ISSN: 3107-6513

APB Verification Using SystemVerilog

Piali Chakraborty[#1], Syed Lukhman[#2], Tejas K Y[#3]
[#]Department of Electronics and Communication Engineering, Rajeev Institute of Technology, Hassan
[1]pialichakraborty@rithassan.ac.in
[2]syedlukhman18@gmail.com
[3]tky4779@gmail.com

Abstract:

This paper introduces a novel protocol verification method for the Advanced Peripheral Bus (APB) protocol using SystemVerilog. The microprocessor technology and System-on-Chip (SoC) architectures are advancing day by day, so an effective interface between the high-speed devices on the board and comparatively low-speed components is essential for superior performance. Hence, interconnects have a significant role in SoC design. SoCs mostly employ APB on-chip communication protocol to connect low-bandwidth peripherals because APB is a simple and low-power bus protocol. Additionally, functional verification has been recognized as a crucial step in the design cycle since it ensures the proper functioning of APB-based designs. This work establishes an efficient way to verify APB protocol functionality by developing a structured environment that leverages the additional features of SystemVerilog over Verilog specifically intended for verification. The testbench components of SystemVerilog, such as interface, driver, monitor, scoreboard, and assertions, generate a functional coverage report. Simulation results demonstrate that the APB design operates correctly under different test scenarios, ensuring reliable communication between the master and slave components.

Keywords— AMBA APB, System Verilog, Verification, Testbench, Functional Verification, Protocol Validation.

I. INTRODUCTION

The APB protocol is a low-cost interface, optimized for minimal power consumption and reduced interface complexity. The APB interface is a simple, synchronous protocol handling one transfer at a time. Every transfer requires at least two cycles to complete. The APB interface serves the purpose of accessing the programmable control registers of peripheral devices. An APB bridge is usually used to link APB peripherals to the main memory system. For instance, a bridge from AXI to APB connects a number of APB peripherals to an AXI memory system. An APB bridge initiates the APB transfers. APB bridges can also be referred to as a Requester. A peripheral interface responds to requests. APB peripherals can also be referred to as a Completer. This specification will use Requester and Completer.

In modern System-on-Chip (SoC) designs, multiple functional blocks such as processors, memories, and peripheral devices are integrated on a single chip. Standardized on-chip bus protocols play a significant role to establish efficient communications between these blocks. APB is one such widely used protocol, which belongs to the Advanced Microcontroller Bus Architecture (AMBA) family introduced by ARM. APB is designed for low-bandwidth, low-power peripherals and is commonly used to connect components such as timers, UARTs, GPIOs, and configure registers.

While designing the APB protocol, simplicity and power efficiency are the two key considerations. As the burst transfers or pipelining are not supported by this protocol, verifying protocol behaviour becomes easier, along with reduced hardware complexity. APB communication is synchronous and is controlled by a clock signal (PCLK) and an active-low reset (PRESETn). Each APB transfer comprises two distinct phases: the setup phase and the access phase. The transfer process is initiated at

the setup phase by asserting the address (PADDR), control signal PWRITE, and the peripheral select signal (PSEL). In the access phase, the enable signal (PENABLE) is asserted to complete the transfer.

For a hardware design verification process is crucial to check whether the design operates as per the required specifications. Besides, verification identifies functional bugs, timing issues, and protocol violations at an early stage. Detecting errors after fabrication is time-consuming and expensive. Hence, for a VLSI design flow, verification consumes a significant portion of the overall development cycle. For bus protocols such as APB, verification ensures the correct occurrence of read and write transactions, set control signals high at proper times, and accurate data transfer between communicating modules. Generally, the verification process in hardware is accomplished using simple testbenches written in hardware description languages. However, to cope with the gradually increasing circuit complexity, advanced verification languages and methodologies have become necessary. This paper addresses the issue by developing a structured verification environment using hardware description and verification language SystemVerilog, which has many additional features compared to Verilog. By proper utilization of these features like built-in assertions and coverage, powerful verification capabilities, improved code reusability, better support for large, complex designs, simpler RTL coding, object-oriented programming, this work has successfully demonstrated an effective, reliable APB protocol verification method.

II. LITERATURE REVIEW

To progress in the right direction, both the conventional and current advancements in the AMBA bus architecture with different kind of interconnects are reviewed in substantial quantity. It is found that a gap exists in interconnect designs. Functional improvement of the interconnects can enhance the SoC performance to a great extent. Some of the impactful research works in this context are mentioned below.

