ISSN (Online) 2395-2717

International Journal of Engineering Research in Electrical and Electronic Engineering (IJEREEE)
Vol 7, Issue 7, July 2021

Development of VIP for AMBA AXI v4.0 Protocol using UVM

[1] Venkatesh.T, [2] Dr.Narendra C.P

[1][2] Dept of Electronics & Comm Engineering,Bangalore Institute of Technology

Abstract-— **System on Chip(SoCs) allows the integration of different Intellectual Properties(IPs) which makes verification of IP very complex and time-consuming. In order to have proper communication between the IPs, Advanced Microcontroller Bus Architecture(AMBA)-based Advanced eXtensible Interface(AXI) protocol is used as a system bus that provides high bandwidth, low latency, and can able to operate at high frequencies. And comparing with other AMBA protocols AXI provides better efficiency while utilizing the bus. Verification of complex SoCs does take more time to verify. so, by developing a reusable Verification IP (VIP)that allows to reuse this verification environment to verify other SoCs by means of this reusable VIP time taken to verify the SoCs will be greatly reduced. The focus of this research is on creating a VIP for the AMBA AXI v4.0 protocol by utilizing the Universal Verification Methodology (UVM). Here the Design Under Test(DUT) was developed using Verilog and SystemVerilog and the testbench environment has been developed using SystemVerilog verification language and UVM Methodology. And all five independent channels supported by the AXI protocol have been verified. Simulation results and Verification of channels along with functional coverage have been carried out using Mentor Graphics Questa Sim tool**

Index Terms - AMBA,AXI,SoC,VIP,UVM.

I. INTRODUCTION

The Because of the AMBA protocol's flexibility, it may be used in a variety of SoC architectures with different size, power, and performance requirements[4]. AMBA protocols are widely used open standards, ensuring compatibility across IPs from different providers for SoCs. Because of this interoperability, low-friction integration and IP reuse are possible, resulting in a faster time to market. Fig.1 shows the evolution of different AMBA protocols.

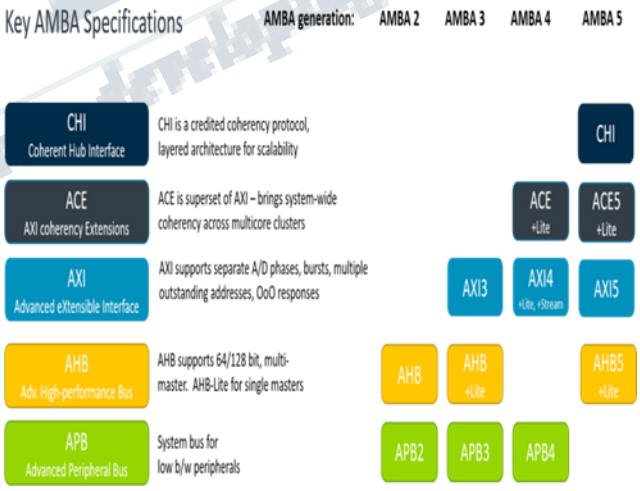

Fig.1 Evolution of AMBA Protocols

For validating designs, the so-called traditional verification method of developing test benches in Verilog or VHSIC Hardware Description Language (VHDL) and driving stimulus to the design is inefficient.

As design complexity has increased, these techniques have become burdensome and insufficient to validate the

ISSN (Online) 2395-2717

International Journal of Engineering Research in Electrical and Electronic Engineering (IJEREEE)
Vol 7, Issue 7, July 2021

IP. System Verilog's advanced features aid in the creation of a possible verification environment.

SystemVerilog's[3] goals are to reduce the cost of verification tool adoption and ownership, as well as to create a larger market through industry-wide support[5]. By becoming an industry standard, SystemVerilog will achieve these goals. To reduce verification effort and increase verification efficiency, a systematic strategy is required. In this study, we apply UVM[6], an SV-based methodology approach that aids in the development of reusable, reliable, and robust verification environments. The UVM approach is used to create VIP for AXI.

The remaining sections mainly focus on, Section II discusses different AXI channels, along with interface signals of each channel.Section III discuss on development of Verification IP by developing a SystemVerilog-based Universal Verification Methodology(UVM) testbench environment. And Simulation and Verification results along with the coverage report are shown in section IV.

