International Journal of Emerging Engineering Research and Technology
Volume 3, Issue 8, August 2015, PP 125-129
ISSN 2349-4395 (Print) & ISSN 2349-4409 (Online)

Design of I2C BUS Controller using VHDL

[1]Spandana Sunku, [2]Lavanya Latha Mutyala

[1]*M.Tech, Audhisankara Institute of Technology, Gudur, India*
[2]*Assistant Professor, Dept of ECE, Audhisankara Institute of Technology, Gudur, India*

ABSTRACT

I2C bus Controllers have been an integral part of many SoCs in the industry and also have been logically implemented individually. For serial data communication with low performance peripherals we design I2C Controller. In this paper, I2C controller for Master and Slave Device controller. This module was designed in VHDL hardware descriptive language. This design was synthesized using Xilinx ISE design suit 13.2. The design can be used to interface low speed peripherals like set top boxes, DVD, or other electronic devices.

Keywords: I2C, FPGA, master, serial data communication, slave, Spartan 3E, Xilinx.

INTRODUCTION

In the world of serial data communication [6], there are protocols like RS-232, RS-422, RS-485, SPI (Serial peripheral interface), and 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 I2C 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 [11] whereas, other bus protocols require more pins and signals to connect devices.

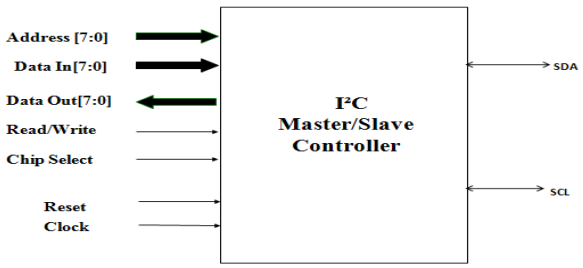

Fig1. *I/O Diagram of I^2C Controller*

In this project, we are implementing I2C bus protocol for interfacing low speed peripheral devices on FPGA [3]. It is also the best bus for the control applications, where devices may have to be added or removed from the system. I2C protocol [1] 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.I2C bus is a medium for communication where master controller [9] is used to send and receive data to and from the slave. The low speed peripheral, is interfaced with I2C master bus and synthesized on Spartan 3E. Fig.1 shows the I2C bus system with the I2C controller implemented on a FPGA.

Address for correspondence:

lavanayamutyala84@gmail.com

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 scale up in section 5.

PROPOSED WORK

I2C Protocol

I2C is a two wire, bidirectional serial bus that provides effective data communication between two devices. I2C bus supports many devices and each device is recognized by its unique address. In Fig.1 detain and addr_in is the 8 bit address given as an input. Clk and reset are the input lines used to initiate the bus controller process. The R/ w signal are given as an input to indicate whether master or slave acts as a transmitter in the data transmission.

Fig2. *(a) "START" Sequence. (b) "STOP" Sequence*

The physical I2C bus consists of just two wires, called SCLand SDA. SCL is the clock line; it is used to synchronize all data transfers over the I2C bus. SDA is the data line; the SCL and SDA lines are connected to all devices on the I2C bus. As both SCL and SDA lines are "open drain" drivers they are pulled up using pull up resistors.

The I2C bus is said to be idle when both SCL and SDA area logic 1 level. When the master (controller) wishes to transmit data to a slave. It begins by issuing a start sequence on the I2C 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 condition, 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.

Serial Data Communication

The I2C bus has two modes of operation [4]: master transmitter and master receiver. The I2C master bus initiates data transfer and can drive both SDA and SCL lines Slave device is addressed by the master. It can issue only data on the SDA line.

Fig3. *Acknowledgement on the I^2C Bus*

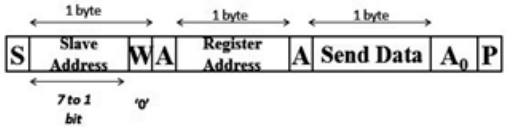

Fig4. *Master Transmission Mode*

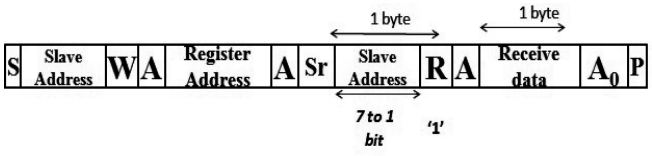

Fig5. *Master Receiver Mode*

In master transmissionn mode, after the initiation of the START sequence, the master sends out a slave address. The address byte contains the 7 bit [2] address, which is 1101000, followed by the direction bit (R/ w). After receiving and decoding the address byte the device outputs acknowledge on the SDA line. After the acknowledges the slave address + write bit, the master transmits a register address to the 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 address, which is 1101000, 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 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.

