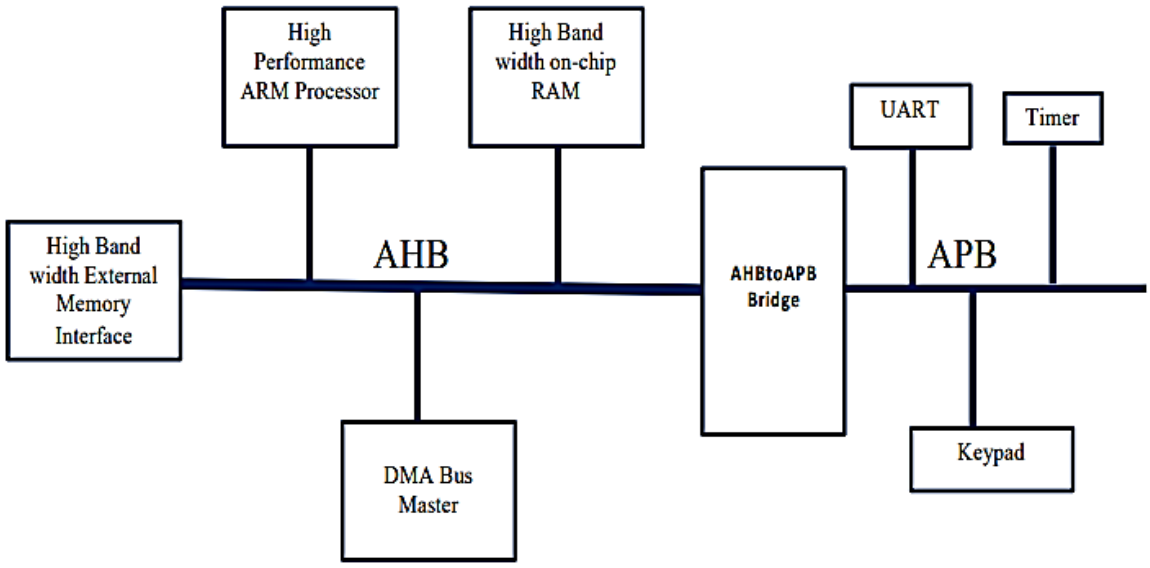
CHAPTER 1 – INTRODUCTION

1.1 Overview of AMBA Bus Architecture

The **Advanced Microcontroller Bus Architecture (AMBA)** is a widely used bus protocol standard developed by ARM for designing **System-on-Chip (SoC)**.
It provides multiple types of buses to connect processors, memories, and peripherals efficiently. The two most common buses are:

- **AHB (Advanced High-performance Bus):**
  - High-speed bus used for processor, memory, and high-bandwidth peripherals.
  - Supports burst transfers, pipelining, and multiple masters.
  - Suitable for performance-critical blocks.
- **APB (Advanced Peripheral Bus):**
  - Low-speed bus used for simple peripheral devices like UART, GPIO, Timers, SPI, etc.
  - Very simple interface, low power, and minimal area requirement.
  - No pipelining or burst transfer, only single data transfers.



Thus, AMBA architecture provides both **performance (via AHB)** and **simplicity/low power (via APB)**.

1.2 Need for Bridging AHB and APB

In modern SoCs, the CPU and memory work on **AHB**, but most simple peripherals (UART, GPIO, I2C, etc.) are designed for **APB**.
Since these two buses have **different protocols, speeds, and signals**, a **Bridge** is required to connect them.

The **AHB to APB Bridge** acts as a **translator**:

- Receives AHB transactions from master (CPU/memory).
- Converts them into APB-compatible transfers.
- Sends responses back to the AHB master.

Without the bridge, AHB masters cannot directly communicate with APB peripherals.

1.3 Applications of AHB to APB Bridge

The AHB to APB Bridge is used in almost all SoCs and microcontrollers, mainly to connect **high-performance cores to low-power peripherals**. Some examples include:

- ARM-based microcontrollers (Cortex-M series).
- Mobile SoCs for connecting GPU/CPU to UART, SPI, I2C.
- Embedded processors in automotive and IoT devices.
- Medical and industrial image processing chips where control peripherals run on APB.

