System Verilog Assertion Based Design and Verification of AHB Lite Protocol

[1]Monisha H M,[2]Meghashree A C
[1]Student,[2]Asst Professor
[1]Digital Electronics and Communication, M.Tech,
[1]VTU Post Graduation Center, Mysore, India

Abstract: Nowadays, Increasing technology increases the amount of logic that can be placed in a silicon chip driving highly integrated SoC design development. Verification is the important part of SoC manufacturing, it gives particular implementation and functionality to DUT for check whether it has achieved specification or not. In this Paper, AHB-Lite protocol subset of AHB which is a part of AMBA (Advanced Microprocessor Bus Architecture), is verified using Hardware Verification Language System Verilog Assertion (SVA). The AHB-Lite is the on-chip-interconnect which manages the functional block in SoC design and finds the presence of error/bug in the design by generating simulation result, coverage report and assertion report using QUESTA SIM tool.

Index Terms **- SoC, AHB Lite, SVA (System Verilog Assertion), coverage and assertion.**

I. INTRODUCTION

Technology evolution, in part, enabled the transition of multi-million gate designs from large printed circuit boards to SoC(System On Chip). The increased complexities of SoCs have led to significant increase in the verification efforts that are imperative to meet the time to market demands[4].Verification is a process of checking whether the interaction of protocol entities satisfy certain properties or conditions. Here functionality of AHB-Lite interconnect in the SoC design is verified.AMBA3 AHB Lite is the subset of AMBA2 AHB used in such SoC design where only one master ,one slave or multiple slaves are required like cortex M. In this paper we have design the AHB-Lite protocol using system verilog coding language. Then this DUT is verified using system verilog assertion and inserted into TB(test bench) to simulate and justify the functional correctness of the SoC interconnect. The simulation result, 100%coverage report and 100%assertion report using system verilog are given.

II. AMBA AHB-LITE INTERCONNECT

AHB-Lite [7]is a subset of the full AHB specification and is intended for use in designs where only a single bus master is used. This can either be a simple single master system or a multi-layer AHB system where there is only one AHB master per layer. The AHB-Lite specification features are:

- It is a single master system, there is only one source of address, control and write data, so no master to slave multiplexor is required.
- Supports the pipelined operation, burst transfer.
- Separates bus for read and write transfer.
- There is no arbitration. The AHB-Lite master always has control of bus.
- There is no SPLIT or RETRY slave responses. The AHB-Lite deals only with a slave ERROR responses.
- The AHB-Lite has no request phase.

Designing of AHB-Lite interconnect requires the following components:

1. Master
2. Slave
3. Address decoder
4. Multiplexer

The block diagram for AHB LITE interconnects is shown in Figure 1, with four components i.e. single master, three slaves, address decoder and slave to master multiplexer. The bus interconnect logic between master to slave consists of one address decoder and multiplexer.

The AHB LITE interconnect supports data bus configuration from 8-bit to 1024-bits but microcontroller
Supports up to 32-bits data configuration. The 32-bits Address and Control signal sent out of AHB LITE master goes to AHB LITE slave as well as address decoder. Decoder calcifies the transaction to particular AHB LITE slave depending upon the address sent from the AHB-LITE master. Same information is sent to the multiplexer and it will know that which response and data signal multiplexer to the AHB LITE master. At single transfer only one slave will accept the information and other two remains in de-active state.

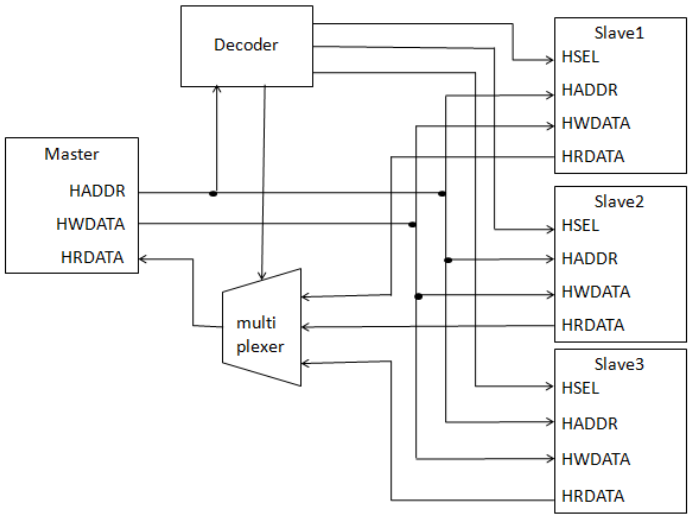


Figure1: Block diagram of AHB-Lite protocol

AHB LITE Master:

The AHB-Lite master provides the address and control information to initiate read and write operation Figure 2.shows the AHB-Lite master interface.

