MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 34 Issue 2, July-Dec 2024, Pages: 486-496

UVM design functional verification of Advanced Peripheral Bus

Dr.K.Uma Maheshwari
Dept. Of Electronics And Communication Engineering
SRM TRP Engineering College
Trichy, India
umamaheswari.k@trp.srmtrichy.edu.in

Dr. G V Shrichandran
Dept. Of Computer Science And Enginnering SRM IST, Ramapuram
Chennai, India
shrichag@srmist.edu.in

Ms. S.Priyadharshini
Dept. Of Electronics And Communication Engineering
SRM TRP Engineering College
Trichy, India
priyadharshini.acs@gmai l.com

Mr. S.Anbarasan
Dr.Navalar
Nedunchezhiyan College of Engineering
sanbarasan767@gmail.com

Mr. G.Parameswaran
SRM TRP Engineering College
paramesall@gmail.com

Abstract—

In hardware design, functional verification identifies the design mistake descriptions that design engineers deliver. To check the functionality, see if the output matches the expected value, and then modify the design to achieve the DUT (Device under Test) required functionality. The Advanced Peripheral Bus (APB) execution protocol links the peripherals through transport customs. ABP has poor data moving capacity and low data transfer volume. These test cases cover manifold write transactions with and without wait states, multiple read and write transactions with and without a wait state, and single and multiple write transactions with and without a wait. The design was programmed using Verilog HDL and testing was verified on a Verilog test bench. The master and slave mode operation is carried out in which the master sends a packet containing the slave's addresses and control signal to the slave. In the consequences, they comprise the slave's addresses and a control signal to the slave, as they are equivalent. Later, it receives a write-and-read data transaction in which the master provides the address, and then the slave identifies it and transmits it to the master as read data. The APB protocol structure is verified using UVM for applications in ASIC (Application Specific Integrated Circuits) or SOC (System on Chip) configurations.

Keywords—Advanced Peripheral Bus , Device under Test, Verilog HDL component, universal verification methodology.

1. INTRODUCTION

Very Large Scale Integration (VLSI) working on Integrated Circuits (IC) through integrating millions of transistors into a mono chip. VLSI technology evolved when microchips became extensively employed in 1970, enabling the fast progress of mutual semiconductor and telecommunications technologies. Before VLSI technology, most Integrated Circuits (IC) had limited capability. Logic circuits, RAM, and ROM comprise the electrical circuit. Using VLSI technology, these are all merged into a single chip as a System- on-chip (SOC). A SOC, or single-chip integrated circuit, contains the whole component. Today, the ARM business standard for Application Specific Integrated Circuits (ASIC) construction for diverse applications is the gold standard. Reusable Intellectual Property (IP) is suitable for upgrading an ARM and is especially important for any ASIC configuration emphasis. Many validated IP square configurations depend on Hardware Description Language (HDL) rather than schematic representations for IP blocks and connections. The Advanced Microcontroller Bus Architecture (AMBA) is an on-chip technology that provides a high-performance communication bus with high power and bandwidth. The Advance Peripheral Bus (APB) and the Advance High-Performance Bus (AHB/ASB) are the two protocols produced by ARM technology. All external devices connected to the bus, including a high-performance ARM processor and high-bandwidth external memory, remain connected to the on-chip RAM via the advanced high-performance ARM processor. All external devices connected to the bus, including a high performance ARM processor and a highbandwidth external memory, are connected to the on-chip RAM via the Advanced high-performance bus. It connects to the outside bus, with the system bus taking independent read-and-write buses. It acts as an interface between the AHB and the bus along both on-chip and off-chip memory supported by AHB. The APB is a low-peripheral device with minimal power and bandwidth needs employed by the Internet of Things. It supports a broad range of systems on chips, such as subsystems, mobile phones, and networking systems on chips, all of which differ in power, performance, and available space.