SLAVE Controlling

The I2c Controller supports a bi-directional, 2-wire bus and data transmission protocol. Master sending data at slave side receive the data byte and check the slave address and check the write or read control bit and perform the operation in slave registers. Get back the ack to master.

SOFTWARE IMPLEMENTATION

I2C master controller is designed using VHDL HDL [5] based on Finite State Machine (FSM). 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

State1: An idle condition: I2C bus doesn't perform any operation (SCL and SDA remains high).

State2: Start condition: master initiates data transmission by providing START (SCL is high and SDA is from high to low).

State3: Slave address - write: master sends the slave address- write (11010000) to the slave.

State4: If the slave address matches with the slave, it sends an acknowledgement bit in response to the master.

State5: 8 Bit Register Address will be transmitted to the slave. Again acknowledgement is sent to the master by the slave.

State6: Data to be transmitted is sent to the slave by the master. After receiving the data, slave acknowledges the master.

State7: Stop condition: Slave sends a stop bit to the master to terminate the communication (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 is11010001. (State 7 will not be performed for read operation)

State8: Master transmits slave address for read operation to the slave.

State9: Master receives the data from the slave and acknowledges the slave.

State10: Master sends a STOP bit to terminate the connection (SCL is high and SDA is from Low to high).

Fig.8 shows the flowchart for I2C master bus communication [8] 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 VHDL HDL [12]. F ig.10 shows the Xilinx ISim

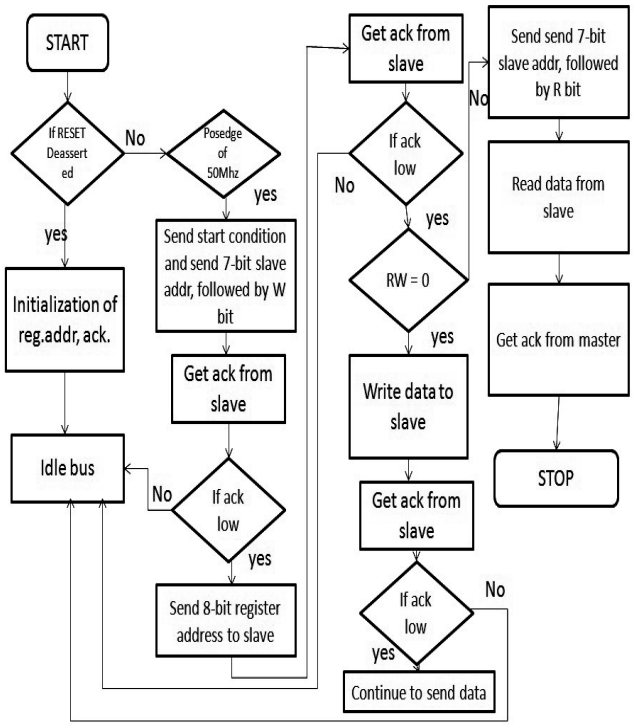

Fig8. *Flowchart for I2C bus communication*

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 VHDL coding is synthesized on Spartan 3E through Xilinx ISE Design Suite13.2 [7]. The design is analyzed using Chip Scope Analyzer in Spartan 3E platform.

