VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belagavi – 590018, Karnataka.

A

Project Work

On

"DESIGN AND VERIFICATION OF AMBA APB PROTOCOL USING SYSTEM VERILOG"

Submitted in partial fulfillment for the award of degree of

BACHELOR OF ENGINEERING

In

ELECTRONICS & COMMUNICATION ENGINEERING

Project Associates

NAME	USN
1. Mr. ABHINANDAN K	4GM20EC001
2. Mr. BASAVARAJ ARUN SUNADHOLI	4GM20EC018
3. Mr. DANESH S ANGADI	4GM20EC025
4. Mr. LIKHITH G K	4GM20EC054

for the Academic Year 2023-24

Under the Guidance of

Dr. Praveen J

Head of the Department

Dr. Rajashekhar Somasagar
Project Coordinator

Dr. Praveen J
Head of the Department

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

Srishyla Educational Trust (R), Bheemasamudra

GM INSTITUTE OF TECHNOLOGY, DAVANGERE

Approved by AICTE | Affiliated by V.T.U Belagavi | Recognized by Govt. of Karnataka

GM INSTITUTE OF TECHNOLOGY

Department of Electronics & Communication Engineering
Post Box No. 4, P.B. Road, Davangere – 577 006, Karnataka State.
Email: info@gmit.info Website: www.gmit.ac.in

CERTIFICATE

This is to certified that, the project entitled **"DESIGN AND VERIFICATION OF AMBA APB PROTOCOL USING SYSTEM VERILOG"** carried out by **ABHINANDAN K(4GM20EC001), BASAVARAJ ARUN SUNADHOLI (4GM20EC018), DANESH S ANGADI(4GM20EC025) and LIKHITH G K(4GM20EC054),** bonafide students of VIII semester in partial fulfillment for the award of Bachelor of Engineering in **ELECTRONICS & COMMUNICATION ENGINEERING** of the Visvesvaraya Technological University, Belagavi for the academic year 2023-24.

Dr. Praveen J **Project Guide**	**Dr. Rajashekhar Somasagar** **Project Coordinator**
Dr. Praveen J **Head of the Department**	**Dr. Sanjay Pande M B** **Principal**

Name of Examiners	**Signature with date**
1.	**1.**
2.	**2.**

ACKNOWLEDGEMENT

The joy and satisfaction that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible.

We would like to express our gratitude to the Principal, **Dr. Sanjay Pande M B** for providing us a congenial environment for engineering studies and also for having showed us the way to carry out the project.

We consider it a privilege and honor to express our sincere thanks to, **Dr. Praveen J,** IQAC-Director, Professor and Head, Department of Electronics & Communication Engineering for his support and invaluable guidance throughout the tenure of this project.

We would like to thank our Guide **Dr. Praveen J,** IQAC-Director, Professor and Head, Department of Electronics & Communication Engineering for support, guidance, motivation, encouragement for the successful completion of this project.

We would also like to thank Project Coordinators **Dr. Rajashekhar Somasagar,** Associate Professor, Department of Electronics & Communication Engineering for their continual support and guidance from the scratch and throughout the process of project.

We intend to thank all the **teaching and non-teaching staff** of Department of Electronics & Communication Engineering for their immense help and co-operation.

Finally, we would like to express our gratitude to **parents and friends** who always stood by us.

Project Associates

1. **ABHINANDAN K (4GM20EC001)**
2. **BASAVARAJ A S (4GM20EC018)**
3. **DANESH S A (4GM20EC025)**
4. **LIKHITH G K (4GM20EC054)**

ABSTRACT

The Advanced Microcontroller Bus Architecture (AMBA), Advanced Peripheral Bus (APB) protocol stands as a cornerstone in modern System-on-Chip (SoC) designs, serving as the vital link between peripheral components and the central processing unit (CPU). Crafting this interface demands meticulous attention to detail, encompassing the development of an AMBA APB interface that strictly adheres to the protocol's exacting standards. This entails designing and integrating the APB controller, the APB bus interface, and various peripheral modules. Leveraging System Verilog facilitates a modular and adaptable design approach, accommodating diverse data widths, address spaces, and peripheral requisites.

In the verification phase, a robust System Verilog testbench is constructed to validate the accuracy of the devised AMBA APB interface. The process entails creating both master and slave functionalities in accordance with the AMBA APB protocol's specifications. The verification environment, constructed with System Verilog, meticulously evaluates the design's capacity for data transfers (read/write), proficient handling of bus arbitration, and adept management of wait states.

This methodology guarantees the functional correctness and resilience of the design by subjecting it to a battery of simulations, encompassing various scenarios, including corner cases and error conditions. Furthermore, the emphasis on modularity and reusability fosters efficiency in adapting the design for future projects reliant on AMBA APB.

CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

In the domain of digital system design, industry standards is crucial for guaranteeing the compatibility, dependability, and seamless operation of integrated circuits, microcontrollers, and System-on-Chip (SoC) designs. The advanced peripheral bus protocol, a vital element of ARM's AMBA, exemplifies the importance of these standards. AMBA APB has emerged as the preferred option for crafting interfaces between microcontroller peripherals and the central processing unit (CPU) in ARM-based systems, and its remarkable balance of simplicity and effectiveness.

The design and verification of an AMBA APB interface is a process that encompasses two pivotal phases. In the design phase, the goal is to create an AMBA APB interface that adheres to the stringent protocol requirements. This phase involves the design and integration of various components, including the APB controller, the APB bus interface, and a host of peripheral modules that are designed to serve as bridges between the digital world and the physical environment. This process requires a keen eye for detail and a deep understanding of the AMBA APB protocol to ensure that the designed interface aligns with the protocol's specifications.

System Verilog, with its rich set of features and flexibility, provides the canvas for this design phase, allowing for modular and adaptable architecture that can be tailored to diverse data widths, address spaces, and peripheral specifications. Following the design phase is the equally critical verification phase. Here, the focus shifts to ensuring the correctness and reliability of the designed AMBA APB interface. A systematic approach to verification is paramount, involving the creation of a comprehensive System Verilog testbench.

This testbench becomes the proving ground where various test cases, encompassing both typical and edge-case scenarios, are executed. The goal is to verify that the designed interface not only complies with the AMBA APB specification but also operates flawlessly in real-world scenarios. Techniques such as simulation, code coverage analysis, assertion-based verification, and functional coverage analysis play a crucial role in this phase, helping to unearth any anomalies and issues in the design. By adhering to a structured methodology encompassing both the design and verification phases, endeavors to equip designers with a systematic process for creating, validating, and integrating AMBA APB interfaces.

Embedded within the broader framework of the ARM (Advanced RISC Machines) AMBA architecture, the AMBA APB protocol holds a pivotal role, particularly in System-on-Chip (SoC) designs. Functioning as an on-chip communication protocol, AMBA APB is tailored for facilitating communication among low-bandwidth peripherals, providing an equilibrium between performance and resource utilization. Its hierarchical arrangement encompasses master and slave interfaces, facilitating smooth data transmission among different components within the SoC.

The architecture of the AMBA APB protocol involves a master-slave relationship, where masters initiate data transfers, and slaves respond to these requests. Additionally, bridge components may be utilized to connect different AMBA buses, allowing for enhanced scalability and flexibility in system design. The data transfer mechanism of the APB protocol is characterized by a burst-based approach, optimizing data flow between master and slave components. This approach enables the efficient utilization of system resources and minimizes latency, making it well-suited for connecting peripherals with relatively lower data transfer requirements.

Designing with the AMBA APB protocol involves crucial considerations such as protocol configuration and timing. The former involves tailoring the protocol to specific application requirements, including choices related to clock frequency, address mapping, and peripheral selection. Addressing timing considerations is essential for achieving reliable and synchronous communication, necessitating meticulous attention to setup and hold times, clock-to-data relationships, and clock domain crossings.

Verification methods for the AMBA APB protocol include simulation-based approaches and formal verification techniques. Simulation tools and testbenches are utilized to confirm the functional accuracy and efficiency of APB designs. Conversely, formal methods like model checking and theorem proving offer stringent mechanisms to guarantee protocol compliance and system characteristics, especially as design complexity escalates.

Chapter 2

LITERATURE REVIEW

Anushka Dwivedi and Dr. Anand Jatti published [1] "Design and Implementation of AMBA APB Protocol Using System Verilog" in July 2022. The paper would likely start with an introduction to the AMBA APB protocol, explaining its significance in ARM-based system designs. It might discuss the need for efficient communication between peripherals and the CPU and how the APB protocol addresses this need. System Verilog Introduction: Since the paper focuses on implementing the protocol using System Verilog, it would probably include a brief overview of System Verilog, highlighting its features and advantages for hardware design and verification.

Shankar and Dipti Girdhar authored [2] "Design and Verification of AMBA APB Protocol" in June 2014. This paper provides an overview of the AMBA bus architecture and extensively examines the APB bus. It specifically focuses on designing the APB bus using Verilog HDL, adhering to the specifications, and verifying it using the Universal Verification Methodology (UVM). The simulation outcomes demonstrate that the data retrieved from a designated memory location aligns precisely with the data initially written to that same memory location.

Vaishnavi R.K, Bindu. S, and Sheik Chandbasha conducted research titled [3] "Design and Verification of APB Protocol using System Verilog and Universal Verification Methodology" in June 2019. The paper authored by Vaishnavi R.K, Bindu. S, and Sheik Chandbasha, published in June 2019 in the International Research Journal of Engineering and Technology (IRJET), provides an overview of the AMBA bus architecture and extensively discusses the APB bus. The APB bus is meticulously designed using Verilog HDL, in compliance with the specified standards, and is subsequently verified using System Verilog and the Universal Verification Methodology (UVM). The simulation results affirm that the data retrieved from a specific memory location precisely matches the data originally written to that location.

