QSPI Controller Specification 0.1

Used as the final project for RTL Design with IP implementation course

By Quang Le

1. Introduction

This document specifies a Quad Serial Peripheral Interface (QSPI) controller IP core designed for interfacing with QSPI flash memory devices. The controller supports two primary operating modes: Execute-In-Place (XIP) mode for direct memory-mapped access to the flash, and Command mode for programmable command execution with optional DMA transfers to/from DRAM. The Control and Status Registers (CSRs) are accessed via an APB slave interface. For DMA operations in Command mode, the controller acts as an AXI master to transfer data over the AXI bus. For XIP mode, the controller includes an AXI slave interface to allow memory-mapped reads (and optionally writes) from the system address space, translating them into QSPI transactions.

The controller is designed for high-performance applications, supporting data rates up to quad mode (4 data lines) and configurable transaction formats to accommodate various QSPI flash devices.

2. Features

- Supports Standard SPI, Dual SPI, and Quad SPI modes.
- XIP mode: Memory-mapped access via AXI slave port for low-latency reads (execute code directly from flash).
- Command mode: Programmable commands with optional DMA to/from DRAM via AXI master port.
- Configurable transaction formats: Number of lanes for command, address, mode bits, dummy cycles, and data phases.
- Clock divider for flexible SPI clock generation from system clock.
- Support for 24-bit or 32-bit addressing.
- Interrupt generation for completion, errors, and status changes.
- Error detection: Timeout, overrun, underrun.
- APB slave interface for CSR access (32-bit data width, address space up to 4KB).
- AXI4 master interface for DMA (burst support up to 16 beats, 32/64-bit data width configurable).
- AXI4 slave interface for XIP (supports incremental bursts, cacheable if enabled).

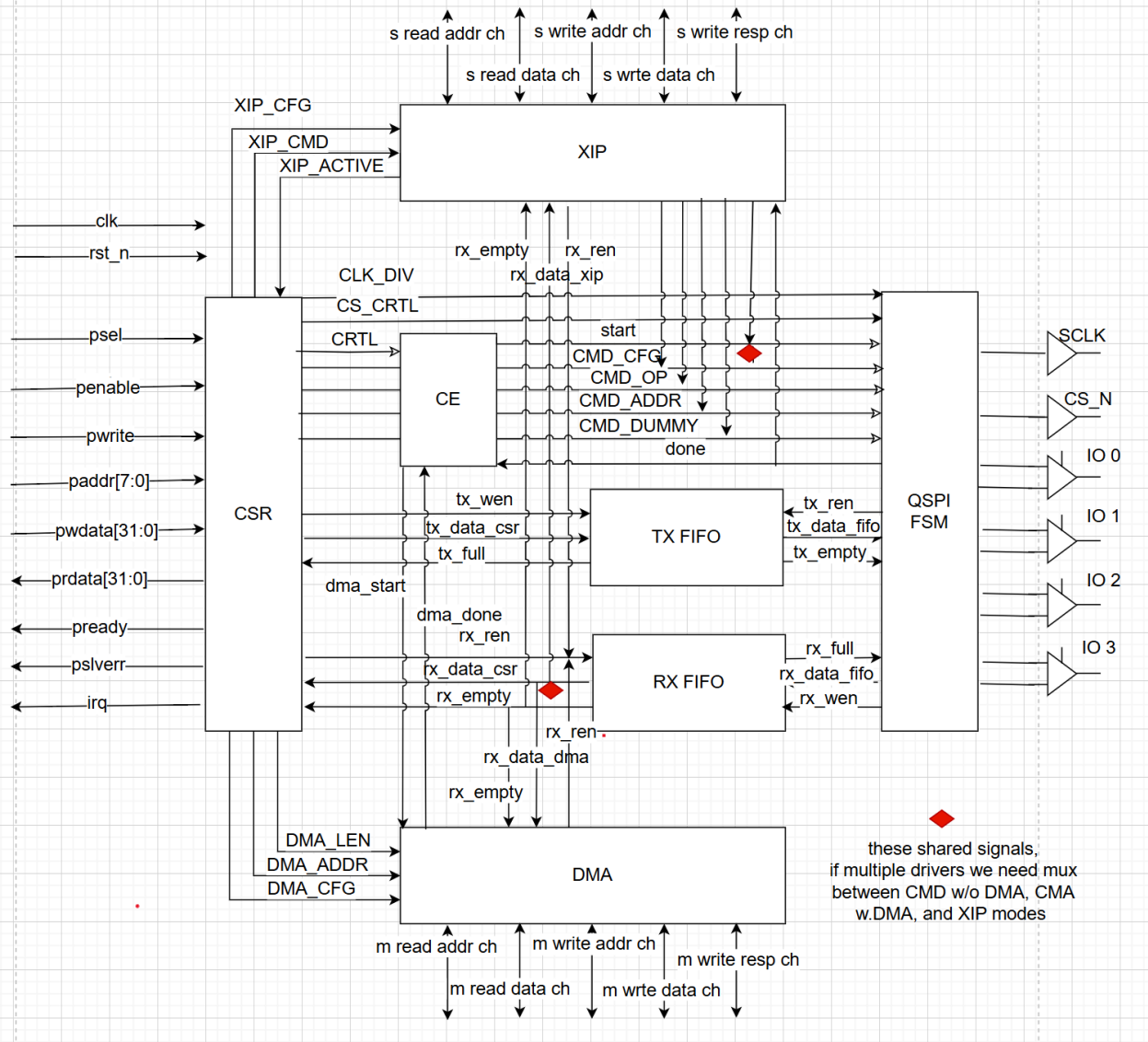
- Power management: Clock gating and low-power modes.

2.1 Configuration Parameters

To enhance flexibility, the QSPI controller IP core includes the following top-level configuration parameters (e.g., generics in VHDL or parameters in Verilog/SystemVerilog). These allow customization at synthesis time for different applications.

