Design and Verification of Single Master Single Slave SPI

Parnika Sisodia1; Poulami Paul2; Pragya Gupta3; Prerna Mittal4; Pujithapydi5; Sakshi Dattatraya Jawanjal6

*1Women in Tech, NXP Mentorship Program*

*1NXP Semiconductors, Noida, India*

[1parnika.sisodia@gmail.com](mailto:1parnika.sisodia@gmail.com), [2pp2528@cse.jgec.ac.in](mailto:2pp2528@cse.jgec.ac.in), [32021eeb1192@iitrpr.ac.in](mailto:32021eeb1192@iitrpr.ac.in), [4prernaharshi@hotmail.com](mailto:4prernaharshi@hotmail.com), [5poojithapydi237@gmail.com](mailto:5poojithapydi237@gmail.com), [6jawanjalsakshi345@gmail.com](mailto:6jawanjalsakshi345@gmail.com)

***Abstract -- This report includes an overview of the Verilog code for both an SPI master and an SPI slave module and talks about the Serial Peripheral Interface (SPI) communication protocol. The synchronous serial communication protocol SPI is frequently used to transfer data between microcontrollers and peripheral devices. This report's Verilog code describes how an SPI master and slave for FPGA- and ASIC-based devices operate.***

1. INTRODUCTION

A full-duplex, synchronous serial communication protocol called SPI (Serial Peripheral Interface) is frequently used in embedded systems to transfer data between microcontrollers and various peripheral devices. It uses four essential signal lines to function:

* **SCK (Serial Clock):** The clock signal generated by the master device, which dictates the timing of data transmission.
* **MISO (Master In, Slave Out):** The data line on which the slave device sends data to the master.
* **MOSI (Master Out, Slave In):** The data line on which the master sends data to the slave.
* **SS (Slave Select):** An optional line used by the master to select a specific slave device.

SPI is characterized by its simplicity and efficiency, making it suitable for various applications, such as sensors, display controllers, and memory devices.

This report describes how to implement both an SPI master and an SPI slave module. While the slave reacts to incoming input, the SPI master starts data transfers.

1. SPI PROTOCOL
2. *SPI Master Module*

The following are the main parts of the SPI master module:

* **State Machine:** In order to function, the SPI master moves through three states: IDLE, WAIT\_HALF, and TRANSFER. To ensure effective data transmission, these states regulate the timing of the clock and data streams.
* **Clock Division:** To manage the data rate, the clock division factor is specified using the **‘CLK\_DIV’** parameter.
* **Data Transmission:** When the **‘start’** signal is asserted, data transmission starts. While concurrently receiving 8 bits of data from the slave device, the SPI master shifts out 8 bits of data.
* **Data Output and Flags:** The module offers a **‘data\_out’** data output, a busy flag (**‘busy’**), and a new\_data flag (**‘new\_data’**) that indicates that new data is available. These flags represent the SPI's current condition.

1. *SPI Slave Module*

The following characteristics are included in the SPI slave module:

* **State Machine:** The SPI slave runs on a condensed version of a state machine. Data transmission is timed to the rising and falling edges of the **‘sck’** signal after waiting for the **‘ss’** signal to become low (signaling the master's selection).
* **Data Transmission:** The slave shifts in 8 bits of data. The slave acknowledges the completion of the master's 8-bit transmission with the **‘done’** flag and gets ready to receive the next byte of data.
* **Slave Select:** Multiple slave devices can be connected to a single master by using the **‘ss’** signal to select the slave device.

1. SPI TRANSMISSION IN THE VERILOG CODE
2. *SPI Master*

* **Idle State:** The SPI master starts in the IDLE state, with the clock counter and bit counter reset. If the **‘start’** signal is asserted, the input data is loaded, and the state transitions to WAIT\_HALF.
* **Wait Half State:** In this state, the clock counter increments, and when half of the clock period elapses, it resets the counter and transitions to TRANSFER.
* **Transfer Status:** The clock counter is still running while data is being transmitted. The most important bit of the data is broadcast on the rising edge of the clock, while the slave transmits the least important bit to the master on the falling edge. When all 8 bits have been sent, the state changes back to IDLE and new data becomes available.

1. SPI Slave

* **State Machine:** The SPI slave awaits the assertion of the selection signal, or **‘ss;** signal, by the master. After that, it synchronizes data transfer with the **‘sck’** signal's rising and falling edges.
* **Data Transmission:** While the master sends, the slave receives data. The slave saves the received data, sets the **‘done’** flag to signal completion, and gets ready to accept the next byte when all 8 bits have been received.
* **Slave Select:** By permitting several slaves to be linked to a single master, the **‘ss’** signal selects the slave device.

1. SYSTEM VERILOG VERIFICATION

Verification is a critical step in the design and development process, and the purpose of these components is to collectively ensure that the design operates correctly and meets its specifications. The generator and driver generate and apply stimulus, the DUT represents the design, the scoreboard checks correctness, the monitor captures data, and coverage analysis ensures thorough testing. This combination of components helps in achieving a high level of confidence in the correctness and reliability of the design.

Verification Importance:

* **Correctness Assurance:** Verification ensures that the design operates as intended and meets the specified requirements.
* **Bugs Identification:** It helps in identifying and fixing bugs or issues in the design before fabrication or production.
* **Cost Reduction:** Finding and addressing design flaws early in the development process reduces the cost of fixing issues in later stages.
* **Quality Assurance**: Verification enhances the overall quality and reliability of the final product.