Shankar, D. Girdhar, N. K. Shukla in this paper analyse the AMBA bus architecture, emphasizing the APB bus. For design and verification purposes, Verilog HDL and Universal Verification Methodology are used, respectively. The correct functionality of APB is proven from the simulated result, that the data read from a particular memory location is the same as the data written to the given memory location. [1].

K. B. Raju, and B. K. Konda in this paper also discusses the AMBA bus architecture and APB bus where a memory controller is introduced in the ABP as an advanced high-performance slave bus. This memory controller is a digital circuit that can be integrated into the system chipset or can be a separate chip. It manages the flow of data to and from the main memory. The authors here used Verilog HDL to design and Xilinx for verification [2].

N. Sahu, A. Kumar and R. Kandari, in this paper, describe a particular type of IP core for APB bus architecture using high level verification methodologies. Here, the authors have used an advanced extensible interface (AXI) along with an advanced high-performance bus APB for interfacing with AMBA APB. APB DUT has been designed on Cadence NcSim using SV and UVM, and its simulations are verified using the Cadence SimVision tool [3].

D. P. Kumar and K. S. Pande have conducted a comprehensive review of relevant noteworthy research works emphasizing the design and verification of the APB protocol. [4].

V. S. Reddy, J. Sachin Rewaria, R. Kumar Thota, C. Anil Kumar and V. Kumar Chikoti, in this paper, designed and verified APB based on a Finite State Machine model. Different operational phases of the structured and standard implementation are delineated by the FSM model. Simulation results demonstrate that the protocol can support the low-power communication behaviour pertinent for modern advanced SoC architectures and

ISSN: 3107-6513

applications like embedded systems or IoT devices
[5].

