Design of AXI4 Protocol Checker for SoC Integration

Veena Abraham, Soumen Basak, Sabi S

Abstract—System-on-a-Chip (SoC) design has become more and more complex because Intellectual Property (IP) core with different functions are integrated within a single chip. Each IPs have completed design and verification but the integration of all IPs may not work together. The bus-based architecture has become the major integration methodology for implementing a SoC. The main issue is to ensure that the IP works correctly after integrating to the corresponding bus architecture. Advanced extensible interface 4 (AXI4) is an on chip bus architecture introduced by ARM to interact with its peripherals. A synthesizable AXI4 protocol checker which contains a set of rules to check on-chip communication properties accuracy is proposed to ensure proper SoC integration. A prototype of AXI4 Master and AXI Slave is also designed to generate the AXI4 signals. The protocol checker will continuously monitor the signals from AXI4 Master and AXI4 Slave to check whether any rule is broken or not. The proposed AXI4 protocol checker will check both the Write Channel and Read Channel transactions. As the AXI4 checker is synthesizable it can be used in FPGA (Field Programmable Gate Array) and Emulation where functional checks are difficult to detect and pin point. The AXI4 Master, AXI Slave and AXI4 protocol checker have been modeled using Verilog and implemented on Digilent Basys2 Spartan 3E FPGA board.

Index Terms— System on a Chip, AXI4 protocol, Intellectual Property cor, Write Channel, Read channel..

I. INTRODUCTION

Process technologies are shrinking and design sizes are increasing during recent years. This has led to highly complex integrated circuits (ICs). Hence manufacturers are integrating large number of components on a single chip. System-on-a-chip (SoC) [1] designs have evolved to address the ever increasing complexity of different applications. A System- on – a Chip (SoC) contains a wide variety of elements like random logic, memory, CPU and analog circuitry. In order to increase productivity, SoC design make use of intellectual property cores (IP). IP blocks are predesigned components that are used in larger design. When IP core with different functions, which have completed design and verification are integrated within a single chip all IPs may not work together. The important problems faced are violation of bus protocol or transaction error.

The major integrated methodology for implementing SoC is bus based architecture. The on-chip communication specification provides a standard interface that facilitates IPs integration and easily communicates with each IPs in a SoC. Some of the popular bus-based communication architecture standards are IBM CoreConnect [2], OpenCores Wishbone [3], ARM Microcontroller Bus Architecture (AMBA) versions 2.0 and 3.0 [4], STMicroelectronics STBus, Sonics SMARRT Interconnect, and Altera Avalon.

Manuscript received June, 2013.

Veena Abraham, Department of Electronic and Communication, Sree Buddha College of Engineering, Kerala.

Soumen Basak, Chief Technology Officer, DSipher Design Solutions Pvt. Ltd, Bangalore.

Sabi S, Department of Electronic and Communication, Sree Buddha College of Engineering, Kerala..

AMBA one of the leading on-chip bus standard used in high performance SoC design was developed by ARM in 1996. The three buses specified within the AMBA bus are: ASB (Advanced System Bus) [4] , APB (Advanced Peripheral Bus and AHB (Advanced High-performance Bus) . ARM introduced the 3rd generation in 2003 which included Advanced eXtensible Interface (AXI) and the Advanced Trace Bus (ATB). AMBA 3.0 AXI supports [5] channel-based specification, with five separate channels. The 4th generation was introduced in 2010 which included AMBA 4 AXI4 [6], AXI4-Lite, and AXI4-Stream Protocol. AXI4 which is the latest revision of the AXI3 protocol have addressed several known issues in AXI3.

Hyun-min Kyung et al proposed a Performance Analysis Unit (PAU) for monitoring the AMBA AXI bus system [7] and the usage of the PAU with the H.264 decoding application. Micro architectures with Netwok Interface [8] which were compatible with the AMBA AXI protocol was proposed by Masoud Daneshtalab et al.

Many verification works based on formal verification techniques exist. Device under test (DUT) is modeled as finite-state transition and its properties are written by using computation tree logic (CTL). A dynamic SDRAM [9] access scheduler (DSAS) is introduced as part of memory controller which will record the early access request in request buffer and when new request comes, it will compare the bank address and row address with the access records. DSAS works as a slave device on the bus. Different verification tools were used to verify DUT's behaviors. A bus interface design technique, called Efficient Bus Interface (EBI) [10] was introduced, to reduce the communication delay between the Intellectual Property (IP) core and the memory connected through AMBA3 AXI bus for mobile systems. NoC [11] research over the past decade has aided the development of a number of techniques that are relevant to the future of on-chip interconnects. Although formal verification can verify DUT's behaviors thoroughly, there are still unpredictable bug in the chip level which need to be verified.