Mukunthan J, A Daniel Raj, and Shruthi T authored a paper on [4] "Design and Implementation of AMBA APB Protocol" in 2021. The paper explains its significance in embedded system designs. It would discuss how the APB protocol facilitates communication between the CPU (Central Processing Unit) and peripherals in ARM-based systems. Protocol Specifications This section would outline the key specifications and features of the AMBA APB protocol. It would cover aspects such as the structure of APB transactions, addressing scheme, data transfer methods, and control signals design Architecture. The paper would describe the architecture proposed for implementing the AMBA APB protocol.

Dr. Dileep Reddy and Dr. Sathya Srikanth Palle published research on [5] "Design and Implementation of Advanced Peripheral Bus (APB) Protocol in System Verilog" in 2020.The paper authored by Dr. Dileep Reddy and Dr. Sathya Srikanth Palle, titled "Design and Implementation of Advanced Peripheral Bus (APB) Protocol in System Verilog,". The paper would initiate with an introduction to the Advanced Peripheral Bus (APB) protocol, emphasizing its role in embedded system designs, particularly within ARM-based architectures. It may discuss the necessity for streamlined communication between peripherals and the CPU, illustrating how the APB protocol effectively addresses this need. system Verilog Overview given the focus on implementing the APB protocol through System Verilog, the paper would offer an introduction to System Verilog, spotlighting its capabilities and benefits for hardware design and verification tasks design Specifications The authors would delineate the design specifications required for implementing the APB protocol in System Verilog. This entails defining the various protocol components, including address phases, data phases, control signals, and transaction types, ensuring a comprehensive understanding of the protocol's structure and functionality.

Panikkar A, Gavankar R, and Umarikar N conducted research on the [6] "Design and implementation of an area-efficient, low-power AMBA-APB Bridge for SoC." In 2022. The paper highlights the significance of bridges in SoC designs, particularly those enabling communication between different bus protocols such as AMBA and APB. It may underscore the importance of achieving area efficiency and low power consumption in contemporary SoC implementations. Background on AMBA and APB. This segment would furnish an overview of the AMBA and APB bus architectures, delineating their respective features, communication methodologies, and common applications within SoC designs. The authors would likely articulate the rationale behind developing an area-efficient and low-power AMBA-APB bridge. This rationale might encompass addressing the complexities associated with integrating peripherals utilizing diverse bus protocols into an SoC, all while minimizing area and power overheads.

Motivation

Utilizing System Verilog for APB design and verification holds paramount importance in the semiconductor sector, serving as the cornerstone for SoC advancement. This protocol elevates system efficiency, provides modularity to accommodate scalable designs, and undergoes meticulous verification to ensure reliability. Its practical implications span across a wide array of electronic devices, presenting substantial learning prospects and paving the way for potential research and innovation in digital system design and semiconductor technology.

2.1 Problem Statement

The digital interfaces in embedded systems are crucial for ensuring the correct functionality and reliability of modern microcontroller-based applications. The rapid evolution of System-on-Chip (SoC) designs demands robust communication protocols for seamless data transfer between integrated circuit components. The APB protocol holds significant importance in this field, demanding accurate implementation and thorough verification. This project tackles the necessity for a comprehensive framework in designing and verifying the APB protocol using System Verilog. It aims to enhance the dependability and effectiveness of data communication within contemporary digital systems.

2.2 Objectives

- To ensure compliance with the AMBA APB protocol, it's essential to guarantee that the designed AMBA APB interface strictly aligns with the specifications outlined in the protocol. This entails adhering to all prescribed control signals, data transfer mechanisms, and timing constraints specified within the AMBA APB protocol.
- To ensure that the AMBA APB interface operates correctly, enabling seamless communication between the central processing unit (CPU) and peripheral modules without errors.
- To design the AMBA APB interface in a modular manner to facilitate reusability and flexibility for various data widths, address spaces, and peripheral configurations.
- To Develop a comprehensive System Verilog testbench and a diverse set of test cases to rigorously verify the functionality of the AMBA APB interface, covering typical, edge-case, and error scenarios.

Chapter 3

INTRODUCTION TO AMBA APB

3.1 Regarding the APB protocol

The APB protocol stands out as a cost-effective interface, prioritizing minimal power usage and simplified interface structures. It operates on a non-pipelined, straightforward synchronous protocol, ensuring each transfer completes in at least two cycles. Widely adopted, the AMBA APB protocol serves as a prevalent standard for on-chip communication in system-on-chip (SoC) designs, especially prevalent in ARM-based architectures. Originating from ARM Holdings, the APB protocol streamlines communication among diverse components within an SoC, including CPUs, memory controllers, and peripheral devices.

The APB interface is tailored for accessing the programmable control registers found in peripheral devices. Usually, APB peripherals link to the primary memory system via an APB bridge. For instance, an AXI to APB bridge facilitates the connection of multiple APB peripherals to an AXI memory system. Initiating APB transfers falls under the responsibility of an APB bridge, also known as a Requester. Meanwhile, a peripheral interface acknowledges requests, alternatively known as a Completer. This specification will employ the terms Requester and Completer.

Transactions in the APB protocol consist of address and data phases, where the master device asserts control signals to indicate the start and end of each transaction. The protocol supports single-cycle transfers, allowing for relatively fast communication between components. One notable aspect of the APB protocol is its ease of integration into SoC designs, enabling efficient utilization of resources and minimizing design complexity. Additionally, its widespread adoption within ARM-based systems ensures compatibility and interoperability among various hardware components and peripherals the APB protocol include simplicity, low complexity, and efficiency, making it suitable for connecting peripheral devices with moderate bandwidth requirements. It operates as a simple master/slave protocol, where a master device initiates transactions to read from or write to slave devices.

3.2 AMBA's bus architecture

The Advanced Microcontroller Bus Architecture (AMBA) comprises a suite of interconnect specifications developed by ARM (Advanced RISC Machines) to facilitate the creation of high-performance, low-power systems-on-chip (SoC). Among its essential elements, the Advanced Peripheral Bus (APB) stands out, designed to furnish a cost-effective, energy-efficient interface for linking peripherals to the system. AMBA APB operates on a straightforward, low-frequency bus

protocol, commonly utilized for interfacing various low-bandwidth peripherals like timers, interrupt controllers, and supplementary components to the primary system bus. Specifically crafted for applications where top-tier performance isn't imperative, APB proves ideal for connecting slower peripherals.

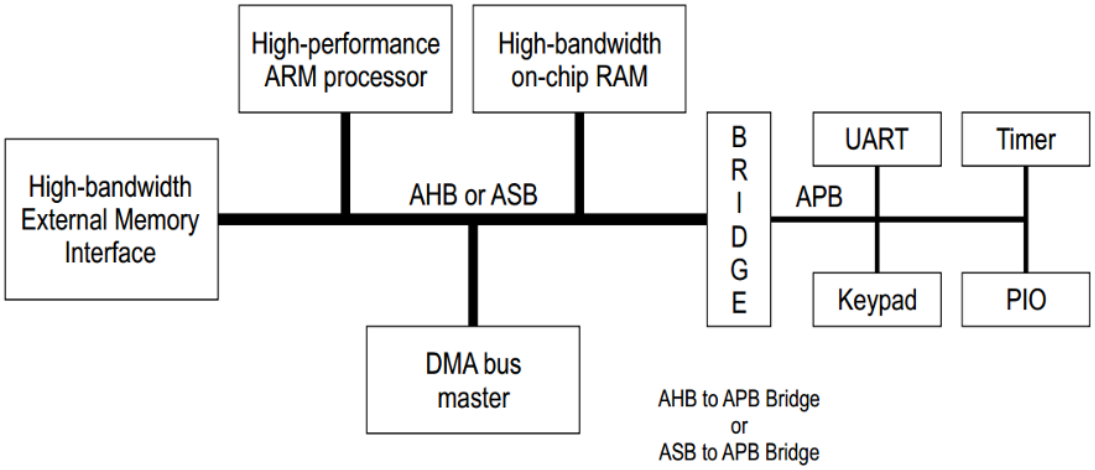

Figure 1: AMBA's bus architecture

AMBA encompasses a collection of interconnect protocols facilitating communication among diverse components within a computer system, including processors, peripherals, and memory. Illustrated in the diagram are an advanced ARM processor and high-bandwidth on-chip RAM, with a central bridge connecting the processor to the external memory interface. Various bridges exist, tailored to different bus types, such as the AHB to APB Bridge or ASB to APB Bridge.

The Advanced High-performance Bus is engineered for swift, high-bandwidth transfers between masters and slaves, while the Advanced System Bus serves as a high-performance bus primarily utilized for memory system interfaces. Contrarily, the Advanced Peripheral Bus caters to simpler peripherals not necessitating high-performance transfers and emphasizes low-power consumption. Notable peripherals include the UART for serial communication, Timer for generating regular electrical pulses, Keypad for data entry, PIO for byte-by-byte data transfer through a parallel port, and DMA, a technique enabling peripherals to transfer data directly to memory without CPU involvement.

