Scientific Journal of Impact Factor (SJIF): 5.71

e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 5, Issue 04, April -2018

DESIGN AND IMPLEMENTATION OF SERIAL PERIPHERAL INTERFACE

VADLA SIVA PRASAD[1], Shaik.TAJMAHABOOB[2]

[1]Student, Dept. of Electrical &Electronics Engineering, JNTUACEP, Pulivendula, A.P., India.
[2]Assistant Professor & Head Department of ECE, JNTUACEP, Pulivendula, A.P., India.

Abstarct : *SPI stands for Serial Peripheral Interface, A typical processor based system not only has to acquire data and process it, but also requires secondary devices like sensors, real time clocks, or communication channels to receive and pass process control information to and from other processors. SPI bus protocol is mainly used for IC to IC communication within the system. SPI provides synchronous serial communication between two IC's. SPI is developed by Motorola for Interfacing micro controllers, microprocessors and various devices such sensors, memory chips, printers, and data converters. SPI architecture consists of one Master and one or more slaves. The aim of the project is to develop SPI master-slave interface in verilog and verify the same using self checking test bench. Verilog standardized as IEEE 1164, is a Hardware Description Language (HDL). This paper presents a full explanation of a Serial Peripheral Interface bus protocol. The SPI Master and Slave can be designed and implemented on Spartan FPGA kit and results can be simulated using Xilinx ISE 12.4 simulator.*

Keywords: *Motorola, Serial Peripheral Interface(SPI), FPGA, Interfacing, Xilinx ISE 12.4*

I. INTRODUCTION

In any communication system requires a channel between the transmitter and receiver. There exists lot of communication channels. Channels are of two types. They are 1. Guided(wired) channels 2. Unguided(wireless) and further classifications of these channels given below in fig1.

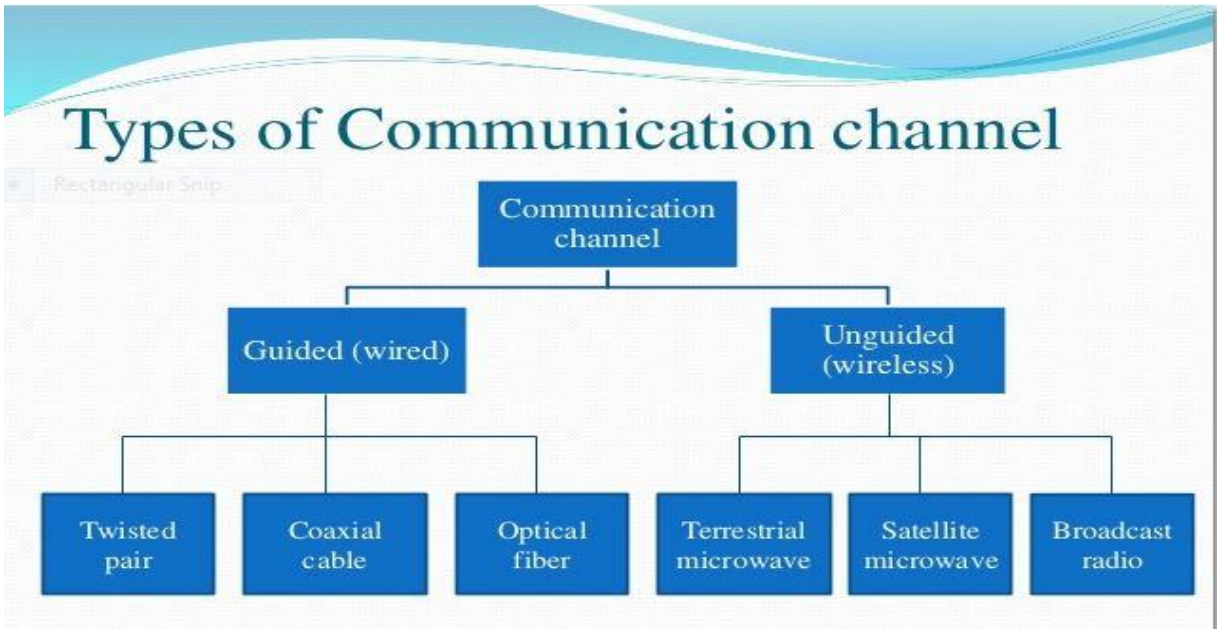

Fig: Different types of communication Channels

Others types are under water acoustic channels, storage channels like magnetic tapes, magnetic disks etc. The above provided channels can be used for system to system communication. A system consists of number of sub-systems such as microprocessors, micro-controller units (MCU), sensors, memories, etc. It is compulsory that , we have to provide communication between these sub-systems. Serial Peripheral Interface (SPI), Universal Asynchronous Receive and Transmit (UART), Inter-Integrated circuit (I2C)

Protocol are used for providing communication within the system (i.e inbetween the sub-systems). UART doesn't provide operation between more than two sub-systems. It is suitable for only between two sub-systems. In I2C protocol the complexity of the circuit is increased when increasing the no.of slave devices and master devices. I2C protocol provides only a half-duplex communication. All these drawbacks are overcome by using SPI protocol.

