AHB TO APB BRIDGE

(MAVEN SILICON DESIGN INTERNSHIP PROJECT)

SUBMITTED BY: AAKRITI

INTRODUCTION

AMBA

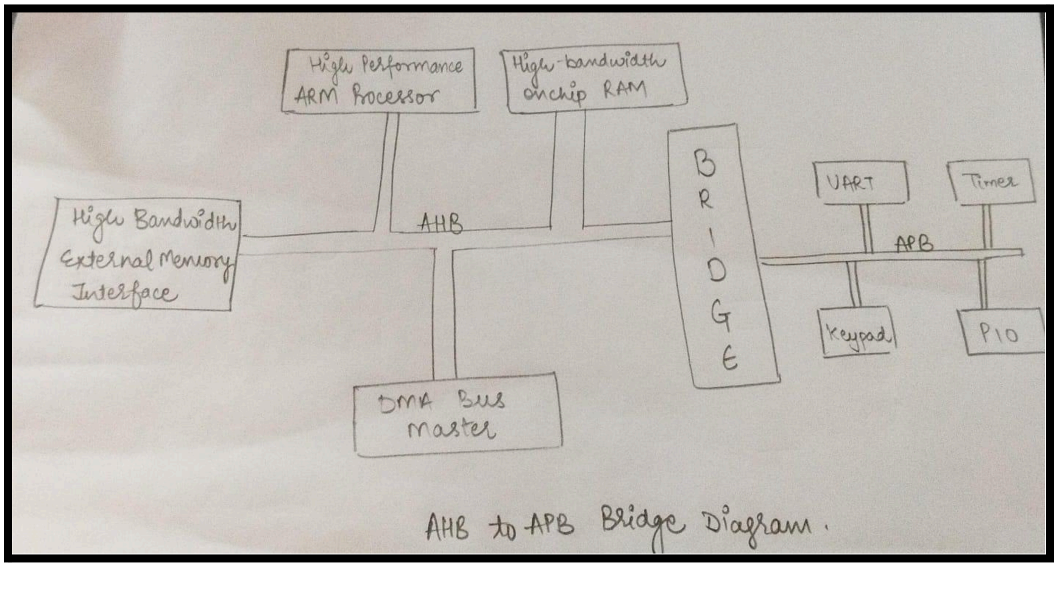
ARM developed the *Advanced Microcontroller Bus Architecture* (AMBA) as an open standard for connecting various blocks or components within a System-on-chip (SoC) design. Given that a typical SoC comprises both high-performance and low-performance components, AMBA was designed to facilitate efficient communication between different peripherals. The AMBA bus architecture includes three primary types of buses, each serving distinct purposes:

1. **Advanced High-Performance Bus (AHB):**
    - The AHB is designed for high-performance, high-bandwidth communication.
    - It supports multiple bus masters, providing efficient access to shared resources.
    - Features include burst transfers, pipelining, and split transactions, which enhance data throughput and system performance.
    - AHB is commonly used for connecting high-speed components such as processors, DMA controllers, and high-speed memory.

2. **Advanced Peripheral Bus (APB):**
    - The APB is optimized for low-power and low-bandwidth communication.

- It is designed to connect peripherals that do not require the high performance of the AHB.
- APB operates with a simpler interface and lower power consumption, making it suitable for peripherals like UARTs, timers, and GPIOs.
- The APB ensures a simple, cost-effective connection for these slower peripherals while maintaining system efficiency.

3. **Advanced System Bus (ASB):**
    - The ASB provides a high-performance, multi-master, and pipelined interface.
    - It is suitable for connecting high-speed system components and supports burst transfers.
    - ASB allows for multiple bus masters, similar to AHB, but with different arbitration schemes.
    - It is used in scenarios where a balance between performance and complexity is required, bridging the gap between AHB and APB.


High Performance ARM Processor
High-bandwidth onchip RAM
High Bandwidth External Memory Interface
AHB
BRIDGE
UART
Timer
APB
Keypad
PIO
DMA Bus master

AHB to APB Bridge Diagram.

BASIC TERMINOLOGY:

Bus cycle

A bus cycle is a basic unit of one bus clock period and for AMBA AHB or APB protocol descriptions are defined from rising-edge to rising-edge transitions.

