IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

APB PROTOCOL OF AMBA TECHNOLOGY

[1] Chirag Saxena, [2] Shivansh Prakash, [3] Deepmala Srivastava
[1]Student, [2]Student, [3] Assistant Professor,
[1] Electronics and communication engineering,
[1]Babu Banarasi Das Northern India Institute of Technology(056)
Lucknow, India

Abstract: The Advanced Peripheral Bus (APB) is a crucial component of AMBA (Advanced Microcontroller Bus Architecture) developed by ARM, designed for low-power, low-bandwidth communication with peripherals. This paper analyzes the architecture, signal interface, and data transfer mechanism of APB, focusing on its simplicity, synchronous operation, and ease of integration in System-on-Chip (SoC) designs. Unlike AHB and AXI, APB is tailored for energy-efficient communication with slower devices like UARTs, timers, and GPIOs. The research highlights APB's role in scalable, reliable peripheral connections and its impact on VLSI design, contributing to optimized embedded system performance.

***Index Terms* - Low-Power, Low-Bandwidth Design, Simplicity and Ease of Integration, Peripheral Connectivity, Part of AMBA Family, Synchronous Data Transfer, Support for Scalable SoC Design**

I. INTRODUCTION

In modern System-on-Chip (SoC) architectures, efficient communication between hardware components is critical for optimizing performance, scalability, and power efficiency. The Advanced Peripheral Bus (APB) protocol, an integral part of the ARM Advanced Microcontroller Bus Architecture (AMBA), enables the seamless integration of low-speed peripheral devices with high-speed processing cores. Designed for simplicity, APB is a low-cost, synchronous, non-pipelined interface that minimizes power consumption while maintaining reliable data transfer.

APB is primarily used for accessing programmable control registers of peripheral devices, such as UART, SPI, I2C, and GPIO, within an SoC. An APB bridge acts as an intermediary, enabling communication between high-speed bus architectures, such as AXI, and lower-speed APB peripherals. Within this framework, the bridge functions as a Requester, initiating transactions, while peripheral interfaces serve as Completers, responding to these requests.

The AMBA protocol suite, developed by ARM, provides a standardized on-chip communication framework, making sure of compatibility and reusability across diverse SoC implementations. As VLSI technology advances, integrating multiple functional blocks—including

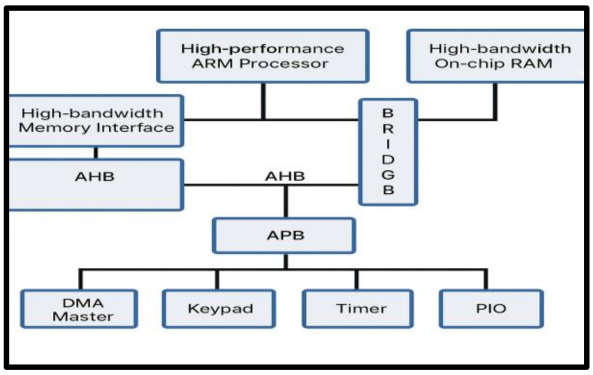

High-performance ARM Processor
High-bandwidth On-chip RAM
High-bandwidth Memory Interface
B R I D G B
AHB
AHB
APB
DMA Master
Keypad
Timer
PIO

fig.1- AMBA block diagram

microcontrollers, processors, memory bridges, and hardware accelerators—into a single chip has become essential for high-performance, cost-effective system design. AMBA's open-standard nature, extensive third-party support, and widespread industry adoption make it a preferred choice for semiconductor applications.

By streamlining peripheral interactions and reducing interface complexity, the APB protocol enhances the overall efficiency and reliability of SoCs. This paper provides an in-depth analysis of the APB protocol's structure, operational mechanisms, and its role within AMBA-based designs, highlighting its significance in modern embedded systems.