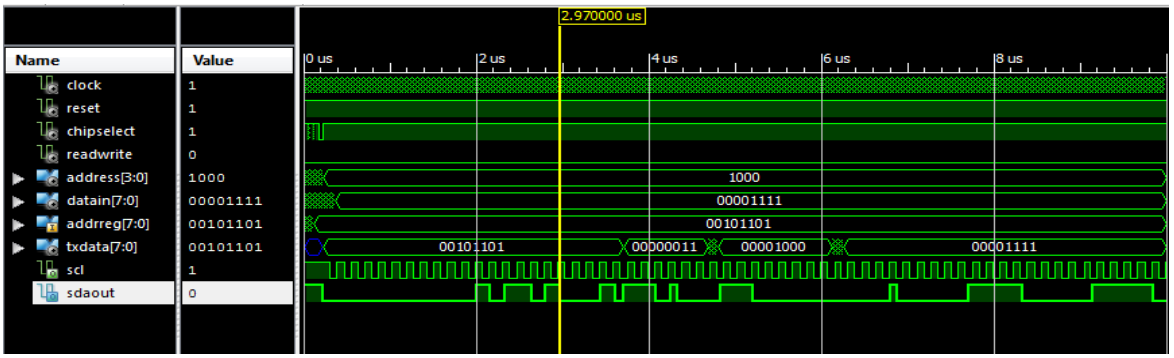

Fig9. *Simulation Result for Master transmitted data*

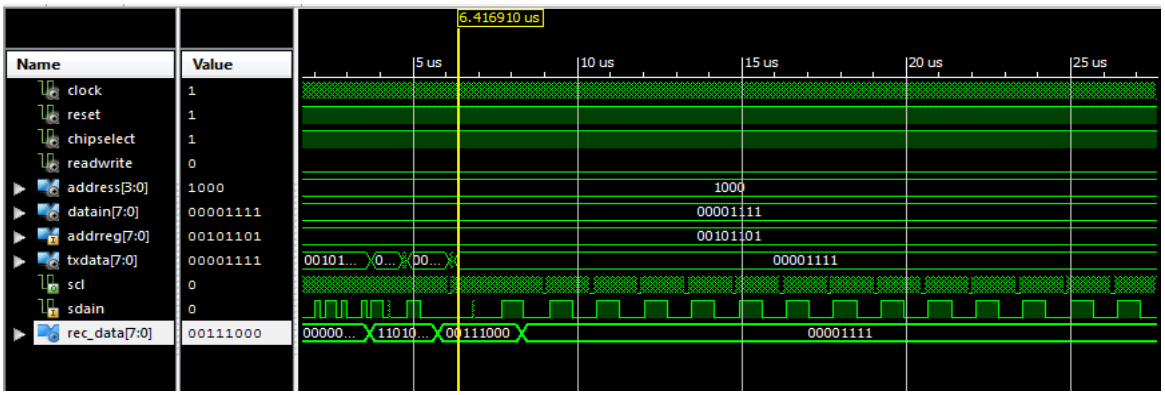

Fig10. *Simulation Result for Slave receive data*

CONCLUSION

This project demonstrates how I2C Controller (Master) transmits data and slave receiving. So that any low speed peripheral devices can be interfaced using I2C bus protocol as master/slave. In future, this can be implemented with multiple masters and multiple slaves to co-ordinate the entire system by clock synchronization techniques.

REFERENCES

[1] I2C Bus Specification, Philips Semiconductor, version 2.1, January2000.

[2] DS1307 64 x 8, Serial, I2C Real Time Clock, and Maxim integrated, 2008.

[3] Prof. Jai Karan Singh et al "Design and Implementation of I2C master controller on FPGA using VHDL," IJET, Aug-Sep 2012.

[4] Raj Kamal, "Devices and Communication Buses for Devices Network," in Embedded system: Architecture programming and Design, Shalini Jha Ed. New Delhi, India: Tata McGraw-Hill Education, 2008, pp.160-165.

[5] Verilog® HDL Quick Reference Guide, IEEE Standard 1364-2001.

[6] Tim Wilmshurst, "Starting with Serial," in Designing Embedded Systems with PIC Microcontrollers: Principles and Applications, 2ndEd.Burlinton: Newnes, 2009, pp.307-327.

[7] Spartan-3A/3AN FPGA Starter Kit Board User Guide, Xilinx, version1.1, 2008.

[8] A.P. Godse, D.A.Godse, "Bus Standards," in Microprocessors and its Applications, 3rd Ed. Pune, India: Technical publications, 2008.

[9] Vincent Himpe, "Historical background of I2C," in Mastering the I2C Bus, Aachen, Germany: Elektor Verilog publications, 2011.

[10] Pong P.Chu, "I/O Modules," in FPGA Prototyping by Verilog Examples: Xilinx Spartan – 3 Versions, New Delhi, India: Wiley, 2008.

[11] AN10216-01 I2C Manual, Philips Semiconductor, March 2003.

[12] Frank Vahid, "Hardware Description Language," in Digital Design with RTL Design, Verilog and VHDL, 2nd Ed. Katie Singleton Ed. Hoboken, New Jersey: VP and Executive publisher, 2010, pp 487-532.

AUTHORS' BIOGRAPHY

Spandana Sunku received the B.Eng. degree in electronics and communication engineering in Visvodaya engineering college, Kavali Affiliated by Jawaharlal Nehru Technological University Ananthapur with Disinction in 2013 and Pursuing M.tech. Degree in VLSI in Audhisankara Institute of Technology, Gudur affiliated by Jawaharlal Nehru Technological University Ananthapur. Area of interest: VLSI.

Lavanya latha Mutyala received the B.Eng. degree in electronics and communication engineering in Narayana engineering college, Gudur Affiliated by Jawaharlal Nehru Technological University Hyderabad with Disinction in 2007 and the M.tech. Degree in Digital Systems Computer Electronics engineering in P.B.R Visvodaya Institute Of Technology, Kavali Affiliated by University of Jawaharlal Nehru Technological University Ananthapur in 2013. Present working as assistant professor in Audhisankara Institute of Technology, Gudur.