Generated Specification Document

INTRODUCTION

-- Overview --

spi_top is a Wishbone-slave SPI master optimized for software-friendly, deterministic transfers up to 128 bits per frame. It exposes a 32-bit bus with a simple address map (four RX readback words, CTRL, DIVIDER, and SS) and implements registered data-out, single-cycle acknowledge, and no bus error generation. Internally, it wraps two submodules: spi_clgen, a 16-bit programmable SCK generator that produces launch/capture edge strobes and supports a max-rate special case (divider=0), and spi_shift, a 128-bit TX/RX shifter supporting MSB- or LSB-first, selectable launch/sample edges, per-byte enables, and a length field where 0 encodes 128 bits. Safe programming is enforced by accepting control, divider, SS, and TX writes only while idle (not tip) to prevent mid-transfer glitches. Transfers are initiated by a start/go control bit, run with precise edge timing determined by tx_negedge/rx_negedge, and auto-clear the busy/start state at the final bit. Chip-selects are flexible: up to 8 active-low outputs are driven either continuously by a manual mask or automatically asserted only during an active transfer. An optional interrupt is raised at end-of-frame and is cleared by any acknowledged Wishbone access. Overall, spi_top provides a wide-frame SPI master with accurate clocking, configurable bit ordering and timing, simple software sequencing, and robust completion signaling.

-- Key Features --

- Wishbone slave with 32-bit, word-aligned register map (RX[0..3] @0x0–0xC, CTRL @0x10, DIVIDER @0x14, SS @0x18); byte enables; registered read data; single-cycle acknowledge; no bus-error generation.
- Programmable SPI clock via 16-bit divider with exact frequency control (SCK = clk_in / (2*(divider+1))); special divider==0 handling for maximum rate; one-cycle-early pos/neg edge strobes for precise timing.
- Flexible framing: 1–128-bit transfers (len==0 encodes 128); MSB-first or LSB-first; independent TX/RX edge selection for CPHA-like mode support.
- Robust transfer control: explicit start (go), transfer-in-progress (tip) tracking, deterministic first-bit preload, automatic end-of-frame busy/start-bit clear.
- End-of-transfer interrupt: asserted on final bit when IE=1; cleared by any acknowledged Wishbone access.
- Safe programming model: CTRL/DIVIDER/SS writable only when idle (!tip); TX buffer updates blocked during active transfers; sticky lower-byte behavior on CTRL to prevent inadvertent clearing.
- Chip-select management: up to 8 active-low CS outputs; manual mode mirrors SS; auto mode asserts CS only during transfers (ASS=1).
- Modular datapath: 128-bit shifter with 32-bit lane writes (per-byte enables) and concurrent RX capture; clean separation of clock generation and shift engine.
- Deterministic edge shaping: optional last_clk gating to control leading/trailing SCK edges and levels across transfers.
- Observability: full RX frame readable via four words; tip provides busy indication.

-- Design Goals --

Design an easy-to-integrate Wishbone SPI master (32-bit slave) with deterministic bus behavior (registered read data, single-cycle ack, no errors). Enforce safe, glitch-free operation by gating configuration and TX writes when idle and isolating TX/RX buffers during transfers. Offer protocol flexibility: 1–128-bit frames, MSB/LSB selectable, independent TX/RX edge selection with precise per-bit strobes. Support up to 8 active-low chip-selects with manual and auto-assert modes. Provide deterministic completion: busy/start auto-clear at the last bit and an interrupt-on-done cleared by any acknowledged access. Deliver broad timing control: 16-bit SCK divider (max rate at 0), last-edge/level gating, no spurious toggles at frame end. Streamline software use: word-aligned address map, byte enables, consistent 128-bit RX readback, visible busy status, and zero bus errors.

-- Block Diagram and Components --

Overview
- spi_top integrates five sub-blocks: Wishbone slave/register file, SPI clock generator (spi_clgen), 128-bit shift engine (spi_shift), chip-select driver, and interrupt logic. All logic is synchronous to wb_clk_i with synchronous reset wb_rst_i.

Sub-blocks
1) Wishbone slave and register file
- Role: Exposes configuration, TX/RX data, and status to the host via a Wishbone classic slave; provides control and data to internal blocks.
- Address decode (wb_adr_i[4:2]):
- 0x0, 0x4, 0x8, 0xC: RX readback lanes for the 128-bit shift buffer (32-bit each)
- 0x10: CTRL (control/status bits: go, char_len[6:0], lsb, tx_negedge, rx_negedge, ie, ass; zero-extended on read)
- 0x14: DIVIDER (SCK divider[15:0])
- 0x18: SS (chip-select mask[7:0], active-low)
- Write/Read behavior:
- CTRL/DIVIDER/SS and TX data writes are accepted only when !tip to avoid mid-transfer glitches
- wb_sel_i honored for lower bytes (e.g., DIVIDER); CTRL lower byte merge makes go sticky until HW clears at end-of-transfer
- Reads always allowed; wb_dat_o is registered (one-cycle latency)
- Wishbone handshake: wb_ack_o pulses per accepted access; wb_err_o is constant 0
- Outputs to internal logic: go, char_len[6:0], lsb, tx_negedge, rx_negedge, ie, ass; divider[15:0]; ss_reg[7:0]; TX write bus (latch[3:0], byte_sel[3:0], p_in[31:0])
- Inputs from internal logic: p_out[31:0] (RX read slice), tip (for write gating and CTRL auto-clear)

2) SPI clock generator (spi_clgen)
- Role: Generates SCK and single-cycle pos_edge/neg_edge strobes in the system clock domain for precise launch/capture timing
- Inputs: clk_in=wb_clk_i, rst=wb_rst_i, enable, divider[15:0], go, last_clk
- Outputs: clk_out (SCK), pos_edge, neg_edge
- Behavior:
- SCK half-period = divider + 1 system clocks; divider==0 yields maximum rate with synthesized strobes
- pos_edge/neg_edge are one-cycle strobes indicating intended SCK edges
- last_clk gating suppresses the first or shapes the final edge/idle level at transfer start/stop

3) Shift engine (spi_shift)
- Role: 128-bit parallel-to-serial/serial-to-parallel engine; manages bit counting, order, and edge phasing
- Inputs: clk=wb_clk_i, rst=wb_rst_i, go, len[6:0] (char_len; 0 encodes 128 bits), lsb, tx_negedge,

rx_negedge, pos_edge, neg_edge, s_in (MISO), s_clk (SCK), latch[3:0], byte_sel[3:0], p_in[31:0]
- Outputs: s_out (MOSI), p_out[31:0] (read slice), tip (transfer-in-progress), last_bit (asserts on final bit time)
- Behavior: Counts on pos_edge, supports MSB/LSB-first, launches/captures on selected edges, blocks host writes while tip=1, produces last_bit for clean stop

4) Chip-select driver
- Role: Drives active-low SS[7:0] based on SS register and auto-select control
- Inputs: ss_reg[7:0], ass, tip
- Output: ss_pad_o[7:0]
- Behavior: Manual mode (ass=0) reflects ss_reg continuously; Auto mode (ass=1) asserts SS only while tip=1

5) Interrupt logic
- Role: Signals transfer completion to the host
- Assertion: wb_int_o = 1 when (ie && tip && last_bit && pos_edge)
- Clear: wb_int_o cleared on any acknowledged Wishbone access (wb_ack_o)

Top-level interconnect and data/control flow
- CTRL.go/char_len/lsb/tx_negedge/rx_negedge drive spi_shift control inputs
- DIVIDER register drives spi_clgen.divider; top derives spi_clgen.enable and last_clk from transfer state (go/tip/last_bit)
- spi_clgen outputs: clk_out drives external SCK pad and spi_shift.s_clk; pos_edge/neg_edge feed spi_shift timing
- Serial data path: spi_shift.s_out -> MOSI pad; MISO pad -> spi_shift.s_in
- RX readback: spi_shift.p_out (32-bit slice) -> Wishbone read mux -> wb_dat_o (registered)
- TX write path: From Wishbone to spi_shift via latch/byte_sel/p_in, accepted only when !tip
- tip and last_bit feed: write gating, CTRL auto-clear on end-of-transfer (tip && last_bit && pos_edge), SS auto mode, and interrupt generation

External interfaces
- Wishbone: wb_clk_i, wb_rst_i, wb_cyc_i, wb_stb_i, wb_we_i, wb_adr_i, wb_sel_i, wb_dat_i, wb_dat_o, wb_ack_o, wb_err_o
- SPI pads: sck_pad_o (from spi_clgen.clk_out), mosi_pad_o (from spi_shift.s_out), miso_pad_i (to spi_shift.s_in), ss_pad_o[7:0] (from chip-select driver)
- Interrupt: wb_int_o

-- Role within System --

spi_top is the system’s memory■mapped SPI master peripheral on a 32■bit Wishbone bus, bridging CPU/software register accesses to the SPI core to control timing, chip■selects, and data movement. It exposes a compact register map for configuration (CTRL, DIVIDER, SS) and data (128■bit TX/RX via 32■bit slices), provides standard Wishbone slave behavior (registered read data, acknowledge handshaking, no bus error generation), and arbitrates safely between bus writes and on■going transfers. Operationally, it generates SCK and edge strobes, serializes MOSI/captures MISO for variable frame lengths and bit order, and manages up to 8 active■low chip■selects in manual or auto modes. For system integration, it raises a “transfer done” interrupt (wb_int_o) on completion when enabled and clears it on any acknowledged bus access, delivering a deterministic, software■programmable interface to external SPI devices.