METHODOLOGY

Architecture of AXI Protocol

This project aims to have communication between the single master and single slave using AXI Interface. And it has been designed using Verilog and SystemVerilog and verified using UVM methodology

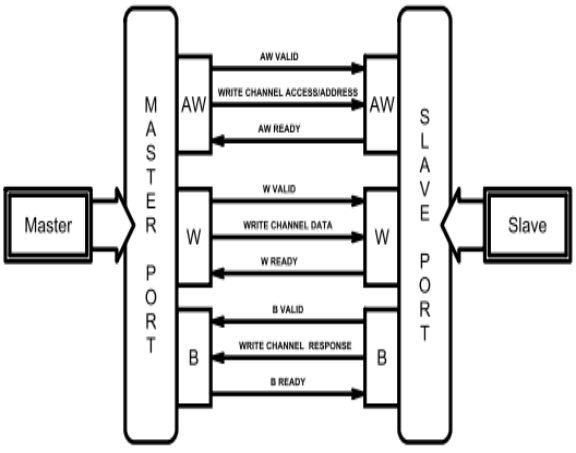

Fig.2 Write Channels of AXI

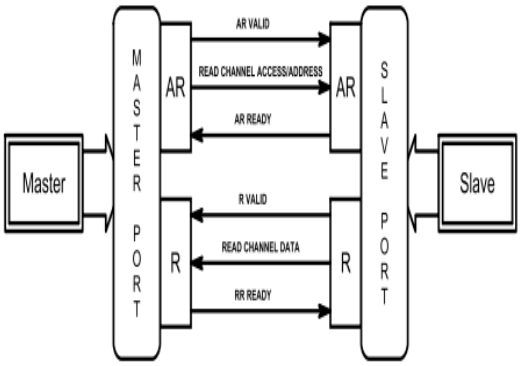

Fig.3 Read Channels of AXI

The AXI design establishes five unique channels and introduces the concept of ID to allow many transactions to occur at the same time. Fig.2 and 3 shows the write and read channel architectures, respectively.

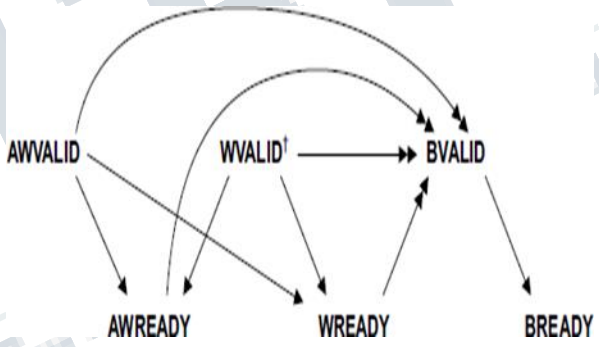

Fig.4 Write Channel handshake

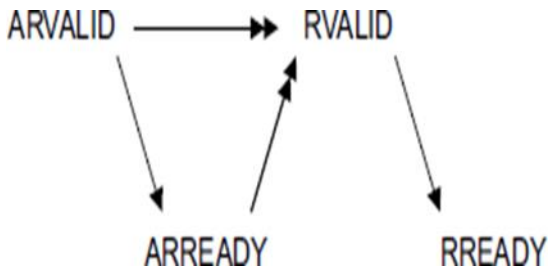

Fig.5 Read Channel handshake

1. Address Write Channel (AW): The channel transfers address and associated control signal information for a write transaction.
2. Write Data Channel(W): The channel carries data and bus with a data bus width of 8 or, 16 or, 32 or, 64 or, 128 or, 256 or, 512 or, 1024.

ISSN (Online) 2395-2717

International Journal of Engineering Research in Electrical and Electronic Engineering (IJEREEE)
Vol 7, Issue 7, July 2021

3. Address Read Channel(AR): The channel transfers address and associated control signal information for the read transaction.
4. Read Data Channel(R): The channel carries data and the read response from the slave to master and the data bus width can be8 or, 16 or, 32 or, 64 or, 128 or, 256 or, 512 or, 1024 wide.
5. Write Response Channel(B): This channel ensures by a slave to reply to write transaction by a master to indicate the transfer completion. It is a 2-bit signal.
2'b00 is an OKAY response, which means the success of a normal transaction.
2'b01 is EXOKAY response, indicates it is a failure to normal transaction and success for an exclusive transaction.
2'b10 is SLEVERR response indicates when master trying to access FIFO/buffer in overflow or an underflow condition. If any request comes with an unsupported transfer size. Trying to write into a readonly location.
2'b11 is DECERR response, returns when interconnect is unable to decode the slave access.
The write transaction dependencies are depicted in Fig.4 and the read transaction requirements are depicted in Fig.5

