Design of AHB2APB Bridge

A

Internship Report

Submitted in partial fulfillment for the award of the degree of

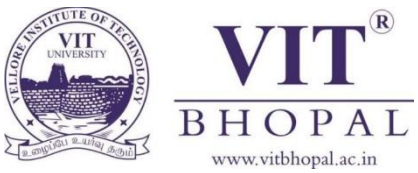
Bachelor of Technology

in

ELECTRONICS AND COMMUNICATION ENGINEERING (AI & CYBERNETICS)

Submitted to

VIT BHOPAL UNIVERSITY (M.P.)



Submitted by

VARUN RAM S

(REGISTRATION NO. – 20BAC10038)

Under the Supervision of
Mr. MANJUNATH N.L
VLSI DESIGN TRAINER, MAVEN SILICON, BENGALURU

SCHOOL OF ELECTRICAL & ELECTRONICS ENGINEERING
VIT BHOPAL UNIVERSITY

BHOPAL (M.P.)-466114

May –2024

VIT BHOPAL UNIVERSITY BHOPAL (M.P.) 466114

SCHOOL OF ELECTRICAL & ELECTRONICS ENGG.

CANDIDATE'S DECLARATION

I hereby declare that the Internship project entitled "Design of AHB2APB Bridge" is my own work conducted under the Supervision of Mr. Manjunath N.L, Academic Trainer, Maven Silicon.

I further declare that to the best of my knowledge, this report does not contain any part of the work that has been submitted for the award of any degree either in this university or other university / Deemed University without proper citation.

Varun Ram S

20BAC10038

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:17th May 2024

Mr. Manjunath N.L

Trainer, Maven Silicon Bengaluru

VIT BHOPAL UNIVERSITY (M.P.) – 466114

SCHOOL OF ELECTRICAL & ELECTRONICS ENGG.

CERTIFICATE

This is to certify that the work embodied in this Internship report entitled **"Design of AHB2APB Bridge"** has been satisfactorily completed by **Mr. Varun Ram S**, Registration No. 20BAC10038 in the School of Electrical & Electronics Engineering of VIT University, Bhopal. This work is a bonafide piece of work, carried out under our guidance at Maven Silicon, Bengaluru for the partial fulfillment of the degree of Bachelor of Technology.

Forwarded by

Dr. Soumitra K Nayak
Program Chair

Approved by

Dr.M Suresh
Professor & Dean

COMPANY CERTIFICATE

MS/VIT-INT/2023-264

aceic™ Design Partner

MAVEN SILICON Centre of Excellence in VLSI

TO WHOMSOEVER IT MAY CONCERN

This is to certify that Varun Ram S, a student of B.Tech ECE, 2024 from Vellore Institute of Technology, Bhopal has successfully completed his VLSI Design Internship Program from 15/05/2023 to 07/07/2023

During his internship program with us, he worked on the below mentioned project.
AHB2APB bridge design using Verilog HDL.

We wish him all the best in his future endeavors.

For Maven Silicon Softech Ltd.

Sweety Dharamdasani
Head - Learning & Development

Place: Bangalore
Date: 24/07/2023

Maven Silicon Softech Pvt Ltd.,
Registered office : # 21/1A, III Floor, MS Plaza, Gottigere, Uttarahalli Hobli, South Taluk
Bannerghatta Road, Bangalore - 560076
CIN No.:U72200KA2010PTC052736 Phone : +91 7406709555 Email : hr@maven-silicon.com www.maven-silicon.com

Acknowledgement

In the first place, I would like to record my gratitude to my Project Guide and Instructor, **Mr. Manjunath NL** , for his supervision, advice and guidance from the very start of the project as well as giving an extraordinary experience throughout the work. Above all and most needed, his guidance was beyond amazing for completing this project work on "Designing AHB2APB Bridge".

I would also like to extend my gratitude to **Mr. Sivakumar PR** , CEO and Founder of Maven Silicon for his guidance and teaching in the Verilog Programming lectures. I'd also not like to forget to thank the Support team, instructors and other staff members of Maven Silicon who were crucial in solving queries and offered great support by timely responses in the email and WhatsApp group.

I am also grateful to my Program Chair **Dr. Soumitra K Nayak** and Dean **Dr. M Suresh** for their continuous support and motivatiuon during the period of the internship. I'm also thankful to my former dean and current registrar**, Dr. Debashis Adhikari** who always gave us awareness about the emerging fields of VLSI and Core Electronics and was always happy and supportive for internships such as this one. The other teachers of SEEE and my Vice Chancellor **Dr. Senthil Kumar Arumugham** also played an important role in my academic journey during and after this internship.

Varun Ram S

20BAC10038

Executive Summary

The design and implementation of an Advanced High-performance Bus (AHB) to Advanced Peripheral Bus (APB) bridge will be thoroughly discussed in this study. The main goal of this project is to provide an effective interface between the AHB and APB buses that will allow high-performance devices linked through the APB bus to exchange data and control signals. This study will start by providing a thorough block diagram of the top module, which symbolizes the main structure of the AHB to APB bridge. The block diagram will provide a visual representation of the overall design by highlighting the important elements and how they link to one another. The detailed explanation of each sub-block inside the bridge will outline the precise function it performs in facilitating communication between the AHB and APB buses. The output waveform, which shows the signals and data flowing through the AHB to APB bridge during operation, will also be examined in this report. In order to assess the bridge's efficacy in allowing communication between the high-performance devices linked through the APB bus, a greater knowledge of the timing, synchronization, and overall performance of the bridge is made possible by the waveform analysis. This report will serve as a complete reference for comprehending the project's goals and results by meticulously outlining the design, architecture, and functionality of the AHB to APB bridge. It will offer insightful information about the complex workings of the bridge, assisting in the assessment of its efficiency, dependability, and general performance.

List of Figures

List of Tables

List of Abbreviations

AHB	**Advanced High Performance Bus**
APB	**Advanced Peripheral Bus**
ASB	**Advanced System Bus**
AMBA	**Advanced Microprocessor Bus Architecture**
FIFO	**First In First Out**
SoC	**System on Chip**
UVM	**Universal Verificaiton Module**

TABLE OF CONTENTS

CHAPTER 1: INTROUDCTION

One of the biggest challenges in the field of digital system design is integrating disparate components in an effective and seamless manner. In order to enable communication and interoperability across different subsystems within a system-on-chip (SoC) architecture, the Advanced High-performance Bus (AHB) and the Advanced Peripheral Bus (APB) must be bridged. This is where the AHB2APB Bridge comes into play.

1.1 Aim of the Project

The goal of this project is to create a bridge that connects the Advanced Peripheral bus (APB) and the Advanced High-performance bridge (AHB), two separate buses that are specified in the Advanced Bus Microcontroller Architecture (AMBA) Specification. In actuality, the bridge permits communication between low-peripheral and high-performance devices.

1.2 Objectives of the Project

The project also covers a few objectives and takeaways for an optimal output:

- Study the Top Module Block Diagram
- Write Verilog code for different modules of the APB2AHB Bridge
- Verify the same using a Test Bench.
- Generate relevant output waveforms for better visualization and understanding.

1.3 Software and Sythesis tools used

The project requires a basic understanding and implementation of the following tools/software to proceed with the design:

1) HDL Used: Verilog
2) Simulator Tool Used: ModelSIM
3) Synthesis Tool Used: Quartus Prime

1.4 About the Interning Organization: Maven Silicon

One of Bangalore, India's top VLSI training facilities is Maven Silicon. VLSI Design, System Verilog, UVM, Verilog, and ASIC Verification are their areas of expertise. Additionally, they support applicants' training for positions in the rapidly expanding VLSI Design Market.

Figure 1.1: Maven Silicon Logo

CHAPTER 2: SIGNIFICANCE OF VERILOG

Verilog is commonly used in the design of AHB to APB bridges due to its capability to describe hardware behaviour at different abstraction levels. Here's how Verilog is utilized in the AHB to APB bridge design:

- *Module Definition*: Verilog modules define the AHB to APB bridge functionality, encapsulating its behavior and interface. This module typically includes ports for AHB signals (such as address, data, control) and APB signals (such as address, data, control), as well as clock and reset signals.

- *Behavioral Modeling*: Verilog behavioral modeling is used to describe the bridge's functionality. This includes decoding AHB transactions, generating corresponding APB transactions, and managing data transfer between the two buses. Behavioral blocks are defined to handle various AHB and APB transfer types.

- *Structural Modeling*: Structural modeling in Verilog is employed to instantiate lower-level modules or components within the bridge design. These components may include FIFOs, state machines, and control logic necessary for protocol conversion and bus arbitration.

- *Simulation and Verification*: Verilog testbenches are developed to simulate the AHB to APB bridge design under different scenarios and stimuli. Testbenches generate AHB transactions, apply them to the bridge input ports, and verify the correctness of APB transactions generated by the bridge.

- *Synthesis and Implementation*: Verilog code is synthesized using hardware synthesis tools to generate the gate-level netlist for the AHB to APB bridge design. This netlist is then used for implementation on FPGA or ASIC devices.

- *Timing Analysis*: Verilog designs undergo timing analysis to ensure that the bridge meets timing constraints and operates correctly at the desired clock frequency. Timing constraints are specified to ensure proper synchronization between AHB and APB buses.

- *Hierarchy and Reusability*: Verilog allows for hierarchical design, enabling the AHB to APB bridge to be structured into modular components. This promotes reusability and facilitates easier maintenance and debugging of the design.

CHAPTER 3: INTRODUCTION TO AMBA BUSES

For the purpose of creating high-performance embedded microcontrollers, an on-chip communications standard is defined by the Advanced Microcontroller Bus Architecture (AMBA) specification.

Three distinct buses are defined within the AMBA specification:

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

The AMBA standard includes a test methodology that offers a diagnostic and test infrastructure for modular macrocells.

3.1 Advanced High Performance Bus (AHB)

For high-performance, high clock frequency system modules, use the AMBA AHB. The high-performance system backbone bus is the AHB. With low-power peripheral macrocell operations, AHB facilitates the efficient connectivity of CPUs, on-chip memories, and off-chip external memory interfaces. Additionally, AHB is designed to be user-friendly in an effective design flow through the use of automated test procedures and synthesis.

3.2 Advanced System Bus (ASB)

For high-performance system modules, use the AMBA ASB.
An alternate system bus, AMBA ASB, can be used in situations when AHB's high-performance capabilities are not needed. Additionally, ASB facilitates the low-power peripheral macrocell operations' effective connectivity to processors, on-chip memories, and off-chip external memory interfaces.

3.3 Advanced Peripheral Bus (APB)

This is a low-power peripheral AMBA APB.
To enable peripheral functions, AMBA APB is designed with low power consumption and simplified interface complexity in mind. It is possible to utilise APB with either system bus version.

