International Journal of Multidisciplinary Engineering in Current Research - IJMEC

Volume 11, Issue 7, July-2026, http://ijmec.com/, ISSN: 2456-4265

RTL Implementation Of Ambaapb Protocol For Peripheral Communication

Ms. Mariam[1], Bhanavath Kavya[2], Domala Meghana[3], Chindam Akshaya[4]

[1]Assistant Professor: Department of Electronics and Communication Engineering, Bhoj Reddy Engineering College for Women Hyderabad, India

[2,3,4]B. Tech Students: Department of Electronics and Communication Engineering, Bhoj Reddy Engineering College for Women Hyderabad, India

Mail Id's: bhanavathkavya8@gmail.com[2], meghanadomala28@gmail.com[3], chindamakshaya0@gmail.com[4]

Accepted 4 July 2026

Abstract

The Advanced Microcontroller Bus Architecture (AMBA) Advanced Peripheral Bus (APB), developed by Arm, is a simple, low-power, and low-complexity bus protocol designed for efficient communication between processors and low-bandwidth peripheral devices. This project presents the Register Transfer Level (RTL) implementation of the APB protocol to enable reliable data transfer between a master and multiple peripheral slave devices. The proposed design consists of APB master, slave, and decoder modules that support both read and write operations using a two-phase transfer mechanism comprising the setup and enable phases. The RTL architecture is implemented using Verilog Hardware Description Language (HDL) to achieve synchronous operation, low power consumption, and minimal hardware complexity. Functional verification is performed through simulation to validate address decoding, control signal generation, and data transfer operations. The simulation results confirm the correct functionality of the APB protocol, ensuring efficient communication between the master and peripheral devices with simplified timing requirements. The implemented system provides a scalable, reliable, and cost-effective solution for integrating low-speed peripherals in embedded systems and System-on-Chip (SoC) applications.

Keywords: *AMBA APB, Register Transfer Level (RTL), Verilog HDL, Bus Protocol, APB Master, APB Slave, Address Decoder, Read/Write Operations, Embedded Systems, System-on-Chip (SoC), Functional Verification.*

Introduction

The Advanced Microcontroller Bus Architecture (AMBA) is a widely adopted on-chip communication standard developed by Arm for connecting various components within a System-on-Chip (SoC). AMBA defines a family of bus protocols that facilitate efficient communication between processors, memories, and peripheral devices. Among these protocols, the Advanced Peripheral Bus (APB) is specifically designed for low-bandwidth peripherals such as Universal Asynchronous Receiver Transmitters (UARTs), General-Purpose Input/Output (GPIO) modules, timers, and other simple devices that do not require high-speed data transfer. Owing to its simple architecture, low power consumption, and minimal hardware complexity, the APB protocol has become a preferred interface in many embedded systems and SoC designs.

The APB protocol operates using a straightforward two-phase transfer mechanism consisting of a setup phase and an enable phase, ensuring reliable communication between the bus master and peripheral slave devices. Unlike high-performance buses such as AHB and AXI, APB does not support pipelined or burst data transfers, thereby reducing implementation complexity and power consumption. This simplified communication mechanism makes APB particularly suitable for peripheral devices where high throughput is not a primary requirement.

Register Transfer Level (RTL) design is a fundamental abstraction in digital system development that describes the flow of data between hardware registers and the logical operations performed on that data. RTL designs are typically implemented using Hardware Description Languages (HDLs) such as Verilog HDL or VHDL, enabling designers to model, simulate, verify, and synthesize digital circuits before hardware fabrication. RTL implementation plays a crucial role in ensuring functional correctness, timing reliability, and efficient hardware utilization.

In this project, the RTL implementation of the AMBA APB protocol is developed using Verilog HDL. The design includes the APB master, APB slave, and address decoder modules, which collectively support read and write operations between the processor and multiple peripheral devices. Functional verification is carried out through simulation to validate address decoding, control signal generation, timing behavior, and data transfer operations. The proposed architecture demonstrates reliable communication

with reduced hardware overhead while maintaining synchronous operation and low power consumption.
The implemented APB protocol provides a scalable and efficient communication interface for integrating low-speed peripherals into modern embedded systems and System-on-Chip (SoC) applications. Its simple architecture, reliable operation, and ease of implementation make it well suited for a wide range of digital designs requiring cost-effective and energy-efficient peripheral communication.