The AHB to APB module's design is explained by T. Koundinya [1] as Verilog is used to code the AHB module and the AHB to APB Bridge. The AHB driver or monitor designed here does not use an FPGA and only discusses RTL simulation. APB technique is further discussed by dr, dileepreddy.et.al study [2] along with a comparison of AHB and ASB. Verilog code used for coding and state diagram of the APB and required signals and its functions discussed in the article. The APB protocol is the only topic covered in the document, although it is helpful in understanding. AHB module and the VHDL code described by Varshavishwarkama.et.al publication [3]. The AHB arbiter's design and its associated AHB signals are well understood. You can apply the AHB module by understanding it with the aid of this document. The study of Pravin S. Shete.et.al [4] describes how understanding the various signals of the AHB master allowed for the development of an efficient Finite State Machine (FSM) for the AHB master. Designing an effective FSM and, consequently, the AHB master requires the operation of the AHB master. The scheme to create the AHB module is the only option, so AHB arbiter architecture is highlighted in the paper of Pravin S. Shete.et.al [5] description of the AHB arbiter design. It also explains the arbiter works and the focal study to build a Verilog arbiter for an AHB. The study describes the growth of burst transfer design for AHB's high performance as mentioned by MitalMungra.et.al [6]. The emphasis of this research is on burst transfers and their ability to maximise AHB performance. In this work, the AHB burst performance programmed using Verilog. The study of Radhika Koti and Divya Meshram [7] contrasts the AHB, ASB, and APB AMBA bus protocols from version 2.0. Their performance application is the basis for comparison. In this investigation, the Verilog code for AHB burst performance applied. The article of Vani.R.M and M.Roopa [8] defines the pattern of AHB to APB segment for diverse frequencies and phases. The article explains the layout of AHB to APB and how they work at different frequencies. The article helps to understand the interplay between the two protocols and describes the design. Research by Anurag Shrivastava G.S. Tomar and Ashutosh Kumar Singh illustrates the functioning of several AMBA protocol variants and examined. [9]. The approach for AHB master wrappers described in the publication of Marc Bertola and Guy Bois.et.al [10] for usage in Intellectual Property (IP) cores.

2. METHODOLOGY

2.1. APB DESIGN

As a part of the AMBA 3 protocol family, the APB offers an inexpensive interface that is easier to use and consumes less power. Since APB's protocol not pipelined further, thus it connects to peripherals with limited bandwidth that does not need a pipelined bus interface to improve the performance. Since every signal transition in an APB peripheral linked to the clock's rising edge, integrating them into any design flow is straightforward. APB connects together the AMBA Advanced Extensible Interface (AXI) and AMBA AHB-Lite. APB is also employed to gain access the peripheral devices' programmable control registers.

2.2. APB Block Diagram

The design specifications give rise to the Advanced Peripheral Bus (APB). The fundamental interface signals portrayed in the AMBA APB's basic block design in Figure 2. The APB slave receives 32-bit PADDR and PWDATA inputs from the bridge together with input control signals PCLK, PRESET, PSEL, PENABLE, and PWRITE, and it outputs 32-bit PRDATA.

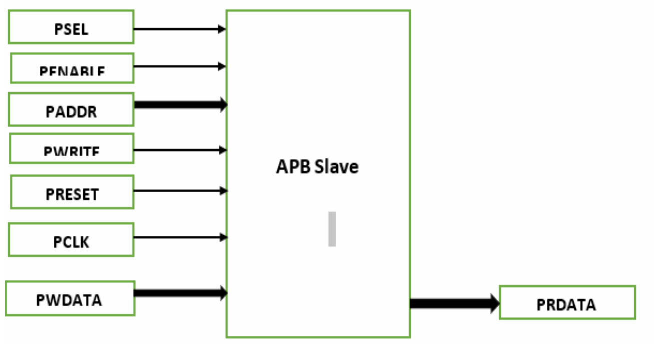


Fig 2: APB Functional diagram

The APB slave receives 32-bit PADDR and PWDATA inputs from the bridge together with input control signals PCLK, PRESET, PSEL, PENABLE, and PWRITE, and it outputs 32-bit PRDATA as listed in Table 1.

