SYNTHESIZABLE AXI4 PROTOCOL CHECKER

Prasanth M and Juhi Raj
Department of Electronics and Communication Engineering, Nehru College of Engineering and Research Centre, Pampady, Thrissur, Kerala, India.

ABSTRACT
Bus based system-on-a-Chip (SoC) design has become the major integrated methodology for shortening SoC design time. The main challenge is how to verify on- chip bus protocols efficiently. Although traditional simulation-based on-chip bus protocol checkers are still lack of an efficient bus protocols verification environment such as FPGA level or chip-level. To overcome the shortage, propose a synthesizable AMBA AXI4 protocol checker. ARM introduced the Advanced Microcontroller Bus Architecture (AMBA) 4.0 specifications which includes Advanced extensible Interface (AXI) 4.0. AMBA bus protocol has become the de facto standard SoC bus. That means more and more existing IPs must be able to communicate with AMBA4.0 bus. This paper presents a project aimed to do data transactions on SoC bus using AMBA AXI4 protocol and verifies the transactions using a protocol checker, which is modelled in Verilog hardware description language (HDL). Icarus verilog and GTKWave viewer are used for verification strategy.

KEYWORDS: *System-on-a-chip (SoC), FPGA, AMBA, AXI, HDL, Icarus verilog.*

I. INTRODUCTION

In recent years, the improvement of the semiconductor process technology and the market requirement increasing. More difference functions IPs are integrated within a chip. Maybe each IPs had completed design and verification. But the integration of all IPs could not work together. The more common problem is violation bus protocol or transaction error. The bus-based architecture has become the major integrated methodology for implementing a SoC. The on-chip communication specification provides a standard interface that facilitates IPs integration and easily communicates with each IPs in a SoC.

To speed up SoC integration and promote IP reuse, several bus-based communication architecture standards have emerged over the past several years. Since the early 1990s, several on-chip bus-based communication architecture standards have been proposed to handle the communication needs of emerging SoC design. Some of the popular standards include ARM Microcontroller Bus Architecture (AMBA) versions 2.0 and 3.0, IBM CoreConnect, STMicroelectronics STBus, Sonics SMARRT Interconnect, OpenCores Wishbone, and Altera Avalon [1]. On the other hand, the designers just integrate their owned IPs with third party IPs into the SoC to significantly reduce design cycles. However, the main issue is that how to efficiently make sure the IP functionality, that works correctly after integrating to the corresponding bus architecture.

Kanna Shimizu et al.[11] proposed an early rule-based monitor to check PCI bus protocol. Oliveira and Alan J. Hu [12] proposed a high-level specification style that can generate a hardware monitor. Interface-monitor-based methodologies to watch the interface between a block and the rest of the system [13] [14]. The DUT is checked cycle by cycle during simulation to make sure the DUT obeys all these rules. Therefore, this approach is very efficient when we integrate several IPs. However, many errors may occur in real-time that the monitored-based approach often cannot find errors in simulation environment. The simulation-based verification IP is non-synthesizable code; they cannot be used in FPGA or chip-level. There are still need more efficient way verify the system, only violated rules cannot help designer to find errors rapidly. To overcome this problem, an AMBA AHB on-chip bus protocol checker with an efficient mechanism called HP Checker, is proposed in [17]. In recent years, for high-performance SoC requirements, the AMBA AXI on-chip bus protocol is defined [18] 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 benefits of using rule-based design include improving observability, reducing debug time, improving integration through correct usage checking, and improving communication through documentation. In the final purpose, increasing design quality while reducing the time-to-market and verification costs [19]. AMBA AXI protocol checking technique will be more and more important in the future.

1.1. AXI4 Protocol

AMBA AXI4 supports data transfers up to 256 beats and unaligned data transfers using byte strobes. In AMBA AXI4 system 16 masters and 16 slaves are interfaced. Each master and slave has their own 4 bit ID tags. AMBA AXI4 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 AXI4 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 AWCACHE and ARCACHE signalling details.

Each transaction is burst-based which 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.

The write operation process starts when the master sends an address and control information on the write address channel shown in figure 1. 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. The allowable responses are OKAY, EXOKAY, SLVERR, and DECERR. After the read address appears on the address bus, the data transfer occurs on the read data channel shown in figure 2. 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 allowable responses are OKAY, EXOKAY, SLVERR, and DECERR [20].

