Verification of an AXI4 slave device using System Verilog

[1]Harish T L., [2]Dr. Chandrashekhar M C

[2]Professor
Department of Electronics and Communication Engineering
Sri Siddhartha Institute of Technology
Tumkur-05

Abstract: **Because of its accessibility, comprehensive documentation, and lack of license costs, AMBA protocols have become the de facto standard for 32-bit embedded processors. High-performance, high-frequency system topologies are now feasible thanks to the AMBA AXI 4 protocol. It is great for systems that need plenty of bandwidth and little latency, and it lets you run at high frequencies without resorting to elaborate bridges. The existing AHB and APB interfaces may be used with it, and it allows for a variety of connection topologies to be built. The Verification Environment for this project is constructed using System Verilog code. This project's goal is to verify the AMBA-based design of the AXI4 Slave Interface.**

Keywords- **Advanced Microcontroller Bus Architecture (AMBA), Advanced Peripheral Bus (APB), AMBA High performance Bus (AHB), Advanced Extensible Interface (AXI).**

1.INTRODUCTION

The abbreviation for "Advanced Extensible Interface" is "AXI." It was created by the ARM (Advanced RISC Machines) corporation as a component of the Advanced Microcontroller Bus Architecture, also known as AMBA. It is a communication protocol that takes place on the chip itself. Assisting in the creation of high-performance and high-frequency system designs is the AMBA AXI protocol. Systems that need both high bandwidth and low latency might benefit from the AXI protocol. It allows for high-frequency functioning without the need for a complex bridge. It conforms to the standards of a wide variety of parts' interfaces. Memory controllers that have a high initial access latency are a good candidate for using the AXI protocol. It offers flexibility in the manner in which interconnect topologies may be put into practice. It maintains compatibility with the already-established AHB and APB interfaces.

A distinct address/control phase and data phase are two of the most important aspects of the AXI protocol. Additionally, the AXI protocol provides support for unaligned data transfers by making use of byte strobes. Transactions occur in short bursts, and just the first start address is issued. It offers Direct Memory Access (DMA) at a cheap cost and is equipped with independent data channels for reading and writing. It allows for the issuance of many unique addresses simultaneously. It allows for the completion of transactions in a non-ordered manner. It makes the adding of register stages to enable timing closure simple and straightforward.

Peripheral Bridge, APB, and other memory interfaces are among the most common types of AHB and ASB slaves. However, certain low bandwidth peripherals will be resting on the APB side, which is the side of the A Bridge that is routinely engaged for linking the low bandwidth designs with a high-performance system bus. This is seen in Figure 1.

2. LITERATURE REVIEW

Connecting and managing functional blocks in system-on-a-chip (SoC) designs is the focus of the Advanced Microcontroller Bus Architecture (AMBA). This specification is an open standard that is implemented as an on-chip interface. It makes it easier to construct designs for multi-processor systems that have a high number of controllers and peripherals. AMBA's purview, despite the fact that it was originally intended for use with microcontrollers, has expanded significantly since the organization's foundation. Application processors used in today's smartphones are only one example of the ASIC and SoC components that make use of the Advanced Message-Passing Architecture (AMBA).

In 1996, ARM introduced AMBA to the public for the first time. Both the Advanced System Bus (ASB) and the Advanced Peripheral Bus (APB) were the original AMBA buses [1, 2]. The AMBA High-performance Bus (AHB) was introduced by ARM in AMBA 2, and it uses a single clock-edge protocol[3-5]. ARM released AMBA 3 in 2003; it included AXI for enhanced performance connection and the Advanced Trace Bus (ATB) as a component of the CoreSight on-chip debug and trace solution [6&7]. AMBA 3 was the third iteration of the Advanced Micro Devices Architecture (AMBA). Beginning in 2010, the AMBA 4 specifications were released, the first of which was AMBA 4 AXI4. Released in 2011, AMBA 4 ACE aimed to reduce congestion by redesigning the high-speed transport layer and extending system-wide coherency [8-10].

