Design and Implementation of SPI IP core for ARM based SoC

Vidyasaraswathi H N, Veena H S

Abstract— The Serial Peripheral Interface (SPI) is a full duplex, serial data link that is standard across several processors and peripherals. It enables data communication between processor and a peripheral or between processors. This 4-wire protocol has MISO, MOSI, SCLK and slave select pins. The master can transmit data at different baud rates. AHB (Processor Interface) is the proprietary bus standard of ARM. It is high-bandwidth full duplex systems interconnect with parallel transfer capability. It also provides several configurable options. Now a days System design becomes more and more complex, SoC design plays most predominant role in current silicon market. Advantages of SoC include the conservation of on-chip space, reduced parasites, and savings on package costs. ARM processor is widely used in SoC design. SPI protocol is essentially preferable in SoC Design, used for serial communication between inter blocks and also outside the block. To meet timing constraint, companies will prefer readily available IP cores for their design. IP cores save both time and cost. This work mainly focuses on such one IP core SPI-AHB bus for ARM based SoC design.

Keywords—AMBA* (*Advanced Microcontroller Bus Architecture), AHB*(*Advanced High Performance Bus), ARM, SPI* (Serial Peripheral Interface), *SoC(system on chip).

I. INTRODUCTION

The Serial Peripheral Interface (SPI) master is a full duplex, serial data link that is standard across several processors and peripherals. It enables data communication between processor and a peripheral or between processors. This 4-wire protocol has MISO (Master In Slave Out), MOSI (Master Out Serial In), SCLK (Serial Clock) and SS_N (slave select) pins. SS_N can be extended to select four slaves. The SCK control line is driven by the SPI master and regulates the flow of data bits. The master may transmit data at a variety of baud rates; the SCK line transitions once for each bit in the transmitted data. The data can be transmitted either MSB or LSB first. The SPI specification allows a selection of clock polarity and a choice of two fundamentally different clocking protocols on an 8-bit oriented data transfer. Data is shifted on one edge of SCK and is sampled on the opposite edge when the data is stable. The MOSI data line carries the output data from the master which is shifted as the input data to the selected slave. The MISO data line carries the output data from the selected slave to the input of the master. The SPI system is flexible enough to interface directly with numerous commercially available peripherals.

Manuscript received on April, 2013.

Vidyasaraswathi H N, M.Tech, VLSI DESIGN, Dept of E & C, BIT, Bangalore, India.

Veena H S, Associate.Prof, Dept of E & C, BIT, Bangalore, India.

The processor can be used in several applications like digital camera, Net cam, flash memory card etc. The master can transmit data at different baud rates up to 60Mbps

Advanced Microcontroller Bus Architecture (AMBA) specification defines on-chip communication standard for designing high-performance embedded microcontrollers. The AMBA AHB is for high performance, high clock frequency system including burst transfer, split transaction [4]. The AHB acts as a high-performance system backbone bus. The AHB supports the efficient connection of processors, on chip-memories and off-chip external memory interfaces. The typically AMBA-AHB system design contains the fallowing components.

- **AHB master:** A bus transfer is able to initiate read and write operations by providing an address and control information. Only one bus master is allowed at a time.
- **AHB SLAVE:** A bus slave responds to a read or writes operation within a given address range. The bus slave signals back to the active master the success, failure or waiting of the data transfer.
- **AHB ARBITER:** Used for multi master mode.
- **AHB DECODER:** Used to decode the address of each transfer and provide select signal for the slave that is involved in the transfer.

II. FUNCTIONAL BLOCK DIAGRAM

The functional block diagram is as shown in Fig.1

The data from the master (processor) on the AHB bus is received by the AHB slave and given to the SPI master. In case it is to be transmitted out of the SPI master then the data register of the SPI has to be addressed. It is assumed that the remote slave devices understand a protocol for the read and write operations and the AHB master sends data based on that protocol. The core does not interpret the data. Burst mode read and writes and split transaction read and writes transactions supported. AHB slave complies with the processor interface protocol of ARM processor.