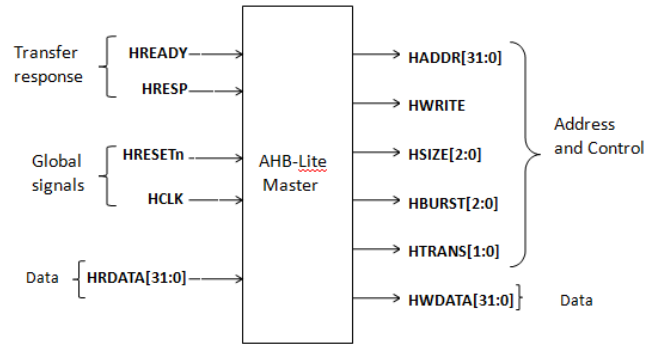


Figure2:AHB-Lite master interfaces

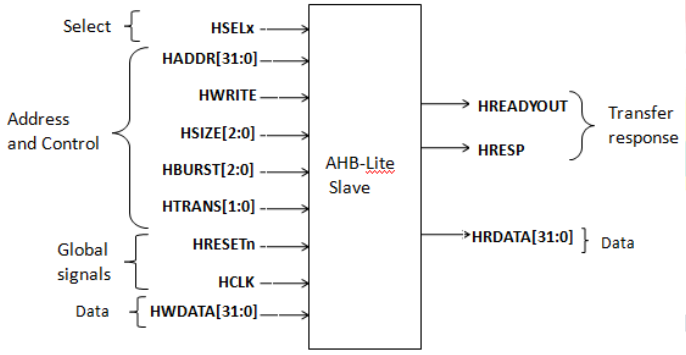


Figure3:AHB-Lite slave interfaces

AHB LITE Slave:

An AHB-Lite slave responds to transfers initiated by masters in the system. The slave uses the HSELx select signal from the decoder when it responds to a bus transfer.Figure3:shows the AHB-Lite slave interface.

AHB LITE Decoder:

This component decodes the address of each transfer and provides a select signal for the slave that is involved in the transfer. It also provides a control signal to multiplexer. A single centralised decoder is required in all AHB-Lite implementation that use two or more slaves.

AHB LITE Multiplexer:

A slave to master multiplexer is required to multiplex the read data bus and response signal from the slave to master. The decoder provides control for the multiplexer. A single centralised multiplexer is required in all AHB-Lite implementation that use two or more slaves.

III. LITERATURE SURVEY

This section describes the previous work done on verification of protocol.

Rishab Singh, Kumari, Shruthi Bhargava and Ajay Somkuwar[1] proposed the AMBA AHB protocol design which acts as an interface between two different IP cores.its working is identified based on signal flow diagram and specifications using VHDL. For design and implementation of a flexible arbiter scheme for the AHB busmatrix based on burst operation.

Sravya Kante, Hari Kishore Kakarla and Avinash Yadlapati [2].represents the definition of AMBA AHB-Lite protocol. Explains about major AHB-Lite transfers like single transfer and the sequential transfers have been verified with individual test cases using Verilog HDL.

Pallavi T lambe and Meghana Kulkarni proposed verification analysis of AHB-Lite protocol using system verilog language. This verification is based on the coverage program included in the verification environment. complexity was found to code verification environment using verilog coding language[3].

In general, all existing papers discussed above includes some disadvantages. So this paper introduces a new type and new way for verification of Protocol.

IV. PROPOSED VERIFICATION ENVIRONMENT

To know about verification environment of AHB-Lite interconnect, the test bench architecture must be known in the broader view. The Figure 4 shows block diagram of AHB-Lite interconnects.