Interface signals of AXI Protocol

Table 1:Write Address Channel Signals

Signal	Source	Description
AWID[3:0]	Master	Write address ID. This signal is the identification tag for the write address group of signals.
AWADDR[31:0]	Master	Write address. The write address bus gives the address of the first transfer in a write burst transaction. The associated control signals are used to determine the addresses of the remaining transfers in the burst.
AWLEN[3:0]	Master	Burst length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address. See Table 4-1 on page 4-3.
AWSIZE[2:0]	Master	Burst size. This signal indicates the size of each transfer in the burst. Byte lane strobes indicate exactly which byte lanes to update. See Table 4-2 on page 4-4.
AWBURST[1:0]	Master	Burst type. The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated. See Table 4-3 on page 4-5.
AWLOCK[1:0]	Master	Lock type. This signal provides additional information about the atomic characteristics of the transfer. See Table 6-1 on page 6-2.
AWCACHE[3:0]	Master	Cache type. This signal indicates the bufferable, cacheable, write-through, write-back, and allocate attributes of the transaction. See Table 5-1 on page 5-3.
AWPROT[2:0]	Master	Protection type. This signal indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access. See *Protection unit support* on page 5-4.
AWVALID	Master	Write address valid. This signal indicates that valid write address and control information are available: 1 = address and control information available 0 = address and control information not available. The address and control information remain stable until the address acknowledge signal, **AWREADY**, goes HIGH.

Table 2:Write Data Channel Signals

Signal	Source	Description
WID[3:0]	Master	Write ID tag. This signal is the ID tag of the write data transfer. The **WID** value must match the **AWID** value of the write transaction.
WDATA[31:0]	Master	Write data. The write data bus can be 8, 16, 32, 64, 128, 256, 512, or 1024 bits wide.
WSTRB[3:0]	Master	Write strobes. This signal indicates which byte lanes to update in memory. There is one write strobe for each eight bits of the write data bus. Therefore, **WSTRB[n]** corresponds to **WDATA[(8 × n) + 7:(8 × n)]**.
WLAST	Master	Write last. This signal indicates the last transfer in a write burst.
WVALID	Master	Write valid. This signal indicates that valid write data and strobes are available: 1 = write data and strobes available 0 = write data and strobes not available.
WREADY	Slave	Write ready. This signal indicates that the slave can accept the write data: 1 = slave ready 0 = slave not ready.

Table 3:Write Response Channel Signals

Signal	Source	Description
BID[3:0]	Slave	Response ID. The identification tag of the write response. The **BID** value must match the **AWID** value of the write transaction to which the slave is responding.
BRESP[1:0]	Slave	Write response. This signal indicates the status of the write transaction. The allowable responses are OKAY, EXOKAY, SLVERR, and DECERR.
BVALID	Slave	Write response valid. This signal indicates that a valid write response is available: 1 = write response available 0 = write response not available.
BREADY	Master	Response ready. This signal indicates that the master can accept the response information. 1 = master ready 0 = master not ready.

Table 4:Read Data Channel Signals

Signal	Source	Description
RID[3:0]	Slave	Read ID tag. This signal is the ID tag of the read data group of signals. The **RID** value is generated by the slave and must match the **ARID** value of the read transaction to which it is responding.
RDATA[31:0]	Slave	Read data. The read data bus can be 8, 16, 32, 64, 128, 256, 512, or 1024 bits wide.
RRESP[1:0]	Slave	Read response. This signal indicates the status of the read transfer. The allowable responses are OKAY, EXOKAY, SLVERR, and DECERR.
RLAST	Slave	Read last. This signal indicates the last transfer in a read burst.
RVALID	Slave	Read valid. This signal indicates that the required read data is available and the read transfer can complete: 1 = read data available 0 = read data not available.
RREADY	Master	Read ready. This signal indicates that the master can accept the read data and response information: 1= master ready 0 = master not ready.

