Generated Specification Document

INTRODUCTION

-- Overview --

dbg_top is the JTAG-facing router/controller for the SoC debug scan chain. With the Debug IR active, it interprets each Shift-DR as either a module-selection command or pass-through data to the currently selected endpoint, while centralizing CRC-32 generation/checking and TDI/TDO multiplexing.
- Function: Selects one of three endpoints (Wishbone, CPU0, CPU1) and routes JTAG traffic to it; manages a shared 1-bit-serial CRC-32 engine (poly 0x04C11DB7, MSB-first, seed=FFFF_FFFF) for selection validation and submodule transfers.
- Modes: The first Shift-DR bit chooses behavior: 1 = selection command (captures 4-bit module_id, then 32 CRC bits and streams a short status); 0 = pass-through (enables only the chosen endpoint, gates TDI to it, sources TDO from it unless a CRC serialization is requested).
- Endpoints: module_id 0 → dbg_wb (Wishbone bridge), 1 → dbg_cpu instance 0, 2 → dbg_cpu instance 1; any other ID flags a selection error and does not change the current selection.
- TDO priority: When any submodule requests shift_crc, TDO is crc_out; otherwise TDO comes from the selected endpoint; during selection, TDO streams the status after CRC.
- JTAG timing: TDI/TMS sampled on TCK rising edge; TDO updated on TCK falling edge.
- Reset/Update-DR: Clears selection, counters, and chip-enable signals; default module_id=0xF (no selection).
- Submodules: dbg_crc32_d1 (shared CRC engine), dbg_wb (JTAG↔Wishbone bridge with per-transfer CRC, CDC to wb_clk), dbg_cpu (two instances, JTAG↔CPU bus/control with per-transfer CRC, CDC to CPU clock, integrates dbg_cpu_registers for stall/reset).
- Host workflow: Load Debug IR; perform a selection scan (1 + module_id + 32 CRC) and read status; then issue pass-through scans (0 + payload) using the selected module’s protocol.

-- Key Features --

- Unified JTAG debug switch with secure module selection (1+4-bit header + 32-bit CRC-32, poly 0x04C11DB7, MSB-first, seed all 1s); selection commits only on CRC-OK; unknown IDs are rejected and reported.
- Multi-target routing to dbg_wb, dbg_cpu#0, dbg_cpu#1; isolates non-selected endpoints via per-endpoint chip-enable and TDI/TDO gating.
- Deterministic JTAG timing: sample TDI/TMS on TCK rising edge; update TDO on TCK falling edge for clean host capture.
- Immediate selection status: 5-bit stream on TDO after CRC {CRC error, bad ID, 0, 0, CRC MSB} for quick diagnostics.
- Shared 1-bit serial CRC engine across top and submodules; ORed crc_en; high-priority shift_crc forces TDO=crc_out; crc_match indicates zero remainder.
- Predictable TDO mux priority: CRC streaming > selected module TDO > selection-status stream.
- Clean scan lifecycle: counters and chip-enables reset on Update-DR; module_id latches only on CRC-OK; safe reset default of "no target" (0xF).
- Pass-through operation when header bit=0: DR payload and CRC control are routed to the selected module; top layer does not interpret submodule payload semantics.

- Submodule highlights:
- dbg_wb: JTAG-to-Wishbone single-beat 8/16/32-bit read/write, address auto-increment, per-transfer CRC, compact status, busy/overrun/underrun reporting, safe CDC to wb_clk.
- dbg_cpu: JTAG-to-CPU bridge with commands (GO, RD_COMM, WR_COMM, RD_CTRL, WR_CTRL), 32-bit streaming with per-transfer CRC, address auto-increment, read FIFO, busy/overrun/underrun reporting, CDC to CPU clock.
- dbg_cpu_registers: 2-bit control (stall, reset) with breakpoint-aware ownership transfer and dual-clock synchronization.
- Simple host protocol: select Debug IR, then use DR scans for module selection (1 + id + CRC + status) or pass-through operations (0 + module-defined payload including CRC/status).
- Lightweight implementation: compact counters/muxing, 1-bit-serial CRC, and reuse of submodule CRC/status logic to minimize area and complexity.

-- Supported Submodules --

- dbg_wb (Wishbone Debug Bridge)
- Module ID: 0x0
- Role: JTAG-to-Wishbone access endpoint for classic single-beat reads/writes.
- Features: byte/half/word transfers, address auto-increment, CRC-validated scans, compact status.
- Interfaces:
- JTAG/TCK: shift_dr_i, update_dr_i, pause_dr_i, tdi_i, tdo_o, wishbone_ce_i, crc_en_o, shift_crc_o, crc_match_i.
- Wishbone: wb_clk_i, wb_cyc_o, wb_stb_o, wb_we_o, wb_adr_o, wb_sel_o, wb_dat_o, wb_dat_i, wb_ack_i, wb_err_i.

- dbg_cpu (CPU Debug Bridge)
- Module IDs: 0x1 = CPU0, 0x2 = CPU1 (both reserved; design may instantiate one or two endpoints).
- Role: JTAG-to-CPU bus endpoint for 32-bit data reads/writes and control.
- Features: streaming data, CRC-validated scans, compact status, CPU stall/reset control via internal registers.
- Interfaces:
- JTAG/TCK: shift_dr_i, update_dr_i, pause_dr_i, tdi_i, tdo_o, cpu_ce_i, crc_en_o, shift_crc_o, crc_match_i.
- CPU domain: address/data/ack/stb/we and busy/status handshakes as implemented by the instance.
- Internal submodule: dbg_cpu_registers
- Role: CPU control register block (stall/halt and reset) with safe TCK↔CPU clock crossings and breakpoint-driven stall ownership.
- Interfaces: cpu_clk_i, tck_i, rst_i, bp_i, we_i, data_i[1:0]; outputs cpu_stall_o, cpu_rst_o, ctrl_reg_o[1:0].

- Shared CRC Engine: dbg_crc32_d1
- Role: Common 1-bit-serial CRC-32 generator/checker used by dbg_top and all selected submodules.
- Conventions: MSB-first, polynomial 0x04C11DB7, seed all 1s; zero-remainder indicates match (crc_match==1).
- Signals: crc_en (accumulate), shift_crc (serialize CRC), crc_out (MSB), crc_match.
- Integration: Submodules assert crc_en_o/shift_crc_o; when any shift_crc is asserted, dbg_top routes TDO from crc_out.

- Selection and muxing notes for supported submodules
- Valid IDs: 0x0 (dbg_wb), 0x1 (dbg_cpu CPU0), 0x2 (dbg_cpu CPU1). Others flag module_select_error.
- dbg_top gates TDI only to the currently selected submodule (via its *_ce_i) and multiplexes its TDO

unless a CRC shift is active.
- During module-select scans, dbg_top owns the chain and emits brief status (CRC error and unknown-ID conditions).

