VLSI Design Internship - AHB to APB Bridge Design

AN INDUSTRIAL INTERNSHIP TRAINING REPORT

Submitted by

22BEC1478 – PRADESH KUMAR M

BECE399J – INDUSTRIAL INTERNSHIP

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

ELECTRONICS AND COMMUNICATION ENGINEERING

November 2024

School of Electronics Engineering

DECLARATION BY THE CANDIDATE

I hereby declare that the Industrial Internship Report entitled "**VLSI Design Internship - AHB to APB Bridge Design**" submitted by me to VIT University, Chennai in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in **Electronics and Communications Engineering** is a record of bonafide industrial training undertaken by me under the supervision of **Mr. Rahul Vinod, Maven Silicon, Bengaluru**. I further declare that the work reported in this report has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Location: Bengaluru Signature of the Candidate

Date: 15.11.2024

aceic™
Accelerate Collaborate Excel

MAVEN SILICON
Centre of Excellence in VLSI

CERTIFICATE OF COMPLETION

This is to certify that Mr./Ms. Pradesh Kumar M (22BEC1478) has successfully completed VLSI Design Internship Program from 05-June-2024 to 13-July-2024

During the internship program with us, he/she worked on AHB2APB Bridge Design project.

Date: 02/08/2024

Place: Bengaluru

MSUID: MS/V-DI/2024-25/504

MAVEN SILICON SOFTECH PVT. LTD. BANGALORE

SIVAKUMAR P R

FOUNDER & CEO, MAVEN SILICON

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Electronics Engineering

BONAFIDE CERTIFICATE

This is to certify that the Industrial Internship Report entitled "**VLSI Design Internship - AHB to APB Bridge Design**" submitted by **Pradesh Kumar M (22BEC1478)** to VIT, Chennai in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology** in **Electronics and Communication Engineering** is a record of bonafide industrial internship undertaken by him/her fulfills the requirements as per the regulations of this institute and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

_____

Signature of the Examiner **Signature of the Examiner**

Date: Date:

Head of the Department (B.Tech ECE)

ACKNOWLEDGEMENT

I extend my sincere gratitude to all those who made this internship experience both educational and enriching. I am immensely grateful to Maven Silicon and the Career Development Center (CDC) at my college for providing me with this invaluable opportunity to pursue a design internship focused on the AHB to APB bridge design. This experience has been crucial in helping me apply theoretical knowledge to practical design challenges in the field of digital design.

I would like to express my heartfelt thanks to my mentor, Ms. Sathya Priya, for her invaluable guidance and encouragement throughout this journey. Her support in helping me grasp the fundamentals of digital design and progress toward advanced concepts has been instrumental to my development. She generously dedicated her time to teaching me Verilog and guiding me through each stage of my AHB to APB bridge design project, which has deepened my understanding and appreciation for the field.

I am also grateful to the entire team at Maven Silicon for fostering a conducive learning environment and sharing their expertise. Their openness and willingness to provide insights into the industry have significantly enriched my learning experience. This internship has been a transformative experience, offering me practical skills and knowledge that will be invaluable as I move forward in my career.

PRADESH KUMAR M

(22BEC1478)

TABLE OF CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

AHB	:	Advanced High-performance Bus
APB	:	Advanced Peripheral Bus
SoC	:	System on Chip
AMBA	:	Advanced Microcontroller Bus Architecture
IoT	:	Internet of Things
ASB	:	Advanced System Bus
RAM	:	Random Access Memory
DSP	:	Digital Signal Processor
ASIC	:	Application Specific Integrated Circuit
DFF	:	D Flip Flop
VLSI	:	Very Large Scale Integration
DMA	:	Direct Memory Access
UART	:	Universal Asynchronous Receiver/Transmitter
GPIO	:	General Purpose Input Output

ABSTRACT

The increasing complexity of System-on-Chip (SoC) designs necessitates efficient communication mechanisms between high-speed processing cores and low-speed peripheral devices. To address this need, the Advanced Microcontroller Bus Architecture (AMBA) specification, developed by ARM, provides a suite of bus protocols, including the Advanced High-performance Bus (AHB) and the Advanced Peripheral Bus (APB). This project focuses on designing an AHB to APB Bridge, a critical component that enables seamless data transfer between the high-performance AHB and the low-power APB buses.

The primary objective of this project is to develop a fully functional bridge that efficiently converts AHB transactions into APB-compatible signals, ensuring smooth communication across different bus protocols within an SoC. The design encompasses two main components: the AHB Slave Interface, which handles address decoding and transaction control on the AHB side, and the APB Controller, which manages the conversion of signals for APB peripherals. This bridge supports single and burst transfer modes, with an emphasis on low latency, high throughput, and minimal power consumption.

The implementation was carried out using industry-standard tools, such as ModelSim and Quartus Prime, and was thoroughly tested using comprehensive testbenches to validate its functionality under various operating conditions. This project demonstrates the importance of bus architecture optimization in embedded systems and highlights the significance of the AHB to APB Bridge in enhancing the performance of SoC designs. The successful integration of this bridge within AMBA-based systems can significantly improve the scalability, modularity, and power efficiency of complex digital systems, making it a versatile solution for applications in automotive, consumer electronics, and IoT devices.

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

The AMBA architecture, developed by ARM, is widely used in embedded systems to provide a standardized, flexible, and scalable interface for connecting different functional blocks. The AMBA specification includes several bus protocols, each catering to specific performance and power requirements:

1. Advanced High-performance Bus (AHB): Designed for high-performance and high-frequency systems, AHB serves as the system backbone, connecting processors, on-chip memory, and other high-speed peripherals. It supports burst transfers, pipelined operations, and high throughput, making it ideal for performance-critical applications.

2. Advanced System Bus (ASB): A simpler alternative to AHB, the ASB provides high-performance capabilities but with reduced complexity. It is suited for systems where the full features of AHB are not necessary but moderate performance is still required.

3. Advanced Peripheral Bus (APB): Optimized for low power and minimal complexity, APB is typically used for connecting slower peripheral devices like UARTs, GPIOs, and timers. Unlike AHB, APB does not support burst transfers or pipelining, focusing instead on simplicity and energy efficiency.

The AHB to APB Bridge serves as a crucial component in bridging the gap between the high-speed AHB and the low-speed APB buses. Its primary function is to convert AHB bus transactions into APB bus transactions, enabling peripherals connected to the APB to communicate seamlessly with the core system connected to the AHB. This bridge ensures that the data transfer is handled efficiently without compromising system performance or increasing power consumption.

1.2 OBJECTIVE OF THE PROJECT

The objective of this project is to design and implement a fully functional AHB to APB Bridge that can facilitate data transfer between the AHB and APB bus protocols. The bridge must be capable of handling various types of transactions, including single reads/writes and burst operations, while ensuring low latency and minimal power consumption. The design also aims to meet the following key requirements:

- **Compatibility**: Seamless integration with existing AMBA-based SoC designs.
- **Performance**: High-speed data transfer from AHB to APB with minimal delay.
- **Scalability**: Ability to support a wide range of peripherals connected to the APB.
- **Low Power**: Optimized for energy efficiency, particularly in idle states.

1.3 SCOPE OF THE PROJECT

The project encompasses the following design and implementation phases:

1. **AHB Slave Interface Design**: This module decodes AHB addresses and initiates read/write transactions based on the control signals received from the AHB master.
2. **APB Controller Design**: This component manages the conversion of AHB transactions into APB-compatible signals, ensuring proper peripheral selection, address mapping, and data handling.
3. **Integration and Testing**: A top-level module integrates the AHB slave interface and APB controller, forming the complete AHB to APB bridge. Comprehensive testbenches are created to verify functionality under various operational scenarios, including single and burst transactions.
4. **Synthesis and Optimization**: The design is synthesized using industry-standard tools, focusing on achieving optimal performance, area, and power metrics.

1.4 IMPORTANCE OF THE PROJECT

The AHB to APB Bridge plays a vital role in modern SoCs by allowing high-speed processors to communicate efficiently with low-speed peripherals. By providing a dedicated bridge for protocol conversion, it offloads the complexity from the main system bus, resulting in:

- **Improved System Efficiency**: Allows high-speed and low-speed devices to coexist without impacting overall system performance.
- **Reduced Power Consumption**: Enables selective access to low-power peripherals, thus conserving energy in power-sensitive applications.
- **Modular Design Approach**: Facilitates a modular SoC architecture where additional peripherals can be easily integrated without major redesigns. By focusing on both functional correctness and system-level optimizations, the AHB to APB Bridge design aims to be a versatile solution for embedded systems across various domains, including automotive, consumer electronics, and IoT devices.