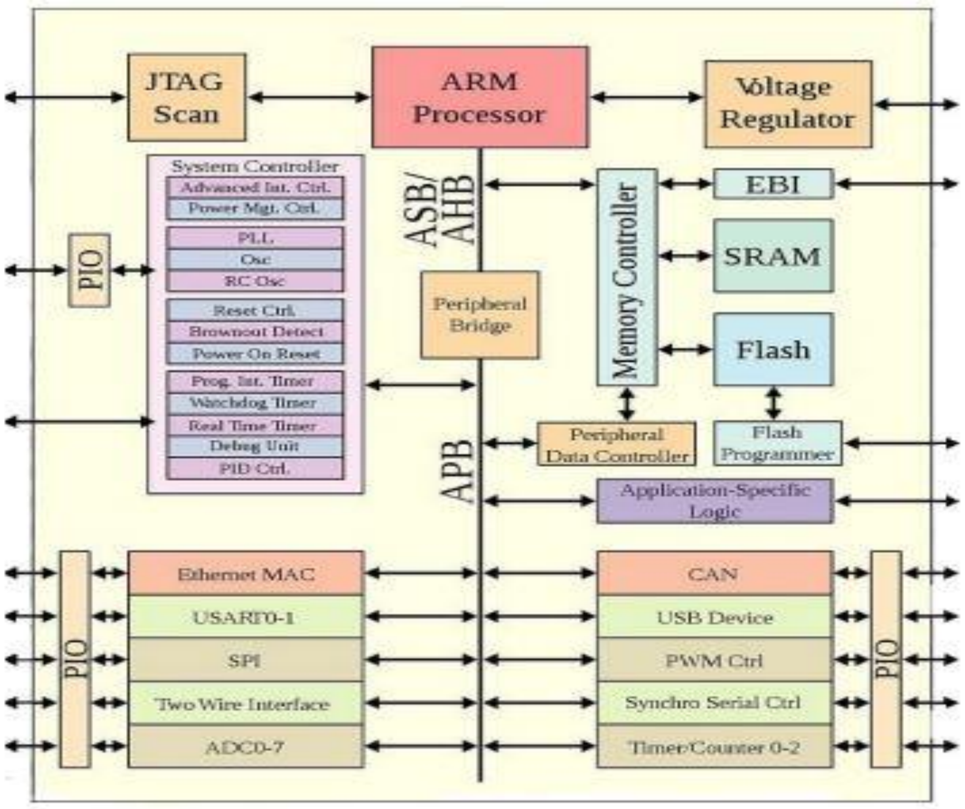

Figure 1 AMBA system bus in SoC

Complete testing of interface IP blocks and system-on-chip (SoC) designs is achievable with the AMBA protocol family because of the metric-driven verification of protocol compliance it provides. The AMBA advanced extensible interface 4 (AXI4) standard improves upon the AMBA AXI3 specification in several ways, including: the addition of component compatibility details; the ability to support burst durations of up to 256 beats; the modification of the write response criterion; the removal of locked transactions. Interfacing between 16 masters and 16 slaves is supported by the AMBA AXI4 protocol system. Verilog-HDL is used to create an actualization of the design [11-13].

3. Research Methodology

The operating frequency of an AMBA AXI4 slave is intended to be 100 MHz, giving each clock cycle a length of 10 nanoseconds, and it is capable of supporting a maximum of 256 data transfers during a single burst. Figure 1 shows that the AMBA AXI4 system has two halves, a master and a slave. There are five distinct channels that may be used to communicate between the AXI master and the AXI slave. The readaddress channel, the writeresponse channel, the read data channel, and the write data channel are the names given to these communication pathways.

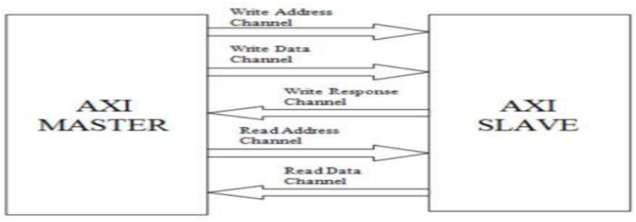

Figure 2 Block diagram of the system.

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 bidirectional flow control device allows both the master and the slave to regulate the rate at which data and control signals are sent. The VALID signal is sent by the source when either the data or the control information may be accessed. The destination sends out the READY signal when it is ready to receive the data or control information. The VALID and READY signals must both be high in order for the transfer operation to take place. There cannot be any possible combinations of paths taken by the input and output signals on either the master or the slave side of the interface.
In Fig-3, the whole of the verification environment that is used to validate the AXI transactions is shown. It is made up of a generator module that is responsible for reading the test cases in order to meet the verification requirements. In this study, we focus on two separate test cases that verify read and write transactions at the same address and at different addresses. The mailbox is being utilized as a synchronization channel since both test cases are being driven to the bus functional model. When it comes to getting the generated transactions to the AXI interface, the Bus Functional Model is crucial. The verification environment is constructed

using UVM and may be adapted to any Device under Verification (DUV) in line with the verification methods and goals. The verification environment, also known as Verification Intellectual Property (VIP), is increasingly appearing as a built-in verifying option with all System on Chips (SOCs).

