Indian Journal of Science and Technology, Vol 9(48), DOI: 10.17485/ijst/2016/v9i48/101912, December 2016
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Development of VIP for AMBA AXI-4.0 Protocol

Renduchinthala H H S S Prasad* and Ch Santhi Rani

Department of ECE, DMS SVH College of Engineering, Manginapudi Beach Road, Machilipatnam, Krishna - 521002, Andhra Pradesh, India; rhhssprasad@gmail.com, santhirani.ece@gmail.com

Abstract

Objective: With the recent trend of Intellectual Property (IP) based designs in developing a System On Chip (SoC), it became a complicated task to verify an System On Chip (SoC). In this situation, a Verification Intellectual Property (VIP) is very helpful. Many SoC Projects are following this trend of design flow, which is IP Core based and verification flow, which is VIP based, to reduce the time to release to market and to accelerate the Verification process for SoC. This work focuses on developing a Verification Intellectual Property (VIP) for Advanced eXtensible Interface (AXI)-4.0 Protocol, which belongs to family of Advanced Micro controller Bus Architecture (AMBA). **Method:** The VIP is developed by using latest Methodology named Universal Verification Methodology (UVM). **Findings:** The functionality of all the five channels of AXI is verified successfully. Multiple outstanding transaction and out of order transaction completion scenarios are also verified by using Questa sim tool. The functional coverage of 100% is obtained. **Applications:** The AXI is used as a bus to interconnect the functional blocks inside a SoC.

Keywords: AMBA, AXI, Functional Coverage, IP, Multiple Outstanding Transaction, Out of Order Transaction Completion, SoC, UVM, VIP

1. Introduction

The complexity inside the chip has been increased with the recent advancements in Semiconductor industry. This led to the invention of SoC. SoC consists of Processor, Peripherals like Timers, Universal Asynchronous Receiver Transmitter (UART), Serial Peripheral Interface (SPI), General Purpose Input/output (GPIO), Analog to Digital Converter (ADC), Digital to Analog Converter (DAC), memory peripherals etc. So in order to complete the design in a less time, most of the SoC's are being developed from IP's, which is used as a building block for ASIC and FPGA based designs. So in a SoC based designs, these pre-designed IP cores are becoming popular. Chips have to be verified for their functional correctness ie, whether they are working according to specification given or not.

2. VIP

Since complex designs are using reusable, pre-designed logic blocks of IP based designs, so must the verification. A Verification Intellectual Property (VIP) is a reusable verification model that can be inserted into the test bench which can be used for verifying a design. By using a VIP, the verification process will be accelerated and the time to run the first test case can be significantly reduced and also reduces the time to release the product to market. A VIP can be defined as a pre-defined functional block which is designed as configurable component that can be configured by the user and can be integrated easily into different verification environments. VIP's will have the necessary elements for the generation of test bench and protocol checking mechanisms.

In proposed functional verification of AXI2OCP bridge using System Verilog approach, AXI2OCP bridge connects two different protocols ie.AXI and OCP and this bridge helps in converting AXI 3.0 format signals to OCP format signals[1]. In proposed method, verification of all the memory transactions in AXI protocol for all the five channels ie., write address, write data, write response, read address and read data channels are done. In this work, the entire test bench environment is modeled using

*Author for correspondence

System Verilog[2]. In proposed method, design and verification of slave block in Ethernet management is done by using UVM[3].

