DESIGN AND SIMULATION OF I2C PROTOCOL

[1]Ankit Kumar Yadav, [2]Prof. Usha Mehta
Institute of Technology Nirma University, Ahmdabad, Gujrat, India[1,2]
19mecv04@nirmauni.ac.in[1], usha.mehta@nirmauni.ac.in[2]

ABSTRACT

With developing technology, complex electronic circuits are being developed. These electronic components are integrated on a single chip known as a system on chip (SoC). Different electronic components need to communicate with each other for transferring information. To decrease the further complexity by decreasing interconnects we can use serial communication which requires less hardware as information is passed serially on a single line for transferring information between different electronic components for this we will use I2C protocol. I2C protocol works on serial communication, it has only two bus one is for serial clock(SCL) and the other one is for serial data(SDA). It is used for connecting high-speed peripherals to low-speed peripherals. Verilog Code has been written for design FSM, clock divider, data storage then RTL schematic has been generated. Through Simulation proper functionality has been checked for the reading of data from slave 1, slave 2, slave 3 to check the operation of a single master and multiple slaves.

Keywords—SDA, SCL, I2C, SPI, UART, HDL, RTL

I. INTRODUCTION

Philips Semiconductors introduced the I2C Protocol for communication between High-speed Peripherals with Low-speed Peripherals which includes Keyboard, LCD, EEPROM and connects them with high speed peripherls like- microprocessor, FPGA, microcontroller etc.

Mode of I2C:

a) low which runs at 100 Kbps

b) Fast which runs at 400 Kbps

c) High speed which runs at 3.4 Mbps

We use I2C protocol in electronics product as it uses only two buses Data bus and Clock bus which reduces the hardware complexity.It also has advantage of using Multiple master with multiple slaves which helps us to do the communication between large number of devices. The basic operation of I2C is widely used bidirectional interface that uses master and slave for their operation. Master read or write the data to the slave on getting acknowledgement for particular address. Each device whether master or slave has specific address for their unique identification. I2C interface consist of only serial clock(SCL) and serial data(SDA) lines. Specification.

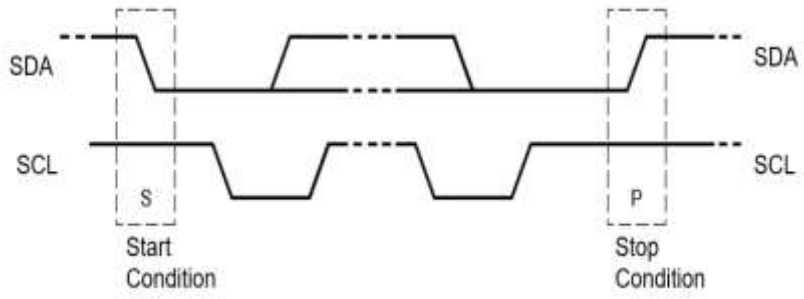
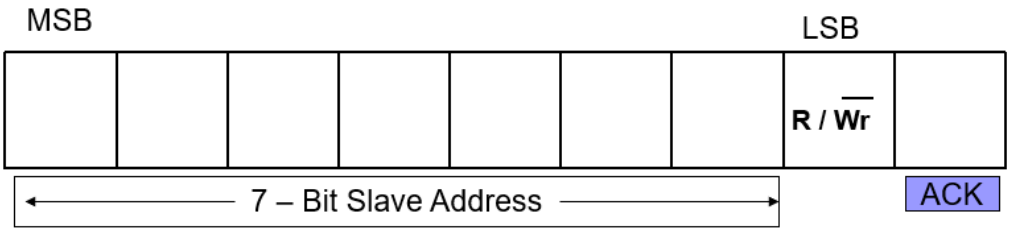


Fig.1 Start Condition and Stop Condition

When data line move from high to low or low to high level during high level of clock line is start and stop condition respectively. Start and stop conditions are generated by the master to show the start and end of the operation respectively.The bus is considered busy after a start condition, until a stop condition occurs which there is a one-to-one correspondence between the inputs and the outputs.

II. DATA TRANSFER BYTE



R/Wr

0 – Slave written to by Master

1 – Slave read by Master

ACK – Generated by the slave whose address has been output.