Parameter Name	Type	Default Value	Description
DATA_WIDTH	Integer	32	AXI data bus width (32 or 64 bits). Affects AXI master/slave data signals.
AXI_ADDR_WIDTH	Integer	32	AXI address width. Can be increased to higher than 32 bit for 64-bit architecture
FIFO_DEPTH	Integer	16	Depth of TX/RX FIFOs in bytes (power of 2, e.g., 8, 16, 32). For non-DMA operations.
SUPPORT_XIP_WRITE	Boolean	False	Enable write support in XIP mode (typically read-only for execution).
SUPPORT_HOLD_WP	Boolean	False	Enable optional HOLD# and WP# pins for flash devices that require them.
MAX_BURST_LEN	Integer	16	Maximum AXI burst length for DMA (1 to 256, but capped at 16 by default).
APB_ADDR_WIDTH	Integer	12	APB address width (up to 12 bits for 4KB space).

3. Block Diagram



The QSPI controller consists of the following major blocks:

- **Register Bank(CSR)**: Handles read/write APB access to control and status registers. Have registers for control, status, and configuration registers. This block also generate interrupt signal based on status events.
- **Command Engine(CE)**: Executes programmed commands in Command mode, generating QSPI transactions. Data to and from QSPI flash device is delivered by Tx FIFO and Rx FIFO.
- **XIP Engine(XIP)**: Translates AXI slave accesses into QSPI read (or write) transactions. This mode is mutually exclusive with Command mode.

- **DMA Engine(DMA)**: Manages AXI master transactions for data transfer in Command mode. DMA commands are set up by
- **QSPI IO Controller(QSPI FSM)**: Handles serialization/deserialization, clock generation, and pin control (SCLK, CS#, IO0-IO3). QSPI command opcode and flash address come from CSR or CE blocks. Transmited data can be read from Tx FIFO and received data is written into RxFIFO.
- **FIFO Buffers(FIFO)**: TX/RX FIFOs (default to 16-byte but can be changed if needed) for buffering for all operations. QSPI FSM will read Tx fifo to write to QSPI device. And write the Received data to RxFIFO. Depends on the current mode and command. Received data will be read by CSR (Command mode without DMA), or by DMA (Command mode with DMA), or by XIP block (Execution-In-Place Mode).

4. Interfaces

- **APB Slave**: Compliant with AMBA APB v2.0. Address width: 12 bits (4KB space). Data width: 32 bits. Used for CSR configuration and status monitoring.
- **AXI4 Master (for DMA)**: Compliant with AMBA AXI4. Data width: Configurable (32/64 bits). Supports burst transfers (INCR, up to 16 beats). Used to DMA data to/from DRAM in Command mode.
- **AXI4 Slave (for XIP)**: Compliant with AMBA AXI4. Data width: Configurable (32/64 bits). Supports burst reads (INCR, up to 256 beats for prefetching). Base address configurable in system; maps flash address space.
- **QSPI Pins**: SCLK (clock output), CS# (chip select, active low), IO0 (MOSI), IO1 (MISO), IO2, IO3 (for quad modes). Optional: HOLD#, WP# (if supported by flash).
- **Clock and Reset**: System clock (clk), reset (rst_n). Interrupt output (irq).

4.1 Port List

The following tables list all top-level signals for the interfaces, including directions from the perspective of the QSPI controller IP core. Widths are parameterized where applicable (e.g., DATA_WIDTH for AXI data buses). AXI master signals use m_ prefix, and AXI slave signals use s_ prefix to ensure uniqueness.

4.1.1 Clock, Reset, and Interrupt Signals

Signal Name	Direction	Width	Description
clk	Input	1	System clock.

rst_n	Input	1	Active-low reset.
irq	Output	1	Interrupt output.

4.1.2 APB Slave Interface Signals

Signal Name	Direction	Width	Description
pclk	Input	1	APB clock.
presetn	Input	1	APB active-low reset.
paddr	Input	12	Address bus.
psel	Input	1	Slave select.
penable	Input	1	Access enable (strobe).
pwrite	Input	1	Write enable (1: write, 0: read).
pwdata	Input	32	Write data bus.
prdata	Output	32	Read data bus.
pready	Output	1	Ready signal (1: transfer complete).
pslverr	Output	1	Slave error (1: error occurred).

4.1.3 AXI4 Slave Interface Signals (for XIP)

Note: Clock and reset are shared with system clk/rst_n, but AXI may use a separate aclk/aresetn if asynchronous. Here, assumed synchronous. Directions are from slave perspective.

Channel	Signal Name	Direction	Width	Description
Global	s_aclk	Input	1	AXI clock (may be same as clk).
Global	s_aresetn	Input	1	AXI active-low reset. Same with rst_n
Write Address	s_awid	Input	4	Write address ID.

Write Address	s_awaddr	Input	AXI_ADDR_WIDTH	Write address.
Write Address	s_awlen	Input	8	Burst length (0-255).
Write Address	s_awsize	Input	3	Burst size (bytes per beat).
Write Address	s_awburst	Input	2	Burst type (e.g., 01: INCR).
Write Address	s_awvalid	Input	1	Write address valid.
Write Address	s_awready	Output	1	Write address ready.
Write Data	s_wdata	Input	DATA_WIDTH	Write data.
Write Data	s_wstrb	Input	DATA_WIDTH/8	Write strobes.
Write Data	s_wlast	Input	1	Write last beat.
Write Data	s_wuser	Input	1	User signal (optional).
Write Data	s_wvalid	Input	1	Write valid.
Write Data	s_wready	Output	1	Write ready.
Write Response	s_bid	Output	4	Write response ID.
Write Response	s_bresp	Output	2	Write response (e.g., 00: OKAY).
Write Response	s_buser	Output	1	User signal (optional).
Write Response	s_bvalid	Output	1	Write response valid.
Write Response	s_bready	Input	1	Write response ready.

Read Address	s_arid	Input	4	Read address ID.
Read Address	s_araddr	Input	AXI_ADDR_WIDTH	Read address.
Read Address	s_arlen	Input	8	Burst length.
Read Address	s_arsize	Input	3	Burst size.
Read Address	s_arburst	Input	2	Burst type.
Read Address	s_arvalid	Input	1	Read address valid.
Read Address	s_arready	Output	1	Read address ready.
Read Data	s_rid	Output	4	Read ID.
Read Data	s_rdata	Output	DATA_WIDTH	Read data.
Read Data	s_rresp	Output	2	Read response.
Read Data	s_rlast	Output	1	Read last.
Read Data	s_ruser	Output	1	User signal.
Read Data	s_rvalid	Output	1	Read valid.
Read Data	s_rready	Input	1	Read ready.

