QUAD SPI FLASH CONTROLLER

AMNA ARSHAD KHAN

AUGUST 1, 2024

Contents

1. Introduction

The Quad Serial Peripheral Interface (QSPI) is an advanced extension of the Serial Peripheral Interface (SPI) protocol, designed to enhance data transfer rates by utilizing multiple data lines. This report details the design, implementation, and testing of a flexible and configurable QSPI flash memory controller, which aims to interface with various flash memory devices through multiple SPI modes and clock settings.

2. QSPI Flash Controller Report

2.1 Objective and Learning Outcomes

The primary objective of this project is to design a flexible and configurable QSPI flash memory controller capable of interfacing with a variety of flash memory devices. The controller supports multiple SPI modes and accommodates various CPOL and CPHA configurations. Key learning outcomes include:

- **Understanding SPI and QSPI Protocols:** In-depth knowledge of Single, Dual, and Quad SPI interfaces.
- **FPGA Design and Verilog:** Skills in FPGA design using Verilog, with a focus on state machines and tri-state buffer management.
- **Flexible Controller Design:** Creation of a controller that can handle multiple communication modes and configurations.
- **Simulation and Testing:** Verification of controller functionality across different modes and parameters.

2.2 Project Description

The QSPI Flash Controller interfaces with flash memory devices using the SPI protocol. It operates in three modes:

- **Single SPI (SPI) Mode:** Standard SPI mode with one data line for both sending and receiving.
- **Quad SPI (QSPI) Mode:** Enhanced mode with four data lines, allowing for higher data transfer rates.
- **Dual SPI (DPI) Mode:** Uses two data lines, providing a compromise between standard SPI and Quad SPI.

The controller supports all four CPOL and CPHA combinations:

- **CPOL (Clock Polarity):** Determines the idle state of the clock line.
- **CPHA (Clock Phase):** Determines whether data is sampled on the rising or falling edge of the clock.

3. Basics of SPI

3.1 SPI Protocol

SPI (Serial Peripheral Interface) is a synchronous serial communication protocol that facilitates data exchange between a master device and one or more slave devices. It supports full-duplex communication and operates using a master-slave architecture.

Basic SPI Communication Lines:

- **MOSI (Master Out Slave In):** Line for transmitting data from the master to the slave.
- **MISO (Master In Slave Out):** Line for transmitting data from the slave to the master.
- **SCK (Serial Clock):** Clock signal line generated by the master to synchronize data transfer.
- **SS (Slave Select):** Line used to select the specific slave device for communication.

4. QSPI Modes

QSPI extends the SPI protocol by increasing the number of data lines, which enhances data throughput. It operates in several modes:

4.1 Standard SPI Mode (Single SPI)

- **Data Lines:** Utilizes one data line for input (MOSI) and one for output (MISO).
- **Data Transfer:** Achieves simplex or full-duplex communication.
- **Synchronization:** Managed via the clock signal (SCK).

4.2 Dual SPI (DPI)

- **Data Lines:** Employs two data lines for input and two for output.
- **Data Transfer Rate:** Doubles the data transfer rate compared to standard SPI.
- **Configuration:** Supports both simplex and full-duplex communication.

4.3 Quad SPI (QSPI)

- **Data Lines:** Uses four data lines for both input and output.
- **Data Transfer Rate:** Quadruples the data transfer rate compared to standard SPI.
- **Data Transfer:** Supports full-duplex communication, allowing simultaneous read and write operations.

5. Working of QSPI

QSPI enhances data transfer rates by utilizing four parallel data lines. Key aspects of QSPI operation include:

- **Data Lines:** Four data lines (IO0, IO1, IO2, IO3) enable simultaneous data transmission and reception.

- **Clock Line:** A single clock line (SCK) synchronizes data transmission. The clock frequency affects data transfer speed.
- **Command/Address Lines:** May use additional lines for commands or addresses, depending on the implementation.
- **Data Frames:** Data is transmitted in frames, with multiple bits sent simultaneously across the four data lines.

