Generated Specification Document

INTRODUCTION

-- Purpose and Scope --

eth_top is a synthesizable, top-level 10/100 Mb/s Ethernet MAC IP core intended for SoC integration. Its purpose is to provide a complete IEEE 802.3-compliant MAC datapath, control/management plane, and system bus/DMA subsystem through a Wishbone interface. The scope includes: full transmit and receive paths with CSMA/CD for half-duplex and compliant preamble/SFD, padding, CRC generation/checking, IFG and length enforcement; 802.3x flow control with PAUSE frame generation on request and PAUSE enforcement on receive; a Wishbone slave register interface for configuration, status, and interrupts, and a Wishbone master DMA engine that moves frame data to/from system memory using internal buffer descriptors and burst transfers; a MIIM (MDIO/MDC) PHY management controller with programmable MDC division and read/write/scan support; address filtering for unicast, broadcast, and multicast via a 64-bit hash, plus promiscuous mode; optional internal MII loopback; robust multi-clock operation across system, MII TX, and MII RX domains using synchronizers and handshake pulses; and comprehensive per-packet status and interrupt reporting for TX/RX complete/error, busy, and flow-control events. The design targets nibble-wide MII only, relies on external system memory for frame buffers, and excludes PHY implementation, GMII/RGMII operation, advanced offloads, time stamping/PTP, VLAN, and switching.

-- Features and Capabilities --

- 10/100 Mbps Ethernet MAC top-level with IEEE 802.3 MII (4-bit) TX/RX, CRC-32 FCS, IFG (96 bit-times), CSMA/CD, and 802.3x PAUSE support.
- Multi-interface I/O:
- Wishbone slave for registers and BD RAM with clean decode, read-data mux, and one-shot ack/err pulses (bus error on invalid address or no byte select).
- Wishbone master DMA with word-aligned addresses, incrementing bursts (cti=010/111), byte-lane strobes, and efficient 4-beat bursts gated by FIFO thresholds.
- MII nibble TX (MTxClk) and RX (MRxClk) data/enable/error lanes.
- MDIO/MDC (MIIM) with programmable MDC divider, optional preamble suppression, read/write/scan, proper turnaround tri-state, and MIIRX_DATA auto-update.
- Transmit MAC (TX):
- Preamble/SFD generation, payload transmit, optional padding to minimum frame, CRC-32 insertion.
- Half-duplex CSMA/CD: carrier sense, collision detect, 32-bit jam, binary exponential backoff (LFSR), retry/late-collision handling, programmable max retries.
- Programmable IPG (IPGT/IPGR1/IPGR2), delayed-CRC window, no-backoff, excessive-defer enable, hug (oversize) mode.
- Status/events: TxDone, TxRetry, TxAbort, underrun, late collision, excessive defer, carrier-sense lost; “will transmit” indication; MTxErr on oversize/underrun.
- 802.3x control frame injection (PAUSE) on request or flow-control state with automatic padding/CRC and isolation from normal TX data.
- Receive MAC (RX):
- Nibble-to-byte assembly with aligned RxValid/Start/End; dribble-nibble safe end detection.

- Preamble/SFD recognition; IFG enforcement (drop on violation); CRC-32 check with correct residue and per-frame good/bad.
- Length enforcement: programmable min/max with hug mode; short/oversize detection.
- Address filtering: unicast match, broadcast enable/disable, 64-bit multicast hash filter, promiscuous mode; AddressMiss and RxAbort on DA failure.
- Auxiliary classification: broadcast/multicast flags and 6-bit CRC hash export; RX symbol error capture.
- IEEE 802.3x flow control:
- RX PAUSE detect (Type 0x8808, Opcode 0x0001) with pause-time capture and timer in 512-bit-time quanta.
- TX PAUSE generation; gating prevents new normal TX frame starts while timer active.
- PHY management (MIIM): clean request edge-detect, in-progress arbitration, MDC-aligned bit timing, read data latch points, and LinkFail status extraction.
- DMA engine and buffer descriptors:
- Internal 256x32 single-port RAM for BDs and pointers; CPU/TX/RX arbitration.
- TX: BD fetch, pointer/length handling, burst reads to TX FIFO, valid-tail byte handling, start/end frame generation, underrun detection.
- RX: byte packing to 32-bit words honoring alignment/byte enables, burst writes to memory, tail flush, overrun detection.
- Descriptor ring with wrap bits; TX/RX partition via TX_BD_NUM; per-BD IRQ enable and status writeback (B vs E IRQs).
- Register block and interrupts:
- Comprehensive control/status: MODER, IPG, lengths, collision config, TX_BD_NUM, MAC address, multicast hash, MIIM control/addr/data/status, TX control (pause).
- Interrupt controller with RW1C sources and mask: TxB, TxE, RxB, RxE, Busy, TxC, RxC; single int_o output.
- Safety gating: TxEn requires TX_BD_NUM>0; RxEn requires TX_BD_NUM<0x80; field-width-safe readback.
- Duplex and loopback:
- Full- and half-duplex operation; half-duplex uses carrier/collision; RX is blocked while TX in half-duplex.
- Internal MII loopback routing TX pins to RX path for diagnostics.
- Robust multi-clock design:
- Three domains (Wishbone, MTxClk, MRxClk) with multi-flop synchronizers and pulse handshakes for control/status crossings (pause requests, aborts, interrupts, TX/RX status).
- Error handling and bus robustness:
- Clean one-shot ack/err to prevent double-servicing; Wishbone bus error on decode miss/no byte select.
- TX/RX error signaling: MTxErr, late-collision qualification, RX symbol error capture, RX abort causes.
- Diagnostics and observability:
- Debug data register, latched retry count/late collision, defer latch, address-miss reporting, LinkFail via MIIM, consolidated per-frame TX/RX status in macstatus.
- Parameterizable FIFOs with almost-full/empty flags for burst gating and throughput.
- Standards compliance: IEEE 802.3 MII timing, CRC-32 polynomial/residue, IFG, CSMA/CD rules, and 802.3x flow control.

-- Standards Compliance --

IEEE 802.3 MAC (10/100 Mb/s, MII): TX generates preamble/SFD (55…55 D5), enforces 64-byte minimum with pad, and appends reflected CRC-32 (poly 0x04C11DB7, standard init/final invert); RX detects SFD, enforces 96 bit-time IFG, assembles LSB-first nibbles to bytes, handles dribble-nibble, and verifies FCS by standard residue (0xC704DD7B). CSMA/CD (half-duplex): Deferral on carrier sense, 32-bit JAM, binary exponential backoff with randomized slots, collision window and late-collision

classification, retry up to configurable MaxRet; full-duplex disables CSMA/CD effects per 802.3. Addressing/filtering: Accepts local unicast, broadcast (configurable), and multicast via 64-bit CRC hash; promiscuous/pass-all provided without altering wire protocol. IEEE 802.3x flow control: RX recognizes MAC Control PAUSE (DA 01-80-C2-00-00-01 or local MAC; EtherType 0x8808; OpCode 0x0001), loads Pause Time; slot timer decrements in 512 bit-time quanta; TX honors PAUSE by gating new data-frame starts; TX can generate compliant PAUSE frames (proper format, pad, CRC) and blocks normal data during control transmission. MDIO/MIIM Clause 22: Emits 64-bit management frames (optional 32-bit preamble), ST=01, OP=01 write/10 read, correct PHYAD/REGAD fields, read turnaround tri-state, data latched at defined bit positions; MDC is a programmable divide with ~50% duty; preamble suppression (NoPre) supported per spec. MII data ordering and alignment: LSB-first nibble ordering within bytes; correct CRC enable/init at SFD boundaries. Configurable but standard-consistent limits: r_MinFL/r_MaxFL (optional HugEn to permit non-standard oversize), r_IFG test override without changing wire behavior; loopback/promiscuous for diagnostics. System bus: Wishbone Rev. B master/slave behavior (CTI/BTE bursts, byte enables, aligned addresses, single-cycle ACK/ERR); not an Ethernet standard but conforms to Wishbone conventions. Limitations/assumptions: 10/100 MII only (no GMII/RGMII/SGMII); Clause 22 only (no Clause 45); MDC must be configured within PHY max (typically ≤2.5 MHz); delayed-CRC controls are internal timing aids and do not affect on-the-wire CRC.

-- Clock Domains and Reset Strategy --

Clock domains
- System/Wishbone domain (wb_clk_i): Hosts top-level bus decode/ack/err, eth_registers (all config/status/IRQs), eth_wishbone (DMA/control engine), eth_miim (MDIO shifter) and eth_clockgen (MDC divider). Wishbone read-data is muxed synchronously; wb_ack_o/wb_err_o are one-cycle registered pulses.
- Transmit MII domain (MTxClk = mtx_clk_pad_i): Hosts eth_txethmac (TX FSM/CRC/backoff), eth_maccontrol (PAUSE arbitration), eth_macstatus TX latches, and TX-side control from eth_wishbone.
- Receive MII domain (MRxClk = mrx_clk_pad_i): Hosts eth_rxethmac (RX front-end/byte align/CRC control), address check and counters, eth_macstatus RX latches, and RX-side control from eth_wishbone.
- MIIM/MDC: MDC is derived from wb_clk_i inside eth_clockgen. MIIM logic remains in wb_clk_i; all MDIO state updates are qualified by one-cycle MdcEn/MdcEn_n strobes. MDC is an external serial clock, not a separate design clock domain.

Reset strategy
- A single asynchronous reset input (wb_rst_i) fanouts to all modules; each domain exposes an async Reset (e.g., Reset/TxReset/RxReset). Assertion forces a known idle state in every domain.
- Safe deassertion is ensured by CDC structures: 2–3 flip-flop synchronizers on single-bit crossings, edge-detected one-shot requests (q2 & ~q3), and symmetric request/acknowledge handshakes. Pulses are always re-created in the destination domain.
- Module timing under reset:
- wb_clk_i domain (eth_registers, eth_wishbone, eth_miim, eth_clockgen): synchronous operation to wb_clk_i with async Reset; some registers also provide local SyncReset (lower priority than async Reset).
- MTxClk and MRxClk domains (TX/RX datapaths, MAC control/status): all state is synchronous to its local MII clock with async Reset.
- CDC and gating rules that protect reset release and normal operation:
- Level/event crossings use multi-FF sync and explicit clears returned to the source domain (e.g., RxAbort latch MRxClk↔wb_clk_i handshake).
- Software-originated controls (e.g., r_TxPauseRq) and hardware status returns (e.g., WillSendControlFrame) traverse domains via triple-sync and edge-detect; acknowledgments are

synchronized back to clear source latches.
- Asynchronous external senses are synchronized before use (e.g., CarrierSense 2-FF into MTxClk; Collision sampled and latched). RX enable is captured only when MRxDV is low to avoid mid-frame glitches.
- No dual-clock FIFOs are used; each domain keeps single-clock FIFOs, and inter-domain transfers are coordinated by synchronized strobes and blocking guards in the wb_clk_i domain to prevent duplicate processing.

-- Operating Modes (Full/Half Duplex, Loopback) --

Operating modes are controlled by MODER and related register fields decoded in eth_registers. MODER[10] (r_FullD) selects duplex; MODER[7] (r_LoopBck) enables internal loopback; r_RxEn gates RX.

Full■duplex (r_FullD=1): CSMA/CD is disabled by gating TxCarrierSense and Collision (both forced inactive), so no jam/backoff paths are taken. RX remains active during TX because the Transmitting indication into the RX domain is deasserted in full■duplex, allowing simultaneous TX/RX. Inter■packet gap uses IPGT timing. IEEE 802.3x flow control is supported: PAUSE frames are detected and pause the transmitter; PAUSE generation is performed when TxFlow is enabled.

Half■duplex (r_FullD=0): CSMA/CD is active. TX defers to carrier sense; collisions trigger jam (8 nibbles) and exponential backoff with a retry window that grows with collision count up to r_MaxRet; r_NoBckof can suppress backoff. The collision window is defined by CollValid; late collisions cause immediate abort; exceeding r_MaxRet causes MaxCollision abort. RX is blocked while TX is active: if MRxDV asserts during a transmit, the RX state machine drops the frame. IPG uses IPGR1/IPGR2 sequencing; excessive defer detection is present and can be masked with r_ExDfrEn. RX■side late collision reporting is active only in half■duplex.

Internal MII loopback (r_LoopBck=1): MRxD/MRxDV/MRxErr are sourced from MTxD/MTxEn/MTxErr, providing an internal TX→RX path. MRxDV/MRxErr are gated by RxEnSync to avoid mid■packet enables; carrier■sense■loss status excludes loopback. Duplex rules still apply in loopback; best results are in full■duplex (in half■duplex, the Transmitting gate can force RX drops).

RX enable synchronization: r_RxEn is synchronized into the RX clock domain (RxEnSync) and used to gate MRxDV/MRxErr in both normal and loopback operation to prevent mid■packet glitches.

-- Submodule Overview --

eth_top comprises seven coordinated submodules that implement an Ethernet MAC with Wishbone control/DMA, MII TX/RX datapaths, flow control, PHY management, and status capture. Each submodule’s role, clock domain, principal interfaces, and key subcomponents are summarized below.

- eth_registers (System register file & IRQ controller)
- Clock domain: wb_clk_i (async reset resynchronized per domain)
- Role: Wishbone slave decode; holds configuration registers (duplex, CRC/pad, IFG/IPG, max/min lengths, flow control enables, MIIM settings, MAC address, hash filters, TX/RX enable gating via TX_BD_NUM); manages RW1C interrupt latches/masks; exposes r_* control signals; MIIM command/data/address registers; propagates Busy/NValid/UpdateMIIRX_DATAReg.
- Interfaces: wb_* slave bus (adr/dat/we/sel/stb/cyc/ack/err); emits config/status to MAC/DMA; receives event/status inputs from eth_macstatus/eth_wishbone/eth_miim.
- Submodules: eth_register (byte-lane-enabled storage with async/sync reset).

- eth_wishbone (DMA/descriptor engine)

- Clock domain: wb_clk_i
- Role: Arbitrates CPU vs TX/RX access to internal 256x32 BD RAM; performs TX descriptor fetch/data reads and RX data/status writes; manages Wishbone master transactions (bursts, byte strobes, alignment), TX/RX FIFOs, and CDC handshakes to MTxClk/MRxClk; detects underrun; generates TxB/RxB and TxE/RxE/Busy IRQ sources.
- Interfaces: wb_* master (m_wb_*); BD RAM port; TX/RX FIFOs; control from eth_registers; status/handshakes to eth_txethmac/eth_rxethmac; IRQ lines to eth_registers.
- Submodules: eth_spram_256x32 (BD RAM), eth_fifo (parameterized single-clock FIFOs).

- eth_txethmac (Transmit MAC engine)
- Clock domain: mtx_clk_pad_i
- Role: Frame transmit sequencing (preamble/SFD, payload, PAD, FCS); CSMA/CD (jam/backoff/retries) in half-duplex; generates MTxD/MTxEn/MTxErr; produces TxDone/TxRetry/TxAbort, TxUsedData, WillTransmit; honors IFG/IPG, max length, delayed CRC windows; integrates flow-control overrides.
- Interfaces: MII TX pads (MTxD[3:0], MTxEn, MTxErr); data/control from DMA (TxStartFrm, TxData, TxPad, TxCRCEn); collision/carrier inputs; flow-control inputs (CtrlMux, ControlData, TxCtrlStart/End, BlockTxDone); status to eth_wishbone/eth_macstatus.
- Submodules: eth_txstatem, eth_txcounters, eth_crc (generator), eth_random (LFSR backoff).

- eth_rxethmac (Receive front-end)
- Clock domain: mrx_clk_pad_i
- Role: Assemble MII nibbles to bytes; drive RxValid/Start/End; control CRC enable/init and delayed CRC; enforce IFG/SFD; count bytes; perform destination address acceptance (unicast/broadcast/multicast hash/promiscuous); flag short/too-big, dribble nibble, invalid symbols; emit RxAbort/AddressMiss.
- Interfaces: MII RX pads (MRxD[3:0], MRxDV, MRxErr, MColl optional via loopback); outputs to DMA (RxData, RxEndFrm, RxAbort); configuration from eth_registers; status to eth_macstatus; flow-control PAUSE detect to eth_maccontrol.
- Submodules: eth_rxstatem, eth_rxcounters, eth_rxaddrcheck, eth_crc (checker).

- eth_maccontrol (IEEE 802.3x flow control)
- Clock domain: Primarily mtx_clk_pad_i (TX-side arbitration), with RX-side detection and WB-side configuration synchronized as needed.
- Role: Detect PAUSE frames on RX; generate PAUSE timer/quanta; arbitrate TX path to insert control frames; force PAD/CRC during control; mask/retime TxDone/TxAbort; block normal TxStart while PAUSE active; handshake signals across domains (TPauseRq, WillSendControlFrame).
- Interfaces: Inputs from RX detect and WB config (r_TxFlow/r_RxFlow, quanta); overrides to TX engine (CtrlMux, ControlData, TxCtrlStart/End, BlockTxDone); status handshakes to registers.
- Submodules: eth_receivecontrol (PAUSE detect/timer), eth_transmitcontrol (control-frame generator).

- eth_miim (MDIO/MDC master)
- Clock domain: wb_clk_i (MDC generated internally)
- Role: Executes MIIM write/read/scan; aligns requests to MdcEn; builds/shifts MDIO frames with optional preamble; manages read turnaround; provides Busy/EndBusy/NValid; pulses UpdateMIIRX_DATAReg to update MIIRX_DATA in eth_registers.
- Interfaces: WB-side control/status via eth_registers; PHY pads (mdc_pad_o, md_pad_i, mdo_pad_o, mdo_en_pad_o); clock enable strobes (MdcEn).
- Submodules: eth_clockgen (programmable MDC divider), eth_shiftreg (serializer/deserializer with Prsd/link-fail latch), eth_outputcontrol (MDIO drive/enable with preamble/turnaround).

- eth_macstatus (TX/RX status collection)

- Clock domains: Dual (mrx_clk_pad_i for RX latches, mtx_clk_pad_i for TX latches), synchronized to WB for register/IRQ updates.
- Role: RX: latch CRC error, short/too-big, invalid symbol, dribble nibble, late collision; TX: latch retry count, retry limit, late collision, defer, CarrierSenseLost; provide per-frame outcomes to DMA and interrupt sources to eth_registers.
- Interfaces: Inputs from eth_txethmac/eth_rxethmac; outputs to eth_registers and eth_wishbone (for BD/status writes and IRQ aggregation).

Top-level integration highlights
- Clock domains: wb_clk_i (registers/DMA/MIIM), mtx_clk_pad_i (TX), mrx_clk_pad_i (RX); asynchronous reset with per-domain resynchronization.
- CDC: Multi-flop synchronizers and one-shot pulses for cross-domain events (flow-control handshakes, RX abort, TX outcomes, FIFO write/shift-end), with Block* flags to prevent double-counting.
- Wishbone: Slave decode/mux in eth_registers; master transactions in eth_wishbone (m_wb_cti_o burst use, alignment, sel strobes); BD RAM arbitrated between CPU and MAC.
- MII loopback: Optional r_LoopBck routes TX to RX; Rx enable gating via synchronized RxEn to avoid mid-packet glitches.
- Configuration ties: r_FullD (collision handling), r_NoPre (MIIM preamble), r_HugEn (max-length), r_RecSmall (RX abort policy), r_TxFlow/r_RxFlow (flow control), TX_BD_NUM (TX/RX BD partition).

-- Assumptions and Constraints --

- Clocks and resets
- Three independent, stable clock domains are required: wb_clk_i (system/Wishbone), mtx_clk_pad_i (TX MII), mrx_clk_pad_i (RX MII).
- wb_rst_i is asynchronous; deassertion must meet recovery/removal and be clean. Internal logic resynchronizes resets and strobes; external logic must not rely on level-sensitive CDC.

- Clock-domain crossing semantics
- Cross-domain controls/events use one-shot request/ack pulses and multi-flop synchronizers; software/logic must not depend on persistent levels and must wait for completion pulses before reissuing commands.
- Pause updates and RxEn gating are timed to idle/low-activity windows; do not toggle enables mid-packet.

- MII datapath and duplex assumptions
- Only 4-bit MII at 10/100 Mb/s is supported; GMII/RGMII/1G are not supported.
- Half-duplex requires valid PHY CRS/COL (mcrs_pad_i/mcoll_pad_i); in full duplex, CRS/COL are ignored by MAC decisions.
- Internal loopback (r_LoopBck=1) routes TX MII into RX and ignores external MRx*; intended for test only.
- RX enable (r_RxEn) is sampled when MRxDV is low; do not change r_RxEn while a frame is active.

- Framing, CRC, and timing
- TX preamble/SFD is fixed (0x55…55 D5). CRC-32 conforms to IEEE 802.3; CRC enable must remain asserted over the entire CRC-covered region.
- Frame length is constrained by r_MinFL/r_MaxFL; with r_HugEn=0, oversize frames are aborted/flagged. Padding to minimum length occurs as required.
- IFG enforcement is 96 bit-times (24 nibbles) unless overridden; SFD before IFG completion forces drop.
- Collision window and late-collision handling depend on r_CollValid and duplex; late collisions in half-duplex cause aborts without retry.

- 802.3x flow control
- RX PAUSE detection requires valid DA (reserved multicast DA or station MAC), EtherType 0x8808, OpCode 0x0001; r_RxFlow=1 enables timer in quanta of 512 bit-times (derived from MRxClk).
- TX PAUSE occurs only when r_TxFlow=1 and a request pulse is observed; control frames preempt at safe idle points. While pause timer > 0, new normal-data frame starts are suppressed; frames in flight complete.

- MIIM/MDIO interface
- MDC is generated by a programmable divider; divider < 2 is clamped to 2. Odd dividers round down to preserve ~50% duty; system must program within PHY MDC limits.
- Only one MIIM operation (write/read/scan) may be active; Busy stays asserted conservatively during alignment/scan. Software must wait for Busy deassertion or EndBusy before starting a new command.
- No-preamble mode (r_MiiNoPre=1) requires PHY tolerance for missing preamble.
- MDIO must be tri-stated: mdio = MdoEn ? Mdo : 'bz. Output pipeline adds two MdcEn_n-qualified cycles of latency; upstream timing is pre-aligned.

- Wishbone slave (register/BD space)
- 32-bit data bus with byte enables; at least one wb_sel_i bit must be asserted or an error pulse is produced.
- Address decoding uses wb_adr_i[11:10]: 00=registers, 01=BD/SRAM, 1x=error. wb_ack_o and wb_err_o are one-cycle pulses; masters must sample edges (not level-held).
- Register fields enforce width limits; out-of-range writes (e.g., TX_BD_NUM > 0x80) are ignored. Some registers zero unused upper bits on readback; avoid RMW of reserved bits.
- INT_SOURCE is RW1C; software must write 1s to clear. Interrupt outputs are masked OR of latched sources.

- Wishbone master (DMA) and memory assumptions
- m_wb_adr_o is word-aligned; m_wb_sel_o conveys valid byte lanes (RX writes may be partial on first/last word, TX reads are full 32-bit words).
- Only classic/incrementing bursts: m_wb_bte_o=2’b00; m_wb_cti_o=3’b010 during bursts and 3’b111 on last beat. Targets must support these or accept fallback to single beats.
- Each transfer retires on m_wb_ack_i or m_wb_err_i; targets must not hold both low indefinitely for an issued cycle.
- 4-beat bursts are opportunistic, gated by FIFO levels and lengths; insufficient bandwidth reduces to non-bursting behavior.

- BD RAM, FIFOs, and rings
- Single-port 256x32 BD RAM arbitrated among CPU, TX, RX; only one access per wb_clk_i cycle. BD read latency is 1 cycle due to registered address.
- TX_BD_NUM partitions the 0..0x7F ring: 0..TX_BD_NUM-1 = TX, TX_BD_NUM..0x7F = RX. r_TxEn requires TX_BD_NUM>0; r_RxEn requires TX_BD_NUM<0x80.
- TX requires BD Ready and legal length; zero/too-small lengths may hit edge-case handling—program standards-compliant lengths.
- Internal FIFOs are single-clock; do not read when empty or write when full. Avoid simultaneous read+write at empty/full boundaries.

- Address filtering and promiscuous
- Frames pass if DA matches station MAC, is broadcast (unless r_Bro=1), or multicast with hash hit (HASH1:HASH0 indexed by CRC[31:26]).
- r_Pro=1 bypasses abort on DA mismatch (promiscuous), but AddressMiss can still be reported. DA decision is at end of DA byte 6; rejection pulses then.