III. METHODOLOGY

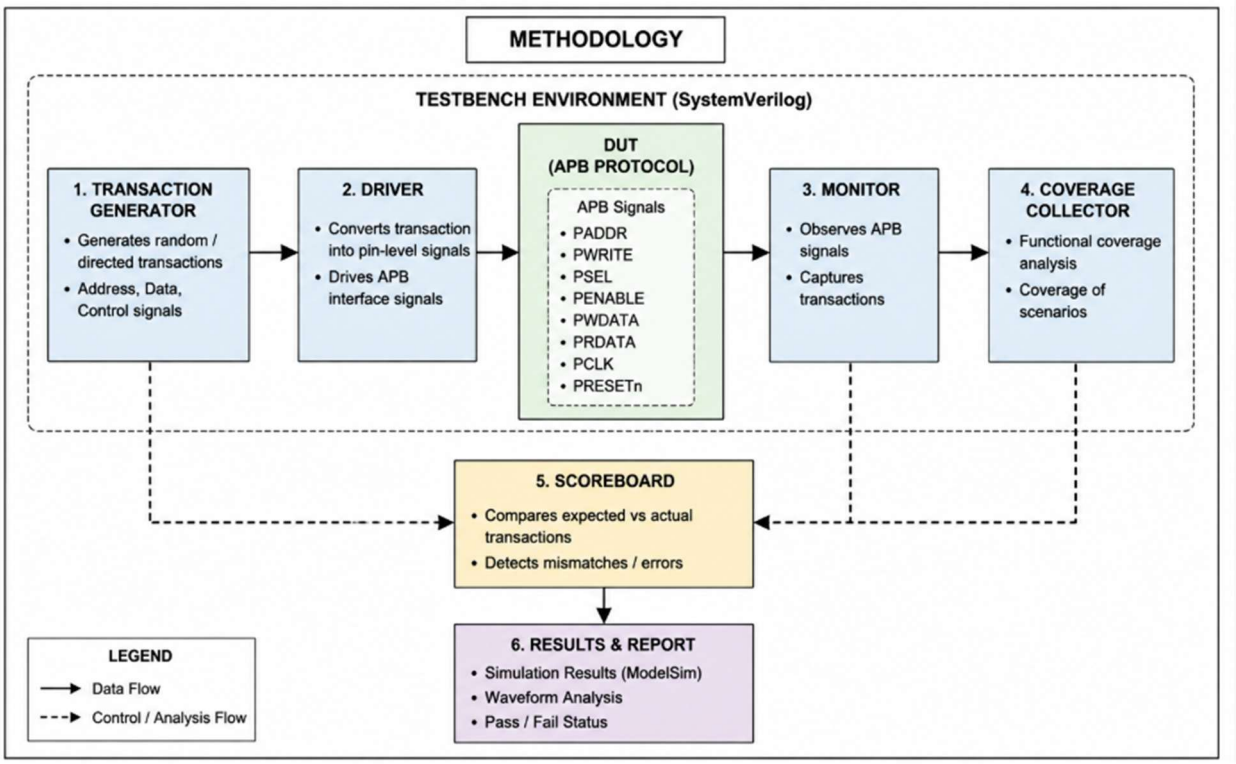


Fig 1: Testbench environment of SystemVerilog

ISSN: 3107-6513

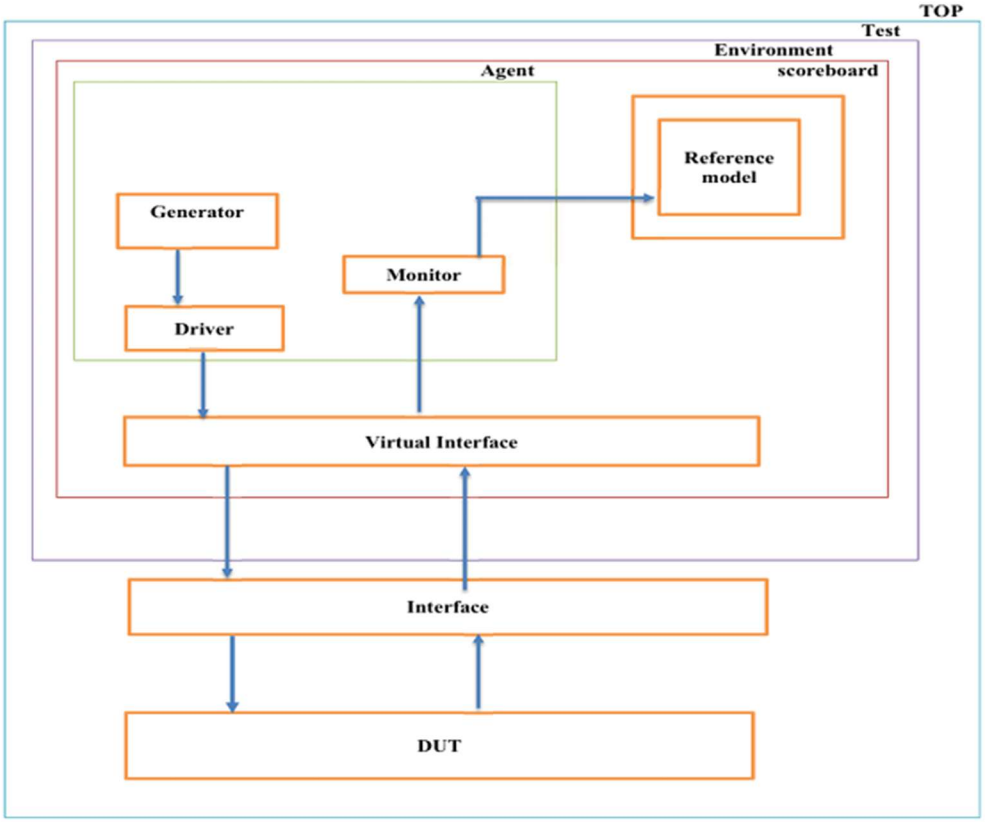


Fig 2: Block diagram of SystemVerilog testbench architecture

The verification methodology for the APB protocol was developed using SystemVerilog to ensure the correct functionality of read and write operations. A structured verification environment is created, consisting of a generator, driver, monitor, scoreboard, and Device Under Test (DUT). The methodology follows a modular approach to improve verification efficiency and error detection capability.

Initially, the APB design module was implemented according to the AMBA APB protocol specifications. A SystemVerilog testbench architecture is then developed to generate and apply different transaction scenarios to the DUT. Different randomized and directed test cases are considered to verify various protocol operations, such as address transfer, data transfer, read/write control, and enable signal generation.

The generator module produces transaction packets containing address, control, and data information. These transactions are sent to the driver, which converts them into APB interface signals and applies them to the DUT. The DUT processes the input transactions and generates corresponding outputs.

The monitor continuously observes DUT signals during simulation and captures transaction activities. The captured data is forwarded to the scoreboard for comparison between expected and actual outputs. Any mismatch detected by the scoreboard is reported as an error, confirming protocol validation.