II. SPI PROTOCOL

SPI protocol was developed by Motorola. It also called as Microwire, MicrowirePlus, QSPI (queued SPI). SPI circuit implementation is very easy when compared to other protocols . It supports full-duplex mode in its operation**.** SPI protocol consists of one Master device and one or more slave devices. Since it operates in full-duplex mode that means at a time the data has been transferred from Master device to Slave and receiving data from Slave device . SPI offers a data rate upto 10 Mbps.since SPI is a serial synchronous control Protocol, due to this reason the transmission of data with more than 16 bits at a time can be implemented simply using control signals.

The SPI design consists of master- slave mode architecture as shown in fig2. The Slave device can be controlled by four control signals from the Master device. They are SCK (Serial Clock), MISO (Master In Slave OUT), MOSI (Master Out Slave In), SS (Slave Select). SPI is a Data Exchange protocol. As data is being clocked out, new data is clocked in. Data is exchanged - no device can just be a transmitter only or receiver only. The master controls the exchange by manipulating the clock line (SCK). There is only one Master device and number of Slave devices depending upon the number of chip select lines. Due to synchronous operation , latch is on for rising edge or falling edge of the clock that shows SDI (Serial Data In) is on rising edge of the clock and SDO (Serial Data Out) is on falling edge or vice versa.The range of operating frequency of Serial Peripheral Interface (SPI) is upto 2MHz. Master sends out Clock and chip select signals to the Slave device , these signals will activates the particular Slave it wants to communicate.

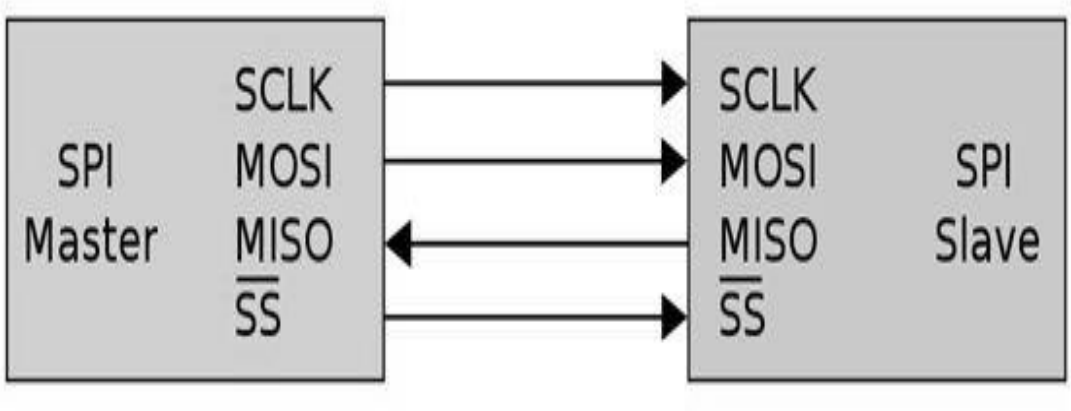

Fig1 : signal representation of Master and Slave devices

SCK (Serial Clock) is the signal which activates the clock signal for each slave device.

MOSI (Master Out Slave In) is the signal which carries the input data from the one IC (subsystem) to the Slave device , that is MOSI acts as a input to the Slave device as shown in fig2.

MISO (Master In Slave Out) is the signal which transmit the data from the another subsystem to the Master device, that MISO acts as a input to the Master device as shown in fig2.

SS is the signal which is used to select the particular Slave device the master wants to communicate. If mater pulls down the SS signal to low, then that particular Slave is selected and transferring of data takes place between the Master device and that particular Slave device. If Logic 0 is transmitted because the chip select signal is low means that its off state is Logic 1. If a waiting period is required then the master device must for atleast that period of time time before starting to issue the clocl cycles. Transmission of data normally may involve in two shift registers of some given word size,such as 8-bit, that is one byte, one byte is in the Master and the another byte is in the Slave; they are connected as the a ring. MSB bit of data is shifted out first, while shifting a new least significant bit into the same shift register. In this data is exchanged for each clock pulse between the two shift registers present in the Master and Slave devices. Still more data is there to transfer or exchange, these shift registers are loaded with new data byte and this process is repeats. These transmissions may involve in any number of clock periods. If there is no data to transfer, the master device stops toggling its clock. Normally, it now deselects the Slave device. To initiate all these processes the Master device must also be configured the clock phase (CPHA) and clock polarity (CPOL) with respect to the data.These CPOL and CPHA are present in the SPI control register.This timing applies to both the Slave device and the Master device. This combination of clock phase(CPHA) and clock polarity (CPOL)gives the different modes of operation shown in table1.

