INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 6.078
(Volume 6, Issue 4)
Available online at: https://www.ijariit.com

VLSI implementation of SPI and I^{2}C communication protocols

Muhammad Bakir
muhammadbakir550@gmail.com
N.M.A.M. Institute of Technology, Nitte, Karkala, Karnataka

Niju Rajan
nijurajan@nitte.edu.in
N.M.A.M. Institute of Technology, Nitte, Karkala, Karnataka

ABSTRACT

For the frequent communication between the integrated circuits in an electronic system, which are designed to work in a group rather than a standalone unit, it becomes necessary to adapt serial communication, which is an effective and simple protocol, that provides efficient communication among these various components as compared to parallel communication in terms of the pin count and the ease of implementation. The two most widely accepted, tried and true serial global standards are Serial Peripheral Interface (SPI) and Inter-Integrated Circuit (I^{2}C), which are used for inter-chip and intra-chip serial communication for a low or medium bandwidth. The Paper's focus is on the simulationof SPI and I^{2}C protocols.

Keywords: *Integrated Circuit, I^{2}C Protocol, SCL (Serial Clock), SDL (Serial Data Line) SPI Protocol, and Protocol Conversion Unit.*

1. INTRODUCTION

For low-end low bandwidth serial communication the two widely accepted standards areI^{2}C and SPI. The two protocols have their own prospects and are strong opponents to each other in the market. By and large there is a high possibility that they exist in the digital domain. With a lot of peripheral devices from a different vendor there is always a choice to be made between the I^{2}C and SPI protocols.

1.1 Serial Peripheral Interface (SPI)

SPI facilitates a Synchronous, Duplex and Serial Communication between the Peripheral devices. It supports Data transfer speed up to 400 Mbps between the Microprocessor and various peripherals. Similar to its counterpart SPI provides a simplest interface with only four pins they are MOSI, MISO, SSN and SCLK are as shown in Figure 1 and 2.

1.1.1 Pin description

- **SCLK**: Data transfer is synchronized by Serial Clock line. Master can only generate the Clock.
- **MOSI**: Master Out Slave In also denoted as Serial Data out line (SDO) and it is used to send the data from Master to Slave.
- **MISO**: Master In Slave Out also denoted as Serial Data Input line (SDI) is used to send data froma Slave toMaster.
- **SSN**: Slave Select signal, which is always active low line and for each device connected on the communication bus the Master dedicates a special line for every single Slave device. In order to drive 'n' number of Slaves on the communication bus, Master should have 'n' number of SSN lines.