II. LITERATURE SURVEY

The literature review examines various research contributions related to the design, implementation, and verification of the AMBA Advanced Peripheral Bus (APB) protocol. These studies focus on improving integration efficiency, reducing power consumption, and making sure of functional correctness in System-on-Chip (SoC) environments.

Ma et al. [1] introduce an AXI4-Lite to APB bridge designed according to AMBA 4.0 specifications, addressing the challenge of integrating heterogeneous Intellectual Property (IP) cores with the AMBA bus system. The study emphasizes seamless communication between high-performance AXI domains and low-power APB peripherals, detailing key aspects such as the handshake mechanism, clock domain crossing techniques, and finite state machine (FSM) implementation. The proposed bridge is validated through simulation and synthesis results, demonstrating its efficiency in SoC applications.

Roopa et al. [2] propose an APB controller for low-bandwidth peripherals in high-performance SoC architectures. The study emphasizes power-efficient communication, explaining the APB state machine, which consists of IDLE, SETUP, and ENABLE states. The authors implement an APB-based system integrating peripherals such as UART, timers, and keypad controllers, optimizing transaction handling between master and slave devices. Experimental results validate the efficiency of the proposed design in reducing power consumption during peripheral communication.

Akhila et al. [3] investigate the implementation of the AMBA APB protocol using SystemVerilog, incorporating a reusable verification methodology to test various transaction scenarios, including single and multiple read/write operations with and without wait states. The design is verified using QuestaSim, focusing on signal interactions, clock synchronization, and transaction efficiency. The simulation results confirm that the APB implementation meets functional correctness and low-power operation requirements, making it a viable solution for modern embedded applications.

ARM Ltd. [4] provides the official AMBA APB protocol specification, outlining its evolution from APB2 to APB5. The document highlights key enhancements such as wait-state management, error handling mechanisms, transaction protection, and wake-up signaling. Additionally, it specifies the signal interactions, transfer protocols, and operating states necessary to ensure industry-standard compliance. This specification serves as a foundational reference for APB implementation, facilitating seamless integration with other AMBA components in SoC designs.

Ravikumar et al. [5] focus on the design and verification of the AMBA APB protocol using Verilog and Cadence tools. The study highlights the importance of formal verification in making sure of protocol compliance. The authors develop a test environment for simulating APB transactions and validating design specifications through RTL synthesis. The research also addresses clock skew minimization, peripheral integration, and power optimization. Experimental results confirm that the APB protocol efficiently supports low-power embedded applications, making sure of reliable communication within an SoC framework.

The reviewed literature collectively highlights advancements in APB bridge design, power-efficient communication strategies, and verification methodologies. These contributions provide valuable insights into digital circuit design, SoC development, and AMBA-based embedded systems, making sure of continued improvements in microcontroller and processor architectures.

III. APB BLOCK DIAGRAM

A. APB Master

The APB Master is responsible for initiating all transactions on the Advanced Peripheral Bus (APB). It supports fundamental operations, including read, write, and idle states, making sure of efficient communication with peripheral devices. The APB Bridge, which connects the system bus to the APB, plays a crucial role in data conversion and address translation. It ensures that the address remains valid throughout the transaction while generating a peripheral select signal (Select), allowing only one peripheral to be active at a time.

During a write operation, the bridge transfers data from the system bus to the APB, making sure of seamless integration of peripheral registers. Conversely, in a read operation, the bridge gets data from an APB peripheral and forwards it to the system bus. The APB Master Bridge thus serves as the interface between high-speed system components and low-power peripherals, making sure of reliable and efficient data exchange.

B. APB Slave

The APB Slave is a simplified yet highly effective interface that enables communication with multiple peripherals. It operates based on control signals such as select, enable, and write, which dictate the execution of read and write operations.

During a write cycle, when the select and enable signals are turned on high, and the write signal is active, the APB Slave captures the address and data from the bus. This information is then used to update the appropriate register within the peripheral. In a read transaction, the slave places the requested data onto the data bus when the write signal is low, provided that both select and enable signals are high. The paddr (peripheral address) is used to determine the specific register that is being accessed, making sure of precise data retrieval.

