International Journal of Research in Engineering, Science and Management
Volume-2, Issue-1, January-2019
www.ijresm.com | ISSN (Online): 2581-5792

Assertion based Verification of AXI Protocol using UVM

K. V. S. Kumar[1], K. Siva Nagendra[2]
[1]*Student, Dept. of Electronics and Communications Engg., Aditya College of engineering, Surampalem, India*
[2]*Professor, Dept. of Electronics and Communications Engg., Aditya College of engineering, Surampalem, India*

***Abstract*: This paper mainly focuses on verifying the important features of advanced extensible interface (AXI). It verify burst transfers, split transactions, single-cycle bus master handover, single-clock edge operation, wider data bus configuration (16/32bits). Verification of AXI protocol in universal verification methodology. UVM is used for the verification of AXI Protocol which provides the best framework to achieve CDV (Coverage Driven Verification) which combines automatic test generation. Validating the transactions of AXI includes the validation of all the five channels read address, read data, write address, write data and write response. In the VIP design the entire test environment is modeled using UVM and the read, write transactions from the same and different memory locations.**

***Keywords*: UVM, AXI, VIP Architecture, Verification.**

1. Introduction

The Advanced Extensible Interface (AXI) is a part of the Advanced Microcontroller Bus Architecture (AMBA) which is developed by ARM (Advanced RISC Machines) company. It is an On-Chip communication protocol. The AMBA AXI protocol supports high-performance, high frequency system designs. The AXI protocol is suitable for high-bandwidth and low latency designs. It provides high-frequency operation without using complex bridge. It meets the interface requirements of a wide range of components. AXI protocol provides flexibility in the implementation of interconnect architectures. It is backward-compatible with existing AHB and APB interfaces. The key features of the AXI protocol are that it has separate address/control and data phases & support for unaligned data transfers, using byte strobes. It utilizes burst-based transactions with only the start address issued. It has separate read and write data channels that provide low-cost Direct Memory Access (DMA). It supports for issuing multiple outstanding addresses. It support for out-of-order transaction completion. It permits easy addition of register stages to provide timing closure.

2. Planning of verification

The check plan is such a great amount of identified with the equipment spec and incorporates a portrayal Of what situations should be examined and the coding style to be utilized. These means contain formal evidences, affirmations, copying, HW/SW co-confirmation irregular or coordinated testing and utilization of check IP. Coding this style of test seat. Takes longer than a regular coordinated test bench, particularly the self-checking divides, bringing about a postponement before the principal test can be run. This hole can bring about our outcomes freeze, so attempt some portion of our standard that we can see the underlying deferral before the first irregular test case runs.

3. Architecture of UVM test bench

A. UVM test bench and environment

An UVM test bench is composed of reusable verification environments called Verification Components (VCs). The VCs are applied to the device under test (DUT) to verify the implementation of the AXI protocol. Architecture of UVM test Bench is shown in Figure.1

B. Building blocks of test bench

The three main building blocks of a test bench in UVM based verification are

UVM env: It is the top level component of the verification components. Uvm env is extended from uvm component. It is used to create and connect the uvm components like drivers, monitors, sequencers. It can also use as sub environment in another environment.

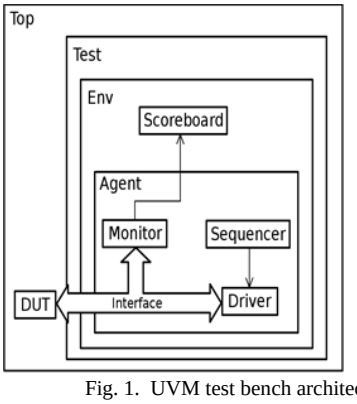

Fig. 1. UVM test bench architecture

UVM test: The uvm test class defines the test scenario for the test bench specified in the test. The test class enables configuration of the test bench and verification components.

UVM verification components: Sequencer (stimulus generator): The sequencer generates stimulus data and passes it to a driver for execution. The uvm sequencer is the base class of uvm class library contains all of the base functionality required to allow a sequence to communicate with a driver.