To check PCI bus protocol Kanna Shimizu et al. [12] proposed an early rule-based monitor. A high-level specification style was proposed by Oliveira and Alan J. Hu [13] that could generate a hardware monitor. An extremely easy way to generate monitors for common interface idioms was provided. M. S. Jahanpour et al. proposed an interface-monitor-based methodology [14] to watch the interface between a block and the rest of the system.

When several IPs are integrated the approach of checking DUT cycle by cycle to make sure the DUT obeys all the rules during simulation, is very efficient. The monitored-based approach often cannot find errors in simulation environment since many errors may occur in real-time.

HPChecker is an efficient AMBA AHB on-chip bus protocol checker [15]. For high performance SoC requirements, the AMBA AXI on-chip bus protocol is defined that targets at high-performance, high frequency system design and includes a number of features that make it suitable for a high-speed submicron interconnects. The AXI protocol checker [16] consists of configuration register, protocol checker and error reference table.

This paper is organized as follows. Section II gives an introduction to AXI4 Protocol. Section III presents proposed prototype of AXI4 Master, AXI4 Slave and the AXI4 checker. The results will be discussed in Section IV followed by conclusion in section V.

II. AMBA AXI4 ARCHITECTURE

The AMBA AXI4 protocol [6] has a number of features which make it suitable for high-performance, high-frequency system designs and high-speed submicron interconnect. The important features of the AXI protocol are: it has separate address/control and data phases, unaligned data transfers using byte strobes are supported, only start address is being issued for burst-based transactions, separate read and write data channels and out-of-order transaction can be completed.

AMBA AXI4 system consists of master, slave and bus. Each of the master and slave are identified by their own four bit ID tags. The system consists of five channels namely write address channel, write data channel, write response channel, read data channel and read address channel. The AXI4 also includes the following features: burst lengths up to 256 beats are supported, removal of WID and locked transactions, AWCACHE and ARCACHE signaling are updated, multiple region interfaces are supported and optional Quality of Service (QoS) and User signalling.submit your final version, after your paper has been accepted, prepare it in two-column format, including figures and tables.

A. Architecture of Read and Write Channel

Read transaction uses the read address and read data channels. Write transaction uses the write address, write data, and write response channels. Burst-based transaction has address and control information on the address channel, for both read and write, which describes the nature of the data to be transferred. The data is transferred from slave to the master in a read data channel. In write transactions all the data flows from the master to the slave. On the completion of the write transaction reponse signal is send to the master through an additional write response channel.

B. AXI4 Signals

The AXI4 signals [6] can be categorized as global signals, write address channel signals, write data channel signals, write response channel signals, read address channel signals and read data channel signals. Table I gives a list of AXI4 signals. Global signals are ACLK and ARESETn. All signals are sampled on the rising edge of the global clock signal ACLK.

TABLE I. AXI4 SIGNALS

Write Address Channel	AWVALID, AWREADY, AWID[3:0], AWADDR[31:0] AWLEN[7:0],AWSIZE[2:0],AWBURST[1:0],AWCACHE [3:0], AWPROT[2:0] and AWLOCK].
Write Data Channel	WVALID, WREADY, WDATA[31:0], WSTRB[3:0] and WLAST.
Write Response Channel	BVALID, BREADY, BID[3:0] and BRESP[1:0].
Read Address Channel	ARVALID, ARREADY, ARID [3:0], ARADDR[31:0], ARLEN[7:0], ARSIZE[2:0], ARBURST[1:0], ARLOCK, ARCACHE[3:0] and ARPROT[2:0].
Read Data Channel	RVALID, RREADY, RDATA[31:0], RLAST, RID[1:0] and RRESP[1:0]

C. AXI4 Protocol Transactions

AXI protocol transactions [6] are based on VALID and READY handshake mechanism. The VALID/READY handshake is used by all five channels of AXI to transfer data, address and control information. When data or control information is available the source generates the VALID signal. When the destination accepts the data or control information it generates the READY signal. Transfer occurs only when both the VALID and READY signals are HIGH.

III. PROPOSED MODEL OF AXI4 MASTER, AXI4 SLAVE AND AXI4 CHECKER

The proposed AMBA AXI4 architecture consists of a model of AXI4 Master and AXI4 Slave. The communication between one master and one slave is carried out in the thesis. A set of rules have been implemented in the AXI4 Checker. Fig. 1 shows the proposed AXI4 architecture. The important blocks of the architecture are Transaction Generator, Clock and Reset Generator, AXI4 Master, AXI4 Slave and AXI4 Checker . Clock and Reset Generator generates the clock and reset signal.

