Department of Computer & Software Engineering

National University of Science and Technology, College of E&ME, Rawalpindi

Final Project Report

Computer System Architecture

Project Name:

AMBA Bus Implementation in Verilog

Submitted to
Dr. Muhammad Yasin

Student Name	Registration #	Degree
Muhammad Sumair	415339	CE-44-A
Usama	424024	CE-44-A
Muskan Fatima	414875	CE-44-B

Submission Date
05 June 2024

CONTENTS

Abstract:

In the realm of system-on-chip (SoC) design, communication protocols play a crucial role in achieving high performance, low power consumption, and testability. The AMBA-APB (Advanced Microcontroller Bus Architecture - Advanced Peripheral Bus) protocol is particularly valuable for SoC designs. In this work, we present a simplified design of an APB controller responsible for managing transactions between a designed master and peripheral devices. The final integrated design, which includes peripheral devices connected to the APB controller, is implemented on an FPGA device. Our approach ensures efficient communication within the SoC, meeting the essential requirements of performance and power efficiency.

AMBA

The AMBA bus is a widely adopted and versatile interconnect standard in the field of system-on-chip (SoC) design. Developed by ARM (Advanced RISC Machines), the AMBA architecture provides a framework for connecting various functional blocks within an SoC, enabling efficient communication and data transfer. Let's delve into the key aspects of the AMBA bus:

AMBA Components:

AHB (Advanced High-Performance Bus)*:* The AHB is a high-bandwidth, pipelined bus that connects high-performance modules such as processors, memory controllers, and DMA (Direct Memory Access) controllers. It supports multiple masters and slaves, allowing concurrent data transfers.

APB (Advanced Peripheral Bus): The APB is a simpler, low-power bus designed for connecting peripheral devices (e.g., UARTs, timers, GPIO controllers). It operates at a lower clock frequency compared to the AHB.

AXI (Advanced extensible Interface): Although not part of the project, the AXI protocol has become prevalent in modern designs. It provides high bandwidth, multiple transaction channels, and advanced features like burst transfers and out-of-order execution.

Bus Transactions:

The AMBA bus uses a transaction-based model. Each transaction involves a master (initiator) requesting data from or writing data to a slave (target). Read transactions involve the master sending an address to the slave, which responds with the requested data. Write transactions follow a similar process, with the master providing both the address and data to be written.

Bus Protocols:

The AHB and APB buses define specific protocols for addressing, data transfer, and control signals.AHB supports split transactions (address phase followed by data phase) and burst transfers.

APB uses a single-phase protocol with separate read and write channels.

Benefits:

Modularity: The AMBA bus promotes modularity by allowing easy integration of IP (Intellectual Property) cores from different vendors.

Scalability: Designers can scale their systems by adding or removing modules without significant changes to the overall architecture.

Standardization: The AMBA standardization ensures compatibility across different SoCs and accelerates development.

AHB (Advanced High-Performance Bus)

The Advanced High-Performance Bus (AHB) is a pivotal component of the AMBA architecture, designed to handle high-bandwidth and high-performance operations within an SoC. Here are some key aspects of the AHB:

High Bandwidth:

The AHB supports high-speed data transfer and pipelined operations, allowing multiple instructions to overlap in execution. This makes it suitable for connecting high-performance modules that require rapid data throughput, such as processors, memory controllers, and DMA controllers.

Multiple Masters and Slaves:

The AHB architecture supports multiple bus masters and slaves. This capability allows different modules to initiate transactions and access shared resources concurrently, enhancing the overall efficiency and performance of the system.

Single-Clock Edge Operation:

AHB transactions are synchronized to a single clock edge, which simplifies the design and ensures consistent timing throughout the communication process.

Split and Retry Responses:

To handle bus congestion and ensure smooth operation, the AHB includes mechanisms for split and retry responses. These features allow a slave to signal that it cannot immediately service a request, prompting the master to retry the transaction later.

Special Multiplexer and Decoder:

Avoid errors and help in writing or reading correct data from selected slave. They prevent slave errors that can be caused by out of bound address and garbage results. Decoder selects the appropriate slave based on selection line and multiplexer selects the correct data coming from all the slaves.

APB (Advanced Peripheral Bus)

The Advanced Peripheral Bus (APB) is designed for connecting peripheral devices that do not require the high bandwidth and performance characteristics of the AHB. Key characteristics of the APB include:

Simplicity and Low Power:

The APB is a simpler and lower-power bus compared to the AHB. Its design focuses on minimal complexity, making it ideal for peripherals such as UARTs, timers, and GPIO controllers that do not demand high data transfer rates.

Single-Clock Domain:

The APB typically operates within a single-clock domain, which simplifies its integration with other low-speed peripheral components.

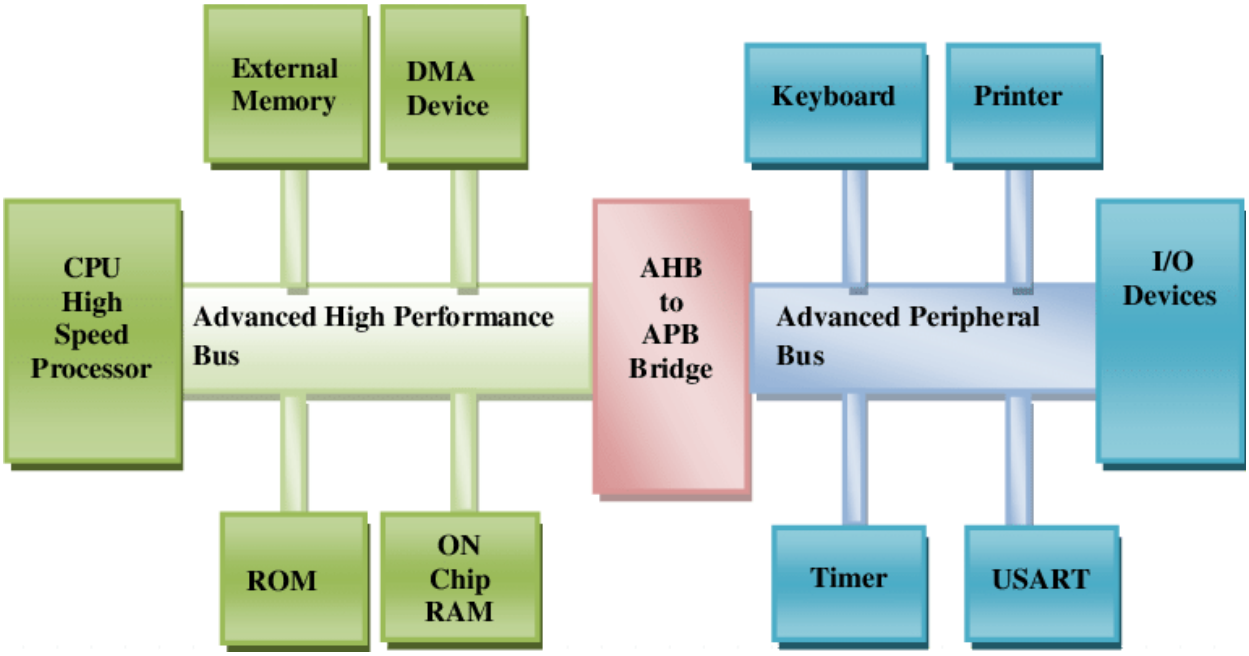
Bridge Interface:

In an AMBA-based SoC, the APB is often connected to the AHB through an AHB-to-APB bridge. This bridge translates high-speed AHB transactions into the simpler APB protocol, enabling efficient communication between high-performance and low-power domains within the SoC.