- *Driver:* The driver drives data items to the bus following the interface protocol. The driver obtains the data items from the sequencer for execution. The UVM Class Library provides the uvm driver base class.
- *Monitor:* The monitor extracts signal information from the bus and translates it into events, struts, and status information.
- *Agent:* An agent has two basic operating modes
- *Active mode:* In this mode, the agent emulates a device in the system and drives DUT signals. This mode requires that the agent instantiate a driver and sequencer. A monitor also is instantiated for checking and coverage.
- *Passive mode:* In this mode, the agent does not instantiate a driver or sequencer and operates passively. Only the monitor is instantiated and configured. This mode is used only when checking and coverage collection is desired.
- *UVM top:* Test Bench top is the module, it connects the DUT and Verification environment components.

4. Architecture of AXI protocol

The AXI based protocol generally implies Bust based. Every transaction stores a control and the information of the address channel which contains the nature of the address which stands for the nature of data that need to be transferred. The data communication among the master and slave by using a signal for writing the data to slave, AXI has a additional response, write response. During the transaction of write transaction the data flow generally from master to slave, the AXI is having a extra write response channel and it allows the slave to indicate the compilation of write signal.

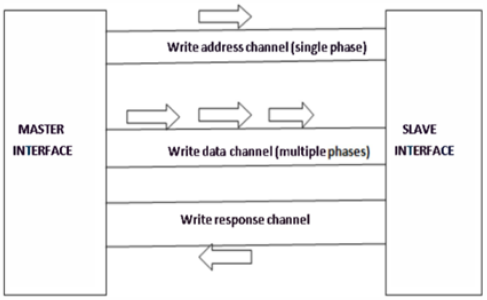

Fig. 2. Write channel architecture

AXI protocol generally enables the address transfer before the exact data transfer and the address can be multiple addresses which also support the out of order transaction. The architecture of AXI architectures are explained as per the “fig. 2,” “fig. 3,” mentioned. The read and write frame work is verified using the system Verilog. For that the verification IP context is being developed in the system Verilog.in the figure explained the framework for write framework the write address is a mono phase signal with a size of 32 bit.

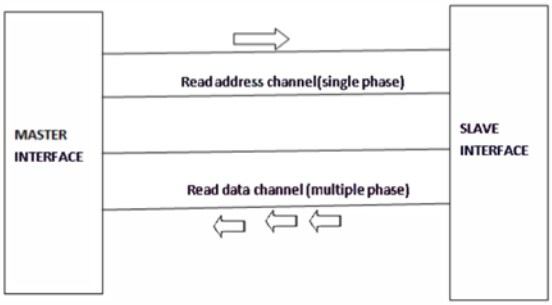

Fig. 3. Read Channel Architecture

The compose information starts from this address in a numerous stage's mode. Amid the compose exchanges in AXI the apportioning of memory totally relies on upon the slave interface reaction. WDATA is a flag which is in charge of various information to be enlisted. This can be plainly conceptualized from the outcomes that are gotten. Vindicating the compose deliver and read deliver to be same and the information that is acquired ought to likewise measure up to in both the compose and read system. Every one of the five channels utilizes the same VALID/READY handshake to exchange information and control data. Exchange of either address data or information Occurs when both the VALID and READY signs are HIGH. ACLK and ARESETn are the Global signs. All signs are inspected on the rising edge of the clock and ARESETn flag is dynamic LOW.

A. Signal descriptions

In AXI signs are arranged into five channels.

- Write address channel signals
- Write data channel signals
- Write response channel signals
- Read address channel signals
- Read data channel signals

A general system consist of master and slave are connected together with the help of some form of interconnect. Signal and with a data of multiple phase. ARADDR is read address signal of length 32 bit. In this test case both the AWADDR and the ARADDR signals should be same, as the write and read are occurring in the same location. This paper mainly focuses in verifying the test cases practically with the help of the obtained waveforms. Each of the five channels utilizes the same VALID/READY handshake to exchange information and supervision. In AXI protocol write address operation is happening at single phase whereas the write data operation is occurring at multiple phases. AWADDR is a write address signal of length 32 bit, representing the particular address where

the multiple data has to be written. Similarly read address is a single phase.

