W	0x0000_0000
0x084	Hash register top [63:32]	R/W	0x0000_0000
0x088	Specific address 1 bottom [31:0]	R/W	0x0000_0000
0x08C	Specific address 1 top [47:32]	R/W	0x0000_0000
0x090	Specific address 2 bottom [31:0]	R/W	0x0000_0000
0x094	Specific address 2 top [47:32]	R/W	0x0000_0000
0x098	Specific address 3 bottom [31:0]	R/W	0x0000_0000
0x09C	Specific address 3 top [47:32]	R/W	0x0000_0000
0x0A0	Specific address 4 bottom [31:0]	R/W	0x0000_0000
0x0A4	Specific address 4 top [47:32]	R/W	0x0000_0000
0x0A8	Type ID match 1	R/W	0x0000_0000
0x0AC	Type ID match 2	R/W	0x0000_0000
0x0B0	Type ID match 3	R/W	0x0000_0000
0x0B4	Type ID match 4	R/W	0x0000_0000
0x0B8	Wake on LAN	R/W	0x0000_0000
0x0BC	IPG stretch	R/W	0x0000_0000

0x0C0	Stacked VLAN	R/W	0x0000_0000
0x0C4	Transmit PFC Pause Register	R/W	0x0000_0000
0x0C8	Specific address 1 mask bottom [31:0]	R/W	0x0000_0000
0x0CC	Specific address 1 mask top [47:32]	R/W	0x0000_0000
0x0D0	AHB Address mask for RX data buffer accesses	R/W	0x0000_0000
0x0D4	PTP RX unicast IP destination address	R/W	0x0000_0000
0x0D8	PTP TX unicast IP destination address	R/W	0x0000_0000
0x0DC	TSU timer comparison value nanoseconds	R/W	0x0000_0000
0x0E0	TSU timer comparison value seconds	R/W	0x0000_0000
0x0FC	Module ID	RO	0x0002_xxxx
0x100	Octets transmitted [31:0] (in frames without error)	RO	0x0000_0000
0x104	Octets transmitted [47:32] (in frames without error)	RO	0x0000_0000
0x108	Frames transmitted without error, excluding pause frames	RO	0x0000_0000
0x10C	Broadcast frames transmitted without error, excluding pause frames	RO	0x0000_0000
0x110	Multicast frames transmitted without error, excluding pause frames	RO	0x0000_0000
0x114	Transmitted pause frames (unless sent through external FIFO interface)	RO	0x0000_0000
0x118	64 byte frames transmitted without error, excluding pause frames	RO	0x0000_0000
0x11C	65 to 127 byte frames transmitted without error	RO	0x0000_0000
0x120	128 to 255 byte frames transmitted without error	RO	0x0000_0000
0x124	256 to 511 byte frames transmitted without error	RO	0x0000_0000
0x128	512 to 1023 byte frames transmitted without error	RO	0x0000_0000
0x12C	1024 to 1518 byte frames transmitted without error	RO	0x0000_0000
0x130	Greater than 1518 byte frames transmitted without error	RO	0x0000_0000
0x134	Transmit under run errors	RO	0x0000_0000
0x138	Single collision frames	RO	0x0000_0000
0x13C	Multiple collision frames	RO	0x0000_0000
0x140	Excessive collisions	RO	0x0000_0000
0x144	Late collisions	RO	0x0000_0000

0x148	Deferred transmission frames	RO	0x0000_0000
0x14C	Carrier sense errors	RO	0x0000_0000
0x150	Octets received [31:0] (in frames without error)	RO	0x0000_0000
0x154	Octets received [47:32] (in frames without error)	RO	0x0000_0000
0x158	Frames received without error, excluding pause frames.	RO	0x0000_0000
0x15C	Broadcast frames received without error, excluding pause frames	RO	0x0000_0000
0x160	Multicast frames received without error, excluding pause frames	RO	0x0000_0000
0x164	Pause frames received	RO	0x0000_0000
0x168	64 byte frames received without error, excluding pause frames	RO	0x0000_0000
0x16C	65 to 127 byte frames received without error	RO	0x0000_0000
0x170	128 to 255 byte frames received without error	RO	0x0000_0000
0x174	256 to 511 byte frames received without error	RO	0x0000_0000
0x178	512 to 1023 byte frames received without error	RO	0x0000_0000
0x17C	1024 to 1518 byte frames received without error	RO	0x0000_0000
0x180	1519 to maximum byte frames received without error	RO	0x0000_0000
0x184	Undersize frames received	RO	0x0000_0000
0x188	Oversize frames received	RO	0x0000_0000
0x18C	Jabbers received	RO	0x0000_0000
0x190	Frame check sequence errors	RO	0x0000_0000
0x194	Length field frame errors	RO	0x0000_0000
0x198	Receive symbol errors	RO	0x0000_0000
0x19C	Alignment errors	RO	0x0000_0000
0x1A0	Receive resource errors	RO	0x0000_0000
0x1A4	Receive overrun errors	RO	0x0000_0000
0x1A8	IP header checksum errors	RO	0x0000_0000
0x1AC	TCP checksum errors	RO	0x0000_0000
0x1B0	UDP checksum errors	RO	0x0000_0000
0x1C8	1588 timer sync strobe seconds	R/W	0x0000_0000
0x1CC	1588 timer sync strobe nanoseconds	R/W	0x0000_0000