-- Assumptions and Dependencies --

- Single synchronous domain: spi_top, spi_clgen, and spi_shift share one system clock; submodule

resets are asynchronous, active-high.
- Wishbone bus dependency: Requires a 32-bit Wishbone master with word-aligned accesses (wb_adr_i[4:2] used for decode). Byte enables must be driven; multi-byte registers honor wb_sel_i (e.g., DIVIDER uses wb_sel_i[1:0]). No error signaling (wb_err_o=0).
- Wishbone timing: Read data is registered and returned with a single-cycle wb_ack_o pulse; one ack per access. Any acknowledged Wishbone access clears the interrupt.
- Register write gating: CTRL, DIVIDER, and SS writes are accepted only when idle (tip=0). Hardware blocks or ignores mid-transfer writes; software must poll tip/busy before writing.
- TX data write dependency: Parallel TX buffer updates are permitted only when idle (tip=0); software must not modify TX lanes during an active transfer.
- Address map assumption: RX data occupies 0x00–0x0C (4 words), CTRL at 0x10, DIVIDER at 0x14, SS at 0x18. Reads return zero-extended values. Integration should confirm or override these addresses.
- RX read timing: Reading RX mid-transfer may yield partially updated data; software should read RX after tip=0.
- Interrupt behavior: wb_int_o asserts at end-of-transfer when IE=1 (ie && tip && last_bit && pos_edge). Cleared by any acknowledged Wishbone access; no separate clear register.
- CTRL start semantics: The start/go bit ORs with software writes and cannot be cleared while active; it auto-clears at end-of-frame (on last_bit at pos_edge) together with busy.
- SPI clock generation: SCK is derived from spi_clgen using divider[15:0]. For divider > 0, SCK half-period = divider + 1 system clocks (f_sck = f_clk / (2*(divider+1))). For divider == 0, SCK ≈ f_clk/2 via special strobe synthesis. System must choose divider values compatible with slave timing.
- Edge strobes: pos_edge and neg_edge are one-clock pulses that precede the actual SCK toggle by one system cycle when divider > 0. Shifter launch/capture and end-of-transfer detection depend on these strobes.
- Bit order/phase dependency: lsb, tx_negedge, and rx_negedge configure serialization order and sample/launch phase (CPHA-like). No explicit CPOL control; idle SCK level and first-edge behavior follow clgen last_clk gating. Connected slaves must be compatible with the selected phase and implicit idle polarity.
- Chip-select policy: ss_pad_o is active-low. Manual mode (ass=0) drives SS continuously from the SS register; auto mode (ass=1) asserts SS only while tip=1. Up to 8 CS outputs are provided; board-level mapping must align SS bits to devices.
- External I/O timing: MOSI updates on the configured launch edge; MISO must meet setup/hold at the capture edge. No extra timing margin is inserted; divider selection must ensure compliance with slave timing.
- Divider special case: For divider == 0, clgen emits strobes each cycle, including a go-assisted first pos_edge; systems must not rely solely on enable gating for the first maximum-rate edge.
- Limits: Max frame length is 128 bits (len==0 encodes 128). No DMA/burst; operation is programmed via register accesses. No bus error generation; masters must issue valid accesses.
- Reset assumptions: All logic resets synchronously to the system clock with asynchronous assertion; registers default to known values (assumed 0), with SS deasserted (all-high). Integrators should confirm reset defaults.
- Integration dependencies: All logic remains in a single clock domain; if used otherwise, CDC must be added externally. Divider range is 0..65535. Provide timing margins per target system clock and slave requirements. Unused chip-selects may be left unconnected or tied off.

IO PORTS

-- Clock and Reset --

- System clocking
- Single synchronous clock domain driven by wb_clk_i. All internal logic (spi_top, Wishbone slave interface, spi_clgen, spi_shift) operates in this domain.
- No internal clock-domain crossings; SPI serial timing uses one-cycle strobes (pos_edge/neg_edge) generated in the wb_clk_i domain.

- SPI SCK generation
- SCK is derived inside spi_clgen from wb_clk_i via a programmable 16-bit divider; SCK is not used as a clock for internal logic.
- For divider > 0, pos_edge/neg_edge assert one wb_clk_i cycle before the corresponding SCK transitions, giving launch/capture lead time.
- For divider == 0 (maximum rate), SCK toggles on every wb_clk_i cycle when enabled; a strobe is produced each cycle to maintain launch/capture timing.
- When enable == 0, SCK holds its current level, the counter reloads to the programmed divider for deterministic restart, and timing strobes are suppressed.
- last_clk gating suppresses the next low-to-high SCK transition to shape the final clock of a transfer and enforce a defined idle level between transactions.

- Reset
- Top-level reset wb_rst_i is active-high and distributed asynchronously to submodules.
- spi_clgen on reset: divider counter initializes to 16'hFFFF; SCK (clk_out) = 0; pos_edge = 0; neg_edge = 0.
- spi_shift on reset: tip (transfer-in-progress) = 0; bit counter cleared; s_out = 0; 128-bit data buffer cleared; no shifting while reset is asserted.
- Configuration registers (e.g., CTRL, DIVIDER, SS) reset to 0; wb_int_o deasserted.
- After reset release, SCK idles low until enable is asserted; the first enable reloads the divider and resumes normal cadence.

- Wishbone bus timing under clock/reset
- All Wishbone handshake signals are synchronous to wb_clk_i; wb_ack_o asserts for one cycle per valid transfer; wb_dat_o is registered.
- No bus error generation (wb_err_o = 0); outputs remain deasserted during reset.

- Design guidance
- Treat pos_edge/neg_edge as timing strobes in the wb_clk_i domain, not as separate clocks.
- Ensure system timing meets divider == 0 operation (SCK at f_sys/2) if used.

-- Wishbone Slave Interface --

Wishbone Classic 32-bit synchronous slave interface. Signals: inputs wb_clk_i, wb_rst_i, wb_cyc_i, wb_stb_i, wb_we_i, wb_sel_i[3:0], wb_adr_i[4:2] (word-aligned), wb_dat_i[31:0]; outputs wb_dat_o[31:0], wb_ack_o, wb_err_o (tied low), wb_int_o. Handshake: single-cycle acknowledgement with one-cycle latency; when wb_cyc_i & wb_stb_i are asserted, wb_ack_o pulses for exactly one clock in the next cycle (no back-pressure). Read timing: wb_dat_o is registered and valid in the cycle wb_ack_o is asserted. Write timing: data is sampled on the wb_ack_o cycle; byte enables wb_sel_i[3:0] are honored per-register rules. No stall or retry signals; all bus accesses receive an ACK; wb_err_o is never asserted. Addressing: word-aligned decode via wb_adr_i[4:2]; register map implements offsets 0x00..0x18. Undefined addresses' behavior is not specified, but the interface still acknowledges accesses per the unconditional ACK rule. Byte-enable usage: RX/TX data window uses all four byte lanes; DIVIDER uses only wb_sel_i[1:0] for the lower 16 bits; SS uses only the lower 8 bits; CTRL uses

defined bits within 14-bit width, with special sticky-OR behavior for the start/busy bit in the lower byte. Write acceptance gating: for CTRL, DIVIDER, SS, and TX window, writes are internally accepted only when the SPI transfer-in-progress flag (tip) is 0 (idle); while tip=1, the slave still ACKs the bus access but the internal register write is ignored. Read semantics: side-effect free for register contents; zero-extension on fields narrower than 32 bits; any acknowledged bus access clears the level interrupt wb_int_o if set. Interrupt: wb_int_o asserted at end of transfer when enabled; cleared by any acknowledged Wishbone access regardless of address or direction.

-- SPI Interface (SCK, MOSI, MISO) --

SPI Interface (SCK, MOSI, MISO)
- SCK generation: Produced by spi_clgen.clk_out from the system clock; rate set by 16-bit DIVIDER: f_SCK = f_sys / (2 × (DIVIDER + 1)). DIVIDER=0 yields ~f_sys/2.
- SCK activity: Toggles only while a transfer is in progress (spi_shift.tip=1); otherwise holds a steady idle level.
- SCK edge shaping: A last-edge gate ensures a controlled end-of-transfer (final edge forced high→low) and can suppress the initial low→high transition, maintaining a consistent low idle after reset.
- SCK timing strobes: For DIVIDER>0, pos_edge/neg_edge assert one system clock before the actual SCK transition to allow MOSI setup; for DIVIDER=0, strobes are synthesized each system clock and SCK toggles every cycle when enabled.
- MOSI drive: Preloaded with the first TX bit while idle so data is valid before the first active SCK edge; thereafter updates on the selected edge (tx_negedge=0 → rising, tx_negedge=1 → falling).
- MOSI bit order: lsb=0 sends MSB-first; lsb=1 sends LSB-first.
- Frame length: len[6:0] encodes 1–127 bits directly; len=0 encodes 128 bits. MOSI updates stop after the final bit to avoid spurious toggles.
- MISO sampling: Captured on the selected edge aligned to SCK (rx_negedge=0 → rising, rx_negedge=1 → falling). Sampling is gated off after the last bit to prevent oversampling.
- Signal mapping: SCK=spi_clgen.clk_out; MOSI=spi_shift.s_out; MISO=spi_shift.s_in. Internal pos_edge/neg_edge strobes drive shifter timing.
- Mode coverage: Independent TX/RX edge selection supports common CPHA-style timing; idle polarity defaults low (CPOL not exposed in this excerpt). In auto-SS operation, SCK activity follows transfer enable/SS assertion.

-- Chip-Select Outputs --

ss_pad_o[7:0] are eight independent, active-low chip-select outputs. Polarity: low = asserted, high = deasserted (idle high). Control is via the 8-bit SS register at address 0x18 (reads are zero-extended); each SS bit directly drives its corresponding output, so a 0 in SS asserts that CS line and a 1 deasserts it. SS writes are accepted only when the SPI engine is idle (!tip) to prevent mid-transfer changes. Behavior is selected by the ASS bit in CTRL: Manual mode (ASS=0) drives ss_pad_o continuously from SS, allowing CS to be held across back-to-back frames; Auto mode (ASS=1) deasserts all CS lines when idle (ss_pad_o=8'hFF) and asserts only during an active transfer (tip=1), with ss_pad_o=SS from transfer start (tip rising) to end (tip falling). CS assertion in Auto mode precedes the first SCK toggle by at least one system cycle when a clock divider is used, and CS deassertion coincides with transfer completion (tip falling), potentially aligning with the transfer-done interrupt. Any subset of the eight CS lines can be asserted simultaneously by programming SS accordingly; unused lines remain high. Writes to SS and CTRL are gated by !tip for safety to avoid CS glitches.