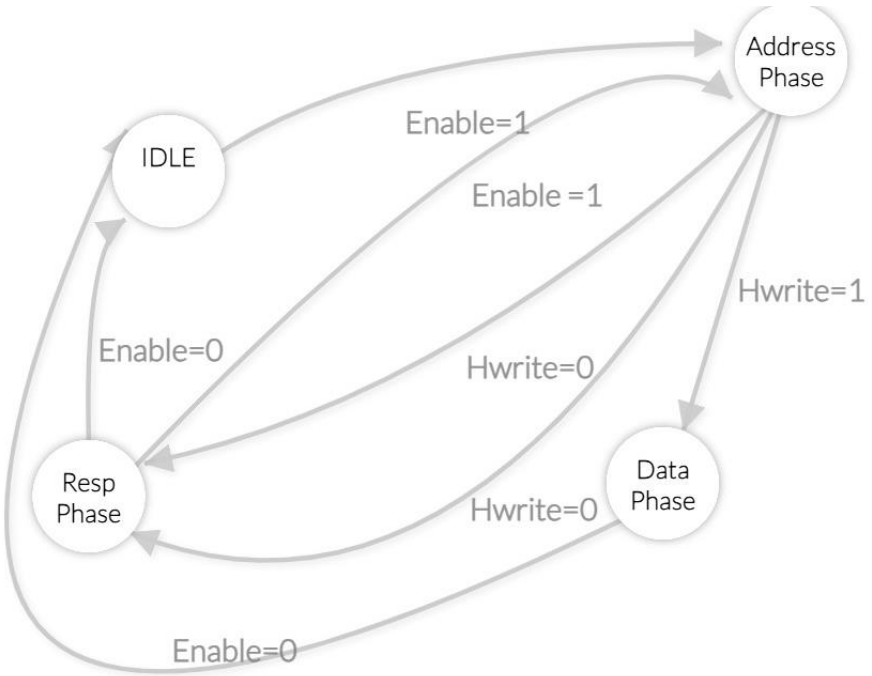
Low Latency for Control Registers:

The APB is particularly well-suited for accessing control registers of peripheral devices. The low-latency access provided by the APB ensures that control operations can be performed efficiently, without the need for the high bandwidth of the AHB.

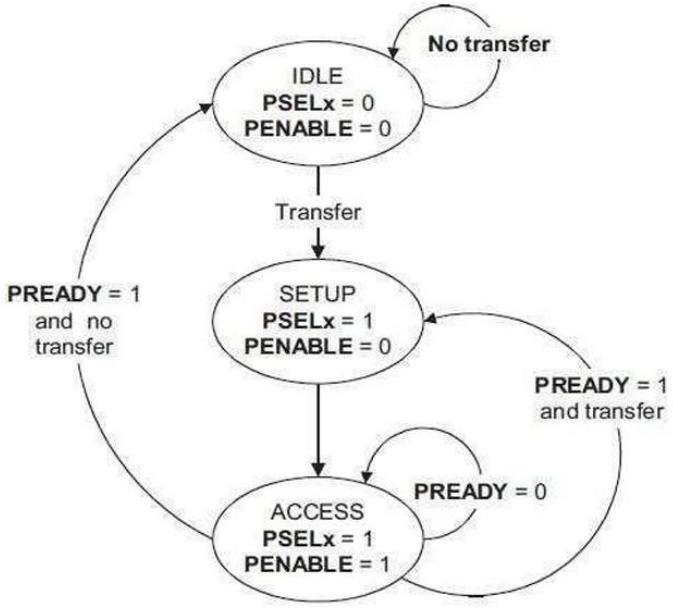
Bird view Image and Block Diagram:



FSM of AHB protocol:



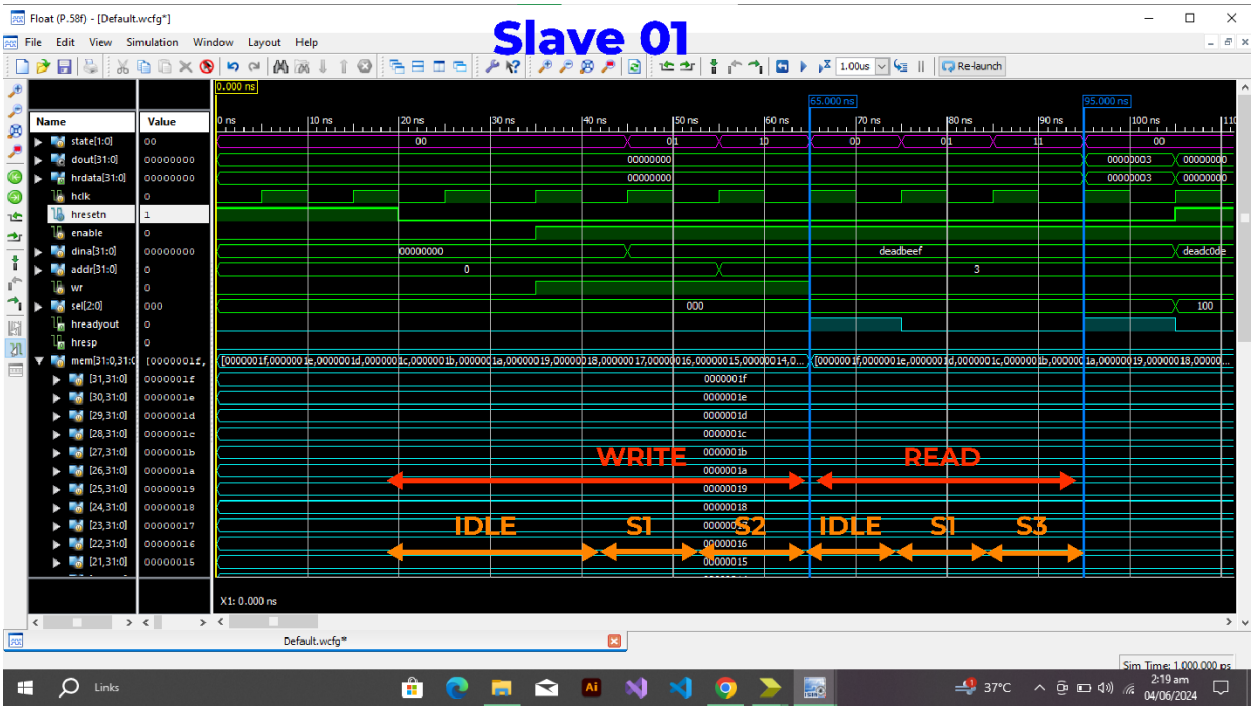
FSM of APB protocol:

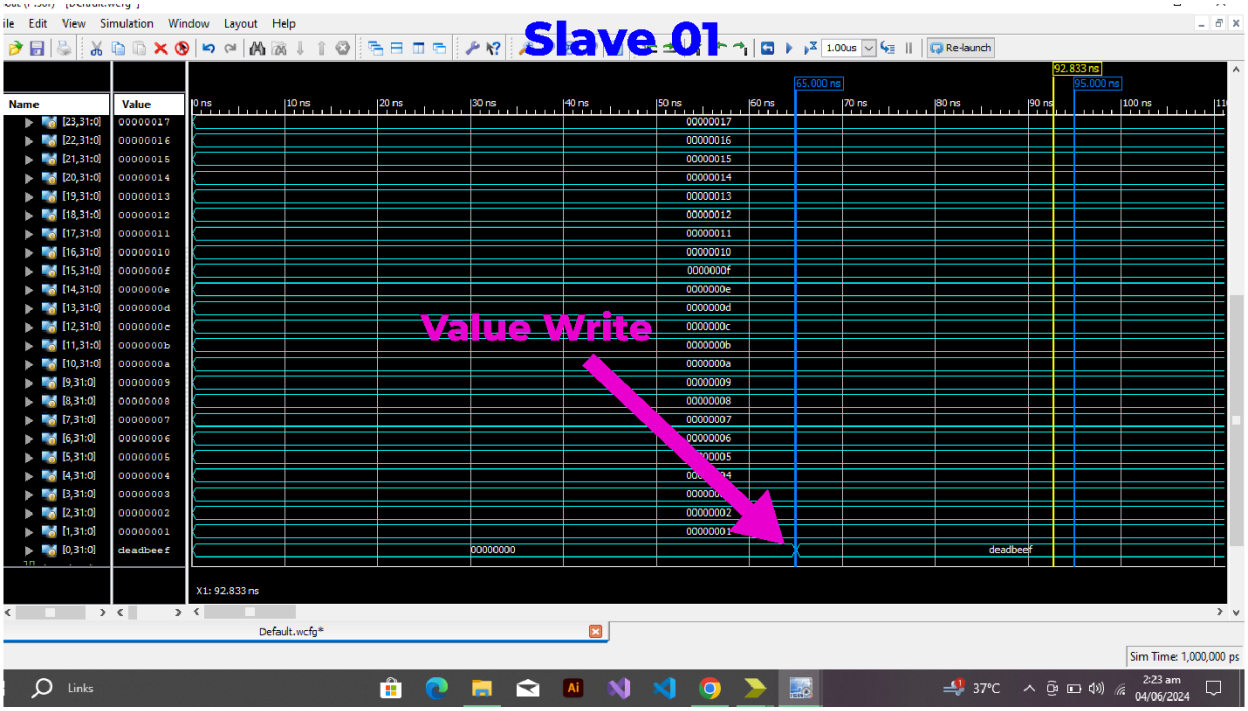


Screenshots with Indentation:

Write state: Enable= 1'b1, wr=1b1, slave_sel= 3'b000, hwdata= 32'hDEADBEEF, haddr=32'b0

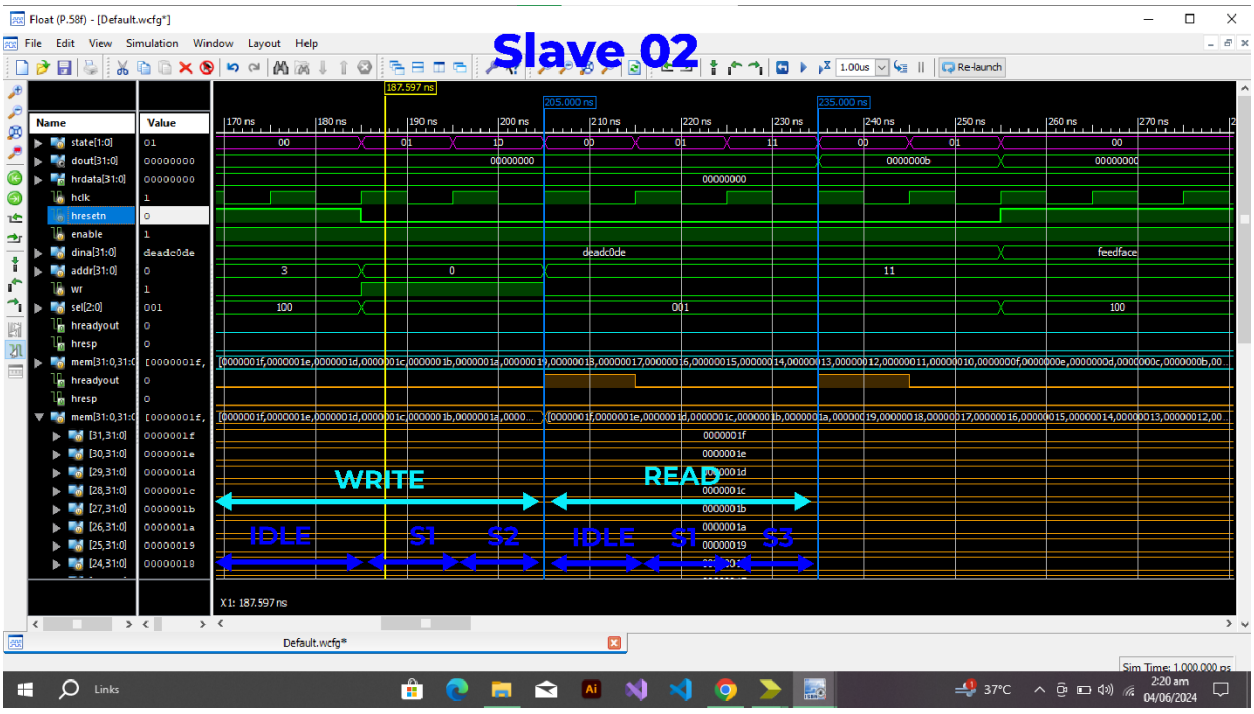
Read state: Enable= 1'b1, wr=1b0, slave_sel= 3'b000, haddr=32'd3