0x1D0	1588 timer seconds	R/W	0x0000_0000
0x1D4	1588 timer nanoseconds	R/W	0x0000_0000
0x1D8	1588 timer adjust	WO	
0x1DC	1588 timer increment	R/W	0x0000_0000
0x1E0	PTP event frame transmitted seconds	RO	0x0000_0000
0x1E4	PTP event frame transmitted nanoseconds	RO	0x0000_0000
0x1E8	PTP event frame received seconds	RO	0x0000_0000
0x1EC	PTP event frame received nanoseconds	RO	0x0000_0000
0x1F0	PTP peer event frame transmitted seconds	RO	0x0000_0000
0x1F4	PTP peer event frame transmitted nanoseconds	RO	0x0000_0000
0x1F8	PTP peer event frame received seconds	RO	0x0000_0000
0x1FC	PTP peer event frame received nanoseconds	RO	0x0000_0000
0x200	PCS control	R/W	0x0000_9040
0x204	PCS status	RO	0x0000_0109
0x208	PCS upper PHY identifier	RO	0x0000_0002
0x20C	PCS lower PHY identifier	RO	0x0000_xxxx
0x210	PCS auto-negotiation advertisement	R/W	0x0000_0060
0x214	PCS auto-negotiation link partner ability	RO	0x0000_0000
0x218	PCS auto-negotiation expansion	RO	0x0000_0004
0x21C	PCS auto-negotiation next page	R/W	0x0000_0000
0x220	PCS auto-negotiation link partner next page	RO	0x0000_0000
0x224	Reserved		
0x228	Reserved		
0x22C	Reserved		
0x230	Reserved		
0x234	Reserved		
0x238	Reserved		
0x23C	PCS extended status	RO	0x0000_C000
0x270	Received LPI transitions	RO	0x0000_0000
0x274	Received LPI time	RO	0x0000_0000
0x278	Transmit LPI transitions	RO	0x0000_0000

0x27C	Transmit LPI time	RO	0x0000_0000
0x280	Design Configuration Register 1	RO	N/A
0x284	Design Configuration Register 2	RO	N/A
0x288	Design Configuration Register 3	RO	N/A
0x28C	Design Configuration Register 4	RO	N/A
0x290	Design Configuration Register 5	RO	N/A
0x294	Design Configuration Register 6	RO	N/A
0x298	Design Configuration Register 7	RO	N/A
The following registers are used when Priority Queueing is enabled			
0x400	Interrupt Status Register – Priority Queue 1	RO	0x0000_0000
0x404	Interrupt Status Register – Priority Queue 2	RO	0x0000_0000
0x408	Interrupt Status Register – Priority Queue 3	RO	0x0000_0000
0x40c	Interrupt Status Register – Priority Queue 4	RO	0x0000_0000
0x410	Interrupt Status Register – Priority Queue5	RO	0x0000_0000
0x414	Interrupt Status Register – Priority Queue 6	RO	0x0000_0000
0x418	Interrupt Status Register – Priority Queue 7	RO	0x0000_0000
0x440	Transmit Buffer Queue Base Address – Queue 1	RW	0x0000_0000
0x444	Transmit Buffer Queue Base Address – Queue 2	RW	0x0000_0000
0x448	Transmit Buffer Queue Base Address – Queue 3	RW	0x0000_0000
0x44c	Transmit Buffer Queue Base Address – Queue 4	RW	0x0000_0000
0x450	Transmit Buffer Queue Base Address – Queue 5	RW	0x0000_0000
0x454	Transmit Buffer Queue Base Address – Queue 6	RW	0x0000_0000
0x458	Transmit Buffer Queue Base Address – Queue 7	RW	0x0000_0000
0x480	Receive Buffer Queue Base Address – Queue 1	RW	0x0000_0000
0x484	Receive Buffer Queue Base Address – Queue 2	RW	0x0000_0000
0x488	Receive Buffer Queue Base Address – Queue 3	RW	0x0000_0000
0x48c	Receive Buffer Queue Base Address – Queue 4	RW	0x0000_0000
0x490	Receive Buffer Queue Base Address – Queue 5	RW	0x0000_0000
0x494	Receive Buffer Queue Base Address – Queue 6	RW	0x0000_0000
0x498	Receive Buffer Queue Base Address – Queue 7	RW	0x0000_0000
0x4a0	Transmit Buffer Queue Base Address – Queue 1	RW	0x0000_0002