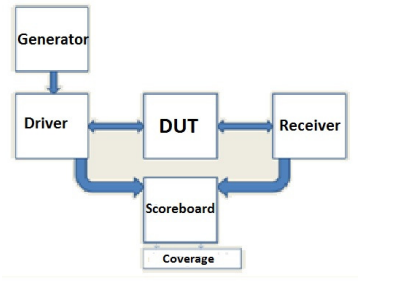


Fig.3. Verification Environment

A generator starts by creating a transaction, randomizing it, and sending it to the driver's mailbox. The class known as the driver creates the packets, drives them to the DUT input interface, and inserts them into the mailbox. The receiver gathers the bytes of data from the signal of interface. after which it pushes the packet into the mailbox after unpacking the bytes. In the scoreboard are two mailboxes. One is employed to obtain the packets from the receiver and the driver, respectively. Next, the packets are compared; in the event that they diverge, error is then claimed.

The tests have been conducted to ensure functional coverage of design features. The method of calculating and applying functional covering involves a number of steps. First, update the testbench with code to track the stimulus that enters the device, its response, and response, ascertain which functionality has been used, and, following simulation, examine the coverage outcomes to identify determine which test scenarios need to be exercised and create tests to do so.

1. IMPLEMENTATION AND SIMULATION RESULTS

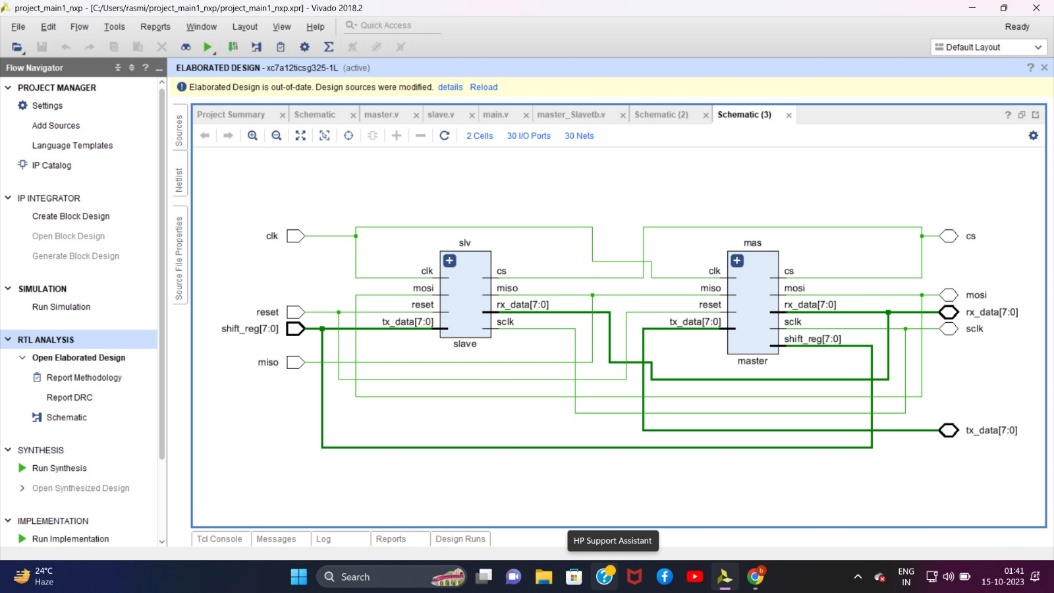
****

Fig.4. RTL Schematic of Top module showing blocks of individual master and slave

**Simulation Results (in Verilog):**

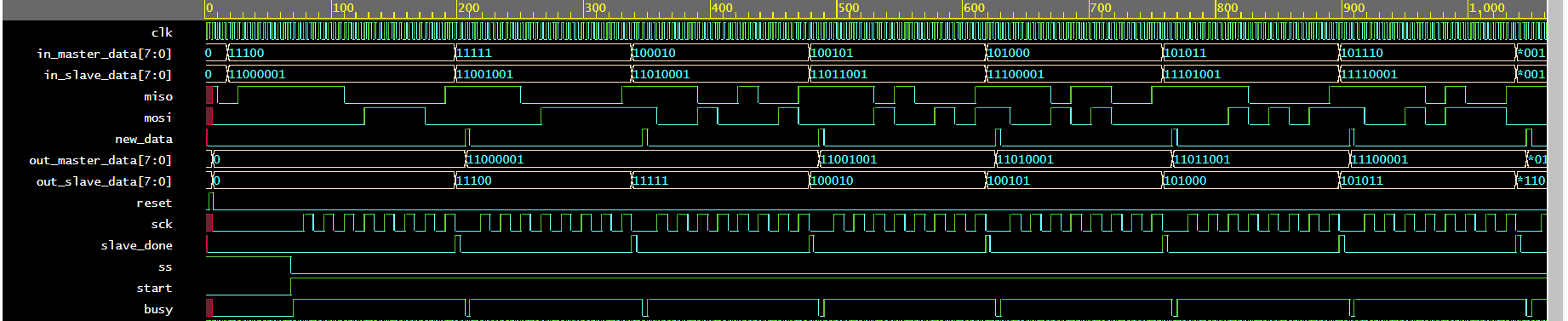


Fig.4. Overall Output waveform showing Full Duplex operation of the top module.

1. CONCLUSION

One popular standard for data exchange between microcontrollers and peripheral devices is the SPI communication protocol. The SPI master and slave modules' functioning is demonstrated by the Verilog code that is supplied with them, enabling smooth data transfer between the master and slave devices. Engineers and designers can integrate SPI communication into their FPGA and ASIC-based designs with efficiency if they comprehend the protocol's operation and the matching Verilog implementation.

1. REFERENCES

[1] <https://www.ijedr.org/papers/IJEDR1303026.pdf>