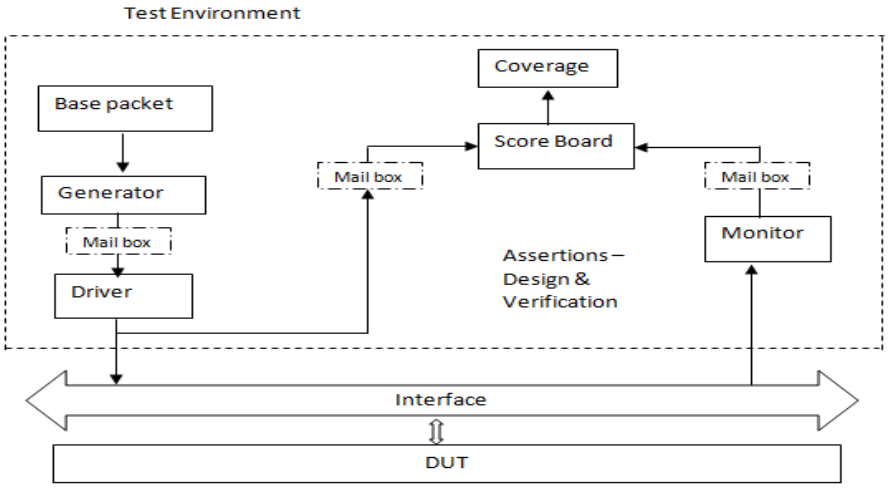


Fig4: Block Diagram of AHB-Lite interconnect verification environment

4.1Base Packet

It consists of input and output ports involved in the environment which can not interchangeable once set. It is a 2 state data types are used in this module.

4.2 Generator

Generates the stimulus ie, creates and randomize the transaction class(test cases) , and send it to the driver. The system verilog provide construct to manage random generator order and distribution.

4.3 Driver

It receives the stimulus from the generator and drives the packet level data inside the transaction into pin level ie,DUT. It achieves back response from design under test and again converts into packet level data.

4.4 Interface and DUT

Interface allows same subprograms to operate on different portions of a design and to dynamically control the set of signals associated with the subprogram. Instead of referring to the actual set of signals directly, we can able to manipulate a set of signals.

DUT(Design Under Test),is a micro architecture of a protocol functionality and implementation. It is a simple view of protocol design to look at the data transfer function.DUT interacts with external blocks through interfaces. Interfaces in the DUT are in the form of buses.

4.5 Monitor

It just monitors the bus traffic of the interface signals on the AHB-Lite protocol. Monitor checks the AHB-Lite protocol is being followed by the ongoing transaction using assertions. And the results are given to the scoreboard.

4.6 Scoreboard

It is also know as a tracker , consists of the reference model in which the expected output is stored. As random stimuli from generator is sent to DUT and the same stimuli are sent to scoreboard and saved into DUT till it gives the output. Then the scoreboard compares the actual output and the golden output and gives the result of the verification.

4.7 Mailboxes and Coverage

Mailboxes are used just to store the outputs from each stage. The data can be pushed and popped anytime from the mailboxes in various levels of the verification.

Coverage are used to check the design under certain rules. In this paper we have used the functional coverage methodology to evaluate the design.

V. RESULTS AND DISCUSSION

Using hardware language ie, system verilog language AHB-Lite protocol is designed. Considering some functionalities like basic read operation, write operation and burst operation which include wrapping of 4&8 bytes. Then the using verification environment the design is verified based on coverage and assertion percentage. The test cases are used to verify the AHB-Lite interconnect.

5.1 Basic Read and Write Operation

When HRESETn is high, the whole design or a system is in active state. Read and write operation depend upon two factor they are HTRANS and HWRITE. When HWRITE is '1', write operation is done and when HWRITE is '0',read operation is done. When HREADY signal is high, master drives 32bit address & data and slave accept it after one clock cycle. The write and read transfer will not continue in same clock cycle. The HTRANS consists of four states 'idle', 'busy', 'non sequential', 'sequential'.