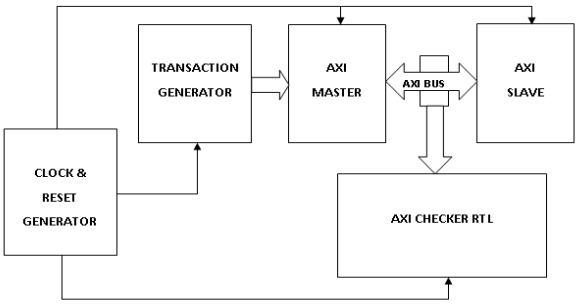

Fig. 1.AXI4 Architecture

A. Transaction Generator

Transaction generator generates random transactions. The Transaction Generator contains Write Data Channel Task, Write Address Channel Task and Read Address Channel Task. Identification ID, control information and address are randomly generated by Write Address Channel Task and Read Address Channel Task respectively.WDATA is randomly generated by means of Write Data Channel Task. A synthesizable transaction generator is also developed to demonstrate the working in FPGA.

B. The AXI4 Master

The AXI Master component is FSM based. The different components in AXI4 Master is shown in Fig. 2. It takes input from 'Transaction Generator' and initiates it on the AXI bus.

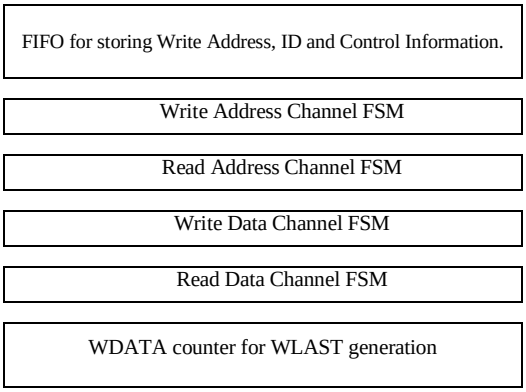

Fig. 2.AXI4 Master

1) FIFO For Storing ID and Control Information of Write Channel: As Write Address and Write Data channel are independent a FIFO is used for storing ID and control information for Write Address Channel. The write data must come in the same order as the correponding write addresses. Write transfer are tracked in the following way. AWVALID and AWREADY both being high indicates address and control signals for that AWID should be stored in the FIFO. WVALID and WREADY both being high i ndicates a valid data beat for that ID has been transferred. WVALID , WREADY and WLAST being high indicates last data beat for that AWID has been transferred and corresponding AWID can be removed from the FIFO. BVALID and BREADY both being high indicates a valid write response for an ID which terminates that write access. Write address or write data may come earlier or simultaneously.

Example of FIFO Storing AWID is shown in Fig.3. When AWVALID and AWREADY is high AWID 7,1 and 3 are stored in FIFO respectively. When the last data beat of AWID 7 comes it is removed from the FIFO.

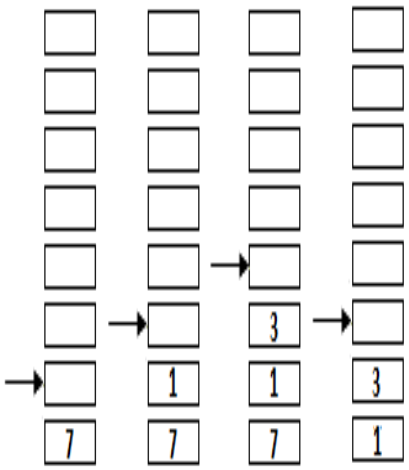

Fig. 3.Example of FIFO Storing AWID

2) *FSM of Master Read Address Channel:*

FSM of Master Read Address Channel shown in Fig. 4 asserts ARVALID high and initiates the ARID, ARADDR and control information like ARLEN, ARSIZE, ARBURST etc on the AXI bus. The triggering inputs are ARREADY from Slave and "xfr" which is user defined input from 'transaction generator' indicating a read transfer. In the FSM "control" indicates read control signal values, "x" stands for don't care and "val" indicates random values from 'transaction generator for Read Address Channel.

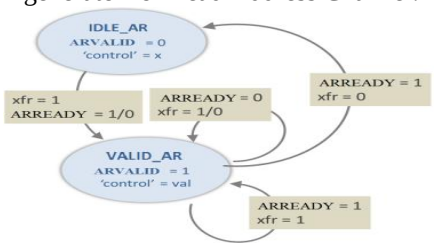

Fig. 4.FSM of Master Read Address Channel

3) *FSM of Master Read Address Channel:* FSM of Master Read Data Channel shown in Fig. 5 asserts RREADY signal high. The triggering inputs are RVALID and RLAST from Slave and "busy" which is internal signal to denote master is not ready to receive data. The different states in the FSM are IDLE_RD, READY, BEAT_IDLE and BEAT_READY. As AXI is burst based if ARLEN is greater than zero more beats need to be transferred and it moves on to the BEAT_IDLE and BEAT_READY states.

