International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 05 Issue: 07 | July - 2021 ISSN: 2582-3930

Practical Approach to UVM

Rishabh Kaur Dingra[1], Dr.Bharthi S.H[2]

[1]ECE Department ,REVA University
[2]ECE Department, REVA University

Abstract - With the advancement of technology or IP a flexible methodology is required for verification. Universal verification Methodology (UVM) is one such methodology derived from system Verilog which gives scalability, plug and play feature and flexibility. In this paper UVM is used for the verification of AXI protocol which is one of the most used specification of Advanced Microcontroller Bus specification (AMBA).

***Key Words*:** NP-hard , polynomial time , Directed acyclic graph.

1.INTRODUCTION *(Size 11, Times New roman)*

For the verification of IC's Universal Verification Methodology (UVM) is a standard or more automated methodology which was derived from OVM on 21st February 2011 by Accellera. UVM provides base class for each of the required component of verification environment like Transactor, Driver, Monitor etc. which makes the methodology more easy and time saving. Frequently used functions like copy, print, verbosity etc are defined in main class uvm_object. UVM supports Object Oriented programming (OOPs) which provides encapsulation, abstraction, polymorphism and inheritance . OOPs concept allows to create objects and provides methods to handle or use those objects. In support of OOPs concept UVM provides factory for instantiating objects which can handle class properties. User need to call `uvm_object_utils(class_name) or `uvm_component_utils (class_name) from factory to instantiate an object. In this paper verification of AXI protocol is done using UVM. AXI protocol is an important and one of the most widely used specification of AMBA. There are two versions one is AXI release in 2003 other is AXI4 which was released in 2010.Further AXI4 is divided as AXI4 , AXLI-Lite and AXI-Stream. It is an enhancement of AXI with much improved performance and some other features. AXI consists of 5 transaction channels which works independent of each other. Those channels are write address , write data, write response , read address and read data. It consist of some special features like burst mode i.e transaction happens in burst mode in which user need to specify only starting address other feature is strobe which provides unaligned transfer of data. In all AXI is a very flexible and most compatible protocol which provides low latency , high bandwidth.

2. UVM TESTBENCH ARCHITECTURE

A typical architecture environment of UVM Testbench consist of :

(1) Sequence : It is a parent sequence class used to generate stimuli.
(2) Sequencer : It is a child sequence class which extract or use the properties of sequence to control the flow mechanism of stimuli or transaction generated by sequence.
(3) Driver : Uvm_driver is used to get the transaction from sequencer and put it to the DUT with the help of TLM(Transaction Level Modelling) .
(4) Monitor : Uvm_Monitor is used to transmit the information from DUT for further analysis.
(5) Agent: Uvm_Agent consist of other components which includes sequencer , Driver and Monitor which performs above mentioned operation.
(6) Scoreboard : Uvm_Scoreboard is to verify the functionality or behaviour of the Design Under Test. It compares the actual result received from Uvm_Agent with the expected transaction .(7) Environment : Uvm_Component combines the above mentioned components (scoreboard , sequencer, agent) which maintains a proper hierarchy and provides encapsulation.
(8)Test : Uvm_test is the top-level hierarchy that creates the environment, configures it (through factory overrides or the configuration database), and provides stimulation to the DUT by invoking UVM Sequences through the environment.
(9)Testbench : Uvm_Testbench provides the communication or connection between the DUT and Uvm_Test.

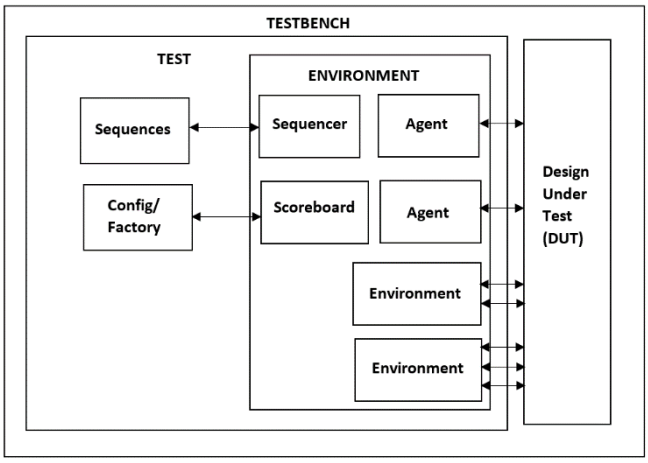


Fig -1: UVM Testbench Architecture

3. AXI READ AND WRITE ARCHITECTURE

The Five Independent channels of AXI are as follows:

