System Verilog/UVM Verification of AMBA APB Protocol

Prameeth.H[1], Aayushii Goswami[2], Prajwal.M[3], Vikas.N.G[4], Dr. Rachana.S. Akki[5]

[1] *Student, Department of Electronics and Instrumentation Engineering, RVCE, Bengaluru, India*
[2] *Student, Department of Electronics and Instrumentation Engineering, RVCE, Bengaluru, India*
[3] *Student, Department of Electronics and Instrumentation Engineering, RVCE, Bengaluru, India*
[4] *Student, Department of Electronics and Instrumentation Engineering, RVCE, Bengaluru, India*
[5]*Assistant Professor, Department of Electronics and Instrumentation Engineering, RVCE, Bengaluru, India*

Abstract - *The advancements in Very Large-Scale Integration (VLSI) technology have enabled packaging of billions of transistors on a single chip. Consequently, this has led to the increase in complexity of the System-on-Chip (SoC) design. One of the major components of SoC are the bus protocols which are the components that assist in communication on-chip or off-chip.*

In the scope of our research, we focus on a widely used on-chip bus protocol used for connection and management of different modules on a SoC i.e., AMBA (Advanced Microcontroller Bus Architecture). AMBA has several versions including AHB, APB, AXI etc. AHB, AXI are high performance system buses used for interconnecting CPU cores, DMA etc. Hence due to the wide scale usage of AMBA APB protocol it is essential to reduce the verification time to meet design time constraints.

This paper presents the architecture of the AMBA APB bus protocol and verifies the design using a custom built UVM based testbench to develop a standard AMBA APB verification IP(VIP) and discuss the obtained results.

Keywords: ***AMBA, VLSI, VIP, SoC, APB, UVM, Design Verification.***

1. INTRODUCTION

With the advancement of deep-submicrometric technology, it is now possible to design and build a system-on-a-chip (SoC) with several intellectual-property (IP) cores while fulfilling short time-to-market requirements.

Although using reusable IP cores can cut down on design time, the SoC's high complexity means that testing time is greatly enhanced [1]. In order to sustain in the developing silicon industry, verification quality must be improved while testing costs are kept low. To reduce silicon overhead caused by design-for-testability (DFT), it becomes extremely advantageous to reuse on-chip functional blocks as much as feasible in order to achieve the shortest feasible test time [2]. Therefore, the most crucial step in the VLSI design process is verification. Its objective is to determine errors in the RTL (Register Transfer Level) design early on so that they don't turn out to be destructive later on in the design process. The verification process usually takes up approximately 70% of the total time [3].

The verification process is similar to how a design is created. A designer reads a block's hardware Configuration and human language definition is interpreted, then the logic is written in a format, normally RTL text as represented in the SoC design flow in figure 1[4]. To do the same, one must be familiar with the initial input, the transformation function, and the output format. This interpretation is always ambiguous, perhaps due to different possibilities in missing details in the original document, or contrasting depictions. Therefore, the first step of design is to understand the design under verification. In our research we focus on the AMBA APB Protocol which is our design under test.

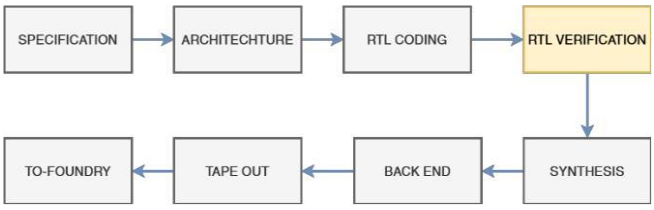

Figure1: SoC Design Flow [4]

The Advanced Microcontroller Bus Architecture is a standard for designing and developing embedded processors. AMBA helps in modular system design and is highly reusable [5]. The peripherals including timers, UART, PIO, Keypad are of low bandwidth and do not require a pipelined bus interface and AMBA APB, which is also non-pipelined, caters to this need [6]. On the other hand, the CPU(ARM) cores, DMA, high bandwidth memory require a high performance, high bandwidth bus and AMBA AHB caters to this need. All the transitions and transactions are associated with the positive clock edge

