International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-2 Issue-2, January 2013

SPI Implementation on FPGA

Trupti D. Shingare, R. T. Patil

Abstract: ***The SPI is a full-duplex, synchronous, serial data link that enables communication between a host processor and peripherals. Based upon Motorola's SPI-bus specifications, version V03.06, release February 2003, the designs are general purpose solutions offering viable ways to controlling SPI-bus, and highly flexible to suit any particular needs. However, Field programmable gate array devices offer a quicker and more customizable solution. This paper provides a full description of an up to- date SPI Master/Slave FPGA implementations, In conformity with design-reuse methodology.***

Keywords: ***Serial Peripheral Interface (SPI), Field Programmable Gate Array (FPGA), System on Chip (SOC).***

I. INTRODUCTION

In the world of communication protocols, SPI is often considered as "little" communication protocol compared to Ethernet, USB, SATA, PCI-Express and others that present throughput in the ×100 Mb/s range, if not Gb/s. It is important to not forget the purpose of each protocol. Ethernet, USB, and SATA are meant for "outside the box communications" and data exchanges between whole systems. While SPI, as well as others serial protocols such as I2C and 1-wire for instance, are well suited for communications between integrated circuits for low/medium data transfer speed with on-board peripherals. Consequently, when there is a need to implement a communication between an integrated circuit such as a microcontroller and a set of relatively slow peripherals, there is no point in employing any excessively complex protocols. In this case, SPI stands among the best candidates. This is why it becomes a worldwide standard for modern digital electronics systems and it will probably continue to compete in the future

II. SERIAL PERIPHERAL INTERFACE

The Serial Peripheral Interface (SPI) is a high speed (up to 400 Mhz) synchronous serial interface/protocol designed by Motorola. It is a popular interface used for connecting peripherals to each other and to microprocessors. Most literature indicates that the interface can only be used for eight or sixteen bit block data transfers, but many Motorola microcontrollers allow transfers of any range of blocks between two and sixteen bits at a time. Because of the serial nature of the interface, data transfers of more than sixteen bits at a time can be implemented easily through control signals.

Manuscript published on 30 December 2012.
***Correspondence Author(s)**
Ms. Trupti, M. Tech (Digital System)-II, R.I.T. Rajaramnagar, Sakhrale, Islampur, Maharastra
Prof. Mr. R. T. Patil, Associate Professor, E & Tc Dept. R.I.T. Sakhrale, Islampur, Maharastra

The interface uses a 3-wire bus plus a chip/slave select line for each device connected to the bus. The three bus lines are as follows:

- SCLK - the clock signal used for synchronizing data transfers. It is generated by the bus "Master"
- MISO - Master In Slave Out. Line used for sending data from a slave to the master.
- MOSI - Master out Slave In. Line used for sending data from the master to a slave.

Each device connected to the bus can be selected by the bus master using a dedicated SS (Slave Select) line for each slave device. It is possible to have more than one master hanging off the bus, but only one master can be active at any given time. The implication of this configuration is that the bus master has to have as many lines as there are devices to drive each of the SS lines.

When the master initiates a data transfer, the master writes a bit to the MOSI line and reads a bit from the MISO at the same time on every cycle of the SCLK signal. The data is transferred through a simple shift register transfer scheme where the data is clocked into and out of devices on a first-in, first-out basis. This means that every data transfer results in an exchange of bits between the master and the slave (each device is simultaneously a transmitter and a receiver), making it a full duplex serial interface. When a device is not selected, it must tri-state (release) the output (MISO) line. Through buffering, it would be possible to drive more than one receive-only device, but not more than one transmit-only or receive and transmit device since there would be a contention issue on the MISO line.

