ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Verification of AXI Bus Protocol using SystemVerilog

Sundararajan PH
M.Tech VLSI,
CMR Institute of Technology,
Kandlakoya Village, Medchal Road,
Hyderabad, Telangana 501401.

S.Balaji
Professor & Dean ,
CMR Institute of Technology,
Kandlakoya Village, Medchal Road,
Hyderabad, Telangana 501401.

Abstract:

Advanced microcontroller bus architecture (AMBA) protocol family provides a metric-driven verification of protocol compliance,enabling the comprehensive testing of interface intellectual property (IP) blocks and system-on-chip (SoC) design. The AMBAadvanced extensible interface 4 (AXI4) update to AMBA AXI3 includes: the support for burst lengths up to 256 beats. It is updatedwrite response requirements and removal of locked transactions. Verification has become the dominant cost in the design process.This paper proposes a work, how to build up the verification environment of AXI bus using SystemVerilog is introduced. Functional-coverage, score-boarding and assertions is implemented with the proposed integrated verification environment.

Keywords:

AMBA, AXI, Verification, System Verilog, CDV Coverage Driven Verification etc…

1 Introduction:

Significant efforts have been made over the past couple of years to improve the quality and the productivity of functional digital verification. Only few years ago engineers wrote directed testcases which were composed of simple sequence of '0' and '1' as input stimulus for the design. In addition checking was done manually. It was part of the test and verification engineer's task to explicitly predict the expected response of the DUT for a specific test. In some cases it required even visual inspection of a waveform. Since this extra work had to be repeated for each and very testcase in case of a design change, engineers tended to minimize the amount of hard coded checks within their tests. Advanced verification techniques have been developed and introduced into today's digital design flows to overcome most of those limitations and productivity restrictions. Moreover, the prediction of verification quality is a major improvement in the state of the art verification methods. Main components of these verification techniques are:

- Automated Stimulus Generation
- Automated Self-Checking (Assertions, Reference Models, etc.)
- Automated Coverage Measurements and Tracking

There are two contrasting approaches to coverage-driven verification in current use. "Classical" constrained random verification starts with random stimulus and gradually tightens the constraints until coverage goals are met, relying on the brute power of randomization and compute server farms to cover the state space. More recently, graph-based stimulus generation (also known as Intelligent Testbench) starts from an abstract description of the legal transitions between the high-level states of the DUT, and automatically enumerates the minimum set of tests needed to cover the paths through this state space.

To speed up SoC integration and promote IP reusability, many bus-based communication architecture standards have emerged over the past several years. Since the first 1990s, many onchip bus-based communication architecture standards are projected to handle the communication needs of emerging SoC design. Some of the popular standards include ARM Microcontroller Bus Architecture (AMBA) versions of 2.0 and 3.0, IBM Core Connect, STMicroelectronics STBus, Sonics SMARRT Interconnect, Open Cores Wishbone, and Altera Avalon [2]-[6]. On the other side, the designers simply integrate their owned IPs with third party IPs into the SoC to significantly reduce design cycles.

However, the main issue is that a way to efficiently ensure the IP functionality, that works properly after integrating to the corresponding bus architecture.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

The AMBA AXI protocol is a standard bus protocol and most of the semiconductor companies design interconnects which supports AXI bus interface. AXI protocol is complex protocol because of its ultra-high-performance. On current projects, verification engineers are maximum number designers, with this ratio reaching 2 or 3 to one for the most complex designs. Therefore an efficient verification environment is needed [9]. Verification of such a complex protocol is challenging. This can be easily verified using the verification environment. This verification environment can be reused for other IPs also.