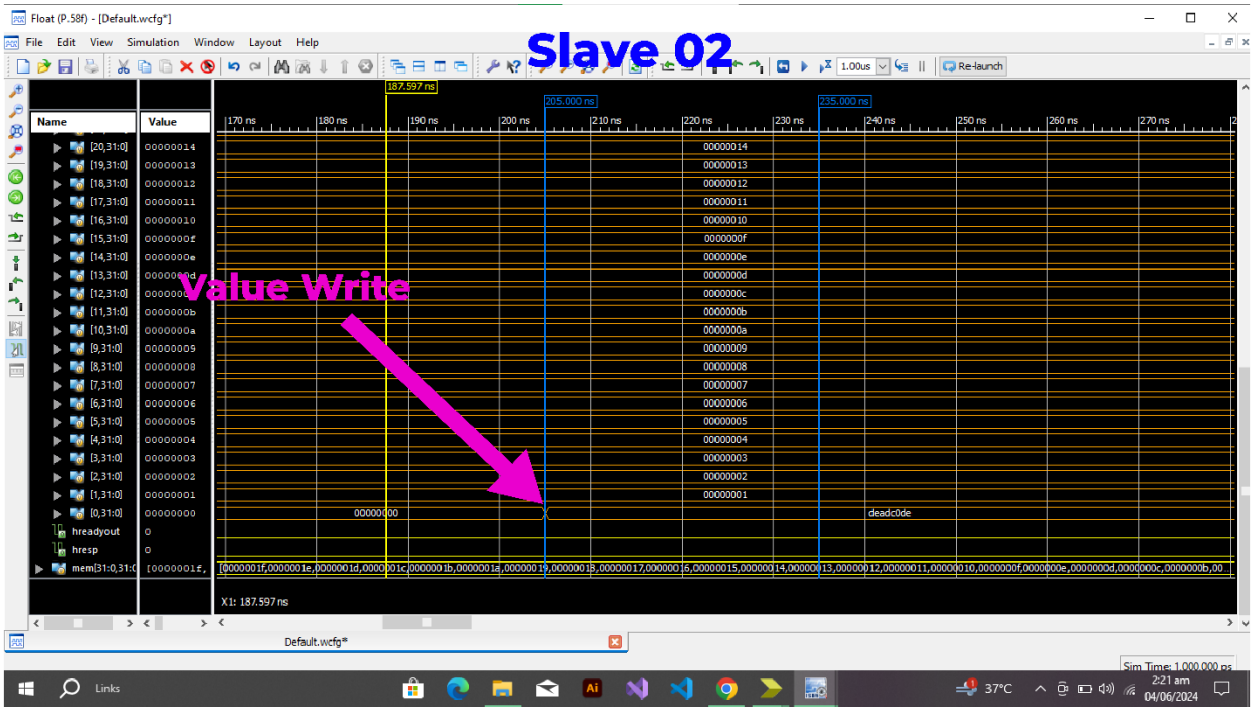




Write state: Enable= 1’b1, wr=1b1, slave_sel= 3’b001, hwdata= 32’hDEADC0DE, haddr=32’b0

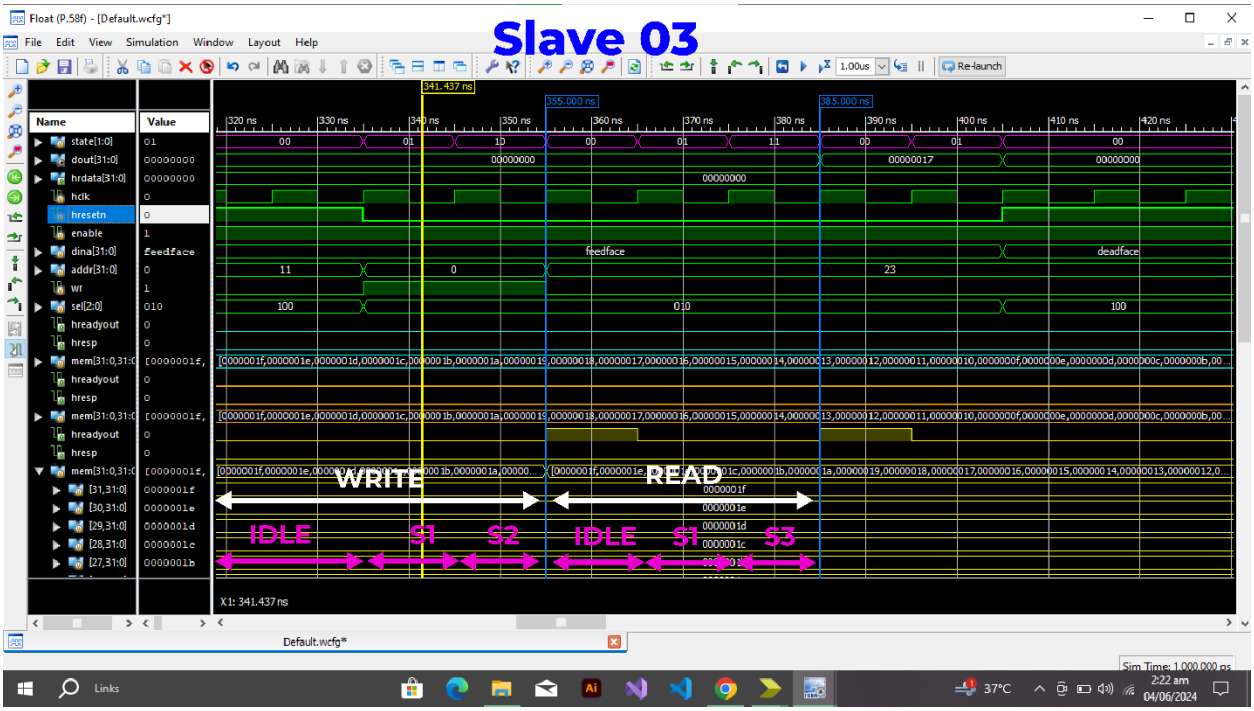
Read state: Enable= 1’b1, wr=1b0, slave_sel= 3’b001 , haddr=32’d11