Signal	**Signal Name**
PCLK	Clock signal
PRESETn	Reset signal
PADDR	32-bit address bus
PSELx	Select signal
PENABLE	Enable signal
PWRITE	Direction signal
PWDATA	32-bit Write Data bus
PREADY	Ready signal
PRDATA	32 bits read data bus

Table 1. APB control signals

Advanced Peripheral Bus (APB) Operating States
The fundamental state machine that depicts how the peripheral bus functions viewed in Figure 3. There are three states: SETUP, ACCESS, and IDLE.

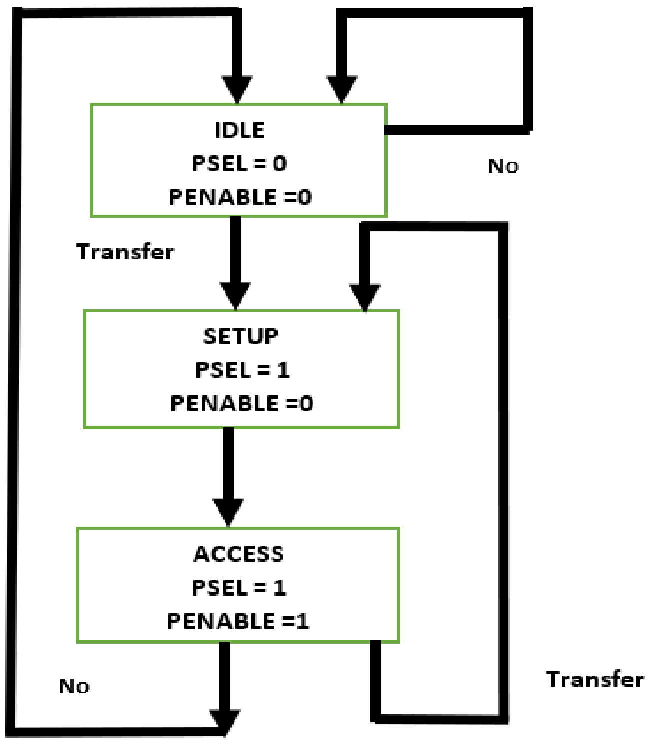


Fig 3: Flow Chart

The default state, known as the idle state or IDLE, is one in which the protocol not activated for performance. The SETUP phase gets started when the PSEL signal asserted. When data transfer is necessary, the bus goes into the SETUP phase. At this stage, the PWRITE, PADDR, and PWDATA are also made available. After one clock cycle in the SETUP phase, the bus moves to the ACCESS state on the subsequent rising edge of the clock. The ACCESS phase begins with the assertion of the PENABLE signal. When the SETUP phase changes to the ACCESS phase, all control, address, and data signals stay the same. The PRDATA present on the bus at this time in the event of a read operation. Furthermore, while the data is on the bus, the PENABLE signal remains high for one clock cycle. If no more data transfer is required, the bus will enter the IDLE state. If more data transfer is required, the bus moves on to the SETUP phase.

2.3. Write Cycle of APB

The PSEL, PWRITE, PADDR, and PWDATA control signals assert during the write transfer process at the T1 clock edge, also known as the SETUP cycle. At the next rising edge of clock T2, the PENABLE and PREADY signals are recognized and reported as an ACCESS cycle, as seen in Figure 4. If more data transfer is required, the PREADY signal transitions from high to low at clock edge T3 and the PENABLE signal is deactivated at this point.

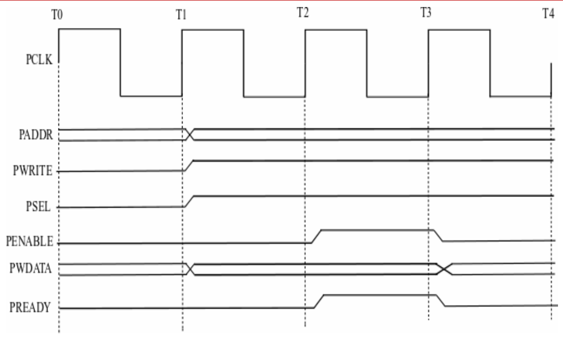


Fig 4: Write cycle of APB

READ THE CYCLE OF APB

During the reading operation, the control signals PSEL, PENABLE, PWRITE, and PADDR were asserted at clock edge T1 (SETUP cycle). The variables PENABLE and PREADY claimed, then the PRDATA is read at clock edge T2, also referred to as the access cycle depicted in Figure 5.