-- Protocol and Scan Format Summary --

Protocol and Scan Format Summary

Link and timing
- All DR scans are MSB-first. TDI and TMS are sampled on TCK rising edge; TDO updates on TCK falling edge.
- A shared CRC32 engine (poly 0x04C11DB7, MSB-first, seed=0xFFFF_FFFF, non-inverted) is used across top and submodules. A zero remainder indicates crc_match.

Top-level DR scan (dbg_top)
- First DR bit is the header selector:
1) Module-select frame:
- Layout: [1][module_id(4)][CRC32(32)][status(5)]. CRC covers the 5 selection bits [1 + id(4)].
- Module IDs: 0=Wishbone (dbg_wb), 1=CPU0, 2=CPU1; others invalid.
- Status (shifted after CRC): bit0=~crc_match (1=CRC error), bit1=module_select_error (invalid ID), bit2=0, bit3=0, bit4=crc_out (CRC MSB).
- On CRC OK, dbg_top latches module_id; subsequent pass-through scans target that module.
2) Pass-through frame:
- Layout begins [0] then the selected submodule's protocol takes over (only its CE sees TDI). CRC enable and serialization are driven by the active submodule.
- TDO priority during Shift-DR: if any submodule asserts shift_crc, TDO=crc_out; else TDO from active submodule in CE order (wishbone, cpu0, cpu1); else module-select status stream.
- Scan boundary: Update-DR terminates the current scan, clears CE and selection, and resets dbg_top/submodule counters.

Wishbone submodule (dbg_wb) pass-through scans
- Active when latched module_id==0 and wishbone_ce=1.
- Command nibble (4 bits) follows the leading [0]:
- 0x2 WR_COMM: shift header {AccType[3:0], Addr[31:0], Len[15:0]} (52 bits), then CRC32; accept on CRC OK.
- 0x1 RD_COMM: CRC phase first, then module preloads and shifts out the 52-bit header, followed by status nibble (4) and CRC out.
- 0x0 GO: CRC phase first, then data phase per AccType/Len, followed by status nibble and CRC out.
- AccType: writes 0x0/0x1/0x2 => 8/16/32-bit; reads 0x4/0x5/0x6 => 8/16/32-bit.
- Data phase: host streams write data (byte/half/word aligned) or reads are streamed out from an internal 4-byte buffer; each transfer is covered by CRC and reports a 4-bit status.
- CRC control: dbg_wb asserts crc_en during accumulation and shift_crc to serialize CRC via dbg_top.

CPU submodule (dbg_cpu) pass-through scans
- Active when latched module_id==1 (cpu0) or 2 (cpu1) and the corresponding CE=1.
- Command nibble (4 bits) follows the leading [0]: 0x0 GO, 0x1 RD_COMM, 0x2 WR_COMM, 0x3 RD_CTRL, 0x4 WR_CTRL.
- WR_COMM: shift 52-bit header {acc_type[3:0], adr[31:0], len[15:0]} then CRC32; accept on CRC OK.
- RD_COMM/RD_CTRL: CRC phase first, then preload and shift out requested fields; status nibble and CRC out follow.
- WR_CTRL: uses the top 2 bits of the 52-bit window for control data; applied on CRC OK.
- GO data phase: write (acc_type==0x2) streams 32-bit words from host to CPU writes; read (acc_type==0x6) streams buffered 32-bit words out with underrun protection; each ends with status

nibble and CRC out.
- CRC control mirrors dbg_wb (crc_en, shift_crc).

Operational notes
- Only the selected CE receives TDI during pass-through; others are gated off.
- Pause-DR: selected submodules may drive busy indications on TDO during pauses.
- Each Update-DR cleanly ends the in-flight sequence and resets counters/flags.

-- JTAG Timing Conventions --

Clocking and sampling
- TCK rising edge: dbg_top and all children sample TDI and TMS during Shift-DR.
- TCK falling edge: dbg_top registers TDO; hosts should sample TDO on the next rising edge per IEEE 1149.1.

Bit ordering and shift direction
- MSB-first throughout: DRs shift left, TDI enters LSB, TDO outputs MSB.
- CRC-32 uses non-reflected 0x04C11DB7, seed all 1s; computed and serialized MSB-first, non-inverted.

Debug IR Shift-DR framing
- First bit is a header: 1 = module-select handled by dbg_top; 0 = pass-through to the currently selected child.
- Module-select timing: bits 0..4 (header+4-bit id) captured on rising edges; bits 5..36 (32-bit CRC) captured on rising edges; immediately after CRC, 5 status bits appear on TDO on falling edges in order: ~crc_match, module_select_error, 0, 0, crc_out.
- Pass-through timing: dbg_top asserts CE for the latched child on Shift-DR entry; TDI is gated only to that child while CE is active; CE and counters clear on Update-DR.

TDO multiplexing priority
- Highest: when any child asserts shift_crc, TDO = CRC remainder (MSB-first).
- Else: TDO = selected child's TDO while its CE is active.
- Else: TDO = dbg_top's selection/status stream.

Pause-DR behavior
- Children (dbg_wb/dbg_cpu) may drive busy on local TDO; dbg_top forwards it per normal mux priority, so busy can be observed externally when CE is active.

Update-DR boundary
- Defines transaction boundary: dbg_top and children reset counters, CE, selection flags, and status. Hosts should not rely on residual state across Update-DR.

CRC timing and control
- Shared CRC engine runs in TCK domain; crc_en asserts during payload/CRC accumulation windows.
- After payload+CRC, children may assert shift_crc to stream the 32-bit remainder MSB-first over 32 TCKs; crc_match=1 indicates zero remainder under the stated convention.

Host guidance
- Keep TDI/TMS stable around TCK rising edge; sample TDO on rising edge.
- Respect the Debug IR header: 1 for module-select (header+CRC+status follows), 0 for pass-through.
- When shift_crc is observed, clock 32 TCKs to read the CRC remainder.
- Use Update-DR to delineate scans; expect clean state at each Update-DR.

-- Clocking and Reset Overview --

- Clocks and clock domains
- TCK (JTAG scan clock): Primary clock for all JTAG-side logic. dbg_top, the scan sides of dbg_wb and dbg_cpu, and dbg_crc32_d1 run entirely in the TCK domain. TDI/TMS are sampled on TCK rising edge; TDO is selected in dbg_top and registered on TCK falling edge for standard JTAG timing.
- wb_clk_i (Wishbone): Drives the Wishbone-side state machines and staging buffer inside dbg_wb. Treated asynchronous to TCK; all crossings use explicit synchronizers/handshakes.
- cpu_clk_i (CPU): Drives the CPU-side state machines in dbg_cpu and dbg_cpu_registers. Treated asynchronous to TCK; crossings use explicit synchronizers/handshakes.

