Verification of AHB Protocol for AHB-Wishbone Bridge using System Verilog

Ayshi J. Shah
P.G Student
Department of Electronics Engineering
Ahmedabad, Gujarat, India
e-mail: ayushishah6@gmail.com

Samir Shroff
Director
Department of Electronics Engineering
Ahmedabad, Gujarat, India
e-mail:samirshroff@yahoo.com

***Abstract*—**Verification is the process to demonstrate the functional correctness of design and checks that a product or system meets a set of design specifications. This paper implements is a novel approach to enable data transfer between two different bus architectures, AHB and WISHBONE which have different functionalities and characteristics. The coding for this module is designed in the SystemVerilog HDL and simulated in Questa Sim 10.0b. The Communication is done with AHB as Master and WISHBONE as Slave, hence, achieve error free data transfer between the two different bus architectures. The DUT has been verified for all possible test cases.

Keywords*-*AHB;WISHBONE; Verification; SystemverilogEnviornment; Testbench; DUT

*****

I. INTRODUCTION *(HEADING 1)*

With the increasing complexity of digital systems driven by ever increasing demand for faster devices with more features, standard and compatible design. Verification of a design is the most critical phase in the chip design cycle. The progress of VLSI technology enables the integration of more than several million transistors in a single chip to make a SoC (System-on-Chip). This has made verification the most critical bottleneck in the chip design flow. Roughly 70 to 80 percent of the design cycle is spent in functional verification.

Different verification languages are there in the VLSI industry like VHDL, Verilog, systemverilog. Systemverilog is a special hardware verification language is mostly used in functional verification. Systemverilog is the industry's first unified Hardware Description and Verification Language (HDVL). It became an official IEEE standard in 2005 under the development of Accellera[1]. The Systemverilog's aim is to be a single language that is sufficiently expressive to model digital systems at various levels of abstraction from untimed functional models all the way through to netlist level. To support the diverse needs of verification and modelling, it also provides general-purpose object-oriented (OO) programming capabilities.

One goal of verification tool designers is in reducing the complexity of the test bench environment. In this paper, the development of the verification environment of Advance High-performance Bus (AHB) using SystemVerilog. Generator, driver, checker. Monitor, scoreboard is implemented with the proposed integrated verification environment.

II. SYESTEMVERILOG FUNCTIONAL VERIFICATION ENVIORNMENT

As an extension to Verilog HDL, SystemVerilog has characteristics of both hardware description languages and hardware verification language. The key features of SystemVerilog from the verification point are as follows.

- Functional coverage
- Assertion
- Constrained-random stimulus generation
- Higher-level structures
- Multithreading and interprocess communication
- Verification components
- Tight integration with event-simulator for control of the design.

The SystemVerilog code generated has the following components. The hierarchy of the code is as shown in Fig 1.

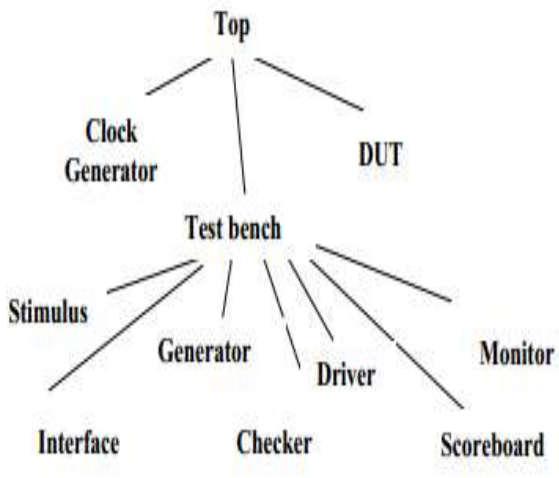

Fig.1. Hierarchy of Developed SystemVerilog Environment[3]

The main purpose of a test bench is to check the correctness of the design under test (DUT). For this following steps have to be followed

- Generate stimulus.
- Apply stimulus to the DUT.
- Capture the response.
- Check for correctness.
- Measure coverage.