3.4 Objectives of the AMBA Specification and Comtrollers

Four essential needs have been met in the development of the AMBA specification:

- To be technology-independent and ensure that highly reusable peripheral and system macrocells can be migrated across a diverse range of IC processes and be appropriate for full-custom, standard cell, and gate array technologies;
- To encourage modular system design to improve processor independence, providing a development road-map for advanced cached CPU cores and the development of peripheral libraries;

- To promote the development of modular systems to increase processor independence by offering a roadmap for the advancement of sophisticated caching CPU cores and the creation of peripheral libraries.
- To reduce the amount of silicon infrastructure needed for effective on-chip and off-chip connectivity throughout manufacturing testing and operation.

An AMBA-based microcontroller often includes a high-performance system backbone bus (AMBA AHB or AMBA ASB) that may support external memory bandwidth and houses the CPU, on-chip memory, and other Direct Memory Access (DMA) devices. This bus offers a high-bandwidth link between the parts involved in the vast majority of transfers. A bridge to the lower bandwidth APB, which houses the majority of the system's peripheral devices, is also present on the high-performance bus.

3.5 Introduction to the AMBA AHB

AHB is a new generation of AMBA buses designed to meet the needs of high-performance synthesizable designs. It is a high-performance system bus capable of supporting multiple bus masters and operating at high bandwidths.

AMBA AHB incorporates the features necessary for high-performance, high clock frequency systems, such as:

- 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 present ASB/APB may be completed swiftly, allowing any existing designs to be simply incorporated.
An AMBA AHB design may have one or more bus masters, whereas a system usually includes at least the processor and test interface. It is also typical to incorporate a Direct Memory Access (DMA) or Digital Signal Processor (DSP) as bus masters.
The most typical AHB slaves are the external memory interface, an APB bridge, and any internal memory. Any other peripheral in the system might be configured as an AHB slave. However, low-bandwidth peripherals are often located on the APB.

A typical AMBA AHB system design contains the following components:

AHB Master: A bus master can commence read and write operations by specifying an address and control information. Only one bus master is permitted to actively operate the vehicle at any given time.

AHB slave: A bus slave responds to read or write operations within a specific address space range. The bus slave notifies the active master of the data transfer's success, failure, or waiting status.

AHB arbitrer: The bus arbiter guarantees that only one bus master at a time can begin data transfers. Regardless of whether the arbitration protocol is established, any arbitration method, such as highest priority or fair access, can be applied based on the application needs.
An AHB would only have one arbiter, which would be easy in single-bus master systems.

AHB decoder: The AHB decoder is used to decode the address of each transfer and provide a select signal to the slave engaged in the transfer. All AHB implementations require a single, centralised decoder.

3.6 Introduction to AMBA APB

The APB is part of the AMBA bus hierarchy and is designed to consume as little power as possible while maintaining interface simplicity.
The AMBA APB appears as a local secondary bus that is contained within a single AHB or ASB slave device. APB adds low-power functionality to the system bus by directly building on AHB or ASB signals.
The APB bridge appears as a slave module, performing the bus handshake and control signal retiming on behalf of the local peripheral bus. By designing the APB interface from the system bus's beginning point, the benefits of system diagnostics and test methods may be realised.
The AMBA APB should be used to connect low bandwidth peripherals that do not need the high performance of a pipelined bus interface.
The newest iteration of the APB specifies that all signal transitions are exclusively connected to the clock's rising edge. This innovation assures that the APB peripherals may be simply incorporated into any design flow, with the following benefits:

- 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.

An AMBA APB implementation generally includes a single APB bridge, which is necessary to transform AHB or ASB transfers into a format that the slave devices on the APB understand. The bridge latches all address, data, and control signals and performs a second level of decoding to provide slave select signals for 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).

3.7 System and Peripheral Bus

Building all peripherals as fully working AHB or ASB modules is possible, but may not always be desired.

- In systems with a large number of peripheral macrocells, increasing bus loading may result in increased power dissipation and reduced performance.
- When timing analysis is necessary, the slowest component on the bus will limit maximum performance.
- Many basic peripheral macrocells require latched addresses and control signals, but high-bandwidth macrocells benefit from pipelined signalling.
- Many peripheral operations just require a selection strobe to indicate macrocell selection and read/write bus activity, eliminating the need to broadcast the high-frequency clock signal to each peripheral.

3.8 Uses, Advantages and Dependencies

When talking about a full AHB or ASB, it is typically used for:

- Bus Masters
- On-Chip Memory Blocks
- High Bandwidth peripherals with FIFO Interfaces
- Extrernal Memory Interfaces
- DMA Slave Peripherals

A simple APB interface is kind of recommended for:

- Simple register mapped slave devices
- Very low power interfaces where clocks cannot be globally routed
- Grouping narrow-bus peripherals to avoid loading the system bus.

Some of the following dependencies should be taken into consideration when reading the AMBA specification:

Technology Impedance: AMBA is a technology-independent on-chip protocol. The standard solely describes the bus protocol at the clock cycle level.

Electrical Characteristics: No information on electrical properties is included in the AMBA standard because this is totally reliant on the manufacturing process technology used for the design.

Tming Specification: The AMBA protocol specifies the behaviour of several signals at the cycle level. The exact time requirements will vary depending on the process technology and frequency of operation. The AMBA protocol does not specify the exact timing requirements, hence the system integrator has complete freedom in allocating the signal timing budget among the various modules on the bus.

The below figures would give an idea of the overall structure of the controller intended to be designed in the project:

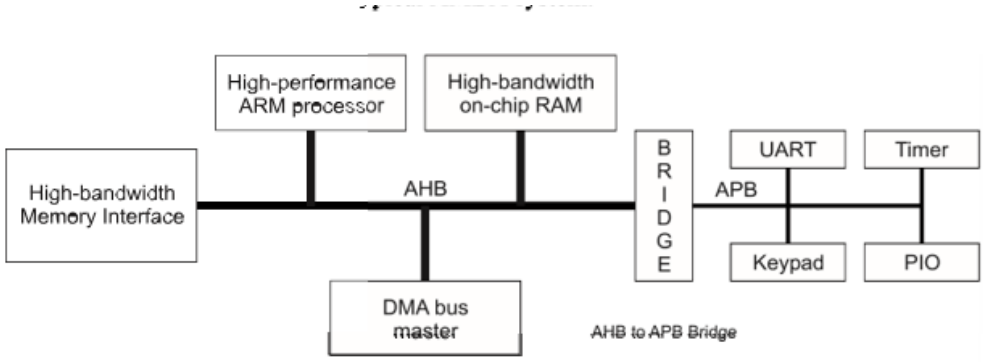


Figure 3.1: Block diagram of the AHB2APB Bridge

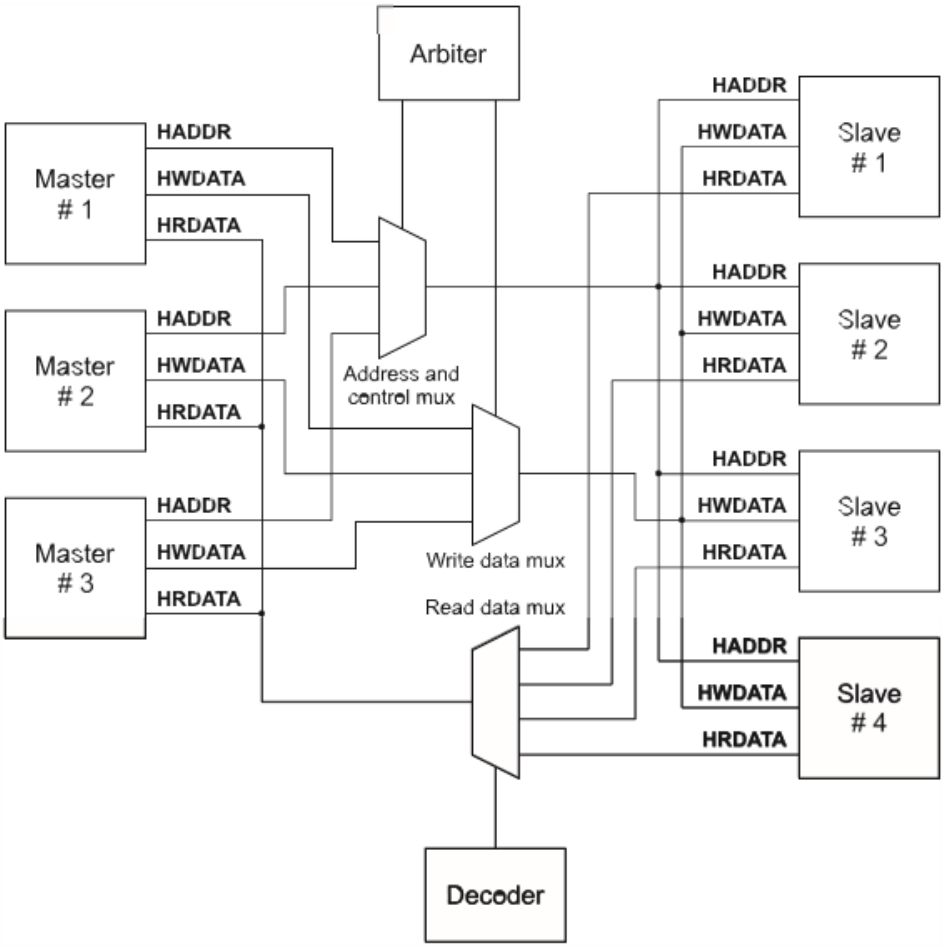


Figure 3.2: Multiplexer Interconnection for the AHB2APB Bridge

CHAPTER 4: AMBA SIGNALS

AMBA (Advanced Microcontroller Bus Architecture) signals are a standardized set of signals used for on-chip communication within System-on-Chip (SoC) designs. These signals facilitate the transfer of data, addresses, and control information between various components integrated on the same chip. Key AMBA signals include:

Clock and Reset Signals: Clock signals serve as a time reference for all system processes, whilst reset signals restore the system to a known condition after power-up or reset.

Address and Data Signals: Address signals provide information about the memory location or peripheral register being accessed, whilst data signals send the data being read from or written to the given address.

Read and Write Signals: Read signals send data from a peripheral or memory location to the requesting component, whereas write signals move data from the requesting component to the given address.

Control signals: They influence several elements of bus operation, such as data transfer readiness, component selection, and transaction type definition.

Burst Signals: Burst signals govern sequential data transfers by setting the burst type and size.

Error and Acknowledge Signals: Error detection and acknowledgment signals improve data integrity and dependability by signalling whether a bus transaction succeeded or failed.

