Functional Verification of AHB Bus Matrix with UVM Methodology

Praveena H U, Santhosh N S, Amaresha S K

***Abstract*— The AHB Bus matrix is enables the parallel access to shared slaves by multiple masters. The routing of the transfers from masters to the slaves is based on the arbitration scheme. Bus matrix enables the multiple AHB masters to connect to the multiple AHB slaves. Bus matrix will decode the transfer control signals, routes the transfer from master to the corresponding slave and response back from slave to the master with valid ready handshake which obey the AMBA AHB protocol specification. The design is verified with verification environment developed with the UVM methodology, which enables the more flexibility and greater control with the reusability.**

***Index Terms*— Parallel Access, Shared Slave, Arbitration, AMBA, UVM, Reusability**

I. INTRODUCTION

In a chip there will be many masters and slaves, there will be more probability of accessing the same slave by multiple masters. In order for secure and safe operations, one must take care of providing access grant to single master at a time. Bus Matrix is the module which acts as arbiter in between masters and slaves, it make sure the single master accessing the slave at any point of time, also it enables parallel access between the different masters and slave.

Functional verification is usually one of the most challenging areas in chip design. It aims to verify that a specific model implements the specification correctly. To implement the specification the design team interprets sentences and paragraphs describing functionality into RTL code. Since both the process is manual and the specification inevitably leaves room for interpretation, there are numerous areas for RTL designers to make mistakes. Functional verification checks the correctness of design.

Verification is performed by generating the random stimulus, driving the stimulus to design and observing the DUT behavior for the driven stimulus. Verification is measured based on Code coverage and Functional coverage numbers. This method is referred as simulation method.

There are many languages and methodologies for writing the verification Testbench. SystemVerilog for verification is the language and UVM Verification methodology is adopted for the verification environment development.

II. FUNCTUIONAL VERIFICTION

A. AHB Bus Matrix Design

AHB Bus Matrix is the top level component which connects the Input node, Decoder and arbitration node, and Output node.

AHB Bus Matrix design supports the following features:

- Architecture type AHB-2
- slave ports used for making connection to masters,
- master ports used for making connection to slaves,
- 32 bits wide of address bus and data bus,
- Arbiter type 'fixed',
- Connectivity mapping: S0, S1 --> M0,M1, M2.

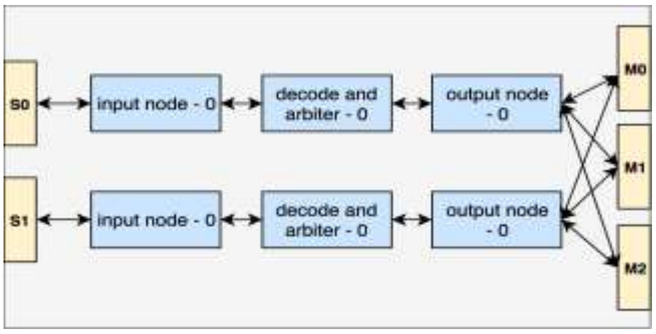


Fig 1: AHB Bus Matrix Design Block Diagram

Above block diagram shows the two slave ports, to communicate with the masters and three master ports, to communicate with the slaves. Also internal blocks are shown.

Below is the pin diagram of the design with the input and output ports are shown.