6. Types of QSPI Modes

QSPI operates in various modes based on clock polarity (CPOL) and clock phase (CPHA) settings:

- **SPI Mode 0 (CPOL = 0, CPHA = 0):** Clock is idle low; data is sampled on the rising edge of the clock.
- **SPI Mode 1 (CPOL = 0, CPHA = 1):** Clock is idle low; data is sampled on the falling edge of the clock.
- **SPI Mode 2 (CPOL = 1, CPHA = 0):** Clock is idle high; data is sampled on the falling edge of the clock.
- **SPI Mode 3 (CPOL = 1, CPHA = 1):** Clock is idle high; data is sampled on the rising edge of the clock.

7. Digital Hardware Requirements

Implementing a QSPI controller requires the following digital hardware components:

- **Master and Slave Devices:** Devices that support QSPI communication.
- **Clock Generator:** Provides the clock signal for synchronization.
- **Data Buffers:** Manage the multiple data lines (IO0, IO1, IO2, IO3).
- **Control Logic:** Manages data transfer, commands, and address lines.
- **Interfacing Logic:** Converts between different data formats or protocols if necessary.

8. Block Diagram of QSPI System

Below is a simplified block diagram for a QSPI system:

9. How SPI, Dual SPI, and Quad SPI Work

- **SPI:**
    - Uses a single data line for both sending and receiving.
    - Achieves full-duplex communication with one clock line.
    - Suitable for low-speed communication.
- **Dual SPI (DPI):**
    - Employs two data lines for sending and receiving.
    - Doubles the data transfer rate compared to SPI.

    - Supports full-duplex communication with one clock line.
- **Quad SPI (QSPI):**
    - Utilizes four data lines for sending and receiving.
    - Quadruples the data transfer rate compared to SPI.
    - Supports full-duplex communication with one clock line.
    - Ideal for high-speed applications and large data transfers.

10. Design and Implementation

10.1 Controller Architecture

The QSPI Flash Controller is implemented with the following key components:

- **Parameters:**
    - **DATA_WIDTH:** Defines the width of the data bus.
    - **ADDRESS_WIDTH:** Defines the width of the address bus.
    - **COMMAND_WIDTH:** Defines the width of the command bus.
    - **MODE:** Selects the operational mode (SPI, QSPI, or DPI).
- **State Machine:** Manages the controller operation with states such as IDLE, COMMAND, ADDRESS, DATA, and DONE.
- **Registers:**
    - **data_reg, address_reg, command_reg:** Store input data, address, and command respectively.
    - **spi_io_out, qspi_io_out, dpi_io_out:** Store output data for SPI, QSPI, and DPI modes.
- **Tri-State Buffer Management:** Handles driving of IO lines based on selected mode and CPOL/CPHA settings.

10.2 State Machine and Operation

The state machine transitions through various states:

- **IDLE:** Waits for a start signal.
- **COMMAND:** Outputs the command based on the selected mode.
- **ADDRESS:** Outputs the address based on the selected mode.
- **DATA:** Outputs the data based on the selected mode.
- **DONE:** Indicates completion and returns to IDLE.

Mode-specific logic ensures correct data output to IO lines and appropriate controller behavior based on selected SPI mode and clock settings.

10.3 Clock Polarity and Phase Handling

The CPOL and CPHA inputs dictate SPI clock configuration. The controller adjusts its behavior to match selected clock settings:

- **Mode 0:** Sample on rising edge, shift on falling edge.
- **Mode 1:** Sample on falling edge, shift on rising edge.
- **Mode 2:** Sample on falling edge, shift on rising edge.
- **Mode 3:** Sample on rising edge, shift on falling edge.

11. Testing and Verification

To ensure the QSPI Flash Controller operates correctly, a detailed testbench was developed, which includes the following components:

- **Clock Generation:** A clock signal with a 10 ns period is generated to drive the controller and synchronize its operation.
- **Reset and Initialization:** The testbench applies a reset signal to initialize the controller and ensure it starts in a known state.
- **Mode Testing:** Various test cases are applied to verify the controller's functionality across different modes (SPI, QSPI, DPI) and with different CPOL and CPHA settings.
- **Verification:** The outputs of the controller are checked against expected values to confirm correct operation. This involves validating data transmission, command execution, and response handling to ensure compliance with the desired functionality.

