Verification of SPI

Subhadip Sen

6th March, 2025

Contents

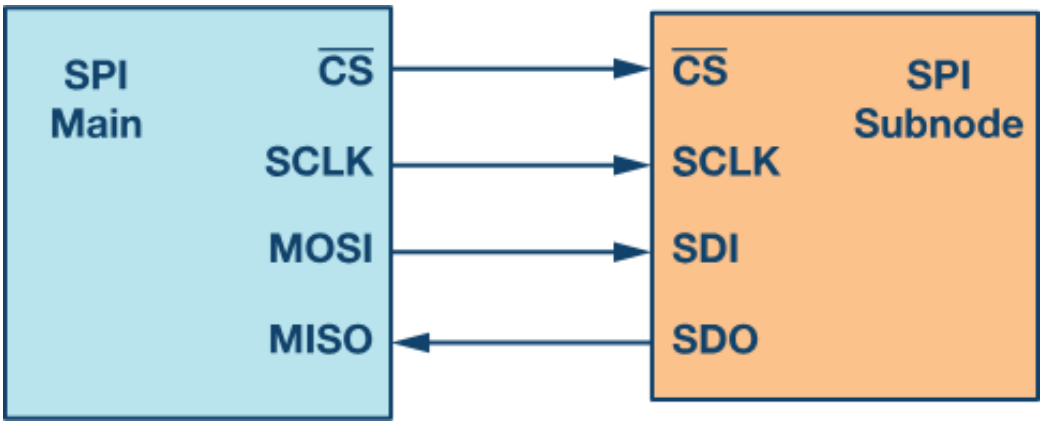
1. Introduction:

The Serial Peripheral Interface (**SPI**) is a **synchronous**, **full-duplex** communication protocol used for high-speed data transfer between a master and one or more slave devices. It operates using four main signals: **MOSI**, **MISO**, **SCLK**, and **CS**. SPI is widely used in embedded systems to interface microcontrollers with peripherals like sensors, memory, and displays due to its simplicity, speed, and flexibility.

- **MOSI (Master Out Slave In):** Carries data from the master to the slave.
- **MISO (Master In Slave Out):** Carries data from the slave to the master.
- **SCLK (Serial Clock):** Clock signal generated by the master to synchronize data transfer.
- **CS (Slave Select):** Active-low signal used by the master to select and enable a specific slave.

Key features:

- **Four-wire** interface (**MOSI**, **MISO**, **SCLK**, **CS**) → faster data rates, but higher pin count.
- **Single master, multiple slaves** supported (each slave needs a separate SS line).
- Master generates the clock and controls communication.
- No formal addressing; slaves selected via CS line.
- Supports **high-speed** data transfer (typically up to several Mbps or more).
- **Full-duplex** communication – simultaneous send and receive.
- Mode selection via clock polarity (**CPOL**) and phase (**CPHA**) → 4 SPI modes.
- **No built-in acknowledgment**; relies on application-level error handling.



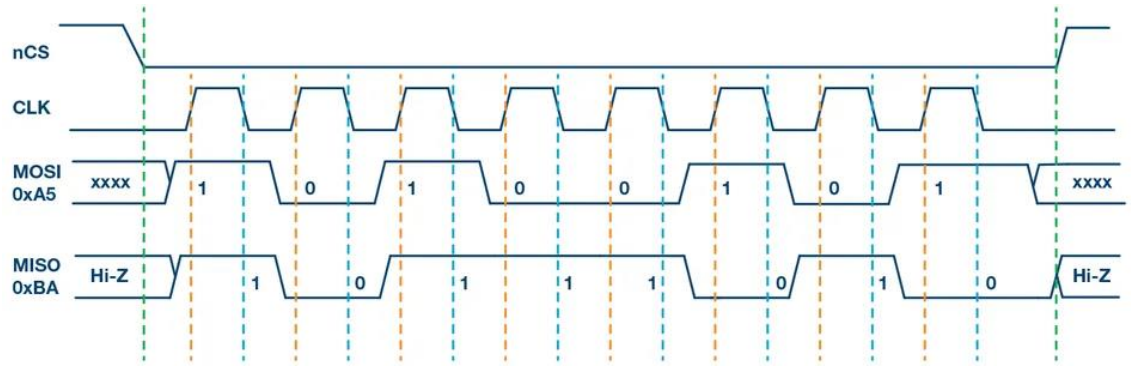


Fig 2: Frame Structure

2. Objective:

The objective of this project is to design and implement a **Serial Peripheral Interface (SPI)** communication protocol for reliable and efficient data exchange between a master device and slave devices. The SPI module will support.