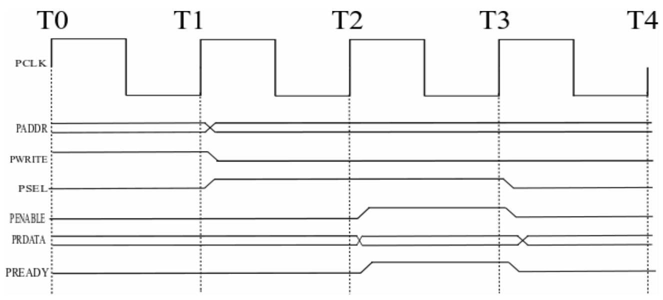
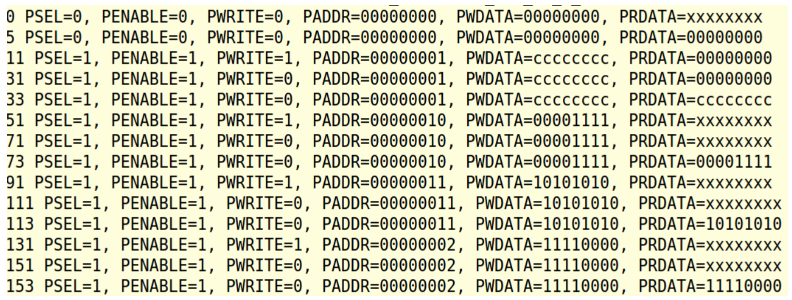


Fig 5: Read cycle of APB

2.4. RESULTS AND DISCUSSION

The design and testbench written in Verilog code constructed using ICARUS Verilog. The read-and-write operations are evident in the results. Figure 6 illustration makes it abundantly evident that when PWRITE=1, the data and address are written on the same clock edge. When PWRITE = 0 after the address is sent, the data is read on the next clock edge.



```
0 PSEL=0, PENABLE=0, PWRITE=0, PADDR=00000000, PWDATA=00000000, PRDATA=xxxxxxxx
5 PSEL=0, PENABLE=0, PWRITE=0, PADDR=00000000, PWDATA=00000000, PRDATA=00000000
11 PSEL=1, PENABLE=1, PWRITE=1, PADDR=00000001, PWDATA=cccccccc, PRDATA=00000000
31 PSEL=1, PENABLE=1, PWRITE=0, PADDR=00000001, PWDATA=cccccccc, PRDATA=00000000
33 PSEL=1, PENABLE=1, PWRITE=0, PADDR=00000001, PWDATA=cccccccc, PRDATA=cccccccc
51 PSEL=1, PENABLE=1, PWRITE=1, PADDR=00000010, PWDATA=00001111, PRDATA=xxxxxxxx
71 PSEL=1, PENABLE=1, PWRITE=0, PADDR=00000010, PWDATA=00001111, PRDATA=xxxxxxxx
73 PSEL=1, PENABLE=1, PWRITE=0, PADDR=00000010, PWDATA=00001111, PRDATA=00001111
91 PSEL=1, PENABLE=1, PWRITE=1, PADDR=00000011, PWDATA=10101010, PRDATA=xxxxxxxx
111 PSEL=1, PENABLE=1, PWRITE=0, PADDR=00000011, PWDATA=10101010, PRDATA=xxxxxxxx
113 PSEL=1, PENABLE=1, PWRITE=0, PADDR=00000011, PWDATA=10101010, PRDATA=10101010
131 PSEL=1, PENABLE=1, PWRITE=1, PADDR=00000002, PWDATA=11110000, PRDATA=xxxxxxxx
151 PSEL=1, PENABLE=1, PWRITE=0, PADDR=00000002, PWDATA=11110000, PRDATA=xxxxxxxx
153 PSEL=1, PENABLE=1, PWRITE=0, PADDR=00000002, PWDATA=11110000, PRDATA=11110000
```