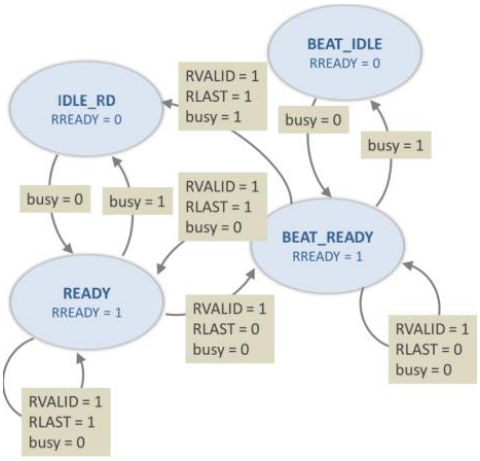

Fig. 5.FSM of Master Read Data Channel

4) *WLAST Generation and Other FSM's:* FSM of Master Write Address Channel is similar to the FSM of Master Read Address Channel but the triggering inputs are AWREADY from Slave and "xfw" which is input from 'transaction generator' indicating a write transfer. In the VALID state valid write address and control information is available. FSM of Master Write Data Channel asserts WVALID high and initiates random data to WDATA. The triggering input is busy signal which indicates whether Master is busy or not. A WDATA counter designed is incremented each time a Write Data transaction occurs and is compared with the current AWLEN for WLAST generation.

C.

D. The AXI4 Slave

The AXI4 slave component interacts with AXI4 Master to mimic AXI transactions and is FSM based. Parameterized memory space is implemented in the Slave for storage of data. Different components in AXI4 Slave is shown in Fig.6

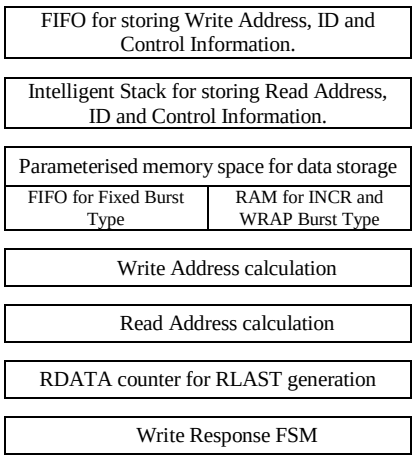

Fig. 6.AXI4 Slave

1) *Intelligent Stack For Storing ID and Control Information of Read Channel*: An intelligent stack is used for storing ID and control information for Read Address Channel as the Read Data and Read Address channel are independent. For Read Data Channel data beats with one ID can come in between data beats of other ID . Hence for Read channel an Intelligent stack is used instead of a FIFO. Read transfer are tracked in the following way. Both ARVALID and ARREADY being high indicates address and control signals for that ID should be stored. RVALID and RREADY both being high indicates a valid data beat for that id is transfered. RVALID , RREADY and RLAST being high indicates last data beat for that id and it should be removed from the stack. Read data cannot precede read address.

The working of the intelligent stack is different from ordinary stack. Example of Intelligent Stack storing ARID is shown Fig. 7. When ARVALID and ARREADY is high ARID 7,1 and 3 are stored in Intelligent Stack respectively. For Read Data Channel corresponding to each data beat RID also appears on the AXI bus. When last data of RID 1 comes, ARID 1 is removed from stack.

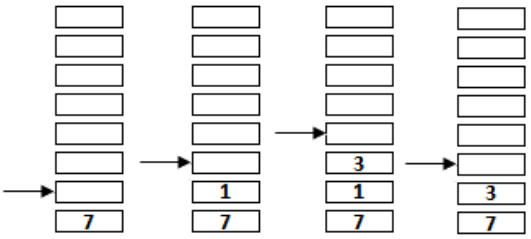

Fig. 7.Example of Intelligent Stack storing ARID

2) *FSM for Write Response Channel:* FSM for write response channel is shown in Fig. 8 Here the triggering input is WLAST signal which indicates last data of the write burst has been transferred. When the FSM is in WR_V_VALID1 state BAVLID is asserted and BID is equal to AWID corresponding to the last data beat transferred.

3) Parameterized Memory Space and Address Calculation: Parameterized memory space is implemented in the Slave for storage of data. WDATA is stored in fixed address when AWBURST is FIXED type and stored in RAM when burst type is INCR or WRAP. Only the intial address of the read and write transaction gets transferred in the first cycle and all the address calculation are done by the AXI4 Slave.

