



ISSN : 2321-9416

Indo - American Journal of Mechanical Engineering



www.iajme.org

Email : iajme.editor@gmail.com or editor@iajme.org



VERILOG-BASED IMPLEMENTATION OF THE AMBA AXI PROTOCOL

Mr.Pitta Sankara Rao¹, Thottempudi Kavya Sree²

*1 Assistant Professor, Department of ECE, Malla Reddy College of Engineering for Women.,
Maisammaguda., Medchal., TS, India*

*2, B.Tech ECE (21RG1A0455),
Malla Reddy College of Engineering for Women., Maisammaguda., Medchal., TS, India*

Abstract - The key characteristics of the advanced extensible interface (AXI) are the primary emphasis of this study. When checking the AXE I memory transactions, be sure to check the write address, write data, write response, read address, and read data on each of the five channels. The verification process in this work is carried out using a technique based on verification intellectual property cores (VIP). Using the quantitative values of Busy Count, Valid Count, and its Bus Utilisation, the VIP design validates read and write transactions from the same and different memory locations in the test environment that is modelled using system verilog. One of the basic aspects validated in this work is the system connection throughout write and read cycles.

KEY WORDS: WRITE AND READ TRANSACTIONS, AXI PROTOCOL, VERIFICATION IP, BUS UTILIZATION, AND COVERAGE MODE ANALYSIS.

INTRODUCTION

A great deal of Intellectual Property Cores (IP) are used in complicated digital designs as a result of the fast development of CMOS technology and Computer Aided Design [1]. The incorporation of these IP cores has contributed to the increasing popularity and extensive usage of the SOC (System on Chip) architecture in many applications nowadays [2]. For data transfer and synchronisation, every SOC architecture also employs bus protocols [4-6]. Therefore, the functional verification process is so important (i.e., it takes 70% of the time, compared to 30%

for the design phase) because of the reusability of a high number of IP cores in the complicated design. Verification engineers devised a strategy for checking the chip's operation using an integrated verification environment called Verification IP (VIP) to circumvent this challenge and lengthy time span [7-10]. In most cases, the bus protocols that are used by contemporary SOC are

ABA, APB, and AXI stand for "Advanced Peripheral Bus," "Advanced High Performance Bus," and



"Advanced Extensible Interface," respectively. The AXI bus protocol outperforms the other two while using very little power when compared side by side. Therefore, the AMBA AXI bus is an integral part of the internal architecture of any single-board computer design. It is common practice to conduct the design's fault resilience and failure mode analysis during functional verification. A approach based on the verification environment, using System Verilog in coverage enabled mode, is suggested to circumvent the functional verification process snag. Two tasks are performed in this verification mode environment: (i) defining the test cases for the device under verification (DUV) and (ii) determining the number of test cases to analyse in order to cover all the design's functions. By detecting the code coverage and evaluating the functional behaviour of the design for all test cases, the coverage enabled mode makes it easy to find design issues during verification.

EXISTING METHOD

AMBA-AXI BUS ARCHITECTURE

The Advanced Microcontroller Bus Architecture (AMBA) is a protocol for on-chip bus communication developed by The arm. The document details the requirements for the on-chip interconnect that link the functional blocks, peripherals, and CPU. These

days, SOCs and FPGAs are best suited to be made to use the AXI bus protocol architecture. This is because of its big memory access, high level of backward compatibility with peripherals, high bandwidth capabilities, and ability to interface with sophisticated bridges. In addition, the AXI bus design has burst mode access, unaligned data transactions, discrete data channels, and distinct read/write data phases. Here we have a look at the AXI protocol's bus design, which is crucial for data transfer and synchronisation. There are two stages to the establishment of data transactions in this bus design, as

- i) There is a read-only phase in the bus, and a write-only phase. Following the master's explanations of the relevant operational stages,
- settings for slaves.

READ TRANSACTION PHASE

While synchronising and communicating data, the read data response (R) and read address (AR) signals are used in the read transaction phase. The read channel access and valid address (AR) are the first components of a read transaction.

sent from the master to the slave in order to initiate the transactions, and the slave in turn makes the address available for communication by responding with the signal AR ready. The master gets the RVALID and read channel data signals after verifying the address upon the arrival of AR ready. The master then initiates read transactions and sets up channel connection with a control signal and a valid RR ready. Figur.1 is a schematic depicting the whole design and functioning of the read transaction.

