Dilip K, et. al. International Journal of Engineering Research and Applications
www.ijera.com
ISSN: 2248-9622, Vol. 11, Issue 7, (Series-V) July 2021, pp. 32-41

RESEARCH ARTICLE OPEN ACCESS

Design and Verification of AHB Protocol Using System Verilog and Universal Verification Methodology (UVM)

Dilip K*, Vijaya Prakash A M*
** M.Tech, Department of ECE, Bangalore Institute of Technology, Karnataka, India*
** Professor Department of ECE, Bangalore Institute of Technology, Karnataka, India*

ABSTRACT :
Recently, VLSI technology has improved significantly and more transistors can be incorporated in a chip. A System on-Chip (SOC) Configuration have number of blocks are integrated on a single chip. Numerous blocks are integrated in single IC, but to access their function, they requires a powerful communication architecture. This can only be achieved by using on-chip bus architecture to meet their requirements. Different Companies has various on-Chip Bus architectures but one of the most suitable architecture is AMBA by ARM. AMBA consist of three buses, namely, Advanced System Bus (ASB), Advanced Peripheral Bus (APB) and Advanced High Performance Bus (AHB).when compared to other two buses AHB is high performance, high bandwidth and for high clock frequency system modules the System designers select AHB as their primary choice. The AHB (Advanced High-performance Bus) is a superior bus in AMBA (Advanced Microcontroller Bus Architecture) family. It is a norm for intercommunication of modules in a framework. The AHB (Advance High performance) bus Standards are characterized by ARM which supports for the communication of on-chip memories, processors and interfaces of external off-chip memory. Here the basic blocks such as master, slave, decoder, and arbiter are used to design and verify an AHB that supports multiples master and multiples slave. The conventional way of verification is simulation based. As the Technology improves the complexity of IC's has been increased. Thus, time spent on verification has also been increased. The main focus is to design of AHB protocol in Verilog and verify using Hardware verification language such as System Verilog and standard Methodology such as Universal Verification Methodology (UVM). QuestaSim (Advanced verification tool from Mentor Graphics) is an EDA tool used to simulate and verify the design and obtain Coverage report.
Keywords – AMBA, AHB, APB, ASB, OCB, SOC, UVM

Date of Submission: 10-07-2021 Date of Acceptance: 26-07-2021

I. INTRODUCTION

1.1 About AMBA bus

The AMBA is an Advanced Microcontroller Bus Architecture defined by ARM, it is an open standard widely used for an on-chip bus system. The standard is intends to simplify the component design by allowing the use of interchangeable parts the within the SoC style. It promotes the use of holding parts, so that a minimum of a neighborhood of the SoC can be reconstructed, instead of having to rewrite it entirely each time. AHB (Advanced High-performance Bus), ASB (Advanced System Bus), and APB (Advanced Peripheral Bus) are the bus groups defined in the AMBA AHB. The AHB is employed for high-performance, high frequency architecture. These applications includes are ARM cores and high-speed RAM inside the system, Nand Flash, DMA and Bridge links. [1] The APB is used for connecting external devices such as UART, keypad and timer, and has low performance requirements, while it is used for optimizing power consumption. AMBA is the Standard bus-based microcontroller typical feature a high-performance system hub bus (AHB or ASB) that supports for external memory bandwidth, including CPUs, on-chip memories, and other direct data access (DMA) devices. For most of the transmission between various units, such as CPUs, on-chip memories, and DMA, the bus serves as a high bandwidth interface.

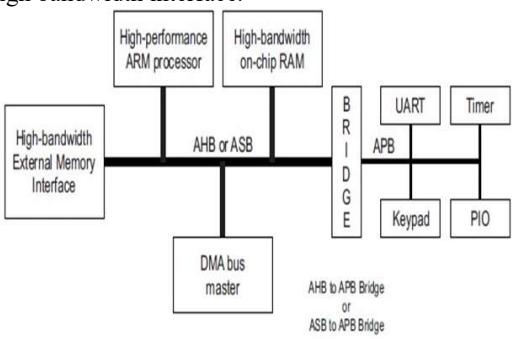

Fig.1: AMBA bus Block Diagram

1.2 Advance System Bus (ASB)

The ASBs (Advanced system buses) are used for defining high performance buses that can be used in embedded microcontrollers with 16-bit and 32-bit architecture. An ASB provides a high performance pipelined bus that can provide access to multiple masters. The Flow of essential operations of ASB is:

- Master communicates with the bus.
- Arbiter observes master's status.
- Then, master begins communicating with the bus.
- The decoder uses the accurate address lines to choose a bus slave.
- Then, a signal is given back to the bus master by the slave.

