International Journal of VLSI System Design and Communication Systems

ISSN 2322-0929
Vol.04, Issue.10,
October-2016,
Pages:0994-0998

www.ijvdcs.org

Design and Verification of APB to SPI Interface

D. ANJALI[1], A. RADHIKA[2]

[1]PG Scholar, Dept of VLSI System Design, Anurag College of Engineering, Aushapur, Ghatkesar, TS, India.
[2]Associate Professor, Dept of ECE, Anurag College of Engineering, Aushapur, Ghatkesar, TS, India.

Abstract: For effective functioning of the system these modules need to be in sync with each other and share resources. Problem starts when one module follows different protocol as others and each module has its different bit rate or baud rate of data transfer which can be either asynchronous or synchronous. The paper takes an example of I2C protocol and AMBA APB protocol to describe the architecture which defines show data are transferred from one protocol to another. It exploits the flexible protocols of I2C to make it compatible with APB protocol. The proposed architecture is a bridge between I2CMaster and APB Salve. The data travels from a serial bus (I2C) to parallel bus (APB) to serial (I2C) in sync with the respective domain clock. This forms a bidirectional interface between I2Csupported module and APB supported module. It has the disadvantages of interface diagram complexity is more and requires more pins. to overcome this problem we can design this project. In this project we design an APB slave which acts as an SPI master and interacts with an off chip SPI slave through the SPI bus. Our design acts as in interface between the peripheral bus of the SOC (APB) and SPI which is one of the popular serial protocols for communication between ICs.

Keywords: AMBA APB, SPI Interface, SPI Controller.

I. INTRODUCTION

The Advanced Peripheral Bus (APB) is part of the Advanced Microcontroller Bus Architecture (AMBA) hierarchy of buses and is optimized for minimal power consumption and reduced interface complexity. The AMBA APB should be used to interface to any peripherals which are low band width and do not require the high performance of a pipelined bus interface. The latest revision of the APB ensures that all signal transitions are only related to the rising edge of the clock. This improvement means the APB peripherals can be integrated easily into any design flow, with the following advantages:

- Performance is improved at high-frequency operation
- Performance is independent of the mark-space ratio of the clock
- Static timing analysis is simplified by the use of a single clock edge
- No special considerations are required for automatic test insertion
- •Many Application-Specific Integrated Circuit (ASIC) libraries have a better Selection of rising edge registers
- Easy integration with cycle based simulators.

These changes to the APB also make it simpler to interface it to the new Advanced High-performance Bus(AHB). An AMBA-based microcontroller typically consists of a high-performance system Back bone bus, able to sustain the external memory bandwidth, on which the CPU and Other Direct Memory Access (DMA) devices reside, plus a bridge to a narrower APB Bus on which the lower bandwidth peripheral devices are located. Fig.1 shows the APB in a typical AMBA system.

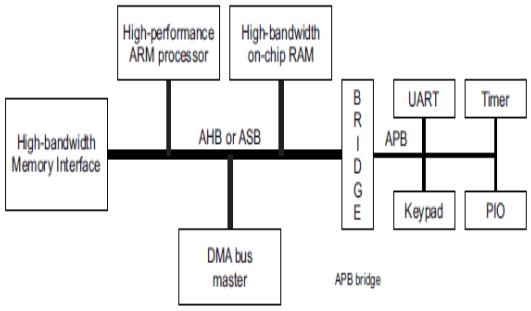

Fig.1 APB in a typical AMBA system.

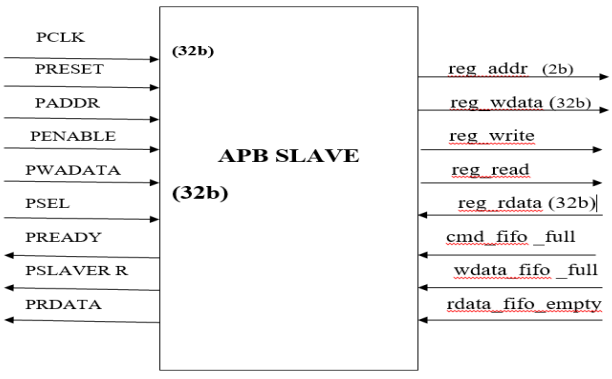

Fig. 2. APB slave block diagram.

