MANIPAL ACADEMY *of* HIGHER EDUCATION

(Deemed to be University under Section 3 of the UGC Act, 1956)

MANIPAL SCHOOL OF INFORMATION SCIENCES

(A Constituent unit of MAHE, Manipal)

Modified SPI Protocol Implementation using Verilog HDL

Reg. Number	Name	Branch
221039026	Nikhilkumar Agasar	Embedded Systems
221039020	Rahul V	Embedded Systems

Under the guidance of

Dr. Mohan Kumar Jayasubramanian

Associate Professor,

Manipal School of Information Sciences,

MAHE, MANIPAL

22/12/2022

MANIPAL SCHOOL OF INFORMATION SCIENCES

MANIPAL

(A constituent unit of MAHE, Manipal)

MANIPAL SCHOOL OF INFORMATION SCIENCES
MANIPAL
(A constituent unit of MAHE, Manipal)

CERTIFICATE

This is to certify that this project report entitled **Modified SPI Protocol Implementation using Verilog HDL** is a bonafide project work carried out and completed by us, **Nikhilkumar Agasar** and **Rahul V**. This is an original piece /emulated work, done as per the regulations of the **Manipal Academy of Higher Education, Manipal** for the purpose of project learning.

ACKNOWLEDGEMENT

It is great pleasure to thank the people behind the success of this PSD 1 activity. I owe my deepest gratitude to all those who guided, inspired, and helped me to complete this project.

I would like to take this opportunity to express my gratitude and heartiest thanks to Associate Professor Dr. Mohan Kumar Jayasubramanian, Manipal School of Information Sciences for his inspirational support and guidance throughout project period.

My heartful gratitude to Dr Keerthana Prasad, Professor & Director, Manipal School of Information Sciences for her full support and encouragement during the activity.

I would like to thank all sources mentioned in the references.

CONTENTS

List of Figures

ABBREVIATIONS

SPI	Serial Peripheral Interface
MOSI	Master Out Slave In
MISO	Master In Slave Out
CS/SS	Chip/Slave Select
BIST	Built In Self Test
FSM	Finite State Machine
FPGA	Field Programmable Gate Array
UART	Universal Asynchronous Reception Transmission
CPOL	Clock Polarity
CPHA	Clock Phase

Abstract

SPI is a common communication protocol used by many different devices. For example, SD card reader modules, RFID card reader modules, and 2.4 GHz wireless transmitter/receivers all use SPI to communicate with microcontrollers.

One unique benefit of SPI is the fact that data can be transferred without interruption. Any number of bits can be sent or received in a continuous stream. With I2C and UART, data is sent in packets, limited to a specific number of bits. Start and stop conditions define the beginning and end of each packet, so the data is interrupted during transmission.

SPI works on the basis of Master and Slave. The master is given the freedom to select slave on the basis of slave select lines. For different slaves, it requires different slave select lines i.e, n different slaves use n different slave select lines for communication.

Modified SPI comes in view here in which a single slave selection line is used for channelizing the signal for different slaves which reduces slave select lines helping out to reduce complexity of space and time.

CHAPTER 1

1.1 Introduction

Communication between electronic devices is very much essential in today’s scenario as it needs different devices to interact and required to transmit data from device to device. Both sides need to speak the same language. In electronics, these languages are called communication protocols.

In our project we are using SPI protocol which is synchronous and full duplex type. SPI is a serial communication bus developed by Motorola. It is a full-duplex protocol that functions on a master-slave paradigm that is ideally suited to data stream application.

SPI is a common communication protocol used by many different devices. For example, SD card modules, RFID card reader modules, and 2.4 GHz wireless transmitter/receivers all use SPI to communicate with microcontrollers. One unique benefit of SPI is the fact that data can be transferred without interruption. Any number of bits can be sent or received in a continuous stream. Devices communicating via SPI are in a master-slave relationship. The master is the controlling device (usually a microcontroller), while the slave (usually a sensor, display, or memory chip) takes instruction from the master. The simplest configuration of SPI is a single master, single slave system, but one master can control more than one slave.

The master can choose which slave it wants to talk to by setting the slave’s CS/SS line to a low voltage level. In the idle, non-transmitting state, the slave select line is kept at a high voltage level. Multiple CS/SS pins may be available on the master, which allows for multiple slaves to be wired in parallel. If only one CS/SS pin is present, multiple slaves can be wired to the master by daisy-chaining. multiple slaves SPI can be set up to operate with a single master and a single slave, and it can be set up with multiple slaves controlled by a single master.

The current working of SPI is, for communicating with multiple slaves, master has to select multiple slaves using multiple slave select or chip select lines. That means, for n number of slaves, n number of CS/SS lines are used. The main focus of the project is to reduce this number of slave select lines for multiple slaves as well.

Modified SPI implementation uses single slave select line for connecting with multiple slaves or to communicate with multiple slaves. That means, which reduces the selection lines for slaves.

1.2 Objectives

- To study and understand the SPI protocol and the controller aspects
- To design modified SPI for reducing the chip selection pins for Slaves
- To simulate the waveforms of the Modified SPI protocol and show its working.

1.3 Literature Survey

1.3.1 Design and Implementation of SPI Bus Protocol with BIST Capability over FPGA [1]

The authors of this paper Shumit Saha, Md. Ashikur Rahman, Amit Thakur have explained and implemented SPI using BIST technology over FPGA. This BIST technology which is very much required in today's scenario helps in self testing. It gave us an idea about the architecture of the SPI module. It talks about the SPI pin diagrams and the modules used in the design. The comparator used to verify the bit format using some value which gave us idea about using comparator. The waveforms which are given in the paper made us to understand the working of different modules and we could figure out the nature of the waveform in what way we should get that.

1.3.2 Design and Simulation of SPI Master / Slave Using Verilog HDL [2]

This paper which is written by the authors, T. Durga Prasad, B. Ramesh Babu mainly aims about the working of SPI protocol with master and slave. This aims at the data transmission from master to slave. It also talks about the master mode operation and slave mode operation by using master control bit. Master control bit is set means, it is writing data to the shift register in the master and if Master control bit is clear means, it is operating in slave mode. This paper gives the total structure about Master and Slave verification of Testing and Design along with functional verification.

1.3.3 SPI to I2C Protocol Conversion Using Verilog [3]

This paper which is written by authors Dvijen Trivedi, Aniruddha Khade, Kashish Jain, Ruchira Jadhav describes about conversion of SPI protocol to I2C protocol using Verilog The purpose of this paper is to design and simulate a Protocol Conversion Unit (PCU) for seamless communication between the two widely accepted serial communication protocols SPI and I2C. Design given in this paper takes data from a sender device working on SPI protocol and sends it to a receiver device working on I2C protocol, which otherwise without such design would not be possible. This also gives an extract about how to design an SPI FSM for the simulation purpose a top-level design SPItoI2C was implemented.

1.3.4 Design and Implementation of High-Speed Serial Peripheral Interface [4]

This paper which is written by Anand N, George Joseph, Suwin Sam Oommen, and R Dhanabal, gives a great idea about the implementation about the high-speed serial peripheral interface which is having a control register to implement the work of Master mode or slave mode. The control register can be programmed by the user to initiate the data transfer between the master and the slaves. By appropriately programming the control register, the SPI module can be made to operate in master mode or slave mode. The status register gives information on the current position of the data transfer operation. From the SPI status register, information on whether the data transfer has completed or not can be inferred. We also came to know about the detailed SPI protocol architecture from this paper which was functionally implemented and logically synthesised in Xilinx.

