International Journal of
Human Computations & Intelligence
Vol. 02, Issue 06, Nov/Dec 2023

RESEARCH ARTICLE OPEN ACCESS

System Verilog Verification Techniques for AXI4 Slave Devices: A Practical Approach

Harish T L . Chandrashekhar M C . Eshwarappa M N

Department of Electronics and Communication Engineering
Sri Siddhartha Institute of Technology, Tumkur, India.

DOI: **10.5281/zenodo.10444427**
Received: 18 October 2023 / Revised: 16 November 2023 / Accepted: 01 December 2023
©Milestone Research Publications, Part of CLOCKSS archiving

Abstract – In this study, the primary emphasis is placed on validating the critical aspects of the advanced extensible interface (AXI). When verifying the memory transactions of AX I, it is necessary to check all five channels: read address, read data, write response, write address, and write response. In this particular piece of work, a technique that is based on Verification Intellectual Property cores (VIP) is employed to carry out the verification Process. The complete testing environment is modelled in the VIP design using system verilog, and the read and write transactions from the same and different memory locations have been confirmed with the quantitative values of Busy Count, Valid Count, and its Bus Utilization. In addition, the Busy Count, Valid Count, and its Bus Utilization have been compared. One of the main qualities checked for in this article is the connectedness of the system throughout the writing and reading cycles.

Index Terms – Write and Read transactions, AXI protocol, Verification IP, Bus utilization, Coverage mode analysis

I. Introduction

The fast rise of CMOS technology and computer-aided design has led to the use of a huge number of IP (Intellectual Property Cores) in the construction of sophisticated digital designs [1]. Today, the System on Chip (SOC) architecture is becoming more popular and is being utilized extensively in a wide variety of applications [2]. This is made possible with the assistance of the integration of these IP cores. Additionally, every design of SOC makes use of bus protocols in order to facilitate data transfer and synchronization [4-6]. since of this, the reusability of a high number of IP cores in the complicated design makes the functional verification process so important (i.e., since it includes 70% of the time span, as opposed to the design process, which involves 30% of the time span). Verification engineers

MILESTONE
RESEARCH.IN
OPEN ACCESS
Electronic ISSN: **2583-5696**

devised a way for validating the functioning of the chip by making use of an inbuilt verification environment that they referred to as Verification IP (VIP) [7-10]. This was done in order to circumvent the difficulties and significant amount of time that was required for the process. APB stands for "Advanced Peripheral Bus," AHB stands for "Advanced high performance Bus," and AXI is for "Advanced Extensible Interface."

These three bus protocols are often employed in today's latest SOCs. When compared to these other three bus protocols, the AXI bus protocols provide superior performance while only using a minimal amount of electricity. Because of this, the AMBA AXI bus is used throughout the internal architecture of every single SOC design. The failure mode analysis and fault robustness testing are normally carried out for the design as part of the functional verification process. A verification environment-based technique that makes use of System Verilog in coverage enabled mode has been suggested as a means of overcoming the slag that occurs throughout the functional verification process [11-17]. The following two operations are carried out in this verification mode environment: i) Proper test-cases are developed for the DUV (Device under Verification); ii) Determine the number of test-cases to examine and cover all of the functionality of the design. During the process of verification, the coverage enabled mode makes it simple to find faults in the design by determining the extent to which the code has been executed and verifying that the functional behavior of the design has been validated for each and every test case. The remaining portions of the paper are structured as follows: AMBA Bus Architecture, Proposed Verification Environment, Verification Plan, and Results and Discussion are the sections that are included in this report.

II. PROPOSED WORK

Verilog is going to be used to accomplish communication between one master and one slave, and then System Verilog will be used to validate the design. This is the kind of work that will be done as part of this project.