- **8-bit addressing**
- **Single master, single slave support**
- **Standard mode of operations**
- **Configurable read/write operations**
- **Start, repeated start, stop condition handling**

The verification will be done using **System Verilog UVM methodology**, covering functional and corner cases with comprehensive test scenarios to ensure correctness, reliability, and protocol compliance.

3. Block Diagram:

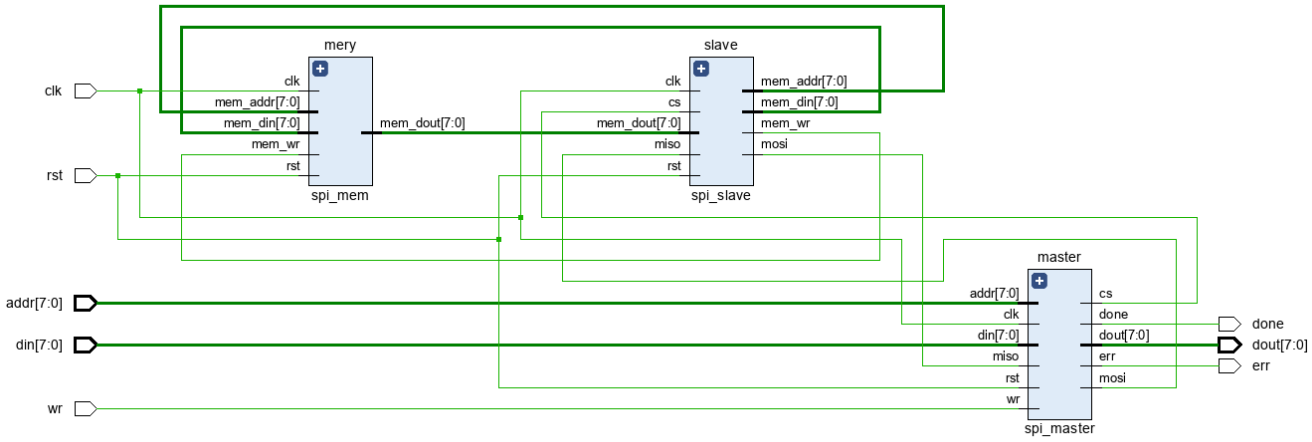


Fig 3: DUT block diagram

a. master:

Designed and implemented a custom SPI Master controller in Verilog as part of an SPI-based master-slave memory communication system. The SPI Master initiates and controls serial communication with a slave module to perform both read and write operations using the SPI protocol.

- **SPI Master – Write Operation**

  The SPI Master takes inputs: addr [7:0], din [7:0], wr = 1, clk, rst, and handles communication with the slave using control signals cs, mosi, and miso, it performs the following steps:

  - **Prepare Frame –** Combines {din, addr, 1'b1} into a 17-bit write frame.
  - **Start Communication –** Pulls cs low to activate the slave.
  - **Send Write Frame –** Transmits all 17 bits over mosi, LSB first.
  - **Complete –** Pulls cs high, sets done = 1, and returns to idle.

  FSM States: IDLE, LOAD, CHECK_OP, SEND_DATA, COMPLETE.

  The SPI Master ensures proper sequential bit-wise transfer to the slave for memory write operations.

- **SPI Master – Read Operation**

  For a read (wr = 0), the SPI Master initiates an 8-bit read address command to the slave and receives 8-bit data back, it performs the following steps:

  - **Prepare Read Frame –** Forms {addr, 1'b0} as an 8-bit read command.
  - **Send Address –** Pulls cs low and sends the 8-bit frame over mosi.
  - **Wait for Data –** After sending, releases cs to signal read request complete.
  - **Receive Data –** Reasserts cs, then clocks in 8 bits from slave via miso.
  - **Complete –** Stores received byte in dout, sets done = 1, and goes idle.

  FSM States: IDLE, LOAD, CHECK_OP, SEND_ADDR, READ_DATA, COMPLETE.

  This structured FSM enables full-duplex SPI communication while respecting slave timing for read operations.

b. slave:

Designed and implemented a custom SPI Slave module in Verilog as part of a master-slave memory communication system. The SPI Slave interfaces directly with internal memory and responds to read and write commands initiated by the SPI Master, following SPI protocol conventions.

❖ **SPI Slave – Write Operation**

The SPI Slave listens for incoming write frames from the Master and writes data into its internal memory, it performs the following steps:

- **Idle Detection** – Waits for cs = 0 and mosi = 1, indicating a write operation.
- **Receive Frame** – Shifts in a 17-bit frame {data [7:0], addr [7:0], wr=1} via miso, one bit per clock.
- **Store to Memory** – On frame completion, writes data into mem[addr].
- **Return to Idle** – Clears state and waits for the next command.