4.1.4 AXI4 Master Interface Signals (for DMA)

Directions from master perspective (opposite of slave).

Channel	Signal Name	Direction	Width	Description
Global	m_aclk	Input	1	AXI clock.
Global	m_aresetn	Input	1	AXI reset.
Write Address	m_awid	Output	4	Write address ID.

Write Address	m_awaddr	Output	AXI_ADDR_WIDTH	Write address.
Write Address	m_awlen	Output	8	Burst length.
Write Address	m_awsize	Output	3	Burst size.
Write Address	m_awburst	Output	2	Burst type.
Write Address	m_awvalid	Output	1	Write address valid.
Write Address	m_awready	Input	1	Write address ready.
Write Data	m_wdata	Output	DATA_WIDTH	Write data.
Write Data	m_wstrb	Output	DATA_WIDTH/8	Write strobes.
Write Data	m_wlast	Output	1	Write last.
Write Data	m_wuser	Output	1	User signal.
Write Data	m_wvalid	Output	1	Write valid.
Write Data	m_wready	Input	1	Write ready.
Write Response	m_bid	Input	4	Write response ID.
Write Response	m_bresp	Input	2	Write response.
Write Response	m_buser	Input	1	User signal.
Write Response	m_bvalid	Input	1	Write response valid.
Write Response	m_bready	Output	1	Write response ready.
Read Address	m_arid	Output	4	Read address ID.
Read Address	m_araddr	Output	AXI_ADDR_WIDTH	Read address.
Read Address	m_arlen	Output	8	Burst length.
Read Address	m_arsize	Output	3	Burst size.
Read Address	m_arburst	Output	2	Burst type.
Read Address	m_arvalid	Output	1	Read address valid.
Read Address	m_arready	Input	1	Read address ready.
Read Data	m_rid	Input	4	Read ID.

Read Data	m_rdata	Input	DATA_WIDTH	Read data.
Read Data	m_rresp	Input	2	Read response.
Read Data	m_rlast	Input	1	Read last.
Read Data	m_ruser	Input	1	User signal.
Read Data	m_rvalid	Input	1	Read valid.
Read Data	m_rready	Output	1	Read ready.

4.1.5 QSPI Interface Signals

Signal Name	Direction	Width	Description
sclk	Output	1	Serial clock output.
cs_n	Output	1	Chip select (active low).
io0	Inout	1	Data I/O 0 (MOSI in single mode).
io1	Inout	1	Data I/O 1 (MISO in single mode).
io2	Inout	1	Data I/O 2 (for dual/quad modes).
io3	Inout	1	Data I/O 3 (for quad modes).
hold_n	Output	1	Hold signal (optional, if enabled).
wp_n	Output	1	Write protect (optional, if enabled).

5. Operating Modes

5.1 Command Mode

In Command mode, the CPU programs the controller via CSRs to execute specific QSPI commands (e.g., read, write, erase). Data can be transferred via small FIFOs or DMA to/from DRAM.

- Configure transaction format (lanes, dummy cycles, etc.).
- Set command opcode, address, data length.
- If DMA enabled, set DRAM address and direction.
- Trigger execution.
- Wait for completion interrupt or poll status.

- Supports read/write/erase commands with optional mode bits.
- DMA: Controller initiates AXI bursts to fetch/write data (if full AXI support)

5.2 XIP Mode

In XIP mode, the flash is memory-mapped via the AXI slave port. System accesses to the mapped address space are translated into QSPI fast read commands.

- Configure XIP settings (read opcode, dummy cycles, lanes).
- Enable XIP mode.
- CPU or other masters access the AXI slave address space; controller performs on-the-fly QSPI reads.
- Writes can be supported if configured, but typically XIP is read-only for execution.
- Supports continuous read modes for performance.
- Optional prefetching to reduce latency.

6. Register Map

All registers are 32-bit wide, accessed via APB. Addresses are offsets from the base address. Reserved bits should be written as 0 and ignored on read.

Offset	Register Name	Access	Description
0x000	ID	RO	Device ID and Version
0x004	CTRL	RW	Control Register
0x008	STATUS	RO	Status Register
0x00C	INT_EN	RW	Interrupt Enable
0x010	INT_STAT	RW1C	Interrupt Status
0x014	CLK_DIV	RW	Clock Divider
0x018	CS_CTRL	RW	Chip Select Control
0x01C	XIP_CFG	RW	XIP Configuration
0x020	XIP_CMD	RW	XIP Command Configuration
0x024	CMD_CFG	RW	Command Mode Configuration

0x028	CMD_OP	RW	Command Opcode
0x02C	CMD_ADDR	RW	Command Address
0x030	CMD_LEN	RW	Command Data Length
0x034	CMD_DUMMY	RW	Command Dummy Cycles
0x038	DMA_CFG	RW	DMA Configuration
0x03C	DMA_ADDR	RW	DMA DRAM Address
0x040	DMA_LEN	RW	DMA Length
0x044	FIFO_TX	WO	TX FIFO Data
0x048	FIFO_RX	RO	RX FIFO Data
0x04C	FIFO_STAT	RO	FIFO Status
0x050	ERR_STAT	RO	Error Status

(Additional reserved space up to 0xFFF.)

7. Register Descriptions

7.1 ID (Offset 0x000, RO)

Device identification and version.

- Bits [31:16]: Vendor ID (e.g., 0x0A10 for example).
- Bits [15:8]: Device ID (e.g., 0x01 for QSPI).
- Bits [7:0]: Version (e.g., 0x01 for v1.0).

7.2 CTRL (Offset 0x004, RW)

Main control register.

- Bit 0: ENABLE -- 1: Enable controller; 0: Disable.
- Bit 1: XIP_EN -- 1: Enable XIP mode; 0: Disable (Command mode active).
- Bit 2: QUAD_EN -- 1: Enable quad mode by default; 0: Single/dual based on config.
- Bit 3: CPOL -- Clock polarity (0: idle low; 1: idle high).
- Bit 4: CPHA -- Clock phase (0: sample on first edge; 1: second edge).
- Bit 5: LSB_FIRST -- 1: LSB first; 0: MSB first.