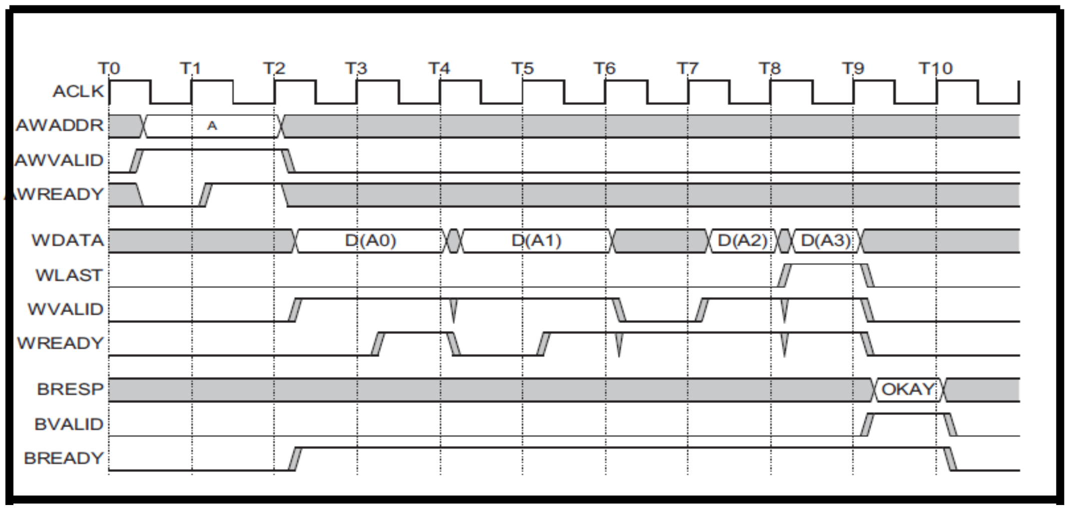


Figure 1. Write burst

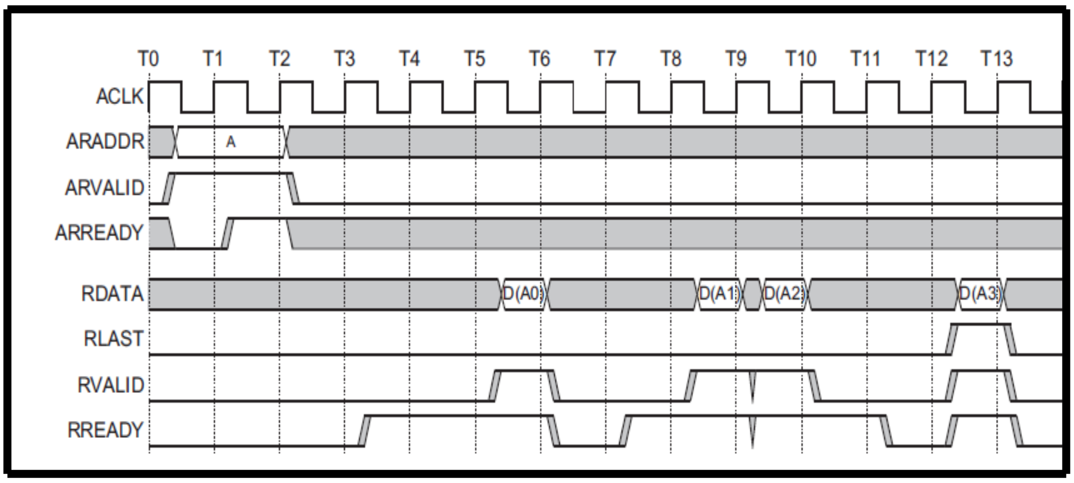


Figure 2. Read burst

The protocol supports 16 outstanding transactions, so each read and write transactions have ARID[3:0] and AWID [3:0] tags respectively. Once the read and write operation gets completed the module produces a RID[3:0] and BID[3:0] tags. If both the ID tags match, it indicates that the module has responded to right operation of ID tags. ID tags are needed for any operation because for each transaction concatenated input values are passed to module.