FSM States: IDLE, READ_DATA, COMPLETE.

Ensures valid writes only when wr = 1 and cs is asserted.

❖ **SPI Slave – Read Operation**

For read commands (wr = 0), the Slave receives an address and responds with the corresponding memory content, it performs the following steps:

- **Address Reception –** Detects cs = 0 and mosi = 0, then shifts in a 9-bit frame {addr [7:0], wr=0} from miso.
- **Fetch Data –** Uses received address to read from memory.
- **Send Data –** Outputs 8-bit memory data sequentially on mosi, one bit per clock.
- **Return to Idle –** Completes transfer and awaits the next request.

FSM States: IDLE, GET_ADDR, SEND_DATA, COMPLETE.

Implements proper synchronization and response timing to ensure accurate data delivery.

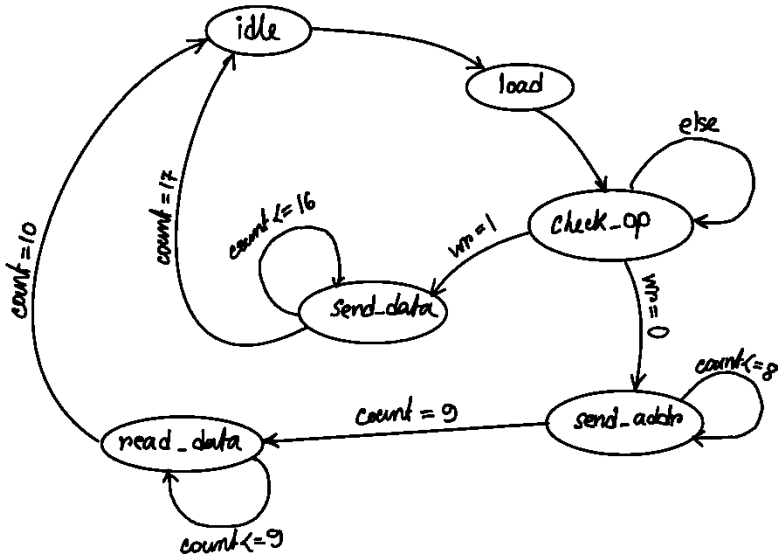
c. memory:

The memory module is a simple 256-byte register array designed to store and retrieve 8-bit data as directed by the I2C Slave Controller.

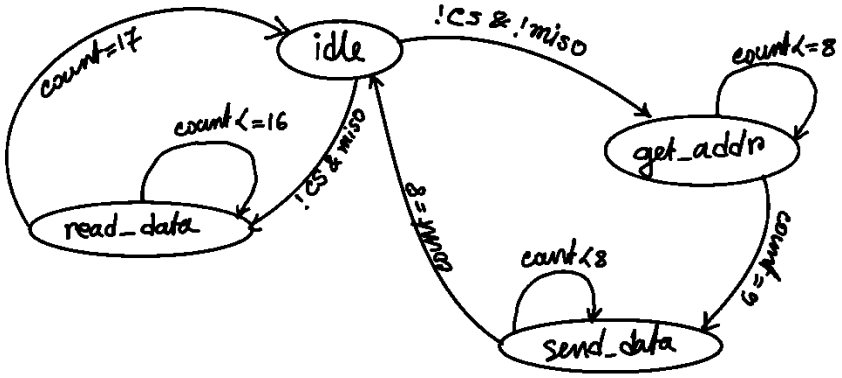
Key Features:

- **Size:** 256 × 8-bit memory array.
- **Read/Write Interface:**
  - **Write:** Accepts write_enable, address, and data_in signals from the slave to store data.
  - **Read:** Provides data_out based on the given address when read_enable is asserted.
- **Synchronous Operation:** Write operations are triggered on the rising edge of the clock. Read is done through continuous assignment.

4. FSM of Master:



FSM of Slave:



5. Testbench:

The testbench verifies the functionality of the SPI Master-Slave system, including reset, write, and read sequences. It stimulates the DUT (Device Under Test) with realistic SPI transactions and checks correctness using random input patterns.

Key Features:

- **Reset Sequence**:
  Initializes all DUT signals and waits for a few clock cycles before starting transactions.
- **Write Sequence**:
  - Generates a **random 8-bit address** and **8-bit data**.
  - Sets wr = 1 and triggers a write transaction via the **SPI** master.
  - Waits for done to be asserted, indicating transaction completion.
- **Read Sequence**:
  - Uses the same address to initiate a read (wr = 0) from the slave.
  - Waits for done, then compares datard (data read) with expected data.
- **Randomization**:
  Uses $random to generate varied address and data values for robustness.