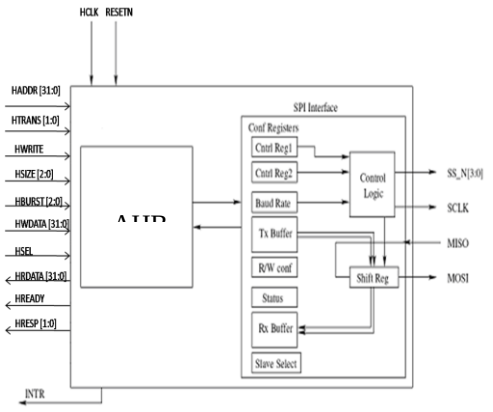

Fig.1. Functional block diagram

A. AHB Slave

This is ARM's proprietary interface. The slave device supports read and write transactions. The protocol is completely handshake based. The data received on the AHB bus is written into the specified SPI registers (conf registers / data register), in case of a write transaction. In case of read request, the slave responds with the requested data. There is a minimum of one clock cycle latency.

B. Control Logic

Control logic implements a state machine to handle the Tx and Rx operations of the SPI master. The data to be transferred is loaded into the output shift register from the data register (This is implemented as Tx buffer.) and the incoming data from the input shift register is loaded back to the receive buffer. The address sent by the master is decoded and the corresponding slave is selected. The serial clock on the SPI bus is generated here.

C. Tx and Rx Buffers

These are configurable FIFOs. Each of these is used as an interface between the PIF slave and the SPI master. Data from the PIF slave that is to be sent over the SPI bus is put into the Tx buffer and loaded onto the transmit shift register for serialization. The data received from the SPI slave is loaded into the Rx buffer. This data in Rx buffer is transmitted to PIF slave if it is a read operation.

D. Input And Output Shift Registers

These are the shift registers working at SCLK, used to shift the data serially in and out of the SPI master.

III. HARDWARE INTERFACE

Table.1 shows the Clocks and reset pins

Table 1: Clocks and Resets

No	Clock name	Frequency	Functional blocks driven by clock
01	SCLK	25MHz nominal	SPI Master
02	i_hclk		AHB Slave
03	i-rstn		Reset

Table.2 shows the pin description of the SPI core

Table.2

No	Signals	Size	Direction	Description
01	MOSI	01	Output	Master Out serial in
02	MISO	01	Input	Master In Serial Out
03	SCLK	01	Output	SPI Serial Clock
04	SS_n	01	Output	SPI Slave select
05	HADDR	32	Input	The 32 bit system addresses bus.
06	HTRANS	2	Input	Indicates the type of current transfer, which can be nonsequential, sequential, idle or busy .
06	HWRITE	1	Input	When HIGH this signal indicates a write transfer and when LOW a read operation.
07	HSIZE	3	Input	Indicates the size of transfer, which is typically byte, half word or word.
08	HBURST	3	Input	Indicates if the transfer forms part of burst. Four, eight and sixteen beat bursts are supported.
09	HWDATA	32	Input	The write data bus is used to transfer data from the master to the bus slave during write operation.
10	HSEL	1	Input	AHB slave select signal
11	HRDATA]	32	Output	The read data bus is used to transfer data from bus slave to the bus master during read operation.
13	HREADY	1	Output	When HGH indicates that a transfer has finished on the bus.
14	HRESP	2	Output	The transfer response provides additional information on the status of a transfer four different responses are provided, OKAY, ERROR, RETRY and SPLIT.

IV. TIMING DIAGRAM

The timing diagram for both AHB and SPI block is as explained below. The data from the master (processor) on the AHB bus is received by the AHB slave and given to the SPI master. In case it is to be transmitted out of the SPI master then the data register of the SPI has to be addressed. It is assumed that the remote slave devices understand a protocol for the read and write operations and the AHB master sends data based on that protocol. The core does not interpret the data. Burst mode read and writes and split transaction read and writes transactions supported. AHB slave complies with the processor interface protocol of ARM processor

A. AHB Interface

An AHB data transfer has two sections, Address phase lasts for only single cycle and data phase which may require several cycles. This is achieved using HREADY signal. The master drives the address and control signals on to the bus after the rising edge of HCLK. The slave then samples the address and control information on the next rising edge of the clock. After the slave has sampled the address and control it can start to drive the appropriate response and this is sampled by the bus master on the third rising edge of the clock. Figure.2 shows the timing diagram of AHB simple transfer.

