AHB to APB Bridge Design and Implementation Report

1. Introduction

1.1 Background

The Advanced Microcontroller Bus Architecture (AMBA) specification defines industry-standard on chip communication protocols for designing high-performance embedded microcontrollers. The AHB to APB bridge addresses the critical need to interface between high-performance system components operating on the AHB and low-power peripheral devices on the APB.

1.2 Project Objectives

- Design a synthesizable AHB to APB bridge interface using Verilog HDL
- Implement address latching and decoding functionality
- Generate appropriate control signals (PSELx, PENABLE)
- Support both read and write data transfer operations
- Validate functionality through comprehensive testbench simulation

2. AMBA Bus Architecture Overview

2.1 AMBA Protocol Family

The AMBA specification encompasses three primary bus protocols:

Advanced High-performance Bus (AHB)

- High-performance, high clock frequency operation
- System backbone bus for processors and memory interfaces
- Supports burst transfers and pipelined operations
- Single clock-edge operation with non-tristate implementation
- Wide data bus configurations (64, 128, 256, 512, 1024 bits)

Advanced System Bus (ASB)

- Alternative high-performance system bus
- Suitable where full AHB performance features are not required
- Efficient connection of processors and memory interfaces

Advanced Peripheral Bus (APB)

- Optimized for low-power peripherals
- Minimal power consumption and reduced interface complexity
- Two-cycle transfer protocol (non-pipelined)
- Typically used for configuration registers and low-bandwidth devices

2.2 System Architecture

The typical AMBA system architecture positions the AHB as the high-performance backbone, with APB segments connected through bridge interfaces. This hierarchical approach optimizes both performance and power consumption by placing high-speed components on AHB and low-power peripherals on APB.

3. AHB to APB Bridge Design

3.1 Functional Requirements

The AHB to APB bridge operates as an AHB slave and the sole APB master, performing the following critical functions:

1. **Address Management**: Latches AHB addresses and maintains validity throughout APB transfers
2. **Address Decoding**: Decodes addresses and generates peripheral select signals (PSELx)
3. **Data Routing**: Manages bidirectional data flow between AHB and APB domains
4. **Control Signal Generation**: Produces APB enable signals (PENABLE) with proper timing
5. **Transfer Conversion**: Converts AHB transfers into equivalent APB protocol transfers

3.2 Architecture Components

3.2.1 AHB Slave Interface

The AHB slave interface responds to transfers initiated by AHB masters, utilizing:

- HSELx select signal for transfer recognition
- Address and control signal monitoring
- Response generation (HREADY, HRESP) back to AHB master

3.2.2 APB Controller State Machine

A finite state machine controls APB transfer generation with eight primary states:

State Machine :

- **ST_IDLE**: Default state, monitors AHB for transfers
- **ST_WWAIT**: Write wait state for AHB ready signal
- **ST_WRITEP**: Write setup state, address/data preparation
- **ST_WRITE**: Write transfer state with PSELx asserted

- **ST_WENABLEP**: Write enable preparation state
- **ST_WENABLE**: Write enable state with PENABLE asserted
- **ST_READ**: Read transfer state with PSELx asserted
- **ST_RENABLE**: Read enable state with PENABLE asserted