Literature Survey

The Advanced Microcontroller Bus Architecture (AMBA), introduced by Arm, has become the industry-standard on-chip communication architecture for System-on-Chip (SoC) designs. It provides standardized communication between processors, memories, and peripheral devices through protocols such as the Advanced High-performance Bus (AHB), Advanced Peripheral Bus (APB), and Advanced eXtensible Interface (AXI). Among these, the APB protocol is specifically designed for low-bandwidth peripherals, offering a simple architecture, low power consumption, and reduced hardware complexity. These characteristics make APB an ideal choice for connecting devices such as timers, UARTs, GPIO controllers, watchdog timers, and configuration registers.
Several researchers have investigated the design and implementation of the APB protocol at the Register Transfer Level (RTL) using Verilog HDL. RTL implementation enables designers to model the communication between master and slave modules, simulate protocol behavior, and verify functional correctness before hardware realization. The APB protocol employs a two-phase transfer mechanism consisting of Setup and Enable phases, which simplifies timing requirements while ensuring reliable read and write operations. Compared with high-performance buses such as AHB and AXI, APB requires fewer control signals and consumes less power, making it well suited for low-speed peripheral communication in embedded systems.
Recent research has also focused on the functional verification of APB designs using simulation tools such as Xilinx Vivado, ModelSim, and other HDL simulators. These studies validate the correct operation of important control signals, including PSEL, PENABLE, PWRITE, PREADY, and PSLVERR, to ensure proper synchronization between the master and slave devices. Simulation results demonstrate that APB provides reliable peripheral communication, efficient hardware resource utilization, and simplified timing analysis. Although APB offers several advantages for low-speed communication, its limited bandwidth and lack of pipelined or burst transfer support make it unsuitable for high-performance applications, where protocols such as AHB or AXI are preferred.

Table 1 Summary of Reviewed Literature

Sl. No.	Technology/Study	Description	Advantages	Limitations
1	AMBA Bus Architecture	Standard communication architecture for SoC systems	Modular, scalable, and standardized	Requires multiple bus protocols
2	Advanced Peripheral Bus (APB)	Low-speed peripheral communication protocol	Simple architecture and low power consumption	Limited bandwidth
3	RTL Implementation of APB	Verilog HDL implementation of APB protocol	Easy verification and efficient hardware utilization	Requires simulation before implementation
4	APB Functional Verification	Simulation using Vivado/ModelSim	Verifies protocol timing and communication	Limited to simulation environment
5	Recent APB Research (2023–2025)	APB optimization for embedded SoC applications	Reliable peripheral communication	Lower performance than AHB/AXI

Motivation

Modern **System-on-Chip (SoC)** designs integrate processors, memory modules, and peripheral devices onto a single chip, making efficient on-chip communication essential for reliable system operation. As the number of integrated peripherals continues to increase, there is a growing need for communication protocols that are simple, energy-efficient, and easy to implement. While high-performance bus protocols are suitable for processor and memory communication, they introduce unnecessary complexity for low-speed peripheral devices.
The **Advanced Peripheral Bus (APB)** addresses this challenge by providing a lightweight communication

protocol with minimal control logic and low power consumption. Its simple architecture enables efficient communication between the processor and peripheral devices while reducing hardware overhead. These advantages make APB one of the most widely used bus protocols in embedded systems and VLSI applications.

The primary motivation for this project is to understand the architecture and operation of the AMBA APB protocol and to implement it at the **Register Transfer Level (RTL)** using **Verilog HDL**. The project aims to design APB master, slave, and decoder modules capable of performing reliable read and write operations. Functional verification is carried out using **Xilinx Vivado** simulation to validate protocol behavior, timing, and control signal generation. Through this implementation, the project provides practical experience in digital hardware design, bus protocol implementation, RTL modeling, and functional verification, which are fundamental concepts in modern embedded system and VLSI design.