1.3 Advance Peripheral Bus (APB)

The Advanced Peripheral Bus (APB) is utilized for connecting low bandwidth peripherals. APB is a simple non-pipelined protocol that can be utilized to communicate from a master to a multiple slaves for read and write through the shared bus. [5] The read and write bus shares the same set of signals and no burst data transfers are supported.

1.4 Advance High Performance Bus (AHB)

An AHB bus is a latest generation of AMBA bus that is intended to handle the necessities addresses the prerequisites of high performance synthesizable design styles. It is a Standard system bus that supports multiple bus masters and provides high-bandwidth operation. AMBA AHB implements the features needed for standard, high clock frequency systems including:

- Burst transfers.
- Single-cycle bus master relinquishing.
- Single-clock edge operation.
- Non-tristate implementation.
- Wider information bus configurations (32/128 bits).

The AMBA AHB bus protocol is designed using a central multiplexer interconnection design. By design, all bus masters transmit address and control signals indicating the data transfer they want to perform and the arbiter determines which master has its address and control signal based on that information it is provided to all of the slaves. The decoder is used to control the read data and response signal from multiplexer, which chooses the proper signals from the slave that is engaged in the transfer.

II. LITERATURE REVIEW

Design and verification of AMBA AHB bus which consist of one master and multiple slave designed in Verilog Hardware descriptive language and shown the output for read and write operation. [1] The Design under test is verified using the system Verilog environment and obtained the coverage report around 65%.This paper tells us the coverage report obtained is less. The Questa sim is the EDA tool used to obtain the simulation output.

The paper presents a Method for Designing an efficient[2] master interface and slave interface based on the finite state machines in Verilog hardware description language and used the Mentor graphics tool Model sim 10.03a to simulate and the synthesis of the design is performed in Xilinx ISE design tool. The completed AMBA AHB system is then inspected for proper lossless communication between master and slave interface. This article tells us does not used the verification language such as system Verilog.

In this the efficient design of an AMBA controller is designed and tested [3] for read and write operations using a Xilinx simulator. The read and write operations using AMBA are illustrated with simple examples.

As reported in paper [4], The AHB master interface and arbiter interface are designed using the finite state machines in Verilog hardware description language and the design is simulated with the help of Questa Sim. AMBA AHB system is then tested to ensure that the master and the slave interfaces communicate in the lossless manner

The above review tell about the most of the related work is on the Verilog hardware descriptive language and the more work to be done in the verification. The present work use the verification language such as system Verilog and standard methodology such as UVM (universal verification methodology).

III. DESIGN METHODOLOGY

The Design under Test (DUT) Block diagram consist of multiple masters and slaves. The masters are namely m1, m2, and m3 and slaves are namely s1, s2, and s3. The master as address and control signals such as HADDR, HTRANS, HBURST and HSIZE. Slave as address and control signal such as HADDR, HREADY and HRESP. The block diagram of DUT as the blocks such as master, slave, decoder and arbiter. The Arbiter consist of both decoder and multiplexer.

Initially the master as the data_1, data_2, and data_3 each masters as the data and it is driven by address and control signals. At the output of master block the HWDATA_tb1, HWDATA_tb2 and HWDATA_tb3 are the write data transactions obtained when wr=1; then the write data is send to multiplexer based on the address on the decoder. The arbiter perform the operation such that it does not allows the masters to send the data to same slave at a particular time for the particular address it

allows master m1 as the write data HWDATA_tb1 is send to slave i.e. HSEL1, then the slave1 read the data, rddata1 and send the response signal HRESP1 to the m1 through slave multiplexer based on the address on the decoder that it complete the transaction of data when m1 receive the response.

For the next particular transaction the initial three master m1, m2 and m3 as the data_1, data_2 and data_3 are send and obtained the HWDATA_tb1, HWDATA_tb2 and HWDATA_tb3 as the write data transactions, now the arbiter is allows the m1 and m2 write data to transfer and the write data is read by the slaves i.e HSEL1 and HSEL2 when the wr=0 and obtained the output rddata1, rddata2, and the slave as HRESP signal which goes high and additional HREADY signal indicates that the transaction of rddata1 and rddata2 are completed and ready to receive the other data.