3.3 Signal Interface

AHB Interface Signals

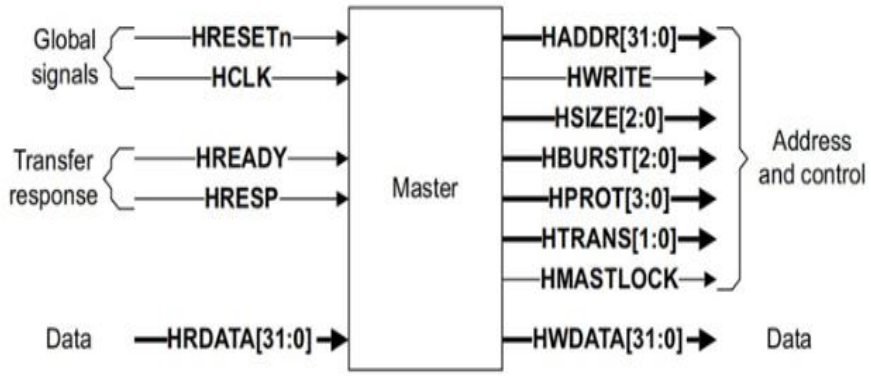
- **HCLK**: System clock
- **HRESETn**: Active-low reset
- **HSEL**: Slave select signal
- **HADDR[31:0]**: Address bus
- **HTRANS[1:0]**: Transfer type
- **HWRITE**: Transfer direction
- **HSIZE[2:0]**: Transfer size
- **HWDATA[31:0]**: Write data
- **HREADY**: Transfer completion
- **HRDATA[31:0]**: Read data
- **HRESP[1:0]**: Transfer response

APB Interface Signals

- **PCLK**: APB clock (typically same as HCLK)
- **PRESETn**: APB reset
- **PADDR[31:0]**: APB address
- **PSELx**: Peripheral select
- **PENABLE**: Enable signal
- **PWRITE**: Direction signal
- **PWDATA[31:0]**: Write data
- **PRDATA[31:0]**: Read data
- **PREADY**: Ready signal (optional)

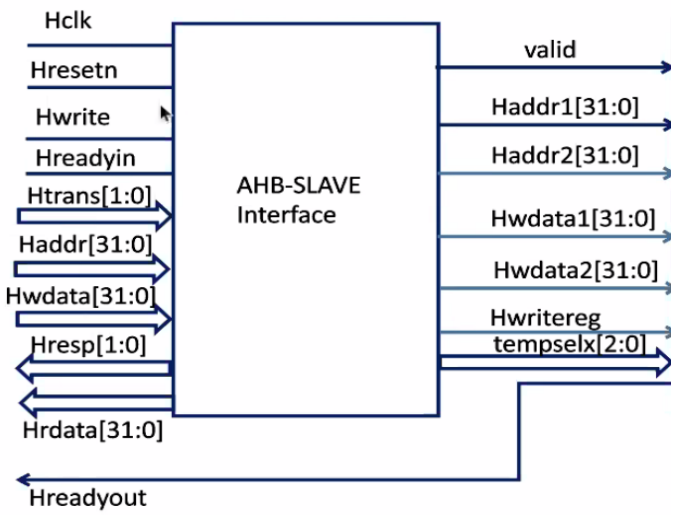
4. BLOCK DIAGRAM

4.1 AHB Master Block



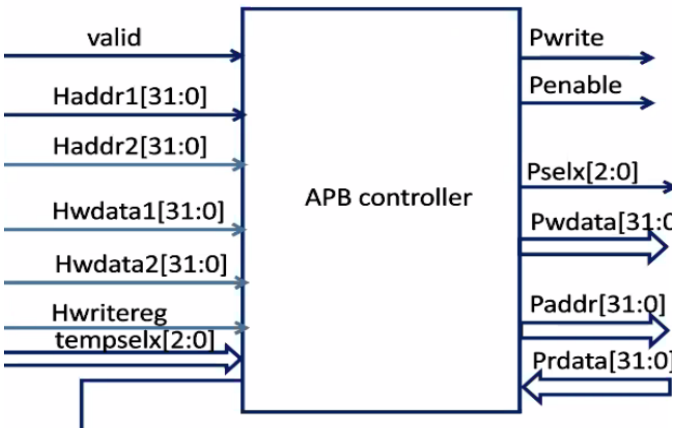
- Initiates read/write transfers on AHB bus
- Generates address, control signals (HADDR, HTRANS, HSIZE, HBURST)
- Provides write data (HWDATA) and receives read data (HRDATA)

