International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 10 (2015) pp. 25563-25570

http://www.ripublication.com

I^2C Master Bus Controller Implementation on FPGA with RAM as Slave

Ch Monica[1], Dr. K Hari Kishore[2], B.Praveen Kitti[3]

1. M.Tech VLSI Student, Department of ECE,
KL University, monicachatla@gmail.com
2. Professor, VLSI Research Group, Department of ECE, KL University.
3.Asst. Professor, Department of ECE, PSCMR College of Engineering.
praveenkitty17@gmail.com

Abstract

This paper implements serial data communicationusing I^2C (Inter-Integrated Circuit) master controller using a field programmable gate array (FPGA).Itprovides a designer a foundation from which the user can customize the I²C Master Controller to meet a particular design requirement.The aim of this project is to design the I²C Master Controller with slave as RAM and its individual blocks are designed using VHDL. The coding is done in VHDL which is compiled and verified by test bench in modelsim. They are simulated and synthesized by using theXilinx design suite 14.2 and optimised for area and power.I^2C master initiates data transmission and inorder slave responds to it. It can be used to interface low speed peripherals like motherboard, embedded and other electronic devices.

Index Terms: I^2C, master, slave, Modelsim, serial data communication, Spartan 3AN, Xilinx.

I. INTRODUCTION

In the world of multiple application based products it is very essential to have a multiple connections to a system, which includes peripherals following different communication protocols as well. Today's world of serial data communication, there are many 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 connections 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. Mostly UARTS are all just 'point to point' data transfer bus systems, to service multiple devices they use multiplexing of the data path and forwarding of messages. To overcome this problem, the I^2C protocolwas introduced by Phillips that requires only two lines for communication with two or more devices whereas, other bus protocols require more pins and signals to connect devices.

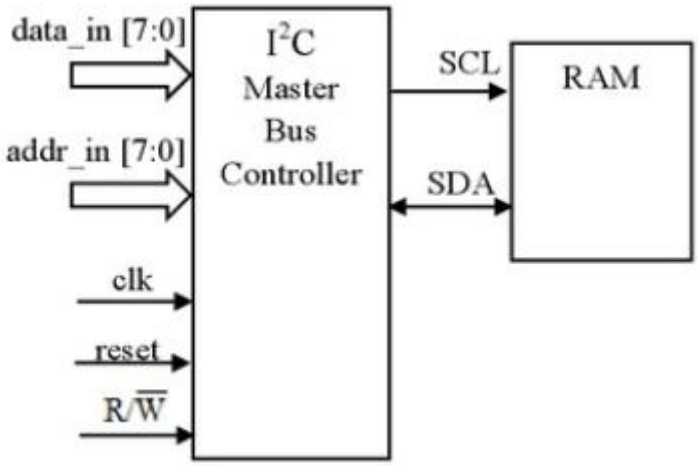

Fig1. I/O Diagram of I^2C Master Controller interfaced with RAM as slave device.

In this undertaking, we are actualizing I^2C bus protocol for interfacing low speed peripheral devices on FPGA. Contingent upon the separation and pace of information exchange. I^2C protocol can likewise be utilized for correspondence between numerous circuit with or without utilizing a protected link. I^2C bus is a medium for communication where master controller is utilized to send and get information to and from the slave RAM. The low speed peripheral is interfaced with I^2C master bus and synthesized on Spartan 3AN. Fig1 shows the I^2C bus system with the I^2C master controller implemented on a FPGA and the RAM device which acts as the slave.The synopsis of the paper is as follows: In section 2, we discussed I^2C 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, I^2C master bus controller in Spartan 3AN FPGA Design kit using Xilinxsoftware and concluded in section 5.

II. PROPOSEDWORK

A. I²C Protocol

I2C is the best bidirectional serial bus for viable information correspondence between two devices. I²C bus underpins many devices and every device is perceived by its one of a kind location.In Fig.1data_in and addr_in are the 8 bit address given as input. Clk and reset are the input lines usedto initiate the bus controller process. The R/*w* signal aregiven as an input to indicate whether master or slave acts as atransmitter in the data transmission.PhysicallyI²C bus comprises of only two wires, called SCL and SDA. SCL is the clock line; it is utilized to synchronize all information exchange over the I²C bus. SDA is the information line; the SCL and SDA lines are associated with all devices on the I²C bus. As both SCL and SDA lines are "open drain" drivers they are pulled up using pull up resistors.