CHAPTER 2

AMBA SPECIFICATIONS

2.1 AMBA BUSES

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:

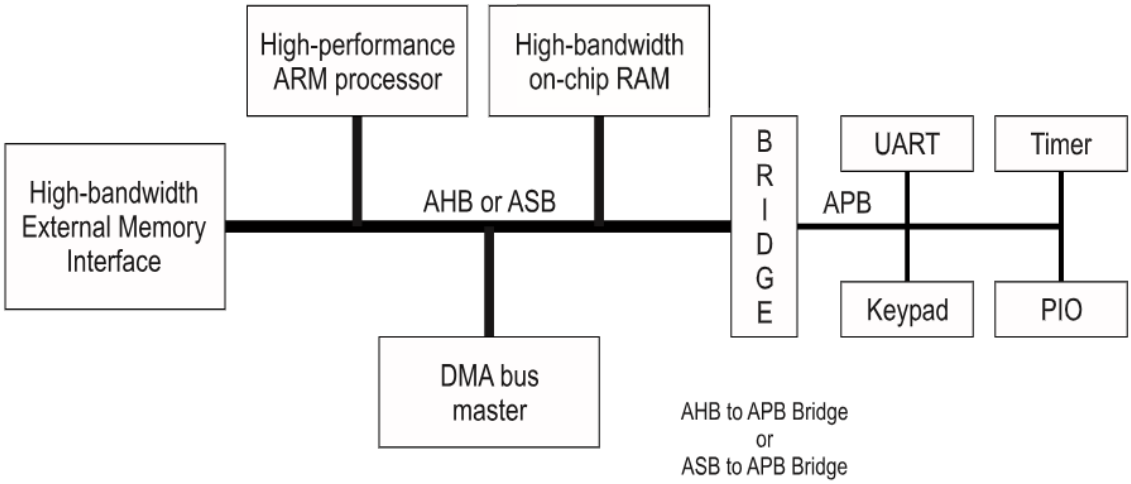
- Advanced High-performance Bus (AHB)
- Advanced System Bus (ASB)
- Advanced Peripheral Bus (APB)

A test methodology is included with the AMBA specification which provides an infrastructure for modular macrocell test and diagnostic access.

- 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 high-performance system modules. AMBA ASB is an alternative system bus 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.

2.2 TYPICAL AMBA-BASED MICROCONTROLLER

An AMBA-based microcontroller typically consists of a high-performance system backbone bus (AMBA AHB or AMBA ASB), able to sustain the external memory bandwidth, on which the CPU, on-chip memory and other Direct Memory Access (DMA) devices reside. This bus provides a high-bandwidth interface between the elements that are involved in the majority of transfers. Also located on the high- performance bus is a bridge to the lower bandwidth APB, where most of the peripheral devices in the system are located.



AMBA APB provides the basic peripheral macrocell communications infrastructure as a secondary bus from the higher bandwidth pipelined main system bus. Such peripherals typically:

- have interfaces which are memory-mapped registers
- have no high-bandwidth interfaces
- are accessed under programmed control.

The external memory interface is application-specific and may only have a narrow data path, but may also support a test access mode which allows the internal AMBA AHB, ASB and APB modules to be tested in isolation with system-independent test sets.

2.3 AMBA AHB

AHB is a new generation of AMBA bus which is intended to address the requirements of high-performance synthesizable designs. It is a high-performance system bus that supports multiple bus masters and provides high-bandwidth operation.

AMBA AHB implements the features required for high-performance, high clock frequency systems including:

- burst transfers
- split transactions
- single-cycle bus master handover
- single-clock edge operation
- non-tristate implementation
- wider data bus configurations (64/128 bits).

Bridging between this higher level of bus and the current ASB/APB can be done efficiently to ensure that any existing designs can be easily integrated. An AMBA AHB design may contain one or more bus masters, typically a system would contain at least the processor and test interface. However, it would also be common for a Direct Memory Access (DMA) or Digital Signal Processor (DSP) to be included as bus masters.

The external memory interface, APB bridge and any internal memory are the most common AHB slaves. Any other peripheral in the system could also be included as an AHB slave. However, low-bandwidth peripherals typically reside on the APB.

A typical AMBA AHB system design contains the following components:

- AHB master: A bus master is able to initiate read and write operations by providing an address and control information. Only one bus master is allowed to actively use the bus at any one time.
- AHB slave: A bus slave responds to a read or write operation within a given address-space range. The bus slave signals back to the active master the success, failure or waiting of the data transfer.
- AHB arbiter: The bus arbiter ensures that only one bus master at a time is allowed to initiate data transfers. Even though the arbitration protocol is fixed, any arbitration algorithm, such as highest priority or fair access can be implemented depending on the application requirements. An AHB would include only one arbiter, although this would be trivial in single bus master systems.
- AHB decoder: The AHB decoder is used to decode the address of each transfer and provide a select signal for the slave that is involved in the transfer. A single centralized decoder is required in all AHB implementations.

2.4 AMBA APB

The APB is part of the AMBA hierarchy of buses and is optimized for minimal power consumption and reduced interface complexity. The AMBA APB appears as a local secondary bus that is encapsulated as a single AHB or ASB slave device. APB provides a low-power extension to the system bus which builds on AHB or ASB signals directly.

The APB bridge appears as a slave module which handles the bus handshake and control signal retiming on behalf of the local peripheral bus. By defining the APB interface from the starting point of the system bus, the benefits of the system diagnostics and test methodology can be exploited. The AMBA APB should be used to interface to any peripherals which are low bandwidth and do not require the high performance of a pipelined bus interface.

The latest revision of the APB is specified so that all signal transitions are only related to the rising edge of the clock. This improvement ensures the APB peripherals can be integrated easily into any design flow, with the following advantages:

- high-frequency operation easier to achieve
- performance is independent of the mark-space ratio of the clock
- static timing analysis is simplified by the use of a single clock edge
- no special considerations are required for automatic test insertion
- many Application Specific Integrated Circuit (ASIC) libraries have a better selection of rising edge registers
- easy integration with cycle-based simulators.

These changes to the APB also make it simpler to interface it to the new AHB. An AMBA APB implementation typically contains a single APB bridge which is required to convert

AHB or ASB transfers into a suitable format for the slave devices on the APB. The bridge provides latching of all address, data and control signals, as well as providing a second level of decoding to generate slave select signals for the APB peripherals.

All other modules on the APB are APB slaves. The APB slaves have the following interface specification:

- address and control valid throughout the access (unpipelined)
- zero-power interface during non-peripheral bus activity (peripheral bus is static when not in use)
- timing can be provided by decode with strobe timing (unclocked interface)
- write data valid for the whole access (allowing glitch-free transparent latch implementations).

2.5 AMBA AHB SIGNALS

NAME	SOURCE	DESCRIPTION
HWDATA[31:0] Write data bus	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. However, this may easily be extended to allow for higher bandwidth operation.
HSELx Slave Select	Decoder	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. However, this may easily be extended to allow for higher bandwidth operation.
HRDATA[31:0] Read data bus	Slave	The read data bus is used to transfer data from bus slaves to the bus master during read operations. A minimum data bus width of 32 bits is recommended. However, this may easily be extended to allow for higher bandwidth operation.
HREADY Transfer done	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. Note: Slaves on the bus require HREADY as both an input and an output signal.
HRESP[1:0] Transfer response	Slave	The transfer response provides additional information on the status of a transfer. Four different responses are provided, OKAY, ERROR, RETRY and SPLIT.

2.6 AMBA APB SIGNALS

NAME	DESCRIPTION
PCLK Bus clock	The rising edge of PCLK is used to time all transfers on the APB.
PRESETn APB reset	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] APB address bus	This is the APB address bus, which may be up to 32-bits wide and is driven by the peripheral bus bridge unit.
PSELx APB select	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 APB strobe	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 APB transfer direction	When HIGH this signal indicates an APB write access and when LOW a read access.
PRDATA APB read data bus	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 APB write data bus	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

CHAPTER 3

AMBA AHB

3.1 BUS INTERCONNECTION

The AMBA AHB bus protocol is designed to be used with a central multiplexor interconnection scheme. Using this scheme all bus masters drive out the address and control signals indicating the transfer they wish to perform and the arbiter determines which master has its address and control signals routed to all of the slaves. A central decoder is also required to control the read data and response signal multiplexor, which selects the appropriate signals from the slave that is involved in the transfer.

