THE INTERNATIONAL JOURNAL OF SCIENCE & TECHNOLEDGE

Assertion Based Verification of I2C Master Bus Controller with RTC

Sagar T. D.
M.Tech Student, VLSI Design and Embedded Systems
BGS Institute of Technology, BG Nagar, Mandya, India
Balaji B. S.
Assistant Professor, Department of ECE
BGS Institute of Technology, BG Nagar, Mandya, India

Abstract:
This paper implements serial data communication using I^2C(Inter –Integrated Circuit) master bus controller using a field programmable gate array(FPGA).The I^2C master bus controller is interfaced with MAXIM DS1307,which acts as a slave. This module is designed in Verilog HDL and simulated in Questa sim 6.4 c. The design is synthesized using Xilinx ISE Design suite 14.2. I^2C master initiates data transmission and in order slave responds to it. It can be used to interface low speed peripherals like mother board, embedded system, mobile phone, set top boxes ,DVD,PDA's or other electronics devices.

***Key words**: DS1307, FPGA, I^2C, master, Questa sim , serial data communication, slave, Spartan 3AN, Xilinx*

1. 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 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.USB/SPI/Micro wire 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 require only two lines for communication with two lines 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.

"N.Ponmagal is with VLSI design group, National Institute of Electronics and Information Technology, Kerala, India (E-mail: ponmagal91@gmail. com)."

"K.Preethi is with VLSI design group, National Institute of Electronics and Information Technology, Kerala, India (E-mail: arutselvipreethi@gmail.com)."

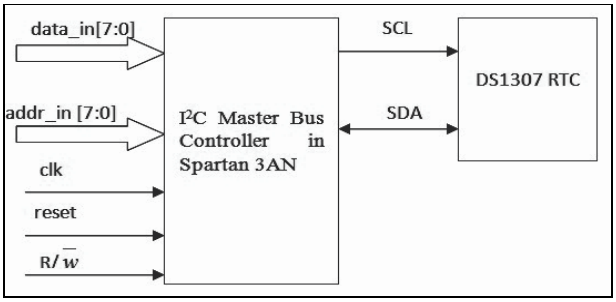

Figure 1: I/O Diagram of I^2C Master Controller interfaced with DS1307 RTC slave device

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. I^2C 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.

I^2C bus is a medium for communication where master controller [9] is used to send and receive data to and from the slave DS1307. The low speed peripheral, DS1307 is interfaced with I2C master bus and synthesized on Spartan 3AN. Fig.1 shows the I^2C bus system with the I^2C master controller implemented on a FPGA and the real time clock device acting 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. In section 4, holds the detailed description of hardware implementation of I^2C master bus controller in Spartan 3AN FPGA Design kit using Xilinx software. Finally, concluded with future scale up in section 5.

2. Proposedwork

2.1. I^2C 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 data_in 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 is given as an input to indicate whether master or slave acts as a transmitter in the data transmission. The physical I^2C 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 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 I^2C 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 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.

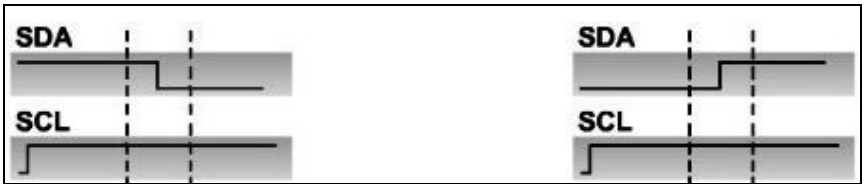

Figure 2: (a) "START" Sequence & (b) "STOP" Sequence

2.2. 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 (DS1307) is addressed by the master. It can issue only data on the SDA line

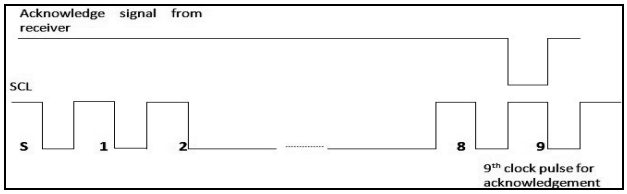

Figure 3: Acknowledgement on the I^2C Bus

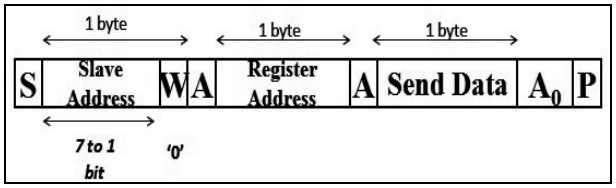

Figure 4: Master Transmission Mode

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 [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 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.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 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.

2.3. MAXIM DS1307

The DS1307 supports a bi-directional, 2-wire bus and data transmission protocol. The pin assignment of DS1307 is given in Fig.6. The DS1307 operates as a slave on the 2- wire bus. Fig.7 shows the interface connection of I2C bus in Spartan 3AN with the DS1307 RTC chip.

