Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 08: 2023
DOI10.17605/OSF.IO/QSD2V

ENHANCING VLSI VERIFICATION: INVESTIGATING APB AND AHB LITE PROTOCOLS FOR EFFICIENT IC COMMUNICATION

Dr. AJITH A S
Associate Professor, Department of ECE, MVJ College of Engineering, Bengaluru, India.
Email: ajithnce@gmail.com, ajith.as@mvjce.edu.in

P AKHIL REDDY
PG Student (M.Tech.), NIT, Calicut. Email: akhilreddypadala@gmail.com

Abstract

VLSI verification plays a pivotal role in ensuring the reliable functionality of integrated circuits (ICs). This paper delves into the investigation of the Advanced Peripheral Bus (APB) and Advanced High-performance Bus (AHB Lite) protocols, which are essential components in IC communication. The focal point of this study is the critical aspect of data transfer between master and slave cells. Both APB and AHB Lite protocols are instrumental in streamlining communication processes. The paper provides an in-depth understanding of these protocols and evaluates their efficiency in transmitting data from master to slave. The research emphasizes the importance of selecting a protocol that enables higher data transfer speeds, offers multiple transmission modes, ensures data security, and minimizes power consumption. The ultimate goal of this investigation is to identify protocols that enhance VLSI verification by optimizing IC communication. By uncovering the strengths and weaknesses of APB and AHB Lite, this study aims to contribute valuable insights into the selection and implementation of communication protocols for advanced integrated circuits.

Keywords: VLSI Verification, APB, AHB Lite, AMBA, Communication Protocols, IC Communication, Master-Slave Communication, Speed Optimization, Protocol Selection

1. INTRODUCTION

The field of Very Large-Scale Integration (VLSI) has revolutionized modern electronics, enabling the integration of complex functionalities into a single chip. As the demand for high-performance and power-efficient integrated circuits (ICs) continues to grow, ensuring the reliability and functionality of these intricate designs becomes increasingly challenging. VLSI verification, a critical phase in the chip design process, aims to validate the correctness and robustness of the ICs before they are fabricated.

One crucial aspect of VLSI verification is efficient IC communication, as it directly impacts the overall performance and power consumption of the chip. To facilitate seamless data exchange between various components of the IC, standardized communication protocols are employed. Among these, the Advanced Peripheral Bus (APB) and Advanced High-performance Bus Lite (AHB Lite) protocols, part of the widely used Advanced Microcontroller Bus Architecture (AMBA), are of prime focus in this study.

The APB and AHB Lite protocols cater to different communication requirements within the IC [4]. While APB follows a non-pipelined approach [15], AHB Lite offers enhanced performance with pipelining capabilities. The choice of the appropriate protocol significantly influences the speed, efficiency, and power consumption of the data transfer process.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 08: 2023
DOI10.17605/OSF.IO/QSD2V

In this paper, we delve into a comprehensive investigation of the APB and AHB Lite protocols, aiming to enhance VLSI verification through optimized IC communication [10]. We explore the fundamental concepts of these protocols, their strengths, and their limitations. Utilizing Verilog language and test-bench, we conduct synthesis and design experiments to evaluate their performance.

The primary objective is to identify the most efficient communication protocol that ensures swift data transfer, multiple transmission modes, data security, and minimal power consumption. We believe that such a protocol will not only accelerate VLSI verification but also contribute to the development of advanced integrated circuits with superior functionality and performance.

In the subsequent sections, we will delve into the intricacies of APB and AHB Lite protocols, examine their roles in master-slave communication within the IC, and analyse the impact of these protocols on data transmission efficiency. Through this exploration, we aim to provide valuable insights for selecting the most suitable communication protocol to optimize the VLSI verification process and advance the domain of integrated circuit design.

Fig 1 shows AMBA Bus architecture diagram. The AMBA bus architecture is a widely used and standardized on-chip communication protocol developed by ARM (Advanced RISC Machines) for designing complex System-on-Chips (SoCs) and integrated circuits. It defines a set of interconnect protocols that facilitate communication between different components within the SoC.

The primary components of the AMBA bus architecture are as follows:

Advanced High-performance Bus (AHB): The AHB is a high-performance bus protocol designed for fast and efficient communication between high-speed modules and components within the SoC [5]. It supports pipelining and is suitable for high-bandwidth data transfers.

Advanced Peripheral Bus (APB): The APB is a slower and simpler bus protocol compared to AHB. It is intended for connecting low-speed peripherals and other components that do not require high bandwidth.

Advanced Extensible Interface (AXI): The AXI is a more advanced and flexible interconnect protocol [2], designed to cater to the increasing complexity of modern SoCs. It supports multiple channels, burst transfers, out-of-order transactions, and is suitable for connecting complex subsystems and memory interfaces.

The AMBA bus architecture diagram would typically show these components interconnected within the SoC. The AHB and APB buses would connect various peripherals and modules to the central processing unit (CPU) or other processing elements. The AXI bus, being more versatile, could act as the main interconnect between different subsystems and memory elements, bridging the communication gap between high-performance and low-speed components.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 08: 2023
DOI10.17605/OSF.IO/QSD2V

Each bus (AHB, APB, and AXI) might have multiple masters and slaves connected to it, allowing for simultaneous data transfers and efficient data flow throughout the SoC.

It's important to note that the exact architecture and interconnections within an AMBA-based SoC can vary depending on the specific design and requirements of the system. However, the key components mentioned above are the fundamental building blocks of the AMBA bus architecture.

In fig 1 the ARM processor acts as the central processing unit (CPU) and is responsible for coordinating the communication between different components using the AMBA bus architecture.

Here's an explanation of the connections:

The ARM processor is the primary processing unit that executes instructions and controls the overall operation of the system.

The UART is a communication interface used for serial communication with external devices. It allows the SoC to exchange data with external peripherals or other systems [8]. Chip RAM serves as a fast-access memory that the CPU can use for storing and accessing data during its operations. It provides a high-speed memory interface for the processor.

The Memory Interface is responsible for connecting the CPU to the external memory subsystem. It facilitates data transfers between the processor and external memory, allowing the system to store and retrieve data efficiently. The Keypad DMA is a specialized component that enables direct memory access for data coming from a keypad or other input devices. It allows data from the keypad to be directly transferred to memory without involving the CPU, reducing CPU overhead and improving data transfer efficiency.

