Institute Of Technology, Nirma University

Department of Electronics and Communication Engineering

2EC202-FPGA-based System Design

22BEC030 [Semester-IV]

Topic: SPI Master

Abstract: SPI (Serial Peripheral Interface) is a commonly accepted 4-wire standard configuration for synchronous serial communication used primarily for short-distance wired communication between a microcontroller and peripheral ICs. For electronic devices to communicate effectively, they require a common means of exchange, similar to how humans need a shared language for successful communication. This communication protocol is a master-slave relationship where the master is the controlling device, and the slave (sensor, display, or memory chip) takes instruction from the master. SPI uses four wires for communication: MOSI, MISO, SCLK, and SS/CS. The master outputs the clock signal determining the speed of data transfer. The data is transferred bit by bit serially via the MOSI and MISO lines. This project focuses on the implementation of an SPI Master using the simplest configuration of SPI which is a single master, single slave system, on a Field-Programmable Gate Array (FPGA). SPI is a widely used communication protocol in various electronic devices, including SD card reader modules, RFID card reader modules, and 2.4 GHz wireless transmitters/receivers.

Keywords: SPI Master, MISO, MOSI, Slave select, Full-duplex serial, FPGA

Literature Survey: SPI is a full-duplex synchronous serial communication, allowing data to be simultaneously transmitted from both directions. It was first developed by Motorola in the mid-1980s for inter-chip communication.

- SPI is a serial communication protocol used for low-speed devices, commonly used for communication with flash memory, sensors, real-time clock, and analog-to-digital converters.
- SPI has evolved from its original use for communication within single integrated circuits to support communication between various microcontrollers, sensors, displays, memory devices, and other peripherals.
- SPI communication involves a master device and one or more slave devices connected through four essential lines: MOSI, MISO, SCK, and SS/CS.
- **MOSI (Master Output/Slave Input)** – Line for the master to send data to the slave.
- **MISO (Master Input/Slave Output)** – Line for the slave to send data to the master.
- **SCLK (Clock)** – Line for the clock signal.
- **SS/CS (Slave Select/Chip Select)** – Line for the master to select which slave to send data to.

- There are four SPI modes (0, 1, 2, and 3) that differ in clock polarity and phase configurations, enabling precise data transfer.
- Researchers have explored various clocking schemes and techniques to enhance the performance of SPI communication, including increasing data rates, minimizing latency, and reducing power consumption.
- The main advantage of SPI is the ability to transfer data without interruption.
- Functional verification of SPI protocol implementation on FPGA can be done through functional simulation using various EDA tools.

Methodology/Proposed solution:

Here is an algorithm based on the flow of the Verilog code for the SPI master:

1. Initialize the i and j registers to 0.
2. Initialize the ss signal to 1, deselecting the slave.
3. On the rising edge of the sclk signal, check if the ss_initiate signal is 1, i is less than 8, and j is less than 8.
4. If all conditions are met, select the slave by setting ss to 0.
5. Shift out the data bit by bit by setting mosi to the value of data_in[i] and storing the received data from miso in data_out[j].
6. Increment the i and j registers by 1.
7. If any of the conditions in step 3 are not met, deselect the slave by setting ss to 1.
8. Set mosi and data_out to high impedance by assigning them to 1'bZ.
9. Repeat steps 3-8 for each clock cycle until the transmission is complete.

This algorithm describes the flow of the Verilog code for the SPI master, which selects the slave, shifts out data bit by bit, and stores the received data in a register. The transmission is complete when all data has been transmitted and received, at which point the slave is deselected and the mosi and data_out signals are set to high impedance.

Limitations:

Number of wires: SPI requires four wires for communication, which can be a limitation in applications where the number of available wires is restricted. In contrast, I2C and UART use only two wires for communication.

Data Acknowledgement: SPI does not provide an acknowledgment mechanism to confirm that the data has been successfully received by the slave device. This can lead to data transmission errors and make it difficult to troubleshoot communication issues. I2C, on the other hand, includes an acknowledgment mechanism to ensure successful data transmission.

Error Checking: SPI lacks a built-in error-checking mechanism, such as the parity bit in UART. This can result in data corruption during transmission, especially in noisy environments or over long distances.