Fig 6: Read and Write Cycle results

The VLSI design flow's most crucial step is verification. To prevent them from being damaged later in the design process, it seeks to identify RTL (Register Transfer Level) design problems are early on. The verification procedure is shown in Figure 7 takes up about 70% of the total time. Therefore, it is the process that takes the longest. Better design tools, more transistor density, and smaller feature sizes contribute to increase the integrated circuit (IC) complexity. This inturn increases the likelihood that the design may have bugs. Consequently, the requirement for IC verification became imperative.

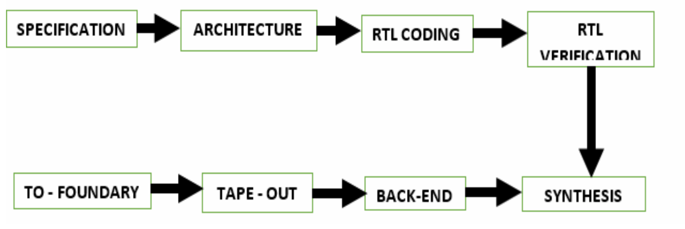


Fig 7: RTL Verification Design Flow

The RTL (Register Transfer Level) design is verified using the universal verification methodology (UVM), a standard verification approach. It consists of a base class library written in System Verilog code. By extending these classes, the verification engineer can produce various verification components. Additionally, UVM offers a variety of other helpful verification capabilities, including the ability to create objects using a factory and employ macros to construct complex functions . Figure 8 displays the several UVM verification elements developed to validate the APB design.

SEQUENCE ITEM

The uvm_sequence_item is the source of the extended transactions. The address and data are in random using this component. This class's data members are subject to the field automation macros.

SEQUENCES

A sequence consists of a group of transactions. The sequence class's users are capable of creating intricate stimuli. They could remain concatenated, expanded, and randomized to produce new sequences.

Sequencer

The UVM sequencer organizes the driver and sequencer. After providing the driver with the transaction to execute, it waits for the driver's response. It acts as a mediator between numerous parallel sequences as well.

Driver

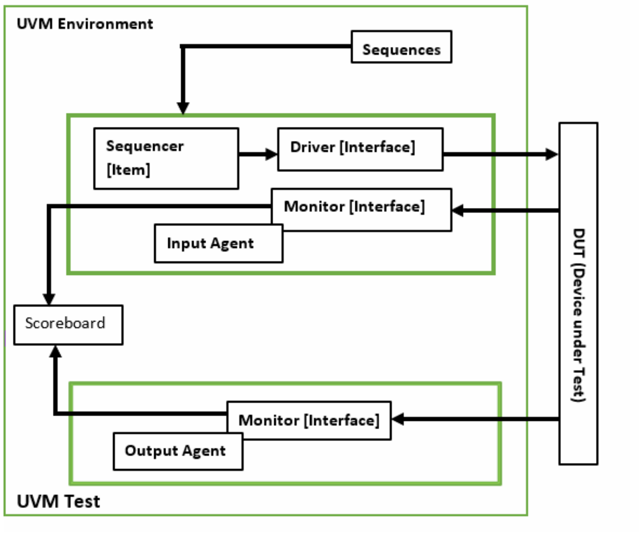
The driver initiates the next transaction request, which is forwarded to the components at a lower level. By expanding the uvm_driver, it is produced.

COLLECTOR AND MONITOR

TO ENABLE FURTHER SIMILARITY, THE COLLECTOR TRANSMITS THE TRANSACTION-BASED DATA FROM THE BUS SIGNAL DATA TO THE MONITOR.