Table 5:Read Address Channel Signals

Signal	Source	Description
ARID[3:0]	Master	Read address ID. This signal is the identification tag for the read address group of signals.
ARADDR[31:0]	Master	Read address. The read address bus gives the initial address of a read burst transaction. Only the start address of the burst is provided and the control signals that are issued alongside the address detail how the address is calculated for the remaining transfers in the burst.
ARLEN[3:0]	Master	Burst length. The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address. See Table 4-1 on page 4-3.
ARSIZE[2:0]	Master	Burst size. This signal indicates the size of each transfer in the burst. See Table 4-2 on page 4-4.
ARBURST[1:0]	Master	Burst type. The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated. See Table 4-3 on page 4-5.
ARLOCK[1:0]	Master	Lock type. This signal provides additional information about the atomic characteristics of the transfer. See Table 6-1 on page 6-2.
ARCACHE[3:0]	Master	Cache type. This signal provides additional information about the cacheable characteristics of the transfer. See Table 5-1 on page 5-3.
ARPROT[2:0]	Master	Protection type. This signal provides protection unit information for the transaction. See *Protection unit support* on page 5-4.
ARVALID	Master	Read address valid. This signal indicates, when HIGH, that the read address and control information is valid and will remain stable until the address acknowledge signal, **ARREADY**, is high. 1 = address and control information valid 0 = address and control information not valid.
ARREADY	Slave	Read address ready. This signal indicates that the slave is ready to accept an address and associated control signals: 1 = slave ready 0 = slave not ready.

ISSN (Online) 2395-2717

International Journal of Engineering Research in Electrical and Electronic Engineering (IJEREEE)
Vol 7, Issue 7, July 2021

Write Strobe Signal

The WSTRB signal can be used by a manager to indicate to a subordinate which bytes of data are significant. For efficient mobility, WSTRB are beneficial for caching sparse data arrays.

The write channel on the data bus has one strobe bit per byte. These bits make up the WSTRB signal. According to the manager, the write strobes for valid data must be set to high.

Take into consideration the 64-bit write data bus. In the WSTRB signal, each byte is represented by one bit. WSTRB values decide which byte lanes are valid, as indicated in the diagram below

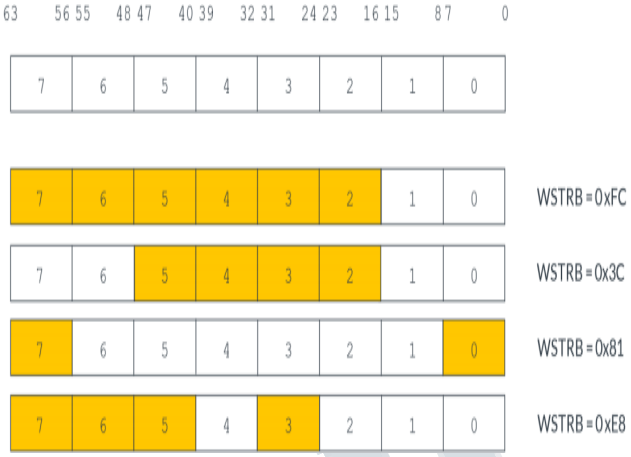

Fig.6 Example for WSTRB signal

Burst Type

AxBURST[1:0] specifies the transaction's burst type: fixed, incrementing, or wrap.

Table 6: Burst type encoding

Burst Type	Value	Defination	Length of transfers	Unaligned Support
FIXED	2'b00	Write and Read repeaditely to the same address	1 to 16	No
INCR	2'b01	Incrementing type: From the previous transaction address, Subordinate will increments the address for each specific beat in the transfer. It depends on the value of AxSIZE.	AXI4: 1-256 AXI3: 1-16	YES
WRAP	2'b10	Wrap type burst: The address will be incremented till it hits the upper or end border of the transaction from the starting address. once it reaches end address.	2, or 4, or 8, or 16	No. Always issues aligned address to the transfer size.

ISSN (Online) 2395-2717

International Journal of Engineering Research in Electrical and Electronic Engineering (IJEREEE)
Vol 7, Issue 7, July 2021

		it get back to the transaction's initial address.		
RESERVED	2'b11	-	-	-

QoS Signalling

AXI4 added two 4 bit QoS signal each for read and write: AxQOS represents either AWQOS for write or ARQOS for read.

AXI4 helps to priortize the transactions based on the QoS value provided by the manager/master. The transaction with the high QoS value has higher priortization compared with the low QoS value.

Fig.7, shows an example for QoS signaling. Where the components such as CPU requires higher access to the memory compared to the other components. So, by assigning suitable QoS values to each transaction. Interconnect will arbitrate the transactions with higher QoS values first compare to the transactions with lower QoS values.