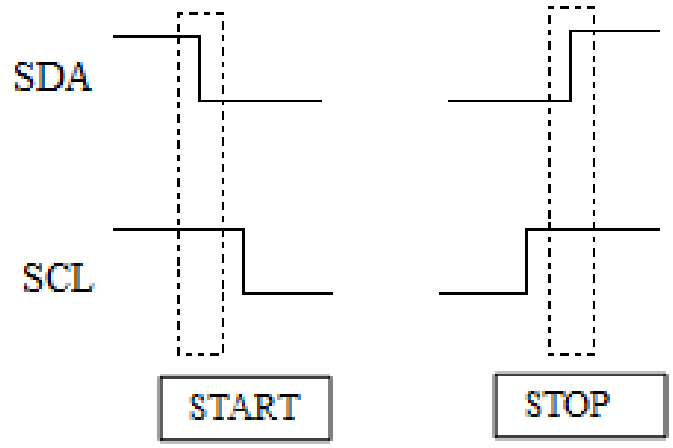

Fig 2. (a) 'START' sequence (b) 'STOP' sequence

TheI²C bus is actually considered to be idle when both SCL and SDA are at logic 1(high) level. At the point when the master (controller) needs to transmit information to a slave (RAM) it starts 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). At that point the bus is thought to be occupied after the START condition. After the START condition, the 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. On the off chance that the accepting device sends back a low affirmation bit, then it has gotten the information and is prepared to acknowledge another byte. On the off chance that it sends back a high flag then it is showing that it can't acknowledge any further

information and the expert ought to end the exchange by sending a STOP arrangement. In Fig.2 (b) which demonstrates the STOP sequence, where the SDA line is driven low while SCL line is high. This flags the end of the exchange with the slave device.

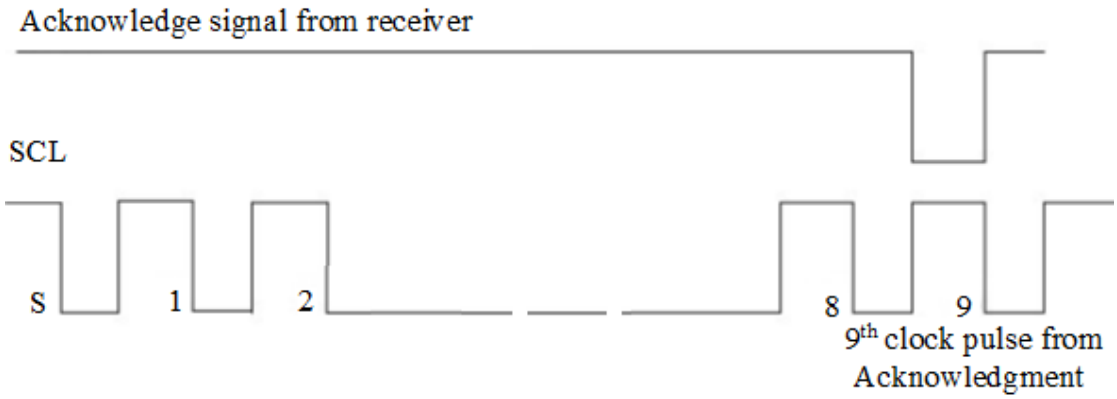

Fig.3 Acknowledgement on the I^2C Bus

B. Serial Data Communication

The I^2C bus has two modes of operation master transmitter and master receiver. The particular I^2C master buslaunches information exchange and can drive both SDA and SCL lines. Slave device(RAM) is tended by the master. It can issue just information on the SDA line.

In master transmission mode, after the initiation of theSTART sequence, the master sends out a slave address. Theaddress byte contains the 7 bit RAM address, which is 1101000, followed by the direction bit $(R|\overline{W})$. After receivingand decoding the address byte the device outputs acknowledgeon the SDA line.

