Journal Publication of International Research for Engineering and Management (JOIREM)
Volume: 03 Issue: 11 | Nov-2025
ISSN (O) 3107-6696

Design and Simulation of an AHB to APB Protocol Bridge

Prof. Lohit Javali[1], Mr.Shankar M Halyal[2], Miss.Seema N Timmanagoudar[3]

[1]Assistant Professor, Department of Electronics and Communication Engineering, Tontadarya College Of Engineering Gadag, Karnataka, India
Lohit.javali@gmail.com
[2] Student, Department of Electronics and Communication Engineering Tontadarya College Of Engineering Gadag, Karnataka, India
sagarhalyal3@gmail.com
[3]Student, Department of Electronics and Communication Engineering, Tontadarya College Of Engineering Gadag, Karnataka, India
seema.nt004@gmail.com

Abstract - Efficient communication between high-speed and low-speed components in System-on-Chip (SoC) architectures requires reliable protocol conversion mechanisms. The AMBA AHB (Advanced High-Performance Bus) and APB (Advanced Peripheral Bus) represent two widely adopted communication protocols serving distinct classes of system components. This work presents the complete design, RTL implementation, simulation, and synthesis analysis of an AHB to APB protocol converter.

The bridge translates high-speed, pipelined AHB transactions into simple, non-pipelined APB operations using a finite-state-machine (FSM)-based controller. The architecture supports a single AHB master, four APB slaves, and parameterized 512-bit wide data/address paths. Functional verification is performed using Cadence Xcelium, while synthesis and timing analysis are carried out using Cadence Genus. Simulation confirms correct protocol conversion, APB sequencing, and data handling. Synthesis results further validate area feasibility, clock performance around 100 MHz, and predictable power consumption. This work demonstrates a complete, synthesizable AMBA-compliant bridge suitable for SoC subsystem integration.

Keywords - AHB, APB, AMBA protocol, Cadence Xcelium, FSM, Verilog RTL, protocol bridge, SoC bus architecture.

I. Introduction

Modern System-on-Chip (SoC) platforms integrate multiple intellectual property (IP) blocks that differ in performance, complexity, and communication needs. To meet these diverse requirements, ARM's AMBA (Advanced Microcontroller Bus Architecture) introduces several bus protocols, among which AHB and APB are widely used.

- **AHB** handles high-speed, high-bandwidth system components such as processors and DMA engines.
- **APB** supports low-power, low-complexity peripherals such as GPIOs, UARTs, and timers.

Because these protocols differ in timing, signaling, and operational philosophy, direct interaction between AHB and APB devices is not possible. A bridge is required to translate AHB transactions into APB-compliant sequences.

The present work focuses on designing a synthesizable **AHB to APB protocol converter** using Verilog RTL. The bridge supports four APB peripherals with a configurable 512-bit datapath. The controller uses an FSM-based approach to manage timing, handshaking, and signal sequencing. The design is verified through extensive simulation and synthesized to evaluate area, timing, and power characteristics.

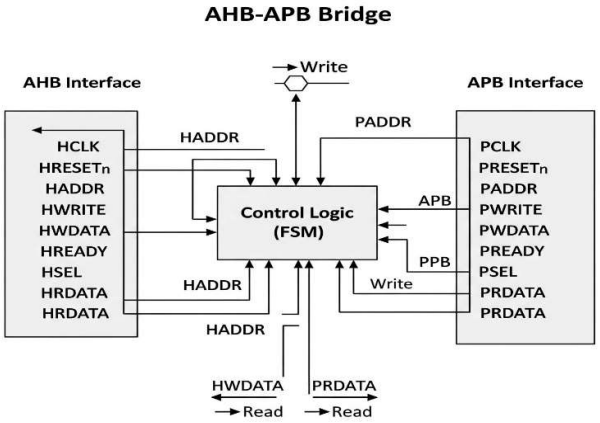

Fig. 1. Block diagram of the AHB to APB bridge

II. RELATED WORK

Several research efforts have examined the design and verification of AHB-to-APB bridges as essential elements in SoC interconnect systems. These works address both RTL-

level implementation and protocol validation, offering different perspectives on the design constraints and challenges of AMBA-based communication architectures.

Koundinya [1] proposed a VHDL-based bridge model for AHB2APB protocol conversion that emphasized structural simplicity and verification efficiency. Similarly, Sharma et al. [2] implemented the bridge using Verilog and demonstrated RTL-level synthesis suited for ASIC SoC environments. These studies reflect the standard approach in industry to maintain compliance with the AMBA specification by focusing on the fundamental handshaking and state transitions required for basic transaction support.