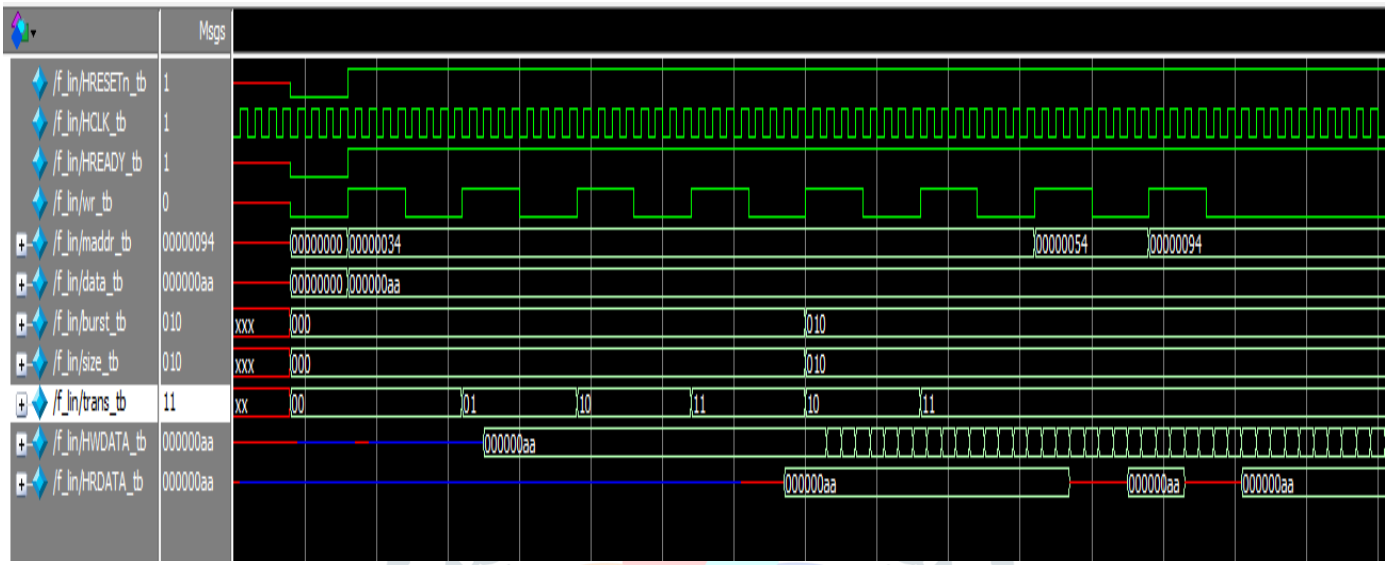


Figure 5: Basic Read and Write Operation

5.2 Burst Transfer

In this transfer, the burst wraps when it hits the address boundary. The boundary is determined by multiplying number of bytes in a burst with the data size. HBURST determines number of bytes, while HSIZE indicates the data size. The waveform below refers to the wrapping burst of 4 bytes and of 32 bit/word data size.

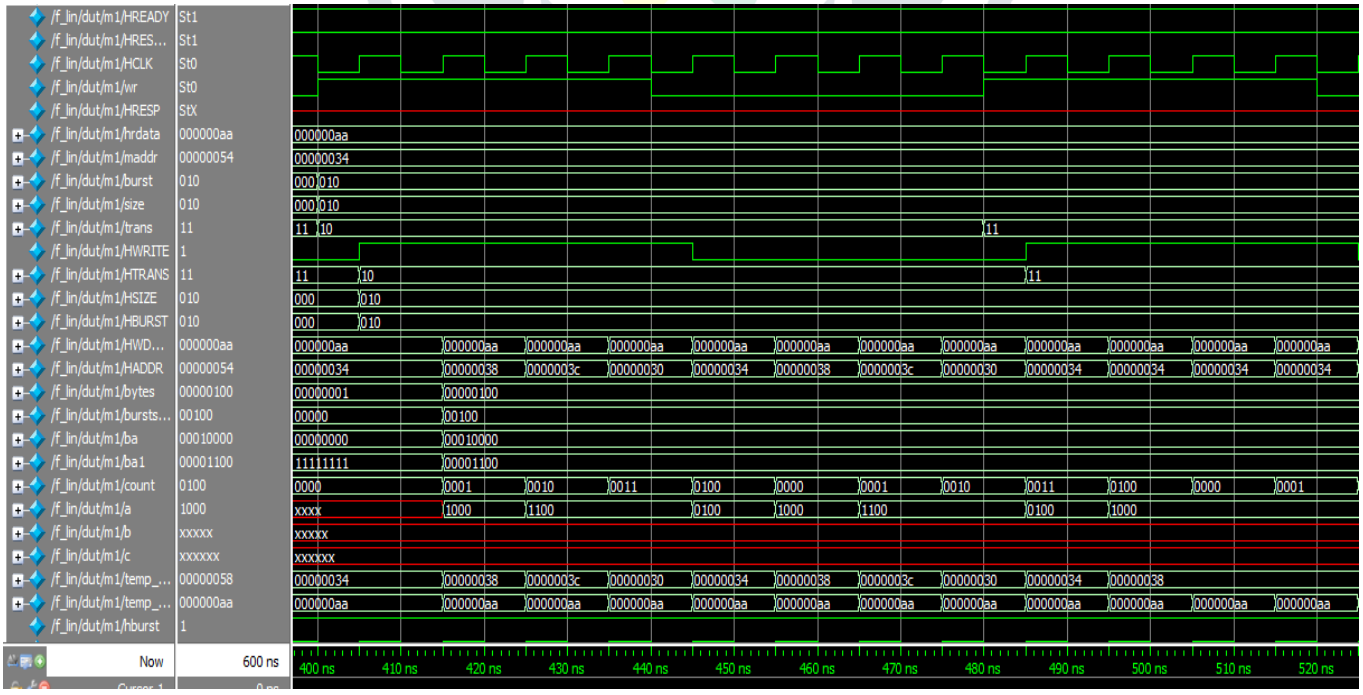


Fig.6 Four byte Brust Transfer.