- Bits [7:6]: Reserved.
- Bit 8: CMD_TRIGGER -- Write 1 to start Command mode execution (self-clearing).
- Bit 9: DMA_EN -- 1: Enable DMA for Command mode; 0: Use FIFO.
- Bits [31:10]: Reserved.

7.3 STATUS (Offset 0x008, RO)

Current status.

- Bit 0: BUSY -- 1: Operation in progress; 0: Idle.
- Bit 1: XIP_ACTIVE -- 1: XIP mode active.
- Bit 2: CMD_DONE -- 1: Command completed.
- Bit 3: DMA_DONE -- 1: DMA transfer completed.
- Bits [31:4]: Reserved.

7.4 INT_EN (Offset 0x00C, RW)

Interrupt enable mask.

- Bit 0: CMD_DONE_EN -- Enable interrupt on command completion.
- Bit 1: DMA_DONE_EN -- Enable interrupt on DMA completion.
- Bit 2: ERR_EN -- Enable interrupt on error.
- Bit 3: FIFO_TX_EMPTY_EN -- Enable on TX FIFO empty.
- Bit 4: FIFO_RX_FULL_EN -- Enable on RX FIFO full.
- Bits [31:5]: Reserved.

7.5 INT_STAT (Offset 0x010, RW1C)

Interrupt status (write 1 to clear).

- Bit 0: CMD_DONE -- Command completed.
- Bit 1: DMA_DONE -- DMA completed.
- Bit 2: ERR -- Error occurred.
- Bit 3: FIFO_TX_EMPTY -- TX FIFO empty.
- Bit 4: FIFO_RX_FULL -- RX FIFO full.

- Bits [31:5]: Reserved.

7.6 CLK_DIV (Offset 0x014, RW)

Clock divider for SCLK generation (SCLK = clk /2^DIV)

- Bits [2:0]: DIV -- Divider value (0-7).
    - 0x0 – no divider value 0 SCLK is same frequency with clk
    - 0x1 – SCLK = clk /2
    - 0x2 – SCLK = clk/4
    - 0x3 – SCLK = clk/8
    - 0x4 – SCLK = clk/16
    - 0x5 – SCLK = clk /32
    - 0x6 – SCLK = clk/64
    - 0x7 – SCLK = clk/128
- Bits [31:3]: Reserved.

7.7 CS_CTRL (Offset 0x018, RW)

Chip select control.

- Bit 0: CS_AUTO -- 1: Automatic CS# management; 0: Manual.
- Bit 1: CS_LEVEL -- Manual CS# level (0: assert; 1: deassert).
- Bits [3:2]: CS_DELAY -- Inter-transaction delay cycles (0-3).
- Bits [31:4]: Reserved.

7.8 XIP_CFG (Offset 0x01C, RW)

XIP mode configuration.

- Bits [1:0]: CMD_LANES -- 0: Single; 1: Dual; 2: Quad.
- Bits [3:2]: ADDR_LANES -- 0: Single; 1: Dual; 2: Quad.
- Bits [5:4]: DATA_LANES -- 0: Single; 1: Dual; 2: Quad.
- Bits [7:6]: ADDR_BYTES -- 0: 0 bytes; 1: 3 bytes; 2: 4 bytes.

- Bit 8: MODE_EN -- 1: Send mode bits. (Mode bits are sent with same number of lanes as in ADDR_LANES.)
- Bits [12:9]: DUMMY_CYCLES -- 0-15 cycles.
- Bit 13: CONT_READ -- 1: Enable continuous read mode.
- Bit 14: WRITE_EN -- 1: Allow writes in XIP (if flash supports).
- Bits [31:15]: Reserved.

7.9 XIP_CMD (Offset 0x020, RW)

XIP command settings.

- Bits [7:0]: READ_OP -- Read opcode (e.g., 0xEB for quad fast read).
- Bits [15:8]: WRITE_OP -- Write opcode (if enabled).
- Bits [23:16]: MODE_BITS -- Mode bits to send after address.
- Bits [31:24]: Reserved.

7.10 CMD_CFG (Offset 0x024, RW)

Command mode transaction format.

- Bits [1:0]: CMD_LANES -- 0: Single; 1: Dual; 2: Quad.
- Bits [3:2]: ADDR_LANES -- 0: Single; 1: Dual; 2: Quad.
- Bits [5:4]: DATA_LANES -- 0: Single; 1: Dual; 2: Quad.
- Bits [7:6]: ADDR_BYTES -- 0: 0 bytes; 1: 3 bytes; 2: 4 bytes.
- Bit 8: MODE_EN -- 1: Send mode bits. Mode bits are sent with same number of lanes as in ADDR_LANES.)
- Bits [12:9]: DUMMY_CYCLES -- 0-15 cycles.
- Bit 13: DIR -- 0: Write (TX); 1: Read (RX).
- Bits [31:14]: Reserved.

7.11 CMD_OP (Offset 0x028, RW)

Command opcode and mode.

- Bits [7:0]: OPCODE -- Command opcode.
- Bits [15:8]: MODE_BITS -- Mode bits (if enabled).

- Bits [31:16]: Reserved.

7.12 CMD_ADDR (Offset 0x02C, RW)

Command address.

- Bits [31:0]: ADDR -- Flash address (lower 32 bits; upper if needed via extension).

7.13 CMD_LEN (Offset 0x030, RW)

Data length for command.

- Bits [31:0]: LEN -- Number of bytes (0 for no data phase).

7.14 CMD_DUMMY (Offset 0x034, RW)

Additional dummy config (if needed beyond CMD_CFG).

- Bits [7:0]: EXTRA_DUMMY -- Extra dummy cycles.
- Bits [31:8]: Reserved.

7.15 DMA_CFG (Offset 0x038, RW)

DMA configuration.

- Bits [3:0]: BURST_SIZE -- Burst length (0:1, 1:2, ..., 4:16).
- Bit 4: DIR -- 0: DMA write to flash (from DRAM); 1: DMA read from flash (to DRAM).
- Bit 5: INCR_ADDR -- 1: Increment DRAM address; 0: Fixed.
- Bits [31:6]: Reserved.

