International Journal of Scientific Engineering and Technology Research

ISSN 2319-8885
Vol.05,Issue.08,
April-2016,
Pages:1706-1709

www.ijsetr.com

Design of I2C Master Bus Controller with Single Slave(Ds1307)

A. VENKATA SUMA[1], B. HEMANTH NAG[2], L. ESWAR PRAKSH[3], M. HARISH KUMAR[4]

Abstract: This paper focus on efficient designing of Inter Integrated Circuit (I2C) master controller. The design follows I2C specifications for address sending and data transfer operations. Here it uses DS1307 as slave. The design is software based that provides easy up-gradation of the whole electronic system by simply clipping or unclipping of the desired IC. It is equipped to ensure no data loss and bus conflicts with multiple device connections on an I2Cbus. The paper demonstrates how I2C Master Controller can achieve appropriate data transmission without collapse. The master controller is designed in VHDL and is simulated in Model Sim. For synthesizability the design is further implemented in Xilinx XST 14.1.

Keywords: Inter Integrated Circuit, Master, Slave, DS1307, Arbitration, Serial Clock, Serial Data Communication, FPGA.

I. INTRODUCTION

The bus protocol which manages the communication between the ICs within a system and between the systems is called the Inter-IC bus or I2C bus. In the world of multiple application based product it is very much a mandatory to have multiple devices connected to a system, this includes peripherals following different communication protocols as well. This requirement gives rise to the need for an intermediate system which can act as a bridge between two or more devices following different communication protocols. This is where I2C master protocol design is very useful. Today a system is connected to a number of devices and make the communication smooth and fast, I2C bus protocol is considered as one of the best. DS1307 is low speed peripherals is taken as slave. The I2C master protocol on one end is connected to micro-controller interface and on the other end it is connected to ds1307 interface. Its main function will be, to understand the control register transmitted by micro-controller and convert it into DS1307 low speed signals. In this project, we are implementing I2C 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. I2C 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. I2C bus is a medium for communication where master protocol is used to send and receive data to and from the slave DS1307. The low speed peripheral, DS1307 is interfaced with I2C master bus interface and synthesized. Fig.1 shows the I2C bus system with the I2C master protocol implemented on a FPGA and the DS1307 interface acting as the slave. 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. In section 4, holds the detailed description of hardware implementation of I2C master bus controller in Spartan 3AN FPGA Design kit using Xilinx software. Finally, concluded with future scale up in section 5.

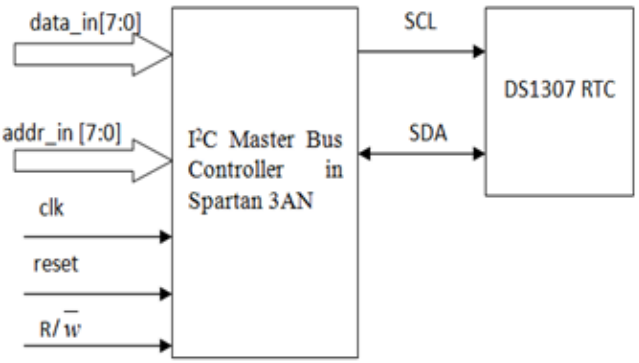

Fig.1. I/O Diagram of I2C Master Controller interfaced with DS1307 RTC slave device.

II. PROPOSED WORK

I2C bus consists of only two lines, SDA and SCL. Both of these lines are open-drained, and must be pulled up to VCC with a 5.6KΩ resistor. Regardless of how many devices are connected to the bus, only one pull-up resister is needed per line. Furthermore, the SCL line needs to be pulled up with a resistor only if there will be two or more masters in the system, or when the slave will do clock stretching as a flow-control measure. In the first case, the pull-up resistor on the SCL line is required for arbitration purposes when two or more masters try to initiate a data transfer at the same time.

A. I2C Protocol

The I2C bus is idle when both SCL and SDA are at a logic 1 level. The master initiates a data transfer by issuing a START condition, 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, a slave address is sent out on the bus by

the master. This address is 7 bits long followed by an eighth bit which is a data direction bit (R/W) where a 0 indicates a write from the master to the slave and a 1 indicates a read from the slave to the master. The master, who is controlling the SCL line, will send out the bits on the SDA line, one bit per clock cycle of the SCL line, with the most significant bit sent out first. The value on the SDA line can be changed only when the SCL line is at a low.

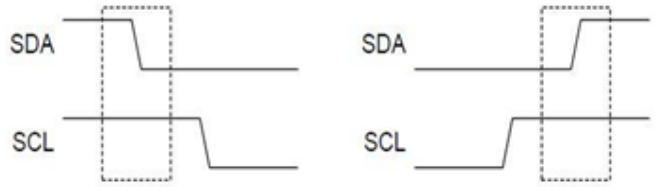

Fig. 2. (a) Start condition (b) Stop Condition.

