Design and Verification of Wishbone Compliant Serial Peripheral Interface

Purushotham S
Dept. of Digital communication
Siddaganga institute of technology (SIT),
Tumkur, Karnataka, India

Naveenkumar M
Dept. of Telecommunication
Siddaganga institute of technology (SIT),
Tumkur, Karnataka, India

***Abstract*— Synchronous serial interfaces give practical on-board correspondence between the processor, computerized to simple and simple to advanced converters, memory, and other building hinders present on the chip. Various Integrated Circuit (IC) makers create and deliver parts that are perfect with serial interfaces. The basic serial interfaces incorporate Serial Peripheral Interface (SPI), Inter-Integrated Circuit (I2C), Universal Asynchronous Receiver Transmitter (UART), and Universal Serial Bus (USB). SPI is generally utilized and worthwhile over other serial interfaces because of its highlights of effortlessness, minimal effort, synchronous clock, and non-intruding on rapid information exchange rate. An open source equipment PC transport Wishbone is chosen as the host controller empowering parallel information trade for quicker correspondence. Both the equipment transports utilize a master slave setup which makes the transport interfacing less demanding.**

Keywords*— SPI (serial peripheral interface); Wishbone; Verilog HDL.

I. INTRODUCTION

SPI is the highly used serial communication protocols that is mainly used for the intra-chip high speed data transfers. This is applied to interface among a microcontroller and other peripherals like ADCs, DACs, and EEPROMs.

We are using different data rate communication protocols in the field of serial data communication. SPI is considered as small communication protocol. Each and every protocols have their specific purpose and usage. USB, Ethernet and Serial AT Attachment, are meant for “inter-system communications” and data transmit in the whole system, while in serial peripheral interface the communication happens between the integrated circuits for little or middle data transfer rate with peripheral devices [2].

II. SERIAL PERIPHERAL INTERFACE

Serial to Peripheral Interface (SPI) is a hardware communications protocol developed by Motorola for the transmission of data between the microcontroller and the peripherals. This protocol was later adapted by others in the industry. The Serial Peripheral Interface or SPI-bus is a simple four-wire serial communications interface used by many microprocessor/microcontroller peripheral chips that enables the controllers and peripheral devices to exchange the data with each other. Even though it is initially developed for the communication between host processor and peripherals, a connection of two processors using SPI is possible. Both single-master and multi-master protocols are possible in SPI. The SPI Bus is mainly used only on the PCB. The SPI Bus was designed to transfer data between various IC chips, at very high speeds.

Most literature review indicates that the interface can only be used for eight or sixteen bits 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 made easily through control signals. SPI is a protocol that has four significant signal lines (as shown in Fig. 1).

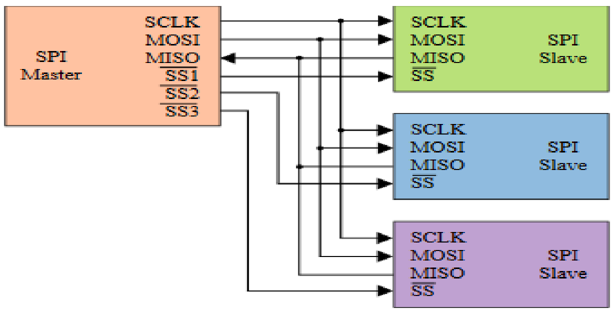

Fig. 1. SPI block diagram.

- MOSI - Output data from the master to the inputs of the slaves.
- MISO - Output data from a slave to the input of the master.
- SCLK - Clock driven by the master to slaves, used to synchronize the data bits.
- SS - Select signal driven by the master to individual slaves, used to select the target slave. If there are multiple slaves, then master makes the particular SS line low to which the data has to be transmitted.

SPI Master drives the SCLK line and controls the stream of information bits. The Slave Select (SS) line must be low to choose a slave. SPI supports single master communication protocol. This implies one focal master starts every one of the correspondences with the slaves. At the point when the SPI master wishes to send information to a slave as well as demand data from it, it chooses slave by pulling the relating SS line low and it actuates the time motion at a clock recurrence usable by the master and the slave. Consequently, when there is a need to implement a communication between an integrated circuit such as a microcontroller and a set of peripheral devices, SPI turns into the best decision to move with. SPI has turned into the de facto standard for current computerized hardware frameworks and it will most likely keep on competing later on.

