Verification of AHB2APB Bridge Protocol using UVM

Saroja V. Siddamal
School of Electronics and Communication
KLE Technological University Hubli, India
sarojavs@kletech.ac.in

Suneeta V. Budihal
School of Electronics and Communication
KLE Technological University Hubli, India
suneeta_vb@kletech.ac.in

Apoorva Narode
School of Electronics and Communication
KLE Technological University Hubli, India
anapoorva08@gmail.com

***Abstract*—**It is vital to update tools and methodologies in pace with technological innovation in order to handle the problems posed by the changing verification environment. The authors propose a verification of AHB2 APB bridge protocol using Universal Verification Methodology (UVM). It is the intention of this study to discuss the benefits of using the Universal Verification Methodology for AHB2APB verification. This work focuses on creating a standard AHB2APB Bridge protocol architecture that is efficient and on enhancing the authentication environment utilizing a System Verilog UVM implementation. An automated authentication platform developed by UVM will produce an AHB2APB Bridge debugging test for any given DUT. UVM-based performance verification adds randomized test scenarios to ensure high performance by covering all potential cases. On the other hand, Verilog cannot be used to examine the active cover model in a typical verification location. System Verilog is used for coding, while Questasim is the simulation tool of choice. The code coverage for RTL design is acquired, and 98.81 percent of the code coverage and 100 percent of the functional coverage are recovered.

***Keywords*—** AHB2APB Bridge, UVM, code coverage, Questasim

I. INTRODUCTION

To implement the AHB2APB bus protocol, multiplexers are a connecting technique. In this technique, the master and arbitrator communicate as follows: The address and control signals driven by the bus master decide whether a transfer is a read or write transfer. The address is sent to the chosen slave when the arbitrator selects the master that possesses its control signal. The data read and signal response multiplexer, which selects the pertinent signals from the slaves needed for the transfer, are primarily under the control of the central decoder.

The AHB bus is for high clock frequency system for providing:

- It offers the High performance, high bandwidth and pipelined operation
- AHB protocol supports Split transactions, burst transfers, and multiple bus masters.
- It has outstanding latency and uses little power.

The advanced peripheral bus (APB): It is a straightforward, non-pipelined protocol that enables reading and writing from or to a bridge or master to a number of slaves via a common bus. The same sets of signals are used for both reads and writes. The transfer of bursts is not supported. .

II. RELATED WORKS

Several works on verification of communication protocols is contributed to its development and overall performances. In paper [2] authors have worked on the AMBA AHB is for system modules with high performance and high clock frequencies. The high-performance backbone system bus is the AHB. The effective connectivity of processors is supported by AHB. The AMBA APB is designed to use less power and have a simpler interface to support peripheral operations. Functions of the AHB2APB Bridge protocol are implemented in this project by writing the code in VERILOG and simulating it in XILINX ISE. In this work, we write UVM verification code and use several test cases to validate all Bridge protocol functionalities.

Authors in their [3] work explains that an open System-on-Chip bus protocol called Advanced Microcontroller Bus Architecture (AMBA) allows high performance buses and low-power devices to interface with one another. The AHB and APB buses are connected by a bridge that is also present. Bridges are common bus-to-bus interfaces that enable standardised communication between IPs linked to various buses. In order to develop, implement, and test the AHB2APB bridge, verilog and UVM were used, and the results are described in this work The AHB2APB Bridge is a sophisticated interface bridge between AHB and APB that has been designed as a synthesizable RTL code.

Authors in [4] worked on the fast time-to market requirements, the rising expense of testing makes producing a highly integrated SoC a significant strain. An effectively tested design with on chip peripherals technique is presented in this study. Utilizing the bridge function to its fullest extent allows for effective functional and structural testing. By adding more test channels and condensing the test-control protocols, the testing time can be greatly decreased. According to experimental findings, both functional and structural test modes significantly minimise testing durations and area overhead. The suggested method can be applied to many on/off-chip bus bridge types.