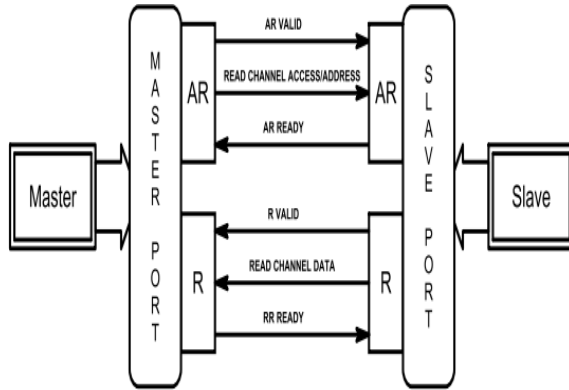


Figure 1: AXI Read Transactions

Write Transaction Phase

First, there is the write address channel; second, there is the write data channel; and third, there is the write response channel, all of which are used for write transactions. When the master is in write address channel mode, AW control signals initiate communication.

The master provides the slave the address and channel control information that is both valid and writeable. A valid AW is immediately ready by the slave. Similar to how the master initiates transactions in Write data channel mode with a WVLAID and Write channel data, the slave acknowledges with a valid WREADY signal thereafter. In a manner similar to that of data and address mode, the master initiates a transaction in write response channel mode by use of the control signals BVALID. The slave initiates communication by sending a valid BREADY signal when it receives the control signal. In figure 2, we can see how the write transactions work.

One common way to check whether the DUV is working properly in code coverage mode is to use the verification environment. The suggested verification environment includes the following parts: I Test-suite,

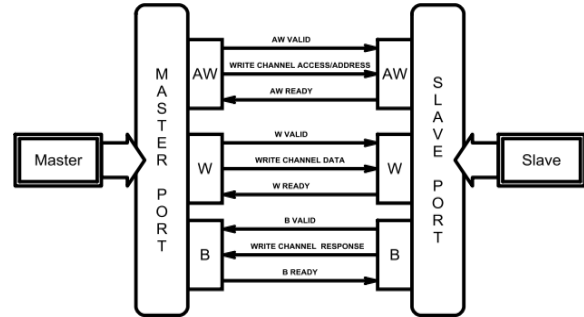


Figure 2: AXI Write transactions

II Virtual sequencer, III Sequencer, IV Driver, V AXI-Monitor, VI Protocol checker, VII Bus Monitor, VIII Coverage Collector, and XI Scoreboard.

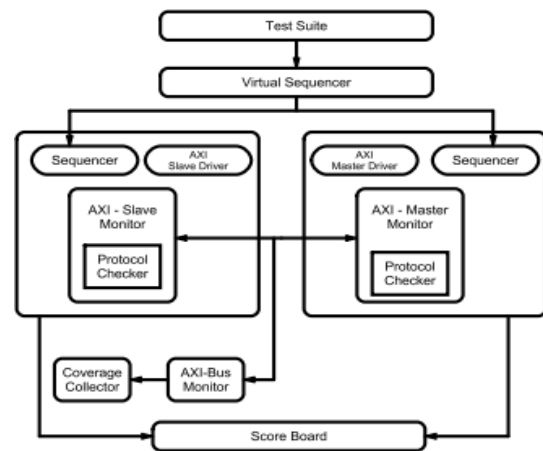


Figure - 3 Verification Environment

PROPOSED VERIFICATION ENVIRONMENT



Test-Suite: In a restricted random mode, the test-suite includes all the test-cases that verify the functioning and certain essential aspects of the DUV. The test-case may be either a direct random test-case or a limited random mode. We run the test on the design and analyse the results regressively. As a rule, it has a collection of stimuli for executing the design.

Virtual Sequencer: In most cases, the sequencer will produce random sequences to check the design's critical features. The produced sequence is divided into two parts: the top level sequence and the base level and primitive sequences that relate to it. Consequently, the virtual sequencer checks whether each subcomponent works by driving or pushing it to follow the set of sequences.

Sequencer: The component known as a sequencer is used to create sets of sequences. Active and passive sequences are the two main categories of the created ones. The signal is driven by the active sequence, not the passive. The sequencer creates and aligns the sequence based on the functionality check criteria.