2 Verification Goals and Approaches :

Verification has basically two goals (Figure 1): Firstly, to check that the system fulfills the specification. That means that it does what it is required/specified to do and does not exceed certain limits, e.g. the gain of an amplifier must be above 10db. Secondly,it has to be verified that the system does not do anything "bad" that might have a negative influence on other components or the environment. E.g. the amplifier starts oscillating during power up phase. This second category covers mostly implicit assumption that are naturally made but not explicitly specified. Check specification Avoid "bad" behavior Check corner conditions:

- Operations mode
- Bus protocol/error
- Vdd, temp, … variation
- Process variation Run specific tests Run failure tests Sweep/test different conditions, modes Figure 1: Verification goal and methods.

Figure 1: Verification goal and methods.

Specific simulation runs and tests are required to implement the verification of those two categories above. Moreover, it has to be assured that the verification goals are met in all different operating modes and corners of the system, such as:

- System's operation modes
- Different phases of the system operation
- Silicon process variations and device mismatch
- Varying environment parameters, like temperature, supply voltage:

To ensure this, the tests have to be repeated in those different constellations. It is already obvious that an exhaustive search through the whole space of different tests, operation modes and corners might be impossible to do. Trade-offs have to be made between the verification effort and the level of confidence in the correct behavior of the system.

3 System Verilog :

It is the Hardware Verification Language (HVL). This language is mainly used for the verification purpose. Initially, test bench (TB) is written in Verilog language using tasks and functions [11].

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

But it was a very lengthy process. It overcomes this lengthy process. System Verilog is the updated version of Verilog, it also supports the features like OOPs concept, Randomization and constrained randomization, etc., by the help of these features we can easily generate all the possible combinations of inputs, and thereby we can successively verify the Design.

4 AMBA AXI Architecture:

The AMBA AXI protocol is aimed towards high-frequency system designs and includes a number of features that make it suitable for a high - speed submicrons interconnect. In this project proposes a feature that supports a maximum of 256 data transfers per burst. In AMBA AXI system 16 masters and 16 slaves are interfaced. Every master and slave has their own 4 bit ID tags. The system consists of master, slave and Interconnect bus .

The AXI4 protocol supports the following mechanisms:

- Two kinds of address mode: aligned and unaligned.
- Three types of burst: FIXED, INCR and WRAP.
- Sixteen choices of burst length in the range of 1-256.
- Four varieties of response types: OKAY, EXOKAY, SLVERR and DECERR.

The Advanced Microcontroller Bus Architecture (AMBA) is a protocol that is used as an open standard; on-chip interconnects specification for the connection and management of functional blocks in a system-on-chip (SoC). The AMBA bus is applied easily to small scale SoCs. Therefore, the AMBA bus has been the representative of the SOC market though the bus efficiency. Three distinct buses are defined within the AMBA specification:

1. Advanced Peripheral Bus (APB).
2. Advanced High performance Bus (AHB).
3. Advanced extensible Interface Bus (AXI).

The AMBA specification defines all the signals, transfer modes, structural configuration, and other bus protocol details for the APB, AHB, and AXI buses. The AMBA APB is used for interface to any peripherals which are low bandwidth and do not require the high performance of a pipelined bus interface. APB peripherals can be integrated easily into any design flow, with the following specification:

- Peripheral bus for low-speed devices
- Synchronous, non multiplexed bus
- Single master (bridge)
- 8, 16, 32-bit data bus
- 32-bit address bus

Non-pipelined AMBA AHB is a new level of bus which sits above the APB and implements the features required for high performance, high clock frequency systems, with the following specification:

- Burst transfers
- Split transactions
- Single cycle bus master handover
- Single clock edge operation
- Wider data bus configurations (64/128 bits)

AXI extends the AHB bus with advanced features to support the next generation of high performance SoC designs. The goals of the AXI bus protocol include supporting high frequency operation without using complex bridges, flexibility in meeting the interface, and performance requirements of a diverse set of components, and backward compatibility with AMBA AHB and APB interfaces. The features of the AXI protocol are:

- Separate address/control and data phases
- Support for unaligned data transfers
- Ability to issue multiple outstanding addresses
- Out-of-order transaction completion.