- Software-visible constraints
- Many status/handshake signals are one-shot pulses; drivers must poll/clear latched status registers and not assume level persistence.
- Some config bits self-mask (e.g., r_TxEn/r_RxEn by TX_BD_NUM); read back to verify effective configuration.
- MIIRX_DATA is hardware-updated only; CPU writes are ignored.

- Environmental/integration limits
- Design targets 10/100 Mb/s MII; gigabit variants are out of scope.
- External PHY must present correctly timed MII, CRS, COL and tolerate chosen MDC frequency; MDIO requires pull-up and proper tri-state wiring.
- CRS/COL wiring must reflect duplex; MAC masks CRS/COL in full duplex.
- Asynchronous reset distribution must be clean despite internal synchronizers.

- Implementation limits and behaviors
- Descriptor space is fixed at 128 entries per direction (0..0x7F); larger rings are unsupported.
- FIFO depths and 4-beat burst lengths are fixed; low bandwidth degrades throughput but remains functional.
- Backoff uses a 10-bit LFSR; retry windows grow with RetryCnt and saturate at 10 bits; NoBckof bypasses backoff.
- Excessive defer is masked when r_ExDfrEn=1; otherwise a fixed nibble-count threshold triggers abort/flag.
- DlyCrcEn adds a 4-byte accounting delay; ByteCntOut reports ByteCnt+4.
- Dribble nibble and InvalidSymbol are tracked across a frame and clear at frame end.

- Verification/simulation notes
- Initial block prints are informational only.
- RAM model exhibits write-first on same-cycle read/write due to registered read address.

- Handshake style caveat
- Wishbone slave returns edge pulses for wb_ack_o/wb_err_o; some classic masters expect level-held ACK—verify compatibility.

IO PORTS

-- Clocks and Resets --

- Clock domains
- wb_clk_i (system/Wishbone): Hosts register file (eth_registers), bus master/DMA and BD RAM control (eth_wishbone), MIIM controller base clock (eth_miim.Clk), interrupt aggregation, and all bus-side handshakes.
- mtx_clk_pad_i (TX MII): Drives transmit datapath (eth_txethmac), MAC control insertion/arbitration (eth_maccontrol), TX-side status latching (eth_macstatus TX), CSMA/CD timing, and TX IPG logic.
- mrx_clk_pad_i (RX MII): Drives receive datapath (eth_rxethmac), address filtering/counters/state machines, RX-side status latching (eth_macstatus RX), RX IFG logic, pause detection and timer.
- MDC (MIIM management clock): Generated inside eth_miim by eth_clockgen from wb_clk_i with 50%

duty. Provides single-wb_clk_i-cycle strobes MdcEn/MdcEn_n for MIIM bit timing; MIIM shift/output stages update only on MdcEn_n.

- Global reset
- wb_rst_i is a global, asynchronous reset input that fans out to all modules and domains (as Reset/TX Reset/RX Reset equivalents).
- Deassertion is resynchronized per clock domain (wb_clk_i, mtx_clk_pad_i, mrx_clk_pad_i) using multi-flop synchronizers and one-shot handshakes to avoid metastability and double actions.
- System requirement: Hold wb_rst_i asserted until all involved clocks (wb_clk_i, mtx_clk_pad_i, mrx_clk_pad_i) are stable. Deassertion must meet recovery/removal timing; included synchronizers handle domain alignment.

- Module-specific reset behavior
- TX/RX MAC blocks (eth_txethmac, eth_rxethmac, eth_maccontrol, eth_macstatus): Asynchronous reset; all internal state synchronous to their respective MTxClk/MRxClk domains.
- MIIM (eth_miim and subblocks eth_clockgen, eth_shiftreg, eth_outputcontrol): Asynchronous reset; transaction start/stop aligned to MdcEn; output pipelines update only on MdcEn_n; NoPre support handled by bit counter presets.
- Registers (eth_registers): Runs in wb_clk_i; asynchronous reset; optional per-register synchronous clears (SyncReset); W1C interrupt latches clear on reset.
- Wishbone master/DMA (eth_wishbone): Spans wb_clk_i, MTxClk, MRxClk; per-domain asynchronous resets with explicit CDC handshakes for frame start/end, FIFO moves, and status/result pulses. Uses m_wb_ack_i/m_wb_err_i in wb_clk_i as "access finished" to advance state safely.
- Single-port RAM (eth_spram_256x32): rst forces read data to 0; memory contents are not cleared (software initialization required if known contents are needed).
- FIFOs (eth_fifo): Asynchronous reset clears pointers and counters. Synchronous clear safely re-bases pointers/outputs and tolerates one read/write during the clear cycle.

- Cross-domain synchronization and reset handshakes
- Use of 2–3 stage synchronizers with edge-detected one-shots for all control/status crossings among wb_clk_i, mtx_clk_pad_i, and mrx_clk_pad_i. All ack/err/done/abort indications are formed as single-cycle pulses in the destination clock.
- Representative crossings and guards:
- CarrierSense into TX: two-FF synchronize into MTxClk; external collision latch cleared by ResetCollision asserted only when TX is idle.
- Transmitting into RX: synchronized via a 3-FF chain to MRxClk to gate RX state (e.g., drops during TX in half-duplex).
- RX enable: r_RxEn is captured into MRxClk (RxEnSync) only when MRxDV=0 to avoid mid-packet changes; loopback paths use TX signals gated by RxEnSync.
- Flow control: r_TxPauseRq triple-synchronized into MTxClk to produce a clean TPauseRq pulse; WillSendControlFrame synchronized into wb_clk_i to generate RstTxPauseRq; PauseTimerEq0 from RX synchronized into MTxClk with pause updates at TX idle/finish points.
- RX abort: Latch set in MRxClk, synchronized into wb_clk_i (RxAbort_wb) and acknowledged back (RxAbortRst) to clear the MRxClk latch.
- IRQ handshakes: TX/RX flow-control and MAC event latches set in their native domains; synchronized into wb_clk_i with 3-FF chains and edge-detected; acknowledgments return to clear source-domain latches.
- MIIM requests: Starts (write/read/scan) triple-registered and edge-detected in wb_clk_i; Start/Busy aligned to MdcEn; read-data update is a one-shot at EndBusy with write-start suppression.

- Clock-local timing resources
- RX IFG counter (MRxClk): Enforces 96 bit-times minimum IFG; resets on SFD or drop; r_IFG can force IFG satisfied.

- TX IPG selection (MTxClk): Applies IPGT/IPGR2; Rule1 latch selects appropriate gap; full-duplex prefers IPGT.
- CRC engines (TX/RX domains): Nibble-rate; asynchronous reset seeds 0xFFFFFFFF; Initialize asserts at SFD/start-of-data; Enable must remain asserted over the covered region.

- Design rules and guarantees
- All domain crossings use explicit synchronizers; pulses are generated as single cycles in the destination clock to prevent multi-counting.
- Reset-driven clears that span domains use request/ack pairs; source latches clear only after destination acknowledges.
- Collision windows and duplex masking are handled entirely within their native clock domains.
- No assumption of memory clearing on reset; software should initialize BD RAM as needed.

-- Wishbone Slave Interface --

A 32-bit Wishbone slave in the wb_clk_i domain that exposes MAC control/status registers and buffer-descriptor/internal SRAM to the CPU with clean acknowledge/error signaling and per-byte write enables.

Ports
- Inputs: wb_clk_i, wb_rst_i (async), wb_adr_i, wb_dat_i[31:0], wb_sel_i[3:0], wb_cyc_i, wb_stb_i, wb_we_i.
- Outputs: wb_dat_o[31:0], wb_ack_o, wb_err_o, int_o.

Address decode and map (wb_adr_i[11:10])
- 2'b00: Register space via eth_registers. Lower word address selects a MAC register (e.g., MODER..TXCTRL, 0x00..0x16). Unmapped register addresses read as 0.
- 2'b01: Buffer-descriptor/internal SRAM via eth_wishbone. 256x32 single-port RAM indexed by wb_adr_i[9:2].
- 2'b1x: Address miss; transaction terminates with wb_err_o.

Transfer acceptance and byte enables
- A transaction is considered valid when wb_cyc_i & wb_stb_i are high and ByteSelected = |wb_sel_i is non-zero.
- Write strobes: wb_sel_i[3:0] are per-byte enables. For register writes, lanes are honored per byte and eth_registers enforces field semantics (e.g., RW1C for INT_SOURCE, MIIRX_DATA not CPU-writable, TX_BD_NUM constrained to <= 0x80). For BD/SRAM writes, wb_sel_i passes directly to the RAM.
- Word alignment is assumed; data lanes map to the 32-bit word selected by wb_adr_i.

Read data mux
- On reads (~wb_we_i): wb_dat_o returns RegDataOut for register space; returns BD/SRAM RAM data for BD/SRAM space.

Acknowledge and error generation
- wb_ack_o: one-cycle pulse per accepted transaction. Asserted on a decoded register access or when BD/SRAM access completes (BDAck). Ack pulses are self-masked to remain single-cycle.
- wb_err_o: one-cycle pulse when wb_stb_i & wb_cyc_i and either ByteSelected=0 or address miss (wb_adr_i[11:10]=2'b1x). Ack and err are mutually exclusive.
- Data on wb_dat_o is valid only in cycles where wb_ack_o is asserted.

Interrupt output
- int_o is sourced by eth_registers: OR of latched INT_SOURCE bits masked by INT_MASK. IRQ bits

are read/write-1-to-clear (RW1C) via the register interface.

Reset behavior
- wb_rst_i asynchronously resets the slave interface and submodules: wb_ack_o/wb_err_o clear, registers revert to reset defaults, and subsequent transactions adhere to the address/byte-enable rules.

Clocking and domain
- All slave-interface logic and submodule register storage are in the wb_clk_i domain. The MAC's separate Wishbone master for DMA is independent and does not affect this slave interface.

-- Wishbone Master Interface (DMA) --

Purpose
- A 32-bit Wishbone master (DMA) moves Ethernet frame data between system memory and the MAC FIFOs. Implemented in eth_wishbone, driven under WB_CLK_I, reset by wb_rst_i.

Interface signals (master side)
- m_wb_adr_o[31:2]: word-aligned address; eth_top forces {addr_msb, 2'b00}.
- m_wb_dat_o[31:0]: write data (RX path: MAC → memory).
- m_wb_dat_i[31:0]: read data (TX path: memory → MAC).
- m_wb_sel_o[3:0]: byte enables; TX reads use 4'b1111; RX writes use alignment-aware masks.
- m_wb_we_o: 0 for reads (TX), 1 for writes (RX).
- m_wb_cyc_o: asserted for the duration of a transaction.
- m_wb_stb_o: equals m_wb_cyc_o; one beat per acknowledge.
- m_wb_ack_i: handshake; each asserted ack completes one beat and advances address/length.
- m_wb_err_i: error; terminates the active transaction immediately.
- m_wb_cti_o[2:0]: 3'b010 for incrementing bursts; 3'b111 on the final beat of a burst/transaction.
- m_wb_bte_o[1:0]: 2'b00 (linear burst).

Protocol and timing
- Classic/incrementing burst Wishbone. The master keeps cyc/stb asserted across beats; each beat completes on ack. No address or data change is considered valid until ack_i.
- On the last beat of any planned sequence, cti_o is driven to 3'b111 to end the burst cleanly; otherwise 3'b010 during multi-beat bursts.

Addressing and alignment
- All master addresses are word-aligned on m_wb_adr_o. RX starting byte offset (RxPointerLSB) is honored via m_wb_sel_o:
- LSB=0 → 1111, LSB=1 → 0111, LSB=2 → 0011, LSB=3 → 0001.
- Subsequent RX words typically use 1111 once alignment is satisfied. TX always reads full words (sel=1111); byte steering to the MAC TX path is handled internally.

Burst policy
- TX: 4-beat incrementing bursts when TX FIFO has headroom (txfifo_cnt < TX_FIFO_DEPTH – 4) and remaining TxLength > ~20 bytes; otherwise single-beat cycles.
- RX: incrementing bursts when RX FIFO occupancy supports it (≥4 bytes or continuing an active burst); otherwise single-beat writes.

TX operation (memory → MAC TX FIFO)
- m_wb_we_o=0, m_wb_sel_o=1111. Address sourced from TxPointerMSB; incremented once per ack_i. TxLength is decremented per beat with initial adjustment for TxPointerLSB. Transaction ends when all valid bytes are fetched or upon err_i.

RX operation (MAC RX FIFO → memory)
- m_wb_we_o=1. m_wb_sel_o derived from RxPointerLSB for the first word; subsequent words usually 1111. Address sourced from RxPointerMSB; incremented per ack_i. Assembles 32-bit words from RX FIFO; on end-of-frame, flushes partial word with zero padding and masks unused lanes via sel.

Completion, errors, and reset
- A transaction ends on planned completion (length/frame done) or any m_wb_err_i. Errors are reflected in descriptor/status logic. wb_rst_i resets engine state and deasserts cyc/stb; cti defaults idle (not driven valid when cyc=0).

Arbitration and increments
- A single DMA engine arbitrates between TX and RX transactions. Address increments and length counters are qualified by MasterAccessFinished (ack_i or err_i) to avoid double increments.

-- MII Transmit Interface --

Function: Drives the IEEE 802.3 MII transmit interface toward the PHY in the TX MII clock domain.
Clocking and signals
- Clock: mtx_clk_pad_i (MTxClk ~25 MHz for 100 Mb/s).
- Outputs: MTxD[3:0] (nibble data), MTxEn (transmit enable), MTxErr (transmit error), all registered on MTxClk to meet PHY timing.
- Inputs (CSMA/CD): mcrs_pad_i (carrier sense), mcoll_pad_i (collision), synchronized into TX domain; gated off in full-duplex.
Framing and serialization
- Nibble-wide, LSB-first serialization aligned to MTxClk; for each byte, lower nibble then upper nibble.
- Preamble/SFD: Generates 0x55 nibbles followed by 0xD to form 0x55..55 D5; MTxEn asserted across preamble, SFD, data/pad, FCS, and jam.
- Data phase: Streams payload bytes from TX FIFO; optional padding inserted to satisfy configured minimum frame length.
FCS (CRC-32)
- CRC computed over payload and pad only (preamble/SFD excluded).
- Transmitted FCS is ones' complement of the running CRC remainder, emitted LSB-first, nibble-by-nibble (starting from Crc[31:28] down per design mapping).
- DlyCrcEn affects internal timing only; wire format is unchanged.
CSMA/CD and duplex behavior
- Half-duplex: Implements IEEE 802.3 CSMA/CD.
- Early collision: Transmits jam sequence (MTxD=0x9 for 8 nibbles), then performs binary exponential backoff and retries.
- Late collision: Forces abort with no retry.
- Full-duplex: CSMA/CD disabled (carrier/collision ignored).
Flow control and control-frame insertion
- Supports 802.3x PAUSE frame injection when TxFlow enabled.
- Control frames seize the TX path, transmit fixed DA 01-80-C2-00-00-01, SA, Ethertype 0x8808, opcode 0x0001, and pause time; pad/CRC forced as needed.
- While a received PAUSE timer is active, normal frame starts are blocked.
Data sourcing and underrun handling
- Consumes bytes from the TX FIFO; TxUsedData strobes mark consumption at byte/nibble boundaries.
- Underrun (FIFO empty when data required) asserts MTxErr and aborts the frame.
Frame length and timing constraints
- Enforces minimum/maximum frame length per configuration; pads to minimum length when enabled; flags TooBig oversize conditions (asserts MTxErr during frame).

- Inter-Packet Gap enforced per configuration (r_IFG) before starting next frame.
Reset, CDC, and loopback
- Global async reset (wb_rst_i) initializes TX state to idle with MTxEn/MTxErr deasserted.
- All cross-domain inputs (carrier, collision, pause requests) are safely synchronized into MTxClk domain.
- r_LoopBck provides internal loopback of TX to RX paths only; MII TX outputs to the PHY are unaffected.
Compliance
- Conforms to 4-bit MII transmit conventions: LSB-first nibble order; MTxEn qualifies all transmitted nibbles including jam; outputs registered to meet PHY setup/hold.

-- MII Receive Interface --

Connects the PHY MII receive pins to the internal RX pipeline and defines sampling, qualification, and timing. Signals: mrx_clk_pad_i (MRxClk), mrxd_pad_i[3:0] (MRxD[3:0]), mrx_dv_pad_i (MRxDV), mrx_err_pad_i (MRxErr). Sampling occurs in the MRxClk domain; data nibbles are captured per MII convention with the high nibble first. Loopback: when r_LoopBck=1, MRxD/MRxDV/MRxErr are sourced from the internal TX MII (MTxD/MTxEn/MTxErr); otherwise they come from the pads. RX enable gating: MRxDV and MRxErr are AND-gated with RxEnSync (r_RxEn synchronized into MRxClk and only captured when MRxDV is low) to ensure clean RX disable without mid-packet glitches. Preamble/SFD recognition: the RX state machine hunts for 0x55…55 D5; once SFD is found, nibbles are assembled into bytes. Byte alignment and strobes: RxValid asserts on byte boundaries, RxStartFrm and RxEndFrm are single-cycle pulses aligned with RxData, and dribble nibble (half-byte) ends are detected to produce a clean RxEndFrm. IFG enforcement: an internal counter enforces a minimum 96 bit-times idle between frames; r_IFG can override IFG satisfaction. Half-duplex interlock: when r_FullD=0, a Transmitting flag (synchronized into MRxClk) prevents starting reception; if MRxDV asserts while Transmitting and the RX FSM is idle, the frame is dropped. Error qualification: MRxErr is latched only when MRxDV is high in valid RX states; InvalidSymbol (nibble 0xE) is detected and handled specially in abort logic. CRC checking: a nibble-wide CRC-32 processes data LSB-first; Initialize_Crc asserts at SFD, Enable_Crc remains high during the data field, residue 0xC704DD7B indicates a good FCS, and CrcError flags a bad FCS. Length control: byte counters track position, enforce min/max frame length (with r_HugEn and r_RecSmall options), and can force early drop on over-length. Clock-domain crossing hygiene: all control and status signals crossing in or out of MRxClk (e.g., Transmitting, Pause status, abort/status strobes) use multi-flop synchronizers or explicit handshake pulses.

-- MDIO/MDC Management Interface --

IEEE 802.3 Clause 22 MIIM master for PHY management over MDC/MDIO. Supports single read, single write, and periodic scan of PHY registers with programmable MDC rate and optional preamble omission. External interface: MDC is a 50%-duty clock generated by a divider of Clk; MDIO is bidirectional, driven by the master during preamble and control fields, and tri-stated during read turnaround and data; MDI samples MDIO during reads. Register interface (via eth_registers): MIIMODER (0x0A) provides r_MiiNoPre (preamble omit) and r_ClkDiv (MDC divider); MIICOMMAND (0x0B) provides r_WCtrlData, r_RStat, r_ScanStat one-shot requests; MIIADDRESS (0x0C) holds r_FIAD[4:0] and r_RGAD[12:8]; MIITX_DATA (0x0D) holds r_CtrlData[15:0]; MIIRX_DATA (0x0E) captures read/scan data at completion; MIISTATUS (0x0F) reports NValid_stat (scan pending/aligned), Busy_stat (overall busy), and LinkFail (active-high). Operations are edge-detected and aligned to MdcEn; one transaction at a time. Busy asserts conservatively during requests, operations, completion, and while a scan is pending; EndBusy pulses for one Clk at operation end; MIIRX_DATA updates only at end of a read/scan and is suppressed if a write starts the same cycle. MDC generation: divider N = max(r_ClkDiv, 2); frequency = Clk / (2 * floor(N/2)); odd N rounds down to maintain 50% duty;

MdcEn/MdcEn_n strobe on MDC edges. Frame timing: 64-bit transaction including optional 32-bit preamble; no-preamble mode jumps the bit counter to skip preamble; byte load strobes at fixed counts; read data bytes latched near counts 0x37 and 0x3F. Serialization/deserialization: shift MSB-first; sample MDI on MdcEn_n; assemble Prsd[15:0]; derive LinkFail when reading PHY reg 1 (Basic Status). Output control: Mdo/MdoEn pipelined to MdcEn_n; write drives MDIO throughout; read tri-states MDIO after control fields (after bit 45) through turnaround and data. Sequences: write uses ST=01, OP=01, TA=10; read uses ST=01, OP=10 with master tri-state for TA/data; scan mirrors read, started when idle and aligned to MdcEn. Reset clears pipelines, counters, Busy, Mdo/MdoEn, Prsd, and LinkFail. Guards: divider clamped ≥2; scan synchronized to MdcEn and blocked while busy; UpdateMIIRX_DATAReg gated against simultaneous write start. Integration: MDIO pad driven as mdio = MdoEn ? Mdo : Z; MIIRX_DATA is written only by MIIM; commands self-clear on EndBusy; software polls Busy_stat or EndBusy; MIISTATUS upper bits read as 0; preamble omission requires PHY support.

-- Interrupts --

eth_top provides a single consolidated interrupt output (int_o) generated and controlled by eth_registers. Two registers define behavior: INT_SOURCE (0x01, RW1C) and INT_MASK (0x02, R/W). Latches for INT_SOURCE are synchronous to wb_clk_i and asynchronously cleared by reset. int_o is a level output defined as OR over (INT_SOURCE[i] & INT_MASK[i]) for bits [6:0]. Masking a bit prevents it from contributing to int_o but does not clear the corresponding latch.

INT_SOURCE bit mapping (bits [6:0]):
- [0] TXB: Transmit buffer event without error
- [1] TXE: Transmit error event
- [2] RXB: Receive buffer event without error
- [3] RXE: Receive error event
- [4] BUSY: RX not ready when a frame started
- [5] TXC: Transmit control (PAUSE) event complete
- [6] RXC: Receive control (PAUSE) event
Unused bits read as 0 and ignore writes.

Generation and classification:
- TXB/TXE (eth_wishbone): Asserted on TxStatusWrite completion when the buffer descriptor IRQ enable bit (BD status[14]) is set. Classified as TXE if TxError=1, else TXB. TxError is OR of: TxUnderRun, RetryLimit (MaxCollisionOccured), LateCollLatched, CarrierSenseLost.
- RXB/RXE (eth_wishbone): Asserted on RxStatusWrite completion when BD status[14] is set and internal pause-related gating permits. Classified as RXE if RxError=1, else RXB. RxError is the OR of configured RX error flags (e.g., CRC error, length violations per RecSmall/HugEn policy, invalid symbol, dribble nibble, overrun, late collision in half-duplex, address miss).
- BUSY (eth_wishbone): Asserted when a receive start occurs while the RX engine was not ready.
- TXC (eth_maccontrol/transmit control): Asserted when a PAUSE control frame transmission completes while TX flow control is enabled. Generated in MTxClk, synchronized into wb_clk_i via a multi-flop (3-FF) edge-detect pipeline; clearing INT_SOURCE[5] handshakes back to release the TX-domain set.
- RXC (eth_receivecontrol): Asserted on a valid received PAUSE control event while RX flow control is enabled. Generated in MRxClk, synchronized into wb_clk_i via a multi-flop (3-FF) edge-detect pipeline; clearing INT_SOURCE[6] handshakes back to release the RX-domain set.

Signaling and software model:
- int_o is asserted while any enabled source is latched and deasserts only when software clears all asserted sources (write 1 to INT_SOURCE bits) or on reset.
- Multiple sources may be asserted simultaneously; software should read INT_SOURCE to identify

active sources, then write 1s to those bits to clear. INT_MASK enables reporting per bit without affecting latched state.

Clock and CDC considerations:
- All INT_SOURCE latches reside in wb_clk_i domain. Events generated in MTxClk/MRxClk domains are converted to single-cycle pulses in wb_clk_i using multi-flop synchronizers and explicit edge detection, with RW1C clear handshakes to the source domains.

Out-of-scope signals:
- Wishbone bus acknowledge/error (wb_ack_o/wb_err_o) are independent of the interrupt controller.
- MIIM/MDIO does not generate an interrupt; status is polled via MIISTATUS (Busy, NValid, LinkFail).

-- Cross-Domain Handshake Signals --

Scope: Defines all cross-domain handshakes between wb_clk_i (Wishbone/system), MTxClk (TX MII), and MRxClk (RX MII). Handshakes use a source-side set/latch, multi-flop synchronization into the destination, edge-pulse generation for one-shot actions, and an explicit return acknowledge/clear synchronized back to the source to prevent metastability and double-counting.