Bus transfer

An AMBA ASB or AHB bus transfer is a read-or-write operation of a data object, which may take one or more bus cycles. The bus transfer is terminated by a completion response from the addressed slave. An AMBA APB bus transfer is a read-or-write operation of a data object, which always requires two bus cycles.

Burst operation

A burst operation is defined as one or more data transactions, initiated by a bus master, which have a consistent width of transaction to an incremental region of address space. The increment step per transaction is determined by the width of the transfer (byte, halfword, word). No burst operation is supported on the APB.

AMBA AHB Signals

Name	Source	Description
HCLK	Clock source	This clock times all bus transfers. All signal timings are related to the rising edge of HCLK.
HRESETn	Reset controller	The bus reset signal is active LOW and is used to reset the system and the bus.This is the only active LOW signal.
HADDR[31:0]	Master	The 32-bit system address bus.
HTRANS[1:0]	Master	Indicates the type of the current transfer, which can be NONSEQUENTIAL, SEQUENTIAL, IDLE or BUSY.
HWRITE	Master	When HIGH this signal indicates a write transfer and when LOW a read transfer.
HSIZE[2:0]	Master	Indicates the size of the transfer, which is typically byte (8-bit), halfword (16-bit) or word (32-bit). The protocol allows for larger transfer sizes up to a maximum of 1024 bits.
HBURST[2:0]	Master	Indicates if the transfer forms part of a burst. Four, eight, and sixteen-bit bursts are supported and the burst may be either incrementing or wrapping.
HWDATA[31:0]	Master	The write data bus is used to transfer data from the master to the bus slaves during write operations. A minimum data bus width of 32 bits is recommended.
HSELx	Decoder	Each AHB slave has its slave select signal and this signal indicates that the current transfer is intended for the selected slave. This signal is simply a combinatorial decode of the address bus.

HRDATA[31:0]	Slave	The read data bus is used to transfer data from bus slaves to the bus master during read operations.
HREADY	Slave	When HIGH the HREADY signal indicates that a transfer has finished on the bus. This signal may be driven LOW to extend a transfer.Slaves on the bus require HREADY as both an input and an output signal.
HRESP[1:0]	Slave	The transfer response provides additional information on the status of a transfer. Four different responses are provided, OKAY, ERROR, RETRY and SPLIT.

AMBA APB Signals

Name	Source	Description
PCLK	Clock Source	The rising edge of PCLK is used to time all transfers on the APB.
PRESETn	Reset Controller	The APB bus reset signal is active LOW and this signal will normally be connected directly to the system bus reset signal.
PADDR[31:0]	Master	This is the APB address bus, which may be up to 32-bits wide and is driven by the peripheral bus bridge unit.
PSELx	Decoder	A signal from the secondary decoder, within the peripheral bus bridge unit, to each peripheral bus slave x. This signal indicates that the slave device is selected and a data transfer is required. There is a PSELx signal for each bus slave.

PENABLE	Master	This strobe signal is used to time all accesses on the peripheral bus. The enable signal is used to indicate the second cycle of an APB transfer. The rising edge of PENABLE occurs in the middle of the APB transfer.
PWRITE	Master	When HIGH this signal indicates an APB write access and when LOW read access.
PRDATA[31:0]	Slave	The read data bus is driven by the selected slave during read cycles (when PWRITE is LOW). The read data bus can be up to 32-bits wide.
PWDATA[31:0]	Master	The write data bus is driven by the peripheral bus bridge unit during write cycles (when PWRITE is HIGH). The write data bus can be up to 32-bits wide.

DESIGN MODULES:

AHB Master:

- The AHB Master module initiates communication and generates requests to be sent over the AHB bus. It is responsible for issuing commands, addressing, and data transfers. This module controls the data flow by sending signals to the AHB Slave, which includes read and write operations.

AHB Slave:

- The AHB Slave module receives signals from the AHB Master. It acts as an intermediary, interpreting the commands sent by the master and preparing the necessary data transactions. This module ensures that the commands and data are correctly formatted and ready to be processed by the bridge module.