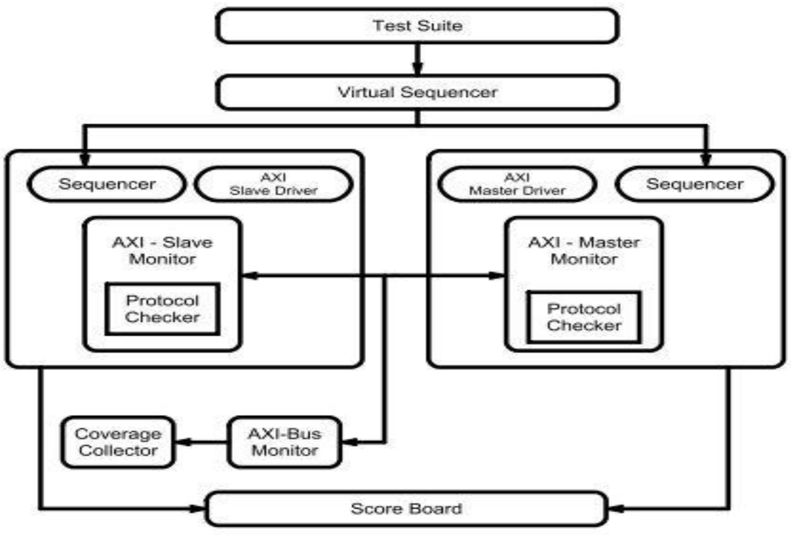

Fig. 1: Verification Environment

- **Design of AXI Protocol AMBA**

A maximum of 256 data transfers may be completed in one single burst using an AXI4 slave that has been configured with an operating frequency of 100 MHz, which results in each clock cycle lasting 10 nanoseconds. As can be seen in Figure 1, the AMBA AXI4 component in the system comprises of both a master and a slave. There are five distinct channels that may be used to communicate between the AXI master and the AXI slave. These channels are referred to as the read address channel, the write response channel, the read data channel, and the write data channel.

Every transmission in the AXI protocol is completed via a process called the hand shake. When it comes to transferring control and data information, every channel employs the same VALID/READY handshake. This system for controlling flow in two directions allows both the master and the slave to exercise control over the rate at which data and control information is sent. When either the data or the control information is accessible, the source will emit the VALID signal to signify this fact. The READY signal, which indicates that the destination is prepared to receive the data or control information, is generated by the destination. The VALID and READY signals must both be high in order for the transfer operation to take place. On both the master and the slave interfaces, there must not be any combinatorial routes between the input signals and the output signals.

Verifying the DUV in code coverage mode may be accomplished using the verification environment, which is a conventional way. The suggested verification environment has the following components: i) Test-suite, ii) Virtual sequencer, iii) Sequencer, iii) Sequencer, iii) Driver, v) AXI-Monitor, vi) Protocol checker, vii) Bus Monitor, viii) Coverage Collector, and xi) Scoreboard.

- **Test-Suite**

The test-suite is made up of all of the test-cases that are used to validate the functioning of the DUV when it is operating in a restricted random mode. The test case may either be of the limited random mode or of the direct random test-case kind. The test is carried out in a backwards fashion on the structure, and then the results are examined. In most cases, it will include a stimulus set for the purpose of running the design. Virtual Sequencer In most cases, the sequencer will produce random sequences in order to validate the fundamental characteristics of the design. The sequence that was formed is referred to as the top level sequence, and it is then divided into the base level sequences and primitive sequences that correspond to it. Therefore, the functioning of each subcomponent is tested by the virtual sequencer by driving or forcing it with the set of sequences.

- **Sequencer**

A component known as the sequencer is responsible for producing a collection of sequences. There are two categories, active and passive, for the sequences that are formed. In contrast, the passive sequence does not drive the signal in any way, which is driven by the active sequence. The criteria of functionality checks are taken into account while the sequencer aligns and creates the sequence in accordance with those requirements. Driver The driver is the component that interacts with or causes all other components to react in accordance to the fed stimuli (that is, it is the component that, to put it more simply, drives the input to the relevant blocks).

- **Monitor**

The coverage collector receives the results of the monitor's observations of all the data transactions carried out by each and every component. It is equipped with a TLM connection for conducting analyses as well as a virtual interface for managing the points of DUV signals. Checker for the protocol In this module, the DUV functionality is examined, along with the interface compliance and the temporal behavior of the inputs. Lastly, the outputs are analyzed for their outputs. In addition, the responses are observed for each and every possible combination of the input sequences. Bus monitor The bus monitor is responsible for monitoring the proper functionality of each individual component as well as connecting the master and slave monitors. In the last step, the transactions are compiled into a database using the coverage collector.