In code Mode is Parametrize, so in whatever mode you want to run the Controller you have to uncomment that from module and test bench both.

```
//    parameter MODE = "QSPI",  // QSPI Mode will be active
    parameter MODE = "SPI",  // SPI Mode will be active
//    parameter MODE = "DPI",  // DPI Mode will be active
```

12. Results:

The testing of all the waveforms is done using ***dummy data*** to verify the concept and ensure that the communication is functioning correctly.

12.1 QSPI Controller Working Result:

In this configuration, all 4 data lines send and receive data.

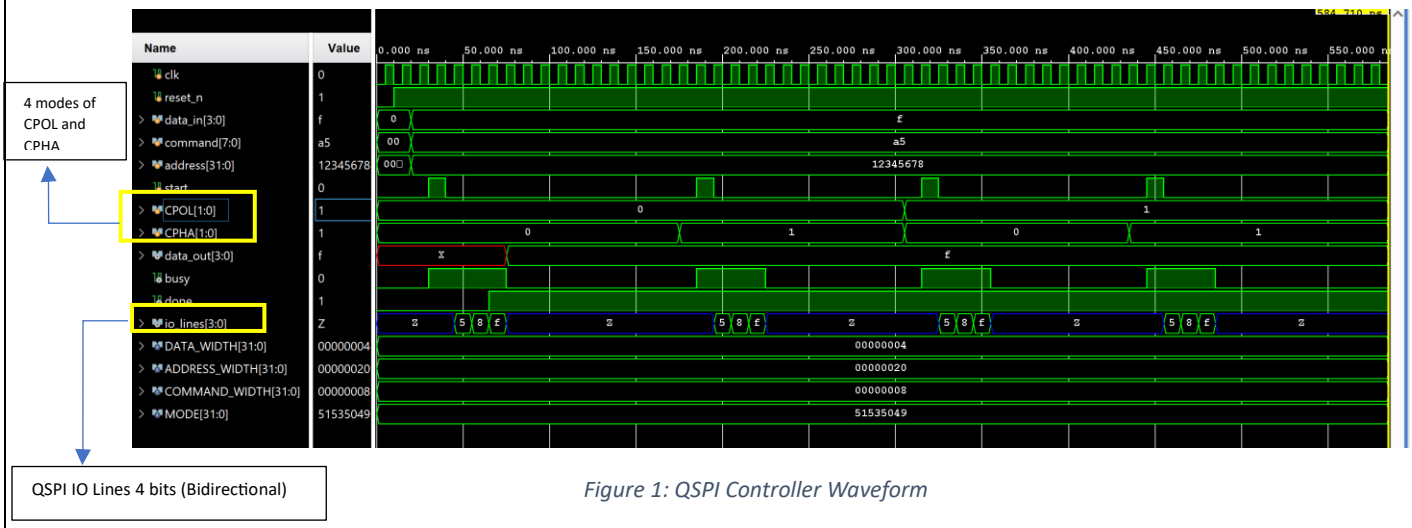

Figure 1: QSPI Controller Waveform

12.1.1 Waveform Analysis:

The provided waveform illustrates the operation of a Quad SPI (QSPI) flash controller. Here's a detailed breakdown of the signals and their behavior, focusing on various aspects including the clock polarity (CPOL) and clock phase (CPHA) settings.

Key Signals

1. **clk (Clock)**
    - **Description:** This is the primary clock signal for the QSPI controller, oscillating between 0 and 1 every 5 time units, representing a 100 MHz clock frequency.
    - **Behavior:** The clock toggles consistently, providing the timing reference for synchronous operations.
2. **reset_n (Active-Low Reset)**
    - **Description:** This active-low signal is used to reset the QSPI controller. It starts low (0) and transitions to high (1) after 10 time units.
    - **Behavior:** Initially, reset_n is low, putting the controller in a reset state. After 10 time units, reset_n goes high, deasserting the reset and allowing the controller to begin operation.
3. **data_in[3:0] (Input Data Bus)**
    - **Description:** This 4-bit input bus supplies data to the QSPI controller.

- **Behavior:** Starts at 0000 and changes to 1100 according to testbench stimuli.