4.1 Design of AXI Protocol

AMBA AXI4 slave is designed with operating frequency of 100MHz, which gives each clock cycle of duration 10ns and it supports a maximum of 256 data transfers per burst. The AMBA AXI4 system component consists of a master and a slave as shown in Figure 2. There are 5 different channels between the AXI master and AXI slave namely write address channel, write data channel, read data channel, read address channel, and write response channel.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

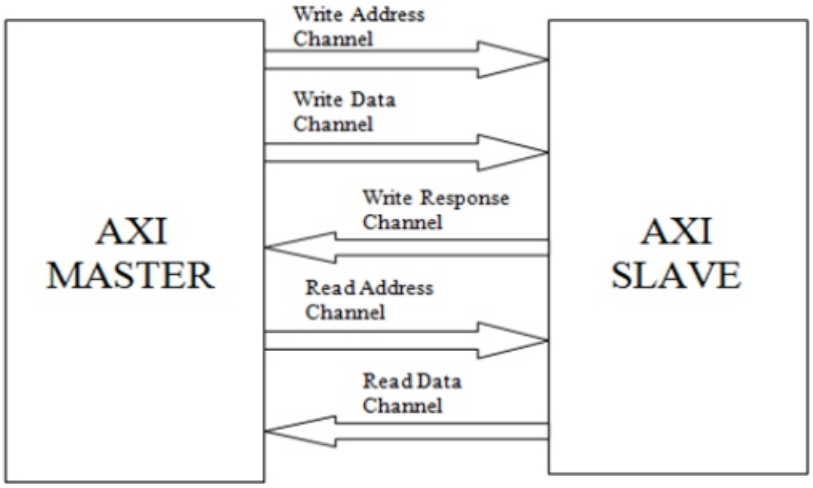

Figure 2 : AXI Master Slave Protocol

4.2 Verification Environment of AXI Protocol:

The verification environment for AXI bus is developed with SystemVerilog, this verification environment is shown in below Figure 3. This environment is organized in a hierarchical layered structure which helps to maintain and reuse it with different designs under verification.

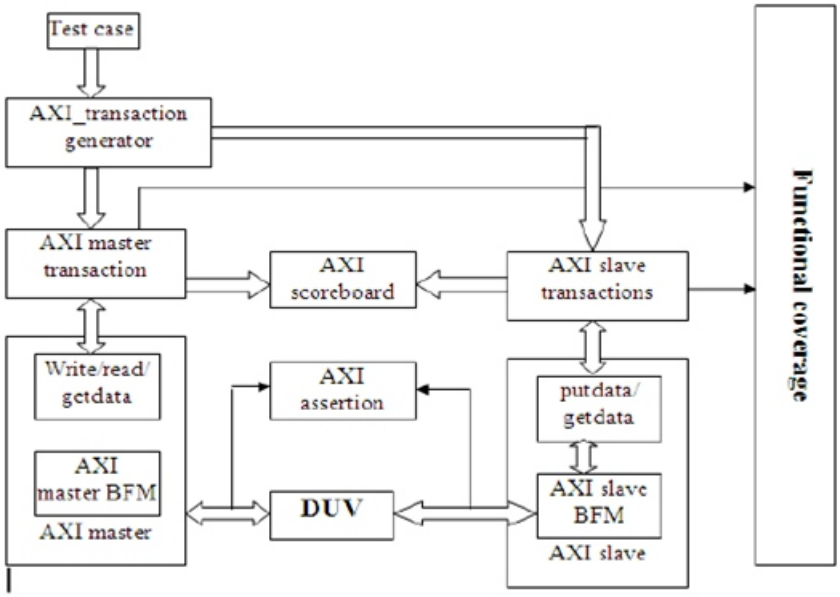

Figure 3 : AXI Verification Environment

4.3 Coverage driven verification flow:

Given that an exhaustive search through the wholeverification space is not practical implies that theverification process is limited in time. However, thecoverage figure still provides an accurate number ofthe verification quality with respect to the definedgoal.Figure 4 assembles the pieces together that have beendiscussed above. The simulation results areautomatically checked and problems are beingreported. An automatic stimuli generator creates testson a random basis within given constraints. On top,the designer might have a certain amount of tests thatare pre-defined and need to be run (directed tests) toreach certain corner cases.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