- **Coverage Collector**

A coverage collector is responsible for gathering all of the code coverage information, such as whether a branch, statement, or FSM in the DUV is covered by a particular test case or not. During the functional verification, the verification engineers should always strive for a coverage rate of one hundred percent. Within the AMBA AXI bus design, the components that have been specified above may function in either the master or the slave mode. The separate modules are each modeled in system Verilog, and then they are combined to form the topmost main module.

III. RESULTS AND DISCUSSION

This section describes the simulation results of the AMBA AXI protocol for the five distinct test-scenarios and the performance evaluation using the quality metrics for the data metrics. Also included are the results of the simulation of the AMBA AXI protocol. First, the various modules of AXI Master/Slave are modelled using System Verilog, and then they are combined with the modelled test environment so that the verification can be carried out. This is done so that the simulation analysis can be carried out. To evaluate the functionality of the DUV inside the limited mode within the verification environment, it is combined with the sequencer, the interface, the driver, and the monitor. Following the completion of the top module's connection with DUV, the simulation is carried out with the Mentor Graphics QuestaSim® software. The test case for this section has been created to accommodate the following five distinct scenarios: i) the read phase; ii) the write phase; iii) the read and write from the same address; iv) the read and write from a different address; and lastly, v) the overlapping of transactions.

- **Verification of Read Phase**

During this phase, the Master/Slave configuration mode is used in order to validate the read data transactions that are part of the AMBA AXI protocol. During the transactions that take place during the ready cycle, the signals ARVALID, ARREADY, RVALID, RREADY, RLAST, and RDATA as well as ARSIZE are monitored and checked. The functionality check in this instance focuses largely on two different modes of operation, which are referred to as i) Read Address and ii) Read Response channel. The read address channel will get the address at the high state values of ARVALID and ARREADY for each positive edge of the clock. This occurs whenever the clock advances. Similarly, once a certain amount of time has passed, the response will be instantiated to high mode, and both RVALID and RREADY will have high values corresponding to them. High values of the RLAST variable indicate the completion of the most recent transaction during the read process. In read mode, the values of ARSIZE and ARLEN are measured, and the results provide an indication of the number of transactions that took place. In order to validate the read phase, a test case is developed and run via simulation in order to validate the read operation's capabilities.

The operation of the read mode makes use of two channels, as was previously described, along with the measurement of the quality metrics for the read address, read channel data, and channel response. Figure 2 provides a graphical representation in the form of a waveform for the whole of the read phase's operations.

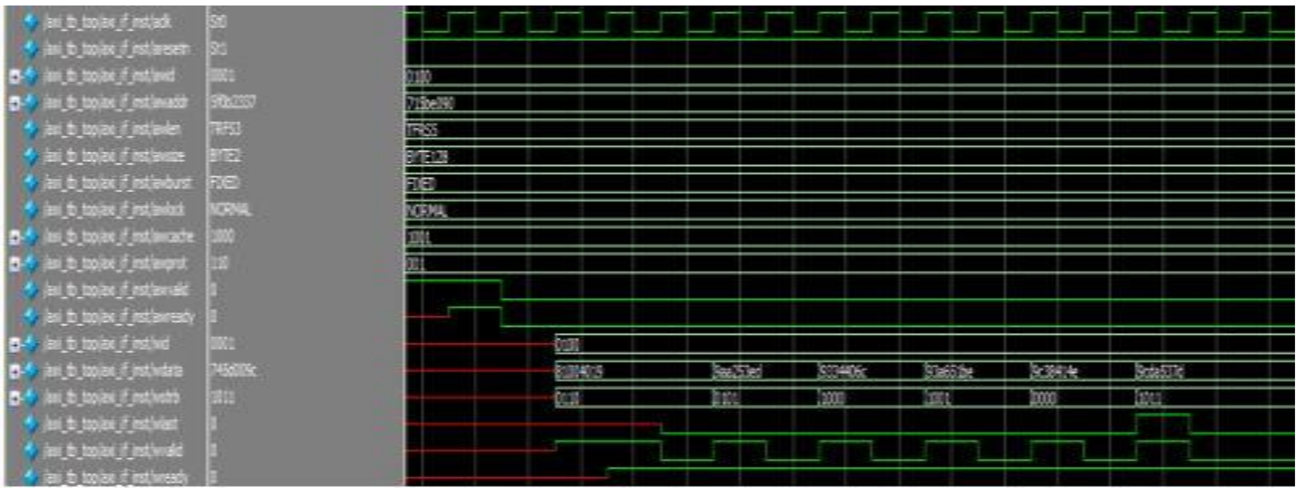
/axi_tb_top/axi_if_inst/aclk
/axi_tb_top/axi_if_inst/aresetn
/axi_tb_top/axi_if_inst/awid
/axi_tb_top/axi_if_inst/awaddr
/axi_tb_top/axi_if_inst/awlen
/axi_tb_top/axi_if_inst/awsize
/axi_tb_top/axi_if_inst/awburst
/axi_tb_top/axi_if_inst/awlock
/axi_tb_top/axi_if_inst/awcache
/axi_tb_top/axi_if_inst/awprot
/axi_tb_top/axi_if_inst/awvalid
/axi_tb_top/axi_if_inst/awready
/axi_tb_top/axi_if_inst/wid
/axi_tb_top/axi_if_inst/wdata
/axi_tb_top/axi_if_inst/wstrb
/axi_tb_top/axi_if_inst/wlast
/axi_tb_top/axi_if_inst/wvalid
/axi_tb_top/axi_if_inst/wready