1.4 Objectives of the Project

The main objectives of this project are:

- To design and implement a **functional AHB to APB Bridge** using Verilog HDL.
- To study the working of **AHB and APB protocols** and their interfacing requirements.
- To develop an **FSM-based bridge** that converts high-speed pipelined AHB transactions into simple APB single transfers.
- To perform **functional verification** using a testbench and analyze simulation waveforms.
- To highlight the **advantages (low power, simple design)** and possible **future enhancements** of the bridge.

CHAPTER 2 – DESIGN AND IMPLEMENTATION

2.1 AHB Protocol Basics

The **Advanced High-performance Bus (AHB)** is a pipelined system bus used for **high-speed data transfer**. Key features:

- Supports multiple masters and multiple slaves.
- Provides burst transfers for efficiency.
- Uses **address phase** and **data phase** (pipelined).
- Important signals:
    - HADDR → Address bus
    - HTRANS → Transfer type (IDLE, BUSY, NONSEQ, SEQ)
    - HWRITE → Write/read control
    - HWDATA / HRDATA → Write and Read data
    - HREADY → Transfer done indication
    - HRESP → Transfer response

2.2 APB Protocol Basics

The **Advanced Peripheral Bus (APB)** is a simple low-power bus designed for peripherals. Features:

- No pipelining, only **single transfers**.
- Each transfer has **Setup** and **Enable** phases.
- Signals are simpler compared to AHB:
    - PADDR → Address bus

- PWRITE → Write/read control
- PWDATA / PRDATA → Write and Read data
- PSEL → Peripheral select
- PENABLE → Enable signal for transfer
- PREADY → Slave ready signal

