Serial Peripheral Interface Design for Advanced Microcontroller Bus Architecture Based System-on- Chip

Mukthi. S. L[1] | Dr. A. R. Aswatha[2]

[1]Department of Electrical & Electronics Engineering, Jain University, Bangalore, Karnataka, India.
[2]Department of Electronics and Communication Engineering, DSCE, Bangalore, Karnataka, India.

ABSTRACT

Design of System-on-Chip(SoC) receives a great deal of attention in recent days. The number of peripherals used in one SoC becomes larger and larger. Processors and peripherals usually communicate with each other using some protocol. With the development of SoC technique, the communication between processor and peripherals become a problem as processors have limited ports hence we design Serial Peripheral Interface to maximize the usage of the existing ports. We are designing a SPI Flash Controller which is compatible with the Advanced Microcontroller Bus Architecture (AMBA) developed by ARM. We are developing a SPI Module which can operate based on the Advanced High Performance Bus (AHB) signals. Any SPI Flash Slave can be interfaced with the SPI Module directly from the ARM Processor. This paper is concerned with design and implementation of a SPI Flash Controller on FPGA which can communicate with SPI Slaves. Tool used for simulation is ModelSim 10.1b and finally implementation include FPGA kit Spartan3 xc3s400-pq208 comprising of 208 pins synthesis carried on Xilinx v9.1.Implantable and medical devices (IMDs) have been advanced with the advancements in engineering and medical science. IMDs are used for applying new therapies to patients, monitoring human body parameters and making diagnosis as per the monitoring result. Increased use of IMDs has enhanced the chances of attacks to them. Therefore, to make use of IMDs for various applications, they need to be secured. A system is developed to achieve the security. The system monitors various human body parameters wirelessly and detects anomaly if unauthorized node participates in communication. The system uses request response protocol in wireless communication. Experiments show that body parameters can be successfully monitored and signal characteristic can be used to detect anomaly.

KEYWORDS: *AMBA, DMC, AHB, SPI ,ARM.*

I. Introduction

An AMBA based microcontroller typically consists of a high performance system backbone bus, able to sustain the external memory bandwidth, on which the CPU on-chip memory and other direct memory access [DMA] devices reside. This bus provides a high bandwidth interface between the elements that are involved in the majority of transfers also located on the high performance bus is a bridge to the lower bandwidth APB, where most of the peripheral devices in the system are located .SPI Controller is one of the high performance system bus slaves. The AHB slave main function is an interface unit that allows AHB logic to initiate a data transfer on the AHB. The AHB specifies the type transaction to be executed on the slave through a user friendly interface. Serial Peripheral Interface (SPI) is one of the widely accepted communication interfaces. It was developed by Motorola. SPI protocol has become a standard but does not have officially released specification or agreed by any International committee. This gives some flexibility in Implementing SPI Protocol in the Electronic devices but also requires programmable flexibility of the host micro controller. The SPI Controller provides access to devices with SPI Interface. It can perform Read Operation, Write Operation or Read/Write Operation.

This paper is on designing a SPI Flash Controller which is compatible with the Advanced Microcontroller Bus Architecture (AMBA) developed by ARM[1]. The proposed SPI Module can operate based on the Advanced High

Performance Bus (AHB) signals. Any SPI Flash Slave can be interfaced with the SPI Module directly from the ARM Processor.

II. PROPOSED DESIGN

The Interfacing of the SPI controller to the ARM Processor using the AMBA AHB Bus is done[2]. The design of AMBA AHB SPI controller is carried out. Fig.1 shows the internal architecture.