Software Requirement

Xilinx Vivado

Xilinx Vivado Design Suite is a comprehensive Electronic Design Automation (EDA) tool developed by Xilinx (AMD) for designing, implementing, and verifying digital systems on FPGA devices. It is widely used in VLSI, embedded systems, and digital hardware design for developing applications using Verilog HDL and VHDL. Vivado provides an integrated platform that supports Register Transfer Level (RTL) design, synthesis, functional simulation, implementation, timing analysis, and debugging.

In this project, Vivado is used to develop the RTL implementation of the AMBA APB protocol, including the APB master, slave, and decoder modules. The built-in simulator is used to verify read and write operations, address decoding, control signal generation, and communication between the master and peripheral devices. The waveform viewer allows detailed analysis of important APB signals such as PCLK, PRESETn, PSEL, PENABLE, PWRITE, PADDR, PWDATA, PRDATA, PREADY, and PSLVERR, ensuring that the protocol operates according to the AMBA APB specification.

Vivado also provides synthesis and implementation reports that help evaluate hardware resource utilization, timing performance, and design correctness before deployment on FPGA hardware. Its user-friendly interface, integrated debugging tools, and efficient simulation environment make it an ideal platform for developing and validating RTL-based communication protocols. Consequently, Xilinx Vivado plays a crucial role in ensuring the functional correctness, reliability, and performance of the proposed AMBA APB implementation.

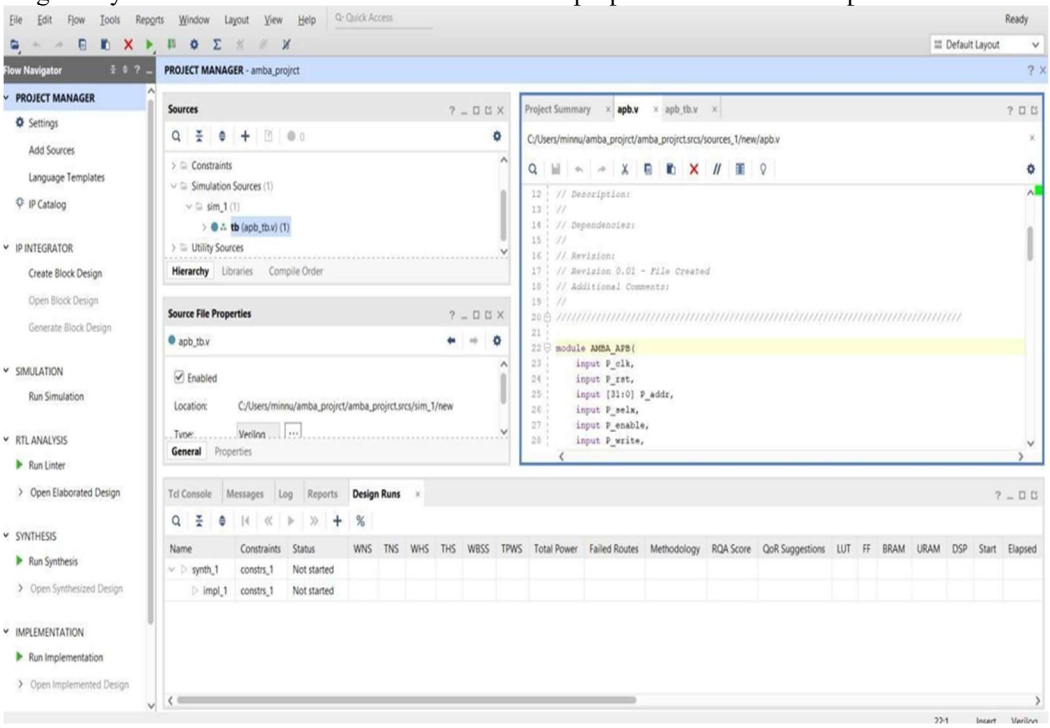

Figure 1: Xilinx Vivado Design Suite.