-- Interrupts --

wb_int_o is a level-latched, transfer-complete interrupt for the SPI master. Assertion occurs at the final bit of a frame when CTRL.ie=1, driven by (ie && tip && last_bit && pos_edge) from spi_shift/spi_clgen.

Scope is global across all chip-selects; one interrupt per completed transfer with no per-CS or per-word events. Clearing is solely by any acknowledged Wishbone access (wb_ack_o), including reads of RX data or writes to CTRL/DIVIDER/SS; starting a new transfer via a CTRL write also clears a prior interrupt through wb_ack_o. There is no event queue; wb_int_o remains asserted until serviced, and frequent polling that causes wb_ack_o will clear a pending interrupt. If CTRL.ie=0, no interrupt is generated and software must poll. Divider==0 does not alter behavior; end-of-transfer auto-clears the internal start/busy control independently of interrupt clearing.

-- Signal Polarity and Levels --

- Reset: rst is asynchronous, active-high; propagated to submodules (spi_clgen, spi_shift). After reset, SCK idles low.
- Wishbone bus: wb_clk_i is rising-edge clock. wb_cyc_i, wb_stb_i, wb_we_i, wb_sel_i[3:0] are active-high (only sel[1:0] affect the 16-bit DIVIDER). wb_ack_o is an active-high, single-cycle pulse on valid accesses (cyc & stb). wb_err_o is tied low. wb_dat_o is registered and valid the cycle after acknowledge. wb_int_o is an active-high level asserted at end-of-transfer when IE=1 and remains high until any Wishbone access is acknowledged.
- SPI clock (SCK from spi_clgen): Initializes low after reset and holds its level when disabled. While enabled, toggles with half-period = divider + 1 input clocks; divider==0 gives max rate (~clk/2). pos_edge/neg_edge are internal, active-high, single-cycle strobes (one clk early when divider>0; alternate each clk when divider==0). last_clk=1 suppresses low-to-high transitions (permits only high-to-low) to shape the final edge/idle level.
- MOSI (s_out): Actively driven; preloaded with the first TX bit while idle so the first selected edge drives valid data.
- MISO (s_in): Input-only; sampled on the selected capture edge; never driven by this core.
- Edge selection: tx_negedge=1 updates MOSI on SCK falling edge (else rising). rx_negedge=1 samples MISO on SCK falling edge (else rising). No explicit CPOL register; default idle SCK is low.
- Chip-selects ss_pad_o[7:0]: Active-low outputs (0=asserted, 1=deasserted). Manual (ASS=0): ss_pad_o continuously reflects SS register (active-low). Auto (ASS=1): ss_pad_o deasserted (all 1s) when idle; asserted per SS only while TIP=1.
- Control/status bits: GO=1 starts a transfer; auto-clears to 0 at end-of-transfer. IE=1 enables wb_int_o. ASS=1 enables auto slave select behavior. LSB=1 selects LSB-first (0=MSB-first). TIP is active-high during a transfer. LEN=0 encodes 128 bits; otherwise LEN[6:0] encodes 1..127 bits.
- I/O drive: SCK, MOSI, and CS outputs are actively driven; MISO is input-only.

-- Interface Timing and Handshake --

- Clock and reset: All interface timing is synchronous to the Wishbone clock. Reset is asynchronous, active high, and clears the SPI clock generator, shifter, and internal status.
- Wishbone request/ack: A request is present when wb_cyc_i and wb_stb_i are high. wb_ack_o is a single-cycle pulse per request when the core is not already acknowledging. wb_err_o is never asserted (constant 0).
- Read latency/data alignment: wb_dat_o is registered; read data is valid in the cycle wb_ack_o pulses (one-cycle latency from request).
- Byte enables: wb_sel_i are honored for multi-byte registers (e.g., DIVIDER[15:0]); partial writes update only the selected bytes.
- Safe-write gating: CTRL, DIVIDER, SS, and the 128-bit data buffer are writable only while idle (tip=0). Writes attempted during tip=1 are ignored, but the Wishbone handshake still completes (ack without error).
- Busy indication: tip is the transfer-in-progress flag used to gate writes and coordinate transactions. Software may poll tip via CTRL/status readback.
- Interrupt handshake: wb_int_o asserts synchronously when the final bit completes (tip and last_bit at the designated edge) and IE=1. wb_int_o clears on any acknowledged Wishbone access (wb_ack_o)

— reading or writing any register.
- SPI edge timing: SCK is generated by a divider; half-period equals divider+1 Wishbone clocks. For divider>0, pos_edge/neg_edge strobes assert one clock before the SCK toggle to lead launch/capture; for divider==0, strobes are synthesized each cycle to preserve per-edge timing.
- Edge selection: tx_negedge selects the MOSI launch edge; rx_negedge selects the MISO sample edge. On start when idle, the first TX bit is preloaded so the first SCK edge drives valid MOSI.
- Transfer termination: cnt decrements on pos_edge; last_bit ends the transfer (tip clears on last_bit at pos_edge). TX/RX clocks are gated by last to prevent extra toggles. last_clk shapes the final SCK level for deterministic idle.
- Chip-select timing: Manual mode (ass=0) drives ss_pad_o directly from SS[7:0] (active-low) and may be changed only while idle. Auto mode (ass=1) asserts ss_pad_o for the duration of tip=1 and deasserts when idle, providing per-frame SS timing.

ARCHITECTURE

-- Top-Level Block Overview --

spi_top is a 32-bit Wishbone-slave SPI master that integrates two submodules: spi_clgen (programmable SCK generator) and spi_shift (128-bit shift engine). It exposes a single synchronous clock/reset domain (Wishbone clock), drives SCK/MOSI and captures MISO, and provides up to 8 active-low chip-select outputs. The top maps control/status and data onto a word-aligned register set decoded by wb_adr_i[4:2]: RX readback at 0x0..0xC (four 32-bit slices of the last 128-bit frame, read-only), CTRL at 0x10 (frame length, bit order, TX/RX edge selects, start/busy, interrupt enable, auto-SS), DIVIDER at 0x14 (SCK divisor), and SS at 0x18 (manual chip-select mask). Wishbone behavior includes a registered wb_dat_o (one-cycle read latency), wb_ack_o pulsing on cyc&stb;, byte-lane writes honored, and wb_err_o tied low. To ensure safe operation, writes to CTRL/DIVIDER/SS are accepted only when idle (tip=0); CTRL lower-byte writes OR-merge the start/busy bit to prevent software from clearing it mid-transfer. spi_clgen creates SCK and early pos_edge/neg_edge strobes from DIVIDER; spi_shift uses those strobes and CTRL fields to serialize MOSI, capture MISO, track busy (tip) and signal last_bit. The top-level auto-clears the start/busy bit at the end of a frame and asserts wb_int_o on the final bit when IE is set; any acknowledged Wishbone access clears the interrupt. Chip-selects can be driven manually via SS or automatically (ASS) for the duration of tip. Overall, the block arbitrates host access against active transfers, provides deterministic SCK generation across divider settings (including divider==0), and offers readback of the most recent 128-bit RX frame.

-- Control Path and Register Decode --

- Wishbone register decode and timing
- 32-bit word-aligned Wishbone slave; address decode uses wb_adr_i[4:2] (8 words total).
- Data reads are registered: wb_dat_o is driven from an internal register, producing one-cycle read latency; wb_ack_o pulses for one cycle per qualified access (wb_cyc_i & wb_stb_i) and aligns with valid wb_dat_o. wb_err_o is tied low.

- Address map (read/write semantics)
- 0x00, 0x04, 0x08, 0x0C: 128-bit RX/TX data window
- Read: returns RX frame as four 32-bit words: [31:0], [63:32], [95:64], [127:96].
- Write: when idle (!tip), loads the corresponding 32-bit TX lane; per-byte enables honored via

wb_sel_i[3:0]. Writes while tip=1 are blocked.
- 0x10: CTRL (14-bit effective)
- Read: zero-extended to 32 bits.
- Write: accepted only when !tip; lower byte honors wb_sel_i[0]. On lower-byte writes, bit0 (go/start) is merged by OR with existing value (software cannot clear a live start/busy). Hardware auto-clears bit0 at end-of-frame (tip && last_bit && pos_edge).
- 0x14: DIVIDER (16-bit)
- Read: zero-extended to 32 bits.
- Write: accepted only when !tip; lower halfword honors wb_sel_i[1:0] (upper half ignored/reads as 0).
- 0x18: SS (8-bit active-low chip-select mask)
- Read: zero-extended to 32 bits.
- Write: accepted only when !tip; lower byte honors wb_sel_i[0].
- Unmapped addresses: reserved; behavior not specified (typical implementation reads 0 and ignores writes while acknowledging).

- Control path routing
- CTRL fields (14-bit): go/start, char_len, lsb, tx_negedge, rx_negedge, ie, ass; unused upper bits read as 0.
- go/start -> spi_shift.go (start transfer) and to spi_clgen to kick edges when divider==0.
- char_len -> spi_shift.len[6:0]; 0 encodes 128 bits, non-zero encodes 1..127 bits.
- lsb -> spi_shift.lsb (bit order select).
- tx_negedge, rx_negedge -> spi_shift TX/RX edge selection.
- ie -> interrupt enable mask.
- ass -> chip-select mode: 0=manual, 1=auto.
- DIVIDER[15:0] -> spi_clgen.divider (SCK half-period = divider + 1 input clocks); spi_clgen drives pos_edge/neg_edge strobes to spi_shift.
- SS[7:0] -> chip-select drive:
- Manual (ass=0): ss_pad_o mirrors SS continuously (active-low).
- Auto (ass=1): ss_pad_o asserts according to SS only while tip=1; otherwise deasserted.

- Status gating and side-effects
- tip (transfer-in-progress) from spi_shift gates all register writes (!tip required), controls auto-SS, and qualifies end-of-transfer actions (CTRL auto-clear and interrupt set).
- Interrupt: wb_int_o is asserted on (ie && tip && last_bit && pos_edge); it is cleared on any acknowledged Wishbone access (wb_ack_o).