For the next transaction the initial three master m1, m2 and m3 as the data1, data2 and data3 are send and obtained the HWDATA_tb1 ,HWDATA_tb2 and HWDATA_tb3 as the write data transactions, now the arbiter allows the m1 , m2 and m3 write data to transfer and the transaction write data is read by the slaves i.e HSEL1 , HSEL2 and HSEL3 when the wr=0; and obtained the output rddata1 , rddata2 and rddata3, and the slave as HREADY signal indicates that the transaction of rddata1 , rddata2 and rddata3 are completed and ready to receive the other data from the masters.

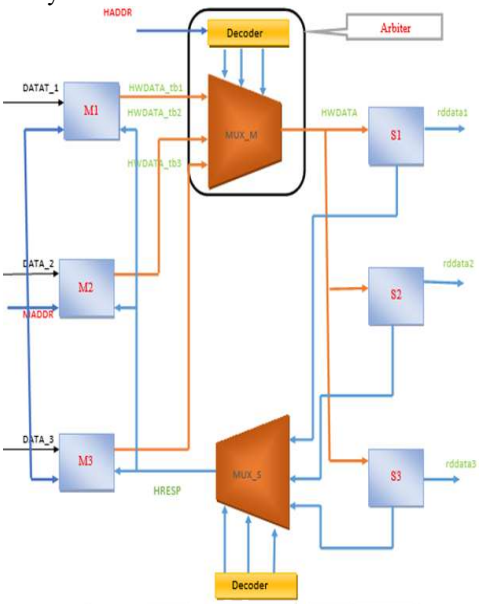

Fig.2: AHB multi master and slave DUT

3.1 Components of AHB

The AHB bus as four components namely

3.1.1 AHB Master
3.1.2 AHB Slave
3.1.3 AHB Arbiter
3.1.4 AHB decoder

3.1.1 AHB Master

The AHB bus master initiate the read/write operations by providing address and control information. The maximum of 16 masters are allowed in our design we are using 3 masters.

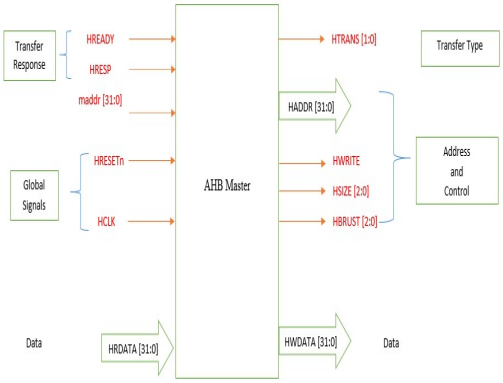

Fig.3: AHB Master

- HRESETn: The bus reset signal it is used to reset the system and the bus.
- HCLK: The clk is used for all bus transfers.
- HADDR [31:0]: The 32-bit address bus.
- HTRANS [1:0]: Indicates the type of transfer such as idle, busy, sequential and non-Sequential.

HTRANS	Type	Description
00	Idle	Master uses the idle transfer when it does not wish to perform the data transfer the slave must ignore the data by sending the OKAY response to the master
01	Busy	The busy transfer type used to insert the idle cycle in between the burst and the master address and control reflect in the next burst transfer the slave must provide the zero OKAY state response to the master

11	Non sequential	The address and control signal is not related to previous transfer and it of single burst type or fixed type burst
10	sequential	The remaining transfer of burst is related to sequential the address and control signal is related to previous and based on the HSIZE[2:0] and burst operation such as wrap and increment

Table.1: HTRANS Types

- HWRITE: when this signal is low indicate the read Transfer and when it is high indicates the write transfer.
- HSIZE [2:0]: indicate the size of transfer such as Byte (8-bit), half-word (16-bit), word (32-bit) etc.

HSIZE [3]	HSIZE [2]	HSIZE [0]	Size (bytes)	Description
0	0	0	1	Byte
0	0	1	2	Half word
0	1	0	4	word

Table.2: HSIZE type

- HBRUST [3:0]: Indicates Type of burst operation such as fixed type, increment type and 4-beat wrap and 4-beat increment type [6]

.

HBURST[2:0]	Type	Description
b000	Fixed/Single	Fixed burst
b001	Single INCR	Increment type
b010	WRAP4	4-beat wrapping burst
b011	INCR	4-beat Increment burst

Table.3: BURST type

- HWDATA [31:0]: The write data bus which is used to transfer the data from master to the slave during write operation.

3.1.2 AHB Slave

The AHB slave response to the transfer given by the master and the decoder is used to select the slave based on the HSELx signal it is the slave signal used to select the slaves and the response is send to the master based on HRESP. There are two types of response given to the master if the data is successfully read by the slave it give OKAY response for the slave if it unsuccessful it provide ERROR response.

