SPI to I2C Protocol Conversion using Verilog

Dvijen Trivedi
Dept. of EXTC Engineering
K.J. Somaiya College of Engineering
Mumbai - 400077, India
Email: dvijen.trivedi@gmail.com

Aniruddha Khade
Dept. of EXTC Engineering
K.J. Somaiya College of Engineering
Mumbai - 400077, India
Email: aniruddha.khade@somaiya.edu

Kashish Jain
Dept. of EXTC Engineering
K.J. Somaiya College of Engineering
Mumbai - 400077, India
Email: kashish.j@somaiya.edu

Ruchira Jadhav
Associate Prof. Dept. of EXTC Engineering
K.J. Somaiya College of Engineering
Mumbai - 400077, India
Email: ruchirajadhav@somaiya.edu

***Abstract*—The purpose of this paper is to design and simulate a Protocol Conversion Unit (PCU) for seamless communication between the two widely accepted serial communication protocols SPI and I2C. Design given in this paper takes data from a sender device working on SPI protocol and sends it to a receiver device working on I2C protocol, which otherwise without such design would not be possible. SPI supports full duplex communication unlike I2C which is half duplex. Also SPI is faster than I2C. I2C on the other hand is just a two wire interface as unlike SPI it does not use a dedicated Slave Select line. Instead I2C relies on Address and Acknowledgement scheme to communicate with slave. Thus in areas where the controlling device needs to communicate with a lot of peripheral devices, it is essential for the controller to send commands and data to the concerned peripheral device quickly using high speed of SPI and at the same time save on dedicated pins for each peripheral device using a rather simple Two Wire Interface of I2C, a design capable of providing conversion between SPI and I2C formats becomes essential. In this paper support for just one peripheral device is given. The design in this paper can be upgraded to support large no of peripherals by providing a First In First Out (FIFO) Queue for storing commands and data along with corresponding addresses of peripheral devices in the Protocol Conversion Unit (PCU).**

***Index Terms*—Serial Peripheral Interface (SPI), Inter-Integrated Circuit (I2C), Protocol Conversion Unit (PCU), Serial In Parallel Out (SIPO), Master Out Slave In (MOSI), Master In Slave Out (MISO), Serial Clock (SCLK) for SPI, Slave Select (SS), Serial Clock (SCL) for I2C, Serial Data (SDA), First In First Out (FIFO) Queue.**

I. Introduction

For serial communication two protocols namely Serial Peripheral Interface (SPI) and Inter-Integrated Circuit (I2C) are widely used. More often there are wide variety of peripherals from different manufacturers using either of the two mentioned protocols for serial communication. Protocol Conversion Unit (PCU) designed in this paper enables a SPI master (sender) device (usually a controller) to communicate and send data to I2C slave (receiver) device (usually a peripheral device). First both the serial communication protocols SPI and I2C are discussed, then a discussion on the design for Protocol Conversion unit (PCU) is given. Finally simulation results for the top level design SpitoI2c with SPI master, PCU and I2C slave is given. The top level design SpitoI2c is synthesized in Xilinx ISE 14.7 and simulation is performed using softwares inbuilt simulator ISIM. Both the top level design and testbench are written in Verilog.

II. SPI

SPI (Serial Peripheral Interface) is a synchronous serial communication protocol. It was developed by Motorola in the mid-1980s. SPI has a four wire interface and supports full duplex communication. SPI has a master-slave architecture. It has a single master and multiple slave devices. SPI supports data rates up to 400 Mbps. Each SPI device has the following four pins:

- SCLK Serial Clock.
- MOSI Master Out Slave In.
- MISO Master In Slave Out.
- SS Slave Select.

Master is responsible for generating the clock (SCLK) and giving appropriate signal on the slave select (SS) line. Following steps are followed in SPI communication:

- Master generates clock (SCLK), typically in MHz range.
- Ideally slave select (SS) line is high. Master pulls down the SS line of the slave device it wishes to communicate with. This signal's the start of communication.
- With each clock pulse on the SCLK, the Data contents stored in registers of MOSI and MISO are shifted out and transferred to each other.
- After all the Data bits (usually 8 bits) are sent, Master pulls up the SS line of the slave to indicate end of communication. At this point Master has the initial contents stored in Slave and Slave has the initial contents stored in Master. Hence it is a full duplex communication.

III. I2C

I2C (Inter-Integrated Circuit), pronounced I-squared-C is a synchronous serial communication protocol. It was developed by Philips Semiconductor in 1982. I2C has a two wire interface

978-1-5386-5257-2/18/$31.00

and supports half duplex communication. I2C has a master-slave architecture. I2C supports data rates 100 Kbps (standard mode), 400 Kbps (Fast mode) and maximum up to 3.4 Mbps (high speed mode). Each I2C device has the following two pins:

- SCL Serial Clock.
- SDA Serial Data.