CDC policy:
- Synchronization depth: 3 flip-flops for IRQ/critical handshakes; 2–3 flip-flops otherwise, per module.
- Pulse formation: destination pulse = sync2 & ~sync3, yielding one destination-clock cycle.
- Clears/acks: returned as levels from the destination domain, re-synchronized to the source domain to release the source latch.

Handshakes:
1) TX Pause request (WB -> TX, with completion ack TX -> WB)
- Source (WB): r_TxPauseRq asserted by software when r_TxFlow=1.
- Destination (MTxClk): r_TxPauseRq is triple-synchronized and edge-detected to produce TPauseRq (1-cycle pulse) to start PAUSE control-frame transmission.
- Completion ack (TX -> WB): WillSendControlFrame is triple-synchronized into wb_clk_i and edge-detected to generate RstTxPauseRq, which clears the WB-side pause request bit; guarantees exactly one PAUSE frame per request.

2) TX flow-control IRQ (TX -> WB with WB ack -> TX)
- Source (MTxClk): SetTxCIrq_txclk asserted by TX logic (set/latch).
- Destination (wb_clk_i): triple-synchronized, edge-detected to SetTxCIrq (one-shot IRQ pulse to WB domain).
- Ack/clear (WB -> TX): ResetTxCIrq driven by WB logic; synchronized back (2–3 FF) to MTxClk to clear the TX-side latch; ensures one interrupt per TX event.

3) RX flow-control IRQ (RX -> WB with WB ack -> RX)
- Source (MRxClk): SetRxCIrq_rxclk asserted by RX logic (set/latch).
- Destination (wb_clk_i): triple-synchronized, edge-detected to SetRxCIrq (one-shot IRQ pulse to WB domain).
- Ack/clear (WB -> RX): ResetRxCIrq driven by WB logic; synchronized back (2–3 FF, commonly 3) to MRxClk to clear the RX-side latch; ensures one interrupt per RX event.

4) RX abort handshake (RX -> WB with WB clear -> RX)
- Source (MRxClk): RxAbort_latch asserted on receive abort conditions.
- Destination (wb_clk_i): synchronized to form RxAbort_wb (level or event for status/ISR in WB domain).
- Clear (WB -> RX): RxAbortRst_wb asserted by WB logic; synchronized back (2–3 FF) as RxAbortRst

to MRxClk to release RxAbort_latch; prevents repeated abort reports.

Notes and constraints:
- All destination pulses are exactly one destination clock cycle and must be consumed by edge-sensitive logic; no multi-cycle pulse stretching is provided across the boundary.
- Source-side latches remain asserted until the returned clear is observed in the source domain, guaranteeing no lost events and preventing double-counting.
- Non-handshake CDCs (one-way pulses/levels) such as TxDone/Abort/Retry, TxUnderRun, WriteRxDataToFifo, ShiftEnded, PauseTimerEq0, and WillTransmit are synchronized but are not part of this handshake specification.

-- Configuration/Status Outputs (r_* Signals) --

- Overview
- r_* are configuration/status outputs produced by eth_registers in the Wishbone clock domain (wb_clk_i). They are readback-visible; upper/unused bits of many registers read back as 0. All r_* are synchronized before use in MTxClk/MRxClk (and MDC for MIIM).

- Core mode/configuration (MODER)
- r_RecSmall: Allow RX of undersized frames.
- r_Pad: Enable TX padding to minimum length.
- r_HugEn: Permit oversized frames beyond r_MaxFL.
- r_CrcEn: Enable TX FCS generation.
- r_DlyCrcEn: Enable delayed-CRC feature.
- r_FullD: Full-duplex; disables CSMA/CD paths.
- r_ExDfrEn: Mask excessive defer signaling.
- r_NoBckof: Disable randomized backoff (half-duplex).
- r_LoopBck: Internal MII loopback (RX uses TX signals).
- r_IFG: Force RX IFG satisfied.
- r_Pro: Promiscuous mode (accept any DA).
- r_Iam: Individual address mode (address/mode control).
- r_Bro: Disable broadcast acceptance.
- r_NoPre: Suppress TX preamble.
- r_TxEn: TX enable, AND-gated by r_TxBDNum>0.
- r_RxEn: RX enable, AND-gated by r_TxBDNum<0x80.

- Inter-packet gap (IPGT/IPGR1/IPGR2)
- r_IPGT[6:0]: Back-to-back IPG timing.
- r_IPGR1[6:0], r_IPGR2[6:0]: General IPG timing (Rule1 selection).

- Frame length limits (PACKETLEN)
- r_MinFL[15:0]: Minimum frame length (incl. FCS) for TX/RX.
- r_MaxFL[15:0]: Maximum frame length; disabled if r_HugEn=1.

- Collision/backoff (COLLCONF)
- r_MaxRet[3:0]: Max retransmissions before abort.
- r_CollValid[5:0]: Early/late collision boundary window.

- Descriptor partitioning (TX_BD_NUM)
- r_TxBDNum[7:0]: TX BD count (0..0x80); gates r_TxEn/r_RxEn. Writes constrained to <=0x80.

- Flow control (CTRLMODER/TXCTRL)
- r_TxFlow: Enable TX PAUSE generation.

- r_RxFlow: Enable RX PAUSE handling.
- r_PassAll: Pass all control frames.
- r_TxPauseTV[15:0]: PAUSE quanta value.
- r_TxPauseRq: Software PAUSE request; honored only if r_TxFlow=1. Auto-cleared when PAUSE frame transmission completes.

- MII management (MIIMODER/MIICOMMAND/MIIADDRESS/MIITX_DATA)
- r_MiiNoPre: Suppress MDIO preamble.
- r_ClkDiv[7:0]: MDC divider for eth_clockgen.
- r_WCtrlData: MIIM write request.
- r_RStat: MIIM single read request.
- r_ScanStat: MIIM continuous scan request.
- r_RGAD[4:0]: PHY register address.
- r_FIAD[4:0]: PHY address.
- r_CtrlData[15:0]: MIIM write data payload.

- Addressing/filtering
- r_MAC[47:0]: Station MAC address (MAC_ADDR1/MAC_ADDR0).
- r_HASH0[31:0], r_HASH1[31:0]: 64-bit multicast hash filter for RX address check.

- Synchronization/consumption
- TX domain consumes: r_TxEn, r_FullD, r_NoBckof, r_ExDfrEn, r_IPGT/IPGR1/IPGR2, r_MinFL/r_MaxFL, r_MaxRet, r_CollValid, r_Pad, r_CrcEn, r_DlyCrcEn, r_TxFlow, r_TxPauseTV, r_TxPauseRq.
- RX domain consumes: r_RxEn, r_RecSmall, r_HugEn, r_MinFL/r_MaxFL, r_Pro, r_Iam, r_Bro, r_PassAll, r_HASH0/1, r_MAC, r_IFG, r_RxFlow, r_LoopBck.
- MIIM/MDC domain consumes: r_ClkDiv, r_MiiNoPre, r_WCtrlData, r_RStat, r_ScanStat, r_RGAD, r_FIAD, r_CtrlData.

- Side effects and constraints
- r_TxEn/r_RxEn are internally gated by r_TxBDNum to enforce a valid TX/RX BD split.
- r_TxPauseRq is synchronized into TX, converted to a one-shot, and auto-cleared in WB domain on completion.
- MIIM command bits are edge-detected and aligned to MDC; MIIRX_DATA is updated only by MIIM (not CPU-writable).
- Readback upper/unused bits are zeroed per register definition.

-- Test/Debug Signals --

Scope and usage
- Provides software-visible status and internal observability to aid bring-up, verification, and in-system debug across three domains: wb_clk_i (system), mtx_clk_pad_i (TX/MII), mrx_clk_pad_i (RX/MII).
- Many indications are single-cycle one-shots in native domains; rely on status latches (eth_macstatus, INT_SOURCE) or synchronize in testbenches.

Software-visible debug/status (eth_registers)
- INT_SOURCE/INT_MASK: RW1C interrupt latches/masks per source (TXB, TXE, RXB, RXE, BUSY, TXC, RXC); int_o is the masked OR for system-level debug.
- MIISTATUS: [2]=NValid_stat, [1]=Busy_stat, [0]=LinkFail for MIIM/PHY checks.
- MIIRX_DATA/MIITX_DATA: Read-back/write data for MDIO transaction verification.
- MIICOMMAND/MIIMODER/MIIADDRESS: Command/divider/address programming with readable control state.

- TXCTRL: r_TxPauseRq, r_TxPauseTV to trigger/parameterize PAUSE-frame tests.
- MODER/CTRLMODER: Readbacks of r_LoopBck, r_TxEn/r_RxEn, r_TxFlow/r_RxFlow, and other feature gates.
- dbg_dat at 0x16: implementation-defined viewport for tapping internal signals.

Top-level hooks and built-in modes
- Internal MII loopback: r_LoopBck routes TX MII (mtxd/mtxen/mtxerr) to RX; RxEnSync gates MRxDV/Rx when disabled.
- Duplex observability: CarrierSense and Collision synchronized into TX; Transmitting synchronized into RX for TX/RX correlation.

MIIM/MDIO debug (eth_miim)
- Lifecycle: Busy, EndBusy (one-shot), UpdateMIIRX_DATAReg (one-shot).
- Timing: MdcEn/MdcEn_n (bit-time strobes), BitCounter[6:0] (frame index), ByteSelect[3:0], LatchByte[1:0].
- MDIO IO: Mdo, MdoEn (drive/tri-state), read turnaround visibility.
- Link: LinkFail (raw link indication latched on MIIRX completion).

TX-path observability
- On-wire: MTxEn, MTxD[3:0], MTxErr.
- End conditions: TxDone, TxRetry, TxAbort and internal StartTxDone/Retry/Abort one-shots.
- Collision/backoff: WillTransmit, ResetCollision, LateCollision, MaxCollisionOccured, RetryCnt.
- Control frames (PAUSE): CtrlMux, SendingCtrlFrm, TxCtrlStartFrm, TxCtrlEndFrm, WillSendControlFrame, BlockTxDone.

RX-path observability
- Stream/framing: RxData[7:0], RxValid, RxStartFrm, RxEndFrm.
- Address/filter: Broadcast, Multicast, UnicastOK/BroadcastOK/MulticastOK, CrcHash[5:0], CrcHashGood.
- Preamble/IFG: MRxDEq5, MRxDEqD, IFGCounterEq24.
- Length/timing: ByteCntEq*/Range flags, ByteCntMaxFrame, DlyCrcEn, DlyCrcCnt.
- Flow control (RX-driven pause): ReceivedPauseFrm, ReceivedPauseFrmWAddr, LatchedTimerValue, SetPauseTimer, PauseTimer, Pause (TX-domain reflect).
- Address outcomes: RxAbort (one-shot) and AddressMiss latches for DA filter diagnostics.

Consolidated status latches (eth_macstatus)
- RX domain: ReceivedPacketGood, ShortFrame, ReceivedPacketTooBig, LatchedMRxErr, InvalidSymbol, DribbleNibble, RxLateCollision.
- TX domain: RetryCntLatched, RetryLimit, LateCollLatched, DeferLatched, CarrierSenseLost.
- Sampling pipeline: TakeSample, LoadRxStatus, ReceiveEnd to align end-of-frame measurements.

DMA/Wishbone engine debug (eth_wishbone)
- Interrupts: TxB_IRQ, TxE_IRQ, RxB_IRQ, RxE_IRQ, Busy_IRQ (mirror INT_SOURCE causes).
- Transaction strobes: TxStatusWrite, RxStatusWrite (descriptor updates), ReadTxDataFromMemory, WriteRxDataToMemory (data moves).
- Bursts/handshake: tx_burst_en, rx_burst_en, burst counters; m_wb_cti_o (010 inc / 111 last), m_wb_bte_o=00; m_wb_ack_i/m_wb_err_i observation.
- Errors/flow: TxUnderRun_wb, RxOverrun; BD Ready/Wrap decisions; pointer/length handling and byte select alignment.

FIFO health (eth_fifo)
- Flags: empty, almost_empty, full, almost_full; depth counter (cnt) for backpressure and throughput

tuning.

Simulation-only hooks
- Initial block SystemCalls in eth_top print trace messages (non-synthesizable) to correlate with waveforms.

Operational cautions
- One-shots include: SetTxCIrq/SetRxCIrq, EndBusy, UpdateMIIRX_DATAReg, TxStatusWrite, RxStatusWrite, RxAbort, TxDone/TxRetry/TxAbort. Use latches or synchronizers to observe cross-domain.
- Feature gates affect behavior: r_IFG (IFG satisfied), r_HugEn (oversize handling), r_NoPre (MIIM preamble skip), r_Pro (promiscuous). Account for configuration when interpreting signals.
- dbg_dat taps and internal counters/comparators (e.g., IFGCounterEq24, ByteCntEq*, BitCounter) are implementation-defined; document selection and domain when exposing.

ARCHITECTURE

-- Top-Level Block Composition --

eth_top integrates the Ethernet MAC, DMA, and control subsystems across three clock domains (wb_clk_i, mtx_clk_pad_i, mrx_clk_pad_i) with wb_rst_i reset. Top-level blocks and roles:
- eth_registers: CPU-visible register file and field decode (r_*), MIIM command/response registers, TX/RX enable gating (incl. TX_BD_NUM partition), maskable interrupt controller, and RW1C status aggregation.
- eth_wishbone: Buffer-descriptor ring and internal SRAM, Wishbone master for TX (reads) and RX (writes), TX/RX FIFOs, alignment/byte steering, clock-domain crossings, DMA control, and IRQ generation.
- eth_txethmac: Transmit MAC datapath and state machines (nibble/byte timing, CSMA/CD jam/backoff/retry), CRC generation, IFG handling, and per-frame status/error flags.
- eth_rxethmac: Receive MAC datapath (nibble-to-byte assembly), CRC checking, address filtering (unicast/multicast with hash), start/end/valid generation, dribble and symbol error handling, and per-frame status.
- eth_maccontrol: IEEE 802.3x flow control arbitration between normal TX and PAUSE frames; enforces Pause gating on TxStart; contains eth_transmitcontrol (PAUSE frame generation) and eth_receivecontrol (PAUSE detection/timer).
- eth_miim: MDIO/MDC (MIIM) controller providing MDC generation, MDIO serialization/deserialization, read/write/scan sequencing, and Busy/Done strobes to registers; composed of eth_clockgen, eth_shiftreg, and eth_outputcontrol.
- eth_macstatus: Consolidated RX/TX status latching across domains for interrupt and software consumption.
External interfaces and routing:
- Wishbone CPU slave terminates in eth_top and decodes wb_adr_i[11:10] to eth_registers (00) or eth_wishbone (01); read-data mux, ack/err pulse generation, and byte-lane enforcement are handled in eth_top.
- Wishbone master signals are driven by eth_wishbone (word-aligned addresses, classic/burst sequencing) toward system memory.
- MII pads connect TX (eth_txethmac via eth_maccontrol) and RX (eth_rxethmac); internal loopback (r_LoopBck) can route TX MII into RX; RX enable is gated and synchronized.

- MDIO pads are driven/observed by eth_miim (MDC and tri-stated MDIO per Mdo/MdoEn protocol).
Hierarchy (leaf blocks):
- eth_txethmac includes eth_txstatem, eth_txcounters, eth_crc, eth_random.
- eth_rxethmac includes eth_rxstatem, eth_rxcounters, eth_rxaddrcheck, eth_crc.
- eth_wishbone includes eth_spram_256x32 and parameterized eth_fifo.
- eth_registers includes multiple eth_register instances for byte-lane writeable storage.

-- Address Decode and Bus Error Handling --

- Address decode: wb_adr_i[11:10] selects target region. 00 -> register space (eth_registers via RegCs[3:0]); 01 -> Buffer Descriptor/Internal SRAM (eth_wishbone via BDCs[3:0]); 1x -> unmapped, treated as chip-select miss (CsMiss).
- Byte-lane validation: ByteSelected = OR-reduction of wb_sel_i. If ByteSelected == 0, the access is invalid and generates a bus error.
- Read-data muxing: On reads (~wb_we_i) with a register-space hit (|RegCs), wb_dat_o <= RegDataOut; otherwise (BD/SRAM space or writes) wb_dat_o <= BD_WB_DAT_O. Write cycles acknowledge per region; wb_dat_o is not meaningful on writes.
- Acknowledge and error generation:
- temp_wb_ack_o = (|RegCs) | BDAck; converted to a single-cycle wb_ack_o with edge/pulse logic.
- temp_wb_err_o = (wb_stb_i & wb_cyc_i) & (~ByteSelected | CsMiss); converted to a single-cycle wb_err_o with edge/pulse logic.
- wb_ack_o and wb_err_o are mutually exclusive for any request; no ack is asserted on error cycles.
- Unmapped/invalid accesses: Addresses with wb_adr_i[11] == 1 (the "1x" regions) or with wb_sel_i == 0 cause wb_err_o; no wb_ack_o is issued. Read data during an error is undefined per protocol and not driven to a valid payload.
- Register-space writes: eth_registers applies per-byte write enables internally; reads of unused/upper bits return 0 per register definition.
- BD/SRAM CPU access path: eth_wishbone arbitrates CPU requests with TX/RX engines; its WB_ACK_O maps to BDAck and its data path to BD_WB_DAT_O. Internal SRAM supports per-byte write enables.
- DMA master error handling: m_wb_err_i or m_wb_ack_i terminates a DMA transfer (MasterAccessFinished). m_wb_err_i aborts the burst and is reflected into BD/status/IRQ outcomes for TX/RX. Master addresses are word-aligned (m_wb_adr_o = {m_wb_adr_tmp, 2'b00}); CTI uses incrementing bursts (010) and end-of-burst (111).
- Pulse shaping and exclusivity: Ack and error outputs are self-masked one-shot pulses, independent of the hold time of request/miss conditions, ensuring clean single-cycle responses and preventing simultaneous assertion.

-- Data Path Multiplexing and Ack/Err Generation --

Wishbone slave data-path and handshake logic are partitioned by address decode on wb_adr_i[11:10]. Addresses 00 select register space, served by eth_registers; 01 select buffer-descriptor/internal SRAM space, served by eth_wishbone; 1x is a decode miss. Read-data is multiplexed so that wb_dat_o is driven by RegDataOut when in register space and ~wb_we_i; otherwise it is driven by BD_WB_DAT_O from the BD/SRAM path. This guarantees a single active source per access. Byte lane validity is checked via ByteSelected = |wb_sel_i; invalid patterns are treated as errors.

Acknowledgement and error are generated as single-clock pulses in the wb_clk_i domain. temp_wb_ack_o asserts for a valid hit (any register chip-select active) or when the BD/SRAM path returns BDAck. temp_wb_err_o asserts when wb_stb_i & wb_cyc_i & (~ByteSelected | address decode miss). Both are converted to one-shot outputs (wb_ack_o, wb_err_o) using edge-detect/self-masking to prevent repeated pulses while stb/cyc remain asserted. Err and Ack are mutually exclusive per access,

with Err taking precedence on invalid byte selects or address misses.

Inside eth_wishbone, a single-port BD RAM is time-multiplexed among CPU (Wishbone), TX engine, and RX engine via WbEn/TxEn/RxEn arbitration that selects ram_addr/we/oe. CPU data-out (WB_DAT_O) is sourced from ram_do when WbEn & oe. BD-space WB_ACK_O is generated as a one-cycle pulse aligned to BDRead/BDWrite completion and the RAM's registered latency (using WbEn/WbEn_q).

Additional data-path multiplexing occurs in other clock domains: eth_maccontrol (MTxClk) arbitrates normal TX data versus MAC control (PAUSE) frames, gating TxStart/UsedData and forcing Pad/CRC during control insertion; TxDone/TxAbort are edge-shaped pulses. On receive (mrx_clk_pad_i), inputs are multiplexed between external PHY and internal TX for loopback, with MRxDV/MRxErr gated by RxEnSync. The MIIM path (Clk domain) tri-states MDIO via MdoEn, driving preamble/ShiftedBit during writes and releasing the line during read turnaround/data. Integrators must treat wb_ack_o and wb_err_o as single-cycle pulses and sample them accordingly.

-- Buffer Descriptor RAM and Arbitration --

Memory map and organization
- Buffer Descriptors (BDs) and internal SRAM share a single-port 256x32 RAM (eth_spram_256x32) clocked by WB_CLK_I.
- Top-level decode selects this region when wb_adr_i[11:10]==01; the word index is WB_ADR_I[9:2].
- CPU/Wishbone reads from this region return ram_do; writes use per-byte strobes and generate a BD-specific acknowledge.

RAM behavior
- Single-port synchronous RAM with one registered read address latency; valid data appears one WB_CLK_I cycle after ce & addr.
- Byte write enables we[3:0]; write-first semantics for enabled bytes; output driven only when ce & oe are asserted.
- rst does not clear the memory array; while rst is asserted, the read data output is forced to 0.

Arbitration (WB clock domain)
- Three contenders: CPU/Wishbone (WbEn), TX engine (TxEn), RX engine (RxEn).
- A simple arbiter in WB_CLK_I asserts exactly one of WbEn/RxEn/TxEn per cycle based on level "needed" inputs from TX/RX and the presence of a CPU BD access; no concurrent RAM access.
- Grants are mutually exclusive and may change every cycle; TX/RX request intents are synchronized into WB_CLK_I.

Address/data/control multiplexing
- ram_addr:
- WbEn: WB_ADR_I[9:2].
- TxEn: composite addressing {TxBDAddress, TxPointerRead} to step BD and pointer words.
- RxEn: composite addressing {RxBDAddress, RxPointerRead} to step BD and pointer words.
- ram_di:
- CPU writes: WB_DAT_I with byte enables we[3:0].
- TX/RX status writes: full-word data from engines (we=4'b1111) for atomic BD updates.
- ram_oe asserted for descriptor/pointer reads and CPU reads; ram_we asserted per byte for CPU writes and full-word for TX/RX status writes.

Wishbone slave handshake
- WB_ACK_O is a one-cycle pulse derived from BDRead/BDWrite and registered WbEn (WbEn_q); read acks align with the RAM's 1-cycle read latency.

- Read data path returns ram_do when in BD space; address misses and ByteSelected==0 produce WB_ERR_O per eth_top decode.

Descriptor ring interaction (address sequencing)
- TX: TxBDRead fetches the BD when Ready==1 and length>4, then TxPointerRead fetches the buffer pointer/alignment LSBs; TxStatusWrite updates BD status/length and increments TxBDAddress unless WrapTxStatusBit wraps to 0; IRQ enable (status[14]) gates TxB_IRQ/TxE_IRQ on status write.
- RX: On RX enable, RxBDAddress initializes to r_TxBDNum (partition boundary); RxBDRead occurs on enable/abort/completion of prior status write; RxPointerRead fetches destination pointer; RxStatusWrite updates BD and increments RxBDAddress unless WrapRxStatusBit wraps to r_TxBDNum; IRQ enable (status[14]) gates RxB_IRQ/RxE_IRQ.

Clock-domain crossing protections
- TX/RX "needed" flags are synchronized into WB_CLK_I; status/abort/results cross back with multi-flop synchronizers.
- Blocking flags (e.g., BlockingTxStatusWrite, BlockingTxBDRead) ensure single read/write per packet across the CDC boundary, preventing duplicates.

Timing and single-port constraints
- Control sequences account for the 1-cycle read latency: sample ram_do one WB clock after ce/addr; ack generation follows this latency.
- The single-port RAM precludes simultaneous CPU and TX/RX access; the arbiter serializes all BD/Internal SRAM cycles.

-- DMA Engine and Burst Control --

The eth_top DMA engine (eth_wishbone) performs descriptor-driven memory moves between system memory and the MAC FIFOs and enforces conservative burst policies on a classic Wishbone master. A single-port 256x32 BD RAM holds buffer descriptors and pointers; an internal arbiter grants BD RAM access among the CPU (Wishbone slave), TX DMA, and RX DMA. Descriptor rings are partitioned by TX_BD_NUM: TX traverses BD indices 0..(TX_BD_NUM-1) and wraps to 0 using the per-BD WrapTxStatusBit; RX traverses TX_BD_NUM..(N-1) and wraps using WrapRxStatusBit.