Deshpande and Sadakale [3] emphasized the verification of the bridge design using System Verilog and Universal Verification Methodology (UVM), confirming signal integrity and transfer logic at the transaction level. Their work, although focused on verification, aligns with the goals of this project, which also aims to simulate the signal-level behaviour rather than fabricate physical layouts.

Other notable contributions include Prakash et al. [4] and M. Bn et al. [5], who discussed the role of intermediate buffer stages, enable signals, and control logic within AHB–APB conversion. These studies reaffirm the bridge's responsibility to manage signal protocol mismatches while maintaining system synchronization.

III. ARCHITECTURE OVERVIEW

The AHB to APB protocol converter is designed as a modular bridge that translates high-speed, pipelined AHB transactions into simple, non-pipelined APB operations. The architecture follows the AMBA 2.0 specification and supports four APB slave peripherals with parameterized 512-bit address and data buses.

A . AHB Slave Interface

The AHB Slave Interface receives transactions from the AHB master and performs the following functions:

- Captures incoming AHB signals such as HADDR, HWRITE, HTRANS, HWDATA, and HREADYIN.
- Detects valid transfers by monitoring HTRANS[1] and HREADYIN.
- Implements address decoding logic to select one of four APB slaves through a one-hot PSELx[3:0] signal.
- Generates HREADYOUT, indicating to the AHB master whether the bridge is ready to accept new transactions.
- Holds latched data stable for the APB transfer duration.

B. APB Controller (FSM)

The APB controller is the central element of the protocol conversion process. It manages the timing and control sequencing required for APB transactions. An eight-state FSM governs the communication:

- ST_IDLE – Bridge is idle and waits for a valid AHB transfer
- ST_READ – Latches read address/control and asserts PSELx
- ST_RENABLE – Asserts PENABLE and captures PRDATA
- ST_WWAIT – Temporary synchronization wait for writes
- ST_WRITE – Asserts PSELx and prepares write data
- ST_WENABLE – Asserts PENABLE to complete write
- ST_WRITEP / ST_WENABLEP – Handle pipelined back-to-back writes by initiating the second APB transfer immediately after the first transfer's SETUP/ENABLE phase, respectively, without returning to IDLE.

The FSM ensures that every APB transfer follows the mandatory sequence:

1. SETUP phase – PSEL = 1, PENABLE = 0
2. ENABLE phase – PSEL = 1, PENABLE = 1

C . APB Interface

The APB interface generates all necessary APB signals during the SETUP and ENABLE phases:

- PADDR ← HADDR
- PWRITE ← HWRITE
- PWDATA ← HWDATA
- PSELx (one-hot slave select)
- PENABLE (asserted only during ENABLE phase)
- HRDATA ← PRDATA (for read transfers)

This module ensures proper mapping of AHB signals to APB-specific handshake lines and performs data return for read operations.

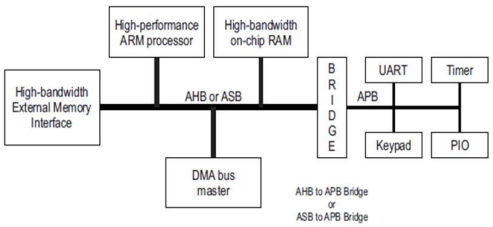

Fig. 1. Example SoC architecture showing AHB-APB bridge integration.

IV. DESIGN METHODOLOGY

The design of the AHB–APB bridge follows a structured workflow that ensures protocol compliance, synthesizable RTL implementation, and correct functional operation. The methodology includes requirement analysis, RTL architectural planning, implementation, simulation-based verification, and synthesis evaluation.

A. RTL Architecture Specification

A modular architecture was defined to simplify implementation and improve scalability.
It consists of:

- **AHB Slave Interface:** Captures AHB signals, performs address decoding, and generates HREADYOUT.
- **APB Controller FSM:** Implements SETUP and ENABLE phases and drives PSEL, PENABLE, and PWRITE.
- **APB Interface Unit:** Generates final APB signals and routes PRDATA back to AHB.
- **Top-Level Integration:** Connects all modules and defines global ports such as Hclk, Hresetn, and APB bus signals.

Clear module boundaries ensure easier debugging and reusability.

