On-chip AMBA Bus Based Efficient Bridge between High Performance and Low Peripheral Devices

Anurag Shrivastava, Sudhir Kumar Sharma
Department of ECE, Jaipur National University, India

Article Info

Article history:

Received Dec 21, 2016
Revised Feb 10, 2017
Accepted Feb 21, 2017

Keywords:

AMBA Bus Protocol
Asynchronous FIFO
AXI 2.0
VHDL
Xilinx Virtex 4

ABSTRACT

Today's scenario of SOC deals with integrity and sharing of information or data with various level of communication. AMBA bus protocol has been proposed by ARM community to justify the uneven demand of integrity .In this paper functional description and implementation of high peripheral devices supporting protocol AXI2.0 and its interface between low peripheral devices has been proposed. The connection named as bridge take care of the protocol mismatch and operates on data transfer for uneven speed demand. Asynchronous FIFO has been considered to avoid the complex handshaking mechanism. The design has been implemented within VHDL and implemented on Xilinx Virtex 4.

Corresponding Author:

Anurag Shrivastava,
Department of ECE,
Jaipur National University, India.
Email: shrivastavaanurag@rediffmail.com

1. INTRODUCTION

The AMBA™ on-chip interconnect system is an established open specification that details a strategy on the interconnection and management of functional blocks that makes up a System-on-Chip (SoC). It is a high-speed, high-bandwidth bus that supports multi-master bus management to maximize system performance.

On March 8, 2010, ARM announced availability of the AMBA 4.0 specifications. As the de facto standard SoC bus, AMBA bus is widely used in the high-performance SoC designs. The AMBA specification defines an on-chip communication standard for designing high-performance embedded microcontrollers. The AMBA 4.0 specification defines five buses/interfaces [1]:

- Advanced extensible Interface (AXI)
- Advanced High-performance Bus (AHB)
- Advanced System Bus (ASB)
- Advanced Peripheral Bus (APB)
- Advanced Trace Bus (ATB)

AXI, the next generation of AMBA interface defined in the AMBA 4.0 specification, is targeted at high performance, high clock frequency system designs and includes features which make it very suitable for high speed sub-micrometre interconnect. Typically the AXI2.0 protocol [2] has five channels for dedicated operation which can initiated the data transfer and control.

1.1. Character of AXI Bus

To correctly grasp the complexity of the design challenge facing an engineer of an AMBA 3 AXI protocol-based design, [3] we must first understand the technical features of the AMBA 4 AXI 2.0 protocol itself. [4] The key feature is that AXI separates the data channels into five independent data channels (write data channel, write address channel, write response channel, read data channel, read address channel). Each channel transfers data in only one direction, and there is no requirement for a fixed relationship [5] between the various channels. This is important because it enables the insertion of a register slice in any channel. It is also possible to use register slices at almost any point within a given interconnection. It is very advantageous to use a direct, fast connection between a processor and high-performance memory, but to use simple register slices to isolate a longer path to less performance critical peripherals.

Besides the above feature, AMBA 4 AXI 2.0 protocol supports many advanced accessing manners, such as exclusive access and unaligned transfer. The AXI protocol enables out-of-order transaction completion and the issuing of multiple outstanding addresses. These features enable the implementation of a high performance. Interconnection, [6], [7] maximizing data throughput and system efficiency. Furthermore, the protocol also defines low power interface to meet the need of reducing power in MPSOC system by issuing how to entry and exit low power state.

1.2. AXI Bus Architecture

At the point of data transaction, AXI protocol defines five independent channels as shown in Figure 1. Both write and read address channels have their own addresses to transfer, as well as the control information that describes the nature of the data to be transferred. AXI bus uses a write data channel to transfer data from master to slave and a read data channel to transfer data from slave to master. In write transaction, there is an additional write response channel to indicate the state of the transaction.