4. **command[7:0] (Command Register)**
    - **Description:** An 8-bit register for storing the command to be executed by the QSPI controller.
    - **Behavior:** Begins at 00000000 and later changes to 10101010.
5. **address[31:0] (Address Register)**
    - **Description:** A 32-bit register holding the address for the memory operation.
    - **Behavior:** Starts with 00000000 and later updates to A5A5A5A5.
6. **start (Start Signal)**
    - **Description:** This control signal initiates the QSPI operation.
    - **Behavior:** Initially low, it goes high for one clock cycle to start the QSPI operation.
7. **busy (Busy Signal)**
    - **Description:** Indicates if the QSPI controller is currently processing a command.
    - **Behavior:** Transitions from low to high after the start signal is asserted, indicating the controller is busy.
8. **done (Done Signal)**
    - **Description:** Shows whether the QSPI operation is complete.
    - **Behavior:** Remains low during the busy state and transitions to high once the operation is finished.
9. **data_out[3:0] (Output Data Bus)**
    - **Description:** A 4-bit output bus that carries the data processed by the QSPI controller.
    - **Behavior:** Starts in an undefined state (X) and is expected to match data_in after the operation completes.
10. **qspi_io[3:0] (Quad SPI I/O Lines)**
    - **Description:** Represents the four data lines used for communication in QSPI.
    - **Behavior:** Initially in high impedance (Z), the lines later show defined values as per the command, address, and data phases.

Phases:

1. **Reset Phase:**
    - **reset_n** is low initially, indicating the QSPI controller is in a reset state. During this period, all outputs are in their initial states: busy is 0, done is 0, and qspi_io is Z (high impedance).

2. **Initialization:**
    - After 10 time units, **reset_n** transitions to high, deasserting the reset.
    - Inputs (command, address, data_in) are applied, and **start** is asserted to initiate the operation.
3. **QSPI Operation Phases:**
    - **Command Phase:** The **command** value (10101010) is loaded, and the lower nibble (1010) is driven onto **qspi_io**.
    - **Address Phase:** The **address** value (A5A5A5A5) is loaded, with each nibble driven onto **qspi_io** sequentially.
    - **Data Phase:** The **data_in** value (1100) is driven onto **qspi_io**.

During these phases, the **busy** signal is high, indicating active processing.

4. **Completion:**
    - After the data phase, the controller transitions to the DONE state. **done** goes high, and **data_out** updates to match **`data_in**.
    - **qspi_io** returns to high impedance (Z), marking the end of data transmission.

Analysis of CPOL and CPHA Settings

1. **CPOL = 0, CPHA = 0:**
    - **Clock Behavior:** The clock is idle low. Data is sampled on the rising edge.
    - **Data Transfer:** The qspi_io lines show data synchronized with the rising edge of the clock.
2. **CPOL = 0, CPHA = 1:**
    - **Clock Behavior:** The clock remains idle low. Data is sampled on the falling edge.
    - **Data Transfer:** Data on qspi_io is now aligned with the falling edge of the clock.
3. **CPOL = 1, CPHA = 0:**
    - **Clock Behavior:** The clock is idle high. Data is sampled on the falling edge.
    - **Data Transfer:** Data on qspi_io is synchronized with the falling edge of a high-idle clock.
4. **CPOL = 1, CPHA = 1:**
    - **Clock Behavior:** The clock is idle high. Data is sampled on the rising edge.
    - **Data Transfer:** Data on qspi_io aligns with the rising edge of the clock when idle is high.

The waveform demonstrates that the QSPI controller correctly processes commands, addresses, and data while respecting the configured CPOL and CPHA settings.

12.2 SPI Controller Working Result:

In this configuration, out of the 4 data lines, only I0 and I1 are active for both sending and receiving data. Specifically, I0 and I1 are used for data transfer on both the MOSI (Master Out Slave In) and MISO (Master In Slave Out) lines.