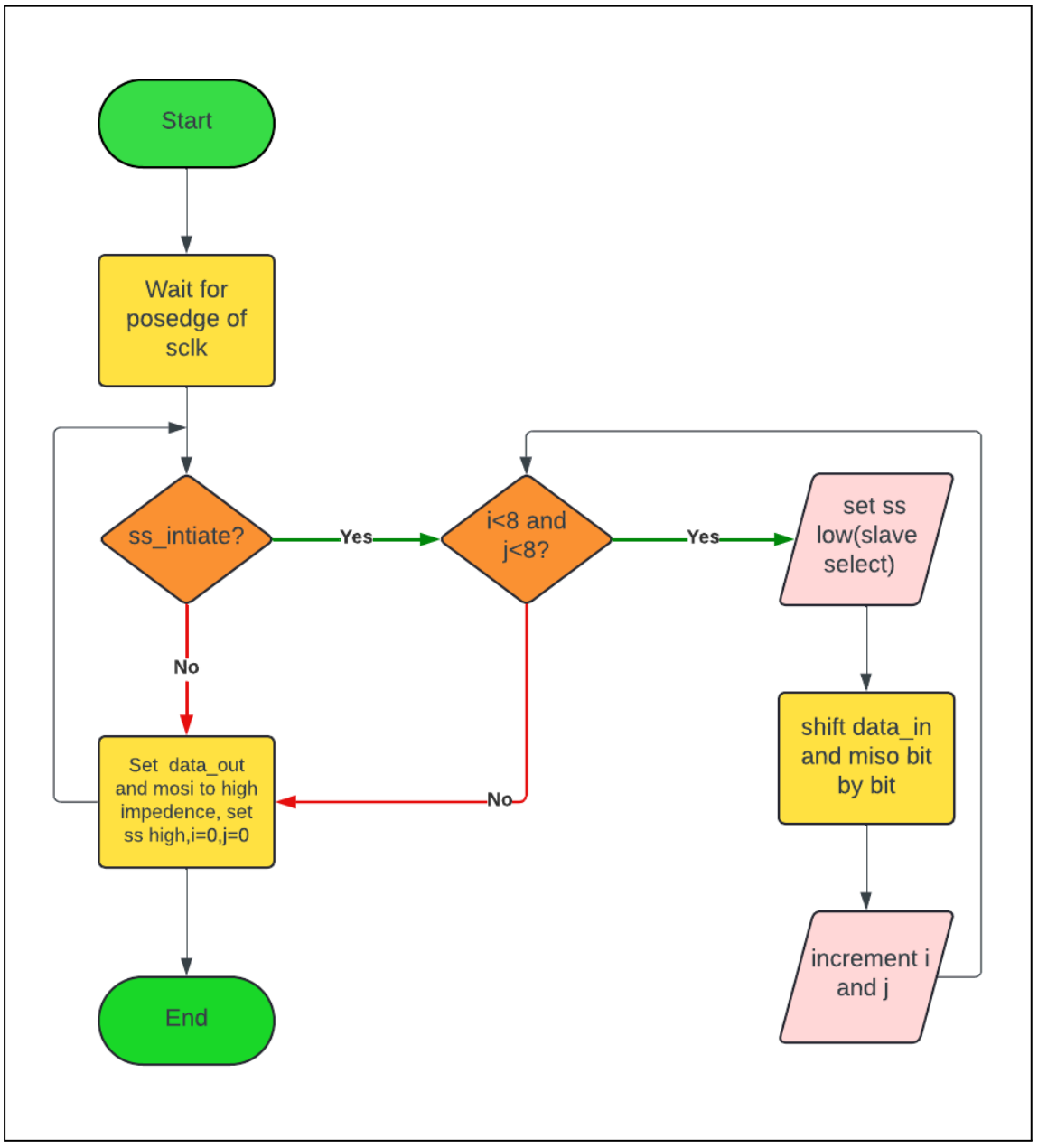
Number of Masters: SPI only supports a single master configuration, which can be a limitation in applications where multiple devices need to communicate with each other. In contrast, I2C supports both multi-master and multi-slave configurations.

Distance: SPI Master can only be used for short-distance communication with slave devices.

Data Security: SPI Master doesn’t offer a data security feature.