- Readback and initialization
- RX window returns the most recent captured 128-bit frame; CTRL/DIVIDER/SS return their stored values zero-extended to 32 bits.
- Reset values are unspecified; software must initialize CTRL, DIVIDER, and SS before use.

-- Datapath: Shift Engine --

Datapath: Shift Engine (spi_shift) is the serial datapath of spi_top. It converts a 128-bit parallel buffer into MOSI and simultaneously captures MISO back into the same buffer, with programmable frame length, bit order, and launch/capture edge selection.
- Parallel buffer and host access: A single 128-bit data register serves as both TX source and RX destination. While idle (tip=0), software writes update selected 32-bit quarters via latch[3:0] with per-byte enables byte_sel[3:0] from p_in[31:0]. p_out[31:0] reads a 32-bit slice of the 128-bit buffer (slice mapping is internal to this block). All host writes are blocked while tip=1.
- Start/stop and length: A rising go when idle asserts tip to begin a frame. char_len (len[6:0]) encodes bit count: len=0 means 128 bits; len=1..127 selects that many bits. On start, cnt loads 8'h80 for 128-bit frames or {1'b0,len} otherwise. cnt is an 8-bit down-counter that decrements on each pos_edge while

tip=1; last is effectively (cnt==1). Transfer completes at (tip && last && pos_edge), deasserting tip; spi_top uses tip/last to auto-clear start and signal done.
- Bit timing and edges: The block consumes pos_edge and neg_edge strobes from spi_clgen. For divider>0, strobes are presented one input-clock early; for divider==0, strobes are synthesized per-cycle to maintain per-edge timing. tx_clk is a one-cycle strobe selecting rising or falling SCK via tx_negedge and is gated with last to prevent an extra end-of-frame toggle. rx_clk is a one-cycle strobe selecting sampling edge via rx_negedge (aligned to s_clk) and is gated with last to avoid oversampling the terminal bit.
- Serialization (MOSI): While idle, s_out is preloaded with the first TX bit so the first SCK edge launches valid data. On each tx_clk, s_out updates from data[tx_bit_pos]. tx_bit_pos is computed from len, cnt, and lsb to support MSB-first or LSB-first ordering across arbitrary frame lengths.
- Deserialization (MISO): On each rx_clk, data[rx_bit_pos] <= s_in, capturing MISO into the shared buffer. rx_bit_pos is computed from len, cnt, lsb, and rx_negedge so bits land in correct positions for the selected bit order and sampling phase.
- Bit order: lsb selects LSB-first vs MSB-first, applied consistently in both TX (tx_bit_pos) and RX (rx_bit_pos) index calculations for any frame length.
- Reset behavior: rst clears cnt, tip, and s_out, and zeroes the 128-bit data register.
- Integration: spi_top provides go, char_len (len), lsb, tx_negedge, rx_negedge; and receives tip and last for auto-clear/interrupt. After completion, software reads the received 128-bit frame via spi_top's RX address map. Chip-select timing and interrupt policy are managed by spi_top; spi_shift focuses on bit timing and data movement.

-- Datapath: Clock Generator --

Datapath: Clock Generator (spi_clgen)
- Purpose: Derive SPI serial clock (SCK) from clk_in and produce single-cycle edge-aligned strobes for TX/RX timing.
- Inputs: clk_in, rst, enable, divider[15:0], last_clk, go.
- Outputs: clk_out (SCK), pos_edge, neg_edge (strobes in clk_in domain; 1 clk_in cycle wide).
- Core operation:
- 16-bit down-counter cnt. While enable==1, cnt decrements each clk_in cycle; when cnt==0, clk_out toggles and cnt reloads to divider.
- When enable==0, cnt reloads to divider and clk_out holds its level; strobes suppressed (except divider==0 go-assisted start on re-enable).
- Half-period = divider + 1 clk_in cycles; SCK frequency = clk_in / (2*(divider + 1)).
- Edge strobes:
- divider > 0: strobes asserted one clk_in cycle before the actual SCK toggle (cnt==1).
- pos_edge: enable==1 && clk_out==0 && cnt==1 (precedes rising edge).
- neg_edge: enable==1 && clk_out==1 && cnt==1 (precedes falling edge).
- divider == 0 (fastest mode): clk_out toggles every clk_in when enable==1; generate strobes every cycle aligned to the next toggle.
- pos_edge aligns to the cycle preceding a low→high transition; a go term seeds an initial pos_edge when re-enabling to guarantee a starting cadence.
- neg_edge aligns to the cycle preceding a high→low transition.
- Last-edge gating:
- last_clk suppresses the next low→high transition near transfer end while still allowing the high→low transition, shaping the final observed edge/idle level.
- Reset:
- On rst: cnt=16'hFFFF; clk_out=0; pos_edge=0; neg_edge=0.
- Integration:
- divider sourced from DIVIDER[15:0] register; writes accepted only when the core is idle (!tip) to avoid mid-transfer changes.
- pos_edge/neg_edge feed the shift engine for TX/RX edge selection; pos_edge provides the cadence

for bit counting and end-of-frame detection.
- clk_out is driven to pads; toggling is gated by enable and last_clk per transfer activity.
- Timing notes:
- For divider>0, strobes are generated one clk_in cycle ahead of the SCK toggle to provide launch/capture lead time.
- divider==0 yields approximately clk_out = clk_in/2; ensure timing closure for this fastest configuration.
- Polarity:
- CPOL is not explicitly modeled; idle-level behavior across frames is governed by enable deassertion and last_clk gating.

-- Interrupt Generation and Clear Logic --

wb_int_o asserts exactly at the end-of-transfer timing strobe when IE=1 and tip && last_bit && pos_edge is true. Here, ie is the CTRL register's interrupt-enable bit (spi_top), tip and last_bit come from spi_shift, and pos_edge is the final-bit strobe from spi_clgen. The assertion is aligned to the final bit's pos_edge; tip deasserts immediately after, yielding at most one interrupt per frame. wb_int_o is sticky and remains high until cleared by any acknowledged Wishbone access: a wb_ack_o pulse clears the interrupt regardless of IE state or SPI busy/idle. Reads or writes to any decoded register will clear it; writes may be blocked while busy, but wb_ack_o still pulses and clears. wb_err_o is not used for clearing (always 0). Independently, the CTRL start/busy bit auto-clears at the same end-of-transfer event and does not depend on bus activity. If IE=0, no interrupt is generated; clearing via wb_ack_o does not modify IE. pos_edge is reliably produced for all DIVIDER values, including 0, ensuring deterministic interrupt timing. Auto-SS (ASS) and chip-select timing are independent of interrupt generation and clearing. Typical software clears by reading RX or any register during the ISR.

-- Chip-Select Control (Manual/Auto) --

Up to 8 active■low chip■selects are driven on ss_pad_o[7:0]. Each bit of the SS register (0x18) maps 1:1 to a CS line; writing a 0 asserts that CS (low), writing a 1 deasserts it (high). The ASS bit in the CTRL register (0x10) selects the CS control mode. Manual mode (ASS=0): ss_pad_o continuously mirrors SS (active■low) regardless of transfer state; software is responsible for asserting/deasserting CS and may hold CS across multiple frames. Writes to SS are accepted only when the SPI is idle (tip=0) to prevent mid■transfer glitches; changes take effect immediately after acceptance. Multiple CS lines can be asserted simultaneously by programming multiple zeros in SS (use only if attached devices support it). Auto mode (ASS=1): when idle (tip=0) ss_pad_o is forced deasserted (all 1s). On transfer start (tip 0→1 after go), ss_pad_o asserts per SS and remains asserted for the entire frame; on transfer completion (tip 1→0 after the final bit), ss_pad_o is immediately deasserted (all 1s), producing exactly one CS pulse per frame. SS must be preprogrammed while idle and is not changed during an active transfer. Interrupts do not affect chip■select timing; assertion/deassertion is driven solely by transfer progress (tip transitions).

-- Clocking and Timing Domains --

Single core clock domain: All internal logic (Wishbone register file, control, interrupt, spi_clgen, spi_shift) runs synchronously on a single system/bus clock (wb_clk_i/clk_in). wb_dat_o is registered (one-cycle read latency); wb_ack_o and wb_int_o are generated synchronously to this clock.

SPI clock and strobes: SCK is derived from the core clock via a reloadable divider. With last_clk gating inactive, SCK toggles every (divider + 1) core cycles (fSCK = fcore / (2 × (divider + 1))). pos_edge/neg_edge are one-core-cycle pulses generated in the core domain to mark SCK edges: for divider > 0 they lead the actual SCK toggle by one core cycle; for divider == 0 they occur every cycle without the one-cycle lead. last_clk can suppress the next rising transition to control the final edge/idle level; SCK initializes low after reset.

Shifter timing domain: The shifter builds tx_clk (launch) and rx_clk (capture) from pos_edge/neg_edge according to tx_negedge/rx_negedge and bit order. These strobes are core-synchronous and align with intended SCK edges. The internal bit counter decrements on pos_edge while tip=1; transfer completes on (pos_edge && last_bit). External MISO is captured in the core clock domain on rx_clk; board-level timing must ensure the SCK→MISO return path meets setup/hold relative to rx_clk.

Chip-select timing: In auto mode (ASS=1), ss_pad_o asserts with tip and deasserts synchronously at the final pos_edge. In manual mode (ASS=0), ss_pad_o follows the programmed SS register synchronously to the core clock and is independent of SCK.

Reset behavior: Asynchronous, active-high assertion; deassertion should be synchronized to the core clock. spi_clgen uses this async reset; spi_shift and top-level state share the same reset. SCK comes out of reset low.

Clock domain crossings: There are no internal asynchronous CDCs. SCK is an output derived from the core clock; datapath uses core-synchronous strobes rather than sampling on SCK. Special case at divider==0 (SCK ≈ fcore/2) reduces launch/capture margin to core-cycle granularity; ensure external device and PCB timing support this.