1.2. Comparison of AXI3 and AXI4

AMBA AXI3 protocol has separate address/control and data phases, but AXI4 has updated write response requirements and updated AWCACHE and ARCACHE signaling details. AMBA AXI4 protocol supports for burst lengths up to 256 beats and Quality of Service (QoS) signaling. AXI4 has additional information on Ordering requirements and details of optional user signaling. AXI3 has the ability to issue multiple outstanding addresses and out-of order transaction completion, but AXI4 has the ability of removal of locked transactions and write interleaving. One major updation seen in AXI4 is that, it includes information on the use of default signaling and discusses the interoperability of components which can’t be seen in AXI3.
In this paper features of AMBA AXI4 listed above are designed and verified. The rest of the paper is organized as follows: Section 2 discusses related work. Section 3 of this paper, discusses proposed work. In Section 4, simulation RESULTS are discussed. Future scope and concluding remarks are given in Section 5.

II. Related Work

In a SoC, it houses many components and electronic modules, to interconnect these a bus is necessary. There are many buses introduced in the due course some of them being AMBA [21] developed by ARM, CORE CONNECT developed by IBM, WISHBONE developed by Silicore Corporation, etc. Different buses have their own properties the designer selects the bus best suited for his application.
The AMBA bus was introduced by ARM Ltd in 1996 which is a registered trademark of ARM Ltd. Later advanced system bus (ASB) and advanced peripheral bus (APB) were released in 1995, AHB in 1999, and AXI in 2003[18]. AMBA bus finds application in wide area. The advantages of introducing Network-on-chip (NoC) within SoC such as quality of signal, dynamic routing, and communication links was discussed in [22].
To verify on-chip communication properties rule based synthesizable AMBA AXI protocol checker [23] is used**.**

III. Proposed Work

The work carried out in this paper is to verify whether any bus protocol violations occur between one master and one slave. To verify this first generates the transaction between master and slave according to AMBA AXI4 protocol. After the transactions had been created design the AXI checker and verifies the transactions.
The Figure 3 shows the AXI4 checker testbench.

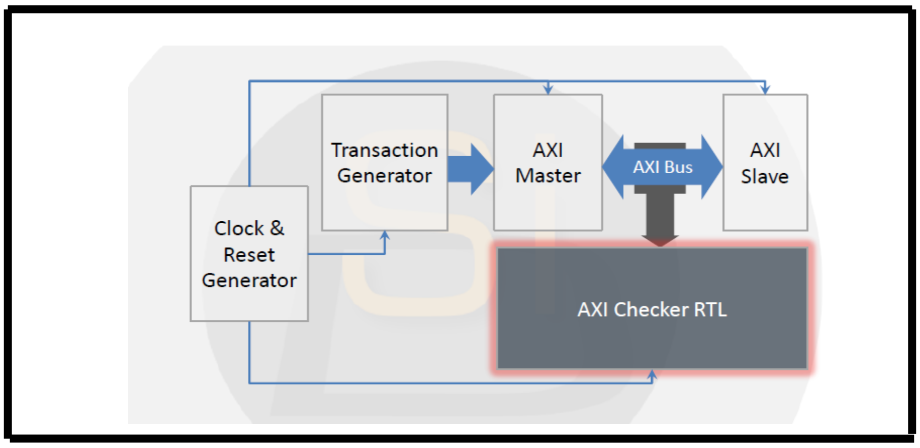


Figure 3. Test bench

The components of the test bench are clock and reset generator, AXI master, AXI slave, transaction generator and AXI checker. The AXI checker perform protocol check using internal data structures and module inputs. The output of the checker is the structured display messages. The clock and reset generator produce the clock and reset signals for the modules. Transaction generator generates random transactions, randomization for address, type (read or write) and control values (size, burst, prot, length etc). the AXI master takes the input from transaction generator and initiates it on the AXI bus. The master is fsm based. AXI slave interacts with AXI master to mimic AXI transactions. The slave also is fsm based.

3.1. Clock and Reset Generator

The clock and reset generator is a stimulus generator used for generating clock (ACLK) and reset (ARESETn). It consists of separate tasks and blocks for generating resets and clocks respectively. These signals will be inputs to all other modules in the test bench.