TABLE I. SPI PROTOCOL: MAIN FEATURES

Device	Features
Originator	Motorola (1979)
Plug & Play	Plug & Play
Interface Type	Serial (3+N wires)*
Distance	Short (In-Box Communication)
Application	Transfer of Data-Streams
Protocol Complexity	Low
Design Cost	Low
Transfer Rate	Free (n x MHz to 10n x MHz)
Power Consumption	Low
Transfer Type	Full Duplex
Time Constraint	Synchronous
Multi Master	No
Multi Slave	Yes
I/O Constraints	No Constraint
Addressing	Hardware (ChipSelect)
Flow Control	No
Clock Stretching	No

III. WISHBONE INTERFACE

The communication of the SPI master and slave devices with the processor on a chip through the Wishbone bus is shown in Fig. 2.

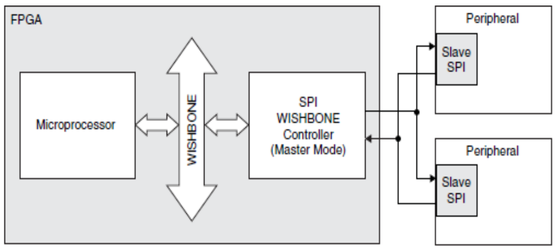

Fig. 2. Processor-SPI Interaction

This SPI WISHBONE controller provides an interface between a microprocessor with a WISHBONE bus and a SPI device. The controller can either act as the SPI Master or SPI Slave device. The selection of the Master or Slave mode is done using parameters in the Verilog HDL code. The design uses a single module [8].

A. *SPI Data Transfer Modes:*

A full-duplex communication is established between the microcontroller or microprocessor and the peripheral devices. The data word or character is simultaneously shifted out from the master to slave and shifted in from the slave to master serially one bit a clock cycle.

The data sampling and shifting are synchronized by the SCLK and only after a slave peripheral device is selected by the microcontroller or microprocessor. Two bits in the SPI control register control the clock phase and polarity. There are four modes of operation. As depicted in the table.3 it defines the modes of operation of the data transfer. The phase and polarity should be identical to both the master and slave throughout the transfer. As SPI is synchronous to the serial clock, the data shift occurs on one edge of the serial clock (either positive or negative) and the data capture is on the other edge. Ideally, there are only two data transfer formats based on the clock phase.

TABLE II. APPLICATIONS OF SPI

Device	Application
Sensors	Temperature, Pressure, Touch Screens, controllers, ADCs
Control Devices	CODECs, DACs, Digital Potentiometers
Communications	Ethernet, USB, CAN, USART, IEEE 802.11, IEEE 802.15.4
Memory	Flash, EEPROM, SD card
Clocks	Real Time Clocks

TABLE III. MODES OF OPERATION

Mode	Polarity (CPOL)	Phase (CPHA)	Data Shift	Data Sample
0	0	0	Falling (Negedge)	Rising (Posedge)
1	0	1	Rising (Posedge)	Falling (Negedge)
2	1	0	Rising (Posedge)	Falling (Negedge)
3	1	1	Falling (Negedge)	Rising (Posedge)

IV. METHODOLOGY

The design methodology of the SPI follows the below flow

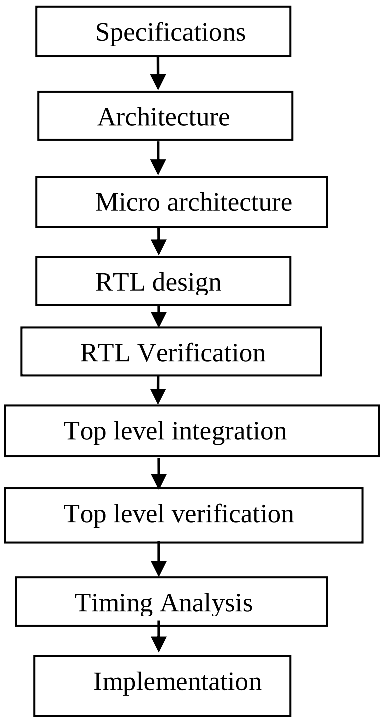

Fig. 3. Flow chart of the VLSI design.

V. ARCHITECTURE

The figure below shows the architecture of the complete SPI to be designed. There are three blocks for which RTL design for each block is written and verified. Then the top level integration is followed by top level verification. After which the timing analysis of the integrated architecture is performed.