In paper [9] the AMBA is widely adopted connectivity standard and on-chip-bus-architecture is used to increase the reusability of IP core (SOC). It is difficult to analyse embedded systems that use AMBA. The authors have synthesized and simulated the AHB2APB Bridge, which is

978-1-6654-8716-0/22/$31.00 ©2022 IEEE

a complex interface bridge between the AHB abd APB Here, a list of Bridge modules is constructed using the Paper Synthesized Net. To perform functional and timing simulation Xilinx and modelsim tools are used.

Authors in paper [10] have designed and verified the AMBA-APB Protocol. AHB, ASB, and AXI, for example, are high performance bus components that are utilized as interfaces with APB, a low performance bus, and other components of the AMBA Bus. APB connects to slaves such as UART, TIMER, Keypad, INTERRUPT CONTROLLER, etc. using minimal peripheral bandwidth. Simulation-based verification is the conventional approach. The complexity of ICs has risen as technology has advanced. As a result, verification time has also increased. The design of the APB protocol in Verilog and verification in two languages—System Verilog and Universal Verification Methodology—are the primary topics of this work (UVM).

III. METHODOLOGY

All the high-performance peripherals will be connected to an AHB bus, while the low power peripherals will all be connected to an APB bus.

Bridge transforms AHB transferals to APB transferals; it is both an AHB slave and an APB master; it serves as an interface between high-performance IPs and low-power peripherals. For building high-performance embedded microcontrollers, AMBA specification defines an on-chip communications standard.

.

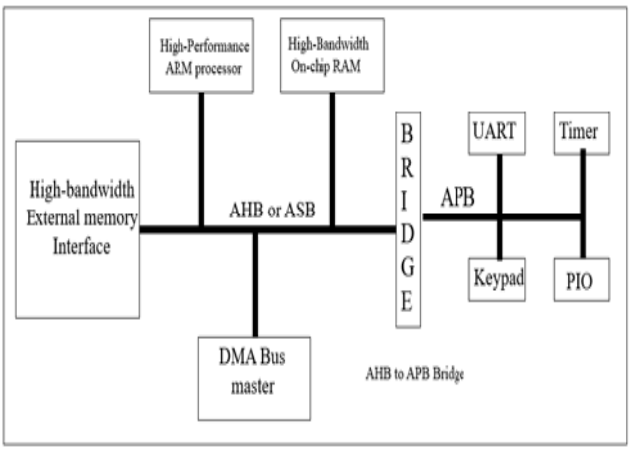

Fig.1: AMBA System

In the AMBA Structure there are two buses AHB and APB. High performance is needed for the CPU (ARM) cores, DMA, and high bandwidth memory, even though the low bandwidth peripherals are connected through the APB bus. An AHB to APB bridge connects AHB with APB. While all peripherals linked to the APB act as slaves, the AHB-APB bridge, also referred to as the APB bridge, serves as the Master and initiates all transactions

This paper discusses the architecture of the AMBA APB bridge protocol and offers a design verification strategy for creating an AMBA APB bridge verification IP with a UVM-based custom test bench. It also looks at the outcomes.

A. ADVANCED HIGH-PERFORMANCE BUS (AHB)

AHB is a high performance and high clock frequency system modules which acts as the system's high-performance backbone bus. Low power peripheral microcell operations can be successfully coupled with CPUs, on-chip memories, and off-chip external memory interfaces with the help of AHB.

AHB also makes use of automated test methodologies and synthesis to ensure usability in an efficient design flow.

Any internal memory, the APB bridge, and the external memory interface are the most often used AHB slaves. The following components are present in an AMBA AHB system architecture that is typical: 3 AHB master A bus master can begin read and write operations by providing an address and control information. There can only ever be one active bus master driving the bus at a time. AHB employee A bus slave responds to a read or write operation within a particular address-space range.