3. Proposed Work

In System Verilog, the environment (ENV) is not reusable for many test scenarios. Since, different test scenarios require different type of transactions to be generated. So, generation of packets must be done without the dependency on the environment (ENV). Reusability and portability cannot be achieved in System Verilog. It is not possible to change the objects from top level in System Verilog. System Verilog does not have configuration facility for test bench infrastructure. System Verilog uses mail boxes for standard communication. But mailboxes can be recognized only by System Verilog language. No other language can recognize mailbox. These are all the drawbacks of System Verilog. All these drawbacks can be overcome by using latest methodology called Universal Verification Methodology (UVM). Methodology is a set of best practices by verification experts by which Standard framework can be built for the verification environment. UVM have a set of base class libraries[3]. (ex:- uvm_component, uvm_object etc). UVM has a standard communication mechanism to achieve interoperability. (Ex:- TLM ports). In UVM, the testbench can be configured from top level. (Ex:- factory, configuration methods etc). Configuration methods help to build a reusable testbench. UVM helps us to generate scenarios independent of testbench environment. (Ex:- sequence, sequence_items etc.) In UVM, we can achieve reusability in plug and play manner. By using all these advantages of UVM, in this paper, a VIP is developed for AXI.

4. AMBA AXI-4 Protocol

Advanced Micro controller Bus Architecture (AMBA) is an on-chip bus protocol from ARM Limited. It tells about the on-chip interconnect specification for the management and connection of functional blocks including processors and peripheral devices. AXI-4 is the latest version of AMBA family industry standard which is used for on-chip communication[4,5]. AXI specifies about higher performance and also about the existing bus architectures. AXI is the most preferred bus architecture in the recent complex SoC's and FPGA's.

AXI protocol is suitable:-

a. For applications which require high bandwidth.
b. For applications which require high frequency operation, but without using complex bridges.
c. For applications which has to meet the interface requirements of a wide range of components.
d. For memory devices which has high initial access latency.
e. To provide flexibility in the implementation of interconnect architectures.
f. To provide backward compatibility with the existing Advanced High-Performance Bus (AHB) and Advanced Peripheral Bus (APB) components.

The special features of AXI-4 are as follows:-

a. It has separate address/control and data phases, which are spread over 5 different channels.
b. It supports for issuing unaligned data transfers by using byte strobes.
c. It uses the burst based transactions by issuing only the starting address.
d. It supports for issuing multiple outstanding addresses.
e. It supports for out-of-order transaction completion.
f. It has separate read and write data channels are shown in Figure 1.

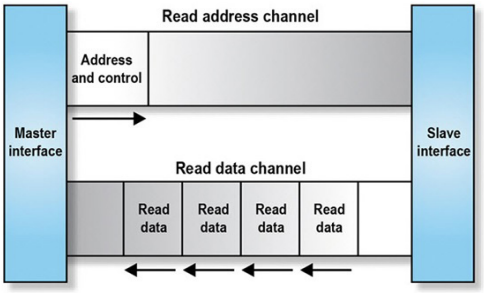

Figure 1. Read channel architecture.

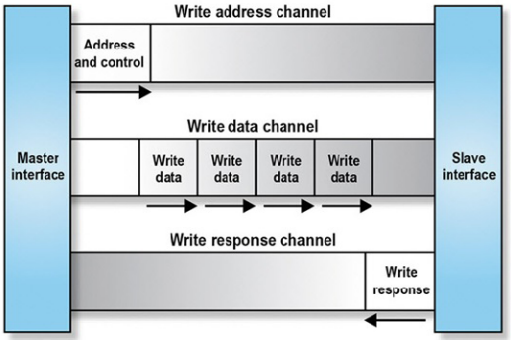

Figure 2. Write channel architecture.

A Master device is one who initiates the transfers and one which has the decision taking capability[6]. A Slave device is one which has no decision taking capability and it has to follow the instructions of master. In AXI protocol, there are 5 channels which work independent to each other. So there is no dependency between each channel. The AXI protocol is meant to offer high data transfer speed. The reasons for this are as follows. ie.,

(A) By using the write data channel, we can perform write transaction and concurrently we can also perform the read transaction by using read data channel, at the same time. So both write and read operations can be performed individually and concurrently. So, this is one of the reason for which AXI offers more data transfer speed[7].