The AHB bus acts as a high-performance interconnect that connects high-speed modules and components within the SoC [1]. It links the ARM processor to other high-bandwidth components, such as the Memory Interface and Chip RAM. The APB bus serves as a simpler and slower interconnect compared to AHB. It connects low-speed peripherals, like UART and Keypad DMA, to the ARM processor. It provides a convenient means of communication for slower devices that don't require high bandwidth.

In this representation, the AMBA Bus architecture facilitates efficient communication between the ARM processor, UART, Chip RAM, Memory Interface, and Keypad DMA. The AHB bus is used to connect high-speed modules like the ARM processor, Memory Interface, and Chip RAM, while the APB bus connects lower-speed peripherals like the UART and Keypad DMA. This hierarchical arrangement optimizes data transfer within the system, ensuring high-performance and efficient operation of the SoC.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 08: 2023
DOI10.17605/OSF.IO/QSD2V

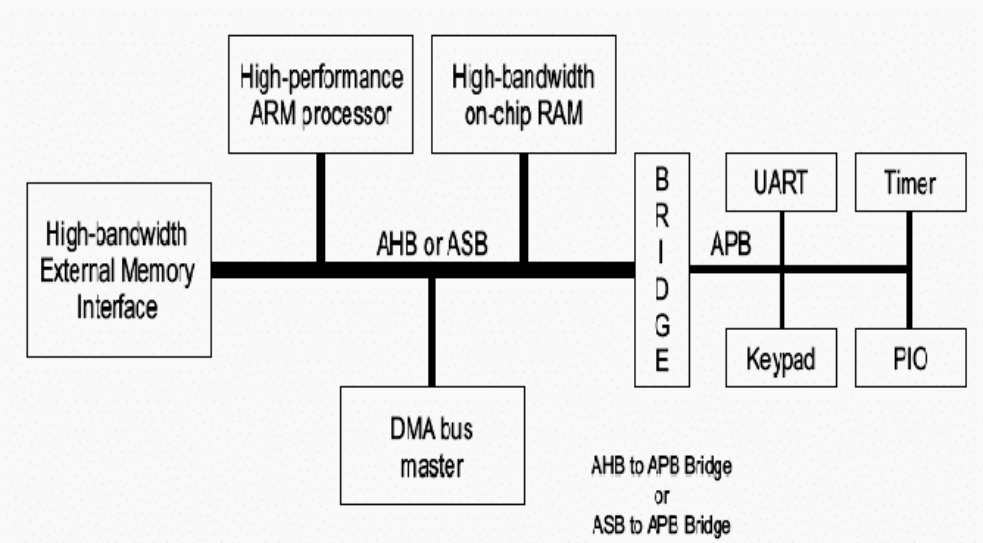

Fig. 1: AMBA Bus functional block diagram

While the AMBA standard defines the basic architecture and interconnect protocols, the User AMBA Bus architecture allows customization and extension to accommodate specific needs [14]. In the User AMBA Bus architecture, the primary components typically include:

User-specific Master(s):

These are custom masters that could be CPUs, DSPs, or other processing units tailored to the user's specific application needs. These masters initiate data transfers and control the overall communication within the system.

User-specific Slave(s):

These are custom slaves that may represent various peripherals, memory elements, or custom IP (Intellectual Property) blocks. The slaves respond to read and write requests from the masters and perform data transactions as per the protocol.

User-specific Interconnect:

This part of the architecture represents the interconnection mechanism between the masters and slaves. Depending on the application and performance requirements, the user can design a specific interconnect that suits the data transfer needs. It could be hierarchical, mesh, or any other custom topology.

AMBA Protocol Interfaces:

Although the User AMBA Bus architecture is customized, it often leverages the AMBA protocol interfaces (AHB, APB, AXI, etc.) to maintain compatibility with existing designs and standards. These interfaces provide a set of rules and protocols for communication between the masters and slaves.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 08: 2023
DOI10.17605/OSF.IO/QSD2V

User-defined Custom Protocols:

In certain cases, users might introduce their custom protocols to cater to unique requirements. These custom protocols would define how specific masters and slaves interact within the system.

User-specific Peripherals or IP blocks:

The User AMBA Bus architecture might include specific peripherals or custom IP blocks that are not part of the standard AMBA architecture. These could be sensor interfaces, accelerators, or any other components tailored to the application.

2. RELATED WORK

In paper [1], this defines the performance of ARM (Advanced RISC Machines) for high-speed and efficient data transfer between IP cores and subsystems within a System-on-Chip (SoC). It is part of the AMBA (Advanced Microcontroller Bus Architecture) family of protocols and is commonly employed in complex SoC designs to achieve high-performance communication.

Furthermore, the AMBA AXI3 protocol offers a powerful and efficient solution for high-speed communication within complex SoC designs. Its capabilities for high throughput, low latency, and flexibility make it well-suited for a wide range of applications, from consumer electronics to advanced computing systems.

In paper [2], deals with the automated design and generation of complex System-on-Chip (SoC) architectures. In the context of VLSI (Very Large Scale Integration) design, SoC refers to an integrated circuit that combines multiple functional blocks or IP (Intellectual Property) cores, such as processors, memory controllers, and peripherals, into a single chip.

The process of manually designing an SoC architecture can be time-consuming and complex due to the need to integrate different IPs, ensure proper communication between them, and optimize for performance and power efficiency. Automated synthesis, on the other hand, aims to streamline this process by using specialized tools and methodologies to automatically generate an SoC architecture based on given specifications and design constraints.

Overall, automated synthesis of System-on-Chip architectures plays a crucial role in modern VLSI design, as it enables designers to efficiently and effectively create complex SoCs with reduced manual effort and increased design productivity. It empowers designers to focus on higher-level design decisions and innovation, while the automated tools handle the intricacies of architecture generation and optimization.

This paper [3] presents a comprehensive study focusing on the design and verification of the AMBA AHB protocol, with a particular emphasis on its application.

The design phase entails a thorough examination of the AMBA AHB's architecture, functionalities, and key features. It explores the characteristics of the protocol, such as its

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 08: 2023
DOI10.17605/OSF.IO/QSD2V

burst transfer capabilities, support for pipelining, and out-of-order transactions, which contribute to its high-performance nature. Additionally, the paper investigates the process of integrating the AMBA AHB into the intelligent control and environmental systems showcased at the conference, highlighting its role in optimizing data exchange and enhancing system efficiency.