Bridge Module:

- The Bridge Module serves as the core component that links the AHB and APB buses. It comprises two main sub-modules: the AHB Slave interface and the APB Controller. This module translates high-performance AHB transactions into the simpler APB protocol, ensuring seamless communication between the two bus architectures.

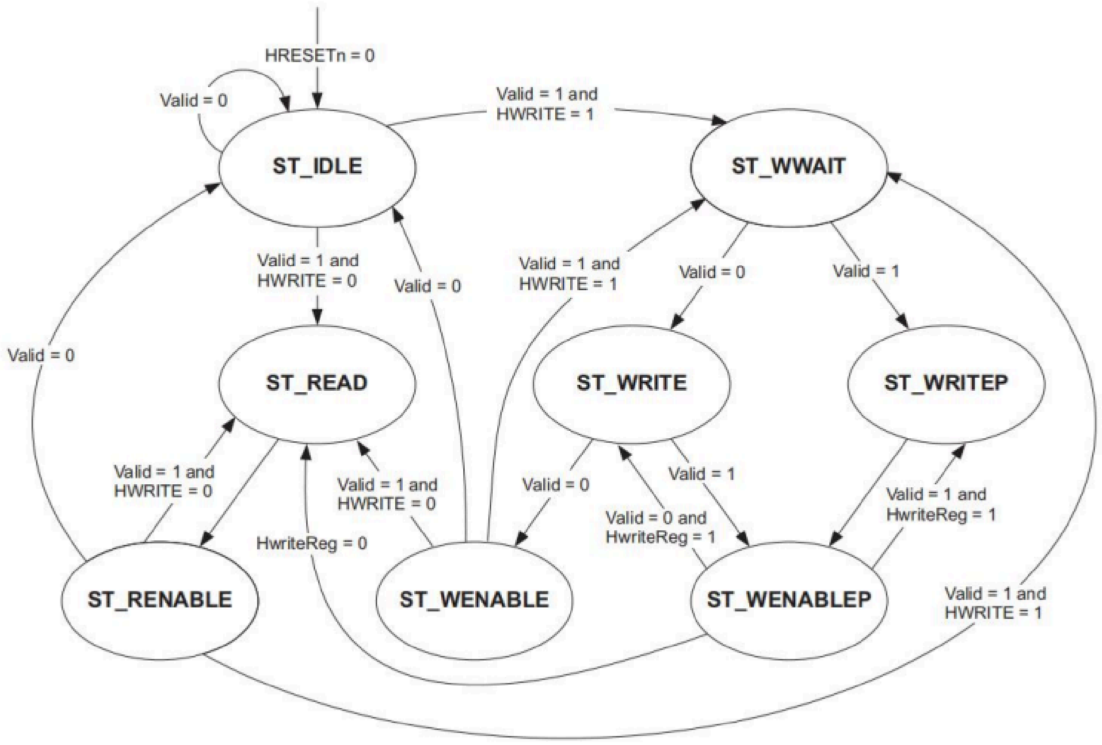
APB Controller:

- The APB Controller, as part of the bridge module, receives translated commands and data from the AHB Slave. It controls the timing and flow of data on the APB bus, ensuring that signals are correctly interpreted and forwarded to the appropriate APB peripherals. This module ensures that the low-power, low-bandwidth requirements of the APB are met.

APB Interface:

- The APB Interface module receives signals from the APB Controller and manages communication with the connected APB peripherals. It handles the final stage of data transfer, ensuring that the signals sent by the controller are correctly routed to the intended peripheral devices, such as UARTs, timers, or GPIOs.

APB CONTROLLER STATE DIAGRAM:



DIFFERENT STATE USED IN APB CONTROLLER:

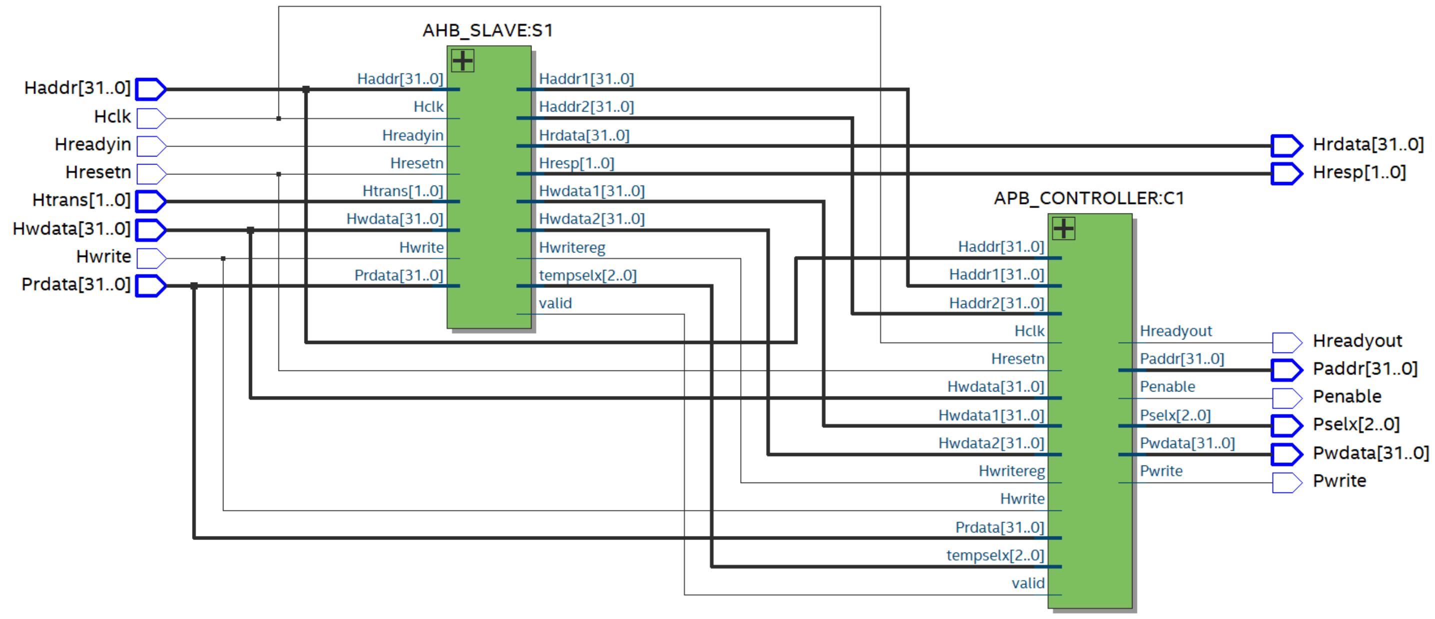
```
parameter ST_IDLE = 3'b000;
parameter ST_WWAIT = 3'b001;
parameter ST_WRITEP = 3'b010;
parameter ST_WENABLEP = 3'b011;
parameter ST_WRITE = 3'b100;
parameter ST_WENABLE = 3'b101;
parameter ST_READ = 3'b110;
parameter ST_RENABLE = 3'b111;
```

TOOLS USED: Quartus Prime, Modelsim.