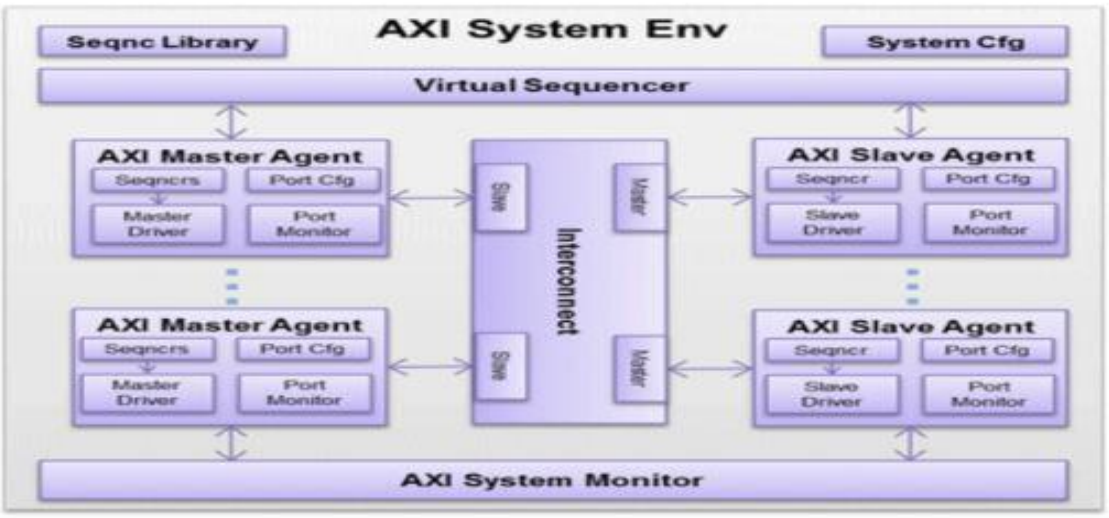

Figure 3. verification environment

4. RESULTS AND DISCUSSIONS

Verification of read architecture

Throughout the course of this read cycle verification, each and every one of the read architecture signals will have their respective transcations checked. The read architecture is comprised of two channels, which are referred to as the read address channels and the read response channels respectively. When the global clock reaches a positive edge in its cycle, the read address channel will begin the process of address fetching by setting both the ARVALID and ARREADY signals to their high states. The read response is similarly converted to high mode for each positive edge of the RVALID and RREADY signals after a delay that defines a gap. The most recent transaction in the RDATAsignal is indicated by the value of RLAST. When compared to the write architecture, ARSIZE and ARLEN are identical. Figure 4 depicts the waveform that was generated by the simulation.

The primary need for this particular test case is to validate only the read phase. Leaving out the values of the write phase signals and concentrating solely on those relating to the read operation, we next proceed to compute the parameters while ignoring the write phase signal values. The read phase, which uses the same two-channel setup as the previous phase, is largely concerned with verifying the operational parameters that, when adhered to, provide reliable measurements of actual bus use. These are the signals that are being tested in this particular test scenario. The reading step is broken up into three distinct channels: read address, read data, and read response. In order to ensure that the read phase is accurate, it is essential to do a thorough cross-check of the handshaking of signals for each channel. Only after this step can the read phase be verified as being accurate.

Verification of write architecture

Each transaction is examined for correctness with respect to the three write signals (write address, write data, and write response) at this stage of the verification process. These signals flip at each rising edge of the world clock, where the address is then stored in the channel. Each write address has its own unique identifier, or AWID, which must match the WID that was used to write the data. When the clock is toggling on positive edges and the enable logic value in WVALID and WREADY is high, the write data channel will acknowledge. In a similar fashion, the write response will take place when both the BVALID and BREADY signals are in their high states. In this case, the size of the AWLEN signal is four bits [0:3], and it creates a variety of transactions ranging from one to sixteen. If the AWLEN value is 0100 when that generation procedure is carried out, the result will be 0101 transactions, which indicates that the number of transactions will be increased by one. This is represented quite well in the waveform is shown in Figure 5. It can be seen from the waveform that the AWSIZE value gives an indication of the magnitude of each transaction. Figure 5's waveform clearly illustrates that all of the signal toggle counts have been simulated and verified for the whole write architecture. This can be seen clearly in the waveform.