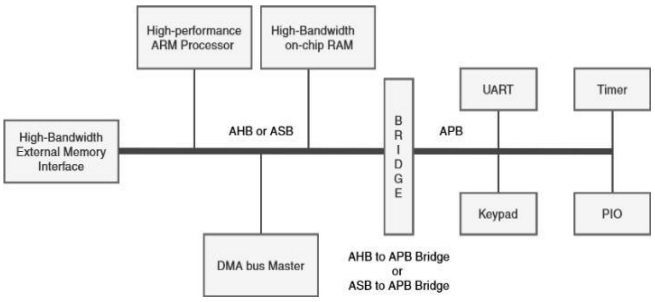

Figure 2: AMBA Bus Architecture [10]

Figure 2 shows the AMBA architecture. Notice the presence of two buses, namely AHB and APB buses. As the CPU(ARM) cores, DMA, high bandwidth memory demand high performance, they are connected to AHB bus while the low bandwidth peripherals are interconnected through APB bus [7]. There is an AHB to APB bridge which connects AHB and APB. All APB connected peripherals act as slaves while the AHB-APB bridge (simply APB bridge) acts as the Master and initiates all the transactions [8].

This paper presents the AMBA APB bus protocol architecture and proposes a design verification strategy using UVM-based custom test bench for AMBA APB verification IP (VIP) development and discusses the results obtained.

2. METHODOLOGY

In order to perform design verification, it is essential to know the working and operation of the various components and hierarchy of the AMBA APB Architecture. In this section as displayed in Figure 2, we briefly discuss about the various APB signals, functioning of AMBA APB Master and Slave, the 3 APB Operating states as well as the Read and Write transactions. After having a good understanding of the various details and critical design aspects of the AMBA APB protocol we proceed to implement a suitable verification strategy after understanding the various aspects of UVM and finally, running the regression to obtain and discuss the results.

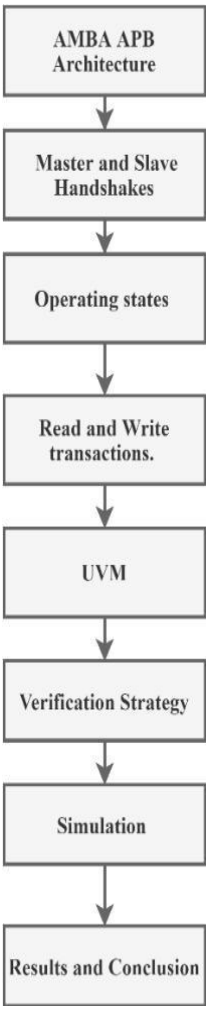

Figure 3: Methodology of research

2.1 APB Master

AMBA that connects to low-level peripheral devices has APB as a part of it. AMBA-APB is made up of two parts: an APB Master/Bridge and a Slave APB, and it is used for connecting a large number of slaves.

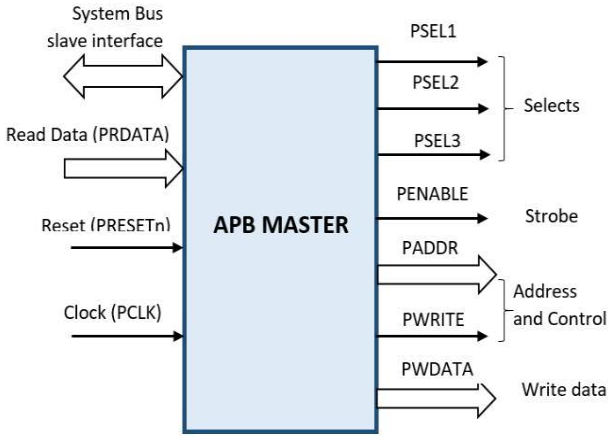

Figure 4: APB Master [10]

The APB bridge/Master and APB slave are seen in Figure 2 as block diagrams. On AMBA APB, the APB bridge is simply a bus master. Furthermore, the APB bridge serves as Slave on the high-level system shuttle.

