# AHB to APB

# Bridge Design

## Project Report


By- Sanskar Verma

College-VIT Vellore

Instructor-Sathyapriya

# Table of Contents

# Introduction

In modern embedded systems, efficient communication between high-performance processors and low-power peripherals is essential. The Advanced Microcontroller Bus Architecture (AMBA) facilitates this through standardized bus protocols, specifically the Advanced High-performance Bus (AHB) and Advanced Peripheral Bus (APB). The AHB is designed for high-speed, high-frequency modules, while the APB is optimized for low-power peripheral functions. This project focuses on designing an AHB to APB bridge, enabling seamless communication between these two buses. The bridge translates high-speed AHB transactions into the simpler, lower-power APB protocol, ensuring efficient data transfer and synchronization in complex embedded systems.

# AMBA Overview

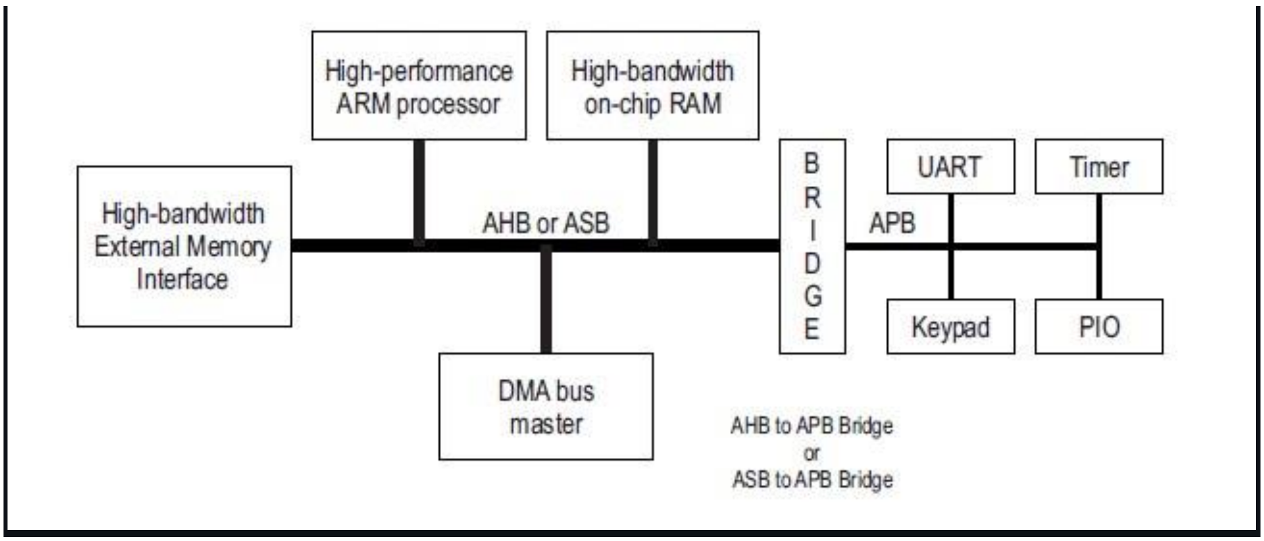
The Advanced Microcontroller Bus Architecture (AMBA) specification defines an on-chip communications standard for designing high-performance embedded microcontrollers. Three distinct buses are defined within the AMBA specification:

1. Advanced High-performance Bus (AHB)
2. Advanced System Bus (ASB)
3. Advanced Peripheral Bus (APB)

Advanced High-performance Bus (AHB) The AMBA AHB is for high-performance, high clock frequency system modules. The AHB acts as the high-performance system backbone bus. AHB supports the efficient connection of processors, on-chip memories and off-chip external memory interfaces with low-power peripheral macrocell functions. AHB is also specified to ensure ease of use in an efficient design flow using synthesis and automated test techniques.

Advanced System Bus (ASB) The AMBA ASB is for highperformance system modules. AMBA ASB is an alternative bus system suitable for use where the high-performance features of AHB are not required. ASB also supports the efficient connection of processors, on-chip memories and off-chip external memory interfaces with low-power peripheral macrocell functions.

Advanced Peripheral Bus (APB) The AMBA APB is for low-power peripherals. AMBA APB is optimized for minimal power consumption and reduced interface complexity to support peripheral functions. APB can be used in conjunction with either version of the system bus.