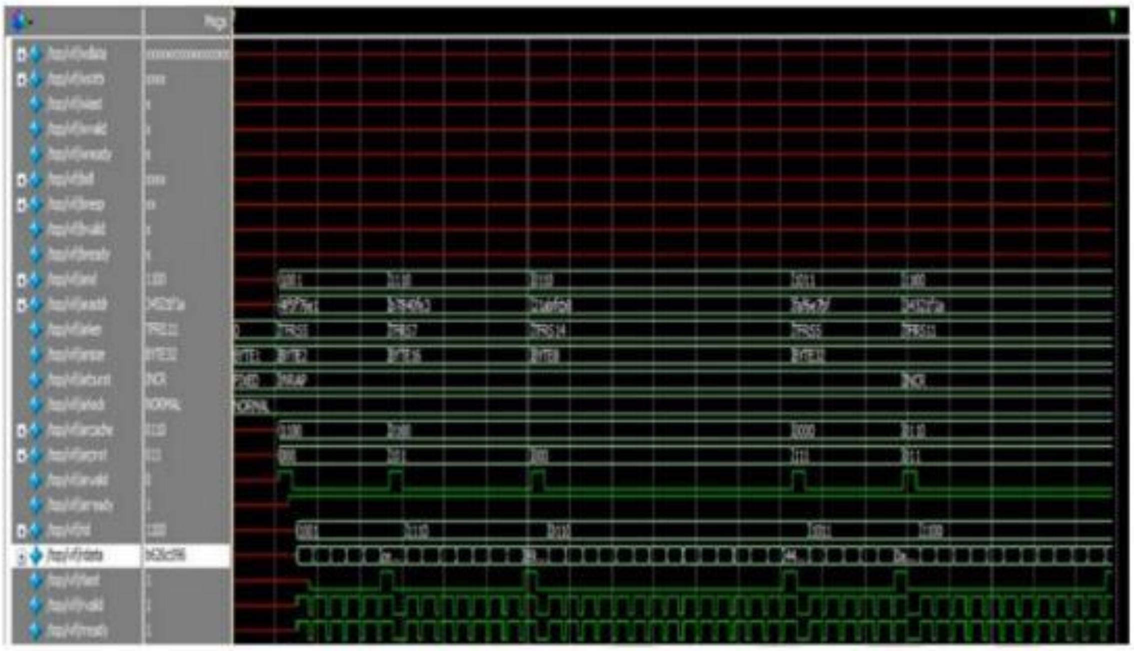
Figure 4 read cycle waveform

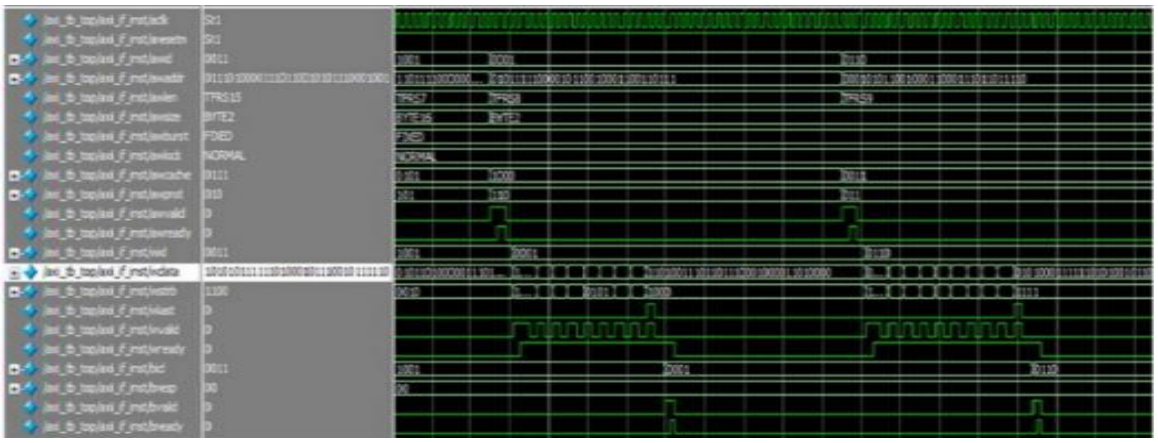
Figure 5 AXI Write Cycle Response

AWADDR, AWVALID, AWREADY, and WDATA, in addition to their write address and write data signals, are the signals that are being tested and confirmed in this test case. Additionally, the signals WLAST, WVALID, and WREADY, as well as BRESP, BVALID, and BREADY, are checked for each and every transaction. With the help of this test case, the values for the parameters VALID COUNT, BUSY COUNT, and BUS UTILIZATION are computed realistically, and the bus utilization is shown in the form of percentage figures.

CONCLUSION

SystemVerilog is used throughout this article to provide an efficient verification environment for the AXI bus. The code coverage mode analysis is used in order to do verification and analysis on the AXI protocol verification as well as the signals that are utilized in each channel. The most significant benefit of using this method of verification is the use of the pseudo randomcoverage driven verification, which results in a shorter time to market and is usable for the verification of complicated designs via UVM. In future, we will have a test case ready to verify read and write phases simultaneously from the same location and from different geographical outposts.

REFERENCES:

1. “AMBA Peripheral Bus Controller Data Sheet” Copyright © 1996 Advanced RISC Machines Ltd (ARM).
2. Ramagundam, S.; Dept. of Computer Sci., Troy Univ., Montgomery, AL, USA ; Das, S.R. ; Morton, S. ; Biswas, S.N. , “Design and implementation of high-performance master/slave memory controller with microcontroller bus architecture”, Instrumentation and Measurement Technology Conference (I2MTC) Proceedings, 2014 IEEE International, 12-15 May 2014.
3. “AMBA™ Specification (Rev 2.0)” 13th May 1999-A, First release, Copyright ARM Limited 1999.