Fig. 2: Read and Write Phase at the Same Address Response

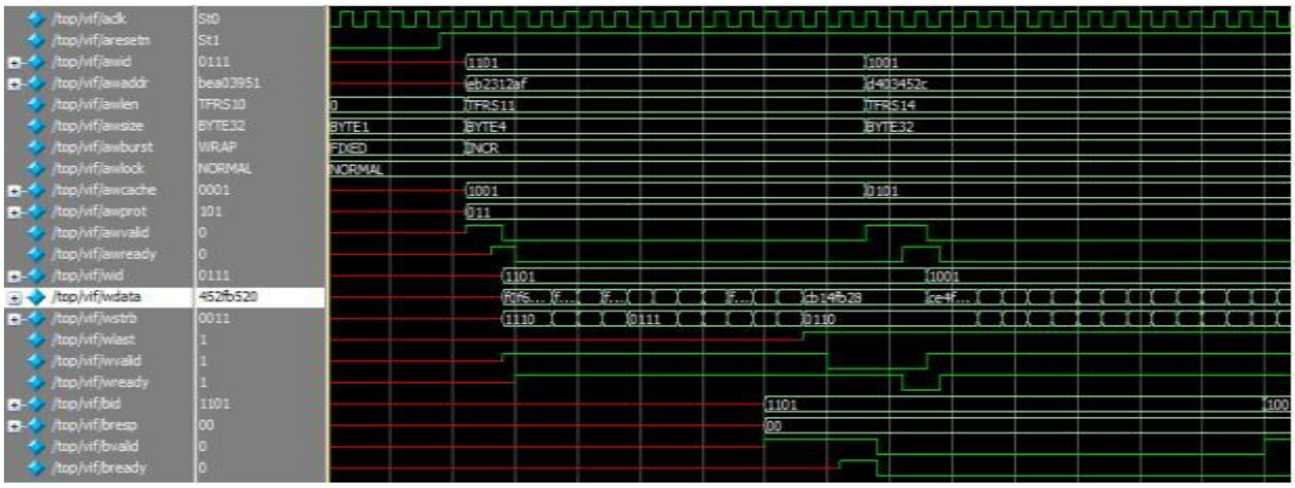

Fig. 3: Transactions that happened at the same time in the simulation

From the modeling results, the use of the bus in the different test-case situations can be compared. When the read and write processes are done at the same time, the bus is used more quickly than when the read and write phases are done separately. For each step of reading and writing, the bus usage factor is only about 85%. Then, when the read and write operations are done together, the bus is used up to 95% of the time.

IV. CONCLUSION

In this study paper, System Verilog HDL is used to create and develop the AMBA-AXI bus system. Then, the Mentor Graphics EDA tool is used to run a program to test the functions. Here, the bus is checked for each of the five test-case situations, and code coverage is also done. The speed measures are also used to figure out how efficient the bus is in the suggested test setting.