4.2 AHB Slave Interface Block



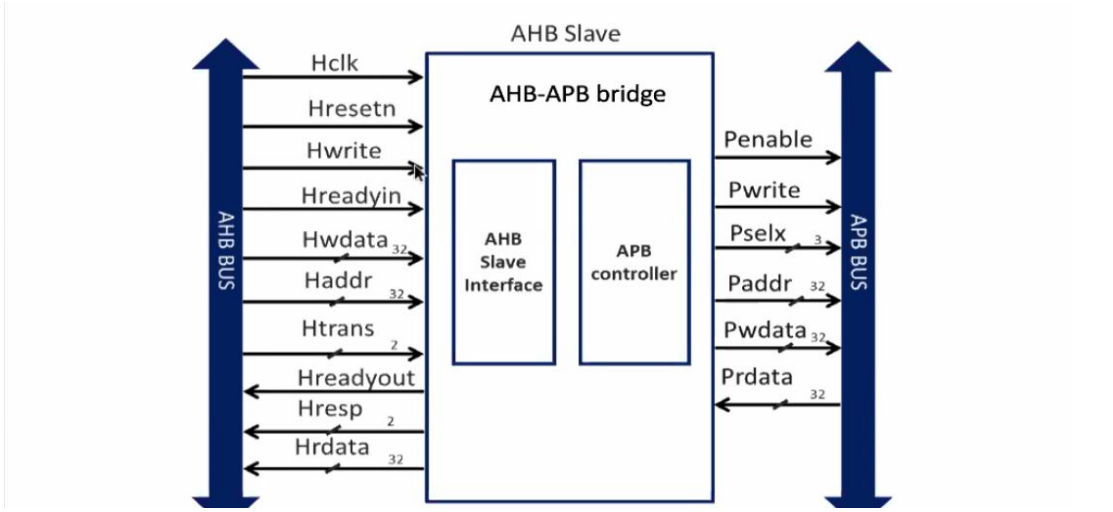
- Handles AHB protocol compliance and signal conditioning
- Latches incoming address and control signals
- Generates internal valid signal for APB controller
- Manages address registers (Haddr1, Haddr2) for transfer tracking
- Buffers write data (Hwdata1, Hwdata2) for APB transfers

4.3 APB Controller Block



- Implements 8-state FSM for APB transfer control
- Generates APB control signals (PENABLE, PWRITE, PSELx)
- Controls APB address (PADDR) and write data (PWDATA) outputs
- Manages state transitions based on valid signal and transfer type
- Handles temporary select signal (tempselx) for peripheral selection