The APB Bridge transfers data/addresses from the System bus to the APB and performs the following tasks.

[1] It latches the address and keeps it current for the duration of the transfer

[2] The address is decoded and a peripheral pick, PSELx, is created. During a switch, only one select signal can be active.

[3] The data is written to the APB using this method.

[4] The APB data is transferred to the system bus for reading.

2.2 APB Slave

APB Slave is simple and have a flexible interface which can be used to multiple slaves. The description of the APB Slave is explained below.

[1] On either PCLK has rising edge or when HIGH signal is seen in PSEL.

[2] On either PENABLE has rising edge or when HIGH signal is seen in PSEL.

[3] The address PADDR, the select signal PSELx and the PWRITE write signal can be integrated to decide whether the write operation updates the register.

[4] The data bus drives the data when PWRITE is LOW signal and both PENABLE and PSELx are HIGH signal for the read transfers. The PADDR decides which register should be read.

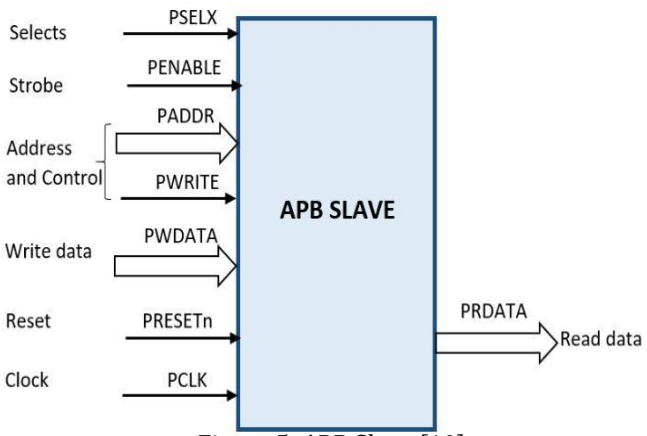

Figure 5: APB Slave [10]

2.3. APB Operating States

There are 3 states in the operating status of an APB as listed below.

[1] STATE 1-IDLE

[2] STATE 2-SETUP

[3] STATE 3-ENABLE OR ACCESS

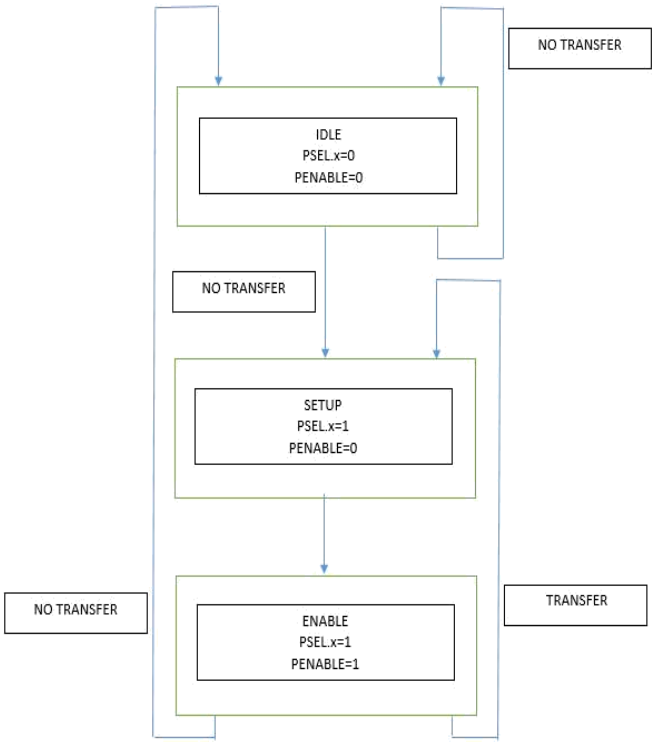

Figure 6: The 3 states of operating Status [11]

[1] **STATE 1- IDLE***:* There will be no data transfer in the IDLE state, and considered as default state.