After the RAM recognizes the slavelocation and write bit, the master transmits a register location tothe RAM this will set the register pointer on the RAM.The master will then start transmitting every byte of information with the RAM recognizing every byte got. The masterwill create a stop condition to end the data write.

In master receiver mode, the first byte is received andhandled as in the master transmission mode. However, in thismode, the direction bit will indicate that the transfer direction is reversed. Serial data is transmitted on SDA by the RAM while the serial clock is input on SCL. START and STOPconditions are recognized as the beginning and end of a serial transfer (Fig.3). The address byte is the first byte received afterthe start condition is generated by the master. The address bytecontains the 7-bit RAM address, which is 1101000, followed by the direction bit $(R|\overline{W})$. After receiving anddecoding the address byte the device inputs acknowledge onthe SDA line. The RAM then begins to transmit data starting with the register address pointed to by the registerpointer. If the register pointer is not written before theinitiation of a read mode, the first address that is read is the last one stored in the register pointer. The RAM mustreceive a “not acknowledged” to end a read.

III. SOFTWARE IMPLEMENTATION

I²C master controller is composed utilizing VHDL in view of Finite State Machine (FSM). FSM is a sequentialcircuit that uses a limited number of states to stay informed regarding its history of operations, and in light of history of operation and current data, decides the following state.There are a few states in acquiring the outcome.

Algorithm

State1: An not doing anything (Idle) condition: I²C bus doesn't conduct almost any procedure (SCL in addition to SDA continues to be high).
State 2: Begin (Start) condition: master launches information transmission byproviding STA (SCL will be high in addition to SDA will be by high to be able to low).
State 3: Slave address – write: master posts the slave addresswrite(11010000) to the slave.
State 4: If the slave address matches with the slave, it sends anacknowledgement bit in response to the master.
State 5: 8 Bit Register Address will be transmitted to theslave. Again acknowledgement is sent to the master by theslave.
State 6: Data to be transmitted is sent to the slave by the master. After receiving the data, slave acknowledges themaster.
State 7: Stop condition: Slave sends a stop bit to the master to terminate the communication (SCL is high and SDA is fromLow 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)
State 8: Master transmits slave address for read operation to the slave.
State 9: Master gets the information from the slave and recognizes the slave.
State 10: Master sends a STOP bit to end the association (SCL is high and SDA is from Low to high).

Fig.4 shows the flowchart for I²C master bus communication with slave device. Fig.5 shows the Modelsim simulationresult for write operation, the given data input is written in toslave register address in each state of FSM programmed in VHDL.

Fig.5 shows the Modelsim 10.1c simulation result for read operation, to read the written data from the slave. Fig.6 shows the write operation. The write operation takes place first followed by the repeated start condition and sending the slave adderss read (11010001) in each state of FSM. Fig.7 shows the implementation of code in Spartan kit.