Direct Memory Controller(DMC) is used to inverse the transactions inorder to ensure that higher QoS values given more importance.

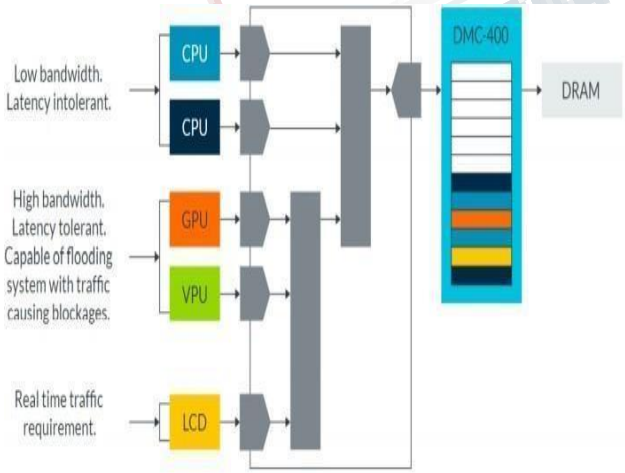

Fig.7 Example for QoS signaling

TESTBENCH ARCHITECTURE

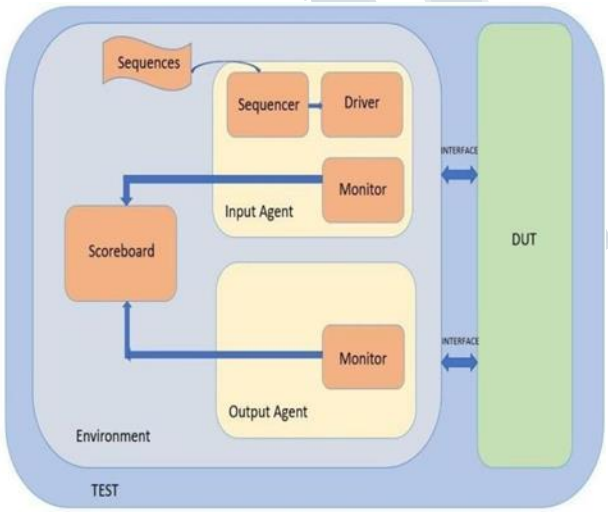

Fig.8 UVM Architecture

UVM Test:
First step is to set up the testbench
Begin the process of creating testbench components by creating the next level down in the hierarchy, for example, env.

Start the process to initiate the stimuli.
UVM Environment: The environment, sometimes known as env. It is a container module that contains agents and scoreboard.
UVM Agent:
A user-defined agent is derived from uvm agent, and uvm component inherits uvm agent.
A driver, a sequencer, and a monitor are common components of an agent.
Agents can be set to be active or passive.
UVM Driver: This component is responsible of converting base data from sequence-item to internal data (to Design Under Test).

ISSN (Online) 2395-2717

International Journal of Engineering Research in Electrical and Electronic Engineering (IJEREEE)
Vol 7, Issue 7, July 2021

UVM-Sequence: It is used to develop a series of random items. That will be passed to the driver through sequencer.
UVM-Sequencer: The sequencer is in charge of directing the data packets between the driver and sequences.
UVM-Monitor: Monitors signal activity via the design interface and convert it to transaction-level data for delivery to the scoreboard.

UVM Scoreboard: Used to compare the reference model data with the orginal data(DUT).

SIMULATION RESULT

All the five channel waveforms of AXI Protocol is shown in this section by using Questa sim tool. To conduct the simulation analysis, the different modules of the AXI Slave are first modelled in Verilog and SystemVerilog and then combined with the modelled test environment for verification, which serves as the master. The Design Under Verification (DUV) environment is developed by developing verification components such as transgenerator, driver, and reference model etc..

Read Phase Verification

The read phase contains two types of channels one is the address read channel and another is the read data channel. Initially master or manager sends the request to the slave or subordinate using the address read channel by asserting the read address signal along with the associative interface signal which are present in that channel. Address Read READY signal is asserted by the slave to indicate that slave is ready for the following transaction. Once the request is initiated by the master at the positive edge of the clock, data can be read using the read data channel by asserting Read Valid and Read Ready signals for each transfer in a transaction. And the response signal is used to acknowledge that the transfer is completed successfully. Fig.9 shows the simulation result and fig.10 shows the verification results of the read phase where the actual data which is coming from DUT is compared with the expected result.