The AXI bus consists of five independent channels, [3] each of which contains a set of information signals and uses a two-way VALID and READY handshake mechanism. The VALID signal from source indicates that the data or control information is available, and the READY signal from destination indicates that it is ready to accept data. When both VALID and READY signals go to high, data is transferred. The AXI protocol is burst based transaction, so both read data channel and write data channel use LAST signal to show when the transfer of the final data item within a transaction takes place. At the point of master and slave devices, it can be divided into master, slave and the interconnection architecture which connects masters and slaves together. A typical system consists of a number of master and slave devices connected together via some form of interconnection. And we are free to choose the interconnection according to the need of system

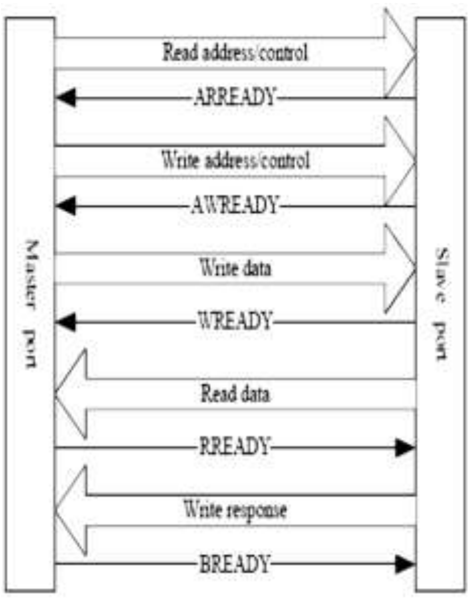

Figure 1. AXI bus architecture

1.3. APB

The AMBA APB is for low-power peripherals. AMBA APB [6] is optimized for minimal power consumption and reduced interface complexity to support peripheral functions. APB can be used in conjunction with either version of the system bus.

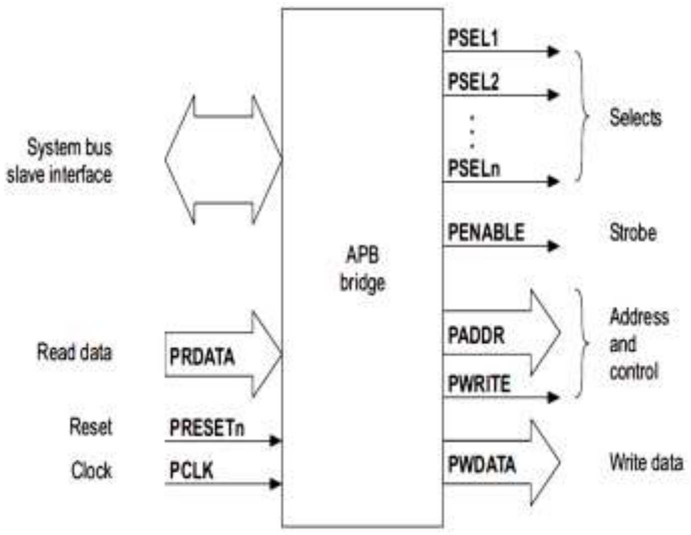

Figure 2. APB bridge

The Bridge unit converts system bus transfers into APB transfers and performs the following function [8]

a. Latches the address and holds it valid throughout the transfer.
b. Decodes the address and generates a peripheral select, PESLx. Only one select signal can be active during a transfer.
c. Drives the data onto the APB for a write transfer.
d. Drives the APB data onto the system bus for a read transfer.
e. Generates a training strobe, PENABLE, for the transfer.