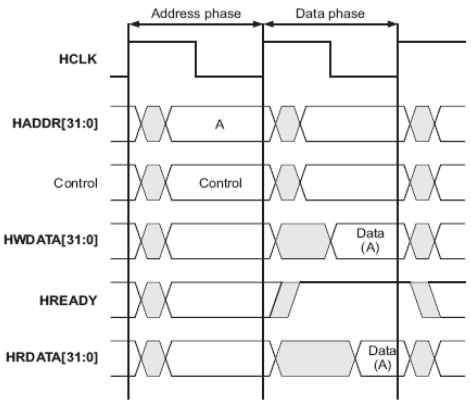

Fig. 2.Simple transfer

B. SPI Interface

The data transmission is as shown. The slave select signal is de-asserted to select a particular slave and depending on the clock polarity and phase the data is transmitted as shown in the following figures. Fig. 3 shows the data transfer on the SPI bus with CPHA = 0.

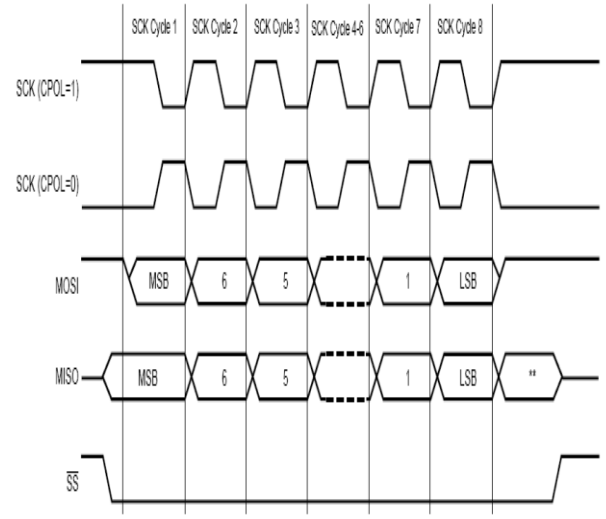

Fig. 3 Data transfer on the SPI Bus with CPHA = 0

Fig. 4 shows the data transfer on SPI data bus with CPHA is equal to 1.

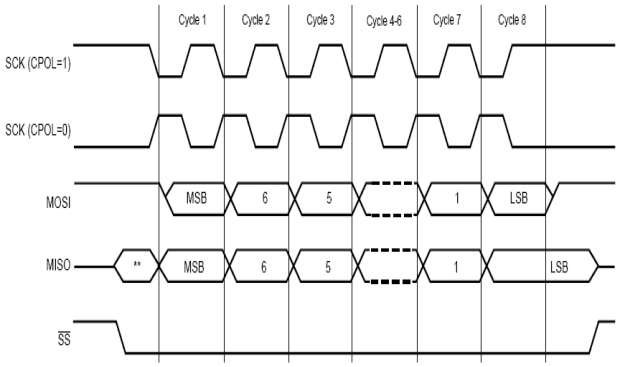

Fig. 4 Data transfer on the SPI Bus with CPHA = 1

SPI protocol has also supports data transmission using with or without streaming. Fig. 5 Data transfer on the SPI Bus with breaks in SS_B signal.

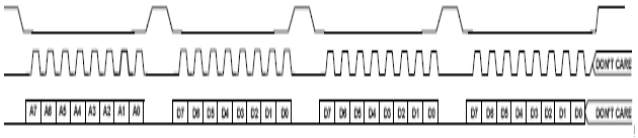

Fig. 5 Data transfer on the SPI Bus with breaks in SS_B signal.

Fig.6 shows the streaming data on the SPI bus (notice that SS_B signal is constantly low between bytes)

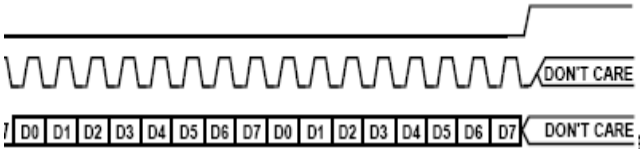

Fig. 6 streaming data on the SPI bus

V. REGISTER DESCRIPTION

Table.3 shows the register map of SPI-AHB IP core

Table 3: Register Map

Address	Register Name	Type	Default value	Description
0x0000	SPICR	R/W	32'h00	Control Register
0x0004	SPISR	R	32'h20	Status Register
0x0008	TXBUF	R/W	32'h00	Transmit Buffers (size 16)
0x000C	RXBUF	R	32'h00	Receive Buffers (size 16)
0x0010	SLV_ADDR	R/W	32'hffffffff	Slave address register.
0x0014	Reserved	R/W	32'h0	Reserved
0x0018	Reserved	R/W	32'h0	Reserved