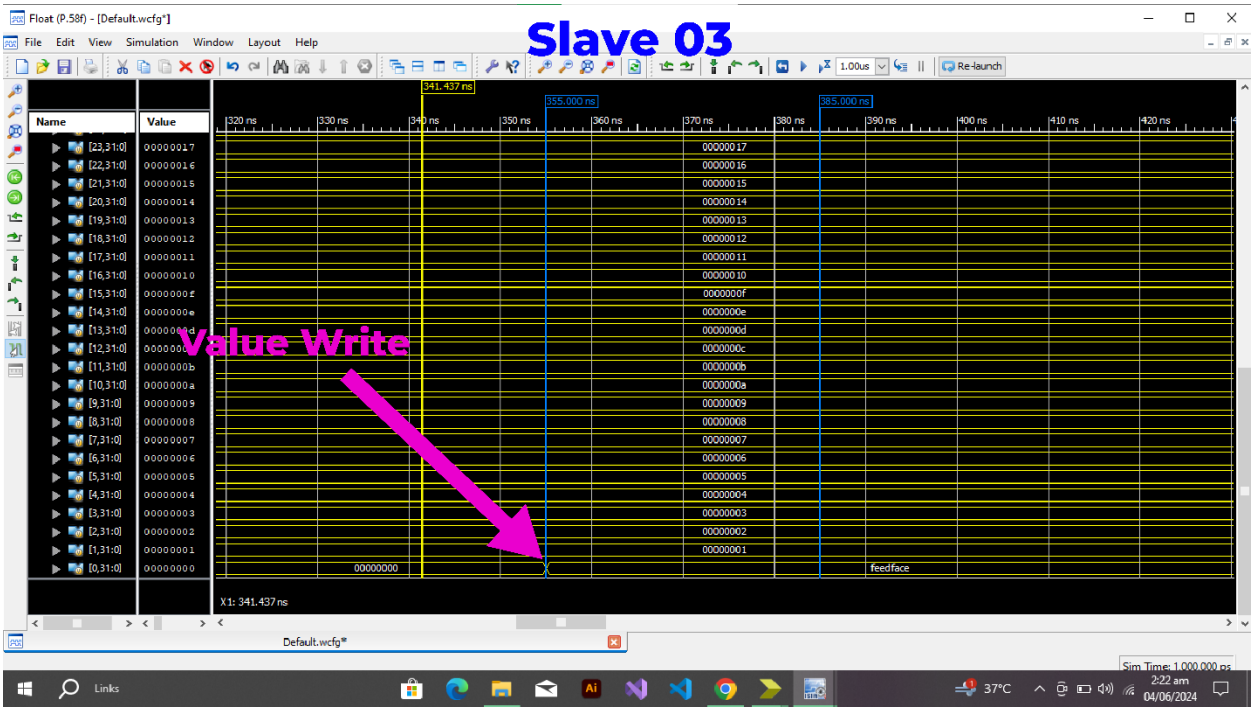




Write state: Enable= 1’b1, wr=1b1, slave_sel= 3’b010, hwdata= 32’hFEADFACE, haddr=32’b0

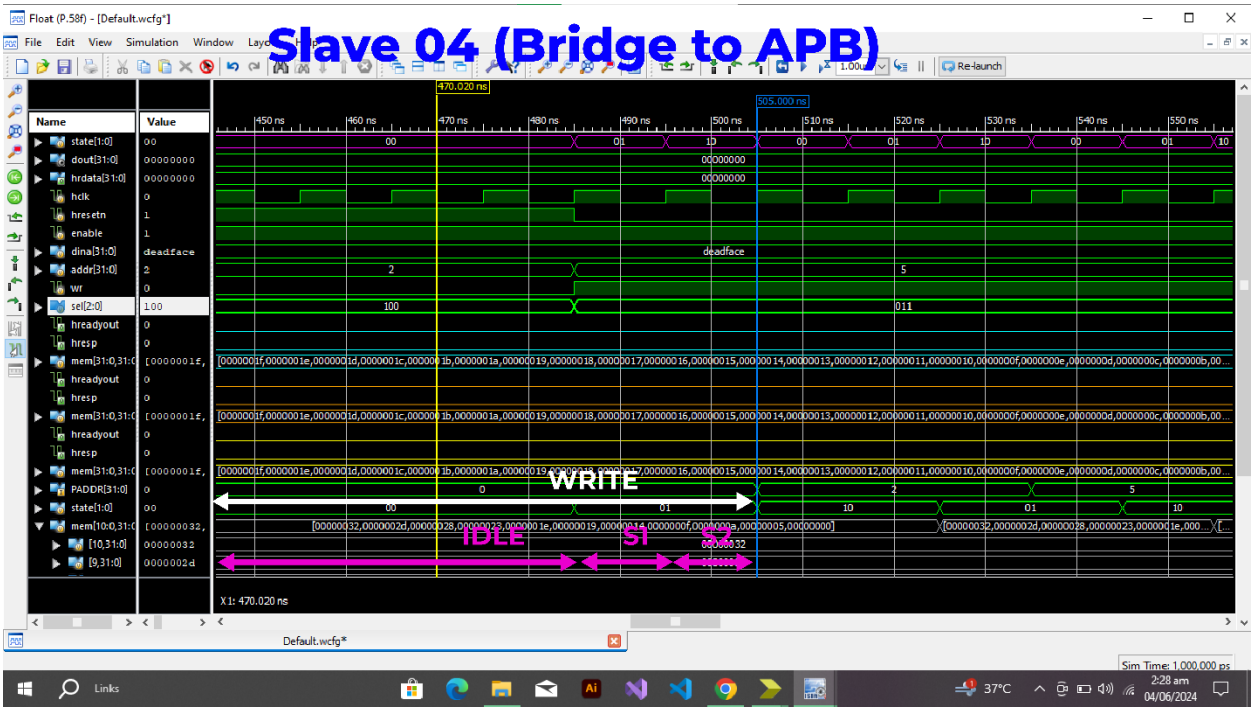
Read state: Enable= 1’b1, wr=1b0, slave_sel= 3’b010 , haddr=32’d23