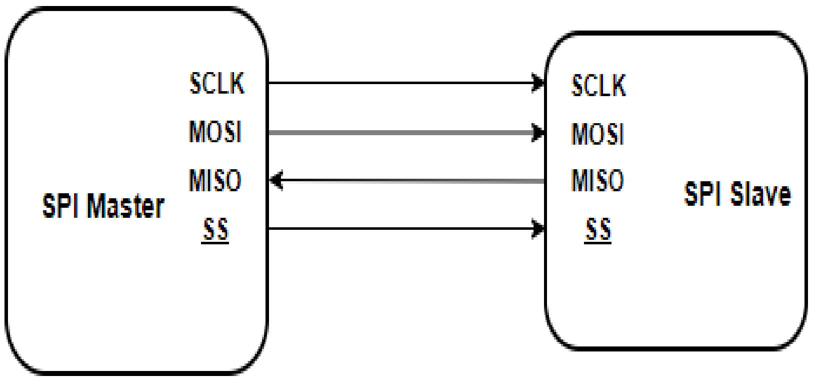


Fig. 1: SPI Master connected single Slave

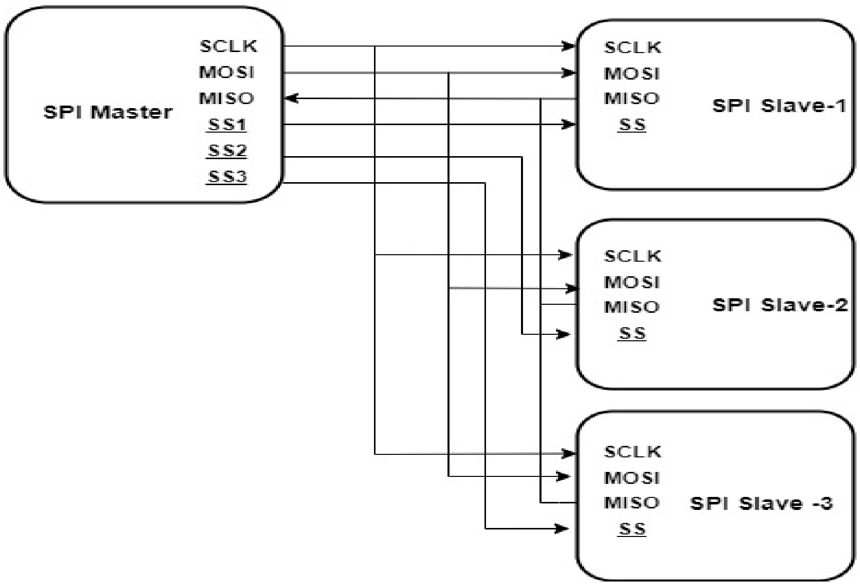


Fig. 2: SPI Master connected to multiple Slaves

1.2 Inter-Integrated Circuit (I^{2}C)

I^{2}C is a serial, short range, easy communication without any loss and this protocol has multi Master capability. It is the simplest of its kind with only two external wire, they are Serial Clock (SCL) and Serial Data (SDA) line.
There are three modes operation in I^{2}C, they areas following:
a. Standard mode –rate of data transfer is up to 100 Kbps
b. Fast mode –rate of data transfer is up to 400 Kbps
c. High Speed mode –rate of data transfer is up to 3.4 Mbps

Data transfer takes place through the SDA and SCL lines of I^{2}C. Any data changeover can occur on the SDA line only when the SCL line is at Logic Low and all these changeovers have to settle down until SCL becomes Logic high. I^{2}C is a bi-directional bit-oriented communication protocol hence it facilitates a Full Duplex communication between Master and Slave.

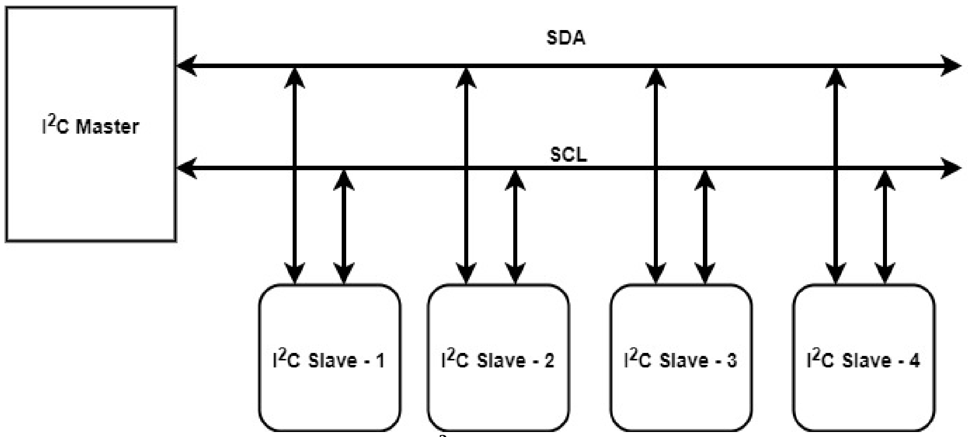


Fig. 3: I^{2}C block diagram

2. LITERATURE REVIEW

I^{2}C protocol gives easy communication with no data loss. It gives excellent speed compared to former protocols such as UART, USB and CAN. It is lightweight, omnipresent and economical. It also increases rate of data transfer when compared other protocols. The main goal of the paper [1] is to develop the protocol to control the data that can be saved on registers as well as registers inside the devices and to get high speed communication, through this they were able to control different parameters. I^{2}C is used in data surveillance for efficiency and accuracy. The ideal surveillance architecture with I^{2}C protocol will be having the following features: high flexibility, performance, low development cost, easy upgradability, and a migration path to lower cost as the application grow up and volume upgrades.

