Modelling of Amba Axi Protocol

Guruprasad Hedge[1] Bharath B[2] Darshan G M[3] Gnanesh G[4]
[1,2,3,4]Student
[1,2,3,4] Electronics and Communication
[1,2,3,4] BMS College of Engineering

Abstract---Shrinking process technologies and increasing design sizes have led to highly complex billion-transistor integrated circuits (ICs). As a consequence, manufacturers are integrating increasing numbers of components on a chip. The need for high performance applications is driving the requirement for a new age of on chip communication infrastructure. On-chip bus organized communication architecture is among the top challenges in CMOS SoC technology due to rapidly increasing operation frequencies and growing chip size. This project is developed with the objective of enabling data transactions on SoC bus using AMBA AXI4 protocol modeled in Verilog hardware description language (HDL) and simulation results for read and write operation of data and address are obtained using XILINX ISE simulation tool. The operating frequency is set to 250MHz. Four test cases are run to perform multiple read and multiple write operations. The architecture developed here is not the ultimate solution. Here we had made an attempt to know about AMBA AXI protocol and developed an interconnect design for multi-masters and multi-slaves.

I. INTRODUCTION

In recent years due to the miniaturization of semiconductor process technology and computation for survival in the current market conditions constant customization is required. The semiconductor process technology is changing at a faster pace during 1971 semiconductor process technology was 10µm, during 2010 the technology is reduced to 32nm and future is promising for a process technology with 10nm. Intel, Toshiba and Samsung have reported that the process technology would be further reduced to 10nm in the future. So with decreasing process technology and increasing consumer design constraints SoC[has evolved, where all the functional units of a system are modeled on a single chip. SoC buses are used to interconnect an Intellectual Property (IP) core to the surrounding interface [1]. These are not real buses, but they reside in Field Programmable Gate Array (FPGA). The AMBA data bus width can be 32, 64, 128 or 256 byte, address bus width will be 32bits wide. The AMBA AXI4 specification to interconnect different modules in a SoC was released in March 2010

A. Amba Evolution:

Advanced Microcontroller Bus Architecture (AMBA), created by ARM as an interface for their microprocessors. The Advanced Microcontroller Bus Architecture (AMBA) specification defines an on chip Communications standard for designing high-performance embedded microcontrollers. Easy to obtain documentation and can be used without royalties

AMBA 1.0 released in 1996, includes ASB and APB AMBA 2.0 released in 1999, includes AHB AMBA 3.0 released in 2003, includes AXIAMBA 4.0 released in 2010, includes AXI4.

II. AXI4ARCHITECTURE:

The AMBA AXI protocol is targeted at high performance, high-frequency system designs and includes a number of features that make it suitable for a high-speed submicron interconnect. AMBA AXI4 [3] supports data transfers up to 256 beats and unaligned data transfers using byte strobes. The system consists of five channels namely write address channel, write data channel, read data channel, read address channel, and write response channel.

Separate address/control and data phases, Burst-based transactions with only start address issued separate read and write data channels to enable low-cost Direct Memory Access. Ability to issue multiple outstanding addresses Out-of-order transaction completion. Easy addition of register stages to provide timing closure. Every transaction has address and control information on the address channel that describes the nature of the data to be transferred. The data is transferred between master and slave using a write data channel to the slave or a read data channel to the master. In write transactions, in which all the data flows from the master to the slave, the AXI protocol has an additional write response channel to allow the slave to signal to the master the completion of the write transaction.

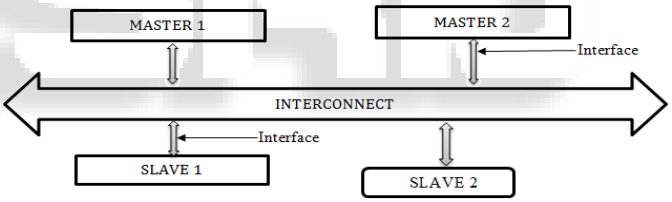

Fig. 1: AXI based System

A. Write Address And Data

The write operation process starts when the master sends an address and control information on the write address channel. The master then sends each item of write data over the write data channel. The master keeps the VALID signal low until the write data is available. The master sends the last data item, the WLAST signal goes HIGH. When the slave has accepted all the data items, it drives a write response signal BRESP[1:0] back to the master to indicate that the write transaction is complete. This signal indicates the status of the write transaction.

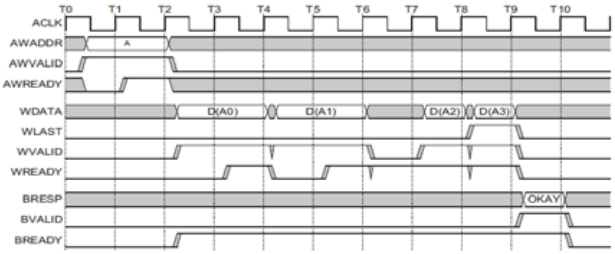

Fig. 2: Write Transaction Example

B. Read Address And Data:

After the read address appears on the address bus, the data transfer occurs on the read data channel .The slave keeps the VALID signal LOW until the read data is available. For the final data transfer of the burst, the slave asserts the RLAST signal to show that the last data item is being transferred. The RRESP [1:0] signal indicates the status of the read transfer.

The data bus that can be 8, 16, 32, 64, 128, 256, 512, or 1024 bits wide. A read response indicating the completion status of the read transaction.

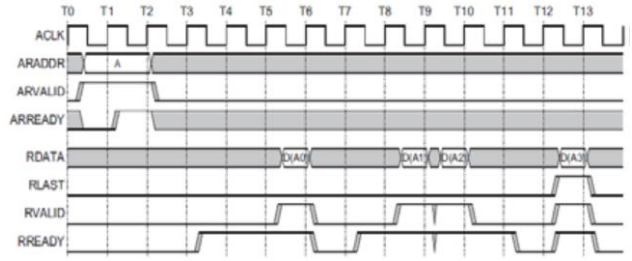

Fig. 3: Read Transaction Example

C. Channel Handshake

All five channels use the same VALID/READY handshake to transfer data and control Information. This two-way flow control mechanism enables both the master and slave to control the rate at which the data and control information moves. The source generates the VALID signal to indicate when the data or control information is available. The destination generates the READY signal to indicate that it accepts the data or control information. Transfer occurs only when both the VALID and READY signals are HIGH.

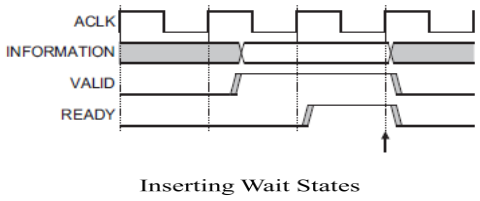

Fig. 4: Wait state example

The source presents the data or control information and drives the VALID signal HIGH. The data or control information from the source remains stable until the destination drives the READY signal HIGH, indicating that it accepts the data or control information. The arrow shows when the transfer occurs.

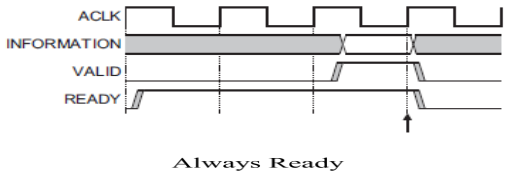

Fig. 5: Handshake in single cycle

The destination drives READY HIGH before the data or control information is valid. This indicates that the destination can accept the data or control information in a single cycle as soon as it becomes valid. The arrow shows when the transfer occurs.

Figure Shows both the source and destination happen to indicate in the same cycle that they can transfer the data or control information. In this case the transfer occurs immediately. The arrow shows when the transfer occurs

D. Burst Length:

The length of the burst must be 2, 4, 8, or 16 transfers. Every transaction must have the number of transfers specified by ARLEN or AWLEN. No component can terminate a burst early to reduce the number of data transfers. During a write burst, the master can disable further writing by de-asserting all the write strobes, but it must complete the remaining transfers in the burst. During a read burst, the master can discard further read data, but it must complete the remaining transfers in the burst.

ARLEN[3:0] AWLEN[3:0]	Number of Data transfers
b0000	1
b0001	2
b0010	3
.....	
b1101	14
b1110	15
b1111	16

Fig. 6: Burst length signal values

E. Burst Size:

AXI determines from the transfer address which byte lanes of the data bus to use for each transfer. For incrementing or wrapping bursts with transfer sizes narrower than the data bus, data transfers are on different byte lanes for each beat of the burst. The address of a fixed burst remains constant, and every transfer uses the same byte lanes. The size of any transfer must not exceed the data bus width of the components in the transaction.

ARSIZE[2:0] AWSIZE[2:0]	Bytes in transfer
b000	1
b001	2
b010	4
b011	8
b100	16
b101	32
b110	64
b111	128

Figure 7 Various Burst Sizes

III. COMMUNICATION BETWEEN MASTER AND SLAVE INTERFACE

During a write transfer, the user logic (AXI slave block) waits for AWVALID signal from the master and accepts the address and control information by asserting the associated AWREADY signal. The slave waits for the WVALID signal when the master drives valid write data. The slave acknowledges receipt of the write data by asserting the WREADY signal. The slave receives data until the WLAST signal is asserted by the master. The slave indicates receipt of the data by asserting the BVALID signal with a valid write response including write response ID. The BVALID signal must remain asserted until the Master accepts the write response and asserts the BREADY signal.

During a read transfer, the user logic waits for the ARVALID signal from the master and accepts the address and control information and asserts the associated ARREADY signal. The slave reads (or drives) the read data from the user interface and indicates a valid read data by driving the RVALID signal High. The slave waits for the RREADY signal before driving new data. The slave sends the RLAST signal when it drives the last data.

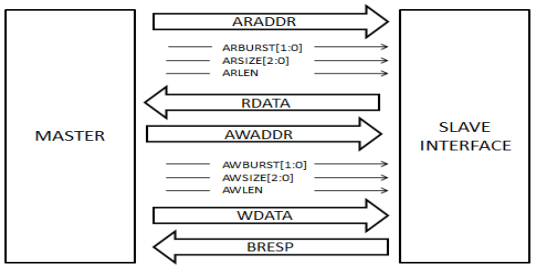

Fig. 8: Master to slave interface signals

IV. INTERCONNECT DESIGN

Interconnect is designed to connect two slave interface in the left side and two master interface in the right side. Interconnect is a highly configurable IP that can be optimized to suit the requirements of a complex SoC using the AMBA protocols. Massive growth in system integration places on-chip communication and interconnect at the center of system performance.

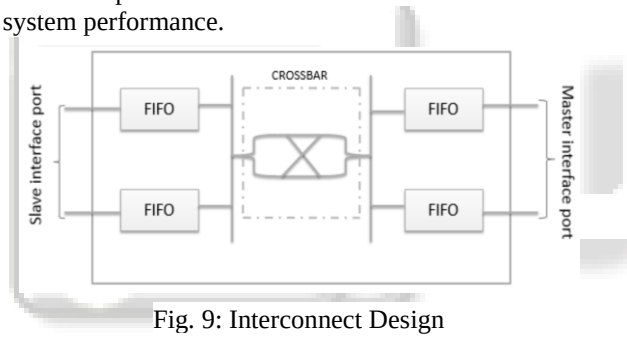

Fig. 9: Interconnect Design

A. FIFO(First In First Out):

In computer programming it is an approach to handling program work requests from queues or stacks so that the oldest request is handled first. In hardware it is either an array of flops or Read/Write memory that store data given from one clock domain and on request supplies with the same data to other clock domain following the first in first out logic. The clock domain that supplies data to FIFO is often referred as WRITE OR INPUT LOGIC and the clock domain that reads data from the FIFO is often referred as READ OR OUTPUT LOGIC. It is used to decide which signal to choose from slave interface.

B. Crossbar:

AXI Crossbar connects one or more similar AXI memory mapped masters to one or more similar memory-mapped slaves. When data sent by one master enters the crossbar, based on the decoder output, data is put in the right demux output channel. In our design i.e. two masters and two slaves, at a time both masters may try to communicate with one slave. Crossbar design consists of De-Multiplexer and Multiplexer with Priority Encoder

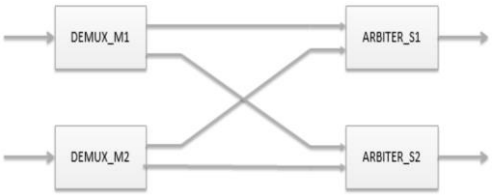

Fig. 10: Crossbar Module

C. De-Multiplexer:

It is used when a circuit wishes to send a signal to one of many devices. Demux output line will be chosen based on the value of the signal at Selection lines. Master address decoder logic will be given as input to selection lines to select memory mapped slave device and data will be routed to selected slave.

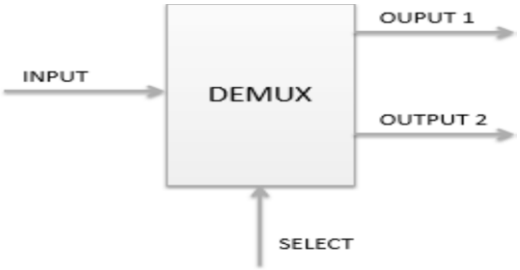

Fig. 11: Demux I/O Diagram

D. Multiplexer With Priority Encoder(Arbiter):

It is used when a circuit wishes to send signal from one of many devices. Mux input will be chosen as output based on the signal value at the selection lines. Priority Encoder is a circuit in which output corresponds to the input with highest designated priority. Priority encoder takes request from all the masters and grants permission to only one device with highest priority. Priority encoder output is connected to Mux to transfer data from access granted master.

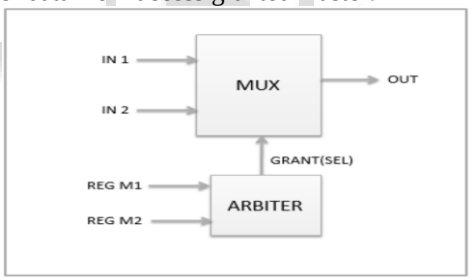

Figure 12 Arbiter Block

V. SIMULATION RESULTS

A. Write Burst Operation:

The ARESETn signal is active low. Master drives the address, and the slave accepts it one cycle later. The write address values passed to module is 0 and control signals related to burst operation are driven. The simulated result for burst operation is shown in figure 9.1. Input AWID [3:0] value is 0, which is same as the BID [3:0] signal, which is identification tag of the write response. The BID [3:0] value is matching with the AWID [3:0] value of the write transaction which indicates the slave is responding correctly. BRESP [1:0] signal that is write response signal from slave is 0 which indicates OKAY.

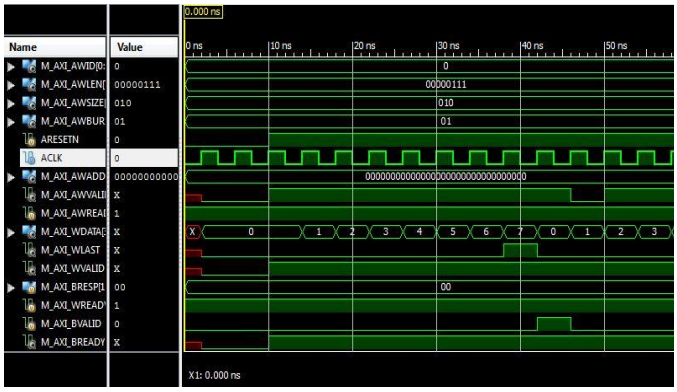

Fig. 13: Write Burst Operation

B. Read Burst Operation:

The read address values passed to module is 0 and the simulated result for burst read data operation is shown in fig 9.2. Simulation result of slave for read address operation input ARID [3:0] value is 0, which is same as the RID [3:0] signal which is the identification tag of the read response. The RID [3:0] and ARID [3:0] values are matching, which indicates slave has responded properly.

Fig. 14: Read Burst Operation

C. Write Outstanding:

Master 1 passed two consecutive addresses for burst write with transaction ids (AWID) as 0 and 1. Base addresses of two transactions are 0 and 8 respectively. Out of order transaction is completed in the order it is issued.

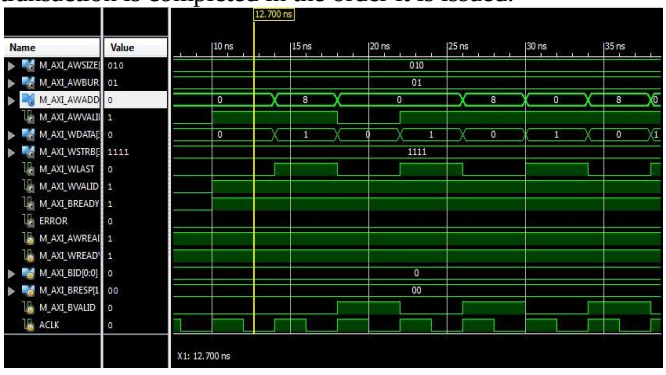

Fig. 15: Write Outstanding

VI. CONCLUSION

We implemented the AXI4 bus protocol which has the following features

- Supports 32 bit master and slave
- Supports burst transfer of length 16 for both read and write operation
- Supports maximum of 4 outstanding read/write transactions

REFERENCES

[1] Shaila S Math and Manjula R B "Design of AMBA AXI4 protocol for System-on-Chip communication", IJCNS, Vol-1, Issue-3 ISSN: 2231-1882.

[2] K. Lahiri, S. Dey and A. Raghunathan, "Design of communication architectures for high-performance and energy-efficient system-on-chip in Multiprocessor System on chips", A. A. Jerraya and W. Wolf Eds. Amsterdam: Elsevier, 2005, pp. 187-222.

[3] V.N.M Brahmanandam K, Choragudi Monohar, "Design of Burst Based Transactions in AMBA AXI Protocol for SoC Integration", International Journal of Scientific & Engineering Research Volume 3, Issue 7, July-2012 1 ISSN 2229-5518 IJSER.

[4] Hari, Venkanna and Anusha Ranga ,"Design and Implementation Of AMBA-AXI Protocol Using VERILOG For SoC Integration", International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com Vol. 2, Issue4, July-August 2012, pp.1102-1106.

[5] "AXI4 Overview-Benefits of Adopting AXI4-Protocol Overview", published by XILINX.

[6] "AMBA AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite ACE and ACE-Lite", published by ARM.

[7] "AMBA APB Protocol Specification", published by ARM.

[8] "LogiCORE IP AXI Interconnect v2.1". Product Guide for Vivado Design Suite, published by XILINX, PG059 December 18, 2013.

[9] "LogiCORE IP AXI to APB Bridge v2.0", Product Guide for Vivado Design Suite, published by XILINX, PG073 December 18, 2013.

[10] "AXI Reference Guide", published by XILINX, UG761 (v13.4) January 18, 2012.