Fig.2 Data Transfer Byte

III. ACKNOWLEDGEMENT

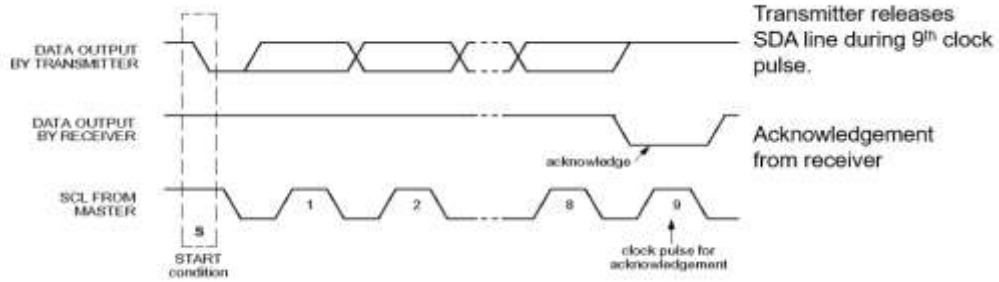


Fig.3 Acknowledgement

Master/slave receivers pull data line low for one clock pulse after reception of a byte. Master receiver leaves data line high after receipt of the last byte requested. Whenever receiver of the slave has data line high on the byte following the last byte it can accept

IV. NEGATIVE ACKNOWLEDGEMENT

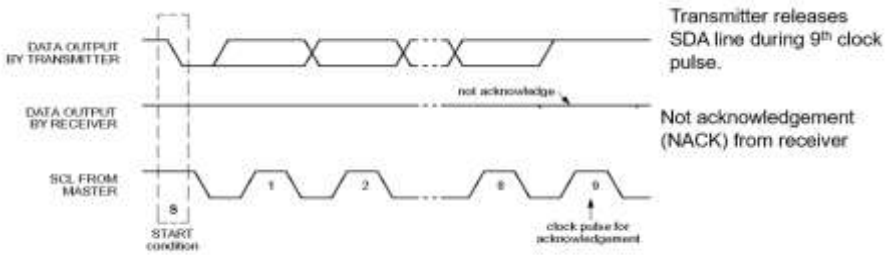


Fig.4 Negative acknowledgement

Data line will be high for duration of one clock cycle after receiving a byte.

V. COMPLETE OPERATION ON I2C BUS

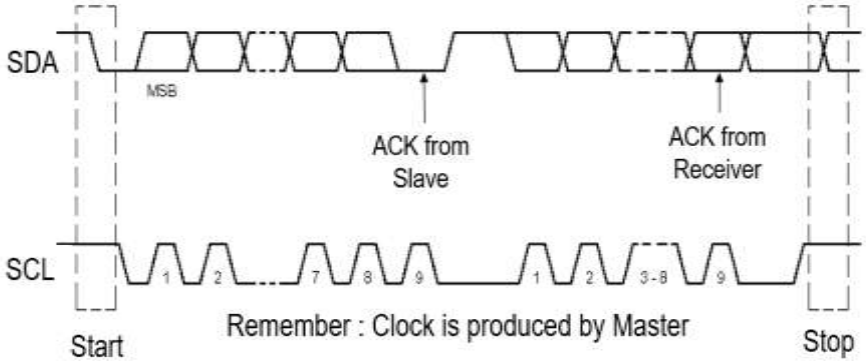


Fig.5 Complete operation on I2C Bus

It consist of Start Condition followed by Slave address with Read or Write then slave acknowledges with ACK. All Data bytes will be transferred each will be followed by ACK.After Completion of operation which will be followed by Stop Condition.

VI. DATA FORMAT

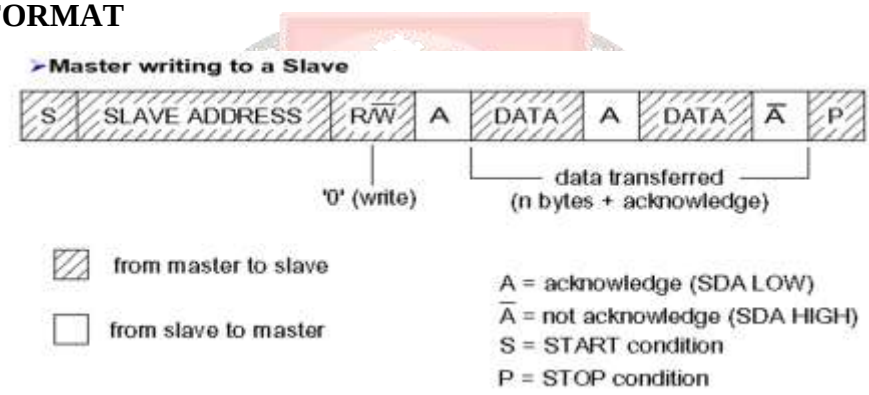


Fig.6 Data Format

Here we can distinguish the operation done by master to slave and slave to master. Shaded part represents the operation done from master to slave. Unshaded part represents the operation done from slave to master.

