DESIGN OF DUAL MASTER I^{2}C BUS CONTROLLER

Ashwini s. Tadkal[1], **Padmapriya Patil**[2]

[1]*M Tech (Final year), VLSI, VTU Regional centre Gulbarga, Karnataka, India*
[2]*Assistant professor, Electronics and communication, PDA College Gulbarga, Karnataka, India*

Abstract

This paper presents an experimental design and implementation of serial data communication using I2C (Inter-Integrated Circuit) multi master and multi slave bus controller using a field programmable gate array (FPGA). The I2C master bus controller was interfaced with slave. This module was designed in Verilog HDL and simulated in Modelsim 10.1c. The design was synthesized using Xilinx ISE Design Suite 14.2. I2C master initiates data transmission and in order slave responds to it. It can be used to interface low speed peripherals like motherboard, embedded system, mobile phones, set top boxes, DVD, PDA's or other electronic devices.

Keywords – *FPGA, I^{2}C, master, Modelsim, serial data communication, slave, Spartan 3AN, Xilinx.*

---***---

1. INTRODUCTION

In the world of serial data communication [5], there are protocols like RS-232, RS-422, RS-485, SPI (Serial peripheral interface), Micro wire for interfacing high speed and low speed peripherals. These protocols require more pin connection in the IC(Integrated Circuit) for serial data communication to take place, as the physical size of IC have decreased over the years, we require less amount of pin connection for serial data transfer. USB/SPI/Microwire and mostly UARTS are all just one point to one point' data transfer bus systems. They use multiplexing of the data path and forwarding of messages to service multiple devices. To overcome this problem, the I^{2}C [1] protocol was introduced by Phillips which requires only two lines for communication with two or more chips and can control a network of device chips with just a two general purpose I/O pins whereas, other bus protocols require more pins and signals to connect devices.

In this project, we are implementing multi master and multi slave I ^{2}C bus protocol for interfacing low speed peripheral devices on FPGA It is also the best bus for the control applications, where devices may have to be added or removed from the system. I^{2}C protocol can also be used for communication between multiple circuit boards in equipments with or without using a shielded cable depending on the distance and speed of data transfer. I^{2}C bus is a medium for communication where master controller [6] is used to send and receive data to and from the slave. The low speed peripheral, is interfaced with I^{2}C master bus and synthesized on Spartan 3AN.Fig-1 shows the I^{2}C bus system with multi master and multi slave.

When using multiple master Arbitration and clock stretching need to considered. Arbitration decides which master is going to rule the bus.
The synopsis of the paper is as follows: In section 2, we discussed I2C protocol of our proposed design which also presents module description for our proposed system. In section 3, we present the software implementation along with algorithm and flow chart.. Finally, concluded with future scaleup in section 4.

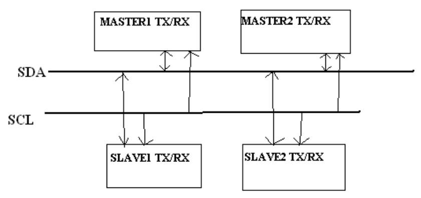

Fig-1 I/O Diagram of I^{2}C Multi Master and Multi Slave bus controller

2. PROPOSED WORK

2.1. I^{2}C Protocol

I^{2}C is a two wire, bidirectional serial bus that provides effective data communication between two devices. I^{2}C bus supports many devices and each device is recognized by its unique address.

The physical I^{2}C bus consists of just two wires, called SCL and SDA. SCL is the clock line. It is used to synchronize all data transfers over the I^{2}C bus. SDA is the data line. The SCL and SDA lines are connected to all devices on the I ^{2}C bus. As both SCL and SDA lines are "open drThe I^{2}C bus is said to be idle when both SCL and SDA are at logic 1 level. When the master (controller) wishes to transmit data to a slave (DS1307) it begins by issuing a start sequence on the I^{2}C bus, which is a high to low transition on the SDA line while the SCL line is high as shown in

Fig– 2(a).The bus is considered to be busy after the START condition. After the START cond

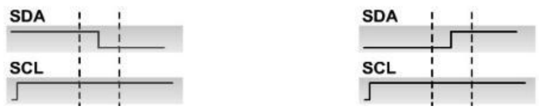

Fig- 2 (a) "**START"** Sequence (b) "**STOP"** Sequence

slave address is sent by the master. The slave device whose address matches the address that is being sent out by the master will respond with an acknowledgement bit on the SDA line by pulling the SDA line low. Data is transferred in sequences of 8 bits. The bits are placed on the SDA line starting with the MSB (Most Significant Bit). For every 8 bits transferred, the slave device receiving the data sends back an acknowledge bit, so there are actually 9 SCL clock pulses to transfer each 8 bit byte of data this is shown in Fig-3. If the receiving device sends back a low ACK bit, then it has received the data and is ready to accept another byte. If it sends back a high then it is indicating it cannot accept any further data and the master should terminate the transfer by sending a STOP sequence. In Fig-2(b) which shows the STOP sequence, where the SDA line is driven low while SCL line is high. This signals the end of the transaction with the slave device.