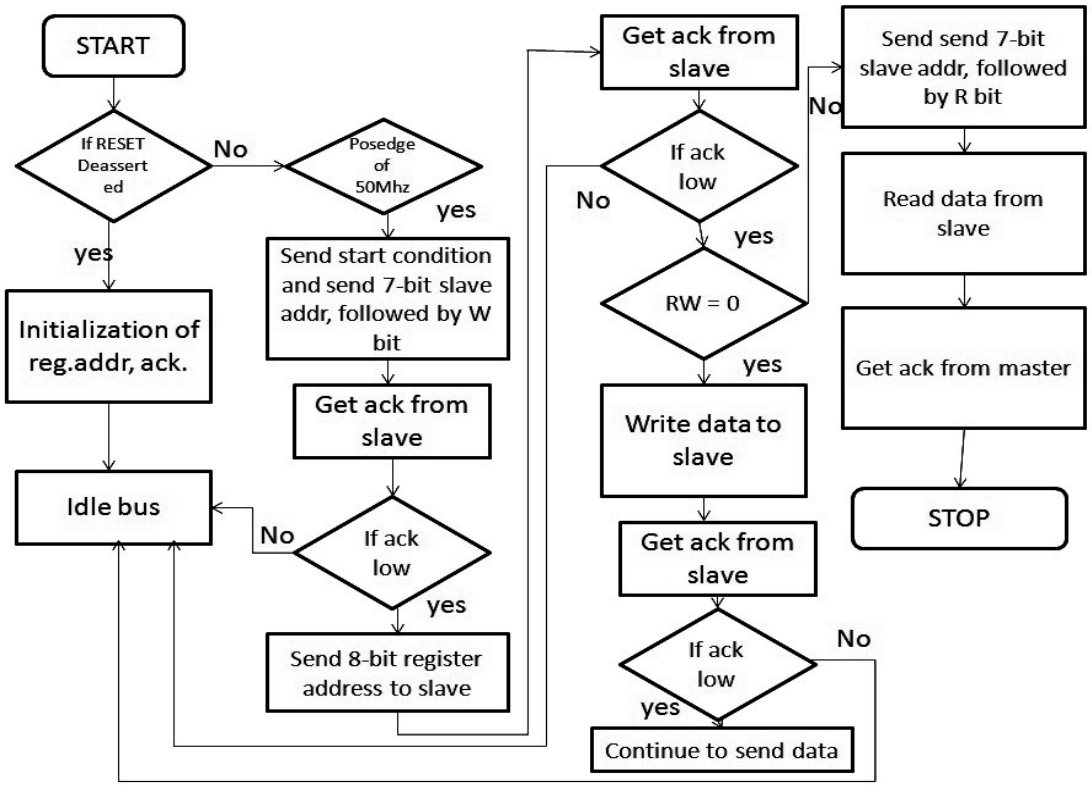

Fig.4: Flow chart for I^{2}C master bus communication with slave device

The simulated VHDL coding is synthesized on Spartan 3AN through Xilinx ISE Design Suite 14.2. Table-1 shows the Xilinx Device Utilization Summary. The result shows that minimal resources are utilized in designing the I^{2}C master along with slave as only 5% slices, 3% flip-flops and 4% LUT's.