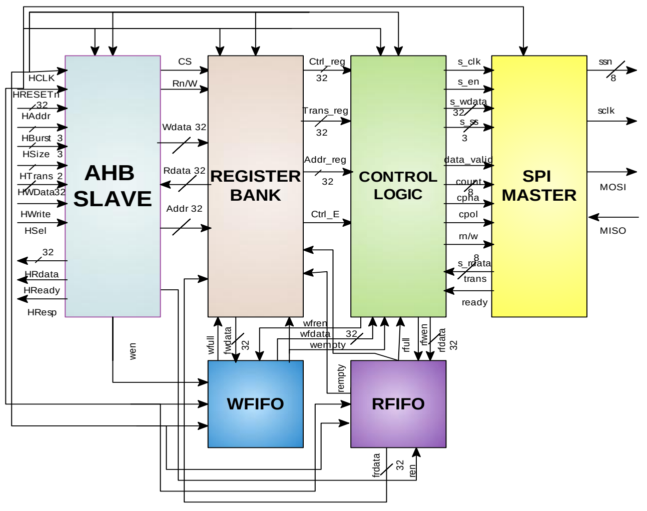

Figure 1: Internal architecture

The architecture is divided mainly into 4 blocks namely the

- AHB Slave Interface
- Register bank
- Control Logic
- SPI Master Interface.

A. AHB Slave Interface

An AHB bus slave responds to transfers initiated by bus masters within the system. The slave uses a HSEL select signal from the decoder to determine when it should respond to a bus transfer. All other signals required for the transfer, such as the address and control information, will be generated by the bus master.

Fig.2 shows the state diagram of AHB slave interface.

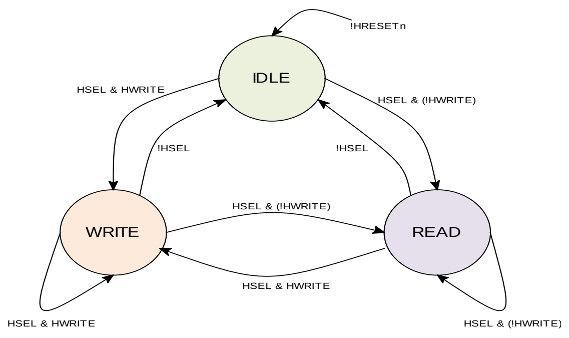

Figure 2: AHB State Diagram

B. Register Bank

Register bank is used to store the control information obtained from the CPU in an intermediate location so that the control logic can obtain the information and do its operation based on the information stored in the register bank. It also has a decoder which will decode the address of the FIFO and enable the respective FIFO. Also the Status register is updated by the Control Logic which is used by the CPU to know the current status of the Operation.

In our Register Bank there are 4 registers namely. This is shown in Table I.

Table I: Registers in the register bank

Address	Register Name	Default Values	Attributes
000	SPI Control Register	0x0000	Read/Write
001	SPI Transfer Register	0x0000	Read/Write
010	SPI Address Register	0x0000	Read/Write
011	SPI Status Register	0x0000	Read
100	Read FIFO	0x0000	Read
101	Write FIFO	0x0000	Write

C. Control Logic

In the architecture, a separate block called as control logic has been used. This block acts as the Finite State Machine (FSM) of the system, which will provide all the control information to the SPI Master based on the CPU's instructions. The Fig.3 shows the state diagram of control logic.

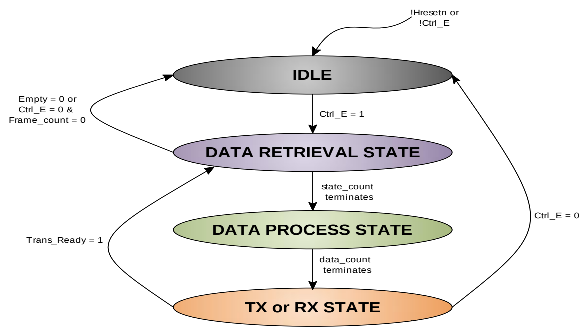

Figure 3: State Diagram for SPI Control Logic

D. SPI Master Interface

SPI slave is used to transform the data serially and communicate with the master when required. It forms the interface between control logic and SPI master. The data is transferred to SPI master in accordance with SPI protocol.