7.16 DMA_ADDR (Offset 0x03C, RW)

DRAM address for DMA.

- Bits [31:0]: ADDR -- Starting DRAM address (AXI-aligned).

7.17 DMA_LEN (Offset 0x040, RW)

DMA transfer length.

- Bits [31:0]: LEN -- Number of bytes to transfer.

7.18 FIFO_TX (Offset 0x044, WO)

TX FIFO data port (for non-DMA).

- Bits [31:0]: DATA -- Write data (byte-packed).

7.19 FIFO_RX (Offset 0x048, RO)

RX FIFO data port (for non-DMA).

- Bits [31:0]: DATA -- Read data (byte-packed).

7.20 FIFO_STAT (Offset 0x04C, RO)

FIFO status.

- Bits [3:0]: TX_LEVEL -- TX FIFO level (0-16).
- Bits [7:4]: RX_LEVEL -- RX FIFO level (0-16).
- Bit 8: TX_EMPTY -- 1: TX empty.
- Bit 9: RX_FULL -- 1: RX full.
- Bits [31:10]: Reserved.

7.21 ERR_STAT (Offset 0x050, RO)

Error status.

- Bit 0: TIMEOUT -- Transaction timeout.
- Bit 1: OVERRUN -- RX overrun.
- Bit 2: UNDERRUN -- TX underrun.
- Bit 3: AXI_ERR -- AXI bus error.
- Bits [31:4]: Reserved.

This table compiles all register fields, bit ranges, access types, and descriptions for easy reference. Defaults are assumed 0 unless specified. Reserved bits are not listed.

Register Name	Offset	Bit Range	Field Name	Access	Description	Default
ID	0x000	31:16	Vendor ID	RO	Vendor ID (e.g., 0x0A10).	-

ID	0x000	15:8	Device ID	RO	Device ID (e.g., 0x01).	-
ID	0x000	7:0	Version	RO	Version (e.g., 0x01).	-
CTRL	0x004	0	ENABLE	RW	1: Enable controller; 0: Disable.	0
CTRL	0x004	1	XIP_EN	RW	1: Enable XIP mode; 0: Disable.	0
CTRL	0x004	2	QUAD_EN	RW	1: Enable quad mode by default.	0
CTRL	0x004	3	CPOL	RW	Clock polarity (0: idle low).	0
CTRL	0x004	4	CPHA	RW	Clock phase (0: sample on first edge).	0
CTRL	0x004	5	LSB_FIRST	RW	1: LSB first; 0: MSB first.	0
CTRL	0x004	8	CMD_TRIGGER	RW	Write 1 to start Command mode (self-clearing).	0
CTRL	0x004	9	DMA_EN	RW	1: Enable DMA for Command mode.	0
STATUS	0x008	0	BUSY	RO	1: Operation in progress.	-

STATUS	0x008	1	XIP_ACTIVE	RO	1: XIP mode active.	-
STATUS	0x008	2	CMD_DONE	RO	1: Command completed.	-
STATUS	0x008	3	DMA_DONE	RO	1: DMA transfer completed.	-
INT_EN	0x00C	0	CMD_DONE_EN	RW	Enable interrupt on command completion.	0
INT_EN	0x00C	1	DMA_DONE_EN	RW	Enable interrupt on DMA completion.	0
INT_EN	0x00C	2	ERR_EN	RW	Enable interrupt on error.	0
INT_EN	0x00C	3	FIFO_TX_EMPTY_EN	RW	Enable on TX FIFO empty.	0
INT_EN	0x00C	4	FIFO_RX_FULL_EN	RW	Enable on RX FIFO full.	0
INT_STAT	0x010	0	CMD_DONE	RW1C	Command completed.	-
INT_STAT	0x010	1	DMA_DONE	RW1C	DMA completed.	-
INT_STAT	0x010	2	ERR	RW1C	Error occurred.	-
INT_STAT	0x010	3	FIFO_TX_EMPTY	RW1C	TX FIFO empty.	-
INT_STAT	0x010	4	FIFO_RX_FULL	RW1C	RX FIFO full.	-

CLK_DIV	0x014	7:0	DIV	RW	Divider value (0-255).	0
CS_CTRL	0x018	0	CS_AUTO	RW	1: Automatic CS# management.	0
CS_CTRL	0x018	1	CS_LEVEL	RW	Manual CS# level (0: assert).	0
CS_CTRL	0x018	3:2	CS_DELAY	RW	Inter-transaction delay cycles (0-3).	0
XIP_CFG	0x01C	1:0	CMD_LANES	RW	0: Single; 1: Dual; 2: Quad.	0
XIP_CFG	0x01C	3:2	ADDR_LANES	RW	0: Single; 1: Dual; 2: Quad.	0
XIP_CFG	0x01C	5:4	DATA_LANES	RW	0: Single; 1: Dual; 2: Quad.	0
XIP_CFG	0x01C	7:6	ADDR_BYTES	RW	0: 0 bytes; 1: 3 bytes; 2: 4 bytes.	0
XIP_CFG	0x01C	8	MODE_EN	RW	1: Send mode bits.	0
XIP_CFG	0x01C	12:9	DUMMY_CYCLES	RW	0-15 cycles.	0
XIP_CFG	0x01C	13	CONT_READ	RW	1: enable 0: disable	0
XIP_CFG	0x01C	14	WRITE_EN	RW	1: enable 0: disable	0
XIP_CMD	0x020	7:0	READ_OP	RW	Read opcode (e.g., 0xEB).	0