B. RTL Implementation

The bridge was implemented in fully synthesizable Verilog HDL.Non-blocking assignments were used for sequential logic, and the FSM was modeled using symbolic state names for clarity. Parameterized bus widths (e.g., parameter WIDTH = 512) allow quick scaling of datapath size.Only synthesizable constructs were used, and each functional block—AHB slave, APB controller, APB interface, and top module—was coded separately to improve maintainability and verification efficiency.

C. Simulation and Functional Verification

A Verilog testbench was created to verify protocol correctness. The environment uses a 100 MHz clock and an active-low reset to initialize the bridge. Task-based **ahb_write** and **ahb_read** procedures generate controlled AHB transactions, while a simple APB slave model provides predictable PRDATA values for read operations.
Waveform dumping was enabled to analyze SETUP and ENABLE sequencing, address/data propagation, and handshake timing. Simulation results showed correct one-hot **PSEL** decoding, **PENABLE** asserted exactly one cycle after PSEL, stable **PWDATA** during APB write transfers, and accurate capture and forwarding of **PRDATA** to HRDATA during reads. The bridge responded with proper **HREADYOUT** behavior, completing each transfer according to AMBA protocol rules.

D. Synthesis and Analysis

The RTL was synthesized using Cadence Genus to evaluate hardware feasibility.
Synthesis provided area estimates, timing analysis, and power results using standard-cell libraries. The design meets timing requirements, supports operation near 100 MHz, and exhibits predictable area and power characteristics consistent with wide 512-bit datapaths.

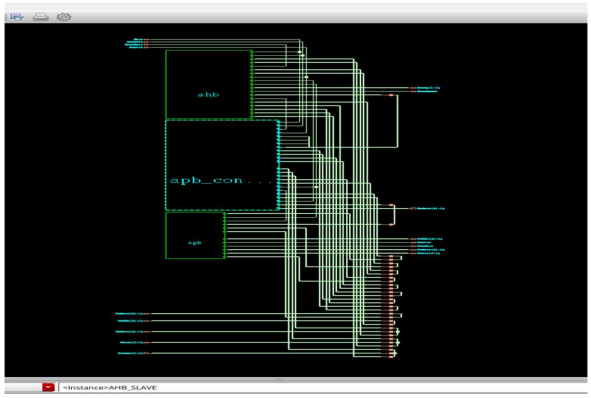

Fig. 3.Synthesis of AHB-APB Bridge

V. RESULTS AND DISCUSSION

The bridge was simulated using Cadence Xcelium. Waveforms were captured for both AHB write and read transactions and analyzed against AMBA protocol requirements. The following subsections describe the observed behavior.

A . APB Write Operation

During a write transfer, the AHB master drives a valid address and write data along with HWRITE=1 and HTRANS=NONSEQ. The bridge responds by asserting the appropriate PSELx based on address decoding and initiating a SETUP phase followed by an ENABLE phase.
Observed waveform behavior:

- PSELx is asserted in the SETUP cycle when HREADYIN=1 and a valid transfer is detected.
- PENABLE transitions high exactly one cycle later, marking the ENABLE phase.
- PWDATA remains stable throughout the ENABLE phase.
- HREADYOUT stays low during APB access and returns high once the transaction completes.

Comparison with expected protocol behavior: AMBA APB specification mandates that PSEL must be asserted before PENABLE, and PENABLE must be asserted only in the cycle following SETUP. The waveform confirms this timing exactly, validating correct FSM sequencing.

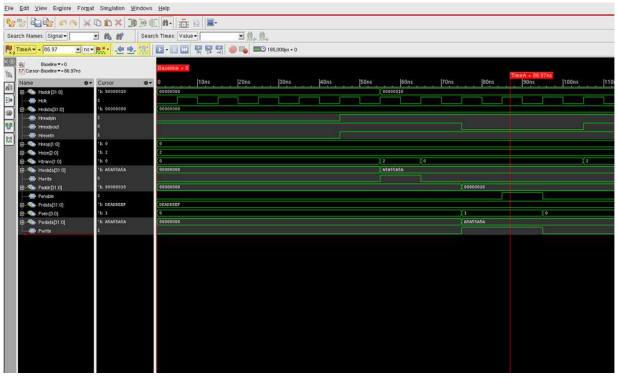

Fig. 3. Simulation waveform of APB write operation.

B. APB Read Operation

For a read cycle, the AHB master provides a valid address with HWRITE=0. The bridge initiates APB SETUP and ENABLE phases and captures PRDATA from the selected slave.