HDL: Verilog

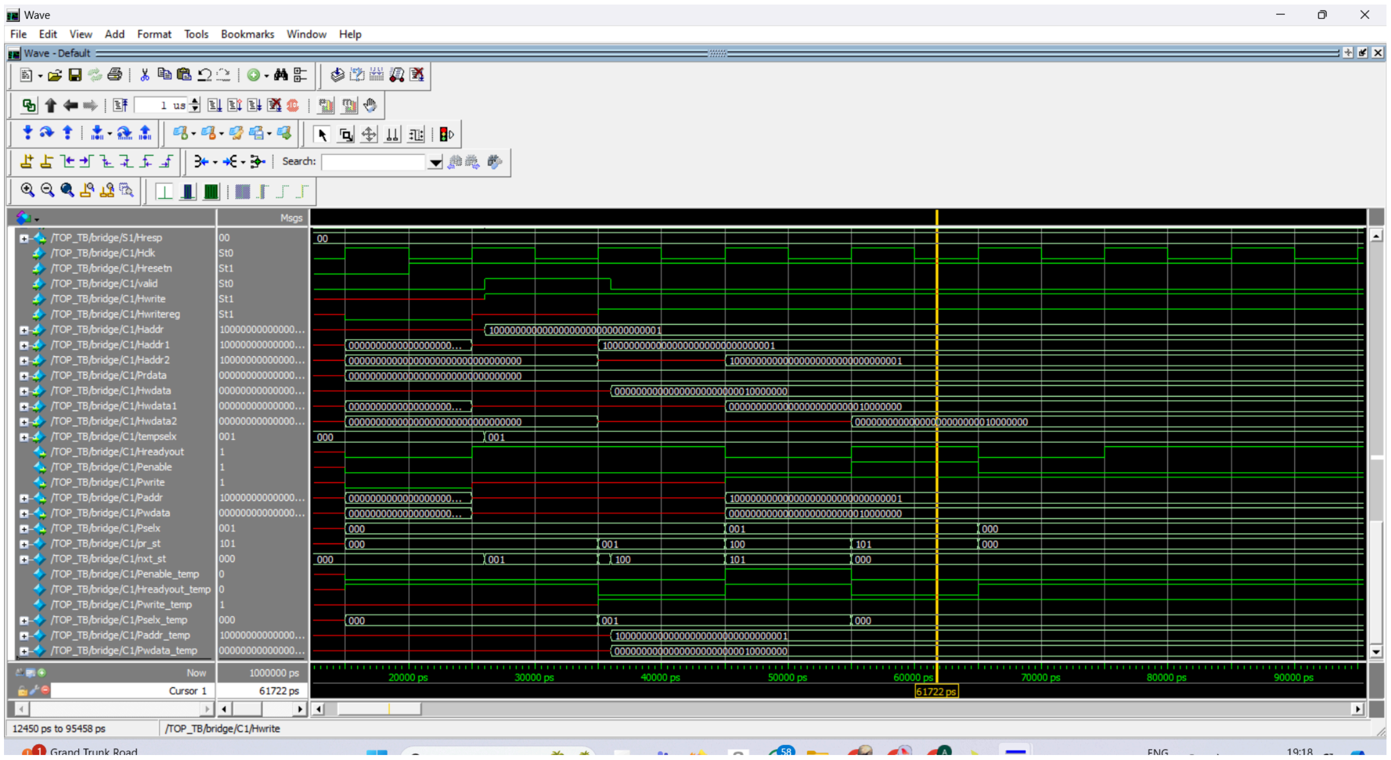
SYNTHESIS RESULT:

The following image shows the result of the synthesis of the bridge top module.