2.2 Serial Data Communication

The I^{2}C bus has two modes of operation: master transmitter and master receiver. The I^{2}C master bus initiates data transfer and can drive both SDA and SCL lines.[2] Slave device is addressed by the master. It can issue only data on the SDA line.

In master transmission mode, after the initiation of the START sequence, the master sends out a slave address. The address byte contains the 7 bit slave address, which is followed by the direction bit (R/ *w*). After receiving and decoding the address byte the device outputs

Acknowledge on the SDA line. After the slave device acknowledges the slave address + write bit, the master transmits a register address to the slave this will set the register pointer on the slave. The master will then begin transmitting each byte of data with the slave acknowledging each byte received. The master will generate a stop condition to terminate the data write.

In master receiver mode, the first byte is received and handled as in the master transmission mode. However, in this mode, the direction bit will indicate that the transfer direction is reversed. Serial data is transmitted on SDA by the slave while the serial clock is input on SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer (Fig-5). The address byte is the first byte received after the start condition is generated by the master. The address byte contains the 7-bit slave address, which is followed by the direction bit (R/ *w*). After receiving and decoding the address byte the device inputs acknowledge on the SDA line. The slave then begins to transmit data starting with the register address pointed to by the

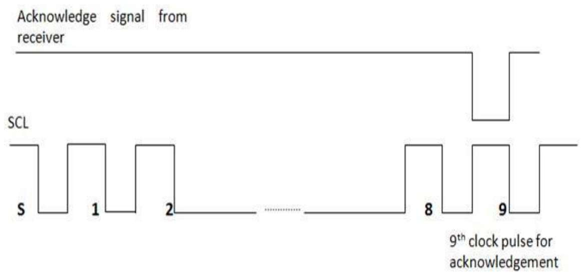
Acknowledge signal from receiver
SCL
S
1
2
8
9
9th clock pulse for acknowledgement

Fig-3 Acknowledgement on the I^{2}C Bus register

Pointer. If the register pointer is not written before the initiation of a read mode, the first address that is read is the last one stored in the register pointer. The slave must receive a "not acknowledged" to end a read.

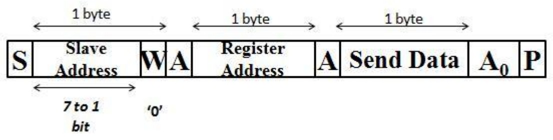

Fig -4 Master Transmission Mode

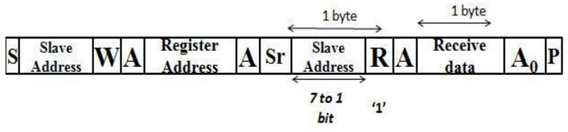

Fig- 5 Master Receiver Mode

The master transmission mode and master receiver mode is shown in above figure.

3. SOFTWARE IMPLEMENTATION

I^{2}C bus is a medium for communication where master controller is used to send or receive data to and from slave and it is developed by Philips.

I^{2}C master controller is designed using Verilog HDL [3] based on Finite State Machine (FSM) [8]. FSM is a sequential circuit that uses a finite number of states to keep track of its history of operations, and based on history of operation and current input, determines the next state. There are several states in obtaining the result.

Algorithm

State 1: An idle condition: I^2C bus doesn't perform any operation. (SCL and SDA remains high).
State 2: Start condition: master initiates data transmission by providing START (SCL is high and SDA is from high to low).
State 3: Slave address - write: master sends the slave address-write (11010000) to the slave.
State 4: If the slave address matches with the slave, it sends an acknowledgement bit in response to the master.
State 5: 8 Bit Register Address[7] will be transmitted to the slave. Again acknowledgement is sent to the master by the slave.
State 6: Data to be transmitted is sent to the slave by the master. After receiving the data, slave acknowledges the master.
State 7: Stop condition: Slave sends a stop bit to the master to terminate the communication (SCL is high and SDA is from Low to high).
State 8: Master transmits slave address for read operation to the slave.
State 9: Master receives the data from the slave and acknowledges the slave.
State 10: Master sends a STOP bit to terminate the connection (SCL is high and SDA is from Low to high).