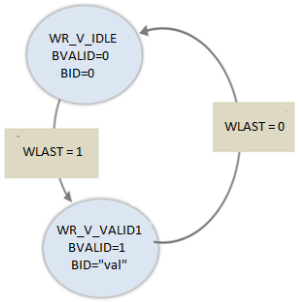

Fig. 8.FSM for Write Response Channel

1) RLAST Generation and Other Blocks: RLAST is generated by means of a counter which is continuosly being compared with AWLEN. As Read data cannot precede read address the generation of RVALID signal depends on ARVALID. Generation of AWREADY and ARREADY depends on whether FIFO and Intelligent Stack is full or not. FIFO for Storing ID and Control Information of Write Channel is explained in Section B.

E. The AXI4 Checker

The AXI4 Checker consists of internal data structure and protocol checker. Protocol checks are done using internal data structures and different module inputs. Different components of the Protocol checker are shown in Fig. 9.

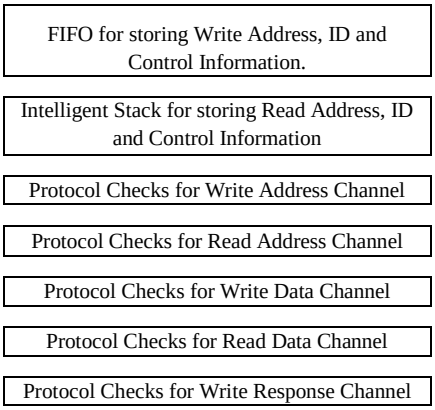

Fig. 9.AXI4 Checker

The AXI4 protocol specification has been studied and a list of rules has been defined for Read Address Channel, Read Data Channel, Write Address Channel, Write Data Channel and Write Response Channel.Table II shows some of the protocol checks included in the checker. All the checks implemented are not specified in Table II instead the different types of checks are shown.Corresponding to each of the checks a flag is defined which will be high if the check fails. Also message will be displayed which gives the check name, severity of error, a description of the error, which part of specification [6] is violated and the time at which the check fails. Additionally an error flag and warning flag is defined which will be high if an error or check occurs.

A Check number indicator which indicates the check number as shown in Table II is also defined.

TABLE II. LIST OF DIFFERENT TYPES OF AXI4 PROTOCOL CHECKS

Spec. Section	Chk . No	Check Name	Functionality	Severity
3.1	1	AXI4_arid_stable	Read Address Channel	Error
13.1.2	2	AXI4_arlen _limitation1	Read Address Channel	Error
3.1.4	3	AXI4_arready_default	Read Address Channel	Warning
13.1.2	6	AXI4_arlen _limitation2	Read Address Channel	Error
3.1.5	E	AXI4_rready_default	Read Data Channel	Warning
13.1.2	11	AXI4_awlen _limitation1	Write Address Channel	Error
4.3	16	AXI4_awsize	Write Address Channel	Error
3.1	1C	AXI4_wdata _stable	Write Data Channel	Error
3.1.3	1F	AXI4_bready_default	Write Response channel	Warning

IV. RESULTS AND DISCUSSION

A prototype of AXI4 Master and AXI4 Slave and AXI4 checker has been designed using Verilog. Simulation and testing has been carried out by means of test cases, using Icarus Verilog and the waveforms were viewed using GTKWave. The different modules have been synthesized using Xilinx ISE, implemented on Spartan 3E FPGA [17] and also waveforms were verified using ISIM. For interfacing the PC with FPGA, Adept application [18] has been used.

A. Simulated Output Using Test Cases

During the design phase the different types of checks have been checked, by purposely creating the errors to show that the AXI4 checker can catch these errors.Simulated result obtained using two test cases will be discussed. Simulation results of both Read and Write channel signals generated by AXI4 Master and AXI4 Slave which are error free is shown in Fig. 10. Snapshot of the contents of RAM and FIFO at the Slave side are shown in Fig. 11, where the data is stored.