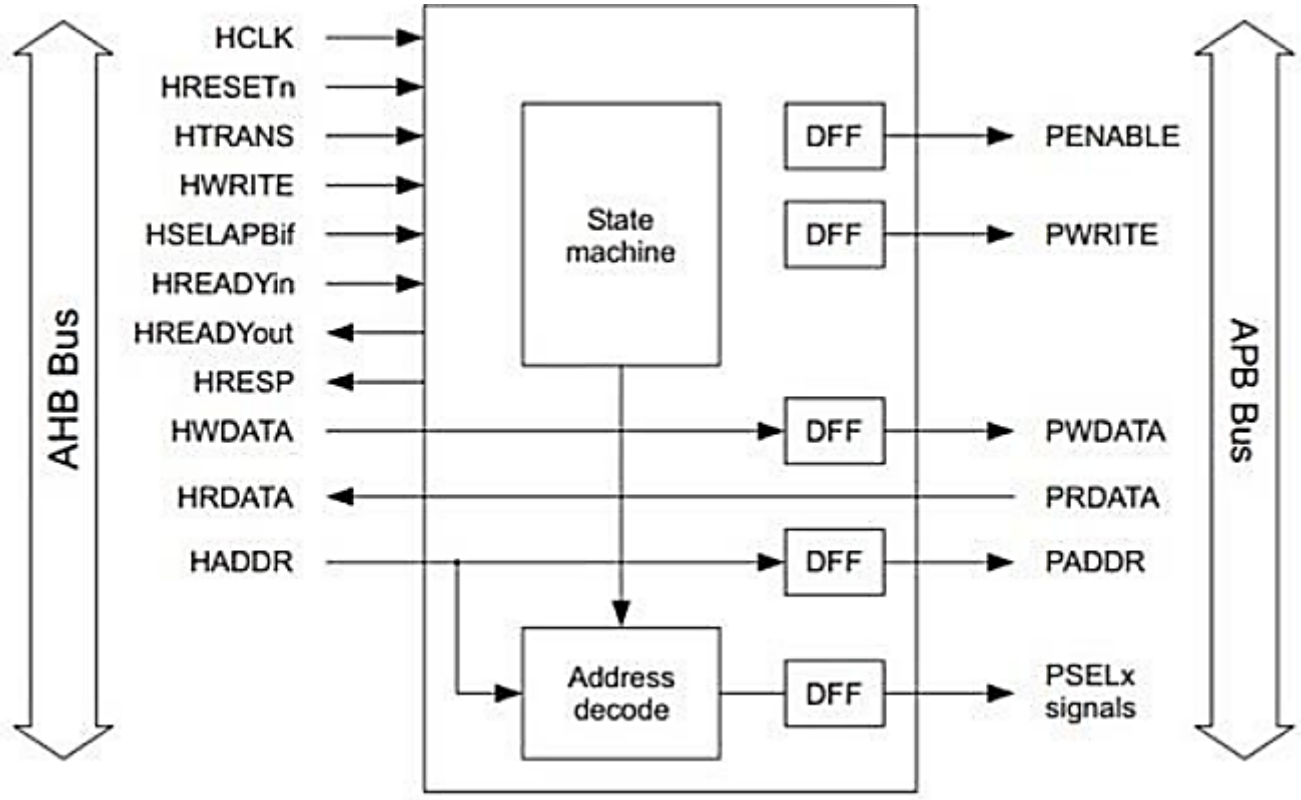
2.3 Role of the Bridge in SoC Design

The **AHB to APB Bridge** sits between high-performance masters (CPU/Memory) and low-power peripherals.

- On the **AHB side**: accepts transactions with pipelined control.
- On the **APB side**: generates control signals for simple peripherals.
- Acts as a **protocol converter** and ensures correct timing/handshaking.

2.4 Functional Block Diagram of AHB to APB Bridge

The bridge internally contains:



- **AHB Interface Unit** – captures AHB transfers.
- **FSM Controller** – generates APB control signals.
- **APB Interface Unit** – drives APB signals for peripheral access.

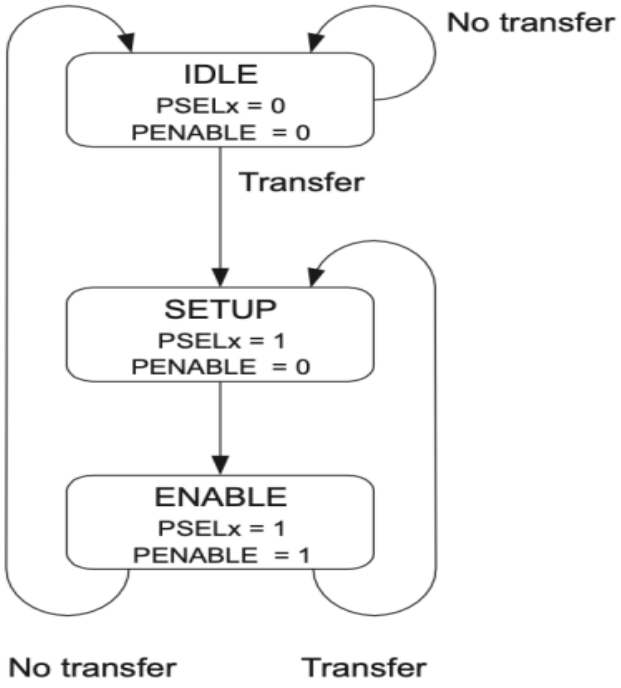
2.5 FSM (Finite State Machine) for Bridge Operation

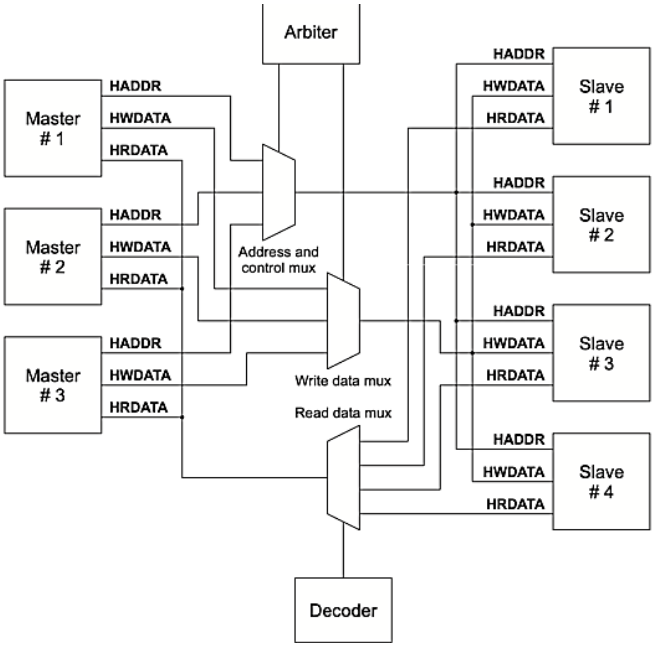
The AHB to APB bridge typically works in **3 states**:

1. **IDLE State**
    - No transfer happening.

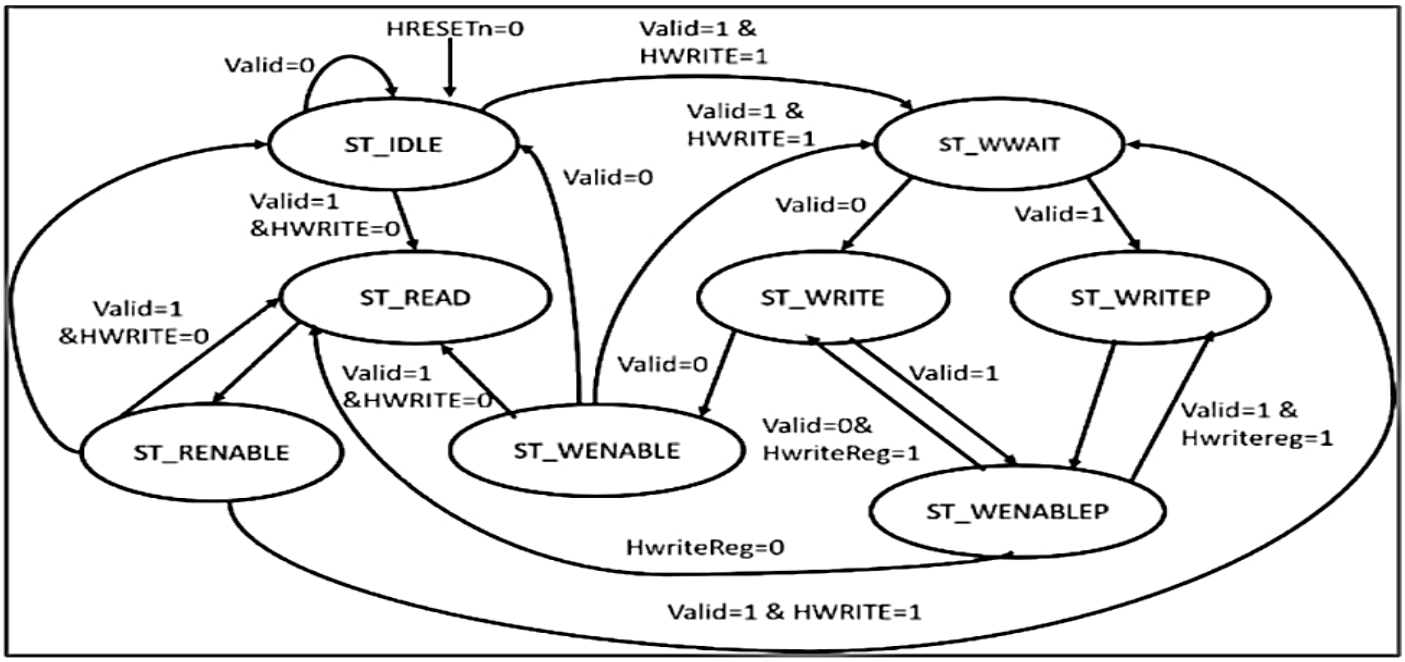
- Waits for a valid AHB transaction (HTRANS ≠ IDLE).

2. **SETUP State**
   - Latches AHB address, control, and write data.
   - Asserts PSEL to select the APB peripheral.
   - Prepares for transfer.
3. **ENABLE State**
   - Asserts PENABLE to start APB transaction.
   - Waits for PREADY from peripheral.
   - On completion → send response back to AHB and return to **IDLE**.