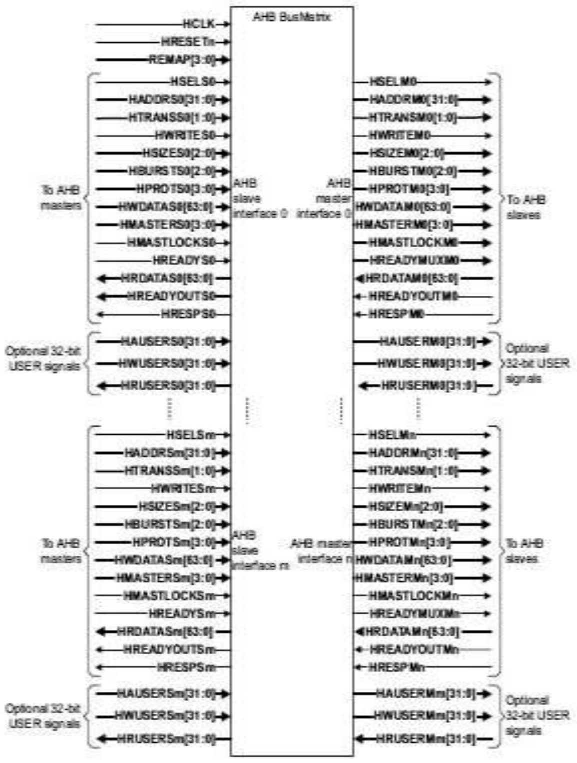


Fig 2: AHB Bus Matrix Pin Diagram

B. Verification environment development

Verification environment is developed to drive the random input stimulus to the design and observe the design behavior, on driving the particular input expected design behavior is as not expected then the error reporting will be done. Also data integrality check is done for error free data handling. Steps involved and components of verification environment are explained in detail with the help of block diagram.

1) Top Level environment Overview:

The top-level environment has AHB Bus Matrix as DUT for verification. On one side, the Bus Matrix will act as slave to master, hence, the Bus Matrix should respond to master as slave. On the other side, the Bus Matrix will act as master to slave, hence, the Bus Matrix should respond to slave as master.

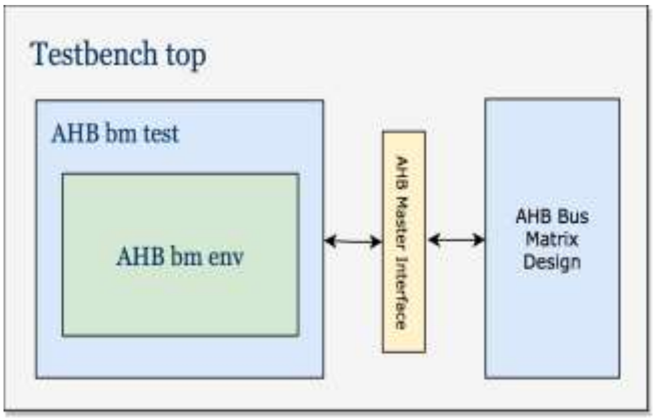


Fig 3: TestBench Top Block diagram

Below are the building blocks of verification environment:

- AHB Master agent and AHB Slave agent
- Scoreboard and Checker
- Coverage Model
- Environment
- Testbench top

2) AHB Master and Slave Agent:

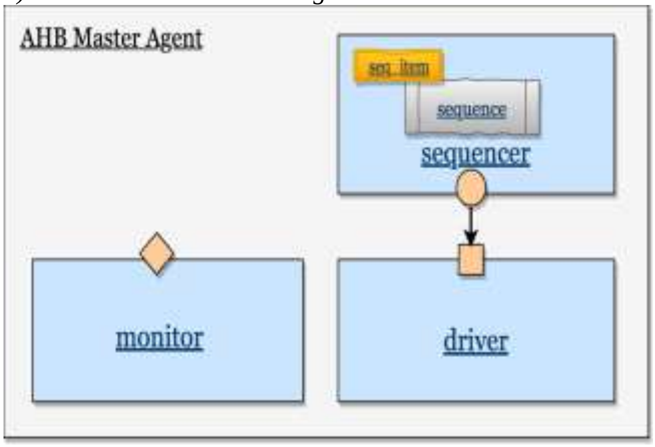


Fig 4: AHB Master Agent

As captured in the above block diagram, Master agent components are;

- Sequence item - consists of control and data variables declared in it with the constraints, these are used to generate the input stimulus to the design.
- Sequences - There are different sequences which are implemented to generate the particular stimulus pattern to drive to design;
- Sequencer - Sequencer provides the sequences to the driver
- Driver - Receives the generated input stimulus from the driver and drives to the Design. The driving of the signals is in accordance with the protocol.
- Monitor - Monitor samples the interface signals and converts the signal level activity into transaction level and send it to the scoreboard.