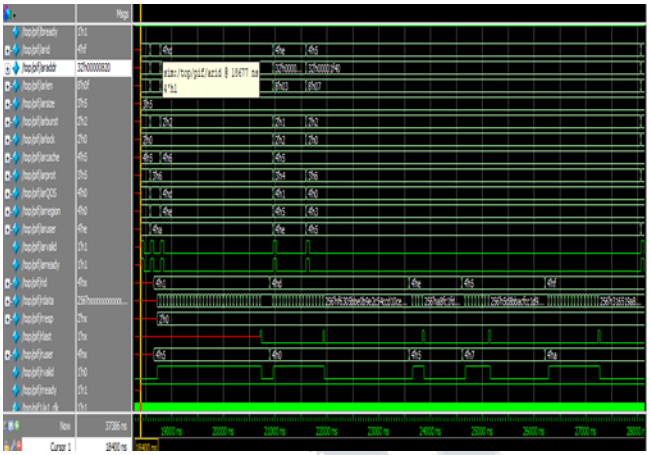

Fig.9 Read Channels with Muiltiple Outstanding Requests

Fig.9 displays the waveforms of read channels with Multiple Outstanding Requests and Out of Order transaction completion with respect to the values of Quality of Service (QoS) passed by the master.

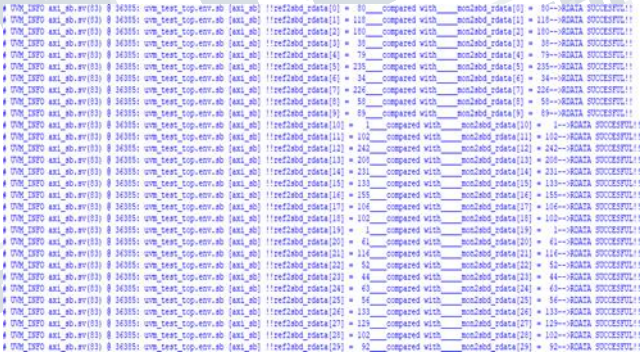

Fig.10 Verification output for Read data

Fig.11Transcript Result for Read data

ISSN (Online) 2395-2717

International Journal of Engineering Research in Electrical and Electronic Engineering (IJEREEE)
Vol 7, Issue 7, July 2021

Write Phase Verification

Fig.12 shows the simulation result of the two diferent Write channels. Here AWSIZE refers to the size of each transaction the address write channel will fetch the address at every high state value of AWVALID and AWREADY. Now AWID is an address write ID that represents an individual tag for each write address and it should match with the write data-id(WID). Then as per the request given by the master, WDATA and WSTRB signals are driven respectively and accordingly. When the last data of each burst has been driven, WLAST in the write data channel is driven high for one clock cycle. This implies it is the end of the burst and at the positive edge of the clock when WVALID and WREADY, get logic high state, the write data channel acknowledgment will take place.

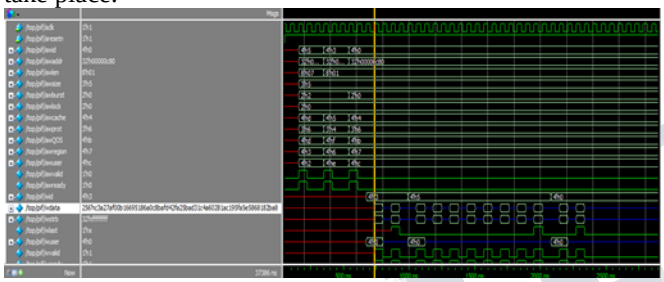

Fig.12 Write Channels with Muiltiple Outstanding Requests

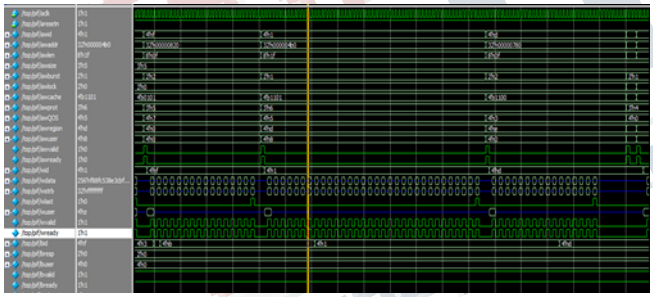

Fig.13 Response Bufferablity

AXI has a buffering feature that permits responses to be sent by intermediate components rather than waiting for a response from the end slave, which may cause the arbiter number of cycles to reach master to be delayed. Fig.13 shows the simulation result for response with buffering concept.