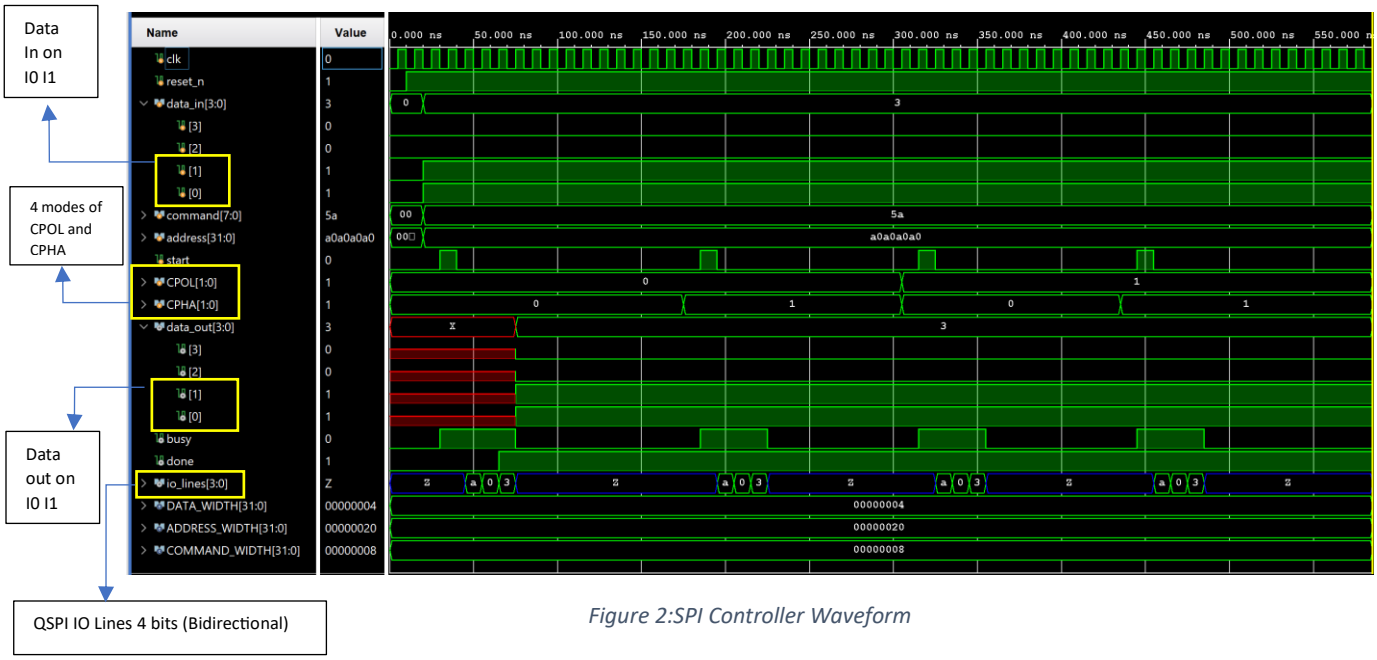

Figure 2:SPI Controller Waveform

12.2.1 Waveform Analysis:

1. **Clock (clk) Signal**

   - **Description:** The clk signal provides the timing for SPI communication with consistent clock pulses throughout the simulation period.
   - **Label:** "SPI Clock Signal - Provides timing for data transfer."

2. **Reset (reset_n) Signal**

   - **Description:** The reset_n signal is high (1) throughout the simulation, indicating the system is operational and not in a reset state.
   - **Label:** "System not in reset - Active mode."

3. **Data Input (data_in[3:0]) Signal**

   - **Description:** Shows the 4-bit input data with a value of 3 used for SPI communication.
   - **Label:** "Input Data - Represents the 4-bit input data for SPI communication."

4. **Command (command[7:0]) Signal**

   - **Description:** Displays an 8-bit command value (5a), typically used to configure or instruct the SPI device.

- **Label:** "SPI Command - 8-bit instruction for SPI device."

5. **Address (address[31:0]) Signal**

- **Description:** Shows a 32-bit address value (a0a0a0a0), indicating the memory or register address targeted for SPI communication.
- **Label:** "Target Address - 32-bit address for read/write operations."

6. **Start (start) Signal**

- **Description:** Transitions to high (1) around 150 ns, initiating the SPI data transfer.
- **Label:** "Start Signal - Initiates SPI data transfer."