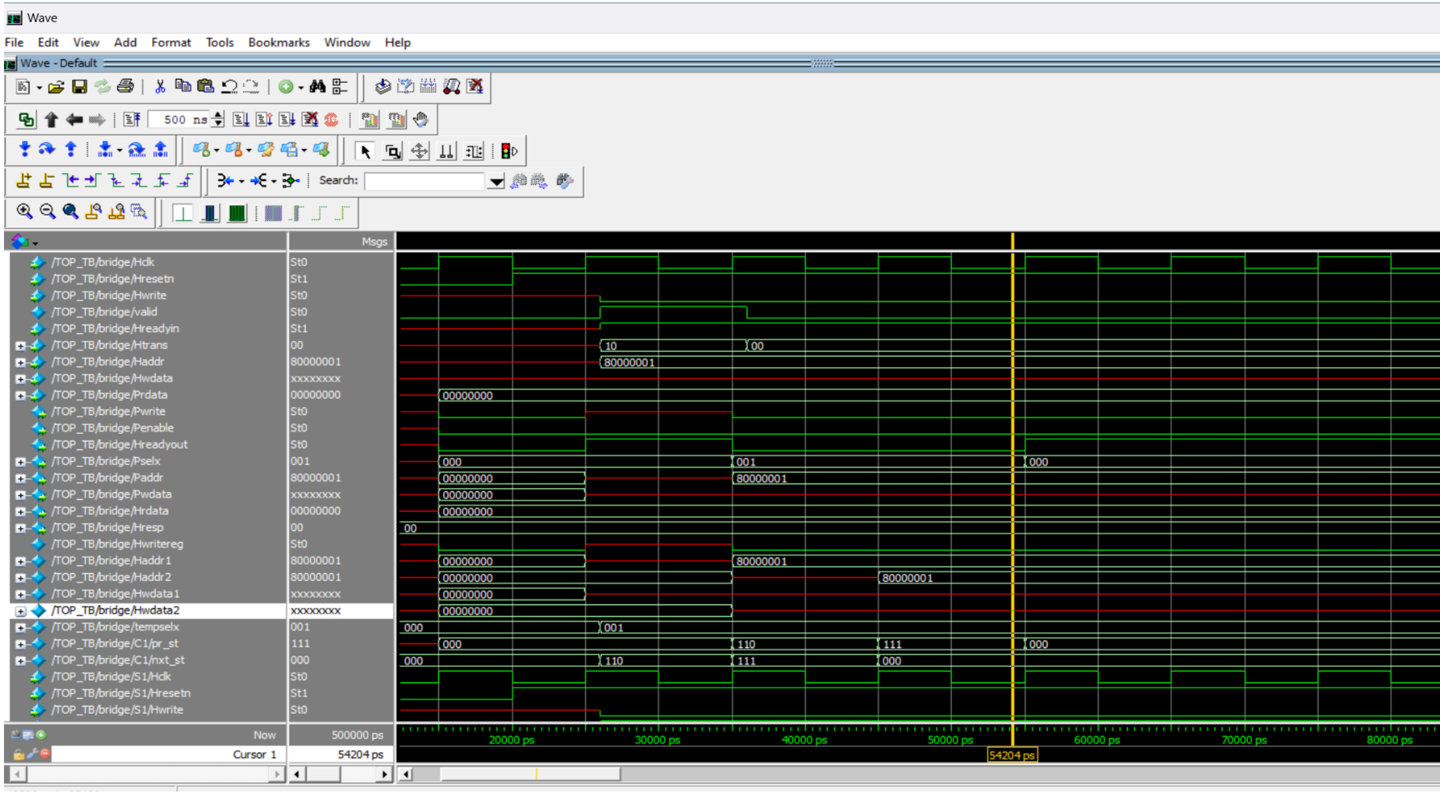
<u>OUTPUTS:</u>

SINGLE WRITE:



```
/TOP_TB/bridge/S1/Hresp            00
/TOP_TB/bridge/C1/Hclk             St0
/TOP_TB/bridge/C1/Hresetn          St1
/TOP_TB/bridge/C1/valid            St0
/TOP_TB/bridge/C1/Hwrite           St1
/TOP_TB/bridge/C1/Hwritereg        St1
/TOP_TB/bridge/C1/Haddr            10000000000000...
/TOP_TB/bridge/C1/Haddr1           10000000000000...
/TOP_TB/bridge/C1/Haddr2           10000000000000...
/TOP_TB/bridge/C1/Prdata           00000000000000...
/TOP_TB/bridge/C1/Hwdata           00000000000000...
/TOP_TB/bridge/C1/Hwdata1          00000000000000...
/TOP_TB/bridge/C1/Hwdata2          00000000000000...
/TOP_TB/bridge/C1/tempselx         001
/TOP_TB/bridge/C1/Hreadyout        1
/TOP_TB/bridge/C1/Penable          1
/TOP_TB/bridge/C1/Pwrite           1
/TOP_TB/bridge/C1/Paddr            10000000000000...
/TOP_TB/bridge/C1/Pwdata           00000000000000...
/TOP_TB/bridge/C1/Pselx            001
/TOP_TB/bridge/C1/pr_st            101
/TOP_TB/bridge/C1/nxt_st           000
/TOP_TB/bridge/C1/Penable_temp     0
/TOP_TB/bridge/C1/Hreadyout_temp   0
/TOP_TB/bridge/C1/Pwrite_temp      1
/TOP_TB/bridge/C1/Pselx_temp       000
/TOP_TB/bridge/C1/Paddr_temp       10000000000000...
/TOP_TB/bridge/C1/Pwdata_temp      00000000000000...
Now                                1000000 ps
Cursor 1                           61722 ps
12450 ps to 95458 ps    /TOP_TB/bridge/C1/Hwrite
```