Monitor captures the send address and data during write and read operation, broadcast the packet through analysis port. **Scoreboard** has reference memory which also store the data during write operation and compares the data with DUT during read operation, reports pass or fail status.

Fig 4: Simulator Output

Simulation results:

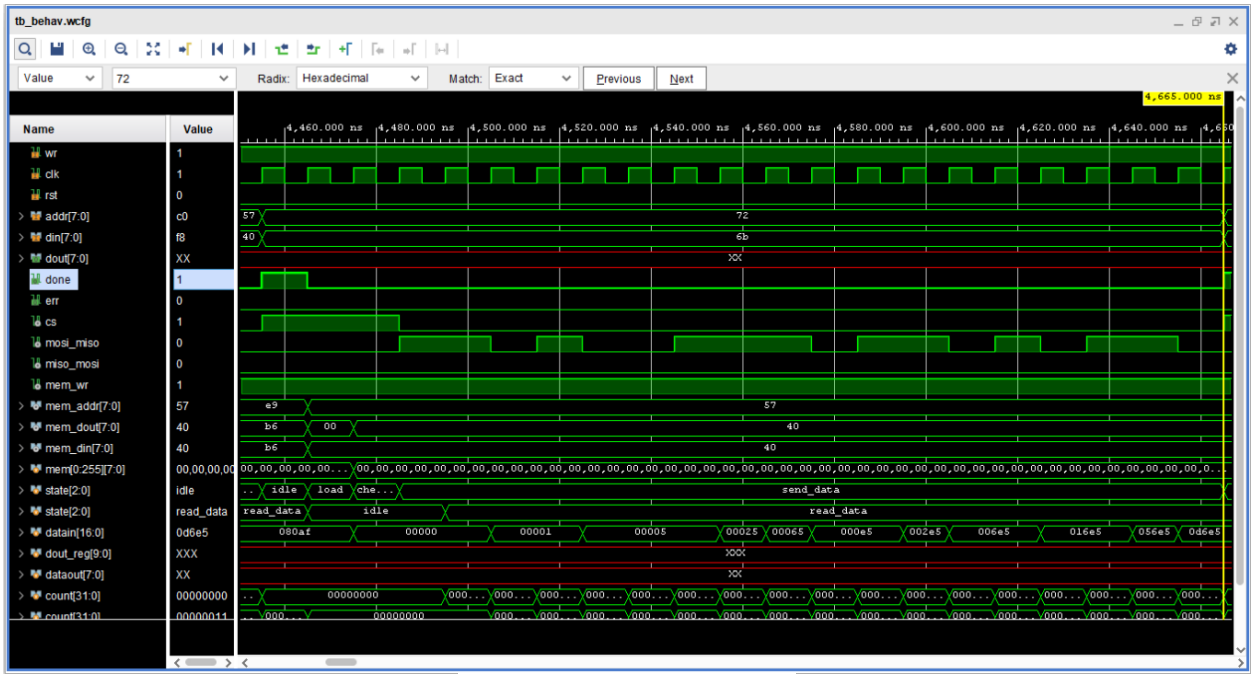