A. Control Register (SPICR)

Table.4 shows the register bit map and Table.5 register bit description of SPICR

Table 4: Register bit map

SPICR (0x0000)							
Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
0	0	0	0	0	0	0	0
Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
SPR7	SPR6	SPR5	SPR4	SPR3	SPR2	SPR1	SPR0
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Streaming	0	Flush fifo	Buf_hdl rst	PIF rst	SPI rst	SWAI	0
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE

Table 5: Register bit description

Bit	Field Name	Type	Default value	Description
23	SPR7	R/W	0	SPI Baud rate setting
22	SPR6	R/W	0	SPI Baud rate setting
21	SPR5	R/W	0	SPI Baud rate setting
20	SPR4	R/W	0	SPI Baud rate setting
19	SPR3	R/W	0	SPI Baud rate setting
18	SPR2	R/W	0	SPI Baud rate setting
17	SPR1	R/W	0	SPI Baud rate setting

16	SPR0	R/W	0	SPI Baud rate setting
15	Streaming	R/W	0	Enable SPI streaming
14	Reserve bit			Reserve bit
13	Flush Fifo	R/W	0	Soft reset to flush FIFO
12	Buf_hdl rst	R/W	0	Soft reset for buffer handler
11	PIF rst	R/W	0	Soft reset for PIF slave
10	SPI rst		0	Soft reset for SPI master
9	SPISWAI	R/W	0	SPI Stop in Wait Mode bit
8	Reserve bit	R/W		Reserve bit
7	SPIE	R/W	0	SPI Interrupt Enable bit
6	SPE	R/W	0	SPI System Enable bit
5	SPTIE	R/W	0	SPI Transmit Interrupt Enable
4	Reserve Bit			Reserve Bit
3	CPOL	R/W	0	SPI Clock Polarity bit
2	CPHA	R/W	0	SPI Clock Phase bit
1	Reserve bit			Reserve bit
0	LSBFE	R/W	0	LSB – First Enable

B. Status Register (SPISR)

Table.6 shows the register bit map of status register (SPISR).

Table 6: Register bit map

SPISR (0X0004)							
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
0	0	Rx int	Rx fifo cnt[4]	Rx fifo cnt[3]	Rx fifo cnt[2]	Rx fifo cnt[1]	Rx fifo cnt[0]
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SPI intr	Rx Full	Tx Empty	0	Spi busy	Spi_data_lost_err	0	0

Table.7 shows the register bit description of SPISR.

Table 7: Register bit description

Bit	Field Name	Type	Default value	Description
15	Reserved Bit		0	Reserved Bit
14	Reserved Bit		0	Reserved Bit
13	Rx int	I	0	Receive interrupt
12	Rx fifo cnt[4]	R	0	Rx FIFO count
11	Rx fifo cnt[3]	R	0	Rx FIFO count
10	Rx fifo cnt[2]	R	0	Rx FIFO count
9	Rx fifo cnt[1]	R	0	Rx FIFO count
8	Rx fifo cnt[0]	R		Rx FIFO count
7	SPI intr	I	0	SPI interrupt flag
6	Rx Full	I	0	Receive buffer full flag
5	Tx Empty	I	1	Transmit buffer empty flag
4	0	R	0	0
3	SPI Busy	R	0	Data is being transmitted on SPI Bus
2	Spi_data_lost_err	R	0	SPI data lost error (if Rx Buf is full and there is a write)
1	Reserve bit		0	Reserve bit
0	Reserve bit		0	Reserve bit

C. Slave Address Register (SLV_ADDR)

Table.8 shows the register bit map of slave address register (SLV_ADDR).

Table 8: Register bit map

SLV_ADDR (0x0010)							
Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
Slave address 3							
Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
Slave address 2							
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Slave address 1							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Slave address 0							

Table.9 gives the register bit description of Slave-address register

Table 9: Register bit description

Bit	Field Name	Type	Default value	Description
31:24	Slave address 3	R/W	0xff	Slave 3 address
23:16	Slave address 2	R/W	0xff	Slave 2 address
15:8	Slave address 1	R/W	0xff	Slave 1 address
7:0	Slave address 0	R/W	0xff	Slave 0 address

VI. EXPERIMENTS

The individual models of the core are implemented in verilog. DUT mainly contains two parts, SPI master and AHB slave. For verification of core, two BFM's are used, AHB BFM at the input side and SPI BFM at the output side. Fig. 7 shows the core with two BFM's for verification.