Fig. 10. Simulated output of Error Free Read and Write channels

Considering the first write and read transaction from Fig. 10 and Fig. 11 we can see that WDATA is first written to RAM[8]. The Write address and Read address channel information are stored in FIFO and Intellegent stack respectively. From FIFO AWLEN[0]=8'h02 and AWBURST[0]=2'b01, WLAST is generated after three data beats are transferred. Proper address calculation are done and WDATA is stored in RAM[12] and RAM[16]. Also BID=8 which is equivalent to AWID[0] showing that the transaction is successfull. From stack ARID[0]=4'h8, ARLEN[0]=8'h02 and ARBURST[0]=2'b01. RID=8 for the first three data transfer and RDATA is read from addresses RAM[8], RAM[12] and RAM[16] and RLAST is also generated at proper time. The other transactions are also successful.

```
Contents of RAM and FIFO storing data at time                230
RAM[0] = 00000000
RAM[1] = 00000000
RAM[2] = 00000000
RAM[3] = 80151600
RAM[4] = 00000000
RAM[5] = 00000000
RAM[6] = 00000000
RAM[7] = 800a8c00
RAM[8] = 80151600
RAM[9] = 800a8c00
RAM[10] = 00000000
RAM[11] = 80151600
RAM[12] = 80151600
RAM[13] = 800a8c00
RAM[14] = 00000000
RAM[15] = 00000000
RAM[16] = 80054600
RAM[17] = 800a8c00
RAM[18] = 00000000
RAM[19] = 00000000
FIFO[0] = 800a8c00
```

Fig. 11. Snapshot of contents of RAM and FIFO at slave side.

Errors are purposely introduced in the second test case to show how the checker catches the error. Simulated output and display messages when different error occurs is shown in Fig. 12 and Fig. 13. The check number indicator signal AXI_error shows check number 0E, 11, 16 and 06 corresponding to the checks AXI4_rready_default , AXI4_ awlen_limitation1, AXI4_awsize and AXI4_arlen _limitation2 respectively.

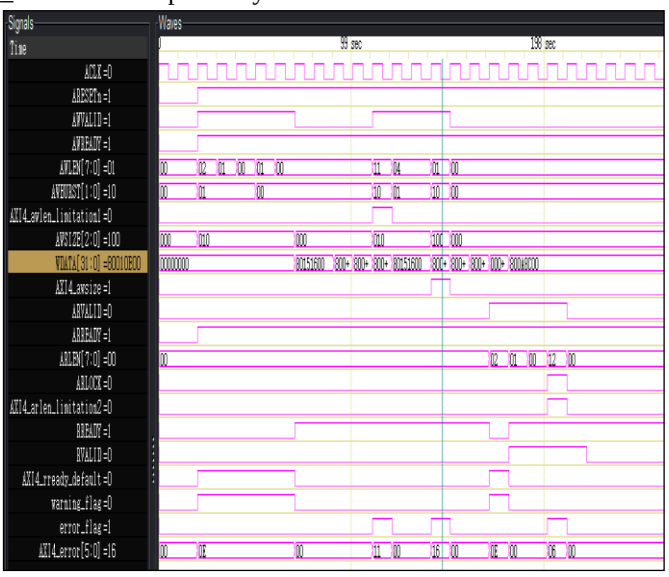

Fig. 12. Simulated output showing different checks.

Corresponding flags of the checks are also high. The error indicator flag and warning indicator flag becomes high when an error or warning occurs.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

```
AXI4_rready_default=1
WARNING [AXI4_rready_default - AXI VER:2 Spec.Sec.3.1.5] at time                30
The default value of RREADY can be HIGH, but only if the master is able to accept read data  immediately, whenever it pe
rforms a read transaction.
AXI4_awlen_limitation1=1
ERROR [AXI4_awlen_limitation1 - AXI VER:2 Spec.Sec.13.1.2] at time              120
Bursts longer than 16 beats are only supported for the INCR burst type. Both WRAP and FIXED burst types remain constrain
ed to a maximum burst length of 16 beats.
AXI4_awsize=1
ERROR [AXI4_awsize - AXI VER:2 Spec.Sec.4.3] at time                150
The size of write transfer must not exceed the data bus width of the components in the transaction.
AXI4_arlen_limitation2=1
ERROR [AXI4_arlen_limitation2 - AXI VER:2 Spec.Sec.13.1.2] at time               210
Exclusive accesses are not permitted to use a burst length greater than 16.
```

Fig. 13. Snapshot of Error Messages

B. Experimental Setup

The AXI4 Master, AXI4 Slave and AXI4 checker have been synthesized using Xilinx ISE and implemented on Digilent Basys2 board built around Xilinx Spartan-3E 100 FPGA. For the FPGA setup a synthesizable transaction generator has been designed to drive the AXI4 Master. Simulated output with FPGA setup using ISIM is shown in Fig.14. Adept application has been used to interface the PC with Digilent's Spartan 3E FPGA [17] board. Fig. 15 shows the snapshot after successfully burning the program to board by means of Adept [18] and Fig. 16 shows the FPGA setup.