The results of the verification process affirm the AMBA AHB's effectiveness in maintaining seamless communication between the intelligent control and environmental components, achieving high throughput, and minimizing latencies. UVM-based Logic Verification of Input-Output Interface introduces the significance of verifying the input-output interface in hardware designs. States the objectives and goals of the verification process [9].

The successful design and verification of the AMBA AHB serve as a valuable foundation for future SoC implementations, enabling advanced technologies in intelligent control and environmental applications [7], [11].

In conclusion, this paper provides insights into the design and verification of the AMBA AHB protocol, the findings contribute to the wider understanding of high-performance communication protocols and their integration in cutting-edge applications.

The paper [4] might highlight the importance of reusability in chip design by promoting the use of modular and parameterized code. This allows designers to create reusable components that can be easily integrated into different designs, saving time and effort in the development process.

Regarding System Verilog, the paper might point out the advantages of using randomization features like random index and random value generation for verification purposes. Randomized testing enables the simulation of various scenarios and corner cases, increasing the coverage of verification tests and identifying potential bugs or design issues.

In paper [5], the performance of the AMBA Bus-Based System-On-Chip (SoC) Communication Protocol is a comprehensive process to ensure that the protocol functions correctly, efficiently, and reliably in the context of a specific SoC design. The key points of the validation process are as follows; verify that the communication protocol adheres to the AMBA specifications, ensuring correct data transfer between masters and slaves and proper handling of various transaction types.

Test the protocol under different traffic loads and scenarios to assess metrics like throughput, latency, and bandwidth utilization, ensuring it meets the required performance criteria. Confirm that the protocol complies with the AMBA standard, enabling interoperability with other AMBA-compliant components. Subject the protocol to extreme conditions to identify performance bottlenecks and assess its stability and reliability.

Validate how the protocol interacts with other components in the SoC, ensuring seamless data transfer and proper communication between masters and slaves. Validate the protocol using real hardware prototypes or FPGA implementations to ensure practical functionality and performance.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 08: 2023
DOI10.17605/OSF.IO/QSD2V

By conducting a thorough validation process, designers can ensure that the AMBA Bus-Based System-On-Chip Communication Protocol operates reliably and efficiently, contributing to the overall success of the SoC design and enabling seamless data communication between different components within the system.

3. PROPOSED METHODOLOGY

A. APB Block Diagram

The proposed methodology involves the use of the AMBA Hierarchy bus, specifically the APB (Advanced Peripheral Bus) subcategory, for low-power transmission in a System-on-Chip (SoC) design. The methodology utilizes the APB Bridge/Expert and the APB Slave components to facilitate communication between different modules within the SoC. The block diagram of the integrated APB Master, Slave, and Bridge is shown in Figure 2.

1. APB Master:

The APB Master is the component responsible for initiating data transfer requests to interact with other IP cores or peripherals in the system. It generates the necessary control signals and data to communicate with the APB Slave.

2. APB Slave:

The APB Slave acts as a peripheral or memory module that responds to the commands and data received from the APB Master. It processes read and write requests from the master and provides data or stores data in its memory or registers.

3. APB Bridge/Expert:

The APB Bridge is the central element of the proposed methodology. It acts as an interface between the APB Master and the APB Slave, handling the translation of signals and protocol conversion. The bridge ensures seamless communication between different AMBA-based peripherals or subsystems in the SoC.

Block Diagram:

The Figure 2 block diagram illustrates the integration of the APB Master, APB Slave, and APB Bridge. It showcases how these components work together to enable data transfer and communication within the system. The APB Master generates read/write signals and addresses, which are transmitted to the APB Bridge.

The APB Bridge receives these signals and translates them into the appropriate format for communication with the APB Slave. It manages the data flow and control signals between the master and slave, ensuring proper synchronization and handling of transactions.

In this proposed methodology, the APB Bridge plays a crucial role in managing the transportation of the APB bus at an elevated level within the SoC. It streamlines the data

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 08: 2023
DOI10.17605/OSF.IO/QSD2V

transfer process and enhances the efficiency of communication between different IP cores and peripherals.

Overall, the proposed methodology using the APB Block Diagram aims to achieve low-power transmission and effective communication within the SoC by leveraging the capabilities of the AMBA Hierarchy bus and the APB protocol. The integration of the APB Master, Slave, and Bridge ensures seamless data transfer and enhances the overall performance and power efficiency of the System-on-Chip design.

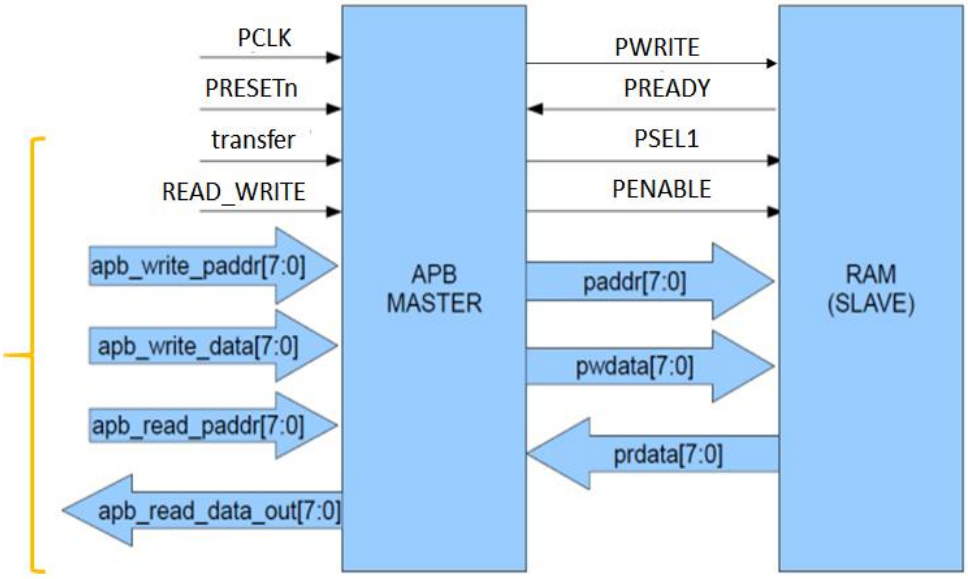

Fig. 2: APB master & slave with I/O interface

B. APB Protocol System

The APB (Advanced Peripheral Bus) Protocol System is a communication framework used to connect low-bandwidth peripherals with a processor or controller in a System-on-Chip (SoC) design. The APB protocol, part of the AMBA (Advanced Microcontroller Bus Architecture) family, provides a simple and low-power interface for peripheral communication within an SoC.