The APB only supports single-word 32-bit accesses. Bits [1:0] of the PADDR signal are not used within the memory controller, resulting in byte and half-word accesses being treated as word accesses. The APB interface enables the memory

controller state of operation in addition to programming the memory controller with the correct timings and settings for the connected memory type. The APB interface also initializes the connected memory devices see Initialization .The APB interface is clocked by the same clock as the AXI domain clock, ack. However, the interface has also a clock enable, enabling it to be slowed down to execute at an integer divisor of ack. The PSLVERR output is included for completeness, and the DMC permanently drives it low. To enable a clean registered interface to the external infrastructure, the APB interface always adds a wait state for all reads and writes by driving PREADY low. In the following instances, a delay of more than one wait state can be generated:

- when a direct command is received and there are outstanding commands that prevent a new command being stored in the command FIFO
- When a memory command is received, and a previous memory command has not been completed.

Fig.2 shows the APB slave block diagram. it consists the PCLK, PRESET, PADDR, PENABLE, PSEL and REGRDATA as inputs and PEREADY, PSLAVERR and PRDATA as outputs.

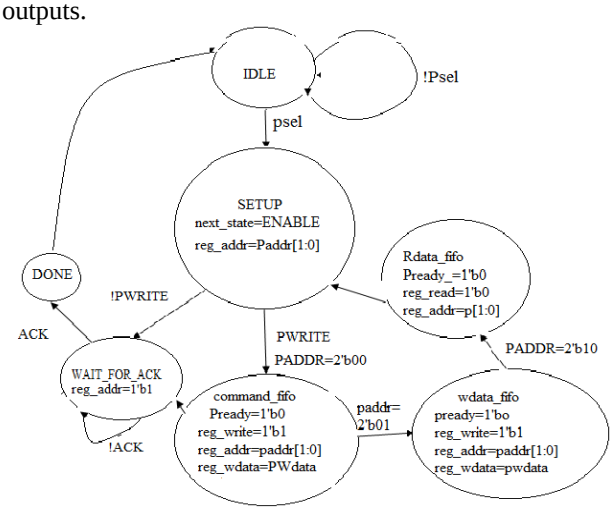

Fig. 3 state diagram for APB slave.

Fig .3 shows the state diagram for the APB slave.IDLE is the normal state of the APB. When a transfer is necessary the bus relocates into the SETUP state, where the suitable select signal, PSEL is asserted .The bus only waits in the SETUP state for one clock cycle and always moves to the ACCESS state on the next rising edge of the clock. ACCESS will enable signal, PENABLE, is asserted in the ACCESS state. The write, write data signals, select, and address must remain stable during the transition from the SETUP to ACCESS state. ACCESS state is controls when to exit by the PWRITE signal from the slave. These are the conditions one is if PWRITE is selected their three commands are performed, one is if paddr=2"b00 then it goes for command FIFO and paddr=2'b01then slave performed write operation. If paddr=2'b10 then slave performed read operation another is PWRITE is not selected o by the slave then the ACCESS state is exited and the bus returns to the IDLE state if no more transfers are required after that it will start the same cycle.

II. SPI PROTOCOL

The SPI is a synchronous serial interface in which data in an 8-bit byte can be shifted in and/or out one bit at a time. It can be used to communicate with a serial peripheral device or with another microcontroller with an SPI interface. The SPI system contains the four signals as shown in Fig. 4. In the master SPI, the bits are sent out of the MOSI pin and received in the MISO pin. The bits to be shifted out are stored in the SPI data register, and are sent out most significant bit (bit 7) first. When bit 7 of the master is shifted out through MOSI pin, a bit from bit 7 of the slave is being shifted into bit 0 of the master via the MISO pin. After 8 clock pulses or shifts, this bit will eventually end up in bit 7 of the master .The clock, which controls how fast the bits are shifted out and into SP0DR, is the signal SCLK. The SSpin must be low to select a slave. This signal can come from any pin on the master, including its SSpin when it is configured as an output.