(B)AXI-4 protocol supports for issuing multiple outstanding transactions. It means that before the completion of one transaction, master can initiate one more transaction. In this way master can initiate 'N' number of transactions, so that all these transactions which are initiated by master can be kept in a pipeline[8]. By this way data transfer speed increases and it also improves the efficiency of the bus. So this is the reason for which AXI is best suitable for memory devices which has high initial access latency. This will avoid a high initial access latency slave blocking the channel. As shown in Figure 3, by this way the performance of the bus also will be increased.

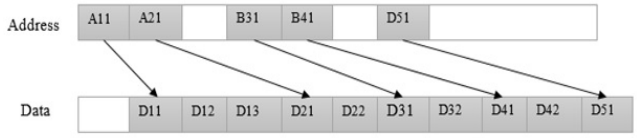

Figure 3. Diagram which explains Multiple Outstanding addresses Transaction scenario.

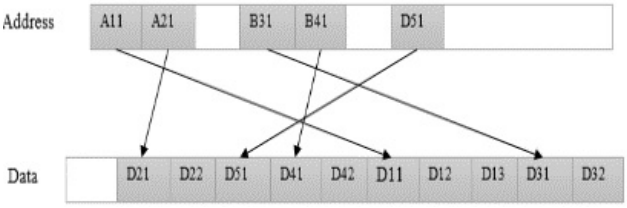

Figure 4. Diagram which explains Out Of Order Transaction completion scenario.

(C)AXI-4 protocol will also support for issuing Out-Of-Order (OOO) transaction completion in write response channel and read data channel only. In AXI-4, in write data channel, data is always sent in order as per the addressed slaves. In read data channel, for suppose if first slave is busy, so it cannot send data immediately. But second slave is ready to send data. So second slave doesn't need to wait for the first slave to complete it's task. So the second slave can send data before first slave. Whenever first slave is ready, it can send it's data then. So faster slaves can respond first, before slower slaves. It effectively reduces the transaction latency. Similarly in write response channel, the faster slaves can give response first to master, such that, that particular transaction is completed. Slave has the capability to send the responses in any manner. Slave may or may not send the responses back in the same order in which master has initiated the transactions. By this way, performance and also the speed of the bus can be improved. The Interconnect will make sure that the response will be going back to the proper Master, whichever has initiated the transaction by matching with AWID is shown in Figure 4.

In AXI, the Master device gives only the starting address of the burst transaction and it is the responsibility of the slave to calculate the next addresses depending upon the burst length and size of each transfer in the burst, which is initiated by Master. Along with address, master will give AWID to the address. Data is sent to slaves and response will come back to proper master by identifying these AWID's. Whenever master is giving valid address, it will assert VALID signal to HIGH. When slave is ready to accept the signals sent by master, slave acknowledges master by making READY signal to HIGH. Master has to hold address and other control information like AWBURST, AWLEN, AWSIZE etc. until slave accepts those signals by asserting READY signal to HIGH. Different signals are grouped into one channel. Through write address channel, master will send starting write address of the burst and some control signals like AWSIZE, AWBURST, AWLEN etc. Through write data channel, master will send write data to slave. Through write response channel, slave will acknowledge to master. For write operation for the entire burst operation, only one response is sent to master and that is also through a separate channel. In AXI a transaction is said to be completed when the slave has sent a response back to master for the data transmitted. Through read address channel, master will send address and some control signals like ARSIZE, ARBURST, ARLEN etc. to Slave. The slave will calculate the next addresses according to the control information sent by master. Now slave will send read data to Master by read data channel. Slave has to send acknowledgment for each and every read data. So slave has the capability

to send both read data and acknowledgment by single channel ie, read data channel. So there is no need of read response channel for AXI.