XIP_CMD	0x020	15:8	WRITE_OP	RW	Write opcode.	0
XIP_CMD	0x020	23:16	MODE_BITS	RW	Mode bits after address.	0
CMD_CFG	0x024	1:0	CMD_LANES	RW	0: Single; 1: Dual; 2: Quad.	0
CMD_CFG	0x024	3:2	ADDR_LANES	RW	0: Single; 1: Dual; 2: Quad.	0
CMD_CFG	0x024	5:4	DATA_LANES	RW	0: Single; 1: Dual; 2: Quad.	0
CMD_CFG	0x024	7:6	ADDR_BYTES	RW	0: 0 bytes; 1: 3 bytes; 2: 4 bytes.	0
CMD_CFG	0x024	8	MODE_EN	RW	1: Send mode bits.	0
CMD_CFG	0x024	12:9	DUMMY_CYCLES	RW	0-15 cycles.	0
CMD_CFG	0x024	13	DIR	RW	0: Write (TX); 1: Read (RX).	0
CMD_OP	0x028	7:0	OPCODE	RW	Command opcode.	0
CMD_OP	0x028	15:8	MODE_BITS	RW	Mode bits (if enabled).	0
CMD_ADDR	0x02C	31:0	ADDR	RW	Flash address.	0
CMD_LEN	0x030	31:0	LEN	RW	Number of data bytes (0 for no data).	0

CMD_DUMMY	0x034	7:0	EXTRA_DUMMY	RW	Extra dummy cycles.	0
DMA_CFG	0x038	3:0	BURST_SIZE	RW	Burst length (0:1, ..., 4:16).	0
DMA_CFG	0x038	4	DIR	RW	0: DMA write to flash; 1: Read to DRAM.	0
DMA_CFG	0x038	5	INCR_ADDR	RW	1: Increment DRAM address.	0
DMA_ADDR	0x03C	31:0	ADDR	RW	Starting DRAM address.	0
DMA_LEN	0x040	31:0	LEN	RW	Number of bytes to transfer.	0
FIFO_TX	0x044	31:0	DATA	WO	Write data (byte-packed).	-
FIFO_RX	0x048	31:0	DATA	RO	Read data (byte-packed).	-
FIFO_STAT	0x04C	3:0	TX_LEVEL	RO	TX FIFO level (0-16).	-
FIFO_STAT	0x04C	7:4	RX_LEVEL	RO	RX FIFO level (0-16).	-
FIFO_STAT	0x04C	8	TX_EMPTY	RO	1: TX empty.	-
FIFO_STAT	0x04C	9	RX_FULL	RO	1: RX full.	-
ERR_STAT	0x050	0	TIMEOUT	RO	Transaction timeout.	-

ERR_STAT	0x050	1	OVERRUN	RO	RX overrun.	-
ERR_STAT	0x050	2	UNDERRUN	RO	TX underrun.	-
ERR_STAT	0x050	3	AXI_ERR	RO	AXI bus error.	-

8. Base design implementation

8.1 Work on base design first

The specification has a lot of configuration bits that can make implementation complicated to plan and implement logic for them right at the beginning. So usually designer can work on a base design then add new features later. So we should ignore some configuration register bits in base design.

Here is the suggestion.

- CRTL reg bit 5 to bit 0

- CLK_DIV reg

- CS_CRTL reg

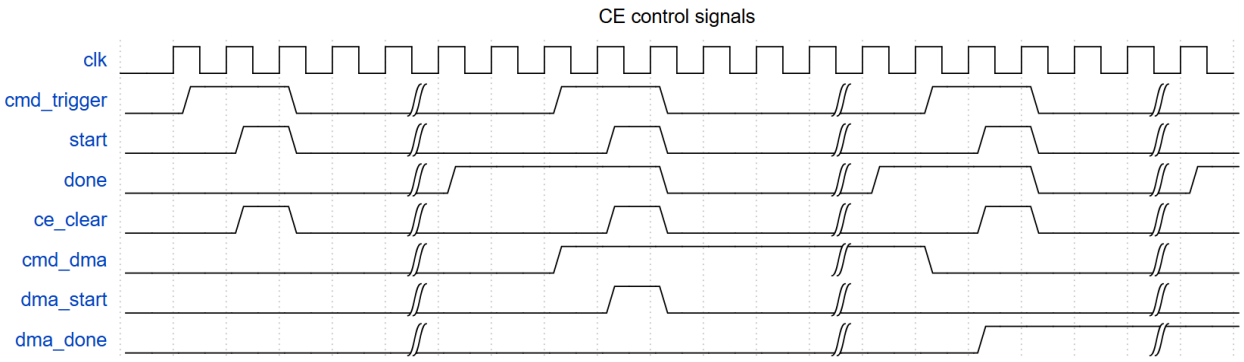
8.2 Command Engine (CE) Guide

Command engine block is a simple block but critical to help start a command based on write to CRTL register and manage the control of QSPI FSM and DMA block (if enable) until the command is complete then it can be ready for a next command.

CE takes the signals CMD_TRIGGER and CMD_DMA from the CSR block. It sends back a ce_clear signal to help clear the CSR bit CMD_TRIGGER so cpu can issue a future command. It sends a start signal to QSPIFSM to start the command for QSPI and wait for a a new done signal from the QSPIFSM (note that QSPIFSM also uses the start signal to clear the done signal of the last command.

Similarly, CE sends a dma_start signal to the DMA block if CMD_DMA is also asserted. The DMA block will clear the dma_done signal if it is still asserted and start the DMA operation for the new command. Note that DMA can only start a new AXI transaction to the bus if it sees RXFIO becomes not empty, then it can read the data from there.

The following timing diagram is provided for reference. The actual doesn’t need to follow exact same cycle by cycle.



Note:

1. ce_start in this case have same timing with start so we can use same signal that is sent to qpsifsm as the same signal sent to csr.
2. Every time “start” asserted the “done” signal needs to reset to 0.
3. Only the second command has cmd_dma. The “done” and “dma_done” has different timings since DMA operation complete a little later compared to QSPI operation.
4. Since there is no new dma command “dma_done” remained asserted waiting for the next “dma_start”

8.3 QSPI IO Controller (QSPI-FSM) Guide

QSPI flash devices can specify commands that can be different across multiple vendors so we need to design a flexible multiple lanes SPI state machine that can output any commands in combinations of lanes, address bytes, dummy cycles to meet the requirements of current and new flash devices.

The programmer needs to program all of the fields in XIP_CFG or CMD_CFG so the state machine will use those fields to know if the command has a ADDR, DATA, or DUMMY cycles and MODE bits to use in the command.