Table 1.Different modes of clock

Mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

If CPOL is Logic Low (i.e Logic 0) then the master device initiate the transfer of data from the rising edge of the clock period. If CPOL is Logic High (i.e Logic 1) then the master device initiates the transfer from the falling edge of the clock period.

If CPHA is Logic Low (i.e Logic 0) then the master device samples and sends the data during first rising or falling edge of the clock. If CPHA is Logic High(i.e Logic 1) then master initiates the sampling and transferring of data during second rising or falling edge of the clock. Like that with the combination of CPOL and CPHA there exist four modes of operations given below in the figures

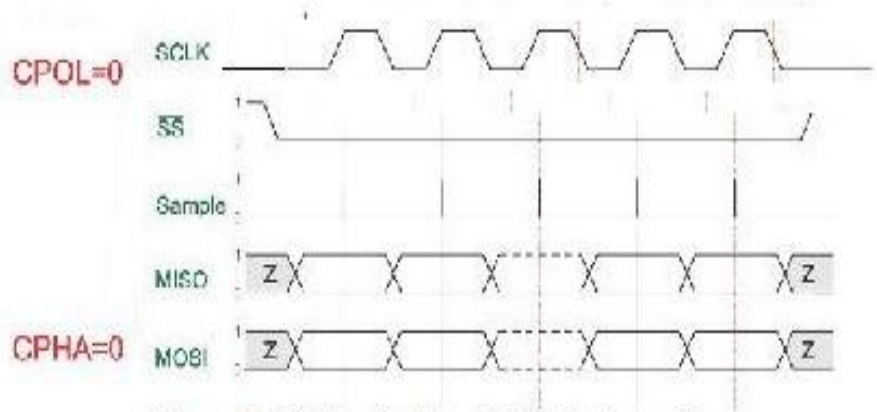

Figure 2: Clock polarity = 0, Clock phase = 1

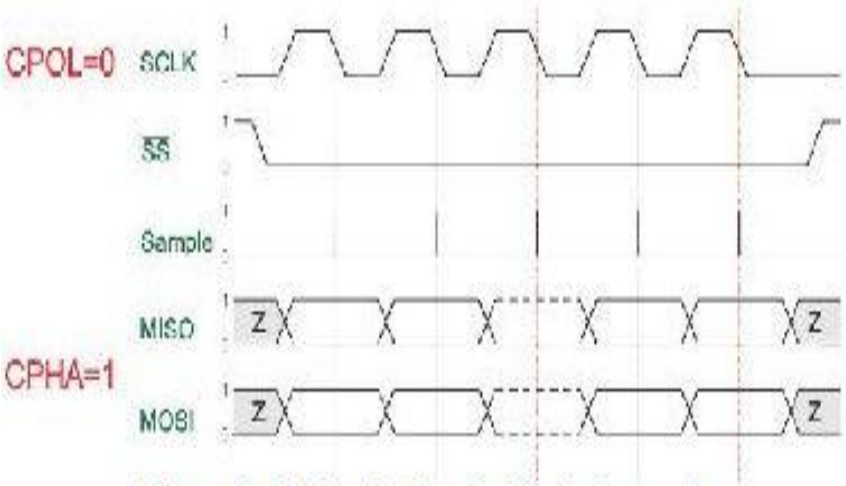

Figure 3: Clock polarity = 0, Clock phase = 1

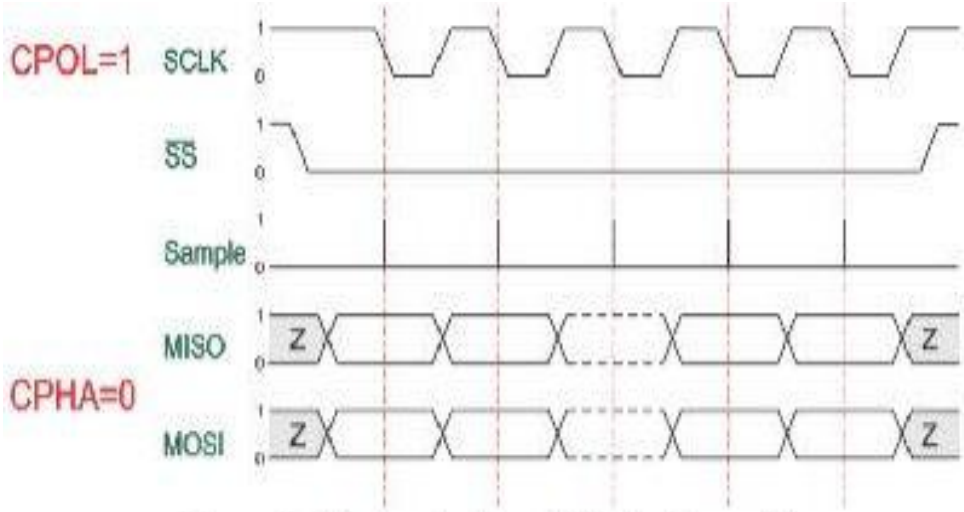

Figure 4: Clock polarity = 1, Clock phase = 0

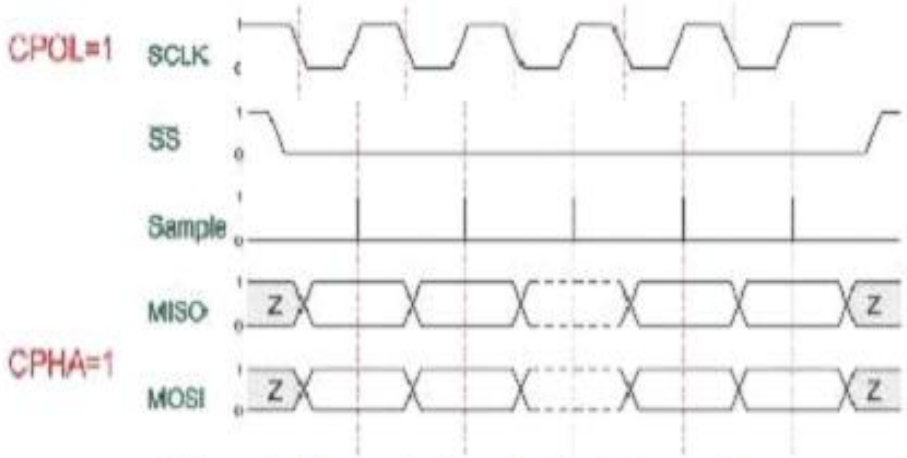

Figure 5 : Clock polarity = 1, Clock phase = 1

In the above figures Z indicates the undefined states or high impedance state.

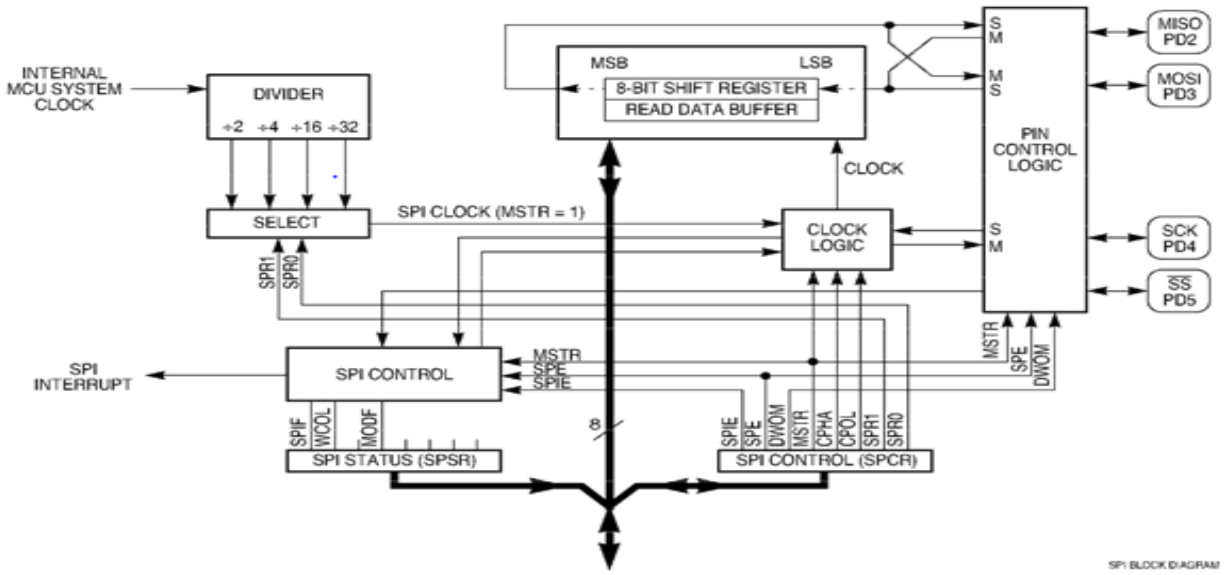

Fig6: Block diagram of the Serial Peripheral Interface

SPI consists of Control Register (SPICR), Data Register(SPIDR) and Status Register(SPISR). These three registers will control the transferring of data from one Sub-system(IC) to another Sub-system(IC).