The I^{2}C protocol is a renowned serial communication protocol developed by Philips Semiconductor (now it is known as NXP Semiconductor) in 1980 to exchange data especially between slow and fast devices [2]. It consists of only two external wires SDA and SCL and its capability to transmit data with no loss makes it simpler and inexpensive than other protocols. This paper [3] is aimed to present the valuable study on I^{2}C protocol by different scholars over the years.

A SPI bus is a communication protocol that allows serial data transfer between a master and a slave device. In the paper [4], their focus was on to present a full explanation of a Serial peripheral interface Master/Slave design and implementation. The Design specification is based on Motorola's Serial peripheral interface Guide version V03.06. The purpose of the paper is to provide information about communication among serial peripheral interfaces (SPI) Master to Serial peripheral interface Slave. The

Complete design was implemented in Verilog HDL [5], and they mapped onto Xilinx FPGA device.
Comparison of physical implementation aspects of SPI and I^{2}C protocols through a number of recent Xilinx's FPGA families, showing which protocol features is accountable for significant area overhead are explained in [6]. This valuable data helps design engineer to make cautious and tightly tailored architecture resolutions. For a comprehensive comparative study [7], both protocols are designed as general-purpose IP solutions, integrating all necessary features required by modern SoC/ASIC applications according to a recent market analysis of an important number of profitable I^{2}C and SPI devices. The Register Transfer Level (RTL) code is technology independent, introducing around 25% area overhead for I^{2}C over SPI, and nearly the same propagation delays for both designs.

In the world of communication protocols, I^{2}C and SPI are often considered as “little” communication protocols compared to Ethernet, USB, SATA, PCI-Express and others that shows throughput in the multiple of 100 Mb/s range, if not Gb/s. It is important that one shouldn't forget the purpose of each protocol [8][9]. Ethernet, USB, and SATA are meant only for “out of the box communications” and data exchanges between whole systems. When there is a necessity to implement a communication among an integrated circuit such as a microcontroller and a set of relatively slow peripherals, there is no need to use excessively complex protocols. Therefore, SPI and I^{2}C perfectly fit the bill and have become so popular that it is very likely that any embedded system engineer will use them during his/her careers.

3. IMPLEMENTATION OF SPI AND I^{2}C PROTOCOLS

For serial communication two protocols namely SPI and I^{2}C are widely used. More often there are wide varieties of peripherals from different manufacturers using either of the two mentioned protocols for serial communication. In this section the detailed implementation of SPI, I^{2}C are explained.

3.1 SPI Protocol

A SPI protocol may consist of microcontroller as Master and another microcontroller as a Slave. It is primarily a Master Slave relationship that exists here. The relation of SPI can have multi-Master and Slave too. Each Master has at least 4 wire lines to communicate with single Slave.

- First the Master decides which Slave it has to send the signal. Then it will make the Slave select line of that particular Slave on. Ie: the resistor is pulled down and then wires value become low thereby selecting that particular Slave. When the Slave gets selected the wire is at low. The other entire Slave wires remains high excluding the one selected. That is, at most one SS will remain low.
- As soon as the SS line turns low the SCL starts ticking, then the Master sends either 0 or 1. If it is 0 then read operation else it is write operation.
- After sending the read/write bit, Master start's sending out the address bits on MOSI line bit by bit at every posedge of SCL on which action has to be performed.
- When the address bits are completed Master sends out the data bits in bit by bit manner on the MISO line.
- If the Master sends a read bit then the Slave reads the address send by the Master and then passes the required data stored at that particular address bit by bit synchronized with SCL clock on the MISO line.
- If the Master sends a write bit then the Slave reads address send by the Master and store the data send by Master in that particular address. Data flow diagram of SPI Master and Slave communication is as shown in Figure 4.