3.2. Transaction Generator

Transaction generator generates random transactions, randomization for address, type (read or write) and control values (size, burst, prot, length etc).

3.3. AXI Master

AXI master takes the input from transaction generator and initiates it on the AXI bus. This should be synthesizable so that it can be put onto FPGA. For the Master, the Slave is a location where it can write or reads data. The master is designed as a FSM with read and writes operations. The FSM has two sub FSMs with read and write happening independently due to separate read and write channels. Each operation is initiated with the asserting of the address signal to the slave with its validity. The master waits to receive the response from the slave to proceed further. After getting the response from the slave the master sends the data to be written to the specific address in the slave or reads data from the specific address in the slave. The read data channel conveys the data read and response back to the master.

A finite state machine or simply a state machine, is a mathematical model used to design computer programs and digital logic circuits. It is conceived as an abstract machine that can be in one of a finite number of states. The machine is in only one state at a time; the state it is in at any given time is called the current state. It can change from one state to another when initiated by a triggering event or condition, this called a transition. A particular FSM is defined by a list of the possible transition states from each current state, and the triggering condition for each transition

3.3.1. AXI master read address FSM

Figure 4 shows the read address FSM for AXI master. xfr is the input from the transaction generator indicating read transfer. control is read control signal values and val is random values from transaction generator. AWREADY is the write address ready. This signal indicates that the slave is ready to accept an address and associated control signals. If AWREADY value is high it remains in a IDLE state. When value of AWREADY is low it goes to VALID state.

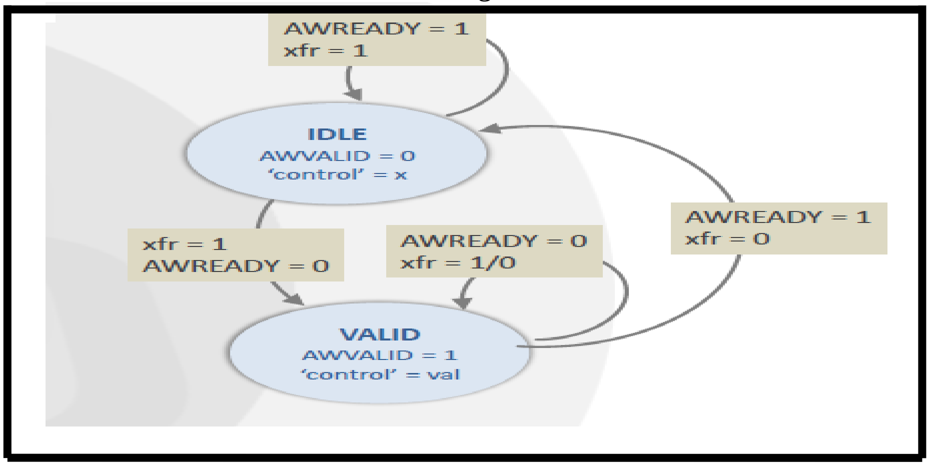


Figure 4. AXI master read address FSM

3.3.2 AXI master read data FSM

Figure 5.shows the read data FSM.