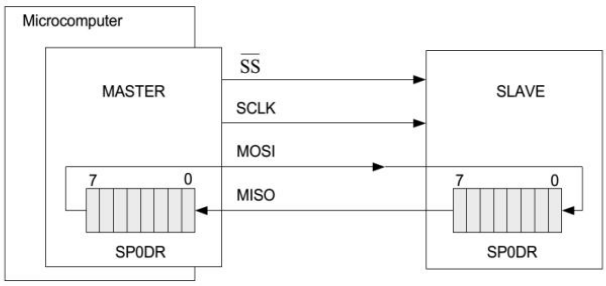

Fig.4 Two SPI modules Connected in a Master-Slave Configuration.

Features of SPI protocol

- Full duplex synchronous serial data transfer.
- Variable length of transfer word up to 128 bit.
- MSB or LSB first data transfer.
- Rx and Tx on both rising or falling edge of serial clock independently.
- 8 slave select lines.
- Fully static synchronous design with one clock domain.
- Technology independent.

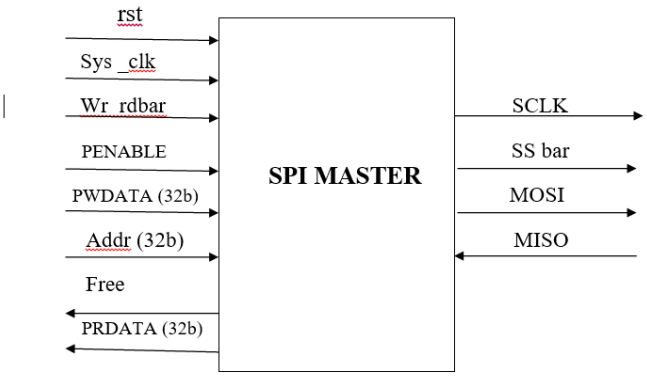

Fig. 5 block diagram of SPI MASTER.

The Serial Peripheral Interface (SPI) bus is a full-duplex serial link used for short-range communication between devices. It is a protocol developed by Motorola that has since become

facto standard for communication between master and slave devices in embedded systems, and is also used by sensors and SD cards. The SPI protocol consists of only four signals, shown in above SCLK (serial clock) is output from the master de-vice to the slave device and controls the rate of the data transfer. MOSI (master output slave input) and MISO (master input slave output) are the two data transfer signals, each of which transmits one bit per SCLK cycle. SS (slave select) is the signal from the master which enables the slave device for serial data transfer.fig 5 shows the block diagram for the SPI master IDLE is the normal state of the SPI. When a transfer is necessary the bus relocates into the SETUP state, where the suitable select signal, ENABLE is asserted .The bus only waits in the SETUP state for one clock cycle and always moves to the ACCESS state on the next rising edge of the clock. ACCESS will enable signal, TX_addr , is asserted in the ACCESS state. The write, write data signals, select, and address must remain stable during the transition from the SETUP to ACCESS state. ACCESS state is controls when to exit by the counter signal from the master. These are the conditions one is if counter is held LOW by the master then the peripheral bus remains in the ACCESS state another is counter is driven HIGH by the slave then the ACCESS state is exited and the bus returns to the IDLE state if no more transfers are required after that it will start the same cycle.fig 6.

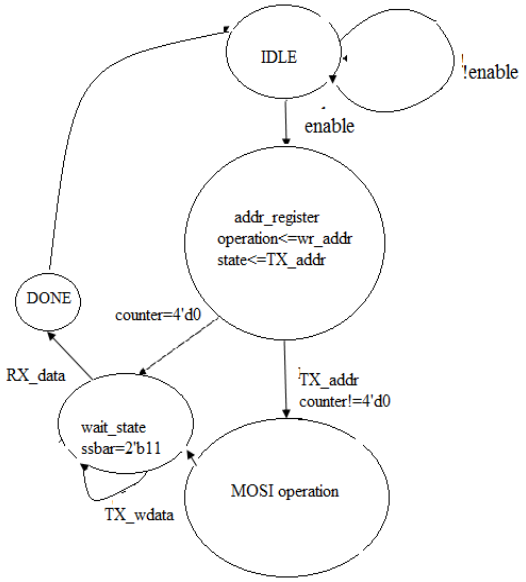

Fig. 6 state diagram for SPI master.