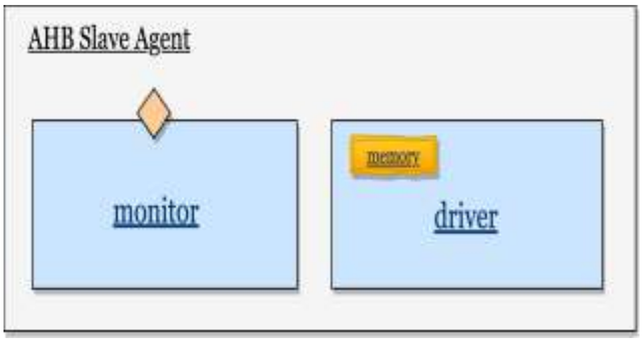


Fig 5: AHB Slave Agent

Captures the address and control information signals by sampling the interfaces signal, Based on address and control information valid data will be stored to slave memory or driven from slave memory to the bus for WRITE and READ transfers respectively.

3) Scoreboard and Checkers:

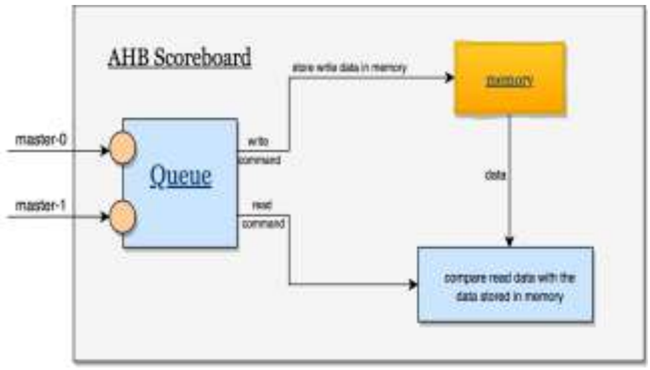

AHB Scoreboard
store write data in memory
memory
master-0
master-1
Queue
write command
read command
data
compare read data with the data stored in memory

Fig 6: Scoreboard

Scoreboard will receive the transaction packets from the monitors. On receiving the packet scoreboard will decode the packet,

If the packet type is WRITE then the write data will be stored into internal memory. If the packet type is READ, then the read data will be compared with the golden data present in the local memory, if there is any mismatch in the READ data with the data written then the ERROR message will be asserted.

Checker will monitor's the signals and checks is there any violation in driving the signal with respect to protocol specification. SystemVerilog assertions are used to write the checker.

In case any protocol violation, the error message will be asserted with the description about the failure. Description helps the easy debugging.

4) Coverage Model:

Coverage model has the list of functional cover points to be covered in order to ensure that all the functional specification of design is tested.

Sl. No	*Cover Point*	*Description*
1	Burst type	Covers the different burst types, example: INCR, SINGLE, WRAP etc
2	Size type	Indicates the beat size, it covers BYTE, HALFWORD, WORD
3	Transfer Type	Indicates WRITE or READ transaction
4	Locked Access	Indicates the Locked access to slave
5	Master ID	Indicates the Master from which the transaction is initiated, it covers; • 0-for Master-0 • 1-for Master-1
6	Slave ID	Indicates the Slave to which the transaction is Sent, it covers; • 0-for Slave-0 • 1-for Slave-1 • 2-for Slave-2 • 3-for Default slave
7	Cross of Master ID and Locked access	Indicates the locked access between all the masters
8	Cross of Master ID, Size, Transfer Type and Burst Type	Indicates combination of Size, Transfer Type and Burst Type across all masters

Table 1: Functional Cover Points

5) Environment:

Environment is the place holder where all the verification components are declared and created, also connection between the different models are done in the environment. Example connecting the Monitor port to the Scoreboard import.