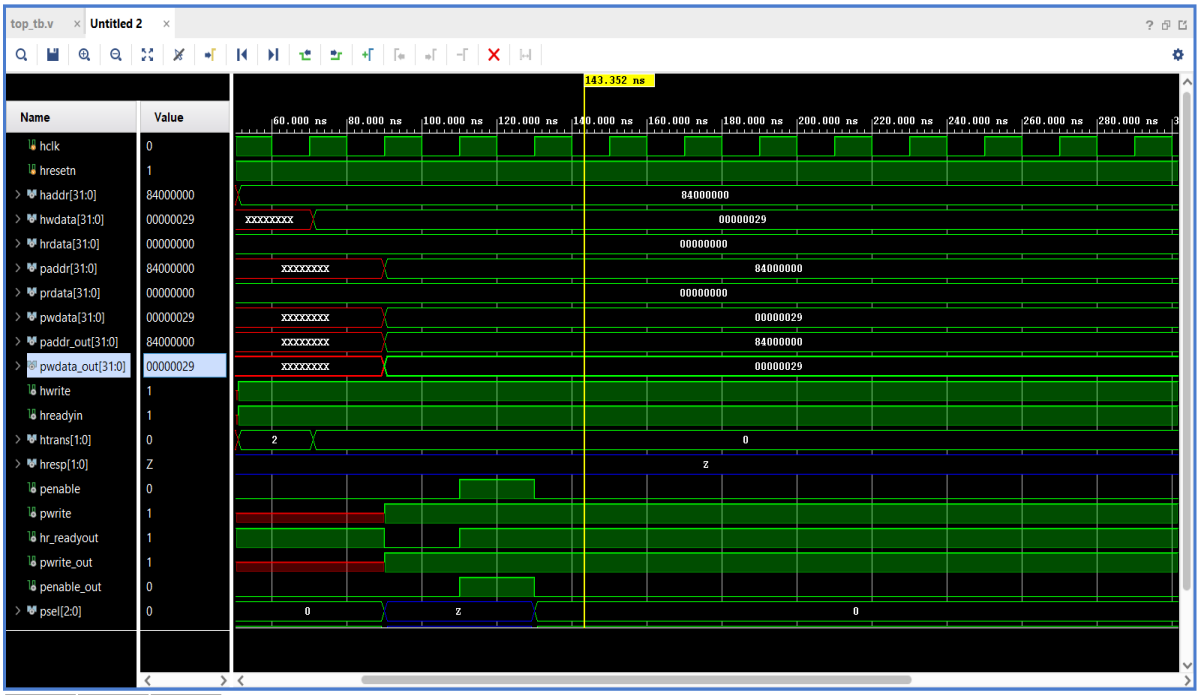
4.4 AHB to APB Bridge Block



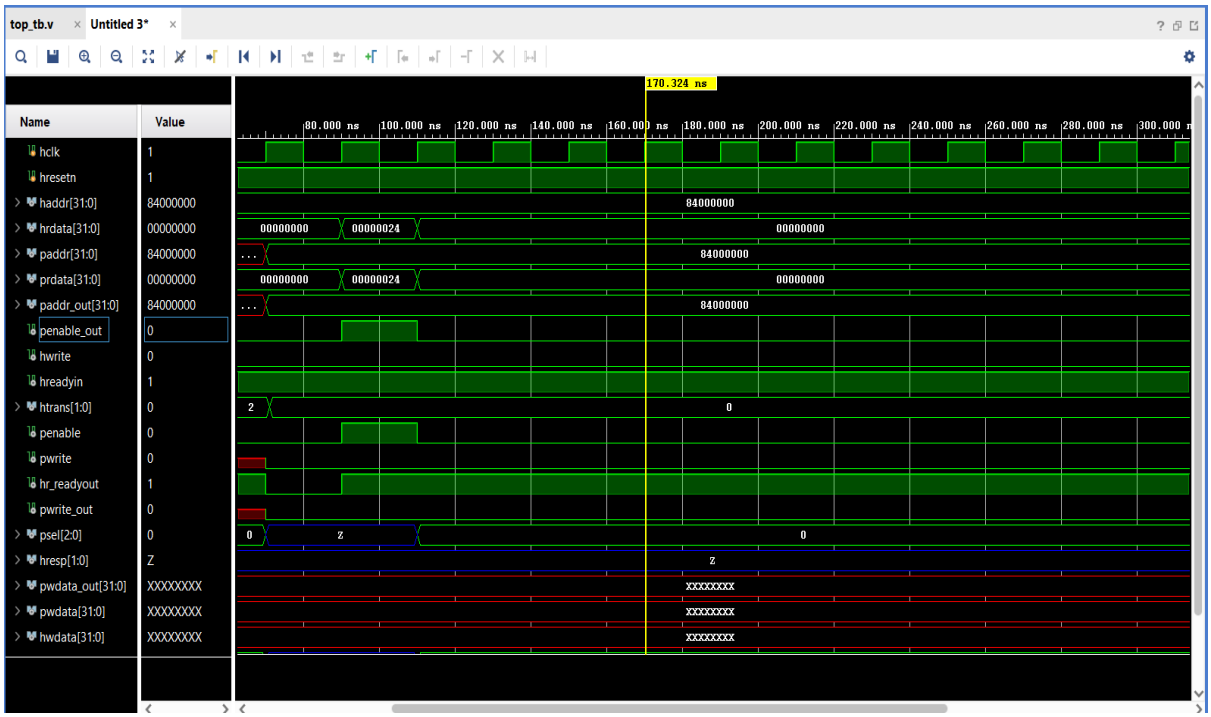
- **Dual Protocol Interface**: Acts as AHB slave and APB master for seamless communication between high-speed and low-power domains.
- **Signal Conversion**: Translates AHB signals (Htrans, Hwrite) to APB protocol signals (Penable, Pwrite, Pselx) with proper timing.
- **Address Decoding**: Generates peripheral select signals (Pselx) based on incoming AHB addresses for target peripheral activation.
- **Synchronized Operation**: Operates on single clock domain (Hclk) with unified reset, eliminating clock crossing complexities.