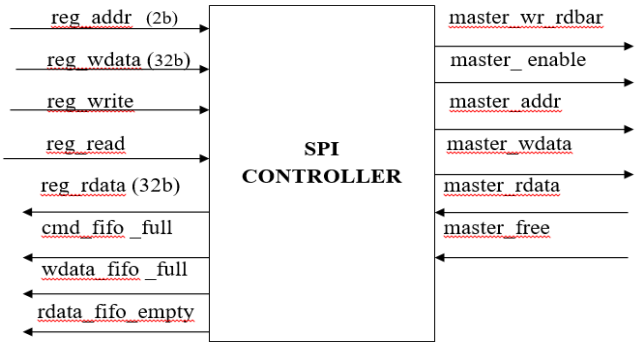

Fig. 7 block diagram for SPI controller.

This SPI controller provides an interface between an APB slaves with a SPI master. The outputs of APB slave are the inputs of SPI controller and the inputs of SPI master are the outputs of SPI controller.SPI controller performs following three operations i) When address is zero it goes to cmd_fifo.ii) when address is one it goes to wdata_fifo and write the data.iii) when address is two it goes to rdata_fifo and read the data.The SPI controller has two modes of operation, master or slave mode. This is selected with the address. In master mode the SPI controls the communication between the master and the slaves, while in slave mode the slave is enabled by the chip select pin is pulled low.fig 7 shows the block diagram for SPI controller. IDLE is the normal state of the SPI controller. When a transfer is necessary the bus relocates into the cmd_fifo empty where performs the operations .i)master_wdata<=wdata_fifo_rdata and wdata_fifo reads the 1bit.When mater is free it performs the operations i)master enable<=1'b1.ii)operation<=write operation.iii)master_wdata<=master_wdata. When master is not free it waits for master finish and then it goes for read done state. When read_done state is exited and the bus returns to the IDLE state if no more transfers are required after that it will start the same cycle as shown in Fig.8.

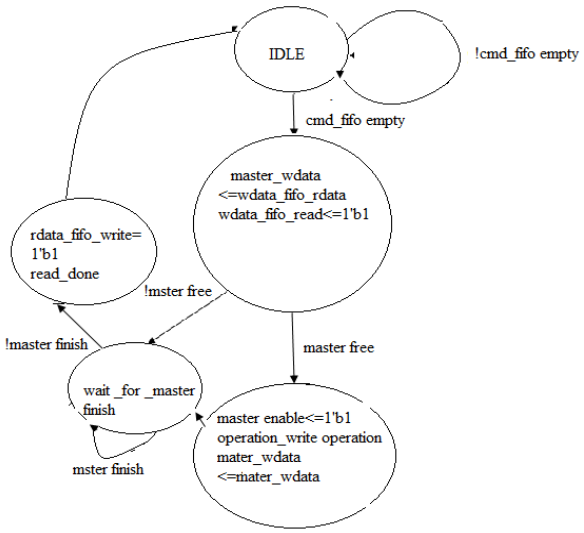

Fig. 8 state diagram for SPI controller.

III. METHOD OF IMPLEMENTATION

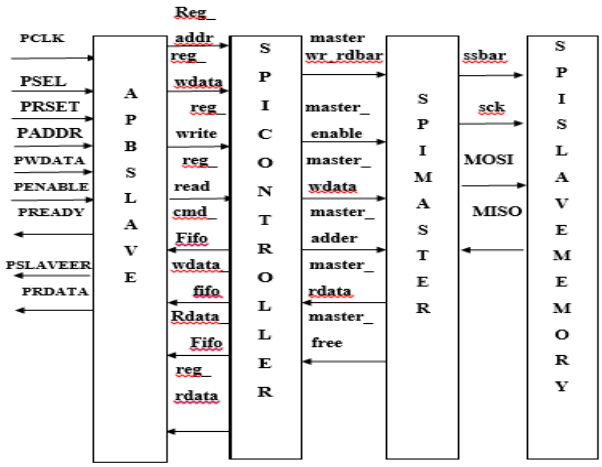

Fig. 9 APB to SPI interface diagram.