Wishbone master protocol uses word-aligned addressing with m_wb_adr_o = {addr_msbs, 2'b00}; m_wb_stb_o mirrors m_wb_cyc_o; m_wb_bte_o = 2'b00; incrementing bursts drive m_wb_cti_o = 3'b010 and terminate with 3'b111 on the last beat. MasterAccessFinished is defined as (m_wb_ack_i | m_wb_err_i) and gates all per-beat actions: address/pointer MSBs increment only on MasterAccessFinished to avoid double increments across wait states or bus errors. TX reads always assert m_wb_sel_o = 4'hF; RX writes drive m_wb_sel_o per first-beat alignment and tail-byte validity on the final beat.

Burst control targets 4-beat bursts for throughput while protecting FIFOs and alignment. TX bursts start when the TX FIFO has at least four free words and remaining frame length exceeds 20 bytes; otherwise the engine issues single or short bursts. RX bursts start when at least four bytes/words are buffered (or when continuing an in-progress burst); otherwise single-beat writes are used. Bursts may be shortened at end-of-frame or when FIFO thresholds or remaining length require; the engine sets m_wb_cti_o = 3'b111 on the final accepted beat. Bus errors count as completed beats, and the engine terminates bursts cleanly before status update.

TX DMA fetches a BD then the buffer pointer; starting alignment and length are latched. The first transfer subtracts the starting misalignment; internal byte steering feeds the 8-bit MAC from 32-bit words; TxValidBytes determines the final word’s valid count and TxEndFrm timing. Reads are throttled

by TX FIFO almost-full conditions to prevent underrun. RX DMA latches pointer/LSB and assembles incoming bytes into words according to alignment; full words or a partial final word trigger writes with appropriate byte lanes enabled. RX overrun is flagged if data arrives when the target buffer is full. All start/end strobes and status cross between MAC clocks and WB_CLK_I via synchronizers; BD status updates generate TxB/TxE and RxB/RxE IRQs.

-- Transmit MAC Pipeline and State Machines --

Overview and clocking\n- The transmit MAC pipeline operates entirely in the MTxClk domain and implements CSMA/CD-compliant frame transmission on the MII nibble interface. It integrates arbitration/control (eth_maccontrol, eth_transmitcontrol), the TX FSM/datapath (eth_txethmac/eth_txstatem), timing/counters (eth_txcounters), CRC-32 generation (eth_crc), and binary exponential backoff (eth_random). Requests and data arrive from the DMA/Wishbone engine (eth_wishbone) in WB_CLK_I and are synchronized into MTxClk; status/results are synchronized back to WB_CLK_I.\n\nComposition and interfaces\n- Inputs: TxStartFrm, TxEndFrm, TxData[7:0], TxUsedData, Pad, CrcEn, r_FullD, r_IPGT/r_IPGR1/r_IPGR2, r_MinFL/r_MaxFL, r_HugEn, r_NoBckof, r_ExDfrEn, r_DlyCrcEn, r_TxFlow, r_TxPauseTV; CarrierSense/Collision (from synchronized mcrs_pad_i/mcoll_pad_i).\n- Outputs: MTxEn, MTxD[3:0], MTxErr, TxDone, TxAbort, TxRetry, UnderRun, WillTransmit; status latches (RetryCntLatched, LateCollLatched, etc.) via eth_macstatus in WB domain.\n\nArbitration: normal data vs MAC control (PAUSE)\n- eth_maccontrol forwards normal TX requests when no control-frame override is active: passes TxStartFrm/TxEndFrm/TxData and propagates Pad/CrcEn.\n- When a transmit pause request is active (TPauseRq synchronized to MTxClk and r_TxFlow=1), eth_transmitcontrol asserts CtrlMux and takes control of the datapath to emit a PAUSE frame (DA 01-80-C2-00-00-01, Type 0x8808, OpCode 0x0001, PauseQuanta=r_TxPauseTV). It forces padding and CRC generation, drives TxCtrlStartFrm/ControlData/TxCtrlEndFrm, and holds TxUsedDataOut low to prevent consuming normal data. Handshakes (TxDone/TxAbort) are cleaned and single-shot.\n- Receive-driven pause (from eth_receivecontrol) blocks new normal-data starts (TxStartFrmOut = TxStartFrmIn & ~Pause) without affecting an ongoing frame.\n\nTX state machine (eth_txstatem)\n- States: Defer, IPG, Idle, Preamble, Data[0], Data[1], PAD, FCS, Jam, BackOff.\n- Key guards: Duplex gating (CarrierSense/Collision only in half-duplex when r_FullD=0); excessive defer mask r_ExDfrEn; optional no-backoff r_NoBckof.\n- Entry/transition synopsis:\n - Defer: Wait while CarrierSense=1 (half-duplex). If idle within excessive-defer threshold, transition to IPG; else ExcessiveDefer triggers abort at frame request.\n - IPG: Enforce interpacket gap. Rule1 selects IPGT vs IPGR2 (Rule1 set in Preamble or when FullDuplex; cleared in Idle/BackOff). StartIdle when IPG counter reaches programmed threshold.\n - Idle: Arm for transmission. StartPreamble when TxStartFrm is asserted and CarrierSense=0. New starts blocked by Pause.\n - Preamble: Emit 0x5 nibble pattern then SFD (0xD) when NibCntEq15. Initialize CRC around SFD per DlyCrcCnt and r_DlyCrcEn. StartData[0] after SFD.\n - Data[0]/Data[1]: Alternating byte phases output lower/upper nibble. Continue while TxEndFrm=0, no collision/underrun, and frame not too big. Transition to Data[1] if data available; otherwise evaluate PAD/FCS.\n - PAD: Insert zeros until minimum frame length (excluding FCS) satisfied (NibbleMinFl=1).\n - FCS: When CrcEn=1, output ones' complement of final CRC nibble-by-nibble. If CrcEn=0, may complete early once minimum length satisfied.\n - Jam: On Collision or UnderRun during transmit states, emit jam pattern (0x9) for 4 bytes (NibCntEq7).\n - BackOff: If collision within ColWindow and retries remain, wait random slot time (from eth_random). End when random backoff equals ByteCnt slot progress; else if disabled or zero, return directly to Defer.\n- Terminal outcomes:\n - TxDone: On clean FCS completion without collision, or early completion rules when CRC disabled and minimum length met.\n - TxRetry: After jam/backoff for early collision within collision window and RetryCnt < r_MaxRet.\n - TxAbort: For TooBig (MaxFrame with r_HugEn=0), UnderRun, ExcessiveDefer, LateCollision (outside ColWindow), or retry limit reached (RetryMax).\n\nTiming and counters (eth_txcounters)\n- Nibble counter (NibCnt): Runs across IPG, Preamble, Data/Pad/FCS, Jam, BackOff; provides NibCntEq15 (terminate preamble/SFD), NibCntEq7 (terminate jam), and supports state terminations.\n- Min frame logic: NibbleMinFl asserts when

minimum frame length (excl. FCS) is satisfied to terminate PAD or allow FCS/early end when CRC disabled.\n- Byte counter (ByteCnt): Increments per byte during data phases; also used as slot-time counter in BackOff (1 byte per 128 nibbles). Signals MaxFrame based on r_MaxFL with r_HugEn override.\n- Delayed CRC counter (DlyCrcCnt): Aligns CRC initialization/gating around SFD and early data when r_DlyCrcEn=1.\n\nCRC-32 generation and FCS (eth_crc in TX)\n- CRC seeded to 0xFFFFFFFF on Initialize_Crc; processes data/pad nibbles LSB-first while Enable_Crc=1; paused during FCS emission. FCS is transmitted as ones' complement of the final CRC in Ethernet order, nibble-by-nibble.\n\nMII outputs and error signaling (eth_txethmac)\n- MTxEn asserted during Preamble, Data, PAD, FCS, and Jam.\n- MTxD[3:0] sources: 0x5 preamble; 0xD SFD; TxData lower/upper nibbles in Data[0]/Data[1]; complemented CRC during FCS; 0x9 during Jam; control-frame bytes via eth_transmitcontrol when CtrlMux active.\n- MTxErr asserted on TooBig or UnderRun during a frame (collisions/retries alone do not assert MTxErr).\n- WillTransmit indicates imminent/active transmission and is used to synchronize a Transmitting indication into MRxClk for RX-side behaviors; ResetCollision is held high only when not transmitting to clear external collision latches.\n\nCollision handling, window, retries, backoff (half-duplex)\n- Collision window (ColWindow) valid until ByteCnt reaches CollValid; collisions afterward are LateCollision and cause abort without retry.\n- On collision within window: enter Jam, increment RetryCnt, optionally BackOff using eth_random LFSR value sampled at Jam. If Random=0 or r_NoBckof=1, reattempt without delay; else wait RandomEqByteCnt alignment to exit BackOff. RetryMax triggers abort when RetryCnt == r_MaxRet.\n\nFlow control interactions\n- TX pause injection: eth_transmitcontrol generates compliant PAUSE frames when requested (software r_TxPauseRq via CDC into MTxClk), forcing Pad/CrcEn and isolating normal data path. Completion interrupts are synchronized back to WB domain.\n- RX-driven pause: Pause signal from RX domain (synchronized to MTxClk) blocks new normal TxStartFrm; does not preempt in-flight frames or control frames already started.\n\nDMA/Wishbone TX interfacing\n- eth_wishbone asserts TxStartFrm when a TX BD is ready, steers 32-bit reads to bytes for TxData, and asserts TxEndFrm on the last valid byte using TxValidBytes/LastWord. TxUsedData pulses when the TX core consumes a byte; underrun detection sets TxUnderRun when data is demanded but unavailable. TxDone/TxAbort/TxRetry/UnderRun are synchronized to WB_CLK_I to update BD status/interrupts.\n\nConfiguration and CDC\n- Programmable registers govern timing/behavior: r_IPGT/r_IPGR1/r_IPGR2 (IPG), r_MinFL/r_MaxFL and r_HugEn (frame size), r_NoBckof and r_ExDfrEn (CSMA/CD nuances), r_FullD (duplex), r_CrcEn/r_Pad/r_DlyCrcEn (framing/CRC), r_TxFlow and r_TxPauseTV (flow control).\n- All cross-domain pulses (pause requests, TxDone/Abort/Retry, underrun) use multi-flop synchronization and one-shot edge detection to avoid metastability and double-counting.

-- Receive MAC Pipeline and Byte/Nibble Handling --

Scope: Defines how the RX MAC converts 4-bit MII nibbles into a byte-qualified stream and aligns control/status to byte boundaries in the MRxClk domain.

Clocking and gating
- All RX pipelines, counters, CRC, and FSM run on MRxClk. MRxDV qualifies enables.
- RxEnSync gates MRxDV/MRxErr to inhibit receive when disabled. Optional loopback sources MRx signals from TX MII when r_LoopBck=1.
- In half-duplex, a synchronized Transmitting indication blocks frame start.

Nibble phasing and byte assembly
- RX FSM: Idle → Preamble (0x5) → SFD (0xD) → Data0 → Data1 → Idle/Drop.
- Data0/Data1 encode nibble position: Data0=first nibble, Data1=second nibble of a byte.
- LatchedByte <= {MRxD[3:0], LatchedByte[7:4]} captures the high nibble first; the low nibble at Data1 completes the byte.
- ByteCnt resets at SFD and increments once per byte on Data1.

CRC handling
- CRC input nibbles reordered LSB-first: Data_Crc[3:0] = {MRxD[3], MRxD[2], MRxD[1], MRxD[0]}.
- Initialize_Crc asserted at SFD; with DlyCrcEn, also during the early delay window to start at DA.
- Enable_Crc = MRxDV & (Data0|Data1) & ~ByteCntMaxFrame.
- Correct frame leaves CRC at residue 0xC704DD7B; LatchedCrcError clears at SFD and updates in data.

Delayed-CRC option and counting
- When DlyCrcEn=1, DlyCrcCnt runs for 8 nibbles after SFD; ByteCnt increments are suppressed during this window.
- ByteCntOut = ByteCnt + 4 while delay is active to keep header parsing aligned.

Byte-qualified outputs and pipelines
- GenerateRxValid = Data0 & (~ByteCntEq0 | (DlyCrcCnt >= 3)); suppresses spurious valid before a full first byte.
- RxData registers LatchedByte only when GenerateRxValid=1; otherwise zero in non-data states.
- GenerateRxStartFrm = Data0 & ((ByteCntEq1 & ~DlyCrcEn) | ((DlyCrcCnt == 3) & DlyCrcEn)).
- GenerateRxEndFrm = Data0 & ((~MRxDV & ByteCntGreat2) | ByteCntMaxFrame).
- Dribble termination: DribbleRxEndFrm = Data1 & ~MRxDV & ByteCntGreat2 ensures clean half-byte end.
- RxValid, RxStartFrm, RxEndFrm are two-stage pipelines of their generate strobes to align with the registered RxData and assert for a single MRxClk.

IFG enforcement interaction
- IFGCounter enforces minimum 96 bit-times (24 nibbles). SFD before completion forces Drop unless r_IFG overrides.

Errors, symbols, and end sampling
- InvalidSymbol sets on MRxErr with nibble 0xE; DribbleNibble sets when MRxDV drops in Data1.
- Status sampling pipeline TakeSample → LoadRxStatus → ReceiveEnd latches per-packet flags (CRC good/bad, length OK/short/oversize, symbol/dribble) aligned to RxEndFrm.

Guards and corner cases
- Starts suppressed until a full first byte is available; delayed-CRC keeps external byte timing consistent.
- Oversize (ByteCntMaxFrame when r_HugEn=0) forces Drop; RX FSM exits to Idle on MRxDV drop with dribble-safe end.
- All control strobes are aligned to registered RxData for cycle-accurate byte framing.

-- Flow Control (PAUSE) Engines --

Implements IEEE 802.3x MAC Control (PAUSE) on both transmit and receive paths with proper arbitration, timing, and robust clock-domain crossings.

- Configuration
- CTRLMODER: r_TxFlow (enable TX PAUSE), r_RxFlow (enable RX PAUSE), r_PassAll (forward received control frames to RX when set).
- TXCTRL: r_TxPauseTV[15:0] (PAUSE quanta), r_TxPauseRq (software request; auto-clears after acceptance).

- TX PAUSE generation (eth_transmitcontrol + eth_maccontrol)
- Request: r_TxPauseRq sampled in wb_clk_i, 3-FF synchronized to MTxClk and edge-detected

(TPauseRq) when r_TxFlow=1. Handshake raises WillSendControlFrame; synced back to wb_clk_i to assert RstTxPauseRq.
- Arbitration/insertion: eth_maccontrol asserts CtrlMux to seize TX datapath, holds TxUsedDataOut low, and forces padding and CRC during SendingCtrlFrm. Normal TxDone/TxAbort are cleaned and masked to avoid contention.
- Frame format: 18-byte MAC Control PAUSE: DA=01-80-C2-00-00-01, SA=local MAC, Type=0x8808, Opcode=0x0001, Pause Time=r_TxPauseTV. Byte-aligned generation across nibble bus; optional DlyCrcEn alignment supported.
- Completion/IRQ: On TxCtrlEndFrm & StartTxDone, set irq_txc (TX flow-control complete), synchronized to system clock and acknowledged.

- RX PAUSE detection/enforcement (eth_receivecontrol + eth_maccontrol)
- Detection: Parse first 18 bytes; accept when DA is reserved multicast or local MAC, Type=0x8808, Opcode=0x0001. Capture pause time (bytes 16–17). Require good length and CRC.
- Timer load: On receive end, if r_RxFlow=1 and frame valid, load PauseTimer (SetPauseTimer) with captured quanta and set irq_rxc; synchronize IRQ to system clock and acknowledge.
- Quanta timing: Decrement PauseTimer once per 512 bit-times via MRxClk-derived Divider2 and 6-bit SlotTimer; operates uniformly across 10/100 MII by counting bit-times.
- TX gating: PauseTimer==0 double-synchronized into MTxClk; Pause asserted in TX while r_RxFlow=1 and timer!=0. Gate new normal frame starts: TxStartFrmOut = TxStartFrmIn & ~Pause. Ongoing frames are not interrupted; MAC control frames bypass Pause gating via CtrlMux.
- Delivery vs filtering: With r_PassAll=0, PAUSE frames are consumed for flow control and typically not delivered to RX buffers; with r_PassAll=1, they may be passed up.

- CDC and hygiene
- Multi-flop synchronizers and edge-detect pulses for requests and IRQs (e.g., TPauseRq, WillSendControlFrame); double-sync for level crossings (e.g., PauseTimerEq0). Updates occur only at defined idle/safe points to prevent mid-frame glitches.

- Standards compliance
- EtherType 0x8808, Opcode 0x0001, DA 01-80-C2-00-00-01; quanta are 512 bit-times each; control frames are always padded and CRC-protected.

-- MIIM Controller (Clockgen, Shiftreg, Output Control) --

- Purpose
- IEEE 802.3 clause 22 MIIM/MDIO master composed of controller, clock generator, shift register, and output control. Provides programmable MDC, preamble insertion/skip, read/write/scan transactions, proper turnaround tri-state, and clean handshakes to the MAC register block.
- External interface
- Clk: system clock for MIIM logic (input).
- MDC: MDIO clock to PHY with 50% duty (output).
- MDIO: bidirectional pad; driven by Mdo when MdoEn=1, otherwise tri-stated; MDI sampled into shifter.
- Register coupling (from eth_registers):
- MIIMODER: r_ClkDiv[7:0] (MDC divider), r_MiiNoPre (preamble suppress).
- MIICOMMAND: r_WCtrlData (write), r_RStat (read), r_ScanStat (scan), edge-detected into one-shot starts.
- MIIADDRESS: r_FIAD[4:0] (PHY address), r_RGAD[4:0] (register address).
- MIITX_DATA[15:0]: write data payload; MIIRX_DATA[15:0]: updated on read completion only.
- MIISTATUS[2:0]: {Nvalid_stat, Busy_stat, LinkFail} from MIIM.
- Controller/operation
- Requests are multi-registered and edge-detected; starts align to MDC rising strobe (MdcEn) when bus is idle. Write/read/scan type latched at start.

- Busy asserts while a request is pending/aligned, during an active frame, or while a scan waits alignment (Nvalid). EndBusy pulses when InProgress deasserts.
- 7-bit BitCounter increments on each MdcEn while InProgress:
- With preamble: counts 0..63 (32 preamble + 32 header/data bits). With r_MiiNoPre=1: skips preamble and starts at header.
- EndOp at 63; InProgress clears on next MdcEn.
- ByteSelect[3:0] strobes load of frame bytes into the shifter; LatchByte[1:0] strobes capture of incoming read data bytes. MIIRX_DATA updates at valid read/scan completion unless a new write starts that same cycle.
- Clock generator (Clockgen)
- Programmable divide of Clk to generate MDC with exact 50% duty; divider clamped to >=2.
- Rising/falling enables: MdcEn asserts one Clk cycle at MDC rising edge; MdcEn_n asserts one Clk cycle at MDC falling edge.
- For even N: MDC = Clk/N. For odd N: MDC = Clk/(N-1) using floor(N/2) half-periods.
- First toggle follows initial countdown to 0; stable duty maintained thereafter.
- Shift register and data capture (Shiftreg)
- 8-bit MSB-first shifter updates only on MdcEn_n (MDC falling edge):
- Byte loads (one-hot ByteSelect):
- [0]: {01, ~WriteOp, WriteOp, Fiad[4:1]} (start + OP + PHYAD[4:1]).
- [1]: {Fiad[0], Rgad[4:0], 10} (PHYAD[0], REGAD[4:0], TA for writes).
- [2]: CtrlData[15:8] (write data high byte).
- [3]: CtrlData[7:0] (write data low byte).
- Otherwise shifts left and samples MDI into LSB on each MdcEn_n.
- Readback latch: on LatchByte[1], Prsd[15:8] <= received byte; on LatchByte[0], Prsd[7:0] <= received byte.
- LinkFail update when latching low byte of PHY reg 1 (MIIBSR): LinkFail <= ~ShiftReg[1] (reflects Basic Status link bit).
- Output control (MDIO drive)
- SerialEn qualifies when the master drives frame bits.
- MdoEn asserted during preamble and whenever SerialEn=1; deasserted for read turnaround and data so PHY can drive the bus.
- Mdo outputs 1’s during preamble (when SerialEn=0 and BitCounter<32); otherwise follows current ShiftedBit. With r_MiiNoPre=1, driving starts at BitCounter==0.
- Mdo and MdoEn are double-registered and update only on MdcEn_n; upstream timing (BitCounter/ShiftedBit) accounts for this two-stage pipeline.
- MDIO pad connection: mdio = MdoEn ? Mdo : 'z; mdio input sampled as MDI by Shiftreg.
- Frame timing (with preamble; BitCounter at MdcEn)
- 0..31: preamble 1’s (MdoEn=1, Mdo=1).
- 0x20: ByteSelect[0] (start+OP+PHYAD[4:1]).
- 0x28: ByteSelect[1] (PHYAD[0]+REGAD+TA for writes).
- 0x30: ByteSelect[2] (write data high byte).
- 0x38: ByteSelect[3] (write data low byte).
- 0x37: LatchByte1 (read upper byte). 0x3F: LatchByte0 (read lower byte).
- 0x3F: EndOp; MIIRX_DATA commit for reads/scans; EndBusy pulse.
- Read/scan vs write
- Write: master continuously drives header, TA=10, and 16 data bits; MdoEn remains asserted after preamble.
- Read/scan: master drives header then tri-states during TA and data; PHY drives 16 data bits sampled into Prsd at defined latch points; MdoEn deasserted for data phase.
- Reset and robustness
- All MIIM state, shifter, Prsd, LinkFail, Mdo/MdoEn, and counters reset to idle/0; outputs safe (MdoEn=0).

- Requests inhibited/cleared on EndBusy to avoid retrigger; scan requests held (Nvalid) and aligned to MdcEn when idle.
- Clk must remain running during active MIIM operations to preserve MdcEn/MdcEn_n strobes and pipeline alignment.
- Integration and compliance
- Controller uses MdcEn for transaction state and bit counting; Shiftreg/Output control use MdcEn_n for data/drive updates, matching clause 22 setup/hold.
- Turnaround tri-state respected on reads; preamble suppress supported via r_MiiNoPre when PHY allows.
- MIIRX_DATA is hardware-updated only; software should poll/IRQ on Busy/EndBusy. MIISTATUS exposes Busy_stat, Nvalid_stat, and LinkFail.
- Configure r_ClkDiv to meet PHY MDC limits (typ ≤ 2.5 MHz for 10/100 unless PHY permits higher).

-- Clock Domain Crossing and Synchronization --

Clock domains
- wb_clk_i: system/Wishbone and MIIM control domain.
- mtx_clk_pad_i (MTxClk): transmit datapath and MAC control domain.
- mrx_clk_pad_i (MRxClk): receive datapath and address/filter/CRC domain.
- MDC/MDIO: eth_miim derives MDC from wb_clk_i via eth_clockgen; MDIO timing is internally aligned to MdcEn/MdcEn_n strobes with no async CDC to MAC datapaths.

Reset strategy
- wb_rst_i is a global asynchronous reset; all per-domain FFs, synchronizers, and latches are cleared locally.
- Cross-domain level latches use explicit, synchronized clear handshakes to guarantee the source deasserts only after the destination samples (e.g., RxAbort, TX control IRQ set/clear).
- Deassertion-sensitive paths are re-synchronized in their destination domains to avoid metastability on reset release.

CDC primitives and patterns
- Multi-flop level synchronizers (2–3 FFs), chosen per path latency/robustness needs.
- Rising-edge pulse generation in the destination domain from synchronized levels (q1/q2[/q3] edge-detect) for one-shot actions.
- Level-latch with explicit clear acknowledgment crossing back to the source to avoid double-counting and metastability.
- Protocol-safe gating of updates to minimize hazards (e.g., RX enable sampled only when MRxDV=0; Pause updated only when TX is idle/finishing).

Key crossings and synchronization
- PHY → TX (MTxClk):
- Carrier sense (mcrs_pad_i) synchronized via two-FF chain (CarrierSense_Tx1/Tx2). TxCarrierSense = ~r_FullD & CarrierSense_Tx2 (half-duplex only).
- Collision (mcoll_pad_i) sampled and level-latched in TX until ResetCollision; Collision = ~r_FullD & Collision_Tx2.
- TX → RX (MRxClk):
- Transmitting indication: WillTransmit synchronized through 2–3 stage chain (WillTransmit → WillTransmit_q → WillTransmit_q2). Transmitting = ~r_FullD & WillTransmit_q2 for RX-side gating.
- RX → TX (flow control):
- PauseTimerEq0 (MRxClk) double-synchronized into MTxClk; Pause is updated only at TX idle/finish points ((TxDoneIn | TxAbortIn | ~TxUsedDataOutDetected) & ~TxStartFrmOut). Pause is observed from RX for slot timing; updates are infrequent and gated, minimizing CDC risk.