# AHB to APB Bridge Design

The AHB to APB bridge design facilitates communication between high-speed AHB modules and low-power APB peripherals in embedded systems. The bridge translates AHB transactions to APB protocol, handling address decoding, data transfer, and control signal synchronization. It includes components like the AHB and APB interfaces, FIFO buffers, and a state machine to manage states such as IDLE, SETUP, and ACCESS. The design ensures efficient data transfer while maintaining low power consumption, critical for embedded applications. A Verilog implementation and thorough simulation verify the bridge's functionality and performance, making it a vital component in modern SoC designs.

# Detailed Design

Components

- AHB Interface: Handles communication with the AHB.
- APB Interface: Handles communication with the APB.

- FIFO Buffers: Manages data flow between the two buses.
- State Machine: Controls the data transfer and ensures proper sequencing.

## State Machine

The state machine manages the transition between different states during the data transfer process:

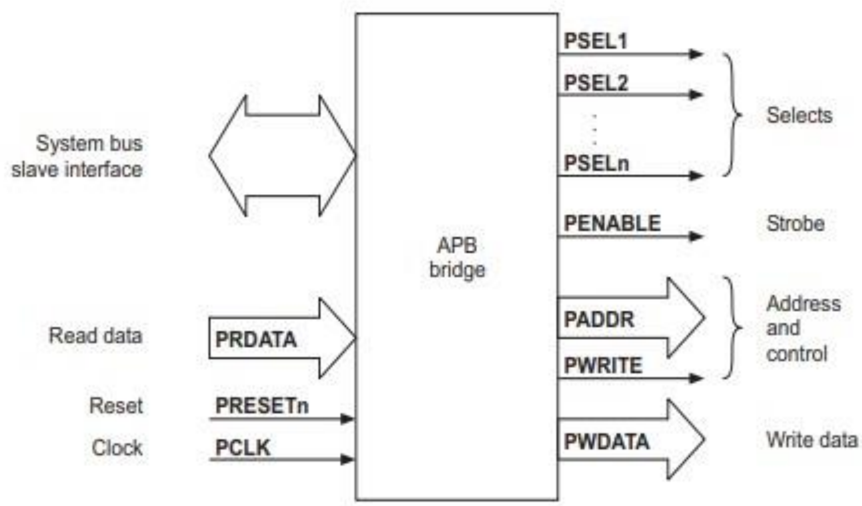
- IDLE: The default state, waiting for an AHB transaction.
- SETUP: Prepares the APB for an upcoming transaction.
- ACCESS: Performs the data transfer between AHB and APB.

To design and simulate a synthesizable AHB to APB bridge interface using Verilog and run single read and single write tests using AHB Master and APB Slave testbenches. The bridge unit converts system bus transfers into APB transfers and performs the following functions:

- Latches the address and holds it valid throughout the transfer.

- Decodes the address and generates a peripheral select, PSELx. Only one select signal can be active during a transfer. Drives the data onto the APB for a write transfer
- Drives the APB data onto the system bus for a read transfer
- Generates a timing strobe, PENABLE, for the transfer Can implement single read and write operations successfully.

The Diagram Below shows the Interface:



# Basic Implementation Tools

HDL Used : Verilog

Simulator Tool Used: ModelSIM

Synthesis Tool Used:  Quartus Prime

Family: Cyclone V

Device: 5CSXFC6D6F31I7ES

# Verilog Implementation

Verilog codes for Bridge with it’s Testbench.