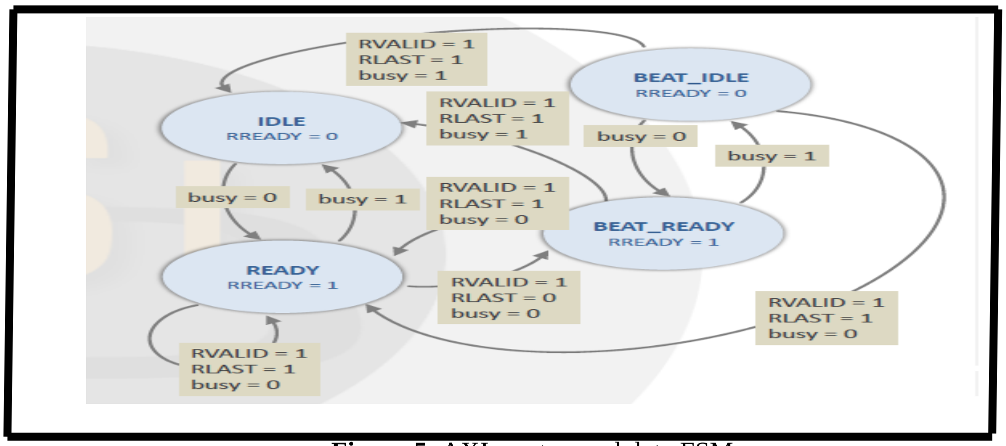


Figure 5. AXI master read data FSM

IDLE is when the master is busy and cannot accept the data beat. BEAT_IDLE is when there are more than one beat and master is busy and cannot accept the data beat. READY is when the master is not busy and can accept the first data beat from the slave. BEAT_READY is when there are more than one beat in the read transaction and master is ready to get the subsequent beats. The value of busy signal is high it remains in either IDLE or BEAT_IDLE state. When busy signal is low the master is ready to accept data so that the next state will be either READY or BEAT_READY.

3.4. AXI Slave

AXI slave component is interact with AXI master to mimic AXI transactions. When both write and read transfers are simultaneously requested, the read request is given more priority than the write request. This should be synthesizable. Implement parameterized memory spaces that are RAM model in one address space for incremental and wrapping burst and FIFO model for fixed bursts. Slave is based on FSM.

3.5. AXI Checker

Figure 6 shows the AXI checker structure.

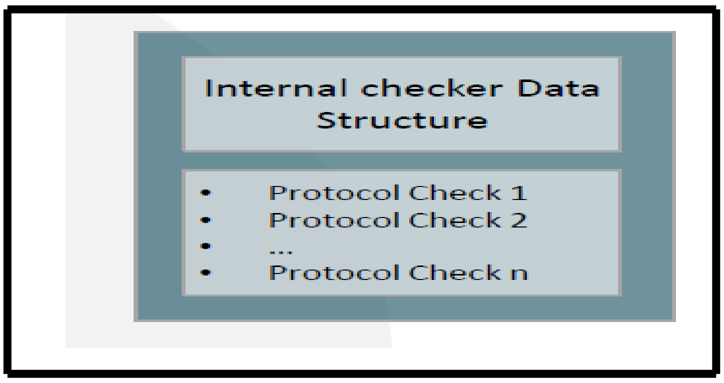


Figure 6. AXI checker

Checker Internal Data Structure includes intelligent CAM /Stack based on transaction IDS (Read & Write) , Exclusive transactions tracking structure and WSTRB calculation functions. In Protocol checks uses the internal data structures and module inputs. Structured Display messages are produced.

IV. SIMULATION RESULTS

The AXI4 Protocol Checker core is designed to monitor AXI interfaces. When attached to an interface, it actively checks for protocol violations and provides an indication of which violation occurred. The checker verifies the protocols in the internal data structure of the AXI4 checker and produces the error messages, when error occurs. The proposed AXI4 checker is a synthesizable hardware module.
Simulation is being carried out on ICARUS VERILOG tool, using Verilog as programming language. The waveforms are visible in GTKWave viewer.

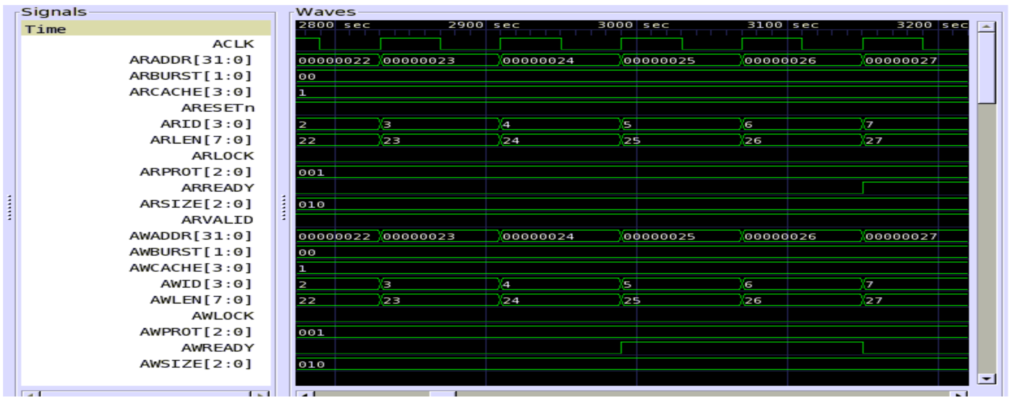
4.1. Simulation Inputs

Here the fixed type of the burst is passed to module. Internal_lock value is 0, burst size is 2, internal_burst type value is 0, internal_cache value is 1 and internal_prot value is 1,for both read and write operations, which indicate that the burst is of fixed type. For write and read operation data passed to modules are 3462EF78.

