DESIGN SPEC DOCUMENT

ECE-593: Fundamentals of Pre-Silicon Validation
Maseeh College of Engineering and Computer Science
Winter, 2025

Project Name: Design and Verification of AHB2APB Bridge using UVM

Members: Raghavendra Davarapalli
Maithreyi Venkatesan
Neil Austin Tauro
Suraj Vijay Shetty

Date:02/26/2025

Project Name	Design and Verification of AHB2APB Bridge using UVM
Location	WCC-310
Start Date	01/27/2025
Estimated Finish Date	03/01/2025
Completed Date	03/04/2025

Prepared by: Team Number	
Prepared for: Prof. Venkatesh Patil	
Team Member Name	Email
Raghavendra Davarapalli	ragha@pdx.edu
Maithreyi Venkatesan	maithven@pdx.edu
Neil Austin Tauro	ntauro@pdx.edu
Suraj Vijay Shetty	surajvs@pdx.edu

Design Features:

The AHB2APB bridge implements the following key features:

- Provides a seamless interface between the high-performance AHB bus and the low-power APB peripheral bus, handling the necessary protocol conversions and signal timing requirements between the two interfaces.
- Supports zero-wait state single transfers for write operations and single-wait state transfers for read operations, optimizing the bridge performance while maintaining protocol compliance.
- Implements separate read and write data paths with configurable data widths (32-bit standard), enabling efficient data transfer between the AHB and APB domains.
- Features an integrated address decoder that generates peripheral select signals (psel) based on the AHB address, simplifying peripheral interfacing and improving design modularity.
- Includes built-in wait state generation and handling capabilities to manage timing differences between the AHB and APB domains when required for specific peripherals.
- Supports different transfer types on the AHB interface, including NONSEQUENTIAL, SEQUENTIAL, IDLE, and BUSY transfers, ensuring proper transaction flow control and efficient bus utilization.
- Incorporates a response mechanism that indicates the status of the AHB transfer, distinguishing between successful, erroneous, retry, or split responses, ensuring robust error handling and system reliability.

Project Description:

The AHB2APB bridge project encompasses both robust design implementation and comprehensive verification methodologies. The key features include:

- Implements a fully compliant AMBA AHB-to-APB bridge interface that handles protocol conversion between the high-performance AHB bus and the low-power APB peripheral bus, supporting efficient system-on-chip integration.
- Features an advanced verification environment developed using SystemVerilog and Universal Verification Methodology (UVM), enabling thorough validation of the bridge functionality through sophisticated test scenarios and coverage metrics.
- Incorporates dedicated Verification IP (VIP) components that facilitate comprehensive testing of all bridge operations, including address decoding, data transfers, and protocol compliance verification.
- The design supports optimized transfer modes with zero-wait state write operations and single-wait state read operations, ensuring efficient data movement while maintaining protocol integrity.
- Implements a sophisticated address decoder for generating peripheral select signals (psel), along with separate read and write data paths supporting 32-bit width transfers.
- The verification environment leverages QuestaSim for simulation and waveform analysis, ensuring thorough validation of all design features and corner cases.
- Project development follows a structured approach, starting from basic functionality verification and progressing to complex test scenarios, ensuring robust operation across various use cases.

Design Signals:

Design Input Signals:

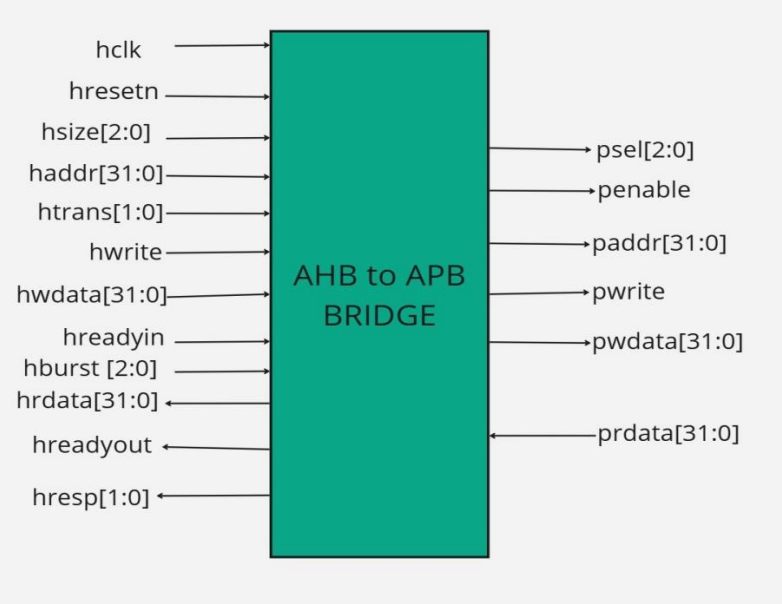
1. hclk - Clock signal for the AHB interface.
2. hresetn -Activelow reset signal for the AHB interface.
3. hsize[2:0] -Size of the transfer on the AHB interface.
4. hburst[2:0] - Specifies the type of burst operation. (Future Scope)
5. haddr[31:0] -32-bit address for the AHB transfer.
6. htrans[1:0] -Transfer type on the AHB interface (NONSEQUENTIAL, SEQUENTIAL, IDLE or BUSY).
7. hwrite - When HIGH this signal indicates a write transfer, and when LOW, a read transfer.
8. hwdata[31:0] -32-bit write data on the AHB interface.
9. hreadyin - Ready signal indicating the availability of the AHB interface.
10. prdata[31:0] - Read data on the APB interface.

Design Output Signals:

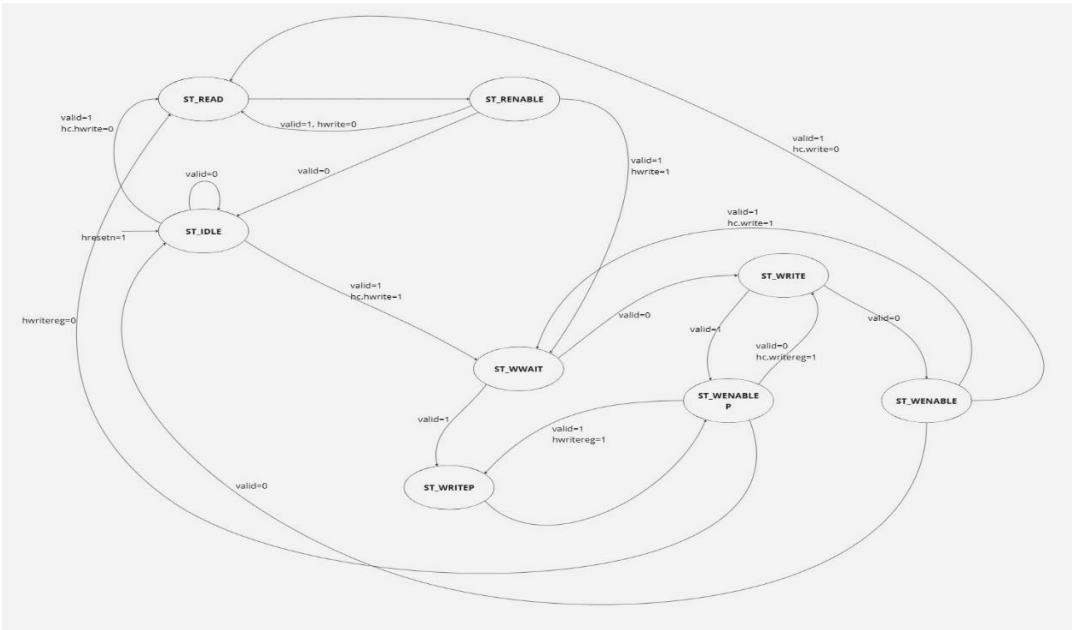
1. hrdata[31:0]-32-bit read data from the AHB interface.
2. hreadyout - Ready signal indicating the readiness of the AHB interface.
3. hresp[1:0] - Response indicating the status of the AHB transfer.
4. psel[2:0] -The signal indicates that the slave device is selected, and that a data transfer is required. It has the same timing as the peripheral address bus. It becomes HIGH at the same time as PADDR, but will be set LOW at the end of the transfer.
5. penable - Enable signal for the APB interface.
6. paddr[31:0]-32-bit address for the APB transfer.
7. pwrite -This signal indicates a write to a peripheral when HIGH, and a read from a peripheral when LOW. It has the same timing as the peripheral address bus.
8. pwdata[31:0]–32-bit write data on the APB interface.