BLOCK DIAGRAM

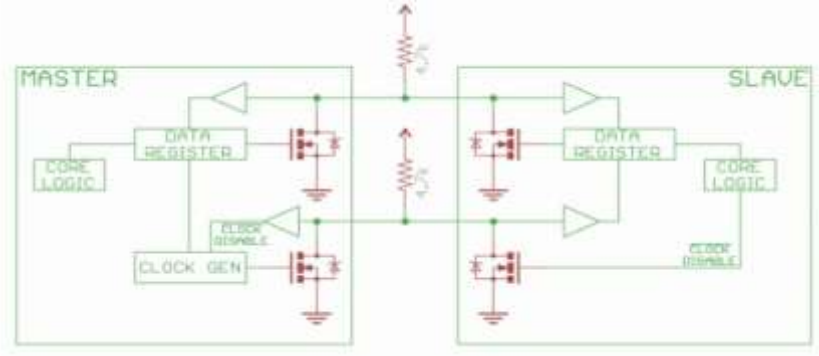


Fig.7 Block Diagram

Master/Slave consist of :

1. Core Logic: Core Logic in Master/Slave design is Finite State Machine (FSM) which has states of Start, Stop, Data, Address, Acknowledge etc

2. Data Register: Data Register in Master/Slave design is Fifo which helps to store data and address into the register.

3. Clock Gen: Clock generator in Master/Slave design is clock divider module which helps to work design at particular frequency.

FSM DESIGN

Ready condition: I2C bus will not perform any operation but will be ready for operation given that enable will be high voltage level.

Start condition: To start the operation SDA line switches from high to low value during SCL remains high or before it moves from high to low value, after this start condition slaves will be get activated for receiving address bits.

Address state: Address bits comprises of 7 bits which represents the address information of the devices which are connected like slave to which master wants to read/write the data. Comparison will be done to different address of slaves and given address.

Acknowledge(ACK) state: Comparison will be done between given address and the slave address, If that will be equal to the any slave address then ACK bit will be switch to low value, else high.

Write state: If Read/Write bit is sent to low then Write operation is performed. Data bit will consist of 8 bits and will be written to the particular slave where master wants to send the data.

Read state: If Read/Write bit is sent to high then Read operation is performed. Data bit will consist of 8 bits and will be read by the master from the particular slave where master wants to send the data.

Stop condition: To stop the operation master will move the SDA line from low to high level during the high value of SCL or before the SCL line switches from high voltage level to low voltage level.