In order to simplify the design generator in the slave, the address is allocated to a slave in 4KB unit. Bursts cannot cross 4KB boundary. In AXI, termination of early burst is not supported. WLAST signal is used to indicate the last transfer in the write burst and RLAST signal is used to indicate the last transfer in the read burst. WSTRB signal is used to indicate which byte lanes hold valid data in write burst. For every 8-bits of write data, there is one write strobe bit. Burst size will intimate about the maximum number of data bytes to transfer in each beat of transfer within a burst. For the incrementing type of burst, the first transfer can have unaligned transfers, but the rest all transfers in a burst should be aligned transfers. For the wrapping type of burst, all transfers in a burst should be of aligned transfers. For the fixed type of burst, the address remains constant within a burst and every transfer should use the same byte lane. AXI also supports for issuing aligned and unaligned transfers. For Wrapping type of bursts, the burst length should be of 2,4,8,16 only. When data size is less than 32-bit, it is called as narrow transfer.

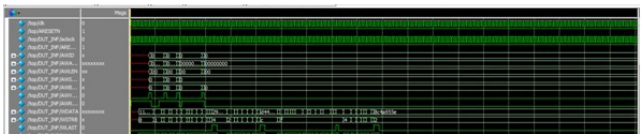

Figure 5. Waveform which explains Write channel's Multiple Outstanding Transaction scenario.

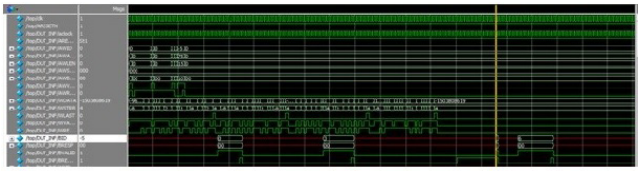

Figure 6. Waveform which explains Out Of Order Transaction completion scenario.

In this work, the VIP is developed without having any Device/Design Under test(DUT) for reference. So, this VIP is having one Master agent and one Slave agent and both of the agents are ACTIVE agents ie., they have all the driver, monitor and sequencer. Each and every transactions initiated by Master agent are assumed that, they are going to a different slave. The number of slaves can be configured from test_case by over_writing the required number of slaves value into the "Number_of_slaves" parameter inside the configuration class. Here, the master agent exactly mimics the master DUT, and slave agent exactly mimics the slave DUT. If the VIP is ready, this VIP will work as per the AXI-4 protocol. Then, if master component (Master DUT) is given, then it should have a slave agent to have communication. Then make Master agent in the VIP as PASSIVE and keep the slave agent as ACTIVE only. Master DUT will drive the data, so only monitor is enough to just monitor those driven signals. If slave component is given, then make the slave agent in VIP as a PASSIVE agent and keep the master agent as ACTIVE only. If both the master component and slave component are given, then make both master agent and slave agent as PASSIVE agents. We can just monitor those signals and verify in the scoreboard.

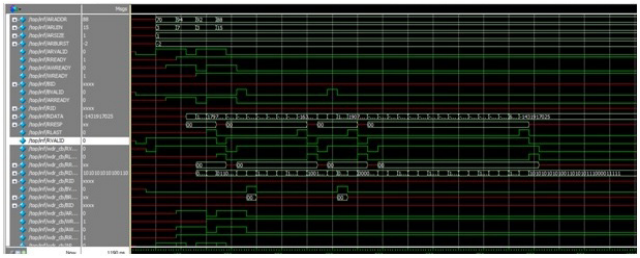

Figure 7. Waveform which explains Read channel architecture.

Coverage Summary by Structure:

Design Scope	Coverage (%)
uvm_pkg	100.00%
uvm_callbacks	100.00%
uvm_phase	100.00%
uvm_component	100.00%
axi_pkg	100.00%
master_seq_base	100.00%
slave_seq_base	100.00%
virtual_seq_base	100.00%
score_board	100.00%

Coverage Summary by Type:

Weighted Average:				100.00%
Coverage Type	**Bins**	**Hits**	**Misses**	**Coverage (%)**
Covergroup	9	9	0	100.00%
Assertion Attempted	28	28	0	100.00%
Assertion Failures	28	0	-	0.00%
Assertion Successes	28	28	0	100.00%

Figure 8. Functional Coverage report generated by Questa Sim.