1.3.5 An Introduction to I2C and SPI Protocols [5]

The author of this paper, Frédéric Leens has explained detailed history of development of both SPI and I2C protocols and the purpose of those. The paper also explains details about the working mode of the I2C and SPI protocols. This paper has helped us in implementing the block diagrams for our project and understanding the entire protocols in the detailed manner. It also gives brief comparison between I2c and SPI with frame formats and timing diagrams respectively. It also mentions about different concepts like clock stretching in I2C.

1.3.6 A Configurable SPI Interface Based on APB Bus [6]

The paper was authored by, Jiang Yang, Yile Xiao, Dejian Li, Zheng Li, Zhijie Chen, Peiyuan Wan. This paper introduces SPI protocol at first, then divides internal structure of SPI module in detail according to the protocol. Immediately afterwards, this paper designs the registers and SPI state machine interface of the module in detail. Finally, through RTL simulation and FPGA verification, the design can achieve SPI communication correctly. A configurable SPI interface based on APB bus is proposed in this paper. The SPI interface transmission information is configured through APB bus, which can realize the master-slave mode switching, 4 kinds of clock transmission mode switching and MSB/LSB communication mode switching. Meanwhile the transmission rate can be selected and the transmission length can be variable from 1 to 32 bits. Based on the SPI transmission protocol, the proposed design uses a finite state machine method to control the transmission timing of the SPI interface. The design was verified through RTL simulation and FPGA verification. The verification results show that the functional of SPI interface is correct and the data transmission is stable.

1.3.7 A SPI Interface Module Verification Method Based on UVM [7]

This paper which was published was penned by Yong Guo, Yubo Wang, Xiaoke Tang, Yi Hu, Jie Gan, Wennan Feng, Yi Hao. In this paper, a reusable SPI interface testbench based on UVM-based verification methodology is built for the verification of a self-developed SPI interface module. The verification efficiency is improved by introducing the register abstraction layer RAL. The restricted random excitation and automated result comparison functions is introduced to achieve the full verification to the function of SPI interface, and 100% coverage has been achieved. The serial peripheral interface (SPI) is an important module for realizing communication between the APB bus in the SOC chip and peripheral SPI devices. Therefore, efficient and sufficient verification of the function of the SPI module is very important for the design and manufacture of the SOC chip. In this article, a verification environment for SPI module is built based on universal verification methodology (UVM). By introducing the register abstraction layer (RAL), the efficiency of register attribute checking and configuration is greatly improved. In addition, the use of constrained random excitation and automatic result comparison function has realized the full verification of the SPI function, with a coverage rate of 100%. Finally, the verification environment of the SPI module has been successfully migrated to the SOC verification environment.

CHAPTER 2
BACKGROUND THEORY

2.1 Protocol and Types

A **communication protocol** is a system of rules that allows two or more entities of a communications system to transmit information via any kind of variation of a physical quantity. The protocol defines the rules, syntax, semantics and synchronization of communication and possible error recovery methods. Protocols may be implemented by hardware, software, or a combination of both.

There are two types of communication when we refer to embedded systems and microcontrollers. They are Serial Communication and Parallel Communication. In serial communication the data bits are transmitted serially one by one i.e., bit by bit on single communication line, It requires only one communication line rather than n lines to transmit data from sender to receiver. Thus, all the bits of data are transmitted on single lines in serial fashion. In parallel communication, all the bits of data are transmitted simultaneously on separate communication lines. In order to transmit n bit, n wires or lines are used.

Serial communication uses two methods: Asynchronous and Synchronous. Asynchronous, this type transfers single byte at a time, it does not need of clock signal. UART (universal asynchronous receiver transmitter) is asynchronous. Synchronous, this type of protocol transfers a block of data (characters) at a time. It requires clock signal. SPI (serial peripheral interface), I2C (inter integrated circuit) are synchronous serial communication protocol. In the view of Data Transmission, again we can divide it into further like, if the data can be transmitted and received, it is a duplex transmission. If the data can be transmitted in only one direction it is Simplex i.e. from TX to RX only one TX and one RX only. In half duplex, Data is transmitted in two directions but only one way at a time i.e. two TX's, two RX’s and one line. In full duplex, Data is transmitted both ways at the same time i.e. two TX's, two RX’s and two lines.

SPI, I2C, and UART are quite a bit slower than protocols like USB, Ethernet, Bluetooth, and Wi-Fi, but they’re a lot simpler and use less hardware and system resources. SPI, I2C, and UART are ideal for communication between microcontrollers and between microcontrollers and sensors where large amounts of high-speed data don’t need to be transferred.

SPI is called as a 4-wire bus as it requires four wires for its communication as shown below. In the case of single slave communications, we need only 3 wires, as slave select (SS) is not required. So, SPI requires more communication lines in contrast to UART, I^{2}C, USB etc.

2.2 SPI

SPI is a serial communication bus developed by Motorola. It is a full-duplex protocol that functions on a master-slave paradigm that is ideally suited to data stream application. SPI is quite straightforward—it defines features any digital engineer would think of if it were necessary to quickly define a way to communicate between two digital devices.

SPI is a protocol on four signal lines.

The working of the SPI module is essentially based on the contents of an eight-bit serial shift register present in both the Master and the Slave. The transmissions take place based on the clock signal which is generated by the master. The Master, when it wants to send a byte of data to the Slave, places the byte in its shift register and similarly, the Slave can place the content in its shift register. As eight clock pulses are generated, the bits contained in the Master's shift register are transferred by means of the MOSI line to the Slave's shift register and the slave transfers its shift register content by means of the MISO signal line back to the master. So the contents of the two shift registers get exchanged. The below given fig2.2.1 describes block diagram of SPI containing Master and Slave and the signals associated with it. SPI uses the following signals for transmissions across its interface.

SS: This stands for Slave Select. When it goes high, the corresponding slave device will be selected. The slave select line is used by master device to select which slave to initiate communication with the master.

SCK: This stands for serial clock. This signal synchronises the transmissions taking place across the bus.

MOSI: It is serial single bit data line, which the SPI master generates based on internally shifted value of the master data register.

MISO: It is serial single bit data line, by which the SPI slave communicates with the master. It sends out the serially shifted out bits from the slave data register.

