Implementation of Different Operations for Data Transfer by AMBA-Advanced Extensible Interface

Manish A Lakhiyar[#1], Dharmesh Khandhar[#2]

[#1] *Student of master of engineering in electronics & communication,c u shah college of engineering and technology,surendranagar,Gujarat,india,*

[#2] *Head of Department of electronics & communication,c u shah college of engineering and technology,surendranagar Gujarat,india*

[1]manish_lakhiyar@yahoo.com

[2]ddk_2008@yahoo.com

***Abstract*— The advanced development in the field of mobile, DSP motivated the design engineer to integrate the complex systems of multimillion transistors in a single chip. This paper investigates open core based SOC design platform. This paper basically described the different operation of data transfer between IP cores in AMBA Advanced extensible interface. This operations are such as simple read/write operation, multiple read/write operation, Open core uses a standard bus AMBA-advanced extensible interface Bus to alleviate System-on-Chip problem. The various issue related to AMBA and AMBA-advanced extensible interface is presented in this paper. These include AMBA specification types of buses/interfaces, and timing specification. All the designs are validated by Modelsim. In this work the design of an AMBA-advanced extensible interface basic block is presented. AMBA-advanced extensible interface is a high-performance bus in AMBA (Advanced Microcontroller Bus Architecture) family. This AXI can be used in high clock frequency system modules. The Advanced Extensible Interface acts as the high-performance system backbone bus.**

Keywords:* *AMBA AHB,AXI, IP cores ,SOC,VERILOG

I. INTRODUCTION

The Advanced Microprocessor Bus Architecture (AMBA) defined by ARM is now most widely used open standard for an on-chip bus system. The main aim of this paper to make ease the component design, by allowing the combination of interchangeable components in the MPSOC design. It promotes the reuse of IP components, so at least a part of the SOC design can become a composition, rather than a complete rewrite every time. By studying The AMBA specification 3.0, it defines an on chip communications standard for designing high-performance embedded microcontrollers. 3 distinct buses are defined by the AMBA specification.

- The Advanced Extensible Interface(AXI)
- The Advanced High-performance Bus (AHB)
- The Advanced System Bus (ASB)

A. *Advanced Extensible Interface(AXI)*

AXI is the 3rd generation protocol defined by AMBA. AMBA AXI protocol is based on the concept point-to-point connection. It is mainly targeted for high-performance, high-frequency system Designs and includes a number of features that make it suitable for a high-speed submicron interconnects.

B. *Advanced High-performance Bus (AHB)*

The AHB bus is used for high-performance, high clock frequency system modules. The AHB acts as the high-performance system backbone bus. AHB supports the efficient connection of processors, on-chip memories and off-chip external memory interfaces with low-power peripheral macro cell functions. AHB is also specified to ensure ease of use in an efficient design flow using synthesis and automated test techniques.

C. *Advanced System Bus (ASB)*

The AMBA ASB is for high-performance system modules. AMBA ASB is an alternative system bus suitable for use where the high -performance features of AHB are not required. ASB also supports the efficient connection of processors, on-chip memories and off-chip external memory interfaces with low-power peripheral macro cell functions.

II. AMBA –AXI

AXI is the 3rd generation protocol defined by AMBA. AMBA AXI protocol is based on the concept point-to-point connection. It is targeted at high-performance, high-frequency system Designs and includes a number of features that make it suitable for a high-speed submicron interconnects.

AXI protocol supports data transfers up to 256 beats and unaligned data transfers using byte strobes. In this system 16 masters and 16 slaves are interfaced. Each master and slave has their own 4 bit ID tags. AMBA AXI system consists of master, slave and bus (arbiters and decoders). The system consists of five channels namely write address channel, write data channel, read data channel, read address channel, and write response channel. The AXI protocol supports the following mechanisms.

- Unaligned data transfers and up-dated write response requirements.
- Variable-length bursts, from 1 to 16 data transfers per burst.
- A burst with a transfer size of 8, 16, 32, 64, 128, 256, 512 or 1024 bits wide is supported.
- Updated two new signal AWCACHE and ARCACHE signaling details.

In AXI each transaction is burst-based which has addressed 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.