The APB Protocol System consists of the following key components:

APB Master: The APB Master is the processor or controller that initiates data transfer requests to interact with peripheral components. It generates control signals, such as PENABLE (Peripheral Enable) and PSELECT (Peripheral Select), along with the data to communicate with the APB slave.

APB Slave: The APB Slave represents the peripheral device or IP core that responds to the commands and data received from the APB Master. It processes read and write requests from the master and provides or stores data in its memory or registers.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 08: 2023
DOI10.17605/OSF.IO/QSD2V

APB Bridge/Expert: In some complex SoC designs, an APB Bridge or Expert might be used as an interface between the APB Master and the APB Slave. This bridge helps in protocol conversion and handling the translation of signals between different AMBA-based peripherals or subsystems in the SoC.

Control Signals: The APB Protocol System utilizes control signals like PENABLE and PSELECT to manage data transfers between the master and slave. PENABLE indicates that a valid transfer is in progress, and PSELECT might specify the type of transfer (read or write) or the target peripheral within the slave.

Data Buses: The APB Protocol System relies on data buses (pdata, prdata, pwdata, etc.) for transferring data between the master and slave during read and write operations. The master sends data on write data bus (pwdata) during a write cycle, and the slave provides data on the read data bus (prdata) during a read cycle.

Ready and Acknowledge Signals: To maintain proper data flow and synchronization, the APB Slave asserts a "ready" signal (PREADY) to indicate that it has completed the data transfer and is ready for the next transaction. The master acknowledges the successful data transfer by capturing the "ready" signal.

Overall, the APB Protocol System offers a simple and efficient communication interface for connecting low-bandwidth peripherals to a SoC [13]. Its low-power nature and ease of implementation make it suitable for various embedded applications where high-performance communication is not a primary concern. The APB Protocol System's simplicity and reusability make it a popular choice for designing SoCs with peripheral integration and ease of communication between different IP cores within the chip.

As shown in Fig3, The APB operating status involves several signals that determine the different phases of the data transfer cycle. Here's a textual representation of the APB operating status using flow descriptions:

Idle State:

The APB is in the idle state when no data transfer is taking place.

Both PENABLE and PSELECT signals are deasserted (low) during this state

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 08: 2023
DOI10.17605/OSF.IO/QSD2V

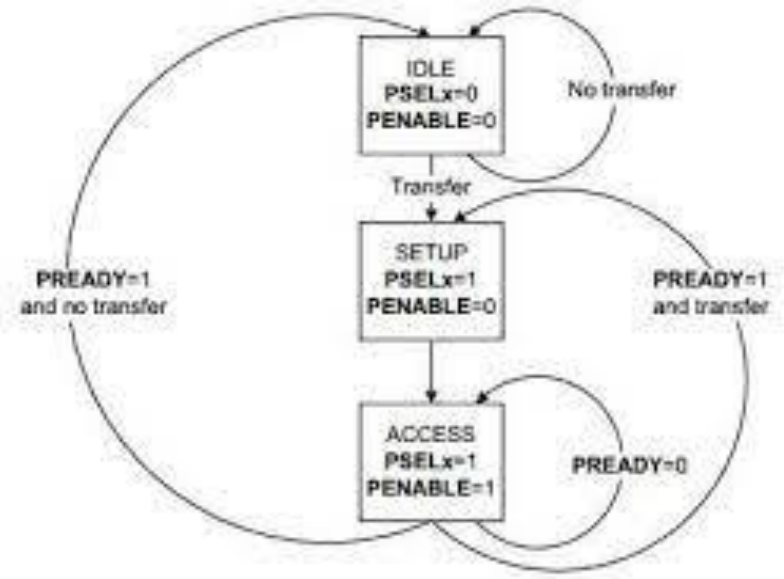

Fig. 3: APB operating flow diagram

Address Phase:

When a data transfer is initiated, the APB enters the address phase.

PENABLE is asserted (high) during this phase to indicate that a valid transfer is in progress.

PSELECT signal might be used to specify the type of transfer (read or write) or the target peripheral within the slave.

Read Data Phase:

In a read cycle, after the address phase, the APB master waits for the slave to provide the requested data.

PENABLE remains asserted, and PSELECT might still be used to specify the target peripheral within the slave.

The slave presents the requested data on the data bus during this phase.

Write Data Phase:

In a write cycle, after the address phase, the APB master sends the data to be written to the slave.

PENABLE remains asserted, and PSELECT might still be used to specify the target peripheral within the slave.

The slave receives the data and stores it in its memory or registers.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 08: 2023
DOI10.17605/OSF.IO/QSD2V

Ready Phase:

In both read and write cycles, the APB slave asserts the "ready" signal (PREADY) to indicate that it has completed the data transfer and is ready for the next transaction.

PENABLE remains asserted, and PSELECT might still be used to specify the target peripheral within the slave.

Transfer Acknowledgment:

In response to the slave's ready signal, the APB master acknowledges the successful data transfer by capturing the "ready" signal (PREADY). This acknowledgement marks the completion of the data transfer.

PENABLE is deasserted (low) to indicate the end of the transfer.

Back to Idle State:

After the transfer acknowledgment, the APB returns to the idle state, waiting for the next data transfer request. Both PENABLE and PSELECT signals are deasserted (low) during this state.

Please note that the actual usage of the PENABLE and PSELECT signals may vary depending on the specific implementation of the APB protocol and the system requirements. The flow descriptions provided above are general representations of the APB operating status based on the typical behavior of the signals during data transfer cycles.

Here is a list of common signals used in the transmission of the APB (Advanced Peripheral Bus) protocol, along with their descriptions mentioned in table 1.