The basic AMBA Signals are described in Table 4.1 and Table 4.2

Table 4.1: AHB Signals in the Project

Name	Source	Description
HWDATA Write data bus	Master	During write operations, the write data bus facilitates the transfer of data from the master to the bus slaves. It's advisable to have a minimum data bus width of 32 bits, but it can be expanded to support higher bandwidth operations if needed.
HSELx Slave select	Decoder	Every AHB slave possesses its individual slave select signal, which signifies that the ongoing transfer is targeted for the selected slave. Essentially, this signal is generated through a combinatorial decoding of the address bus.
HRDATA 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 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.

Table 4.2: APB Signals

Name	**Description**
PCLK Bus clock	The rising edge of PCLK is used to time all the transfers on 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 5: AMBA AHB

Before an AMBA AHB transfer can begin, the bus master must have access to the bus. This procedure begins with the master sending a request signal to the arbiter. The arbitrator then specifies when the master will be permitted access to the bus. A permitted bus master initiates an AMBA AHB transfer by driving the address and control signals. These signals indicate the address, direction, and breadth of the transfer, as well as whether it is part of a burst. Two distinct types of burst transfers are allowed:

- Incrementing bursts, which do not wrap at address boundaries
- Wrapping bursts, which wrap at particular 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
- One or More cycles for the data

The address cannot be elongated, necessitating all slaves to sample it during this period. However, data can be extended by utilizing the HREADY signal. When set to LOW, this signal triggers the insertion of wait states during the transfer, providing additional time for the slave to either supply 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 functioning, a master is permitted to complete all transfers within a certain burst before the arbiter offers another master access to the bus. However, to minimise excessive arbitration latencies, the arbiter may break up a burst, in which case the master must re-arbitrate for the bus in order to finish the remaining transfers in the burst.

5.1 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.

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
- After the slave has sampled the address and control it can start to drive the appropriate response and this is sampled by the bus master on the third rising edge of the clock.

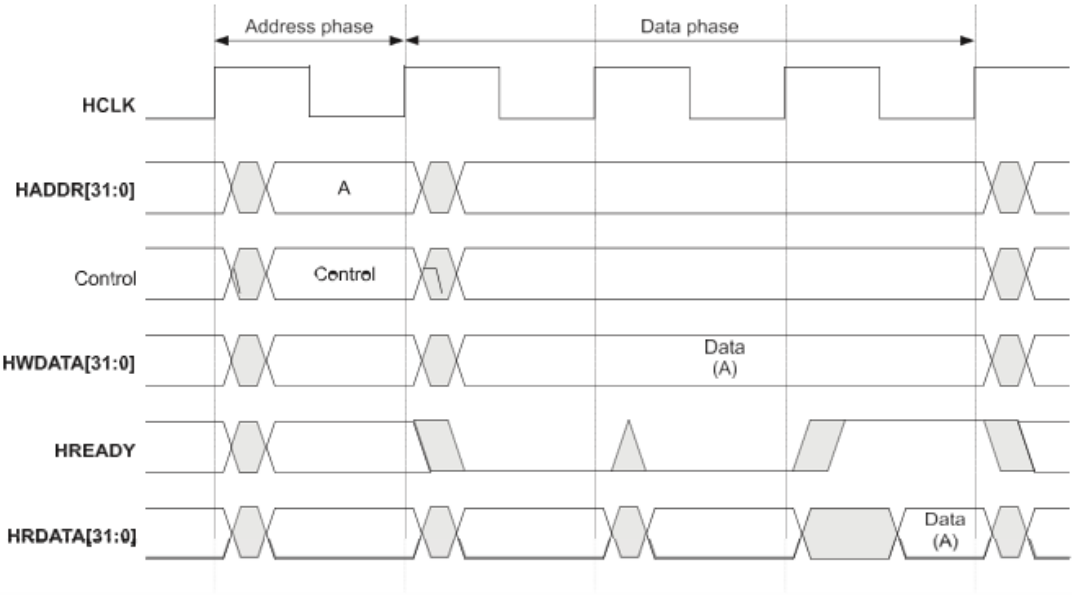


Figure 5.1: Transfer with Wait States

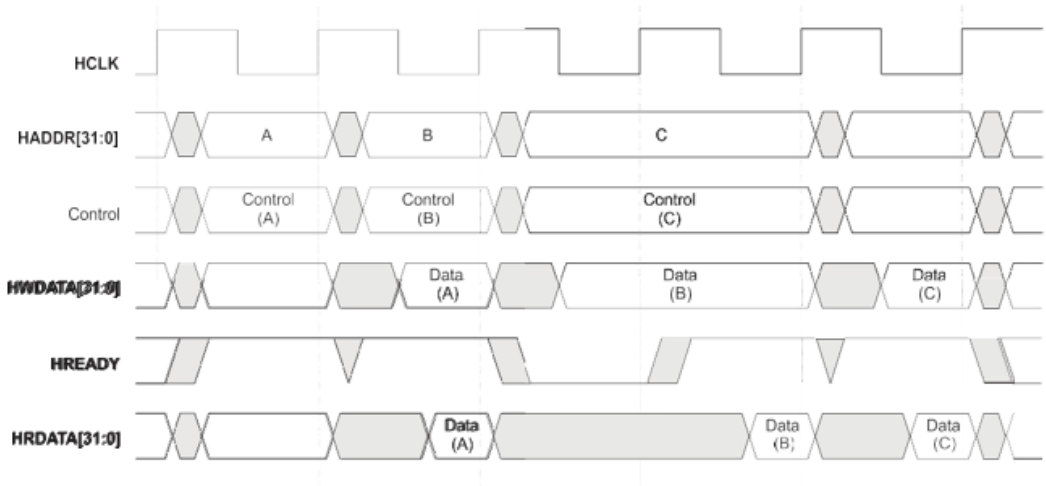


Figure 5.2: Multiple Transfers

Every transfer can be classified into one of four different types, as indicated by the **HTRANS[1:0]** signals as:

Table 5.1: 4 Types of Transfer

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 signals 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 transfers.
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).

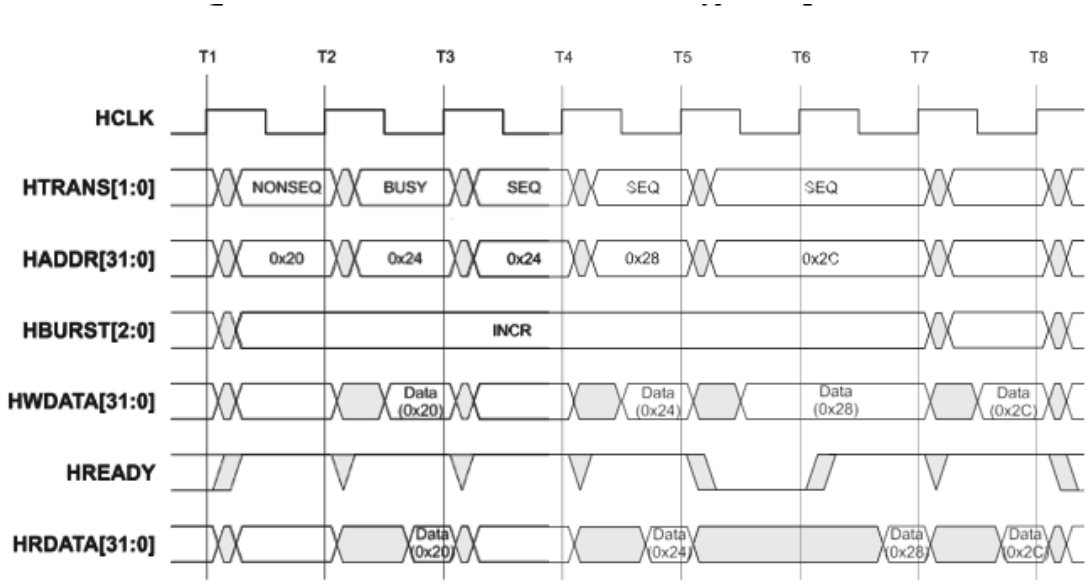


Figure 5.3: Transfer Type Examples

- The first transfer of a burst is NONSEQUENTIAL.
- The master delays the second transfer by using a BUSY transfer. In this case, the master just needs one cycle before it is ready to begin the next transfer in the burst, which completes without any wait states.
- The master initiates the third transfer of the burst, but the slave fails to finish and adds a single wait state.
- The last transfer of the burst is completed with zero wait states.

5.2 Burst Operation

The AMBA AHB protocol defines four, eight, and sixteen-beat bursts, in addition to undefined-length bursts and single transfers. The protocol supports both incrementing and wrapping bursts are as follows:

- Incrementing bursts access sequential locations and the address of each transfer in the burst is just an increment of the previous address
- For wrapping bursts, if the start address of the transfer is not aligned to the total number of bytes in the burst (size x beats) then the address of the transfers in the burst will wrap when the boundary is reached. For example, a four-beat wrapping burst of word (4-byte) accesses will wrap at 16-byte boundaries. Therefore, if the start address of the transfer is 0x34, then it consists of four transfers to addresses 0x34, 0x38, 0x3C and 0x30

Table 5.2: HBURST[2:0] on different transfer addresses

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

5.3 Early Burst Termination

Certain conditions will prevent a burst from completing, thus any slave design that uses the burst information must be able to take the appropriate action if the burst is cancelled early. The slave can detect when a burst has ended prematurely by monitoring the HTRANS signals and verifying that any transfers following the start of the burst are marked as SEQUENTIAL or BUSY. If a NONSEQUENTIAL or IDLE transfer happens, it means that a new burst has begun and the preceding one must have ended.

If a bus master is unable to complete a burst because it has lost ownership of the bus, it must reconstruct the burst correctly when it regains access to the bus. For example, if a master has only completed one beat of a four-beat burst, it must utilise an undefined-length burst to finish the remaining three transfers.