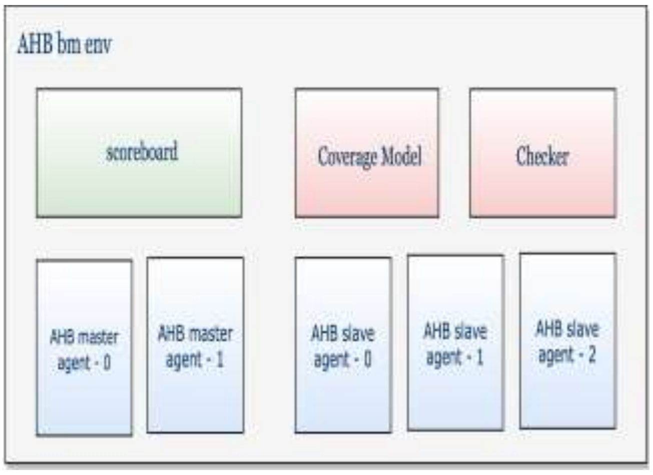


Fig 7: Environment

6) TestBench Top:

This is the top most component in the verification Testbench, in this design and verification environment are integrated and connected together with the help of interfaces.

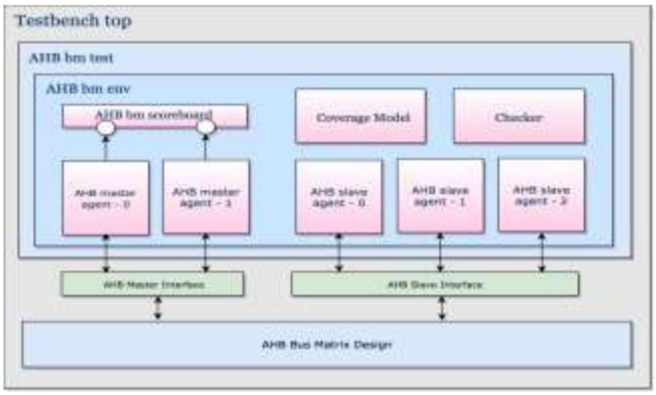


Fig 8: TestBench Top

7) TestCases:

Test Case is the verification component in which the verification environment is declared and created. Declaration, Creation of sequence and running the sequence on particular driver in order to drive the stimulus to design is done in this component.

Testcase examples are single write read test, increment write read test, master-1 access slave-0 test, master-1 unlocked access priority check test, master-1 locked access priority check test etc,

C. Results

1) Single Write and Read

Below are the waveform for the burst type SINGLE and the transfer size of HALFWORD. As shown in waveform, WRITE is done to address 0x0D61D85A with the data 0x000046DD. On reading with the address 0x0D61D85A the read data is same as written data 0x000046DD.

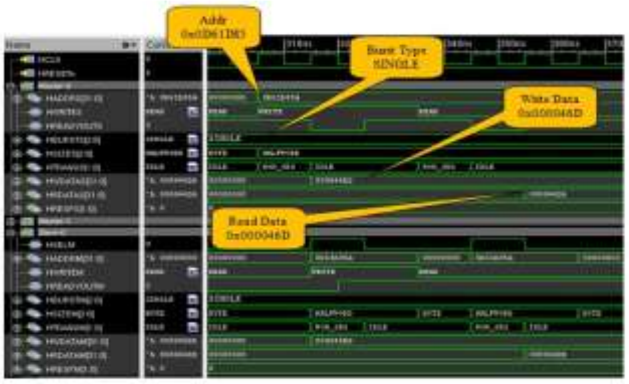


Fig 9: Single Read Waveform

2) Master-1 Locked Access:

As Master-1 requested the locked access to slave-0, on request Master-0 request for the slave-0, access will be granted on completion of transfer from master-1.

