EE214
VHDL Implementation and Simulation of SPI Master-Slave Communication

Varun Mishra (23b3985)
Tanisha Hase (23b3984)

Project Overview

This project focuses on the development and simulation of SPI communication between an SPI Master and an SPI Slave using VHDL. The SPI Master initiates the communication, sending data '5' over the MOSI line, while the SPI Slave responds by sending data '7' over the MISO line. The communication is synchronized using a 10 MHz clock and follows the Mode 0 configuration.

1. SPI Master

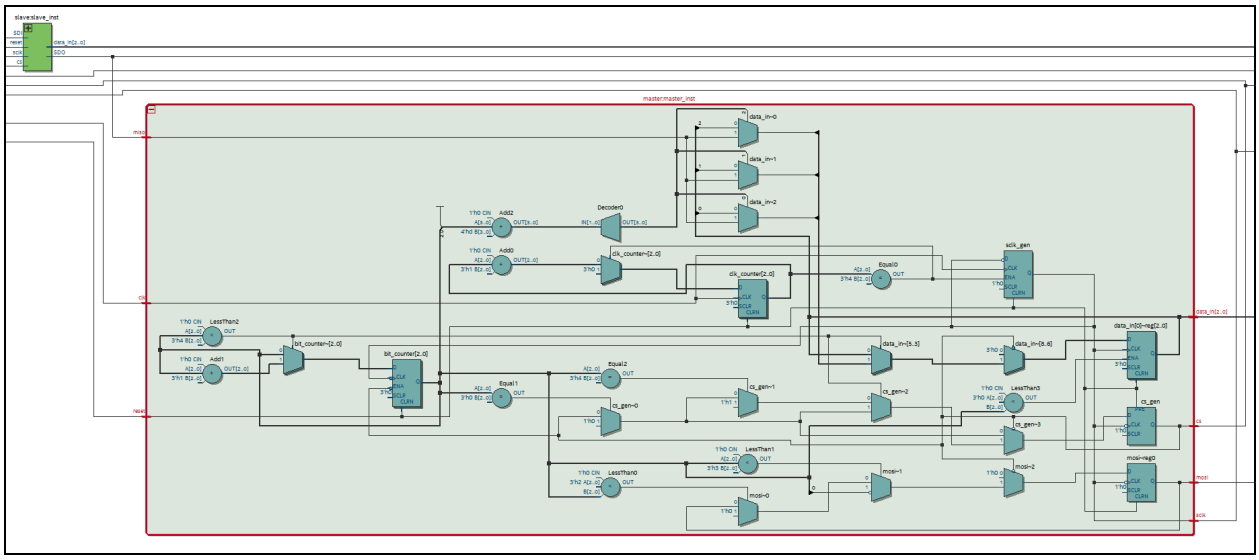
The SPI Master initiates the communication, generates the clock signal, and controls the **CS** signal. The following considerations guided the code development:

- **Clock Generation**: The Master generates the **SCLK** signal, and the communication is based on its rising and falling edges.
- **Data Sending and Receiving**: On every clock cycle, the Master sends its data over **MOSI** while simultaneously receiving data from the Slave over **MISO**.
- **Chip Select Management**: The **CS** signal is pulled low to initiate communication and pulled high to terminate it.
- **Mode 0 Configuration**: In Mode 0, data is transmitted on the falling edge and received on the rising edge of **SCLK** respectively.

Key Points in Code:

- **MOSI** sends the Master's data value (5) to the Slave.
- The Master receives the Slave's data over **MISO** in sync with the clock.
- **CS** ensures the communication is controlled and active only during the appropriate time.