[2] **STATE 2- SETUP***:* In this state relevant PSLEL.x signal is stated and maintained high, the bus is seen for only one clock cycle in SETUP state, and always shift to ENABLE state in the following uprising clock edge.

[3] **STATE 3- ENABLE/ACCESS:** During this state PENABLE signal will be stated and also maintained high. In the course of transition to ENABLE state from the SETUP state, the write, address including the select signal remain stable. The bus is seen for only one clock cycle in ENABLE state, and shift to SETUP state if no additional transfers are needed. During the transition to SETUP state from ENABLE state, the write, address and the select signals can glitch.

2.3 Write Cycle

In the operation of WRITE transfer, the PADDR, PSEL, PWRITE and the PWDATA signals are stated and made high at T1 edge of clock, this is called as SETUP cycle.

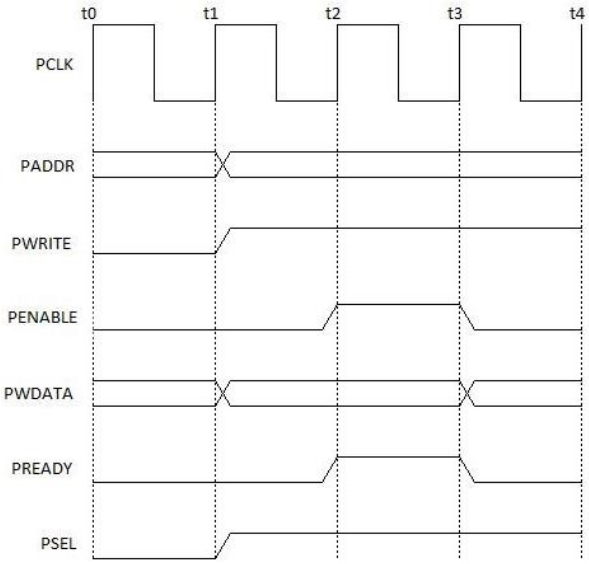

Figure 7: Write Cycle [11]

The PREADY and the PENABLE signal are stated and made high in T2 which is the upcoming rising edge of clock, this is called as ENABLE/ACCESS cycle. At the next clocks rising edge T3, disable PENABLE signal and if further data has to be transferred, a high-low transition happens on the signal PREADY.

2.4 Read Cycle

In the operation of read, the PADDR, PSEL, PWRITE and the PENABLE signals are stated and made high at the T1 edge of clock, this is called as SETUP cycle.

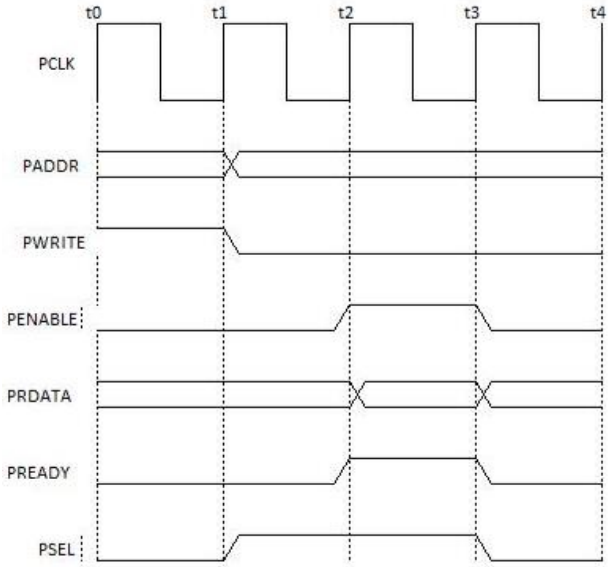

Figure 8: Read cycle [11]

The PREADY, PENABLE are maintained high also the PRDATA is read in this period, this is called as ENABLE/ACCESS cycle.