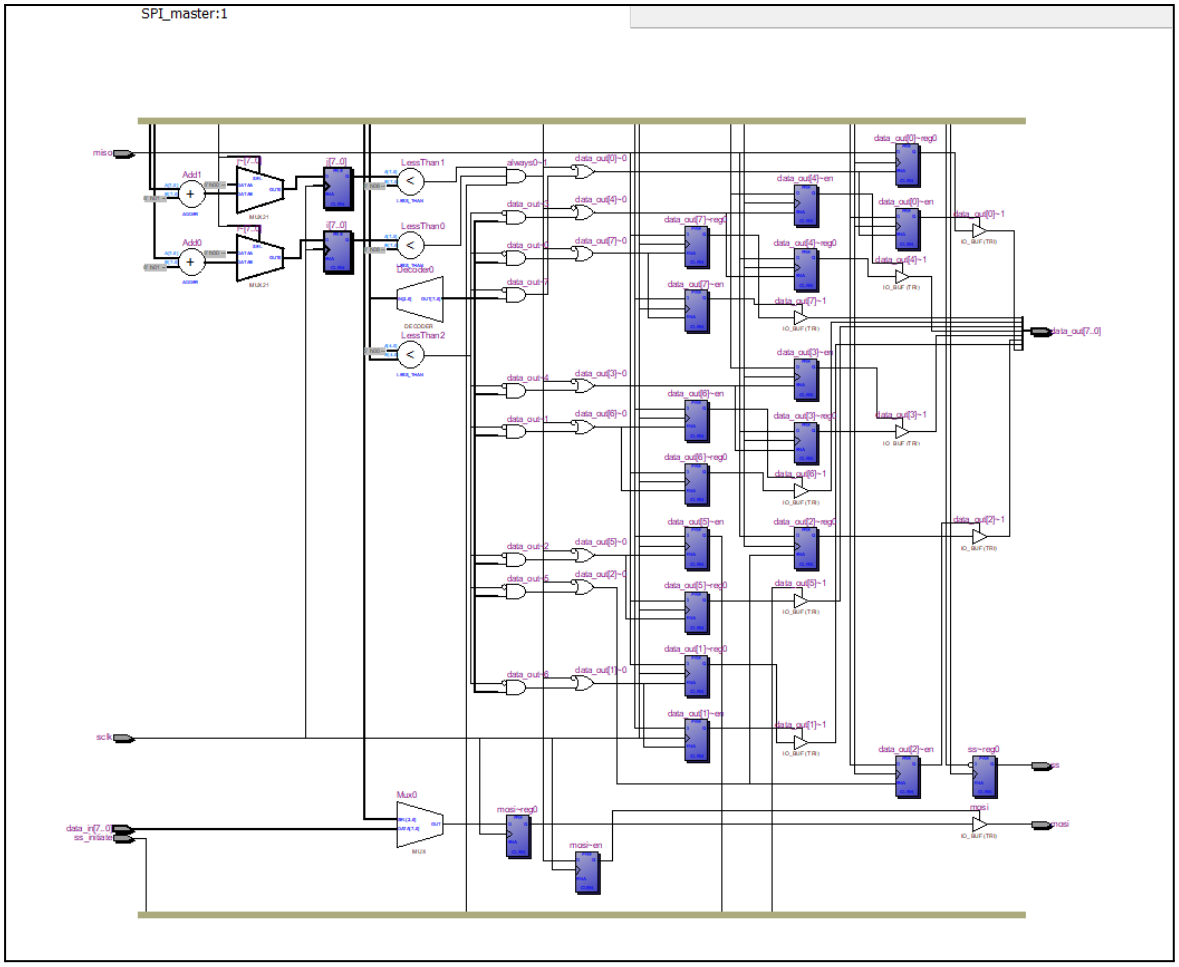
SPI communication is simple and fast, but it lacks some of the advanced features and flexibility of other communication protocols like I2C and UART.