Functional verification is carried out by applying simulation tools such as QuestaSim. Waveforms are analyzed to validate timing behaviour, read/write operations, and signal synchronization. Coverage metrics were also evaluated to ensure that all important functional scenarios are exercised during simulation. The list of required software and their brief function are given in Table 1.

QuestaSim	QuestaSim is a simulation and verification tool used to test and debug Verilog and SystemVerilog designs.

ISSN: 3107-6513

EDA Playground	EDA Playground is an online platform used to write, simulate, and verify Verilog and SystemVerilog code.
gVim	gVim was used to write and edit Verilog and SystemVerilog source files.

Table1: Required software used for APB design and verification

IV. RESULT AND DISCUSSION

The designed APB verification environment is simulated using SystemVerilog to validate the correctness of APB protocol transactions. The testbench successfully generated read and write operations and monitored the response of the DUT under different scenarios.

1. Functional Verification Results

The simulation results confirm that all APB transactions have been executed according to the protocol specification. During the Write operation PCLK, PENABLE signal becomes High at the positive edge of clock. PWRITE becomes High when it Write the Data in PWDATA with a specific address in PADDR. For Read Operation the clock and enable signals need to be High but PWRITE signal remains Low while PADDR shows the address corresponding to output. Write transactions correctly transfer data from the master to the slave during the SETUP and ENABLE phases. Read transactions successfully retrieve the stored data from the slave device. The PREADY signal properly controls wait states when introduced. PSLVERR is asserted correctly for invalid or error conditions (if included in design).

2. Waveform Analysis

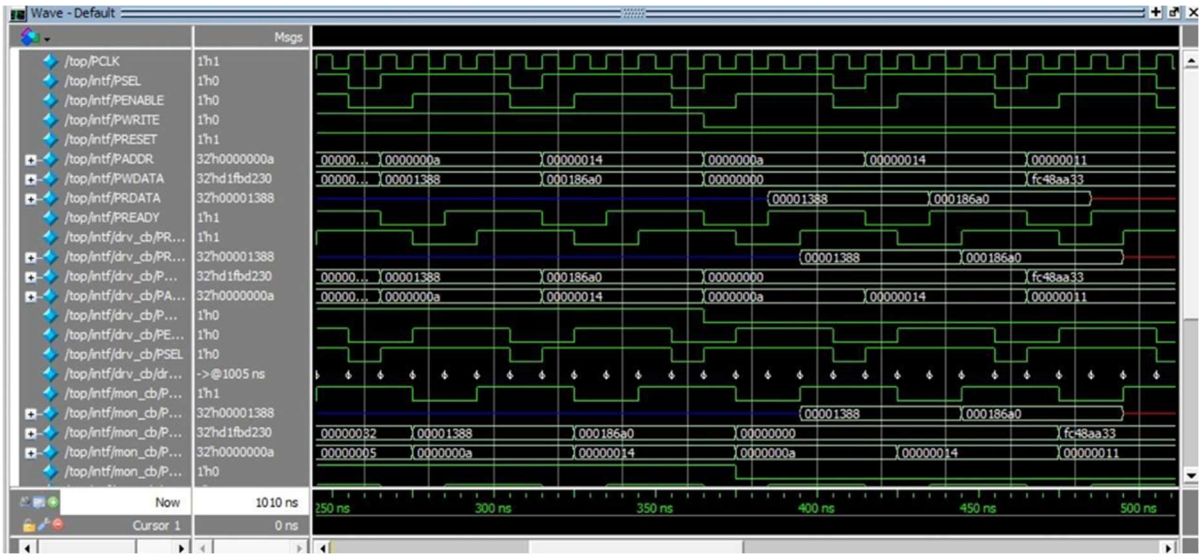
The waveform analysis shows correct synchronization between control and data signals. The PSEL, PENABLE, and PWRITE signals followed proper APB state transitions. In addition, address and data signals remained stable during the ENABLE phase. No race conditions or timing violations were observed as well.

3. Coverage Results

Functional coverage metrics indicate that most protocol scenarios have been exercised. Such as Read/Write operations exhibit 87% coverage. Next, address range testing metric has covered multiple valid addresses. Simultaneously, wait-state conditions are verified through a delayed PREADY signal. All metrics ensure that under normal conditions, the verification environment sufficiently tests the APB protocol behavior.