The architecture of verification environment developed for AHB protocol is shown in the fig 2. The different modules of environment are explained.

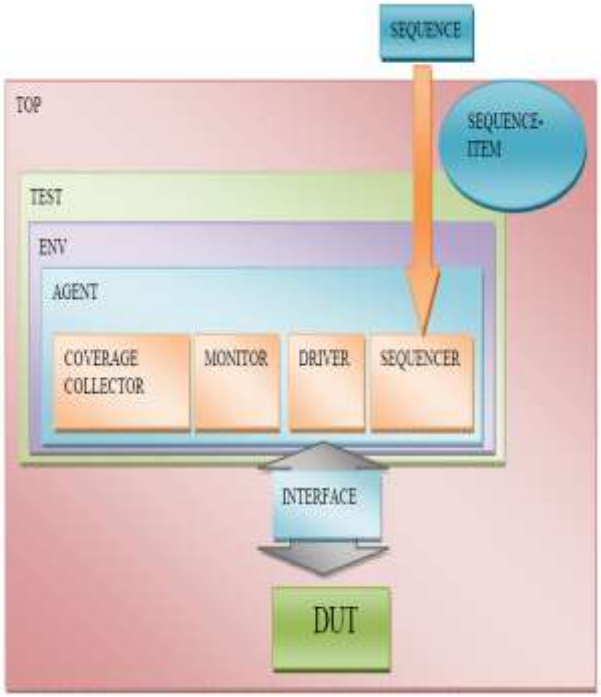

Fig 2: SystemVerilog Verification Environment

As is divided in the Fig 2, there are three parts of the verification environment. The first part is the DUT, which is the design top of the SoC to be verified. The second is the top module (Systemverilog Test bench), which connects the test program and DUT. Test program includes the main test tasks. The interface part (Interface) is the interface to be used to joining the DUT and the test program. All the parts are instanced in the test-top[4]. The verification environment will also be reused, without modifications, by as many test cases as possible to minimize the amount of codes required to verify the DUT.

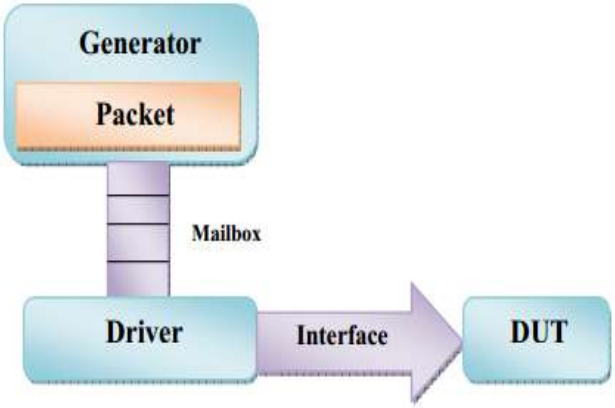

Fig 3: Mailbox, Packet, Interface

TABLE I
SystemVerilog Verification Environment components

Components	Description
Top	Top module connects the Test and DUT. It also has a clock generator which is responsible for running the desired test. This is the top layer of the verification environment.
Test	Test is the top level class Test class initiates the construction process by building the next level down in the hierarchy and initiates the stimulus by starting the main phase. It is controlling the overall verification flow. Test class is instantiated the environment and then configures it for a particular test at a time.
Environment	Assembles the testbench structure, contain one or more agents, scoreboard, depend on design. It has configuration parameters that allow to restructure and reuse it for different scenarios.
Agent	Encapsulates a driver, a sequencer and a monitor. Agent is configurable either as a active or as a passive. Active contains driver, Sequencer and monitor, while passive component contains only Monitor. Agent will also pass the interface of the DUT to each of the sub-sequent component.
Generator and Squencer	Random Stimulus makes the packet. The generator generates the random stimulus from random stimulus packet class. Sequencer runs stimulus generation code and sends this stimulus to driver using mailbox or by other means like callbacks..
Driver	Driver first unpacks the packet and translates the operations produced by the generator into the actual inputs for the DUT . Maps the sequence items to the signal level format.
Monitor	Sample the dut signal from the interface,s but does not drive them. It will keep displays various messages according to the operations being performed like whether it is read or write operation. Similarly, it also shows start, stop and transfer of data operations. In the monitor code Task 'run' is calling start, stop, data tasks.
Scoreboard	Compare the o/p with reference model.
Interface	For communication between classes and modules. It is the mechanism to connect Testbench to the DUT