1. Write Address channel
2. Write Data channel
3. Write Response channel
4. Read Address channel
5. Read Data channel

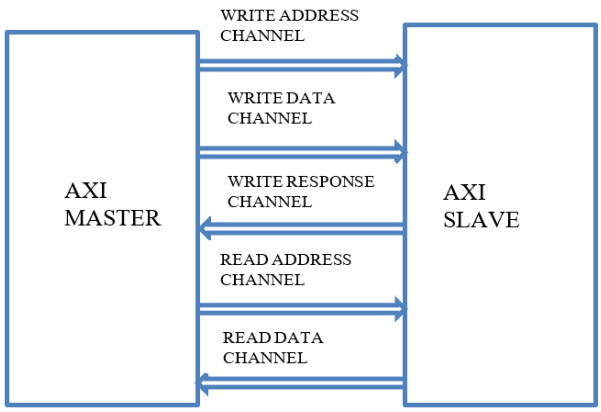


Fig -2: AXI READ &WRITE CHANNEL

According to the operation data can be transfer from mater to slave or slave to master simultaneously. Address of a burst is transferred from AXI Master to AXI Slave through Write address channel and respective data is also transferred from AXI Master to AXI Slave through Write data channel. After receiving the data with corresponding address AXI Slave sends the acknowledgment to AXI Master through write response channel. If AXI wants to read the data of a particular address that address is transmitted to AXI slave through read address channel then AXI Slave transmits that data through read data channel.

A. AXI READ DATA AND ADDRESS CHANNEL

READ ADDRESS CHANNEL :

- ARID, ARADDR, ARBURST, ARLEN, ARSIZE, ARCACHE, ARLOCK, ARPROT are the read address signals driven by the AXI MASTER, with ARVALID as HIGH indicating that the driven signals are valid .
- When ARESET signal is HIGH, only AXI MASTER drives the signals; otherwise, the signals are driven as zero.
- AXI MASTER does not drive the ARVALID signal, which is driven by the ARREADY signal, as LOW until it receives the ARREADY signal.

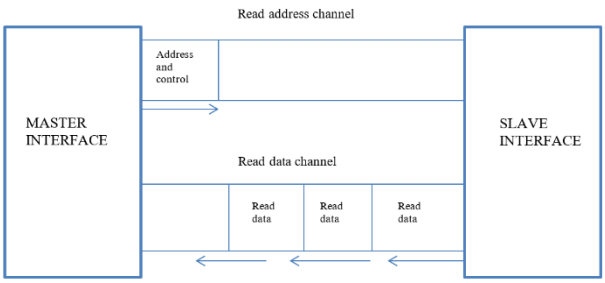


Fig -3: AXI READ FLOW

READ DATA CHANNEL:

- AXI_SLAVE drives the Read data signal after receiving the read address signal.When ARESETn is HIGH only then these signals are asserted otherwise all these signals remains as zero.
- AXI SLAVE only drives read data signal when RVALID is HIGH, and it keeps the same value until it receives the RREADY signal.
- RREADY drives next data only when RDATA is HIGH.
- ARLEN number of data is driven by AXI SLAVE. When the last data is driven, RLAST becomes HIGH.

B. AXI WRITE DATA AND ADDRESS CHANNEL

WRITE ADDRESS CHANNEL:

- AWID,AWADDR, AWBURST, AWLEN, AWSIZE, AWCACHE, AWLOCK, AWPROT are the write address signals driven by the AXI MASTER, with AWVALID as HIGH indicating that the driven signals are valid.
- When ARESETn is HIGH, AXI MASTER drives the write signals; otherwise, it drives all signals as zero until it receives the AWREADY signal, which is driven by AXI SLAVE .

WRITE DATA CHANNEL:

- After transmitting the write address signals, the AXI MASTER drives the Write data signals.
- When the ARESETn signal is HIGH, it drives the signals; otherwise, it drives them as zero.
- When the WVALID signal is high , AXI MASTER drive the WDATA signal, which remains at the same value until the WREADY signal is received.
- WDATA is triggered if WREADY hits HIGH.
- AWLEN no. of data is driven by AXI_MASTER.WLAST becomes HIGH when last data is being transferred.