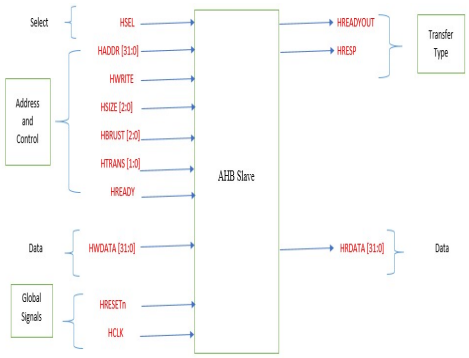

Fig.4: AHB Slave

- HRDATA [31:0]: The read data bus which is used to read the data from the master when HWRITE is low.
- HREADY: when the HREADY signal is high indicates theat the transfer is finished by the slave
- HRESP : it is response given to the if the response is zero then the slave as completed the transfer of data and indictes okay signal and if it is one indicates error in the transfer.
- HREADYOUT: output of slave indicate the status of transfer.

3.1.3 AHB Arbiter

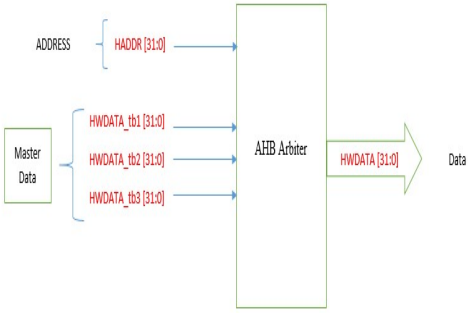

Fig.5: AHB Arbiter

The arbiter controls the three master write data such as HWDATA_tb1, HWDATA_tb2 and HWDATA_tb.It operation is to control the master data such way that different master data should not Send to the single slave.The arbiter work is to filter or control the write data of master such a way that it as to send the write data to a particular Slave.

3.1.4 AHB Decoder

The below Figure 6 shows the AHB decoder The AHB slave has its own slave select signals i.e HSEL1, HSEL2, HSEL3 and this signal indicates that the transfer is particularly for the selected slave. This slave select signal is selected based on the address of the decoder .Then the transfer data is read by the slave and send the response to the particular master and ready signal goes high for that transfer of data. If the response is low indicates okay transfer. If the responsre is high indicates errror transfer.

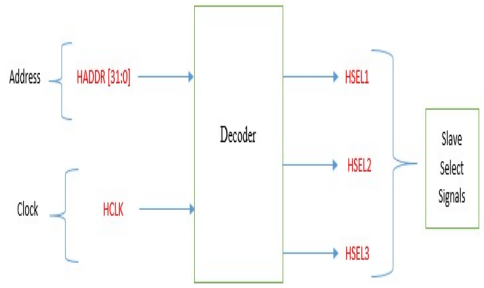

Fig.6: AHB Decoder

IV. VERIFICATION METHODOLOGY

Verification is the significant part in the VLSI technology. Since it is used to find out the bugs in the RTL design at the initial stage so the overall Design should not prove any error. Here we creating a System Verilog Environment and Universal verification methodology (UVM) Environment for an AHB design. The main intention of creating verification environment is to generate the stimulus to the Design under test (DUT), and check the results to verify that the DUT function is correct.

4.1 System Verilog Environment

System Verilog is a Hardware Description language (HDL) and Hardware Verification Language (HVL) based on Verilog. While it has few features to help with design, the purpose of language is to verify of electronic designs. [7] Open Vera, a language denoted by Synopsys, provides the majority of verification functionality. System Verilog as the IEEE standard P1800-2005.

System Verilog is a special hardware verification language and Hardware verification language intended to be used in function verification. It is used to provide the high level data Structures accessible in object-oriented languages, such as C++. The data structures empower a higher level of abstraction and modeling of complex data types. The System Verilog also as constructs necessary for modeling hardware concepts such as cycles, tri-state values, wires, same like Verilog hardware languages. System Verilog can be used to simulate and verify the Verilog HDL design by applying the high level of test input as it is known to be Hardware verification language (HVL). The system Verilog as the test bench architecture which consist of component such as basepkt, generator, driver, monitor and scoreboard.