RTL Implementation of AMBA APB Protocol for Peripheral Communication

The Register Transfer Level (RTL) implementation of the Advanced Microcontroller Bus Architecture (AMBA) Advanced Peripheral Bus (APB) protocol is an essential aspect of modern System-on-Chip (SoC) design. The APB protocol is a simple, low-power, and low-complexity communication bus used to interface low-speed peripheral devices such as timers, Universal Asynchronous Receiver Transmitters (UARTs), General-Purpose Input/Output (GPIO) modules, and control registers with the processor. Developed as part of the AMBA specification by Arm, the APB protocol provides an efficient communication mechanism between the processor and peripheral devices while minimizing hardware complexity and power consumption.

The APB protocol follows a master–slave communication architecture in which the master initiates read and write transactions and the selected slave device responds accordingly. Communication is performed through address, data, and control signals synchronized by a common clock. In this project, the APB protocol is implemented at the Register Transfer Level (RTL) using Verilog HDL and functionally verified using Xilinx Vivado Design Suite. The implementation demonstrates reliable communication between the APB master and multiple peripheral devices while validating the protocol through simulation waveforms. This project provides practical understanding of RTL design, bus communication protocols, and on-chip peripheral interfacing in embedded and VLSI systems.

Block Diagram

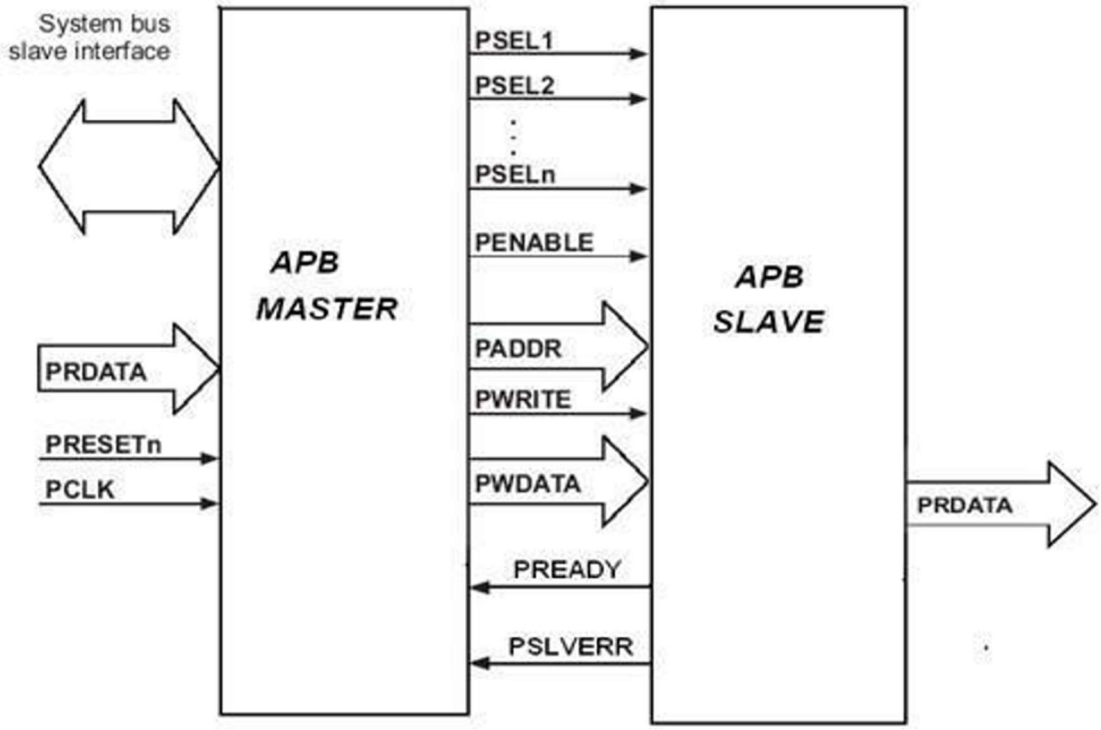

Figure 2; Block Diagram

System Bus Slave Interface

The System Bus Slave Interface acts as the communication bridge between the processor or system bus and the APB master. It receives address, control, and data information from the processor and forwards these signals to the APB master, enabling communication with the selected peripheral devices.