7. **Clock Polarity (CPOL[1:0]) and Phase (CPHA[1:0]) Settings**

- **Description:** Define the SPI mode as follows:
    - **CPOL = 0, CPHA = 0:** Clock is idle low, data is sampled on the rising edge.
    - **CPOL = 0, CPHA = 1:** Clock is idle low, data is sampled on the falling edge.
    - **CPOL = 1, CPHA = 0:** Clock is idle high, data is sampled on the falling edge.
    - **CPOL = 1, CPHA = 1:** Clock is idle high, data is sampled on the rising edge.
- **In this waveform:** CPOL = 1 and CPHA = 1, meaning the clock is idle high, and data is sampled on the rising edge.
- **Label:** "SPI Mode Settings - CPOL = 1, CPHA = 1 (Clock idle high, data sampled on rising edge)."

8. **Data Output (data_out[3:0]) Signal**

- **Description:** Initially unknown (X), then outputs the value 3 once the transfer begins.
- **Label:** "Output Data - Data sent from the SPI master."

9. **Busy (busy) and Done (done) Signals**

- **Description:**
    - Busy signal goes high (1) after the start signal, indicating the SPI controller is actively transferring data.
    - Done signal goes high (1) towards the end of the waveform, indicating the completion of the SPI data transfer.
- **Label for Busy:** "SPI Busy - Data transfer in progress."
- **Label for Done:** "SPI Done - Data transfer completed."

10. **IO Lines (io_lines[3:0])**

- **Description:** Initially in a high-impedance state (Z), changes to active states (a, 0, 3) during data transfer, and returns to high impedance (Z) afterward.
- **Label:** "QSPI Data Lines I0 I1 - Transmit and receive data during SPI communication."

11. **Width and Mode Configuration Signals**
    - **Description:**
        - DATA_WIDTH[31:0] = 00000004 (4-bit data width)
        - ADDRESS_WIDTH[31:0] = 00000020 (32-bit address width)
        - COMMAND_WIDTH[31:0] = 00000008 (8-bit command width)
        - MODE[23:0] = 535049 (combined mode configuration)
    - **Label for Data Width:** "Data Width Configuration - Specifies the width of the data bus."
    - **Label for Address Width:** "Address Width Configuration - Specifies the width of the address bus."
    - **Label for Command Width:** "Command Width Configuration - Specifies the width of the command register."
    - **Label for Mode:** "Mode Configuration - Combined settings for operation modes."

12.3 DUAL SPI Controller Working Result:

In this configuration, out of the 4 data lines, I0 and I1 are active for both sending data and I2and I3 for receiving data.