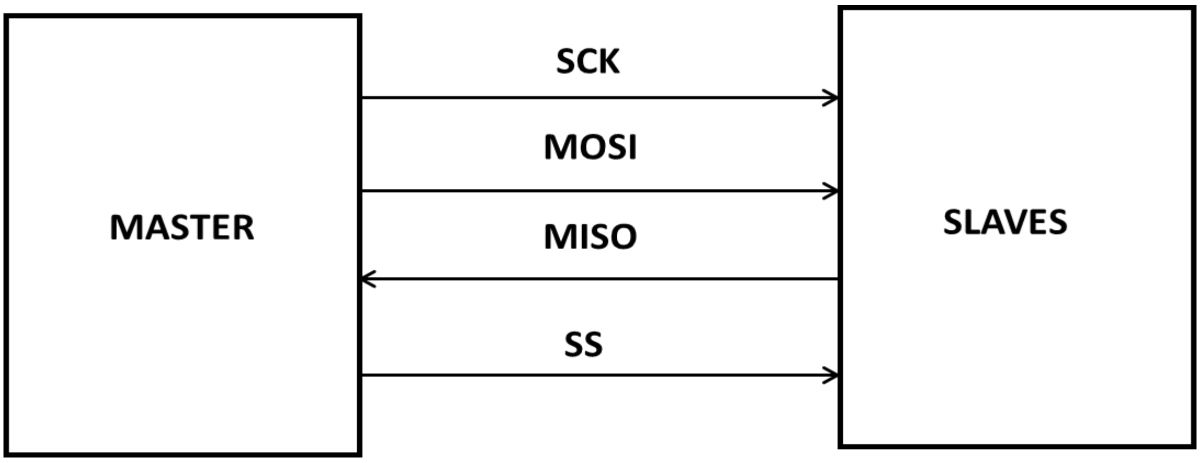

Figure 2.2.1 SPI Block diagram

SPI is a single-master communication protocol. This means that one central device initiates all the communications with the slaves. When the SPI master wishes to send data to a slave and/or request information from it, it selects a slave by pulling the corresponding SS line low, and it activates the clock signal at a clock frequency usable by the master and the slave. The master generates information onto the MOSI line while it samples the MISO line. Four communication modes are available (MODE 0, 1, 2,3) that define

- The SCLK edge on which the MOSI line toggles
- The SCLK edge on which the master samples the MISO line
- The SCLK signal steady level (that is, the clock level, high or low, when the clock is not active)

Each mode is formally defined with a pair of parameters called clock polarity (CPOL) and clock phase (CPHA)

A master/slave pair must use the same set of parameters SCLK frequency, CPOL, and CPHA for a communication to be possible. If multiple slaves are used that are fixed in different configurations, the master will have to reconfigure itself each time it needs to communicate with a different slave. This is basically all that is defined for the SPI protocol. SPI does not define any maximum data rate nor any particular addressing scheme; it does not have an acknowledgement mechanism to confirm receipt of data and does not offer any flow control. Actually, the SPI master has no knowledge of whether a slave exists, unless "something" additional is done outside the SPI protocol. For example, a simple codec will not need more than SPI, whereas a command response type of control would need a higher-level protocol built on top of the SPI interface. SPI does not care about the physical interface characteristics like the I/O voltages and standard used between the devices. Initially, most SPI implementation used a non-continuous clock and byte-by-byte scheme, but many variants of the protocol now exist that use a continuous clock signal and an arbitrary transfer length.

- Master – Out / Slave – In or MOSI, as the name suggests, is the data generated by the Master and received by the Slave. Hence, MOSI pins on both the master and slave are connected together. Master – In / Slave – Out or MISO is the data generated by Slave and must be transmitted to Master.
- MISO pins on both the master and slave are ties together. Even though the Signal in MISO is produced by the Slave, the line is controlled by the Master. The Master generates a clock signal at SCLK and is supplied to the clock input of the slave. Chip Select (CS) or Slave Select (SS) is used to select a particular slave by the master.
- Since the clock is generated by the Master, the flow of data is controlled by the master. For every clock cycle, one bit of data is transmitted from master to slave and one bit of data is transmitted from slave to master.
- This process happens simultaneously and after 8 clock cycles, a byte of data is transmitted in both directions and hence, SPI is a full – duplex communication.
- If the data has to be transmitted by only one device, then the other device has to send something (even garbage or junk data) and it is up to the device whether the transmitted data is actual data or not.
- This means that for every bit transmitted by one device, the other device has to send one bit data i.e. the Master simultaneously transmits data on MOSI line and receive data from slave on MISO line.
- If the slave wants to transmit the data, the master has to generate the clock signal accordingly by knowing when the slave wants to send the data in advance.

During the positive edge of the clock signal, both the devices (master and slave) read input bit into LSB of the register. During the negative cycle of the clock signal, both the master and slave places a bit on its corresponding output from the MSB of the shift register.
Hence, for each clock cycle, a bit of data is transferred in each direction i.e. from master to slave and slave to master. So, for a byte of data to be transmitted from each device, it will take 8 clock cycles.
We have already seen that it is the job of the Master device to generate the clock signal and distribute it to the slave in order to synchronise the data between master and slave. The work of master doesn’t end at generating clock signal at a particular frequency.

In fact, the master and slave have to agree on certain synchronization protocols. For this, two features of the clock i.e. the Clock Polarity (CPOL or CKP) and Clock Phase (CPHA) come in to picture.

Clock Polarity determines the state of the clock. When CPOL is LOW, the clock generated by the Master i.e. SCK is LOW when idle and toggles to HIGH during active state (during a transfer). Similarly, when CPOL is HIGH, SCK is HIGH during idle and LOW during active state.

Clock Phase determines the clock transition i.e. rising (LOW to HIGH) or falling (HIGH to LOW), at which the data is transmitted. When CPHA is 0, the data is transmitted on the rising edge of the clock. Data is transmitted on the falling edge when CPHA is 1.

Depending on the values of Clock Polarity (CPOL) and Clock Phase (CPHA), there are 4 modes of operation of SPI: Modes 0 through 3. Fig 2.2.2 describes different modes of working.

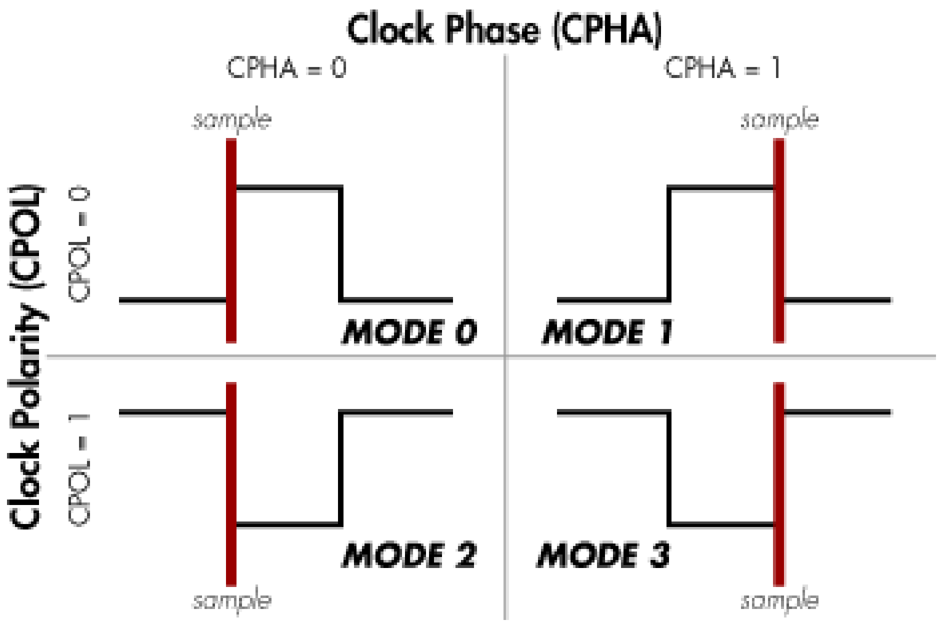

Figure 2.2.2 SPI Working Modes

Mode 0:

Mode 0 occurs when Clock Polarity is LOW and Clock Phase is 0 (CPOL = 0 and CPHA = 0). During Mode 0, data transmission occurs during rising edge of the clock.