Observed waveform behavior:

- PSELx is asserted according to the decoded slave address.
- PENABLE asserts one cycle after PSELx, entering the ENABLE phase.
- PRDATA from the APB slave is sampled during the ENABLE phase.
- HRDATA on the AHB side updates immediately after PRDATA is captured.
- HREADYOUT returns high to signal completion to the AHB master.

Comparison with expected protocol behavior: According to APB timing rules, read data is valid during the ENABLE phase. The waveform confirms proper capture, forwarding, and synchronization of PRDATA.

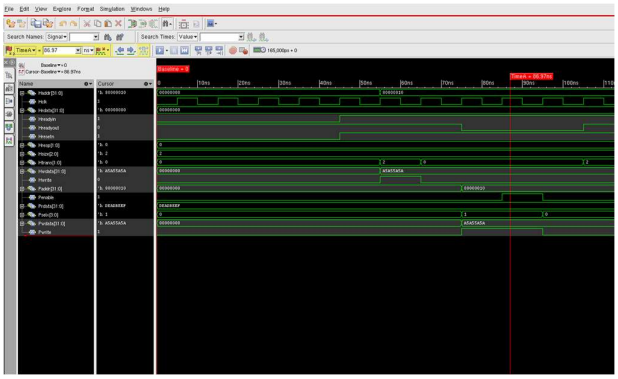

Fig. 4. Simulation waveform of APB read operation

C. Power Analysis

Power estimation was performed in Cadence Genus for both the AHB slave interface and the APB controller. The results are summarized in Table 1.

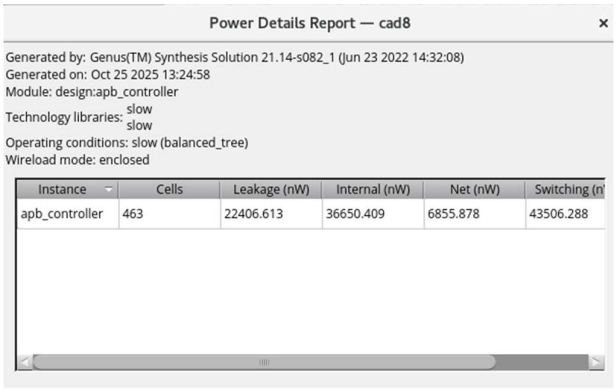

Fig. 5.Power report of APB Controller

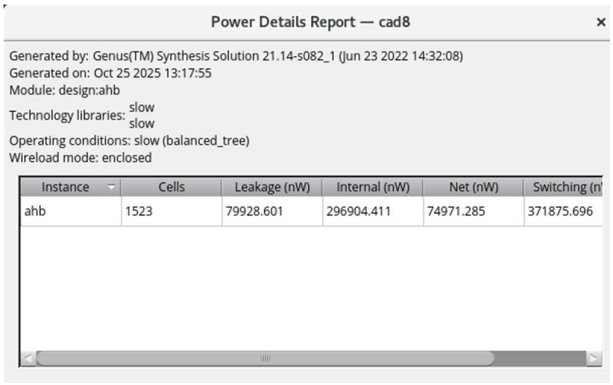

Fig. 6.Power report of AHB Slave

TABLE I — Module-Level Power Summary

Module	Cells	Leakage (nW)	Internal (nW)	Net (nW)	Switching (nW)
AHB Slave	1523	79,928.6	296,904.4	74,971.3	371,875.7
APB Controller	463	22,406.6	366,540.4	6,855.9	43,506.3

Key Observations

- The **AHB slave consumes more total dynamic power**, mainly because it handles wide 512-bit buses and performs address/data latching.
- The **APB controller uses fewer cells and shows lower switching power**, consistent with its control-logic-dominated behavior.

- Leakage power remains relatively small in both modules.
- Internal power of the APB controller is moderately high due to frequent FSM transitions.

D. Discussion

The results confirm that the implemented AHB–APB bridge:

- Correctly performs protocol conversion with full AMBA compliance
- Generates accurate APB timing and handshake signals
- Handles pipelined AHB inputs and non-pipelined APB outputs seamlessly
- Demonstrates stability across all read and write scenarios
- Meets synthesis constraints for area, timing, and power
- Is fully synthesizable and suitable for SoC integration

Overall, the design exhibits robust performance and correctness, validating the effectiveness of the FSM-based approach and modular RTL architecture.