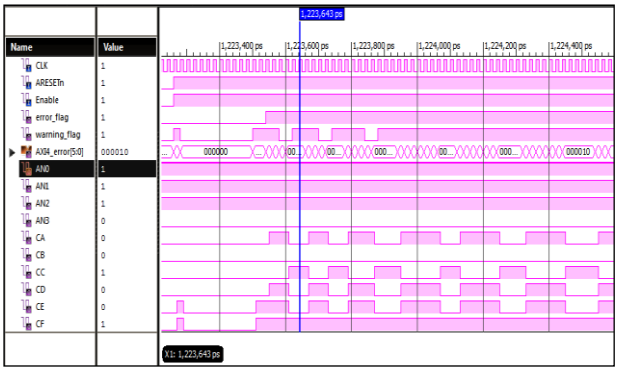

Fig. 14. Simulated Results with FPGA setup

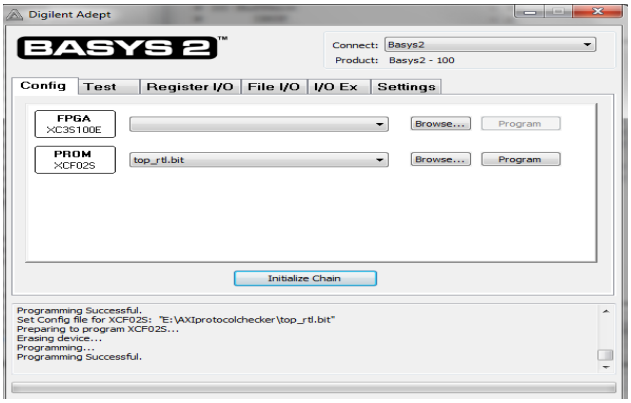

Fig. 15. Snapshot of succecssful burning to board using Adept

LED's one and two represents the warning flag and error flag. The check number corresponding to the error captured is displayed on the LCD screen and also on five LED's. Check number 2 corresponding to check AXI4_arlen _limitation1 is being displayed in Fig.16. If more than one error or warning occurs at the same time then only the check number corresponding to smaller one will be displayed on FPGA, but display messages appear on the console. Once we rectify it and again burn it to FPGA others can be detected.

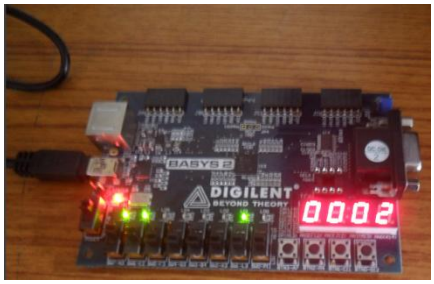

Fig. 16. Snapshot of FPGA setup

The final Device utilization summary is shown in Table III and timing summary is shown in Table IV .

TABLE III. DEVICE UTILIZATION SUMMARY

Selected Device	3s100ecp132-5	
Number of Slices	465 out of 960	48%
Number of Slice Flip Flops	391 out of 1920	20%
Number of 4 input LUTs	860 out of 1920	44%
Number of IOs	23	
Number of bonded IOBs	23 out of 83	27%
Number of GCLKs	2 out of 24	8%

TABLE IV. TIMING SUMMARY

Minimum period	9.511 ns
Maximum combinational path delay	11.289 ns
Total REAL time to Xst completion	636.00 secs
Total CPU time to Xst completion	636.51 secs

V. CONCLUSION

AMBA AXI4 is a plug and play protocol developed by ARM which defines bus specification and a technology independent methodology for designing, implementing and testing customized highly-integrated embedded interfaces. In the thesis the AXI protocol has been studied thoroughly and a list of protocol checks for read and write channel have been defined. The defined rules have been implemented in the checker. A model of AXI4 Master and AXI4 Slave has been designed to generate the read and write transactions. The AXI4 protocol checker continuously monitor the signals from AXI4 Master and AXI4 slave and flags messages (errors, warnings or information) based on protocol checks implemented inside it. The AXI4 Master, AXI4 slave and AXI4 protocol checker has been modeled using Verilog, synthesized using Xilinx ISE and implemented on Spartan 3E FPGA . Different test cases have been used to check the functionality and by observing the display messages and simulated waveforms errors can be rectified. Constrained random verification technique is also used in addition to directed test cases which has made the testing more thorough. As the checker is synthesizable it can be easily converted to system Verilog assertions and used with Formal verification flow. The proposed AXI4 checker can be used to ensure proper SoC integration.

ACKNOWLEDGMENT

The author would like to thank KELTRON, Thiruvananthapuram for the technical support provided and also acknowledge the help of Ms. Betty John(Deputy General Manager, KELTRON, Kerala).