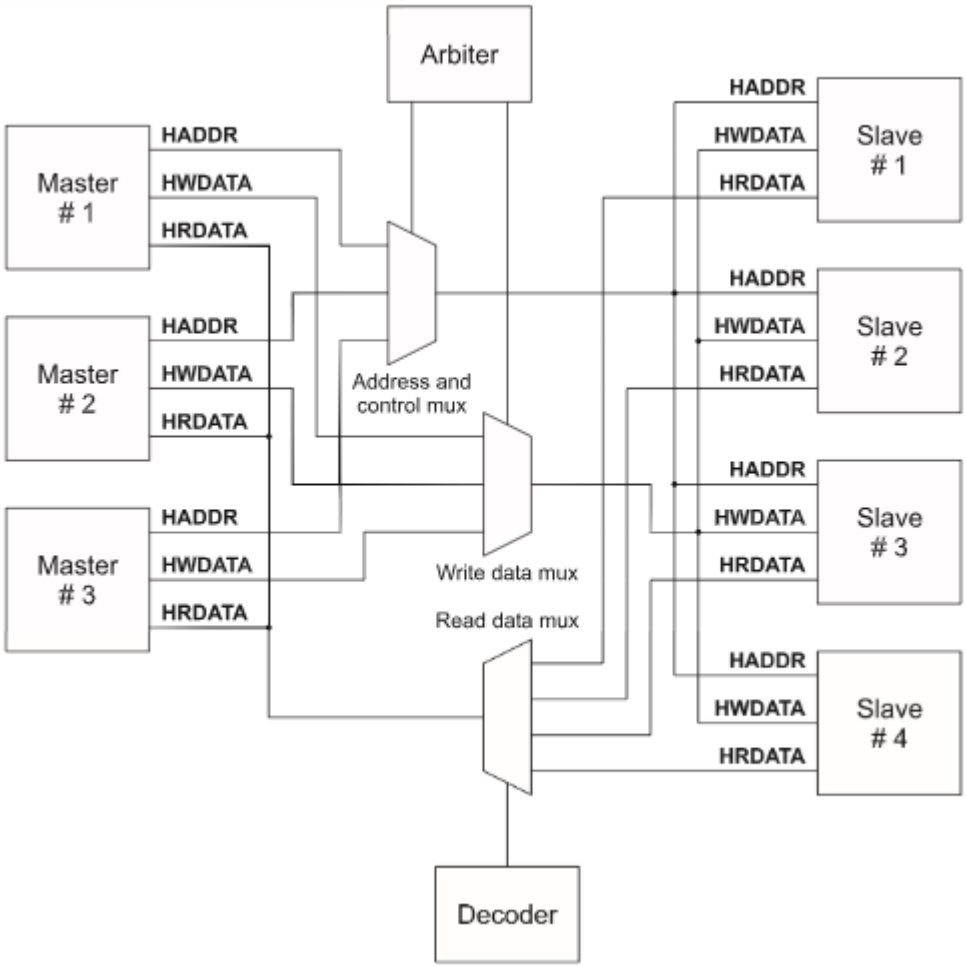


FIGURE: Structure required to implement an AMBA AHB design with three masters and four slaves.

3.2 AHB OPERATION

Before an AMBA AHB transfer can commence the bus master must be granted access to the bus. This process is started by the master asserting a request signal to the arbiter. Then the arbiter indicates when the master will be granted use of the bus. A granted bus master starts an AMBA AHB transfer by driving the address and control signals. These signals provide information on the address, direction and width of the transfer, as well as an indication if the transfer forms part of a burst. Two different forms of burst transfers are allowed:

- incrementing bursts, which do not wrap at address boundaries
- wrapping bursts, which wrap at particular address boundaries.

A write data bus is used to move data from the master to a slave, while a read data bus is used to move data from a slave to the master. Every transfer consists of an address and control cycle and one or more cycles for the data The address cannot be extended and therefore all slaves must sample the address during this time. The data, however, can be extended using the HREADY signal. When LOW this signal causes wait states to be inserted into the transfer and allows extra time for the slave to provide or sample data.

During a transfer the slave shows the status using the response signals, HRESP[1:0]:

- OKAY: The OKAY response is used to indicate that the transfer is progressing normally and when HREADY goes HIGH this shows the transfer has completed successfully.
- ERROR: The ERROR response indicates that a transfer error has occurred and the transfer has been unsuccessful.
- RETRY and SPLIT: Both the RETRY and SPLIT transfer responses indicate that the transfer cannot complete immediately, but the bus master should continue to attempt the transfer.

In normal operation a master is allowed to complete all the transfers in a particular burst before the arbiter grants another master access to the bus. However, in order to avoid excessive arbitration latencies it is possible for the arbiter to break up a burst and in such cases the master must re-arbitrate for the bus in order to complete the remaining transfers in the burst.

3.3 BASIC TRANSFER

An AHB transfer consists of two distinct sections:

- The address phase, which lasts only a single cycle.
- The data phase, which may require several cycles. This is achieved using the HREADY signal.

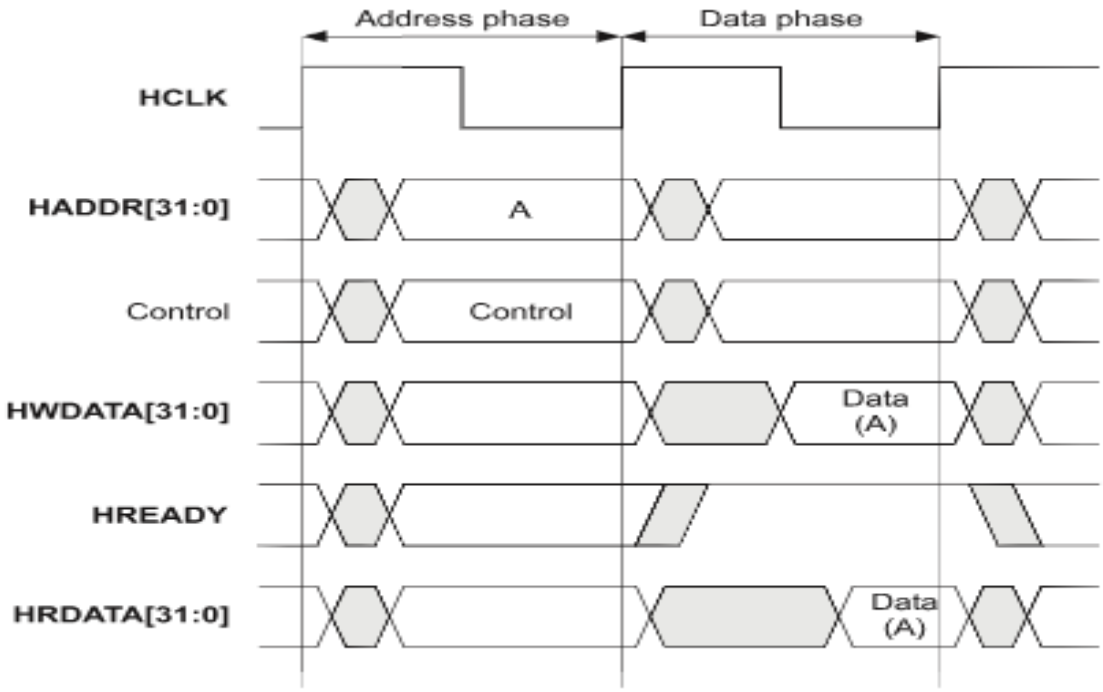


FIGURE: Simple transfer with no wait states

In a simple transfer with no wait states:

- The master drives the address and control signals onto the bus after the rising edge of HCLK.
- The slave then samples the address and control information on the next rising edge of the clock.

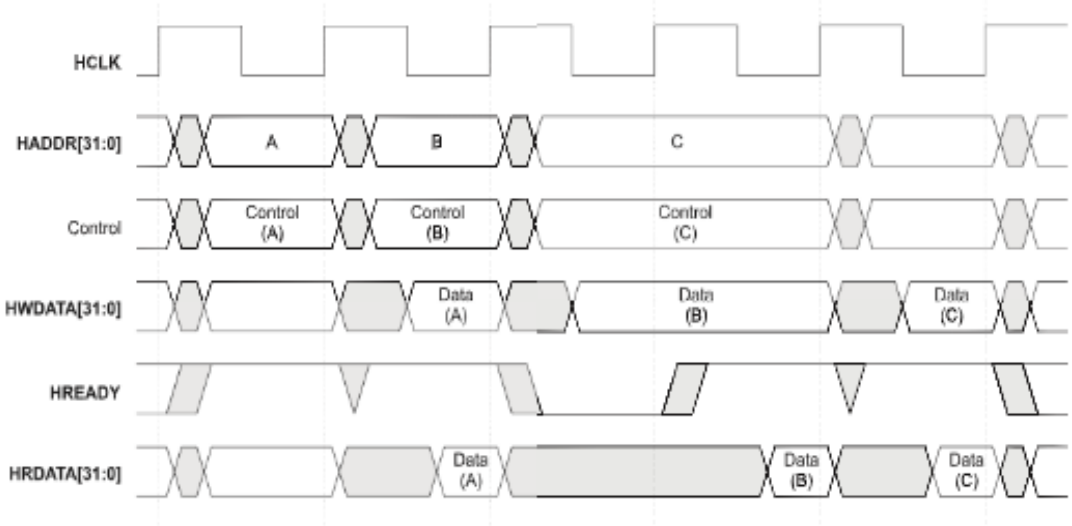


FIGURE: Multiple transfers

In this figure

- the transfers to addresses A and C are both zero wait state
- the transfer to address B is one wait state
- extending the data phase of the transfer to address B has the effect of extending the address phase of the transfer to address C.

3.4 TRANSFER TYPE

HTRANS[1:0]	TYPE	DESCRIPTION
00	IDLE	Indicates that no data transfer is required. The IDLE transfer type is used when a bus master is granted the bus, but does not wish to perform a data transfer. Slaves must always provide a zero wait state OKAY response to IDLE transfers and the transfer should be ignored by the slave.
01	BUSY	The BUSY transfer type allows bus masters to insert IDLE cycles in the middle of bursts of transfers. This transfer type indicates that the bus master is continuing with a burst of transfers, but the next transfer cannot take place immediately. When a master uses the BUSY transfer type the address and control signalı must reflect the next transfer in the burst The transfer should be ignored by the slave. Slaves must always provide a zero wait state OKAY response, in the same way that they respond to IDLE response.
10	NONSEQ	Indicates the first transfer of a burst or a single transfer. The address and control signals are unrelated to the previous transfer. Single transfers on the bus are treated as bursts of one and therefore the transfer type is NONSEQUENTIAL
11	SEQ	The remaining transfers in a burst are SEQUENTIAL and the address is related to the previous transfer. The control information is identical to the previous transfer. The address is equal to the address of the previous transfer plus the size (in bytes). In the case of a wrapping burst the address of the transfer wraps at the address boundary equal to the size (in bytes) multiplied by the number of beats in the transfer (either 4, 8 or 16).