-- Data Widths and Buffers --

Wishbone data bus is 32 bits; all register reads are zero-extended to 32 bits. wb_sel_i is 4 bits; narrow registers ignore unused upper write bits, and DIVIDER supports per-byte writes via wb_sel_i[1:0]. wb_dat_o is a registered output with one-cycle latency after decode. The sole data buffer is a single 128-bit frame register in spi_shift shared by TX/RX; software accesses it as four 32-bit words at 0x00, 0x04, 0x08, 0x0C, with RX readback mapping 0x00→[31:0], 0x04→[63:32], 0x08→[95:64], 0x0C→[127:96]. Host writes to the 128-bit buffer are allowed only when idle (tip=0). There are no FIFOs; transfers are single-shot and software-managed. Parallel data lanes p_in/p_out are 32 bits; lane selection uses latch[3:0] and per-byte enables use byte_sel[3:0]. Serial I/O widths: MOSI (s_out) 1 bit, MISO (s_in) 1 bit, SCK (clk_out) 1 bit. Chip-select bus is 8 bits (active-low) driving ss_pad_o[7:0]. Interrupt output wb_int_o is 1 bit. Edge qualifier signals pos_edge and neg_edge are single-cycle 1-bit pulses in the clk domain.

OPERATION

-- Reset and Initialization --

Reset polarity and domains: A single active-high rst is distributed to all submodules. spi_clgen treats rst asynchronously; spi_shift clears internal state on rst.
Post-reset top-level state:
- Core idle: tip=0; SPI clock (SCK/clk_out)=0; pos_edge=0; neg_edge=0; MOSI (s_out)=0; RX/TX buffers cleared.
- Interrupt: wb_int_o deasserted; it asserts only at end of transfer when IE=1 and is cleared by any acknowledged Wishbone access.
- Wishbone: wb_dat_o is registered with unspecified reset value; wb_ack_o acknowledges per cyc/stb; no specific reset defaults are defined.

Submodule reset specifics:
- spi_clgen: cnt=16'hFFFF; clk_out=0; pos_edge=0; neg_edge=0. When not enabled, cnt reloads DIVIDER and SCK holds; no toggles occur until enable is asserted.
- spi_shift: counters clear; tip=0; s_out=0; 128-bit shift buffer zeroed. While idle, the first TX bit is preloaded on MOSI before the first SCK edge after start.
Initialization sequence (required before starting transfers):
1) Confirm tip=0 (idle).
2) Program DIVIDER[15:0] to the desired SCK rate.
3) Program CTRL fields (char_len/len, lsb, tx_negedge, rx_negedge, ie, ass) with go=0.
4) Program SS[7:0] (active-low chip-select mask) for the target device(s).
5) Load TX data lanes while !tip.
6) Assert go in CTRL to start; the shifter preloads MOSI and the clock generator begins strobes/SCK per DIVIDER and edge settings.
7) On completion, the busy/start bit auto-clears; wb_int_o asserts if IE=1 and is cleared by any acknowledged Wishbone access.
Special/unspecified behaviors and cautions:
- DIVIDER==0 generates maximum-rate strobes; fully program CTRL and SS before asserting go to avoid unintended edges.
- CTRL lower-byte write applies a sticky OR to bit0 (go); do not rely on clearing it during an active transfer.
- Auto SS mode (ass=1) deasserts CS when idle; manual SS mode (ass=0) drives ss_pad_o directly from SS—program SS before start to keep CS inactive during initialization.
- last_clk input reset/default behavior is unspecified; software should not assume a defined value at reset.

-- Bus Handshake and Access Timing --

- Interface: Wishbone Classic slave, zero-wait-state.
- Acknowledge: wb_ack_o pulses for exactly one cycle when wb_cyc_i & wb_stb_i are asserted; internal one-shot prevents multi-cycle acknowledges while a request is held.
- Errors: wb_err_o is permanently 0; no error/retry signaling.
- Read timing: Address decode occurs in the request cycle; wb_dat_o is registered and valid in the same cycle as wb_ack_o. Reads complete in one cycle.
- Write timing: Writes are acknowledged in one cycle with the same wb_ack_o behavior. Byte enables (wb_sel_i) are honored; DIVIDER[15:0] supports partial updates via wb_sel_i[1:0]. CTRL lower byte uses a sticky OR merge on bit0 (start/busy) to prevent software from clearing an active transfer.
- Commit gating: CTRL, DIVIDER, and SS writes are bus-acknowledged immediately but only take effect when !tip (idle). TX shifter lane writes are blocked while tip=1; the bus still returns ack without delay.
- Interrupt interaction: When enabled, wb_int_o asserts at end of transfer and is cleared by any acknowledged Wishbone access; it deasserts in the same cycle as wb_ack_o for that access.
- Addressing and visibility: Address decode uses wb_adr_i[4:2]; accesses are 32-bit word-aligned. RX reads (0x0/0x4/0x8/0xC) return the most recent completed 128-bit frame in 32-bit slices; CTRL, DIVIDER, and SS read back current values.
- Stall behavior: No stall/wait-state mechanism; every access (read or write) completes deterministically in one cycle.

-- Programming Sequence --

1) Reset and verify idle: After reset, ensure the core is idle (tip=0). Software may poll CTRL.go/busy; hardware auto-clears go/busy at the end of a transfer and software cannot clear go while busy (lower-byte OR-merge).

2) Configure clock and mode (idle-only writes): Write DIVIDER at 0x14 (wb_sel_i[1:0] honored; divider=0 gives maximum SCK). Write CTRL at 0x10 without asserting go: set char_len (1..127 bits; 0 encodes 128 bits), lsb (bit order), tx_negedge/rx_negedge (launch/capture edges), ie (interrupt enable), and ass (SS mode). Write SS at 0x18 to select active-low target device(s). Writes to DIVIDER/CTRL/SS are accepted only when tip=0.
3) Load transmit frame (idle-only writes): Write the 128-bit TX payload as four 32-bit words to 0x0, 0x4, 0x8, 0xC (byte enables honored). TX writes are blocked while tip=1. Reads at these addresses return the most recent RX data.
4) Start the transfer: While idle, assert CTRL.go=1 in a write to 0x10. In auto-SS mode (ass=1), ss_pad_o is driven only while tip=1; in manual mode (ass=0), ss_pad_o follows SS continuously.
5) Wait for completion: Either poll CTRL.go/busy until it clears (tip goes low) or enable IE and wait for wb_int_o to assert at end-of-frame.
6) Read results and clear interrupt: Read RX[127:0] from 0x0, 0x4, 0x8, 0xC. Any acknowledged Wishbone access (including these reads) clears wb_int_o.
7) Repeat transfers: While idle, update TX data and control fields as needed, then reassert go. Avoid writing DIVIDER/CTRL/SS while tip=1; such writes are gated/ignored. Address map (wb_adr_i[4:2]): 0x0/0x4/0x8/0xC = RX/TX words, 0x10 = CTRL, 0x14 = DIVIDER, 0x18 = SS.

-- Transfer Lifecycle (Start/Busy/Complete) --

Lifecycle phases and behavior:
- Idle/Program: While tip=0, software may preload TX data, program DIVIDER, CTRL (including len/order/edge/ass/IE), and SS. The start control bit (go) resides in CTRL.
- Start: Writing CTRL with go=1 when tip=0 begins a transfer. The shifter asserts tip=1 and preloads the first MOSI bit before the first SCK edge. The clock generator starts issuing pos_edge/neg_edge strobes; at divider==0 it synthesizes the initial strobe from go to start cleanly at maximum rate.
- Busy/Active: While tip=1, bit timing proceeds. The shifter decrements its bit counter on pos_edge and shifts/samples on the selected edges (tx_negedge/rx_negedge) in MSB- or LSB-first order. Host writes to CTRL, DIVIDER, SS, and TX lanes are blocked to preserve timing. The start/busy bit in CTRL is sticky on writes (OR semantics) and cannot be cleared mid-transfer; busy can be polled via CTRL read.
- Complete: On the final bit boundary (tip && last_bit && pos_edge), the shifter deasserts tip, hardware auto-clears CTRL start/busy, and the clock generator is gated (last_clk) so SCK stops cleanly. If IE=1, wb_int_o is asserted at this event and is cleared by any Wishbone access that yields wb_ack_o. In auto-SS mode (ass=1), ss_pad_o deasserts when tip goes low. RX registers then hold the completed frame.
- Determinism/Safety: Frame length is set by CTRL.len (0 encodes 128 bits). No mid-transfer reprogramming or software abort is supported. End-of-frame auto-clear provides a single-shot start per transaction and a consistent busy/done handshake for polling or interrupt-driven use.

-- Edge Selection and Bit Timing --

Edge timing is coordinated by spi_clgen (SCK generation) and spi_shift (TX/RX alignment) using programmable edge strobes and selectable launch/capture edges. For divider > 0, single■cycle pos_edge/neg_edge strobes are generated one clk_in cycle before the actual SCK rising/falling edges (pos_edge when !clk_out && cnt==1; neg_edge when clk_out && cnt==1), giving the datapath one■cycle setup margin. For divider == 0 (maximum rate), the one■cycle lead is removed and pos_edge/neg_edge are synthesized each clk_in cycle, alternating with clk_out to preserve per■bit timing at minimal margin. TX launch edge is selected by CTRL.tx_negedge: 0 = rising (use pos_edge), 1 = falling (use neg_edge). TX updates are driven by the early strobes so data is valid one clk_in before the physical SCK toggle when divider > 0; while idle, s_out is preloaded with the first bit so the first SCK edge sees valid MOSI. RX capture edge is selected by CTRL.rx_negedge: 0 = rising, 1 = falling. RX sampling is aligned to the actual chosen SCK edge by combining s_clk with the selected edge strobe;

sampling is inhibited on the final bit to avoid oversampling. The bit counter decrements on pos_edge while tip=1; last■bit detection, transfer completion (tip clear), and interrupt assertion occur on (last_bit && pos_edge), independent of launch/capture edge selections. TX/RX strobes are masked on the last bit to prevent extra updates/samples. SCK toggles only when enable=1 and cnt==0; last_clk gating can suppress the final low→high transition to shape the end■of■transfer edge and idle level (no explicit CPOL control; idle depends on last toggle and last_clk policy). With divider > 0, expect one clk_in cycle of setup before the SCK transition; with divider == 0, updates/samples occur every clk_in cycle with minimal external timing margin. Changes to CTRL.tx_negedge, CTRL.rx_negedge, and divider are accepted only while idle (tip=0) to ensure deterministic edge timing.