3.3 Block diagram of APB

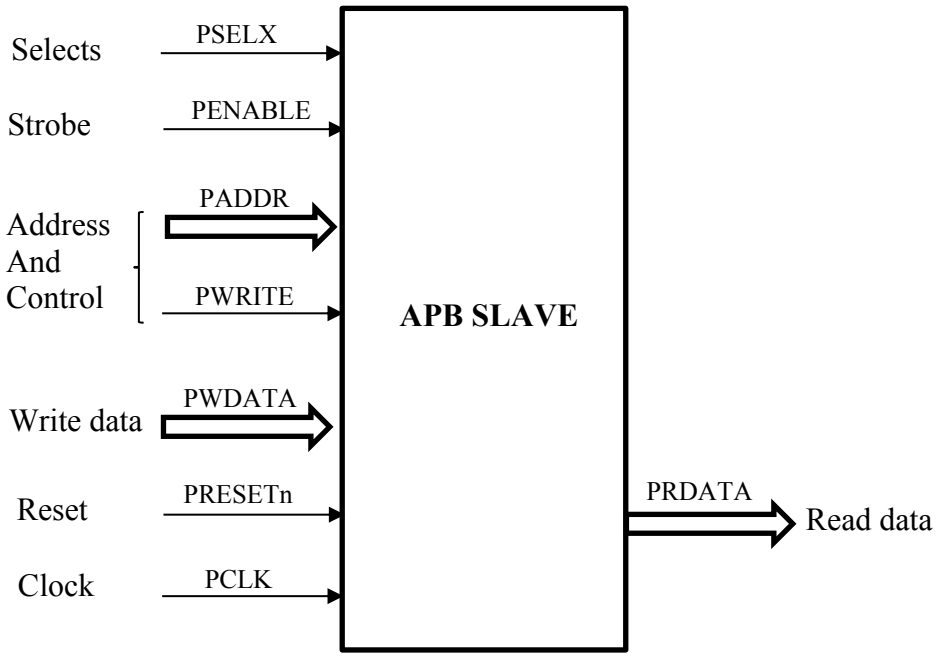

Figure 2: Slave Architecture of APB

The primary block diagram of the AMBA APB depicted in Figure 2 illustrates the fundamental interface signals. Designed in accordance with the specifications, the Advanced Peripheral Bus (APB) configuration is presented in Figure 2. Within this framework, the APB slave module receives PSEL, PENABLE, PWRITE (Address and Control), PRESETn, and PCLK as input signals. PADDR and PWDATA serve as 32-bit input control signals from the bridge and furnish 32 bits of PRDATA as output.

All signals within the AMBA APB architecture are denoted with a single letter P prefix. Certain signals within the APB, such as the clock, may be directly linked to the corresponding signals within the system bus. The APB Slave module boasts a straightforward yet adaptable interface, facilitating its utilization across various slave interfaces. It executes the subsequent functions effectively.

Chapter 4

DESCRPTION OF SIGNALS

This section delineates the signals of the APB interface. While certain signals maintain a fixed width, others possess variable widths. In instances where the width isn't predetermined, it's articulated through a property. A property value of zero denotes the absence of the signal on the interface.

Signal	Source	Width	Description
PCLK	Clock	1	Clock. **PCLK** is a clock signal. All APB signals are timed against the rising edge of **PCLK**.
PRESETn	System bus reset	1	Reset. **PRESETn** is the reset signal and is active-LOW. **PRESETn** is normally connected directly to the system bus reset signal.
PADDR	Requester	ADDR_WIDTH	Address. **PADDR** is the APB address bus. **PADDR** can be up to 32 bits wide.
PPROT	Requester	3	Protection type. **PPROT** indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access.
PSELx	Requester	1	Select. The Requester generates a **PSELx** signal for each Completer. **PSELx** indicates that the Completer is selected and that a data transfer is required.
PENABLE	Requester	1	Enable. **PENABLE** indicates the second and subsequent cycles of an APB transfer.
PWRITE	Requester	1	Direction. **PWRITE** indicates an APB write access when HIGH and an APB read access when LOW.
PWDATA	Requester	DATA_WIDTH	Write data. The **PWDATA** write data bus is driven by the APB bridge Requester during write cycles when **PWRITE** is HIGH. **PWDATA** can be 8, 16, or 32 bits wide.
PSTRB	Requester	DATA_WIDTH/8	Write strobe. **PSTRB** indicates which byte lanes to update during a write transfer. There is one write strobe for each 8 bits of the write data bus. **PSTRB[n]** corresponds to **PWDATA[(8n + 7):(8n)]**. **PSTRB** must not be active during a read transfer.

Signal	Source	Width	Description
PREADY	Completer	1	Ready. **PREADY** is used to extend an APB transfer by the Completer.
PRDATA	Completer	DATA_WIDTH	Read data. The **PRDATA** read data bus is driven by the selected Completer during read cycles when **PWRITE** is LOW. **PRDATA** can be 8, 16, or 32 bits wide.
PSLVERR	Completer	1	Transfer error. **PSLVERR** is an optional signal that can be asserted HIGH by the Completer to indicate an error condition on an APB transfer.
PWAKEUP	Requester	1	Wake-up. **PWAKEUP** indicates any activity associated with an APB interface.
PAUSER	Requester	USER_REQ_WIDTH	User request attribute. **PAUSER** is recommended to have a maximum width of 128 bits.
PWUSER	Requester	USER_DATA_WIDTH	User write data attribute. **PWUSER** is recommended to have a maximum width of DATA_WIDTH/2.
PRUSER	Completer	USER_DATA_WIDTH	User read data attribute. **PRUSER** is recommended to have a maximum width of DATA_WIDTH/2.
PBUSER	Completer	USER_RESP_WIDTH	User response attribute. **PBUSER** is recommended to have a maximum width of 16 bits.

4.1 APB Master and APB Slave

- With only one bus master on the APB, an arbiter becomes unnecessary.
- The master oversees both the address and write buses, performing a combinatorial decode of the address to decide which PSELx signal to activate.
- Additionally, it manages the timing of the transfer by controlling the PENABLE signal.
- In a read transfer, it transmits APB data onto the system bus.

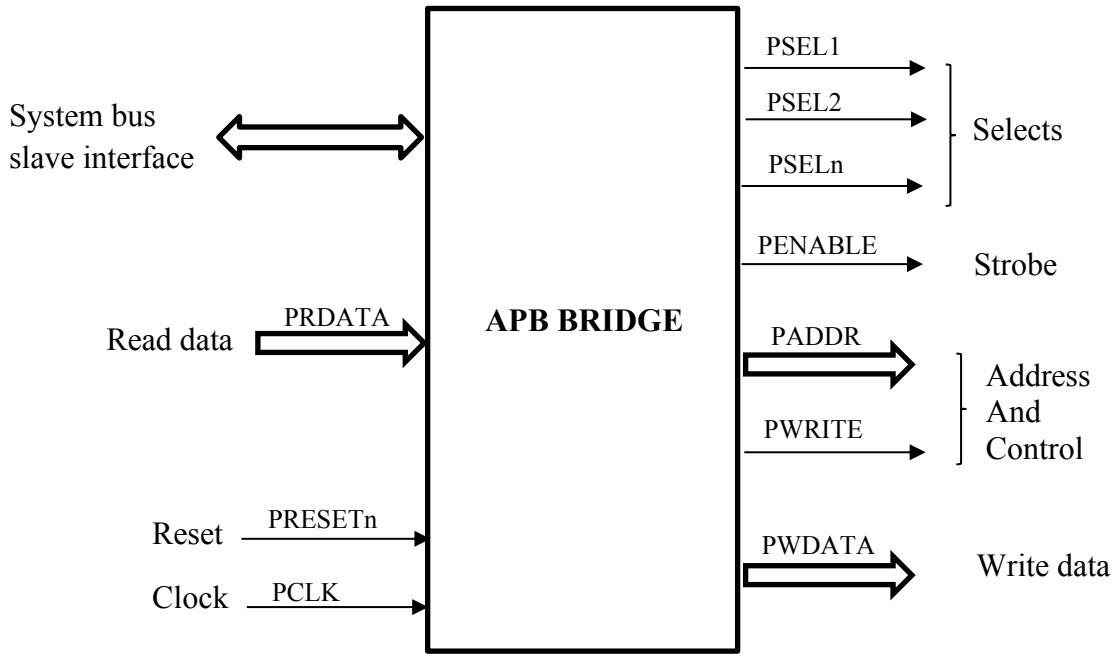

Figure 3: APB Master

- The interface of APB slaves is characterized by its simplicity and flexibility.
- The specific configuration of this interface will vary based on the chosen design approach, allowing for numerous potential options.
- Two crucial signals in this interface, PSLVERR and PREADY, primarily ensure data integrity during data transfer processes.