REFERENCES

1. S. Pasricha, N. Dutt, On-Chip Communication Architectures: System on Chip Interconnect, Morgan Kaufmann, 2008.
2. IBM, Core connect bus architecture. IBM Microelectronics. Available at http://www.ibm.com
3. Wishbone system-on-chip (soc) interconnection architecture for portable IP cores. Available at http://www.opencores.org
4. ARM, "AMBA Specification (Rev 2.0)". [Online] Available: http://www.arm.com
5. A. Shrivastava, G.S Tomarand A.K. Singh, "Performance Comparison of AMBA Bus-Based System-On-Chip Communication Protocol", in Proc. Int. Conf. Communication Systems and Network Technologies (CSNT), June 2011, pp. 449- 44.
6. ARM, "AMBA AXI protocol specifications (Version 2), March 2010", [Online] Available: http://www.arm.com.
7. Hyun-min Kyung , Gi-ho Park , Jong Wook Kwak , Tae-jin Kim and Sung-Bae Park, "Design and implementation of Performance Analysis Unit (PAU) for AXI-based multi-core System on Chip (SOC)" , Elsevier Trans. Microprocessors and Microsystems, vol. 34, pp. 102-116, March 2010.
8. M. Daneshtalab, M. Ebrahimi, P. Liljeberg, J. Plosila and H. Tenhunen, "Memory-Efficient On-Chip Network with Adaptive Interfaces", IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 31, issue 1, pp. 146 -159, Jan. 2012.
9. Jun Zheng, Kang Sun , Xuezeng Pan, and Lingdi Ping "Design of a Dynamic Memory Access Scheduler", in Proc. IEEE 7 th Int. Conf. ASIC, Oct. 2007, pp. 20-23.
10. Na Ra Yang, Gilsang Yoon, Jeonghwan Lee, Intae Hwang, Cheol Hong Kim, Sung Woo Chung and Jong Myon Kim, "Improving the System-on-a-Chip Performance for Mobile Systems by Using Efficient Bus Interface", in Proc. IEEE Int. Conf. Communications and Mobile Computing, Vol 4, March 2009, pp. 606-608.
11. Bruce Mathewson "The Evolution of SOC Interconnect and How NOC Fits Within It", in Proc. IEEE 47 th Int. Conf . Design Automation(DAC), June 2010, pp. 312-313.
12. Kanna, Shimizu, David L. Dill and Alan J. Hu. "A monitor-based formal specification of PCI", in Proc. Springer-Verlag London 3rd Int. Conf. on Formal Methods in Computer-Aided Design, Nov. 2000, pp. 335-353.
13. Marcio T. Oliveira, Alan J. Hu, "High level specification and design: High-Level specification and automatic generation of IP interface monitors", in Proc. 39th Conf. on Design automation, June 2002, pp. 129-134.
14. M. S. Jahanpour, E. Cerny, "Compositional verification of an ATM switch module using interface recognizer/suppliers (IRS)", in Proc. IEEE Int. Conf. International High-Level Design, Validation, and Test Workshop, 2000, pp. 71-76.
15. Y.-T. Lin, C.-C. Wang, and I.-J. Huang, "AMBA AHB Bus Protocol Checker with Efficient Debugging Mechanism," In Proceedings of the IEEE International Symposium on Circuits and Systems(ISCAS'08), May 2008, pp. 929-931.
16. Chien-Hung Chen, Jiun-Cheng Ju, and Ing-Jer Huang, "A Synthesizable AXI Protocol Checker for SoC Integration", in Proc. IEEE Int. Conf. SoC Design (ISOCC) , Nov. 2010, pp.103-106.
17. Digilent Basys2 Board Reference Manual. [Online] Available: http://www.digilentinc.com
18. Adept™ Application User's Manual. [Online] Available: http://www.digilentinc.com

AUTHORS PROFILE

Ms. Veena Abraham, is pursuing M.Tech in Embedded System from Sree Buddha college of Engineering, Kerala. Her areas of interest are Embedded Systems, VLSI Design and Digital Electronics.

Mr. Soumen, is Chief Technology Officer at DSipher Design Solutions Pvt. Ltd, Bangalore and also provides technical support for the VLSI department at KELTRON. He was part of organizations such as Motorola, Freescale and Synopsys and specialized in next-generation verification technologies. He is a graduate from IIT Bombay and his main area of interest is VLSI Design.

Mr. Sabi S, is Assistant professor at Sree Buddha college of Engineering and has pursued M.Tech in Microwave and TV Engineering from College of Engineering Trivandrum. His areas of interest are Microwave and Optical Communication, Electronic Circuits and Digital Electronics.