3.5 BURST OPERATION

Four, eight and sixteen-beat bursts are defined in the AMBA AHB protocol, as well as undefined-length bursts and single transfers. Both incrementing and wrapping bursts are supported in the protocol. Burst information is provided using HBURST [2:0] and the eight possible types are defined below:

HBURST[2:0]	TYPE	DESCRIPTION
000	SINGLE	Single transfer
001	INCR	Incrementing burst of unspecified length
010	WRAP4	4- beat wrapping burst
011	INCR4	4- beat incrementing burst
100	WRAP8	8- beat wrapping burst
101	INCR8	8- beat incrementing burst
110	WRAP16	16- beat wrapping burst
111	INCR16	16- beat incrementing burst

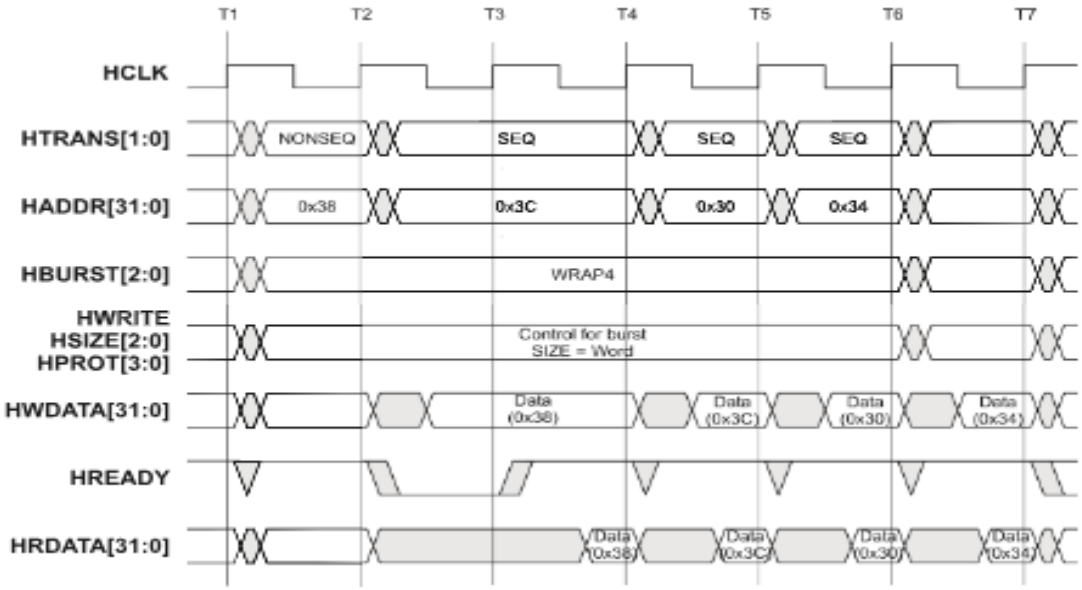


FIGURE: Four-beat wrapping burst

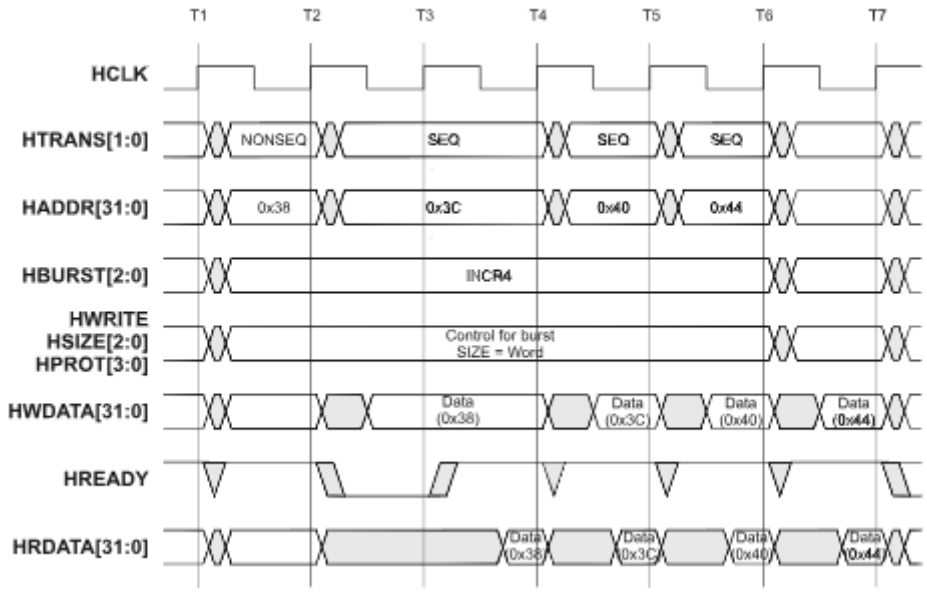


FIGURE: Four-beat incrementing burst

3.6 CONTROL SIGNALS

As well as the transfer type and burst type each transfer will have a number of control signals that provide additional information about the transfer. These control signals. have exactly the same timing as the address bus. However, they must remain constant throughout a burst of transfers.

- Transfer direction: When HWRITE is HIGH, this signal indicates a write transfer and the master will broadcast data on the write data bus, HWDATA[31:0]. When LOW a read transfer will be performed and the slave must generate the data on the read data bus HRDATA[31:0]
- Transfer size: HSIZE [2:0] indicates the size of the transfer.

HSIZE[2:0]	SIZE	DESCRIPTION
000	8 bits	Byte
001	16 bits	Halfword
010	32 bits	Word
011	64 bits	
100	128 bits	4- word line
101	256 bits	8- word line
110	512 bits	
111	I024 bits	

3.7 SLAVE TRANSFER RESPONSE

Whenever a slave is accessed it must provide a response which indicates the status of the transfer. The HREADY signal is used to extend the transfer and this works in combination with the response signals, HRESP[1:0], which provide the status of the transfer. The slave can complete the transfer in a number of ways. It can:

- complete the transfer immediately
- insert one or more wait states to allow time to complete the transfer
- signal an error to indicate that the transfer has failed
- delay the completion of the transfer, but allow the master and slave to back off the bus, leaving it available for other transfers.

The HREADY signal is used to extend the data portion of an AHB transfer. When. LOW the HREADY signal indicates the transfer is to be extended and when HIGH indicates that the transfer can complete.

The encoding of HRESP[1:0], transfer response signals, and their description are given below:

HRESP[1:0]	RESPONSE	DESCRIPTION
00	OKAY	When HREADY is HIGH this shows the transfer has completed successfully. The OKAY response is also used for any additional cycles that are inserted, with HREADY LOW, prior to giving one of the three other responses.
01	ERROR	This response shows an error has occurred. The error condition should be signalled to the bus master so that it is aware the transfer has been unsuccessful. A two-cycle response is required for an error condition
10	RETRY	The RETRY response shows the transfer has not yet completed, so the bus master should retry the transfer. The master should continue to retry the transfer until it completes. A two-cycle RETRY response is required.
11	SPLIT	The transfer has not yet completed successfully. The bus master must retry the transfer when it is next granted access to the bus. The slave will request access to the bus on behalf of the master when the transfer can complete. A two-cycle SPLIT response is required.

3.8 DATA BUSES

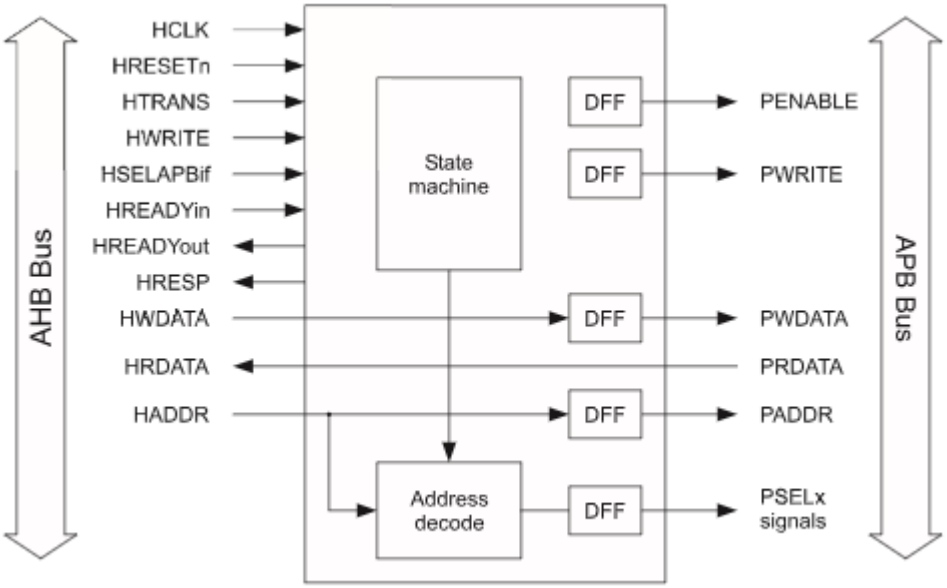
- HWDATA[31:0]: The write data bus is driven by the bus master during write transfers. If the transfer is extended then the bus master must hold the data valid until the transfer completes, as indicated by HREADY HIGH All transfers must be aligned to the address boundary equal to the size of the transfer. For example, word transfers must be aligned to word address boundaries (that is A[1:0] 00), halfword transfers must be aligned to halfword address boundaries (that is A[0] = 0).