C. APB Block Diagram Overview

The Advanced Peripheral Bus (APB) protocol is structured to meet the design specifications necessary for low-power and low-complexity peripheral communication. The block diagram

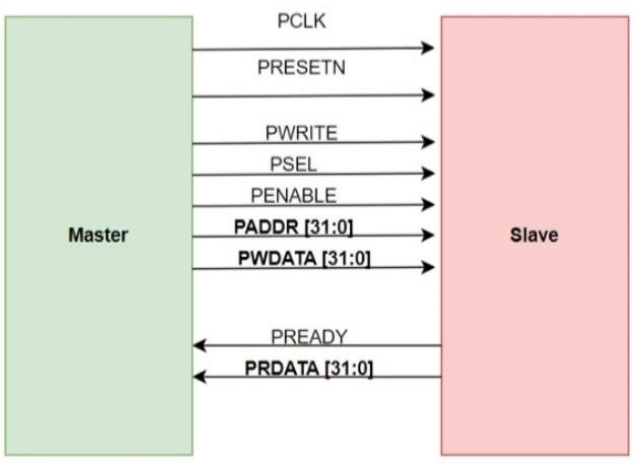


Fig. 2 – APB – Master & Slave flow

representation of APB outlines its fundamental signals and interconnections, making sure of a clear understanding of how data and control signals are managed within the system. The fundamental architecture of the APB, highlighting the key components involved in data transactions between the master and multiple slave peripherals.

IV. APB OPERATING STATUS

The Advanced Peripheral Bus (APB) employs a three-state operational model to streamline data transfer while optimizing power consumption in embedded systems.

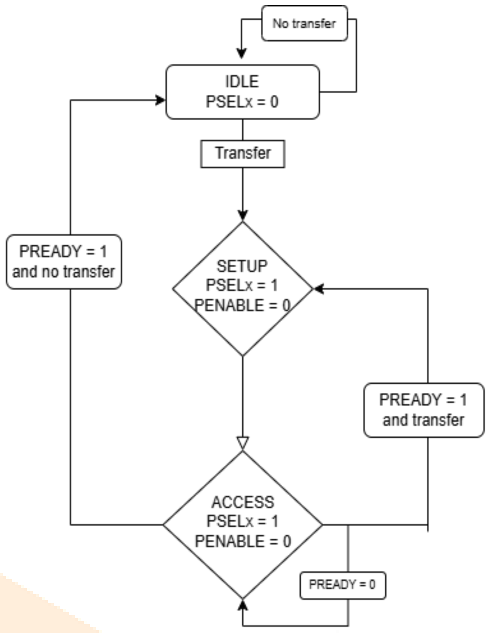
A. State Transition Overview

The APB operates through three primary states:

IDLE State: The default state where no data transfer occurs, allowing the bus to conserve power until a transaction request is initiated.

SETUP State: When a transfer request is received, the bus enters the SETUP state for one clock cycle. During this phase, address selection, control signal assertion, and device selection take place, preparing the system for data transfer.

ENABLE State: Following the SETUP phase, the ENABLE state enables the actual data transfer between the master and slave devices. This phase also lasts one clock cycle. After data transfer completion, the bus either returns to the IDLE state or re-enters the SETUP state if consecutive transactions are required, making



sure of operational efficiency.

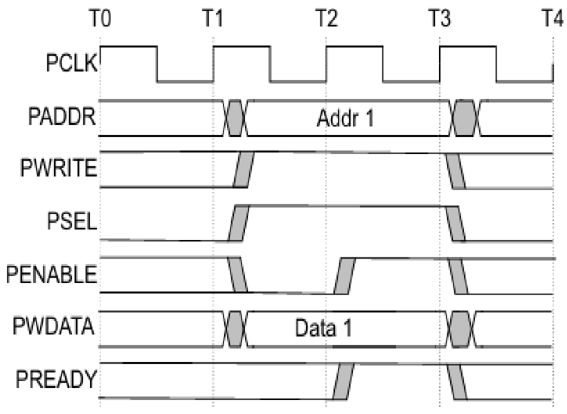
Fig 3. State Diagram of APB Protocol

Minor signal fluctuations may occur during rapid transitions between the SETUP and ENABLE states; however, these remain within the tolerances defined by the APB protocol.