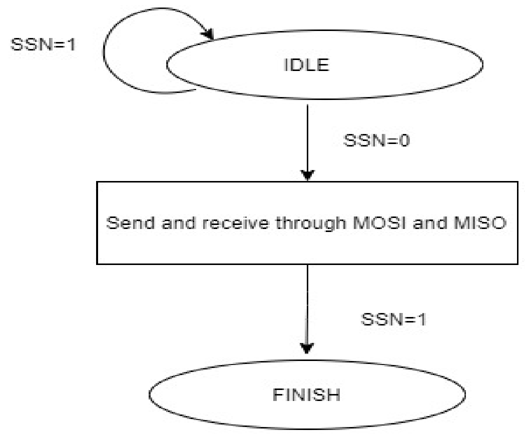


Fig. 4: Flow chart of SPI Master-Salve communication

3.2 I^{2}C Protocol

Following steps are followed in I^{2}C communication.

- Initially SCL and SDA lines are high.
- First SDA line goes high to low, followed by SCL line going high to low. This is start bit.
- Now master generates clock on SCL line in synchronization to clock on SCL all the transfer of bits occur on SDA line.
- Master sends address of slave device it wants to communicate with. It appends a R/W bit at the end of address and only the slave device connected to master receive this address and only the slave and only the slave device with address sent on the SDA line sends an ACK bit. Reset all slave leaves control of SCL and SDA line, R/W ‘0’ for Read and ‘1’ for Write. Finite state diagram is as shown in Figure 5 and 6.