- HRDATA[31:0]: The read data bus is driven by the appropriate slave during read transfers. If the slave extends the read transfer by holding HREADY LOW then the slave only needs to provide valid data at the end of the final cycle of the transfer, as indicated by HREADY HIGH

CHAPTER 4

APB BRIDGE

4.1 BLOCK DIAGRAM



The main sections of this module are:

- AHB slave bus interface
- APB transfer state machine, which is independent of the device memory map.
- APB output signal generation.

To add new APB peripherals, or alter the system memory map, only the address decode sections need to be modified,

4.2 APB BRIDGE SIGNALS

SIGNAL	TYPE	DIRECTION	DESCRIPTION
HCLK	Bus clock	Input	This clock times all bus transfers.
HRESETn	Reset	Input	The bus reset signal is active LOW, and is used to reset the system and the bus.
HADDR[1:0]	Address bus	Input	The 32-bit system address bus.
HTRANS[1:0]	Transfer type	Input	This indicates the type of the current transfer, which can be NONSEQUENTIAL, SEQUENTIAL, IDLE or BUSY.
HWRITE	Transfer direction	Input	When HIGH this signal indicates a write transfer, and when LOW, a read transfer.
HWDATA[31:0]	Write data bus	Input	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. However, this may easily be extended to allow for higher bandwidth operation.
HSELAPBif	Slave select	Input	Each APB slave has its own slave select signal, and this signal indicates that the current transfer is intended for the selected slave. This signal is a combinatorial decode of the address bus.
HRDATA[31:0]	Read data bus	Output	The read data bus is used to transfer data from bus slaves to the bus master during read operations. A minimum data bus width of 32 bits is recommended. However, this may easily be extended to allow for higher bandwidth operation.
HREADYin HREADYout	Transfer done	Input/Output	When HIGH the HREADY signal indicates that a transfer has finished on the bus. This signal may be driven LOW to extend a transfer.
HRESP[1:0]	Transfer response	Output	The transfer response provides additional information on the status of a transfer. This module will always generate the OKAY response.
PRDATA[31:0]	Peripheral read data bus	Input	The peripheral read data bus is driven by the selected peripheral bus slave during read cycles (when

			PWRITE is LOW)
PWDATA[31:0]	Peripheral write data bus	Output	The peripheral write data bus is continuously driven by this module, changing during write cycles (when PWRITE is HIGH)
PENABLE	Peripheral enable	Output	This enable signal is used to time all accesses on the peripheral bus. PENABLE goes HIGH on the second clock rising edge of the transfer, and LOW on the third (last) rising clock edge. of the transfer.
PSELx	Peripheral slave select	Output	There is one of these signals for each APB peripheral present in the system. 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.
PADDR[31:0]	Peripheral address bus	Output	This is the APB address bus, which may be up to 32 bits wide and is used by individual peripherals for decoding register accesses to that peripheral. The address becomes valid after the first rising edge of the clock at the start of the transfer. If there is a following APB transfer, then the address will change to the new value, otherwise it will hold its current value until the start of the next APB transfer..
PWRITE	Peripheral transfer direction	Output	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.

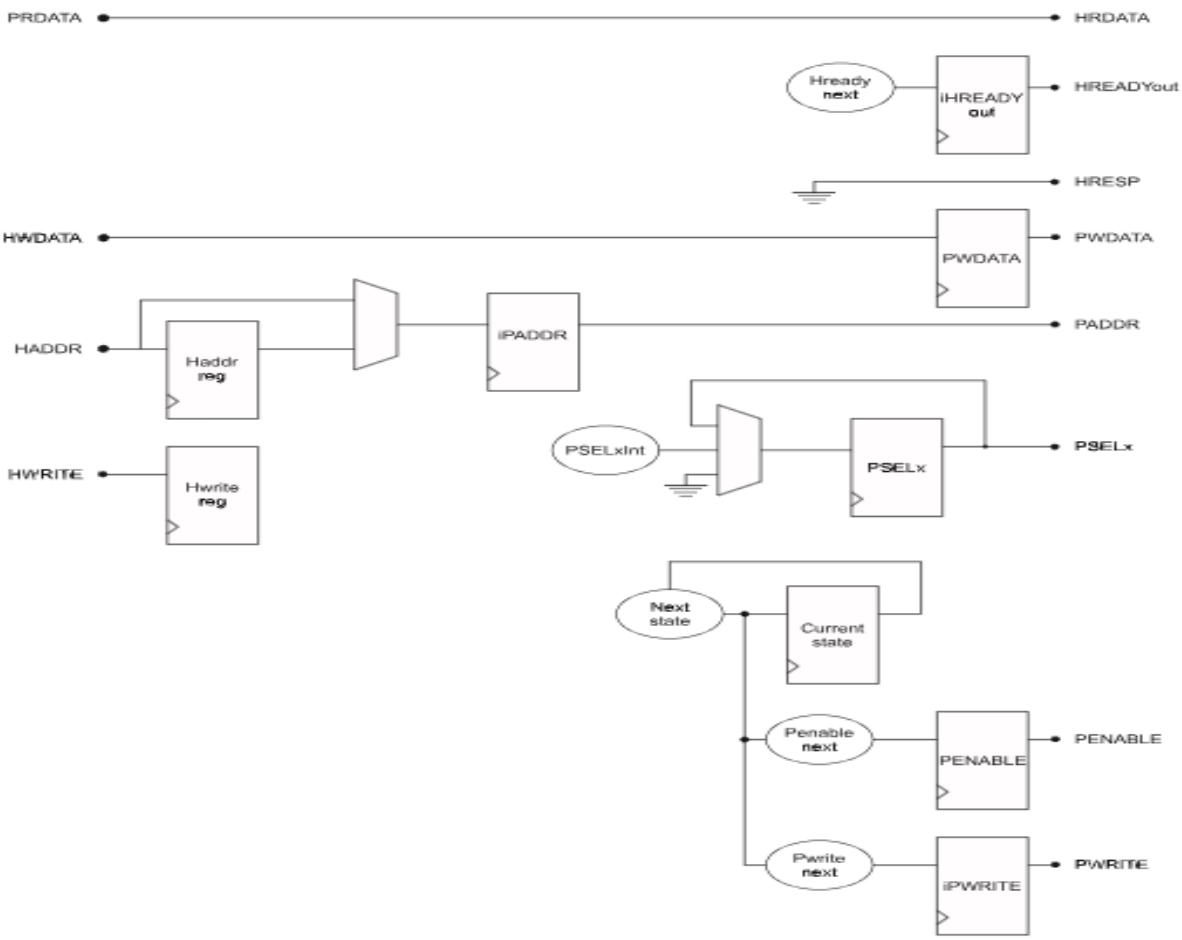
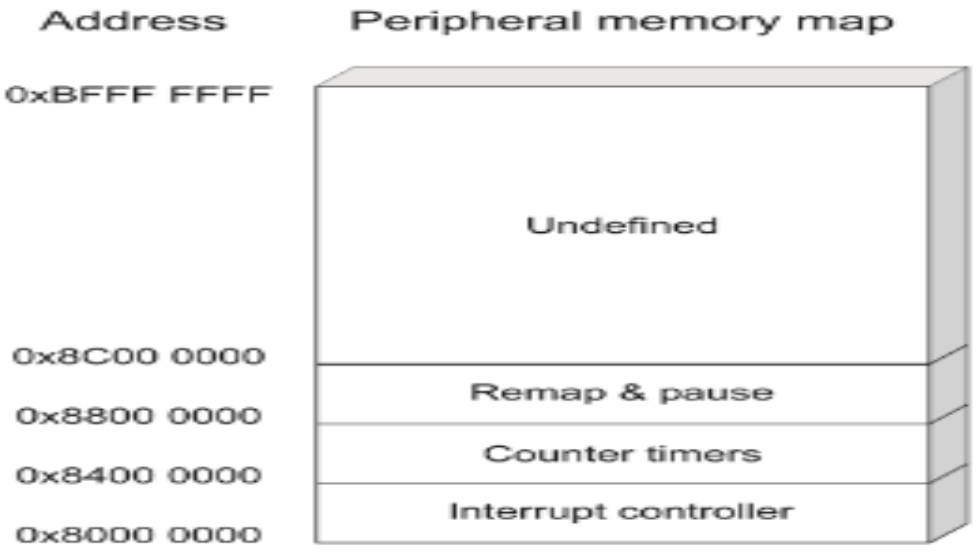


FIGURE: APB BRIDGE MODULE SYSTEM DIAGRAM

4.3 PERIPHERAL MEMORY MAP

APB bridge controls the memory map for the peripherals, and generates a select signal for each peripheral. The default system memory map is given below:



4.4 FUNCTIONING AND OPERATION

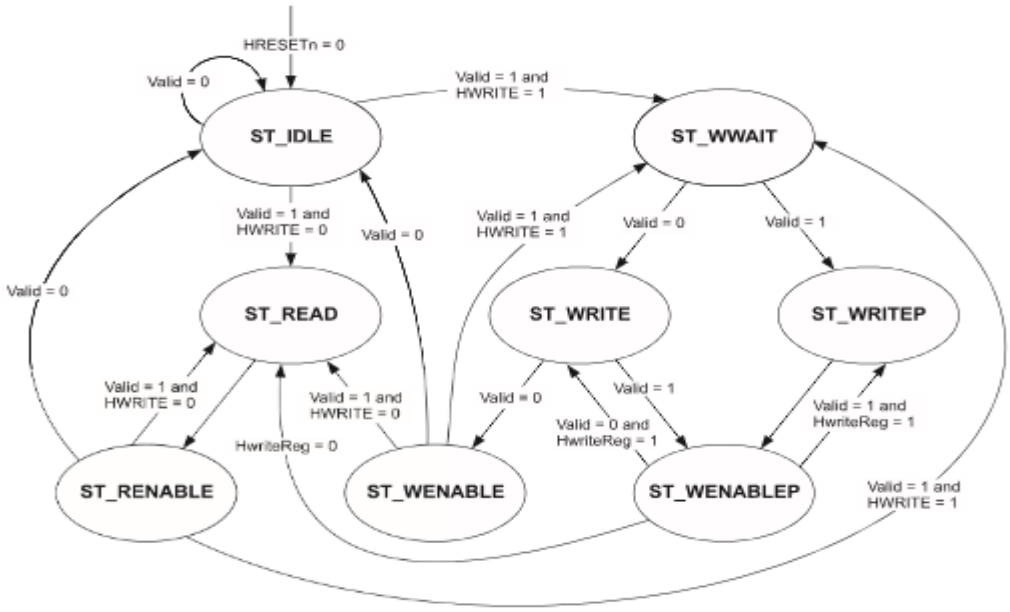


FIGURE: State machine for AHB to APB interface

- ST_IDLE: During this state the APB buses and PWRITE are driven with the last values they had, and PSEL and PENABLE lines are driven LOW. The ST_IDLE state is entered from: reset, when the system is initialized ST_RENABLE, ST_WENABLE, or ST_IDLE, when there are no peripheral transfers to perform. The next state is: 1. ST READ, for a read transfer, when the AHB contains a valid APB read transfer 2. ST_WWAIT, for a write transfer, when the AHB contains a valid APB write transfer.
- ST_READ: During this state the address is decoded and driven onto PADDR, the relevant PSEL line is driven HIGH and PWRITE is driven LOW. A wait state is always inserted to ensure that the data phase of the current AHB transfer does not complete until the APB read data has been driven onto HRDATA. The ST READ state is entered from ST_IDLE, ST RENABLE, ST_WENABLE, or ST_WENABLEP during a valid read transfer. The next state will always be ST_RENABLE.
- ST_WWAIT: This state is needed due to the pipelined structure of AHB transfers, to allow the AHB side of the write transfer to complete so that the write data becomes available on HWDATA. The APB write transfer is then started in the next clock cycle. The ST WWAIT state is entered from ST_IDLE, ST RENABLE, or ST WENABLE, during a valid write transfer. The next state will always be ST_WRITE.
- ST_WRITE: During this state the address is decoded and driven onto PADDR, the relevant PSEL line is driven HIGH, and PWRITE is driven HIGH. A wait state is not inserted, as a single write transfer can complete without affecting the AHB. The ST WRITE state is entered from: 1. ST_WWAIT, when there are no further peripheral transfers to perform 2. ST_WENABLEP, when the currently pending peripheral transfer is a write, and there are no further transfers to perform. The next state is: 1. ST_WENABLE, when there are no further peripheral transfers to perform 2. ST_WENABLEP, when there is one further peripheral write transfer to perform.

- ST_WRITEP: During this state the address is decoded and driven onto PADDR, the relevant PSEL line is driven HIGH, and PWRITE is driven HIGH. A wait state is always inserted, as there must only ever be one pending transfer between the currently performed APB transfer and the currently driven AHB transfer. The ST WRITEP state is entered from: 1. ST_WWAIT, when there is a further peripheral transfer to perform. 2. ST_WENABLEP, when the currently pending peripheral transfer is a write, and there is a further transfer to perform.
- ST_RENABLE: During this state the PENABLE output is driven HIGH, enabling the current APB transfer. All other APB outputs remain the same as the previous cycle. The ST_RENABLE state is always entered from ST_READ. The next state is: 1. ST READ, when there is a further peripheral read transfer to perform 2. ST_WWAIT, when there is a further peripheral write transfer to perform 3. ST_IDLE, when there are no further peripheral transfers to perform.
- ST WENABLE: During this state the PENABLE output is driven HIGH, enabling the current APB transfer. All other APB outputs remain the same as the previous cycle. The ST_WENABLE state is always entered from ST_WRITE.. The next state is: 1. ST_READ, when there is a further peripheral read transfer to perform 2. ST_WWAIT, when there is a further peripheral write transfer to perform 3. ST_IDLE, when there are no further peripheral transfers to perform.
- ST WENABLEP: A wait state is inserted if the pending transfer is a read because, when a read follows a write, an extra wait state must be inserted to allow the write transfer to complete on the APB before the read is started. The ST WENABLEP state is entered from: 1. ST_WRITE, when the currently driven AHB transfer is a peripheral transfer 2. ST_WRITEP, when there is a pending peripheral transfer following the current write. The next state is: 1. ST_READ, when the pending transfer is a read 2. ST_WRITE, when the pending transfer is a write, and there are no further transfers to perform 3. ST_WRITEP, when the pending transfer is a write, and there is a further transfer to perform.

4.5 SYSTEM DESCRIPTION

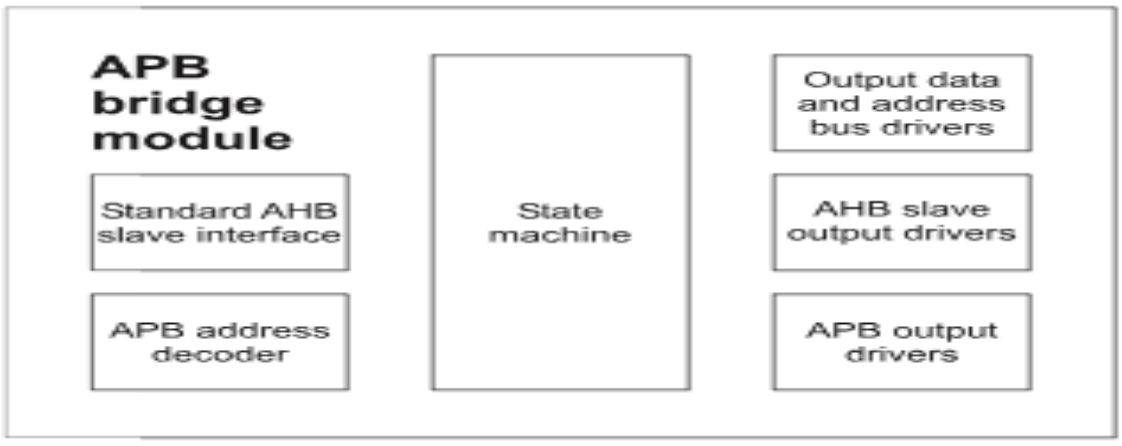


FIGURE: APB bridge module block diagram

AHB slave bus interface: This module uses the standard AHB slave bus interface, which comprises:

- the valid transfer detection logic which is used to determine when a valid transfer is accessing the slave
- the address and control registers, which are used to store the information from the address phase of the transfer for use in the data phase.
- Due to the different AHB to APB timing of read and write transfers, either the current or the previous address input value is needed to correctly generate the APB transfer. A multiplexor is therefore used to select between the current address input or the registered address, for read and write transfers respectively.

APB transfer state machine: The transfer state machine is used to control the application of APB transfers based on the AHB inputs. The state diagram shows the operation of the state machine, which is controlled by its current state and the AHB slave interface signals.

APB output signal generation: The generation of all APB output signals is based on the status of the transfer state machine:

- PWDATA is a registered version of the HWDATA input, which is only enabled during a write transfer. As the bridge is the only bus master on the APB, then it can drive PWDATA continuously.
- PENABLE is only set HIGH during one of three enable states, in the last cycle of an APB transfer. A register is used to generate this output from the next state of the transfer state machine.
- The PSELx outputs are decoded from the current transfer address. They are only valid during the read, write and enable states, and are all driven LOW at all other times so that no peripherals are selected when no transfers are being performed.
- PADDR is a registered version of the currently selected address input (HADDR or the address register) and only changes when the read and write states are entered at the start of the APB transfer.
- PWRITE is set HIGH during a write transfer, and only changes when a new APB transfer is started. A register is used to generate this output from the next state of the transfer state machine.
- The APBen signal is used as an enable on the PSEL, PWRITE and PADDR output registers, ensuring that these signals only change when a new APB transfer is started, when the next state is ST_READ, ST_WRITE, or ST_WRITEP

AHB output signal generation: A standard AHB slave interface consists of the following three outputs:

- HRDATA is directly driven with the current value of PRDATA. APB slaves only drive read data during the enable phase of the APB transfer, with PRDATA set LOW at all other times, so bus clash is avoided on HRDATA (assuming OR bus connections for both the AHB and APB read data buses).
- HREADYout is driven with a registered signal to improve the output timing. Wait states are inserted by the APB bridge during the ST_READ and ST_WRITEP states, and during the ST_WENABLEP state when the next transfer to be performed is a read.
- HRESP is continuously held LOW, as the APB bridge does not generate SPLIT, RETRY or ERROR responses.