Fig 5: Single write to memory

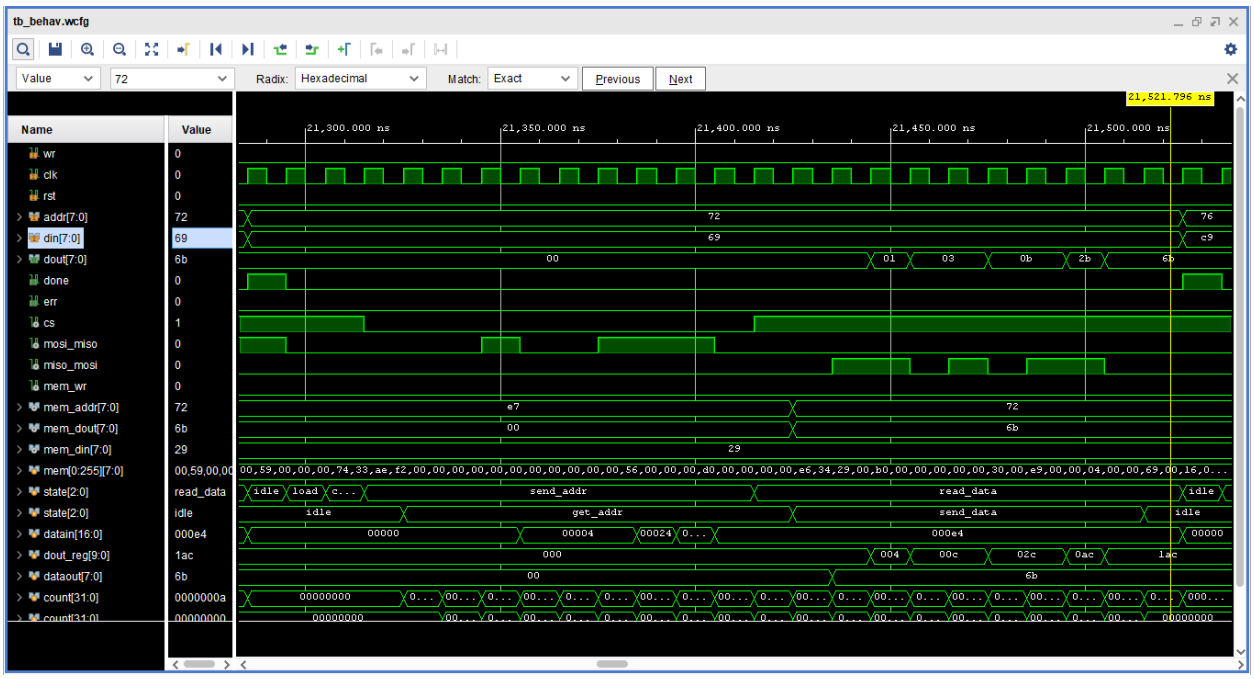


Fig 6: Single read from memory

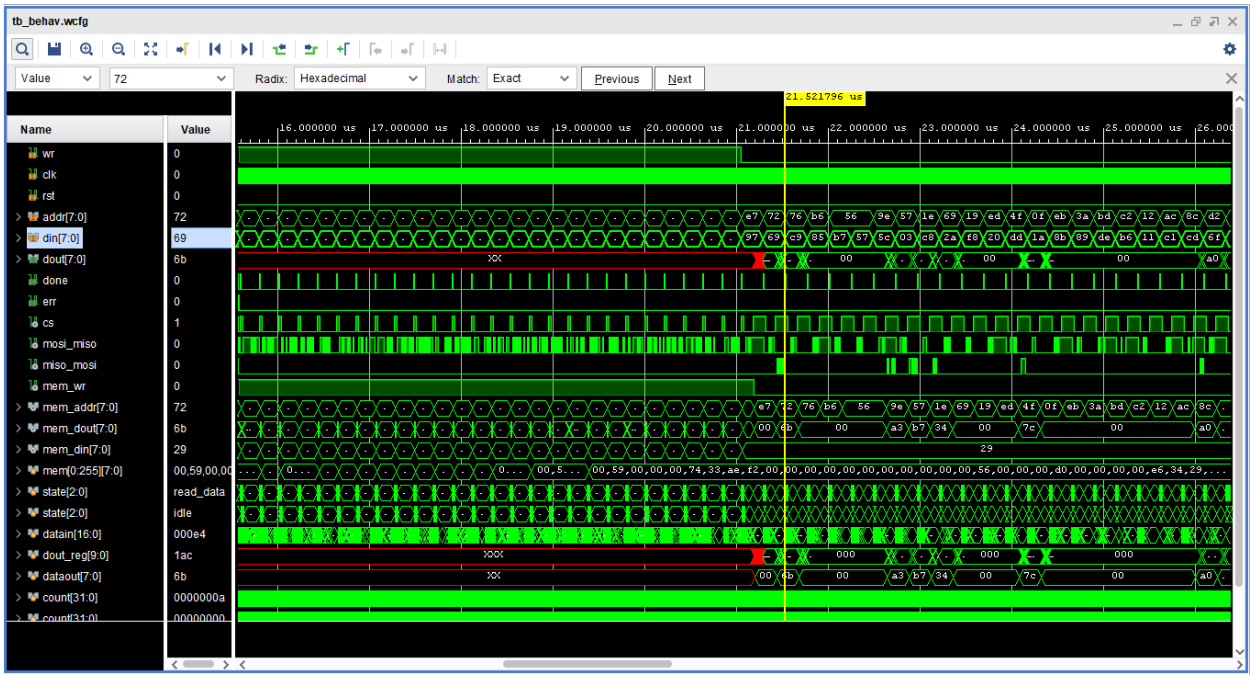


Fig 7: All sequence simulation result

It can be seen that all sequences are passing, read and write data matching for all memory locations.

- System verilog for Design & UVM verification environment written & simulated in Vivado.