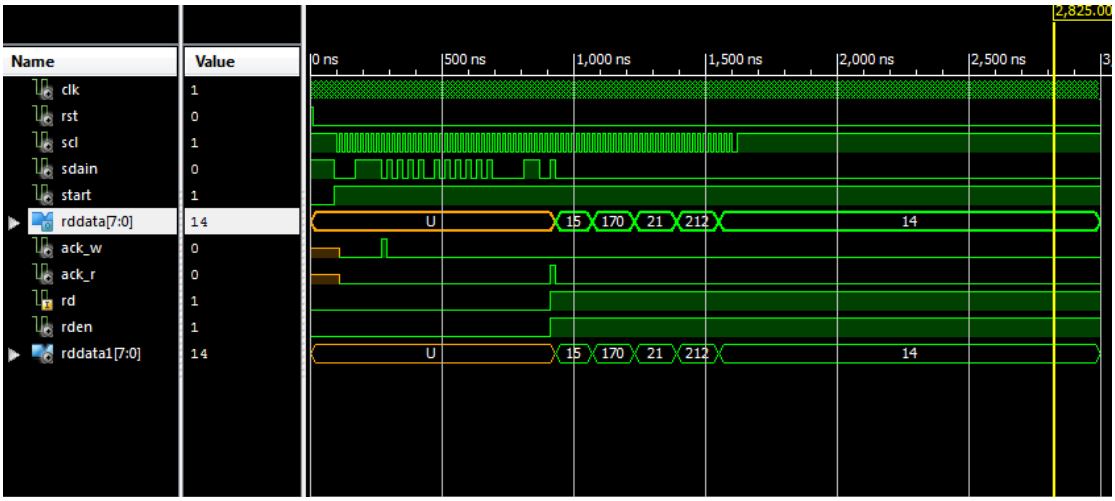

Fig.5 Read operation

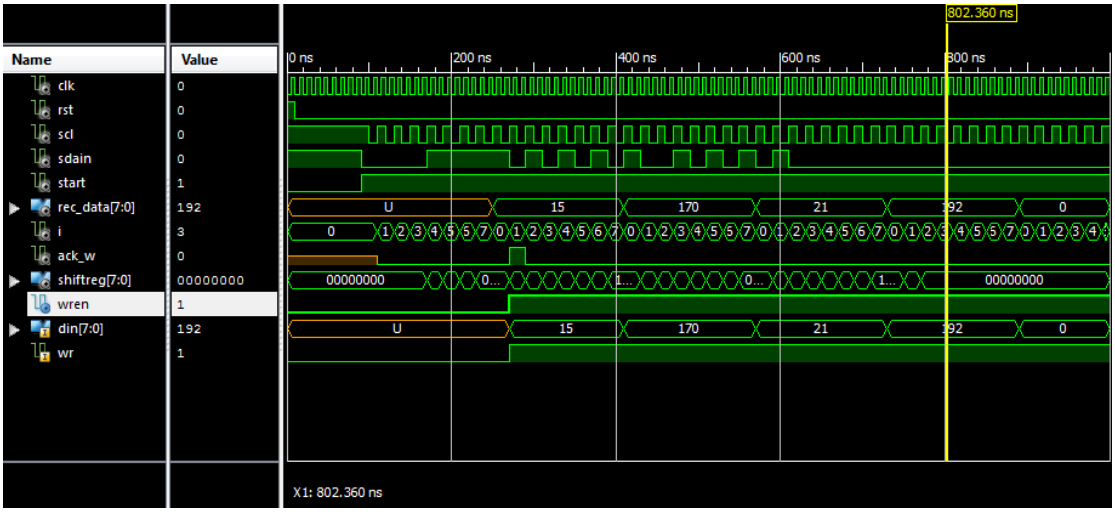

Fig.6 Write operation

IV. CONCLUSION

The I²C Master Controller has been designed, verified and synthesized for implementation. The designed controller is well suited for on-board applications.Though today microcontrollers come with on-chip I^2C interface, there is always a requirement of a controller to interface the microcontroller that do not have on-chip interface with the I^2Cbus. The controller can be used for embedded microprocessor boards, low-power applications, communication systems, cost-effective reliable automotive systems.

TABLE-1 Xilinx Device Utilization Summary

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	238	4656	5%
Number of Slice Flip Flops	294	9312	3%
Number of 4 input LUTs	393	9312	4%
Number of bonded IOBs	10	232	4%
Number of GCLKs	2	24	8%

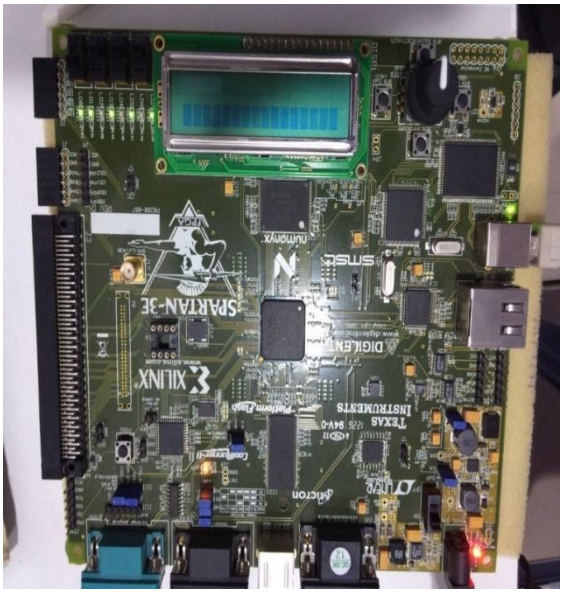

Fig.7: Implementation in Spartan 3AN design kit

REFERENCES

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

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

[3] *I2C Bus Specification*, Philips Semiconductor, version 2.1, January 2000.

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

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

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

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

[8] Vincent Himpe, "Historical background of I2C," in *Mastering the I2C Bus*, Aachen, Germany: ElektorVerlag publications, 2011.

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

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