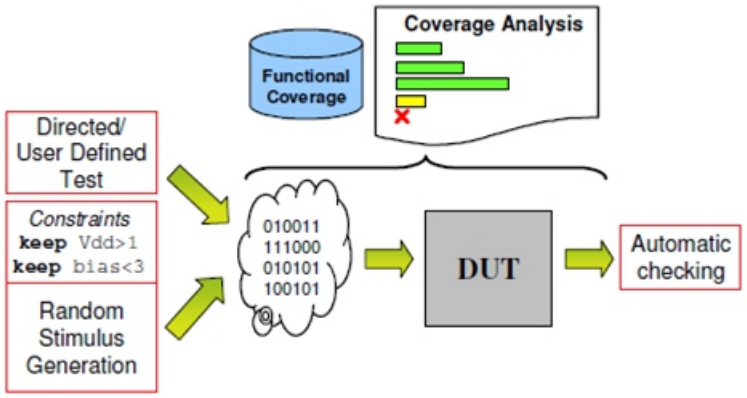

Figure 4: Coverage driven verification flow.

4.3.1 Test Case:

The ,,Test case includes the list of test cases. Each test case is connected to the "sequences" which written for the different scenarios like, single_write_operation, single_read_operation, write_followed_read_operation, multiple_write_single_read, single_wite_multiple_read, etc., Any one of the test case is connected to the Verification Environment to verify the design for a particular scenario.

4.3.2 AXI_Transaction_Generator :

Transaction generator is also known as the "sequence item". Sequence_item is a class which includes all the port signals as its property. All these signals are declared using a "rand" keyword, so that after calling the randomize function this class should assign the random value to the each signal. This generated input values are assigned later to the DUV.

4.3.3 AXI_Master_Transaction:

It includes the signals which are driven from the master. This class has the instance of the AXI_transaction_generator. The master transaction can override the values that are generated in the AXI_transaction_generator. Suppose we have not over ridden any signals, then the values that are generated in the AXI_transaction_generator are passed to the DUV.

4.3.4 AXI_Slave_Transaction:

It includes the functionality similar to AXI_Master_Transaction, except it includes the signals which are driven from the slave.

4.3.5 AXI_Scoreboard :

The values generated in the AXI_Master_Transaction and AXI_Slave_Transaction are also stored in the AXI_scoreboard. Later we can use these signals for the comparison of expected output and the actual output.

4.3.6 Functional Coverage:

This class includes the list different coverage scenarios, which checks for the how much part of the design is covered during verification. AXI_Master_Transaction and AXI_Slave_Transaction classes will invoke this functional coverage.

4.3.7 AXI_Master This is the main block of master part, it includes the two sub-blocks Write/read/get data and AXI master BFM.

Write/read/get data: This sub-block includes the objects of classes sequencer, driver, and monitor. Sequencer picks the assigned sequence and drops it into the driver. It drives these signals according to the protocol. Monitor monitors whether signals are changing according to protocol or not AXI master BFM: This is the class which includes the functions related to the buses. BFM stands for Bus Function Modules. Finally the signals driven from the driver are passed to the DUV. AXI_Slave has the functionality similar to AXI_Master.

4.3.8 AXI_Assertions

It includes the list of assertions which are written according to the signal description. These are written using assert statements. These assertions are applied to the signals that are driving from the driver before applying to the DUV.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

4.4 Results

4.4.1 AXI Write

Fig5 shows a write transaction. The process starts when the master sends anaddress and control information on the write address channel. The master then sendseach item of write data over the write data channel. When the master sends the last dataitem, the WLAST signal goes HIGH. When the slave has accepted all the data items, it-drives a write response back to the master to indicate that the write transaction iscomplete.