C:/Users/HP/OneDrive/Desktop/project_Bridge/bridge_rtl.v (/bridge_tb/bridge) - Default

```
module bridge_rtl(
input hclk,hresetn,hselapb,hwrite,
input [1:0] htrans,input[31:0] haddr,
input [31:0] hwdata,
input [31:0] prdata,
output reg [31:0] paddr,pwdata,
output reg psel,penable,pwrite,
output reg hresp,hready,
output reg [31:0] hrdata
);

  parameter idle = 3'b000;
  parameter read = 3'b001;
  parameter wwait = 3'b010;
  parameter write = 3'b011;
  parameter write_p = 3'b100;
  parameter wenable_p =3'b101;
  parameter wenable = 3'b110;
  parameter renable = 3'b111;
  reg[31:0] haddr_temp,hwdata_temp;
  reg[2:0] present_state,next_state;
  reg valid,hwrite_temp;

  always@(*)
        begin
          if(hselapb == 1'b1 && (htrans == 2'b10 || htrans == 2'b11))
              valid = 1'b1;
              else
              valid = 1'b0;

              if(hresetn == 1'b0)
                 present_state = idle;
               else
                 present_state = next_state;
     end

     always@(present_state)
```

```
     always@(present_state)
         begin
                 case(present_state)
                         idle:
                             begin
                                psel = 1'b0;
                                penable = 1'b0;
                                 hready = 1'b1;

                                 if(valid == 1'b0)
                                     next_state = idle;
                                     else if(valid == 1'b1 && hwrite ==1'b0)
                                         next_state = read;

                                         else if(valid == 1'b1 && hwrite ==1'b1)
                                         next_state = wwait;
                             end

                 read:
                    begin
                         psel = 1'b1;
                         paddr = haddr;
                         pwrite = 1'b0;
                         penable = 1'b0;
                         hready = 1'b0;

                         next_state = renable;
                 end


                 renable :
                         begin
                                 penable = 1'b1;
                                 hrdata = prdata;
                                 hready = 1'b1;
                                 if(valid == 1'b1 && hwrite ==1'b0)
                                         next_state =read;
```

C:/Users/HP/OneDrive/Desktop/project_Bridge/bridge_rtl.v (/bridge_tb/bridge) - Default

```
                                 if(valid == 1'b1 && hwrite ==1'b0)
                                         next_state =read;
                                         else if(valid == 1'b1 && hwrite ==1'b1)
                                                 next_state =wwait;
                                                 else if(valid == 1'b0)
                                         next_state =idle;
                         end

                 wwait:
                    begin
                         penable = 1'b0;
                         haddr_temp = haddr;
                         hwrite_temp = hwrite;
                         if(valid == 1'b0)
                                 next_state = write;
                                 else if(valid == 1'b1)
                                    next_state = write_p;

                 end

                 write :
                   begin
                         psel = 1'b1;
                         paddr = haddr_temp;
                         pwdata = hwdata;
                         pwrite = 1'b1;
                         penable = 1'b0;
                         hready = 1'b0;

                         if(valid == 1'b0)
                                 next_state = wenable;
                                 else if(valid == 1'b1)
                                         next_state = wenable_p;
                 end

                 write_p :
                         begin
                         psel = 1'b1;
```

C:\Users\HP\OneDrive\Desktop\project_Bridge\bridge_rtl.v (\bridge_tb\bridge) - Default

```
                          pwrite = 1'b1;
                          penable = 1'b0;
                          hready = 1'b0;

                          hwrite_temp = hwrite;

                          next_state = wenable_p;

                end

           wenable :
                begin
                     penable = 1'b1;
                     hready = 1'b1;
                     if(valid == 1'b1 && hwrite == 1'b0)
                          next_state = read;
                          else if(valid == 1'b1 && hwrite == 1'b1)
                            next_state = wwait;
                            else if(valid == 1'b0)
                                  next_state = idle;
                end


          wenable_p :
                begin
                     penable = 1'b1;
                     hready = 1'b1;
                     if(valid == 1'b0 && hwrite == 1'b0)
                          next_state = write;
                          else if(valid == 1'b1 && hwrite == 1'b1)
                            next_state = write_p;
                            else if(hwrite == 1'b0)
                                  next_state = read;
                end
                endcase

        end
endmodule
```