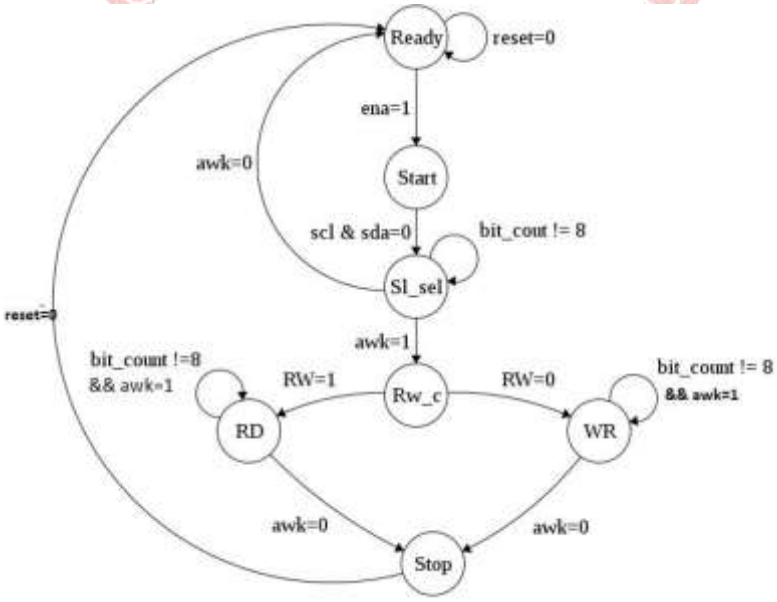


Fig.8 State Diagram

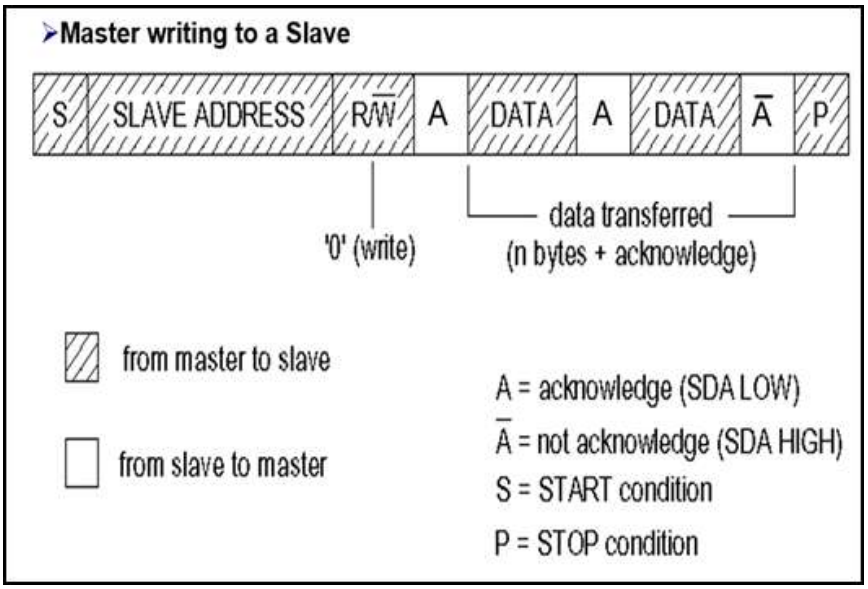


Fig.9 Complete Transfer

STEPS OF DATA TRANSFER:

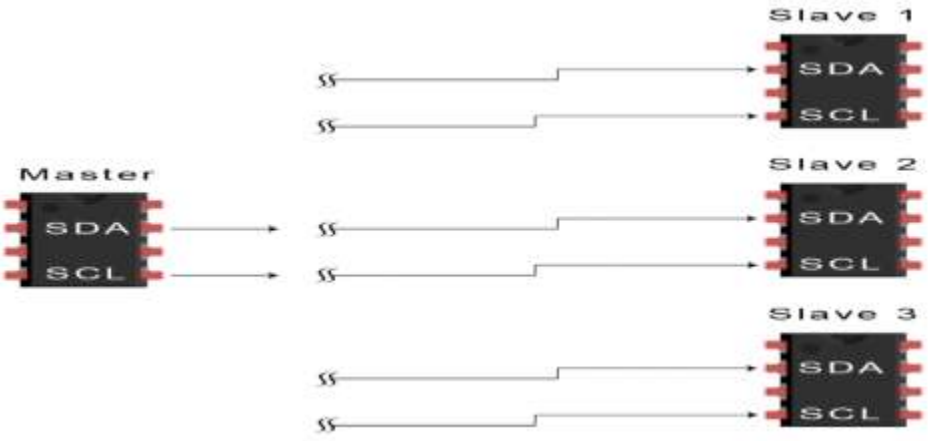


Fig.10 Step 1

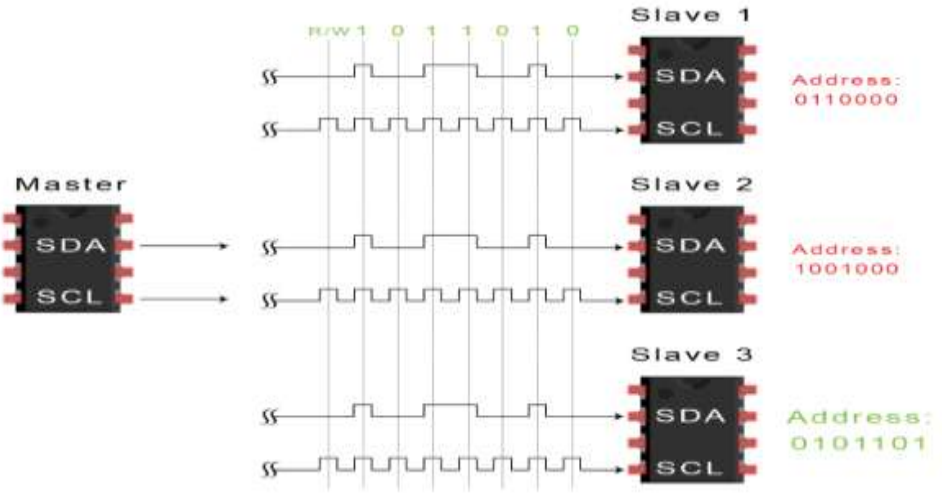


Fig.11 Step 2

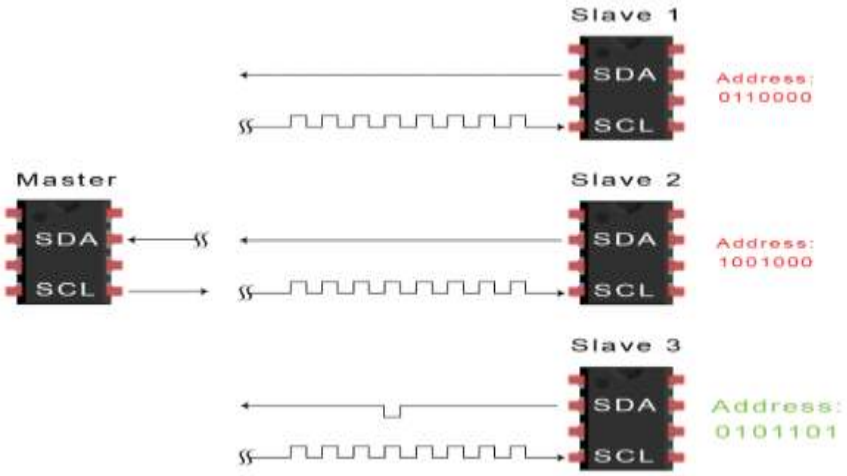


Fig.12 Step 3

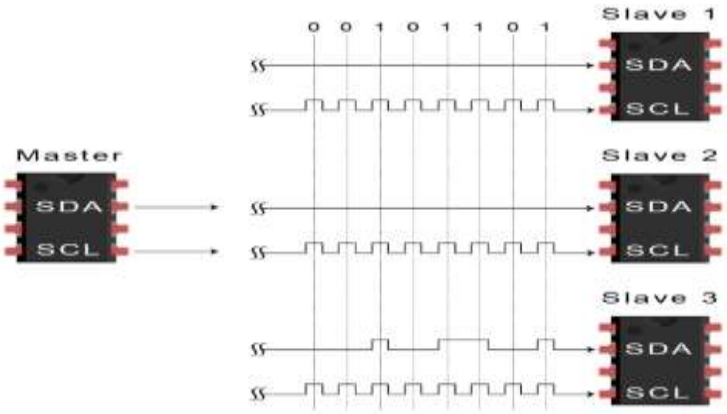


Fig.13 Step 4

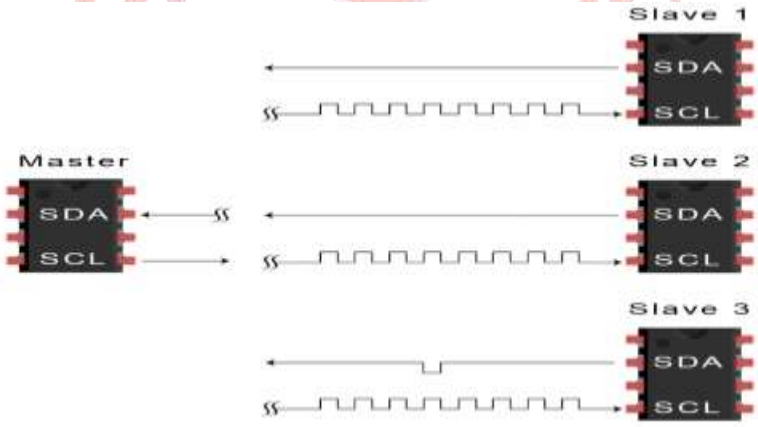


Fig.14 Step 5

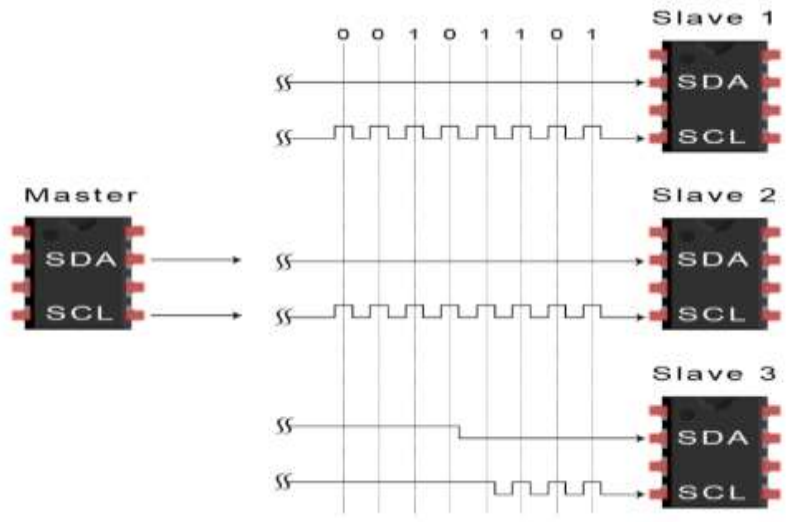


Fig.15 Step 6

Step 1: Master send the start bit which we can see from fig. of step 1 on serial data line (SDA) by switching from high to low level before Serial Clock line(SCL) switches from low to high level.

Step 2: Master sends the address on the Serial Data line (SDA) which will be received by all the three slaves connected

Step 3: All slave will compare the address which the master had sent if it will match then that particular slave will give negative acknowledgement on the serial data line (SDA).