The bus slave notifies the active master of the data transfer's success, failure, or waiting status. Decide for AHB The bus arbitrator makes it possible for just one bus master to initiate data transfers at once. The arbitration protocol is established, however any arbitration procedure, such as highest priority or fair access, may be employed depending on the needs of the application. An AHB would only have one arbiter, even though this would be superfluous in systems with a single bus master. The AHB decoder, decodes each transfer's address and communicates a choose signal to the participating slave. A single centralized decoder must be used by all AHB implementations.

A. ADVANCED PERIPHERAL BUS (APB)

There is APB for low-power peripherals. The interface is made to be as simple as feasible to facilitate peripheral operations while consuming the least amount of power possible. APB is compatible with both system bus iterations. The APB bridge acts as a slave module for the local peripheral bus and controls bus handshake and control signal retiming. In an AMBA APB implementation, an APB bridge is frequently included and is required to convert AHB or ASB transfers into a language that the slave devices on the APB can understand.

The Architecture of AHB to APB Bridge is as shown in Fig. 2.

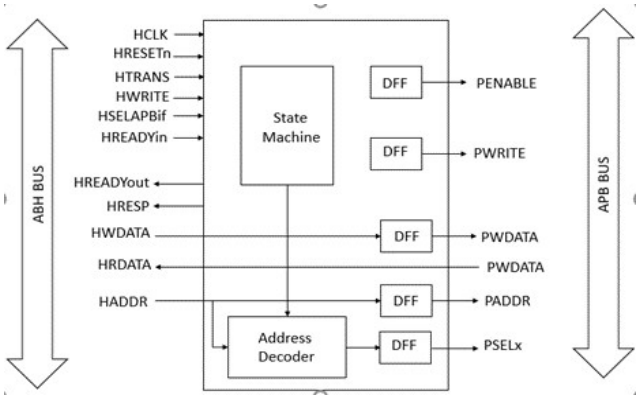

Fig 1: AHB to APB Bridge Architecture

APB Finite State Machine and AHB Slave Interface are the bridge's primary components (FSM). This bridge generates a signal for each connected device and manages the peripherals map to memory addresses.

State machine representation of the process is as shown in Figure 3. The "Hreadyout" signal governs all AHB transactions. All APB output signals are generated under control of the bridge as well.

PWDATA is continuously driven by the bridge, the only master aboard the bus. When the slaves are chosen by the bridge during APB read transfers, PRDATA is only driven.

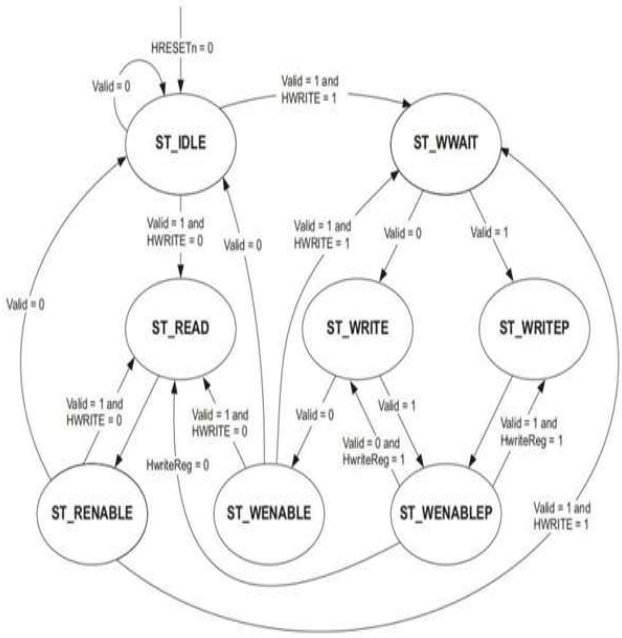

Fig. 3: FSM of AHB to APB Bridge

Verilog HDL is used to create the AHB-APB bridge. The APB FSM controller and AHB slave module are the two submodules that make up the bridge module. It is created as a Register-Transfer-Level module so that it can be physically realized and synthesized. To ensure that the bridge complies with the protocol, two extra modules have been constructed. Bus functional is the realization of AHB Master and APB Slave modules.