Driver: In order for other parts of the system to react to the input stimuli, the driver must be present. This component "drives" the input to the appropriate blocks.

Monitor: The system keeps tabs on all component data interactions and relays that information to the coverage collector. To process the DUV signal points, it has a virtual interface and a TLM port for analysis.

Protocol checker: This section checks the DUV's functionality and, using the inputs' temporal behaviour, the interface's compliance. Additionally, we keep an eye on the results for every possible permutation of the input sequences.

Bus monitor: The bus monitor links the master and slave monitors and checks the functional functioning of individual components. In the end, the coverage collector stores all of the transactions in a database.

Coverage Collector: Coverage collector gathers information on code coverage, including whether or not the requested test-case covers the branch, statement, and FSM in the DUV. During functional verification, verification engineers often strive for 100% coverage.



VERIFICATION PLAN

The coverage requirements are driven to be met by the verification plan, which informs that the property must be confirmed. In order to accomplish the coverage and functionality check, a solid strategy includes quantifiable metrics and successful test-cases. You may find out what needs verifying and how to check it in the strategy for verification. I. System connection during read and write cycles; ii. Transaction routing; and iii. Data integrity are the attributes that need to be confirmed in this AMBA AXI verification. Using the appropriate acknowledgment signals, the system integration is monitored in the system connection checking test. Then, while planning the route, the efficiency and utilisation of the buses check the data occupancy structure. Lastly, at the end of each step, we make note of the data integrity for each transaction and run the functionality check pass.

The covering of the following situations is focused in the verifications plan of this AXI protocol:

Read phase

- **Write Phase**
- **Write & Read Phase from same address**
- **Write & Read Phase from different address**
- **Overlapped transactions**

To ensure the functioning, test cases are created for each situation and simulations are run. To measure the bus's efficiency, we look at its Valid Count, Busy Count, and Bus Utilisation factor. If the bus is not correctly validated in the relevant test-cases, then the test-cases are adjusted to achieve the property checks. Additionally, the verification process includes analysing the DUV using a code coverage based verification mode. If the analysis does not reach 100% coverage, the test-case is modified to cover all the code components of the design.



RESULTS & DISCUSSIONS

Here we go over the AMBA AXI protocol's performance evaluation based on quality metrics for data metrics, as well as the outcomes of the simulations for the five distinct test-scenarios. As a first step in conducting the simulation analysis, the AXI Master/Slave modules are modelled in System Verilog and then merged with the simulated test environment. To evaluate its performance in limited mode, the DUV is connected with the verification environment's sequencer, interface, driver, and monitor. The Xilinx ise is used to conduct the simulation once the top module has been integrated with DUV. Here, we have the test-case prepared for five distinct scenarios: (i) the reading phase, (ii) the writing phase, (iii) reading and writing from the same address, (iv) reading and writing from separate addresses, and (v) overlapping transactions.

VERIFICATION OF READ PHASE

Using the AMBA AXI protocol's Master/Slave configuration mode, the read data transactions are checked at this step. While processing transactions in the ready

cycle, the following signals are checked and validated: ARVALID, ARREADY, RVALID, RREADY, RLAST, RDATA, and ARSIZE. The primary components of the functionality check in this case are two

i) read the address, and ii) read the response, are the two channels of operation. At the high state values of ARVALID and ARREADY, the read address channel will retrieve the address on every positive edge of the clock. The response will also go into high mode once some time has passed, with signals on RVALID and RREADY set to high values. When RLAST is high, it means the read operation has completed its final transaction. The number of read-only transactions is indicated by the measured values of ARSIZE and ARLEN. In order to ensure that the read phase is functioning properly, a test-case is created and then run via a simulation. As mentioned earlier, the read mode operates with two channels, which are used to monitor read address quality metrics, read channel data, and channel response. The whole read phase process is graphically shown as a waveform in figure 4.