A. ***AXI system design***

A typical AMBA AXI bus system design contains the following components:

AXI Master: AMBA AXI master which is able to initiate read and write operations by providing an address and control information with the help signals. For performing write address and data operation the transaction is initiated by input signals awaddr, awid, awcache, awlock, awprot, awburst and on the same lines for read address and data operations enable input by araddr, arid, arcache, arlock, arprot, arburst signals. The addresses of read and write operations are validated by VALID signals and sent to interface unit.

AXI slave: AXI bus Slave responds to a read or write operation within a given address-space range. The bus Slave signals back to the active Master the success, failure or waiting of the data transfer. The AXI slave consists of common read/ write buffer which stores the read/ write address and data. Pending read address register stores the remaining read addresses to be sent; pending write address register which stores the remaining write addresses to be sent and pending write data register which stores the remaining write data to be sent. The read/write state machines receive internal inputs from the read/ write buffer. The AXI slave test bench initiates the read or write transaction and the output from the AXI slave are standard read/write channel signals. The AXI slave receives the write data in the same order as address.

AMBA AXI bus Interconnect: The AXI bus interconnect block consists of arbiter and decoder. When more than one master initiates a transaction simultaneously, the arbiter block gives priority to access the bus. The decoder decodes the address sent by master and the control goes to one slave out of 4. The AMBA AXI interface decoder is centralized digital block. The decoder decodes the address sent by master and goes to one slave out of 4. 0-150 locations are meant for slave-1, next 151-300 addressable locations are meant for slave-2, and so on till slave 4,ideally AMBA AXI support up to 16 slaves .

***Arbitration*:** The arbitration mechanism is used to ensure that multiple Master has access to the bus at any time, so this mechanism enables parallel access path between multiple masters and slaves.

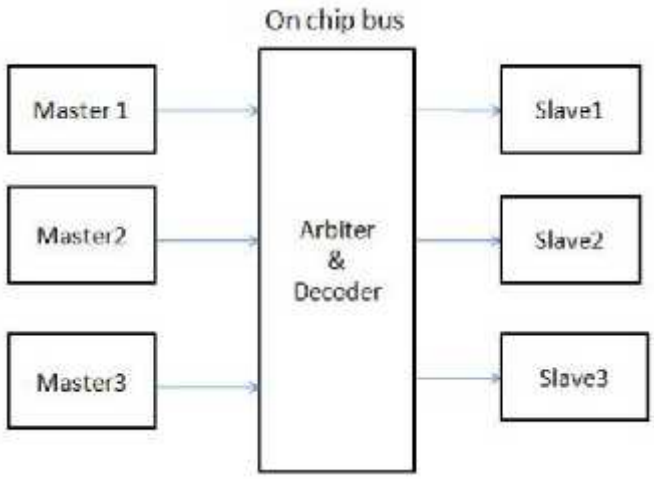

Figure 1. AMBA AXI bus interconnection

B. ***AXI Signals***

This section describes basics the AMBA AXI signals. All signals are prefixed with the letter A, ensuring that the AXI signals is differentiated from other similarly named signals in a system design such as AHB signals.

Aclk	Clock	Global clock signal. Here All the signals are sampled on the rising edge of the clock.
ARESETn	Reset source	Global reset signal. This signal is active LOW.
AWID[3:0]	Master	Write address ID. This signal is the identification tag for the write address group of signals.
AWADDR[31:0]	Master	Write address. The write address bus gives the address of the first transfer in a write burst transaction.
AWLEN[3:0]	Master	Burst length type. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address.
AWSIZE[2:0]	Master	Burst size. This signal indicates the size of each transfer in the burst. Byte lane strobes indicate exactly which byte lanes to update.
AWBURST[1:0]	Master	Burst type. The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated.
AWLOCK[1:0]	Master	Lock type. This signal

		provides additional information about the atomic characteristics of the transfer.
AWCACHE[3:0]	Master	Cache type..
AWPROT[2:0]	Master	Protection type.
AWVALID	Master	Write address valid. This signal indicates that valid write address and control information are available: 1 = address and control information available 0 = address and control information not available. The address and control information remain stable until the address acknowledge signal, AWREADY, goes HIGH.
WREADY	Slave	Write address ready. This signal indicates that the slave is ready to accept an address and associated control signals: 1 = slave ready, 0 = slave not ready
WID[3:0] Write	Master	ID tag. This signal is the ID tag of the write data transfer. The WID value must match the AWID value of the write transaction.
WDATA[31:0]	Master	Write data. The write data bus can be 8, 16, 32, 64, 128, 256, 512, or 1024 bits wide.
WSTRB[3:0]	Master	Write strobes. This signal indicates which byte lanes to update in memory. There is one write strobe for each eight bits of the write data bus.
WLAST	Master	Write last. This signal indicates the last transfer in a write burst.
WVALID	Master	Write valid. available: .

III.SIMULATION AND SYNTHESIS RESULT

A. AXI decoder and Arbiter

Based on starting address issue by masters which slave will going to interface with master determined by Decoder the output shown in figure 2 . Also I have used round robin algorithm for AXI which has rotating type priority for masters shown in figure 3.

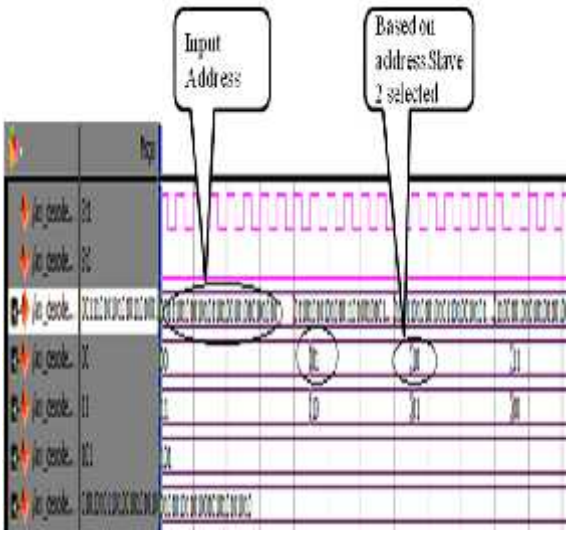

Figure 2 simulation result for AXI decoder

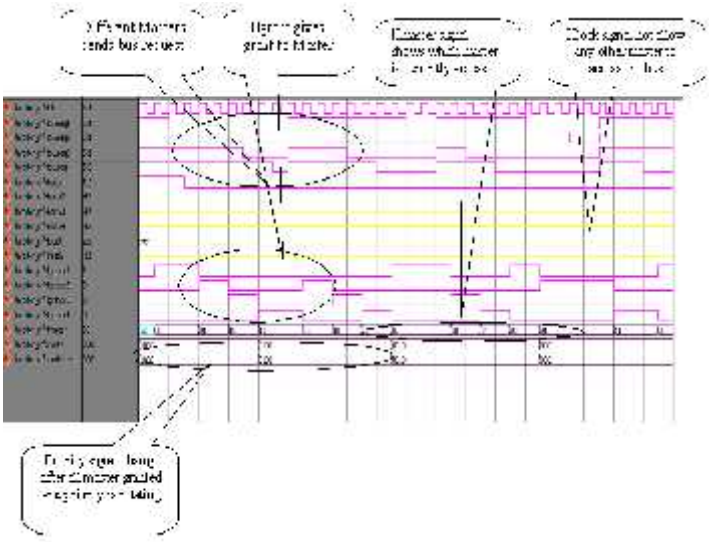

Figure 2 simulation result for AXI Arbiter

B. AXI slave for read and write address

For AXI design slave side arbitration multiple request need only one starting address for read and write operation which is the most important advantage of AXI over AHB. So here by giving just starting address for both read and write operation that can enable the multiple outstanding transactions. The simulation r result for slave for read address is shown in figure 4 and simulation result for write address shown in 5.

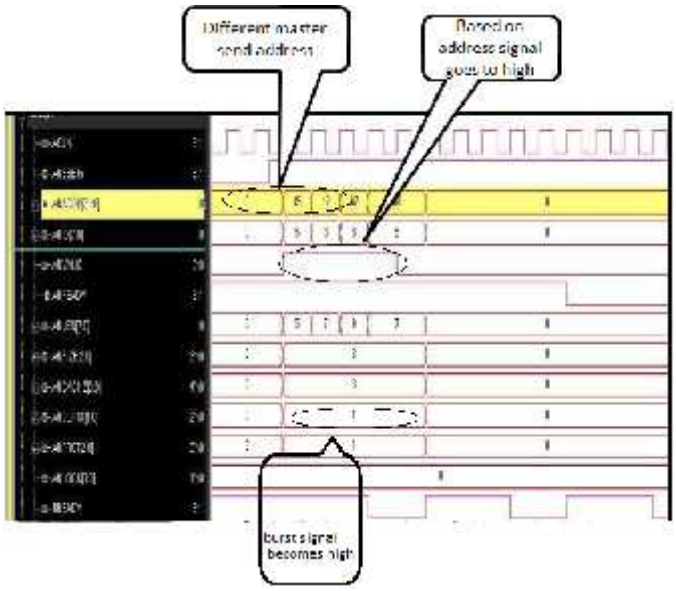

Figure 4. simulation results for read address

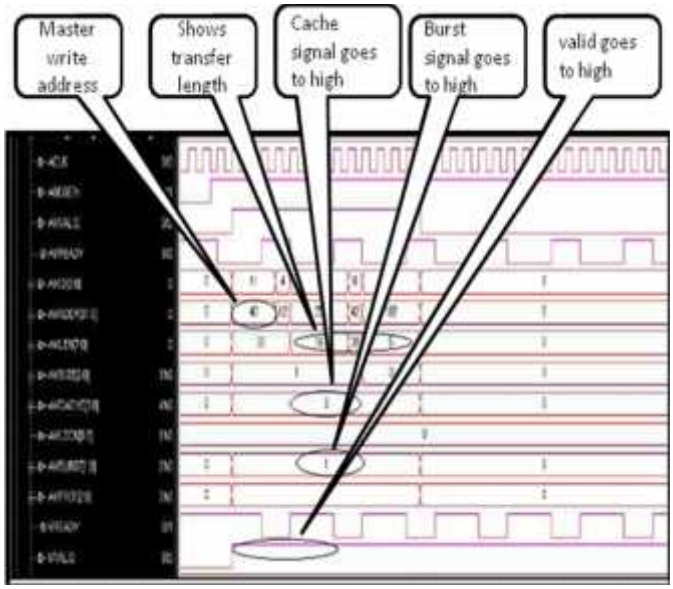

Figure 4. simulation results for write address

C. AXI slave for multiple read & Write

Here the normal type of the burst is passed to module. Internal_lock value is 0, internal_burst value is 1 and internal_prot value is 1,for both read and write operations, which indicate that the burst is of normal type. For write operation address locations passed to module are 40, 12, 35, 42 and 102; for read operations 45, 12, 67 and 98.The simulation output signals generated are From input side the validating signals AWVALID/ARVALID signals are generated by interconnect which gives the information about valid address and ID tags .For write operations RESP[1:0]response signal generated from slave indicates the status of the write transaction. The allowable responses are OKAY, EXOKAY, SLERR, and DECERR. For read operations RLAST signal is raised by slave for every transaction which indicates the completion of operation. The simulation result for multiple read data for slave shown in figure 6.

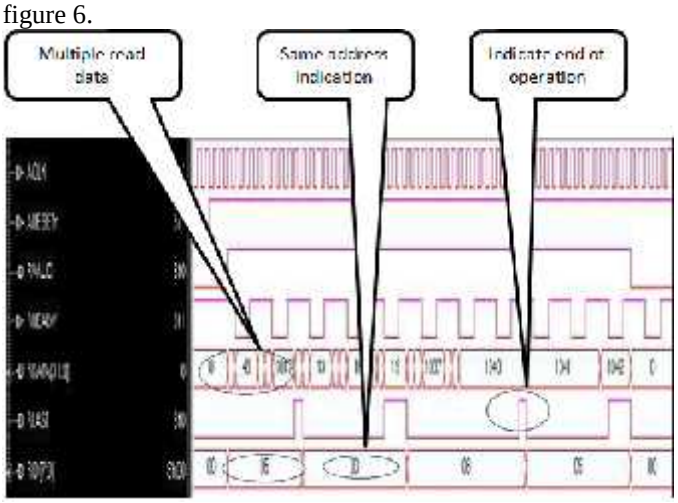

Figure 6.Simulation result for multiple read data for slave

D. *Simulation result for write operation*:

The **AResetn** signal is active low. All Master drives the address, and the based on address slaves accepts it one cycle later. The write address values for slaves passed to module are shown in figure. Input **AWID [3:0]** value is 11 for 40 address location, which is same as the BID [3:0] signal for 40 address location 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. Simulation result of slave for multiple write data operation is shown in figure 7.

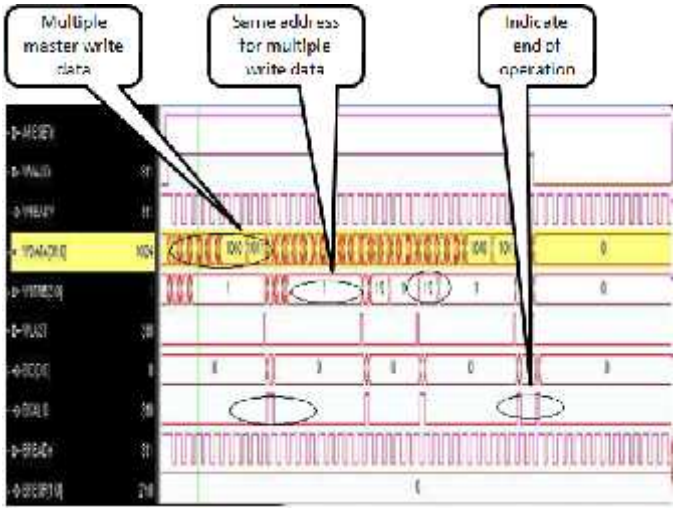

Figure 7. Simulation result for multiple write data operation for slave

IV SYNTHESIS RESULT

First I would like introduce about Synthesis. Synthesis is a nothing but the process of converting to high-level description of design into gate-level representation. Logic synthesis uses a standard cell library which have simple cells, such as basic logic gates like and, or, and nor, or macro cells, such as adder, multiplexers, memory block, and flip-flops/latches. Standard cells put together are called technology library.

A. *RTL schematic for AXI*

Above all result shows the functional simulation results. Now next step is synthesis of the AXI design. I have used Xilinx 12.1 for the synthesis and for the implementation of design. Below Figure shows the RTL schematic diagram for the AMBA AXI design. Also synthesis report is shown in Table 1.

Figure 5.15 RTL schematic for AMBA AXI

B. *Device Utilization Summery for AXI*

This section basically describes Number of flip-flops, Number of LUTs, Number of registers used during Synthesis. This design is synthesized using Xilinx 12.1 Software the value of c lock signal during operation is 110.120 Mhz. I have used Virtex-6 for the implementation purpose.

Slice Logic Utilization	Used	Available
Number of Slice Registers	128	93120
Number used as Flip Flops	87	
Number used as Latches	41	
Number of Slice LUTs	218	46560
Number used as logic	209	46560
Number used as Memory	0	16720
Number of occupied Slices	80	11640
Number of fully used LUT-FF pairs	99	1862
Number of bonded IOBs	69	240

V CONCLUSION

Depending upon the real time application these intellectual properties can be used and design number of Masters and Slaves that can be used in the work. The main goal of this work is to design algorithms for arbiter which is useful to granting the Master for multiple access to slave. The design of decoder is also completed which is generating the select signal for the slave. Also different operations of data transfer like simple read and write for AHB as well as multiple read and write operation using AXI.

REFRENCES

1). Anurag Shrivastava, G.S. Tomar and Ashutosh Kumar Singh, “Performance Comparison of AMBA Bus-Based System-On-Chip Communication Protocol” , 2011 IEEE, pp 449-454

2). Priyanka Gandhani, Charu Patel “Moving from AMBA AHB to AXI Bus in SoC Designs: A Comparative Study “Int. J Comp Sci. Emerging Tech Vol-2 No 4 August, 2011 pp 476-479

3). Andrei R adulescu, John Dielissen, Santiago González Pestana, Om Prakash Gangwal, Member, IEEE, Edwin Rijpkema, Paul Wielage, and Kees Goossens “An Efficient On-Chip NI Offering Guaranteed Services, Shared-Memory Abstraction, and Flexible Network Configuration” IEEE transactions on computer-aided design of integrated circuits and systems, VOL. 24, NO. 1, JANUARY 2005

4). Ms.Usha A. Jadhav and prof.M.M Jadhav” A HIGH THROUGHPUT AMBA AHB Protocol“International Journal of Engineering Science and Technology Vol. 2(5), 2010, 1233-1241

5). Rishabh Singh Kurmi and Miss.Shruti Bhargava, “Implementation of an AMBA Advanced High Performance Bus protocol IP block”, International Journal of Electronics Communication and Computer Engineering (IJECCE) vol.1 Issue.1 01/06/2011

6). Arm.amba axi protocol specification ARM.[online]. Available at http://www.arm.com