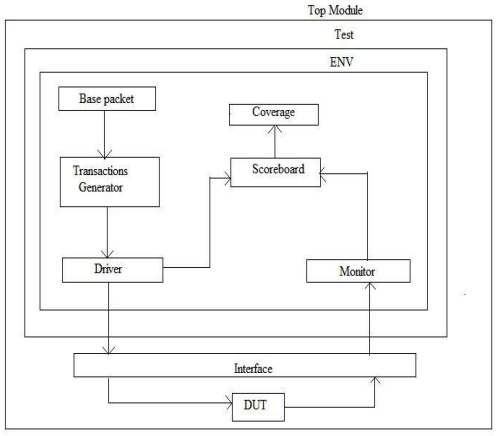

Fig.7: System Verilog Environment

The above Figure 7 shown is a System Verilog environment. The Environment includes Design under test (DUT) written in Verilog Hardware descriptive language (HDL) and System Verilog test bench that includes System Verilog interface, simulation module and test program. In system Verilog test bench, the basepkt contains the all input and output are send as a packet through mailbox to generator. The generator is utilized to create constrained random test vectors. These vectors are sent to the driver, and then through interface can simulate the DUT. A monitor generates verification reports for each state, transaction, and model message. Using the Scoreboard to checks the results of the driven and the monitor driving them through mailboxes any changes in the modifications that need to be done. The development of coverage class based on the coverage plane and we will apply the test cases and analyze the code coverage report. System Verilog advantage is to use of object oriented programming, which enables the reusability of test bench components. The interface is used to combine the DUT and the System Verilog test bench which includes the test program.

4.2 Universal Verification Methodology (UVM)

System verilog is a Hardware descriptive and Hardware verification language just like verilog and has its particular constructs, syntax and features. But universal verification methodology is a structure of system verilog classes from which we can built fully functional test benches. The RTL (Register transfer level) design is verified using the Standard

Methodology such as Universal Verification Methodology. It comprises of base class library coded in System Verilog. [7] The verification engineer can make different verification components by extending these classes. Additionally, UVM provides several useful verification features such as utilizing of macros for implementing complex functions and factories for creating the objects. The below Figure 8 is a UVM environment. The environment includes interface and DUT with test bench. The test bench environment consist of agent, sequencer, and driver and monitor as sub components.

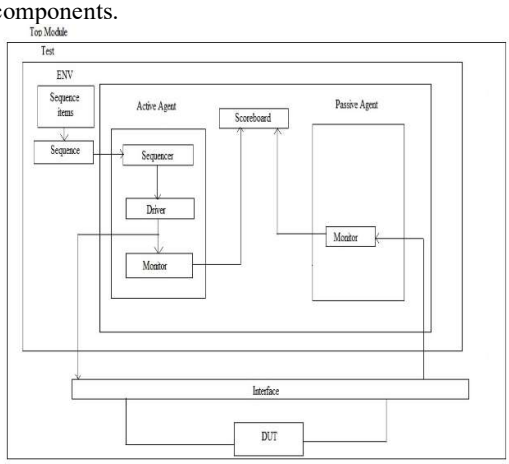

Fig.8: UVM Environment

Sequence item: Using the uvm_sequence_item class, the transactions are extended to send the randomized data to a driver to be driven onto the bus. The field automation macros are applied to these members of the class as well.

Sequences: A sequence is a bulk of transaction. In the sequence class, the users can make complex stimulus. This sequences can be randomized, extended to make another sequence and can be combined.

Sequencer: The data flow between Sequences and driver is signaled by the sequencer verification component. The sequencer has a collection of sequences combined with it called a sequence library. The collection of sequences utilized by a sequencer is called sequence library. This type of component is also known to as a driver sequencer.

Driver: Driver collect the object from the sequencer and drives it to the next lower level such as DUT (Design under Test) through the interface. It is generated by extending the uvm_driver.

Monitor: monitor samples the DUT signals through the Virtual interface signals are converts into packet level which is then sent to other components, such as scoreboards for the analysis. It was generated by extending the uvm_monitor.

Agent: The agent consist the verification components such as driver, monitor, collector and sequencer. It used to connect these components using TLM connections. The agent as one of the operating modes active or passive. In the active mode of operation, the agent initiate driver, sequencer and monitor where as in the passive mode of operation initiate only monitor and configured

Environment: The Environment class consist all the sub components such as agents, driver and monitor etc. and configures them.

Testbench: The uvm_test class defines the test cases for the test bench specified in the test. The Different test cases is applied to enable the configuration of the test bench and verification components. The uvm_test is written by extended from the uvm_component.