- JTAG/TAP phase gating
- shift_dr_i enables scan shift in the TCK domain.
- update_dr_i is a one-shot end-of-scan strobe in TCK and acts as a synchronous transaction boundary clear; it is also synchronized into WB/CPU domains for cleanup.
- pause_dr_i may gate out busy/status from submodules when TAP is paused.
- debug_select_i must be asserted to interpret the first Shift-DR bit (module-select vs pass-through).

- Reset sources and behavior
- rst_i: Asynchronous, active-high global reset across dbg_top, dbg_wb, dbg_cpu, dbg_cpu_registers, and dbg_crc32_d1. Clears control/state, counters, and seeds CRC engines to 32'hFFFF_FFFF.
- update_dr_i: Synchronous per-scan clear in TCK. In dbg_top it clears module selection, CE enables, data/crc/status counters, crc_started, and forces module_id to 4'hF until a valid selection completes. In dbg_wb and dbg_cpu it clears most TCK-side counters/flags and error latches; wb_error is cleared and address/control are (re)loaded on a synchronized set_addr. update_dr_i is synchronized into wb_clk_i and cpu_clk_i domains (update_dr_cpu) to safely clear WB/CPU-side flags.

- Clock-domain crossings (CDC)
- TCK↔WB (dbg_wb): Two-flop synchronizers and handshakes for start_rd_tck, start_wr_tck, set_addr (TCK→WB) and wb_end/busy/update_dr (WB→TCK). Read data uses a WB-domain 4-byte staging buffer; TCK tracks availability with guarded counters.
- TCK↔CPU (dbg_cpu): Two-flop synchronizers for start pulses; cpu_end acknowledgment round-trip; update_dr_i synchronized as update_dr_cpu. A 1-word read staging/throttle prevents underrun.
- dbg_cpu_registers: Two-flop synchronizers for stall and reset controls and breakpoint ownership signals across TCK/CPU.

- CRC/scan interaction
- CRC enable (crc_en) is OR-combined from dbg_top selection and submodule crc_en_o and clocked by TCK into dbg_crc32_d1. When any shift_crc is active, CRC output overrides module TDO; result is still registered on TCK falling edge.

- Timing/constraints
- Treat TCK asynchronous to wb_clk_i and cpu_clk_i; constrain only intra-domain paths. Ensure TDO changes only on TCK falling edge so external tools can sample on the next rising edge.

-- Use Model --

Preconditions and timing:
- Select the Debug IR so debug_select_i=1; all operations occur during Shift-DR/Update-DR with this IR active.
- JTAG timing: TDI/TMS sampled on TCK rising edge; TDO updates on TCK falling edge. Host should sample TDO on the next rising edge.

- Bit order is MSB-first across all fields. CRC polynomial 0x04C11DB7, seed = 0xFFFF_FFFF, non-inverted.

Two-phase DR scans (first shifted bit decides mode):
1) Module-select scan (first TDI bit=1):
- Host shifts: 1 + 4-bit module_id (0=Wishbone, 1=CPU0, 2=CPU1) + 32-bit CRC over these 5 bits.
- Continue clocking in the same Shift-DR to observe status on TDO: bit0=~crc_match (1=CRC error), bit1=module_select_error (1=invalid id), bit2=0, bit3=0, bit4=CRC MSB (crc_out) if needed.
- End with Update-DR. On CRC OK and valid id, selection latches and persists across scans until changed or reset.
- On error (CRC fail or bad id), previous selection remains unchanged.

2) Pass-through scan (first TDI bit=0):
- dbg_top asserts CE for the currently selected submodule for the duration of Shift-DR.
- TDI is routed exclusively to the selected submodule; TDO is sourced from that submodule except when the shared CRC engine is streaming CRC (shift_crc asserted), in which case TDO=CRC MSB.
- Submodules define their own framing (command nibble, headers, payload, status) and CRC windows; host appends CRC for write-like phases, and captures CRC from TDO for read-like phases.
- Finish by exiting Shift-DR and issuing Update-DR to commit/terminate the operation, clear counters, and deassert CE.

Typical session:
- Load Debug IR (debug_select_i=1).
- Do one module-select scan (first bit=1) with CRC OK and valid module_id.
- Perform one or more pass-through scans (first bit=0) to execute the selected submodule’s protocol.
- Repeat pass-through scans as needed; re-select a different module with another module-select scan when desired.
- Always terminate each scan with Update-DR.

CRC responsibilities:
- Module-select: compute CRC over {1, module_id[3:0]} MSB-first; append 32 bits; expect status bit0=0 on success.
- Write-like phases (host→target): host appends 32 CRC bits; at CRC end the device drives immediate ~crc_match on TDO followed by a status nibble. Retry on CRC fail or error status.
- Read-like phases (target→host): after payload/status, the submodule asserts shift_crc to stream 32 CRC bits on TDO; host validates locally.

Pause-DR and pacing:
- Submodules may present a busy flag on TDO during Pause-DR to aid pacing. Resume with Shift-DR to continue or Update-DR to terminate.

Error handling and persistence:
- Treat any nonzero error bit or CRC fail as a failed transaction; host should retry or recover as appropriate.
- Selection persists across DR scans and Pause-DR; it changes only on a new CRC-valid module-select scan or on rst_i.
- On rst_i, module_id resets to 0xF (no target); host must re-select before pass-through.

Submodule summaries (for host framing expectations):
- Wishbone (module_id=0): commands 0x2 WR_COMM and 0x1 RD_COMM to set/read headers {AccType[3:0], Addr[31:0], Len[15:0]}, and 0x0 GO to execute. Write widths 8/16/32 (AccType 0x0/0x1/0x2); read widths 8/16/32 (0x4/0x5/0x6). GO+write: stream data then CRC; status nibble

follows. GO+read: payload then status then CRC stream.
- CPU (module_id=1/2): commands 0x2 WR_COMM, 0x1 RD_COMM, 0x0 GO for 32-bit word transfers; 0x4 WR_CTRL and 0x3 RD_CTRL for 2-bit control (stall/reset). GO+write: stream words then CRC; status nibble follows. GO+read: payload then status then CRC stream.

Throughput:
- All operations shift at 1 bit per TCK; long transfers require proportional TCK cycles. Use Pause-DR to accommodate system-side latencies.