Step 4: After receiving acknowledgement, master will send data bits on the Serial Data line(SDA) to that particular slave.

Step 5: After completion of the data bits slave will send the acknowledgement for ending of the data transmission.

Step 6: Master will send the Stop bit on Serial data line (SDA) to show the end of the operation by switching from low to high voltage level before the Serial clock line(SCL) switches from high to low level.

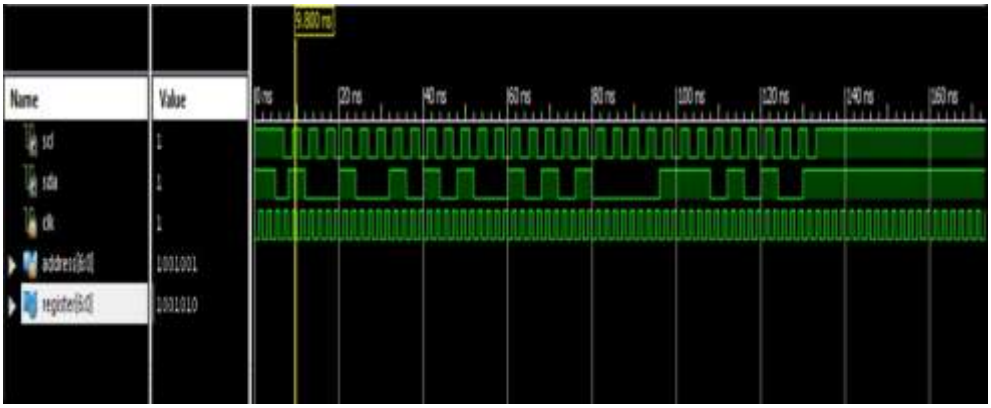


Fig.16 Simulation Waveform(a)

Master is reading data from slave1. Address of slave1 is 1001001 which is sent by the master and acknowledged by slave1 then master sent the register address to slave1 for storing data, then slave1 sent the data which is 00011101 which is read by Master.