APB Master

The APB master controls all communication on the APB bus. It generates the required control signals, including slave selection, read/write control, and address information. The master initiates every transaction and manages the communication sequence between the processor and the selected peripheral slave.

Clock Signal (PCLK)

The **PCLK** signal provides the synchronization clock for all APB operations. Every state transition and data transfer within the protocol occurs on the active edge of this clock, ensuring synchronous communication throughout the system.

Reset Signal (PRESETn)

The **PRESETn** signal initializes the APB system. When activated, all registers and control signals return to their default values, ensuring proper system startup and preventing undefined operation during initialization.

Slave Select Signals (PSEL1, PSEL2 … PSELn)

The slave select signals determine which peripheral device participates in the current transaction. Only the selected slave receives the address and control information and responds to the master's request.

Address Bus (PADDR)

The **PADDR** bus carries the address generated by the master. The selected slave decodes this address to identify the register or memory location involved in the requested read or write operation.

Working Methodology

The AMBA APB protocol operates using a simple two-phase communication mechanism consisting of the Setup Phase and the Access Phase. Communication begins after the system is initialized using the PCLK and PRESETn signals. The reset signal places all registers and control logic into a known state, after which the APB master becomes ready to initiate communication with the peripheral devices.

The communication process starts when the APB master generates the required control signals, including PSEL, PADDR, and PWRITE. These signals identify the target peripheral device and specify whether the requested operation is a read or write transaction. During the Setup Phase, the selected slave is activated through the PSEL signal while the address and control information are placed on the bus. At this stage, the PENABLE signal remains low, indicating that the transaction is being prepared but data transfer has not yet started.

In the Access Phase, the master asserts the PENABLE signal to initiate the actual data transfer. During this phase, the selected slave processes the received request. If the PWRITE signal is high, the master performs a write operation by transferring data through the PWDATA bus to the addressed register within the slave. Conversely, if the PWRITE signal is low, the slave retrieves the requested data from the specified address and returns it to the master through the PRDATA bus.

After completing the requested operation, the slave asserts the PREADY signal to indicate successful completion of the transaction. If any communication error occurs, the PSLVERR signal is asserted to notify the master of the failure. This simple communication mechanism ensures reliable synchronization between the master and slave while maintaining low hardware complexity and power consumption.

The complete RTL design is implemented using Verilog HDL and verified through simulation using Xilinx Vivado Design Suite. Functional verification is performed by analyzing the waveform outputs of key APB signals, including PSEL, PENABLE, PWRITE, PADDR, PWDATA, PRDATA, PREADY, and PSLVERR. The simulation results confirm the correct execution of the setup phase, access phase, read operations, and write operations, thereby validating the functionality and correctness of the proposed AMBA APB protocol implementation.

Results and Discussion

Results

The RTL implementation of the AMBA Advanced Peripheral Bus (APB) protocol was successfully developed using Verilog HDL and functionally verified using Xilinx Vivado Design Suite. The simulation results confirmed correct communication between the APB master and slave modules. Both read and write operations were executed successfully, and the waveform analysis verified the correct operation of important APB signals, including PSEL, PENABLE, PWRITE, PADDR, PWDATA, PRDATA, PREADY, and PSLVERR. The implemented design demonstrated reliable and synchronized data transfer between the master and peripheral devices while complying with the APB communication protocol.

The major outcomes of the implementation are summarized as follows:

1. **Successful RTL Implementation:** The APB protocol was successfully designed and implemented at the Register Transfer Level (RTL) using Verilog HDL.
2. **Successful Functional Verification:** The complete design was simulated using Xilinx Vivado, and waveform analysis verified the correctness of protocol timing, control signals, and data transactions.
3. **Reliable Peripheral Communication:** The implemented APB protocol achieved efficient, reliable, and low-complexity communication between the processor and peripheral devices.