IO PORTS

-- JTAG/TAP Interface Signals --

JTAG/TAP interface ports on dbg_top: tck_i (input) is the JTAG clock; TDI and TAP strobes are sampled on the TCK rising edge, and tdo_o is registered on the falling edge to meet IEEE 1149.1 TDO timing. rst_i (input, asynchronous active-high) resets the JTAG-side logic and shared CRC engine; there is no TRST port. debug_select_i (input) is high when the TAP IR selects the Debug instruction; only then dbg_top participates in DR scans and the TAP mux considers tdo_o. shift_dr_i, pause_dr_i, update_dr_i (inputs) are TAP-derived strobes: shift_dr_i enables shifting, counters, and CRC accumulation; pause_dr_i is forwarded to submodules for their Pause-DR behavior; update_dr_i is a pulse that ends the current transaction and clears dbg_top selection, counters, CE enables, and CRC-start flags (submodules also use it to finalize scans). tdi_i (input) is the serial data in; with debug_select_i and shift_dr_i high, the first DR bit selects the mode: 1 = module-select (1-bit header + 4-bit module_id followed by 32 CRC bits to the shared CRC), 0 = pass-through (TDI gated only to the currently selected submodule via its CE). tdo_o (output) is the serial data out; while debug_select_i is high, its source priority is: shared CRC when any submodule asserts shift_crc_o; else the selected submodule (Wishbone, CPU0, CPU1) by CE; else dbg_top's module-select status stream; dbg_top does not tri-state TDO (the TAP mux ignores it when debug_select_i is low). Not present: tms_i, capture_dr_i, and TRST; these are consumed inside the TAP to generate the used strobes.

-- Scan Data Signals --

- Scope and domain
- All scan data signals are in the JTAG TCK domain. TDI/TMS are sampled on TCK rising edges; TDO is updated on TCK falling edges and held stable for the next rising edge.

- Top-level scan ports
- Inputs: shift_dr_i (active during Shift-DR), update_dr_i (single-cycle at end of scan), pause_dr_i (Pause-DR), debug_select_i (IR selects debug DR chain), tdi_i (serial data in).
- Output: tdo_o (serial data out). Internally driven by a muxed bit (tdo_tmp) that is registered on the falling edge of TCK.

- Bit ordering and shift convention
- All shifters accept TDI into the LSB and shift left; the MSB is the next TDO bit.
- CRC and all scan payloads serialize MSB-first.

- Scan header and routing
- The first DR bit when debug_select_i=1 is the scan header.

- Header=1: selection phase. TDI stream is consumed by dbg_top (module_id + 32-bit CRC); submodules are not enabled.
- Header=0: pass-through. Only the currently selected submodule is chip-enabled (CE) and receives TDI.
- CE gating (examples): tdi_wb = wishbone_ce & tdi_i; cpu0_tdi = cpu0_ce & tdi_i; cpu1_tdi = cpu1_ce & tdi_i. Inactive modules see 0 on their TDI.
- CE is asserted only while shift_dr_i=1 and header=0; all CEs clear on update_dr_i.

- TDO selection and priority (during Shift-DR)
- Highest: if any submodule asserts shift_crc_o, TDO is forced to the shared CRC MSB (crc_out), regardless of submodule TDO.
- Else: if wishbone_ce, TDO = tdo_wb; else if cpu0_ce, TDO = cpu0_tdo; else if cpu1_ce, TDO = cpu1_tdo.
- Else (selection phase or no active CE): TDO = selection/status stream generated by dbg_top.
- tdo_o is driven from tdo_tmp captured on the TCK falling edge.

- Shared CRC-32 interaction
- Single-bit-serial CRC engine (poly 0x04C11DB7, seed 0xFFFF_FFFF, MSB-first, non-inverted).
- crc_en is the OR of dbg_top's selection-phase enable and crc_en_o from the active submodule.
- shift_crc is the OR of submodules' shift_crc_o; when asserted, crc_out is driven on TDO (priority override).
- crc_out is the current MSB of the CRC register; crc_match=1 indicates a zero remainder when the received CRC completes.

- Selection-phase TDO content
- After receiving the 1+4 selection bits and their 32 CRC bits, dbg_top shifts out a 5-bit status stream on TDO: {~crc_match, module_select_error, 0, 0, crc_out} (MSB-first).

- Pause-DR and Update-DR effects
- While pause_dr_i=1, dbg_top maintains normal TDO muxing; submodules may present busy via their TDO paths.
- On update_dr_i: CEs, selection/CRC counters, and CRC-start flags clear; any in-progress selection or pass-through transaction is terminated.

- Reset/initial state
- On reset, no module is selected (module_id=4'hF). Unknown IDs during selection cause module_select_error=1 in the selection status stream.

-- CRC Engine Interface --

Purpose: Single shared CRC-32 engine (dbg_crc32_d1) time-multiplexed between dbg_top selection phase and the currently selected submodule (Wishbone, CPU0, CPU1). Runs entirely in the JTAG TCK domain.

Clock/Reset:
- clk = tck_i (JTAG TCK).
- rst = rst_i; seeds CRC register to 0xFFFF_FFFF on reset only (not on Update-DR).

Polynomial/Convention:
- CRC-32 polynomial 0x04C11DB7, MSB-first, non-inverted.
- Zero remainder indicates acceptance (crc_match=1) after payload + transmitted CRC.

Inputs:
- data_in = tdi_i; one bit consumed per TCK when crc_en=1.
- crc_en = OR of dbg_top's selection-phase CRC enable and the selected endpoint's crc_en_o; only one producer active per DR scan.
- shift_crc = OR of endpoints' shift_crc_o; not an engine input but controls TDO mux behavior.

Outputs:
- crc_out = MSB of the current CRC remainder; routed to dbg_top's TDO mux and used for optional observability.
- crc_match = 1 when the CRC remainder is zero; fanned out to dbg_top and all endpoints as crc_match_i.

Timing:
- TDI (data_in) and crc_en sampled on the rising edge of TCK.
- TDO updated on the falling edge of TCK.
- While shift_crc=1, dbg_top forces TDO=crc_out with highest priority.

Operation:
- Selection phase (header=1): dbg_top asserts its crc_en across the 5-bit header (1-bit header + 4-bit module_id) and the following 32 CRC bits; samples crc_match at the end of this CRC window to accept/reject module_id. dbg_top does not assert shift_crc; it may present crc_out for one status cycle.
- Pass-through (header=0): the selected endpoint drives crc_en_o to define its CRC window and may assert shift_crc_o to serialize its CRC remainder onto TDO. dbg_top's TDO mux gives shift_crc highest priority over normal TDO sources.