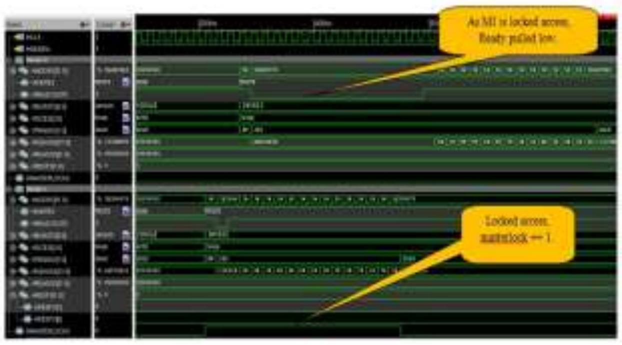
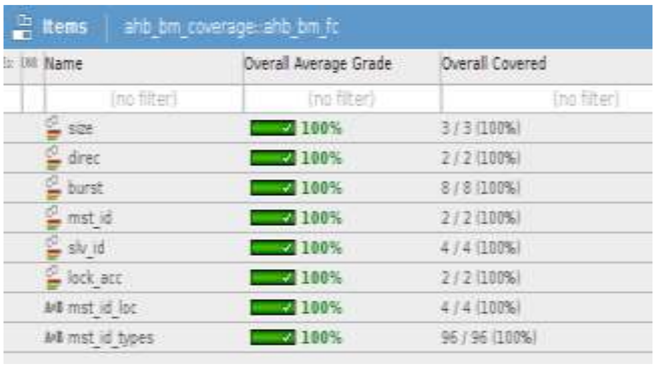

As M1 is locked access, Ready pulled low
Locked access, mastlock = 1

Fig 10: Master-1 Locked Access Waveform

3) Functional Coverage:

Below snapshot shows the functional coverage with the cover points and cross across them.



Items | ahb_bm_coverage::ahb_bm_fc

Name	Overall Average Grade	Overall Covered
(no filter)	(no filter)	(no filter)
size	100%	3 / 3 (100%)
direc	100%	2 / 2 (100%)
burst	100%	8 / 8 (100%)
mst_id	100%	2 / 2 (100%)
slv_id	100%	4 / 4 (100%)
lock_acc	100%	2 / 2 (100%)
mst_id_loc	100%	4 / 4 (100%)
mst_id_types	100%	96 / 96 (100%)

Fig 11: Functional Coverage Snapshot

III. CONCLUSION

For an ASIC design functional verification is necessary in order to check the correctness and quality of the design. In this paper functional verification of AHB Bus Matrix is explained by considering the SystemVerilog as verification language and UVM as verification methodology.

REFERENCES

[1] (Reference paper) "An Overview of On-Chip Buses", Milica Mitic and Mile Stoj ´ cev http://es.elfak.ni.ac.rs/Papers/Facta_2006.pdf

[2] "Multichannel AMBA AHB with multiple arbitration technique" Communications and Signal Processing (ICCSP), 2014 International Conference on Date of Conference: 3-5 April 2014

[3] Cortex-M System Design Kit Technical Reference Manual, http://infocenter.arm.com/help/topic/com.arm.doc.ddi0479b/DDI0479B_cortex_m_system_design_kit_r0p0_trm.pdf

[4] AHB Specification "AMBA® 3 AHB-Lite Protocol" http://mazsola.iit.uni-miskolc.hu/~drdani/docs_arm/IHI0033A_AMBA3_AHB_Lite.pdf

[5] "1800-2017 - IEEE Standard for SystemVerilog--Unified Hardware Design, Specification, and Verification Language"

[6] "1800.2-2017 - IEEE Standard for Universal Verification Methodology Language Reference Manual"

Praveena H. U is doing Mtech in VLSI and Embedded Systems from Visvesvaraya Technological University, Belagavi. Completed BE from BTL Institute of Technology. Area of interest is ASIC Verification.

Santhosh N. S Lecture, Acharya Institution, Bangalore. Area of interest is ASIC Design.

Amaresha S. K Asst. Professor, VTU Ext. Center, UTL Technologies, Bangalore. Area of interest is RTL Design and Physical Design.