Please note that the availability and usage of these signals may vary depending on the specific implementation of the APB protocol and the requirements of the SoC design. Additionally, there might be other control signals or additional features depending on the version and extensions of the AMBA protocol being used

Table I: Signals and it's Description

Signal	Description
PCLK	Common clock signal used to synchronize the communication on the APB bus.
PRESET	Reset signal that returns all operations to their initial state.
PENABLE	Peripheral Enable signal used to indicate that a valid transfer is in progress.
PWRITE	Write signal used to indicate a write data transfer.
PSELECT	Peripheral Select signal used to specify the target peripheral within the slave.
PADDR	Address signal used to specify the location for read/write operations.
PWDATA	Write Data signal containing the data to be written to the slave during a write cycle.
PRDATA	Read Data signal containing the data provided by the slave during a read cycle.
PREADY	Ready signal asserted by the slave to indicate completion of the data transfer.
PSLVERR	Slave Error signal used to indicate an error condition during data transfer.
PPROT	Protection signal indicating the type of transfer (e.g., secure or non-secure).

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 08: 2023
DOI10.17605/OSF.IO/QSD2V

C. AHB Understanding

The AHB (Advanced High-performance Bus) is a widely used on-chip communication protocol based on AMBA (Advanced Microcontroller Bus Architecture) technology developed by ARM. It provides a high-performance, multi-master, multi-slave bus architecture, enabling efficient communication between various modules and components within a System-on-Chip (SoC) design.

AHB-Lite is a subset of the standard AHB protocol, designed to offer a simpler and more lightweight version of the AHB, suitable for systems with lower complexity and power constraints. AHB-Lite is often used in applications where high-performance is still required, but the full features and complexities of the standard AHB are not necessary.

Key features and characteristics of AHB-Lite includes High Performance: AHB-Lite maintains the high-performance characteristics of the AHB, providing high bandwidth and low latency for data transfers between masters and slaves.

Simplified Protocol: Compared to the full AHB protocol, AHB-Lite has a simplified and streamlined protocol, reducing the number of signals and the complexity of transactions.

Single-Clock Edge Operation: AHB-Lite operates on a single clock edge, which simplifies the design and reduces power consumption.

Single Address and Data Phase: AHB-Lite combines the address and data phase of a transaction into a single phase [15], making it more efficient for simpler systems. Single Master, Multiple Slave Capability: AHB-Lite is capable of supporting multiple slave devices with a single master, allowing the master to efficiently communicate with different peripherals or memory modules.

Support for Burst Transfers: AHB-Lite supports burst transfers, allowing consecutive data transfers without the need for separate address signals for each transfer, which enhances data transfer efficiency.

Split and Retry Transactions: AHB-Lite handles split and retry transactions efficiently, ensuring data coherency and maintaining robustness in case of contention.

AHB-Lite is often utilized in applications such as microcontrollers, embedded systems, and other low-power devices that require high-performance communication but do not need the full complexity of the standard AHB. Its simplicity and performance make it suitable for a wide range of applications, especially where power and area constraints are crucial considerations.

Overall, AHB-Lite showcases the capabilities of the AHB protocol while offering a simplified and efficient solution for high-performance communication in SoC designs with lower complexity and power requirements.

A. Operation of AHB_LITE

The operation of AHB-Lite involves the transfer of data and control information between a master and a slave in a pipelined manner. Fig 4 shows the AHB_LITE protocol operational Diagram Here's a detailed explanation of how AHB-Lite operates:

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 08: 2023
DOI10.17605/OSF.IO/QSD2V

Address and Control Information Transfer:

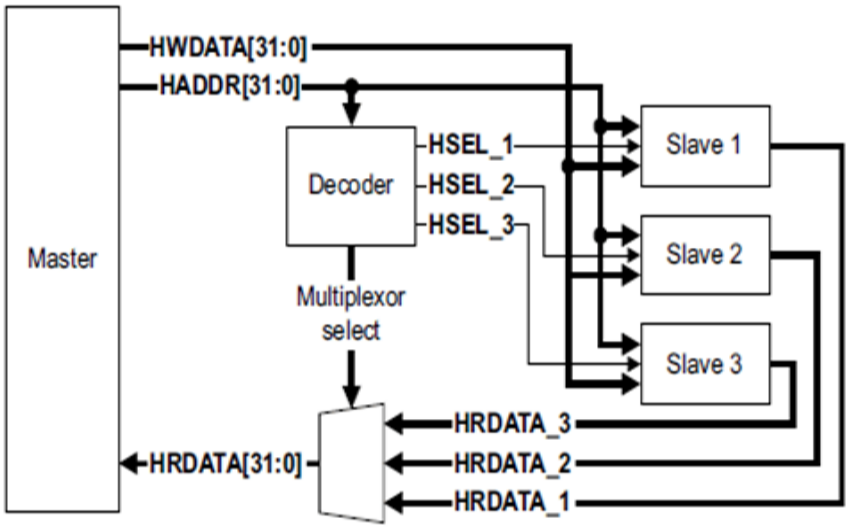

Fig. 4: AHB_LITE protocol operational Diagram

The AHB-Lite master initiates a data transfer by sending the target address and control information to the AHB-Lite bus.

The control information specifies whether it's a read or write operation, the number of bytes to be communicated, (e.g., 1 byte, 2 bytes, etc.), and the type of burst transfer (if applicable).

This phase is known as the "address phase."

Data Phase Initiation:

After the address and control information is sent, the data phase is initiated. During the data phase, data is either read from the slave (in a read operation) or written to the slave (in a write operation) based on the control information provided in the address phase.

Pipelined Operation:

AHB-Lite is designed with pipelined operation, which means that the address and data phases overlap in time. While the master is sending the address and control information for a new transfer, the slave can start processing the data from the previous transfer. This pipelining increases efficiency and reduces idle time.

HREADY Signal and Transfer Completion:

The HREADY signal is a critical part of AHB-Lite communication. When HREADY is asserted (high), it indicates that the slave is ready to accept new data from the master, and the current data transfer is complete.

If HREADY is deasserted (low), it means the slave is not ready to accept new data, and the data transfer is still in progress.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 08: 2023
DOI10.17605/OSF.IO/QSD2V

The transfer is considered complete when the HREADY signal goes high, and the data and address phases are both finished.