5.3 Slave Selection

Slave selection is done by Multiplexer based on the address given by the Master. In this paper, three slaves are used where first slave 'S0' ranges from 0x00 to 0x40, second slave 'S1' ranges from 0x41 to 0x60 and third slave 'S2' ranges from 0x61 to

0x99. When master sends the data and address to slave ,based on the address only one slave will be activated remaining slaves will be de-activated. Below figure 7 shows the slave selection based on address allocated by master.

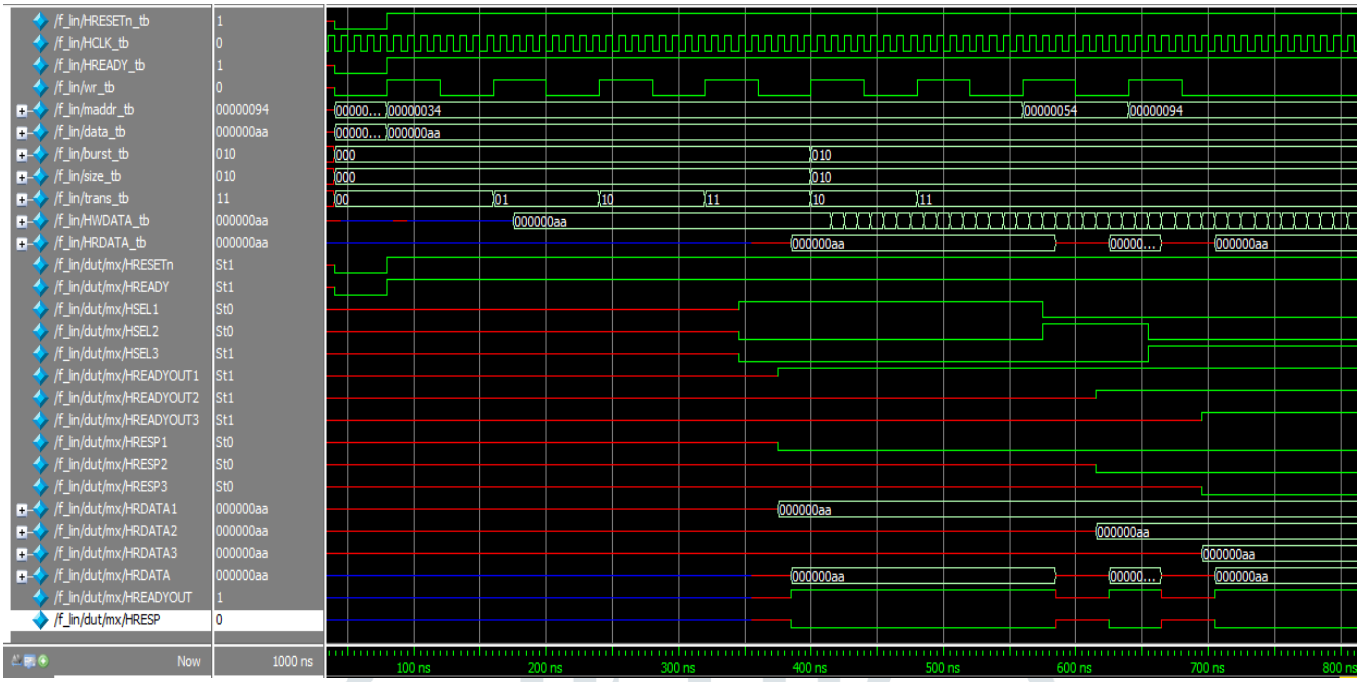


Fig.7 Simulation results to show the Slave Selection.

5.4 Verification Results

Verification of the design is done based on assertion and coverage percentage. The simulation result verification architecture shown in below figure 8, concurrent assertions are used to verify the concurrent different properties and cases of the design (AHB-Lite protocol) below figure:9 shows the 100% assertion percentage of the design and functional code coverage is used to check the design, below figure:9 shows the 100% coverage percentage of the design.