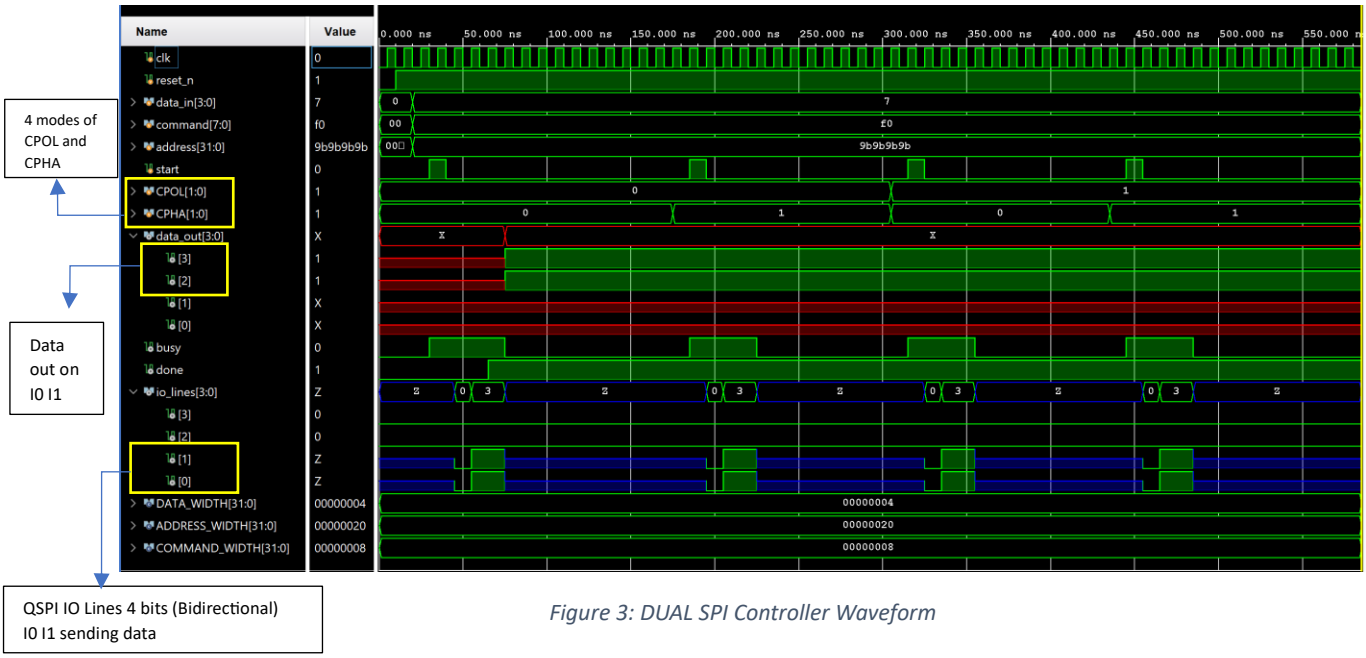

Figure 3: DUAL SPI Controller Waveform

12.3.1 Waveform Analysis:

The waveform illustrates a simulation of SPI (Serial Peripheral Interface) data transfer in dual mode, where two bits are transferred per clock cycle. The SPI protocol parameters, including clock polarity (CPOL) and clock phase (CPHA), are varied during the transfer to test different communication modes.

Explanation of the Waveform

1. **Clock Signal (clk)**
    - **Description:** Represents the clock used for SPI communication. It drives the timing of data transfer, toggling regularly throughout the simulation.
    - **Label:** "SPI Clock Signal - Provides timing for data transfer."
2. **Reset Signal (reset_n)**
    - **Description:** An active-low signal that resets the system when low (0). In the waveform, it is high (1), indicating that the system is operational.
    - **Label:** "System not in reset - Active mode."
3. **Data Input (data_in[3:0])**
    - **Description:** Represents the 4-bit data input into the SPI module, with a value of 7 (binary 0111). This value remains constant throughout the simulation.
    - **Label:** "Input Data - 4-bit data input for SPI communication."

4. **Command (command[7:0])**
    - **Description:** Displays the 8-bit command sent to the SPI module, which is F0 (binary 11110000).
    - **Label:** "SPI Command - 8-bit instruction for SPI device."
5. **Address (address[31:0])**
    - **Description:** Shows the 32-bit address (9B9B9B9B), representing the memory location for reading or writing data.
    - **Label:** "Target Address - 32-bit address for SPI operations."
6. **Start Signal (start)**
    - **Description:** Initiates the SPI communication by transitioning from 0 to 1. This triggers the SPI module to start processing the data.
    - **Label:** "Start Signal - Begins SPI data transfer."
7. **Clock Polarity (CPOL[1:0]) and Clock Phase (CPHA[1:0])**
    - **Description:** Control the clock settings:
        - **CPOL = 0, CPHA = 0:** Clock idle low, data sampled on rising edge.
        - **CPOL = 0, CPHA = 1:** Clock idle low, data sampled on falling edge.
        - **CPOL = 1, CPHA = 0:** Clock idle high, data sampled on falling edge.
        - **CPOL = 1, CPHA = 1:** Clock idle high, data sampled on rising edge.
    - **Label:** "SPI Mode Settings - CPOL and CPHA configurations affect data sampling."
8. **Data Output (data_out[3:0])**
    - **Description:** Initially undefined (X), then outputs the expected data based on CPOL and CPHA settings. The data becomes valid as the communication progresses.
    - **Label:** "Output Data - Data from the SPI master."
9. **Busy Signal (busy)**
    - **Description:** Indicates that the SPI module is engaged in data transfer. It goes high during the transfer process.
    - **Label:** "SPI Busy - Data transfer in progress."
10. **Done Signal (done)**
    - **Description:** Signifies the completion of the SPI transfer by going high at the end of the operation.

- **Label:** "SPI Done - Data transfer completed."