The outputs of APB slave are inputs of SPI controller and inputs of SPI master are outputs of SPI controller. The outputs of SPI controller are inputs of SPI master. The outputs of SPI master are inputs of SPI slave memory. The APB only supports single-word 32-bit accesses. Bits [1:0] of the PADDR signal are not used within the memory controller, resulting in byte and half-word accesses being treated as word accesses as shown in Fig.9. The APB interface enables the memory controller state of operation in addition to programming the memory controller with the correct timings and settings for the connected memory type. The APB interface also initializes the connected memory devices see Initialization. The Serial Peripheral Interface (SPI) bus is a full-duplex serial link used for short-range communication between devices. It is a protocol developed by Motorola that has since become facto standard for communication between master and slave devices in embedded systems, and is also used by sensors and SD cards.

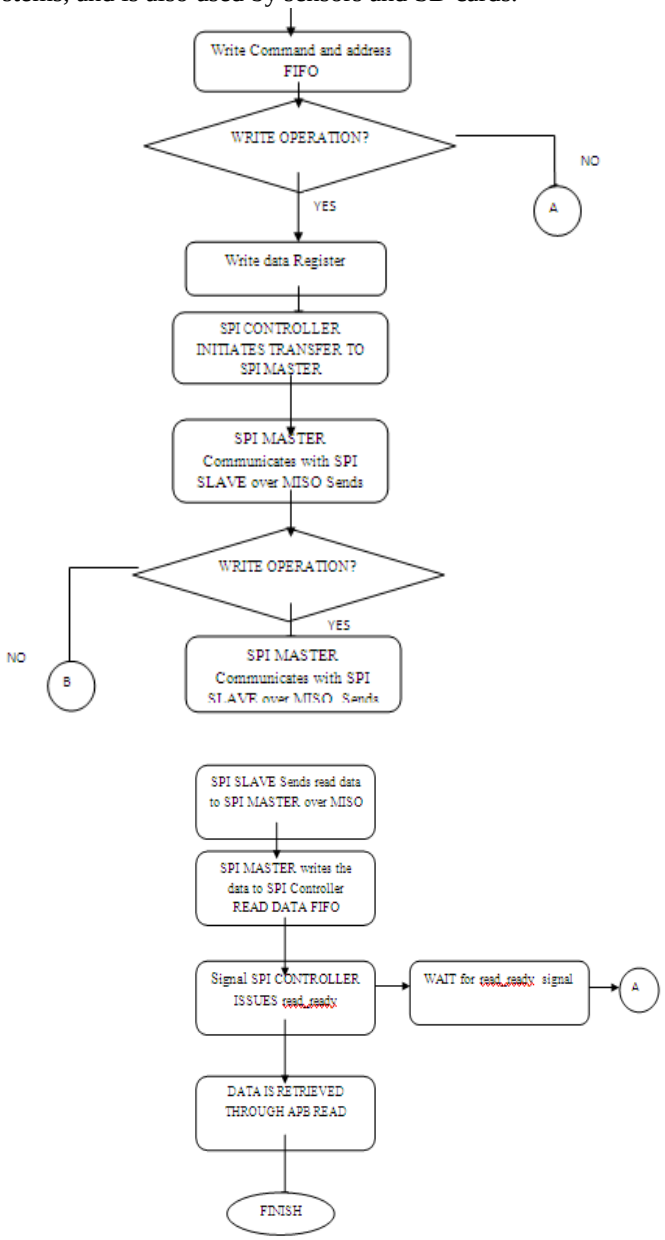

Fig. 10 implementation algorithm for the design of APB to SPI interface.