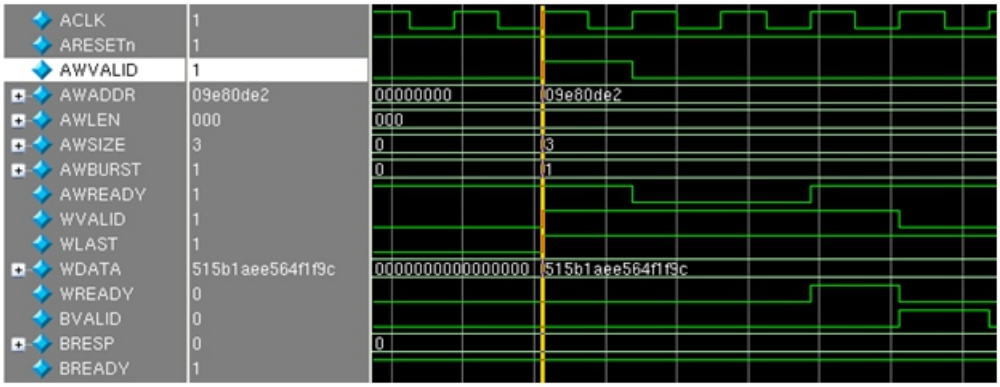

Fig5 : AXI Write

4.4.2 AXI Read

Figure 6 shows a read burst of four transfers. In this example, the master drives the address, and the slave accepts it one cycle later. The master also drives a set of control signals showing the length and type of the burst, but these signals are omitted from the figure for clarity.After the address appears on the address bus, the data transfer oc-curs on the read data channel. The slave keeps the VALID signal LOW until the read data is available. Forthe final data transfer of the burst, the slave asserts the RLAST signal to show that thelast data item is being transferred.

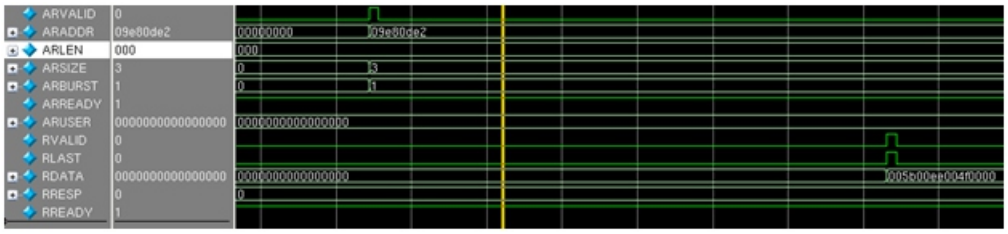

Fig 6: AXI Read

4.4.3 AXI Write_Read

Figure 7 shows a sample AXI Write followed by AXI Read. All five channels use the same VALID/READY handshake to transfer data and controlinformation. This two-way flow control mechanism enables both the master and slave-to control the rate at which the data and control information moves. The source generates the VALID signal to indicate when the data or control information is available. The destination generates the READY signal to indicate that it accepts the data or control information. Transfer occurs only when both the VALID and READY signals are HIGH. AXI Write is completed by WLAST and response is completed when BVALID and BREADY are high. AXI Read is completed by RLAST and RRESP.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