SPI CONTROL REGISTER (SPICR): Read: anytime Write: anytime

SPIE — SPI Interrupt Enable Bit. This bit enables SPI interrupt requests, if SPIF or MODF status flag is set.
1 = SPI interrupts enabled 0 = SPI interrupts disabled.

SPE — SPI System Enable Bit. This bit enables the SPI system and dedicates the SPI port pins to SPI system functions. If SPE is cleared, SPI is disabled and forced into idle state and status bits in SPISR register are reset 1 = SPI enabled, port pins are dedicated to SPI functions. 0 = SPI disabled (lower power consumption).

SPTIE — SPI Transmit Interrupt Enable This bit enables SPI interrupt requests, if SPTEF flag is set.
1 = SPTEF interrupt enabled. 0 = SPTEF interrupt disabled.

MSTR — SPI Master/Slave Mode Select Bit. This bit selects, if the SPI operates in master or slave mode. Switching the SPI from master to slave or vice versa forces the SPI system into idle state.
1 = SPI is in Master mode 0 = SPI is in Slave mode

CPOL — SPI Clock Polarity Bit. This bit selects an inverted or non-inverted SPI clock. To transmit data between SPI modules, the SPI modules must have identical CPOL values. In master mode, a change of this bit will abort a
transmission in progress and force the SPI system into idle state.
1 = Active-low clocks selected. In idle state 0 = Active-high clocks selected. In idle state SCK is low.

CPHA — SPI Clock Phase Bit. This bit is used to select the SPI clock format. In master mode, a change of this bit will abort transmission in progress and force the SPI system into idle state
1 = Sampling of data occurs at even edges (2,4,6,...,16) of the SCK clock
0 = Sampling of data occurs at odd edges (1,3,5,...,15) of the SCK clock

Register Address: $___0

	Bit 7	6	5	4	3	2	1	Bit 0
R W	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
Reset:	0	0	0	0	0	1	0	0

Fig : SPI control register

SPI STATUS REGISTER (SPISR):

Register Address: $___3

	Bit 7	6	5	4	3	2	1	Bit 0
R	SPIF	0	SPTEF	MODF	0	0	0	0
W								
Reset:	0	0	1	0	0	0	0	0

= Reserved

Fig : SPI status register

Read: anytime Write: has no effect

SPIF — SPIF Interrupt Flag. This bit is set after a received data byte has been transferred into the SPI Data Register. This bit is Cleared by reading the SPISR register (with SPIF set) followed by a read access to the SPI Data Register.
1 = New data copied to SPIDR 0 = Transfer not yet complete

SPTEF — SPI Transmit Empty Interrupt Flag. If set, this bit indicates that the transmit data register is empty. To clear
this bit and place data into the transmit data register, SPISR has to be read with SPTEF=1, followed by a

write to SPIDR. Any write to the SPI Data Register without reading SPTEF=1, is effectively ignored.
1 = SPI Data register empty 0 = SPI Data register not empty

MODF — Mode Fault Flag. This bit is set if the SS input becomes low while the SPI is configured as a master and
mode fault detection is enabled, MODFEN bit of SPICR2 register is set. The flag is cleared automatically by
a read of the SPI Status

Register (with MODF set) followed by a write to the SPI Control Register 1.
1 = Mode fault has occurred. 0 = Mode fault has not occurred

SPI DATA REGISTER (SPIDR):

Register Address: $__5

	Bit 7	6	5	4	3	2	1	Bit 0
R W	Bit 7	6	5	4	3	2	2	Bit 0
Reset:	0	0	0	0	0	0	0	0

Fig : SPI Data register

Read: anytime; normally read only after SPIF is set Write: anytime

The SPI Data Register is both the input and output register for SPI data. A write to this register allows a data byte to be queued and transmitted. For a SPI configured as a master, a queued data byte is transmitted immediately afte r the previous transmission has completed. The SPI Transmitter Empty Flag SPTEF in the SPISR register indicates when the SPI Data Register is ready to accept new data. Reading the data can occur anytime from after the SPIF is set to before the end of the next transfer. If the SPIF is not serviced by the end of the successive transfers those data bytes are lost and the data within the SPIDR retains the first byte until SPIF is serviced.