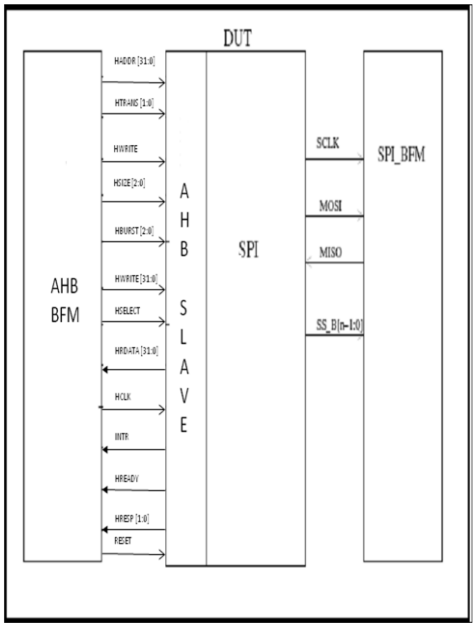

Fig. 7

A. AHB BFM:

AHB BFM mainly generates and drives inputs to AHB-SPI DUT .It supports single byte read/write and multibyte read/write data transfer. This act as a master and control over the AHB part in the DUT. An AHB data transfer has two sections, Address phase lasts for only single cycle and data phase which may require several cycles. This is achieved using HREADY signal. The master drives the address and control signals on to the bus after the rising edge of HCLK. The slave then samples the address and control information on the next rising edge of the clock. After the slave has sampled the address and control it can start to drive the appropriate response and this is sampled by the bus master on the third rising edge of the clock. The timing diagram for simple AHB transfer is as shown in figure.4. The AHB BFM generates the outputs are given to the input of the AHB pins of the DUT.

B. SPI BFM

It mainly acts as a Receiver to the SPI DUT. It receives serial data from MOSI pin of DUT and converts it to parallel data. It will also send data to the DUT during full duplex mode or read mode.

VII. RESULTS

Core has been tested for main specifications of SPI like, different modes of the SPI, Different baud rates, LSB transmitted first, MSB transmitted first etc. Similarly AHB slave tested for both split and burst transfers.

The simulations of specified functions were conducted by the software Icaurus verilog. [4]. Fg. 8 shows the simulation result of CPOL = 0 and CPOH =0

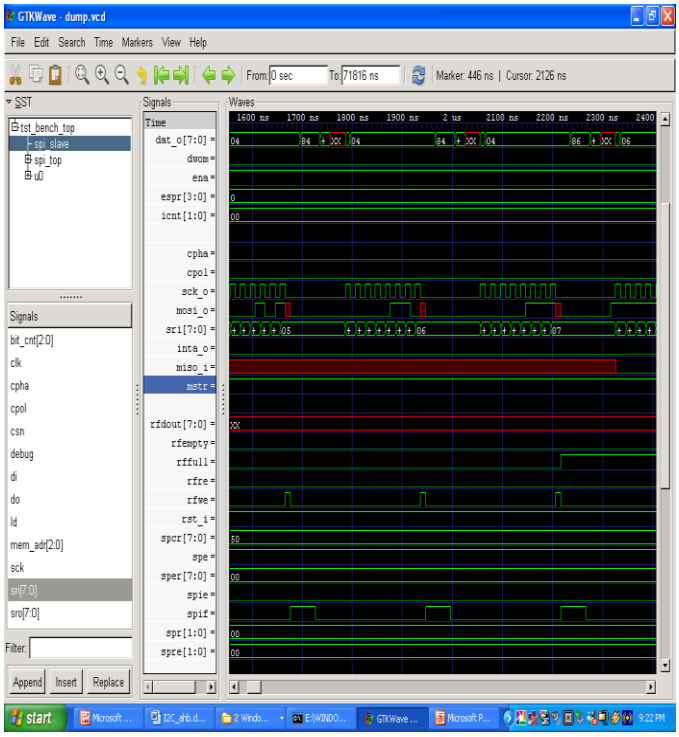

Fig.8 simulation result of CPOL = 0 and CPOH =0

Fig. 9 shows the simulation result of CPOL =0 and CPOH =1for bidirectional mode. The DUT sends serial data through MOSI pin and simultaneously SPI BFM sends serial data to the MISO pin of the DUT.