IV. IMPLEMENTATION DETAILS

. The foundation of the Open Verification Methodology is used by UVM. Version UVM 1.0 EA was made available by Accellera on May 17, 2010. A strong verification strategy is frequently used to support a design's high quality. AHB2APB bridge is incorporated into the project's design. Functional verification of the integrated device is carried out utilising a verification test bench built using UVM and System Verilog (SV). A good verification strategy should test the Design Under Test (DUT functioning)'s in all conceivable situations. An engineer should create an organised, automated test bench to do this. A good verification plan should be created before building a test bench, taking all potential outcomes into consideration.

A. Assertion based Verification Plan

The predicted behavior of the DUT under specific circumstances or events is tested using assertions. The assertions will stop the test and indicate an error if the device doesn't operate as expected. The assertions are among the most potent components of the verification strategy. A condition and a message to be displayed upon test termination make up an assertion. More DUT features are tested the more claims there are. Assertions shorten the test's simulation runtime because they operate in parallel. It is possible to write coverage for assertion. This will determine how frequently the case relevant to the assertion condition appears. Assertions and coverage groups both belong in a good verification test bench.

Building test scenarios that feed inputs of various formats into the DUT in accordance with protocol and then verify output allows for the correctness of the AHB2APB bridge to be validated. In Chapter 4.2, the scenarios that were created to test the functions are mentioned.

B. Assertion based Verification Plan

For verification of design testbench is built using system Verilog/UVM. This test bench is a setting that provides stimuli for the design being tested (DUT). He verification is done using a reference model that implements the DUT capability, and whose outcomes are considered as expected output for any given set of inputs. The DUT's results are contrasted with those of the reference model.

To establish a directed test bench is the conventional method of the aforesaid verification methodology. Verification engineers can only attempt to validate the design using this directed test bench by sending a small number of essential sets of inputs to the DUT. Random stimuli are required in order to take into account all the unlikely-corner cases as well. Some kind of automated test bench environment that can generate random stimulus is required in order to employ random-stimuli. A group of semiconductor companies came up with the UVM standard verification approach as a result of this.

The main elements of the UVM test bench environment include a driver, monitor, sequencer, and scoreboard.

UVM provides a basic class for each component with standardized functions, allowing us to customize the test bench environment to meet our needs. Every part of the UVM test bench relies on data that is constantly flowing through the surrounding environment to function. This information is referred to as a sequence item, and it is essentially dynamic in nature. Reusability will be one of UVM's standout qualities. Any verification engineer with a solid grasp of the UVM approach may quickly comprehend the established test bench and make changes in accordance with his requirements.

Any test scenario can be verified using a variety of techniques, such as creating inputs, driving them at the correct clock cycle, obtaining outputs, and lastly comparing the inputs from the previous cycle with the output. The building of the appropriate test bench will enable these events to take place sequentially.

*B. **UVM based Test Bench Architecture***

Test bench Top: The UVM test bench includes the interface instantiation that connects the DUT with the

UVM test bench as well as the design instantiation that needs to be verified. Transaction Level Modeling (TLM) techniques are used in UVM to connect components. A UVM test that compiles the test bench all at once and performs the several tests linked to it is called during runtime.

Bridge Test: UVM Test is the top-level component class that is located under the UVM test bench. In this class, UVM sequences are called to provide the design under test with the necessary stimuli. Value configuration is possible using configuration classes