5. Simulation Results

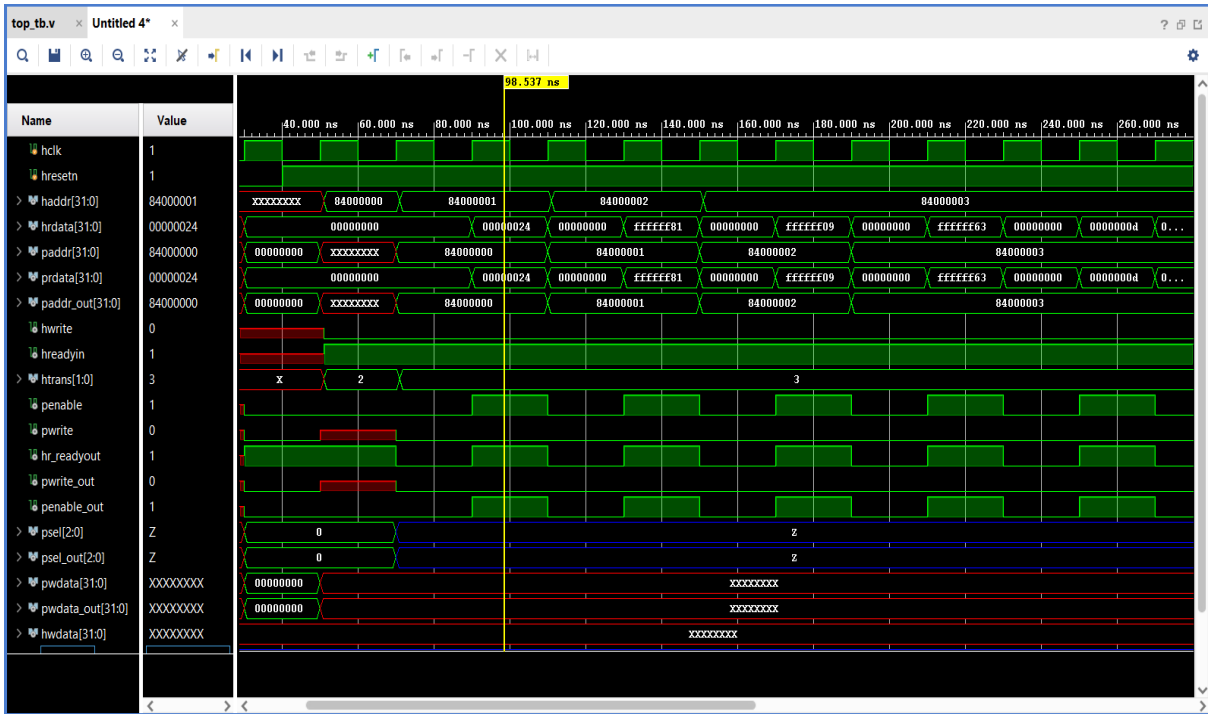
Single Write Operation



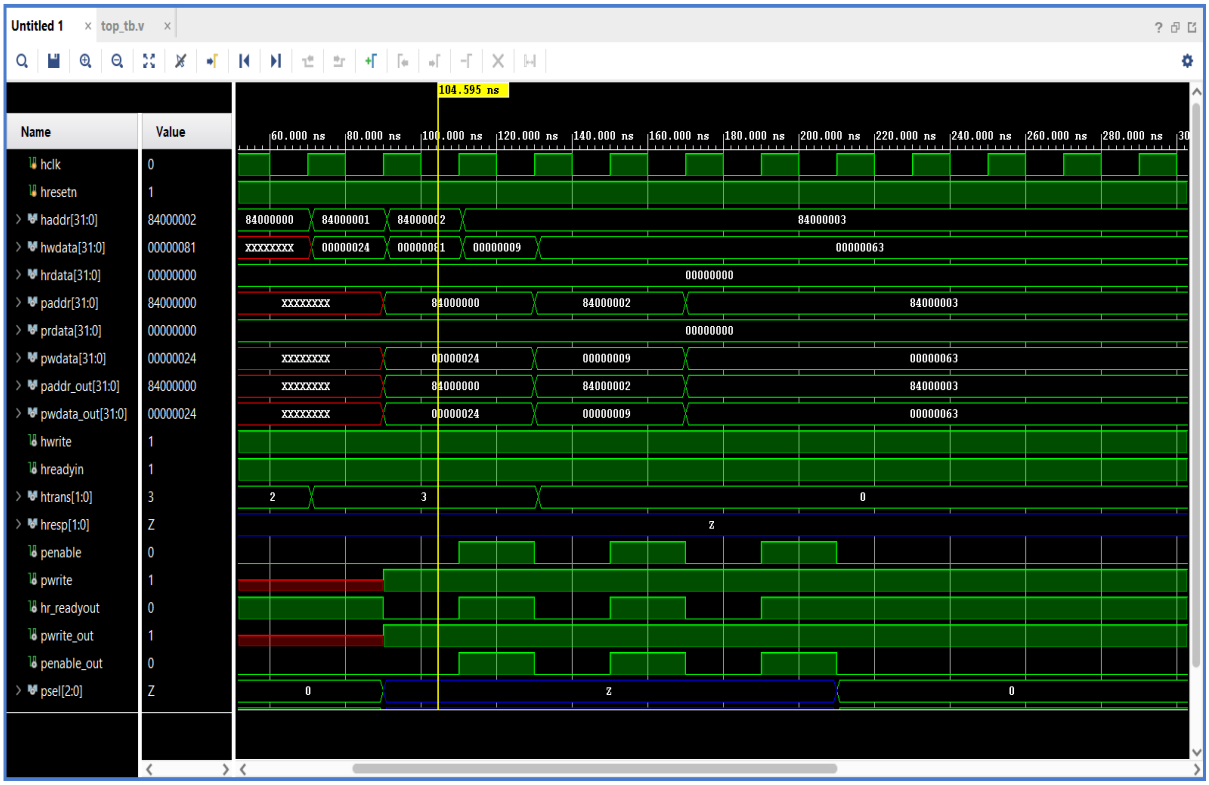