AGENT

For the verification process from Figure 8, the agent generates the sequencer, collector, monitor, and driver. Additionally, it uses TLM connections to link these parts together. The functioning modes of the agent can be active or passive. While an agent is running in the active mode, it creates a driver, a sequencer collector, and a monitor. The monitor and collector are the only components that are built and configured when it is operating in passive mode.



ENVIRONMENT

ALL OF THE SUB-COMPONENTS, INCLUDING DRIVERS, MONITORS, AND AGENTS, ARE INSTANTIATED AND CONFIGURED BY THE ENVIRONMENT CLASS.

TEST

The uvm_component extends to the uvm_test. It is possible to create several test cases for the specified verification environment.

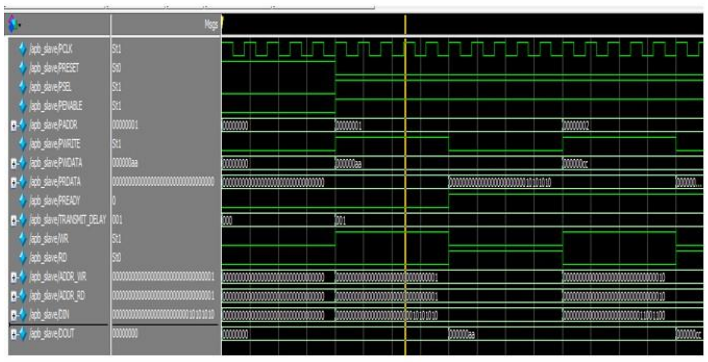

Msgs
/apb_slave/PCLK St1
/apb_slave/PRESET St0
/apb_slave/PSEL St1
/apb_slave/PENABLE St1
/apb_slave/PADDR 00000001
/apb_slave/PWRITE St1
/apb_slave/PWDATA 000000aa
/apb_slave/PRDATA 00000000000000000000000000000000
/apb_slave/PREADY 0
/apb_slave/TRANSMIT_DELAY 001
/apb_slave/WR St1
/apb_slave/RD St0
/apb_slave/ADDR_WR 00000000000000000000000000000001
/apb_slave/ADDR_RD 00000000000000000000000000000001
/apb_slave/DIN 00000000000000000000000010101010
/apb_slave/DOUT 00000000

Fig 9: APB write operation verification

The simulation results achieved by setting up a verification environment indicated in Figure 9. The modeling results include additional signals like DIN, DOUT, ADDR_WR, and ADDR_RD. These signals originate from the memory that has been linked to the APB bus.

The data 000000aa is written to memory location 00000001 in Figure 9, and the same data is read from the same memory address in Figure 10. During the writing process shown in Figure 9, which is triggered by the high signal on the PWRITE port, the data transferred to the PWDATA port is communicated to the storage device through the memory's DIN port. By sending an address to the PADDR port, the data is communicated to the ADDR_WR port of the memory.

The address applied to the PADDR port, which is transferred to the ADDR_RD port of the memory, during the read operation, as seen in Figure 10. When the PWRITE signal goes low, the memory's RD signal gets high. The data obtained at the DOUT port is read via thememory's PRDATA port.