Mode 1:

Mode 1 occurs when Clock Polarity is LOW and Clock Phase is 1 (CPOL = 0 and CPHA = 1). During Mode 1, data transmission occurs during falling edge of the clock.

Mode 2:

Mode 2 occurs when Clock Polarity is HIGH and Clock Phase is 0 (CPOL = 1 and CPHA = 0). During Mode 2, data transmission occurs during rising edge of the clock.

Mode 3:

Mode 3 occurs when Clock Polarity is HIGH and Clock Phase is 1 (CPOL = 1 and CPHA = 1). During Mode 3, data transmission occurs during rising edge of the clock.

The timing diagram of SPI communication is given below in the figure 2.2.3

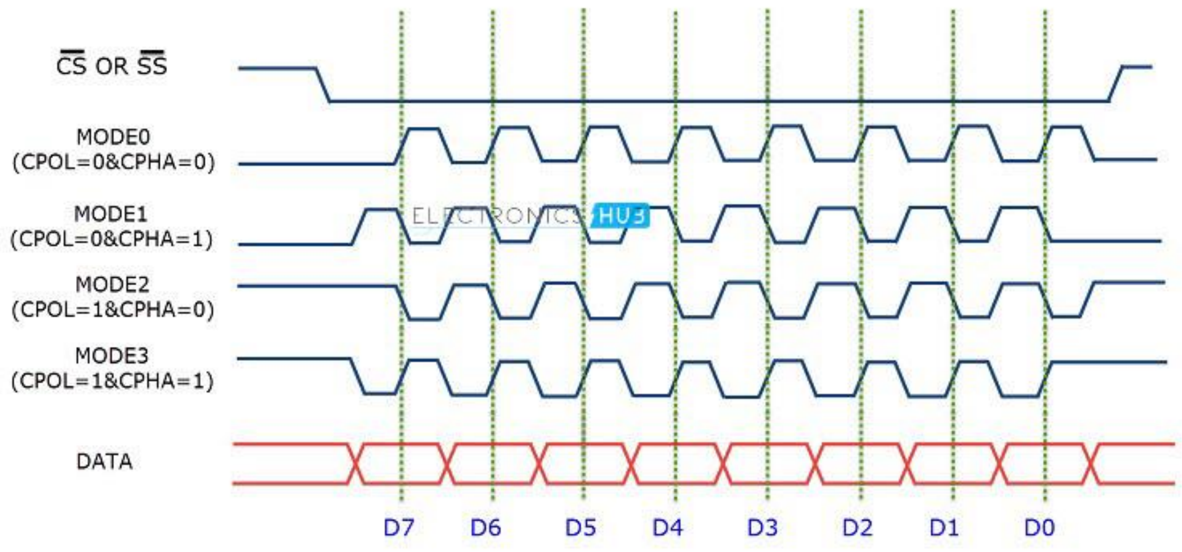

Figure 2.2:3 Timing Diagram of SPI

Multiple Slave concept with Single Master:

In SPI, multiple slaves can be connected to Single master. As we have seen in working of SPI protocol, the connection of slaves or slaves can be selected for communication from both the side by master and slave on the basis of slave select lines.

That means, if a single slave is to be selected means, a particular slave select line should be connected and activated based on the slave position. For different slaves, there are different slave select lines or chip select lines. Which results, for n number of slaves we require n selection lines specifically which are also unique.

Fig.2.2.4 shows the multiple slave connection to single master with different slave selection lines. Slave 1 is activated using SS1 line, Slave 2 is activated using SS2 line, Slave n is activated using SSn line and so on. Hence this is the common mode of activation and addressing of slaves using Slave select lines that too multiple slave select lines for multiple slaves from single master.

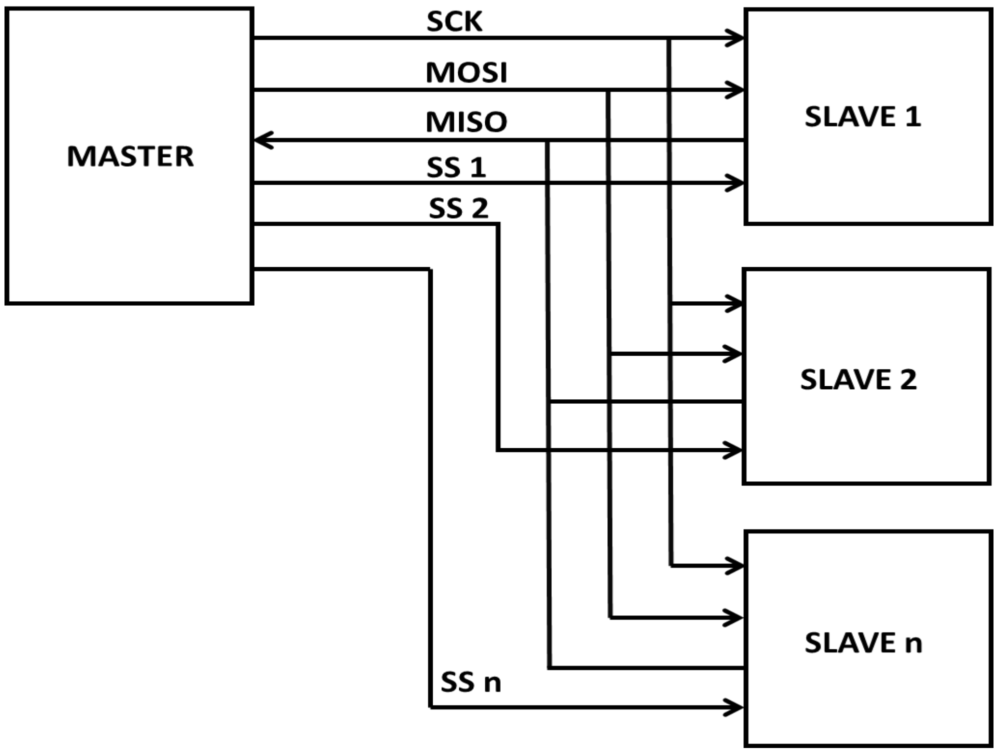

Figure 2.2.4 Multiple Slaves using multiple Slave select lines

Daisy Chain fashion of SPI:

In Daisy chain SPI interface; command propagates from one device to another, connected in serial. Figure2.2.5 shows SPI fashion connected in daisy chain configuration.

Active low slave select signal and SCLK are connected to all devices. Only the first slave in the chain receives the command data directly from the microcontroller. Every other slave in the network receives its MOSI data from the MISO output of the preceding slave in the chain. The SPI port of each slave is designed to send out during the second group of clock pulses an exact copy of what it received during the first group of clock pulses. The whole chain acts as an SPI communication shift register.

In Daisy Chain Configuration, only a single Slave Select line is connected to all the slaves. The MOSI of the master is connected to the MOSI of slave 1. MISO of slave 1 is connected to MOSI of slave 2 and so on. The MISO of the final slave is connected to the MISO of the master. Consider the master transmits 3 bytes of data in to the SPI bus. First, the 1st byte of data is shifted to slave 1. When the 2nd byte of data reaches slave 1, the first byte is pushed in to slave

2. Finally, when the 3rd byte of data arrives in to the first slave, the 1st byte of data is shifted to slave 3 and the second byte of data is shifted in to second slave.