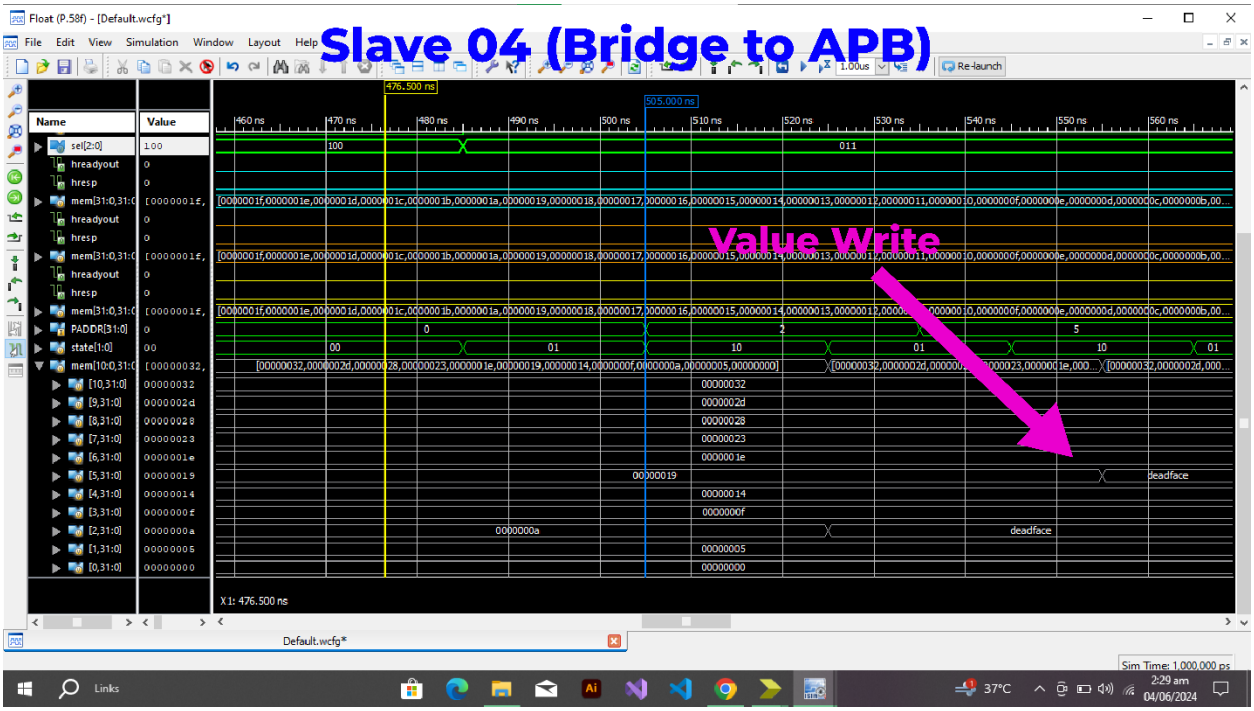




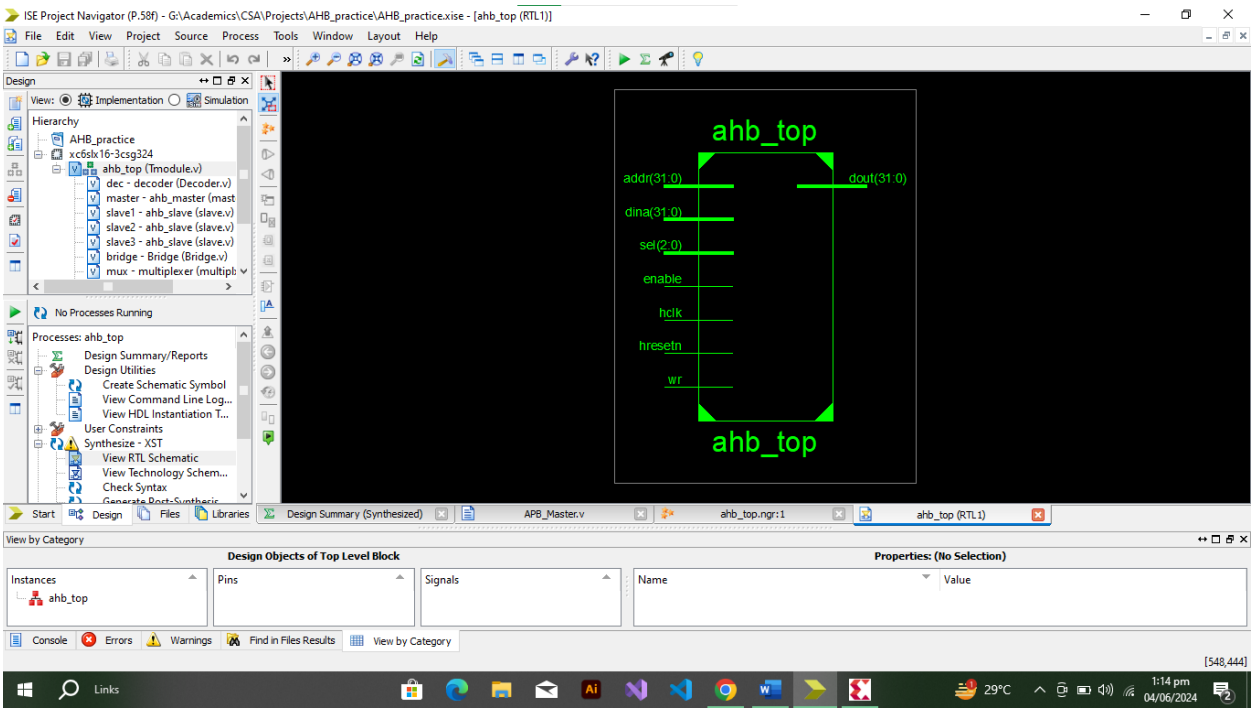
Write state: Enable= 1’b1, wr=1b1, slave_sel= 3’b011, hwdata= 32’hDEADFACE, haddr=32’b0