B. APB Write Cycle

An APB write operation starts with the bus in the IDLE state at the T0 clock cycle. When a write transaction is initiated, control signals PSEL, PWRITE, PADDR, and PWDATA are turned on at the T1 clock edge, marking the beginning of the setup phase.

At the T3 clock edge, both PENABLE and PREADY are turned on, indicating transaction completion. Each write operation requires a minimum of two clock cycles. For subsequent transactions, PENABLE de-asserts



while PREADY transitions from high to low, preparing for the next operation.

Fig. 4 – Signal diagram of write state

1) Write Transfer with Wait States

If the slave device needs additional processing time, wait states are introduced. The PENABLE signal remains high until PREADY is turned on. If the slave is not ready, it holds PREADY low, forcing the master to maintain the address and data signals until readiness is achieved. This mechanism extends the initial write transaction to at least three clock cycles, while subsequent transfers (once the slave is ready) are completed in two cycles.

C. APB Read Cycle

A read transaction begins from the IDLE state. When a read request is initiated, PSEL, PWRITE (set to 0), and PADDR are turned on at the T1 clock edge, initiating the setup phase. At the next clock edge (T3), PENABLE and PREADY are turned on, signaling transaction completion. Each read operation requires at least two clock cycles. For continuous transactions, PENABLE de-asserts, transitioning the system back to the SETUP state, while PREADY moves from high to low, preparing for the next transaction.

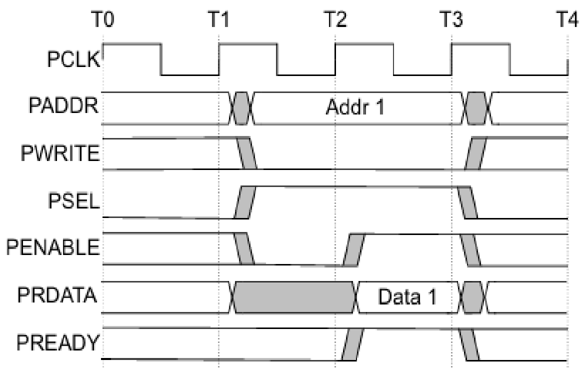


Fig. 5 - Signal diagram of read state

1) Read Transfer with Wait States

If the slave requires extra processing time, PREADY remains low, introducing wait states. The master holds its request signals until PREADY is turned on. Once ready, the master gets data from PRDATA, de-asserts PENABLE, and both devices return to the IDLE state. This process ensures proper synchronization and efficient communication despite variable slave response times.

D. APB Master-Slave Communication Design

The APB master-slave architecture adheres to a synchronized data transfer protocol:

The master initiates communication by asserting PSEL, specifying the address (PADDR), and signaling the transfer through PENABLE.

The PWRITE signal determines whether the transaction is a read (0) or write (1) operation.

The selected slave device asserts PREADY to indicate readiness for transaction execution.

Data transfer occurs on the rising edge of the clock (PCLK) for read operations and on the falling edge for writes.

This structured communication framework ensures low-latency, efficient data exchange between master and slave devices, making APB a preferred choice for low-power embedded applications.

VI. RESULTS AND DISCUSSION

The AMBA APB read and write operations were analyzed through waveform simulations to observe signal interactions and ensure the correctness of data transactions between the master and slave devices. The read operation waveform, shows how key control signals coordinate the data retrieval process. The PCLK signal provides synchronization, while PADDR specifies the target address. The PWRITE signal remains low, confirming a read transaction. The PSELx line activates the appropriate slave, and the transition of PENABLE to high marks the start of the read cycle. The crucial PREADY signal dictates whether data is available immediately or if wait states are required. If the slave is not ready, the PREADY signal remains low, prompting the master to wait until valid data appears on PRDATA upon PREADY assertion.