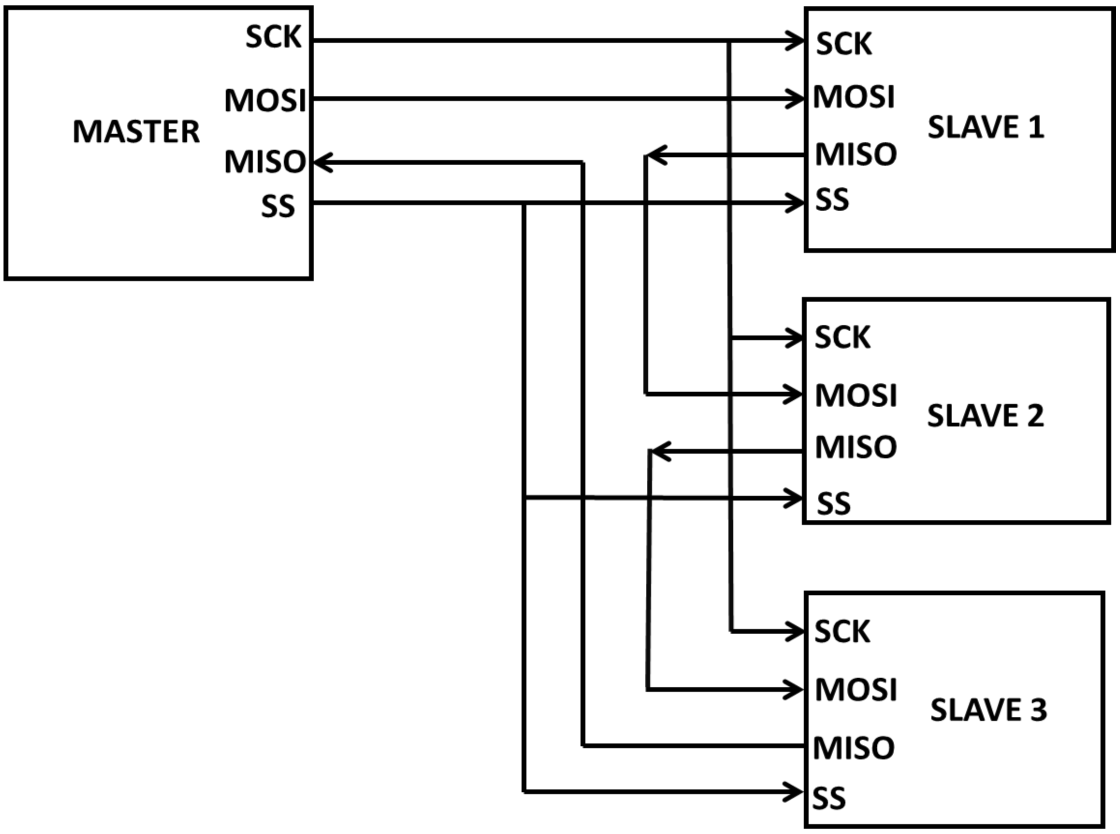

Figure 2.2.5 Multiple Slaves in Daisy Chain Fashion

Modified SPI:

To overcome this multiple slave activation using multiple slave selection lines, modified SPI comes into view. Where the SPI has single slave select line with connection into multiple slaves. For n different slaves, only one slave select line is used for slave activation by channelising it based on the slave address.

The working and design of the proposed model of Modified SPI is explained in next chapters.

CHAPTER 3
PROPOSED BLOCK DIAGRAM & DESIGN

3.1 Proposed Block

A explained in the above section regarding multiple slave selection using slave select lines, to overcome this multiple slave activation using multiple slave selection lines, modified SPI comes into view. Where the SPI has single slave select line with connection into multiple slaves. For n different slaves, only one slave select line is used for slave activation by channelising it based on the slave address.
The working and design of the proposed model of Modified SPI is explained in next chapters. We have used Demux for slave switching.

The slaves are addressed using slave address. The slave address is stored as selection lines of demux. Number of slaves addressed are in number of 16. The 16 slaves are addressed using 4 bit of address lines which are 4-bit selection for a demux. Slave selection is done using slave select signal passing it through as input to demux and switching it over slaves on basis of slave address.

The architecture and block diagram of individual blocks like Master, Slave and Demux has been explained below with the signals of present as input and output from the block.

3.1.1 SPI Master:

SPI master is always defined as a microcontroller in the embedded systems. The microcontroller will be communicating with different slaves, on the basis of slave select lines and serial clock. SPI master circuit is nothing but the combination of a latch and a shift register, which will be switching or shifting the data bits on the basis of data size. SPI master block has different signals as input. The signals involved in SPI master block are CDIV, CLK, D_IN, MLB, RSTB, START, T_DATA as part of Input and D_OUT, R_DATA, SCK, SS as part output.

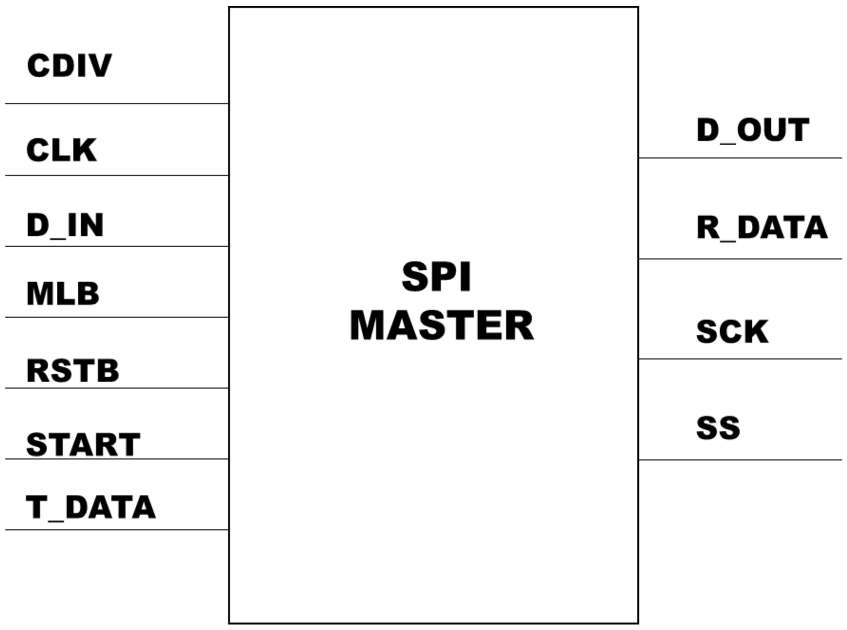

Figure 2.1.1 SPI Master block

CDIV: This register is used to store a divisor for use in generation of the SPI_CLK signal, based on the external clock signal (CLK). As part of the SPI Clock Generation Unit, an internal counter is used to count up to the value written to the CDIV register. The next edge of SPI_CLK will only be generated when the internal counter reaches this divisor value.

CLK: Clock signal. Controlled by the master device. A new data bit is shifted out with each clock cycle.

D_IN: This is the MISO signal that is coming from slave which is abbreviated as data in.

MLB: This is used to select the endianness of a data whether the data to be stored from LSB or MSB. If MLB=0, the reading and storing sequence pattern is from LSB. If MLB=1, the reading and storing sequence pattern is from MSB.

RSTB: This is the reset bit of the master, this will reset the circuit, if rstb=0, then the state is finish that leads to idle state.

START: This signal starts the data transmission.

T_DATA: This signal is input helps in storing data from the system which in turn to be sent to slave.

D_OUT: This is the MOSI signal which is to be sent to slave, the data is to be sent to slave data out.