III. TESTCASE SENARIO

Testcase scenario document includes all the possible combination to to test the functionality of the design under test

ISSN: 2321-8169
2803 - 2806

(DUT). Test cases are identified from the design specification. The verification engineer must design the architecture of the verification environment at the outset to achieve the ability to support constrained random test cases in an efficient manner. In designing our goal is to develop a robust and easy to use mechanism that facilitates the development of test cases with minimal impact to the test bench code.

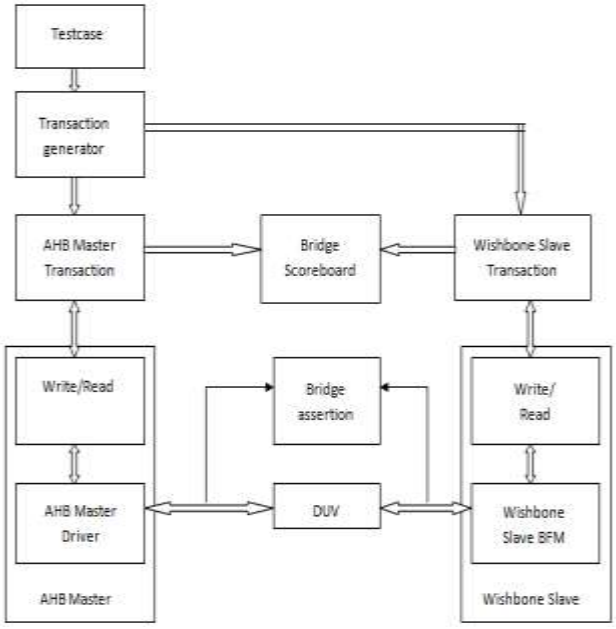

Fig.4. AHB2WB Verification Testcase Environment

Fig.4. shows developed systemverilig environment to implement all testcases for verification.

TABLE II
Testplan for Verification of AHB Interface

Sr no	Cycle operation	Signal required for Operation	Signal required for verification
1	Reset	Hresetn , Hclk	
2	Read	**AHB:**Hwrite,Hready,Haddr, Hrdata **WISHBONE:** adr_o, dat_i, ack_i	adr_o, ack_i , Hrdata
3	Write	**AHB:**Hwrite,Hready,Haddr, Hrdata **WISHBONE:** adr_o, dat_i, ack_i	adr_o, ack_i , Hwdata
4	Write with wait state by wishbone slave	**AHB :** Hwrite , Hready , Haddr , Hwdata **WISHBONE :** adr_o , dat_o , ack_i	adr_o , dat_o , ack_i , Hready
5	Write with wait state by AHB master	**AHB :** Hwrite , Hready , Haddr , Hwdata , Htrans **WISHBONE :** adr_o , dat_o , ack_i , stb_o	adr_o , dat_o , ack_i , Htrans , stb_o
6	Read with wait state	**AHB :** Hwrite , Hready , Haddr , Hrdata	adr_o, ack_i ,
	By wishbone slave	**WISHBONE :** adr_o , dat_i , ack_i	Hrdata , Hready
7	Read with wait state By AHB master	**AHB :** Hwrite , Hready , Haddr , Hrdata , Htrans **WISHBONE :** adr_o , dat_i , ack_i , stb_o	adr_o, ack_i , Hrdata , Htrans , stb_o
8	Read after write	**AHB :** Hwrite , Hready , Haddr , Hrdata , Hwdata **WISHBONE :** adr_o , dat_i , dat_o , we_o	adr_o , dat_o , dat_i , Hwrite , Hrdata , we_o
9	Write after read	**AHB :** Hwrite , Hready , Haddr , Hrdata , Hwdata **WISHBONE :** adr_o , dat_i , dat_o , we_o	adr_o , dat_o , Hwrite , Hrdata , we_o