V. RESULTS AND DISCUSSION

5.1 Master1 to Slave1 data transfer

The Figure 9 show below shows the simulation output for master1 to slave1 data transfer. When reset is high the master data is send to the input of the arbiter. The arbiter comprises of both decoder and multiplexer based on that arbiter filters the other master write data and allows for the master1 to send the data to the slave1 (HSEL1). The slave1 read the data and obtain the output rddata1. When slave1 complete the transfer of data. The response signal goes low indicating to master that the slave as completed the transfer of data and it is ready to accept the other data by indication the HREADY signal high.

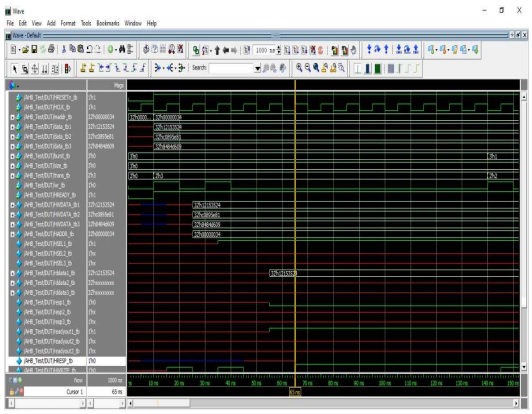

Fig.9: Master1 to Slave1 data transfer

5.2 Multi master to Multi slave data transfer

When reset is high the master data is send to the input of the arbiter. The arbiter consist of both decoder and multiplexer based on that the arbiter filter it allows the masters to transfer the data to the

particular slaves the Figure 10 below shows simulation result for activation of multi slaves.

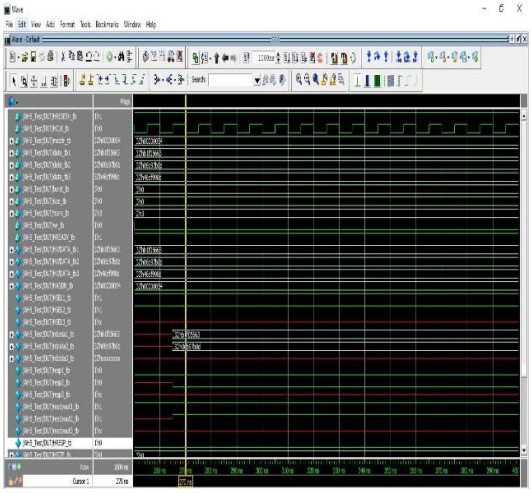

Fig.10: Simulation Result for Activation of Multiple Slaves

Figure 11 shown below shows the simulation result for data transfer from multiple master to multiple slave the arbiter allows the multiple master to transfer the data to multiple slave it control the master write data to send for the particular slave without any mismatching of data. The control and address related to slave will sends the signal to the master after transferring of data.

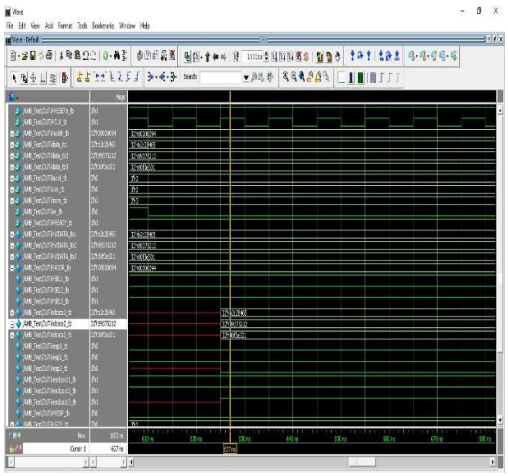

Fig.11: simulation result for multi master and multi slave

5.3 Simulation result for burst Operation

Figure 12 shows the wrap 4 burst operation (HBRUST=b010). The address bus get wrapped when it reach it boundary size. The wrap4 burst operation depends on the HSIZE (010=2). i.e. the wrap4 as 4-beat each beat depend on the size of transfer below figure we taken the size type word i.e. the word as 4 bytes now 1-beat takes 4 bytes for 4-beat it takes 16-bytes of data and address is wrapped when it reached maximum. Initial transfer of burst as transfer type non sequential (11) is given for stating single burst later for next transfer sequential (10) is related to the previous address and next burst operation continuous sequentially when it each address boundary again the address get wrapped to initial boundary.