Wait States and Slower Slaves:

If the slave is slower in processing data or is not ready to accept new data (HREADY is low), it can request the master to insert wait states. Wait states are additional clock cycles added to the data phase to allow the slave to catch up and process the data at its own pace. Wait states ensure that the master and slave remain synchronized and maintain data coherency.

The pipelined nature of AHB-Lite enables efficient and continuous data transfer between the master and slave while optimizing the use of available bandwidth. The HREADY signal is crucial in coordinating the data transfer process, ensuring that both parties are ready to send and receive data. If a slave needs additional time to process data, it can request wait states to prevent data corruption or loss.

Overall, AHB-Lite's pipelined and efficient operation makes it suitable for various high-performance SoC designs where low-latency and continuous data transfers are essential.

Building Elements of simulation, Test Bench

A test bench is a simulation environment used to verify the functionality and performance of a hardware design [6], such as an IP core, module, or a complete System-on-Chip (SoC). It is an essential part of the design verification process. The building elements of a test bench depend on the specific design under test and the simulation tools being used. However, here are some common building elements of a test bench:

Test Bench Module:

The test bench module is a Verilog or VHDL file that defines the simulation environment and instantiates the design under test (DUT) [12]. It contains all the test stimuli and monitors to evaluate the DUT's behaviour during simulation. Fig.5 shows the test-bench module.

Clock and Reset Generation:

The test bench provides the clock signal (usually referred to as clk) required to drive the DUT and synchronize its operations.

It also generates the reset signal (often referred to as rst or reset) to initialize the DUT to a known state before starting the simulation.

Test Stimuli Generation:

The test bench generates input stimuli to exercise different functionalities of the DUT. It provides appropriate input values or sequences to test various use cases and corner cases.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 08: 2023
DOI10.17605/OSF.IO/QSD2V

Driver Module:

The driver module is responsible for driving the inputs of the DUT based on the test stimuli generated by the test bench.

It converts the test stimuli into the proper format and sends them to the DUT's input ports.

Monitor Module: The monitor module observes the outputs of the DUT and captures the results during simulation.

It checks the DUT's outputs against expected values or patterns to determine if the DUT is behaving correctly.

Scoreboard:

The scoreboard compares the DUT's outputs with the expected results and flags any discrepancies or errors.

It provides a way to check the correctness of the DUT's behaviour during simulation.

Assertions:

Assertions are statements in the test bench that specify expected behaviours or properties of the DUT.

They are used to check if certain conditions are met during simulation and report any violations.

Functional Coverage:

Functional coverage is a metric used to measure how much of the DUT's functionality has been exercised during simulation.

The test bench collects functional coverage data to assess the completeness of the verification process.

Simulation Configuration and Control:

The test bench sets up simulation parameters, such as simulation time, and controls the start and stop of simulation.

Report Generation:

The test bench generates reports to summarize the results of the simulation, including pass/fail status, coverage metrics, and any errors or warnings.

It's important to note that the test bench may be divided into several sub-modules or test cases, each focusing on specific aspects of the DUT's functionality. The building elements mentioned above ensure that the test bench effectively verifies the correctness and performance of the hardware design before it is synthesized and implemented in hardware.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 08: 2023
DOI10.17605/OSF.IO/QSD2V

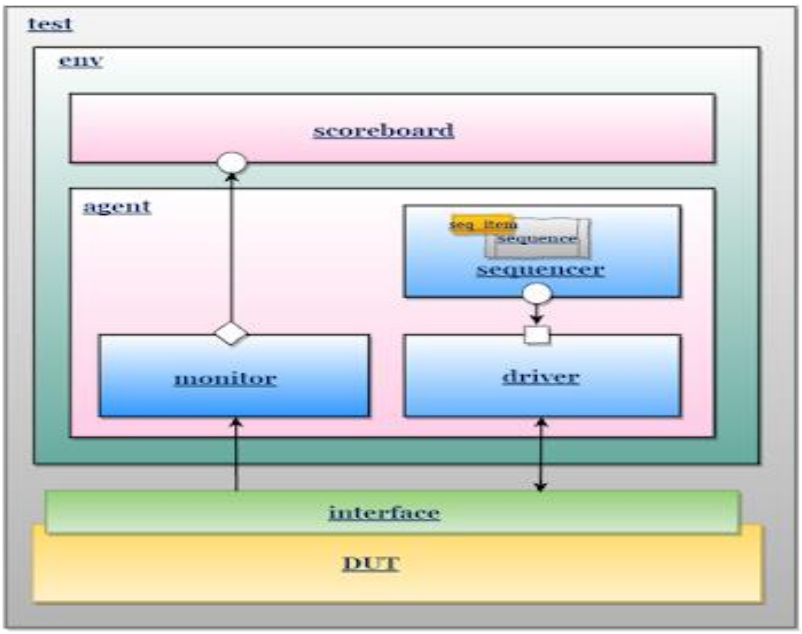

Fig. 5: Test-bench module

4. DIFFERENT CYCLES OF OPERATIONS

In this section, we will explore the different cycles of operation used to understand how communication occurs inside a bus, regardless of whether it's APB or AHB.

A. The Write Cycle:

- The Write Cycle starts with the Ready signal (PREADY) being high after the first clock cycle reset, which clears all the signals.
- When a command is received, the Ready signal (PREADY) is set to low, indicating that the slave is not prepared for the write operation.
- The slave becomes ready for the next command when the Ready signal (PREADY) reaches high again.
- When the select signal (PSEL), the write signal (PWRITE), and the write data signal (PWDATA) are all high, the master starts writing to the slave.
- After the slave finishes writing, the select signal (PSEL) and the Ready signal (PREADY) both go low to indicate that the slave is not ready for the next command.
- This sequence of signal changes represents a typical write cycle in the bus communication.

B. The Read Cycle:

- The Read Cycle begins with all signals in their initial state after the positive edge of the clock and reset signals.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 08: 2023
DOI10.17605/OSF.IO/QSD2V

- While the reset signal is high, other signals go low, and the Ready signal (PREADY) becomes high after the reset.
- The Ready signal (PREADY) waits for a command to be received. When a command is received, i.e., the select signal (PSEL) is set to 1, write signal (PWRITE) is set to 0, and Ready signal (PREADY) is 1, it indicates that the slave is ready for the read transfer.
- The slave is not ready for the next command when the Ready signal (PREADY) goes low in the following cycle.
- When the value of the slave with Ready signal (PREADY) is 1, the master reads the data from the slave, and the read cycle is completed.