IV. Simulation Result

The proposed verification environment applies constrained random technique to fulfill the configuration of verification environment and DUT, and to cover all the corner cases. The functional coverage is employed to make sure that DUT has realized all the expected functional requirements. And the automaticity of the proposed verification environment improves efficiency of verification largely.

Various simulation outputs are obtained after stimulating the DUT, out of which some inputs and outputs are shown below.

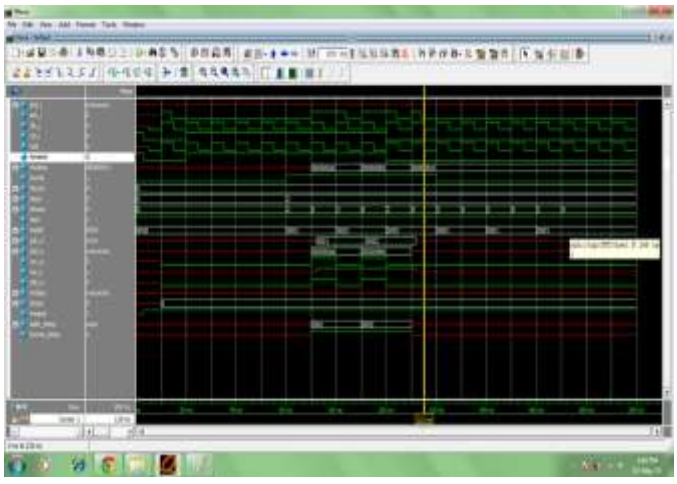

Fig 5: Reset operation

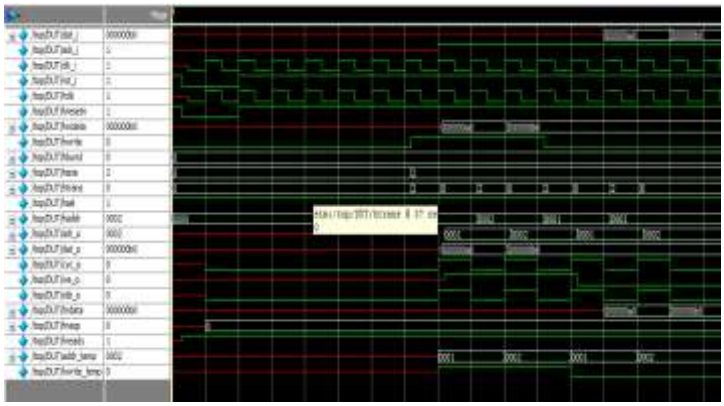

Fig 6: Read and Write operation

V. COCLUSION

The SystemVerilog verification environment developed along with the complete flow of verification for AHB protocol has been discussed. The various classes for driver monitor, stimulus, generator, environment etc. and modules or programs made have been compiled and simulated and the outputs observed are shown. The environment can be reused easily for different design. By using this verification environment the DUT has been verified for its functionality.

REFERENCES

[1] IEEE Standard for System Verilog, IEEE Std 1800[tm] 500 University Science, 1989.

[2] Opencore organization, "AHB-WISHBONE BRIDGE", Released: July 13, 2007.

[3] Rakhi Nangia, Neeraj Kr. Shukla, "Functional verification of I2C core using SystemVerilog", International Journal of Engineering, Science and Technology, Vol. 6, No. 4, 2014, pp. 31-44

[4] Martin Keavency, Anthony McMahon, et al. "The development of advanced verification environments using systemverilog". Proceedings of ISSC2008, pp.325-330, 2008.

[5] www.asicword.com

.