B. Mechanisms of handshake

The source creates the VALID flag to demonstrate when the information or amazingness message is accessible. This two-way stream amazingness component empowers both the ace and slave to control the rate at which the information and matchless quality message moves. The goal creates the READY flag to show that it acknowledges the information or control data. Exchange happens just when both the VALID and READY signs are HIGH. There ought to be no combinatorial ways between tip off and yield motions on both ace and slave interfaces.

5. Result

The following coverage achieved for different test cases.

- Single write and single read test case from same memory location.
- Five write and five read test case from same memory locations.
- Ten write and ten read test case from same memory locations.

The beneath Cover proclamation can be utilized to screen arrangements and other behavioral parts of the outline. The tool can assemble data about the test cases and report the outcomes at the end. At the point when the property for the cover articulation is effective, the pass explanations can indicate a scope capacity, for example, checking all ways for a grouping. A cover property makes a solitary cover point.

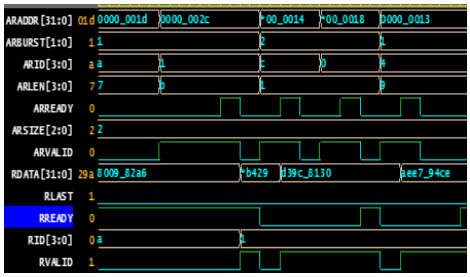

Fig. 4. Write data into address location

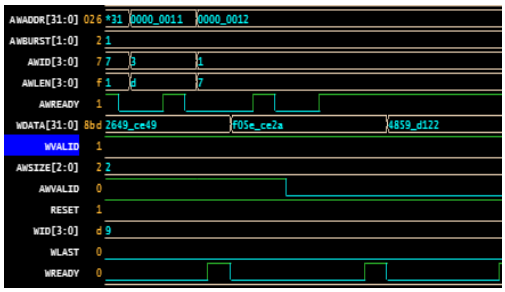

Fig. 5. Read data from address location

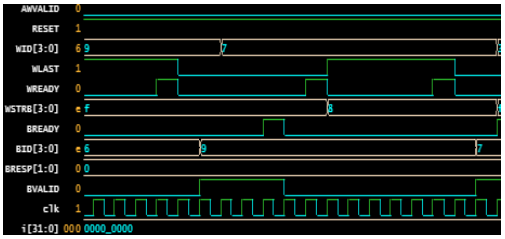

Fig. 6. Response signals

A. Inference

Bin indicates name and count with a set of values or a sequence. If the bin designates a set of values, the count increments each time the coverage point hit one of the values in the set. If the bin describes a sequence of values, the count increases each time the coverage point hits the entire sequence of values. Since we are using three different test case

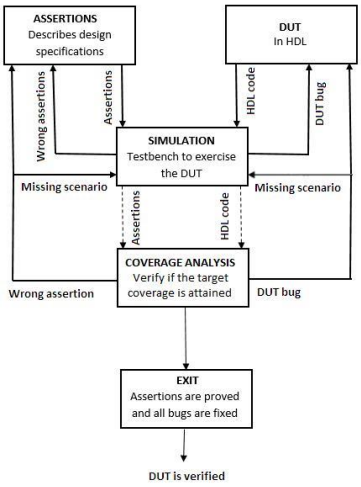

Fig. 7. Assertion flow diagram

For each test case the TYPE indicates how many times it hits the set of values. So among three cases, 12 times write read case achieves larger coverage because it is having 10 chances to validate compared to remaining test cases, so its chances of covering the design will definitely be more always.