R_DATA: Received data signal which is a output which holds the data from master and to send it to tdata in slave.

SCK: Serial Clock, this signal manages clock signal between master and slave.

SS: This signal is used to select the slave for communication.

3.1.2 SPI Slave:

SPI slave is always defined as any sensor or device which is connected in the embedded systems applications. The microcontroller will be communicating with different slaves, on the basis of slave select lines and serial clock. SPI slave circuit is nothing but the combination of a latch and a shift register, which will be switching or shifting the data bits on the basis of data size. It will receive the data communication from Master and writes back to Master for any communication as it is full duplex protocol. SPI slave block has different signals as input. The signals involved in SPI slave block are MLB, RSTB, T_DATA, SCK, TEN, SS, SD_IN as part of Input and SD_OUT, R_DATA, DONE as part output.

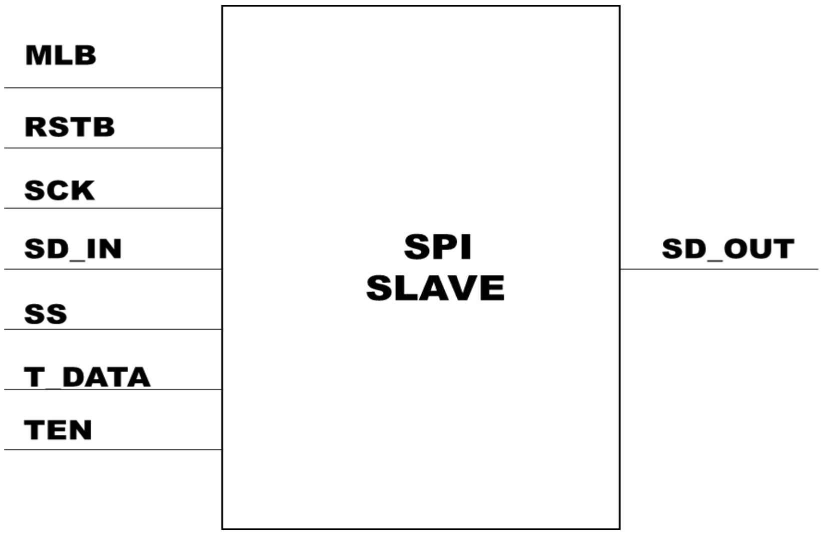

Figure 3.1.2 SPI Slave block

MLB: This is used to select the endianness of a data whether the data to be stored from LSB or MSB. If MLB=0, the reading and storing sequence pattern is from LSB. If MLB=1, the reading and storing sequence pattern is from MSB.

RSTB: This is the reset bit of the master, this will reset the circuit, if rstb=0, then the state is finish that leads to idle state.

SCK: Serial Clock, this signal manages clock signal between master and slave.

SS: This signal is used to select the slave for communication.

T_DATA: This signal is input helps in storing data from the master rdata which in turn to be sent to master.

SD_IN: This signal is MOSI which is used to take serial data from master.

T_EN: Transmission enable signal which is used to enable the transmission back to master from slave.

SD_OUT: This signal is used to transfer data from slave to din of master.

R_DATA: Received data signal which is a output which holds the data received.

Done: This signal indicates the end of transmission of data from master to slave and vice versa.

3.1.3 DEMUX:

The demultiplexer takes one single input data line and then switches it to any one of a number of individual output lines one at a time. The demultiplexer converts a serial data signal at the input to a parallel data at its output lines. Here, demux is taking input as slave select line and channelising it across different slaves present. The demux used here is 1:16 which will communicate with and channelize 16 slaves to communicate on the basis of slave address.

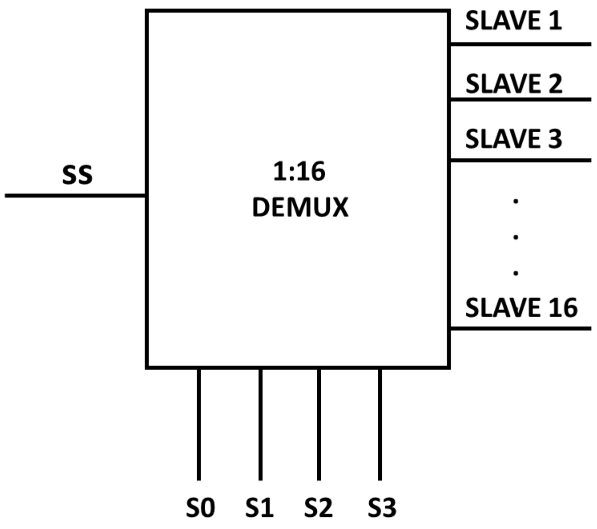

Figure: 3.1.3 Demux Block Diagram

3.1.4 Modified SPI Protocol Block Diagram:

Multiple slave selection can be done using different slave selection lines. But this increasing slave selection lines increases the hardware complexity of the protocol. So, in this project demux is used for slave selection purpose. Usage of demux for designing a modified SPI protocol, as to reduce the number of slaves select lines from the master. We are connecting 16 slaves to master using 1:16 demux. Each slave is defined by the different address, these addresses are mapped with defined selection lines of the demux.