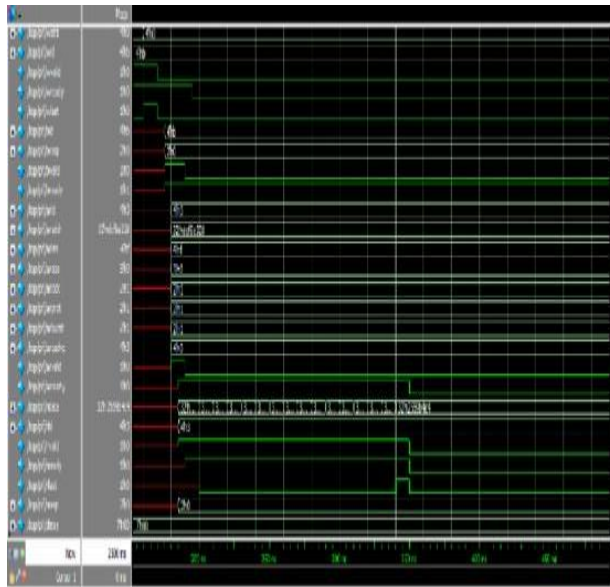


Figure 4: Read Phase Simulation Response

VERIFICATION OF WRITE PHASE

Master/Slave configuration AWID, AWADDR, AWLEN, AWSIZE, and AWVALID AND AWREADY signal verification is part of the AMBA AXI protocol's Write Phase, which is similar to the Read Phase. The three main operations of the Write Phase are as follows: I. Writing the address channel; II. Writing the channel data; and III. Writing the response channel. In order to get the valid write address with an acknowledgment AWID, the signals WVALID and WREADY go into a high logic state for every positive clock cycle. Similarly, when BVALID and BREADY transition high, the write response operation will occur. Each transaction causes the control signal AWLEN to increase from its initial value of 0000, while the signal AWSIZE shows the size of each transaction. Figure 5 is a visual representation of the Write Transactions.

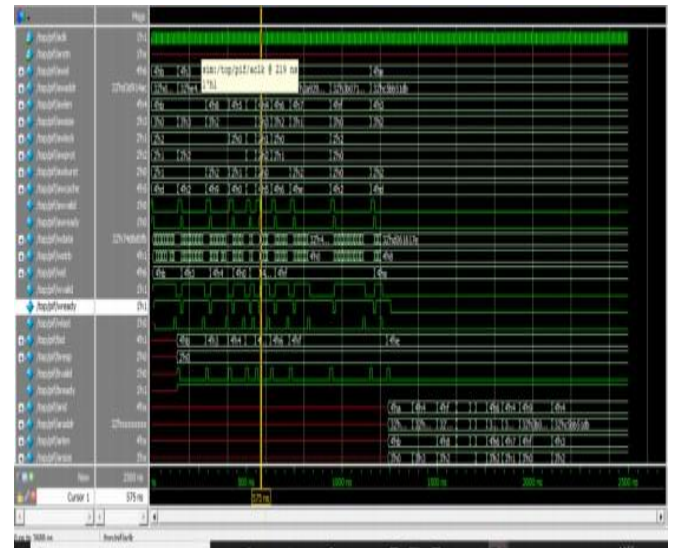


Figure 5: Write Phase Simulation Response

The coverage report is 10 write and read compare transaction compare to previous we are more signals are covering and also 83.3% will be cover the transactions.