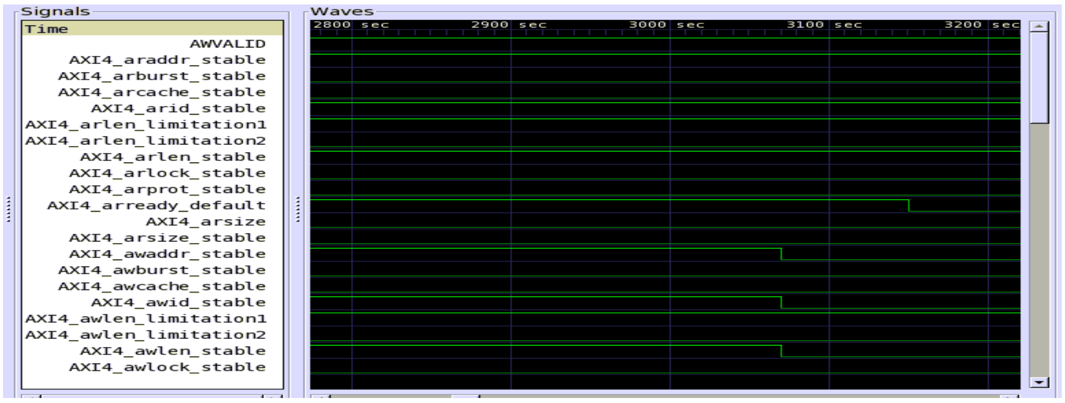
4.2. Simulation Outputs

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 BRESP[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 error_flag, warning_flag and AXI4_error are generated by the AXI4 checker which gives the information about error in the transactions.

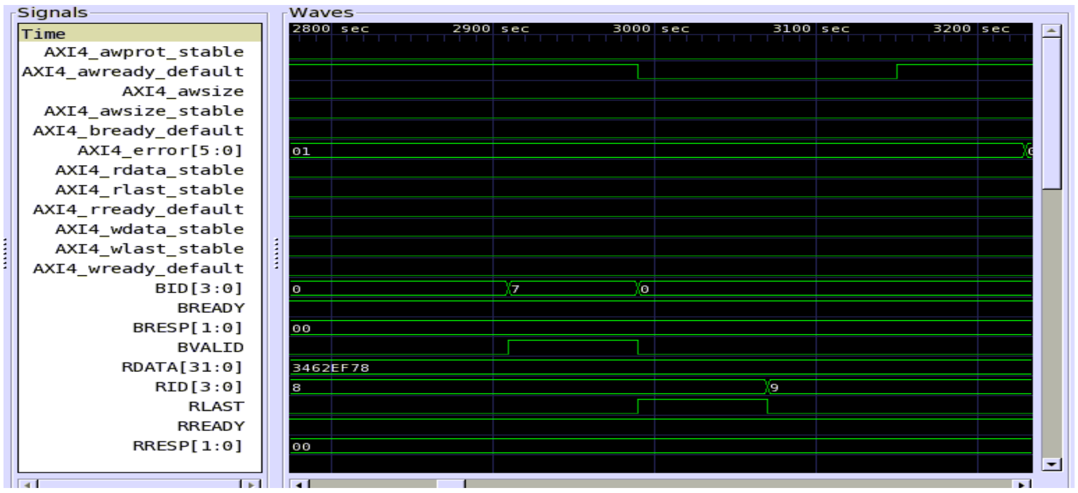
The warning flag, error flag and the AXI4 error signal shows the errors and warnings in the AXI4 bus. The warning flag is high when the variation in the default value of the AXI signals. The error flag is high whether any errors occurs in the read and write channels. Figure 7 shows the simulation results.