-- Readback Behavior --

Reads are supported at all addresses with a one-cycle registered latency: wb_dat_o is valid in the cycle wb_ack_o is asserted, and wb_err_o is never asserted for reads. Any acknowledged Wishbone access (including reads of any address) clears the interrupt (wb_int_o); reads have no other side effects and do not stall an ongoing SPI transfer. Address map readbacks: 0x00, 0x04, 0x08, 0x0C return the 128-bit RX buffer in little-to-big 32-bit word order (rx[31:0], rx[63:32], rx[95:64], rx[127:96]); no byte reordering is performed by the core. 0x10 returns the current 14-bit CTRL value zero-extended to 32 bits; the start/busy bit is auto-cleared by hardware at end of transfer. 0x14 returns the current 16-bit DIVIDER value zero-extended to 32 bits (divider==0 reads back as 0). 0x18 returns the current 8-bit SS mask zero-extended to 32 bits. RX readback coherency: the RX buffer is updated as bits arrive and is exposed combinationally to the bus, so multi-word reads during an active transfer (tip=1) can tear and individual words may change between accesses. For a coherent frame, software should read RX only after transfer complete (tip=0 or after servicing the transfer-done interrupt). After reset, the RX buffer reads as 0 until the first transfer completes. Reads are not gated by tip; they always reflect the live internal state.

-- Chip-Select Modes and Behavior --

Chip-select outputs ss_pad_o[7:0] are active-low (0 = asserted, 1 = deasserted), with up to eight CS lines allowed and multiple lines assertable simultaneously. The SS register (address 0x18) holds the programmed CS mask and reads return the register value; writes to SS and the CTRL.ass mode bit are accepted only when the SPI engine is idle (!tip) to prevent CS glitches. Modes: Manual (ass=0) — ss_pad_o continuously equals SS; software is responsible for asserting/deasserting CS and may hold CS across one or more transfers. Auto (ass=1) — when a transfer is in progress (tip=1), ss_pad_o reflects SS; when idle (tip=0), all CS outputs are forced deasserted (all 1s) regardless of SS contents. On transfer completion (tip clears), CS deasserts immediately in auto mode, while in manual mode it remains as programmed. CS is guaranteed stable for the entire frame length (1–128 bits) due to the !tip write gating and tip-driven auto behavior. If SS is all 1s, no device is selected even during auto mode transfers (SCK/MOSI may toggle without a target). Reading SS always returns the register value; in auto mode while idle the pad state may not match SS. SS reset value is not specified here.

-- Interrupt Handling --

- Interrupt line: Single level-type output wb_int_o.
- Source: “Transfer done” only; no underrun/overflow/error interrupts; wb_err_o=0.
- Assert condition: Set on the final bit boundary when IE=1: (ie && tip && last_bit && pos_edge). Assertion is synchronous with pos_edge; spi_shift deasserts tip at the same boundary.
- Enable/disable: Controlled by IE in CTRL (address 0x10). CTRL writes (including IE changes) are accepted only while idle (!tip). If IE=0, no interrupt is generated; software should poll busy/done via CTRL readback (or wrapper STATUS).
- Clear mechanism: Cleared by any acknowledged Wishbone access (wb_ack_o), regardless of address or read/write direction. Reading any register (e.g., RX/CTRL/DIVIDER/SS) or performing any

write clears a pending interrupt. Clearing does not require the SPI to be idle.
- Persistence: wb_int_o remains asserted after completion until a Wishbone transaction receives wb_ack_o. No throttling—each completed frame reasserts wb_int_o when IE=1.
- Timing details: pos_edge from spi_clgen is used for end-of-transfer detection and is valid even when divider==0. last_bit is derived in spi_shift from its down-counter; tip transitions low on (last && pos_edge), aligning with interrupt set timing.
- Interaction notes: Auto chip-select (ASS) affects ss_pad_o timing only and does not influence interrupt generation. Wishbone acknowledgments occur for valid cyc/stb, making clear behavior deterministic.

-- Error Handling --

The core does not report errors. Wishbone wb_err_o is tied low and wb_ack_o asserts for all qualifying accesses, even if a write has no internal effect. While a transfer is in progress (tip=1), writes to CTRL, DIVIDER, SS, and TX are accepted on the bus but ignored internally; no error or retry is signaled. CTRL.go is OR-merged on lower-byte writes and cannot be cleared by software mid-transfer; hardware auto-clears it at frame end. The RX register always exposes the last completed frame and may be overwritten by a new transfer without any overrun/overflow indication. The interrupt (wb_int_o) indicates completion only; it is cleared by any acknowledged Wishbone read or write, and unintended accesses that clear it are not treated as errors. Programming bounds are permissive: DIVIDER=0 is valid (fastest SCK) and SS is 8-bit; upper-bit writes are ignored without error. An asynchronous reset aborts any active transfer and returns the core to idle with no error reporting; auto-SS deasserts with tip=0. Accesses outside the documented address map are not flagged as errors; returned data is undefined/zero depending on implementation. The core provides no protocol fault detection (e.g., MISO inactivity, SCK violations), parity/CRC, timeouts, or error counters. There are no FIFOs, hence no underrun/overrun flags.

REGISTERS

-- Address Map --

32-bit Wishbone slave; word-aligned decode using wb_adr_i[4:2]. Offsets below are relative to the module base; byte enables are honored as noted. Writes to DATAx/CTRL/DIVIDER/SS take effect only when idle (tip=0); reads are always allowed.
- 0x00: DATA0 — RX[31:0] read; TX[31:0] write when idle. Per-byte write enables; writes ignored during tip=1.
- 0x04: DATA1 — RX[63:32] read; TX[63:32] write when idle. Per-byte write enables; writes ignored during tip=1.
- 0x08: DATA2 — RX[95:64] read; TX[95:64] write when idle. Per-byte write enables; writes ignored during tip=1.
- 0x0C: DATA3 — RX[127:96] read; TX[127:96] write when idle. Per-byte write enables; writes ignored during tip=1.
- 0x10: CTRL — 14-bit control, R/W when idle; reads are zero-extended to 32 bits. Lower byte writes are merged (start/go bit is sticky/ORed); hardware auto-clears start/busy at end of transfer.
- 0x14: DIVIDER — 16-bit SCK divider, R/W when idle. Writes use only lower 16 bits (honor wb_sel_i[1:0]); upper 16 bits ignored on write and read as zero.
- 0x18: SS — 8-bit chip-select mask, R/W when idle. Only lower byte writable; upper 24 bits read as zero.

- 0x1C: Reserved/unused — do not use; behavior is implementation-defined if accessed.
Notes: Read data is registered (appears one cycle after access). wb_ack_o pulses once per valid access; wb_err_o is always 0.

-- CTRL Register --

CTRL (Control) Register
- Offset: 0x10 (32-bit Wishbone word)
- Width: 32; active control bits [13:0]. Read returns zero-extended value; writes to [31:14] are ignored.
- Reset: 0x0000_0000 (all fields deasserted)
- Access: Read/Write, but writes are accepted only when SPI is idle (!tip). Wishbone byte enables (wb_sel_i) are honored.

Fields (14-bit control payload; exact bit positions per RTL):
- go (start/busy): Write 1 (via lower-byte) when idle to start a transfer. Lower-byte writes OR-merge this bit (writing 0 cannot clear it). Hardware auto-clears it at end-of-transfer (tip && last_bit && pos_edge). Read reflects the current busy/start state.
- char_len[6:0]: Frame length provided to the shifter. Value 0 encodes a 128-bit frame; values 1..127 encode N-bit frames.
- lsb: Bit order; 0 = MSB-first, 1 = LSB-first.
- tx_negedge: TX launch edge; 0 = launch on rising SCK (pos_edge), 1 = launch on falling SCK (neg_edge).
- rx_negedge: RX sample edge; 0 = sample on rising SCK (pos_edge), 1 = sample on falling SCK (neg_edge).
- ie: Interrupt enable for transfer completion. When set, wb_int_o asserts when the final bit completes. The interrupt is cleared by any acknowledged Wishbone access.
- ass: Auto slave-select; 0 = manual mode (ss_pad_o follows SS register continuously, active-low), 1 = automatic mode (ss_pad_o asserted only while a transfer is active, tip=1).
- reserved: Remaining bit(s); read as 0, write as 0.

Behavior and notes:
- Write gating: All CTRL field updates (including edge selects, length, bit order) are ignored while a transfer is active (tip=1) to prevent mid-frame glitches. Byte-lane behavior applies; only the lower-byte write OR-merges go.
- Readback: Returns the current latched control fields, zero-extended to 32 bits.
- End-of-transfer: Hardware clears go on the last bit boundary; if ie=1, wb_int_o is asserted at completion and is cleared by any Wishbone transaction that generates wb_ack_o.
- SS interaction: With ass=1, ss_pad_o asserts only during an active transfer; with ass=0, ss_pad_o reflects the SS register regardless of tip.
- Implementation note: CTRL contains 14 effective control bits; bit positions should be confirmed in the RTL. char_len is 7 bits (len[6:0]).

-- DIVIDER Register --

DIVIDER Register: 32-bit R/W at Wishbone word offset 0x14 within spi_top. Bits [15:0] hold the divider value; bits [31:16] read as 0 and ignore writes. Writes are accepted only when the SPI is idle (tip == 0) to prevent mid-transfer rate changes. Byte-lane enables wb_sel_i[1:0] apply to the lower two bytes; upper lanes are ignored. Readback returns the zero-extended 16-bit value; accesses participate in the Wishbone handshake (wb_dat_o registered, wb_ack_o pulses) and any acknowledged access clears wb_int_o. Function: Programs the SPI SCK half-period in units of clk_in cycles for the clock generator. SCK half-period (cycles) = DIVIDER + 1; SCK frequency f_SCK = f_clk_in / (2 × (DIVIDER + 1)); DIVIDER = 0 selects the fastest rate ≈ f_clk_in/2. Timing/behavior: clgen counter reloads to DIVIDER

when enable == 0 (idle) or cnt == 0; SCK toggles at cnt == 0 when enabled. For DIVIDER > 0, pos_edge/neg_edge strobes assert one clk_in cycle before the SCK toggle (cnt == 1), providing a one-cycle lead; for DIVIDER == 0, strobes are synthesized every clk_in cycle to maintain per-edge pulses at the highest rate. Update timing: A new DIVIDER value takes effect cleanly at the next enable/start of transfer; no mid-transfer changes occur. Reset: divider register default is unspecified in this context (clgen cnt resets to 0xFFFF). Example: clk_in = 100 MHz, DIVIDER = 4 $\rightarrow$ SCK $\approx$ 10 MHz.