The slave device whose address matches the address that is being sent out by the master will respond with an acknowledgment bit on the SDA line by pulling the SDA line low during the ninth clock cycle of the SCL line as shown in Fig. 4. The direction bit (R/W) determines whether the master or the slave will be the transmitter in the subsequent data transmission after the sending of the slave address. Every byte put on the SDA line for transmission must be 8-bits long with the most significant bit first. Except for the START and STOP conditions, the SDA line must not changed when the SCL line is high. The number of bytes that can be transmitted is unrestricted. Each byte has to be followed by an acknowledge bit. The acknowledge-related clock pulse is generated by the master. The transmitter releases the SDA line (sets it high) during the acknowledge clock pulse, and the receiver must pull down the SDA line during the acknowledge clock pulse to acknowledge the receipt of the byte. The one exception is when a master-receiver is involved in a transfer. In this case the master-receiver must signal the end of data to the slave-transmitter by not generating an acknowledge on the last byte that was clocked out of the slave. To signal the end of data transfer, the master will send a STOP condition by pulling the SDA line from low to high while the SCL line is at a high as shown in Fig.2 (b). Instead of sending a STOP condition, the master can send a repeated START condition so that the master can change the direction of the data transmission without having to release the bus.

B. Serial Data Communication

The I2C bus has two modes of operation: master transmitter and master receiver. The I2C master bus initiates data transfer and can drive both SDA and SCL lines. Slave device (DS1307) 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 DS1307 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 DS1307 acknowledges the slave address + write bit, the master transmits a register address to the DS1307 this will set the register pointer on the DS1307. The master will then begin transmitting each byte of data with the DS1307 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 DS1307 while the serial clock is input on SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer (Fig.3). The address byte is the first byte received after the start condition is generated by the master. The address byte contains the 7-bit DS1307 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 DS1307 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. The DS1307 must receive a "not acknowledged" to end a read.

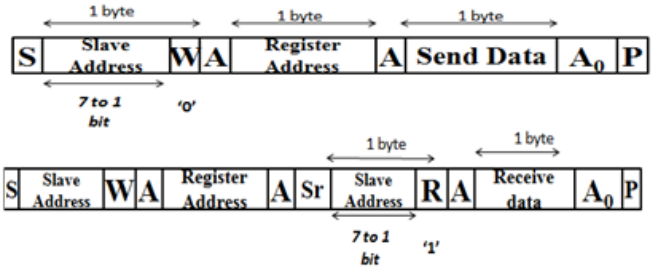

Fig. 3. Master Transmission and Receiver Mode.

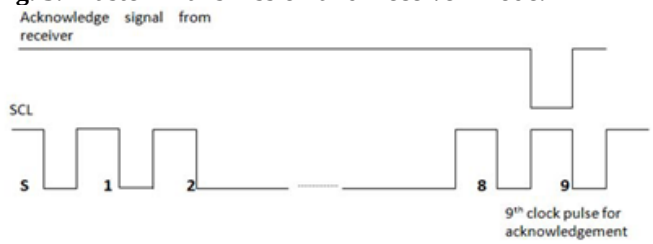

Fig. 4.Acknowledgement on the I2C Bus.

C. MAXIM DS1307

The DS1307 supports a bi-directional, 2- wire bus and data transmission protocol. The pin assignment of DS1307 is given in Fig.5.The DS1307 operates as a slave on the 2- wire bus. Fig.6 shows the interface connection of I2C bus in Spartan 3AN with the DS1307 RTC chip [2].

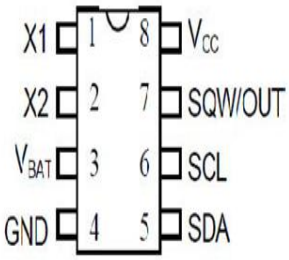

Fig.5. Pin Assignment of DS1307.

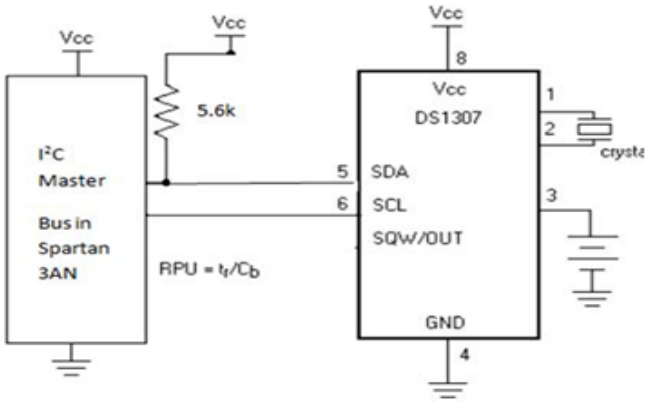

Fig.6. DS1307 connected to two wire data bus.