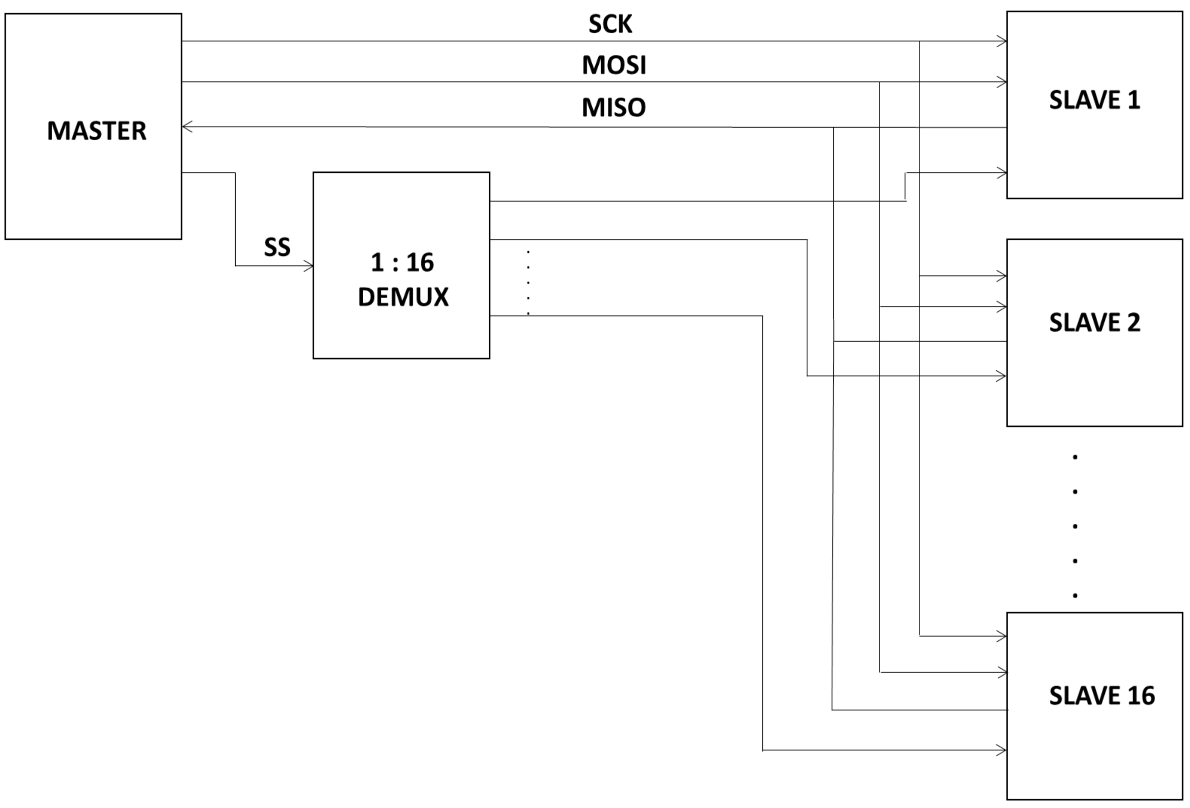

Figure: 3.1.4 Modified SPI Block Diagram

CHAPTER 4
WORK DONE

The proposed block diagram design of the modified SPI protocol which is the main objective of our project has been discussed already in chapter 3. While designing the modified SPI protocol and understanding the SPI protocol controller aspects was fulfilled by implementing Verilog HDL code with Xilinx Vivado software used as a tool for developing master and slave blocks.

4.1 Working of Individual blocks:

The current working of SPI protocol takes place on the basis of MODE 3. That defines, sampling of the data should be done at rising edge of SCK and shifting of the data takes place at falling edge of SCK.

SPI Master:

Using Verilog, SPI Master code has been written, the working goes as follows, the master working is defined with the help of an FSM with different states. The different states are defined as idle, send(working) and finish. The parameter variables cur(current state) and next(next state) are used for transition between states. Based on the value of cdiv, a count value called mid is generated for setting the operation for falling edge of the clock.

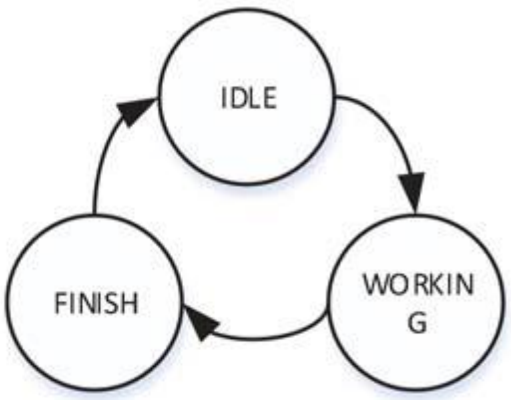

Figure 4.1.1 States of Master's FSM

Here, slave select is used to select the particular slave which is an output of the master block.

SPI Slave:

Using Verilog, SPI Slave code has been written, the working goes as follows, here assign keyword present in Verilog is used which will run concurrently which will help to send the data from slave to master. There are two concurrently running blocks, one block is used for receiving data from master and another one is used for send the data from slave to the master. On the basis of particular slave select line, particular slave is selected as per the working of normal SPI protocol.

Demux:

1:16 demux is used for the selection of multiple or particular slaves based on their address. The slaves are selected on the basis of their address which is selection lines of demux. The active low slave select line is fed into the system of demux input and it is switched for the slaves particularly.

4.2 Working of Modified SPI:

In typical SPI communication protocol if number of slaves increases then number of slaves select signals should be increases.by increasing the slave select signal s in master it increases the space complexity of the protocol.

By analysing these scenarios, we approached a modified SPI protocol with the adding the extra block of demux in between the master and the slaves. Fig 4.1 describes the RTL schematic block of Modified SPI which includes 2 slaves which generates 4 slave address, along with 1:4 demux for enabling the visualization view of the schematic.

In our project, we used 16 slaves and 1 master for the communication. We feed active low slave selection line into the input of demux, on the basis of slave address particular slave is selected for the communication by channelizing output of the demux to the respective slaves. Hence, the communication between the master and slave/s is fulfilled.

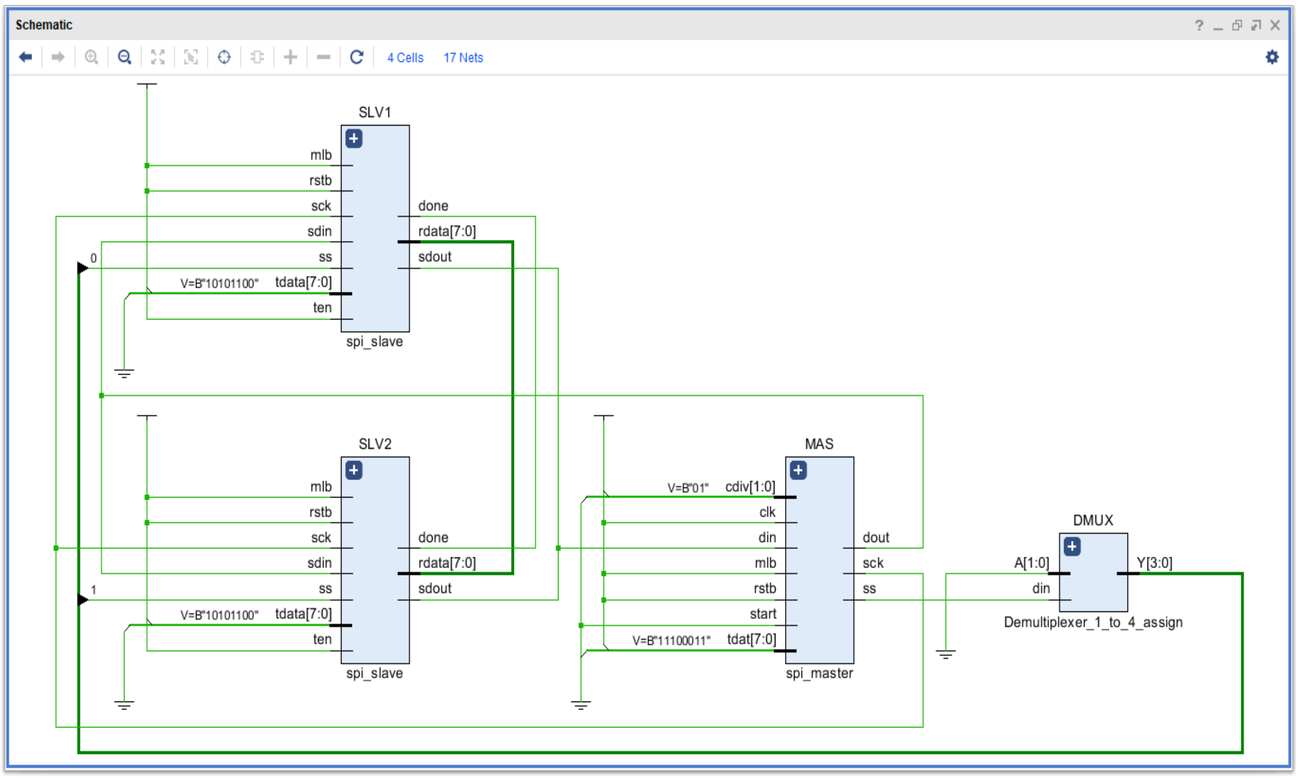

Figure 2.1.2: RTL Schematic of SPI Interface

CHAPTER 5
RESULT AND CONCLUSION

5.1 Result Analysis:

The result analysis of SPI protocol is obtained when the data which is given at the Master is transferred to Slave and vice versa is fulfilled.