Master is responsible for generating clock (typically in KHz range) on SCL line. I2C frame typically consists of 20 bits (Start bit + 7 bit Address + R/$\overline{W}$ bit + ACK bit + 8 bit data + ACK bit + Stop bit). Following steps are followed in I2C communication:

- Ideally both SCL and SDA lines are high.
- First SDA goes high to low, followed by SCL going high to low. This is the Start bit.
- Now master generates clock on SCL line and in synchronization to clock on SCL all the transfer of bits occur on SDA line.
- Master sends address of slave device it wishes to communicate with. It appends a R/$\overline{W}$ bit at the end of address to indicate the type of operation it wishes to perform. All the slave devices connected to master receive this address and only the slave device with address sent on the SDA line sends an ACK bit. Rest all slave devices leave control of SCL and SDA line. R/$\overline{W}$: 1 for Read and 0 for Write.
- Now data is sent, after which SCL line goes low to high, followed by SDA line going low to high. This is the Stop bit.

IV. Design

The Protocol Conversion Unit (PCU), consists of a Serial In Parallel Out (SIPO) register and an I2C master device. SS line of the SPI master (sender device) is connected to the Enable (SS) pin of SIPO. During the transmission of data by the SPI master (sender) a low signal enables SIPO register. SIPO uses clock provided by SPI master on SCLK line. With each clock pulse data on the MOSI line of SPI master is shifted into the internal register of SIPO through DIN (serial data in) of SIPO and data initially stored in the internal register of SIPO is shifted out on the MISO line of SPI master through SISO of SIPO. After 8 clock pulses SPI master stops communication and SS line goes low to high. Also SPI master raises a flag called DONE. This flag is later used to signal reset of the I2C master. The transition on SS line signals SIPO to load the data in its internal register onto the output bus DOUT. This completes the first step in PCU, where the data from sender is received and ready to be sent to the receiver. Address (7 bits) of I2C slave (receiver device) is stored in input address register called AddressI2cMaster (8 bits). R/$\overline{W}$ bit is appended after the address bits and stored in AddressI2cMaster register. Data on DOUT is loaded into input data register DataI2cMaster (8 bits) of I2C master. DONE flag from SPI master sets RESET of I2C master high. This initiates the second step in PCU. Now I2C master initiates its operation. I2C master follows the I2C frame format mentioned above. I2C master stops communication after sending stop bit. At this point data sent by SPI master (sender) is successfully transferred to the concerned I2C slave (receiver) and the protocol conversion successfully takes place.

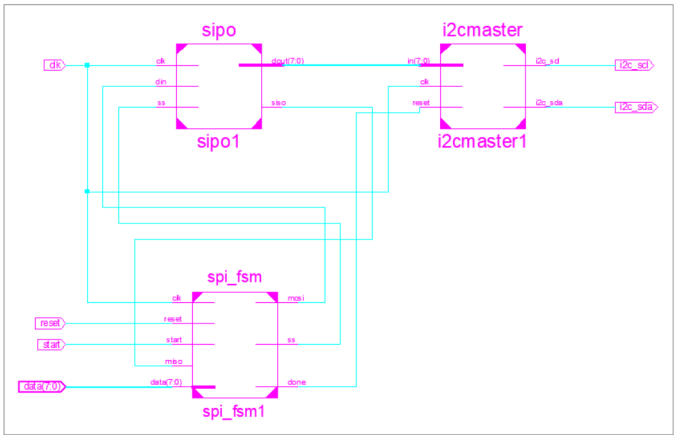

Fig. 1. Top Level Design Module

V. Result

This design was synthesized in Xilinx ISE 14.7 and simulated using software's ISIM simulator. In order to observe simulation results a top level design called SPItoI2C was implemented. It had three instances:

- SpiMaster: SPI master (sender) device. It was implemented using Finite State Machine (FSM) approach [2].
- SIPO: Serial In Parallel Out register of PCU.
- I2cMaster: I2C master device of PCU.It was implemented using Finite State Machine (FSM) approach [2].

An instance for I2C slave (receiver) was created within I2cMaster, named I2cSlave. I2cSlave performs slowly while sending Acknowledgement, hence it holds onto SCL of I2C in order to synchronize and send Acknowledgement effectively. This can be seen by broadening of SCL line during Acknowledgement stage of I2C in simulation waveform. SDA line of I2C is bi-directional. Hence while I2cMaster waits to receive Acknowledgement form I2cSlave it puts SDA line in high impedance state. This can be seen in simulation waveform. Top level design SPItoI2C can be seen in fig. 1 State diagram for SPItoI2C can be seen in fig. 2,3,4,5. Simulation output can be seen in fig. 6. Data 10101010 sent by SPI master (sender) on MOSI line is received successfully on SDA line of I2C.

VI. Conclusion

The design in this paper implemented a Protocol Conversion Unit (PCU) which takes command and data from sender device (usually a controller) working on SPI protocol and sends it to receiver device (usually a peripheral device) working on I2C protocol. Thus such a design would enable the controller to communicate with large number of peripherals using high speed of SPI and save on dedicated pins for each peripheral device using I2C.