TheSPI protocol consists of only four signals, shown in above SCLK (serial clock) is output from the master de-vice to the slave device and controls the rate of the data transfer. MOSI (master output slave input) and MISO (master input slave output) are the two data transfer signals, each of which transmits one bit per SCLK cycle. SS (slave select) is the signal from the master which enables the slave device for serial data transfer. This SPI controller provides an interface between an APB slaves with a SPI master. The outputs of APB slave are the inputs of SPI controller and the inputs of SPI master are the outputs of SPI controller.SPI controller performs following three operations i) When address is zero it goes to cmd_fifo .ii) when address is one it goes to wdata_fifo and write the data.iii) when address is two it goes to rdata_fifo and read the data. The SPI controller has two modes of operation, master or slave mode. This is selected with the address. In master mode the SPI controls the communication between the master and the slaves, while in slave mode the slave is enabled by the chip select pin is pulled low.fig 10 shows the APB to SPI interface diagram.

IV. SYNTHESIS AND SIMULATION

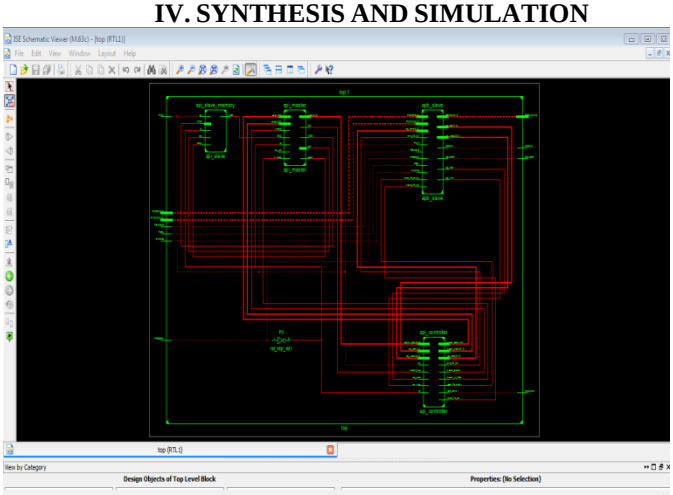

Fig. 11 RTL schematic for APB to SPI interface.

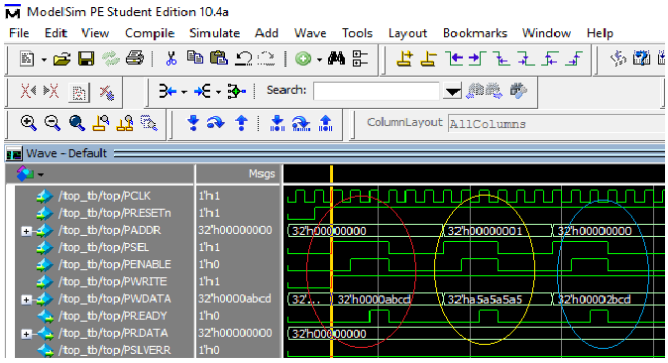

Fig. 12. APB bus driving APB slave.

In the above figs.11 and 12 we can see the APB BUS driving the APB slave. There are 3 write transactions.

PADDR=0, PWRITE=1, PWDATA=32'habcd.

PADDR=1 PWRITE =1, PWDATA = 32'ha5a5a5a5.

.PADDR=0 PWRITE =1, PWDATA = 32'h2bcd.

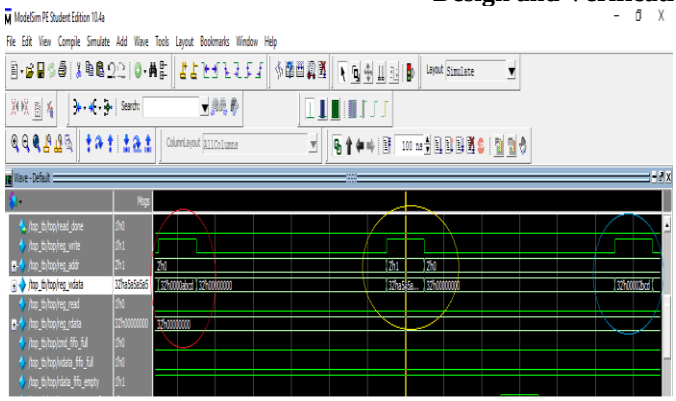

Fig. 13. APB slave writing into SPI controller registers.