The block diagram of this process is as follows:

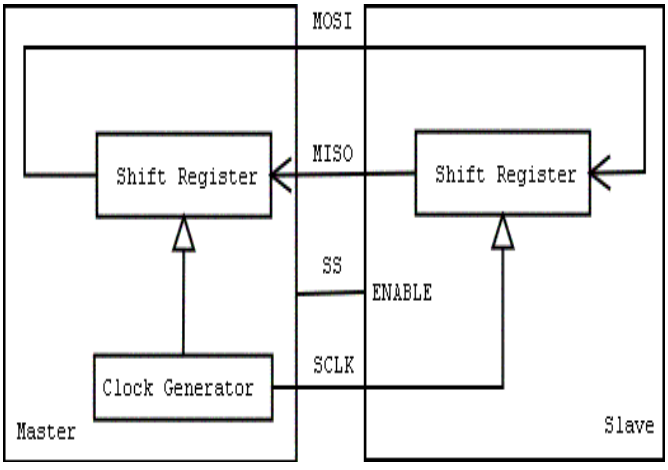

Fig1. Master/Slave Transfer Block Diagram

There are four possibilities for clocking the data based on the clock polarity and the clock phase:

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

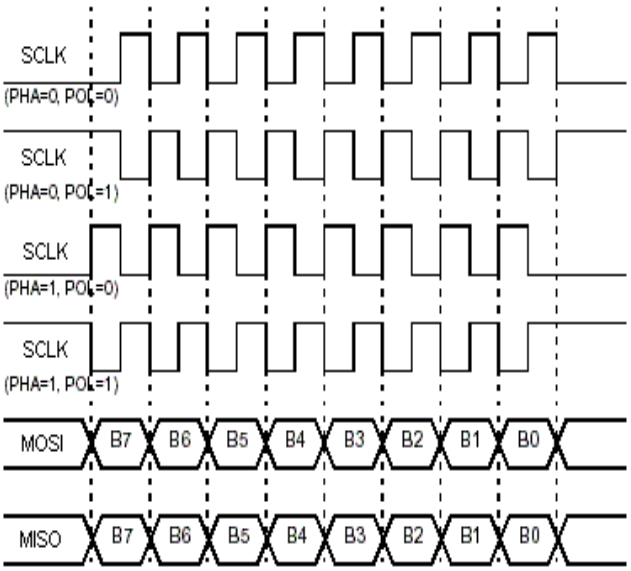

Fig 2. Data transfer w.r.t.CPHA & CPOL

Usually, in synchronous serial protocols, data is clocked out on one edge and clocked in on the other edge to reduce clock skew errors.

III. SPI INTEGRATION ON SOC ENVIRONMENT

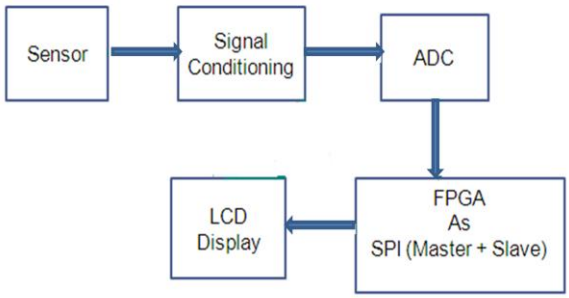

Fig 3. System Block Diagram.

Above block diagram shows the interfacing of sensor with the FPGA via ADC & result is displaying on LCD display. The temperature sensor LM 35 is used for sensing the temperature between -55 to 150°C .The MCP 3201 is used for analog to digital conversion of sensor output.MCP 3201 is 2.7v,12 bit A/D converter with spi serial interface. SPI protocol is used for data transfer from slave to master. Here sensor circuit is used as slave which transferring data to the SPI master. Both these slave & master is coded in vhdl.

IV. SERIAL PERIPHERAL INTERFACE USING VHDL

The SPI was divided into components while implementing the SPI using VHDL. The components were Xtal, sck_logic, Xmit_shift_register, rcv_shift_register, spi_master and spi_slave. All the components made use of the SPI control register, SPI status register and SPI data register. Figure 4 presents a flow chart, which illustrates how the components were instantiated in order to obtain the SPI protocol.