This FSM ensures **AHB pipelined transfers** are safely converted to **APB two-phase transfers**.



2.6 Verilog Design Methodology

The design flow followed:

- **Step 1:** Define module ports for AHB and APB interfaces.
- **Step 2:** Write FSM logic for state transitions (IDLE → SETUP → ENABLE).
- **Step 3:** Implement AHB interface (capture HADDR, HWRITE, HWDATA).
- **Step 4:** Implement APB interface (drive PADDR, PWRITE, PWDATA, PSEL, PENABLE).
- **Step 5:** Generate proper response (HREADY, HRESP) back to AHB.
- **Step 6:** Verify with a **Verilog testbench** simulating both read and write transfers.

2.7 Testbench and Verification Approach

For verification, a **self-checking testbench** is written:

- AHB Master model generates **read and write transactions**.
- APB Slave model responds with **PREADY** and data.
- Monitors check if:
    - Address and data are correctly transferred.
    - Timing (Setup + Enable phases) is maintained.
    - Responses are correctly sent back to AHB.

Simulation waveforms are analyzed in **GTKWave**, showing transitions of HADDR, HWRITE, HWDATA, PADDR, PWDATA, PSEL, PENABLE, and PREADY.

CHAPTER 3 – RESULTS AND CONCLUSION

3.1 Simulation Waveforms and Output Analysis

The **Verilog simulation** of the AHB to APB Bridge was carried out using a testbench. Both **read and write transfers** were tested.
The following observations were made from waveforms:

- **Write Transfer:**
    - AHB master issues HADDR, HWRITE=1, and HWDATA.
    - Bridge moves from **IDLE → SETUP → ENABLE**.
    - On APB side, PADDR, PWDATA, PWRITE=1 are driven.
    - After PREADY=1, the data is successfully written.
- **Read Transfer:**
    - AHB master issues HADDR, HWRITE=0.
    - Bridge generates APB read cycle (PWRITE=0).
    - Peripheral responds with PRDATA.
    - Bridge sends data back on HRDATA to AHB master.

The **waveforms confirm** that the bridge correctly translates AHB transactions into APB operations.

3.2 Performance Metrics (Simplified)

The bridge was analyzed in terms of basic parameters:

Parameter	Observation
Latency	2 APB cycles (Setup + Enable)
Throughput	Single transfer per APB cycle (non-pipelined)
Area (logic size)	Very small (FSM + registers only)
Power	Low (APB design reduces switching activity)
Scalability	Can support multiple peripherals by decoding PADDR

3.3 Key Features Achieved

- Successfully converts **AHB pipelined transactions** into **APB two-phase transfers**.
- Maintains **protocol correctness** and safe handshaking.
- Provides **low power and simple interface** for peripherals.
- Supports **both read and write operations**.
- Easily extendable to **multiple APB slaves**.

3.4 Conclusion and Future Scope

- The project demonstrated the **design and verification of an AHB to APB Bridge** using Verilog.
- The bridge plays a **critical role in SoC design**, connecting high-speed cores to low-power peripherals.
- Simulation results show correct translation of AHB transactions into APB signals with proper timing.

Future Scope:

- Add support for **multiple APB slaves** using address decoding.
- Include **error handling (PSLVERR)**.
- Optimize for **timing closure and synthesis** for ASIC/FPGA implementation.
- Extend design to support **AXI to APB bridge** for advanced SoCs.

REFERENCES

[1] **Integrated AHB to APB Bridge Using Raspberry Pi and Artix-7 FPGA**, *arXiv preprint*, Jan 2025. Link

[2] **Design and Implementation of AHB to APB Bridge using Verilog**, *IJEMR*, Mar–Apr 2025. PDF

[3] **UVM-Based Verification of AMBA AHB-To-APB Bridge**, *ResearchGate*, Jun 2025. Link

[4] **Design and Implementation of AHB to APB Bridge using Verilog**, *IJCRT*, Jan 2025. PDF