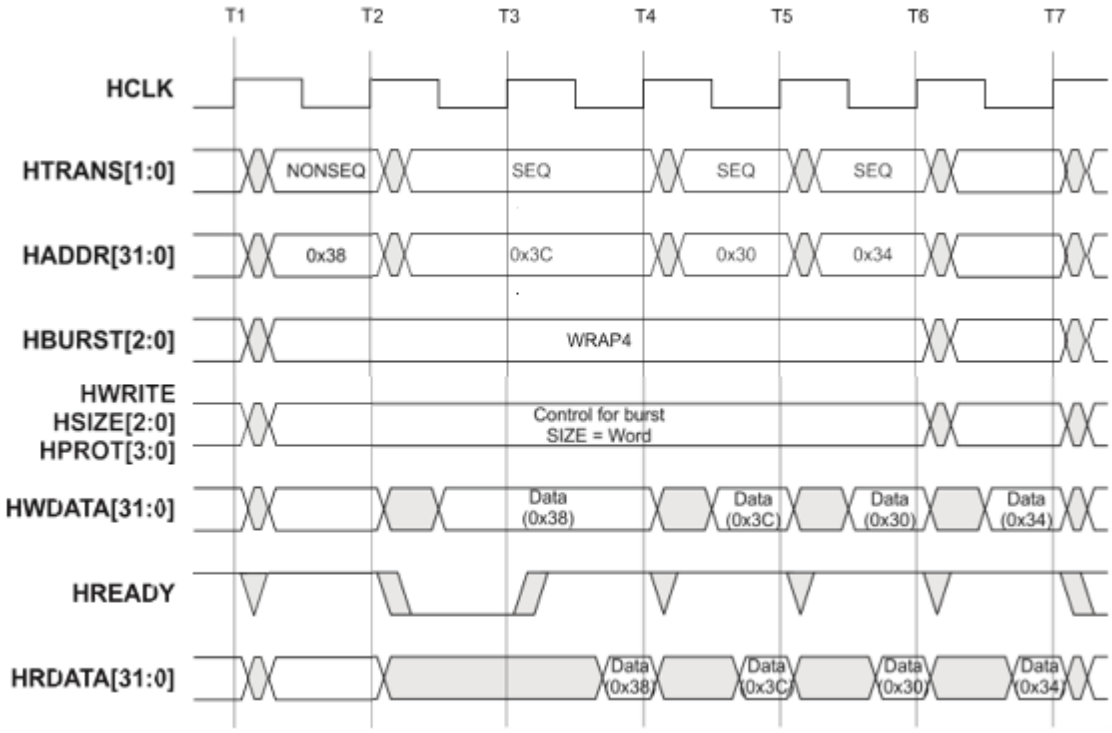


Figure 5.4: Four Beat Wrapping Burst

Considering the burst is a four-beat burst of word transfers, the address will wrap at 16-byte boundaries, therefore the transfer to 0x3C is followed by a transfer to address 0x30. The sole difference from the incrementing burst is that the addresses continue beyond the 16-byte border.

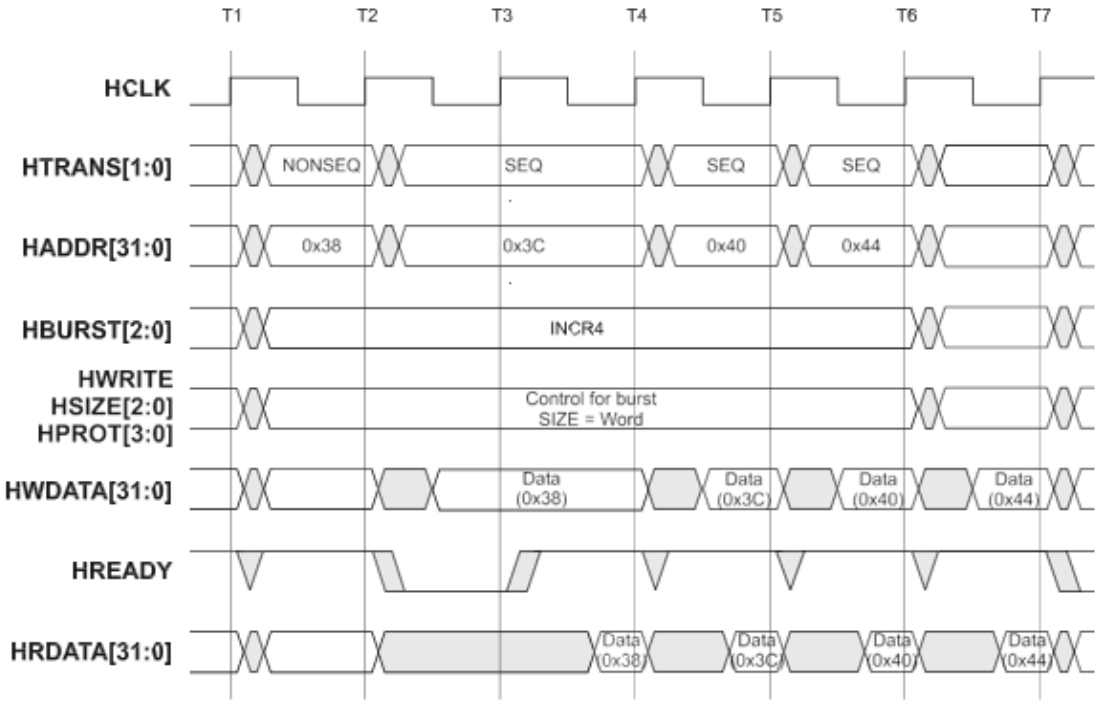


Figure 5.5: Four beat incrementing burst

Figure 5.6: Eight beat wrapping burst

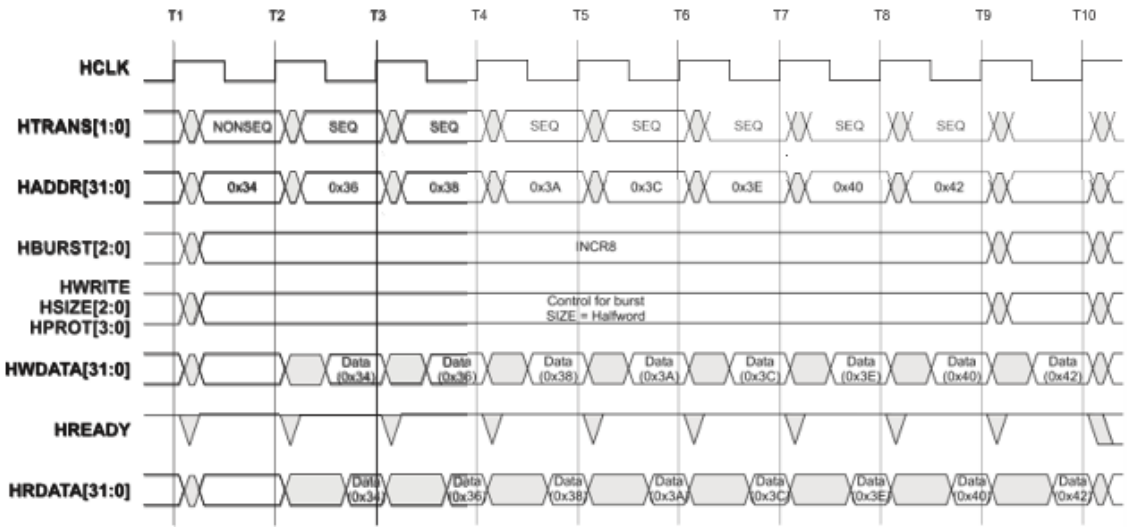


Figure 5.7: Eight beat incrementing burst

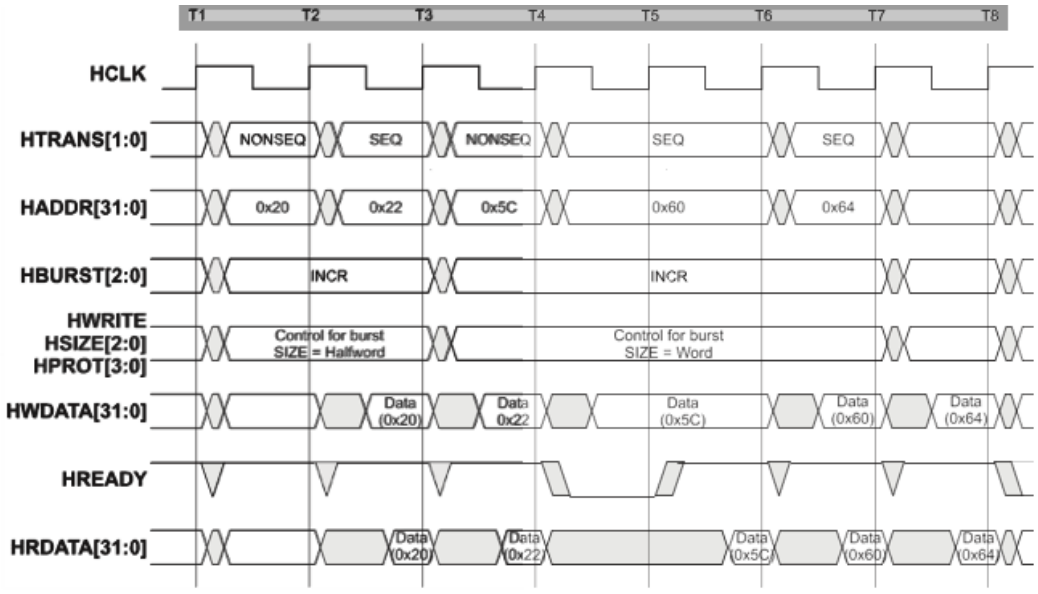


Figure 5.8: Undefined length bursts

Above figure basically describes:

- Two halfword transfers starting at address 0x20. The halfword transfer addresses increment by 2.
- Three word transfers starting at address 0x5C. The word transfer addresses increment by 4

5.4 Slave Transfer Responses

After a master initiates a transfer, the slave selects how it should proceed. There is no provision in the AHB standard for a bus master to cancel a transfer once it begins. When a slave is accessed, it must provide a response indicating the status of the transfer. The HREADY signal is used to extend the transfer and works in conjunction with the response signals, HRESP[1:0], which indicate the transfer's status.

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.

5.5 Write and Read Data Bus – HWDATA[31:0] & HRDATA[31:0]

To create an AHB system without using tristate drivers, separate read and write data buses are necessary. The minimum data bus width is defined as 32 bits, however it can be expanded.

During write transfers, the bus master drives the write data bus. If the transfer is prolonged, the bus master must keep the data valid until the transfer is complete, as signalled by the mode called as HREADY HIGH.
All transfers must be aligned with an address boundary that is equal to their size. For example, word transfers (A[1:0] = 00) and halfword transfers (A[0] = 0) must be aligned to word address boundaries.

Transfers that are narrower than the bus width, such as a 16-bit transfer on a 32-bit bus, need the bus master to only drive the relevant byte lanes. The slave is responsible for picking write data from the appropriate byte lanes. Tables 3-6 and 3-7 on page 3-26 illustrate which byte lanes are active on a little-endian and big-endian system, respectively. If necessary, this information can be expanded for larger data bus systems. Burst transfers with transfer sizes smaller than the data bus's width will have separate active byte lanes for each beat of the burst.

The active byte lane is determined by the system's endianness, although AHB does not specify which endianness is necessary. As a result, all masters and slaves on the bus must share the same endianness.

The suitable slave drives the read data bus during read transfers. If the slave prolong the read transfer by keeping HREADY LOW, it only has to transmit valid data at the conclusion of the transfer cycle, as indicated by HREADY HIGH.
For transfers that are less than the width of the bus, the slave merely has to supply correct data on the active byte lanes. The bus master is responsible for picking data from the appropriate byte lanes.