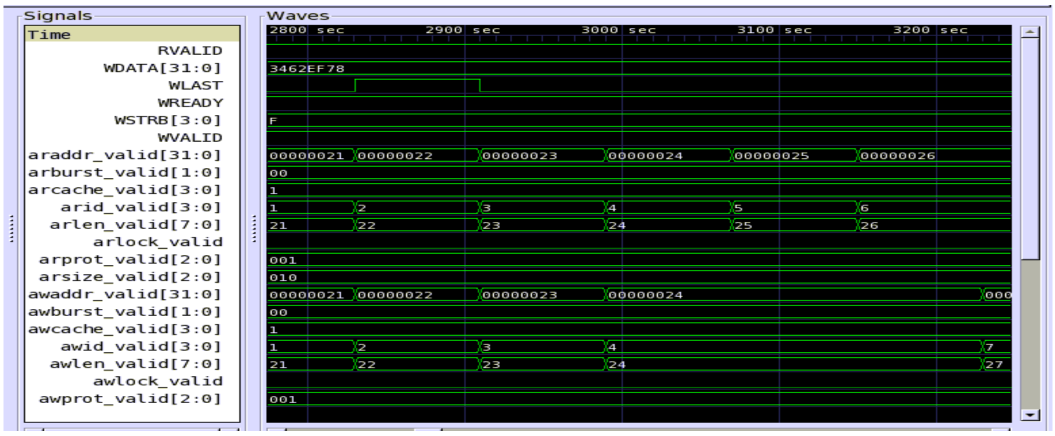
(A)



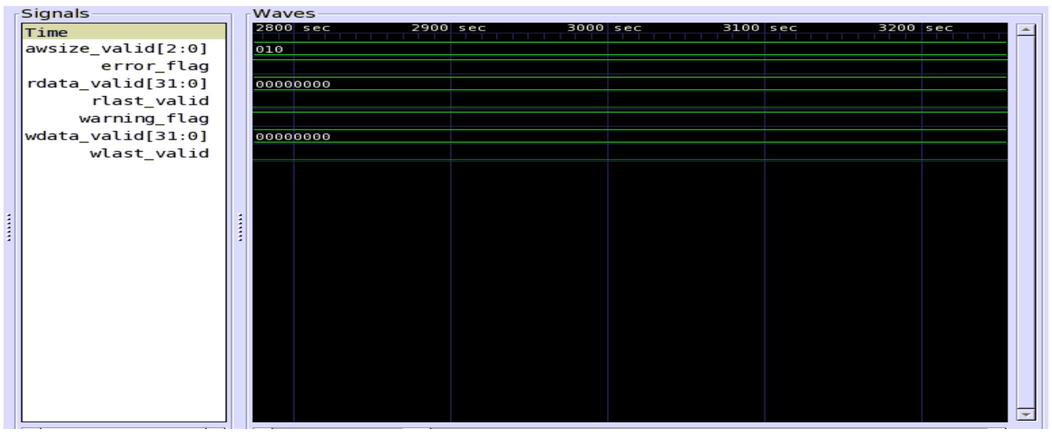
(B)



(C)



(D)



(E)

Figure 7. (A), (B), (C), (D), (E) Simulation result for AXI4 checker

V. Conclusion And Future Scope

5.1 Conclusion

AMBA AXI4 is a plug and play IP protocol released by ARM, defines both bus specification and a technology independent methodology for designing, implementing and testing customized high-integration embedded interfaces. The data to be read or written to the slave is assumed to be given by the master and is read or written to a particular address location of slave. The proposed AXI4 on-chip bus protocol checker has rules that provide AXI master, slave, and default slave issue. The proposed verification mechanism is used for improving verification ability. The checker verifies the transactions AXI master and slave. The errors and warnings of the transactions were showed in the simulation results.

5.2 Future Scope

The AMBA AXI4 has limitations with respect to the burst data and beats of information to be transferred. The burst must not cross the 4k boundary. Bursts longer than 16 beats are only supported for the INCR burst type. Both WRAP and FIXED burst types remain constrained to a maximum burst length of 16 beats. These are the drawbacks of AMBA AXI4 system which need to be overcome.

References

[1]. S. Pasricha, N. Dutt, On-Chip Communication Architectures: System on Chip Interconnect, Morgan Kaufmann, 2008.