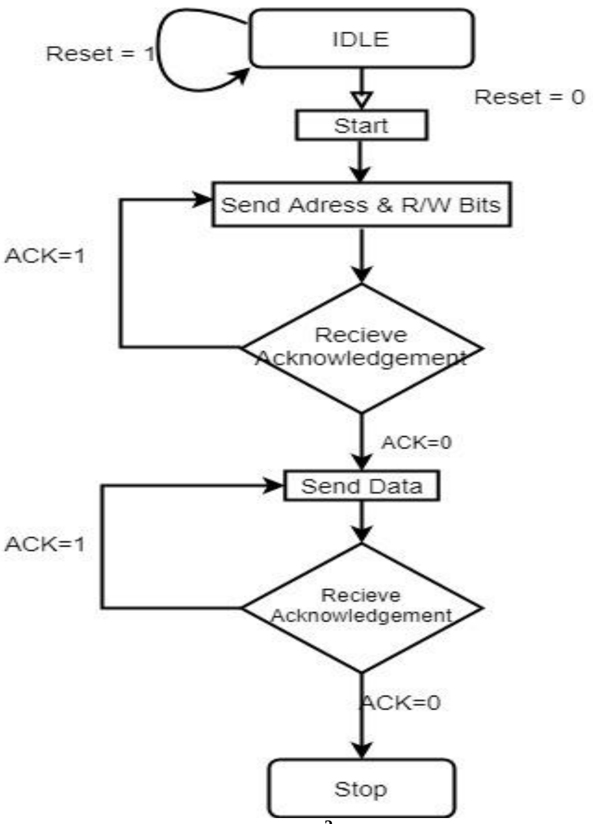


Fig. 5: Data flow diagram of I^{2}C Master Communication

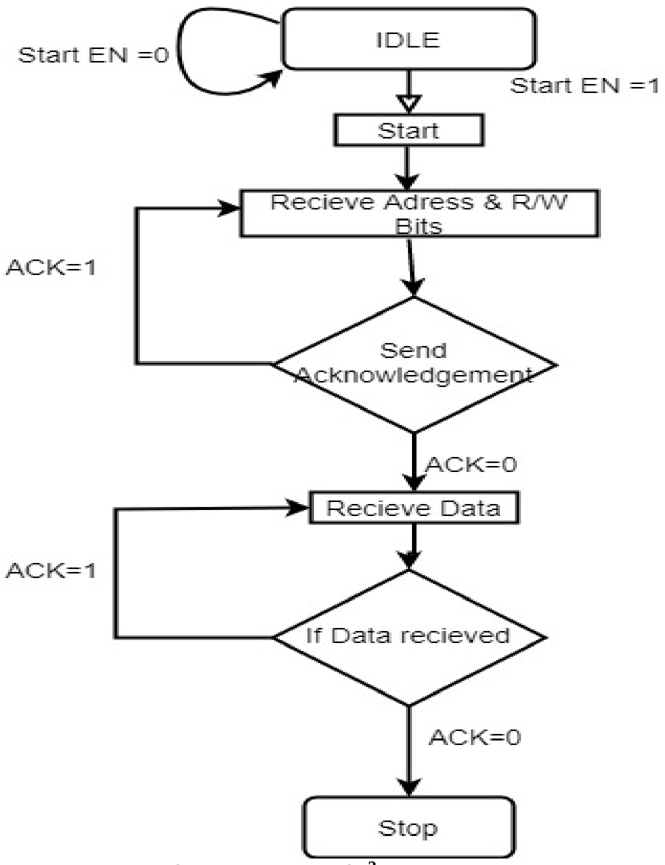


Fig. 6: Data flow diagram of I^{2}C Slave Communication

4. RESULTS AND DISCUSSION

Simulation and synthesis are carried out using Xilinx ISE 14.7.

4.1 SPI Protocol

4.1.1 Single Master-Slave Configuration: When the SS line is turned down to select the particular Slave, then as soon as the ss1 turns down the clock starts ticking of SCL line. Then with the first positive edge of SCL line read/write bit along with address is sent on the line.

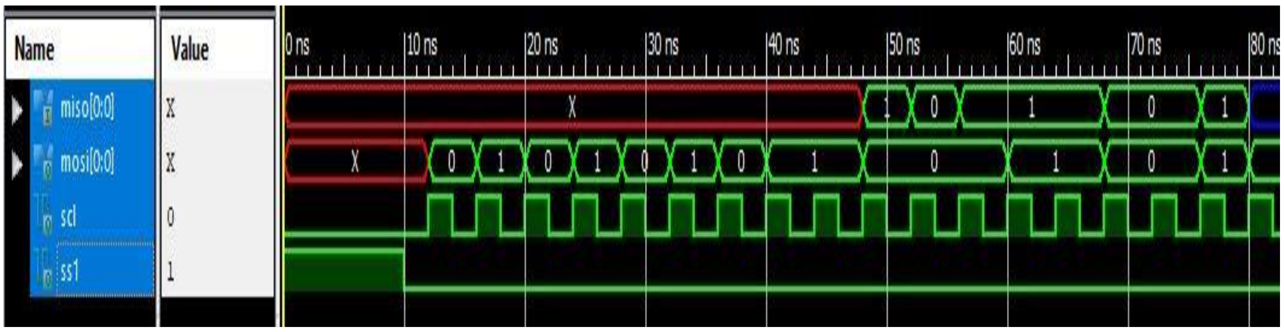


Fig. 7: **Single Master Slave Configuration read operation**

The MOSI lines remain null until the SCL line is high. Here 0 is read bit and 1 is write bit. In Figure 7 the first bit 0 is the read/write bit. Here it is read bit. The next eight bits are the address bits sent one bit at a time. Then data bits are sent one bit at a time. Here it is eight bits. The address bits are 10101011 and data bits are 00011001. When a Slave receives address bits and along with data bits it discards the data bits because it doesn't require data bits when we have read bit and it will be used only when we have write bit.