3. IMPLEMENTATION

As an Engineer doing verification, one has to first go through the specifications of the hardware, construct a plan to verify, then use them to create tests that demonstrate the RTL code correctly to implement the functionality [4]. By this result, one must not just comprehend the design and the intent, as the designer may not have considered all of the corner test cases. So considering all those is necessary.

However, the complexity of the integrated circuit (IC) has increased due to an increase in transistors density in the IC as well as smaller feature sizes and advanced design tools. Integrated Circuits have entered the era of SoC wherein all the components of a computer or a system are integrated into a chip. As a result, the likelihood of faults in the design is increased [9]. As a result, it became important to verify the entire system, its components and communication protocols thoroughly to avoid design re-spin in the later stages.

The Industry standard UVM (Universal Verification Methodology) allows the Engineers to build and reuse verification IP (VIP) and verification environments quickly [14].

It is now an IEEE standard and consists of a collection of class libraries specified using System Verilog (IEEE 1800) syntax and semantics. UVM's key goal is to provide an API platform that can be used through various projects to assist businesses in developing flexible, reusable, and scalable testbench structures [12-13].

It consists of a System Verilog-coded base class library. By expanding these classes, the verification engineer may construct various verification components. UVM also has a number of helpful verification features, such as the use of macros to execute complex functions and a factory for object development. The UVM classes used to verify the designing of the APB protocol are stated below.

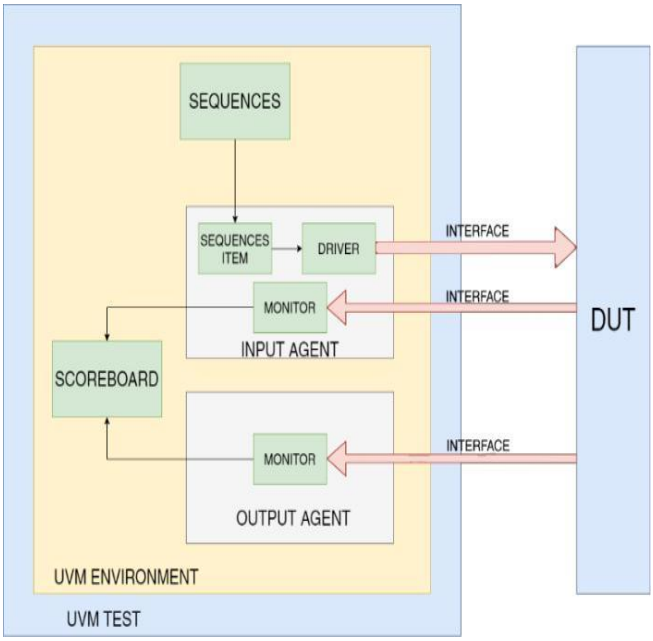

Figure 9: UVM Verification components [15]

3.1 UVM Verification components

[1] Sequence item

The uvm sequence item is used to expand the transactions. The address and data are both randomly generated by this part. The field automation macros are added to this class's data representatives.

[2] Sequences

A collection of transactions is referred to as a sequence. Users in the sequence class will generate complex stimuli. These sequences can be mixed, randomized, and expanded to generate new sequences.

[3] Sequencer

The UVM sequencer serves as a link between the driver and the chain. It sends the transaction to the driver for processing and receives the driver's response. It also serves as an arbitrator for running many sequences in parallel.

[4] Driver

The driver starts the next transaction request and sends it down to the lower-level components. It's made by expanding the uvm driver driver.

[5] Collector and Monitor

The collector derives signal data from the bus, translates it to transactions, and sends it to the display through the analysis port for comparison.

[6] Agent

The verification components - engine, monitor, collector, and sequencer are all instantiated by the agent. It also uses TLM

connections to bind these components. The agent can operate in either active or passive mode. The agent instantiates the driver, sequencer, collector, and monitor in the active mode of operation, while only the monitor and collector are instantiated and configured in the passive mode of operation.