4. Discussion

The developed SystemVerilog testbench effectively validates the APB slave functionality. The results confirm that the design adheres to the AMBA APB protocol specifications. The use of constrained random stimulus improved verification depth, ensuring better coverage of corner cases. Overall, the verification environment demonstrates high reliability and correctness of APB transactions.



ISSN: 3107-6513

Fig 3: Simulated Functional Verification Result

The waveform shown above represents the functional verification of an APB interface implemented using SystemVerilog. It can be interpreted from the waveform that APB read and write transactions are correctly executed, using the setup and enable phases of the standard two-phase protocol. During the setup phase, the PSEL signal is asserted high, and the master drives the address (PADDR) and control signal (PWRITE). The PENABLE signal remains low, indicating no data transfer occurs at this stage. In the subsequent enable phase, the PENABLE signal is asserted high, enabling the actual data transfer between the master and slave. It is evident from the waveform that successful write operations are carried out when data is driven on PWDATA, and read operations when valid data is returned on PRDATA. Additionally, the PREADY signal validates the appropriate handshake behaviour through controlling the wait states. A controlled wait state ensures that data transfer completes only when the slave is ready. Concisely, the resulting waveform demonstrates proper implementation of APB protocol with perfect timing synchronization, stable signal transitions, and accurate read/write functionality without any protocol violations.

Coverage Summary by Structure:

Design Scope	Hits %	Coverage %
top	64.79%	76.05%
intf	62.44%	80.19%
dut	65.56%	76.19%
top_sv_unit	90.05%	96.17%
apb_trans/display	0.00%	0.00%
generator/new	100.00%	100.00%
generator/run	100.00%	100.00%
driver/new	100.00%	100.00%
driver/run	100.00%	100.00%
apb_coverage	81.81%	93.52%
monitor/new	100.00%	100.00%
monitor/run	100.00%	100.00%
agent/new	100.00%	100.00%
agent/run	100.00%	100.00%
scoreboard/new	100.00%	100.00%
scoreboard/run	100.00%	100.00%
environment/new	100.00%	100.00%
environment/run	100.00%	100.00%
base_test/new	100.00%	100.00%
base_test/run	100.00%	100.00%

Coverage Summary by Type:

Total Coverage:					71.60%	84.99%
Coverage Type	**Bins**	**Hits**	**Misses**	**Weight**	**% Hit**	**Coverage**
Covergroups	48	38	10	1	79.16%	87.05%
Statements	144	137	7	1	95.13%	95.13%
Branches	20	19	1	1	95.00%	95.00%
FEC Conditions	7	5	2	1	71.42%	71.42%
Toggles	414	254	160	1	61.35%	61.35%
Assertions	1	1	0	1	100.00%	100.00%

Fig 4: Overall Coverage Report

The functional coverage results indicate a high level of verification completeness for the design under test. The overall coverage achieved is 84.99%, with a total hit percentage of 71.60%, suggesting that most of the defined verification scenarios have been exercised successfully. At the structural level, the design scope shows strong verification depth in most modules such as intf (80.19%), dut (76.19%), and top_sv_unit (96.17%), indicating that the core design components were well stimulated during simulation.

From a coverage type perspective, assertions (100%) and branches (95%) show excellent verification quality, ensuring that key design rules and decision paths are fully exercised. Similarly, statements coverage (95.13%) and cover groups (87.05%) indicate strong functional exploration of the design behaviour. However, toggle coverage (61.35%) and FEC conditions (71.42%) are comparatively lower, suggesting incomplete signal activity and insufficient condition variation in certain logic paths. Overall, while the verification environment demonstrates strong functional and structural coverage, targeted improvements in stimulus generation—particularly for toggling

ISSN: 3107-6513

activity and uncovered modules—would further enhance the robustness and completeness of the verification process.

top_sv_unit	96.17	87.05	93.80	100.00	100.00	100.00
apb_trans/display	0.00	--	0.00	--	--	--
generator/new	100.00	--	100.00	--	--	--
generator/run	100.00	--	100.00	--	--	100.00
driver/new	100.00	--	100.00	--	--	--
driver/run	100.00	--	100.00	--	--	--
apb_coverage	93.52	87.05	100.00	--	--	--
monitor/new	100.00	--	100.00	--	--	--
monitor/run	100.00	--	100.00	100.00	100.00	--
agent/new	100.00	--	100.00	--	--	--
agent/run	100.00	--	100.00	--	--	--
scoreboard/new	100.00	--	100.00	--	--	--
scoreboard/run	100.00	--	100.00	100.00	100.00	--
environment/new	100.00	--	100.00	--	--	--