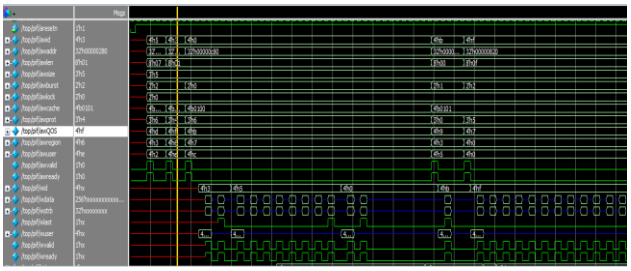

Fig.14 QoS Priortization

AXI4 helps to priortize the transactions based on the QoS value provided by the manager/master. The transaction with the high QoS value has higher priortization compared with the low QoS value. Fig.14 shows the simulation result with different QoS values for each transaction.

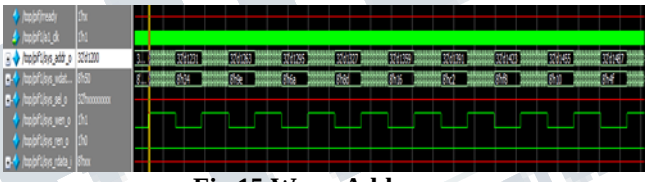

Fig.15 Wrap Address

Fig.15 shows the simulation result for address increament in case of wrap burst type.

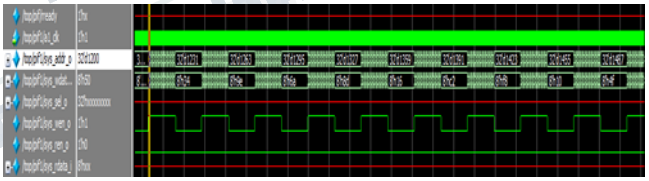

Fig.16 INCR Address

Fig.16 shows the simulation result for address increament in case of INCR burst type.

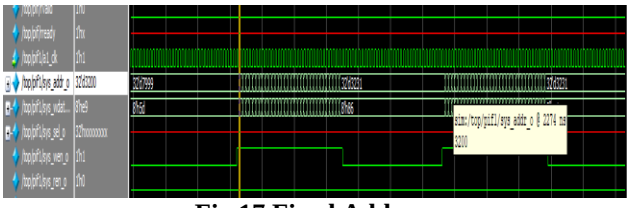

Fig.17 Fixed Address

Fig.17 shows the simulation result for address increament in case of FIXED burst type.

ISSN (Online) 2395-2717

International Journal of Engineering Research in Electrical and Electronic Engineering (IJEREEE)
Vol 7, Issue 7, July 2021

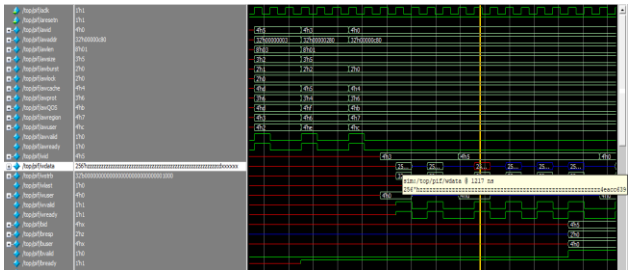

Fig.18 Unaligned Address transaction

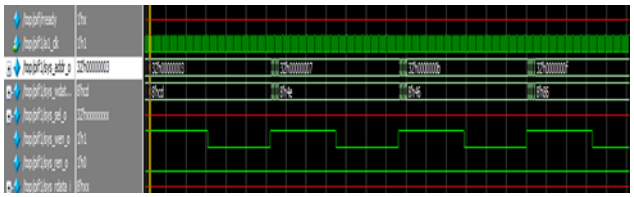

Fig.19 Unaligned Address increament

AXI-3 and AXI-4 supports Unaligned address for INCR type of burst. Fig.18 shows the simulation result for INCR burst type with unaligned address. Fig.19 shows the simulation result of how the data and address will be driven for each transfer.

Fig.20 shows the verification output for write data. Where the expected data coming from reference model is compared with the actual DUT output using scoreboard.