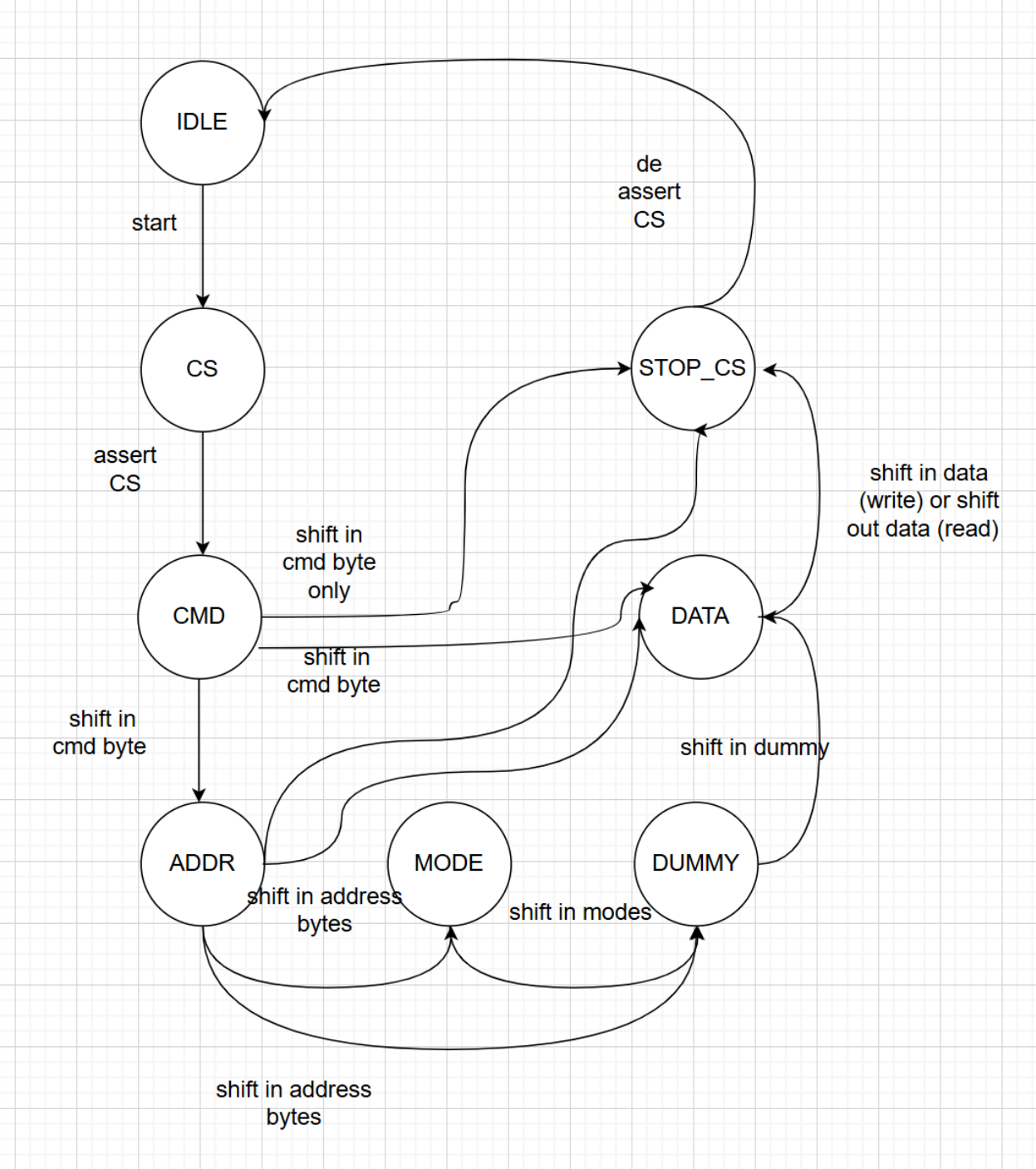
The following is a programming guide for some of QSPI commands in Macronix flash specification NX25L6435F for Command Mode

Command (OPCODE)	CMD LANES	ADDR LANES	DATA LANES	ADDR BYTES	MODE EN	DUMMY CYCLES	DIR	DATA LEN	
0x03	1(0x0)	1	1	3	0	0	Read	>=1	
0x0B	1	1	1	3	0	8	Read	>=1	
0xBB	1	2	2	3	0/1	4	Read	>=1	

0x6B	1	1	4	3	0	8	Read	>=1	
0xEB	1	4	4	3	0/1	6	Read	>=1	
0x20	1	1	0	3	0	0	Write	0	
0xD8	1	1	0	3	0	0	Write	0	
0x60	1	1	0	0	0	0	Write	0	
0x02	1	1	1	3	0	0	Write	256	
0x38	1	4	4	3	0	0	Write	256	
0x06	1	0	0	0	0	0	Write	0	

For XIP mode, ADDR bits and read and write direction come from AXI transactions. Other config bits provided by XIP_CFG and XIP_CMD registers.

A pseudo state machine is enclosed below.



9. Implementation Guidelines

Implement the IP in the following order, targeting completion of step 10 within 4 weeks. Report status, issues, and progress in a presentation.