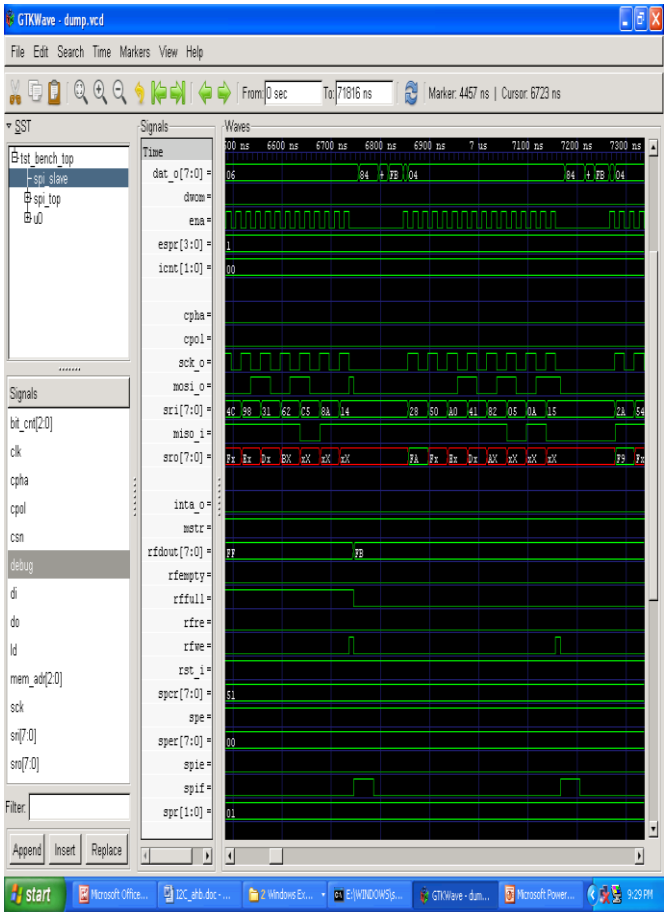

Fig. 9 simulation result of CPOL = 0 and CPOH =1

Fig.10 shows the simulation result of CPOL = 1 and CPOH =0

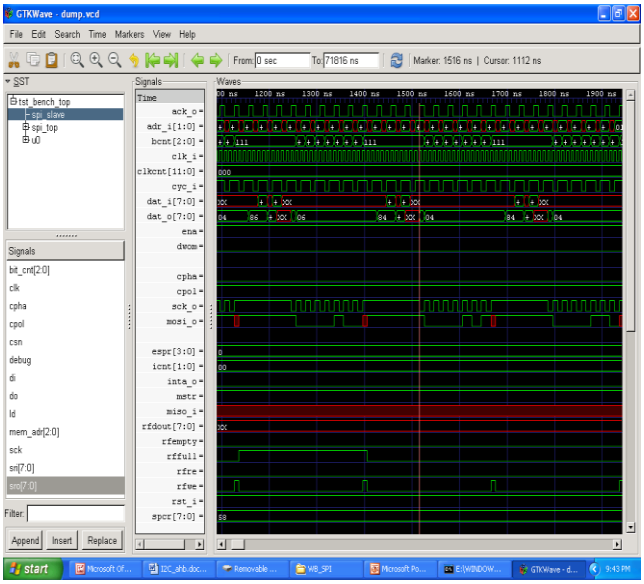

Fig. 10 simulation result of CPOL = 1 and CPOH =0

Fig.11 shows the simulation result of CPOL = 1 and CPOH =1

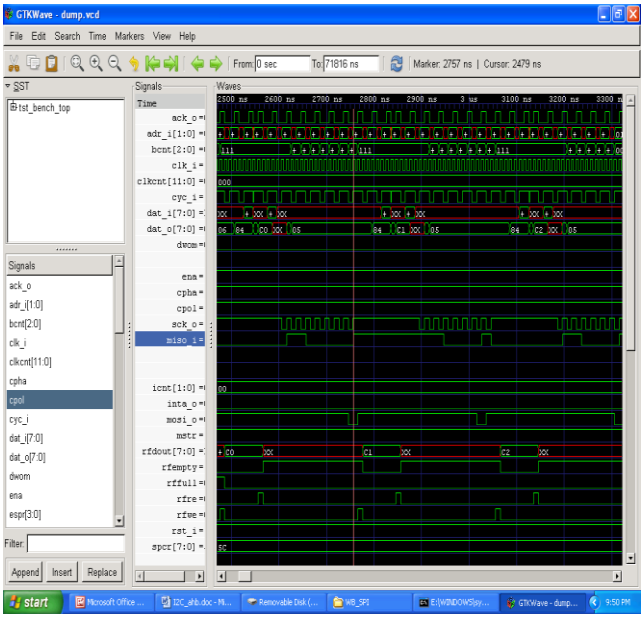

Fig. 11 simulation result of CPOL = 1 and CPOH =1

Our implementation of the design was analyzed by using ISE FPGA tool from Xilinx [2]. Designs are mapped on to Spartan FPGA. The whole system has been placed and routed in to the XC3S500e-5-PQ208 FPGA chip. The numbers of slices utilized by the design are 1340 out of 4656, 28% of the utilization.

VIII. CONCLUSION

The AMBA AHB-SPI IP Core provides a high-speed data communication channel between the AHB and SPI buses. The SPI - AHB bridge enables an AHB host to access a serial device at high-speed through the SPI interface. The controller can be used in applications such as flash memory card and digital camera. Both AHB and SPI support master and slave modes. The AHB - SPI bridge performs either parallel-to-serial conversion or serial-to-parallel conversion. Full SPI duplex mode is supported. The AMBA AHB-SPI IP Core is an RTL design in Verilog that implements an SPI - AHB controller on an ASIC, or FPGA.

REFERENCES

[1]. A. K. Oudjida, M. L. Berrandjia, A. Liacha, et al. “Design and test of general-purpose SPI master/slave IPs on OPB bus,▫ 2010 7th International Multi-Conference on Systems Signals and Devices (SSD). 2010, pp. 1-6.

[2] J. M. Xing, P. He, Z. C. Wang, et al. “Design of data acquisition system based on AD7367 and TMS320F2812,” 2010 International Conference on Computational and Information Sciences (ICCIS), 2010, pp. 380-383.