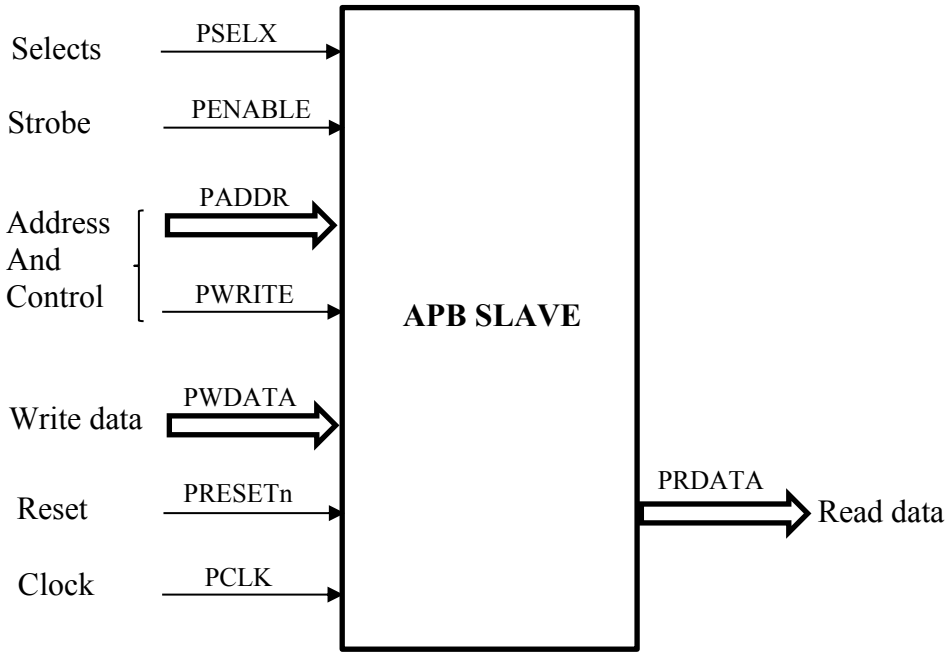

Figure 4: APB Slave

4.2 Communication of Master to Slave

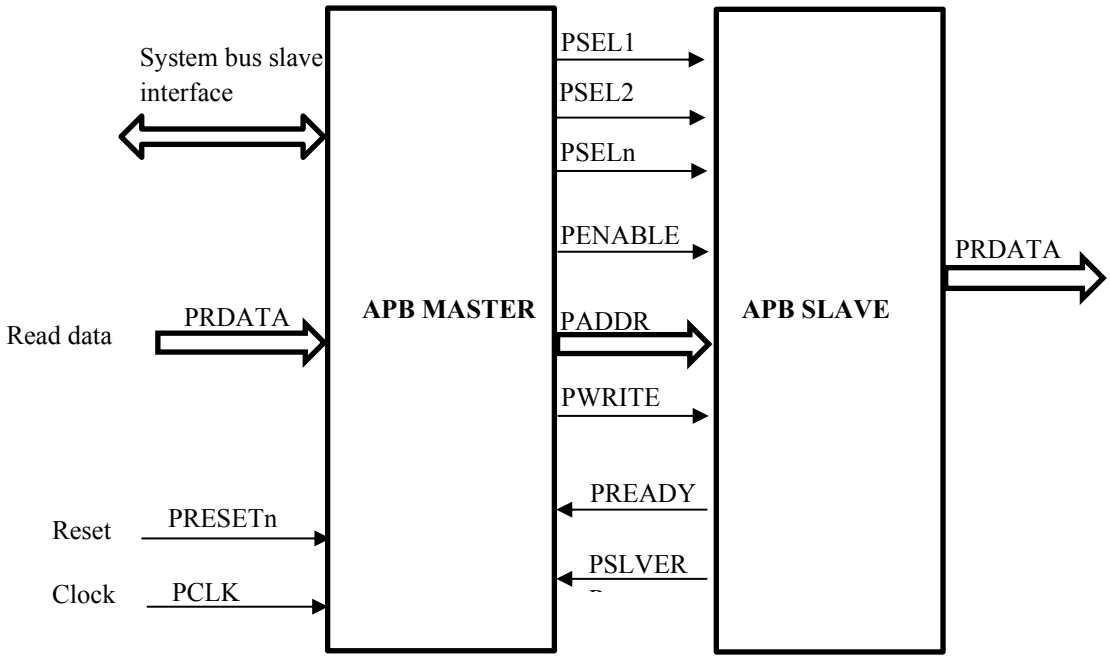

Figure 5: Communication of Master to Slave

• **System bus slave interface:** The interface connecting the peripheral device to the system bus.

• **PCLK:** Symbolizing the system clock.

• **PRESETn:** An active-low asynchronous reset signal.

• **PADDR:** Signifying the address bus from the master to the slave, width of up to 32 bits.

• **PWDATA:** The write data bus from the master to the slave, potentially up to 32 bits wide.

• **PRDATA:** The read data bus from the slave to the master, with a potential width of up to 32 bits.

• **PSELx:** Slave select signals, with one PSEL signal designated for each slave linked to the master.

• **PENABLE:** The enable signal for the data bus.

• **PWRITE:** Indicating a write transfer.

• **PRREADY:** Utilized by the slave to signify readiness to receive data during a write transfer.

• **PSLVER:** Employed by the slave to confirm the validity of data during a read transfer.

These signals facilitate communication between master and slave devices. To initiate a data transfer, the master places the address and data (for a write transfer) on the bus while asserting the necessary control signals. Upon recognizing its address on the PADDR bus, the slave device responds by asserting the PREADY signal, indicating readiness to receive data. Once the data is received, the slave asserts the PSLVER signal to confirm the validity of data on the PRDATA bus. Subsequently, the master retrieves the data from the PRDATA bus.

4.3 Operational States

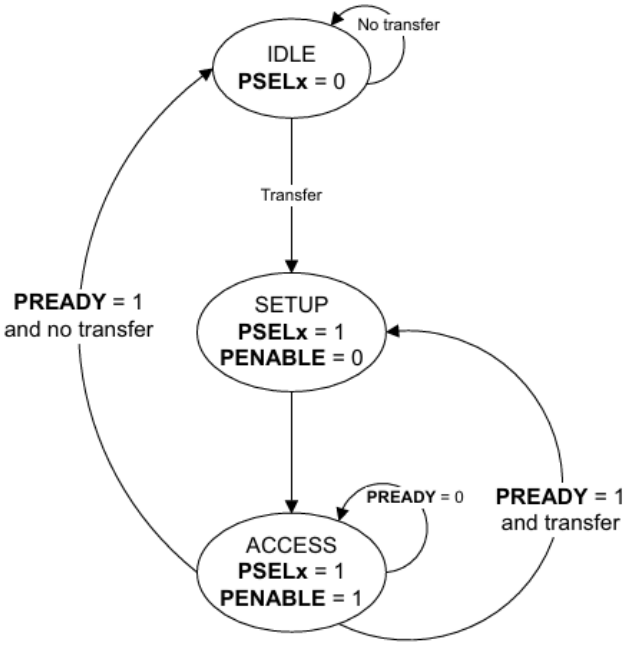

Figure 6: Diagram depicting States

The state machine progresses through the following states:

- **IDLE:** This represents the default state of the APB interface.
- **SETUP:** When a transfer is initiated, the interface transitions into the SETUP state. Here, the relevant select signal, PSELx, is activated. The interface stays in the SETUP state for only one clock cycle and invariably transitions to the ACCESS state on the subsequent rising clock edge.
- **ACCESS:** In the ACCESS state, the enable signal, PENABLE, is activated. It's crucial that the following signals remain unchanged during the transition from SETUP to ACCESS and between cycles within the ACCESS state:
    - **PADDR**
    - **PPROT**
    - **PWRITE**
    - **PWDATA,** only for write transactions
    - **PSTRB**
    - **PAUSER**
    - **PWUSER**

The ACCESS state termination is regulated by the PREADY signal originating from the Completer

- Should the Completer maintain PREADY at a LOW level, the interface persists in the ACCESS state.
- Conversely, if the Completer sets PREADY to a HIGH state, the ACCESS state concludes. If further transfers are unnecessary, the bus reverts to the IDLE state. Alternatively, if another transfer is imminent, the bus transitions directly to the SETUP state.

4.4 Transfers

4.5.1 Write transfers

This section outlines various types of write transfers:

1. **Transfer without any wait states**

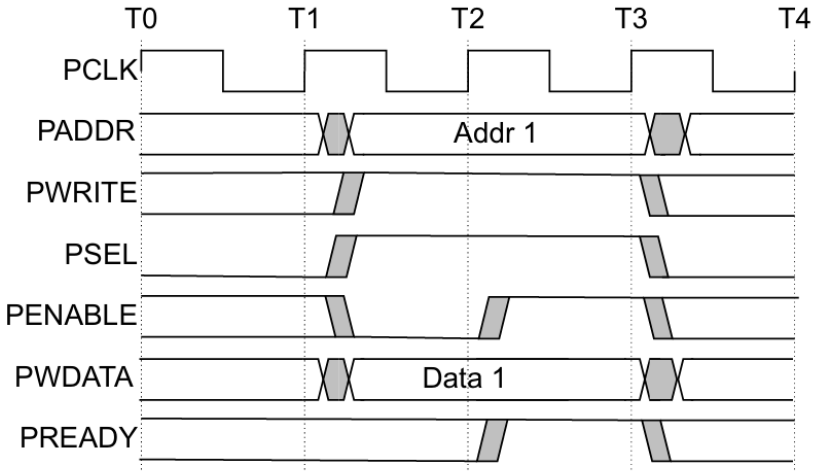

Figure 7: Transfer without wait states

The initiation of the write transfer occurs when the address, write data, write signal, and select signal all undergo changes subsequent to the rising edge of the clock.

- The initial clock cycle of the transfer, termed the setup phase, ensues.
- Subsequent to the ensuing clock edge, the enable signal PENABLE is asserted, signaling the commencement of the Access phase.
- Throughout the Access phase, the address, data, and control signals remain valid, culminating in the completion of the transfer.
- At the termination of the transfer, PENABLE is de-asserted.
- PSELx transitions to a LOW state unless the transfer is promptly succeeded by another transfer to the same peripheral.
- **T1**: The master asserts all the necessary signals to initiate the write transfer. These signals include:

- PCLK: System clock signal.
- PADDR: Address bus that specifies the memory location where the data needs to be written.
- PWRITE: Write control signal (active high).
- PSEL: Slave select signal to identify the specific slave device involved in the transfer.
- PENABLE: Enable signal for the data bus.
- PWDATA: Write data bus that carries the data to be written (Data 1).

- **T2**: The slave asserts the PREADY signal. This indicates to the master that the slave is ready to accept the data on the PWDATA bus. Since PREADY is asserted in the same clock cycle (T2) as the other control signals by the master, there are no wait states inserted by the slave device in this transfer.
- **T3**: The write transfer is complete. The master deasserts PENABLE and PSEL signals.