Similarly, the write operation waveform, captures the data transfer process from the master to the slave. The master initiates the write cycle by placing the target address on PADDR, asserting PWRITE, and activating the corresponding PSELx line. Once PENABLE is turned on, data is placed on PWDATA, marking the start of the write transaction. The PREADY signal plays a pivotal role in determining whether the slave can accept the data immediately or requires additional time. If the slave is ready, it raises PREADY, allowing an instant transfer. Otherwise, it remains low, instructing the master to hold the transaction until it can process the request. This controlled synchronization prevents data corruption and ensures a seamless transfer process.

To further validate the APB implementation, a Cadence-based synthesis and verification process was conducted. The design was developed using Verilog HDL, and comprehensive timing, power, and area analyses were performed using the Genus synthesis tool. The comparison of area and power consumption for different design instances is summarized in Table I. The testbench was designed to evaluate both read and write cycles for two slave devices under scenarios with and without wait states, making sure of robust functionality.

Additionally, a register-transfer level (RTL) representation of the design was generated, followed by the synthesis and netlist generation, shown in respectively. The extracted timing, power, and area reports, illustrated in providing insights into the efficiency of the APB implementation. The final analysis confirms that the design meets the required functional and performance specifications.

VII. CONCLUSION

This research provides a detailed exploration of the AMBA APB architecture, emphasizing its role in facilitating efficient and low-power peripheral communication. The APB protocol was successfully designed, implemented, and verified for a dual-slave system using the Cadence toolchain. Functional verification of both read and write operations confirmed the correct execution of transactions under normal and wait-state conditions.

Furthermore, RTL synthesis, power analysis, and area estimation were conducted to evaluate the design's hardware efficiency. The results demonstrate an optimized power-performance tradeoff, validating APB's suitability for low-power embedded applications. Future work could extend this study towards physical layout generation and performance enhancements by integrating advanced clock gating and power optimization techniques to further improve efficiency.

VIII. ACKNOWLEDGMENT

We would like to express our sincere gratitude to all those who contributed to the successful completion of our research project titled *"Design and Verification of APB AMBA Protocol."*First and foremost, we extend our heartfelt thanks to our advisor, Ms. Deepmala, for her unwavering guidance, insightful feedback, and continuous support throughout the course of this project. Her expertise and encouragement were instrumental in shaping the direction and quality of our work.We are also grateful to the faculty and staff of the [Name of the Department], whose support and provision of necessary resources created a conducive environment for our research.

Our appreciation extends to our colleagues and classmates for their collaboration, shared knowledge, and moral support. Their camaraderie made this journey both intellectually rewarding and personally enjoyable.

REFERENCES

[1] C. Ma, Z. Liu, and X. Ma, “Design and Implementation of APB Bridge based on AMBA 4.0,” Proc. IEEE Conf. on System-on-Chip Design, pp. 193-196, 2011.
[2] R. Ma, V. R. M., and P. V. Hunagund, “Design of Low Bandwidth Peripherals Using High-Performance Bus Architecture,” Procedia Engineering, vol. 30, pp. 274-282, 2012.
[3] G. Akhila, D. Ravali, J. Mallikarjunarao, and B. Ramarao, “Design and Implementation of AMBA APB Protocol Using System Verilog,” Int. J. Innov. Res. Technol., vol. 9, no. 1, pp. 1664-1666, 2022.
[4] ARM Ltd., AMBA APB Protocol Specification, ARM IHI 0024D, 2021.
[5] R. E., G. K. B., C. Y. C., and P. R. T. M., “Design and Verification of Advanced Microcontroller Bus Architecture (AMBA) Advanced Peripheral Bus (APB) Protocol,” Proc. IEEE Int. Conf. on Smart Systems for Electrical Sciences (ICSSES