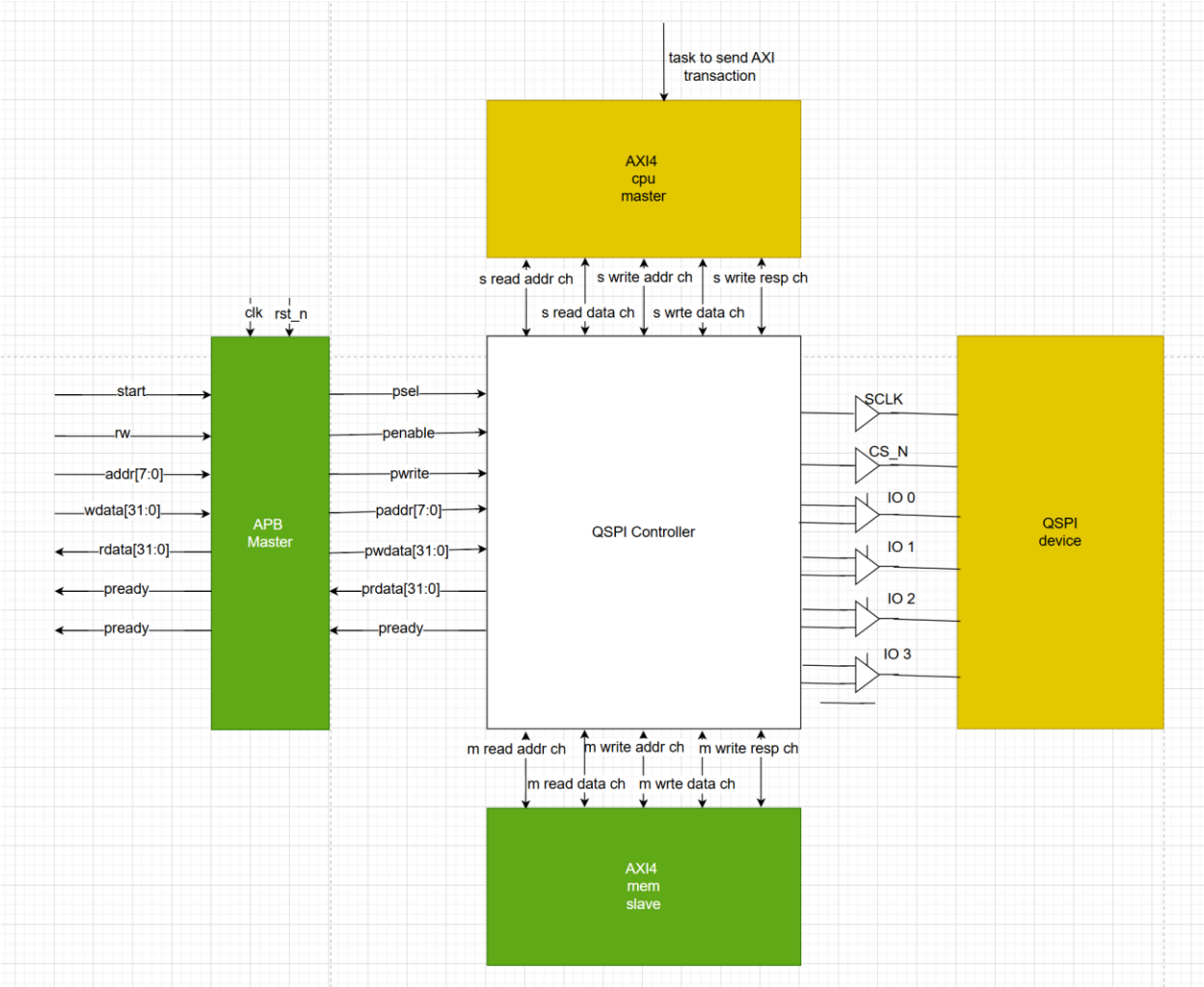
1. Top-Level Parameters: Define all parameters (DATA_WIDTH, AXI_ADDR_WIDTH, etc.) with default values in VHDL/Verilog.

2. APB Interface and CSR: Implement all CSRs with APB decoding. Test write/read functionality. Status bits (e.g., BUSY) may read as X until implemented.
3. Clock Divider: Set CLK_DIV=0x0 (SCLK=clk) for single clock domain operation initially.
4. Command Engine: Implement CE with inputs/outputs to QSPI IO Controller.
5. QSPI IO Controller: Implement FSM to handle read/write commands (e.g., Read Data, Page Program) for single, dual, quad lanes.
6. FIFOs: Reuse TX/RX FIFO designs from Lab 5 (default FIFO_DEPTH=16).
7. Test Command Mode (Non-DMA):
    - a. Write one-byte command (single lane, e.g., Write Enable 0x06).
    - b. Write multi-byte command, no data (e.g., Sector Erase 0xD8).
    - c. Write multi-byte command with data (e.g., Page Program 0x02).
    - d. Read one-byte command (e.g., Read Status 0x05).
    - e. Read multi-byte command (e.g., Fast Read 0x0B).
    - f. Repeat a-e with dual lanes (address/data).
    - g. Repeat a-f with quad lanes (address/data).
8. DMA Read Mode:
    - a. Implement AXI4-Lite (ignore full AXI signals like awlock, awcache).
    - b. Reuse Lab 5 DMA write FSM for AXI master writes.
    - c. Implement concurrent QSPI reads and AXI writes (no RX FIFO full concern due to QSPI rate < AXI rate).
    - d. Single command for full DMA length; pause/resume not required.
9. XIP Structure: Connect XIP Engine signals to CSRs (e.g., XIP_CFG, XIP_CMD).
10. Synthesis: Optimize for area, target >200 MHz. **(This is the milestone for the end of class)**
11. XIP Reads: Implement XIP read mode (one AXI command = one QSPI command, AXI-Lite limited performance).
12. DMA Write Mode: Implement AXI read to QSPI write.

13. XIP Writes: Implement write support for XIP (requires SUPPORT_XIP_WRITE=True).
14. Full AXI4: Implement full AXI4 transactions; measure performance vs. AXI4-Lite.
15. Clock Divider and Clock Phase Modes: Implement CLK_DIV modes (SCLK = clk / 2^DIV). And implement different SPI clock modes (0-3) No full asynchronous clock domain crossing needed.

Verification Environment:

- Reuse APB master and AXI slave from Lab 5.
- Use provided QSPI device model and AXI master BFM.
- Test cases: Validate each command mode test (7a-g) with specific opcodes, addresses, and data patterns.



Challenges:

- Understand all CSR bits before design to avoid conflicts.
- CSRs are fixed for software compatibility; propose new error bits if needed (e.g., WP# blocked write).
- AXI signals follow standard; removed signals (awlock, awcache) are not used.
- Clarify doubts with the instructor to ensure correct assumptions**.**

10. Glossary

- **Lanes**: Number of IO pins used for data transfer (1=Single SPI, 2=Dual SPI, 4=Quad SPI).
- **Burst Beats**: Number of data transfers in an AXI burst (e.g., awlen=7 means 8 beats).
- **CS#**: Chip Select, active-low signal to select the flash device.
- **HOLD#**: Optional pin to pause QSPI communication without deselecting (active low).
- **WP#**: Optional write protect pin to block write/erase commands (active low).
- **SCLK**: Serial clock generated by QSPI IO Controller.
- **XIP**: Execute-In-Place, direct memory-mapped access to flash via AXI slave.