2. Transfer with wait states

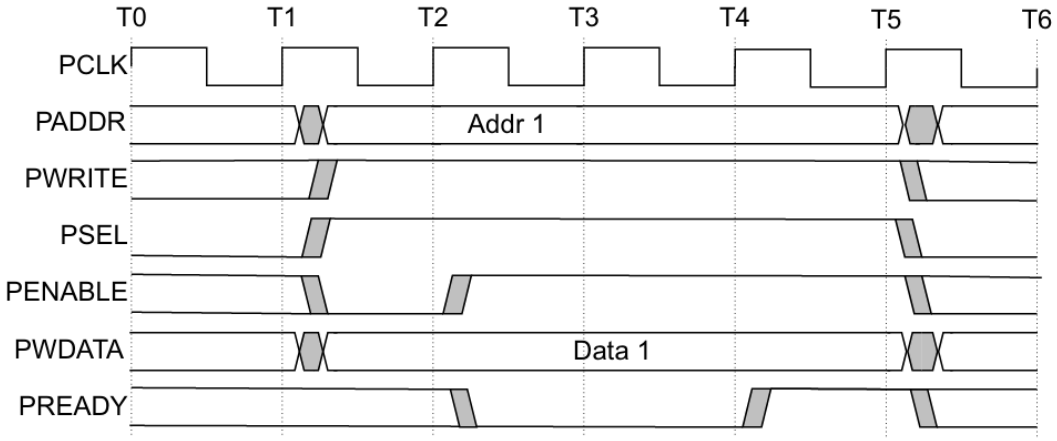

Figure 8: Transfer with wait states

The transfer can be prolonged by the PREADY signal received from the slave.

- In an ongoing Access phase, characterized by PENABLE being HIGH, extending the transfer entails driving PREADY LOW.
- The subsequent signals remain static for the additional cycles:
  - PADDR
  - PWRITE
  - PSEL
  - PENABLE
  - PWDATA

- **T0:** The master initiates communication by activating the PCLK, PADDR, PWRITE (set to HIGH for a write), PSEL, and PENABLE signals. This signifies the intent to write Data 1 to a specified address, Addr 1.
- **T1:** The slave device sets the PREADY signal low, transitioning into a waiting state, indicating to the master its unreadiness to receive data presently. Despite this, the master maintains control over all other signals, including PCLK, PADDR, PWRITE, PSEL, and PENABLE.
- **T2-T5:** These intervals represent wait states introduced by the slave device. The slave keeps PREADY low, indicating its unreadiness.
- **T6:** The slave reasserts the PREADY signal, signifying its readiness to accept the data.
- During the wait states (T1 to T6), the master continues to drive the PCLK, PADDR, PWRITE, PSEL, PENABLE, and PWDATA (Data 1) signals.

4.5.2 Read transfers

This section outlines various types of Read transfers:

1. **Transfer without any wait states**

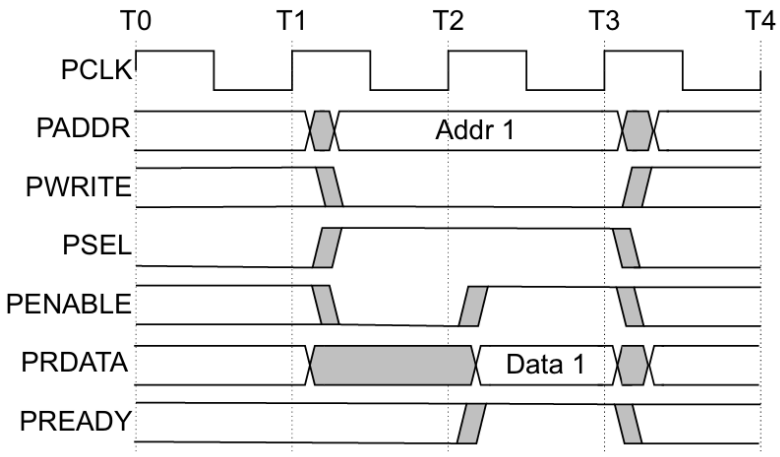

Figure 9: Transfer without any wait states

The initiation of the read transfer occurs when the address, write signal, and select signal undergo changes following the rising edge of the clock.

- The initial clock cycle of the transfer, known as the Setup phase, follows.
- Subsequent to the ensuing clock edge, the assertion of PENABLE indicates the commencement of the Access phase.
- Throughout the Access Phase, the address and control signals retain their validity.
- It is imperative for the slave to furnish the data before the conclusion of the read transfer.
- The transfer reaches completion at the end of this cycle.

- PENABLE is de-asserted upon the conclusion of the transfer.
- **T0**: The master asserts all the necessary signals to initiate the read transfer. These signals include:
    - PCLK: System clock signal
    - PADDR: Address bus that specifies the memory location from where the data needs to be read (Addr 1).
    - PWRITE: Write control signal (active LOW for a read transfer).
    - PSEL: Slave select signal to identify the specific slave device involved in the transfer.
    - PENABLE: Enable signal for the data bus.
- **T1**: The slave activates the PREADY signal simultaneously with transmitting Data 1 on the PRDATA bus. This signals to the master that the slave is prepared to deliver the data. As PREADY is asserted concurrently with the other control signals by the master in the same clock cycle (T1), there are no wait states introduced by the slave device in this transfer.
- **T2**: Upon completion of the read transfer, the master retrieves Data 1 from the PRDATA bus and removes the assertions of the PENABLE and PSEL signals.

This transfer method is optimal for high-performance systems where reducing transfer times is crucial. Ensuring that the slave device has the data readily available is essential to prevent reading outdated or inaccurate data. The capability to execute read transfers without introducing wait states relies on the functionalities of the individual slave device.

2. Transfer with wait states

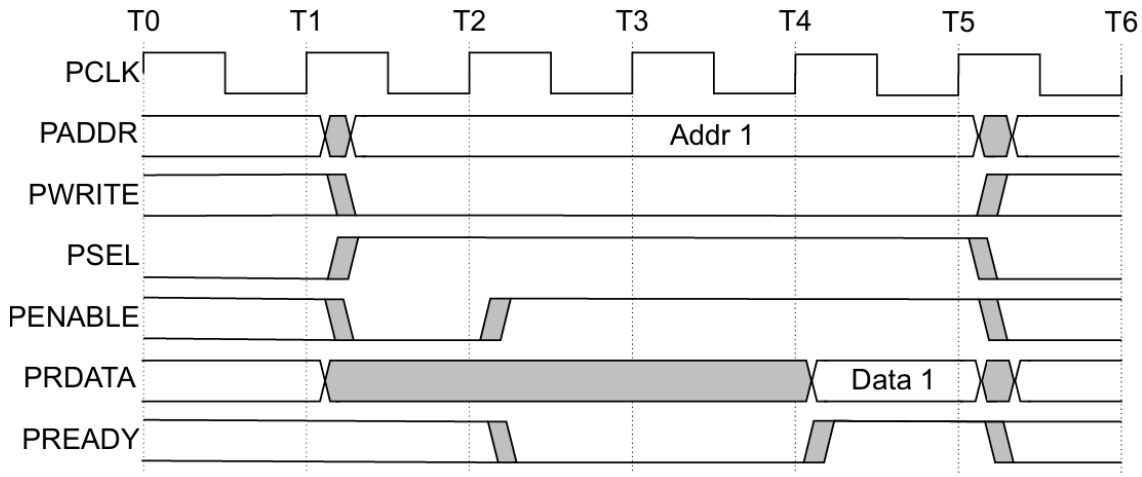

Figure 10: Transfer with wait states

The transfer undergoes extension when PREADY is lowered during an Access phase. The subsequent signals maintain their state while PREADY remains in a LOW state. The Wait states are inserted by the slave device to indicate to the master that it is not ready to provide the requested data yet.

- **T0**: The master asserts the PCLK, PADDR, PWRITE (which is LOW for a read), PSEL, and PENABLE signals. This indicates to the slave device that the master is requesting to read data from a specific address.
- **T1**: The slave device asserts the PREADY signal to indicate that it is ready to accept the request. It then deasserts PREADY to enter a wait state.
- **T2-T3**: These are wait states inserted by the slave device. The slave is not ready to provide the data yet, possibly due to some internal operations it needs to complete first. It keeps PREADY low to signal this to the master. The master continues to drive the address, PWRITE, PSEL and PENABLE signals during this time.
- **T4**: The slave asserts PREADY again to indicate that it has the data ready.
- **T5**: The master reads the data (Data 1) from the PRDATA bus.
- **T6**: The master deasserts PENABLE and PSEL. The read transfer is complete.

4.6 Error response

PSLVERR serves to signal an error condition during an APB transfer, applicable to both read and write transactions. Its validity is confined to the last cycle of an APB transfer, characterized by PSEL, PENABLE, and PREADY being simultaneously asserted. While it is recommended that PSLVERR be set to LOW when PSEL, PENABLE, or PREADY are not asserted, it is not mandatory.

In case of an error, it is permissible for the peripheral state to remain unchanged, as this behavior is peripheral-specific. An error in a write transaction does not imply that the peripheral register has not been updated. In the case of read transactions encountering an error, the returned data may be invalid. Despite receiving an error response to a read transfer, a Requester may still utilize the data. It should be noted that a Completer is not obliged to support PSLVERR. In instances where PSLVERR is not supported by a Completer, the appropriate input to the Requester is connected to LOW.