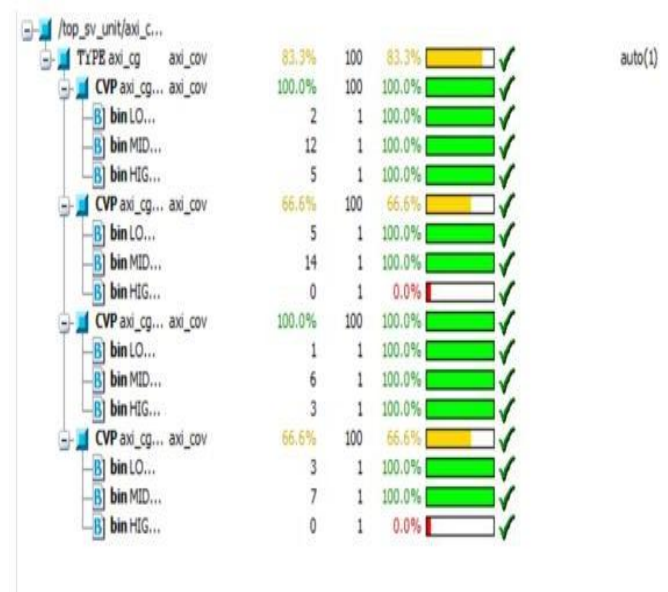


Fig 6: coverage report of 10 write read transaction

Task	Min	Max	Current	Progress	Status
CPUPackager	0.00	100.00	100.00		✓
bin\bin	0	1	1		✓
bin\lib	0	1	1		✓
bin\obj	0	1	1		✓
CPUPackager	0.00	100.00	100.00		✓
bin\bin	1	1	1		✓
bin\lib	2	1	1		✓
bin\obj	0	1	1		✓
CPUPackager	0.00	100.00	100.00		✓
bin\bin	2	1	1		✓
bin\lib	2	1	1		✓
bin\obj	2	1	1		✓
CPUPackager	0.00	100.00	100.00		✓
bin\bin	1	1	1		✓
bin\lib	1	1	1		✓
bin\obj	2	1	1		✓

The coverage report is 16 write and read compare transaction compare to previous we are more signals are covering and also 75.0% will be cover the transactions.



CONCLUSION

9



REFERENCES

- [1] Advanced Microcontroller Bus Architecture Specification. (1997) [Online]. Available: <http://www.arm.com>
- [2] Open Core Protocol Specification. (2006) [Online]. Available: <http://www.ocpip.org/home>
- [3] A. T. Tran and B. M. Bass, “RoShaQ: High-performance on-chip router with shared queues,” in Proc. IEEE 29th Int. Conf. Comput. Design, Oct. 2011, pp. 232–238.
- [4] J. Shao and B. T. Davis, “A burst scheduling access reordering mechanism,” in Proc. IEEE 13th Int. Symp. High Perform. Comput. Archit., Feb. 2007, pp. 285–294.
- [5] J. Pang, L. Yang, L. Shi, T. Zhang, D. Wang, and C. Hou, “A priority expression based burst scheduling of memory reordering access,” in Proc. Int. Conf. Embedded Comput. Syst., Archit., Model., Simul., Jul. 2008, pp. 203–209.
- [6] X. Xiao and J. J. Lee, “A true $O(1)$ parallel deadlock detection algorithm for single-unit resource systems and its hardware implementation,” IEEE Trans. Parallel Distrib. Syst., vol. 21, no. 1, pp. 4–19, Jan. 2010.
- [7] A. Silberschatz, P. B. Galvin, and G. Gagen, Operating System Concepts, 7th ed. New York, USA: Wiley, 1993.
- [8] T. S. Cummins, “Method and apparatus for detecting a bus deadlock in an electronic system,” U.S. Patent 6 292 910, Sep. 18, 2001.
- [9] Technical Reference Manual of PrimeCell AXI Configurable Interconnect (PL300), ARM, Cambridge, U.K., 2010.
- [10] K. Lahiri, A. Raghunathan, and G. Lakshminarayana, “The LOTTERYBUS on-chip communication architecture,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 14, no. 6, pp. 596–608, Jun. 2006.
- [11] K. Sekar, K. Lahiri, A. Raghunathan, and S. Dey, “Dynamically configurable bus topologies for high-performance on-chip communication,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 16, no. 10, pp. 1413–1426, Oct. 2008.



- [12] H.-W. Wang, C.-S. Lai, C.-F. Wu, S.-A. Hwang, and Y.-H. Lin, “Onchip Interconnection Design and SoC Integration with OCP,” in Proc. IEEE Int. Symp. VLSI Design, Autom., Test, Apr. 2008, pp. 25–28.
- [13] N. Y.-C. Chang, Y.-Z. Liao, and T.-S. Chang, “Analysis of shared-link AXI,” IET Comput. Digit. Tech., vol. 3, no. 4, pp. 373–383, Jul. 2009.
- [14] O. Ogawa, S. Bayon de Noyer, P. Chauvet, K. Shinohara, Y. Watanabe, H. Niizuma, T. Sasaki, and Y. Takai, “A practical approach for bus architecture optimization at transaction level,” in Proc. Design, Autom., Test Europe Conf. Exhibit., 2003, pp. 176–181.
- [15] Synopsys Design Compiler. (2010) [Online]. Available: <http://www.synopsys.com/Tools/Implementation/RTLSynthesis/Pages/default.aspx>.