4. “Soo-Yun Hwang; Dept. of Comput. Eng., ChungNam Nat. Univ., Taejon, South Korea; Kyoung-Sun Jhang “An improved implementation method of AHB Bus Matrix”, SOC Conference, 2005. Proceedings. IEEE International, 25-28 Sept. 2005

5. Hu Yueli; Key Lab. of Adv. Display & Syst. Application., Shanghai Univ., Shanghai, China ; Yang Ben “Building an AMBA AHB Compliant Memory Controller”, Measuring Technology and Mechatronics Automation (ICMTMA), 2011 Third International Conference, 6-7 Jan. 2011.
6. “AMBA AXI Specification (AR500-DA-10008)” 16 June, 2003-A, First release, Copyright ARM Limited 2003
7. Paunikar, A.; Sch. of Electron. Eng., VIT Univ., Vellore, India ; Gavankar, R.; Umarikar, N. ; Sivasankaran, K. , “Design and implementation of area efficient, low power AMBA 3-APB Bridge for SoC” , Green Computing Communication and Electrical Engineering (ICGCCEE), 2014 International Conference, 6-8 March 2014
8. “AMBA AXI 4 and ACE Protocol Specification” 28 October 2011 D Non-Confidential First release of AMBA AXI 4 and ACE Protocol Specification.
9. Xu Yang ; Harbin Inst. of Technol., Harbin ; Zhang Qing-li ; Fu Fang-fa ; Yu Ming-yan, “NISAR: An AXI compliant on-chip NI architecture offering transaction reordering processing” ASIC, 2007. ASICON '07. 7th International Conference, 22-25 Oct. 2007.
10. Manjula, R.B. ; Manvi, S.S. ; Kaunds, P. “Data transactions on system-on-chip bus using AXI4 protocol” Recent Advancements in Electrical, Electronics and Control Engineering (ICONRAEeCE), 2011 International Conference, 15-17 Dec. 2011.
11. “Verilog-A Language Reference Manual Analog Extensions to Verilog HDL”, Version 1.0, Open Verilog International August 1, 1996
12. “Verilog-AMS Language Reference Manual”, Release 2.3.1, Accellera Systems Initiative , 06-2009
13. “Verilog-AMS Language Reference Manual”. Release 2.4, Accellera Systems Initiativ, 06-2014.