CHAPTER 5

OUTPUTS

5.1 SYNTHESIS OUTPUT OF BRIDGE MODULE

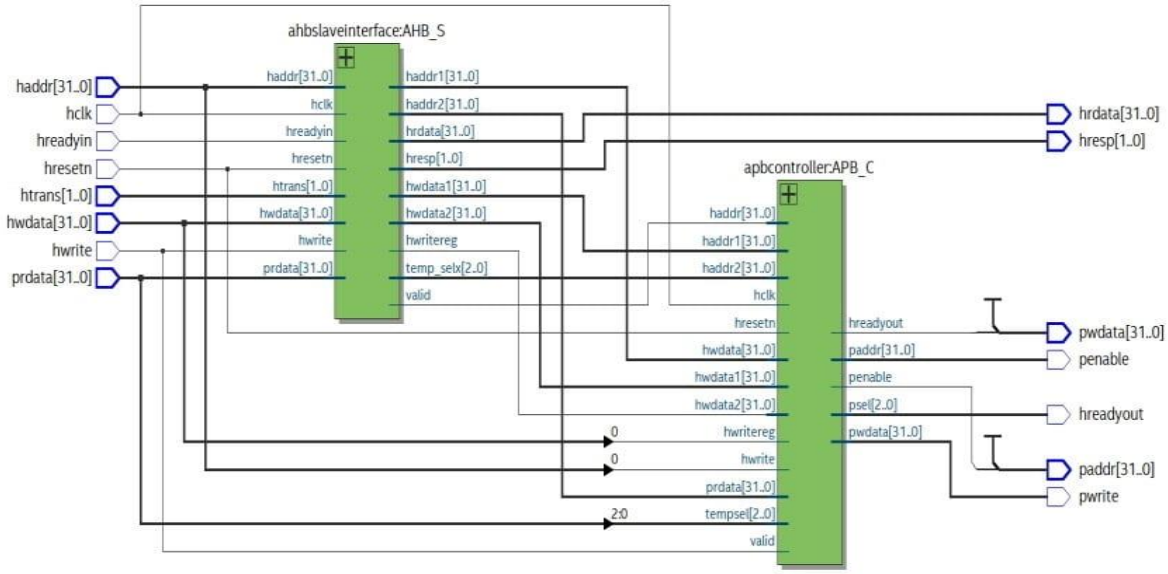
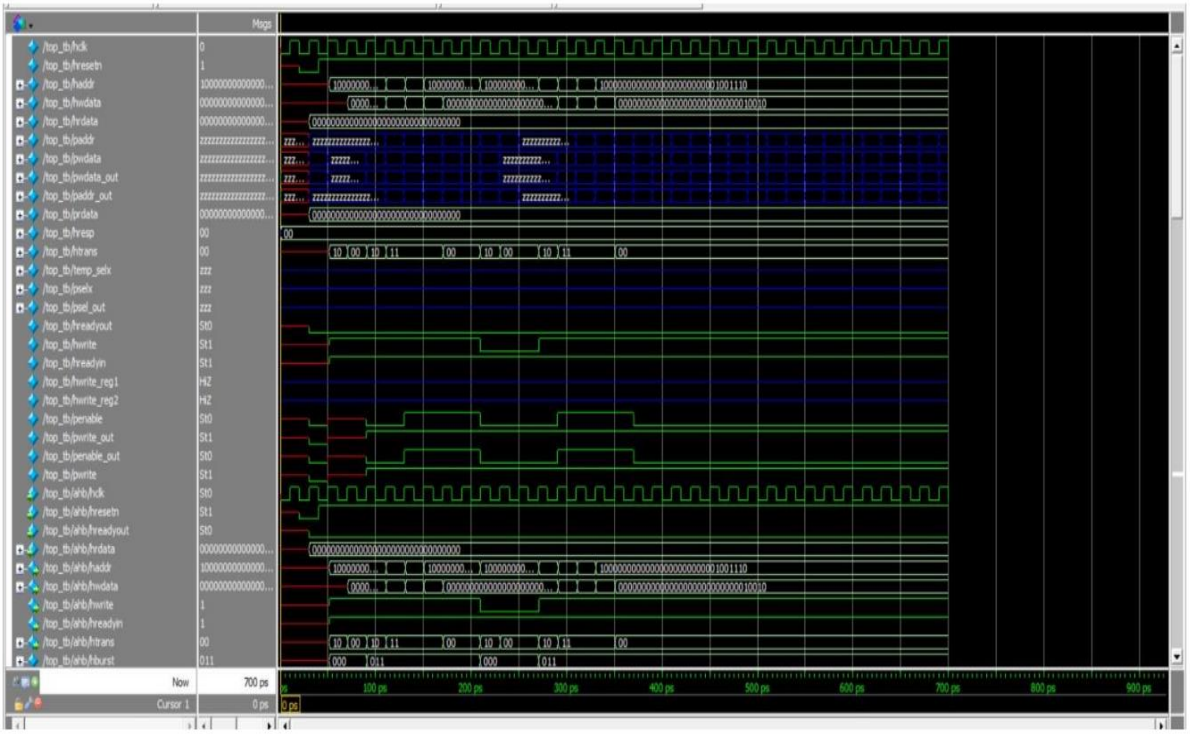
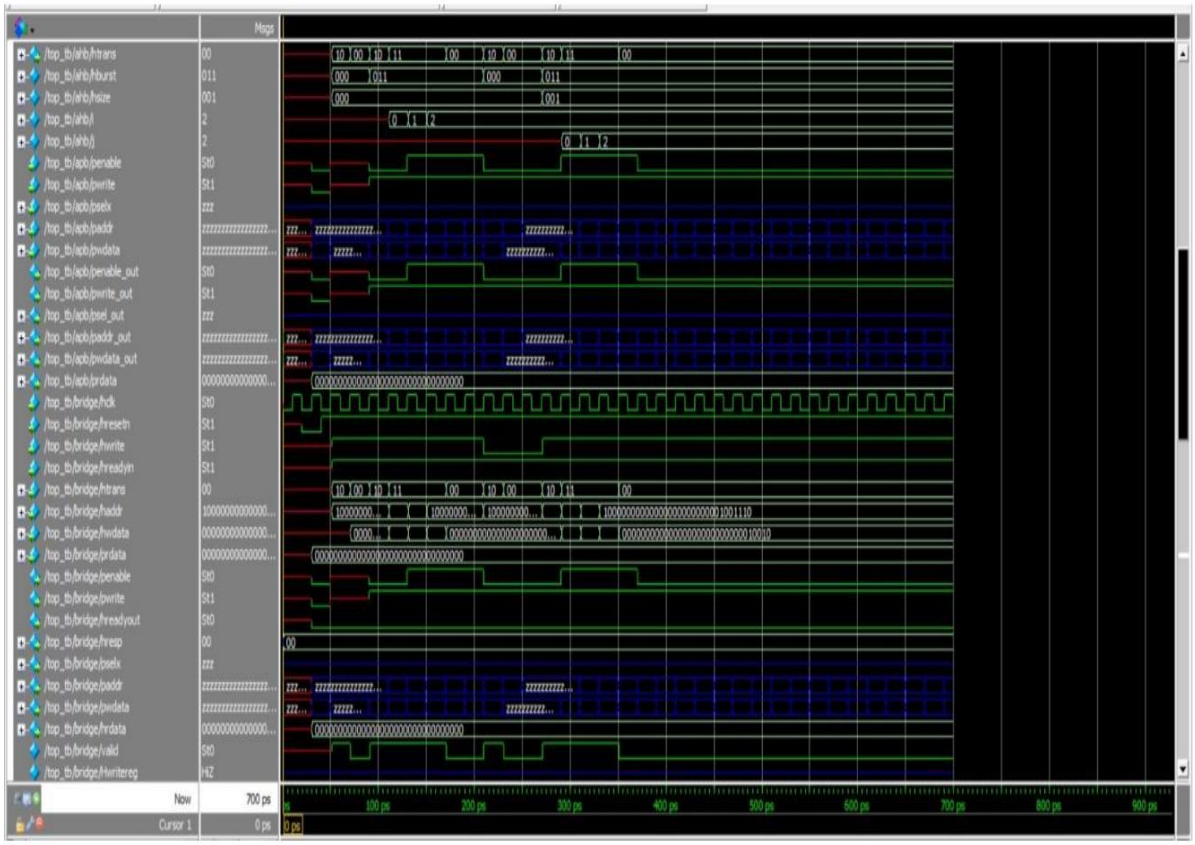
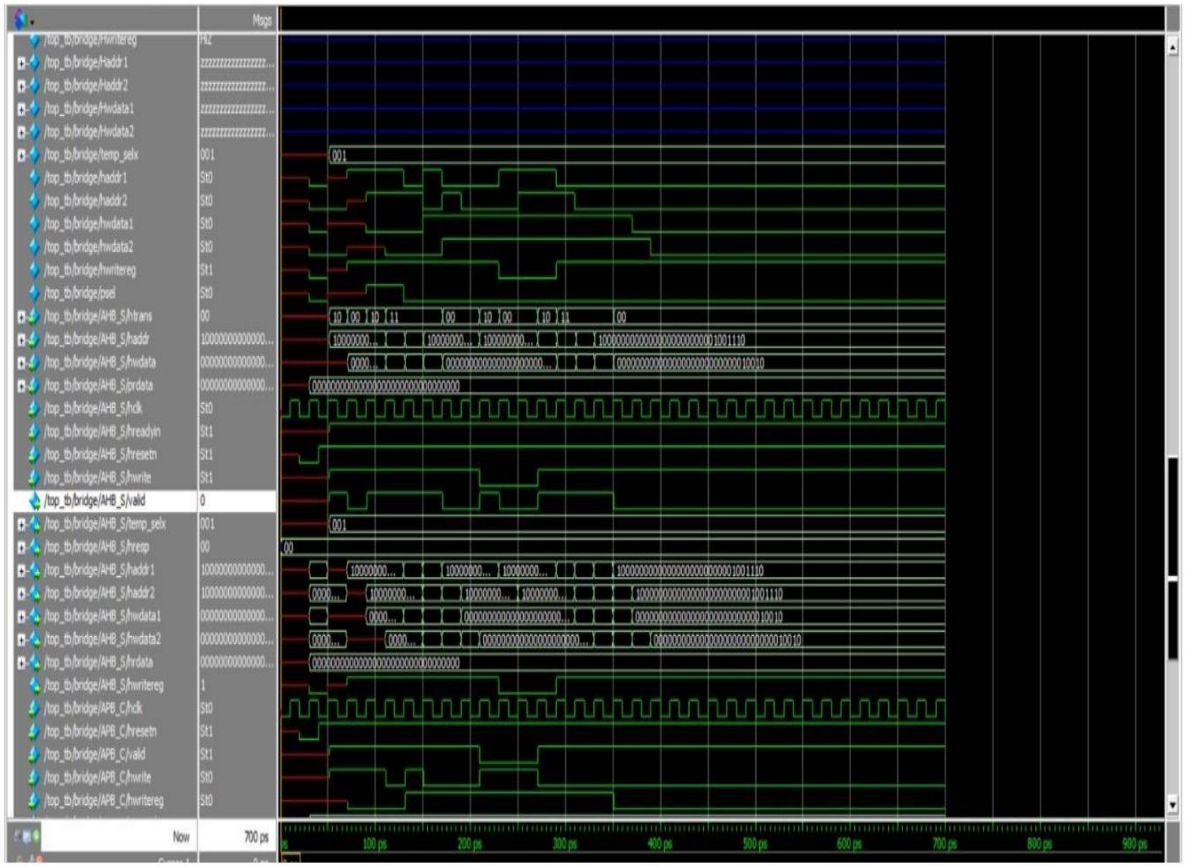