4.6.1 Write Transfer of APB

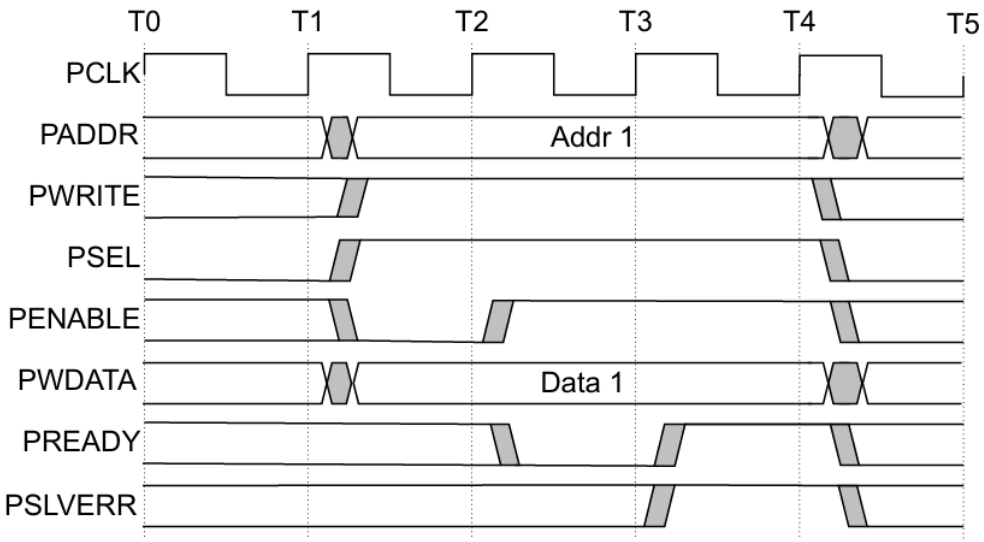

Figure 11: Unsuccessful write transfer

4.6.2 Read Transfer of APB

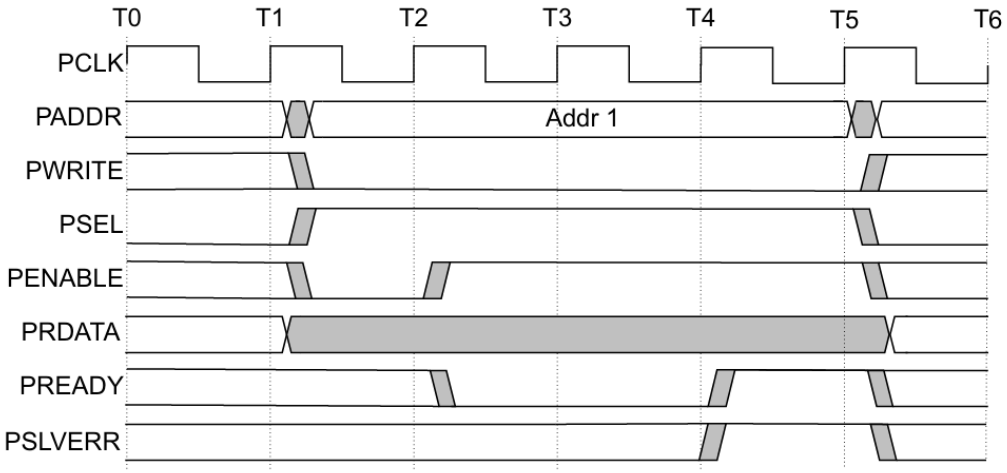

Figure 12: Unsuccessful read transfer

A read transfer can conclude with an error response, signifying the absence of valid read data. This error condition is denoted by PSLVERR in an APB transfer. Such errors can manifest in both read and write transactions. PSLVERR is deemed valid solely during the final cycle of an APB transfer, specifically when PSEL, PENABLE, and PREADY are all asserted. While it is advisable to set PSLVERR to LOW when it is not being sampled, i.e., when any of PSEL, PENABLE, or PREADY are LOW, it is not obligatory.

Chapter 5

METHODOLOGY

Methodology for Design and Verification

1. **Understand the AMBA APB Protocol:** Begin by thoroughly understanding the AMBA APB protocol. Study the official documentation to grasp the details of the protocol's specifications.
2. **Specification and Requirements:** Define the specifications and requirements for your AMBA APB interface. Specify data width, address width, clock frequency, and any specific project requirements.
3. **High-Level Architecture:** Create a high-level architectural design of your AMBA APB interface. Identify the key components, including the APB controller, APB bus interface, and peripheral modules.
4. **Module Design:** Design individual modules for your AMBA APB interface. Key modules include APB Controller, Design the APB controller to manage bus operations, including read and write requests. APB Bus Interface, Design the APB bus interface responsible for address decoding, data multiplexing, and response generation, Peripheral Modules Create specific peripheral modules to connect to the AMBA APB bus.
5. **Testbench Development:** Create a comprehensive System Verilog testbench to verify the functionality of your AMBA APB interface. This testbench should include stimulus generation and response checking.
6. **RTL Coding:** Implement the APB protocol design in System Verilog, emphasizing coding guidelines, naming conventions, and structured coding practices to enhance readability and maintainability. Utilize data structures like arrays and structs to organize and manage protocol specific information efficiently.
7. **Simulation:** Simulate your design using a System Verilog simulator (e.g., ModelSim, VCS, or Questa). Execute the testbench with the generated test cases to validate the design.
8. **Performance Verification:** Evaluate the performance of the APB protocol design by conducting simulations with varying traffic loads, transaction rates, and data transfer sizes. Analyse the results to ensure that the design meets specified throughput and latency requirements.

Chapter 6

Hardware/ Software Requirements

Software Requirements:

1. **HDL Development Environment:** An Integrated Development Environment (IDE) for HDL programming, such as Vivado Design Suite (Xilinx), Cadence Virtuoso, Quartus Prime (Intel/Altera), or any other vendor-specific toolchain compatible with System Verilog.
2. **Simulation Tool:** A simulation tool for verifying the APB protocol design, such as ModelSim (from Mentor Graphics), VCS (from Synopsys), or XSIM (included in Xilinx Vivado), or Cadence Sim vision.
3. **Version Control System**: A version control system (e.g., Git) to manage and track changes in the design files, facilitating collaboration and version management.
4. **Verification and Testbench Tools:** Tools for creating and managing the verification environment, including System Verilog, UVM (Universal Verification Methodology), and any additional verification libraries or methodologies.
5. **Miscellaneous Tools:** Additional tools, as needed, for tasks such as waveform viewing, code analysis, debugging, Code Coverage. For example, GTKWave for waveform viewing, Cadence imc gui for Code Coverage. It's important to ensure compatibility between the chosen hardware and software tools, as well as verify system requirements specified by the vendors of the development tools.

Chapter 7

RESULTS

1. **Correctly Designed AMBA APB Interface:** The primary expected result is a correctly designed AMBA APB interface that adheres to the AMBA APB protocol specification and the specified project requirements. This interface should include the APB controller, the APB bus interface, and any peripheral modules, all working together seamlessly.
2. **Functional Correctness:** The designed AMBA APB interface should function correctly, allowing for the error-free communication of data between the central processing unit (CPU) and connected peripheral modules. It should pass all functional test cases.
3. **Modularity and Reusability:** The design should exhibit modularity and reusability, allowing the interface to be adapted for different data widths, address spaces, and peripheral configurations without requiring a complete redesign.
4. **Simulation Results:** Simulation results should demonstrate that the AMBA APB interface performs as expected under various test cases and operational conditions. The simulation results should confirm functional correctness and adherence to the protocol.
5. **Functional Correctness and Compliance:** Verification results confirming that the APB protocol implementation adheres to the APB standard, ensuring proper data transfer, address decoding, and control signalling between APB master and slave components.
6. **Documentation and Reports:** Comprehensive documentation including design specifications, test plans, verification results, and any design decisions or trade-offs made during the project, providing clear and organized information for future reference.
7. **Timing Analysis Report:** If the design involves timing-critical elements, a timing analysis report should demonstrate that the AMBA APB interface meets the specified timing requirements and operates within acceptable latency limits.
8. **Regression Test Pass Report:** The AMBA APB interface should pass all regression tests, indicating that it remains robust and error-free even after design modifications and enhancements.
9. **Knowledge Transfer and Dissemination:** The successful sharing of knowledge and insights gained from the project through presentations, workshops, or publications, contributing t to the broader community of digital system design and verification.

7.1 Simulation Result

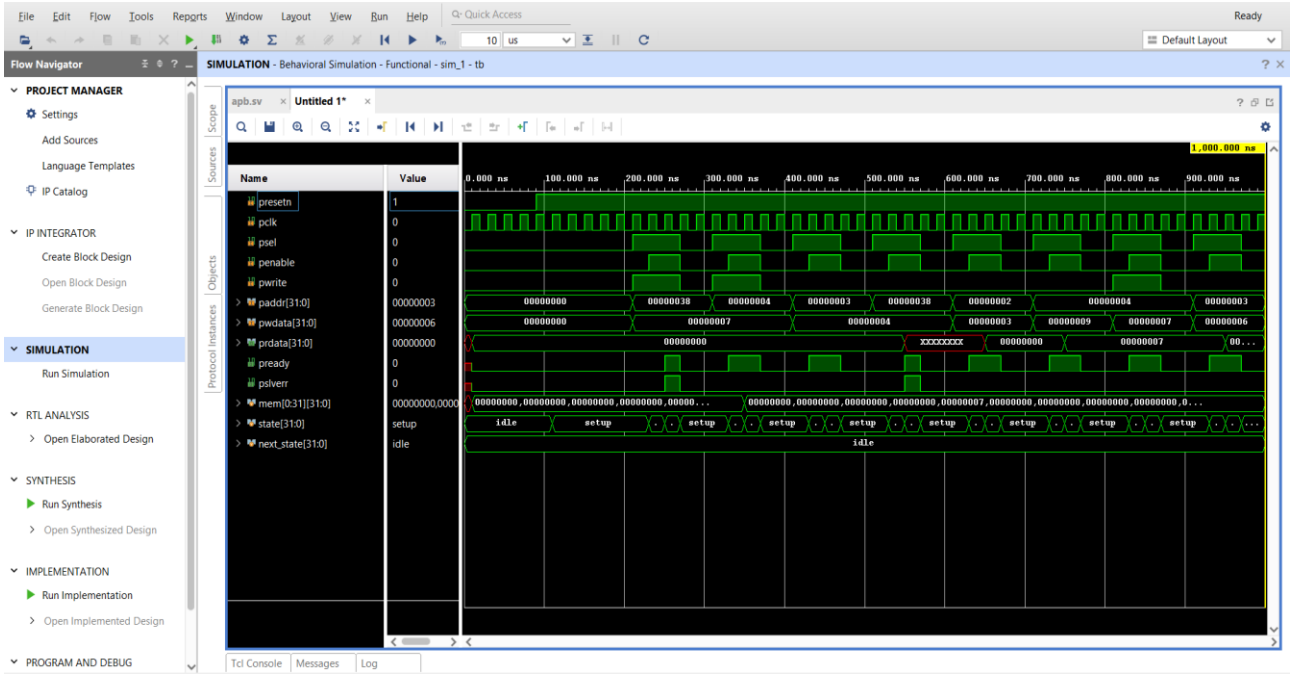

Figure 13: Simulation result

This waveform illustrates the simultaneous read and write operations of data transfer in AMBA APB when both `psel` and `penable` are HIGH.