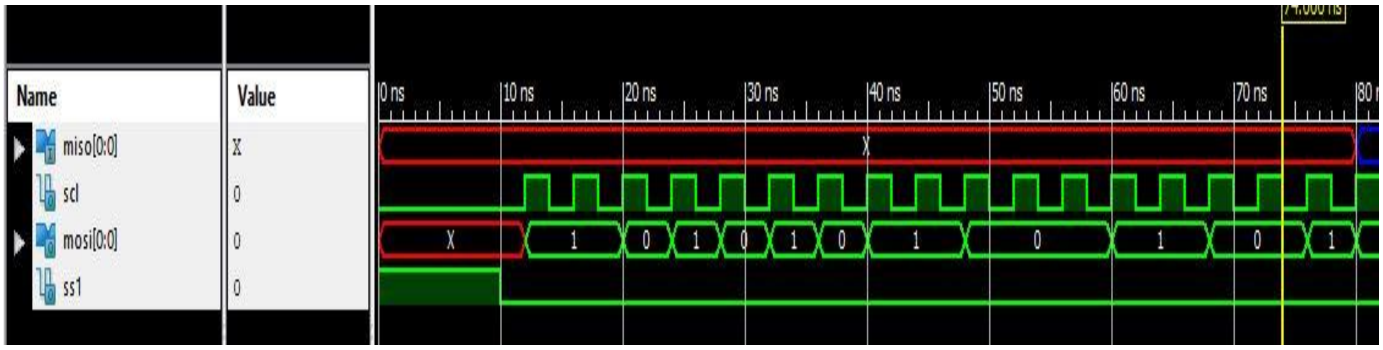
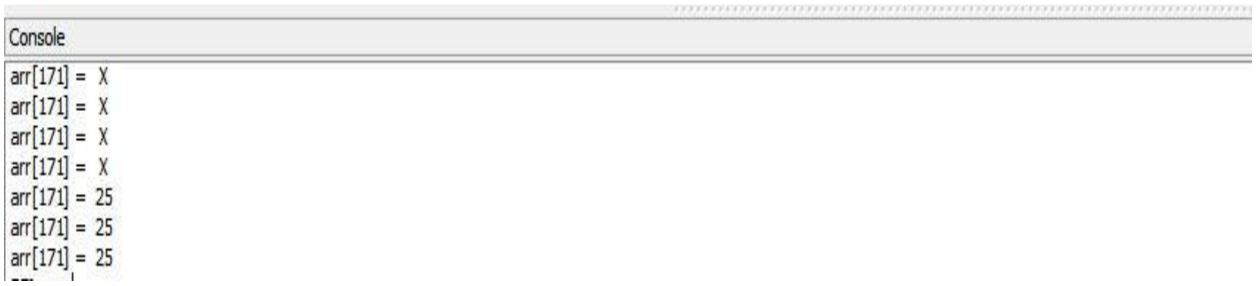


Fig. 8: Single Master Slave Configuration with write-operation

When read-bit is sent, MISO line will be activated with data stored in the address sent to the Slave. MISO line will be active when last bit of address received.



Console

```
arr[171] = X
arr[171] = X
arr[171] = X
arr[171] = X
arr[171] = 25
arr[171] = 25
arr[171] = 25
```

Fig. 9: Writing Data into Slave's memory

When write bit is sent, Slave will store the data in the address sent by the Master and it is as shown in Figure 8 and 9. The address and data in decimals are 171 and 25.

4.1.2 Single Master multiple Slave Configuration: A single Master drives four Slaves, the MOSI line of Master is connected through a 4:1 Mux to Slaves as shown in Figure 7.1.4. The operation is very similar to single Master Slave configuration except that it has multiple Slaves.

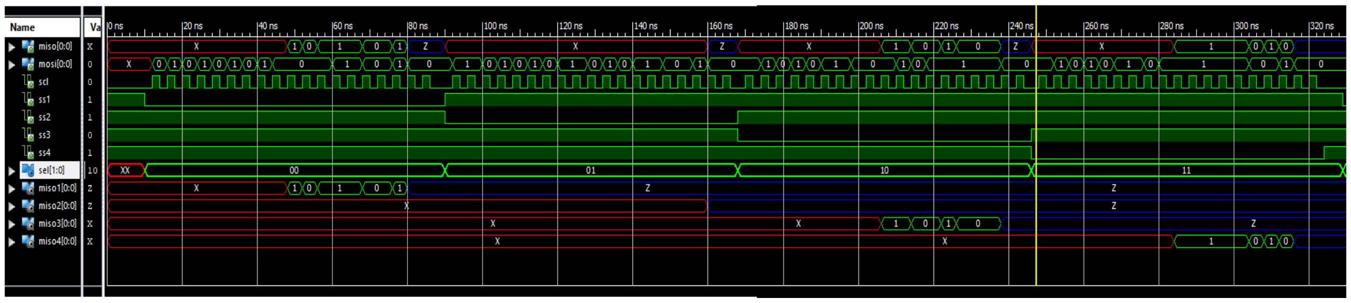
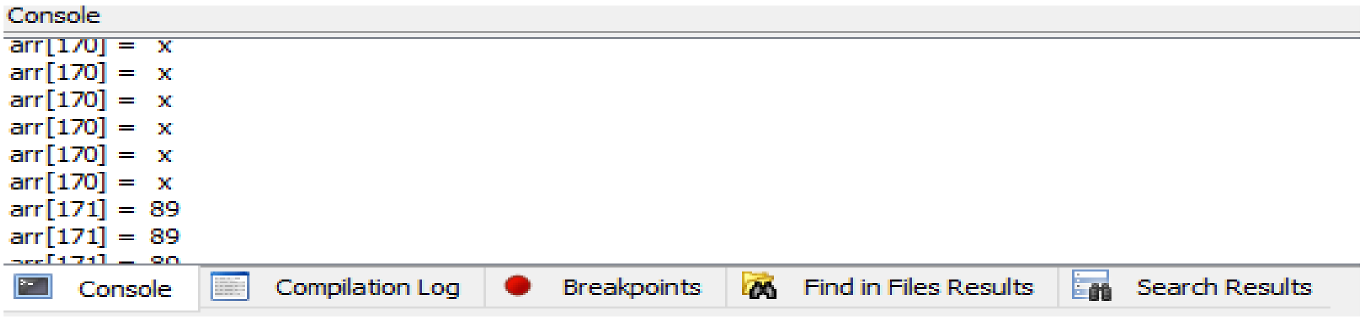