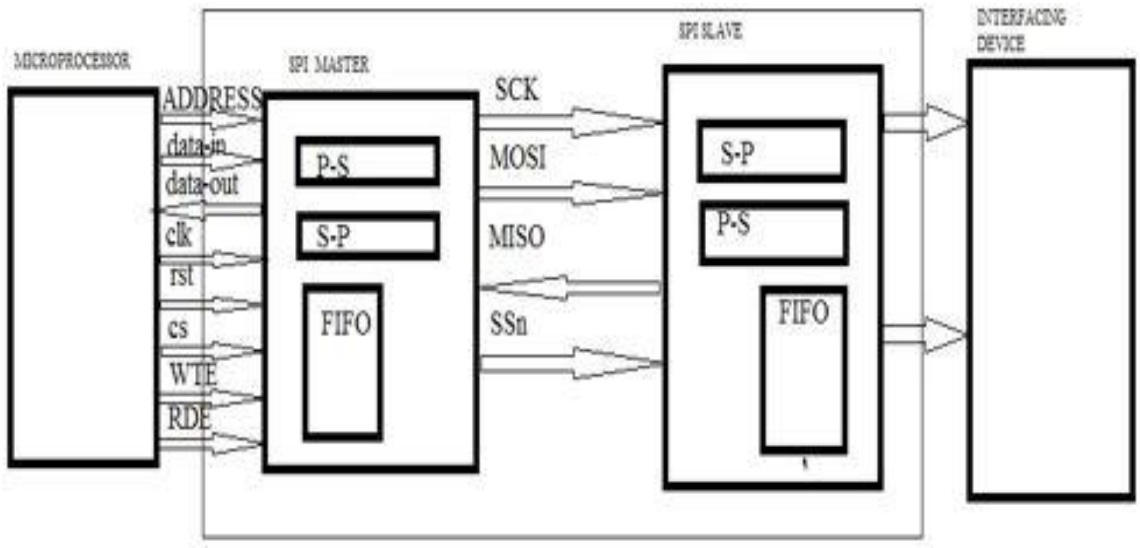

Fig7: Interfacing of two sub-systems through SPI Controllers.

Fig7 shows that how two IC's or sub-systems communicates. One device send address of the second device (or subsystem) to the SPI controller for reading data from the second device or writing the data to the second sub-system. Data-in, data-out are the input and and output signals to the SPI master device. Clk and rst are the inputs to set or reset the SPI controller. CS (chip select) is the signal to the SPI controller which selects or deselect the SPI controller. WTE (write enable) signal is used to write the date to the IC. RDE (read enable) signal is the signal used to read data from the second IC (sub-system).

III.KEY FEATURES OF SPI CONTROLLER

SPI controller offers multiple number of data rates depending on the baud rate of the SPI master module. SPI has data transfer rate upto 2-10 Mbps. SPI controller can supports multi-slave operation. The master and slave device can acts as atransmitter or receiver based on its mode of operation. It is capable of receiving or transmitting on both the rising edge and falling edges of the clock independently.