Fig 5: Coverage Summary

The results demonstrate that the APB verification environment is highly effective, with most components achieving full coverage and only minor gaps remaining in specific APB functional scenarios. In order to close the remaining coverage gap, further enhancement of the directed test cases and improvement of the random testing strategies are recommended. Here, the findings of the functional coverage of the APB verification environment reveal a substantial level of verification completeness across all key components. An overall coverage of approximately 96.17% is attained by the top-level SystemVerilog unit, along with several simulation runs reaching entire 100% coverage, demonstrating strong test effectiveness.

V. CONCLUSION

In this paper, a well-organized and reusable verification environment is successfully implemented to execute APB verification applying SystemVerilog. This work validates the APB protocol functionality by establishing a comprehensive testbench containing stimulus generation, driver, monitor, scoreboard, and coverage components. A number of test scenarios are conducted to verify the proper functionality of read, write, and reset operations under various conditions.

It is interpreted from the simulated outcomes that the Design Under Test (DUT) performs as per the APB specification, along with correct data transfer between master and slave interfaces. Further, the functional coverage analysis corroborates a substantial level of verification completeness by exercising the majority of the critical cases. In addition, the effective detection of protocol violations through assertions and checks improves design robustness. Moreover, SystemVerilog features such as interfaces, classes, and constraints facilitate better modularity, reusability, and scalability in a verification environment. This project work also elucidated practical insight into industry-standard verification techniques used in modern digital design flows. Overall, this project work successfully demonstrates a reliable and efficient verification strategy for the APB protocol, and it has the potential to be extended for more complex bus architectures and SoC-level verification tasks in future designs.

REFERENCES

[1] Shankar, D. Girdhar, N. K. Shukla. "Design and verification of AMBA APB protocol." *International Journal of Computer Applications* vol. 95, no. 21, pp. 29-35, 2014.

[2] K. B. Raju, and B. K. Konda, "Design and verification of AMBA APB protocol."

ISSN: **3107-6513**

International Journal of Engineering Research and Application vol. 7, no. 1, pp. 87-90, 2017.

[3] N. Sahu, A. Kumar and R. Kandari, "Design and Verification of APB IP Core using Different Verification Methodologies," *2022 International Conference on Breakthrough in Heuristics And Reciprocation of Advanced Technologies (BHARAT)*, Visakhapatnam, India, 2022, pp. 150-154.

[4] D. P. Kumar and K. S. Pande, "A Study on Verification of APB Protocol," *2024 5th International Conference on Smart Electronics and Communication (ICOSEC)*, Trichy, India, 2024, pp. 530-534.

[5] V. S. Reddy, J. Sachin Rewaria, R. Kumar Thota, C. Anil Kumar and V. Kumar Chikoti, "Design and Verification of APB Protocol," *2025 3rd International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA)*, Coimbatore, India, 2025, pp. 1-4.

[6] A. Tibeli, S. V. Siddamal, and S. V. Budihal, "Design and Verification of AHB to APB Bridge," 2025 *International Conference on ICT for Sustainable Development*, pp. 298-308.

[7] J Mukunthan, A Daniel Raj, S Keerthivadana, R Kiruthika, T Shruthi and S Snegasri, "Design and implementation of amba apb protocol." 2021 *IOP Conference Series: Materials Science and Engineering*, vol. 1084 012050.

[8] I. H. Shanavas, P. A. Reddy, M. M. Baburaj, N. C. Krishna and N. S, "Design and Analysis of APB and AHB Lite Protocol," *2023 7th International Conference on Design Innovation for 3 Cs Compute Communicate Control (ICDI3C)*, Karnataka, India, 2023, pp. 1-6.

[9] H. G. Rayasa, H. P. N and Y. J. M. Shirur, "Synthesis of AMBA APB BUS Protocol," *2025 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE)*, Bangalore, India, 2025, pp. 1-5.

[10] N. Swathi, J. S. Prasad, U. P. Lakshmi, E. P. Kalyan, K. Shashikanth, "Design of AMBA APB Protocol using Verilog." GRENZE International Journal of Engineering & Technology (GIJET), 2024.