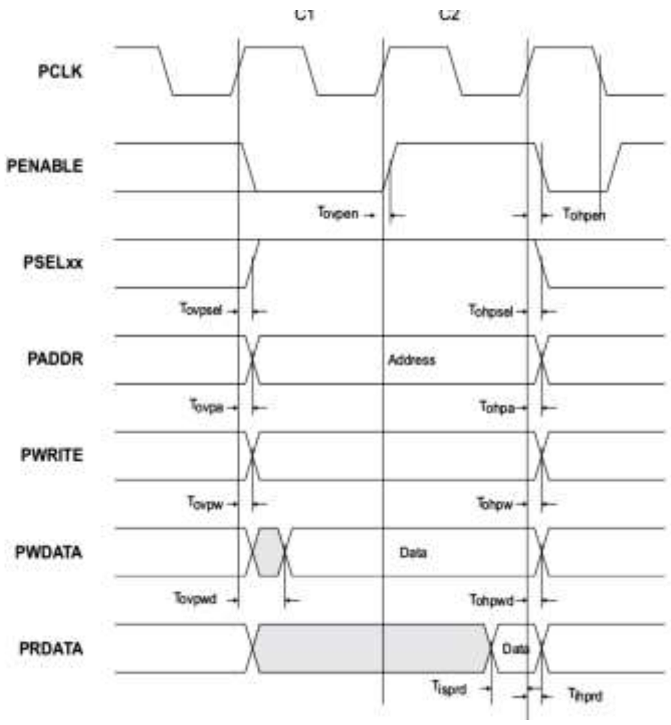

Figure 3. Design description

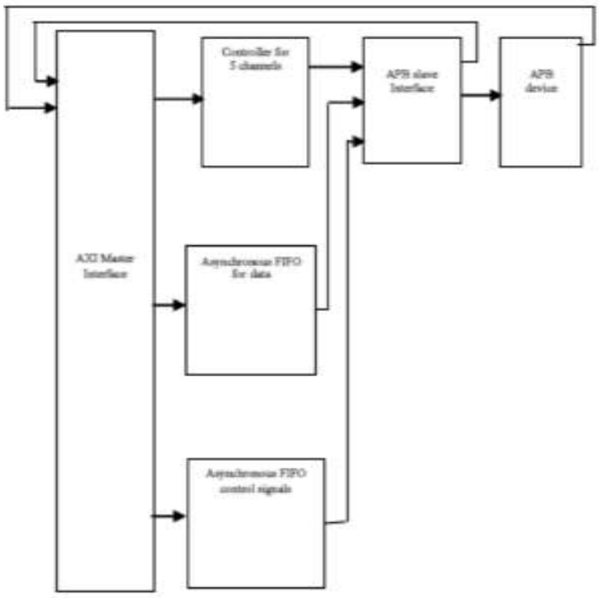
AXI Master Interface
Controller for 5 channels
APB slave Interface
APB device
Asynchronous FIFO for data
Asynchronous FIFO control signals

Figure 4. Basic APB Bridge operation

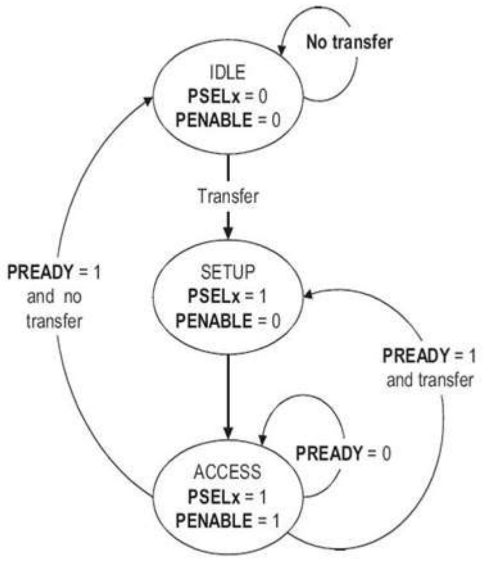

Figure 5. State diagram of APB

There are three basic operation in APB FSM.
Asynchronous FIFO:

This block is responsible for the communication interface between AXI master and APB slave. [8] once the control pin is stored in the FIFO according to CLK frequency of AXI master then FIFO will assert its FIFO full control according to the response of FIFO full, master stop its operation and then control is transferred to APB salve, and same way if read operation is done by APB slave then FIFO empty will be high and read operation will be stop. Asynchronous FIFO will take care of the high speed and low peripheral device interaction with different CLK write and read operation.

AXI Master Interface:- It is initiator of the operation between it take care of the operation by using its five channel and each channel have an asynchronous FIFO interface, so that protocol mismatch is resolved.

APB slave: It take care of the interface between master and device. If device is interacting to processor it has to interact with salve first, and slave will response according to analysis of control signal.it follows following state machine for its control.
ENABLE:-

- In ENABLE state the enable signal, PENABLE is asserted. The address, write and select signals all remain stable during the transition from the SETUP to ENABLE state.
- The ENABLE state also only lasts for a signal cycle and after this state the bus will return to the IDLE state if no further transfers are required. Alternatively, if another transfer is to follow then the bus will move to SETUP sate. [2]
- It is acceptable for the address, write and select signal to glitch during a transition from the ENABLE to SETUP states.

IDLE:-

- The default state for peripheral bus.

SETUP:-

- When the transfer is required the bus will move into SETUP state. Where the appro pate selected signal, PSELx, is asserted. The bus only remain in the SETUP state for one clock cycle and will always move to the ENABLE state on the next rising edge of the clock.

2. RESULTS AND SIMULATION

Figure 6 shows simulation results of APB slave. Figure 7 shows RTL view of APB slave. Figure 8 shows RTL diagram of APB slave. Figure 9 shows RTL view of AXI APB Bridge. Figure 10 shows simulation results of AXI APB bridge

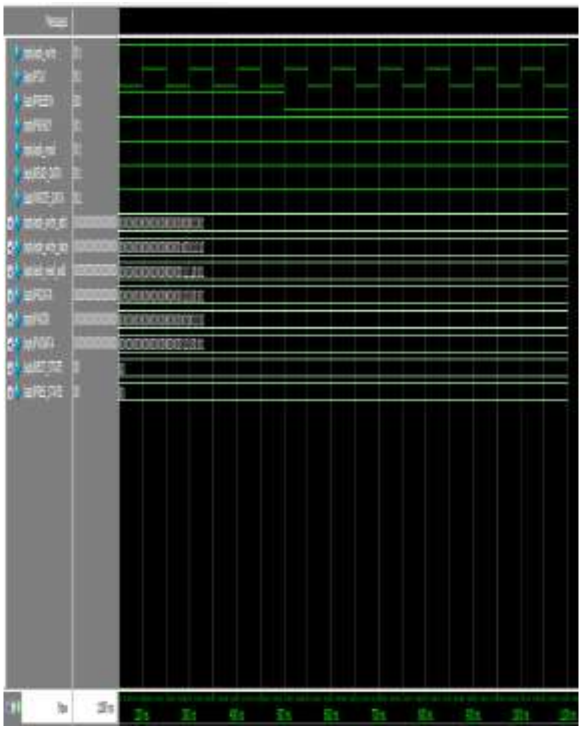

Figure 6. Simulation results of APB slave

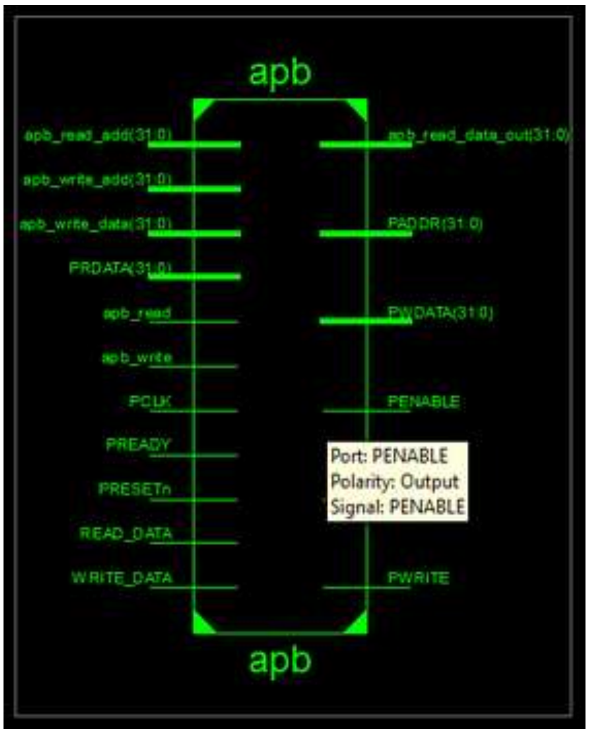

Figure 7. RTL view of APB slave

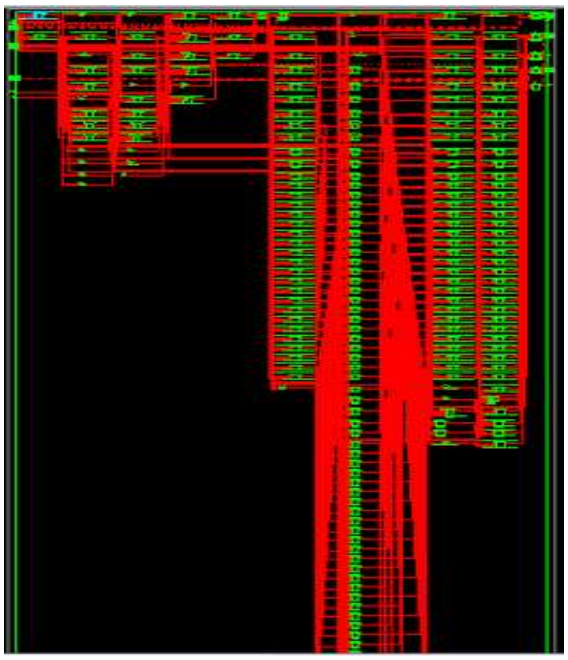

Figure 8. RTL diagram of APB slave

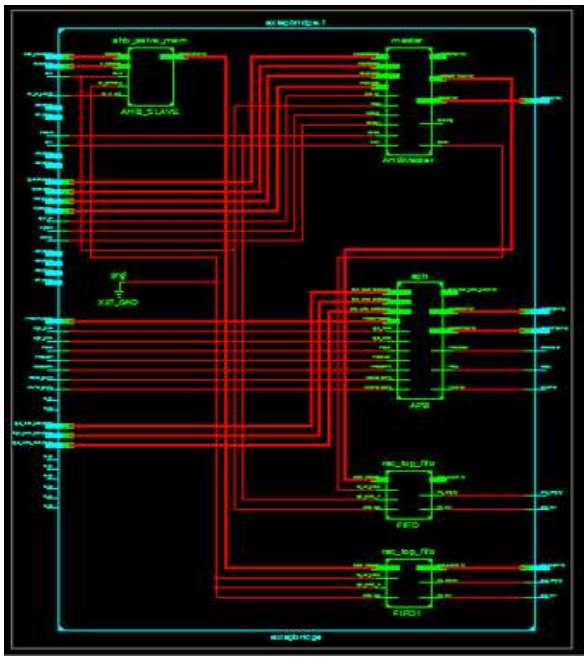

Figure 9. RTL view of AXI APB Bridge

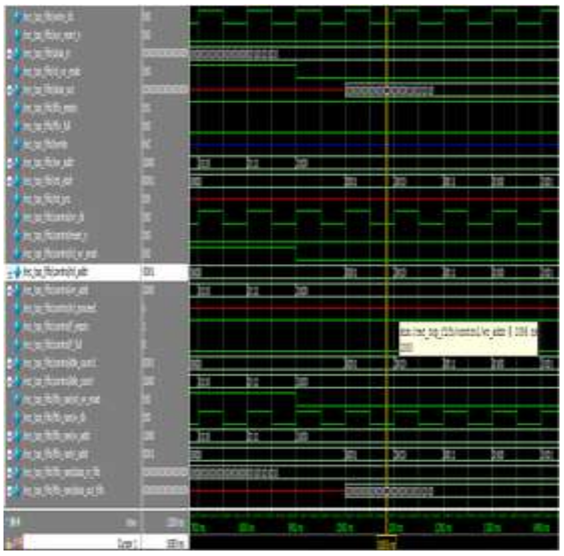

Figure 10. Simulation results of AXI APB Bridge

3. SYNTHESIS REPORT OF THE DESIGN

Final Results:-

Device utilization summary :

Selected Device : 6slx4tqg144-3

Slice Logic Utilization :

Number of Slice Registers :	58	out of	4800	1%
Number of Slice LUTs :	88	out of	2400	3%
Number used as Logic :	82	out of	2400	3%
Number used as Memory :	6	out of	1200	0%
Number used as RAM :	6			

Slice Logic Distribution :

Number of LUT FlipFlop pairs used : 92

Number with an unused FlipFlop : 34 outof 92 36%

Number with an unused LUT : 4 out of 92 4%

Number of fully used LUT-FF pairs : 54 out 92 58%

Number of unique control sets: 14

IO Utilization:

Number of IOs: 587

Number of bonded IOBs : 217 out of 102 212% (*)

IOB Flip Flops/Latches : 37

Specific Feature Utilization:

Number of BUFG/BUFGCTRLs : 4 out of 16 25%

4. CONCLUSION

The design has been implemented in VHDL as synthesized with Xilinx 6slx4tqg144-3 spartan 8. Area constraned is taken care for the implementation.for synchronization fifo is used,so its not a design which is latnecy free , but control has been develop so that it takes minimum latency.

REFERENCES

[1] ARM, "AMBA Specification Overview", available at www.arm.com

[2] ARM, "AMBA Specification (Rev 2.0)", available at www.arm.com

[3] ARM, "AMBA AXI Protocol Specification"2010, available at www.arm.com

[4] ARM, "AMBA Protocol specification 4.0,"www.arm.com, 2010.

[5] Anurag Shrivastava, G.S.Tomar and Singh, *"Performance comparison of AMAB Bus–Based system-on-chip communication protocol,"* in Proc.IEEE International Conference on CSNT 2011, 3-5 June. SMVDU Katra,pp-449-454.

[6] Akhilesh Kumar and Richa Sinha, *"Design and verification analysis of APB3 Protocol with coverage"* in Proc. IJAET 2011 Vol. 1, Issue 5, pp. 310-317.

[7] A Singh, Anurag Shrivastava, and G.S. Tomar *''Design and Implementation of High Performance AHB Reconfigurable Arbiter for on chip Bus Architecture"* in Proc.IEEE International Conference on CSNT 2011, 3-5. June SMVDU Katra.pp-455-459.

[8] Chenghai, Zhijun, *"Design and implementation of APB bridge based on AMBA 4.0"* Consumer Electronics, Communications and Networks (CECNet), 2011 IEEE International Conference 16-18 April 2011,pp-193-196.

[9] Math SS, Manjula, Manavi, *"Data transactions on system-on-chip bus using AXI4 protocol"*. Recent Advancements in Electrical, Electronics and Control Engineering (ICONRAEeCE), 2011 IEEE International Conference on 15-17 Dec, pp- 423-427.

[10] Sangik, Kang *"Implementation of an on-chip bus bridge between heterogeneous buses with different clock frequencies"*. System-on-Chip for Real-Time Applications, 2005. Proceedings. IEEE Fifth International Workshop on 20-24 July 2005, pp-530-534.

[11] Saxena, Bhatnagar, Jaiswal. *"Reconfigurable architecture for IP peripherals"*. Signals and Electronic Systems (ICSES), 2010 IEEE International Conference on 7-10 Set 2010, pp-347-350.

.