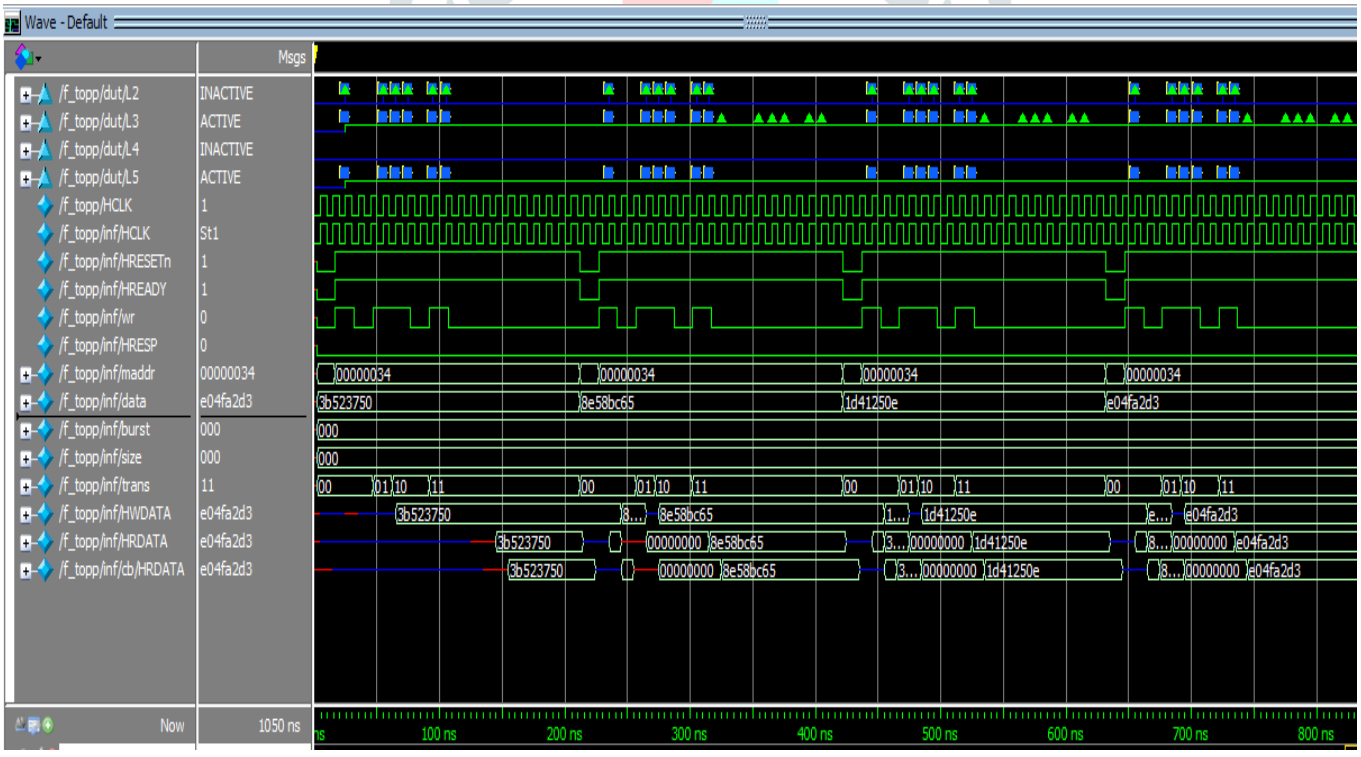
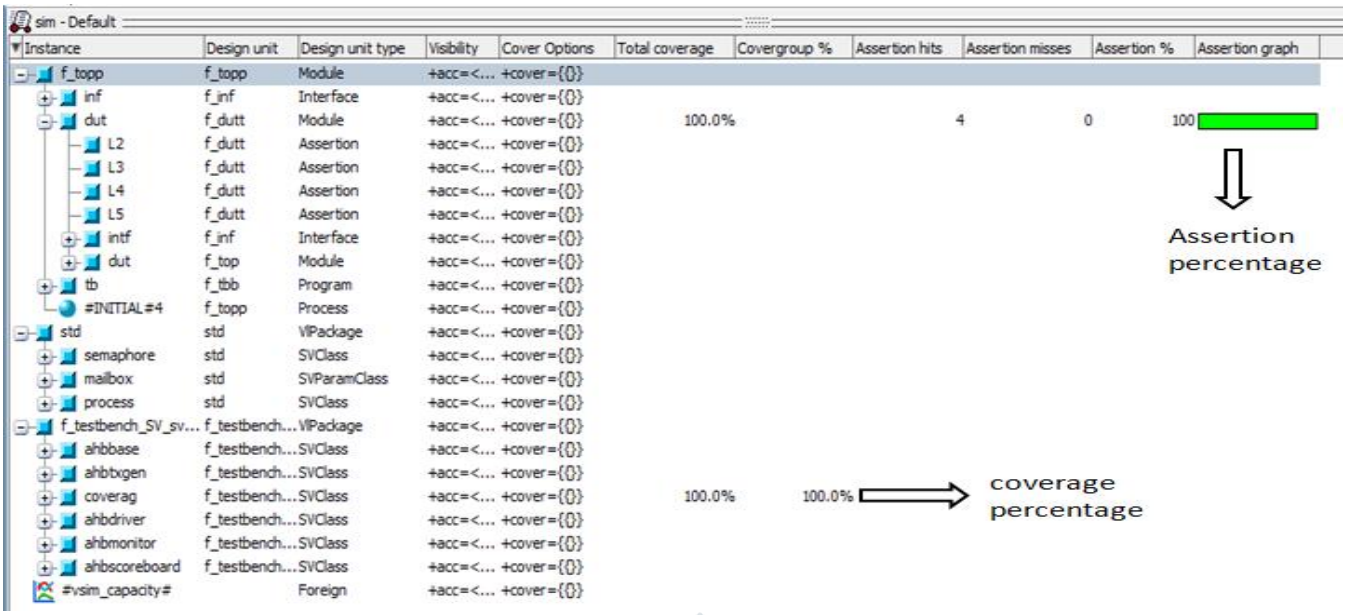