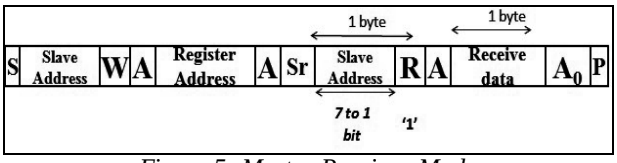

Figure 5: Master Receiver Mode

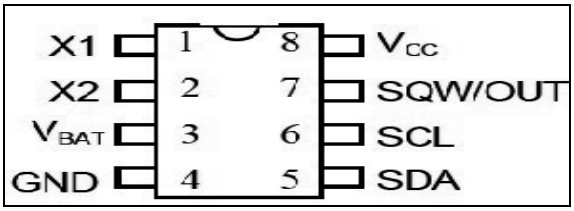

Figure 6: Pin Assignment of DS1307

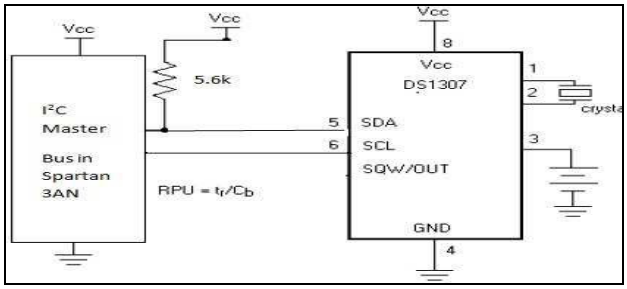

Figure 7: DS1307 connected to two wire data bus

3. Software Implementation

I^2C master controller is designed using Verilog 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.

3.1. 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 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.
- 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).

Fig. 8 shows the flowchart for I2C master bus communication [8] with slave device. Fig.9 shows the Questasim 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 [12]. Fig.10 shows the Questasim 6.4c.

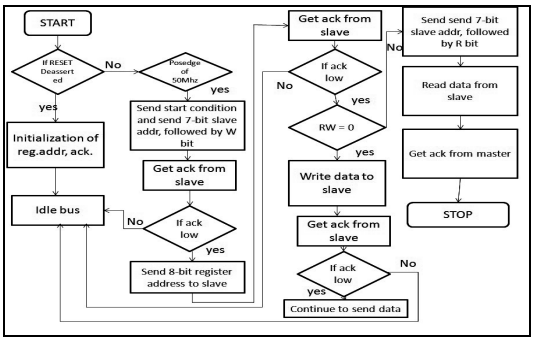

Figure 8: Flowchart for I²C master bus communication with slave device

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 [7]. The design is analyzed using Chip Scope Analyzer in Spartan 3AN platform.

4. Hardware Implementation

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 5.6k resistor to the VCC. Since only one master is connected there is no need to pull up SCL.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 I²C protocol.

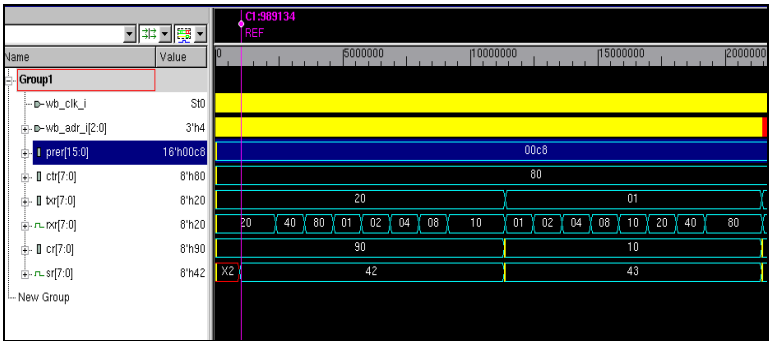

Figure 9: Questasim Simulation Result for register Write/read operation

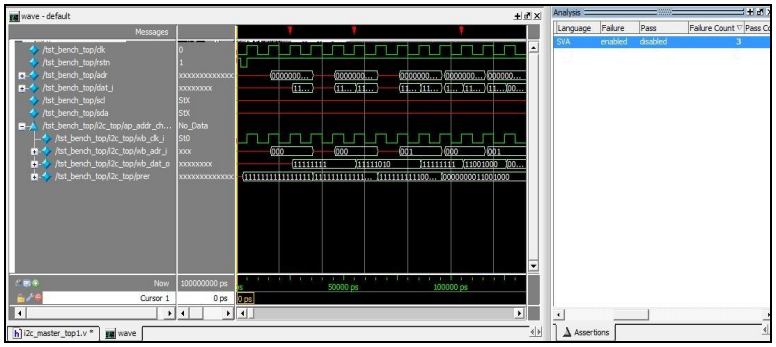

Figure 10: Questasim Simulation Result for SDL and SCL generation and Assertion Result

5. Conclusion

This project demonstrates how I²C Master Controller (Master) transmits and receives data to and from the DS 1307 (Slave). So that any low speed peripheral devices can be interfaced using I²C bus protocol as master and the overall design is verified using System Verilog Assertion Based Verification. In future, this can be implemented as real time clock in networks that contains multiple masters and multiple slaves to co-ordinate the entire system by clock synchronization techniques.

6. References

1. I^2C 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. 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, 2nd Ed.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 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. 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.