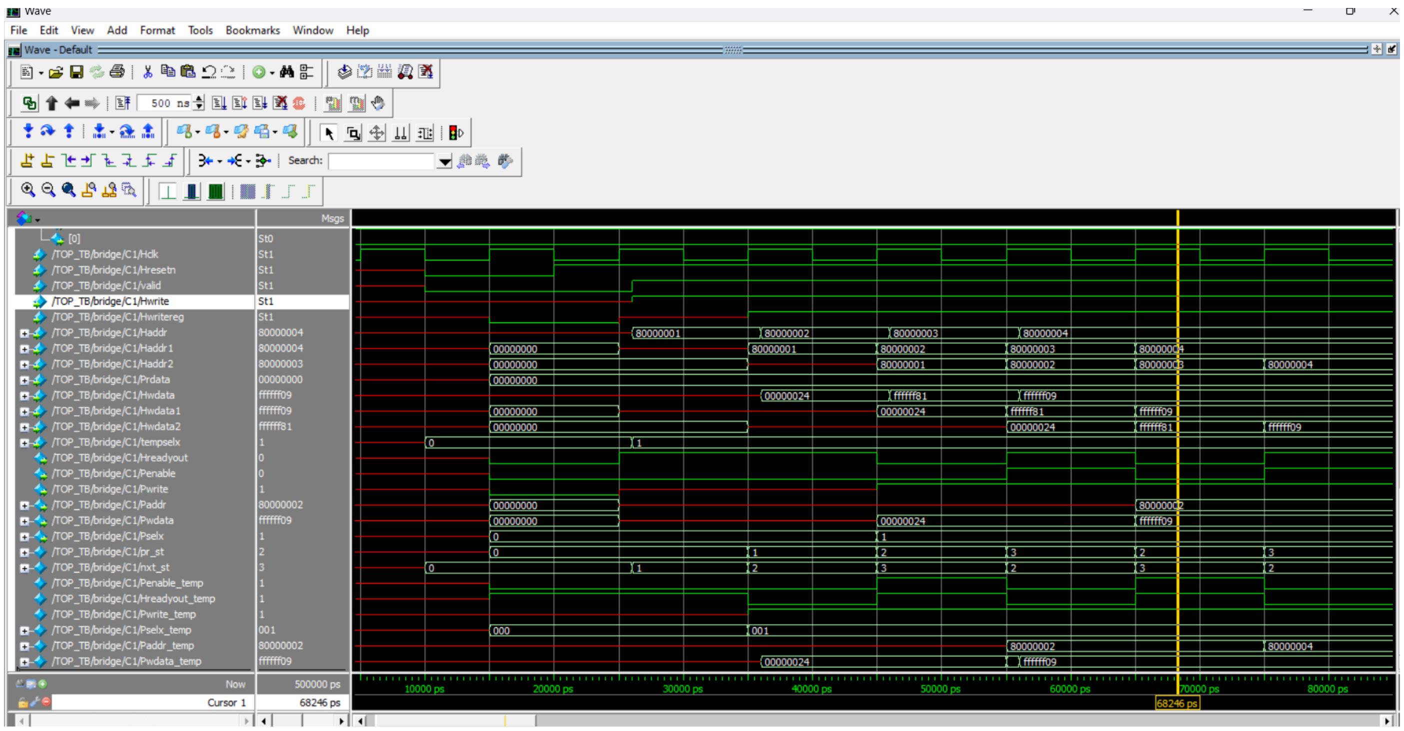
SINGLE READ:



```
/TOP_TB/bridge/Hclk          St0
/TOP_TB/bridge/Hresetn       St1
/TOP_TB/bridge/Hwrite        St0
/TOP_TB/bridge/valid         St0
/TOP_TB/bridge/Hreadyin      St1
/TOP_TB/bridge/Htrans        00
/TOP_TB/bridge/Haddr         80000001
/TOP_TB/bridge/Hwdata        xxxxxxxx
/TOP_TB/bridge/Prdata        00000000
/TOP_TB/bridge/Pwrite        St0
/TOP_TB/bridge/Penable       St0
/TOP_TB/bridge/Hreadyout     St0
/TOP_TB/bridge/Pselx         001
/TOP_TB/bridge/Paddr         80000001
/TOP_TB/bridge/Pwdata        xxxxxxxx
/TOP_TB/bridge/Hrdata        00000000
/TOP_TB/bridge/Hresp         00
/TOP_TB/bridge/Hwritereg     St0
/TOP_TB/bridge/Haddr1        80000001
/TOP_TB/bridge/Haddr2        80000001
/TOP_TB/bridge/Hwdata1       xxxxxxxx
/TOP_TB/bridge/Hwdata2       xxxxxxxx
/TOP_TB/bridge/tempselx      001
/TOP_TB/bridge/C1/pr_st      111
/TOP_TB/bridge/C1/nxt_st     000
/TOP_TB/bridge/S1/Hclk       St0
/TOP_TB/bridge/S1/Hresetn    St1
/TOP_TB/bridge/S1/Hwrite     St0

Now        500000 ps
Cursor 1    54204 ps
```