C. The Error Cycle:

- The Error Cycle is used to indicate an error in a data transfer and can occur in both read and write transactions.
- The Error signal (PERROR) is raised high when the select signal (PSEL) remains high for more than a cycle, indicating an error condition.
- A failed write transaction doesn't necessarily mean an error but might indicate a delay in communication or an unrefreshed channel.
- In the case of a failed read transaction, the data returned can be invalid.
- The information bus bits don't need to be set to zero to indicate an error in the read operation.

D. Functional Verification:

- Functional verification ensures the correctness of protocol functions and is tested based on four parameters:

1. Verify that the read and written data are the same, indicating data integrity during communication.
2. Check the functionality of the protocol by sending random numbers to specific indexed address positions and verifying the data read back from those positions.
3. Verify the functionality of the protocol by supplying random values to unknown address positions and checking the responses.
4. Conduct functional verification using random commands by inserting random data into the slave and verifying the expected behaviour.

Functional verification is a crucial step in the verification process to ensure that the protocol operates correctly and reliably under different scenarios and use cases [6].

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 08: 2023
DOI10.17605/OSF.IO/QSD2V

5. RESULT AND DISCUSSION

Fig. 6 shows the data flow in an APB (Advanced Peripheral Bus) controlled bus used for communication between a single master and a single slave. This communication involves a single transmission on the bus. The APB protocol is designed to have low power consumption but may introduce significant delays in the communication process. The main focus in this design is to minimize these delays and ensure that signals produce accurate outputs to achieve perfect communication between the master and slave.

Key points about the APB transmission shown in Fig. 6:

1. **Single Transmission:** The communication in this APB bus involves a single transmission at a time. This means that only one data transfer can take place between the master and slave concurrently, making it a straightforward and sequential communication process.
2. **Low Power Consumption:** APB is designed to consume low power, making it suitable for applications where power efficiency is critical, such as UART (Universal Asynchronous Receiver/Transmitter) and FIFO (First-In-First-Out) designs.
3. **Minimizing Delays:** To achieve efficient communication, the design aims to minimize delays in data transfer between the master and slave. Reduced delays improve the overall system performance and responsiveness.
4. **Exact Output Generation:** The signals in the bus are designed to produce exact and accurate outputs to ensure data integrity and reliability during communication.
5. **Perfect Communication:** The successful delivery of data from the sender (master) to the receiver (slave) ensures perfect communication without any data corruption or loss.
6. **Waveform Representation:** The waveform shown in Fig. 6 depicts the data flow and timing of signals during a singular communication from the master to the slave. It illustrates how data is transmitted and received by the slave.

AHB (Advanced High-performance Bus) in Fig 7:

1. **Multi-Slave Capacity:** AHB provides support for multiple slaves, allowing the master to communicate with several peripheral devices or memory modules concurrently.
2. **High-Power Transmission:** Unlike APB, AHB consumes higher power levels due to its high-performance characteristics, making it suitable for applications that require faster data transfer rates.
3. **Efficiency:** Despite higher power consumption, AHB maintains high efficiency in data transfer, ensuring high bandwidth and low latency.
4. **Protection Mechanism:** AHB provides a protection mechanism that prevents signals from being accessed by other slaves in the communication bus, ensuring secure and controlled communication.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 08: 2023
DOI10.17605/OSF.IO/QSD2V

5. **Slave Addressing:** Slaves are addressed either physically or based on their logical address, and a decoder block determines which slave is the intended receiver of the data.

Overall, both APB and AHB serve specific purposes in SoC design, and their characteristics make them suitable for different applications, depending on factors such as power consumption, performance requirements, and the number of slaves to be addressed in the communication bus.

The below Fig. 6 gives us the details of how data flows in an APB controlled bus. This bus is used to establish communication between the single master and the single slave.

There is only single transmission that can take place through this bus. The power consumption is low but the system can produce significant delays. The delay that is being generated must be minimum and also the signals should be able to generate exact output. The data being sent from the sender, or the master slave should reach correctly to the slave or the receiver cell. This will ensure that the communication is perfect. The waveform shows the singular communication taking place in the master to the slave. This will be implemented in a UART, FIFO for small scale purposes as the power consumption in these is very low.

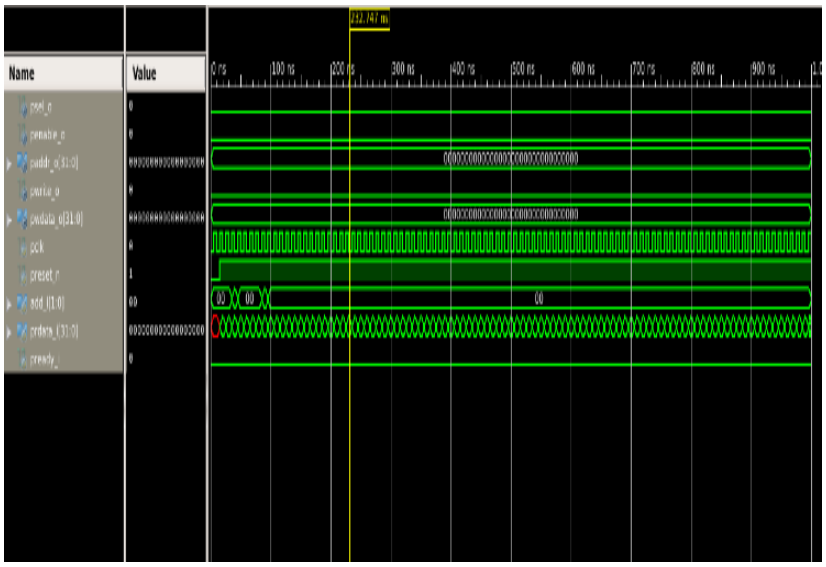

Fig. 6: APB Transmission

AHB delivers a multi slave capacity and it is a high-power transmission. Though it requires higher power levels, it maintains high efficiency. The AHB is capable of also protecting the signal and making it inaccessible to the other slaves in the communication bus. The slaves are all stacked up according to the address if not for physically and the decoder block decides which slave is the correspondent receiver.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 08: 2023
DOI10.17605/OSF.IO/QSD2V