0x4a4	Transmit Buffer Queue Base Address – Queue 2	RW	0x0000_0002
0x4a8	Transmit Buffer Queue Base Address – Queue 3	RW	0x0000_0002
0x4ac	Transmit Buffer Queue Base Address – Queue 4	RW	0x0000_0002
0x4b0	Transmit Buffer Queue Base Address – Queue 5	RW	0x0000_0002
0x4b4	Transmit Buffer Queue Base Address – Queue 6	RW	0x0000_0002
0x4b8	Transmit Buffer Queue Base Address – Queue 7	RW	0x0000_0002
0x500 to 0x53c	Screening – Type 1 Registers (up to 16)	RW	0x0000_0000
0x540 to 0x57c	Screening – Type 2 Registers (up to 16)	RW	0x0000_0000
0x600	Interrupt Enable Register – Priority Queue 1	WO	0x0000_0000
0x604	Interrupt Enable Register – Priority Queue 2	WO	0x0000_0000
0x608	Interrupt Enable Register – Priority Queue 3	WO	0x0000_0000
0x60c	Interrupt Enable Register – Priority Queue 4	WO	0x0000_0000
0x610	Interrupt Enable Register – Priority Queue 5	WO	0x0000_0000
0x614	Interrupt Enable Register – Priority Queue 6	WO	0x0000_0000
0x618	Interrupt Enable Register – Priority Queue 7	WO	0x0000_0000
0x620	Interrupt Disable Register – Priority Queue 1	WO	0x0000_0000
0x624	Interrupt Disable Register – Priority Queue 2	WO	0x0000_0000
0x628	Interrupt Disable Register – Priority Queue 3	WO	0x0000_0000
0x62c	Interrupt Disable Register – Priority Queue 4	WO	0x0000_0000
0x630	Interrupt Disable Register – Priority Queue 5	WO	0x0000_0000
0x634	Interrupt Disable Register – Priority Queue 6	WO	0x0000_0000
0x638	Interrupt Disable Register – Priority Queue 7	WO	0x0000_0000
0x640	Interrupt Mask Register – Priority Queue 1	RW	0x0000_0000
0x644	Interrupt Mask Register – Priority Queue 2	RW	0x0000_0000
0x648	Interrupt Mask Register – Priority Queue 3	RW	0x0000_0000
0x64c	Interrupt Mask Register – Priority Queue 4	RW	0x0000_0000
0x650	Interrupt Mask Register – Priority Queue 5	RW	0x0000_0000
0x654	Interrupt Mask Register – Priority Queue 6	RW	0x0000_0000
0x658	Interrupt Mask Register – Priority Queue 7	RW	0x0000_0000

0x6e0 to 0x6fc	Screener Type 2 – Ethertype Registers (up to 8)	RW	0x0000_0000
0x700, 0x704 to 0x7f8, 0x7fc	Screener Type 2 – Compare Registers (up to 32), each using two 32-bit mapped registers	RW	0x0000_0000

Physical Estimates

Approximate two input NAND equivalent gate counts:

Basic configuration	37,000
The DMA option adds	73,000
The alternate DMA packet buffer option adds	54,000
The PCS option adds	8,000
The TSU option adds	8,500
The GEM statistics registers adds	20,000

As an example, a configuration using the DMA packet buffer without PCS, but with TSU and GEM statistics registers was synthesized with an area of approximately 115,000 gates.

Flip-flop counts:

Basic configuration	2550
The DMA option adds	4870
The alternate DMA packet buffer option adds	2629
The PCS option adds	562
The TSU option adds	528
The GEM statistics registers adds	1090

The total pin count for the basic configuration of the GEM is 970 with external FIFO i/f or 949 with DMA-AHB (internal FIFO configuration) or 1306 with the DMA-AHB (packet buffer single queue configuration)

The SoC external pin count is 27 if using GMII, or 28 using TBI

The power consumption of the GEM in typical configuration is 15mW using TSMC 0.13G

Verification

All our IP modules are verified to one of the following levels:

- Gold IP has been to target silicon.
- Silver IP has been to silicon in FPGA.
- Bronze IP has been verified in simulation with logical timing closure.
- In development IP has not yet been verified.

Please contact the IPGallery™ (ipgallery@cadence.com) for the latest verification information.

Deliverables

The full IP package comes complete with:

- Verilog HDL
- Cadence RTL Compiler synthesis scripts
- Verilog testbench
- *GEM User's guide* with full programming interface, parameterization instructions and synthesis instructions.