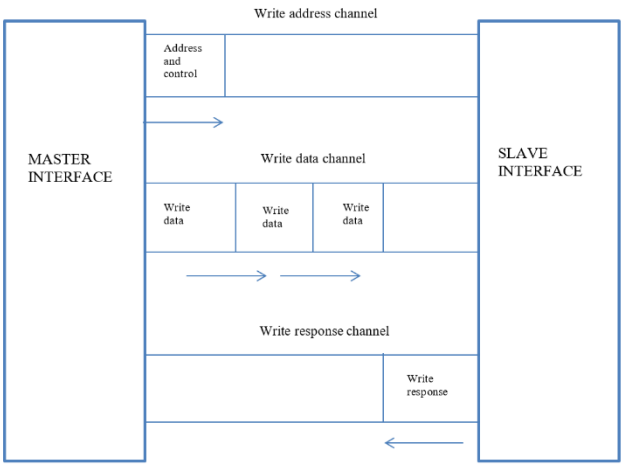
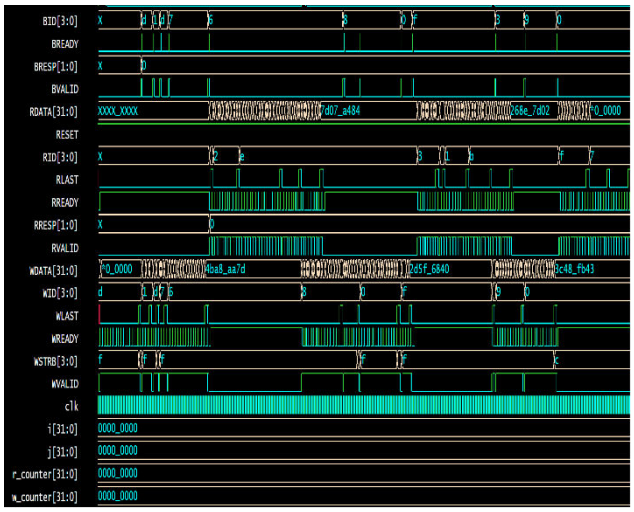
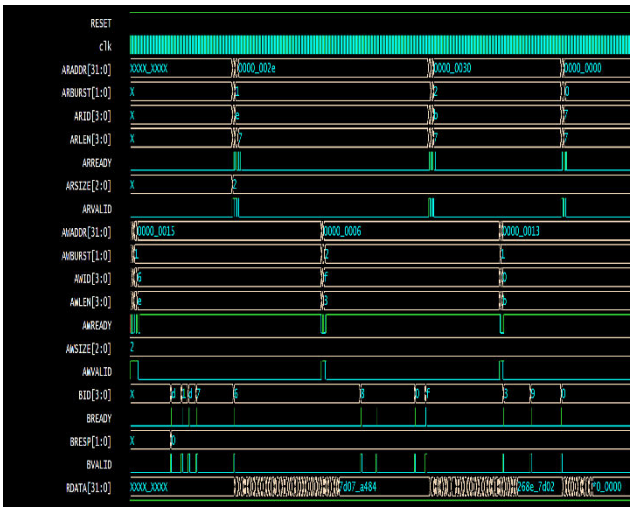


Fig -4: AXI WRITE FLOW

3. RESULT





4. CONCLUSION

The AXI protocol is well-suited for high-bandwidth, low-latency designs because it allows for high-frequency operation without the use of complex bridges and meets the interface requirements of a diverse set of component. It is also super appropriate for memory dash controllers with high low initial access latency, provides hard flexibility in the implementations of inter interconnect architectures, and is upward-backward-compatible with existing interconnect architectures.

ACKNOWLEDGEMENT

I'd like to thank my advisor, Dr. Bharthi S.H., for her unwavering support of my studies and research, as well as her patience, inspiration, excitement, and vast knowledge. Her advice was invaluable during my research.

REFERENCES

1. https://www.xilinx.com/support/documentation/ip_documentation/ug761_axi_reference_guide.pdf
2. http://www.gstitt.ece.ufl.edu/courses/fall15/eel4720_5721/labs/refs/AXI4_specification.pdf
3. https://developer.arm.com/documentation/102202/latest
4. https://www.chipverify.com/uvm/uvm-introduction