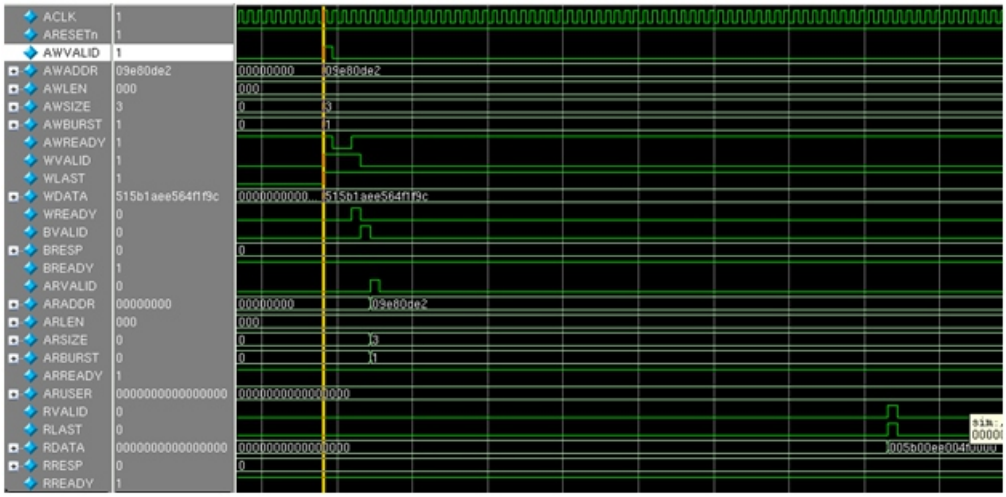

Fig 7: AXI Write followed by Read

4 CONCLUSIONS:

A coverage driven verification approach for digitalsystems has been presented. In the first sectionthe general verification task has been analyzed indetail and the similarities and differences betweenanalog and digital verification are considered.The approach enables advanced verification methodsfor digital design and closes a gap in theincreasing demand of system integration andreliability goals. However, this approach is clearly nota replacement but a complementary measure for pureanalog verification efforts.The advantages of the approach are obvious and havealready been discussed before:

- Advanced verification method
- Well defined verification strategy
- Random generation
- Coverage matrix

AMBA AXI/4 is a plug and play IP protocol. It is released by ARM, defines both bus specification and a technology independent methodology for designing, implementing and testing customized high-integration embedded interfaces. The data is 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. In this paper, an effective verification environment can simulate most cases of the AXI signal, check all the transmitted dataautomatically and complete coverage analysis during the simulation.

So the environment can improve the coverage and reduce the time spending in the verification.

5 FUTURE SCOPE:

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

6 References:

[1] Development of Verification Environment for AXI Bus Using SystemVerilogXu Chen, Zheng Xie, and Xin-An Wang Key Lab of Integrated Micro-Systems Science Engineering and Applicationshttp://www.ijeee.net/uploadfile/2013/0702/20130702105435960.pdf

[2] Coverage Driven Verification for Mixed Signal Systems Cadence Design Systems Walter Hartong, Nils Luetke-Steinhorst, Hannes Froehlich https://www.cadence.com/rl/Resources/conference_papers/107Paper.pdf

[3] IJRET: International Journal of Research in Engineering and Technology eISSN: 2319-1163 | pISSN: 2321-7308 ; DESIGN AND VERIFICATION ENVIRONMENT FOR AMBA AXI PROTOCOL FOR SOC INTEGRATION Pradeep S R1 , Laxmi C2http://esatjournals.org/Volumes/IJRET/2014V03/I15/IJRET20140315066.pdf

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

[4]A Survey of Three System-on-Chip Buses: AMBA, CoreConnect and Wishbone Milica Mitić and Mile Stojčevhttp://es.elfak.ni.ac.rs/Papers/ICEST%20'06.pdf

[5]International Journal of Application or Innovation in Engineering & Management (IJAIEM) ; Design of an AMBA AHB Reconfigurable Arbiter for On-chip Bus Architecture Pravin S. Shete1 , Dr. Shruti Oza2 http://www.ijaiem.org/volume3issue5/IJAIEM-2014-05-29-091.pdf

[6]International Journal of Engineering Inventions ; A Synthesizable Design of AMBA-AXI Protocol for So-CIntegration M. Siva Prasad Reddy1 , B. Babu Rajesh2 , Tvs Gowtham Prasad3 http://www.ijeijournal.com/papers/v1i3/D0131926.pdf
o.