[2]. E. M. Clarke, E. A. Emerson, and A. P. Sistla, "Automatic verification of finite-state concurrent systems using temporal logic specifications," ACM Transactions on Programming Languages and Systems (TOPLAS), vol. 8, pp. 244-263, 1986.
[3]. P. Chauhan, E. M. Clark, Y. Lu, and D. Wang, "Verifying IP-core based System-On-Chip designs," In Proceedings of the Twelfth Annual IEEE International ASIC/SOC Conference, pp. 27-31, Sept. 1999.
[4]. A. Roychoudhury, T. Mitra, and S. R. Karri, "Using formal techniques to debug the AMBA System-on-Chip bus protocol," In Proceedings of the IEEE/ACM Design, Automation, and Test in Europe Conference & Exhibition (DATE'03), 2003.
[5]. L. Ivanov and R. Nunna, "Specification and formal verification of interconnect bus protocols," In Proceedings of the 43rd IEEE Midwest Symposium on Circuits and Systems (MW-CAS'00), Vol. 1, Aug. 2000.
[6]. M. Kaufmann, A. Martin, and C. Pixley, "Design constraints in symbolic model checking," In Proceedings of the 10th International Conference on Computer Aided Verification, Springer-Verlag, 1998.
[7]. K. L. McMillan, "Symbolic Model Checking," Kluwer Academic Publishers, 1993.
[8]. I. Ltd., ''Sugar formal property language reference manual.'' [Online]. Available: http://www.haifa.il.ibm.com/projects/verification/sugar/
[9]. S. Ltd., OpenVera LRM 2.0. [Online]. Available: http://www.openvera.com.
[10]. Property Specification Language Reference Manual. [Online].Available: www.eda.org/vfv/docs/PSL-v1.1.pdf.
[11]. Kanna, Shimizu, David L. Dill and Alan J. Hu. "A monitor-based formal specification of PCI". Proceeding of the 3rd International Conference of Formal Methods in Computer-Aided Design, Nov. 2000.
[12]. Marcio T. Oliveira, Alan J. Hu. "High level specification and design: High-Level specification and automatic generation of IP interface monitors". Proceedings of the 39th conference on Design automation, June 2002.
[13]. M. S. Jahanpour, E. Cerny. "Compositional verification of an ATM switch module using interface recognizer/suppliers (IRS)". International High-Level Design, Validation, and Test Workshop, pp. 71-76. 2000.
[14]. M. Kaufmann, A. Martin, C. Pixley. "Design constraints in symbolic model checking"10th International Conference on Computer-Aided Verification, pp. 477-487. LNCS 1427. Springer, 1998.
[15]. ARM Ltd., "AMBA 3 AXI Protocol Checker: user guide," ARM DUI 0305B, ARM Ltd., 2006.
[16]. Synopsys Inc., "Using the DesignWare Verification Models for the AMBA 3 AXI Protocol," Version 5.60a, Synopsys Inc., April 2009.
[17]. 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.
[18]. ARM Ltd., "AMBA AXI Protocol Specification V1.0", ARM IHI 0022B, ARM Ltd., 2004.
[19]. Harry Foster , David Lacey , Adam Krolnik, Assertion-Based Design, Kluwer Academic Publishers, Norwell, MA, 2003.
[20]. ARM(2010), AMBA Protocol Specification 4.0, www.arm.com.
[21]. ARM(1997), AMBA Specification , www.arm.com.
[22]. Bruce Mathewson "The Evolution of SOC Interconnect and How NOC Fits Within It", IEEE transl, DAC,2010, California, USA,Vol 6, pp. 312-313, June 2010
[23]. 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

AUTHORS

Prasanth M working as Assistant Professor in the Department of Electronics and Communication Engineering at Nehru College of Engineering and Research Centre, Pampady, Thiruvilawamala, Kerala, India, affiliated to University of Calicut. He received B.Tech degree in Electronics & Communication Engineering from Lourdes Matha College of Science and Technology, Kerala, India affiliated to Kerala University and M.E in Applied Electronics from Sardar Raja College of Engineering, Alangulam, Tamil Nadu.

Juhi Raj is doing M.Tech in VLSI Design at Nehru College of Engineering & Research Centre, Thrissur, Kerala under the University of Calicut. She received B.Tech degree from Mahatma Gandhi University in 2011.