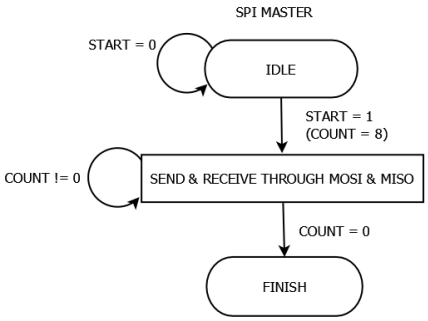

Fig. 2. Flowchart of SPI Master Communication

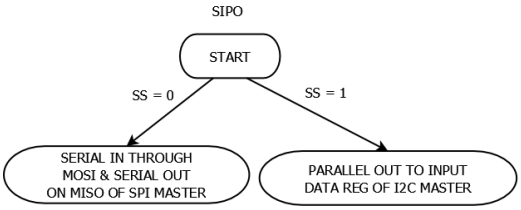

Fig. 3. Flowchart of SIPO

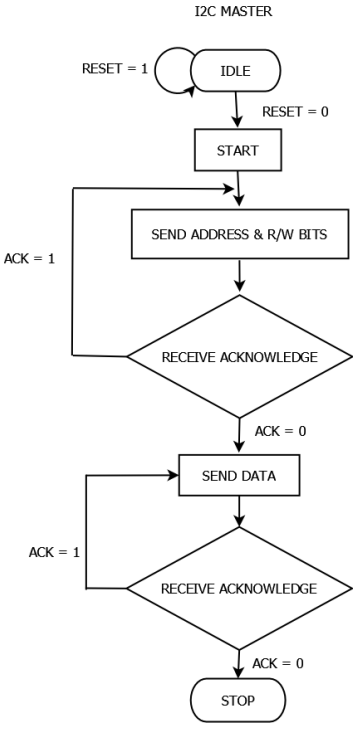

Fig. 4. FLowchart of I2C Master Communication

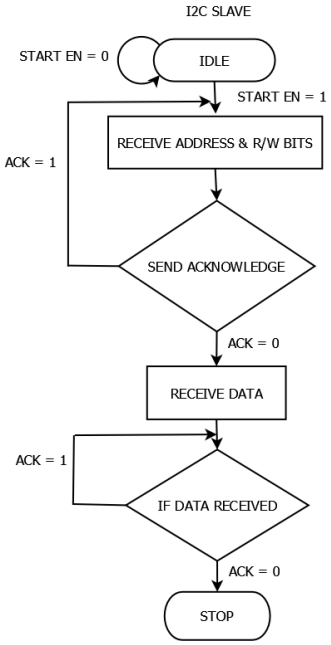

Fig. 5. Flowchart of I2C Slave Communication

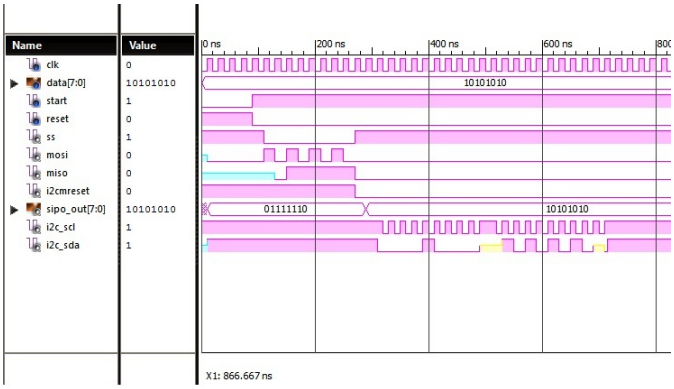

Fig. 6. Simulation Output

VII. FUTURE SCOPE

The design given in the paper can be upgraded to support above mentioned multi slave mode by providing a First In First Out (FIFO) Queue in the Protocol Conversion Unit (PCU). Such a Queue will store commands and data along with addresses of peripheral device sent by the controller. Queue becomes a holding buffer for commands and data sent by controller. Contents of Queue would be erased once they are successfully sent to peripheral devices by the Protocol Conversion Unit. Thus in such a scenario it essentially becomes a producer consumer problem and care must be taken so that Queue does not become full, resulting in loss of command and data sent by controller. A Queue Full flag could be set in such cases to disable controller until Queue is empty again to store more data and commands.

ACKNOWLEDGMENT

This paper is written based on the final year project we did as a part of our curriculum. We would like to thank EXTC Department of our college K.J. Somaiya College of Engineering, Mumbai to allow us to access lab facilities. We would also like to thank our Project Guide Associate Prof. Ruchira Jadhav for guiding us and motivating us to do our final year project in VLSI. We would also like to thank our Department's Assistant Professor, Ankit Khivsara in suggesting us this project and helping us at times needed while developing this project.

REFERENCES

[1] Abhilash S.Warrier, Akshay S.Belvadi, Dhiraj R.Gawhane, Babu Ravi Teja K, FPGA Implementation Of SPI To I2C Bridge, International Journal of Engineering Research & Technology (IJERT), Vol. 2 Issue 11, November - 2013.

[2] M. Morris Mano, Michael D. Ciletti, Digital Design:With an Introduction to verilog HDL, 5e, Pearson, 2013.