Block Diagram

AHB2APB Bridge:



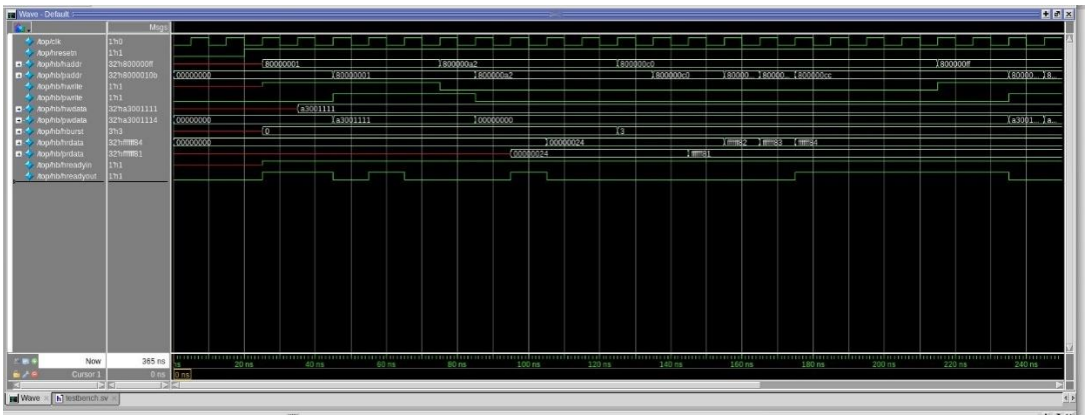
State Machine for AHB2APB Interface:



Compilation and Simulation:

To verify the fundamental accuracy of the selected code, QuestaSim is used to compile the design files, and the process completed successfully without any errors or issues. Additionally, a basic testbench simulation was created to analyze the waveform produced by the design, and a screenshot of the results is attached below.

Simulation:



Transcript:

```
#
# [TEST]: Going through tests!
# 1st transaction!!!!
# [GEN]: single_write_hw_nonseq_okay at 75
# [SCO]: Scoreboard check...
# [SCO]: Input Address: 3fc83
# [SCO]: Input Write Data: 00000006
# [SCO]: Data Stored: 00000006
#
# 2nd transaction!!!!
# [GEN]: single_read_hw_nonseq_okay at 80
# [SCO]: Scoreboard read
# [SCO]: Temp address = 67131
# [SCO]: Read data from DUT 29dbd537
# [SCO]: Data from TB memory 29dbd537
#
# 3rd transaction!!!!
# [GEN]: single_write_nonseq_error at 85
# [SCO]: Scoreboard check...
# [SCO]: Input Address: 082ea
# [SCO]: Input Write Data: 0000000b
# [SCO]: Data Stored: 0000000b
#
# 4th transaction!!!!
# [GEN]: single_read_byte_okay at 90
# [SCO]: Scoreboard read
# [SCO]: Temp address = ba6bb
# [SCO]: Read data from DUT 29dbd537
# [SCO]: Data from TB memory 29dbd537
#
# 5th transaction!!!!
# [GEN]: single_write_hw_seq_okay at 95
# [SCO]: Scoreboard check...
# [SCO]: Input Address: cfb37
# [SCO]: Input Write Data: 00000004
# [SCO]: Data Stored: 00000004
#
# 6th transaction!!!!
# [GEN]: single_read_wrd_okay task in generator at 100
# [SCO]: Scoreboard read
# [SCO]: Temp address = a6499
# [SCO]: Read data from DUT 29dbd537
# [SCO]: Data from TB memory 29dbd537
#
# 7th transaction!!!!
# [GEN]: single_write_byte_seq_error at 105
# [SCO]: Scoreboard check...
# [SCO]: Input Address: 51bb9
# [SCO]: Input Write Data: 0000000d
# [SCO]: Data Stored: 0000000d
#
```