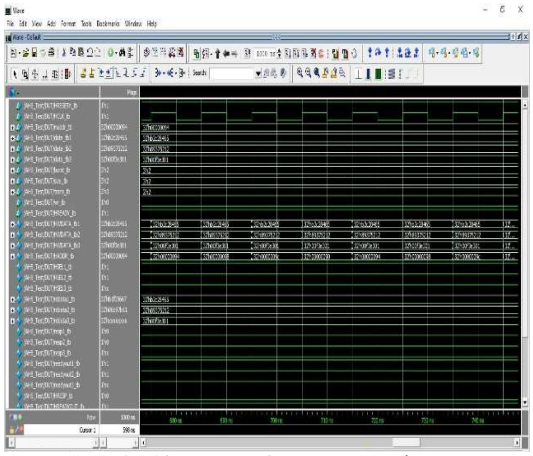

Fig.12: WRAP4 Burst operation

Figure 13 shown below shows the increment type of burst operation (HBRUST=b011) the incr4 is type of burst operation with the HSIZE (010) word size of 4 bytes and the address is increment gradually based on the size of the bytes and each address as the data is stored in the particular address the data are stored in the each address. The figure shows multi master and multi slave increment 4 type of burst operation.

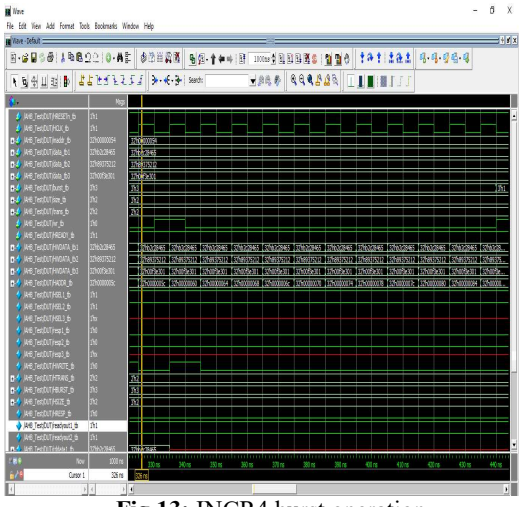

Fig.13: INCR4 burst operation

5.4 System Verilog top module waveform

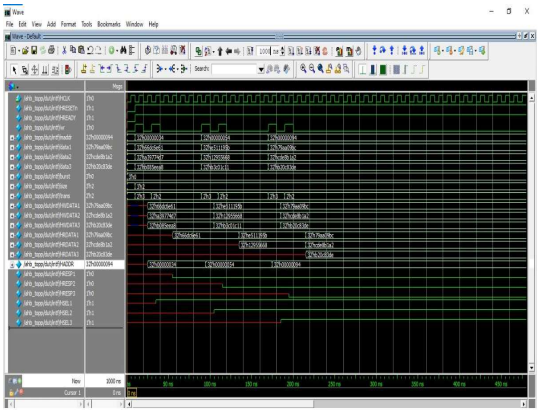

Fig.14: System Verilog Test bench Environment

Figure 14 shows the system Verilog top module waveform for multiple master and multiple slave.

5.5 Code coverage Report

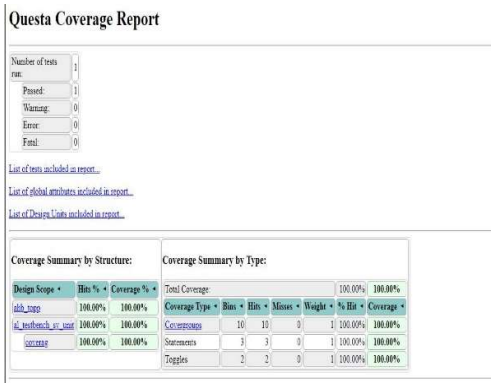

Fig.15: Coverage report

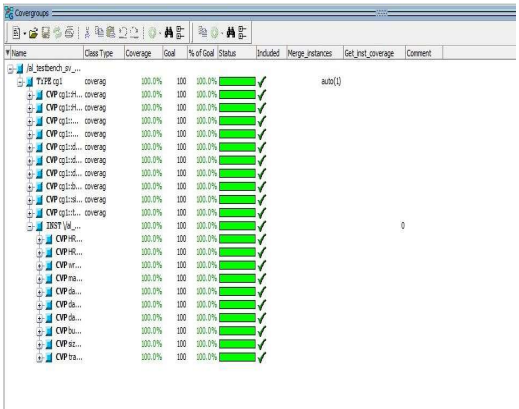

Fig.16: cover point analysis report

Figure 15 and 16 shows the coverage report and Cover point analysis Report. The coverage report gives information about statement, Toggle and coverage group it includes both code coverage and Function coverage. The cover point analysis report it is basically a functional coverage report. The bins are created based on the constraints. And applied the test cases if they met then that functionality is said to be hit or covered. Here the overall coverage report obtained is 100%.