FIGURE: Synthesis output of the top testbench of the bridge module designed in MODELSIM (output obtained from QuartusPrime)

5.2 SIMULATION OUTPUTS




/top_tb/ahb/htrans
/top_tb/ahb/hburst
/top_tb/ahb/hsize
/top_tb/ahb/i
/top_tb/ahb/j
/top_tb/apb/penable
/top_tb/apb/pwrite
/top_tb/apb/pselx
/top_tb/apb/paddr
/top_tb/apb/pwdata
/top_tb/apb/penable_out
/top_tb/apb/pwrite_out
/top_tb/apb/psel_out
/top_tb/apb/paddr_out
/top_tb/apb/pwdata_out
/top_tb/apb/prdata
/top_tb/bridge/hclk
/top_tb/bridge/hresetn
/top_tb/bridge/hwrite
/top_tb/bridge/hreadyin
/top_tb/bridge/htrans
/top_tb/bridge/haddr
/top_tb/bridge/hwdata
/top_tb/bridge/prdata
/top_tb/bridge/penable
/top_tb/bridge/pwrite
/top_tb/bridge/hreadyout
/top_tb/bridge/hresp
/top_tb/bridge/pselx
/top_tb/bridge/paddr
/top_tb/bridge/pwdata
/top_tb/bridge/hrdata
/top_tb/bridge/valid
/top_tb/bridge/hwritereg
Msgs
Now
700 ps
Cursor 1
0 ps
100 ps
200 ps
300 ps
400 ps
500 ps
600 ps
700 ps
800 ps
900 ps


/top_tb/bridge/hwritereg
/top_tb/bridge/haddr1
/top_tb/bridge/haddr2
/top_tb/bridge/hwdata1
/top_tb/bridge/hwdata2
/top_tb/bridge/temp_selx
/top_tb/bridge/haddr1
/top_tb/bridge/haddr2
/top_tb/bridge/hwdata1
/top_tb/bridge/hwdata2
/top_tb/bridge/hwritereg
/top_tb/bridge/psel
/top_tb/bridge/AHB_S/htrans
/top_tb/bridge/AHB_S/haddr
/top_tb/bridge/AHB_S/hwdata
/top_tb/bridge/AHB_S/prdata
/top_tb/bridge/AHB_S/hclk
/top_tb/bridge/AHB_S/hreadyin
/top_tb/bridge/AHB_S/hresetn
/top_tb/bridge/AHB_S/hwrite
/top_tb/bridge/AHB_S/valid
/top_tb/bridge/AHB_S/temp_selx
/top_tb/bridge/AHB_S/hresp
/top_tb/bridge/AHB_S/haddr1
/top_tb/bridge/AHB_S/haddr2
/top_tb/bridge/AHB_S/hwdata1
/top_tb/bridge/AHB_S/hwdata2
/top_tb/bridge/AHB_S/hrdata
/top_tb/bridge/AHB_S/hwritereg
/top_tb/bridge/APB_C/hclk
/top_tb/bridge/APB_C/hresetn
/top_tb/bridge/APB_C/valid
/top_tb/bridge/APB_C/hwrite
/top_tb/bridge/APB_C/hwritereg
Msgs
Now
700 ps
100 ps
200 ps
300 ps
400 ps
500 ps
600 ps
700 ps
800 ps
900 ps

CHAPTER 6

CONCLUSION

6.1 CONCLUSION

The internship at Maven Silicon has been a transformative experience, providing me with an in-depth understanding of digital system design, VLSI technology, and the System-on-Chip (SoC) design process. During my time at Maven Silicon, I was thoroughly trained in fundamental concepts such as Verilog programming, digital logic design, and VLSI design methodologies, which form the backbone of modern semiconductor technology. This solid theoretical foundation enabled me to take on the practical challenges of designing complex digital systems.

One of the key highlights of my internship was the opportunity to work on a real-world project: the AHB to APB Bridge Design. This project aimed to develop a bridge that facilitates efficient communication between the high-speed Advanced High-performance Bus (AHB) and the low-power Advanced Peripheral Bus (APB), which are integral parts of the AMBA (Advanced Microcontroller Bus Architecture). The goal was to ensure seamless data transfer while maintaining low latency and power efficiency, critical requirements in embedded systems and SoC applications.

The project was structured in a step-by-step approach under the expert mentorship of the Maven Silicon team. This phased approach allowed me to gradually build my skills, starting from understanding the basics of bus protocols to implementing and optimizing the design. I was guided through the process of designing the AHB Slave Interface, which involved address decoding and transaction management, and the APB Controller, which focused on converting AHB signals into APB-compatible transactions.

Throughout the project, I faced various challenges, such as ensuring the correct synchronization between the AHB and APB buses, managing different data transfer modes, and optimizing the bridge for both performance and power consumption. However, with the continuous support and insightful feedback from my mentors, I was able to overcome these challenges, thereby gaining a deeper understanding of digital design and debugging techniques.

Moreover, the hands-on experience of using industry-standard tools like ModelSim for simulation and Quartus Prime for synthesis provided me with practical exposure to the design flow used in semiconductor industries. Writing testbenches, performing simulations, and analyzing waveforms to validate the design functionality were key aspects that helped me bridge the gap between theoretical knowledge and practical application.

The internship not only enhanced my technical proficiency but also developed my problem-solving and critical thinking abilities. It taught me the importance of meticulous design practices, attention to detail, and the value of iterative testing and optimization in achieving a robust solution. I also learned how to collaborate effectively within a professional environment, which is crucial for success in the semiconductor industry.

In conclusion, my experience at Maven Silicon has been incredibly rewarding. It has provided me with a strong foundation in digital system design and a clear understanding of the semiconductor design process. The knowledge and skills I gained during this internship have equipped me to take on future challenges in the field of VLSI and SoC design, and I am confident that this experience will serve as a valuable stepping stone in my career as a hardware design engineer. I am grateful to the entire team at Maven Silicon for their mentorship, support, and the opportunity to work on a project of such significance.

Thank you to everyone at Maven Silicon for making my internship journey a memorable and enriching one.

Sincerely,

Pradesh Kumar M