Serial Peripheral Interface (SPI) communication is done between the Master and Slave serially. We require 4 pins for SPI communication. These pins are

- SSn
- SCLK
- MOSI
- MISO

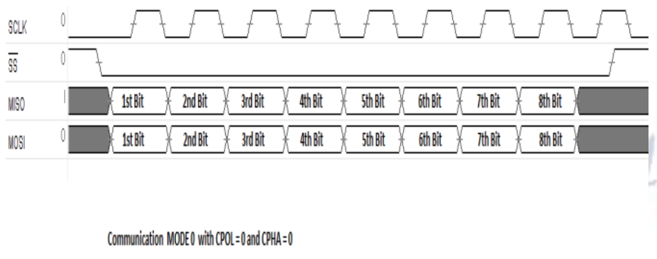

Figure 4: SPI Mode 0 Operation

Multiple slaves can be connected through multiple slave select pins. They are selected based on the content of the Slave Select Register. Before communication starts, SSn should be asserted to logic 0. The actual communication starts whenever SCLK is enabled. Here we have 4 modes of operation with respect to SCLK. These modes are controlled by 2 bits namely CPOL and CPHA. Depending upon the mode of operation, MOSI & MISO is asserted during a particular edge of the clock based on data to be sent or received. For example : In Mode 0 operation the stable data must be present before SCLK starts .Data is sampled at the positive edge and it is propagated in the negative edge. When the slave is not selected its MISO pin will be in high impedance state.

Write Operation (Considering Mode 0)

Before communication starts, SSn is made logic 0. The data should be stable before SCLK starts. Stable data will be present at the MOSI pin even before the SCLK starts. In this mode, Base value of clock is 0.i.e whenever SCLK starts its value will be 0. When SCLK is applied, the communication begins. The data is sent through the MOSI pin to the slave through Shift registers. Then the slave decodes the message and writes operation takes place. In Write Operation data on MISO is neglected by the Master. Whenever the write operation finishes, the SCLK is stopped and the SSn is simultaneously made logic 1.

Read Operation (Considering Mode 0)

Before communication starts SSn is made Logic 0. Since Mode 0 is used stable data should be present before SCLK arrives. The communication begins when the SCLK starts. The command & address is sent through the MOSI pin. The Slave analyses & sends the data through MISO pin. Now the Master starts receiving the data. When read operation completes, SSn is made logic 0 simultaneously with SCLK also stopped. The state diagram for SPI Master is shown in the Fig.5.

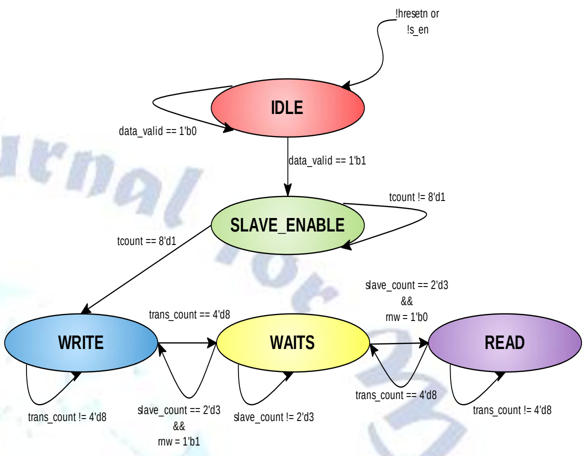

Figure 5: State Diagram for SPI Master

III. RESULTS

The SPI Master Module is simulated using ModelSim 10.1b and the Output Simulation results are observed. The SPI State Diagram is designed in such a way that if read operation has to take place then write operation must also take place as the opcode and address must be sent to the SPI slave for read operation as well as write operation. We are observing the simulation results for a Read operation where both Write and Read Operation are taking place. This is shown in Fig.6.