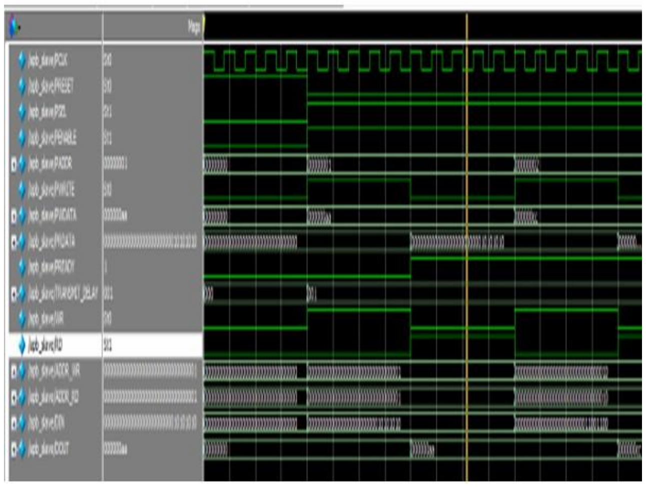


Figure 10: APB read operation verification

The outcomes of the UVM test bench simulation are presented in the UVM report. The UVM report summary produced after completing all UVM phases is displayed in Figure 11. The UVM report description exists in Figure 11 shows 36 information messages, as indicated by the UVM_INFO. Since UVM_ERROR, UVM_WARNING, and UVM_FATAL are equal to zero, the information supplied by the UVM report summary guarantees that the design is error-free and does not result in any warnings or fatal errors.

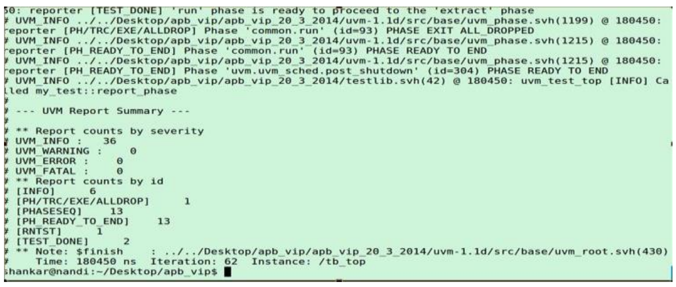

```
50: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO ../../Desktop/apb_vip/apb_vip_20_3_2014/uvm-1.1d/src/base/uvm_phase.svh(1199) @ 180450:
reporter [PH/TRC/EXE/ALLDROP] Phase 'common.run' (id=93) PHASE EXIT ALL_DROPPED
# UVM_INFO ../../Desktop/apb_vip/apb_vip_20_3_2014/uvm-1.1d/src/base/uvm_phase.svh(1215) @ 180450:
reporter [PH_READY_TO_END] Phase 'common.run' (id=93) PHASE READY TO END
# UVM_INFO ../../Desktop/apb_vip/apb_vip_20_3_2014/uvm-1.1d/src/base/uvm_phase.svh(1215) @ 180450:
reporter [PH_READY_TO_END] Phase 'uvm.uvm_sched.post_shutdown' (id=304) PHASE READY TO END
# UVM_INFO ../../Desktop/apb_vip/apb_vip_20_3_2014/testlib.svh(42) @ 180450: uvm_test_top [INFO] Ca
lled my_test::report_phase
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :    36
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [INFO]     6
# [PH/TRC/EXE/ALLDROP]     1
# [PHASESEQ]    13
# [PH_READY_TO_END]    13
# [RNTST]     1
# [TEST_DONE]     2
# ** Note: $finish    : ../../Desktop/apb_vip/apb_vip_20_3_2014/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 180450 ns  Iteration: 62  Instance: /tb_top
shankar@nandi:~/Desktop/apb_vip$ █
```

Fig 11: Summary Report of UVM

CONCLUSION

This article provides an overview of the AMBA bus architecture along with detailed description of the APB bus. According to the standard, the APB bus created using Verilog HDL code, is validated using UVM. The simulation findings demonstrate that data read from one location in memory matches data put to that location in memory exactly. The design is thus correctly functional. The UVM report summary also guarantees the design's functionality. The Advanced Peripheral Bus protocol written in the Verilog programming language, and verification performed using the Verilog Testbench. One advantage of Verilog HDL programming is the ability to produce Tasks for various test cases, which greatly increases the reusability of test bench segments. Designing computer programs with Verilog has the advantage of allowing testbench portions to remain reused more frequently. The interface connects the DUT to the Verilog Testbench, which holds the test program. Testbench examines the protocol's overall functioning for four distinct parameters: the Write, Read, and Error cycles. Test cases address the following topics: grouped read and write values, self-error checking blocks, the protocol's functional soundness by passing a random value with the same index, and the read, write, and error cycles. The re-enactment's outcomes show that there is no distinction between data that has been read from a particular system location and data that is in touch with the allocated memory area. As a result, the design sounds practical. The Xilinx ISE Design tool used for design and verification of the AMBA APB. Future research will focus on power-advancement methods that may be used in different APB modules, resulting in significant improvement in total transport convention exhibitions, further increases the performance.