[3] X. H. Chen, D. Y. Zhang, and H. Y. Yang, “Design and implementation of a single-chip ARM-based USB interface JTAG emulator,” Fifth IEEE International Symposium on Embedded Computing, 2008, pp. 272-275.

[4] B. Qu, D. W. Fan, “Design of remote data monitoring and recording system based on ARM,” 2010 2nd International Conference on Industrial and Information Systems (IIS), 2010, pp. 252-255.

[5] J. M. Ma, L. H. Jiang, W. Du, et al. “The embedded systems based M. CORE microcontroller,” Beijing: National Defence Industry Press, 2003, pp. 141-143. (in chinese)

[6] M. Turner, J.Naber, “The design of a bi-directional, RFID-based ASIC for interfacing with SPI bus peripherals,” 2010 53rd IEEE International Midwest Symposium on Circuits and Systems (MWSCAS), 2010, pp. 554-557.

[7] A. K. Oudjida, M. L. Berrandjia, R. Tiar, et al. “FPGA implementation of I2C & SPI protocols: A comparative study,” 2009 16th IEEE International Conference on Electronics, Circuits, and Systems(ICECS), 2009, pp. 507-510.

[8] Verilog HDL: A guide to digital design and synthesis, Samir Palnitkar,2nd edition, Pearson Education.

[9] AMBA Specification (Rev 2.0), ARM Limited 1999.

[10] A Verilog Primer, J Bhasker, 2nd edition, BS Publication Samir Palnitkar, Pearson 2nd edition “Verilog HDL, A Guide to Digital Design and Synthesis**:**

Vidyasaraswathi H N Obtained her B.E degree from GECR Ramanagar, of Visveswaraya Technological University, Belgaum in the year 2011 and currently pursuing M.Tech in VLSI Design and Embedded Systems from Bangalore Institute of Technology, Bangalore.

Veena H S She is graduated B.E from BIT Bangalore, of Bangalore University in the year 1988. Post graduated in M.Tach from UVCE, Bangalore, of Bangalore University in the year 1995. She has been actively guiding PG and UG students She is presently working as Assostant Professor in Electronics and Communication Engineering Department, BIT, Bangalore.