Transfer size	Address offset	DATA [31:24]	DATA [23:16]	DATA [15:8]	DATA [7:0]
Word	0	✓	✓	✓	✓
Halfword	0	-	-	✓	✓
Halfword	2	✓	✓	-	-
Byte	0	-	-	-	✓
Byte	1	-	-	✓	-
Byte	2	-	✓	-	-
Byte	3	✓	-	-	-

Transfer size	Address offset	DATA [31:24]	DATA [23:16]	DATA [15:8]	DATA [7:0]
Word	0	✓	✓	✓	✓
Halfword	0	✓	✓	-	-
Halfword	2	-	-	✓	✓
Byte	0	✓	-	-	-
Byte	1	-	✓	-	-
Byte	2	-	-	✓	-
Byte	3	-	-	-	✓

Figure 5.9: Tabular Diagram showing active lanes for 32 bit little & big-endian data bus

5.6 Role of Endinaness in the system

Dynamic endianness is not supported because it would result in a considerable and unnecessary silicon overhead in the majority of embedded systems.
It is advised that module designers make only modules that will be utilised in a wide range of applications bi-endian, with endianness selected by a configuration pin or an internal control bit. For more application-specific blocks, changing the endianness to little-endian or big-endian will result in a smaller, lower-power, and higher-performing interface.

CHAPTER 6: AHB MODULES

AHB (Advanced High-performance Bus) modules are a set of standardized interfaces commonly used in System-on-Chip (SoC) designs for high-performance computing systems. AHB modules facilitate communication and data transfer between various components within an SoC, such as processors, memory, and peripherals.

6.1 APB Bridge

The AHB to APB Bridge is an AHB Slave, provising an interface between the high-speed AHB and the low power APB. Read and write transfers on the AHB are converted into equivalent transfers on the APB. As the APB is not pipelined, then wait states are added during transfers to and from the APB when the AHB is required to wait for the APB.

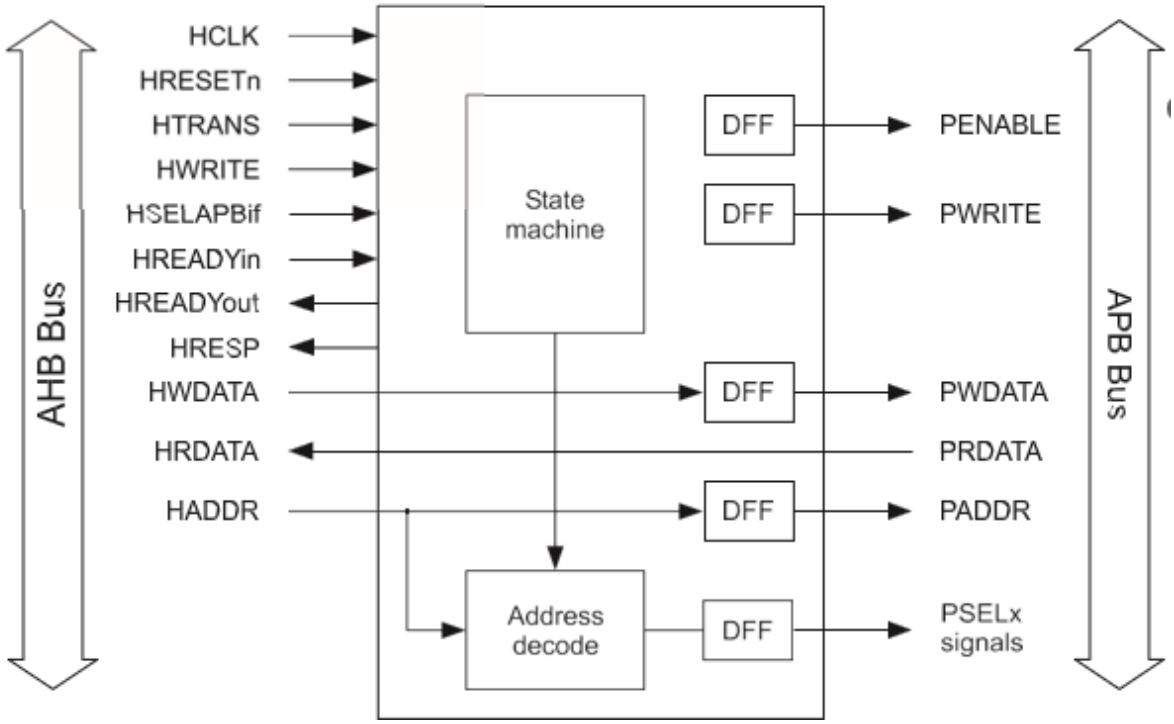


Figure 6.1: Block Diagram of the APB Bridge Module

This module primarily includes the following:

- AHB Slave bus interface
- APB transfer state machine, which is independent of the device memory map
- APB Output Signal generation

In order to add new APB Peripherals, or alter the system memory map, only thr decode sections need to be modified.

6.2 Signal Descriptions in the APB Bridge Module

Some of the APB Bridge Module Signals are descrived in the Table below as follows:

Table 6.1: APB Bridge Module Signals' output

HRESP [1]	HRESP [0]	Response	Description
0	0	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.
0	1	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.
1	0	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.
1	1	SPLIT	The transfer has not yet completed successfully. The bus master might retry the transfer when the access is granted next.

6.3 Peripheral Memory map and Operation of the Module

The APB Bridge controls the memory map for peripherals, and generates a select signal for each peripheral. The default system map is as follows:

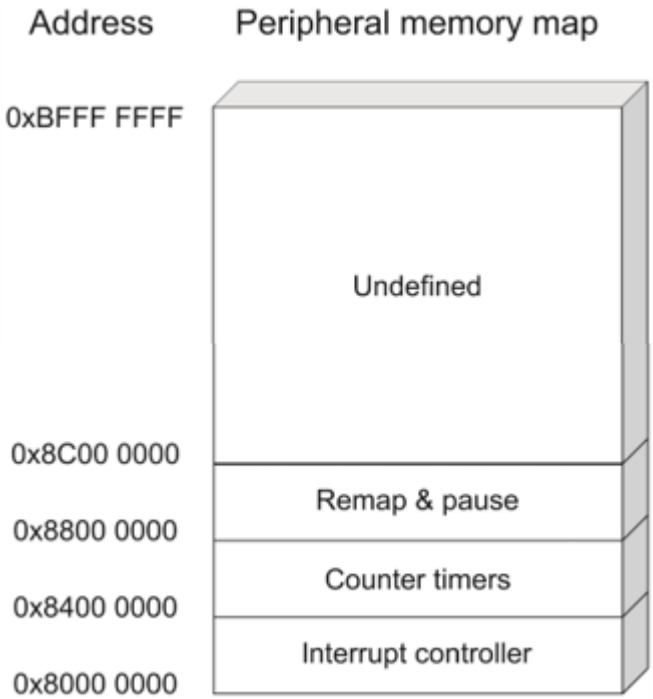


Figure 6.2: Peripheral Memory Map of APB Bridge

The APB Bridge responds to transaction requests from the currently granted AHB master. The AHB transactions are then converted into APB Transactions. The state machine controls the following:

- The AHB transactions with the HREADYout signal
- The generation of all APB output signals.

The individual PSELx signals are decoded from HADDR, using the state machine to enable the outputs while the APB transactions are being performed.

If an Undefined location is accessed, operation of the system continues as normal, but no peripherals are selected.

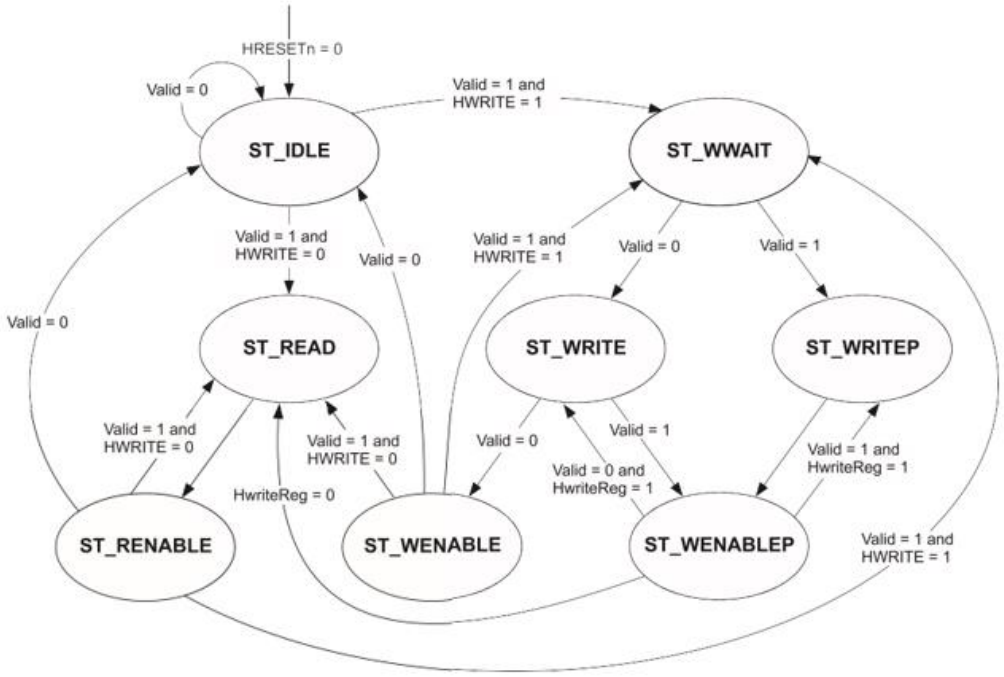


Figure 6.3: State Machine of AHB to APB Interface

6.4 Description of the combined system

This sub chapter shall be describing how the HDL code for the AHB brudge is set out. A simple system block diagram with information about the main parts of the HDL code, is followed by details of the registers, inputs and outputs used in the module. This should more or less be read with the conjunction with the HDL code.

The AHB to APB Bridge basically consists of a state machine, which is used to control the generation of the APB and AHB output signals, and address the decoding logic which is used to generate the APB Peripheral select lines.

All the registers used in the system are clocked from the rising edge of the system clock HCLK and uses the asynchronous reset HRESETn.