For performing read operation, write operation is performed first and then read operation is done. Slave address for read is 11010001. (State 7 will not be performed for read operation)

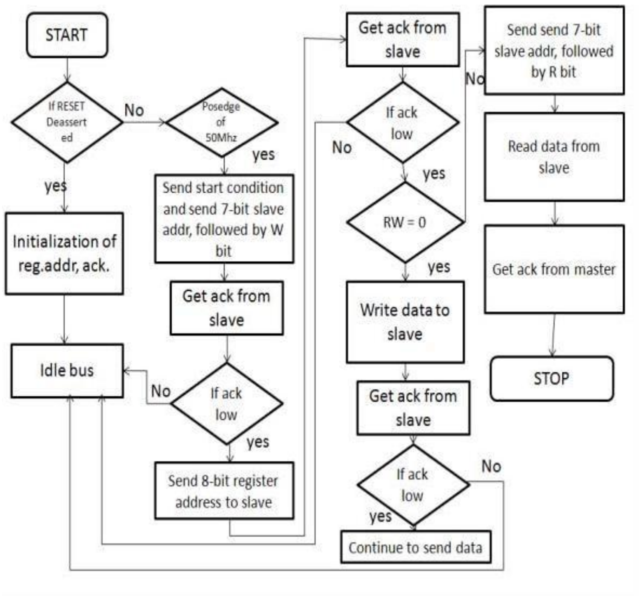

Fig 6: Flow chart of I^2C master bus controller.

Fig-8 shows the flowchart for I^2C master bus communication with slave device. Fig-9 shows the Modelsim simulation result for write operation, the given data input is written in to slave register address in each state of FSM programmed in Verilog HDL. Fig-10 shows the Modelsim simulation result for read operation, to read the written data from the slave the write operation takes place first followed by the repeated start condition and sending the slave address read (11010001) in each state of FSM.

The simulated Verilog coding is synthesized on Spartan 3AN through Xilinx ISE Design Suite14.2 [4]. The design is analysed using ChipScope Analyzer in Spartan 3AN platform.

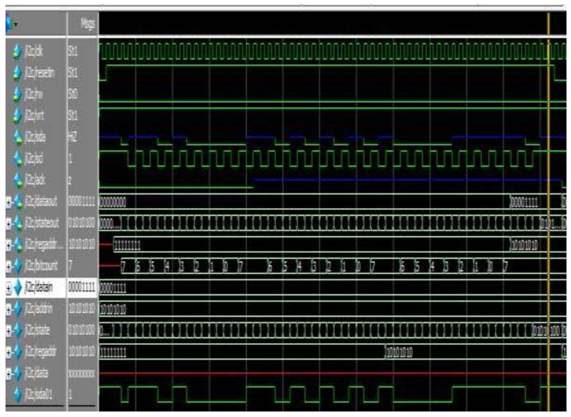

Fig-7.Modelsim Simulation Result for Write operation

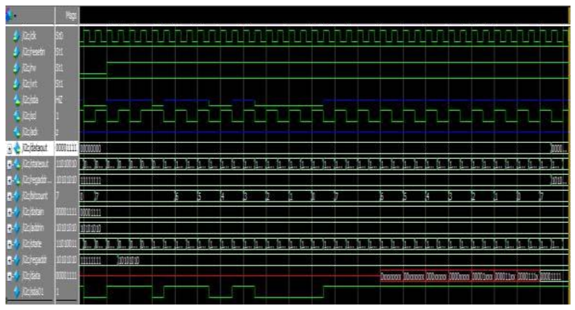

Fig-8 Modelsim Simulation Result for Read operation

After completing the coding in Verilog HDL, it is downloaded to the Spartan 3AN kit using Xilinx software. And corresponding input is given according to the I^2C protocol.

4. CONCLUSIONS

I^2C bus controller is successfully designed in verilog and simulated in ModelSIM. Simulation results verify that the communication has been established between the multi master and the multi slave.

REFERENCES

[1]. Philips Semiconductor "I2C Bus Specification" version 2 1, January 2000.
[2]. Raj kamal ,"Embedded system: Architectureprogramming and Design",Tata McGraw Hill,2008.
[3]. Stuart Sutherland, "Verilog® HDL Quick Reference Guide", IEEE Std 1364-2001.

[4]. Xilinx “Spartan-3A/3AN FPGA Starter Kit Board User Guide,”version 1.1,2008.
[5]. A.P.Godse, D.A.Godse, “Microprocessor, Microcontroller & Applications”Technical publications, 2008.
[6] .Vincent Himpe, “Mastering the I2C bus” Elektor Verlag publications, 2011.
[7]. M.Morris Mano, “Digital Design” EBSCO publishing.Inc., 2002.
[8]. Frank Vahid, “Digital Design with RTL Design, Verilog and VHDL”VP and Executive publisher, 2010.