5. Assertion Based Verification (ABV)

Assertions are used to validate the design's behavior. Assertions capture knowledge about how a design should operate. Assertions are properties that should be true. They also provide information about functional coverage information for a design. Assertions are mainly

used for protocol checking and functional checking also. Assertions will increase the observability and controllability of a design. Assertions improve debug ability. If there are no assertions and if we find some kind of wrong behavior at design output, then it is hard to debug as to figure out where exactly it is going wrong. But if we have assertions embedded to capture all the design rules, then that failure can be debugged much faster as the bug can be caught right at the line of code. Assertions will catch the bugs at the point where they have occurred. So with the help of assertions verification process can be accelerated. So that verification process can be finished in a less time. Assertions depend on the type of stimulus applied. In this VIP, assertions are checked for dependencies between channel handshake signals

6. Test Cases

In this paper, the working of all the five channels are verified. Basic Write and read scenarios for all possible burst types and burst sizes, Multiple outstanding addresses transaction and Out Of Order transaction completion scenarios are also verified successfully by using UVM.

7. Results

In Figure 5, the Master will give four different AWADDR, AWLEN, AWSIZE, AWBURST. Depending on these inputs given by Master, WDATA and WSTRB are being driven respectively. After driving the last data of each burst, WLAST signal is driven high for one clock cycle to indicate the end of the burst.

In Figure 6, the master will give four different AWADDR, AWLEN, AWSIZE, AWBURST. Whichever burst completes the transaction first, it will give response first to Master by BID. BID will match with AWID to indicate response given back to proper master.

In Figure 7, the read channel architecture is shown. The master will give ARADDR, ARLEN, ARSIZE, ARBURST. Depending on these inputs RDATA and response are driven by slave to Master.

Figure 8 shows the Functional Coverage report generated by using Questa Sim Software.

8. Conclusion

A VIP for AMBA AXI-4 protocol is developed successfully and it can be configured and inserted into a test bench for verifying a design. This VIP is used to verify basic write and read transactions for all possible burst types and burst sizes, multiple outstanding addresses transactions, Out of order transaction completions.

9. References

1. Mahesh G, Shaktivel SM. Functional Verification of the Axi2Ocp Bridge using System Verilog and effective bus utilization calculation for AMBA AXI 3.0 Protocol. IEEE Sponsored 2nd International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS). 2015.
2. Mahesh G, Shaktivel SM. Verification of Memory Transactions in AXI Protocol using System Verilog approach. IEEE ICCSP Conference. 2015.
3. Naidu KJ, Srikanth M. Design and Verification of Slave block in Ethernet Management Interface using UVM. Indian Journal of Science and Technology. 2016 Feb; 9(5):1-7.
4. AMBA AXI-4 Specification, Copyright ARM Limited. http://www.gstitt.ece.ufl.edu/courses/eel4720_5721/labs/refs/AXI4_specification.pdf.
5. Sebastian R , Mary SR, Gayathri M, Thomas A. Assertion Based Verification of SGMII IP Core incorporating AXI Transaction Verification Model. International Conference on Control, Communication and Computing India (ICCC). 2015; p. 585-88.
6. Chen X, Xie Z and Wang XA. Development of Verification Environment for AXI Bus Using System Verilog. International Journal of Electronics and Electrical Engineering. 2013; 2(1):112-14.
7. Chen CH, Iu JC, Huang II. A Synthesizable AXI Protocol Checker for SoC Integration. IEEE Transl, ISOCC. 2010; 8:103-6.
8. Ranga A, Venkatesh LH, Venkanna V. Design and Implementation of AMBA-AXI Protocol Using VHDL for SOC Integration. International Journal of Engineering Research and General Science. 2012; 2(4):1-5.