Fig. 10: Single Master multiple Slave configuration of read and write Operations

Slaves 1, 3 and 4 are used for read operation and Slave 2 used for write operation is as shown in Figure 10. The Data stored in Slave-2 is as shown in Figure 11.



```
arr[170] =  x
arr[170] =  x
arr[170] =  x
arr[170] =  x
arr[170] =  x
arr[170] =  x
arr[171] =  89
arr[171] =  89
arr[171] =  89
```

Fig. 11: Writing Data into second Slave's memory

4.2 I^{2}C Protocol

When SCL is High and SDA is at falling edge it starts the communication between I^{2}C Master and Slave. Then master sends 7-bit address 1101000 along with R/W bit '0' (Reading operation) on SDA line then Slave will respond back with an ACK bit '1' it indicates that the particular Slave is present.

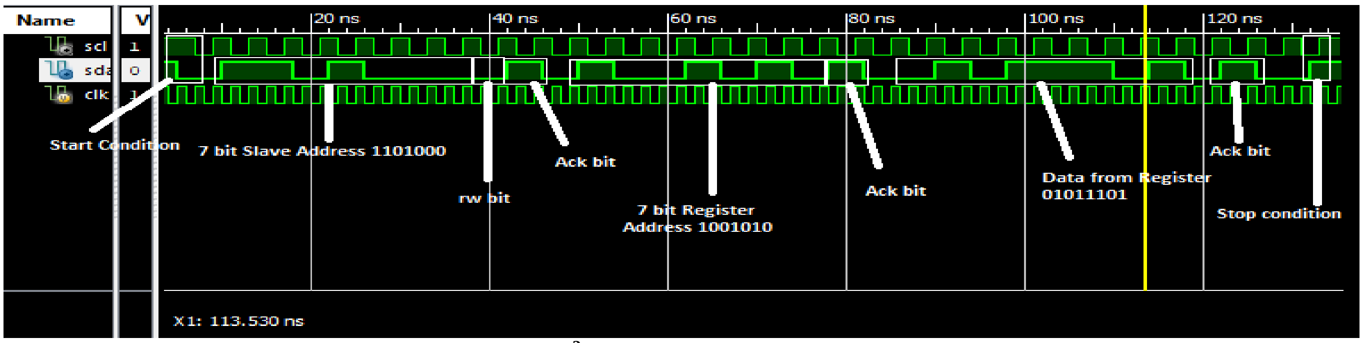


Fig. 12: Standard I^{2}C Protocol between Master and Slave

Master will send the address of the Slave's register i.e. 1001010, then Slave will respond by ACK bit '1' which represents that the address is present in the slave. Then Slave will send Data from received register i.e. 01011101 to Master, Then Master will respond with ACK bit '1' indicates that Master read the data successfully. Communication stops when SDA is at rising edge and SCL at High as shown in Figure 12.