The data received by the APB slave is written into the SPI Controller registers. SPI Controller has 2 register FIFOs, Control and address register FIFO with register address as 0 and Data register FIFO with register address 1. APB write with PADDR =0 in Fig.13 is translated into a SPI Controller register write with address 0 (i.e. control and address register). PWDATA received on the APB BUS will be written as is. After the 3 transactions in Fig.14 Control FIFO has 2 entries – 16'habcd and 16'h2bcd and Data FIFO has 1 entry 32'ha5a5a5a5.

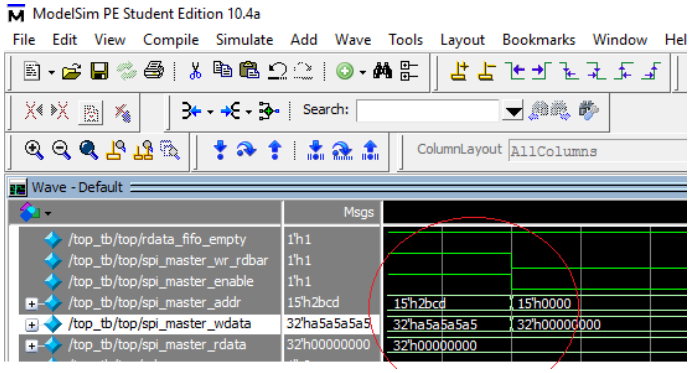

Fig. 14.SPI controller writing into SPI master.

The SPI Controller translates the information in its FIFOs to write and read transactions to be executed by the SPI master. The first entry in Control FIFO is 16'habcd which translates to 1010_1011_1100_1101 .The 16th bit indicates a write or read transaction. The 16th bit here is 1 which means it is a SPI write transaction as shown in Fig.15. The rest of the 15 bits indicate the SPI address which is 15'h2bcd here. The SPI Controller indicates the SPI master to do a SPI write transaction (spi_master_wr_rdbar=1)to address 15'h2bcd (spi_master_addr) and data 32'ha5a5a5a5 (spi_master_wdata) which the Controller fetches from its Data FIFO.

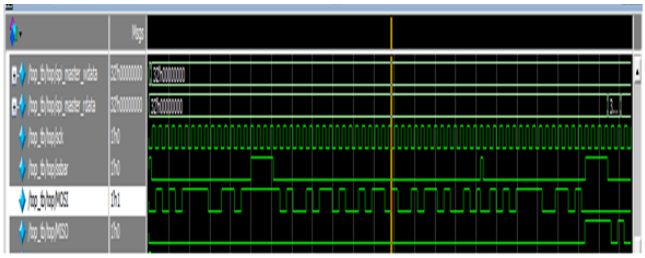

Fig. 15. SPI master to SPI slave.

V. CONCLUSION

To avoid high power requirements we can design this project. In this project we design an APB slave which acts as an SPI master and interacts with an off chip SPI slave through the SPI bus. Our design acts as in interface between the peripheral bus of the SOC (APB) and SPI which is one of the popular serial protocols for communication between ICs.

Future Scope: We have one SPI slave memory in our system. The system could be extended to have multiple SPI slaves with different functionalities like FUSES, camera, etc. Slave address decoding would be required to be added to the SPI Controller to distinguish accesses to different slaves and SPI master would need to generate corresponding SSELs to the corresponding slaves.

VI. BIBLIOGRAPHY

[1]hhttp://arm.com/about/trademarks/arm-trademark-list/AMBA-trademark.php
[2]http://www.corelis.com/products-bus-analyzers/
[3] Queued Serial Module Reference Manual
[4]ww.ti.com/lit/ug/sprufm4i/sprufm4i.pdf
[5]ww,latticesemi.com/~/media/.../SZ/SPISlaveController-Documentation.PDF
[6]ip.cadence.com/uploads/435/Cadence_32bit_APB_SPI_dspdf
[7]https://web.eecs.umich.edu/~prabal/teaching/eecs373-f10/readings/CoreSPI_HB.pdf
[8]https://web.eecs.umich.edu/~prabal/teaching/eecs373-f11/slides/lec5.ppt