In the part of Modified SPI, the slaves are called with their particular address and the data is transferred from master to particular mentioned slave.

The address of a slave to which data has to be transferred is mentioned in Testbench.

Case 1:

The Data which we have given is 7C => 01111100 which is to be transferred from master to slave, from fig. 5.1.1 we can note that the at start the data is present at m_tdat[7:0][5.1.1] which is transferred to SLVrdata[7:0] after 8 SCK[3.1.2] clock signals each SCK signal is of 2 units(200ns) and the slave address given is A=0001

Case 2:

The Data which we have given is AC => 10101100 which is to be transferred from slave to master, from fig. 5.1.1 we can note that the at start the data is present at s_tdat[7:0][5.1.1] which is transferred to Mrdata[7:0] after 8 SCK[3.1.2] clock signals each SCK signal is of 2 units(200ns) and the slave address given is A=0001

Hence, from the above given cases, we can verify that the data is transferred from Master to Slave and as well as Slave to Master which will fulfil our aim to transfer the data as SPI is full duplex.

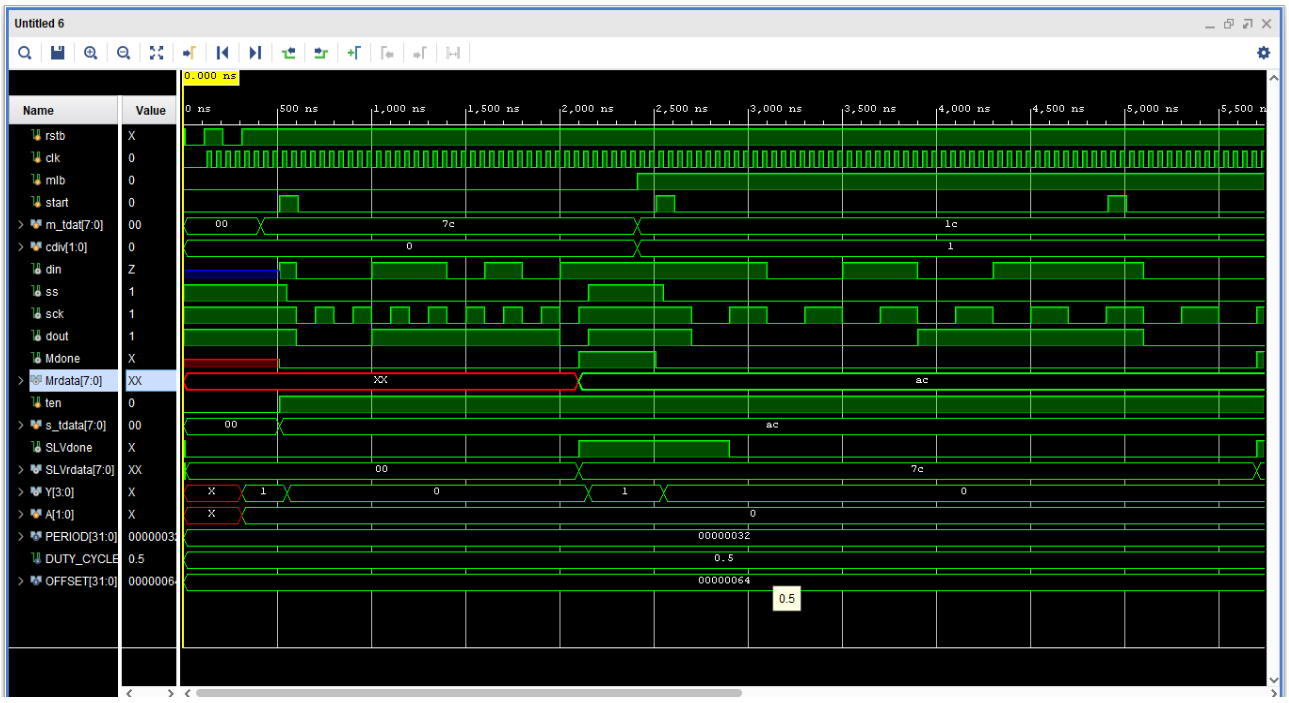

Figure 5.1.1 Waveform output of Modified SPI

5.2 Conclusion:

From the given result analysis [5.1], we can sum up that, understanding and implementing SPI controller aspects using Verilog HDL is achieved. Designing and implementing Modified SPI using Verilog HDL and data transfer is verified using Demux.

Simulation of waveforms of Modified SPI using Verilog HDL is achieved and verified.

After analysing the waveforms of SPI protocol of multiple slaves in typical form and comparing with Modified SPI, the Modified SPI reduces the complexity by space and time.

- The modified SPI protocol has some advantage over the typical SPI protocol.
- This design saves space and makes adding and removing of slaves easy.
- Modified SPI can reduce area and hardware components which are used in designing.

5.3 Scope of further Work:

- After analysing SPI protocol, we can come to know that the one challenge in SPI is about SlaveSelect or ChipSelect lines.
- For n different Slaves there are n different CS lines. This will increase the Hardware size and make the circuit or system more complex.
- Hence, the future scope lies here to modify this protocol to use single select line for multiple slaves. In this project, Demux is used for switching slaves. Further it can be done using Address verification methodology like I2C working in which all ack bits, start and stop bits can be included.
- In any other further ways, in which the protocol, slave select lines can be reduced for working.

References

[1] S. Saha, M. A. Rahman and A. Thakur, "Design and implementation of SPI bus protocol with Built-in-self-test capability over FPGA," 2014 International Conference on Electrical Engineering and Information & Communication Technology, 2014, pp. 1-6, doi: 10.1109/ICEEICT.2014.6919076.

[2] Prasad, Tadi Durga and B. Ramesh Babu. "Design and Simulation of SPI Master / Slave Using Verilog HDL." (2014). Paper ID: 02015624 International Journal of Science and Research (IJSR) ISSN (Online)

[3] D. Trivedi, A. Khade, K. Jain and R. Jadhav, "SPI to I2C Protocol Conversion Using Verilog," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), 2018, pp. 1-4, doi: 10.1109/ICCUBEA.2018.8697415.

[4] Anand N, G. Joseph, S. S. Oommen and R. Dhanabal, "Design and implementation of a high speed Serial Peripheral Interface," 2014 International Conference on Advances in Electrical Engineering (ICAEE), 2014, pp. 1-3, doi: 10.1109/ICAEE.2014.6838431.

[5] F. Leens, "An introduction to I2C and SPI protocols," in IEEE Instrumentation & Measurement Magazine, vol. 12, no. 1, pp. 8-13, February 2009, doi: 10.1109/MIM.2009.4762946.

[6] J. Yang, Y. Xiao, D. Li, Z. Li, Z. Chen and P. Wan, "A Configurable SPI Interface Based on APB Bus," 2020 IEEE 14th International Conference on Anti-counterfeiting, Security, and Identification (ASID), 2020, pp. 73-76, doi: 10.1109/ASID50160.2020.9271704.

[7] Y. Guo et al., "A SPI Interface Module Verification Method Based on UVM," 2020 IEEE International Conference on Information Technology,Big Data and Artificial Intelligence (ICIBA), 2020, pp. 1219-1223, doi: 10.1109/ICIBA50161.2020.9277156.

[8] Opencores Verilog SPIhttps://opencores.org/projects/spi_verilog_master_slave