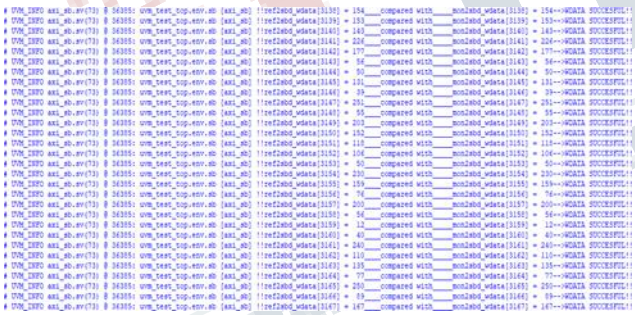

Fig.20 Verification output for Write data

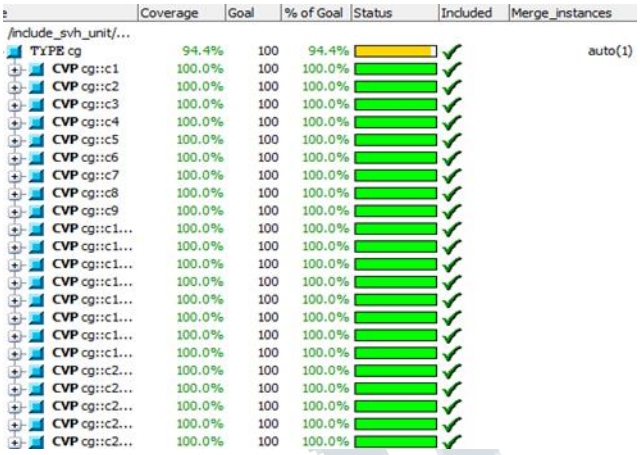

Fig.21 Coverage Report

Fig.21 shows the functional coverage report. Where 94.4% of coverage has been achived for the given DUT.

```
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :32484332
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [RNTST]     1
# [TEST_DONE]     1
# [TIMOUTSET]     1
# [UVMTOP]     1
# [axi_driver]    40
# [axi_sb] 32450112
# [uvm_test_top.env.sb] 34176
# ** Note: $finish    : F:/questasim64_10.4e/verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 37386 ns  Iteration: 54  Instance: /top
# 1
```

Fig.22 UVM Report Summary

Fig.22 shows the UVM report summary with verbosity level for the above AXI protocol.

CONCLUSION

Verification of complex designs or SoCs with Verilog or VHDL does take a lot of time to verify. So, there is a need to develop a standard verification environment that provides reusability to verify other standard IPs and consumes less time to verify the DUT. Most of the SoCs uses AXI as a common bus for on-chip communication. This research work is carried out by designing AXI Slave using Verilog and SystemVerilog and by developing Verification IP(VIP) for AXI protocol in order to verify all types of transactions with respect to writes and reads for all types of burst supported by AXI using UVM. This project also verifies some of the features of AXI which

include QoS prioritization, Bufferablity with respect to the response provided by Slave, and Unaligned address transactions for INCR type of burst. 94.4 % of functional coverage was achieved by this project. And all the components in the verification environment are developed using UVM which helps to reduce verification time.

REFERENCES

[1] AXI, ”AMBA™ Specification(Rev 5.0)”.
[2] Gayathri M, Rini Sebastian, Silpa Rose Mary, Anoop Thomas,” A SVUVM framework for Verification of SGMII IP Core with reusable AXI to WB Bridge UVC”, 3rd International Conference on Advanced Computing and Communication Systems (ICACCS), Jan. 22 , 2016
[3] Nishit Gupta, Sunil Alag,” NTRP: Novel Approach for DUT Testing based on Nonintrusive Timing Randomization Probes using SystemC Verification Library”, International Conference on Information Technology(InCiTe)-The Next Generation IT Summit,2016
[4] AXI, ”AMBA CHI Specification”.
[5] AXI, ”AMBA AXI™ Specification(Rev 5.0)”.
[6] “Mentor Graphics UVM Cookbook”
[7] lasencia-Balabarca, Edward Mitacc-Meza, Mario Raffo-Jara and Carlos SilvaCárdenas,” A Flexible UVM-Based Verification Framework Reusable with Avalon, AHB, AXI and Wishbone Bus Interfaces for an AES Encryption Module”, International Journal of VLSI design & Communication Systems (VLSICS) Vol.3, No.2. 2012.