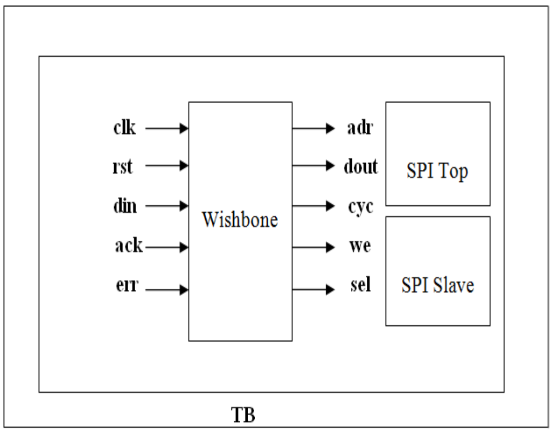

Fig. 4. Design Architecture

The Micro-Architecture is shown below is classified into three bocks

- Wishbone
- SPI Master
- Read FIFO
- Write FIFO
- Control Circuits
- **FIFO Empty:** At the point when the Read Address Register rises to the Write Address Register, the FIFO is named as empty.
- **FIFO Full:** When the read address LSBs equal the write address LSBs and the extra MSBs are different, the FIFO is full. The control circuit block consists of the three main control registers i.e. SPI Control Register (SPCR), SPI Status and Control Register (SPSCR), SPI Data Register (SPDR).

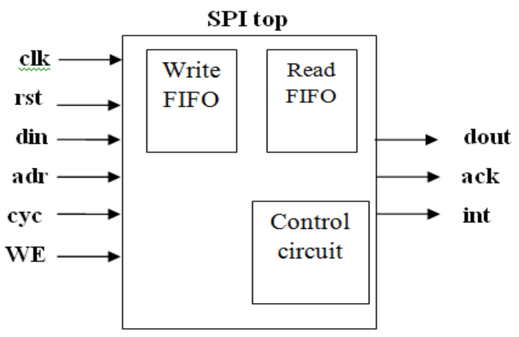

Fig. 5. SPI top module.

SPI Slave contains Control Circuits, the below figure shows the SPI slave block. It contains the control circuit block in which the control registers are there.

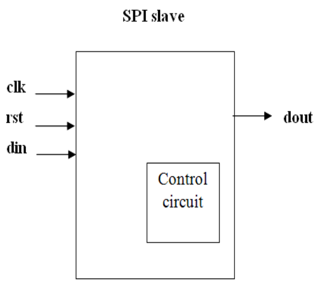

Fig. 6. SPI slave module.

VI. EXPERIMENTAL RESULTS

The verilog code for FIFO is written along with the testbench.The simulation result is shown below.

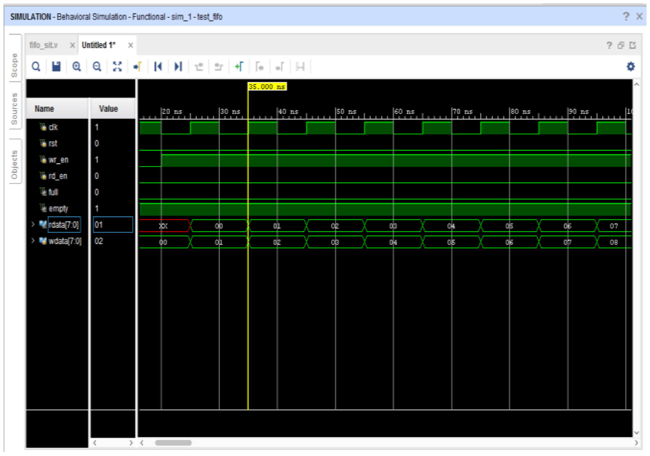
SIMULATION - Behavioral Simulation - Functional - sim_1 - test_fifo
Name
Value
clk
rst
wr_en
rd_en
full
empty
rdata[7:0]
wdata[7:0]
35.000 ns

Fig. 7. Simulation result of the FIFO.

The simulation results shows the operation of FIFO. When the write data is enabled the data bits is written to the register. The data width is taken has 8 bits. The data is shifted to the read pointer at the positive edge of the clock cycle.

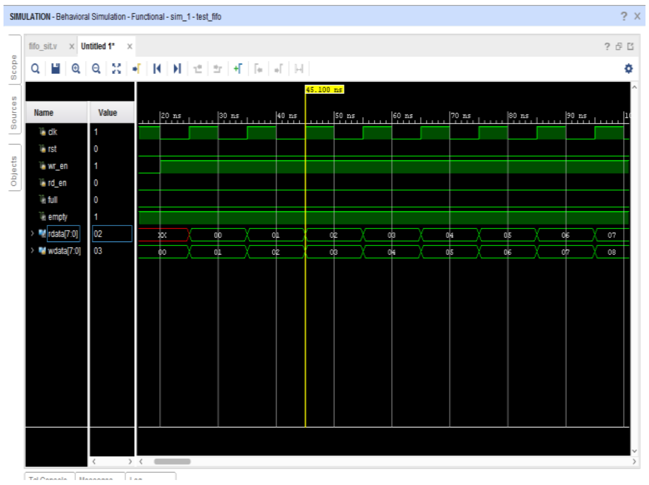
SIMULATION - Behavioral Simulation - Functional - sim_1 - test_fifo
Name
Value
clk
rst
wr_en
rd_en
full
empty
rdata[7:0]
wdata[7:0]
45.100 ns

Fig. 8. Simulation result of the FIFO.