Constraints/Rules:
- Exclusivity: Only one CRC producer (dbg_top selection or selected endpoint) may be active per DR scan; dbg_top's OR gating enforces this.
- Update-DR behavior: dbg_top counters and chip-enables clear on Update-DR; the CRC engine is not automatically reseeded (seed occurs only on rst_i). Hosts/endpoints must respect start-of-scan seeding conventions.
- Sampling of crc_match: Endpoints sample crc_match_i at their CRC end; dbg_top samples it at the end of the 32-bit selection CRC window.
- JTAG alignment: Follow rising-edge sampling for inputs and falling-edge update for TDO; crc_out reflects the current MSB at all times.

-- Wishbone Bus Interface Signals --

32-bit Wishbone Classic (single transfer) interface driven by dbg_wb when dbg_top selects the Wishbone module (module_id==4'h0 with pass-through header==0); activity is gated by wishbone_ce_i. All signals are synchronous to wb_clk_i.
- wb_clk_i (input): Wishbone bus clock domain.
- wb_cyc_o (output): Cycle indicator; asserted at the start of each read/write and held until wb_ack_i or wb_err_i.
- wb_stb_o (output): Strobe; asserted with wb_cyc_o to launch a transfer; deasserts on acknowledge/error.
- wb_we_o (output): Write enable; 1 for writes, 0 for reads.
- wb_adr_o[31:0] (output): Address; loaded from the JTAG command on accept (CRC OK) and auto-increments after each acknowledge by access width (+1 for 8-bit, +2 for 16-bit, +4 for 32-bit).
- wb_sel_o[3:0] (output): Byte lane enables derived from wb_adr_o[1:0] and width:
• 8-bit: adr[1:0]=0/1/2/3 -> 1000/0100/0010/0001
• 16-bit: adr[1:0]=0 -> 1100; adr[1:0]=2 -> 0011

• 32-bit: 1111
- wb_dat_o[31:0] (output): Write data; formed from the TCK-side shifter with lane-filling by width (byte/half/word). 8-bit writes broadcast dr[7:0]; 16-bit writes broadcast dr[15:0] to upper/lower halves; 32-bit writes use dr[31:0]; actual lanes driven are controlled by wb_sel_o.
- wb_dat_i[31:0] (input): Read data; captured on wb_ack_i into a 4-byte staging buffer according to wb_sel_o and reassembled for JTAG serialization.
- wb_ack_i (input): Acknowledge; completes the current transfer, triggers read-data capture and address increment.
- wb_err_i (input): Error; terminates the transfer, latched for JTAG status and cleared on update_dr.
Protocol notes: Classic single transfers only; CTI/BTE/CAB are internally held at 0.

-- CPU Debug Interface Signals --

- Scope: CPU-facing debug interface between dbg_cpu (instances CPU0/CPU1) and the target CPUs.

- Clock and Reset
- cpu0_clk_i, cpu1_clk_i: CPU-domain clocks for dbg_cpu operation and synchronization.
- cpu0_rst_o, cpu1_rst_o: Active-high, level-hold reset outputs to the CPU; asserted by host control (write 1) and released by host (write 0). Reset is synchronized to the respective cpuX_clk_i.

- CPU Bus (32-bit, word-granular streaming)
- cpu0_addr_o[31:0], cpu1_addr_o[31:0]: Current bus address. During GO streaming, auto-increments by 4 on each acknowledged transfer.
- cpu0_data_o[31:0], cpu1_data_o[31:0]: Write data presented to the CPU bus.
- cpu0_data_i[31:0], cpu1_data_i[31:0]: Read data captured on ack_i. Data are assembled big-endian into a 4-byte buffer; underrun detection is implemented on the read path.
- cpu0_we_o, cpu1_we_o: Write-enable (1=write, 0=read). Driven during GO operations to qualify transfer direction.
- cpu0_stb_o, cpu1_stb_o: Bus strobe/request. Asserts to start a read or write and deasserts on ack_i.
- cpu0_ack_i, cpu1_ack_i: Acknowledge inputs from the CPU bus. Each ack_i completes the current transfer and advances addr_o during streaming.

- CPU Control (halt/stall and breakpoint)
- cpu0_stall_o, cpu1_stall_o: Halt outputs to the CPU. Can be asserted by host control (WR_CTRL) and is immediately asserted on cpuX_bp_i. Once asserted, stall remains active under host ownership until explicitly cleared.
- cpu0_bp_i, cpu1_bp_i: Breakpoint inputs from the CPU. On assertion, the CPU is stalled immediately; subsequent release is controlled by the host via the stall register.

- Timing and Synchronization
- All control and status transfers crossing between the TCK/JTAG and CPU clock domains use two-flop synchronizers. Host-driven cleanup (e.g., via update phases) is synchronized into the CPU domain to clear per-operation flags.

- Operational Notes
- Streaming transfers operate on 32-bit words; address increments by 4 per ack.
- Read side uses a 4-byte buffer (big-endian assembly) with underrun detection; write side detects overrun conditions.
- Reset control is level-based (not pulsed); stall behavior is immediate on breakpoint and held until host clear.

-- Control and Status Outputs --

- TDO (tdo_o)
- Driven on TCK falling edge.
- Source priority during Shift-DR when debug_select_i=1:
1) If any submodule asserts shift_crc_o, TDO = crc_out (CRC serialization has absolute priority).
2) Else if wishbone_ce=1, TDO = dbg_wb.tdo_o.
3) Else if cpu0_ce=1, TDO = dbg_cpu(0).tdo_o.
4) Else if cpu1_ce=1, TDO = dbg_cpu(1).tdo_o.
5) Else, TDO = module-selection/status stream (selection command path).
- Pause-DR behavior: when a CE is active, dbg_top passes the selected submodule's tdo_o (e.g., busy flags) through on TDO.

- Module-selection/status stream on TDO (header=1 path)
- After the 32-bit CRC field is shifted, TDO emits 5 status bits:
- Bit0 = ~crc_match (1 = CRC error on selection command).
- Bit1 = module_select_error (1 = bad/unknown module_id).
- Bit2 = 0 (reserved).
- Bit3 = 0 (reserved).
- Bit4 = crc_out (current CRC MSB to aid host finalization).

- Control outputs (chip-enable) to submodules (header=0 pass-through)
- wishbone_ce, cpu0_ce, cpu1_ce assert when the latched module_id selects the corresponding submodule and the scan is a pass-through (header=0) in Shift-DR.
- CE signals are level-true only during the active Shift-DR window; they gate TDI into the chosen submodule and select its TDO source (unless overridden by CRC serialization).
- CE signals deassert on Update-DR and on rst_i.

- Gated TDI into submodules (effective control)
- Only the selected submodule sees TDI: tdi_submodule = (corresponding CE) & tdi_i; all others receive 0.