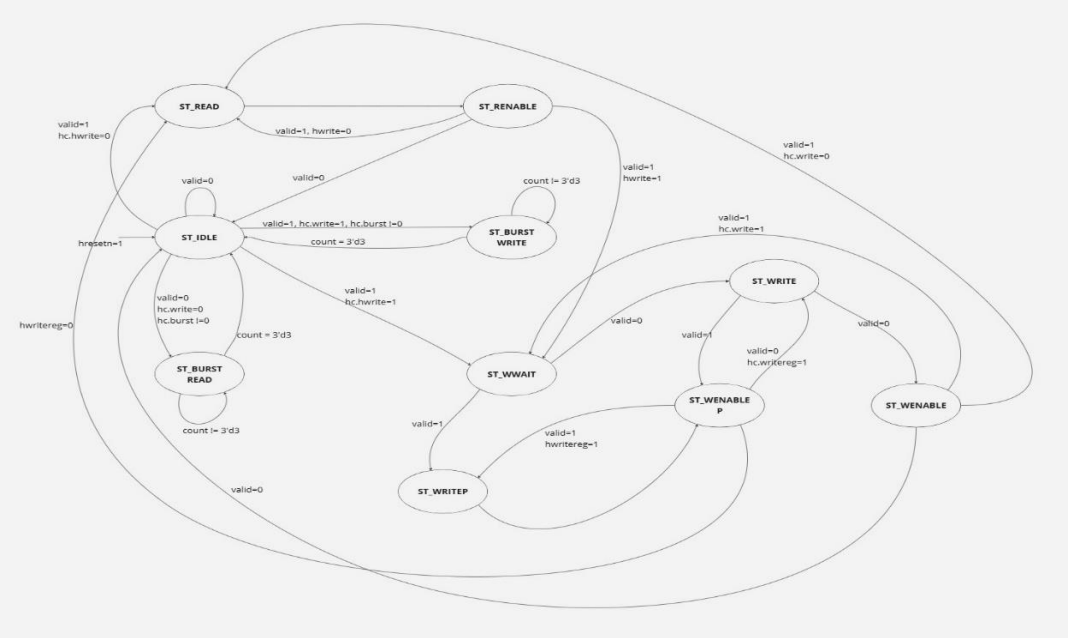
Steps to run the code:

1) Download the ece593w25_14_AHB2APB_Bridge_M5.zip file and unzip the folder and open it.
2) Copy the location of M5/CLASS/ file.
3) Open QuestaSim and in the terminal enter the command “cd {path_name for CLASS/}”
4) Now enter the command “**do run_normalMode.do**”, this should produce the output transcript along with the coverage reports. The coverage files names are code_coverage.txt, functional_coverage.txt and assertions_coverage.txt

Future Scope:

The support for burst mode operation is considered for future scope, where the bridge will handle burst transfers on the AHB side while efficiently translating them into individual APB transactions. This enhancement will help maintain high data throughput for burst transactions, optimizing performance for systems requiring sequential data movement. Future implementations will include support for various burst transfer types such as INCR, WRAP4, INCR4, WRAP8, INCR8, WRAP16, ensuring improved efficiency and throughput in handling consecutive memory accesses.

Modified State Machine for AHB2APB Interface (Including Burst Operations):



References/Citations:

1. https://github.com/prajwalgekkouga/AHB-to-APB-Bridge
2. https://chatgpt.com/
3. https://developer.arm.com/documentation/ihi0033/latest