Single Read Operation



Burst Read Operation



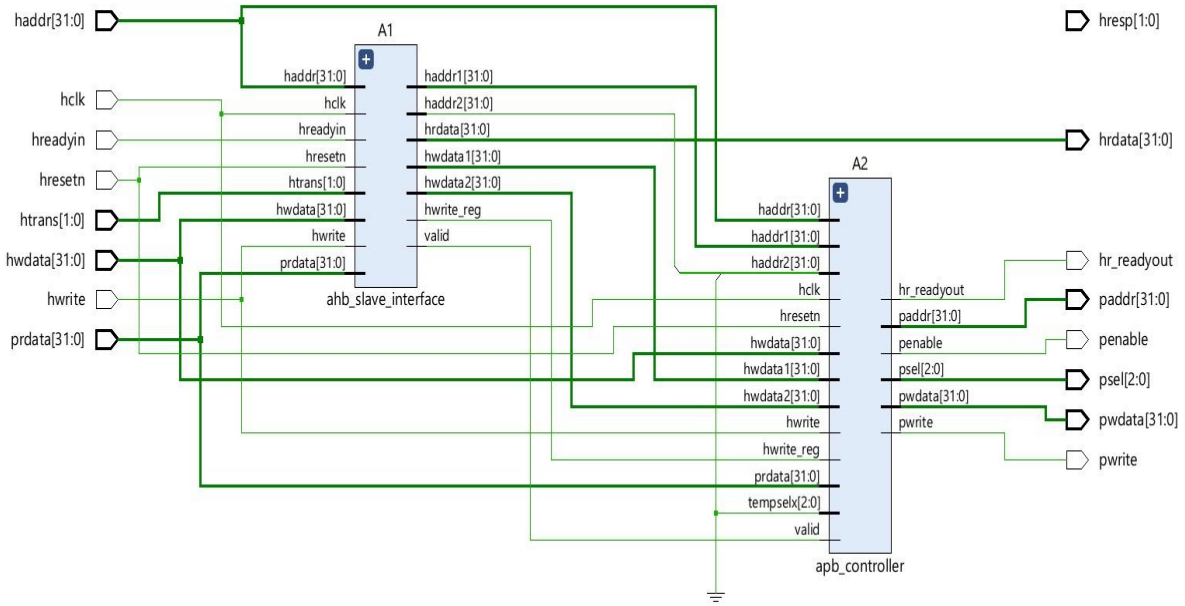
Burst Write Operation