Fig.8 Simulation results on verification of AHB-Lite protocol.



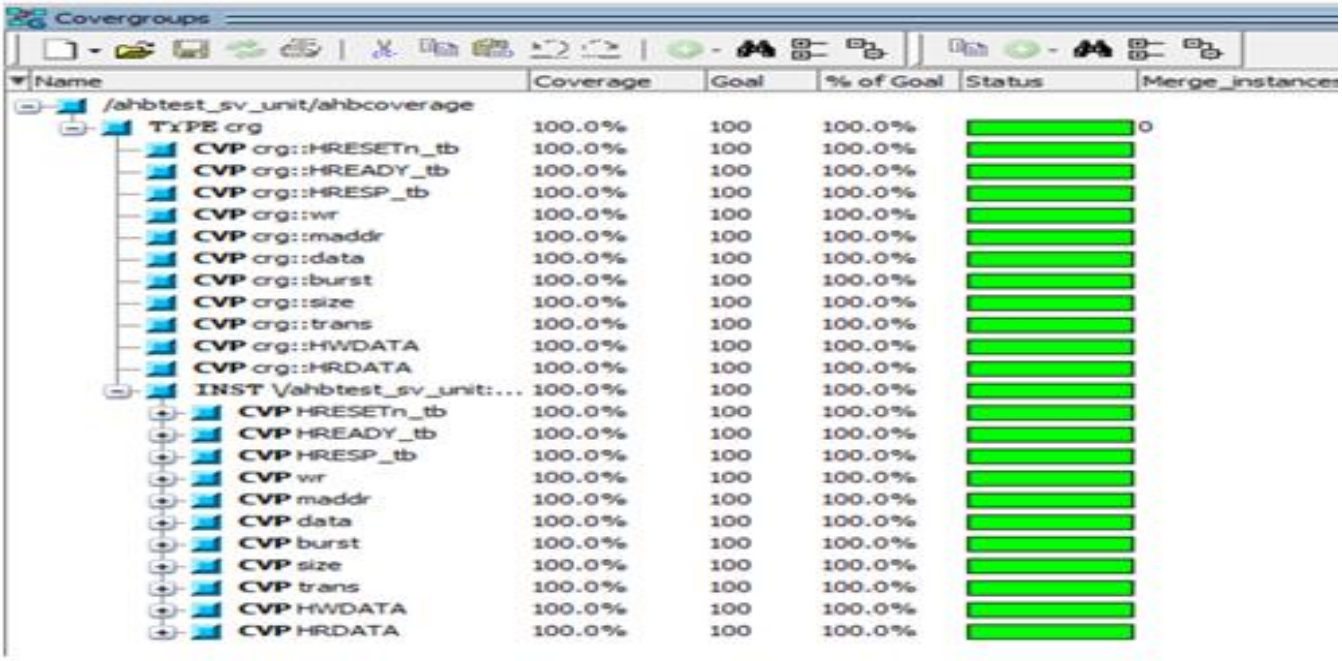
Instance	Design unit	Design unit type	Visibility	Cover Options	Total coverage	Covergroup %	Assertion hits	Assertion misses	Assertion %	Assertion graph
f_topp	f_topp	Module	+acc=<...	+cover={0}						
inf	f_inf	Interface	+acc=<...	+cover={0}						
dut	f_dutt	Module	+acc=<...	+cover={0}	100.0%		4	0	100	
L2	f_dutt	Assertion	+acc=<...	+cover={0}						
L3	f_dutt	Assertion	+acc=<...	+cover={0}						
L4	f_dutt	Assertion	+acc=<...	+cover={0}						
L5	f_dutt	Assertion	+acc=<...	+cover={0}						
intf	f_inf	Interface	+acc=<...	+cover={0}						
dut	f_top	Module	+acc=<...	+cover={0}						
tb	f_tbb	Program	+acc=<...	+cover={0}						
#INITIAL#4	f_topp	Process	+acc=<...	+cover={0}						
std	std	VPackage	+acc=<...	+cover={0}						
semaphore	std	SVClass	+acc=<...	+cover={0}						
mailbox	std	SVParamClass	+acc=<...	+cover={0}						
process	std	SVClass	+acc=<...	+cover={0}						
f_testbench_SV_sv...	f_testbench...	VPackage	+acc=<...	+cover={0}						
ahbbase	f_testbench...	SVClass	+acc=<...	+cover={0}						
ahbtxgen	f_testbench...	SVClass	+acc=<...	+cover={0}						
coverag	f_testbench...	SVClass	+acc=<...	+cover={0}	100.0%	100.0%				
ahbdriver	f_testbench...	SVClass	+acc=<...	+cover={0}						
ahbmonitor	f_testbench...	SVClass	+acc=<...	+cover={0}						
ahbscoreboard	f_testbench...	SVClass	+acc=<...	+cover={0}						
#vsim_capacity#		Foreign	+acc=<...	+cover={0}						