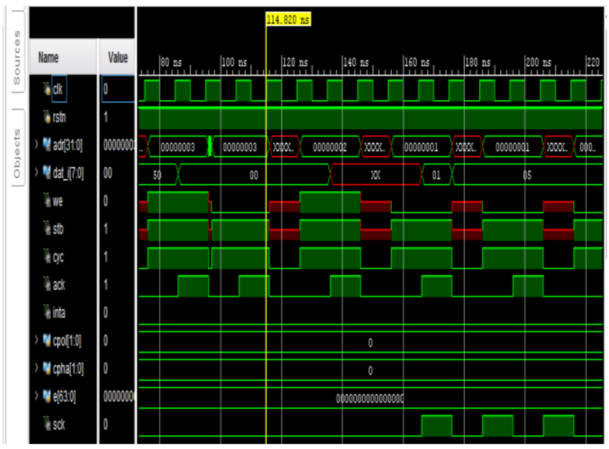

Fig. 9. Simulation result of the master-slave.

VII. CONCLUSION

Our work focuses on the performance analysis of SPI along with the RTL Design work. Design part involves around the specifications and codes written in the Verilog. Serial peripheral interface RTL divided into three blocks i.e. Master, Slave and SPI top module. RTL code written for the complete architecture using the wishbone interface model which open source, work done using XILINX VIVADO Design suite tool which gives simulation and synthesis validation of specification. The complete RTL has been designed for the complete architecture and design summary reports has to recorded.

VIII. FUTURE SCOPE

The development of verification environment can be extended to the verification other Wishbone-compliant peripherals that support additional communication protocols.

REFERENCES

[1] A.K. Oudjida, M.L. Berrandjia, A. Liacha, R. Tiar, K. Tahraoui & Y.N. Alhoumays, “ Design and Test of General-Purpose SPI Master/Slave IPs on OPB Bus,” 7th International Multi-Conference on Systems, Signals and Devices, 2010.

[2] Freescale SemiconductorInc.,“MC68HC912D60 manual V 4.0,” September 2010.

[3] “Design and Implementation of Serial Peripheral Interface Protocol Using Verilog HDL”, International Journal of Engineering Development and Research, Volume 3, Issue 3, pp 2321-9939, 2015

[4] S.Sarns and J. Woehr, “Exploring I2C,” Embedded Systems Programming, vol. 4, p. 46, Sept. 1991.

[5] A. K. Oudjida, M. L. Berrandjia, R. Tiar, A. Liacha, K. Tahraoui, "FPGA implementation of I2C & SPI protocols: A comparative study," in Proc. 16th IEEE International Conference on Electronics, Circuits, and Systems, Dec. 2009, pp.507- 510.

[6] Konstantin Mikhaylov, J. T. and Fadeev, D. "development of energy efficiency aware applications using commercial low power embedded systems". In Embedded Systems - Theory and Design Methodology, 2012, pages 407–430.

[7] Žilvinas Nakuti, "Embedded Systems Power Consumption Measurement Methods Overview" . MATAVIMAI, 44(2):29–35, 2013.

[8] SPI WISHBONE Controller, Reference Design RD1044, Lattice Semiconductor , March 2014.

[9] F.Leens, “An Introduction to I2C and SPI Protocols,”IEEE Instrumentation & Measurement Magazine, February 2009, pp. 8-13.

[10] N. Anand, G. Joseph, S. S. Oommen, and R. Dhanabal, " Design and implementation of a high speed serial peripheral interface", International Conference on Advances in Electrical Engineering (ICAEE), IEEE, 2014, pp. 1-3.

[11] K. Aditya, M. Sivakumar, F. Noorbasha, and T. P. Blessington, "Design and functional verification of a spi master slave core using system verilog", International Journal of Soft Computing and Engineering, , vol. 2, no. 2, 2012, pp. 558-563.

[12] Manish Kundu, Abhijeet Kumar, “A Review on Low Power SPI Protocol”, International Journal of VLSI System Design and Communication Systems, Vol. 02, Issue. 03, 2014, pp. 0193-0195.