- CRC control/status coupling affecting TDO
- crc_en to the shared CRC engine is the OR of dbg_top's selection-path CRC enable and all submodules' crc_en_o.
- shift_crc is the OR of submodules' shift_crc_o; when asserted, TDO is forced to crc_out regardless of CE.
- crc_out is used both during CRC serialization phases and as the final bit (Bit4) of the selection-status stream.
- crc_match is sampled at the end of the 32-bit CRC field during selection; its complement (~crc_match) is reported as Bit0 of the status stream.

- Reset and Update-DR behavior (control/status)
- On rst_i or Update-DR: all CE signals deassert; selection-phase counters and module_select clear; latched module_id resets to 4'hF (invalid).
- With no valid selection latched, CE remain low and TDO ownership defaults to the selection/status path (unless a submodule is actively shifting CRC).

- Default/idle observability
- Prior to a valid selection, only the selection/status stream is observable on TDO during Shift-DR; submodule traffic is suppressed until a valid module_id is accepted by CRC.

-- Clock and Reset Signals --

- Clocks
- JTAG/TCK (tck_i): Primary control and scan clock. All TCK-side logic in dbg_top and submodules advances on posedge tck_i. JTAG inputs (TDI/TMS) are sampled on posedge; TDO is registered on negedge to meet JTAG timing. JTAG state strobes shift_dr_i, update_dr_i, and pause_dr_i are generated and consumed in the TCK domain.
- Wishbone (wb_clk_i): Used only inside dbg_wb for Wishbone bus sequencing and WB-side state machines. Asynchronous to tck_i. TCK-originating control pulses (start_rd, start_wr, set_addr) are synchronized into wb_clk_i; WB-side status/handshake signals are synchronized back to tck_i.
- CPU clocks (cpu_clk_i): Each dbg_cpu instance has an independent cpu_clk_i for CPU bus transactions and control register synchronization. Asynchronous to tck_i. TCK-originating control pulses and update_dr_i are synchronized into cpu_clk_i; CPU-side status/handshake signals are synchronized back to tck_i.
- CRC engine clock: The shared CRC engine (dbg_crc32_d1) is clocked by tck_i. CRC enable/shift controls originate in the TCK domain; when any shift_crc is asserted, crc_out drives TDO.

- Resets
- Global reset (rst_i): Asynchronous, active-high, applied to dbg_top and all submodules. Effects include clearing TCK-side counters/flags and chip-enable selections, seeding the CRC to 32'hFFFF_FFFF, and resetting WB/CPU-side interfaces and control registers.
- JTAG Update-DR (update_dr_i): Acts as a synchronous clear in the TCK domain to terminate the current scan, reset per-scan counters/flags, and deassert chip-enable selections. In multi-clock modules (dbg_wb, dbg_cpu), update_dr_i is synchronized into wb_clk_i and cpu_clk_i to cleanly end in-flight operations and clear local flags.
- Optional synchronous reset (sync_rst) for dbg_crc32_d1 may be used by integration; if unused, tie inactive. CRC still seeds on rst_i.

- Timing and CDC requirements
- Treat tck_i, wb_clk_i, and each cpu_clk_i as fully asynchronous. Do not allow combinational paths between domains.
- Use two-flop synchronizers for all TCK→WB/CPU control pulses (e.g., start_rd, start_wr, set_addr) and for WB/CPU→TCK status/handshake indications (e.g., end/busy/data-available).
- Synchronize update_dr_i into wb_clk_i and cpu_clk_i domains for deterministic cleanup.
- Hold rst_i asserted long enough for all asynchronous domains to observe it; ensure glitch-free release relative to each domain.
- Maintain JTAG timing: sample inputs on posedge tck_i; TDO must be stable at negedge tck_i.

-- Signal Polarity and Active Levels --

- Global/JTAG strobes: debug_select_i, shift_dr_i, update_dr_i, pause_dr_i are active-high.
- Resets: rst_i is asynchronous, active-high across dbg_top and submodules; dbg_crc32_d1 rst and sync_rst are active-high (CRC seed = 32'hFFFF_FFFF).
- DR header bit: 1 = module-select command; 0 = pass-through to latched module.
- Chip-enables: wishbone_ce, cpu0_ce, cpu1_ce are active-high during pass-through; they gate TDI (tdi_sub = CE & tdi_i) and TDO selection.
- CRC control: crc_en and shift_crc are active-high; crc_match is active-high when CRC remainder is zero (OK). Error reporting on TDO commonly uses ~crc_match (error = 1).
- TDO selection: shift_crc = 1 forces TDO = crc_out; else CE-selected submodule TDO; else module-select TDO.
- dbg_top status: module_select_error is active-high.
- Wishbone bridge (dbg_wb): enable = shift_dr_i & wishbone_ce_i (active-high); rst_i active-high; crc_en_o, shift_crc_o active-high; crc_match_i active-high = OK, ~crc_match_i driven as error = 1. Bus

signals wb_cyc_o, wb_stb_o, wb_we_o, wb_sel_o[] active-high; wb_ack_i, wb_err_i active-high. Status flags busy_tck, overrun, underrun active-high; status nibbles use 1 for asserted/error.
- CPU bridge (dbg_cpu): enable = cpu_ce_i & shift_dr_i (active-high); rst_i active-high; crc_en_o, shift_crc_o active-high; crc_match_* active-high = OK, immediate/error bits use ~crc_match_* (error = 1). Bus signals cpu_stb_o, cpu_we_o active-high; cpu_ack_i active-high. pause_dr_i active-high causes TDO = busy_tck (busy = 1). Overrun/underrun flags active-high; status bits use 1 for asserted/error and, where specified, 1 for CRC OK.
- CPU control registers (dbg_cpu_registers): we_i active-high; bp_i active-high (asserts cpu_stall_o immediately). data_i[0] = 1 asserts stall (cpu_stall_o = 1); data_i[1] = 1 asserts reset (cpu_rst_o = 1). ctrl_reg_o reflects active-high semantics for stall and reset.