REFERENCES

1. Swaminathan, K., Lakshminarayanan, G., & Ko, S. B. (2014). Design and verification of an efficient WISHBONE-based network interface for network on chip. *Computers & Electrical Engineering*, *40*(6), 1838-1857.
2. Ahmed, S. T., Ashwini, S., Divya, C., Shetty, M., Anderi, P., & Singh, A. K. (2018). A hybrid and optimized resource scheduling technique using map reduce for larger instruction sets. *International Journal of Engineering & Technology*, *7*(2.33), 843-846.
3. Su, A. P., Kuo, J., Lee, K. J., Huang, J., Jian, G. A., Chien, C. A., ... & Chen, C. H. (2011, April). Multi-core software/hardware co-debug platform with ARM CoreSight™, on-chip test architecture and AXI/AHB bus monitor. In *Proceedings of 2011 International Symposium on VLSI Design, Automation and Test* (pp. 1-6). IEEE.
4. abdelkrim zitouni, B. A., & Tourki, R. (2010, March). Design and implementation of network interface compatible OCP For packet based NOC. In *5th International Conference on Design & Technology of Integrated Systems in Nanoscale Era* (pp. 1-8). IEEE.
5. Mahesh, G., & Sakthivel, S. M. (2015, April). Verification of memory transactions in AXI protocol using system verilog approach. In *2015 International Conference on Communications and Signal Processing (ICCSP)* (pp. 0860-0864). IEEE.
6. Singh, S. P., Bhoj, S., Balasubramanian, D., Nagda, T., Bhatia, D., & Balsara, P. (2006). Generic network interfaces for plug and play NoC based architecture. In *Reconfigurable Computing: Architectures and Applications: Second International Workshop, ARC 2006, Delft, The Netherlands, March 1-3, 2006, Revised Selected Papers 2* (pp. 287-298). Springer Berlin Heidelberg.
7. Chen, C. H., Ju, J. C., & Huang, J. (2010, November). A synthesizable AXI protocol checker for SoC integration. In *2010 International SoC Design Conference* (pp. 103-106). IEEE.
8. Lai, Y. L., Yang, S. W., Sheu, M. H., Hwang, Y. T., Tang, H. Y., & Huang, P. Z. (2006, June). A high-speed network interface design for packet-based NoC. In *2006 International Conference on Communications, Circuits and Systems* (Vol. 4, pp. 2667-2671). IEEE.
9. Shao, J., & Davis, B. T. (2007, February). A burst scheduling access reordering mechanism. In *2007 IEEE 13th International Symposium on High Performance Computer Architecture* (pp. 285-294). IEEE.
10. Fattah, M., Manian, A., Rahimi, A., & Mohammadi, S. (2010, July). A high throughput low power FIFO used for GALS NoC buffers. In *2010 IEEE Computer Society Annual Symposium on VLSI* (pp. 333-338). IEEE.
11. Ragaventhiran, J., Vigneshwaran, P., Kodabagi, M. M., Ahmed, S. T., Ramadoss, P., & Megantoro, P. (2022). An unsupervised malware detection system for windows based system call sequences. *Malaysian Journal of Computer Science*, 79-92.
12. Shrivastavastava, A., Tomar, G. S., & Kalra, K. K. (2010, November). Efficient Design and Performance analysis for AMBA bus Architecture based System-on-Chip. In *2010 International Conference on Computational Intelligence and Communication Networks* (pp. 656-660). IEEE.
13. Wang, C. W., Lai, C. S., Wu, C. F., Hwang, S. A., & Lin, Y. H. (2008, April). On-chip interconnection design and SoC integration with OCP. In *2008 IEEE International Symposium on VLSI Design, Automation and Test (VLSI-DAT)* (pp. 25-28). IEEE.
14. Tran, A. T., & Baas, B. M. (2011, October). RoShaQ: High-performance on-chip router with shared queues. In *2011 IEEE 29th International Conference on Computer Design (ICCD)* (pp. 232-238). IEEE.

15. Mahesh, G., & Sakthivel, S. M. (2015, April). Verification of memory transactions in AXI protocol using system verilog approach. In *2015 International Conference on Communications and Signal Processing (ICCSP)* (pp. 0860-0864). IEEE.
16. Mahesh, G., & Sakthivel, S. M. (2015, March). Functional verification of the Axi2OCP bridge using system verilog and effective bus utilization calculation for AMBA AXI 3.0 protocol. In *2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)* (pp. 1-5). IEEE.
17. Basha, S. M., & Ahmed, S. T. (2023). Real Time Systems: Challenges and Applications.