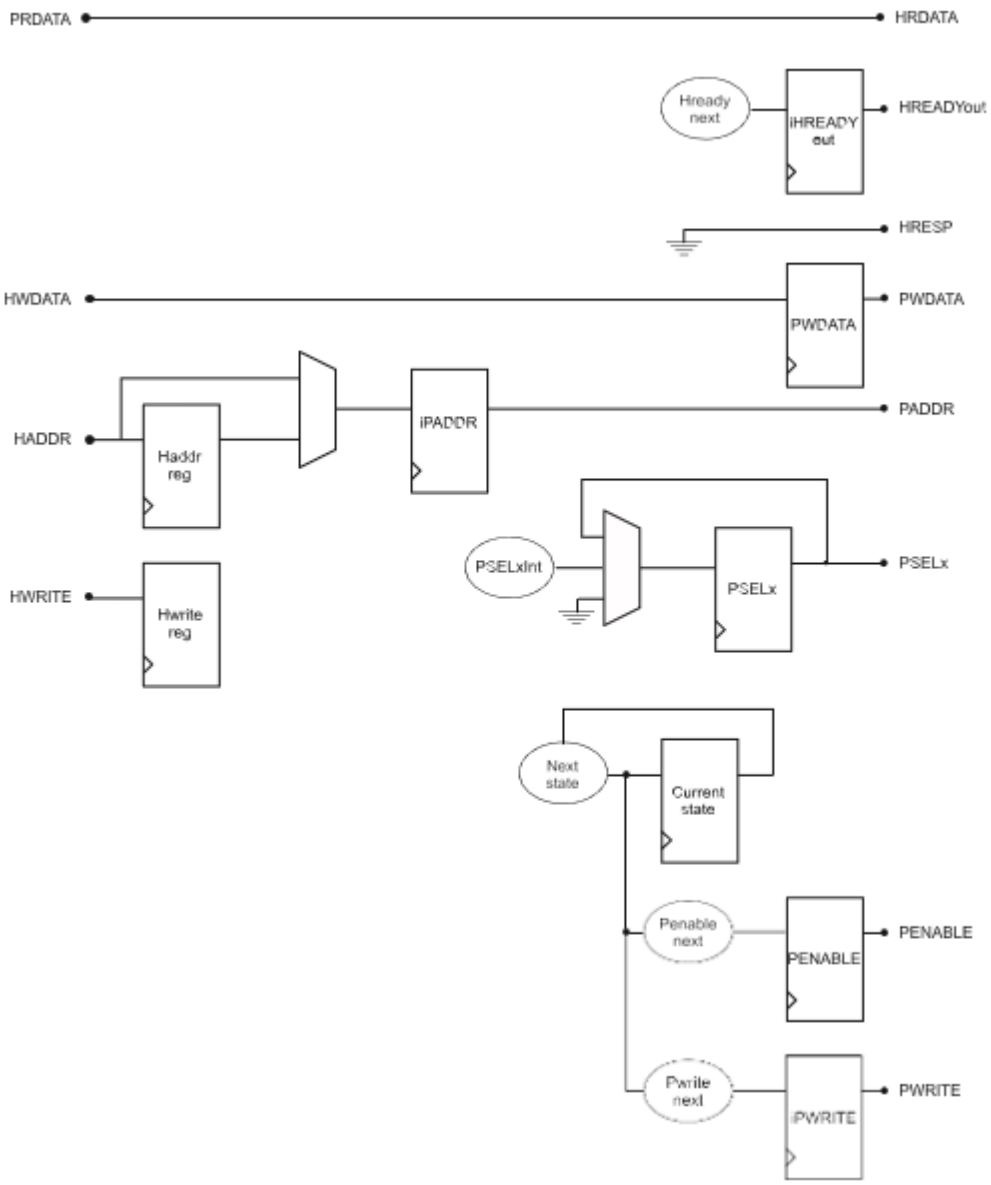


Figure 6.4: APB Bridge Module System

6.5 AHB Bus Salve Interface

The module in the project uses a standard AHB interface which comprises of:

- The valid transfer detection logic which is used to determine when a valid transfer is accessing the slave
- The address and control resgiters, which are used to store the information from the address phase of the transfer for use in the data phase.

Considering 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 multiplexer is therefore used to select between the current address input or the registered address, for the read and write transfers respectively.

6.6 APB Output Signal Generation

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

PWDATA is the registered version of the HWDATA inout 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 dueing 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 state transfer address. They are only valid dueing 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 registered address register). And only changes whne 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 regsiters, ensutring that these signals only change when a new APB Transfer is started, when the next state is ST_READ, ST_WRITE, or ST_WRITEP.

6.7 AHB Output Signal Generation

HRDATA is directly driven with the current value of PRDATA. APB Slaves only read data during the enable phase of the APB Transfer, with PRDATA set LOW ar all times.

HREADYout is driven with a registered signal to improve output timings.

HRESP is continuously held LOW, as the APB Bridge does not generate SPLIT, RETRY or ERROR responses.

6.8 Intefracing the APB to AHB and Transfers involved

In case of the Read transfers, the transfer states in the AH start at time T1 and the address is sampled by the APB Bridge at T2. If the transfer is for the peripheral bus, then this address is broadcast and the appropriate peripheral cycle signal is generated. The first cycle is called the SETUP cycle and this is followed by the ENABLE cycle, when the PENABLE signal is asserted.

During the ENABLE cycle, the peripheral must provide the real data. Normally, it will be possible to route this read data directly back to the AHB where the bus master can sample it on the rising edge of the clock at the end of the ENABLE cycle, which is set at time T4.

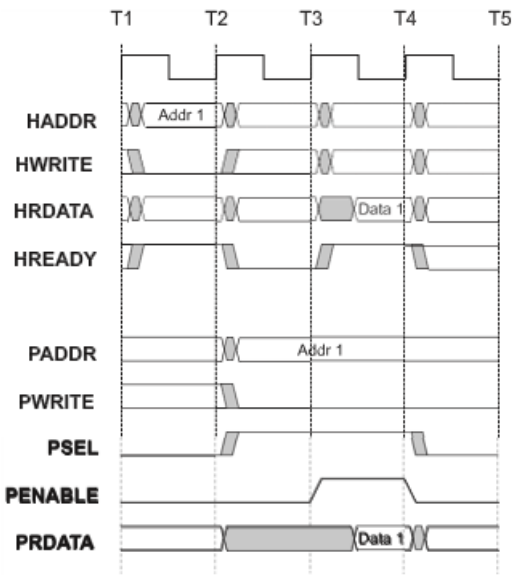


Figure 6.5: Read Transfer on AHB

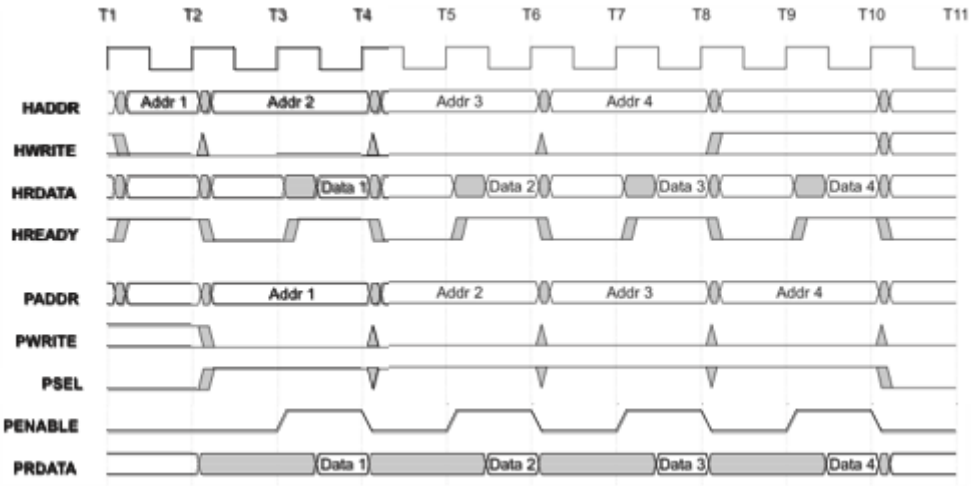


Figure 6.6: Burst of read transfers

A similar kind of sequence is observed in case of the write transfers as well.

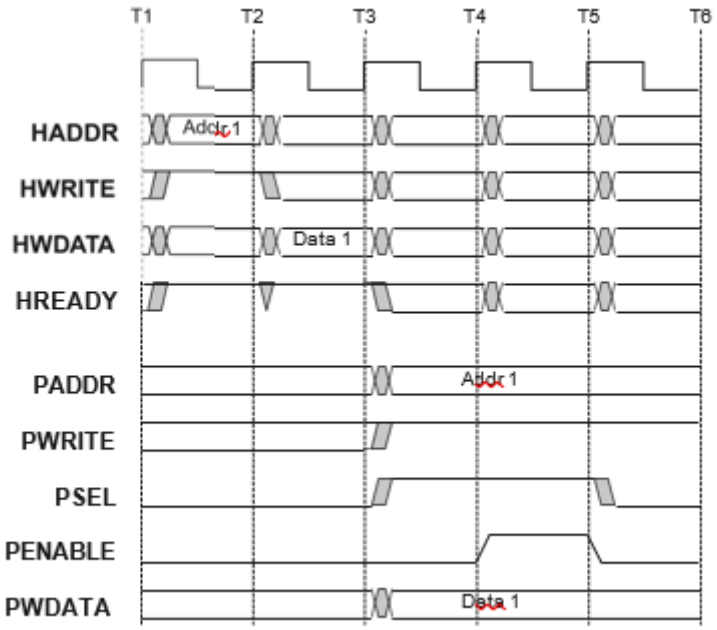


Figure 6.7: Write Transfer from the AHB

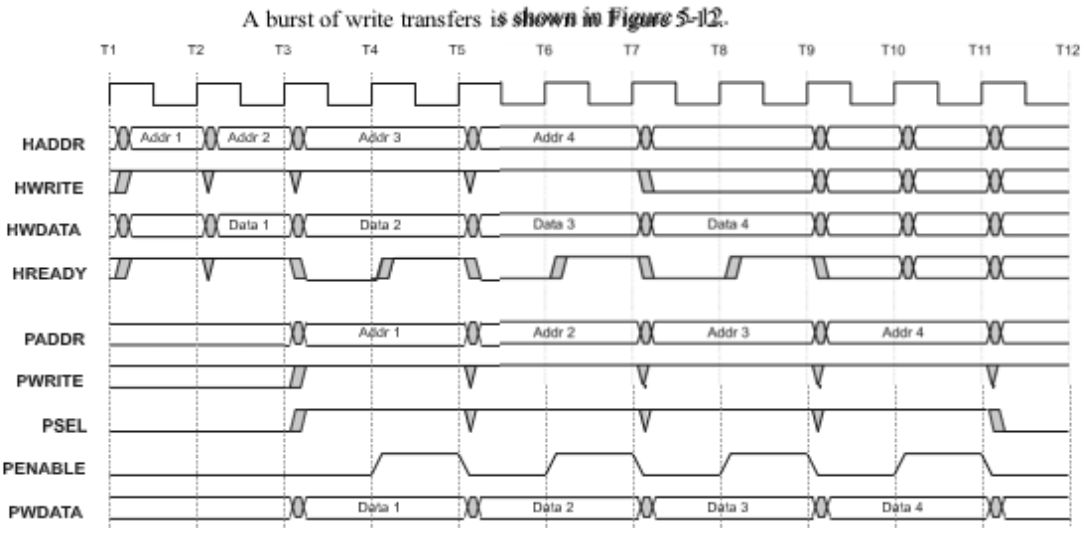


Figure 6.8: A burst of write transfers

While the first transfer can be complete with zero wait states, subsequent tranfers to the peripheral bus will require a single wait state for each transfer performed. It is necessary for the bridge to contain two address registers, in order that the bridge can sample the address of the next transfers while the current transfer continues on the peripheral bus.