- WB (system) ↔ TX (flow control/interrupts):
- Software TX Pause request: r_TxPauseRq (wb_clk_i) triple-synchronized into MTxClk; rising-edge detect creates TPauseRq (one-shot) when r_TxFlow=1.
- Control-frame completion: WillSendControlFrame (TX) triple-synchronized into wb_clk_i; rising edge generates RstTxPauseRq to clear pending SW request.
- TX control IRQs: SetTxCIrq generated in MTxClk, triple-synchronized into wb_clk_i with an edge pulse; WB acknowledgment (ResetTxCIrq) synchronized back to MTxClk to clear TX-side set.
- WB (system) ↔ RX (status/abort/control):
- RX abort: RxAbort_latch (MRxClk) synchronized into wb_clk_i (RxAbort_sync) for SW visibility; WB clear synchronized back to MRxClk (RxAbortRst_sync) to reset the latch reliably.
- RX enable: r_RxEn resampled in MRxClk only when MRxDV=0 (RxEnSync) to avoid mid-packet glitches.
- TX/RX results and DMA control ↔ WB (eth_wishbone):
- TX → WB: TxDone/TxAbort/TxRetry pulses and TxUnderRun flag synchronized via 2–3 FF chains; WB performs single descriptor/status updates guarded by BlockingTxStatusWrite/NotCleared qualifiers to ensure exactly-once processing.
- RX → WB: WriteRxDataToFifo pulses synchronized into wb_clk_i; end-of-shift/end-of-frame and abort indications synchronized for buffer writes and status commits; Busy_IRQ set if a frame starts while not RxReady.
- WB ↔ MTxClk/MRxClk bulk/control strobes use one-shot pulses with blocking/ack mechanisms to prevent replays across domains.

Per-module domain boundaries
- eth_txethmac and eth_maccontrol: entirely in MTxClk; consume synchronized CarrierSense/Collision and produce WillTransmit; expose TX events to WB via synchronized pulses.
- eth_rxethmac: entirely in MRxClk; consumes Transmitting (synchronized from TX); produces RX data/events and local filters/CRC results; exports via synchronized handshakes.
- eth_registers: in wb_clk_i; hosts configuration bits and MIIM control; implements the CDC synchronizers for TX/RX flow-control, interrupts, and status updates.
- eth_macstatus: maintains separate MRxClk and MTxClk latches; no internal CDC, downstream domains synchronize as needed.

Timing for MIIM
- eth_clockgen generates MdcEn/MdcEn_n from wb_clk_i; eth_miim pipelines request/complete and data capture to these strobes (q1/q2/q3 chains). UpdateMIIRX_DATAReg is a one-shot in wb_clk_i; no crossings to MTxClk/MRxClk.

Additional considerations
- All Wishbone ack/err pulses are single-cycle in wb_clk_i and do not cross clock domains.
- FIFOs are single-clock; domain crossings rely on synchronizers and per-domain latches, not dual-clock FIFOs.
- Half-duplex only: CarrierSense/Collision are masked in full-duplex (r_FullD=1).

-- Loopback and RX Enable Gating --

Implements an internal MII loopback and a safe RX-enable gate in the MRxClk domain. Loopback selection: when r_LoopBck (MODER[7])=1, RX inputs are sourced from TX MII signals (MRxD<=MTxD[3:0], MRxDV<=MTxEn, MRxErr<=MTxErr); when 0, RX inputs come from external PHY pads (MRxD/MRxDV/MRxErr). RX-enable gating: MRxDV and MRxErr presented to the RX front-end are AND-gated by RxEnSync, a copy of r_RxEn synchronized into MRxClk and updated only when MRxDV==0 to avoid mid-frame glitches. Effective behavior: MRxDV_to_RX = Selected_MRxDV & RxEnSync; MRxErr_to_RX = Selected_MRxErr & RxEnSync; MRxD_to_RX = Selected_MRxD (data

follows the selected source, DV/ERR are gated). Configuration sources: r_RxEn = MODER[0] & (TX_BD_NUMOut < 0x80), preventing RX enable when no RX BD space exists; r_LoopBck = MODER[7]. Integration: eth_rxethmac and eth_rxstatem consume the gated MRxDV/MRxErr, so deasserting RxEnSync holds the RX path idle regardless of loopback or pads. With loopback enabled and RxEnSync=1, TX can self-test RX without a PHY; in half-duplex, an independent Transmitting indicator (synced into MRxClk and masked by ~r_FullD) still prevents starting a reception while TX is active. CDC safety: r_RxEn is produced in the system/Wishbone clock domain and safely transferred to MRxClk as RxEnSync with the guard that updates occur only when MRxDV=0, ensuring enables/disables do not tear or abort frames.

-- Carrier Sense/Collision Handling (Half-Duplex) --

When r_FullD=0, the MAC operates in IEEE 802.3 half■duplex with CSMA/CD. Carrier sense (mcrs_pad_i) is double■synchronized into MTxClk and gated by ~r_FullD to form TxCarrierSense; the TX FSM defers transmission while TxCarrierSense is high and detects excessive deferral against a fixed nibble threshold, maskable via r_ExDfrEn. CarrierSenseLost is flagged if carrier drops during preamble/data while half■duplex, not in loopback, and no collision is present. Collision (mcoll_pad_i) is synchronized and level■latched in the TX clock domain; Collision = ~r_FullD & Collision_Tx2 feeds the TX MAC until cleared by ResetCollision, which is asserted whenever TX is not actively transmitting. On collision (or underrun), the TX FSM emits a JAM of 0x9 for 8 nibbles, then either retries/backoffs or aborts based on the collision window. The valid collision window (ColWindow) remains active until the byte counter reaches the programmable r_CollValid threshold; collisions within this window initiate binary exponential backoff (slot time = 512 bit■times, random delay from a 10■bit LFSR scaled by RetryCnt) and retry up to r_MaxRet. Collisions outside the window are treated as late and force abort; LateCollision and RetryCnt/RetryLimit are latched for TX status/interrupts. r_NoBckof bypasses backoff (transition from JAM directly to DEFER). On the RX side, the transmit■active indication is synchronized into MRxClk; if MRxDV asserts while transmitting, RX is dropped to enforce no simultaneous TX/RX. An RX collision window mirrors TX policy, marking RxLateCollision for events outside the window (or per r_RecSmall). When r_FullD=1, CSMA/CD is bypassed: carrier sense and collision inputs are ignored, and CarrierSenseLost is suppressed in loopback.

-- Status Collection and Latching --

Status is captured in the local clock domains (MRxClk for RX, MTxClk for TX, wb_clk_i for DMA/Wishbone), converted to single-cycle pulses where needed, and transferred across domains via multi-flop synchronizers and explicit request/clear handshakes to prevent metastability and double-counting. Receive-side (MRxClk) uses a sampling pipeline (TakeSample → LoadRxStatus → ReceiveEnd) to commit per-frame flags; CRC error is latched between SFD and data and inverted to form ReceivedPacketGood; MRxErr occurrences qualify InvalidSymbol and DribbleNibble (clearing at LoadRxStatus and SFD respectively); length checks (ShortFrame, ReceivedPacketTooBig, ReceivedLengthOK) evaluate at TakeSample with r_HugEn gating oversize; collision-window tracking produces RxLateCollision outside the programmed window; address acceptance (UnicastOK, BroadcastOK, MulticastOK) determines RxAddressInvalid; RxAbort is a one-cycle pulse at ByteCntEq7 when no acceptance and not promiscuous, while AddressMiss latches a broader miss for reporting; RxValid, RxStartFrm, and RxEndFrm are pipelined and aligned to data with dribble-safe end generation. Transmit-side (MTxClk) level-latches TxDone, TxRetry, and TxAbort on their start events, clearing on new TxStartFrm; PacketFinished edges qualify one-shot outcomes; RetryCntLatched, RetryLimit (MaxCollisionOccured), and LateCollLatched are captured on completion/abort; DeferLatched records deferral; CarrierSenseLost latches on CS drop during preamble/data in half-duplex and clears on TxStartFrm; external collision is level-latched until ResetCollision and masked in full-duplex; MuxedDone and MuxedAbort are gated one-shots derived from TxDoneIn/TxAbortIn edges to avoid contention with starts and control frames (BlockTxDone). Flow control latches PAUSE reception (LatchedTimerValue, PauseTimer) and generates a TX-domain

Pause that is only updated at TX idle/finish; software TX PAUSE request (r_TxPauseRq) is triple-synchronized into MTxClk and edge-detected (TPauseRq); WillSendControlFrame latches during control-frame transmission and enforces Pad/CrcEn; flow-control events produce SetTxCIrq/SetRxCIrq via 3-FF synchronizers with reset handshakes. MDIO/MIIM status uses InProgress and Busy to gate operations; EndBusy is a one-shot on InProgress falling; UpdateMIIRX_DATAReg pulses at the end of read/scan to commit Prsd to MIIRX_DATA; LinkFail is derived when reading PHY Basic Status; MIISTATUS packs NValid, Busy, and LinkFail. In wb_clk_i, TX outcomes (TxDone_wb, TxAbort_wb, TxRetry_wb) and TxUnderRun_wb are synchronized from MTxClk with guards against double processing; TxError aggregates underrun, retry limit, late collision, and carrier sense loss; TxStatusWrite commits BD status and raises TxB_IRQ/TxE_IRQ based on per-BD IRQ enables. RX-side collects RxStatusIn (pause, address miss, overrun, invalid symbol, dribble, too big, short, CRC error, late collision) and LatchedRxLength; ShiftEnded triggers RxStatusWrite; RxOverrun latches FIFO overflow into BD; RxB_IRQ/RxE_IRQ fire per BD IRQ enables with pause gating; Busy_IRQ indicates RX path not ready at frame start. Wishbone ack/err are edge-pulsed (single-cycle wb_ack_o/wb_err_o) from decoded responses or byte-enable/address misses. Cross-domain handshakes include RxAbort_latch synchronized into wb_clk_i with a return clear (RxAbortRst); WillTransmit synchronized into MRxClk to form Transmitting for RX FSM masking; CarrierSense is two-FF synchronized into MTxClk for defer and carrier-loss logic. All latches reset asynchronously and clear at defined boundaries: RX at SFD or LoadRxStatus, address/multicast latches at RxEndFrm or RxAbort; TX on TxStartFrm, collision latch on ResetCollision, control-mode state at TxDoneIn; MIIM on EndBusy and read/scan completion; Wishbone pulses are generated as one-shots. Per-packet outcomes are reported via BD status writes, while per-event and flow-control statuses are exposed via CPU-visible registers and interrupts.

-- FIFOs and Buffering --

FIFOs and Buffering
- Resources: Two synchronous single-clock data FIFOs (TX and RX) in the Wishbone (WB) domain provide decoupling and staging. A 256x32 single-port descriptor/internal buffer RAM (eth_spram_256x32) holds buffer descriptors, pointers, and status. Small staging registers on each path (TxDataLatched, LatchedByte, RxWord assembly) manage byte/word alignment.
- FIFO implementation: eth_fifo instances feature occupancy counters with empty, almost_empty, almost_full, and full flags; one-cycle read latency; async reset; and synchronous clear that re-centers pointers at index 0. Guardrails require no write on full, no read on empty, and avoiding simultaneous read+write at boundary conditions.
- Clock domains and CDC: All FIFO read/write ports reside in the WB clock domain. Data production/consumption signals from MAC domains are transferred via multi-flop synchronizers and handshake pulses (e.g., TxUsedData from MTxClk, WriteRxDataToFifo from MRxClk). No dual-clock FIFOs are used.
- TX path: The WB master fills the TX FIFO using 4-beat incremental bursts when txfifo_cnt < (DEPTH − 4) and TxLength > 20 bytes (cti=010 then 111 on last). Byte steering uses TxPointerLSB; the final partial beat (1–3 bytes) is driven by TxValidBytesLatched. MAC consumes via TxUsedData; underrun is detected when the FIFO becomes empty while MAC needs data, sets TxUnderRun, asserts error/IRQ, and triggers TX FIFO clear on abort/retry. During 802.3x control frame insertion, TxUsedDataOut is forced low to prevent draining normal TX FIFO; Pause can suppress new frame starts.
- RX path: MRxClk assembles nibbles into bytes and 32-bit words per RxPointerLSB and local counters. When a full or terminal partial word is ready, WriteRxDataToFifo pulses into the WB domain to push the RX FIFO; unused lanes in the final word are zero-padded and masked by RxByteSel. The WB master drains via 4-beat bursts when rxfifo_cnt ≥ 4 or while continuing a burst (cti=010, then 111 on last), with m_wb_sel_o honoring pointer alignment. Overrun latches when RX FIFO is full and more data arrives. RxReady gates acceptance; Busy_IRQ asserts if a frame starts while not ready. End-of-frame forces a flush of the last partial word.

- Descriptor/internal RAM: Single-port 256x32 with per-byte write enables, registered address (1-cycle read latency), and write-first behavior on same-cycle read/write for enabled lanes. The RAM is arbitrated per cycle among CPU (WbEn), TX (TxEn), and RX (RxEn) clients; only one owner drives address/we/oe/di at a time. The array is not cleared by reset; outputs are driven to 0 while rst is asserted.
- Throughput and robustness: 4-beat bursts maximize bus efficiency while respecting TX headroom and RX availability. CDC via handshake pulses ensures reliable cross-domain buffer operation. FIFO flags and guardrails, underrun/overrun handling, and isolation during control frames maintain integrity of buffered data.

-- Random Backoff Generator --

Generates the IEEE 802.3 CSMA/CD binary exponential backoff count and completion indication in the MTxClk domain. A 10-bit LFSR (polynomial x^10 + x^3 + 1, XNOR feedback ~(x[2]^x[9])) advances every MTxClk (period 1023). On each collision/jam event, a random value is latched; the effective backoff window size is k = min(max(RetryCnt, 1), 10), yielding a uniform integer in [0 .. 2^k – 1] by masking higher LFSR bits as RetryCnt grows. RandomLatched resets to 0 on Reset, making RandomEq0=1 initially. StartBackoff is asserted only when jam completes at nibble boundary, the collision is within the window, retries remain, randomized backoff is enabled, and the latched random is non-zero; otherwise the path defers immediately. Backoff duration is measured in slot times: ByteCnt[9:0] is reset at StartBackoff and increments once per 128 nibbles (512 bit-times) while in BackOff; completion occurs when ByteCnt[9:0] equals RandomLatched at a slot boundary, reported by RandomEqByteCnt. RetryCnt increments either when a zero-backoff condition occurs at jam completion or when a non-zero backoff finishes. Control/status: RandomEq0 signals zero-wait; RandomEqByteCnt signals backoff done. Operation is gated to half-duplex (~r_FullD); NoBckof disables randomized backoff (forces immediate defer); RetryMax prevents new backoff attempts; the window saturates at 10 bits. LFSR runs continuously; only the latched sample governs the backoff. Ensure ByteCnt is reset at StartBackoff to avoid stale compares; initial post-reset collision will skip backoff unless a new jam capture occurs first.

-- CRC Engine --

- Function: 32-bit Ethernet CRC-32 (AUTODIN II) engine used in both transmit (TX) and receive (RX) datapaths on an MII 4-bit nibble interface. Generates FCS in TX and checks FCS in RX.
- Algorithm: Polynomial 0x04C11DB7 with reflected implementation (equivalent reversed poly 0xEDB88320). Processes data LSB-first, 4 bits per clock.
- Initialization: CRC register seeds to 0xFFFFFFFF on Reset or a synchronous Initialize pulse. Initialize is asserted prior to the Destination Address (DA) so the seed is active when CRC coverage begins.
- Data ordering: MII uses low nibble first; feed Data[3:0] to the CRC LSB-first. TX and RX map nibbles accordingly (TX uses nibble-reversed mapping from TxData; RX maps MRxD[3:0] directly LSB-first).
- Enable behavior: When Enable=1, each incoming nibble is absorbed into the LFSR. When Enable=0, the network shifts zeros. Enable must remain asserted continuously across the CRC-covered region (DA through payload/pad), excluding preamble/SFD and excluding TX FCS.
- Throughput: 4 bits per MII clock (MTxClk for TX instance, MRxClk for RX instance).
- Residue check (RX): After absorbing the entire frame including the 4-byte FCS, a correct frame leaves CRC at 0xC704DD7B. CrcError = (Crc != 32'hC704DD7B), sampled after the last FCS nibble. RX status logic latches this per frame; ReceivedPacketGood = ~LatchedCrcError.
- FCS generation (TX): The transmitted FCS is the ones' complement of the running CRC. It is emitted nibble-by-nibble LSB-first; eth_txethmac's StateFCS outputs ~Crc starting at Crc[31:28] and down. The FCS is not fed back into the CRC generator. Enable_Crc = ~StateFCS so CRC runs during data/pad and holds while FCS is sent.
- Delayed CRC support: When r_DlyCrcEn is set, DlyCrcCnt windows adjust Initialize timing and early

data gating on both TX and RX to ensure CRC alignment to DA.
- Gating and limits (RX): Enable_Crc is gated by MRxDV and data states, and suppressed at ByteCntMaxFrame to stop CRC accumulation beyond the configured max frame length.
- Clocks and domains: Independent CRC instances in MTxClk (TX) and MRxClk (RX) domains. Results are consumed locally (e.g., RX macstatus latching).
- Configuration and interactions: r_CrcEn (MODER[13]) enables CRC/FCS insertion in TX. eth_maccontrol forces CRC enabled when sending IEEE 802.3x MAC Control (PAUSE) frames. r_DlyCrcEn (MODER[12]) enables delayed-CRC timing adjustments.
- Constraints: Do not include preamble/SFD in CRC. Keep Enable asserted over the covered region. Maintain correct LSB-first nibble mapping. TX does not perform residue checking; it relies on generated ~Crc.

OPERATION

-- Reset and Initialization --

- Global reset and deassertion
- wb_rst_i is an asynchronous, top-level reset applied to all three clock domains: wb_clk_i (CPU/Wishbone), mtx_clk_pad_i (TX MII), and mrx_clk_pad_i (RX MII).
- Reset deassertion is safely managed using multi-flop synchronizers (q1/q2/q3) or explicit request/ack handshakes for all cross-domain events to avoid metastability, double-triggering, or stale pulses; recovery/removal timing must be met.

- Wishbone/CPU domain (wb_clk_i)
- Configuration/status registers (eth_registers) asynchronously reset to defined RESET_VALUEs; all RW/W1C fields clear; interrupt source bits clear and int_o deasserts.
- MIIM register set (MIIMODER, MIICOMMAND, MIIADDRESS, MIITX_DATA, MIIRX_DATA, MIISTATUS) clears; UpdateMIIRX_DATAReg is idle; MIIM Busy deasserts.
- Wishbone slave outputs (data/ack/err) clear; no ack/err until a valid access.
- Wishbone master/BD engine (eth_wishbone) clears m_wb_* signals, BD indices/latches, burst counters, and NotCleared flags; TX/RX FIFOs are asynchronously reset to empty; eth_spram_256x32 read output q is 0 while rst is asserted (RAM contents not cleared).

- MIIM management sub-blocks (wb_clk_i)
- eth_miim: Busy/InProgress/WriteOp/one-shots/Nvalid clear; BitCounter=0; EndBusy/UpdateMIIRX_DATAReg pulses idle.
- eth_clockgen: Mdc=0 and Counter=1 at reset; first MDC toggle/enables (MdcEn/MdcEn_n) occur deterministically after next counter zero.
- eth_shiftreg: ShiftReg/Prsd/LinkFail clear to 0.
- eth_outputcontrol: Mdo/MdoEn clear to 0; serialization pipeline reset.

- TX MII domain (mtx_clk_pad_i)
- eth_txethmac: TX FSM enters Defer/idle; NibCnt/ByteCnt/DlyCrcCnt clear; status latches (TxDone/TxRetry/TxAbort) clear; WillTransmit deasserts; MTxEn/MTxD/MTxErr idle; CRC seeded to 0xFFFFFFFF (Initialize asserted at start-of-frame after reset).
- eth_txstatem resets to Defer; IPG/Rule1 selection and collision-window flags reinitialize; ResetCollision active when not transmitting.
- eth_txcounters clear; excessive-defer/min/max-frame flags reset.

- eth_random: LFSR and RandomLatched clear; RandomEq0 true until a jam capture.
- eth_maccontrol: TxReset clears TxUsedDataOutDetected, edge-detect latches, control-arbitration state; CtrlMux deasserts; BlockTxDone clears.
- Flow-control TX path: TPauseRq synchronizers reset; no PAUSE frame transmission until a new software request is synchronized and edge-detected.

- RX MII domain (mrx_clk_pad_i)
- eth_rxethmac: RxData/RxValid/RxStartFrm/RxEndFrm/Broadcast/Multicast/CrcHash/RxGood clear; CRC seeded to 0xFFFFFFFF; enable/init gating resumes at SFD after reset.
- eth_rxstatem: StateDrop=1 at reset; transitions to Idle only when MRxDV=0 to prevent false frame starts.
- eth_rxcounters: ByteCnt/IFGCounter/DlyCrcCnt clear; IFG measurement restarts at next SFD; ByteCntMaxFrame flags reset.
- eth_rxaddrcheck: UnicastOK/MulticastOK latches clear; RxAbort=0; AddressMiss clears at ByteCntEq0.
- eth_receivecontrol: DetectionWindow/ByteCnt clear; LatchedTimerValue/PauseTimer=0; Pause deasserted; PauseTimerEq0 double-synchronized into TX domain.

- MAC status and interrupts
- eth_macstatus latches/flags clear (CRC errors, late collision, invalid symbol, dribble, retry, defer, carrier loss); INT_SOURCE clears and W1C semantics restart from 0.

- Cross-domain reset/clear handshakes
- RX abort path: RxAbort_latch (MRxClk) resets; synchronized copy (wb_clk_i) clears; return clear (RxAbortRst) is synchronized back to MRxClk.
- Flow-control IRQ set/clear chains in TX/RX domains reset; edge detectors and return clears initialize to suppress spurious interrupts.

- Post-reset enable gating and safety
- r_TxEn/r_RxEn recompute from MODER defaults and TX_BD_NUM partition; RxEnSync initializes to “receive disabled” until a stable enable is captured with MRxDV=0.
- Loopback (r_LoopBck) and duplex (r_FullD) default to MODER reset values; carrier/collision synchronizers clear; Transmitting gating initializes to 0 in RX domain.
- FIFOs: asynchronous reset leaves them empty; optional synchronous clear may be used for alignment; obey no-read-when-empty/no-write-when-full rules to keep pointers/counters consistent.

-- Bus Transactions (Wishbone Slave/Master) --

Wishbone Slave (wb_clk_i domain)
- Address decode: wb_adr_i[11:10] = 00 → register space (eth_registers); 01 → Buffer Descriptors/Internal SRAM window (eth_wishbone); 1x → unmapped, error.
- Byte select requirement: ByteSelected = |wb_sel_i must be 1 on any access; if 0, signal wb_err_o and no write occurs.
- Read data mux: For reads in register space, wb_dat_o = RegDataOut; otherwise wb_dat_o = BD_WB_DAT_O (BD RAM).
- Write semantics: Per-byte enables honored in both register space and BD RAM; data written only to active byte lanes.
- Handshake/timing: wb_ack_o and wb_err_o are one-cycle registered pulses. wb_ack_o asserts on selected register accesses or BD RAM accesses that complete; wb_err_o asserts on address decode miss or ByteSelected=0. Acknowledge and error are mutually exclusive.
- BD RAM timing: BD reads have 1-cycle latency (registered address); WB_ACK_O is generated from BDRead/BDWrite handshake.