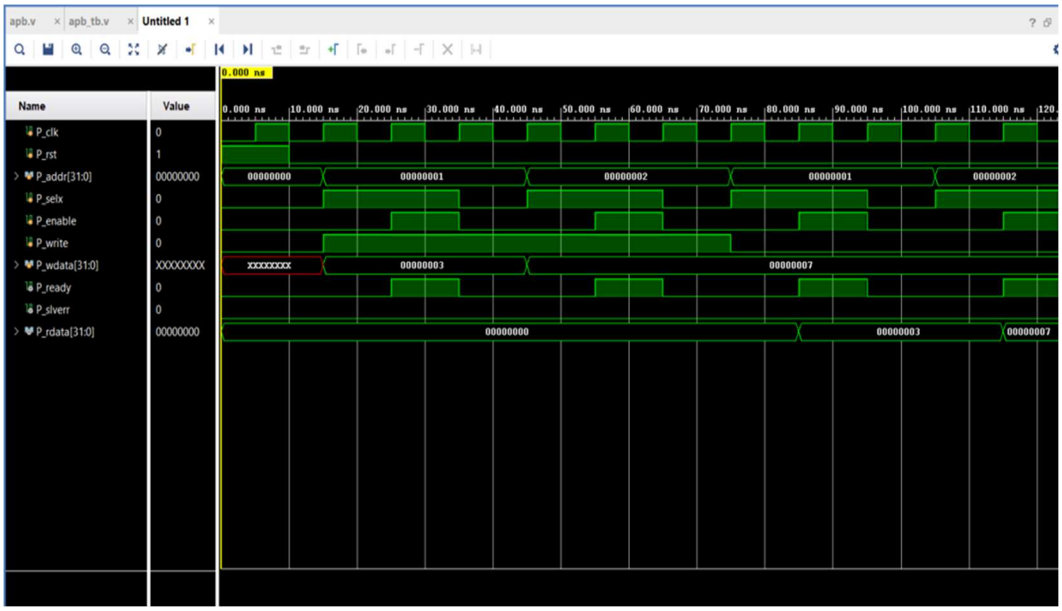

Figure 3: Simulation Result of RTL Implementation of AMBA APB Protocol

Figure 3 presents the simulation waveform of the implemented AMBA APB protocol. The PCLK signal provides synchronization for all bus transactions, while PRESETn initializes the system at the beginning of the simulation. The PADDR signal carries the address generated by the master to access the required slave register. The PSEL signal selects the target slave device, and PENABLE indicates the access phase of the transaction. The PWRITE signal determines whether the current transaction is a read or write operation.

During write operations, data is transferred from the master to the slave through the PWDATA bus, whereas during read operations, the slave returns the requested data through the PRDATA bus. The PREADY signal indicates successful completion of the transaction, while PSLVERR remains inactive, confirming error-free communication. The waveform clearly demonstrates successful execution of both read and write operations, with data values being correctly transferred between the master and slave modules. These results validate the correctness and reliability of the proposed RTL implementation of the AMBA APB protocol.

Discussion

The proposed project successfully demonstrates the Register Transfer Level (RTL) implementation of the Advanced Peripheral Bus (APB) protocol for communication between a processor and low-speed peripheral devices. The APB protocol, which forms part of the AMBA architecture, provides a simple and low-power communication interface suitable for embedded and System-on-Chip (SoC) applications. The APB master and slave modules were designed using Verilog HDL, with the master generating the necessary control signals such as PSEL, PENABLE, PWRITE, and PADDR, while the slave executed the corresponding read and write operations.

The implemented protocol follows the standard APB communication sequence consisting of Setup and Access phases, ensuring synchronized data transfer between the master and slave devices. Functional verification using Xilinx Vivado confirmed correct timing behavior, address decoding, and successful execution of data transactions. The simulation waveforms demonstrated that all control and data signals operated according to the APB specification without communication errors. The implementation highlights the advantages of the APB protocol, including simple architecture, reduced hardware complexity, low power consumption, and reliable peripheral communication, making it suitable for integration into modern embedded and VLSI systems.