Transcript | Wave | Objects | Processes | Library | Project | sim

Ln: 1 Col: 0 | Project : AHB_APB | Now: 1,400 ps Delta: 2

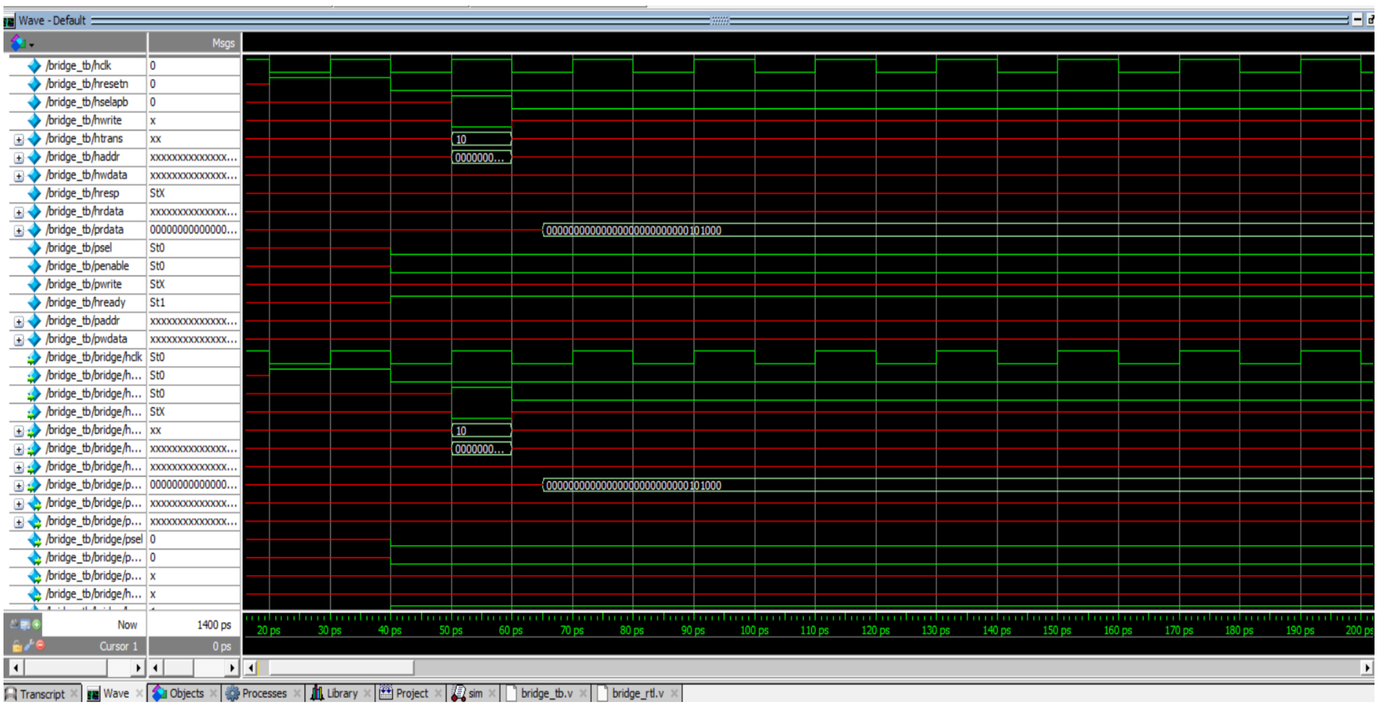
C:/Users/HP/OneDrive/Desktop/project_Bridge/bridge_tb.v (/bridge_tb) - Default

```
module bridge_tb(




                    );

 ////////////////////////ahb slave
reg hclk,hresetn,hselapb,hwrite;
reg [1:0] htrans;
reg [31:0] haddr,hwdata;
wire hresp;
wire [31:0] hrdata;

////////////////////////////////apb outpute signal

reg [31:0] prdata;
wire psel,penable, pwrite,hready;
wire [31:0] paddr,pwdata;

bridge_rtl bridge(hclk,hresetn,hselapb,hwrite,htrans,
                  haddr,hwdata,
                  prdata,
                  paddr,pwdata,
                  psel,penable,pwrite,
                  hresp,hready,
                  hrdata);
  initial
        begin
            hclk = 0;
  end

always #10 hclk = ~hclk;

task reset_bridge();
        begin
```