Flowchart of the code :

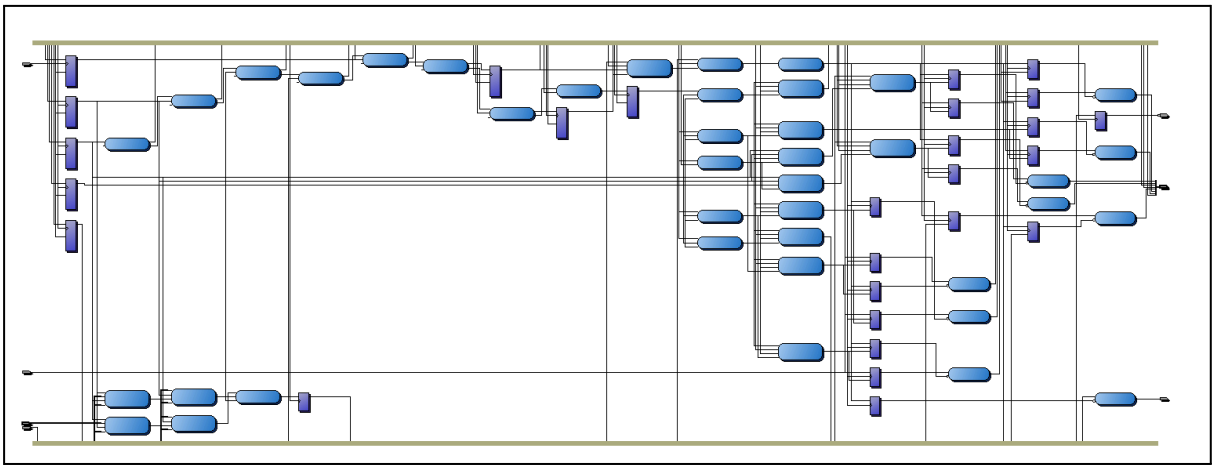


Results in terms of RTL, TTL, and waveform:

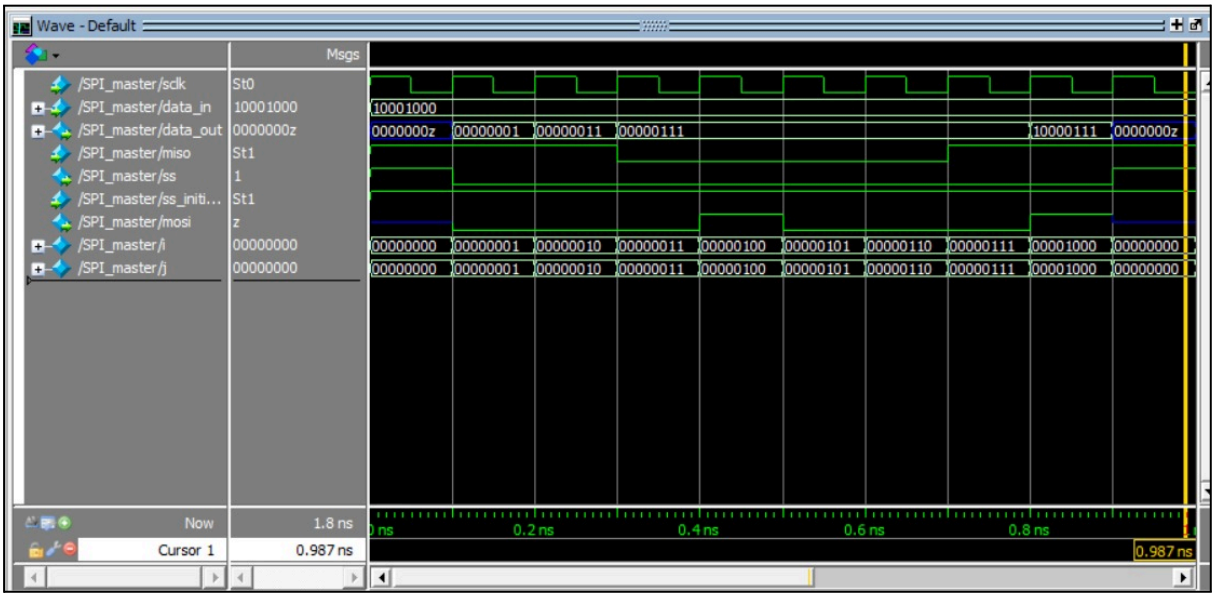
RTL:

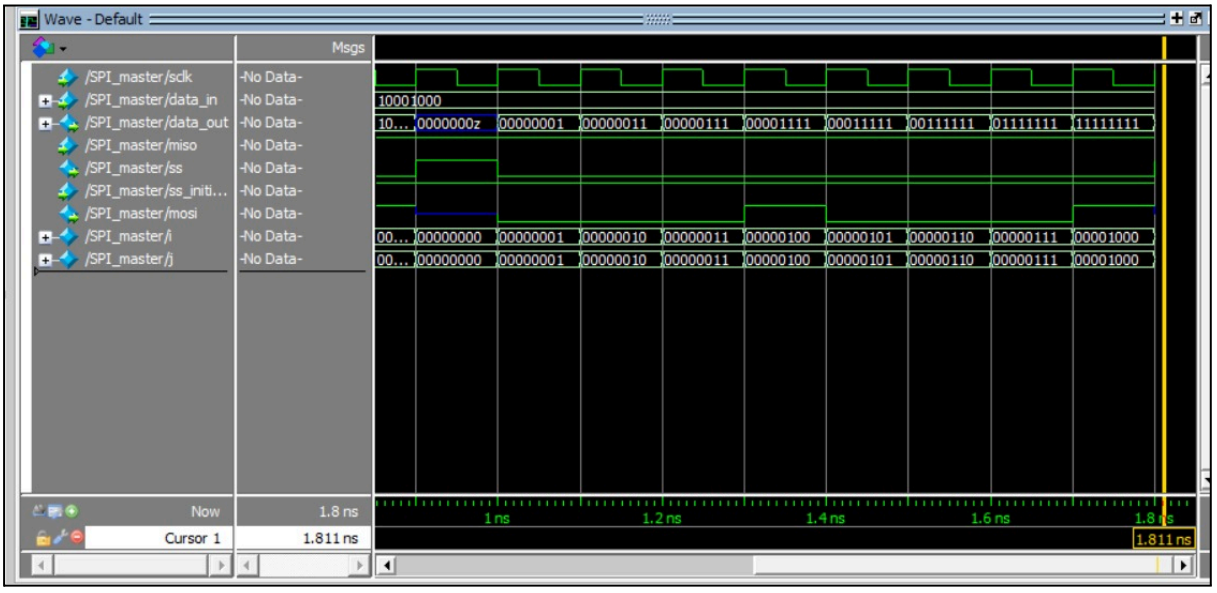


TTL:



Waveforms:





Conclusion:

This Verilog code implements an SPI master in a single master, single slave FPGA system. The SPI protocol offers high data transfer rates, separate MISO and MOSI lines, the absence of start and stop bits, and a simple communication protocol. However, it has limitations such as requiring four wires, lacking acknowledgment for received data, and having no error-checking mechanisms. Additionally, SPI only supports a single master. The code serves as a valuable foundation for SPI-based hardware design and could be enhanced by adding support for multiple masters, error-checking mechanisms, and addressing potential data handling limitations.

Reference:

[1] **https://www.circuitbasics.com/basics-of-the-spi-communication-protocol/**
[2] **https://ijarsct.co.in/Paper14295.pdf**

Appendix:

```
module SPI_master (

  input sclk,    //serial clock from master

  input [7:0] data_in,//data to be transmitted from master

  output reg [7:0] data_out,  // Output for transmitted data from the slave

  input  miso,     // master in slave out

  output reg ss,    //slave select active low

  input  ss_initiate, // transmission signal from master

output reg mosi   // master out slave in

);

reg [7:0] i;

reg [7:0] j;

 initial begin

          i<=0;
           j<=0;
           ss<=1'b1;//deselect slave
  end



 always @(posedge sclk ) begin

   if (ss_initiate==1 && i<8 && j<8) begin

          // Shift out data bit by bit (if the transmission is ongoing)
```

```
                ss<=1'b0;  // select slave
                mosi<=data_in[i];  // shifting data on MOSI line
                 data_out[j]<=miso; // shifting via MISO line

        //Increment
                   i<=i+1;
                   j<=j+1;

    end else begin

          // Transmission complete

                i<=0;
                j<=0;
                ss<=1'b1;  //deselect slave

        // Set to high impedance

          mosi <= 1'bZ;
         data_out<= 1'bZ;

         end
end


 endmodule
```

Working Video Link:
https://youtube.com/shorts/LuTS7woPn2Q?si=KWzcBWHQfiMoqcQA

FPGA Setup: