Design of AXI bus interface modules on FPGA

Ramesh Bhaktavatchalu
Dept of Electronics and Communication Engineering
Amrita School of Engineering, Amritapuri
Amrita Vishwa Vidyapeetham
Amrita University, India
Email: rameshamrita@gmail.com

B. Sasi Rekha, G. Ananta Divya, V. Usha Sai Jyothi
Dept of Electronics and Communication Engineering
Amrita School of Engineering, Amritapuri
Amrita Vishwa Vidyapeetham
Amrita University, India
Email: sasirekha195@gmail.com

***Abstract*-This paper describes the design and implementation of programmable AXI bus Interface modules in Verilog Hardware Description Language (HDL) and implementation in Xilinx Spartan 3E FPGA. All the interface modules are reconfigurable with the data size, burst type, number of transfers in a burst. Multiple masters can communicate with different slave memory locations concurrently. An arbiter controls the burst grant to different bus masters based on Round Robin algorithm. Separate decoder modules are implemented for write address channel, write data channel, write response channel, read address channel, read data channel. The design can support a maximum of 16 masters. All the RTL simulations are performed using Modelsim RTL Simulator. Each independent module is synthesized in XC3S250EPQ208-5 FPGA and the maximum speed is found to be 298.958 MHz. All the design modules can be integrated to create a soft IP for the AXI BUS system.**

Keywords—System-on-a-chip (SoC); ARM; AMBA; FPGA; AXI; HDL.

I. INTRODUCTION

Owing to the advancements in the semiconductor industry, the integration of large number of functional or IP blocks such as Processor cores, Graphic Processors, wireless communication models into a single chip has been possible. The SoC's provide good power efficiency and high speeds and thus are widely used in smart phones, laptops and a variety of embedded systems. For efficient and reliable data transfer between the blocks of a SoC, bus protocols complying with specific standards are used. ARM has introduced AMBA bus protocol standard in the year 1996. Later in 2003, it has introduced AXI as a part of its third generation. It is targeted at high performance and high clock frequency system designs. It is widely used with ARM Cortex A processors such as Cortex A9, Cortex A53 processors.

The AMBA AXI standard specifies the signal descriptions, channel handshake, addressing options and response signaling for data transfers. It also specifies the process and the properties of the transactions such as the number of data transfers in a burst, size of each transfer in bytes, burst type and protection type. The protocol makes use of a two way handshake mechanism using VALID and READY signals as shown in Fig. 1. The overall system can be viewed as five blocks, master interface, master controller, interconnect, slave controller and the slave interface as shown in Fig. 2. The master interface comprises multiple masters, all connected through the interconnect and the controller blocks to the multiple slaves of the slave interface. Successful System-On-Chip Communication between two masters and four slaves has been accomplished in this paper.

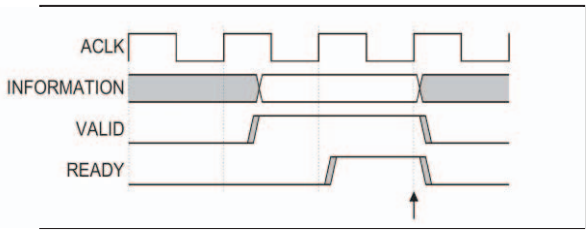

Fig.1. Handshake Mechanism [1] and [2].

The overall system can be viewed as three blocks, the master interface, the slave interface and the interconnect.

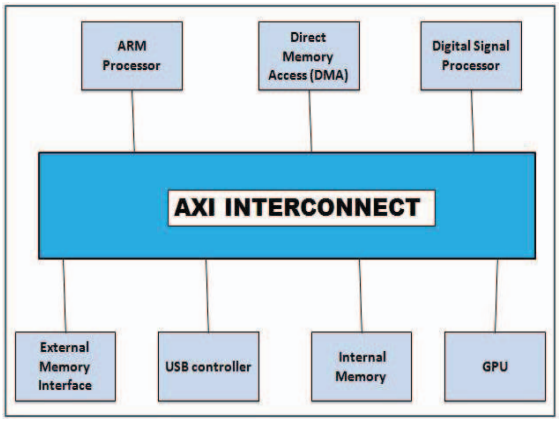

Fig.2. Example of AXI Interconnect on a System on Chip.

II. LITERATURE STUDY

IBM's CORE CONNECT [3], Silicore Corporation's WISHBONE are a few examples of the data buses in the semiconductor industry. AXI supports up to 16 transfers per burst for WRAP and FIXED burst types, but in INCR burst, more than 16 transfers can be performed in a single burst [4]. In order to verify the AXI Protocol, many protocol checkers have been proposed. Modern hardware testbenches and system-level SoC/NoC hardware designs have several subcomponents interconnected with an AMBA bus [5].

Authorized licensed use limited to: Technische Universitat Kaiserslautern. Downloaded on March 14,2024 at 13:54:21 UTC from IEEE Xplore. Restrictions apply.

The AMBA AXI protocol checker system includes configuration register, protocol checker and error reference [6]. The AXI4 protocol checker continuously checks the signals from AXI4 Master and AXI4 Slave and flags messages such as errors, warnings or information based on protocol checks implemented inside it [7]. AXI has applications over a wide range of domains. The multi level Monte Carlo FPGA Architecture which uses AXI interconnect can simulate up to 100 millions of time steps in an asset path simulation using a power supply of less than 3.6W which indicates its high energy efficiency [8]. A Wizard Link Transceiver (TI-TLK2711A) which can be used in flights combined with STAR-Dundee SpaceFibre CODEC IP core and an AXI4-based DMA controller constitute the AXI SpaceFibre IP core [9]. The DDR3 memory controller permits access to the DDR3 memory through an AXI bus interface [10]. The performance of AXI based MPSoC evaluated with the matrix operation benchmark could run at 100MHz and required 8963 Adaptive Look Up Tables and the maximum speed ratio achieved was 3.81 indicating its effectiveness over the AHB bus [11]. Enough communication bandwidth can be achieved using a crossbar router for the network architecture [13]. System Verilog can also be used to model the AXI system components and to perform system verification. The verification system consists of generator, mailbox, BFM and AXI interface modeled in system verilog [14]. The AXI can also be connected to other bus models like APB where AXI slave acts as APB Bridge connecting the AXI4 and the APB4 buses [14] and [15].

III. Proposed Axi Interface Architecture

A. KEY FEATURES

The key features of the AXI protocol are,

- Separate address/control and data phases.
- Unaligned data transfers using 4 bit byte strobe.
- Separate read and write data channels resulting in parallelism.
- Ability to issue multiple outstanding addresses.
- Out-of-order transaction completion.
- Easy addition of register stages in the interconnect to provide timing closure.
- It distinguishes between data and instruction.
- Signals for low power operation support.

B. SYSTEM ARCHITECTURE

The proposed architecture has the following blocks,

- Master.
- Slave.
- Interconnect.

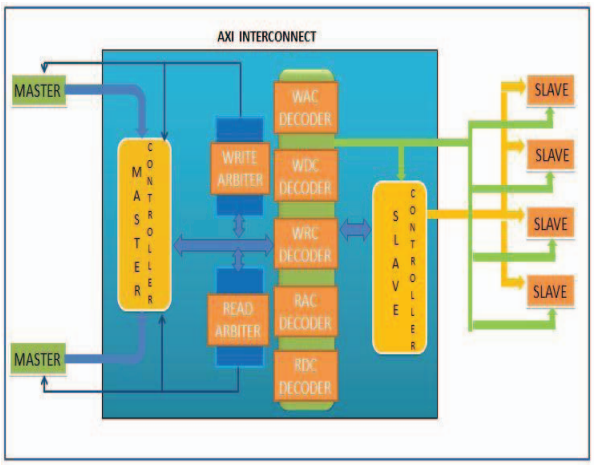

Fig.3. Proposed AXI System Architecture.

1. The protocol specifies the use of five independent Channels,
   a. Write Address Channel.
   b. Write Data Channel.
   c. Write Response Channel.
   d. Read Address Channel.
   e. Read Data Channel.

2. The interconnect consists of the following blocks,
   a. Master Controller.
   b. Arbiter.
   c. Decoder.
   d. Slave Controller as shown in Fig. 3.

3. The interconnect comprises two arbiters, one for the write group of channels and the other for the read group of channels. Each arbiter grants access to one master at a time which has requested for an access of the write group of channels or read group of channels. Thus at an instance, two different masters can perform the data transfer, i.e. one performing the read operation and the other performing the write operation.

4. The decoder performs the function of obtaining the slave address from the given address i.e. AWADDR [31:0], ARADDR [31:0] which corresponds to the address of the memory location for the Write or Read operation.4. Each arbiter gives access to one master at a time which has requested for an access. Thus at an instance, only two different masters can perform the data transfer, i.e. one performing the read operation and the other performing the write operation.

5. The Master Controller multiplexes output bus signals from the granted master to the Slave Controller and other blocks. Similarly, it demultiplexes the signals from the Slave Controller onto the input bus signals of the granted master through the arbiter grant control signal.

Authorized licensed use limited to: Technische Universitat Kaiserslautern. Downloaded on March 14,2024 at 13:54:21 UTC from IEEE Xplore. Restrictions apply.

6. Similarly, the Slave Controller multiplexes output bus signals from the selected slave through the decoder control signals to the Master Controller and other blocks and demultiplexes the signals from the Master Controller onto the input bus signals of the selected slave through the decoder control signals.

7. The proposed architecture includes 5 decoders, one per each channel. This facilitates multiple outstanding transactions.

Table.1. Bus widths of the AXI channels.

Channel Widths	
Channel	***Width(Bits)***
Write Address Channel	56
Write Data Channel	43
Write Response Channel	8
Read Address Channel	56
Read Data Channel	41

C. WRITE OPERATION

The write operation gets initiated when the master gets grant from the arbiter. AWADDR [31:0] signal indicates the address of the destination from which the slave address location is decoded by the decoder. The slave accepts the asserted information after the successful address handshake i.e. when both the AWVALID and AWREADY are asserted high. Then, for every data transfer the master places the data on the write data channel. The number of data transfers, the size of the data transfer and the type of the burst are given by the control information signals AWLEN [3:0], AWSIZE[2:0], AWBURST[1:0]. The termination of the data transfer is indicated by the WLAST signal generated by the master. After the slave accepts all the data, it asserts response signal BRESP [1:0] indicating the transaction status. The allowable response signals are OKAY, EXOKAY, SLVERR, and DECERR. The response signal OKAY indicates the success of a normal access or the failure of an exclusive access. EXOKAY signal indicates the success of either read portion or write operation of an exclusive access. SLVERR occurs, when the access has reached a slave but the slave wishes to return an error condition to the originating master. DECERR signal generated by the interconnect indicates the absence of the slave at the decoded address.

D. READ OPERATION

The read operation gets initiated when the master gets grant from the arbiter. ARADDR [31:0] indicates the address of the destination from which the slave address location is decoded by the decoder. The slave accepts the asserted information after the successful address handshake i.e. when both the ARVALID and ARREADY are asserted high.

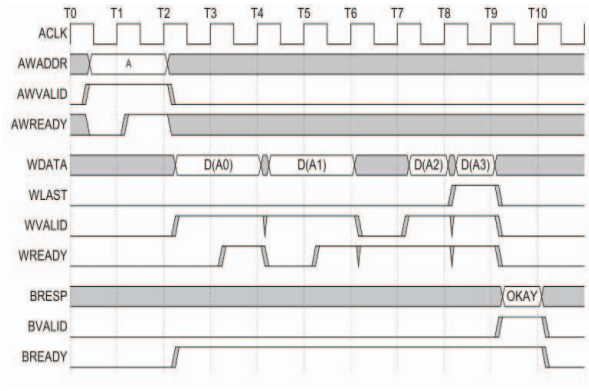

Fig.4. Write Burst Operation [1] and [2].

Then, for every data transfer the slave places the data on the read data channel. The number of data transfers, the size of the data transfer and the type of the burst are given by the control information signals ARLEN [3:0], ARSIZE [2:0], ARBURST [1:0]. The termination of the data transfer is indicated by the RLAST signal generated by the slave. The response signal RRESP [1:0] is given by the slave after every data transfer. The allowable response signals are OKAY, EXOKAY, SLVERR, and DECERR.

Table.2. Control Information Encoding [1] and [2].

CONTROL INFORMATION SIGNALS	
ARLEN[3:0]	1,2,3,4,5,6,7,8,9,10,….,16 transfers
AWLEN[3:0]	
ARSIZE[2:0]	1,2,4,8,16,32,64,128 bytes
AWSIZE[2:0]	
AWBURST[1:0]	FIXED,INCR,WRAP, Reserved
ARBURST[1:0]	

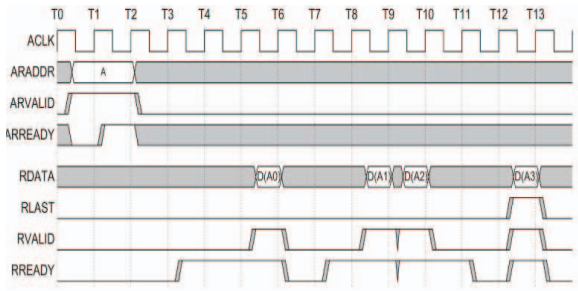

Fig.5. Read Burst Operation [1] and [2].

IV. IMPLEMENTATION DETAILS

This section details with the simulation results performed in Modelsim SE and Xilinx ISE. A clock frequency of 100MHz is chosen for the simulation of write and read transfers. The verilog modules have been simulated using the Spartan 3E FPGA device XC3S250EPQ208-5.

Authorized licensed use limited to: Technische Universitat Kaiserslautern. Downloaded on March 14,2024 at 13:54:21 UTC from IEEE Xplore. Restrictions apply.

A. SINGLE WRITE OPERATION

AWLEN [3:0] is set to 4'b0000, indicating that the number of transfers in the burst is 1.

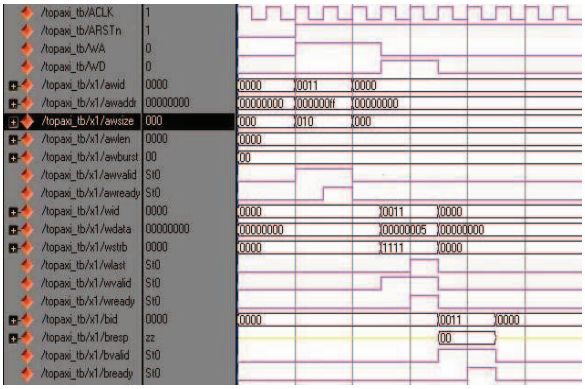

Fig.6. Waveform of single write operation.

The time taken for the single write operation is 70ns excluding the 20ns, the first two clock cycles of ARSTn i.e. the Reset Phase. AWSIZE [2:0] is set as 3'b010, indicating that the size of each transfer is 4 bytes, thus the WSTRB is set as 4'b1111 indicating that all the 4 byte strobes are valid. Thus the throughput that can be achieved while performing a single write transfer is 4.571 bits/clock cycle. A match exists between AWID, WID and BID indicates that the correct slave is responding to the transfer operation. The slave accepts the master asserted information 1 clock cycle after the master asserts it. The termination of the operation occurs at 90ns as indicated in the figure after the response is accepted by the master. The BRESP = 2'b00 (OKAY) indicates the success of the write operation.

B. SINGLE READ OPERATION

ARLEN [3:0] is set to 4'b0000, indicating that the number of transfers in the burst is 1.

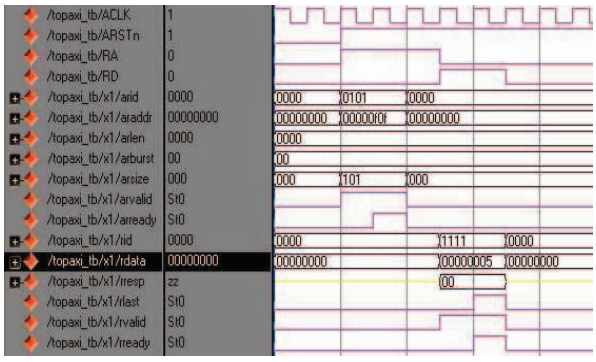

Fig.7. Waveform of single read operation.

The time taken for the single read operation is 50ns excluding the 20ns, the first two clock cycles of ARSTn i.e. the Reset Phase. ARSIZE [2:0] is set as 3'b010, indicating that the size of each transfer is 4 bytes. Thus the throughput that can be achieved while performing a single read transfer is 6.4 bits/clock cycle. A match exists between ARID, RID indicates that the correct slave is responding to the transfer operation. The slave accepts the master asserted address and control information 1 clock cycle after the master asserts it and similarly the master accepts the read data 1 clock cycle after the slave asserts it. The termination of the operation occurs at 90ns as indicated in the figure after the response is accepted by the master. The RRESP = 2'b00 indicates the success of the write operation.

C. BURST WRITE OPERATION

Simultaneous Burst Read and Burst Write operation by master 1, master 2 followed by a Burst Write and Burst Read operation by the same masters have been simulated. A write burst of 5 transfers indicated by AWLEN [3:0] from master 1 to slave 1 has been simulated as shown in Fig.8.

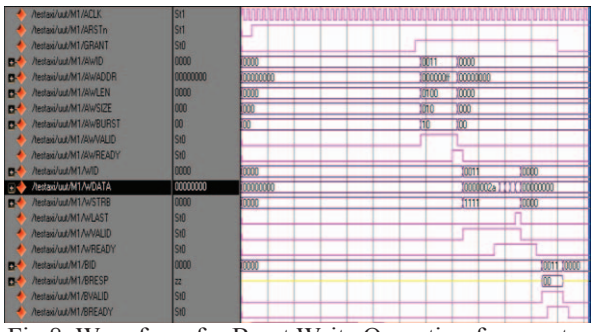

Fig.8. Waveform for Burst Write Operation for master 1 and slave 1 depicting the signals at the master interface.

The time taken for this burst write operation of 290ns. AWSIZE [2:0] is set as 3'b010, indicating that the size of each transfer is 4 bytes, thus the WSTRB is set as 4'b1111 indicating that all the 4 byte strobes are valid. Thus the throughput that can be achieved while performing a burst write transfer is 5.51 bits/clock cycle. The termination of the operation occurs at 320ns as indicated in the figure after the response is accepted by the master 1. The BRESP = 2'b00 (OKAY) indicates the success of the write operation.

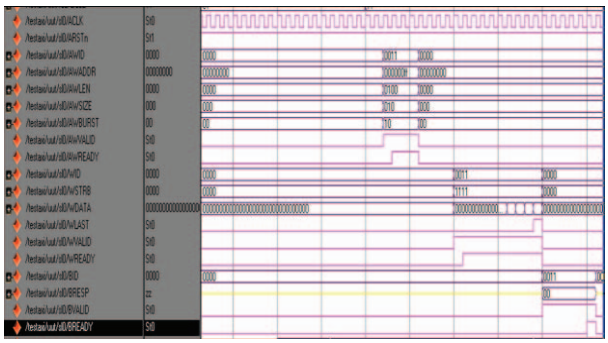

Fig.9. Waveform for Burst Write Operation for master1 and slave1 depicting the signals at the slave interface.

The match between AWID, WID, BID as shown in the Fig. 9 and Fig. 8 indicate that the slave 1 is responding to the write request of master 1 as desired. This burst operation shown in Fig. 10 and Fig. 11 takes a time of 290ns and the throughput that can be achieved is 5.51 bits/cycle. BRESP = 2'b00 (OKAY) indicates the success of the write operation. The match between AWID, WID and BID as shown in the Fig. 10 and Fig. 11 indicate that the slave 3 is responding to the write request of master 2 as desired.

Authorized licensed use limited to: Technische Universitat Kaiserslautern. Downloaded on March 14,2024 at 13:54:21 UTC from IEEE Xplore. Restrictions apply.

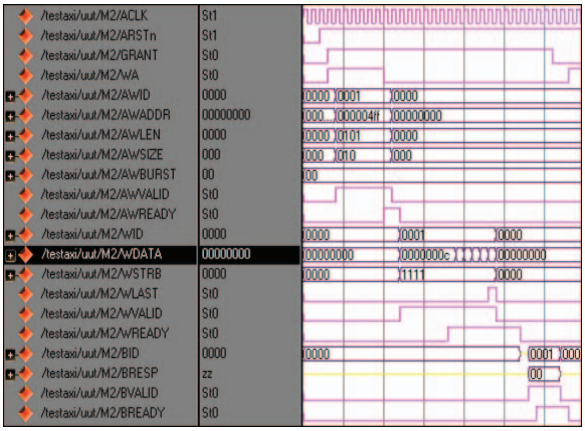

Fig.10. Waveform for Burst Write Operation for master2 and slave3 depicting the signals at the master interface.

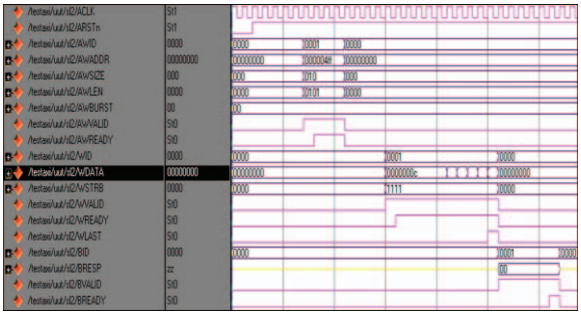

Fig.11. Waveform for Burst Write Operation for master2 and slave3 depicting the signals at the slave interface.

D. BURST READ OPERATION

A read burst of 5 transfers indicated by AWLEN [3:0] from master 1 to slave 2 and master 2 to slave 4 have been simulated as shown in Fig. 12 and Fig. 13. The time taken for this read operation shown in Fig. 12 is 210ns. ARSIZE [2:0] is set as 010, indicating that the size of each transfer is 4 bytes. Thus the throughput that can be achieved while performing a burst read operation of 5 transfers is 7.619 bits/clock cycle.There exists a read response for all the six transfers in the burst, RRESP [1:0] of 2\'b00 indicates the success of every transaction.

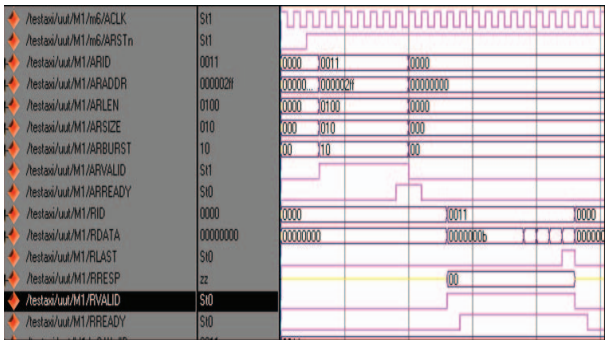

Fig.12. Waveform for Burst Read Operation for master1 and slave2 depicting the signals at the master interface.

A match exists between ARID, RID indicates that the correct slave is responding to the operation.

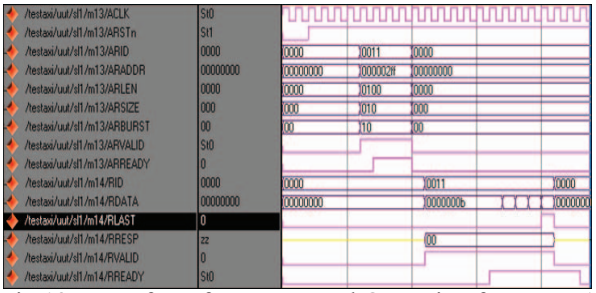

Fig.13. Waveform for Burst Read Operation for master1 and slave2 depicting the signals at the slave interface.

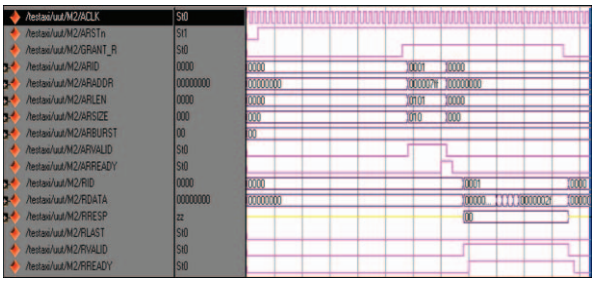

Fig.14. Waveform for Burst Read Operation for master2 and slave4 depicting the signals at the master interface.

This burst operation shown in Fig. 14 takes a time of 210ns and the throughput that can be achieved is 7.619 bits/clock cycle. RRESP = 2\'b00 (OKAY) indicates the success of the read operation.

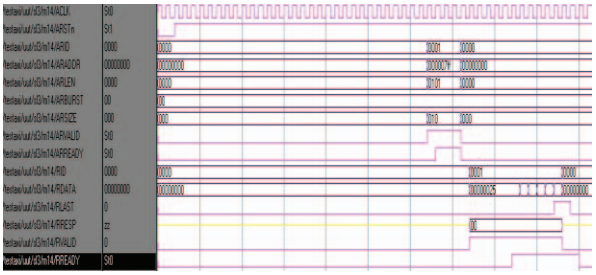

Fig.15. Waveform for Burst Read Operation for master2 and slave4 depicting the signals at the slave interface.

Table.3. Device Utilization Table for Master and Slave blocks.

Logic Utilization	Master Block		Slave Block	
	Used	Utilization	Used	Utilization
Number of Slices	49	2%	60	2%
Number of Slice Flipflops	77	1%	88	1%
Number of 4 Input LUT's	155	3.12%	112	2%
Number of bonded IOBs	155	98%	88	55%
Number of BUFG/ BUFGCTRLs	1	3%	1	3%
Maximum Operating Frequency	188.144MHz		175.982MHz	

Authorized licensed use limited to: Technische Universitat Kaiserslautern. Downloaded on March 14,2024 at 13:54:21 UTC from IEEE Xplore. Restrictions apply.

Table.4. Device Utilization Table for Arbiter and Decoder blocks

Logic Utilization	Arbiter Block		Decoder Block	
	Used	Utilization	Used	Utilization
Number of Slices	9	0%	3	0%
Number of Slice Flipflops	9	0%	4 (IOB Flipflops)	
Number of 4-Input LUT's	14	0%	5	0%
Number of bonded IOB's	9	5%	5	5%
Maximum Operating Frequency	298.958MHz			

.

E. ANALYSIS IN XILINX ISE

The verilog codes of master, slave, arbiter and decoder have been synthesized in Xilinx ISE using the Spartan 3 FPGA of speed grade -5 and the results have been tabulated in this section.

V. CONCLUSION

ARM Cortex processors are used in smart phones which demand high data transfer speeds. Thus the need for high speed, high throughput protocols is increasing. In this paper, AXI bus interface modules were implemented in Modelsim using verilog HDL and simultaneous read and write data transfers among a Soc of two masters and four slaves were simulated in Modelsim SE and all the modules were synthesized on a Spartan 3E FPGA. The maximum throughput achievable with the AXI protocol occurs for 16 data transfers. The calculated throughput is 5.51 bits/clock cycle for write and 7.619 bits/clock cycle for read transfer each of 6 transfers. The AXI protocol also specifies features for protection against illegal transactions, support for locked and exclusive access. Thus AXI protocol accounts for reliability, safety and high bandwidth. It is thus widely accepted and used data transfer protocol.

REFERENCES

[1] AMBA AXI Protocol Specification, 2003, ARM Limited.

[2] AXI for Xilinx System Development Reference Guide, ug 761(v 13.4), January 18, 2012.

[3] IBM, Core connect bus architecture, IBM Microelectronics, http://www.ibm.com/chips/products/ coreconnect, 2000.

[4] Shaila S Math, Manjula R B, "Design Of AMBA AXI4 protocol for System On Chip communication," *International Journal Of Communication Network And Security (IJCNS)*, Vol-1, Issue-3, ISSN: 2231-1822.

[5] Daniel C/K. Kho, Kumar Munusamy, "Transaction-based SoC Design Techniques for AMBA AXI4 Bus Interconnects using VHDL," *11th International Conference Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, 2014, 14-17 May, 2014, pp. 1-6.

[6] Chien-Hung Chen, Jiun-Cheng Ju and Ing-Jer Huang, "A Synthesizable AXI Protocol Checker for SoC Integration", *IEEE transl*, ISOCC, Vol 8, pp 103-106, 2010.

[7] Veena Abraham, Soumen Basak, Sabi S, "Design of AXI4 Protocol Checker for SoC Integration," *International Journal Of Emerging Science and Engineering (IJESE),* ISSN: 2319-6378, vol.1, Issue 8, June 2013.

[8] Christian de Schryver, Pedro Torruella, Nobert Wehn, "A Multi-Level Monte Carlo FPGA Accelerator for Option Pricing in the Heston Model," *Proceedings of the conference on Design, Automation and Test in Europe*, March 2013.

[9] D. Jungewelter, D. Cozzi, D. Kleibrink, S. Korf, J. Hagemeyer, M. Porrman, J. Ilstad, "AXI-based SpaceFibre IP Core Implementation," *SpaceWire Conference (SpaceWire),* 2014, pp.1-6.

[10] Vijaykumar, R K Karunavathi, Vijay Prakash, "Design of Low Power Double Data Rate 3 Memory Controller with AXI compliant," *International Journal of Engineering and Advanced Technology (IJEAT),* ISSN: 2249-8958, vol. 1, Issue 5, June 2012.

[11] Fu-ming Xiao, Dong-sheng Li, Gao-Ming Du, Yu-kun Song, "Design of AXI bus based MPSoC on FPGA," *3rd International Conference on Anti-Counterfeiting,* Security and Identification in Communication (ASID) 2009.

[12] Sanghun Lee, Chanho Lee, Hyuk-Jae Lee, "A new multi-channel on-chip-bus architecture for system-on-chips," in *Proceedings of IEEE international SoC Conference,* September 2004.

[13] Golla Mahesh, Sakthivel.S.M, "Verification IP for an AMBA-AXI Protocol using System Verilog," *International Journal Of Engineering Research and General Science*, vol. 3, Issue 1, January-February 2015, ISSN: 2091-2730.

[14] Chenghai Ma, Zhijun Liu, Xiaoyue Ma, "Design And Implementation of APB Bridge based in AMBA 4," *International Conference on Communications and Networks*, 2011.

[15] Muhammed Asharaf.T.P, Manu Ramesh, Sankarnarayana Bhat.M, "Design and Implementation of High performance Bus Architecture using FPGA," *International Journal of Engineering and Computer Science*, ISSN: 2319-7242, vol. 3, Issue 6, June 2014, pp. 6578-6583.

Authorized licensed use limited to: Technische Universitat Kaiserslautern. Downloaded on March 14,2024 at 13:54:21 UTC from IEEE Xplore. Restrictions apply.