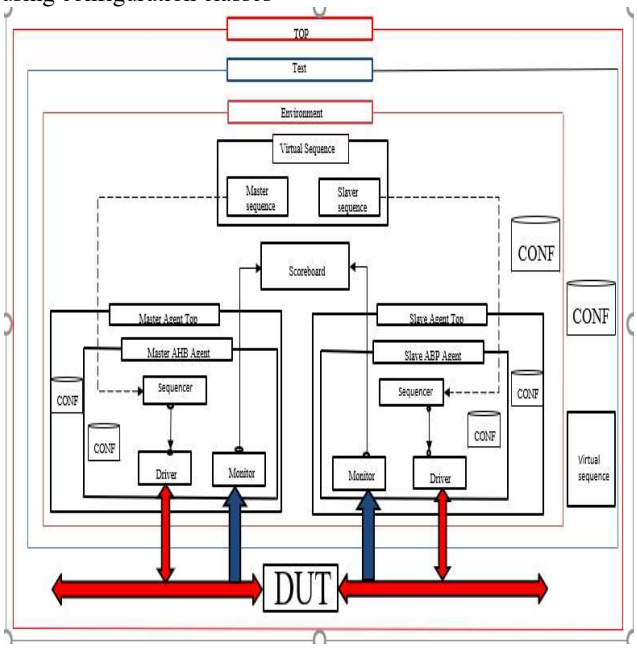

Fig. 4: AHB to APB Bridge Verification TB Architecture

Bridge Environment: This container component class contains all verification components, such as scoreboards and agents, organised according to hierarchy. This environment class aims to encompass all design-focused lower-level verification components. UVM test allows you to modify the UVM environment's default settings.

AHB Agent: Agent comprises of low-level class like sequencer, driver, and monitor.

APB Agent: The UVM class hierarchy's lowest and most basic level is the sequence item class. This class adds the ability to define transaction items with or without limitations to the UVM object class. In order for the sequencer class to communicate the defined transactions to the driver, this class is essential.

The UVM object class, which is intended to produce and randomise a set or collection of transactions as defined by the sequence items class, is extended by the bridge sequence class. To the UVM sequencer, which subsequently delivers it to the driver, are sent the created transactions.

AHB Sequencer: The sequencer makes the UVM component longer. The sequence and the driver communicate with one another through the sequencer. By adhering to the hand-shake procedures between the sequence and the driver, it passes transaction items.

Driver for UVM: The UVM component is expanded by the UVM driver. By transforming the transaction into a pin signal, it forces the transactions into the DUT. Get next item (), item done (), and other methods are used to receive transaction items.

AHB Monitor: The UVM monitor extends the UVM component. It converts the output pin signals of DUT into a transaction item and sends it to the scoreboard component for comparison with golden responses.

V. RESULTS AND OPTIMIZATION

This session presents the findings of the functional simulation of the AHP2APB bridge module. The AHP2APB bridge top level module and all of its submodules were created in Verilog-Hardware Description Language (Verilog-HDL), and they were all tested first on a conventional Verilog-HDL test bench and then on a UVM test bench. Having constructed a test bench in UVM. Using UVM approaches, the Bridge is implemented for the functional verification for both read and write operations.

```
#Width = 32, size 16 bits
#-----------------------------------------------------------------------------------------------------
# Name                          Type Size Value          Size          Value
# -----------------------------------------------------------------------------------------------------
#  req                          ahb_xtn   -   @1094
#      begin time               time      64  1060
#      end_time                 time      64  1060
#      depth                    int       32  'd2
#      Parent sequence (name)   string    7   wr_seqs
#      parent sequence (fullname) string  49  uvm_test_top.env_h.ahb_agt_top.agent.sr_h.wr_seqs
#      sequencer                string    41  uvm_test_top.env_h.ahb_agt_top.agent.sr_h
#      HRESETn                  integral  1   'd0
#      HTRANS                   integral  2   3
#      HBURST                   integral  3   5
#      HSIZE                    integral  3   'd1
#      HWRITE                   integral  1   1
#      HADDR                    integral  32  'h80002e4
#      HWDATA                   integral  32  'hfcb73cd2
#      HRDATA                   integral  32  'h0
#-----------------------------------------------------------------------------------------------------
```

Fig. 5: Test bench Topology

The RTL design's functional verification by the Bridge results in complete code and functional coverage. Bridge Functional Verification Using UVM Because verification methodology is a crucial part of circuit design. For the RTL design, the read operation of the Bridge is completed in XILINX, and the verification techniques are completed in Questasim Verilog is used for the design, and UVM is used for the verification. The Bridge is configured as a DUT, functional verification, and 100 percent code coverage are attained. From the Fig. 5 the signal HRESETn is the reset, it is the only active LOW signal in AMBA AHB, and it is the primary reset for all the bus elements. The reset may b asserted asynchronously but deserted synchronous with the rising edge of HCLK. HTRANS indicates the type of transfer. Every transfer can be classified into four different types that are Ideal, busy, Sequential, and non-sequential. The HSIZE indicates the size of transfer.