BURST WRITE:



```
[0]                                    St0
/TOP_TB/bridge/C1/Hclk                 St1
/TOP_TB/bridge/C1/Hresetn              St1
/TOP_TB/bridge/C1/valid                St1
/TOP_TB/bridge/C1/Hwrite               St1
/TOP_TB/bridge/C1/Hwritereg            St1
/TOP_TB/bridge/C1/Haddr                80000004
/TOP_TB/bridge/C1/Haddr1               80000004
/TOP_TB/bridge/C1/Haddr2               80000003
/TOP_TB/bridge/C1/Prdata               00000000
/TOP_TB/bridge/C1/Hwdata               ffffff09
/TOP_TB/bridge/C1/Hwdata1              ffffff09
/TOP_TB/bridge/C1/Hwdata2              ffffff81
/TOP_TB/bridge/C1/tempselx             1
/TOP_TB/bridge/C1/Hreadyout            0
/TOP_TB/bridge/C1/Penable              0
/TOP_TB/bridge/C1/Pwrite               1
/TOP_TB/bridge/C1/Paddr                80000002
/TOP_TB/bridge/C1/Pwdata               ffffff09
/TOP_TB/bridge/C1/Pselx                1
/TOP_TB/bridge/C1/pr_st                2
/TOP_TB/bridge/C1/nxt_st               3
/TOP_TB/bridge/C1/Penable_temp         1
/TOP_TB/bridge/C1/Hreadyout_temp       1
/TOP_TB/bridge/C1/Pwrite_temp          1
/TOP_TB/bridge/C1/Pselx_temp           001
/TOP_TB/bridge/C1/Paddr_temp           80000002
/TOP_TB/bridge/C1/Pwdata_temp          ffffff09

Now        500000 ps
Cursor 1    68246 ps
```

CONCLUSION:

Through this AHB to APB bridge project, I gained comprehensive knowledge and practical experience in several key areas:

1. **AMBA Architecture**:
   - I learned about the Advanced Microcontroller Bus Architecture (AMBA), which is crucial for the efficient design of system-on-chip (SoC) devices. Understanding the different bus protocols within AMBA, such as AHB and APB, provided me with a solid foundation in SoC communication mechanisms.
2. **AHB and APB Protocols**:
   - I delved into the specifics of AHB (Advanced High-Performance Bus) and APB (Advanced Peripheral Bus) protocols. This included understanding how high-performance and low-performance components communicate within a SoC and the distinct characteristics of each protocol.
3. **Real SoC Transactions**:
   - By working on this project, I observed how transactions occur in real SoC devices. This practical insight into data transfers, command signaling, and peripheral communication was invaluable in understanding the intricacies of SoC design and operation.
4. **Hands-on with Verilog**:
   - I acquired hands-on experience with Verilog, a hardware description language used to model electronic systems. Writing and debugging Verilog code was a crucial part of developing the various modules in the AHB to APB bridge.
5. **Using Questa Prime and ModelSim**:
   - I learned to use Questa Prime and ModelSim, essential tools for simulation and verification of hardware designs. These tools enabled me to test and validate the functionality of the bridge modules, ensuring they met the design specifications and performed correctly.

Github repository link:https://github.com/itsmeAakriti/AHP2APB

References :

- https://developer.arm.com/Architectures/AMBA
- Resources from Maven silicon