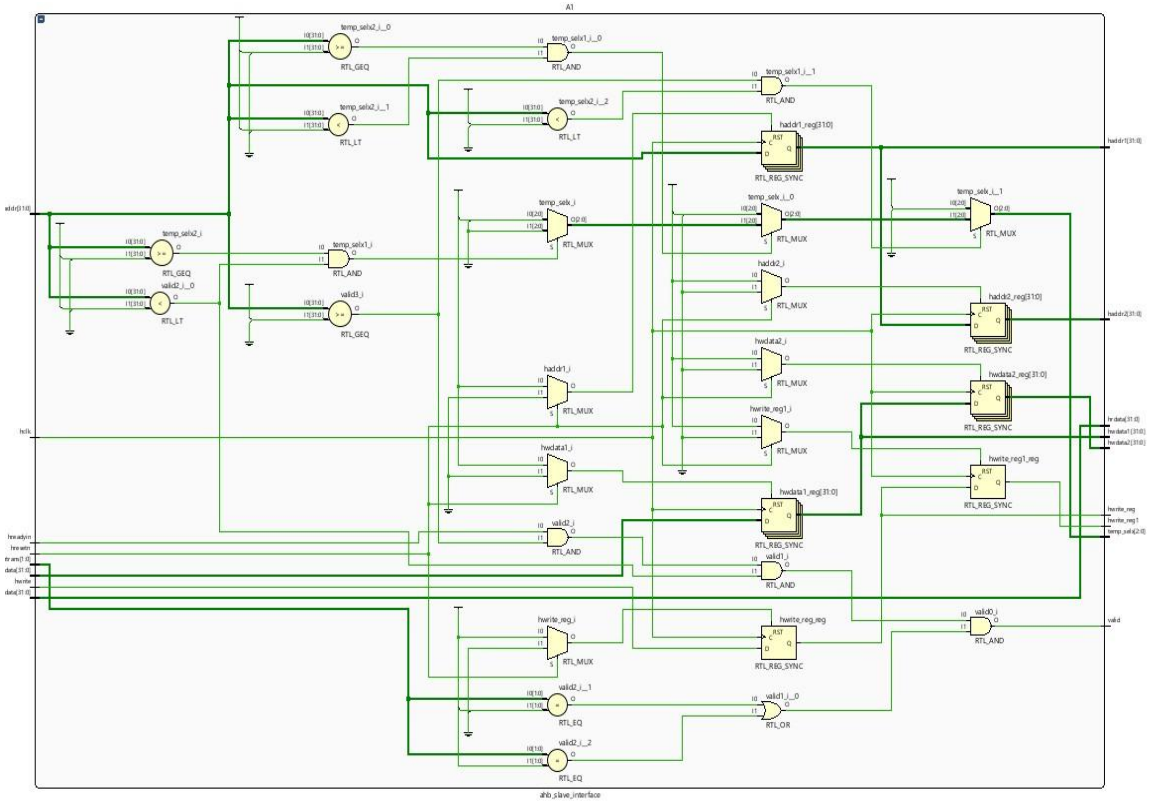
6. Synthesis Results

RTL synthesis generates optimized hardware implementation:

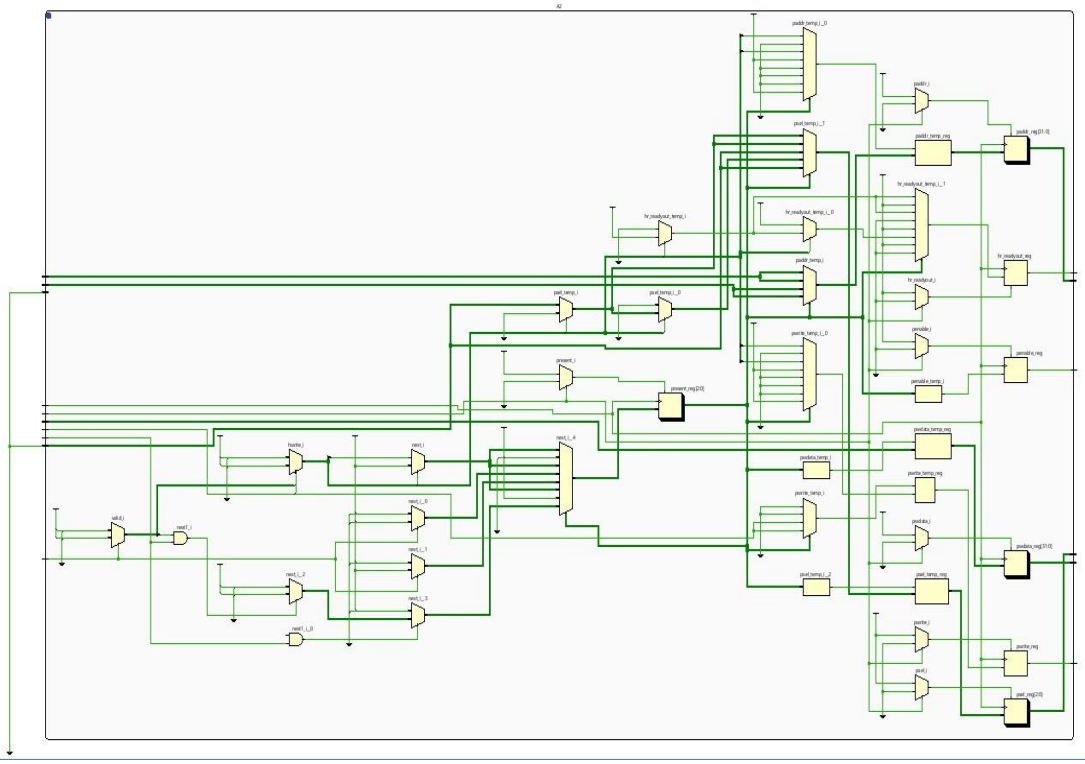
TOP Level RTL Model



AHB Slave Interface



APB Controller



7. Conclusion

The AHB to APB bridge implementation successfully demonstrates the conversion of high-performance bus transfers to low-power peripheral bus operations. The design achieves all specified objectives while maintaining protocol compliance and synthesis compatibility. The modular architecture and comprehensive verification provide a solid foundation for future enhancements and system integration.

The project effectively bridges the performance gap between high-speed system components and low-power peripherals, enabling efficient SoC architectures that optimize both performance and power consumption. The successful synthesis and simulation results validate the design's practical applicability in real-world embedded systems.

.