Fig.9 Verification report based on Assertion and Coverage percentage.

Assertions

Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count
/f_topp/dut/L2	Concurrent	SVA	on	0	30	0
/f_topp/dut/L3	Concurrent	SVA	on	0	24	6
/f_topp/dut/L4	Concurrent	SVA	on	0	0	0
/f_topp/dut/L5	Concurrent	SVA	on	0	1	29

Fig.10 Assertion report.



Name	Coverage	Goal	% of Goal	Status	Merge_instances
/ahbtest_sv_unit/ahbcoverage					
TYPE cvg	100.0%	100	100.0%		0
CVP cvg::HRESETn_tb	100.0%	100	100.0%		
CVP cvg::HREADY_tb	100.0%	100	100.0%		
CVP cvg::HRESP_tb	100.0%	100	100.0%		
CVP cvg::wr	100.0%	100	100.0%		
CVP cvg::maddr	100.0%	100	100.0%		
CVP cvg::data	100.0%	100	100.0%		
CVP cvg::burst	100.0%	100	100.0%		
CVP cvg::size	100.0%	100	100.0%		
CVP cvg::trans	100.0%	100	100.0%		
CVP cvg::HWDATA	100.0%	100	100.0%		
CVP cvg::HRDATA	100.0%	100	100.0%		
INST \/ahbtest_sv_unit:...	100.0%	100	100.0%		
CVP HRESETn_tb	100.0%	100	100.0%		
CVP HREADY_tb	100.0%	100	100.0%		
CVP HRESP_tb	100.0%	100	100.0%		
CVP wr	100.0%	100	100.0%		
CVP maddr	100.0%	100	100.0%		
CVP data	100.0%	100	100.0%		
CVP burst	100.0%	100	100.0%		
CVP size	100.0%	100	100.0%		
CVP trans	100.0%	100	100.0%		
CVP HWDATA	100.0%	100	100.0%		
CVP HRDATA	100.0%	100	100.0%		

Fig.11 Coverage report.

VI. CONCLUSION

In this paper, design code for AHB-Lite interconnect environment is simulated and output is observed. System Verilog Assertion ensures true assertion based verification integrated into verilog/SV language. Using four assertions for the design of AHB-Lite protocol we have obtained the 100% assertion percentage. The functional verification of AHB-Lite interconnects, is developed by the Mentor Graphics QUESTA SIM tool and explored as per expectation without any change in the features of DUT. Bins has been created based on the constraints and 100% functional coverage has been obtained. Thus we have designed and verified the Single On Chip communication protocol such as AHB-Lite Protocol.

VI. ACKNOWLEDGMENT

This project is partly supported by the assistant professors of Visvesvaraya Technological University, Post Graduation Center, Satagalli Mysore.

REFERENCES

[1] Rishabh Singh Kurmi, Shruti Bhargava and Ajay Somkuwar, Article: Design of AHB protocol block for Advanced Microcontrollers, International Journal of Comuter Applications 32(8):23-29,October 2011.

[2] Sravya Kante, Hari Kishore Kakarla and Avinash Yadlapati "Design and Verification Of AMBA AHB-Lite protocol using Verilog HDL", International Journal of Engineering and Technology(IJET),Vol 8 No 2 Apr-Mar 2016.

[3] Pallavi T lambe and Meghana Kulkarni, "Functional Verification of AMBA AHB-Lite Interconnect using System Verilog",International Joural of Advanced Research in Electrical, Electronics and Instrumentation Engineering,Vol.6 Issue 7,july 2017.

[4] Mallesham Boini,"Complex SoC Verification using ARM Processor",from elnfochips.

[5] ARM website" www.arm.com "