A. UVM based Test Bench Architecture

Scoreboard is a class that extends from UVM Component, where we obtain inputs and outputs from Input

monitor and Output monitor respectively, and then compares the output.

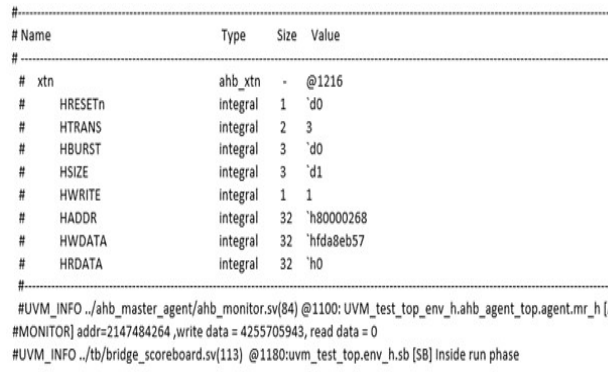

Fig. 6: Scoreboard Result

Data from monitor class is received into scoreboard. Inputs and Outputs are printed in the transcript. Here the Input fed to the DUT in the first cycle is stored in the register. The write-transfer from AHB to APB can occurs with zero wait states. The initial transfer is the start of a burst and therefore is non-sequential. The bridge is responsible for sampling the address, transfer of data from AHB to peripherals through APB and holding the value for duration of APB write transfer. Initially when HADDR gets the address value. Hwrite signal is 1 then HWDATA is broadcasted on the data bus. The above figure 5.1 is the packet generated at the AHB bus. Initially reset signal will resets the bus and the HTRANS transfer type will be sequential. Burst is set for single transfer. The HWRITE signal is set to 1 so that write single is enabled for write operation. And HADDR is address is broadcasted on the address bus. HWDATA is the write data bus, driven by the bus master during write-transfers. HRDTAT is the read data bus is driven by the appropriate slave during read transfers. The generated data HWDATA = 'hfcb73cd2 is transferred from AHB to APB through Bridge.

The fig. 7 represents the packet received by APB. Whenever the data transfer is to be done PSELX signal indicated the slave selection for data transfer. PSELX is 0, and PENABLE is asserted. During the state transition from SETUP to Enable the address, write and select signals are maintained stable. For APB write access the PWRITE is maintained to High. The data and the address from the APB is same as the data and address in AHB. So transaction is successful.

After running the test cases for multiple times and if the test gets passed in all the iterations, then TEST CASE PASS message will be displayed as shown in Figure 5.3 else TEST CASE FAIL message will be displayed.

Coverage Report

The overall code coverage is 98.81 percent while the branches, conditions, FSM states, FSM branch and statements are 94.08 percent. Hence, this proves the design of an AHB2APB bridge is tested for all cases.