-- SS Register --

Chip-Select Mask (SS) register. Address offset: 0x18 on 32-bit Wishbone. Width/Access: 8-bit implemented, read/write; reads return a 32-bit value with bits [31:8]=0. Writes use bits [7:0] only; wb_dat_i[31:8] and upper byte lanes are ignored (only wb_sel_i[0] is meaningful). Function: Each bit controls one active-low chip-select output ss_pad_o[7:0]; 0 asserts (low), 1 deasserts (high). Supports up to 8 chip-selects. Write timing: The SS latch updates only when the core is idle (tip=0). Writes during an active transfer (tip=1) are bus-acknowledged but do not modify the latched SS value, avoiding mid-transfer glitches. Read behavior: Non-destructive; returns the current SS latch, zero-extended. Interaction with CTRL.ASS (auto-slave-select): ASS=0 (manual) — ss_pad_o continuously reflects SS (active-low). ASS=1 (auto) — ss_pad_o are asserted only while a transfer is active (tip=1); when idle, all ss_pad_o are forced high regardless of SS. Usage: Multiple devices can be selected by writing multiple 0s. To keep selection between transfers, use manual mode; in auto mode all SS deassert between transfers. Reset: Not specified in this context.

-- RX Data Windows --

RX Data Windows expose the 128-bit receive buffer as four read-only 32-bit words on the Wishbone bus at word-aligned offsets: 0x0 -> bits [31:0], 0x4 -> [63:32], 0x8 -> [95:64], 0xC -> [127:96] (wb_adr_i[4:2] selects the lane). Read data is registered: wb_dat_o presents the selected 32-bit word with a one-cycle latency. Any acknowledged read of an RX window clears the transfer-done interrupt (wb_int_o). Coherency is not enforced across all four words; there is no read-freeze. Software should read after the transfer completes (tip=0 or upon interrupt) to avoid partial/tearing observations; reads during an active transfer may mix old and new data. For frame lengths < 128, only the received bit positions are updated; untouched bits retain their prior values (all zeros after reset). Bit placement follows the lsb control: lsb=1 (LSB-first) packs earlier bits into lower indices; lsb=0 (MSB-first) packs toward higher indices within the active length. Reset clears the entire buffer to 0, so all windows read back as 0 until a transfer populates them. The same addresses may be used for TX writes when idle, but reads at these addresses always return RX content. The rx_negedge setting only selects the sampling edge and does not alter address or bit mapping.

-- Register Access Rules --

- Address map and widths (32-bit word-aligned):
- 0x00–0x0C: Data window (4 x 32-bit lanes) for RX readback and TX loading.
- 0x10: CTRL (effective 14 bits; read zero-extended to 32 bits).
- 0x14: DIVIDER (16 bits; read zero-extended to 32 bits).
- 0x18: SS (8-bit mask; read zero-extended to 32 bits).
- Read rules:
- Reads are always accepted; no busy gating.
- RX window (0x00/04/08/0C) returns slices of the most recent 128-bit received frame: [31:0], [63:32], [95:64], [127:96].
- CTRL, DIVIDER, and SS read back current values; unused upper bits are zero.
- CTRL.go reflects actual busy state (auto-clears when transfer completes; not software-cleared).
- Write rules and busy gating:
- Effectful writes are honored only when idle (!tip): TX data (0x00–0x0C), CTRL (0x10), DIVIDER

(0x14), SS (0x18).
- While busy (tip=1), writes are acknowledged but have no effect on internal state; no bus error is reported.
- Byte-enable handling (wb_sel_i[3:0]):
- Honored for multi-byte registers and TX lanes.
- TX data lanes accept per-byte updates when idle; partial writes update only selected bytes of the 128-bit TX buffer.
- DIVIDER: only sel[1:0] affect DIVIDER[15:0]; upper lanes are ignored on write.
- SS: only low byte is writable; higher bytes are ignored on write.
- CTRL.go special semantics:
- Writing 1 to bit0 when idle starts a transfer (sets busy/tip).
- Lower-byte writes use OR semantics for bit0: writing 0 cannot clear it.
- Hardware auto-clears go/busy at end of transfer; software must not rely on write-0 to clear.
- Side effects:
- Any acknowledged Wishbone access (read or write at any address) clears wb_int_o.
- Error signaling:
- wb_err_o is never asserted for any access; software must obey idle-only write rule.
- Notes:
- DIVIDER accepts any 16-bit value, including 0; behavior handled internally by clock generator.
- TX window shares addresses with RX readback; host writes are blocked by busy gating as above.

-- Byte-Enable and Alignment Semantics --

- Address alignment: 32-bit Wishbone data bus with word-aligned accesses; wb_adr_i[4:2] selects the 32-bit word, and address bits [1:0] are ignored.
- RX frame mapping: 128-bit RX is exposed as four consecutive 32-bit words: 0x0 -> rx[31:0], 0x4 -> rx[63:32], 0x8 -> rx[95:64], 0xC -> rx[127:96].
- Endianness and lane mapping: wb_sel_i[0..3] map to bus byte lanes [7:0], [15:8], [23:16], [31:24] respectively; narrow fields are placed in least-significant bytes.
- Read semantics: wb_sel_i is ignored; a full 32-bit word is returned. Narrow registers are zero-extended (CTRL 14-bit in low 16 bits, DIVIDER 16-bit, SS 8-bit).
- Write semantics (byte enables): writes are honored only when the SPI is idle (!tip); while tip=1, host writes do not update contents. Per-byte enables apply as follows:
- TX buffer (128-bit): address selects the 32-bit quarter; wb_sel_i[3:0] select which bytes within that 32-bit word are updated.
- CTRL (14-bit at 0x10 in low 16 bits): wb_sel_i[0] updates bits [7:0]; wb_sel_i[1] updates bits [13:8]; wb_sel_i[2] and [3] are ignored. Bit0 of the lower byte is write-OR (sticky) and is auto-cleared by hardware at end of transfer.
- DIVIDER (16-bit at 0x14): wb_sel_i[0] updates bits [7:0]; wb_sel_i[1] updates bits [15:8]; wb_sel_i[2] and [3] are ignored.
- SS (8-bit at 0x18): wb_sel_i[0] updates bits [7:0]; wb_sel_i[1..3] are ignored.
- Invalid/unused lanes: writes to non-implemented byte lanes are benign no-ops; no bus error is generated.

-- Reset Values and Sticky Bits --

Reset values: At reset, spi_top deasserts wb_int_o and drives wb_err_o to 0. Program-visible registers CTRL[13:0], DIVIDER[15:0], and SS[7:0] have unspecified/implementation-defined reset values; firmware must initialize them before starting any transfer. spi_clgen initializes cnt[15:0] to 0xFFFF, clk_out (SCK) to 0, and pos_edge/neg_edge to 0; the divider must be programmed before use. spi_shift initializes cnt to 0, tip (transfer-in-progress) to 0, s_out (MOSI) to 0, and its 128-bit RX/TX buffer to 0, ensuring a clean idle state. Chip-select outputs (ss_pad_o) depend on SS, ASS, and activity (tip); because SS/ASS reset values are unspecified, firmware should program them explicitly to avoid

unintended assertions. Sticky and auto-clear behaviors: CTRL.go is sticky-on-write in the lower byte (new_bit0 = old_bit0 OR write_bit0), preventing software from clearing it with lower-byte writes while active; hardware auto-clears it at end of frame when the final bit completes (tip && last_bit && pos_edge). The interrupt (wb_int_o) sets at the same transfer-complete event when IE=1 and remains asserted (sticky) until any acknowledged Wishbone access occurs; it is cleared on wb_ack_o, not by writing a specific bit. Write gating: writes to CTRL, DIVIDER, and SS are accepted only when idle (!tip); while tip=1 these registers cannot be modified, effectively holding their values through the active transfer.

CORE CONFIGURATION

-- Frame Length Configuration --

Frame length is programmed via CTRL (0x10) field char_len[6:0]. Encoding: char_len=0 selects a 128-bit frame; char_len=N (1..127) selects an N-bit frame. Writes to CTRL (including char_len) are accepted only while the SPI master is idle (tip=0); the programmed length is latched into the shifter when a transfer is started (go asserted while idle). On transfer start, the shifter loads an 8-bit down-counter with {0,char_len} for nonzero lengths or 0x80 for char_len=0; it decrements on each pos_edge strobe, and the final bit occurs when the counter reaches 1. At the final bit, hardware ends the transfer (tip deasserts), auto-clears the start/busy control bit, and asserts an interrupt if IE=1. For frames shorter than 128 bits, only the selected N bit positions are shifted/sampled; bits outside the programmed frame are not modified in the 128-bit RX/TX buffer. Reading CTRL returns the programmed char_len; a read value of 0 indicates the 128-bit setting. Notes: CTRL is 14-bit; char_len is a 7-bit field forwarded to spi_shift.len[6:0]. The sticky lower-byte write behavior applies only to start/busy, not char_len. Software should poll tip=0 before writing char_len, then assert go; frame length can be reprogrammed between transfers without side effects.