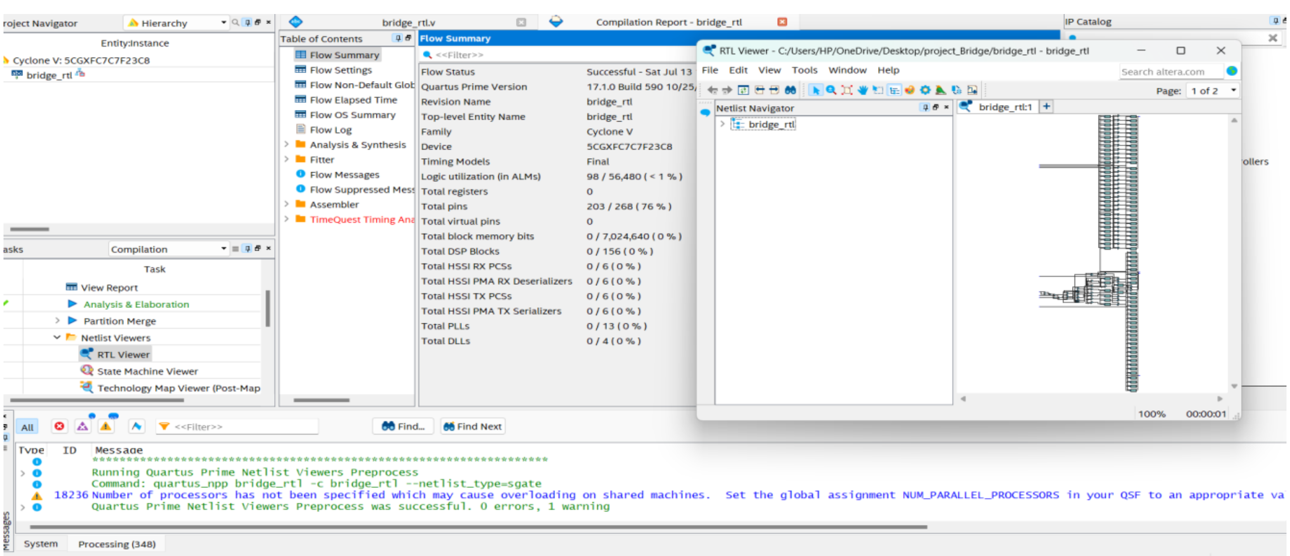
```

always #10 hclk = ~hclk;

task reset_bridge();
        begin
            @(negedge hclk)
                hresetn = 1;
            @(negedge hclk)
                hresetn = 0;
        end
        endtask



        initial
        begin

        reset_bridge ();
        #10 ;
        hwrite = 1'b0;
        hselapb = 1'b1;
        htrans = 2'b10;
        haddr =32;
        #10 ;
        hwrite = 1'bx;
        hselapb = 1'b0;
        htrans = 2'bxx;
        haddr = 32'hxxxx_xxxx;
        #5 ;
        prdata =40;
        end
endmodule
```