7.2 Implementation

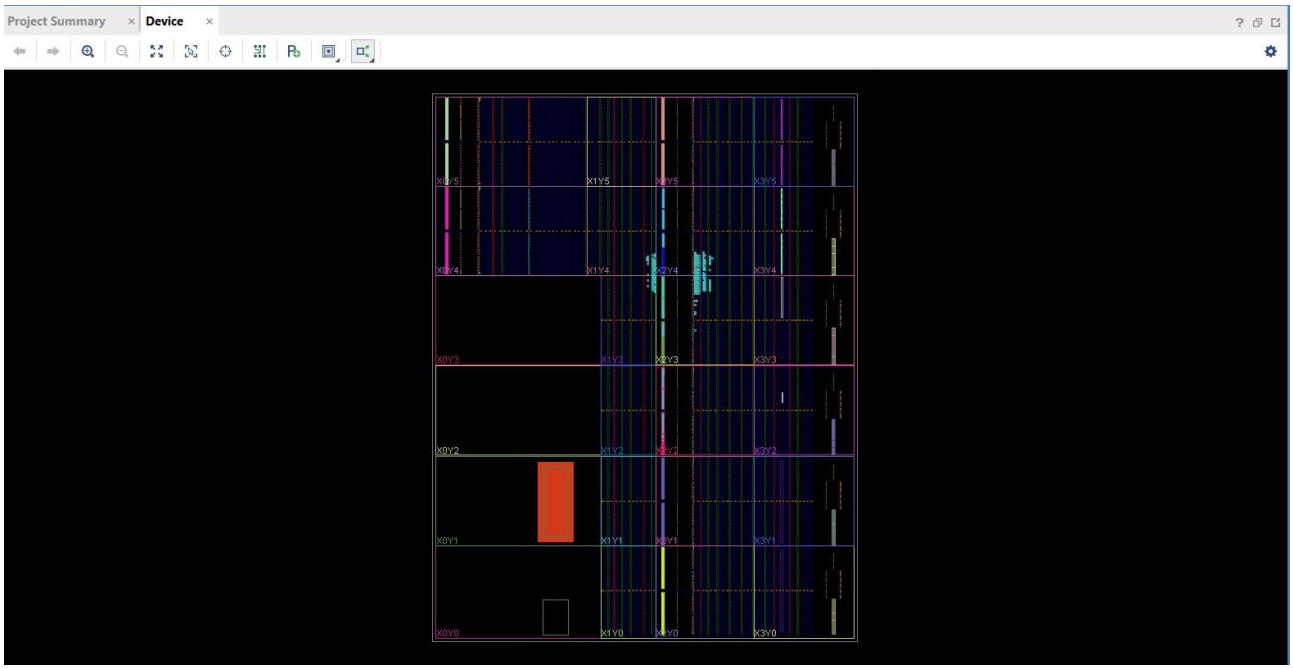

Figure 14: Implementation Design

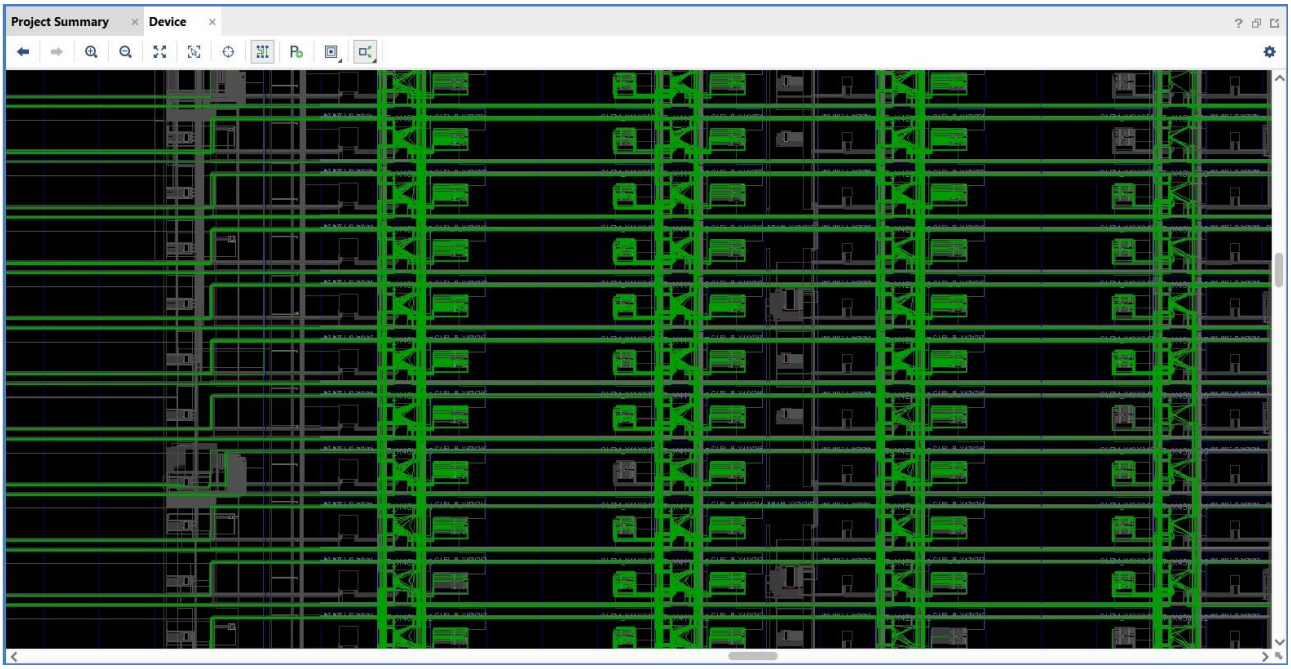

Figure 15: Enlarged View of Implementation

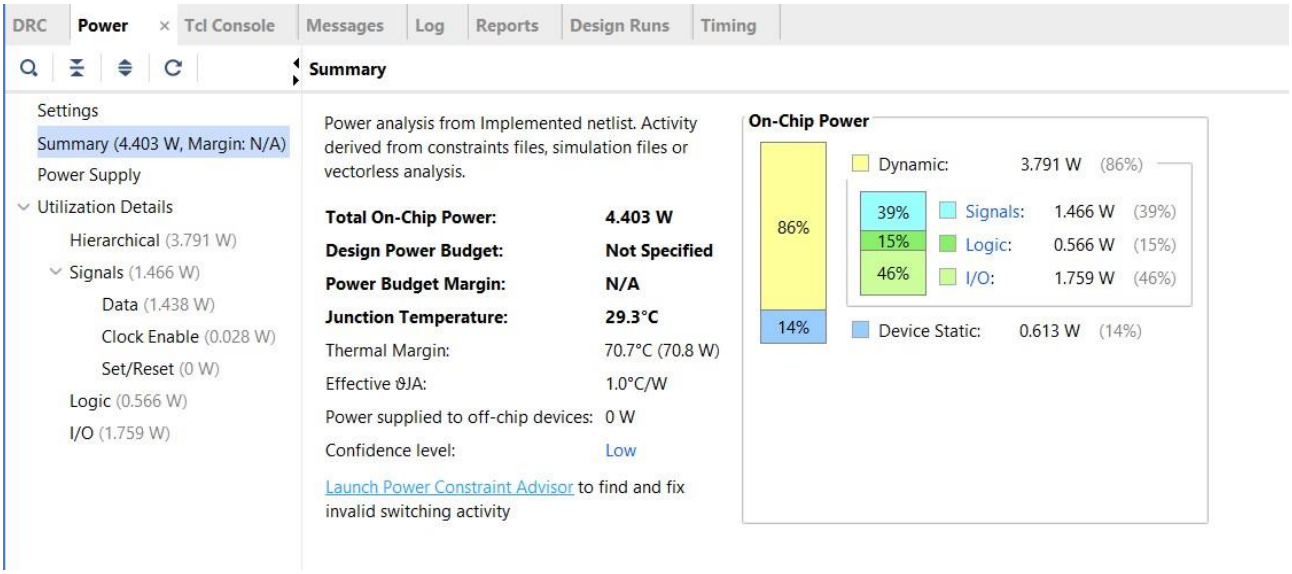

Figure 16: Power report

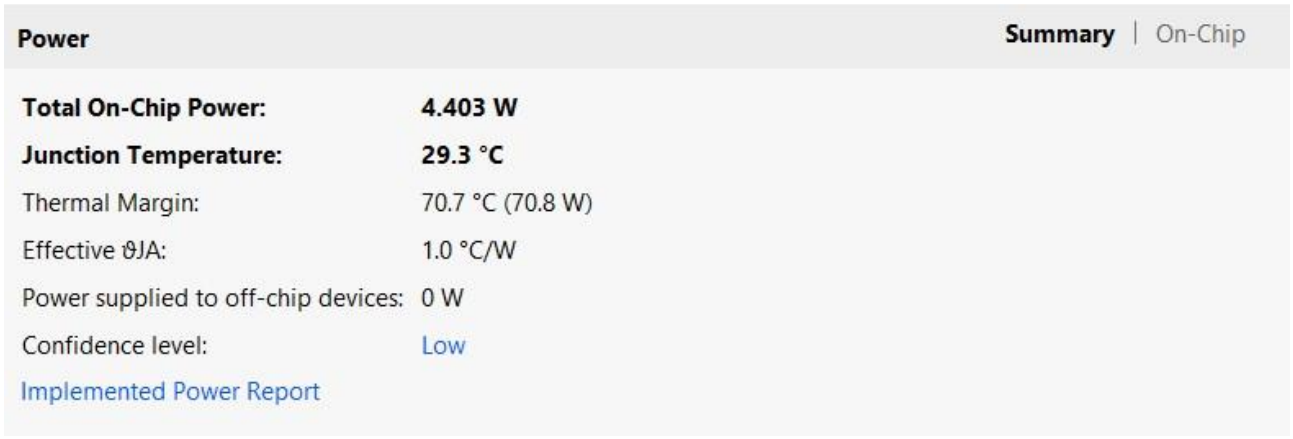

Figure 17: Implemented Power report

- **Summary (4.403 W, Margin: N/A)**: This section likely summarizes the total estimated power consumption of the design, which is 4.403 watts (W) in this case. The margin seems to be not available (N/A).
- **Power Supply**: This section shows the breakdown of the power consumption by voltage source. In the image, there is a single unnamed supply voltage with a dynamic power consumption of 3.791 watts, which is 86% of the total power consumption.
- **Utilization Details**: This section shows how the power is being consumed within the design. Here are some subcategories:
    - **Total On-Chip Power**: This is the total estimated power consumption of the design, which is 4.403 watts in this case.
    - **Design Power Budget**: This is the target or allowable power consumption for the design, which is not specified (Not Specified) in this case.
- **Hierarchical (3.791 W)**: This section shows the breakdown of power consumption across different parts of the design hierarchy. In this case, all the power consumption seems to be at the top level of the design hierarchy.
- **Signals (1.466 W)**: This is the estimated power consumption due to the switching activity of signals within the design.
    - **Data (1.438 W)**: This is the estimated power consumption due to the switching activity of data signals.
    - **Clock Enable (0.028 W)**: This is the estimated power consumption due to the switching activity of clock enable signals.
    - **Set/Reset (0 W)**: This is the estimated power consumption due to the switching activity of set and reset signals.

- **Logic (0.566 W)**: This is the estimated power consumption due to the logic gates within the design.
- **I/O (1.759 W)**: This is the estimated power consumption due to the input/output (I/O) buffers and pins of the design.

- **Junction Temperature**: This shows the estimated temperature at the junction of the die, which is 29.3 degrees Celsius (°C) in this case.
- **Power supplied to off-chip devices**: This shows the estimated power consumption of off-chip devices powered by the design, which is 0 watts (W) in this case.

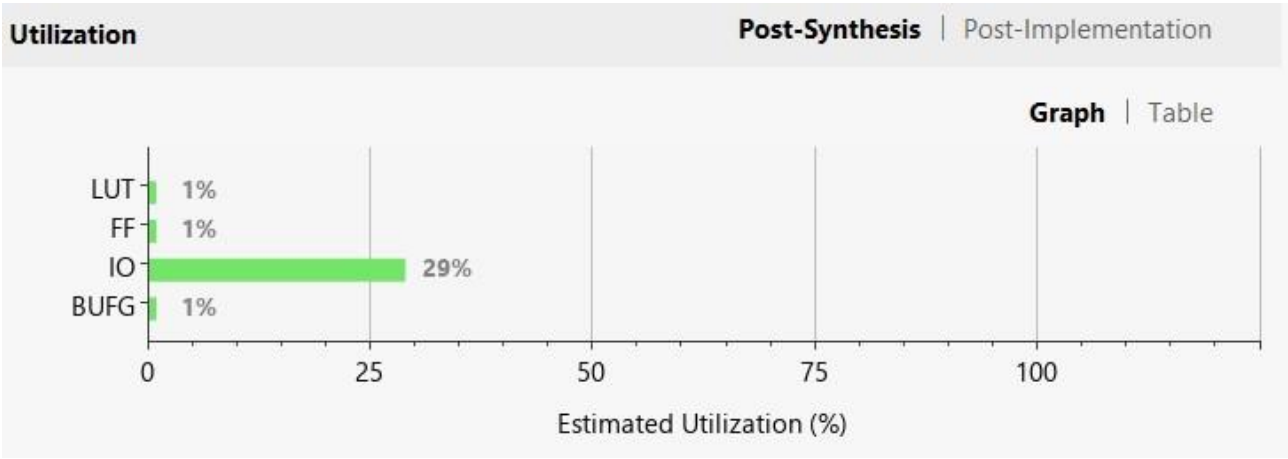

Figure 18: Post Implementation Power Consumption

- **LUT (Look-Up Table):** A LUT is a basic building block in an FPGA that can be used to implement logic functions.
- **FF (Flip-Flop):** A flip-flop is a circuit that can store a single bit of data.
- **BRAM (Block RAM):** A BRAM is a memory block that can be used to store data.
- **DSP (DSP Slice):** A DSP slice is a specialized unit that can be used to perform mathematical operations.
- **IOs (Input/Outputs):** The IOs are the pins on the FPGA that are used to communicate with other devices.
- **BUFG (Buffer Global Clock):** The BUFG is a circuit that provides a clock signal to the FPGA.

The graph shows that the utilization of most of the resources is very low, at around 1%. The only exception is BRAM, which is being used at 29%.

Chapter 8

ADVANTAGES AND APPLICATIONS

8.1 Advantages

1. **Standardization:** Designing the AMBA APB interface using System Verilog allows for adherence to a widely recognized industry standard. This ensures compatibility and interoperability with other ARM-based systems.
2. **Flexibility and Modularity:** System Verilog's flexibility and modularity enable designers to create customizable and reusable AMBA APB interfaces that can adapt to varying data widths, address spaces, and peripheral requirements.
3. **Efficiency:** The use of System Verilog allows for efficient design and verification. It offers a concise and expressive way to specify digital logic, potentially reducing development time.
4. **Verification Capabilities:** System Verilog provides a rich set of verification features, including advanced testbench development, assertions, and code and functional coverage analysis. This enhances the ability to rigorously verify the AMBA APB interface.