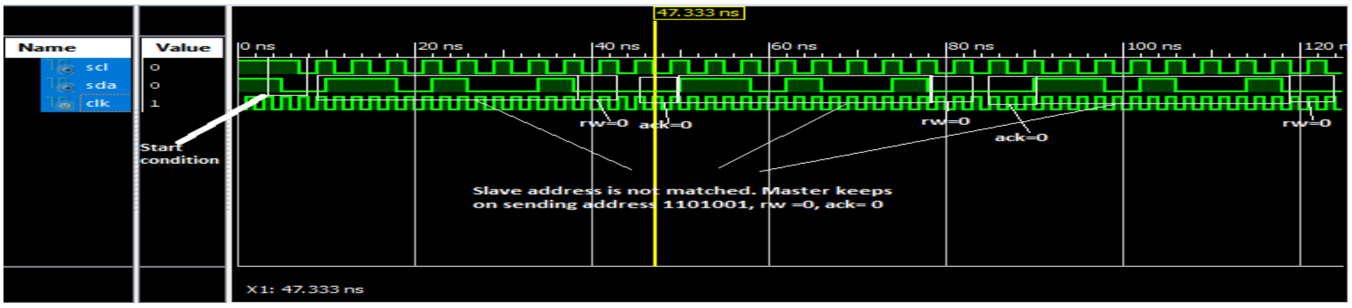


Fig. 13: Waveform of mismatch in address sent by the master

If the address that is send by master doesn't match with the slave then it will keep on sending the same address as shown in Figure 13.

5. CONCLUSION

This paper presents the simulation of SPI and I^{2}C protocol communication. Codes are written in Verilog, simulation and synthesis are carried out in Xilinxs ISE 14.7 and SPI and I^{2}C protocols are industry standard protocols, which are discussed in detail. FSM implementation of SPI and I^{2}C are explained.

6. REFERENCES

[1] Mankar, Jayant, ChaitaliDarode, KomalTrivedi, MadhuraKanoje, and PrachiShahare. "Review of I2C protocol." *International Journal of Research in Advent Technology* 2, no. 1 (2014).

[2] Pandey, Vivek Kumar, Sparsh Kumar, Vimal Kumar, and PankajGoel. "A Review Paper on I2C Communication Protocol." *International Journal of Advance Research, Ideas and Innovations in Technology* 4, no. 2 (2018): 340-343.

[3] Leal-del Río, Tatiana, Gustavo Juarez-Gracia, and L. NoéOliva-Moreno. "Implementation of the communication protocols SPI and I2C using a FPGA by the HDL-Verilog language." *Research in Computing Science* (2014): 31-41.

[4] Oudjida, AbdelkrimKamel, Ahmed Liacha, D. Benamrouche, M. Goudjil, RachidTiar, and A. Ouchabane. "Universal Low/Medium Speed I^{2}C-Slave Transceiver: A Detailed FPGA Implementation." *Journal of Circuits, Systems, and Computers* 17, no. 04 (2008): 611-626.

[5] Ghanekar, Aviraj, BrajKishor, and SachinBandewar. "Design and Implementation of SPI Bus Protocol." *International Journal*

of Advanced Research in Electrical, Electronics and Instrumentation Engineering (IJAREEIE) 5, no. 5 (2016): 4155-4157.

[6] Singh, Jai Karan, MukeshTiwari, and Vishal Sharma. "Design and Implementation of I2c master controller on FPGA using VHDL." *IJET* 4, no. 4 (2012).

[7] Oudjida, AbdelkrimKamel, Mohamed LamineBerrandjia, RachidTiar, Ahmed Liacha, and K. Tahraoui. "FPGA implementation of i^{2}c &spi protocols: A comparative study." In *2009 16th IEEE International Conference on Electronics, Circuits and Systems-(ICECS 2009)*, pp. 507-510. IEEE, 2009.

[8] Leens, Frédéric. "An introduction to I2C and SPI protocols." in 2009 *IEEE Instrumentation & Measurement Magazine*

[9] Oudjida, Abdelkrim K., AbdelghaniOuchabane, and Marco Liem. "Master-slave wrapper communication protocol: A case study." In *Proceedings of the 1st IEEE international computer systems and information technology conference ICSIT*, vol. 5, pp. 461-467. 2006.