Wishbone Master (DMA via eth_wishbone)
- Addressing/controls: m_wb_adr_o is word-aligned ({m_wb_adr_tmp, 2'b00}); m_wb_stb_o = m_wb_cyc_o; bte_o = 2'b00 (linear); cti_o = 3'b010 during bursts and 3'b111 on last beat.
- Progress gating: MasterAccessFinished = m_wb_ack_i | m_wb_err_i; used to advance state and increment pointers.
- TX (memory → MAC reads): m_wb_we_o = 0; m_wb_sel_o = 4'hF. Up to 4-beat bursts when TX FIFO has room and TxLength > 20 bytes. Pointer MSB increments per accepted beat; alignment handled by internal TxPointerLSB with byte steering.
- RX (MAC → memory writes): m_wb_we_o = 1; m_wb_sel_o derives from RxPointerLSB to strobe the correct byte lanes (1111/0111/0011/0001). Up to 4-beat bursts gated by RX FIFO levels; pointer MSB increments on m_wb_ack_i; partial last word flushed with appropriate sel.
- Error handling: m_wb_err_i terminates the current master access; TX/RX engines capture status and proceed to BD/status updates.

BD/Internal SRAM window (slave path)
- Arbitration: CPU accesses to BD RAM share the single-port RAM with TX/RX engines; only one user drives the RAM per cycle.
- Data/ack: WB_DAT_O for BD reads is ram_do; WB_ACK_O is produced from BDRead/BDWrite handshake; per-byte write enables supported.

Summary
- Slave: single-cycle ack/err pulses; decode and byte-select enforced; data sourced by register file or BD RAM.
- Master: linear bursts with aligned addresses; TX uses full-word reads; RX uses alignment-aware sel strobes; bursts end with cti=3'b111; progress strictly gated by ack/err.

-- Descriptor Ring Management (TX/RX) --

The MAC uses a single descriptor ring stored in an internal 256x32 single-port RAM (two 32-bit words per BD: word0=status/length, word1=buffer pointer with byte-alignment LSBs). The ring is partitioned by TX_BD_NUM (r_TxBDNum): TX consumes BDs [0 .. r_TxBDNum-1], RX consumes BDs [r_TxBDNum .. last]. TX_BD_NUM writes are accepted only when DataIn <= 0x80 (max 128 BDs); enable gating prevents invalid partitions: r_TxEn = MODER[1] & (TX_BD_NUMOut > 0) and r_RxEn = MODER[0] & (TX_BD_NUMOut < 0x80).

TX ring management: TxBDAddress selects the current TX BD. TxBDRead fetches status/length; a BD is eligible when Ready=1 and Length > 4 bytes. TxPointerRead fetches the buffer pointer and alignment LSBs. After a frame completes (success, abort, or retry), TxStatusWrite updates the BD and advances TxBDAddress; advance is linear unless the BD’s wrap bit (WrapTxStatusBit) is set, in which case TxBDAddress wraps to 0. TxFifoClear may assert on abort/retry to drain residual data. BlockingTxStatusWrite prevents duplicate status writes across clock domains.

RX ring management: On RX enable, RxBDAddress initializes to r_TxBDNum. RxBDRead fetches status/length; a BD is eligible when Ready=1. RxPointerRead fetches the buffer pointer; pointer LSBs provide byte alignment and derive per-byte write strobes. When a frame ends or is aborted, RxStatusWrite updates the BD and advances RxBDAddress; advance is linear unless the BD’s wrap bit (WrapRxStatusBit) is set, in which case RxBDAddress wraps to r_TxBDNum. RxBDRead is re-armed after RxStatusWrite completion or abort to stage the next BD.

CDC and arbitration: All TX/RX start/end/status/abort events cross between MTxClk/MRxClk and WB_CLK_I via synchronizers with blocking/deglitching. Descriptor reads/writes and pointer advances

are performed in WB_CLK_I and only after acknowledged operations to avoid duplication. The single-port BD RAM is arbitrated per cycle among CPU, TX, and RX; the CPU can read/write BDs via the Wishbone slave path.

Interrupts: On TxStatusWrite or RxStatusWrite, if BD status[14] (per-BD IRQ enable) is set, non-error outcomes raise TxB_IRQ/RxB_IRQ and error outcomes raise TxE_IRQ/RxE_IRQ.

Guards: RX overrun and other per-packet conditions are latched into RxStatus; Busy_IRQ can assert if a frame starts when RX is not ready. These conditions do not advance ring indices until the corresponding status write completes.

-- Transmit Frame Sequence --

- Entry and arbitration
- Transmission is permitted only when r_TxEn=1 and a TX buffer descriptor (BD) marked Ready with a valid length is available. eth_maccontrol arbitrates the TX path between a normal data frame and an injected MAC Control (PAUSE) frame. If a receive PAUSE timer is active and r_TxFlow=1, new normal frames are gated until Pause deasserts; ongoing frames are not interrupted.

- Data sourcing
- eth_wishbone fetches the TX BD, reads the buffer pointer and length, performs Wishbone master reads, and byte-aligns 32-bit words into the TX stream/FIFO. TxStartFrm and TxEndFrm delimit the presented payload; underrun is detected if the MAC requests data while the FIFO is empty.

- Defer and IPG
- After any frame attempt, the MAC enforces an inter-packet gap using r_IPGT/r_IPGRx. In half-duplex, if CarrierSense is present the MAC defers transmission until idle; excessive defer may be flagged per configuration.

- Preamble and SFD
- From Idle with TxStartFrm and no CarrierSense, MTxEn asserts and the MAC transmits 7 bytes of 0x55 followed by SFD 0xD5 (total 16 nibbles). If DlyCrcEn is set, a short alignment window follows; the CRC engine initializes to 0xFFFFFFFF.

- Payload and pad
- Payload bytes are transmitted nibble-by-nibble (two-phase data state), with TxUsedData indicating consumption. If Pad=1 and the minimum frame size is not met at payload end, the MAC inserts pad bytes to reach the minimum length.

- CRC and FCS
- If CrcEn=1, the CRC runs over header/payload/pad; the MAC appends the ones’ complement of the CRC as a 4-byte FCS, least-significant byte first, nibble-by-nibble. If CRC is disabled, the frame ends at payload/pad termination.

- Completion
- StartTxDone asserts at the last FCS nibble (or at data end in CRC-less mode). TxDone is latched and a cleaned TxDoneOut pulse is produced after ensuring real data use; MTxEn deasserts at frame end.

- Errors and limits
- Underrun when the source starves triggers an immediate jam and abort, with MTxErr asserted during the fault. If the maximum frame length is exceeded and r_HugEn=0, the MAC aborts and asserts MTxErr.

- Collisions, jam, and retry (half-duplex)
- During transmission, a collision inside the collision window causes 8 nibbles of jam (pattern 0x9). If retries remain, the MAC backs off by a binary-exponential random slot-time delay and retries; if the collision is late or the retry limit is reached, it aborts without retry. ResetCollision is held inactive only while sending/jamming.

- Flow-control (PAUSE) frame insertion
- When TPauseRq is set and r_TxFlow=1, the MAC seizes the TX path at an idle boundary and injects a standards-compliant PAUSE frame: DA 01-80-C2-00-00-01, SA, Type 88-08, OpCode 00-01, and pause time r_TxPauseTV. Pad and CRC are forced on. Normal data is blocked during insertion; upon completion, control is released and RstTxPauseRq is generated.

- On-wire signaling
- MTxEn is asserted during preamble, payload, pad, FCS, and jam. MTxD outputs: 0x5 nibbles for preamble, SFD ending with nibble 0xD, payload/pad nibbles from data stream, CRC nibbles from the inverted CRC, and 0x9 during jam. MTxErr asserts on underrun or oversize aborts.

- Status and progression
- The TX path reports TxDone, TxRetry, TxAbort, LateCollision, RetryCnt, and CarrierSenseLost for BD status writeback and interrupts. After Done/Abort/Retry, the machine returns to Defer/IPG, then arbitrates the next frame (normal or control).

-- Collision, Jam, Backoff, Retry/Abort --

Half-duplex CSMA/CD only (disabled in full-duplex via r_FullD): CarrierSense (mcrs_pad_i) and Collision (mcoll_pad_i) are sampled into the TX clock domain and masked when r_FullD=1. A top-level collision latch holds Collision level until ResetCollision, which is asserted whenever the transmitter is not in an active TX/jam state. Collision classification uses a programmable window: ColWindow is asserted from frame start and deasserted when ByteCnt reaches r_CollValid[5:0] (COLLCONF), at a defined nibble alignment; collisions while ColWindow=1 are early, otherwise late.

Jam: Any Collision during transmit phases (preamble/data/pad/FCS) or an underrun forces StartJam and transmits a fixed 32-bit jam sequence (8 nibbles, 0x9 pattern). Jam lasts exactly 8 nibbles; upon jam-complete, the MAC chooses BackOff or immediate Defer/IPG based on configuration and random value.

Backoff: Binary exponential backoff implemented by eth_random (10-bit LFSR, x^10 + x^3 + 1). At jam, a random value is latched with k=min(RetryCnt,10) enabled bits. A backoff slot is 512 bit-times; the slot counter (ByteCnt) advances only at slot boundaries indicated by &NibCnt;[6:0]. Backoff ends when ByteCnt[9:0] equals the latched random at a slot boundary (RandomEqByteCnt). If r_NoBckof=1 or the latched random is 0, backoff is skipped (zero-slot); the MAC proceeds directly to Defer/IPG. RetryCnt increments at jam-complete for the skip case, or at backoff-complete otherwise.

Retry/Abort: StartTxRetry is asserted for early collisions within ColWindow when retries remain (~RetryMax) and no underrun. StartTxAbort is asserted for LateCollision (collision after ColWindow), MaxCollisionOccured (RetryCnt == r_MaxRet), UnderRun, TooBig (frame length exceeded when hug disabled), or ExcessiveDeferOccured (unless masked by r_ExDfrEn). After retry decision, the TX state machine transitions through Defer/IPG before a retransmission attempt. In full-duplex (r_FullD=1), CSMA/CD is disabled: CarrierSense/Collision do not trigger jam/backoff, and transmission proceeds without retries.

Status/Reporting: eth_macstatus latches RetryCntLatched, RetryLimit, and LateCollLatched on

StartTxDone/StartTxAbort; it also tracks RxLateCollision in half-duplex when a collision occurs outside the window on the RX side (cleared at end-of-frame). eth_wishbone captures TxRetry/TxAbort, updates buffer descriptor status bits (RetryLimit, LateCollision, etc.), clears the TX FIFO on Abort/Retry, and raises interrupts (TxB_IRQ/TxE_IRQ) per aggregated error status. MTxErr asserts only for TooBig or UnderRun; collisions/retries alone do not assert MTxErr.

-- Receive Frame Sequence --

1) Source select and enable: MRxD/MRxDV/MRxErr are taken from external MII unless r_LoopBck=1, in which case the RX path consumes internal TX (mtxd/mtxen/mtxerr). MRxDV and MRxErr are gated by RxEnSync (inactive when r_RxEn=0). 2) IFG and start qualification: RX idles until MRxDV=1 and, in half■duplex, Transmitting=0. Inter■frame gap must meet 96 bit■times (IFGCounterEq24=1) unless forced by r_IFG. If MRxDV asserts with Transmitting=1 (and ~r_FullD) or IFG is insufficient, the FSM enters Drop until MRxDV deasserts. 3) Preamble/SFD hunt: RX hunts for preamble (0x5 nibbles) followed by SFD nibble 0xD. On SFD, ByteCnt resets and CRC is initialized. 4) Byte assembly and stream control: Nibbles are assembled into bytes (high nibble then low nibble into LatchedByte); RxData registers from LatchedByte. RxStartFrm asserts on the first data byte (ByteCntEq1), or when DlyCrcCnt reaches 3 if DlyCrcEn=1. RxValid qualifies each output byte; with DlyCrcEn=1 the initial byte(s) may be suppressed and ByteCntOut presents ByteCnt+4. 5) Address check window: During DA bytes (ByteCntEq2..7), UnicastOK matches r_MAC, BroadcastOK = Broadcast & ~r_Bro, MulticastOK via 6■bit CRC hash addressing HASH1:HASH0 when CrcHashGood & Multicast. At ByteCntEq7, if none of (UnicastOK | BroadcastOK | MulticastOK | r_Pro) is true, RxAbort pulses to terminate reception; AddressMiss latches DA miss using slightly different rules (considers PassAll & ControlFrmAddressOK but not r_Pro). 6) Control/PAUSE detection: In bytes 0–17, detect control frame with DA = 01■80■C2■00■00■01 or local MAC, Type 0x8808, OpCode 0x0001; capture Pause Time at bytes 16–17. ReceivedPauseFrm pulses at byte 16; the timer is loaded at ReceiveEnd only if ReceivedPauseFrmWAddr & ReceivedPacketGood & ReceivedLengthOK & r_RxFlow=1. The Pause timer decrements in MRxClk quanta and is synchronized to TX to block new data■frame starts while non■zero. 7) Counting and limits: ByteCnt increments once per byte in StateData[1]. Oversize is flagged when ByteCnt exceeds r_MaxFL unless r_HugEn=1. Short frames are identified against r_MinFL (acceptance may be allowed by r_RecSmall). 8) CRC processing: CRC is enabled from DA through FCS; input nibbles are ordered LSB■first for the CRC core. At frame end, a correct FCS yields Crc == 0xC704DD7B (CrcError=0); ReceivedPacketGood reflects ~LatchedCrcError. 9) End■of■frame: RxEndFrm asserts when MRxDV drops after at least two bytes, or when ByteCntMaxFrame triggers. Dribble■nibble endings are normalized to a clean RxEndFrm. 10) DMA to memory: After RX BD pointer fetch, RxReady=1. Incoming bytes are packed into 32■bit words according to RxPointerLSB; full words are written immediately, and the final partial word is flushed and zero■padded. WriteRxDataToFifo pulses in MRxClk and is synchronized into the Wishbone clock to perform writes with proper byte■selects; RxPointerMSB advances on acks and bursts are used when FIFO levels allow. 11) Finalize and interrupts: On end■of■frame, ShiftEnded crosses to WB to latch length and status. RxStatusWrite updates the RX BD and raises RxB_IRQ or RxE_IRQ per aggregated error status. If a frame arrives when not RxReady, Busy_IRQ asserts; RxOverrun latches if buffering/memory cannot keep up. 12) Errors and aborts: At TakeSample/LoadRxStatus, the MAC records ShortFrame, ReceivedPacketTooBig, DribbleNibble, InvalidSymbol (sticky on MRxErr with 0xE), and RxLateCollision (half■duplex outside window). RxAbort_latch is set for fatal conditions (DA reject, ShortFrame when ~r_RecSmall, LatchedMRxErr when ~InvalidSymbol, and ReceivedPauseFrm when ~r_PassAll), synchronized to WB, and cleared by a WB■driven reset handshake. 13) Duplex/loopback interactions: In half■duplex, Transmitting inhibits RX start and late collisions are evaluated; in full■duplex RX runs concurrently with TX. With r_LoopBck=1, the full RX pipeline processes the locally transmitted stream through the same sequence.

-- Flow Control Operation (PAUSE Rx/Tx) --

IEEE 802.3x PAUSE is fully supported across RX detection/timer, TX enforcement, and TX PAUSE frame generation, with safe cross■clock synchronization and dedicated interrupts.
- RX PAUSE detection: Within first 18 bytes, accept if DA = 01-80-C2-00-00-01 (reserved multicast) or local MAC, Type=0x8808, OpCode=0x0001, CRC OK; gated by CTRLMODER.RxFlow=1.
- RX timer: On valid PAUSE, load 16■bit PauseTimer from bytes 16–17; decrements once per PAUSE quanta (512 bit■times) using MRxClk■derived slot timing; RXC interrupt via INT_SOURCE[6] when timer loads (SetPauseTimer & RxFlow).
- TX enforcement: Pause is asserted in TX domain while PauseTimer != 0; synchronized into MTxClk and updated only at TX idle/finishing points; it blocks new normal frame starts (TxStartFrmOut = TxStartFrmIn & ~Pause) but never preempts ongoing frames; control frames are exempt from Pause.
- PAUSE frame storage policy: If CTRLMODER.PassAll=0, received PAUSE frames are not stored (RX abort latch) while the timer still loads; if PassAll=1, PAUSE may be forwarded subject to address acceptance.
- TX PAUSE generation: Software sets TXCTRL.TxPauseRq and TXCTRL.TxPauseTV with CTRLMODER.TxFlow=1; a synchronized one■shot (TPauseRq) latches WillSendControlFrame when TX is available; control mux emits a PAUSE frame (DA 01-80-C2-00-00-01, SA local MAC, Type 0x8808, OpCode 0x0001, Pause Time = TxPauseTV[15:0]); pad and CRC are forced; completion clears WillSendControlFrame and generates TXC interrupt via INT_SOURCE[5]; WB side receives RstTxPauseRq to clear TxPauseRq.
- Clock■domain crossings: r_TxPauseRq (WB) -> TPauseRq (MTxClk) via multi■FF sync/edge pulse; WillSendControlFrame (MTxClk) -> RstTxPauseRq (WB) via multi■FF sync/edge; PauseTimerEq0 (MRxClk) -> TX domain Pause update via multi■FF sync.
- Timing: PAUSE quanta = 512 bit■times; slot timer derived from MRxClk decrements the PauseTimer per quanta.
- Configuration dependencies: CTRLMODER.TxFlow enables TX PAUSE generation and TXC IRQ; CTRLMODER.RxFlow enables RX PAUSE timer and RXC IRQ; DlyCrcEn is handled for control frame alignment; duplex mode does not gate PAUSE operation.

-- MDIO Transactions (Read/Write/Scan) --

Defines how the MIIM/MDIO master executes Write, Read, and Scan transactions via eth_miim under software control of the MIIM register set.

General operation
- Commands: MIICOMMAND.WCtrlData (write), MIICOMMAND.RStat (read), MIICOMMAND.ScanStat (scan). Each command is edge-detected (one-shot); do not hold bits high for repeats.
- Exclusivity: Only one operation may be pending or active at a time. New requests are inhibited while Busy is asserted.
- Clocking: MDC is generated by eth_clockgen using MIIMODER.r_ClkDiv (50% duty). MdcEn pulses on MDC rising edges; MdcEn_n on falling edges. Odd divisors clamp down to the next lower even divide.
- Preamble: If MIIMODER.r_MiiNoPre=0, a 32-bit 1s preamble precedes the frame. If r_MiiNoPre=1 (NoPre), frame fields start immediately (bit counter starts at header).

Busy and completion
- Busy_stat (MIISTATUS[1]) is asserted while an operation is active, during scan pending alignment, or while a synchronized scan request exists.
- EndBusy is a 1-cycle pulse on the falling edge of InProgress (transaction end).
UpdateMIIRX_DATAReg pulses with EndBusy for read/scan (suppressed for write).
- MIIRX_DATA updates only on read/scan at EndBusy; update is suppressed if a write starts in the same cycle a read/scan completes.
- NValid_stat (MIISTATUS[2]) reflects a pending/valid scan window; contributes to Busy.

Frame timing and structure (with preamble enabled)
- BitCounter advances on MdcEn and defines frame landmarks:
- 0..31: preamble (MDIO driven high, MdoEn=1).
- 0x20: header byte 0 (Start=01, OP, PHYAD[4:1]).
- 0x28: header byte 1 (PHYAD[0], REGAD[4:0], TA for write=10).
- 0x30: data byte 1 (data[15:8], writes only).
- 0x38: data byte 0 (data[7:0], writes only).
- 0x37: LatchByte1 (capture read data[15:8]).
- 0x3F: LatchByte0 (capture read data[7:0]) and EndOp.
- No preamble mode: frame begins at the header timing (counter initialized to skip 0..31).

Serialization and turnaround
- eth_shiftreg builds header/data bytes, shifts MSB-first on MdcEn_n, and latches read data at 0x37/0x3F.
- eth_outputcontrol drives MDIO:
- Preamble and writes: MdoEn=1 for all bits (including TA and data).
- Reads: MdoEn=1 through header, then deasserted for turnaround and 16 data bits so the PHY drives MDI; master reasserts only after data completes.
- LinkFail (MIISTATUS[0]) is derived on reads when r_RGAD==1 (PHY status reg) during the low-byte latch.

Transaction types
- Write (WCtrlData):
1) Program MIIMODER (r_ClkDiv, r_MiiNoPre) and MIIADDRESS (r_FIAD, r_RGAD). Load MIITX_DATA with r_CtrlData[15:0].
2) Assert MIICOMMAND.WCtrlData. The request is edge-detected and aligned to MdcEn; InProgress asserts with WriteOp=1.
3) Frame emits preamble (if enabled), header, TA=10, then data[15:0]. MDIO remains driven for the entire frame.
4) At BitCounter 0x3F, EndBusy pulses; Busy_stat deasserts. MIIRX_DATA is not updated.

- Read (RStat):
1) Program MIIMODER and MIIADDRESS.
2) Assert MIICOMMAND.RStat. Request is edge-detected; InProgress asserts with WriteOp=0.
3) Master drives preamble and header, then tri-states for turnaround and data. PHY returns 16 data bits; eth_shiftreg latches at 0x37 (upper) and 0x3F (lower).
4) EndBusy pulses at 0x3F; UpdateMIIRX_DATAReg updates MIIRX_DATA. LinkFail may update if REGAD==1.

- Scan (ScanStat):
1) Assert MIICOMMAND.ScanStat at any time. Request is synchronized to MdcEn and starts only when the bus is idle: SyncStatMdcEn & ~InProgress & ~InProgress_q1 & ~InProgress_q2.
2) NValid_stat indicates a pending/valid scan window; Busy_stat stays high while a scan is pending or active.
3) Executes a read-type frame and latches data as in Read. On EndBusy, UpdateMIIRX_DATAReg updates MIIRX_DATA. Re-assert ScanStat to continue periodic scans.

Software usage and constraints
- Sequence: program MIIMODER and MIIADDRESS; for writes, load MIITX_DATA; assert exactly one command bit (pulse); poll MIISTATUS.Busy_stat until 0; on read/scan, sample MIIRX_DATA after Busy clears; optionally check MIISTATUS.LinkFail when reading REG 1.

- Corner case: If a write starts in the same cycle a read/scan ends, MIIRX_DATA will not update; avoid back-to-back write issue on the exact completion edge.
- Single-operation constraint: overlapping requests are blocked; EndBusy clears command one-shots to prevent retriggering.

-- Interrupt Handling and RW1C Clearing --

Interrupt Handling and RW1C Clearing

- Interrupt sources and bit mapping (INT_SOURCE[6:0])
- [0] TXB: Transmit buffer/status written without error (TxB_IRQ)
- [1] TXE: Transmit error (TxE_IRQ)
- [2] RXB: Receive buffer/status written without error (RxB_IRQ)
- [3] RXE: Receive error (RxE_IRQ)
- [4] BUSY: RX not ready when a frame started (Busy_IRQ)
- [5] TXC: Flow-control transmit event complete (SetTxCIrq via CDC)
- [6] RXC: Flow-control receive event (SetRxCIrq via CDC)
- Upper bits read as 0; writes to them have no effect.

- Latching, masking, and int_o (wb_clk_i domain)
- Each bit has a level latch set by a single-cycle event pulse synchronized to wb_clk_i.
- INT_SOURCE reflects current latched status; additional events while a bit is set do not queue (coalesce).
- INT_MASK gates the interrupt output only; latching is unaffected. int_o = OR over i=0..6 of (INT_SOURCE[i] & INT_MASK[i]).
- wb_rst_i clears all latches and masks; int_o deasserts.

- RW1C clearing semantics (INT_SOURCE)
- Read/write-one-to-clear per bit: writing 1 to bit [6:0] clears that latch; writing 0 leaves it unchanged.
- Standard Wishbone byte-selects apply; only bits with corresponding selected byte lanes are affected.
- Multiple bits may be cleared in one write; mask state does not affect clearing.
- Clearing affects only the CPU-visible latches and does not modify descriptors, status memory, or other logic.
- If an event pulse coincides with a clear, the bit may reassert; software should read back INT_SOURCE to confirm clearing and repeat if needed.

- Event generation and CDC
- TXB/TXE/RXB/RXE/BUSY originate in eth_wishbone (wb_clk_i domain) as one-cycle pulses (e.g., on TxStatusWrite/RxStatusWrite with per-BD IRQ enable, and Busy fault detection).
- TXC (bit 5) and RXC (bit 6) originate in MTxClk and MRxClk domains, are triple-synchronized into wb_clk_i, and edge-detected into single-cycle set pulses.
- For TXC/RXC only, the source-domain "set" flops are auto-cleared by a return handshake from the wb_clk_i pulse; RW1C clears only the CPU-visible latch.