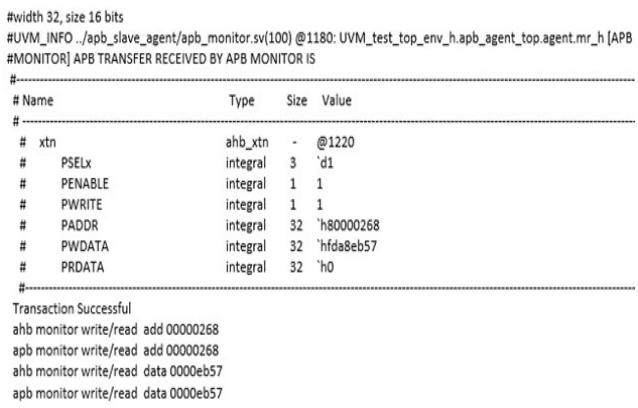

Fig 7: Transaction Packet in AHB

Fig.8: Base Test Result

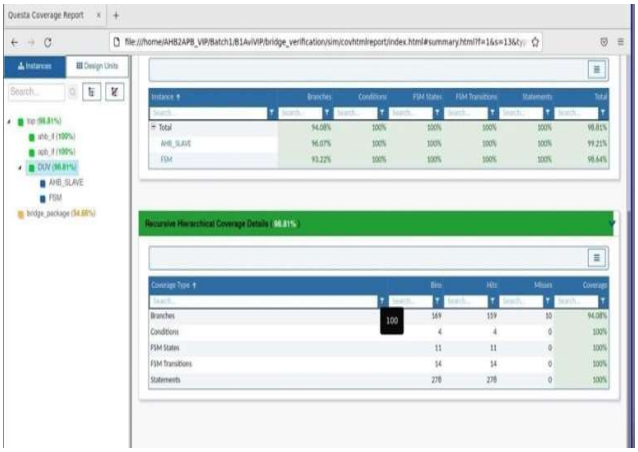

Fig. 9: Code Coverage

VI. Conclusion

The proposed work shows the methodologies to construct and test the AHB2APB Bridge. The code coverage for RTL design is acquired, and 98.81 percent of the code coverage and 100 percent of the functional coverage are recovered. The methodology offers full RTL design coverage in order to produce a bridge protocol design that is error-free. Therefore, it can be integrated into a real-time system. Additional implementation of this is possible for SOC and ASIC applications. Future work will involve implementing an interface between various AMBA protocols and APB.

References

[1] MBA™ Specification (Rev 2.0).

[2] Prarthi Bhatt, Devang Shah, "Design of an Efficient Design for Test (DFT) Architecture and its Verification Using Universal Verification Methodology" International Journal of Recent Technology and Engineering (2019).

[3] P. Mishra, R. Morad, A. Ziv and S. Ray, "Design AMBA Based AHB To APB Bridge Using Verilog HDL", 2018.

[4] Prameeth.H, Aayushii Goswami, Prajwa. M, Vikas. N.G, Dr. Rachana S. Akki. on "UVM Verification of AMBA APB Protocol", International Journal of Recent Technology and Engineering. Vol 8, no. 10 May 2021.

[5] Aparna Kharade and V. Jayashree, "DESIGN OF AMBA BASED AHB2APB BRIDGE". D.K.T.E. Society's Textile and Engineering Institute, Ichalkaranji, Maharashtra, India. 2018.

[6] Ankem Kiran , V Thrimurthulu "Verification of AMBA AHB2APB bridge using Universal Verification Methodology". vol 04, no.12. 2016.

[7] K. Swetha Reddy, Punna Soujanya, D. Kanthi Sudha Sudha "Asic Design and Verification of AMBA APB Protocol using UVM" IJCSNS International Journal of Computer Science and Network Security, Vol 9,no. 9,July 2020.

[8] Vani. R.M and M. Roopa "Design of AMBA Based AHB2APB Bridge" Vol. 10, no.11, November 2010.

[9] Bhagvati Panchal, Yogesh Parmar, Haresh Suthar. "Implementation of AMBA Based AHB2APB Bridge" International Journal of Recent Technology and Engineering (IJRTE), vol 8, no. 6, March 2020.

[10] A, V. Jayashree, "VLSI Design of AMBA Based AHB2APB Bridge", International journal of VLSI design and communication systems (VLSICS). 30 June 2018.

[11] Prarthi Bhatt, Devang shah, "Design AMBA Based AHB to APB Bridge Using Verilog HDL".. Institute of Engineering and Technology. Journal of emerging technologies and innovative research. Vol 9.no.7 2020.