ACKNOWLEDGMENT

The research was supported, by the Department of Electronics and Communication at SRM TRP Engineering College, Trichy, Tamil Nadu, India. We express our gratitude to our management, faculty, and research scholars in SRM TRP Engineering College, Trichy, Tamil Nadu, India, who provided skill and proficiency that entirely helped to develop the performance of APB.

REFERENCES

Design of AMBA Based AHB2APB Bridge by Vani.R.M andM.Roopain IJCSNS International Journal of Computer Science and Network Security, 2010.

Design & Implementation of Advance Peripheral Bus Protocol by Heli Shah, ChinmayModi, BhargavTarpara in International Journal of Scientific Engineering and Applied Science (IJSEAS),2015.

Implementation of AMBA AHB protocol for high capacity memory management using VHDL by Varshavishwarkama,Abhishekchoubey,ArvindSahu in International Journal on Computer Science and Engineering (IJCSE),2012.

Design of an Efficient FSM for an Implementation of AMBA AHB Master by Pravin S. Shete,Dr. ShrutiOza, in International Journal of Advanced Research in Computer Science and Software Engineering, 2014.

Study Of High Performance AmbaAhb Reconfigurable Arbiter For On-Chip Bus Architecture by Pravin S. Shete,Dr. ShrutiOza, International Journal of Electrical, Electronics and Data Communication, 2014.

Design Incrementing Burst Data Transfer Operation for AMBA-Advanced High Performance Bus by MitalMungra and Assi. Prof. Vishal S.Vora in International Journal of Emerging Trends in Electrical and Electronics (IJETEE), 2013.

An Overview of Advanced Microcontroller Bus Architecture relate on APB bridge by RadhikaKoti, DivyaMeshram in International Journal of Scientific and Research Publications, 2013.

Design of AHB2APB Bridge for different phase and Frequency by Vani.R.M andM.Roopa in International Journal of Computer and Electrical Engineering, 2011.

Performance Comparison of AMBA Bus-Based SystemOn-Chip Communication Protocol by AnuragShrivastava G.S. Tomar, Ashutosh Kumar Singh in International Conference on Communication Systems and Network Technologies, 2011.

A methodology for the design of AHB bus master wrappers by Marc Bertola, Guy Bois in IEEE ,2003.

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 hierarchical test methodology for systems on chip,” IEEE Micro, vol. 22, no. 5, pp. 69–81, Sep./Oct. 2002.

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.

Hubert Kaeslin,”Top-Down VLSI Design: From Architectures to Gate-Level Circuits and FPGAs”,Elsevier, 2015

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

David Flynn, “AMBA: Enabling Reusable On-Chip Designs”, IEEE Micro, July/August 2002, pp. 20- 27, vol. 17.

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

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).

Fu-Ching Yang;Yi-Ting Lin;Chung-Fu Kao;IngJerHuang, “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

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

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.

ARM Limited, ARM IHI 0024B “AMBA 3 APB Protocol Specification”, 2004.

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

Frank Plasencia-Balabarca, Edward MitaccMeza, 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)

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)

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)

ARM, "AMBA APB3 Specification Overview", http://www.arm.com

Samir Palnitkar, "Verilog HDL: A guide to Digital Design and Synthesis (2nd Edition), Pearson, 2008.

Chris Spear, "System Verilog for verification (2nd Edition): A guide to learning the testbench features, Springer, 2008.

Bergeron, "Writing test benches using System Verilog,"Springer, 2009.

Pamula P, Gorthy DP, Ngangbam PS, Alagarsamy A. Verification of SoC Using Advanced Verification Methodology. *Engineering Proceedings*. 2023; 34(1):12. https://doi.org/10.3390/HMAM2-14160