5.6 UVM top module waveform

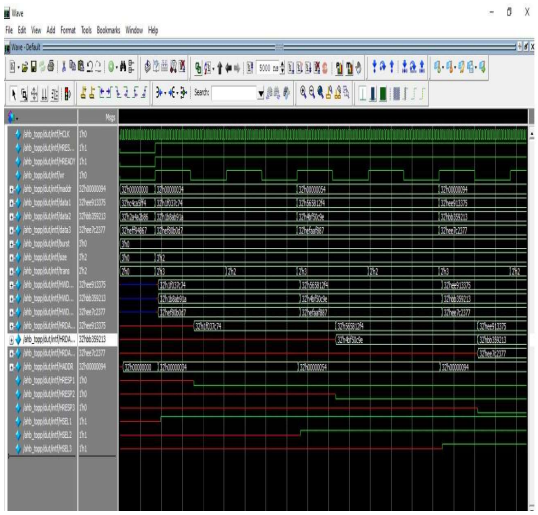

Fig.17: UVM testbench environment

Figure 17 shows the UVM testbech Waveform for multiple master and slaves.

5.7 UVM report summary

```
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :  162
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [RNTST]     1
# [TEST_DONE]     1
# [UVMTOP]     1
# [uvm_test_top.ENV.sb]   159
# ** Note: $finish    : C:/questasim64_10.4e/verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 3200 ns  Iteration: 54  Instance: /ahb_topp
```

Fig.18: UVM report summary

UVM report gives information about the result obtained after Simulation of UVM test bench. Figure 18 shown below the UVM report generated after passing all the UVM phases. UVM_INFO in the report conclude that there are 162 information messages. The UVM report Summary explain the design as no errors and it does not have fatal error.

Because UVM_ERROR, UVM_WARNING and UVM_FATAL is equal to 0.

VI. CONCLUSION AND FUTURE SCOPE

The paper gives an Overview of AMBA bus Architecture and discussed the AHB protocol. The AHB bus is designed which supports for multiple masters and slaves. Which consist of basic blocks such as master, slave, decoder and arbiter. The AHB design Block is designed using Verilog HDL and verified using the system Verilog and Universal verification methodology. The Tool used is Questa sim is an EDA tool used to simulate and verify the design and obtained the coverage report. Which says that the functionality is correct. The UVM report summary also ensure that functional Correctness of the design. In the present work we have designed AHB which support for 3 master and 3 slaves. Developing the design which can support for 16 master and 16 slaves could be the future work of this project design.

REFERENCES

[1]. Dr. Priyanka Choudhury, Perrumalla Giridhar "Design and Verification of AMBA AHB", 1st International Conference on the Advanced Technology in Intelligent Control, Environment, Computing & Communication Engineering (ICATIECE) in IEEE 2019.

[2]. P.Harishankar, Mr. Chosen Duari Mr.Ajay Sharma, "Design and Synthesis of Efficient FSM for Master and Slave Interface in AMBA AHB", International Journal of Engineering Development and Research", IJEDR, Volume 2, Issue 3, ISSN: 2321-9939, 2014.

[3]. Shivakumar B.R Deeksha L, "Efficient Design and Implementation of AMBA AHB Bus Protocol using Verilog", International Conference on Intelligent Sustainable System, (ICISS) in IEEE, 2019.

[4]. Mr. M. Naresh Kumar, K.Manikanta Sai Kishore "Design and Implementation of Efficient FSM for AHB Master and Arbiter", International Journal and magazines of Engineering Technology, Management and Research ISSN No. 2348-4845, 2015.

[5]. Shraddha divekar, Archana Tiwari "Multichannel AMBA AHB with Multiple Arbitration Technique", International Conference on Signal Processing and Communication, Apr 3-5, 2014.

[6]. Jayapraveen and T.G. Priya "Design of Memory Controller based on AMBA AHB protocol", "Elixir International Journal 51A, Vol.2, pp. 11115-11119, 2012.

[7]. Chrisspear, System Verilog for Verification, New York: Springer, Rath A.W, Esen. V and Ecker.W, A transaction Oriented UVM-based Library for Verification of analog behavior Publication year: 2014 pages(s):806-811.

[8]. Divya M, Dr. K. A. Radhakrishna Rao "AHB Design and Verification AMBA 2.0 using System Verilog", in IJARIIT Volume-4, Issue-3, 2018.