- Not generated by INT_SOURCE
- Wishbone slave ack/error responses and MIIM (MDIO) Busy/NValid/LinkFail do not map to INT_SOURCE; these are accessed via bus responses or MIISTATUS and must be polled.

- Software usage
- Enable desired sources in INT_MASK.
- On int_o assertion, read INT_SOURCE to identify causes.
- Service conditions and clear by writing 1s to the serviced bit positions in INT_SOURCE (with correct

byte enables).
- If coalescing is expected, clear all serviced bits together and verify by re-reading INT_SOURCE.

-- Error Handling and Recovery --

eth_top implements end-to-end error detection, containment, and recovery across Wishbone, TX/RX datapaths, flow control, and MIIM. Wishbone slave: unmapped address or no byte select generates a single-cycle wb_err_o; valid register/BD accesses generate a single-cycle wb_ack_o; read-data muxing is glitch-free to prevent spurious ack/err. Wishbone master (DMA): m_wb_ack_i or m_wb_err_i both terminate a transaction; on error, the cycle ends cleanly and pointers advance to avoid hangs; software infers anomalies via per-packet status and interrupts. TX (half-duplex aware): in-window collision triggers 8-nibble jam, LFSR-based binary exponential backoff, and retry up to r_MaxRet; late collision, underrun, oversize (unless r_HugEn), or excessive defer cause abort; MTxErr and latched results (RetryCnt, RetryLimit, LateColl) report outcomes; maccontrol produces one-shot Done/Abort only when TX data is consumed and masks at frame start and during control-frame insertion to avoid false terminations; TX FIFOs are cleared on abort/retry; carrier/collision are double-synchronized. RX: IFG violations, oversize (unless r_HugEn), short frame (when r_RecSmall=0), RX-while-TX in half-duplex, CRC and symbol errors, dribble nibble, and late-collision classification are detected; address filtering asserts RxAbort when no DA match (unless r_Pro); RX overrun and not-ready conditions are latched and can raise Busy_IRQ; erroneous frames enter Drop until MRxDV deasserts, then return to Idle. RX abort latch: set in MRxClk on fatal conditions (address reject, short with r_RecSmall=0, MRxErr without InvalidSymbol, received PAUSE with r_PassAll=0); synchronized to WB for visibility and cleared via a two-way WB->RX handshake (RxAbortRst), ensuring deglitched, metastability-safe recovery under software control. Flow control: received PAUSE loads a timer and gates TxStartFrmOut while active; transmitted PAUSE frames seize TX, block TxUsedData, and are released cleanly at frame end (BlockTxDone prevents races). MIIM: operations are edge-detected and aligned to MdcEn; Busy prevents overlap; EndBusy one-shot signals completion; read-data updates only on read/scan completes to avoid stale results; no explicit PHY error flags are handled. Clock-domain crossings: events and status use multi-flop synchronizers and one-shot pulse generation; RxEn updates only when MRxDV=0; external carrier/collision/loopback inputs are double-registered. Reporting and software recovery: per-packet TX/RX status in WB domain includes detailed error qualifiers; interrupts are RW1C (TxB/E, RxB/E, Busy, flow-control events). Software clears RW1C latches and the RX abort latch via handshake, after which TX returns to Defer/IPG/Idle and RX returns to Idle.

-- Cross-Domain Handshakes and Edge Pulses --

Clock domains and general rules
- Domains: wb_clk_i (system/Wishbone), MTxClk (TX/MII), MRxClk (RX/MII). MIIM logic stays in wb_clk_i and uses MdcEn/MdcEn_n as sub-cycle strobes.
- CDC method: Status/control transfers use 2–3 flip-flop synchronizers. Edge pulses in the destination domain are generated from synchronized levels using rising-edge detection (sync2 & ~sync3) to produce single-cycle one-shots.
- Handshake style: Source domain sets a level latch; destination domain detects a rising edge and processes it once; an acknowledgment is synchronized back to clear the source latch. This guarantees exactly-once processing and avoids retriggering.

WB↔TX Flow-control pause handshake
- WB→TX request: r_TxPauseRq (wb_clk_i) is triple-synchronized into MTxClk. A rising-edge detector produces TPauseRq as a one-cycle pulse in MTxClk to start a PAUSE frame when enabled.
- TX→WB completion: WillSendControlFrame (MTxClk) is triple-synchronized into wb_clk_i; a rising-edge one-shot (RstTxPauseRq) clears the pending software request.

RX↔WB Abort handshake

- RX set/hold: RxAbort_latch asserts in MRxClk on fatal RX conditions and holds until cleared.
- RX→WB report: RxAbort_latch is synchronized into wb_clk_i as a clean level (RxAbort_wb) for software/BD processing.
- WB→RX clear: RxAbort_wb is synchronized back into MRxClk (RxAbortRst) to clear RxAbort_latch, ensuring a single report per event.

Flow-control IRQ handshakes (TX/RX↔WB)
- TX IRQ: SetTxCIrq_txclk set in MTxClk and triple-synchronized into wb_clk_i; rising-edge detection creates a one-cycle SetTxCIrq pulse. ResetTxCIrq is synchronized back to MTxClk to clear the TX-domain set.
- RX IRQ: SetRxCIrq_rxclk set in MRxClk and triple-synchronized into wb_clk_i; rising-edge detection creates a one-cycle SetRxCIrq pulse. ResetRxCIrq is synchronized back to MRxClk to clear the RX-domain set.

TX/RX results and status pulses into WB
- TX results: TxDone, TxAbort, TxRetry, and TxUnderRun are synchronized into wb_clk_i and converted to dedicated one-shot pulses (…Pulse) to drive BD/status updates. Blocking flags prevent duplicate writes across domains.
- RX data/status: WriteRxDataToFifo (MRxClk pulse) is synchronized into wb_clk_i (WriteRxDataToFifo_wb) for Wishbone writes. ShiftEnded is latched in MRxClk at frame end and synchronized to wb_clk_i to trigger a single RxStatusWrite.
- Busy IRQ: An RX start-of-frame while the engine is not ready sets Busy_IRQ via a synchronized indication into wb_clk_i.

Domain-crossing indications and gating
- Carrier sense (pads→TX): mcrs_pad_i sampled through two FFs; TxCarrierSense is the synchronized level gated by ~r_FullD.
- Collision (pads→TX): Collision is sampled and level-latched until ResetCollision; effective Collision = ~r_FullD & latched level.
- Transmit intent (TX→RX): WillTransmit is double-synchronized into MRxClk and gated by ~r_FullD to form Transmitting for RX state machines.
- RX enable (WB→RX): r_RxEn is sampled into MRxClk as RxEnSync only when MRxDV=0 to avoid mid-packet glitches.

Pause enforcement across domains
- MRxClk→MTxClk: PauseTimerEq0 is double-synchronized into MTxClk. Pause gating in TX updates only at TX idle/finish points (no active data usage) to avoid hazards.
- TX→RX observation: Pause is observed in MRxClk for slot-time generation; it changes only at TX idle update points, minimizing CDC risk.

MIIM pulses (wb_clk_i domain with MDC strobes)
- Start one-shots: WCtrlDataStart and RStatStart are formed by rising-edge detection on triple-registered requests and aligned to MdcEn/MdcEn_n to create single-cycle WriteDataOp/ReadStatusOp enables.
- Completion pulses: EndBusy is a one-cycle pulse on InProgress falling edge. UpdateMIIRX_DATAReg is a one-shot when read/scan completes and no write begins that cycle. ScanStart is edge/strobe aligned; Nvalid holds a pending-scan level until completion.

Intra-domain edge shaping (clarifying pulses vs levels)
- TX datapath: StartPreamble/StartData/StartPAD/StartFCS/StartJam/StartBackOff are single-cycle strobes. PacketFinished_d and PacketFinished are latched pulses at terminal events; PacketFinished_q provides clean edge detection.

- RX datapath: RxStartFrm/RxEndFrm are single-cycle pulses aligned to RxData via pipelines; DribbleRxEndFrm ensures a clean end pulse on half-byte endings.
- MAC control: MuxedDone/MuxedAbort are one-shot, edge-cleaned versions of TxDoneIn/TxAbortIn, gated by actual data usage (TxUsedDataOutDetected) to avoid spurious early pulses.
- MAC status (RX side): TakeSample (end event), LoadRxStatus (registered next-cycle pulse), and ReceiveEnd (two-cycle delayed pulse) form a small pipeline to sample per-frame status exactly once.

Reset behavior and safeguards
- Resets: wb_rst_i is asynchronous; each domain resynchronizes local clears. Source-domain latches clear on reset and upon receiving destination-domain acknowledgments.
- Safe update points: Pause updates only when (~TxStartFrmOut & (TxDoneIn | TxAbortIn | ~TxUsedDataOutDetected)); RxEnSync only when MRxDV=0.
- Duplex gating: Half/full duplex affects relevance of CarrierSense and Collision via ~r_FullD masks.
- CDC discipline: All cross-domain communications use 2–3 FF synchronizers; destination pulses are generated by rising-edge detection; bidirectional handshakes use level latches plus synchronized acknowledgments to prevent metastability and double-counts.

-- Loopback Mode Behavior --

Loopback mode is enabled by MODER[7] (r_LoopBck=1). When enabled, the RX MII inputs are internally driven by the MAC's TX MII outputs instead of the external RX pads: MRxD <= MTxD[3:0], MRxDV <= MTxEn & RxEnSync, MRxErr <= MTxErr & RxEnSync. RxEnSync is generated in the MRxClk domain, updates only when MRxDV is low to avoid mid-frame glitches, and forces MRxDV/MRxErr low when r_RxEn=0 so RX is suppressed even in loopback. No additional clock-domain synchronization is added; TX-side signals are sampled by RX logic in the MRxClk domain, so correct operation assumes MRxClk present and compatible with MTxClk (typical MII test conditions). In full-duplex (r_FullD=1), TX and RX can run concurrently; looped frames are received immediately, enabling self-reception tests. In half-duplex (r_FullD=0), the RX FSM is inhibited while transmitting via Transmitting (WillTransmit synchronized into MRxClk), so RX is generally blocked during TX; collision/backoff remains governed by external carrier-sense/collision inputs. Carrier-sense (mcrs_pad_i) and collision (mcoll_pad_i) continue to come from the pads and are gated by ~r_FullD; loopback does not synthesize them. CarrierSenseLost status is masked during loopback; all other TX/RX status and interrupts (CRC/length/short/too big/dribble/underrun/late collision/retry/abort/done) behave as in normal mode. The full RX pipeline (preamble/SFD detect, IFG, byte assembly, CRC check, address filtering, length checks, multicast hash) applies to looped frames; frames not accepted by filtering are aborted. Flow control is active: looped MAC Control PAUSE frames are detected (when enabled and address-accepted), and the extracted pause timer gates new TX frame starts, allowing self-pausing. Only the RX data/valid/error pads are bypassed; MDIO/MDC and MIIM remain unaffected. Deasserting r_RxEn prevents looped data from generating RX activity. For half-duplex tests requiring simultaneous TX/RX observation, use full-duplex or account for the RX inhibit during transmission.

-- Timing and Latency Considerations --

- Clocks, resets, and CDC: Three domains (wb_clk_i, mtx_clk_pad_i, mrx_clk_pad_i). wb_rst_i is async and re-synchronized per domain. CDC uses 2-flop chains (CarrierSense→TX, WillTransmit→RX; ~2 dest cycles) and 3-flop chains (WillSendControlFrame→wb_clk_i, r_TxPauseRq→MTxClk, RX abort handshake; ~3 dest cycles). RX enable changes are sampled only when MRxDV=0, deferring mid-packet enable/disable until inter-nibble idle.
- MII TX timing (MTxClk): MTxD/MTxEn/MTxErr registered. Preamble+SFD: 16 nibbles; Jam: 8 nibbles. CRC runs nibble-parallel; Initialize during Idle/Preamble or DlyCrc window; FCS nibbles emitted in StateFCS. Minimum-frame enforcement inserts PAD until 64 bytes; StartTxDone at FCS nibble 7 (no collision) or at defined data-pad conditions (no CRC). ExcessiveDefer threshold is a fixed nibble count

(masked by ExDfrEn).
- MII RX timing (MRxClk): IFG enforced via 24-nibble counter; early SFD before IFG forces Drop. Bytes assembled high nibble then low; data/control have up to 2-cycle pipeline. SOF depends on DlyCrcEn (first byte vs after 3-nibble delay); EOF aligns to MRxDV drop or max-frame boundary, dribble-safe. CRC Initialize at SFD and early delay window; Enable only in data; CrcError sampled at data end. Multicast hash captured at DA byte 6; address filter decision at byte 7 issues a one-cycle RxAbort.
- Flow control (PAUSE): RX PAUSE loads PauseTimer at valid frame end; decrements once per 512 bit-times via Divider2+SlotTimer. Pause state crosses to TX with 3-flop sync; TX updates only when idle/finishing ((TxDoneIn | TxAbortIn | ~TxUsedDataOutDetected) & ~TxStartFrmOut), adding several MTxClk cycles of latency plus gating. TX PAUSE requests triple-synced to MTxClk and one-shot pulsed; control-frame injection begins when CtrlMux can seize (~TxUsedDataOut=0) and handshakes permit; DlyCrc may shift initial bytes.
- Wishbone slave/master: wb_ack_o/wb_err_o are registered one-cycle pulses and may be delayed one wb_clk cycle relative to decode. Register reads return with ack (implementation-dependent single-cycle) or via the ack pulse pipeline. BD/SRAM reads add 1 wb_clk cycle (registered address). Master bursts use cti_o=010 (incrementing) and terminate with 111; typical 4-beat bursts when FIFO levels permit. TX reads are full-word; RX writes use sel per alignment; last-beat valid-byte handling may introduce a final-cycle stall.
- Internal RAM/FIFOs: eth_spram_256x32 has 1-cycle read latency; byte-write enables are write-first; dato drives only when ce&oe.; eth_fifo has 1-cycle data_out latency after read pointer advance; avoid read+write at empty/full; synchronous clear re-bases pointers and data_out to entry 0 with special-case if read/write asserted.
- MIIM/MDIO: MDC from eth_clockgen; MdcEn/MdcEn_n are one-Clk-cycle strobes per MDC edge. MDC ≈ Clk/(2*floor(Divider/2)); odd Divider rounds down for 50% duty. Operations align to MdcEn; InProgress spans the 64-bit frame (including preamble). EndBusy is a one-cycle pulse on falling edge of InProgress. Read-byte latches at BitCounter 0x37 and 0x3F; write/scan strobes at 0x20/0x28/(0x30/0x38). eth_outputcontrol adds a 2-cycle enable-qualified pipeline to Mdo/MdoEn relative to MdcEn_n; upstream shifts must account for this latency.
- Collision/carrier and TX/RX interaction: CarrierSense sampled into TX via 2 flops; matters in half-duplex. Collision is level-latched until ResetCollision when not transmitting. WillTransmit synchronized into RX via 2 flops, forming Transmitting that blocks RX; ~2 MRxClk cycles latency.
- Status/interrupts: RX status sampling is a two-cycle pipeline (TakeSample → LoadRxStatus → ReceiveEnd). TX status latched on StartTxDone or StartTxAbort. Interrupts edge-detected into wb_clk_i via 3-flop CDC, producing single-cycle pulses; write-one-to-clear. Expect ~3 wb_clk cycles from source event to int_o assertion.
- DMA framing/alignment: TX length decrements per ack with pointer alignment considered; last-beat valid-byte handling may stall one wb_clk cycle; TxStartFrm/TxEndFrm cross to MTxClk; underrun crosses to WB for status. RX word assembly and WriteRxDataToFifo pulse occur in MRxClk; end-of-shift crosses to WB to trigger RxStatusWrite; partial final word flushed with zero padding.
- General expectations: Cross-domain event propagation typically 2–3 cycles in the destination domain plus intentional gating. RX control signals can lag data by up to 2 MRxClk cycles; TX state changes take effect at the next nibble boundary. MIIM transactions consume 64 MdcEn ticks; completion strobes are single-cycle with a 2-cycle Mdo/MdoEn pipeline. Wishbone BD RAM adds 1-cycle read latency; ack/err are pulsed and may be delayed one wb_clk cycle; master-side latency depends on m_wb_ack_i and burst capability.

REGISTERS

-- Register Map Overview --

Register space selection and addressing
- Wishbone slave, 32-bit data, word-aligned addressing. Register space is selected when wb_adr_i[11:10] = 2'b00.
- Buffer Descriptor/Internal SRAM window is selected when wb_adr_i[11:10] = 2'b01 (not part of the register map below).
- Any wb_adr_i[11:10] = 2'b1x is an address miss and returns a bus error on access.
- Register addresses below are word offsets within the register space (i.e., wb_adr_i[11:2]).
- Byte enables (wb_sel_i[3:0]) gate per-byte writes; a write with no byte lanes selected (wb_sel_i == 0) is a bus error. Unwritten bytes retain previous values.
- Unmapped register offsets read as 0; writes to unmapped locations are ignored.

Register map (word offsets)
- 0x00 MODER (RW): MAC mode/control. Fields include RxEn, TxEn, Pro, Iam, Bro, NoPre, IFG, LoopBck, NoBckof, ExDfrEn, FullD, DlyCrcEn, CrcEn, HugEn, Pad, RecSmall. TxEn/RxEn are effectively gated by TX_BD_NUM (see TX_BD_NUM note).
- 0x01 INT_SOURCE (RW1C): Latched interrupt sources; write 1 to a bit clears that source. Bits [6:0] correspond to TxB, TxE, RxB, RxE, Busy, TxC, RxC.
- 0x02 INT_MASK (RW): Interrupt mask; int_o is the OR of (INT_SOURCE & INT_MASK) for bits [6:0].
- 0x03 IPGT (RW): Back-to-back inter-packet gap (TX).
- 0x04 IPGR1 (RW): Non-back-to-back IPG part 1 (TX).
- 0x05 IPGR2 (RW): Non-back-to-back IPG part 2 (TX).
- 0x06 PACKETLEN (RW): MinFL[31:16], MaxFL[15:0] frame length limits.
- 0x07 COLLCONF (RW): Collision/backoff configuration. Includes MaxRet and CollValid[5:0]. Some upper/unused bits read as 0.
- 0x08 TX_BD_NUM (RW, range-limited): Number of TX buffer descriptors (0..0x80). Partitions TX vs RX BD rings; RX ring starts at index TX_BD_NUM. Writes outside 0..0x80 are ignored. Tx/Rx enable gating: r_TxEn = MODER.TxEn & (TX_BD_NUM > 0); r_RxEn = MODER.RxEn & (TX_BD_NUM < 0x80).
- 0x09 CTRLMODER (RW): Control frame options: TxFlow, RxFlow, PassAll.
- 0x0A MIIMODER (RW): MII management clock divider and NoPreamble control.
- 0x0B MIICOMMAND (RW, self-clearing pulses): WCtrlData, RStat, ScanStat initiate MIIM write, read, or scan actions.
- 0x0C MIIADDRESS (RW): PHY and register address fields FIAD[4:0], RGAD[4:0].
- 0x0D MIITX_DATA (RW): MII management write data CtrlData[15:0].
- 0x0E MIIRX_DATA (RO): MII management read data, updated by hardware upon MIIM read/scan completion; software writes are ignored.
- 0x0F MIISTATUS (RO): MIIM status. [2] NValid, [1] Busy, [0] LinkFail. Busy indicates in-progress/pending scan; NValid indicates rx data not valid.
- 0x10 MAC_ADDR0 (RW): MAC address low 32 bits (MAC[31:0]).
- 0x11 MAC_ADDR1 (RW): MAC address high 16 bits in [15:0] (MAC[47:32]). Upper bits read as 0.
- 0x12 HASH0 (RW): Multicast hash table lower 32 bits.
- 0x13 HASH1 (RW): Multicast hash table upper 32 bits.
- 0x14 TXCTRL (RW/WO pulse): [16] TxPauseRq (self-clearing pulse when a pause/control frame is sent), [15:0] TxPauseTV pause time value.
- 0x16 dbg_dat (RO): Debug visibility register.

General register behaviors
- Unless specified, upper/unused bits read as 0 and are not writable.
- MIIM path: Writing MIICOMMAND bits triggers operations; MIIRX_DATA updates on operation completion; MIISTATUS.Busy reflects activity.

- Default reads for addresses with no active read strobe return 0 from the register block; unmapped offsets also return 0.
- Handshake: Valid register accesses produce a single-cycle ACK; address misses or writes with no byte selected produce a single-cycle ERR.

-- Configuration Registers (MODER, CTRLMODER) --

Addresses and access: MODER at 0x00 (byte lanes 0–2 writable); CTRLMODER at 0x09 (lower byte writable). Writes honor Wishbone byte enables. Unimplemented/upper bits read back as 0. Reset: Both registers reset to implementation-defined defaults (typically 0x00000000). Readback: Reserved/read-only/unimplemented bits read as 0.

MODER fields (effects across TX/RX/flow-control/MIIM):
- [16] r_RecSmall: Allow RX of undersized frames; if 0, short frames trigger RX abort.
- [15] r_Pad: Enable TX padding to 64-byte minimum.
- [14] r_HugEn: Permit oversize (“huge”) frames; if 0, MaxFL enforced in TX/RX.
- [13] r_CrcEn: Enable CRC/FCS generation on TX.
- [12] r_DlyCrcEn: Enable delayed-CRC; engages small delay counters and adjusted start/valid timing in TX/RX.
- [10] r_FullD: Full-duplex; masks carrier-sense/collision handling when 1.
- [9] r_ExDfrEn: Mask excessive-defer detection when 1 (active-low enable of detector).
- [8] r_NoBckof: Disable binary exponential backoff when 1.
- [7] r_LoopBck: Internal MII loopback; RX path is sourced from TX; MRxDV/MRxErr gated by synchronized RX enable to prevent RX while disabled.
- [6] r_IFG: Override IFG requirement; RX IFG counter reports IFG satisfied.
- [5] r_Pro: Promiscuous mode; bypass DA checking and accept all frames.
- [4] r_Iam: Individual-address mode (reserved/read-only in this context).
- [3] r_Bro: Broadcast disable; masks broadcast acceptance.
- [2] r_NoPre: Disable MAC TX preamble insertion (distinct from MIIM NoPre).
- [1] r_TxEn: TX enable; hard-gated by descriptor partition (effective only if TX_BD_NUMOut > 0).
- [0] r_RxEn: RX enable; hard-gated by descriptor partition (effective only if TX_BD_NUMOut < 0x80).

CTRLMODER fields (MAC control/PAUSE):
- [2] r_TxFlow: Enable TX PAUSE frame generation; transmit-control can seize TX to inject PAUSE on request.
- [1] r_RxFlow: Enable RX PAUSE processing; gates pause timer load/decrement; Pause is synchronized into TX domain to block normal frame transmission while active.
- [0] r_PassAll: Pass all MAC control frames; when 0, specific control frames contribute to RX abort; when 1, such frames are passed and status latched.

Partitioning and synchronization: TX/RX enables are hard-gated by TX_BD_NUMOut to ensure valid descriptor space. MODER bits are synchronized into respective TX/RX clock domains; RX enable uses an MRxDV-low sampling method to avoid mid-packet glitches.

Implementation notes: MODER is decoded/stored in eth_registers and its r_* outputs drive TX state machine, counters, RX address-check, MAC-control, and MIIM paths. r_Pad and r_CrcEn propagate to TX/MAC control; control frames may force Pad/CRC irrespective of MODER while SendingCtrlFrm. r_NoBckof affects backoff entry; r_IFG forces IFG satisfied; r_DlyCrcEn alters CRC initialization windows and start/valid timing. MIIM’s NoPre (r_MiiNoPre) is separate from MAC’s r_NoPre.

-- Timing Registers (IPGT, IPGR1, IPGR2) --

IPGT, IPGR1, and IPGR2 are programmable timing registers that define the inter-packet gap (IPG) enforced by the transmit path. They are memory-mapped in the Wishbone register space at addresses 0x03 (IPGT), 0x04 (IPGR1), and 0x05 (IPGR2). Each register has an effective field width of 7 bits; upper readback bits are forced to 0. Writes honor Wishbone byte enables, updating only the active lanes; unused upper bits always read as 0. Values are specified in nibble-time units (MTxClk cycles on MII), where 24 nibbles correspond to the IEEE 802.3 minimum IFG of 96 bit-times.