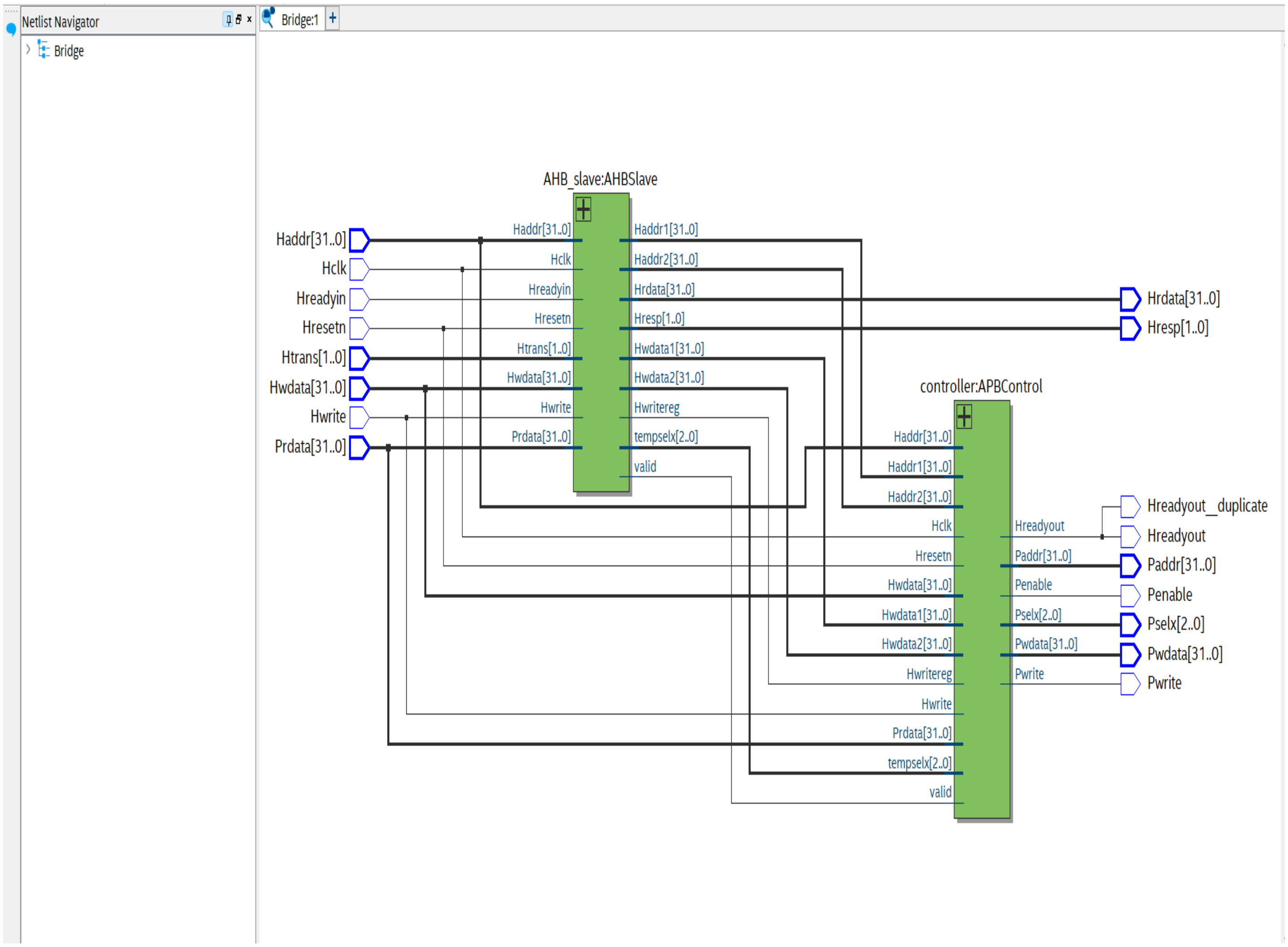
# Simulation and Verification

## Simulation Result :-



## Synthesis Result :-




Netlist Navigator
Bridge
Bridge:1
AHB_slave:AHBSlave
controller:APBControl
Haddr[31..0]
Hclk
Hreadyin
Hresetn
Htrans[1..0]
Hwdata[31..0]
Hwrite
Prdata[31..0]
Haddr1[31..0]
Haddr2[31..0]
Hrdata[31..0]
Hresp[1..0]
Hwdata1[31..0]
Hwdata2[31..0]
Hwritereg
tempselx[2..0]
valid
Hreadyout
Paddr[31..0]
Penable
Pselx[2..0]
Pwdata[31..0]
Pwrite
Hreadyout_duplicate

# Conclusion

During my internship, I focused on designing and simulating an AHB to APB bridge interface, a crucial component in embedded systems for facilitating communication between high-speed AHB modules and low-power APB peripherals. The project involved multiple phases, each contributing to a comprehensive understanding of hardware design and verification.

I started by understanding the requirements and specifications of the AHB and APB protocols, and the functionality needed for the bridge to translate transactions between these two buses. Using Verilog, I designed the bridge to include key components such as the AHB interface, APB interface, a state machine for managing data transfers, and an address decoder for generating peripheral select signals (PSEL).

To verify the functionality of the design, I developed detailed testbenches for both the AHB master and APB slave. These testbenches simulated single read and write operations, ensuring that the bridge could accurately latch addresses, drive data onto the APB for write transfers, retrieve data from the APB for read transfers, and generate the necessary timing strobe (PENABLE).

The project utilized ModelSIM for simulation, allowing me to thoroughly test and debug the design, and Quartus Prime for synthesis, targeting the Cyclone V family of FPGAs. This combination of tools ensured that the design not only met theoretical specifications but also practical implementation criteria.

Through this project, I gained hands-on experience in hardware description languages, simulation and synthesis tools, and the complete design flow from concept to implementation. The successful completion of this project demonstrated the functionality and efficiency of the AHB to APB bridge, highlighting its importance in system-on-chip (SoC) architectures and enhancing my skills in digital design and verification.