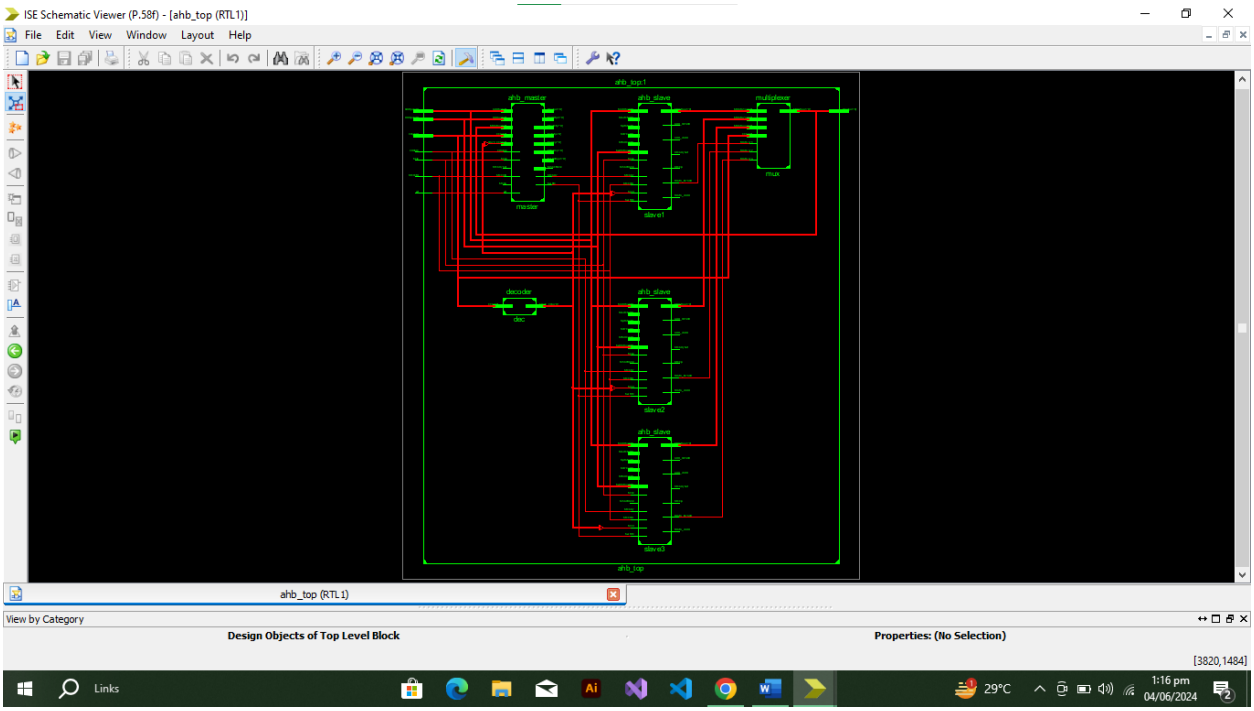
Read state: Enable= 1’b1, wr=1b0, slave_sel= 3’b011 , haddr=32’d5



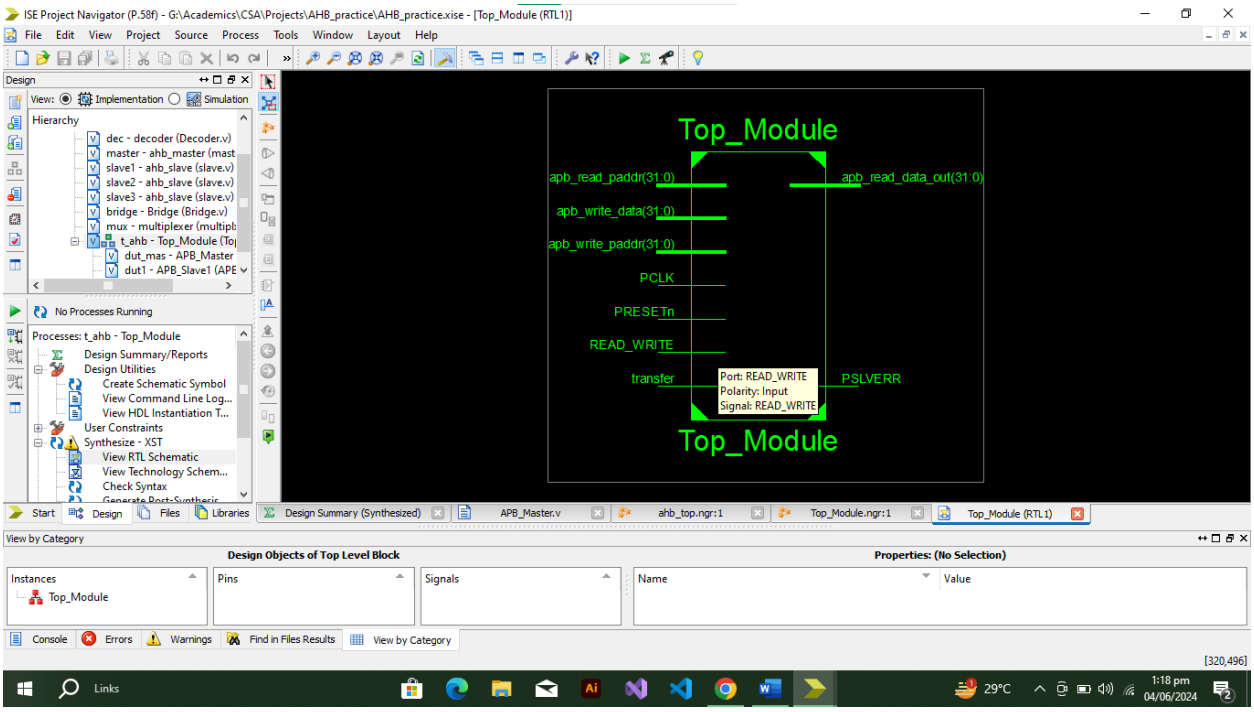

Slave 04 (Bridge to APB)
Value Write
Float (P.58f) - [Default.wcfg*]
File Edit View Simulation Window Layout Help
Re-launch
Name
Value
476.500 ns
505.000 ns
X1: 476.500 ns
Default.wcfg*
Links
37°C
2:29 am
04/06/2024

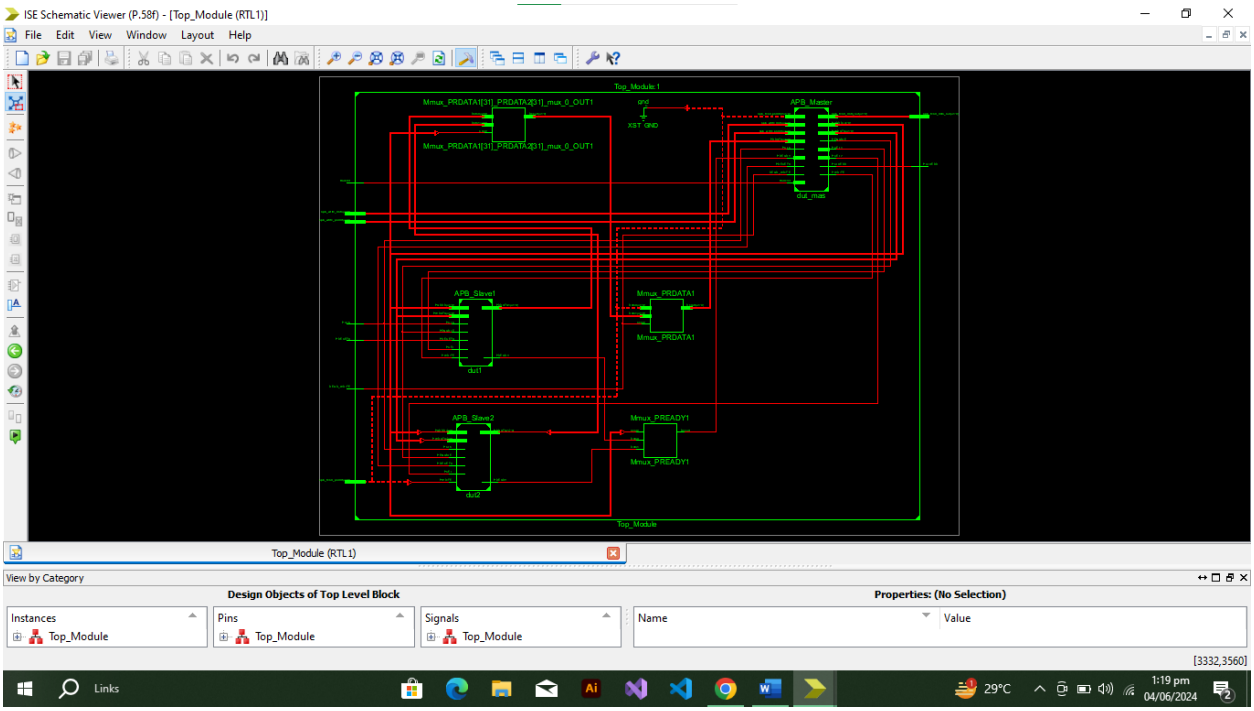
RTL Schematics of AHB Protocol:



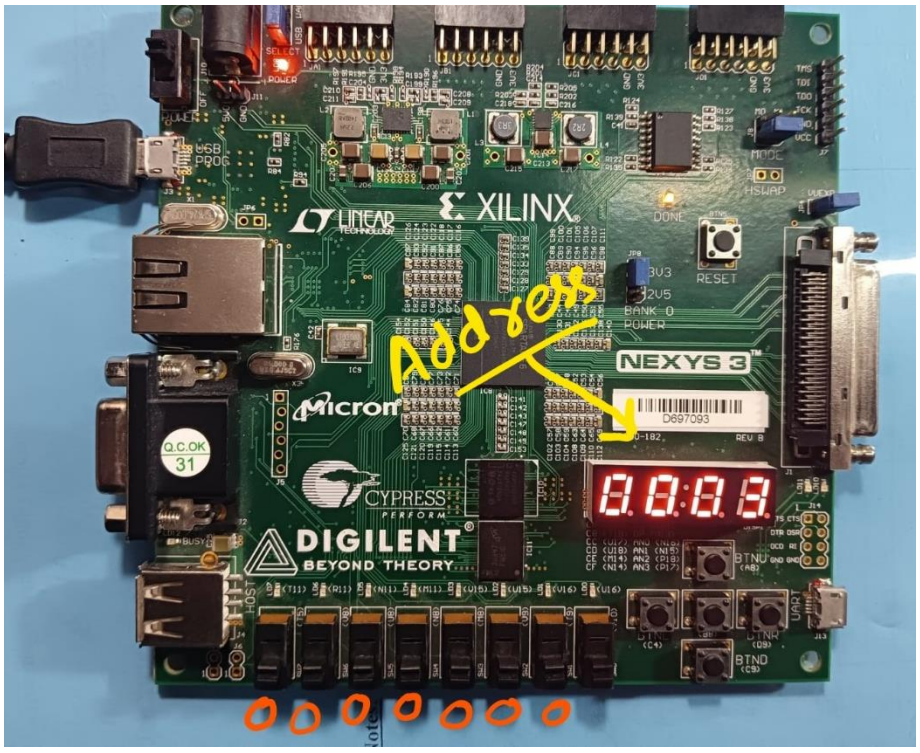


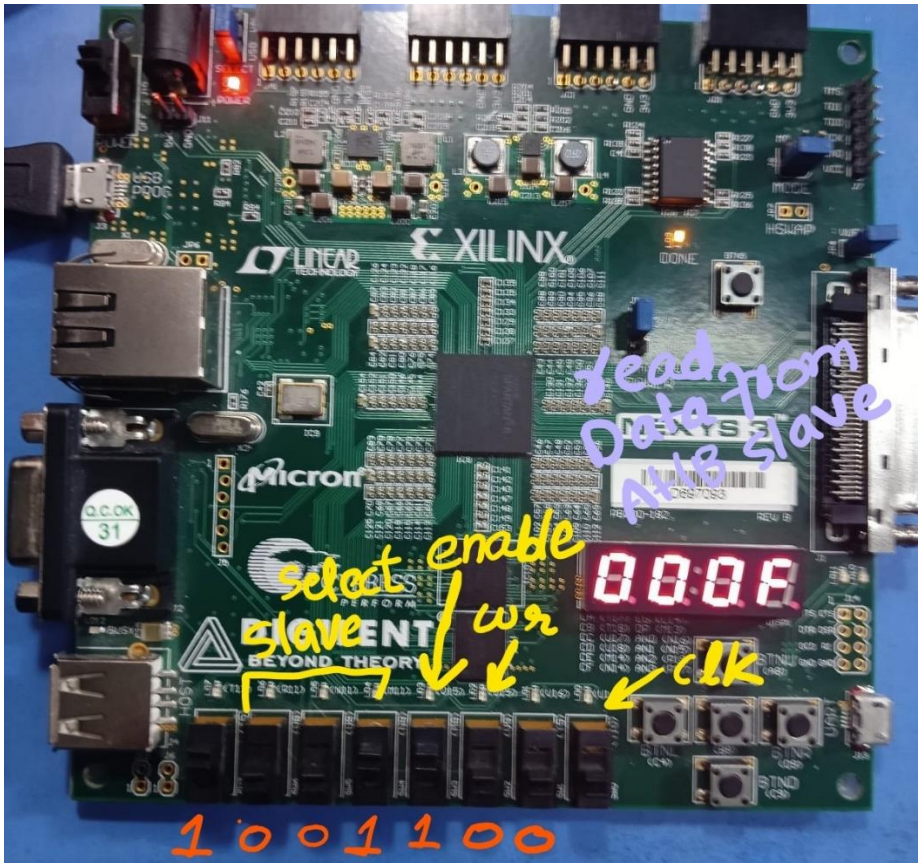
RTL Schematics of APB Protocol:





FPGA Implementation






write data
0000
0000110


read data from APB slave
NEXYS 3
0011
1011110

FPGA Implementation Video Link:

https://drive.google.com/file/d/1q5XFXzAUW5mBnZ7pVNW4mgyltmhztVHr/view?usp=sharing/

References:

We hereby consider the use of information from these sites, and we humbly accept the use of artificial intelligence tools like Gemini and ChatGPT for better understanding:

https://github.com/sure-trust/VLSI-Project-AXI-to-APB-Bridge/blob/main/bridge.v
https://github.com/SamarSarda/AMBA-APB-I2C-Project
https://digilent.com/reference/programmable-logic/nexys-3/start
https://www.allaboutcircuits.com/technical-articles/introduction-to-the-advanced-microcontroller-bus-architecture/