III. SOFTWARE IMPLEMENTATION

I2C master controller is designed using VHDL [5] based on Finite State Machine (FSM) [13]. 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: I2C 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[12] 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). 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)

State 8: Master transmits slave address for read operation to the slave.

State9: 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).

Fig.7. Flowchart for I2C master bus communication with slave device.

Fig.7 shows the flowchart for I2C master bus communication with slave device. Fig.8 shows the simulation result for DS1307. In DS1307, we have activated only 4th order FIR filter for the given data input and filter coefficients are applied and output is observed. Generated square output is also observed programmed in VHDL. Fig.9 shows the simulation result for I2C, a dataout, sda, read and write operations are observed in the I2C Master. 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. Here the I2C bus protocol is designed using VHDL and implemented in Spartan 3E using Xilinx ISE Design Suite14.2. The simulation results are shown below: The below table shows the Xilinx Device Utilization Summary. The result shows that minimal resources are utilized in designing the I2C master as only 2% slices, 1% flip flops and 1% LUTs are utilized.

TABLE I: Device Utilization Summary (Estimated Values)

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	177	126800	0%
Number of Slice LUTs	249	63400	0%
Number of fully used LUT-FF pairs	132	294	44%
Number of bonded IOBs	28	210	13%
Number of BUFG/BUFGCTRLs	2	32	6%

IV. HARDWARE IMPLEMENTATION

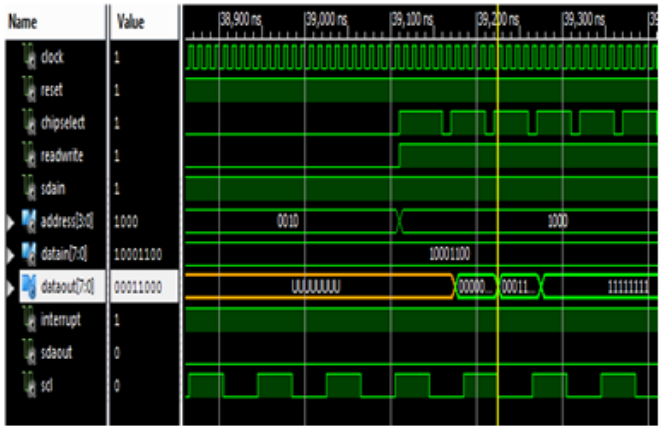

Fig. 8. Simulation of I2C master.

The hardware implementation is done by interfacing DS 1307 with the I2C master present in Spartan 3AN [10] kit. It is an 8 pin DIP. A power supply of 5V is given to the slave. Pin 4 is connected to the ground (GND) and pin 8 is connected to the VCC. Pins 5 and 6 are connected to SDA and SCL of I2C master bus in Spartan 3AN. Both SDA and SCL are open drained lines. When multiple masters are connected, both SDA and SCL are in need to be pulled up with a 5.6kΩ resistor to the VCC. Since only one master is connected, there is no need to pull up SCL. Red wire indicates VCC, Black wire indicates ground (GND), yellow wire indicates SCL line and Green wire indicates SDA line connected with Spartan 3AN FPGA kit.

Fig.9. Hardware implementation in Spartan 3AN design kit.

After completing the coding in VHDL, it is downloaded to the Spartan 3AN kit using Xilinx software.And corresponding input is given according to the I2C protocol.

V. CONCLUSION

The result shows that minimal resources are utilized in Spartan 3A. The design process is simplified using VHDL to design the I2C bus protocol, I2C Master (Master) transmits and receives data to and from the DS1307 (Slave). So that any low speed peripheral devices can be interfaced using I2C bus protocol as master. In future, this can be implemented as real time clock in networks that contains multiple masters and multiple slaves to coordinate the entire system by clock synchronization techniques.

VI. REFERENCES

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

[2] DS1307 64 x 8, Serial, I2C Real Time Clock, 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]Tim Wilmshurst, "Starting with Serial," in Designing Embedded

[6] Spartan-3A/3AN FPGA Starter Kit Board User Guide, Xilinx, version 1.1, 2008.

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

[8]Systems with PIC Micro-controllers: Principles and Applications, 2nd Ed. Burlinton: Newnes, 2009, pp.307-327.

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

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

[11]"Implementationof i2c master bus controller using vhdl on fpga" International conference on Communication and Signal Processing, April 3-5, 2013, India Bollam Eswari, N.Ponmagal, K.Preethi, S.G.Sreejeesh, 978-1-4673-4866-9/13/$31.00 ©2013 IEEE.