11. **I/O Lines (io_lines[3:0])**
    - **Description:** Represent the data lines used in dual-mode SPI communication, where two bits are transferred simultaneously per clock cycle. The lines switch between high-impedance (Z) and active states (a, 0, 3) as data is transferred.
    - **Label:** "QSPI Data Lines I0 I1 (used for input) I2 I3 (used for Output)- Data transfer during SPI communication."
12. **Width Parameters**
    - **DATA_WIDTH[31:0]:** Indicates a 4-bit data width.
    - **ADDRESS_WIDTH[31:0]:** Indicates a 32-bit address width.
    - **COMMAND_WIDTH[31:0]:** Indicates an 8-bit command width.
    - **Label for Data Width:** "Data Width Configuration - Width of the data bus."
    - **Label for Address Width:** "Address Width Configuration - Width of the address bus."
    - **Label for Command Width:** "Command Width Configuration - Width of the command register."

Additional Insights and Details

1. **Impact of CPOL and CPHA on Data Transmission**
    - **Clock Polarity (CPOL):** Determines the idle state of the clock. CPOL = 0 means the clock is low when idle; CPOL = 1 means the clock is high when idle. This affects the data sampling alignment.
    - **Clock Phase (CPHA):** Determines whether data is sampled on the leading or trailing edge of the clock. The combination of CPOL and CPHA defines four modes, affecting how data is captured.
2. **Dual SPI Mode Operation**
    - **Increased Data Throughput:** Dual mode transfers two bits per clock cycle, doubling the data rate compared to single-bit SPI.
    - **Timing Considerations:** Ensures that data setup and hold times are met due to the increased data per cycle.
3. **SPI Control Signals and Flow**
    - **Start Signal (start):** Triggers the data transfer process.
    - **Busy Signal (busy):** Shows when the SPI module is active.
    - **Done Signal (done):** Indicates the end of the data transfer.

4. **Undefined Data States (X)**
    - **Initial X States:** Common during the start of a transfer when data is not yet valid.
    - **Transitioning X States:** May occur during CPOL and CPHA transitions, reflecting reconfiguration of the SPI module.
5. **Error Checking and Signal Integrity**
    - **Signal Glitches:** Any unexpected glitches may indicate issues with signal integrity.
    - **Data Validity and Timing Analysis:** Ensures that data aligns correctly with clock edges for reliable transmission.
6. **Parameter Settings Impact**
    - **DATA_WIDTH, ADDRESS_WIDTH, COMMAND_WIDTH:** Proper configuration is essential to match system requirements for data, address, and command fields.

13. Verilog Concepts Used

The design of the QSPI Flash Controller utilizes several key Verilog concepts:

- **State Machines:** Used to manage the various operational states and transitions of the controller. This includes defining states such as IDLE, COMMAND, ADDRESS, DATA, and DONE, and specifying the transitions between these states based on input signals and current state.
- **Tri-State Buffers:** Employed to manage the driving of IO lines in different modes. Tri-state buffers are crucial for handling the high-speed data lines in SPI, Dual SPI, and Quad SPI modes.
- **Parameterization:** Allows for flexible configuration of the controller's data width, address width, and command width. Parameters make the design adaptable to different requirements and memory devices.
- **Always Blocks:** Used for sequential logic (state transitions) and combinational logic (next state determination). Always blocks ensure that the controller behaves as expected in response to changes in input signals and clock edges.

14. Conclusion

The QSPI Flash Controller project demonstrates a successful implementation of a versatile and configurable interface for flash memory devices. By leveraging the Quad SPI protocol, the controller achieves significant improvements in data throughput compared to traditional SPI and Dual SPI configurations. The controller supports multiple modes and clock settings, ensuring compatibility with a wide range of flash memory devices.

The design was thoroughly verified through extensive testing, which confirmed its functionality and reliability. The use of Verilog for FPGA design allowed for a robust implementation that handles various communication modes and configurations efficiently. The project highlights the importance of understanding both the SPI and QSPI protocols, as well as the practical aspects of FPGA design and simulation.

This report provides a comprehensive overview of the QSPI Flash Controller's design, implementation, and testing. It underscores the key concepts and hardware requirements necessary for effective QSPI communication and the successful application of these principles in a real-world scenario.