Conclusion and Future Scope

Conclusion

This project successfully implemented the AMBA Advanced Peripheral Bus (APB) protocol for peripheral communication using Verilog HDL at the Register Transfer Level (RTL). The developed design demonstrates reliable communication between the APB master and slave modules through standardized address, control, and data signals. Functional verification using Xilinx Vivado Design Suite confirmed the correct execution of read and write

operations, proper address decoding, and accurate timing of APB control signals.

The implemented APB protocol achieves efficient communication with low-speed peripheral devices while maintaining a simple architecture, low hardware complexity, and reduced power consumption. The successful simulation results validate the correctness of the design and provide a practical understanding of AMBA-based communication protocols used in modern embedded systems and System-on-Chip (SoC) architectures. Overall, the project demonstrates that the APB protocol is an effective and reliable solution for integrating peripheral devices into digital systems.

Future Scope

The proposed RTL implementation of the AMBA APB protocol can be extended in several directions to improve its functionality and applicability in advanced SoC designs. Future enhancements include the integration of the APB protocol with high-performance AMBA buses such as AHB and AXI to develop complete multi-bus communication architectures. The implementation of an AHB-to-APB or AXI-to-APB bridge would enable seamless communication between high-speed processors and low-speed peripheral devices.

The current design can also be expanded to support multiple slave devices using advanced address decoding techniques, allowing communication with a larger number of peripherals. Additional low-power optimization methods, including clock gating and power gating, may be incorporated to further reduce energy consumption in battery-powered embedded systems.

Future work may also include advanced verification using SystemVerilog, Universal Verification Methodology (UVM), and formal verification techniques to improve design robustness and functional coverage. Implementing the design on FPGA hardware would enable real-time validation beyond simulation and provide practical performance evaluation. Furthermore, enhanced error detection and fault-handling mechanisms can be incorporated to improve the functionality of the PSLVERR signal. The APB protocol can also be integrated into modern embedded applications such as IoT devices, microcontrollers, and complex System-on-Chip (SoC) platforms, making the design more scalable and suitable for next-generation digital systems.

References

[1] S. K. Yadav, P. Satyanarayana, and B. V. Krishna, "Design and Implementation of AMBA-APB Protocol for System-on-Chip Designs," in *Proc. IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE)*, 2024.

[2] P. Maheshwari, "Design and Synthesis of AMBA APB Protocol for System-on-Chip Applications," *Journal of Communication and Network Engineering*, 2023.

[3] A. Sharma, R. Verma, and S. Gupta, "RTL Design of APB Protocol Based Memory Module," *Engineering Research Journal*, vol. 18, no. 2, pp. 45–52, 2025.

[4] S. R. Savarimuthu, K. Ramesh, and M. Prakash, "UVM-Based Design and Verification of AMBA APB Protocol," in *Proc. IEEE Conference on Computing and Communication Systems*, 2025.

[5] J. Mukunthan, R. Kumar, and P. Karthikeyan, "Design and Implementation of AMBA APB Protocol," *IOP Conference Series: Materials Science and Engineering*, vol. 1084, 2021.

[6] D. P. Kumar and K. S. Pande, "A Study on Verification of APB Protocol," in *Proc. 5th International Conference on Smart Electronics and Communication (ICOSEC)*, IEEE, 2024.

[7] S. K. Yadav, P. Satyanarayana, and B. V. Krishna, "Design and Implementation of AMBA-APB Protocol for System-on-Chip Designs," in *Proc. IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE)*, 2024.

[8] Arm Ltd., **AMBA APB Protocol Specification**, AMBA 5 APB Protocol Specification, Cambridge, U.K., 2019.

[9] C. Saxena, S. Prakash, and D. Srivastava, "APB Protocol of AMBA Technology," *International Journal of Creative Research Thoughts (IJCRT)*, vol. 13, no. 5, pp. 120–126, 2025.

[10] P. Uma Maheshwari, J. Siddhu Nayak, S. Mandava, and M. Shoukath Ali, "Design and Implementation of 32-bit AMBA-APB Protocol Using Cadence," *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, vol. 13, no. 4, pp. 1450–1458, 2025.