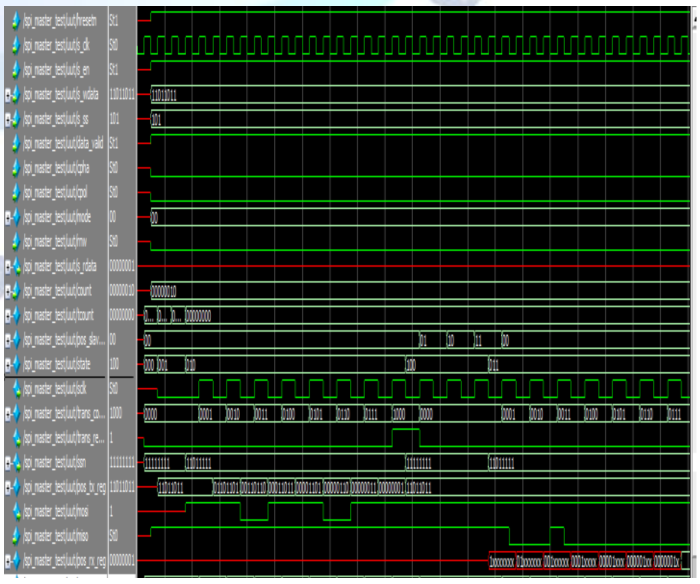

Figure 6: Simulation results of write and read operation

There are 4 modes of operation in SPI and all the Four Modes are simulated and the results are as shown in the Simulation figures. The Fig. 7 and Fig. 8 shows the waveform of MODE 0 and MODE 1 operation.

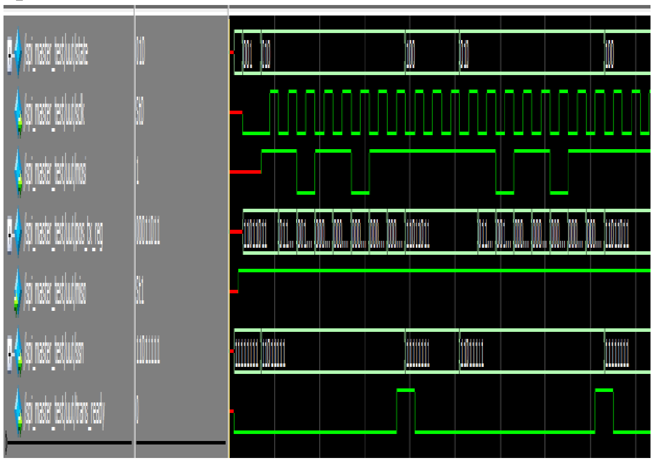

Figure 7: SPI MODE 0 Operation

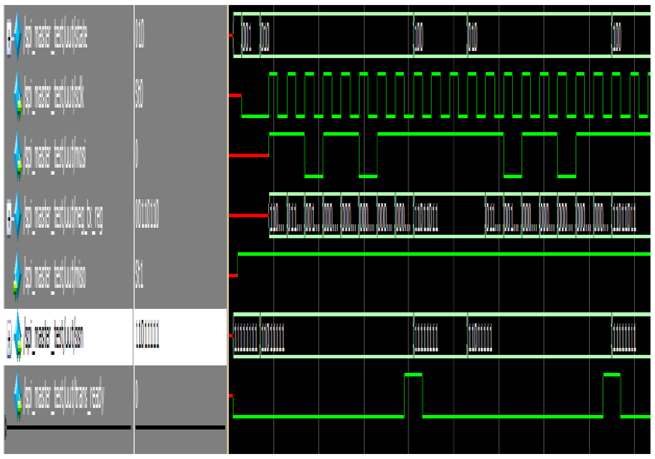

Figure 8: SPI MODE 1 Operation

The simulation waveform for MODE 2 and MODE 3 operation are shown in the Figure 9 and Figure 10.