B. UVM report

```
UVM_INFO axi_sb.sv(81) @ 11745: uvm_test_top.env.scoreboard [SCOREBOARD] Comparison
Successfully of both packet in read part
UVM_INFO /apps/vcsmx/etc/uvm-1.2/src/base/uvm_objection.svh(1270) @ 12635: reporter
[TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
UVM_INFO axi_sb.sv(120) @ 12635: uvm_test_top.env.scoreboard [SCOREBOARD]

  SUCCESSFULLY DATA WRITE=502
  SUCCESSFULLY MATCH=207
  READ BEFORE WRITE=61
  DATA WRITE NOT POSSIBLE DUE TO WSTRB BIT IS 0 =10
  NOT MATCH=0

  write pkt success=12
  write pkt failure=0
  read pkt success=10
  read pkt failure=0
  total packets=22

UVM_INFO /apps/vcsmx/etc/uvm-1.2/src/base/uvm_report_server.svh(847) @ 12635: reporter
[UVM/REPORT/SERVER]
--- UVM Report Summary ---

** Report counts by severity
UVM_INFO :   71
UVM_WARNING :    0
UVM_ERROR :    7
UVM_FATAL :    0
** Report counts by id
[ASSERTION]     7
[RNTST]     1
[SCOREBOARD]    67
[TEST_DONE]     1
[TIMOUTSET]     1
[UVM/RELNOTES]     1

$finish called from file "/apps/vcsmx/etc/uvm-1.2/src/base/uvm_root.svh", line 527.
$finish at simulation time                12635
           V C S   S i m u l a t i o n   R e p o r t
```

C. Assertion fail

```
----------------------------------------------------------------------------------------
UVM_INFO axi_sb.sv(50) @ 10860: uvm_test_top.env.scoreboard [SCOREBOARD] Comparison Success
"axi_intf.sv", 191: top.intf.rvar_are_stable: started at 10885s failed at 10885s
        Offending '(($stable(RRESP) && $stable(RDATA)) && $stable(RID))'
UVM_ERROR axi_intf.sv(192) @ 10885: reporter [ASSERTION] Failure AXI_RVALID_STABLE
UVM_INFO axi_sb.sv(31) @ 11415: uvm_test_top.env.scoreboard [SCOREBOARD] pkt From Monitor t
----------------------------------------------------------------------------------------
```

6. Conclusion and future scope

In this paper, VIP component development for AXI3.0 protocol is achieved. As part of this thesis the VIP components which includes Generator, Monitor, Slave and Coverage models are developed and achieved basic scenarios targeting all features of AXI protocol. Firstly list down features, scenarios from the data specification sheet. The important features of AXI WRITE Address, WRITE Data, Write Response, Read Address and Read Data are validated by taking Test cases like Single Write cycle and Single Read cycle from address location, Five Write cycle and Five Read cycle from successive address locations and Ten Write cycle and Ten Read cycle from successive address locations. After running these different test cases the coverage done by each test case is observed. In future the complexity of AXI Protocol can be increased from the data specifications which means if the basic scenarios are validated successfully the specs can be increased further and validate it. Once the test bench architecture is Setup it can be used for any design.

References

[1] Shaila S Math, Manjula R B, “Survey of system on chip buses based on industry standards”, Conference on Evolutionary Trends in Information Technology(CETIT), Bekgaum,Karnataka, India, pp. 52, May 2011.
[2] ARM, AMBA Specifications (Rev2.0). http://www.arm.com, 1999.
[3] Alan P. Su, JifT Kuo, Kuen Jong Lee, Ing Jer Huang, Guo An Jian" A Multi core Software/Hardware Codebug Platform with ARM CoreSightTM, On chip Test Architecture and AXIIAHB Bus Monitor ".
[4] Synopsys, VCS / VCSi User Guide Version 10.3, www.synopsys.com
[5] Core connect bus architecture. IBM Microelectronics. http://www.ibm.com/chips/products/coreconnect, 2000.
[6] Na Ra Yang, Gilsang Yoon, Jeonghwan Lee, Intae Hwang, Cheol Hong Kim, Sung Woo Chung and Jong Myon Kim, “Improving the System-on-a-Chip Performance for Mobile Systems by Using Efficient Bus Interface”, International Conference on Communications and Mobile Computing, vol. 4, pp. 606-608, March 2009.