There is also a case of back to back tranfers and tristate data bus implementation transfers and the same is shown

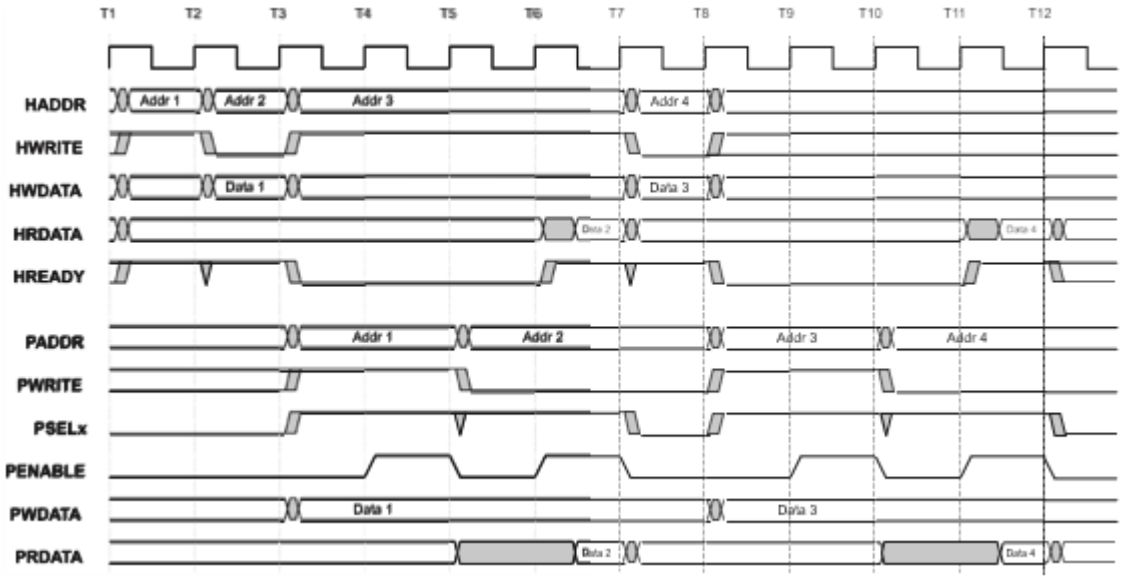


Figure 6.9: Back to back transfers

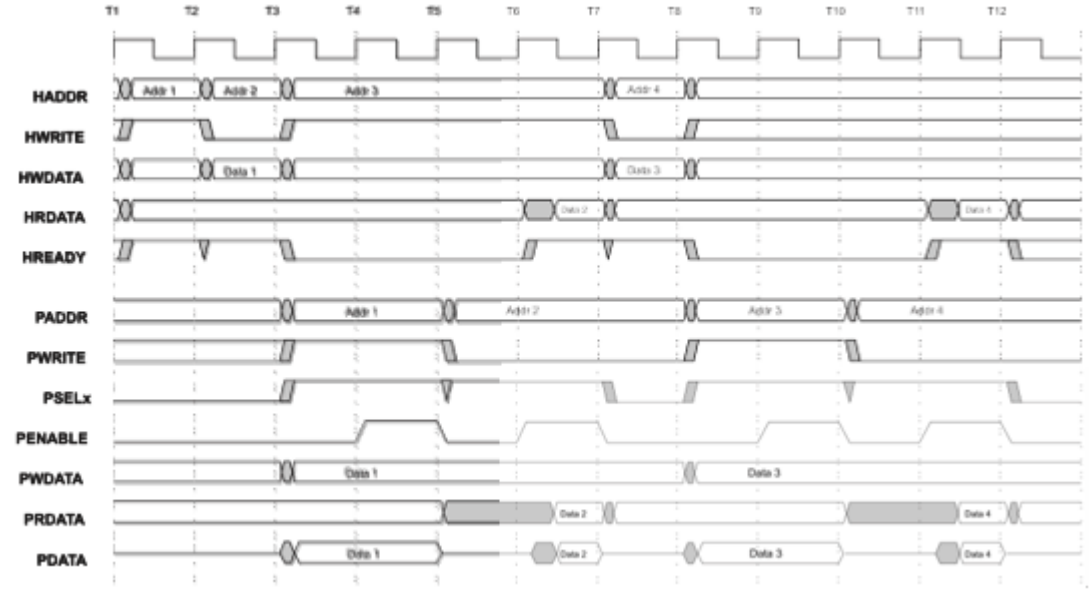


Figure 6.10: Tristate data bus implementations

CHAPTER 7: SIMULATIONS AND RESULTS

After completing the interfacing of the modules and writing a test bench code on Verilog programming language, here are the results of few of the expected waveforms obtainted.