5. **Simulation Support:** System Verilog is supported by well-established simulation tools (e.g., ModelSim, VCS, Questa) that offer powerful simulation capabilities, making it easier to validate the design.

8.2 Applications

1. **System-on-Chip (soc) Designs:** APB protocol is widely used in the development of socs, facilitating efficient communication between processors, memory subsystems, and various peripherals.
2. **Embedded Systems:** APB is employed in embedded systems to manage communication between microcontrollers, memory units, and external devices like sensors, actuators, and communication modules.
3. **Networking Equipment:** In networking devices, APB protocol is crucial for managing data traffic between components such as processors, switches, and memory buffers.
4. **Graphics Processing Units (gpus):** gpus utilize the APB protocol to efficiently manage data transfer between processing units, cache, and memory modules.
5. **High-Performance Computing (HPC):** APB plays a vital role in HPC systems by ensuring efficient data movement between processing units and memory banks, optimizing computational tasks.

Chapter 9

CONCLUSION

The project on "Design and Verification of AMBA APB protocol using system Verilog" addresses the complexities involved in designing and verifying high-performance bus protocols for on-chip communication. Through the application of system Verilog, a powerful hardware description and verification language, the project demonstrates a methodical approach to model the AMBA APB protocol, highlighting its flexibility and efficiency in handling high-speed data transfers and multiple bus masters.

The utilization of system Verilog enabled the creation of a robust verification environment that effectively simulated various operational scenarios and corner cases, ensuring the reliability and functionality of the APB protocol design. The project's outcomes not only reinforce the importance of rigorous verification processes in the development of complex digital systems but also showcase the potential of system Verilog in streamlining these processes.

Moreover, the project contributes valuable insights into the challenges and solutions associated with implementing the AMBA APB protocol, serving as a useful reference for future developments in on-chip communication technologies. The methodologies and findings presented in this project could pave the way for further innovations in the field, potentially leading to more efficient and scalable system-on-chip architectures.

In conclusion, this project exemplifies the critical role of advanced verification techniques in ensuring the integrity and performance of digital communication protocols. It underscores the effectiveness of system Verilog in addressing the intricate requirements of modern on-chip communication standards, marking a significant step forward in the design and verification of complex electronic systems.

REFERNCES

[1] Anushka Dwivedi, Anand Jatti, “Design and Implementation of AMBA APB Protocol Using System Verilog”, International Research Journal of Modernization in Engineering Technology and Science, Vol. 4, Issue. 07, July 2022.

[2] Dipti Girdhar Shankar, “Design and Verification of AMBA APB Protocol”, International Journal of Computer Application, Vol. 95 Issue. 21, June 2014.

[3] Vaishnavi R.K, Bindu. S, Sheik Chandbasha “Design and Verification of APB Protocol using System Verilog and Universal Verification Methodology”, International Research Journal of Engineering and Technology (IRJET), Vol. 6, Issue. 06, June 2019.

[4] Mukunthan J, A Daniel Raj, Shruthi T “Design and Implementation of AMBA APB Protocol”, IOP Conference Series: Materials Science and Engineering, 2021.

[5] Dileep Reddy, Sathya Srikanth Palle “Design and Implementation of Advanced Peripheral Bus (APB) Protocol in System Verilog", GIS Science Journal, Vol. 9, Issue. 01, June 2022.

[6] Panikkar A, Gavankar R, Umarikar N “Design and implementation of area efficient, low power AMBA-APB Bridge for SoC”, International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE), March 2014.