-- Interface Timing Assumptions --

- JTAG edge policy: TDI/TMS sampled on TCK rising while shift_dr_i=1; TDO updates on TCK falling and is valid for the next rising edge.
- Debug chain active only when debug_select_i=1 (Debug IR selected); otherwise dbg_top ignores Shift-DR header/flows.
- Half-cycle settling: All TDO mux inputs/selects (crc_out, shift_crc, CE, submodule TDO, module-select status) must settle within the rising→falling half-TCK.
- All contributors to TDO are generated in the TCK domain and/or registered on TCK rising to meet the negedge TDO register timing.
- Shifting cadence: Exactly one bit is consumed per TCK rising edge when shift_dr_i=1; hosts must provide continuous TCK during scan phases.
- Pause-DR: Halts shifting; counters hold; CE remains asserted; submodules may drive busy on TDO; resume continues seamlessly on the next rising edge.
- Update-DR: Marks end of a scan; clears dbg_top and submodule scan-phase counters/flags, CE, and module_select; new sequences start only after Update-DR.
- Module-select flow (header=1):
- Bit0=1 (header), next 4 bits are module_id (captured on first 5 rising edges).
- Followed by a 32-bit CRC, one bit per rising edge; crc_match is evaluated at CRC end to latch module_id.
- Status bits may stream immediately after CRC; TDO changes on falling edges.
- TDO priority: crc_out when shift_crc is asserted; else module-select status stream.
- Pass-through flow (header=0):
- dbg_top asserts CE for the latched module for the duration of Shift-DR; TDI is gated only to that submodule.
- CE deasserts at Update-DR; submodule handles its own command/data/CRC sequencing while enable = CE & shift_dr_i.
- TDO source: selected submodule unless any shift_crc is active, in which case crc_out overrides.
- CRC engine timing: TCK rising-edge domain; enable consumes one data bit per rising edge; shift serializes MSB-first; reset seeds 0xFFFF_FFFF; crc_match checked at CRC end.
- Cross-clock interactions (dbg_wb/dbg_cpu): TCK-domain scan logic; bus-side on wb_clk_i/cpu_clk_i with two-flop synchronizers for start/ack pulses; pulses >=1 TCK; Pause-DR allowed.
- Throughput/flow control: Host pacing is fixed at 1 bit per TCK rising; submodules may signal overrun/underrun via status without altering JTAG edge timing.
- Reset assumptions: Resets seed CRC, clear counters/CE; module_id defaults to 0xF (no valid target), requiring a successful module-select before pass-through.
- Timing budget: Choose TCK frequency/duty such that combinational TDO mux plus routing meets the half-cycle (rising→falling) requirement with margin.

ARCHITECTURE

-- Top-Level Block Overview --

dbg_top is the JTAG-facing debug switch/router that terminates the TAP Debug Data Register when the Debug IR is active (debug_select_i=1). It interprets the first bit of each Shift-DR to either run a CRC-verified module selection (header=1) or pass scan traffic through to the currently selected endpoint (header=0). Endpoints include a Wishbone bridge (module_id=0), CPU0 debug bridge (1), and CPU1 debug bridge (2); only one target is active at a time and selection persists until changed.

Interfaces and timing:
- JTAG: tck_i, tms_i, tdi_i sampled on TCK rising edge; tdo_o updates on TCK falling edge via a negedge register.
- Control: debug_select_i (Debug IR active), rst_i, Update-DR strobes from the TAP.
- Submodules: gated TDI, returned TDO, chip-enable per endpoint, and shared CRC control signals.

Operating phases:
- Selection (header=1): shifts 5 bits (1 + 4-bit module_id) followed by 32 CRC-32 bits (MSB-first, poly 0x04C11DB7, seed all 1s). On CRC match at the CRC-end edge, module_id latches; invalid IDs flag module_select_error. A short status stream follows: {~crc_match, module_select_error, 0, 0, crc_out}.
- Pass-through (header=0): dbg_top asserts the CE for the latched module for the duration of the scan and gates TDI/TDO to/from that endpoint. The selected submodule owns command/data/CRC sequencing.

CRC and TDO routing:
- A single serial CRC core (dbg_crc32_d1) is shared across selection and endpoint operations: crc_en is the OR of dbg_top's selection enable and submodules' crc_en_o; shift_crc is the OR of submodules' shift_crc_o.
- TDO priority: if any shift_crc is active → crc_out; else selected module TDO; else dbg_top's selection/status stream.

Control and state:
- Counters: data_cnt (first 5 bits), crc_cnt (32 CRC bits), status_cnt (status phase); all clear on rst_i or Update-DR.
- module_id defaults to invalid (0xF) after reset; latches only on CRC-valid selection. CE signals and in-progress selection clear at Update-DR, which serves as the transaction boundary.

Summary:
- Provides a robust, CRC-checked routing layer between JTAG and three debug endpoints with strict JTAG timing, shared CRC resources, deterministic TDO muxing, and concise post-selection status reporting.

-- Scan Chain and Shift Register Structure --

The JTAG scan chain is a single-bit, MSB-first path with TDI/TMS sampled on TCK rising edge and TDO updated on TCK falling edge. The first DR bit gates the path: 1 enters dbg_top's selection micro-protocol; 0 passes through to the previously latched submodule. Selection micro-protocol uses a 5-bit shift register (module_dr) and data_cnt=0..4 to capture {1(header), module_id[3:0]}, followed by a 32-bit CRC window (crc_cnt=32) using a shared dbg_crc32_d1 (poly 0x04C11DB7, init all 1s, MSB-first; crc_match=zero remainder). After CRC, a 5-bit status sequence (status_cnt=0..4) is shifted: ~crc_match, module_select_error, 0, 0, crc_out. On valid crc_match and supported module_id

(0=dbg_wb, 1=dbg_cpu0, 2=dbg_cpu1), dbg_top latches module_id; otherwise pass-through stays disabled. In pass-through DR scans, dbg_top asserts exactly one CE (wishbone_ce/cpu0_ce/cpu1_ce) for the duration of Shift-DR and gates TDI to the selected submodule (tdi_sub=CE & tdi_i); all others are electrically quiescent. TDO priority is: if any source asserts shift_crc, TDO=crc_out (serializes the 32-bit CRC MSB-first); else TDO=selected submodule's TDO; else TDO=dbg_top's selection/status stream. A single shared CRC engine is OR-driven by crc_en from dbg_top (selection phase) or the active submodule; shift_crc from any source overrides the TDO path to serialize CRC. Submodules implement a common shift structure: a 4-bit command nibble, a 52-bit header image, a 32-bit CRC phase, a data stream during GO, and a 4-bit status nibble. Internal DRs shift left (TDI enters LSB; the MSB feeds TDO when selected). dbg_wb opcodes {0x2=WR_COMM, 0x1=RD_COMM, 0x0=GO}: WR_COMM shifts in 52-bit header then CRC; RD_COMM performs CRC first then shifts out the 52-bit header; data streaming after GO is 8/16/32-bit with counters aligning byte/half/word boundaries; shift_crc_o is asserted after status to serialize CRC via dbg_crc32_d1. dbg_cpu opcodes {GO, RD_COMM, WR_COMM, RD_CTRL, WR_CTRL}: WR_COMM shifts in header then CRC; RD_COMM shifts out header after CRC; RD_CTRL/WR_CTRL use the same 52-bit DR image for control paths; GO streams 32-bit words (dr[31] typically drives TDO during reads); pause_dr_i may force TDO=busy_tck per the module's TDO mux rules. Update-DR clears dbg_top counters (data_cnt, crc_cnt, status_cnt), CE enables, and module_select, and submodules similarly reset their phase counters/flags to reestablish boundaries. The selection micro-protocol spans a minimum of 42 clocks (5 header+id, 32 CRC, 5 status), and the default module_id reset value (4'hF) disables pass-through until a CRC-verified selection occurs.