[7] Environment

The Environment class creates and configures all of the sub-components, including the agents, drivers, and monitors.

[8] Test

The uvm_component is used to extend the uvm_test. For a specific verification environment, different testcases may be developed [15].

3.2 Verification Strategy

The UVM testbench is written which models APB slave as a memory. The scoreboard component of the UVM testbench has a local memory:

1. Which is updated in case of write operation along with APB slave memory (DUT)

2. Whose data is compared against the prdata from DUT, for the randomised address, indicating a READ match/fail.

Two sequences namely apb_read_sequence are used for generation of stimulus following the APB bus protocol explained the above sections.

4. SIMULATION OF TESTBENCH

On running the regression using the previously mentioned strategy we acquire the waveforms for the read and write operation along with a detailed

scoreboard and UVM report summary which shall be discussed in the following sections.

[1] Write Operation

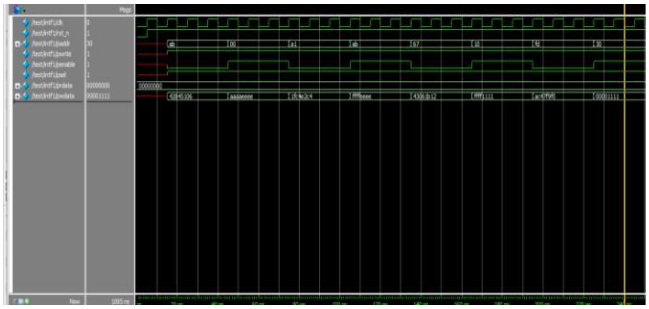

Figure 10: Write operation to the AMBA APB slave

Figure 10 shows the write operation to APB slave. As PWRITE=1, this operation is a APB slave WRITE operation. As PSEL=1, the data gets written when PENABLE=1. Therefore, valid data gets written for addresses 00h, ABh, 10H, 30h as AAAAEEEEh, FFFFEEEEh, FFFF1111h, 00001111h respectively. Note that PSEL is tied to logic 1.