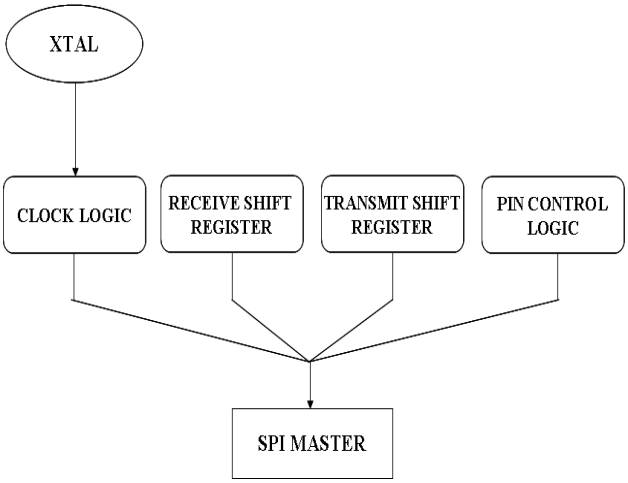

Fig 4. SPI using VHDL

V. SPI ON FPGA

Complete coding of SPI Protocol using VHDL can be implemented on FPGA XC3S400.

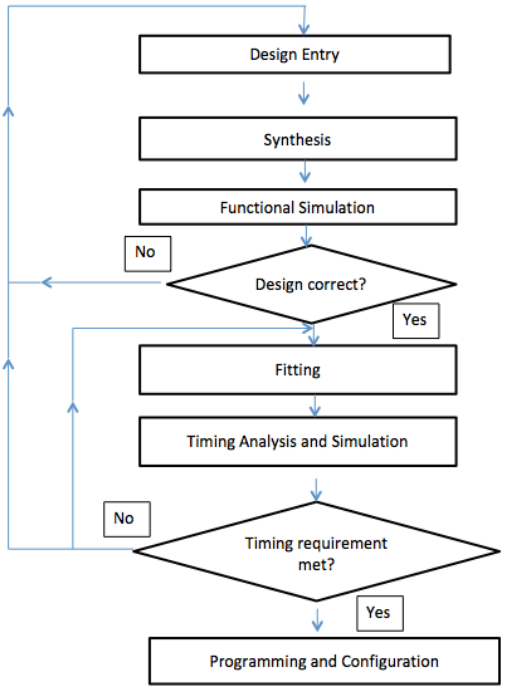

Fig 5. FPGA Design Flow Chart

Using above flow chart with proper mapping, placing & routing FPGA downloading can be done.

VI. CONCLUSION

The paper has shown up the results of an up-to-date FPGA implementation of the Master/Slave sides of the standard SPI protocol. In this paper, we have more particularly focused on design features and utilization issues of high-quality SPI Master/Slave IPs based upon design-reuse methodology. As the RTL-code is technology independent, much faster transfer rate can be obtained in ASIC implementation using a standard cell library.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

REFERENCES

1. F. Leens, "An Introduction to I2C and SPI Protocols," IEEE Instrumentation & Measurement Magazine, pp. 8-13, February 2009.
2. [2] "Design and Implementation of a Reused Interface" CHEN Run1,2, HUANG Shi-zhen1, LIN Wei1, LI Lei2 ,The 1st International Conference on Information Science and Engineering (ICISE2009)
3. "Realising the SPI communication in a multiprocessor system "Akos Szekacs, Tibor Szakaill and Zoltan Hegyk6zi SISY 2007 . 5th International Symposium on Intelligent Systems and Informatics 24-25 August, 2007 Subotica, Serbia
4. "Design and Test of General-Purpos SPI Master/Slave IPs on OPB Bus"A.K. Oudjida, M.L. Berrandjia, A. Liacha, R. Tiar, K. Tahraoui & Y.N. Alhoumays Microelectronics and Nanotechnology Division Centre de Développement Technologies Avancées, CDTA, 2010 7th international multi conference Systems,signal and devices .
5. "FPGA Implementation of I2C & SPI Protocols: a Comparative Study " A.K. Oudjida, M.L. Berrandjia, R. Tiar, A. Liacha, K. Tahraoui Microelectronics and Nanotechnology Division Centre de Développement des Technologies Avancées CDTA 978-1-4244-5091-6/09©2009 IEEE
6. Motorola Inc., "SPI Block Guide V03.06," February 2003.