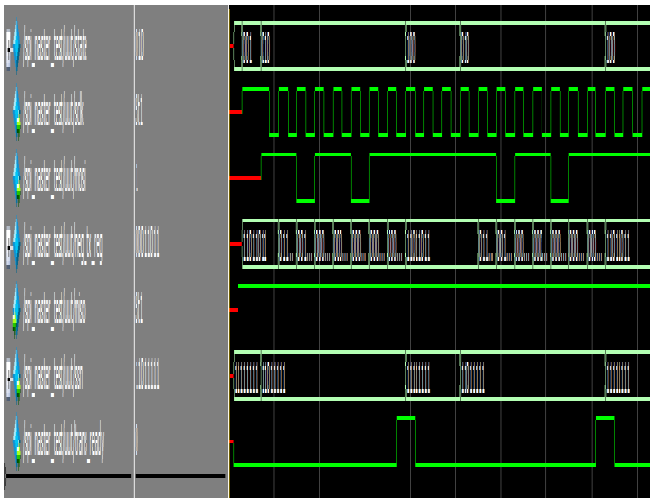

Figure 9: SPI MODE 2 Operation

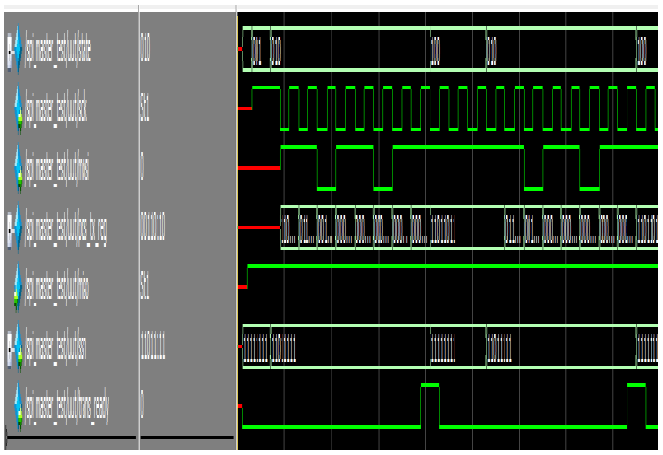

Figure 10: SPI MODE 3 Operation.

The synthesis report is shown in the Table II below.

Table II: RTL synthesis report

PARAMETERS	AHB INTERFACE	REGISTER BANK	CONTROL LOGIC	SPI MASTER
Number of Slices	44 out of 3584 1%	106 out of 3584 2%	120 out of 3584 3%	31 out of 3584 0%
Number of Slice Flip Flops	76 out of 7168 1%	128 out of 7168 1%	113 out of 7168 1%	13 out of 7168 0%
Number of 4 input LUTs	10 out of 7168 0%	136 out of 7168 1%	224 out of 7168 3%	56 out of 7168 0%
Number of bonded IOBs	144 out of 141 102%	232 out of 141 164%	152 out of 141 107%	37 out of 141 26%
Number of GCLKs	1 out of 8 12%	1 out of 8 12%	1 out of 8 12%	1 out of 8 12%

IV. CONCLUSION

A Serial Peripheral Interface has been designed. AHB Bus interface is used for communication with processor and the Serial Peripheral Interface. CPU can configure the Control Register, Transfer Register and the Address Register and inform the SPI Control Logic to do a particular operation and the Control Logic fills the Status register and indicates to the CPU the present status of the Operation. The SPI Master can communicate with its slave in all the different modes of operation as specified in the SPI protocol. We can perform either read or write operation at a time. The SPI can be reconfigured by software if needed. The Future Enhancement is Daisy Chain Supporting, which is often done with shift registers to provide a bank of inputs or outputs through SPI. Such a feature only requires a single SS line from the master, rather than a separate SS line for each slave. In the Future we plan to implement a SPI Master which can perform both the operation simultaneously (Full Duplex).

References

[1] ARM Corporation, AMBA specification 2.0, reference manual, 2006

[2] AMBA Open Specifications- ARM - http://www.arm.com

[3] SPI-www.en.wikipedia.org/wiki/Seria_Peripheral_Interface_Bus

[4] Digital Design Principles and Practises, John F. Wakerly, third edition, Prentice Hall Publication, 2000

[5] J.Basker, a Verilog HDL Primer 3rd Edition, 2003

[6] J. Noseworthy, "Efficient communications between an Embedded processor and reconfigurable Logic on FPGA" IEEE Transaction on Very Large Scale Integration(VLSI) Systems, Vol.16.No 8, pp 1083-1090.August 2008

[7] J.Basker, a Verilog HDL Synthesis 1st Edition, 1998.

[8] Nazeih M. Botros, "HDL PROGRAMMING (VHDL and VERILOG)", Dreamtech Press, 2009 edition.