Following is the netlist viewer of the SPI master :



2. SPI Slave

The SPI Slave is responsible for receiving data from the Master over the **SDI** line and simultaneously sending its own data over **SDO**. The communication is synchronized with the clock signal **SCLK**, and the **CS** signal activates the Slave. The following steps were taken while writing the code:

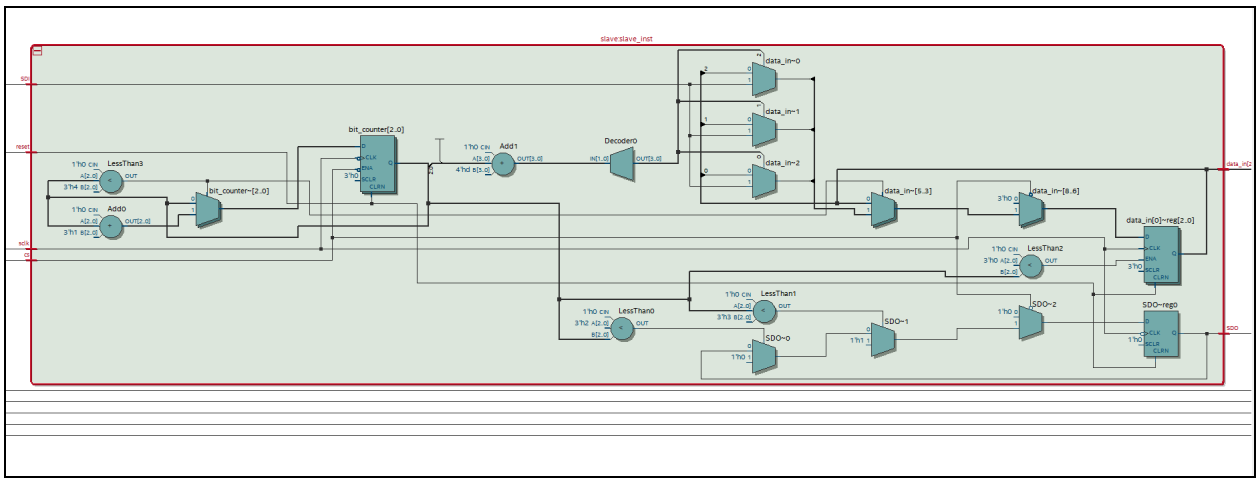
- **Synchronization**: Since SPI is clock-synchronized, the transmission process was triggered on the falling edge and the reception process on the rising edge of **SCLK** respectively. The Slave samples the incoming data from **SDI** while shifting its own data onto **SDO**.

- **Chip Select (CS)**: The **CS** signal is active-low. This means communication only occurs when **CS** is low, and when **CS** is high, the Slave stops communication and resets its internal counters.
- **Data Transmission**: To manage 3-bit communication, a **bit_counter** was used. This counter ensures the correct number of bits are shifted in and out.

Key Points in Code:

- **bit_counter** increments on each clock pulse during active communication.
- **SDI** data is sampled and stored into an internal register, **data_slavein**, while **datatransmitted** is shifted out to the Master over **SDO**.

Following is the netlist viewer of the SPI slave :



3. Top-Level Entity (DUT)

The top-level entity integrates both the SPI Master and Slave modules, connecting the SCLK, MOSI, MISO, and CS signals to simulate the communication. It was essential to ensure:

- **Signal Connections**: Proper signal routing was done between the Master and Slave modules.
- **Clock Sharing**: Both Master and Slave share the same clock signal for proper synchronization.

- **Data Handling**: Data outputs from both the Master and Slave were connected for observation.

Key Points in Code:

- Signals like **MOSI**, **MISO**, **SCLK**, and **CS** are interconnected between the Master and Slave.
- The Master's and Slave's output data was displayed for verification in simulations.