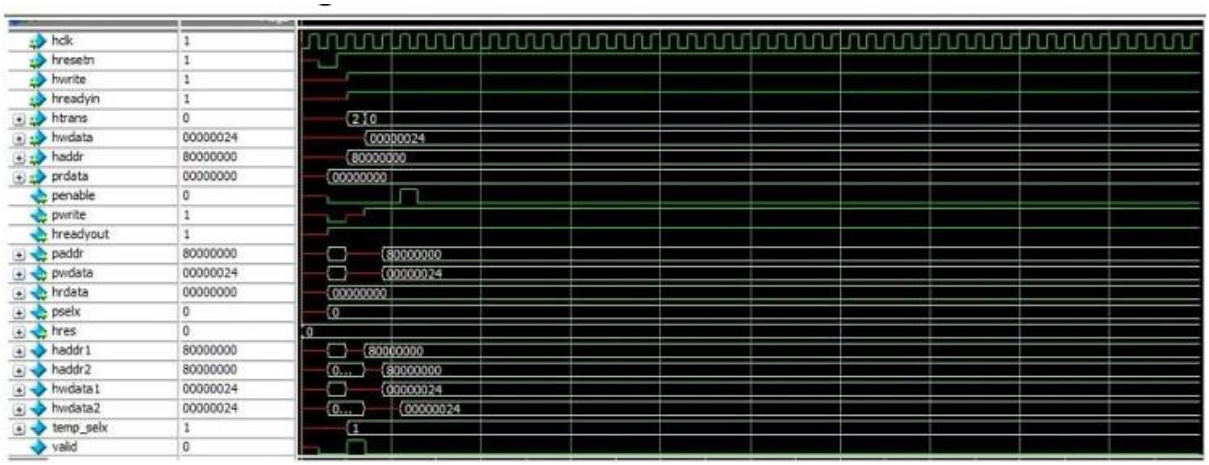


Figure 7.1: Waveform of Single Write

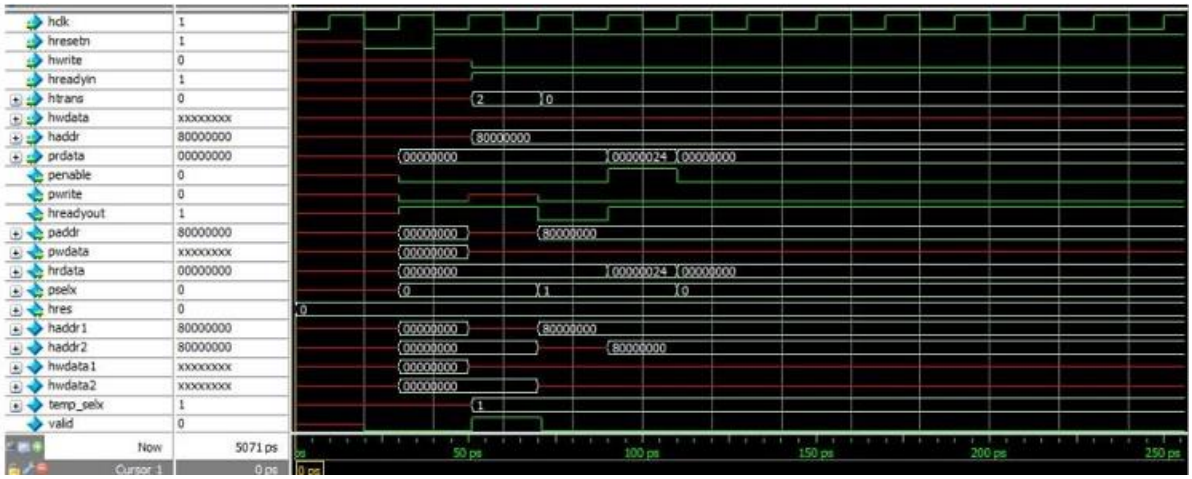


Figure 7.2: Waveform of Single Read

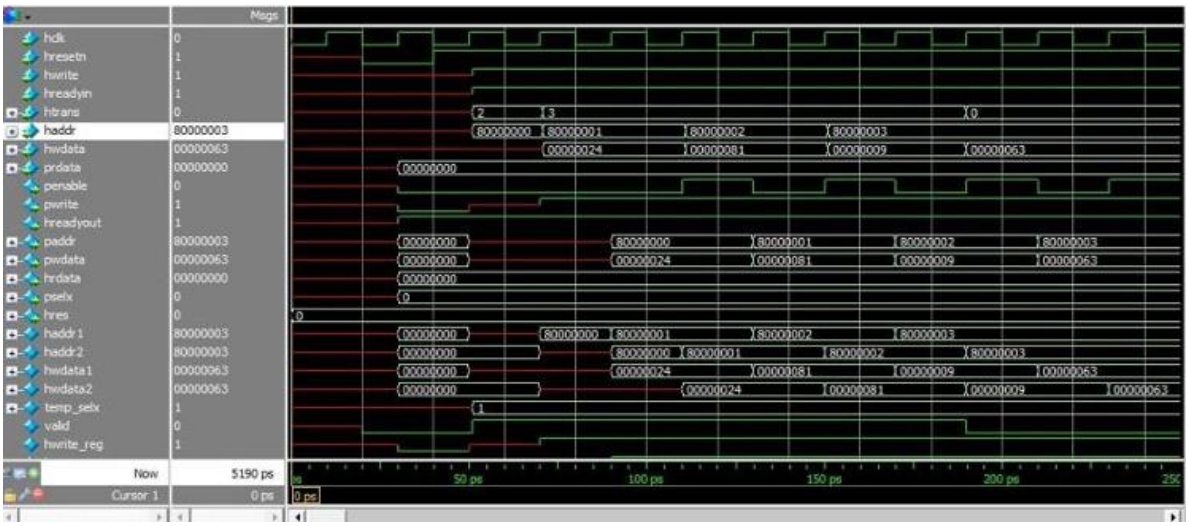


Figure 7.3: Waveform of Burst Write

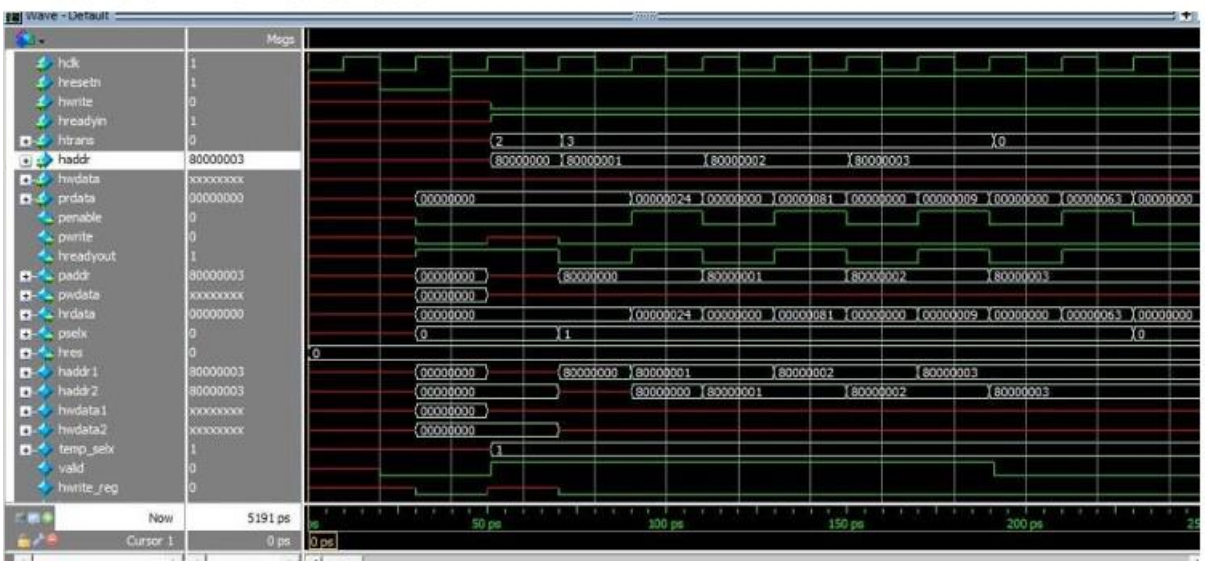


Figure 7.4: Waveform of Burst Read

The generation of these waveforms prove that the AHB2APB bridge has been successfully designed and coded using a Test Bench and that this Entire Module can be used for future applications and scope in the domain of VLSI Design, especially the frontend applications.

CHAPTER 8: FUTURE SCOPE AND CONCLUSION

The AHB2APB bridge, a vital component in System-on-Chip (SoC) designs, holds immense promise for future developments in the realm of embedded systems. As technology continues to advance rapidly, the future scope of the AHB2APB bridge encompasses several key areas of innovation and enhancement.

Firstly, with the increasing complexity and integration of SoCs, there is a growing demand for more efficient and high-performance bridges. Future iterations of the AHB2APB bridge are likely to focus on optimizing data transfer rates, reducing latency, and enhancing overall throughput to meet the evolving needs of advanced applications, such as artificial intelligence, machine learning, and Internet of Things (IoT) devices.

Additionally, as SoCs become more heterogeneous, incorporating diverse processing elements and peripherals, the AHB2APB bridge will play a pivotal role in facilitating seamless communication and interoperability between different components. Future versions of the bridge may feature enhanced compatibility with emerging standards and protocols, ensuring smooth integration with a wide range of system components and peripherals.

Furthermore, with the proliferation of edge computing and decentralized architectures, there is a growing emphasis on power efficiency and low-latency communication. Future AHB2APB bridges may incorporate innovative power management techniques and advanced buffering mechanisms to minimize energy consumption while maintaining high-speed data transfers.

Overall, the future scope of the AHB2APB bridge is marked by continuous innovation and adaptation to the evolving landscape of embedded systems. By addressing the challenges posed by increasingly complex SoC designs and emerging applications, future iterations of the bridge are poised to unlock new possibilities and propel the development of next-generation embedded systems.

One of the key takeaways from this project is the importance of thorough understanding and implementation of industry-standard protocols and interfaces. By adhering to the specifications of the AHB and APB protocols, we have ensured compatibility and interoperability with a wide range of SoC components and peripherals, enhancing the versatility and scalability of the bridge design.

Furthermore, the project has underscored the significance of effective collaboration and teamwork in achieving project goals. Through close coordination among team members, efficient allocation of tasks, and regular communication, we have overcome challenges and successfully delivered a high-quality solution within the specified timeframe.

Moreover, the project has provided valuable insights into the intricacies of hardware design, including signal routing, clock synchronization, and data arbitration. By applying principles of digital design and simulation techniques, we have gained practical experience and honed our skills in designing complex hardware systems.

Looking ahead, the project lays a solid foundation for future advancements in SoC design and embedded systems development. By building upon the lessons learned and leveraging emerging technologies, we can further enhance the performance, efficiency, and functionality of AHB2APB bridges, driving innovation and enabling new applications in diverse fields ranging from automotive and aerospace to consumer electronics and IoT devices.

In conclusion, the design of the AHB2APB bridge represents a significant milestone in the development of System-on-Chip (SoC) designs, offering a crucial interface for seamless communication and data transfer between Advanced High-performance Bus (AHB) and Advanced Peripheral Bus (APB) modules. Through meticulous planning, rigorous testing, and iterative refinement, the project has successfully delivered a robust and efficient bridge solution that meets the demanding requirements of modern embedded systems.

References

[1] ARM AMBA 5 AHB Protocol Specification, ARM

[2] AMBA APB Protocol (Version 2.0), ARM

[3] AHB Example AMBA System (Technical Reference Manual), ARM

[4] Capital Microelectronics, Inc., "AHB2APB Bridge User Guide"

[5] Vani R. M. and M. Roopa, "Design of AMBA based AHB2APB Bridge", IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.11, November 2010

[6] Prof. Ravi Mohan Sairam and Prof. Sumit Sharma, "FSM & Handshaking based AHB to APB Bridge for High Speed Systems", International Journal of Engineering Research & Technology (IJERT) Vol. 2 Issue 11, November – 2013.

Online Resources:

1. https://developer.arm.com/architectures/system-architectures/amba
2. https://www.arm.com/products/silicon-ip-system/embedded-systemdesign/amba-specifications
3. https://en.wikipedia.org/wiki/Advanced_Microcontroller_Bus_Architecture
4. http://paper.ijcsns.org/07_book/201011/20101104.pdf

Appendix

Code for AHB2APB Bridge:

```
`define IDLE 3'b000

`define READ 3'b001

`define WWAIT 3'b010

`define WRITE 3'b011

`define WRITEP 3'b100

`define WENABLE 3'b101

`define WENABLEP 3'b110

`define RENABLE 3'b111

`timescale 1ns/1ps module ahb2apb(HCLK, HRESETn, HSELAPB, HADDR,
HWRITE, HTRANS, HWDATA, HRESP, HRDATA, HREADY, PRDATA, PSEL,
PENABLE, PADDR, PWRITE, PWDATA);
```

Code for AHB Slave Interface:

```
input wire HCLK, HRESETn, HSELAPB, HWRITE;

input wire [1:0]HTRANS;

input wire [31:0]HADDR, HWDATA;

output reg HRESP, HREADY;

output reg [31:0]HRDATA;
```

APB Output Signals:

```
input wire [31:0]PRDATA;

output reg PSEL, PENABLE, PWRITE;

output reg [31:0]PADDR, PWDATA;
```

Implementation Signals:

```
reg [31:0]TMP_HADDR, TMP_HWDATA;
```

```
reg [2:0] ps,ns;
reg valid, HWrite;
always @(*) begin
```

VALID LOGIC:

```
if (HSELAPB==1'b1 && (HTRANS==2'b10 || HTRANS==2'b11))
valid=1'b1;
else
valid=1'b0; if(HRESETn==1'b0) ns=`IDLE; HRESP=1'b0;
end
always @(posedge HCLK) begin ps=ns; end
always @(ps)
begin
 case(ps)
`IDLE :
 begin PSEL=1'b0;
 PENABLE=1'b0;
HREADY=1'b1;
 if(valid==1'b0) ns=`IDLE;
 else if(valid==1'b1 && HWRITE==1'b0)
 ns=`READ;
else if(valid==1'b1 && HWRITE==1'b1)
 ns=`WWAIT;
end
 `READ : begin PSEL=1'b1;
PADDR=HADDR; PWRITE=1'b0;
PENABLE=1'b0; HREADY=1'b0;
ns=`RENABLE;
end
```

```
`WWAIT : begin PENABLE=1'b0;
TMP_HADDR=HADDR;
 HWrite=HWRITE;
if(valid==1'b0) ns=`WRITE;
else if(valid==1'b1) ns=`WRITEP;
end
`WRITE :
begin PSEL=1'b1;
 PADDR=TMP_HADDR;
PWDATA=HWDATA; PWRITE=1'b1;
 PENABLE=1'b0;
 HREADY=1'b0;
 if(valid==1'b0) ns=`WENABLE;
else if(valid==1'b1) ns=`WENABLEP;
 end
```

Testbench Code :

```
 `timescale 1ns/1ps module tb;
 //----------------------AHB Slave Interface----------------------
 reg HCLK, HRESETn, HSELAPB, HWRITE;
reg [1:0]HTRANS;
reg [31:0]HADDR, HWDATA; wire HRESP; wire [31:0]HRDATA;
reg [31:0]PRDATA;
wire PSEL, PENABLE, PWRITE, HREADY;
wire [31:0]PADDR, PWDATA; always #1 HCLK=~HCLK; Page  34 of 45   `ifdef
Single_Read initial begin $dumpfile("Single_Read.vcd"); $dumpvars; end initial begin
=1'b1; HRESETn=1'b0; #2 HRESETn=1'b1; HWRITE=1'b0; HSELAPB=1'b1;
HTRANS=2'b10; HADDR=32; #2.1 HWRITE=1'bx; HSELAPB=1'b0;
HTRANS=2'bxx; HADDR=32'hxxxx_xxxx; #1.9 PRDATA=16; #2 $finish; end
```