-- Module Selection Datapath --

Activates only when Debug IR is selected (debug_select_i=1), Shift-DR is asserted, and the first DR bit is 1. Captures a 5-bit selection header+ID MSB-first into module_dr while data_cnt increments: module_dr[4] must be 1 (command header), module_dr[3:0] is the requested module_id. Immediately follows with a 32-bit CRC phase; crc_cnt counts the CRC bits. The shared dbg_crc32_d1 engine is enabled (crc_en) during the 5 header/ID bits and the 32 CRC bits, processes MSB-first, non-inverted CRC-32, and uses a zero-remainder check to produce crc_match at the end of crc_cnt. The selection path does not assert CRC shift; crc_out remains available for visibility. On the crc_cnt terminal edge (module_latch_en), and only if crc_match=1, module_id is updated to module_dr[3:0]; otherwise the previous selection is retained. Decode supports: 0 -> Wishbone (dbg_wb), 1 -> CPU0 (dbg_cpu[0]), 2 -> CPU1 (dbg_cpu[1]); any other requested ID sets module_select_error=1 and does not change module_id. After the 32 CRC bits, a 5-bit status stream is driven on TDO via tdo_module_select: bit0=~crc_match, bit1=module_select_error, bit2=0, bit3=0, bit4=crc_out; TDO updates on TCK falling edge and is selected when no submodule is forcing CRC serialization. Defaults/reset: module_id resets to 4'hF; data_cnt, crc_cnt, status_cnt, and all selection controls clear on Update-DR or rst_i.

-- CRC Engine Integration --

Shared engine: dbg_top instantiates a single dbg_crc32_d1 that is shared between the top-level module-select path and the currently selected submodule (e.g., dbg_wb, dbg_cpu). Clock/data domain: runs on TCK; data input is the live TDI stream; CRC advances only while shift_dr_i is asserted (Pause-DR does not advance). Polynomial and conventions: CRC-32 (0x04C11DB7), MSB-first, non-reflected, non-inverted; zero-remainder is the match convention; the transmitted CRC is the raw remainder MSB-first. Seeding: initialized to 32'hFFFF_FFFF on reset; each CRC-protected segment starts from the seeded state per protocol. Interface: inputs {clk=TCK, rst (async), optional sync_rst, enable=crc_en, shift=shift_crc, data=TDI}; outputs {crc_out=MSB of remainder, crc_match=(crc==0)}. Gating/wiring: crc_en = OR of top-level selection-phase enable and all selected submodules' crc_en_o; shift_crc = OR of selected submodules' shift_crc_o (no top-level shifting during selection); only one producer is active per DR path, making OR-gating contention-free. Distribution: crc_out and crc_match return to dbg_top; crc_match is fanned out to submodules as crc_match_i for acceptance decisions.

Top-level (module-select) usage: when the first DR bit is 1, dbg_top captures 5 header bits {1, module_id[3:0]} with crc_en asserted, then accumulates the next 32 bits (host-appended CRC); at the end, crc_match gates acceptance (1=accept, 0=reject). Selection/status TDO: after CRC, bit0=~crc_match (CRC error), bit1=module_select_error (unknown id), bit2=0, bit3=0; an extra cycle surfaces crc_out (MSB) to the host. Submodule usage: when the first DR bit is 0, dbg_top enables exactly one submodule via CE; that submodule asserts crc_en_o during its CRC-accumulated phases (payload plus received CRC for checking, or payload-only when generating a remainder), samples crc_match_i at its CRC end, and then raises shift_crc_o to serialize 32 bits of the remainder MSB-first; while shift_crc is high, dbg_top forces TDO=crc_out. TDO mux and timing: priority 1) shift_crc -> crc_out, 2) active submodule TDO, 3) selection/status TDO; dbg_top updates TDO on TCK falling edge to align crc_out timing. Lifecycle/reset: update_dr_i delineates scan segments and clears dbg_top counters/CEs/module_select/crc_started; dbg_crc32_d1 holds the seeded state at segment start; robustness is ensured by single-producer CRC controls per DR path and segment-level resets.

-- TDO/TDI Multiplexing and Priority --

- Timing: TDI and TMS are sampled on TCK rising edge; TDO is registered and updated on TCK falling edge.
- TDI routing modes:
- Pass-through scan (first Shift-DR bit = 0): Only the currently CE-selected submodule receives TDI. Per-endpoint gating: tdi_wb/cpu0_tdi/cpu1_tdi = CE & tdi_i; all non-selected endpoints see 0. The active CE is asserted for the duration of this scan and cleared at Update-DR.
- Module-select scan (first Shift-DR bit = 1): All CEs are deasserted (submodules isolated). dbg_top consumes TDI locally to capture {header=1, module_id[3:0]} and then a 32-bit CRC.
- Top-level TDO priority mux (highest to lowest):
1) If any submodule asserts shift_crc: TDO = crc_out (from dbg_crc32_d1), overriding all other sources.
2) Else if wishbone_ce: TDO = tdo_wb.
3) Else if cpu0_ce: TDO = cpu0_tdo.
4) Else if cpu1_ce: TDO = cpu1_tdo.
5) Else: TDO = tdo_module_select (dbg_top’s status stream during module-selection sequences).
- Module-selection TDO/status stream (when path #5 is active): After all 32 CRC bits of the selection are received, dbg_top shifts a short status: bit0 = ~crc_match (1 indicates CRC error); bit1 = module_select_error (1 indicates unknown/invalid module_id); bits2–3 = 0; bit4 = crc_out (MSB echo).
- CRC integration and override:
- crc_en into dbg_crc32_d1 = OR of dbg_top’s selection-phase CRC enable and all submodules’ crc_en.
- shift_crc = OR of all submodules’ shift_crc. When asserted, TDO is forced to crc_out irrespective of CE selection or status output.
- Pause-DR behavior:
- dbg_top does not impose a Pause-DR override. If a CE-selected submodule drives busy_tck on pause_dr_i, it passes through on TDO per the normal CE-based source selection (unless masked by the shift_crc override).
- Interaction summary:
- In pass-through scans, the active CE simultaneously gates TDI to one endpoint and selects its TDO for forwarding (subject to CRC override precedence).
- In module-select scans, TDI is handled internally by dbg_top and submodules are isolated; TDO comes from dbg_top status unless CRC override is active.
- Default/no-selection sets module_id to 4'hF; invalid selections are reported via module_select_error in the status stream.

OPERATION

REGISTERS