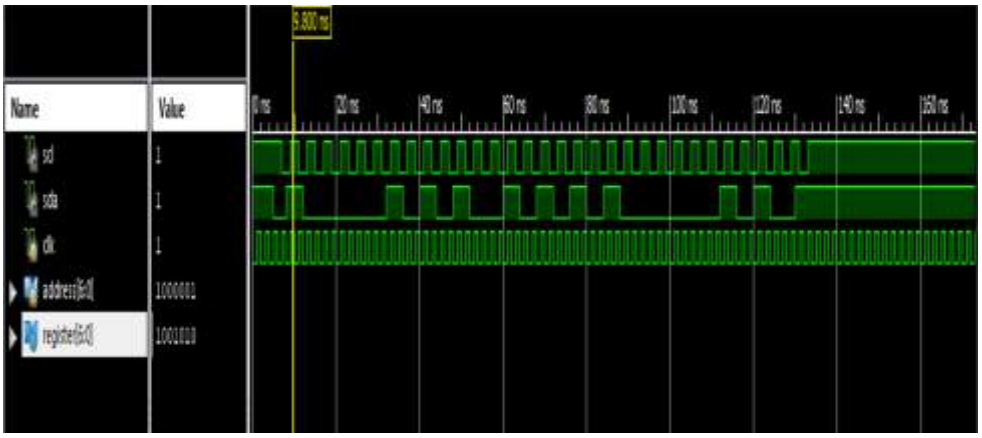


Fig.17 Simulation Waveform(b)

Master is reading data from slave2. Address of slave2 is 1000001 which is sent by the master and acknowledged by slave2 then master sent the register address to slave2 for storing data, then slave2 sent the data which is 10000001 which is read by Master.

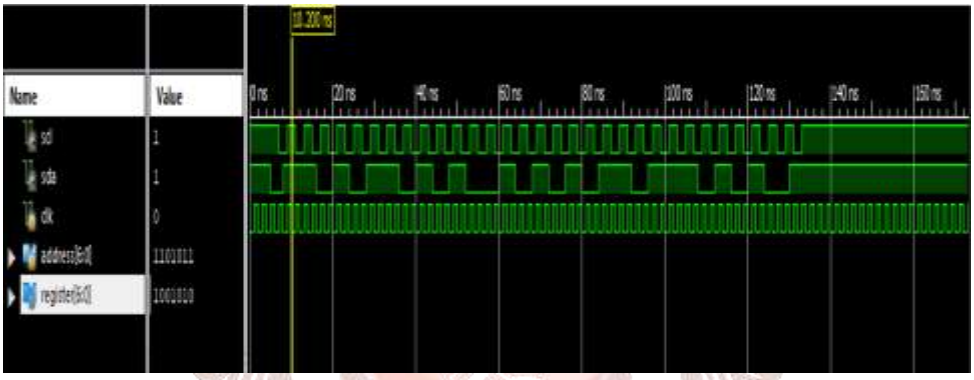


Fig.18 Simulation Waveform(c)

RTL SCHEMATIC:

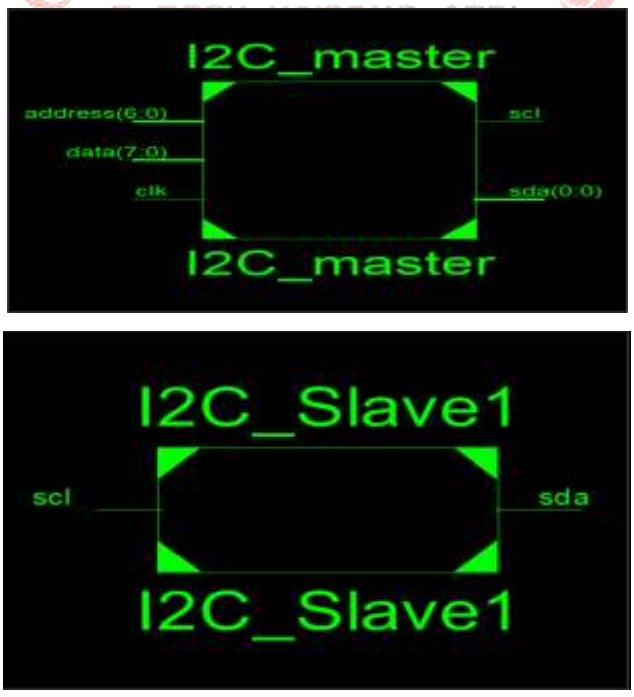


Fig.19 RTL Schematic

POWER CONSUMPTION:

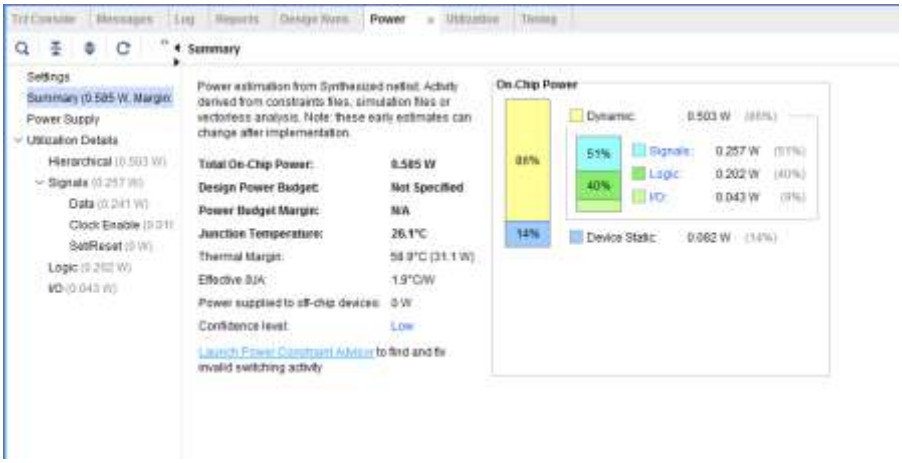


Fig.20 Power Consumption

Here Total On-Chip Power is 0.585 W

Dynamic Power = Signal + Logic + I/O = 0.257+0.202+0.043 = 0.503 W

Static Power = 0.082 W

Total Power = Dynamic Power + Static Power = 0.503 + 0.082 = 0.585 W

VII. CONCLUSION

I2C Master and Slave has been designed using Verilog HDL on Xilinx ISE and simulation has been carried out on Xilinx ISIM for checking the proper functionality of design. This Project has been carried out for Single Master and multiple slaves. In this we have successfully able to read the data from slaves according to the input address. This design is consuming Total On-Chip Power consumption of 0.585 W which has been find out using Xilinx Power Estimator Tool.

More work can be done on this project by extending this project for design of I2C multiple masters and multiple slaves, which helps us to read/write the data from multiple masters which can guide multiple slaves that works on bit arbitration technique.

REFERENCES

[1] Philips Semiconductors, "The I2C-Bus Specifications", version 2.1, January 2000.

[2] P.Venkateswaran, Madhumita Mukherjee, Arindam Sanyal, Snehasish Das and R.Nandi, "Design and Implementation of FPGA Based Interface Model for Scale-Free Network using I2C Bus Protocol on Quartus II 6.0", International Conference on Devices for communication 2009.

[3] BollamEswari, N.Ponmagal, K.Preethi, S.G.Sreejeesh "Implementation of I2C Master Bus Controller on FPGA", International conference on Communication and Signal Processing,pp,1113-1116, 2013.

[4] A. K. Oudjida, M. L. Berrandjia, R. Tiar, A. Liacha, K. Tahraoui, "FPGA Implementation of I2C & SPI Protocols:a Comparative Study", 16th IEEE International Conference on Electronics, Circuits, and Systems, December 2009, pp. 507-510.

[5] Ramesh Bhakthavatchalu, Deepthy G R, Vidhya S and Nisha V, "Design and Analysis of Low power Open Core Protocol Compliant Interface using VHDL", International Conference on Emerging Trends in Electrical and Computer Technology,23-24 March,2011

[6] Steve Golson, "State machine design techniques for Verilog and VHDL", Synopsys Users Group Conference,1994

[7] Wolfgang Grieskamp, Yuri Gurevich, Wolfram Schulte and Margus Veanes, "Generating Finite State Machines from Abstract State Machines", International Symposium on Software Testing and Analysis, July 2002, Vol.27, No. 4, pp. 112-122.

[8] Bijoy Kumar Upadhyaya and Salil Kumar Sanyal, "Design of A Novel FSM based Reconfigurable Multimode Interleaver for WLAN Application", International Conferences on Devices and Communications,24-25 February 2011

[9] Nrusingh Prasad Dash, Ranjan Dasguptay, Jayakar Chepadaz and Arindam Halder, "Event Driven Programming for Embedded Systems - A Finite State Machine Based Approach", The Sixth International Conference on Systems, 23-28 January, 2011

[10] Samir Palnitkar, Verilog HDL-A Guide to Digital Design and Synthesis, Sun Soft Press, 1996.