-- Bit Order Configuration --

Selects the serial bit order for each SPI frame. Field: CTRL.lsb. Values: 0 = MSB-first (most significant bit transmitted and received first), 1 = LSB-first (least significant bit first). Applies to both MOSI serialization and MISO placement inside spi_shift so that the 128-bit parallel RX view reflects the chosen order. Works for any frame length 1–128 bits (len==0 encodes 128); only the active frame-length bits are shifted/stored, other buffer bits remain unchanged. The shifter preloads the first bit per the selected order to ensure the first SCK edge launches valid data, and transfer completion is gated to stop exactly at the configured length regardless of order. Update rules: CTRL.lsb is writable only while idle (tip=0); the value is latched when a transfer starts (go) and mid-transfer changes have no effect; readback returns the currently programmed value. Independent of clock polarity/phase (tx_negedge, rx_negedge) and SCK divider—only the serial bit traversal is affected. Software must pack TX data to align the intended MSB/LSB of the frame with the selected order; RX readback will match that logical ordering. Reset default for lsb is not specified here (see top-level reset defaults).

-- Launch/Capture Edge Selection --

TX/RX edge selection is programmable per transfer via CTRL.tx_negedge and CTRL.rx_negedge (writable only when idle, !tip). A value of 0 selects the SCK rising edge; 1 selects the SCK falling edge. Edge strobes pos_edge and neg_edge originate in the clock generator: for divider > 0 they assert one clk_in cycle before the actual SCK toggle to provide setup time; for divider == 0 strobes are synthesized

every cycle, with go creating the initial start strobe at maximum rate. The shifter builds tx_clk from the selected strobe (pos_edge or neg_edge) to launch MOSI; the first transmit bit is preloaded while idle so the first selected SCK edge sees valid data. tx_clk is gated by the last-bit indicator to prevent any MOSI update after the final bit. Similarly, rx_clk is built from the selected strobe and aligned to SCK so MISO is sampled exactly at the chosen edge, and it is gated by the last-bit indicator to avoid oversampling. Bit accounting is independent of selection: the counter decrements on pos_edge while tip=1; tip deasserts on last && pos_edge and the interrupt uses (ie && tip && last_bit && pos_edge). Independent TX/RX edge selection enables CPHA-like behavior (launch on one edge, capture on the opposite); CPOL is not explicitly controlled, but clock-generator last_clk gating can shape leading/trailing SCK edges across transfers for mode compliance.

-- SCK Rate Configuration --

- Purpose: Selects the SPI SCK rate by programming the SCK half■period in units of the internal clk_in cycles.
- Register: DIVIDER[15:0] at offset 0x14. Writable only when the core is idle (no transfer in progress); writes during a transfer are ignored. Byte enables supported on wb_sel_i[1:0] (lower two bytes). Readback is zero-extended to 32 bits.
- Frequency relation: For DIVIDER >= 1, SCK half■period = (DIVIDER + 1) × clk_in cycles and f_sck = f_clk_in / (2 × (DIVIDER + 1)). For DIVIDER = 0 (maximum rate), SCK toggles every clk_in cycle when enabled, so f_sck ≈ f_clk_in/2.
- Activity gating: SCK is generated only during an active transfer; between transfers the clock holds a stable idle level and does not free■run. On the first enable after idle, the counter reloads from DIVIDER to ensure a deterministic start.
- Timing strobes (internal): For DIVIDER > 0, pos_edge/neg_edge assert one clk_in cycle before the corresponding SCK edge; for DIVIDER = 0, equivalent strobes are produced every cycle to preserve correct TX/RX timing at the fastest rate.
- End■of■transfer shaping: The final SCK transition can be gated so the last observed edge is well■defined; SCK then holds the idle level between frames.
- Range and granularity: DIVIDER is 16■bit (0–65535). Integer half■period control only; SCK rate is constant for the duration of a frame and should not be changed mid■transfer (writes while active are not taken). Reset initializes SCK low and the counter to 0xFFFF; the first enable reloads from DIVIDER.

-- Chip-Select Mode and Width --

Eight active-low chip-select outputs (ss_pad_o[7:0]) controlled by the 8-bit SS register and the ASS bit in CTRL. Width: 8 CS lines, each asserted with SS bit=0 and deasserted with SS bit=1; multiple CS lines may be asserted simultaneously. Modes: Manual (ASS=0) — ss_pad_o mirrors SS continuously. Auto (ASS=1) — ss_pad_o mirrors SS only while a transfer is active (tip=1); when idle (tip=0), all CS lines are forced deasserted (all 1s). Access: SS is writable only when idle (tip=0) to prevent glitches; SS address is 0x18 (readback zero-extended). Recommendation: Leave unused CS lines deasserted by setting their SS bits to 1.

-- Maximum Frame Size and Buffering --

Maximum frame size per transfer is 128 bits. CTRL.char_len is 7-bit: values 1–127 specify that exact number of bits; 0 encodes a full 128-bit frame. An internal 8-bit down-counter supports lengths up to 128 bits. Buffering is single-frame, single-deep: a 128-bit register in spi_shift both preloads the TX payload and captures RX bits back into the same buffer; reset clears it. The buffer is exposed on the Wishbone bus as four 32-bit lanes, with RX readback at 0x0, 0x4, 0x8, and 0xC returning the most recent 128-bit frame (registered, one-cycle latency). TX data is written into 32-bit quarters with per-byte enables; all writes to CTRL/DIVIDER/SS and the TX buffer are allowed only while idle (tip=0) and are blocked during a transfer (tip=1). There is no FIFO: software must segment payloads >128 bits into

multiple frames using load → start → wait for interrupt/end-of-frame → read. Bit-order (lsb) and edge controls (tx_negedge/rx_negedge) affect serialization and bit placement but not capacity.

-- Timing Strobes Usage --

Timing strobes are single-cycle pulses (pos_edge, neg_edge) generated by spi_clgen in the clk_in domain to announce upcoming SCK (clk_out) edges. For divider > 0, each strobe occurs one clk_in cycle before the corresponding SCK toggle, providing a one-cycle lead for launch/sample preparation. For divider == 0, strobes are synthesized every clk_in cycle (alternating with clk_out) and a go-derived kick produces the initial pos_edge when starting at the fastest rate. Strobes are produced only while enable is asserted (during an active transfer); when disabled, the counter reloads and SCK holds. spi_shift derives tx_clk and rx_clk by selecting pos_edge or neg_edge via tx_negedge and rx_negedge and gates these clocks with the last signal to suppress launches/samples after the final bit. The bit counter decrements on pos_edge; last_bit is detected from this countdown. Transfer termination keys off pos_edge: tip deasserts on (last_bit && pos_edge), and spi_top raises wb_int_o and auto-clears the start/busy CTRL bit on (ie && tip && last_bit && pos_edge). last_clk gating in spi_clgen can suppress the next rising SCK toggle to end the frame cleanly (e.g., on a falling edge), while strobes are still issued so that final launch/sample and termination proceed even without a physical SCK transition. All strobe generation, countdown, and termination decisions are synchronous to clk_in; external SPI pins follow on the subsequent hardware toggle. Divider updates are blocked while tip = 1 to keep strobe cadence stable. In ASS mode, SS asserts with tip so the first strobe occurs while SS is active. Frames always count on pos_edge regardless of selected launch/sample edges; the design does not provide termination on neg_edge.

-- Constraints and Safe Programming Rules --

- Bus access discipline:
- All registers are word-aligned; use wb_sel_i for byte/halfword writes where supported.
- wb_dat_o is registered; expect one-cycle read latency and a single wb_ack_o pulse per access.
- wb_err_o is always 0; do not rely on bus error signaling for protection.
- Any acknowledged Wishbone access clears the interrupt; avoid unrelated bus transactions while an interrupt is pending to prevent spurious clears.
- Write gating (idle-only updates):
- CTRL[13:0], DIVIDER[15:0], SS[7:0], and TX data lanes are writable only when tip=0 (shifter idle). Always read busy (CTRL or wrapper STATUS) and wait for tip=0 before writing.
- CTRL lower byte write OR-merges with existing bit0 (start/busy); software cannot clear start while hardware is active. Hardware auto-clears start at end-of-frame.
- DIVIDER honors wb_sel_i[1:0]; upper bytes are ignored on write and read as zero.
- Transfer framing and start/stop:
- Program DIVIDER, CTRL fields (char_len, lsb, tx_negedge, rx_negedge, ie, ass), SS, and preload TX while tip=0, then assert go.
- char_len is 7 bits: 0 encodes 128 bits; 1..127 encode that many bits. Do not program values >127.
- Do not attempt to retrigger start mid-frame; hardware clears start/busy at end-of-frame.
- Data coherency:
- Block parallel TX writes while tip=1; preload only when idle.
- RX updates during shifting; for a coherent 128-bit read (four words at 0x0/0x4/0x8/0xC), read only when tip=0 (e.g., post-interrupt).
- Interrupt handling:
- Transfer-done interrupt asserts at the final bit when IE=1.
- ISR must perform at least one acknowledged Wishbone access to clear the interrupt; minimize unrelated accesses while the interrupt is pending.
- Chip-select behavior and safety:
- SS bits are active-low.

- Manual mode (ass=0): ss_pad_o follows SS continuously; do not modify SS mid-transfer (writes are blocked by tip) to avoid glitches.
- Auto mode (ass=1): ss_pad_o asserts only while tip=1 and deasserts otherwise. If SS must remain asserted across frames, use manual mode and sequence accordingly.
- Clocking and edge timing (spi_clgen):
- divider is 16-bit; divider=0 yields maximum SCK ≈ clk_in/2. Ensure external devices and timing closure can tolerate this rate.
- For divider>0, pos_edge/neg_edge strobes lead the SCK toggle by one clk_in cycle; do not change edge-select fields after go.
- last_clk gating can suppress leading/trailing edges; do not change related mode fields during a transfer.
- Edge selection and bit order (spi_shift):
- Choose tx_negedge/rx_negedge to match the target SPI mode; do not modify these fields while tip=1.
- lsb selects LSB-first serialization; ensure both ends agree on bit order.
- Reset and initialization:
- rst is asynchronous, active-high. After reset, reprogram DIVIDER, CTRL, SS, and preload TX before starting.
- When enable=0 (idle), the clock generator counter reloads to divider for deterministic restart; do not rely on previous counter values across resets or disables.