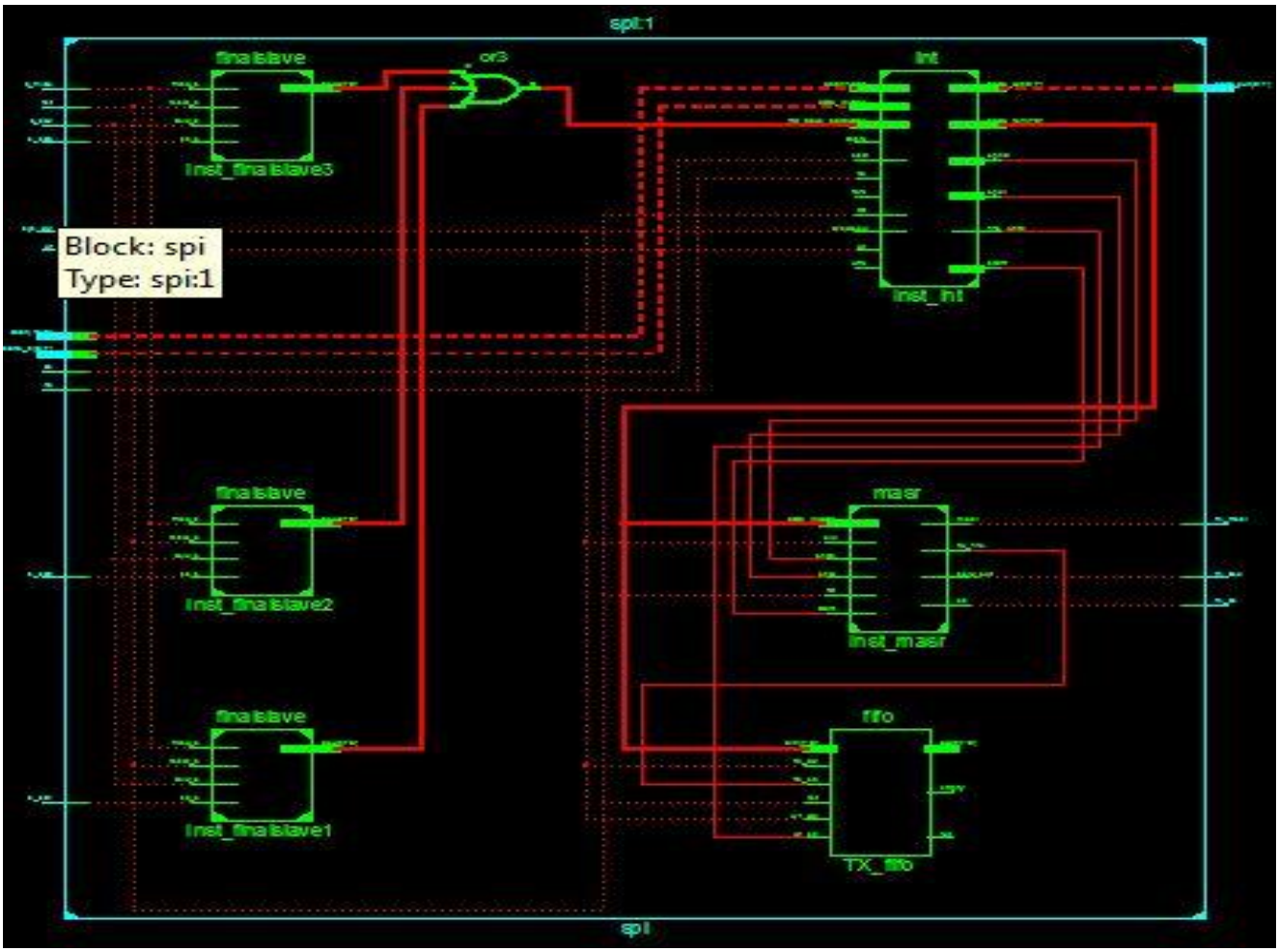

Fig8: Implementation SPI in Xilinx ISE

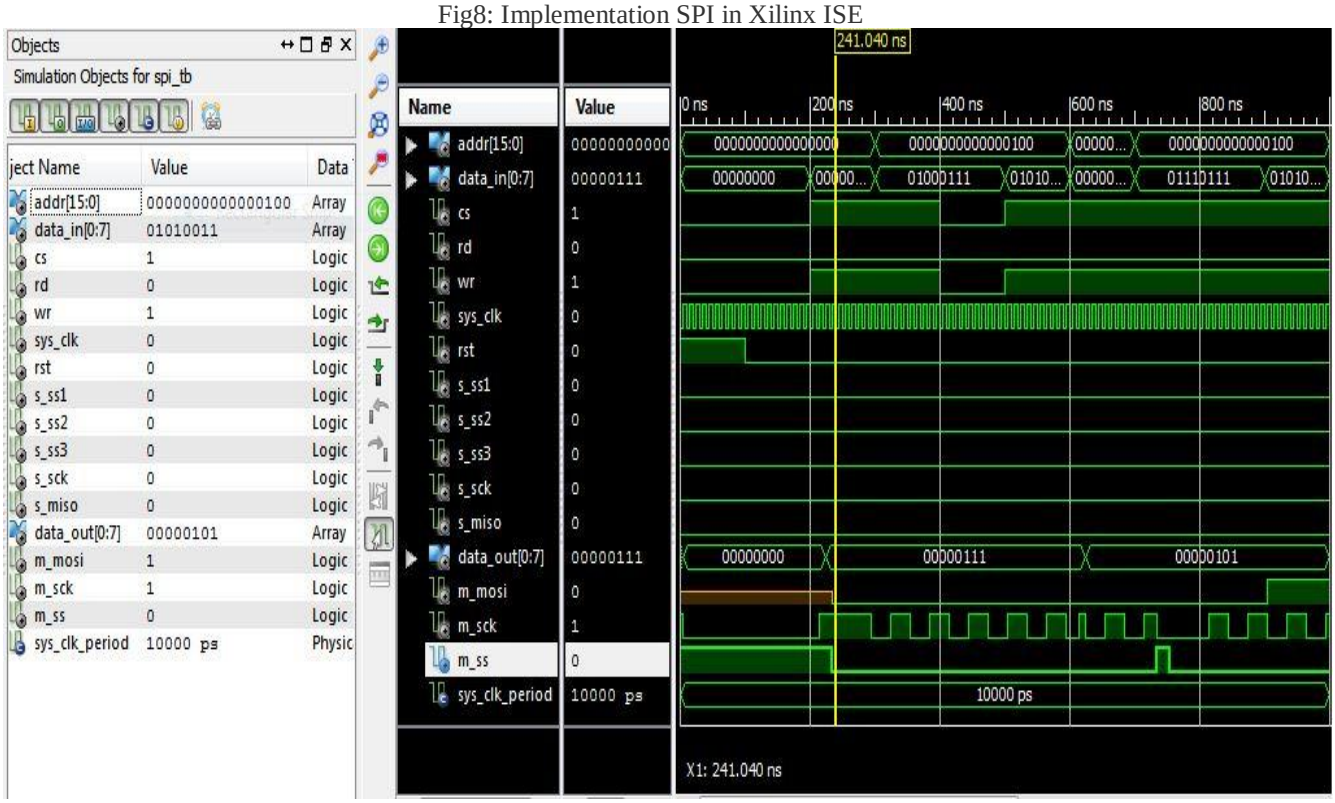

Fig9: Waveforms of SPI controller in Xilinx ISE

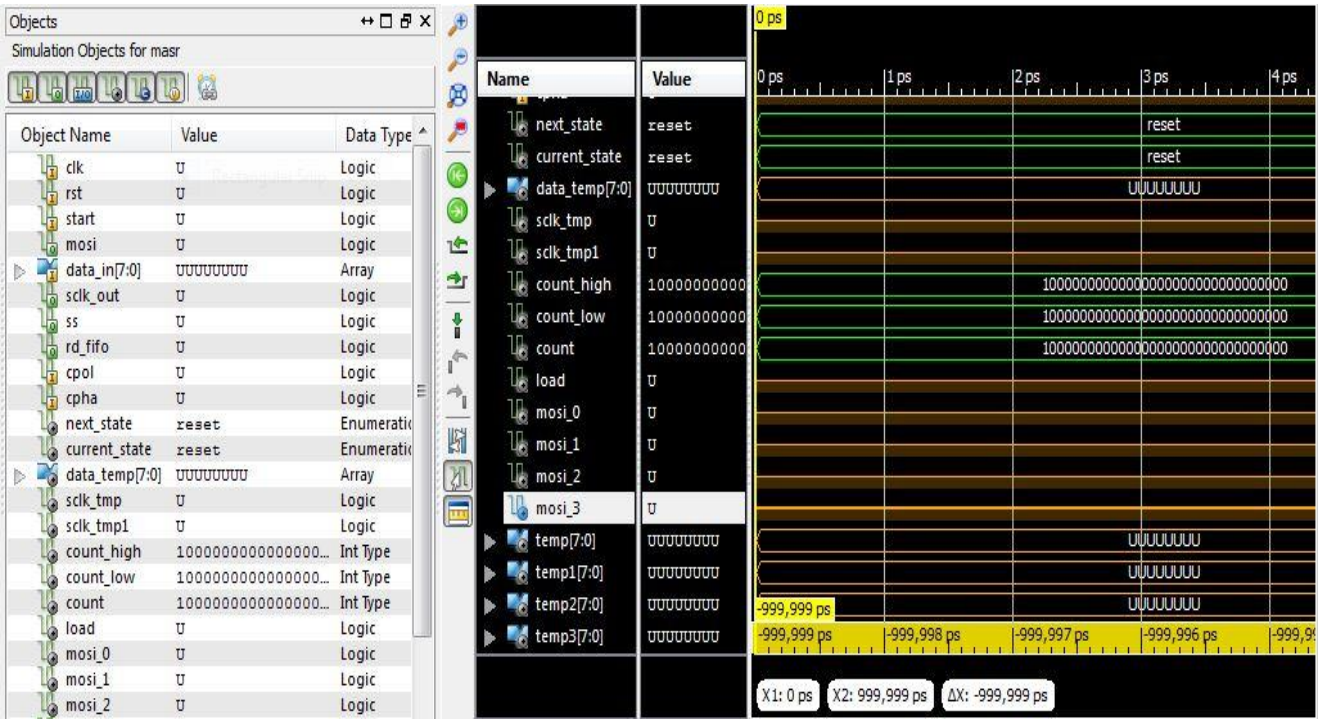

Fig9: Waveforms of SPI controller in Xilinx ISE

IV.CONCLUSION

In this paper SPI master and slave devices are implemented in Xlinx ISE tool using VHDL programming code. Our work is completely focussed on the performance analysis of SPI along with the RTL design work. Design part of SPI controller involves around Motorola 3.0 specifications. This SPI Master device is a flexible programmable logic components that accommodates communication with number Slaves via single parallel interface. The SPI slave device can offers a user definable mode, data and status signals for transmit ready, receive ready, and receive over run errors.

References:

1. F.Leens, "An Introduction to I2C and SPI Protocols,"IEEE Instrumentation & Measurement Magazine, pp. 8-13, February 2009.
2. "Design and Implementation of a Reused Interface" 978-0-7695-3887- IEEE.September 7, 2009
3. Bhaskar. J, A Verilog HDL Primer, Publisher: Syndicate Publications.
4. IP Design of Universal Multiple Devices SPI Interface Tianxiang Liu1, Yunfeng Wang1 * Department of Electronic Engineering, Xiamen University,2011 IEEE.
5. ZHANG Yan-wei, Verilog HDL detailed design procedure, Posts & Telecom Press