The TX state machine (eth_txstatem) and counters (eth_txcounters) consume these registers in the MTxClk domain. The nibble counter (NibCnt) times the IPG and is compared against the selected threshold to assert StartIdle. In full-duplex, IPGT is the single-phase IPG target; StartIdle asserts when NibCnt reaches IPGT. In half-duplex, IPG is two-phase: IPGR1 defines an early window during which CarrierSense forces immediate return to Defer per CSMA/CD, and IPGR2 defines the completion threshold at which StartIdle asserts. An internal Rule1 latch selects which register governs the IPG: Rule1 is set when entering Preamble or when r_FullD=1, and cleared in Idle or BackOff. When Rule1=1, StartIdle uses IPGT; otherwise, it uses IPGR2. The IPG begins on StartIPG when leaving Defer and the medium is idle; after the programmed IPG completes, the FSM transitions to Idle, allowing a new frame if TxStartFrm is present.

These registers are written/read in the Wishbone clock domain and safely sampled into MTxClk for TX logic. They are not used by RX IFG enforcement, which has separate controls. No default/reset values are specified; software must program IPGT/IPGR1/IPGR2 to meet system requirements for the chosen duplex mode.

-- Length/Collision Registers (PACKETLEN, COLLCONF) --

Length/Collision Registers configure frame length limits and CSMA/CD collision behavior.

- Address map and width
- PACKETLEN @ 0x06, 32-bit. All four byte lanes writable.
- COLLCONF @ 0x07, 32-bit. Byte lane 1 (bits 15:8) is not writable; other lanes honor byte enables.

- PACKETLEN (frame length limits; units = bytes including FCS)
- Fields: MinFL = [31:16], MaxFL = [15:0].
- Write/read: All bytes writable; readback of any reserved bits returns 0.
- TX usage (MTxClk domain):
- Padding control: MinFL sets the minimum frame size prior to FCS. TX counters derive a nibble-aligned threshold from (MinFL - 4); TX state machine inserts PAD until this threshold then proceeds to FCS.
- Oversize enforcement: MaxFL compared to TX ByteCnt; when ByteCnt == MaxFL and Huge mode (r_HugEn) is 0, MaxFrame triggers oversize handling (TooBig/StartTxAbort).
- RX usage (MRxClk domain):
- Byte-count limit: When ByteCnt reaches MaxFL and r_HugEn is 0, frame termination/drop is asserted.
- Length status: ShortFrame if RxByteCnt < MinFL; ReceivedPacketTooBig if RxByteCnt > MaxFL (status still flagged even when r_HugEn=1); ReceivedLengthOK when MinFL ≤ RxByteCnt ≤ MaxFL.
- Short-frame policy: If r_RecSmall == 0, ShortFrame causes RxAbort latch (frame rejection).
- Huge mode: r_HugEn disables MaxFL enforcement in both TX and RX; status signals still reflect too-big conditions.

- COLLCONF (collision retry/window; units = bytes from start of frame)
- Fields: MaxRet[3:0] = [19:16]; CollValid[5:0] = [5:0]. Bits 15:8 are not writable.
- Write/read: Writes to byte lane 1 ignored; readback of reserved/disabled bits is 0.
- TX collision handling (MTxClk domain):

- Retry limit: RetryCnt is compared to MaxRet; reaching MaxRet asserts RetryMax and contributes to abort (no further retries).
- Collision window: A ColWindow flag is cleared when TX ByteCnt reaches CollValid during data/pad or FCS at the appropriate nibble boundary. Collisions after ColWindow clears are LateCollision and force immediate abort without backoff/retry.
- Backoff sequencing: Standard random backoff/retry occurs for collisions within the window, bounded by MaxRet; late collisions bypass backoff.
- RX late-collision qualification (MRxClk domain):
- macstatus derives RxColWindow from CollValid and flags RxLateCollision when a collision is detected after the window in half-duplex (~r_FullD).

- Cross-domain behavior
- r_MinFL, r_MaxFL, r_MaxRet, r_CollValid are generated in the system/register clock domain and treated as static configuration in MTxClk and MRxClk domains (slow-changing, no explicit handshake).

- Notes
- All length thresholds include the FCS; CollValid is a byte threshold applied to the running frame ByteCnt.
- Naming references: MaxFrame/TooBig, ShortFrame, ReceivedLengthOK, RxLateCollision, RetryMax, ColWindow are internal logic outcomes driven by these registers.

-- Buffer Descriptor Partition (TX_BD_NUM) --

Buffer Descriptor Partition (TX_BD_NUM) defines the split between transmit (TX) and receive (RX) buffer descriptors in the internal BD RAM. Key properties: - Location/width: Wishbone register at 0x08. Programmed as 8-bit value; readback returns this 8-bit value with upper bits read as 0. - Valid range: 0x00..0x80 (0..128). Writes with value > 0x80 are not accepted (register retains previous value). - Partition semantics (128 total BDs): TX BDs occupy indices [0 .. TX_BD_NUM-1]; RX BDs occupy indices [TX_BD_NUM .. 127]. Each BD consumes two 32-bit words in the internal 256x32 RAM; addressing forms as {BD_index[6:0], word_select}. - Enable gating: Transmit enable = MODER[1] AND (TX_BD_NUM > 0). Receive enable = MODER[0] AND (TX_BD_NUM < 0x80). Thus TX_BD_NUM=0 forces TX disabled; TX_BD_NUM=0x80 forces RX disabled. - Descriptor engine behavior: On RX enable, RxBDAddress initializes to TX_BD_NUM so RX fetches from the RX region; RX wraps back to TX_BD_NUM on the RX wrap status bit. TX wraps back to index 0 on the TX wrap status bit. - Software guidance: Choose 0 < TX_BD_NUM < 128 for full-duplex operation and ensure wrap bits are programmed to bound each ring within its region. Changing TX_BD_NUM while TX/RX engines are active is not defined; update the partition with engines disabled to avoid descriptor overlap.

-- MIIM Registers (MIIMODER, MIICOMMAND, MIIADDRESS, MIITX_DATA, MIIRX_DATA, MIISTATUS) --

MIIM Registers overview and behavior within MAC register space
Address map
- 0x0A MIIMODER
- 0x0B MIICOMMAND
- 0x0C MIIADDRESS
- 0x0D MIITX_DATA
- 0x0E MIIRX_DATA (CPU read-only)
- 0x0F MIISTATUS

MIIMODER (PHY management timing and format)
- Fields

- [8] NoPre: 1 omits the 32-bit MDIO preamble; 0 inserts preamble ones.
- [7:0] ClkDiv: MDC clock divider.
- Behavior
- Divider values less than 2 are forced to 2 to preserve a valid MDC duty cycle.
- MDC frequency = Clk / (2 * floor(ClkDiv/2)); for even ClkDiv, MDC = Clk / ClkDiv.
- NoPre controls preamble emission for all transactions.
- Readback: bits [31:9] read as 0.

MIICOMMAND (operation requests)
- Fields (write 1 to request)
- [2] WCtrlData: start a single write of MIITX_DATA to the PHY register at MIIADDRESS.
- [1] RStat: start a single read of the PHY register at MIIADDRESS.
- [0] ScanStat: enable repeated read operations; a read will start on the next MDC enable when idle and continue to retrigger while ScanStat remains 1.
- Behavior
- Requests are detected on rising edges of the respective bits; only one operation runs at a time and new requests are inhibited while MIISTATUS.Busy is 1.
- Scan requests are aligned to MDC enable boundaries; MIIRX_DATA updates after each scan read completes.
- Software should poll MIISTATUS.Busy to determine completion; reads update MIIRX_DATA automatically.
- Readback: bits [2:0] reflect the latched command bits; bits [31:3] read as 0.

MIIADDRESS (target PHY/register)
- Fields
- [12:8] RGAD: PHY register address (5 bits).
- [4:0] FIAD: PHY address (5 bits).
- Readback: other bits read as 0.

MIITX_DATA (write payload)
- Fields
- [15:0] CtrlData: 16-bit data presented to the PHY for write operations.
- Readback: bits [31:16] read as 0.

MIIRX_DATA (read result; CPU read-only)
- Fields
- [15:0] Prsd: last 16-bit data received from the PHY upon completion of a read or scan.
- Behavior
- Updated by hardware at the end of a read or scan transaction (on EndBusy) via an UpdateMIIRX_DATAReg pulse.
- Update is suppressed if a write operation begins in the same cycle the previous operation ends, ensuring MIIRX_DATA reflects only read or scan results.
- CPU must not write this register.
- Readback: bits [31:16] read as 0.

MIISTATUS (controller status)
- Fields
- [2] NValid: indicates a scan request is present and aligned in the MDC domain; clears when an operation ends.
- [1] Busy: high while a request/start is present, an operation is in progress or ending, or a scan is pending/aligned; low only when fully idle.
- [0] LinkFail: link-fail status derived from PHY Basic Status Register (reg 1) reads; 1 indicates link

down per PHY interpretation.
- Readback: bits [31:3] read as 0.

Operational notes
- Single write: program MIIMODER (ClkDiv, NoPre), set MIIADDRESS (FIAD, RGAD), write MIITX_DATA, then write MIICOMMAND with WCtrlData=1. Poll MIISTATUS.Busy until 0.
- Single read: program MIIMODER and MIIADDRESS, then write MIICOMMAND with RStat=1. Poll MIISTATUS.Busy until 0; read MIIRX_DATA.
- Scanning: program MIIMODER and MIIADDRESS, then write MIICOMMAND with ScanStat=1. Controller starts reads aligned to MDC when idle and repeats while ScanStat stays 1; MIIRX_DATA updates after each completion; MIISTATUS.NValid reflects scan alignment; Busy reflects active or pending scans.

Implementation and CDC notes
- Requests are sampled and edge-detected in the MDC enable domain; overlapping commands are masked so only one operation runs at a time.
- Busy is conservative, asserting during start, in-progress, end, and while a scan is pending/aligned.
- LinkFail is latched when sampling PHY reg 1 during read; exact semantics depend on the PHY definition.
- Upper and unused register bits read as 0.

-- Address/Filter Registers (MAC_ADDR0/1, HASH0/1) --

Purpose and scope
- Provide software-programmable station MAC address and 64-bit multicast hash filter used by RX address checking and control-frame handling.

Register map (Wishbone, 32-bit each)
- 0x10 MAC_ADDR0: station MAC lower 32 bits (bits [31:0]).
- 0x11 MAC_ADDR1: station MAC upper 16 bits in [15:0]; [31:16] reserved, read as 0, writes ignored.
- 0x12 HASH0: multicast hash lower 32 bits.
- 0x13 HASH1: multicast hash upper 32 bits.

Write/read semantics
- Byte-lane writes supported; each enabled wb_sel_i byte updates only that lane.
- MAC_ADDR1[31:16] always read 0; writing these bits has no effect.
- Reads return the currently programmed values; all registers reset to 0.

Decoded datapath signals
- r_MAC[47:32] <= MAC_ADDR1[15:0]; r_MAC[31:0] <= MAC_ADDR0.
- r_HASH0 <= HASH0; r_HASH1 <= HASH1.

RX address filtering
- Unicast: accept if Destination Address (DA) equals r_MAC[47:0] (byte-for-byte match).
- Broadcast: BroadcastOK asserted only if broadcast frame and r_Bro=0; otherwise blocked.
- Promiscuous: r_Pro suppresses RxAbort on address miss (effectively accept all).
- Multicast: CRC-based 6-bit index CrcHash[5:0]=Crc[31:26] captured at DA byte 6 (qualified by CrcHashGood). Select HASH word by CrcHash[5] (0→HASH0, 1→HASH1), select byte by CrcHash[4:3], select bit by CrcHash[2:0]; accept multicast if the selected bit is 1 and the DA indicates multicast (I/G=1).

Control frames (PAUSE)
- RX: accept PAUSE frames when DA is either 01-80-C2-00-00-01 (reserved multicast) or r_MAC.

- TX: generated PAUSE frames use r_MAC as the Source Address.

Clock-domain and integration
- Software programs registers in wb_clk_i domain; RX filtering and hash checks occur in MRxClk domain using r_MAC/r_HASH as static configuration.

Reset and programming
- All registers reset to 0; software must program MAC and hash before enabling RX filtering.

Endianness
- Byte ordering follows Wishbone byte lanes; software should write each MAC/hash octet into the intended lane consistent with system endianness.

-- Flow Control Registers (TXCTRL) --

Flow Control Register: TXCTRL (offset 0x14)
Purpose: Commands transmit-side IEEE 802.3x MAC Control PAUSE frames and holds the pause-time value.
Fields:
- [15:0] TxPauseTV (RW): Pause time/quanta (0..65535); one quanta = 512 bit-times. On the wire, encoded big-endian as [15:8] then [7:0]. Reset = 0x0000.
- [16] TxPauseRq (RW; edge-triggered; auto-clear on start): Writing 1 while CTRLMODER.TxFlow=1 issues a one-shot PAUSE transmit request. Hardware clears this bit when the control frame start is accepted; software may clear it by writing 0. If TxFlow=0, no request is generated and the bit remains set until cleared. Reset = 0.
- [31:17] Reserved (RO=0): Reads return 0; writes ignored.
Write strobes/byte lanes: Bits 0..16 are writable via the lower three byte lanes; upper byte lane (bits 31..24) is ignored.
Operational behavior:
- A rising edge of TxPauseRq with TxFlow=1 arms the control-frame generator; the PAUSE frame is injected at the next eligible TX idle opportunity. Normal TX data is temporarily blocked during control-frame transmission.
- The Pause Time field used in the frame is the current TxPauseTV latched at control-frame start (TxCtrlStartFrm). Software may update TxPauseTV any time before start to change the pending value.
- Retriggering requires a 1→0→1 sequence on TxPauseRq; holding it at 1 does not retrigger.
- If TxFlow=0, setting TxPauseRq does not create a request pulse and no auto-clear occurs; software must clear it.
- Control frames force padding and CRC generation; standard preamble/SFD are emitted.
Status/interrupts:
- On control-frame completion, a TX-control interrupt is asserted (INT_SOURCE bit 5), subject to INT_MASK. Normal TX Done/Abort status is gated during control-frame transmission to avoid contention.
Readback: Returns last written values for TxPauseTV and TxPauseRq; reserved bits read as 0.
Notes:
- RX-driven pause enforcement (receive PAUSE quanta timer) is separate and unaffected by TXCTRL.
- PAUSE frame format: DA=01-80-C2-00-00-01, SA=MAC[47:0], Type=0x8808, OpCode=0x0001, Pause Time=TxPauseTV.

-- Interrupt Registers (INT_SOURCE, INT_MASK) --

Interrupt Registers implement the MAC’s interrupt controller in the Wishbone/CPU clock domain (wb_clk_i). They expose per-source pending status (write■1■to■clear) and per-source masks, and

drive int_o as the masked OR of all sources.

Addressing and width
- INT_SOURCE @ 0x01: 32-bit, RW1C (write ‘1’ clears the bit; ‘0’ no effect).
- INT_MASK @ 0x02: 32-bit, R/W.
- Only bits [6:0] are implemented; [31:7] read as 0 and ignore writes.

INT_SOURCE (pending status, RW1C)
- Read returns the latched pending status for each source.
- Write: write ‘1’ to clear selected bits; writing ‘0’ leaves bits unchanged. Multiple bits can be cleared in one write.
- Reset: all bits clear to 0.
- Bit mapping [6:0] (active-high when pending):
0: TXB — Transmit buffer interrupt (successful TX completion). Set by eth_wishbone on TxStatusWrite when the BD’s TxIRQEn is set and no TX error is flagged.
1: TXE — Transmit error interrupt. Set by eth_wishbone on TxStatusWrite when TxIRQEn is set and a TX error is present (e.g., underrun, retry limit, late collision, excessive defer).
2: RXB — Receive buffer interrupt (successful RX completion). Set by eth_wishbone on RxStatusWrite when RxIRQEn is set, subject to pause/control■frame gating.
3: RXE — Receive error interrupt. Set by eth_wishbone on RxStatusWrite when RxIRQEn is set and an RX error is present (e.g., CRC error, too big, short frame, symbol error, overrun, late collision), subject to pause/control■frame gating.
4: BUSY — Busy/overrun interrupt. Set by eth_wishbone if a frame starts while the RX engine is not ready (e.g., RX buffer not prepared).
5: TXC — Transmit control (flow-control) pulse latched as pending. Generated in MTxClk by MAC control on PAUSE frame TX completion when r_TxFlow=1; synchronized into wb_clk_i with edge detect.
6: RXC — Receive control (flow-control) pulse latched as pending. Generated in MRxClk by receive control when a valid PAUSE loads the timer and r_RxFlow=1; synchronized into wb_clk_i with edge detect.

INT_MASK (per-source enable, R/W)
- Read returns current mask bits.
- Write sets/clears mask bits; mask[i]=1 enables propagation of INT_SOURCE[i] to int_o, mask[i]=0 masks it.
- Reset: defaults to 0 (all sources masked) unless overridden by system integration.

Global interrupt output
- int_o = OR over i=0..6 of (INT_SOURCE[i] & INT_MASK[i]).
- Level-sensitive: int_o remains asserted while any enabled pending bit is set; software clears pending via INT_SOURCE W1C writes.

Clock-domain crossings
- TXC (bit 5) and RXC (bit 6) originate in MTxClk and MRxClk respectively. They are synchronized into wb_clk_i via multi-flop pipelines with rising-edge detection to create single-cycle pulses that set the INT_SOURCE latches. Handshake/ack paths to source domains ensure the set condition is reliably cleared in the source clock once observed in wb_clk_i.
- TXB/TXE/RXB/RXE/BUSY (bits 0–4) are generated in wb_clk_i by eth_wishbone on descriptor/status write events; TXB/TXE depend on BD TxIRQEn, RXB/RXE depend on BD RxIRQEn and pause/control gating.

Software guidance
- Clearing: write ‘1’ only to bits you intend to clear; use read■modify■write or a mask of ones for

cleared bits.
- Masking: enable desired sources in INT_MASK; masked sources can still be polled via INT_SOURCE without affecting int_o.
- Reading INT_SOURCE does not clear bits.
- Multiple pending events may coexist; clear order is software-defined.
- Descriptor-level enables must be set (TxIRQEn/RxIRQEn) for TXB/TXE and RXB/RXE to assert.

-- Debug/Status Registers (dbg_dat) --

dbg_dat is a 32-bit, read-only debug/status register exposed by eth_registers at address 0x16. It provides a snapshot of selected internal MAC and bus signals for bring-up and diagnostics. The register is driven and read in the wb_clk_i domain; reads have no side effects and writes to 0x16 are ignored. Bitfield composition is implementation-defined; typical coverage includes: Wishbone/top decode and handshake (byte select, RegCs/BDCs, CsMiss, pre-ack/err causes); DMA/descriptor engine progress and FIFO/overrun/underrun indicators; TX/RX MAC activity (frame start/end/valid presence, collision/reset/late/retry counters or hints, MTxErr/TxUsedData pulses, compressed state flags); MAC status latches (good/short/too-big frame, invalid symbol, dribble nibble, late/defer/carrier-sense lost, retry limit); flow control (pause active, control-frame send/receive, mux selection, block Tx done); MIIM activity indicators (in-progress/write op, bit counter range); RX abort handshakes; loopback and RX enable gating. All sources crossing into wb_clk_i must be synchronized before packing to ensure stable readback. Favor single-cycle pulse capture or sticky bits that clear on read or on frame boundaries, and avoid duplicating information already provided in formal status registers unless adding finer-grain debug. Asynchronous wb_rst_i clears underlying latches; dbg_dat typically reads as 0 after reset until activity occurs.

-- Write Enable and Byte-Lane Rules --

Global byte-lane model: 4-bit lane enables gate write activity everywhere—wb_sel_i for CPU/regs/BD space, we[3:0] inside BD SRAM, and m_wb_sel_o for DMA master. Unselected lanes retain previous contents.

CPU Wishbone writes (slave side):
- A legal write requires stb & cyc and ByteSelected = |wb_sel_i == 1 and a valid address decode; otherwise wb_err_o pulses and the write is rejected.
- Per-register lane masks apply in addition to wb_sel_i; only lanes enabled by both are writable. Bits outside defined field widths are ignored on write and read back as 0.
- Examples:
- PACKETLEN: lanes [3:0] writable.
- MODER: lanes [2:0] writable; upper bits read as 0.
- MIIMODER: lanes [1:0] writable; upper bits read as 0.
- COLLCONF: lane 1 is not writable (forced 0 in write path); other defined lanes honor field masks.
- INT_SOURCE: RW1C; writing 1 to any bit (in an active byte lane containing that bit) clears that interrupt, writing 0 leaves it unchanged.
- MIIRX_DATA: not CPU-writable; only updated by internal UpdateMIIRX_DATAReg.
- TX_BD_NUM: only low 8 bits meaningful; accepts writes only when DataIn <= 0x80; upper bytes read as 0.
- CPU writes with wb_sel_i == 4'b0000 are illegal and generate wb_err_o.

BD SRAM (eth_spram_256x32):
- CPU BD writes use we[3:0] derived from wb_sel_i; enabled bytes update, unenabled bytes preserve prior contents.
- Read-during-write is write-first per enabled byte: newly written lanes are visible immediately; others reflect previous data.

- MAC-side TX/RX status updates write full words (we = 4'b1111).
- ce gates both write acceptance and read address capture; oe controls output drive.

DMA Wishbone master:
- Master addresses are word-aligned; byte-lane selection is via m_wb_sel_o.
- TX reads use m_wb_sel_o = 4'hF (full-word fetches); byte steering to the MAC is internal.
- RX writes set m_wb_sel_o = RxByteSel from pointer alignment (patterns 1111/0111/0011/0001) so only lanes containing new bytes are written; unwritten lanes in memory are preserved.
- Final partial word uses RxValidBytes to assert only lanes carrying valid bytes; tail lanes are zero-padded internally but not written (masked by m_wb_sel_o).
- Within a burst beat, lane enables remain constant unless the tail flush requires a final-beat mask.

MIIM serialization (internal):
- eth_shiftreg uses one-hot ByteSelect[3:0] as internal byte enables to load specific frame bytes; only one ByteSelect is honored per MdcEn_n.
- Write data bytes use ByteSelect[2] and [3]; loads occur without shifting on those cycles.

Guard/consistency rules:
- Non-writable lanes or masked bits ignore writes; readback shows zeros or prior values per field definition.
- Internal logic prevents DMA RX writes from asserting lanes beyond the valid-byte count of the final word, avoiding unintended overwrites.

-- Reserved/Zeroed Bits Behavior --

General rule: Any undocumented/reserved bits in eth_top registers read back as 0 and ignore writes; fields are width-limited. Word-aligned Wishbone master accesses force m_wb_adr_o[1:0]=2’b00. Per register: MODER (0x00) [31:17]=0; INT_SOURCE (0x01) [31:7]=0, only [6:0] valid (RW1C); INT_MASK (0x02) [31:7]=0, only [6:0] valid; IPGT/IPGR1/IPGR2 (0x03/0x04/0x05) [31:7]=0, only [6:0] valid; PACKETLEN (0x06) no reserved bits (MinFL[31:16], MaxFL[15:0] implemented); COLLCONF (0x07) [31:20]=0 and byte lane [15:8]=0 (reads 0, writes ignored); TX_BD_NUM (0x08) [31:8]=0 (value constrained ≤0x80; upper writes ignored); CTRLMODER (0x09) [31:3]=0, only [2:0] valid; MIIMODER (0x0A) [31:9]=0, only [8:0] valid; MIICOMMAND (0x0B) [31:3]=0, only [2:0] valid; MIIADDRESS (0x0C) [31:13]=0 and [7:5]=0, only RGAD[12:8] and FIAD[4:0] valid; MIITX_DATA (0x0D) [31:16]=0, only [15:0] valid; MIIRX_DATA (0x0E) [31:16]=0 (read-only by CPU; writes ignored); MIISTATUS (0x0F) [31:3]=0, only [2:0] valid; MAC_ADDR0 (0x10) all 32 bits valid (no reserved bits); MAC_ADDR1 (0x11) [31:16]=0, only [15:0] valid; HASH0 (0x12) and HASH1 (0x13) all 32 bits valid; TXCTRL (0x14) [31:17]=0, only [16:0] valid; dbg_dat (0x16) implementation-dependent—no reserved-bit guarantees. Unless explicitly listed as valid above, upper/unused bits read 0 and have no side effects on write.