[2] Read Operation

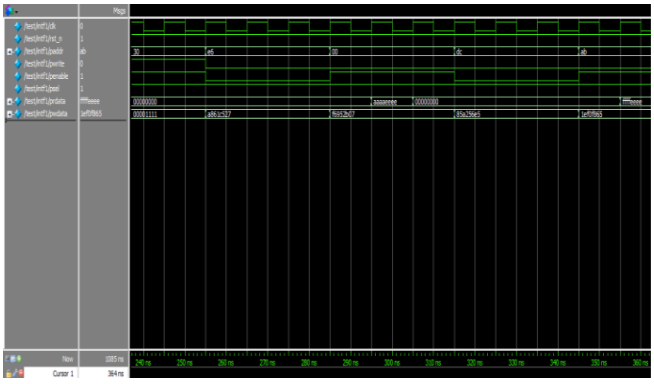

Figure 11: Read operation from slave

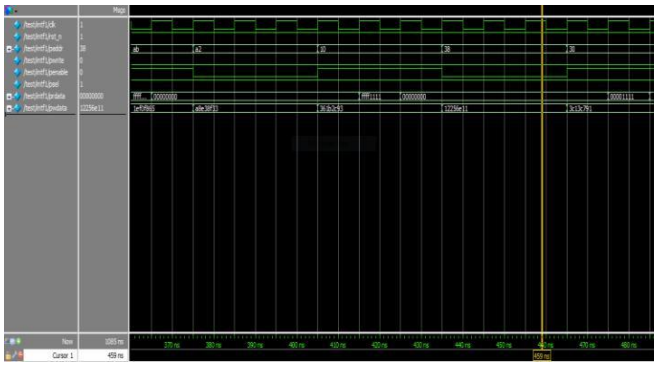

Figure 12: Read operation from slave

From figures 11 and 12 it can be observed that PWRITE=0, indicating a read operation from slave. The data gets read from slave into PRDATA when PENABLE=1. Therefore, value of PRDATA changes as AAAAEEEEh, FFFFEEEEh, FFFF1111h, 00001111h for read addresses 00h, ABh, 10h, 30h respectively. Note that PSEL is tied to logic 1

```
/home/runner/apb_scbd.svh(29) @ 305: uvm_test_top.e0.sc0 [SCBD] readdata match
/home/runner/apb_scbd.svh(30) @ 305: uvm_test_top.e0.sc0 [SCBD] local scoreboard memory for address 00 is aaaaeeee,prdata from dut is aaaaeeee
/home/runner/apb_scbd.svh(29) @ 365: uvm_test_top.e0.sc0 [SCBD] readdata match
/home/runner/apb_scbd.svh(30) @ 365: uvm_test_top.e0.sc0 [SCBD] local scoreboard memory for address ab is ffffeeee,prdata from dut is ffffeeee
/home/runner/apb_scbd.svh(29) @ 425: uvm_test_top.e0.sc0 [SCBD] readdata match
/home/runner/apb_scbd.svh(30) @ 425: uvm_test_top.e0.sc0 [SCBD] local scoreboard memory for address 10 is ffff1111,prdata from dut is ffff1111
/home/runner/apb_scbd.svh(29) @ 485: uvm_test_top.e0.sc0 [SCBD] readdata match
/home/runner/apb_scbd.svh(30) @ 485: uvm_test_top.e0.sc0 [SCBD] local scoreboard memory for address 30 is 00001111,prdata from dut is 00001111
```

Figure 13: UVM Scoreboard.

Finally, it can be clearly seen that PRDATA equals the slave data for valid memory addresses in the UVM scoreboard shown in figure 13.

6. RESULT AND DISCUSSION

The UVM report indicates the summary of results of the applied verification strategy after running the simulation. In figure 14 we present the obtained UVM report that shows 63 UVM info(uvm_info) statements. Among these, 16 have been issued from UVM_DRIVER,20 from UVM_MONITOR and 24 from UVM_SCOREBOARD.

UVM_ERROR=0 and UVM_FATAL=0 indicate that there no errors whatsoever and the verification of the stated design was performed successfully.

The verification strategy utilized in this paper covered all the critical aspects that are needed to be satisfied for the correct functioning of the AMBA APB Protocol. The timing diagrams for the read and write operations were in tandem with the requirements and constraints of the protocol, ensuring successful and valid transaction of data between the components.

```
# KERNEL: --- UVM Report Summary ---
# KERNEL:
# KERNEL: ** Report counts by severity
# KERNEL: UVM_INFO :   63
# KERNEL: UVM_WARNING :    0
# KERNEL: UVM_ERROR :    0
# KERNEL: UVM_FATAL :    0
# KERNEL: ** Report counts by id
# KERNEL: [DRV]     16
# KERNEL: [MON]     20
# KERNEL: [RNTST]     1
# KERNEL: [SCBD]    24
# KERNEL: [TEST_DONE]     1
# KERNEL: [UVM/RELNOTES]     1
# KERNEL:
# RUNTIME: Info: RUNTIME_0068 uvm_root.svh (521): $finish called.
# KERNEL: Time: 1085 ns,  Iteration: 56,  Instance: /test,  Process: @INITIAL#23_1@.
# KERNEL: stopped at time: 1085 ns
# VSIM: Simulation has finished. There are no more test vectors to simulate.
# VSIM: Simulation has finished.
```

Figure 14: UVM report

7. CONCLUSION

This paper gives a detailed overview to the AMBA bus protocol and exhaustively discusses the architecture of AMBA APB bus and the Verification Methodology. The APB bus design is implemented in Verilog HDL whereas the verification testbench is written in UVM. The design is verified using this UVM based testbench through Constrained Random Verification (CRV). The simulation results from the above figures prove that data transactions to and from the memory follow the APB protocol and that the data reads and writes occur from the same memory location. The UVM report proves that the scoreboard reads matches the DUT reads after the memory write operations to scoreboard and DUT therefore meeting the requirements of the said protocol constraints.