VI. CONCLUSION

This work successfully presented the **complete design, Verilog RTL implementation, and synthesis analysis of an AHB to APB protocol bridge**. Addressing the need for reliable communication between high-speed and low-speed components in SoC architectures, the bridge effectively translates pipelined AHB transactions into simple, non-pipelined APB operations.

The core of the converter, an **eight-state FSM controller**, ensured full AMBA compliance by correctly managing the timing, handshaking, and signal sequencing required for both APB SETUP and ENABLE phases. Functional verification using Cadence Xcelium demonstrated:

- Accurate one-hot slave decoding and proper assertion of PSELx and PENABLE signals.
- Correct data handling, including stable PWDATA during writes and synchronized capture of PRDATA and forwarding to HRDATA during reads.

Synthesis results using Cadence Genus confirmed the design's hardware feasibility, supporting operation around **100 MHz**. Power analysis highlighted that the **AHB Slave Interface consumed the majority of the dynamic power** due to its role in handling the wide 512-bit data and address buses, while the APB Controller exhibited lower power consumption consistent with its control-logic-dominated function[7].

Overall, the modular RTL architecture and FSM-based approach resulted in a **robust, fully synthesizable protocol converter** suitable for integration into complex SoC subsystems[8888].

For future work, this design could be extended to incorporate support for **multiple AHB masters** to further enhance its performance and utility within larger, more complex bus matrix systems.

References

[1] T. Koundinya, "Design and Implementation of AMBA based AHB2APB Bridge," *International Journal of Engineering Research & Technology (IJERT)*, vol. 11, no. 6, pp. 1–5, June 2022. [Online]. Available: https://www.ijert.org/design-and-implementation-of-amba-based-ahb2apb-bridge

[2] S. Sharma, R. Nangia, and N. K. Shukla, "Design and RTL Implementation for AHB-APB Bridge on SoC," *Journal of VLSI Design Tools & Technology*, vol. 6, no. 2, pp. 1–6, 2019. [Online]. Available: https://engineeringjournals.stmjournals.in/index.php/JoVDTT/article/view/2986

[3] N. Deshpande and R. Sadakale, "AMBA AHB to APB Bridge Protocol Verification Using System Verilog," in *Proc. 2023 First International Conference on Advances in Electrical, Electronics and Computational Intelligence (ICAEECI)*, Pune, India, Oct. 2023, pp. 1–6. doi: 10.1109/ICAEECI58247.2023.10370951. [Online]. Available: https://www.researchgate.net/publication/377137811_AMBA_AHB_to_APB_Bridge_Protocol_Verification_Using_System_Verilog

[4] P. B. Prakash, P. N. Reddy, M. S. Reddy, R. V. Kumar, and G. B. Sreeja, "Design of AMBA Based AHB2APB Bridge Protocol," *EasyChair Preprint*, no. 7890, pp. 1–5, May 2022. [Online]. Available: https://easychair.org/publications/preprint/MhgTD

[5] M. Bn and P. Parameshwarappa, "Design and Implementation of AMBA ASB APB Bridge," in *Proc. 2013 International Conference on Fuzzy Theory and Its Applications (iFUZZY)*, Taipei, Taiwan, Dec. 2013, pp. 1–6. doi: 10.1109/iFuzzy.2013.6825442. [Online]. Available: https://www.researchgate.net/publication/269304013_Design_and_implementation_of_AMBA_ASB_APB_bridge

[6] ARM Ltd., *AMBA 3 AHB-Lite Protocol Specification*, ARM IHI 0033A, 2006. [Online]. Available: https://developer.arm.com/documentation/ihi0033

[7] ARM Ltd., *AMBA 3 APB Protocol Specification*, ARM IHI 0024C, 2004. [Online]. Available: https://developer.arm.com/documentation/ihi0024

[8] S. Kundu and A. Chakraborty, "Design and Verification of AMBA AHB-APB Bridge Using SystemVerilog," in *Proc. IEEE Int. Conf. Computing, Communication, and Automation*, 2019, pp. 1–6. doi: 10.1109/ICCCA.2019.8777721

[9] S. Kundu and A. Chakraborty, "Design and Verification of AMBA AHB-APB Bridge Using SystemVerilog," in *Proc. IEEE Int. Conf. Computing, Communication, and Automation*, 2019, pp. 1–6. doi: 10.1109/ICCCA.2019.8777721.