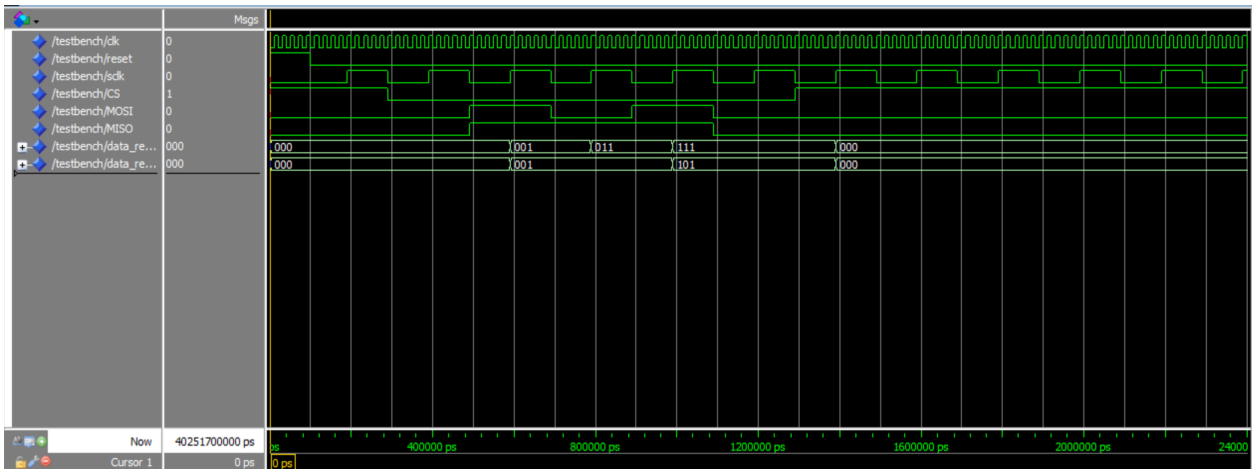
4. Testbench

The testbench simulates the SPI communication by generating the necessary clock and control signals. Key design points included:

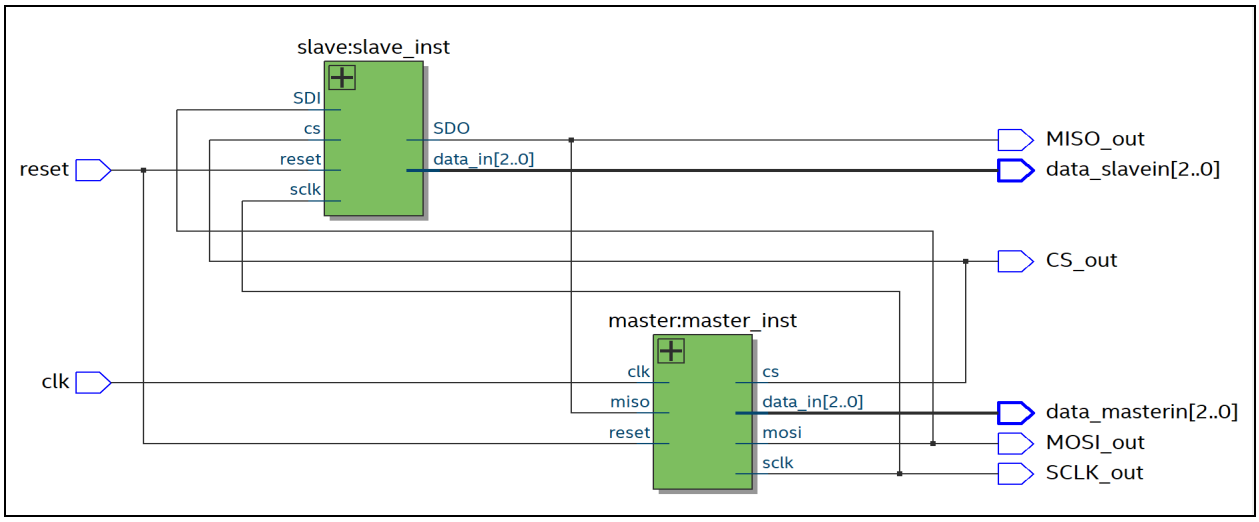
- **Clock Generation**: A 50 MHz clock was created which was then converted into a 10MHz clock which was called **SCLK** in the SPI Master.
- **CS Signal Control**: The **CS** signal was toggled to start and stop communication.
- **Data Verification**: The testbench monitors data sent over **MOSI** and **MISO** to verify that the Master and Slave exchanged the correct values.

Waveform

The simulation in Quartus Prime verified the correct operation of the SPI Master-Slave communication. The waveform demonstrates the exchange of data between the Master and Slave, with data transmitted on the **MOSI** and **MISO** lines in sync with the **SCLK** signal.



Block Diagram



Work Breakdown

- **Varun**: Implemented the SPI Master, created the top-level entity, and developed the testbench for verifying communication between the Master and Slave.
- **Tanisha**: Implemented the SPI Slave and prepared the report, including the block diagram and simulation screenshots.

Conclusion

The SPI Master and Slave modules were designed, simulated, and verified successfully in Quartus. The communication process was validated by observing the correct data transmission and reception, synchronized with the clock signal, ensuring a fully functional SPI system.