8. REFERENCES

[1] Y. Zorian, E. J. Marinissen, and S. Dey, "Testing embedded-core based system chips," in Proc. IEEE Int. Test Conf., Oct. 1998, pp. 130–143. J. F. Li, H. J. Huang, J. B. Chen, C. P. Su, C. W. Wu, C. Cheng, S. I. Chen, C. Y. Hwang, and H. P. Lin, "A A hierarchical test methodology for systems on chip," IEEE Micro, vol. 22, no. 5, pp. 69–81, Sep./Oct. 2002.

[2] M. Amory, M. Lubaszewski, F. G. Moraes, and E. I. Moren, "Test time reduction reusing multiple processors in a network-on-chip based architecture," in Proc. Des., Autom. Test Eur. Conf., Mar. 2005, pp. 62–63.

[3] Hubert Kaeslin,"*Top-Down VLSI Design: From Architectures to Gate-Level Circuits and FPGAs",Elsivier, 2015*

[4] Resve Saleh, Steve Wilton, Shahriar Mirabbasi, Alan Hu, "System-on-chip: Reuse and integration" In Proceedings of the IEEE (2006), IEEE, pp. 1050– 1069.

[5] David Flynn, "AMBA: Enabling Reusable On-Chip Designs", IEEE Micro, July/August 2002, pp. 20-27, vol. 17.

[6] Prashant Dwivedi, Neha Mishra, Amit Singh-Rajput, "Assertion & Functional Coverage Driven Verification of AMBA Advance Peripheral Bus Protocol Using System Verilog", International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), 2021

[7] Padmaprabha Jain, Satheesh Rao, "Design and Verification of Advanced Microcontroller Bus Architecture-Advanced Peripheral Bus (AMBA-APB) Protocol", IEEE- 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV).

[8] Fu-Ching Yang;Yi-Ting Lin;Chung-Fu Kao;Ing-JerHuang, "An On-Chip AHB Bus Tracer With Real-Time Compression and Dynamic Multiresolution Supports for SoC", IEEE Transactions on Very Large Scale Integration (VLSI) Systems (2011) ,Volume: 19, Issue: 4

[9] Chenghai Ma, Zhijun Liu, Xiaoyue Ma, "Design and implementation of APB bridge based on AMBA 4.0", IEEE International Conference on Consumer Electronics, Communications and Networks (CECNet) 2011

[10] Kiran Rawat, Kanika Sahni , Sujata Pandey, "RTL implementation for AMBA ASB APB protocol at system on chip level", IEEE- 2015 2nd International Conference on Signal Processing and Integrated Networks (SPIN), 2015.

[11] ARM Limited, ARM IHI 0024B "AMBA 3 APB Protocol Specification", 2004.

[12] IEEE Standard for Universal Verification Methodology Language Reference Manual IEEE Std 1800.2-2020 (Revision of IEEE Std 1800.2-2017) Year: 2020

[13] Frank Plasencia-Balabarca, Edward Mitacc-Meza, Mario Raffo-Jara, C. S. Cárdenas, "A Flexible UVM-Based Verification Framework Reusable with Avalon, AHB, AXI and Wishbone Bus Interfaces for an AES Encryption Module", 2019 IEEE Latin American Test Symposium (LATS)

[14] Jaehoon Song, Hyunbean Yi, Juhee Han, and Sungju Park, "An Efficient SoC Test Technique by Reusing On/Off-Chip Bus Bridge", IEEE Transactions on Circuits and Systems Part I: Regular Papers (2009)

[15] Lakhan Shiva Kamireddy, Lakhan Saiteja K, "UVM Based Reusable Verification IP for Wishbone Compliant SPI Master Core", International Journal of VLSI Design and Communication Systems (2018)