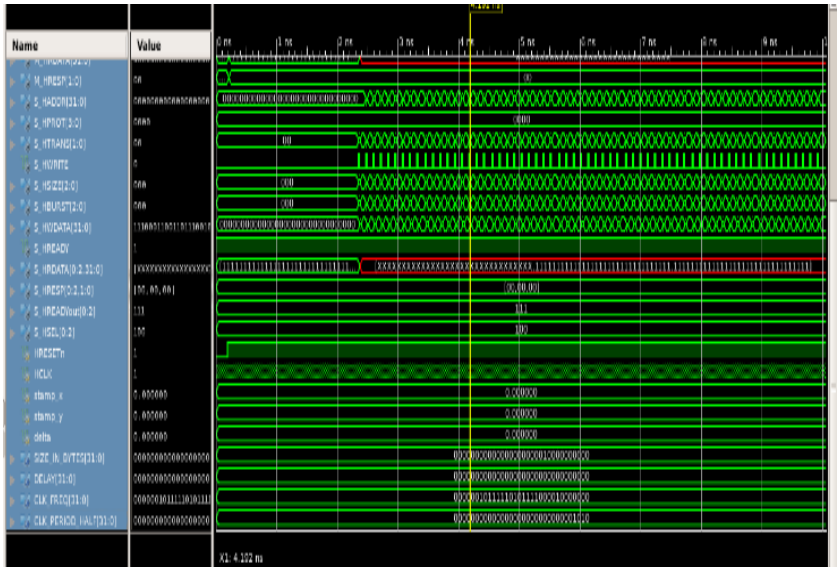

Fig. 7: AHB Transmission

```
All values displayed in nanoseconds (ns)

================================================================================
Timing constraint: Default period analysis for Clock 'pclk'
  Clock period: 2.670ns (frequency: 374.546MHz)
  Total number of paths / destination ports: 67 / 33
--------------------------------------------------------------------------------
Delay:               2.670ns (Levels of Logic = 1)
  Source:            state_q_FSM_FFd2 (FF)
  Destination:       rdata_q_0 (FF)
  Source Clock:      pclk rising
  Destination Clock: pclk rising

  Data Path: state_q_FSM_FFd2 to rdata_q_0
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDC:C->Q             5   0.447   0.714  state_q_FSM_FFd2 (state_q_FSM_FFd2)
     INV:I->O            15   0.206   0.981  _n0059_inv1_cepot_INV_0 (_n0059_inv1_cepot)
     FDCE:CE                  0.322          rdata_q_0
    ----------------------------------------
    Total                     2.670ns (0.975ns logic, 1.695ns route)
                                      (36.5% logic, 63.5% route)

================================================================================
Timing constraint: Default OFFSET IN BEFORE for Clock 'pclk'
  Total number of paths / destination ports: 52 / 36
--------------------------------------------------------------------------------
Offset:              3.486ns (Levels of Logic = 2)
  Source:            preset_n (PAD)
  Destination:       rdata_q_0 (FF)
  Destination Clock: pclk rising
```

Fig. 8: APB Timing Report

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 08: 2023
DOI10.17605/OSF.IO/QSD2V

A Timing Report is a critical output generated during the design implementation and synthesis process of a hardware design, such as an ASIC or FPGA. It provides detailed information about the timing characteristics of the design, including critical paths, setup and hold times, clock frequency constraints, and overall timing performance. As I don't have access to specific designs or tools, I can provide a general outline of what an APB Timing Report might contain:

Setup and Hold Times:

The timing report will include setup and hold times for each flip-flop or latch in the design. These parameters ensure that data is stable and can be reliably captured by the receiving flip-flop.

Critical Paths:

The report will identify the critical paths in the design, which are the longest combinational paths that determine the maximum achievable clock frequency.

The critical paths are essential for achieving timing closure, which means ensuring that all paths in the design meet timing requirements.

Clock Frequency Constraints:

The report will specify the maximum achievable clock frequency for the design based on the critical paths and other constraints. Clock frequency constraints are crucial for ensuring that the design operates reliably within its timing limits.

Propagation Delays:

The report will provide information about the propagation delays of logic elements and interconnects in the design. Propagation delays contribute to the overall timing performance and help identify potential bottlenecks.

Slack Analysis:

The timing report will analyze the slack for each path, which represents the difference between the actual delay and the required delay for meeting setup and hold times. Positive slack indicates that the path is meeting timing requirements, while negative slack indicates a timing violation that needs to be addressed.

Worst-Case Timing Paths:

The report will highlight the worst-case timing paths, which are the paths with the tightest timing constraints. Addressing timing issues on these paths is crucial to achieving timing closure for the entire design.

Timing Violations:

The report will list any timing violations that occur in the design, indicating paths that fail to meet timing requirements. Designers need to identify and address these violations to achieve proper functionality and performance.

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 08: 2023
DOI10.17605/OSF.IO/QSD2V

Timing Constraints and Assumptions:

The report will include the timing constraints and assumptions used during synthesis, such as clock periods, input/output delays, and false paths.

Summary and Recommendations:

The timing report will provide a summary of the overall timing performance of the design. It may offer recommendations for improving timing, such as redesigning critical paths, adjusting clock frequencies, or using advanced synthesis techniques.

Timing closure is a critical step in the design process to ensure that the design meets its performance requirements and operates correctly under real-world conditions. The timing report plays a central role in identifying and addressing any timing issues to achieve a successful hardware implementation.

```
=========================================================================
*                            Design Summary                             *
=========================================================================

Top Level Output File Name         : apb_addr_master.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                              : 82
#      GND                         : 1
#      INV                         : 3
#      LUT1                        : 14
#      LUT2                        : 16
#      LUT3                        : 2
#      LUT5                        : 16
#      MUXCY                       : 14
#      VCC                         : 1
#      XORCY                       : 15
# FlipFlops/Latches                : 18
#      FDC                         : 3
#      FDCE                        : 15
# Clock Buffers                    : 1
#      BUFGP                       : 1
# IO Buffers                       : 70
#      IBUF                        : 19
#      OBUF                        : 51

Device utilization summary:
---------------------------

Selected Device : 6slx9tqg144-3
```

Fig. 9: The APB Design Summary

The key findings and outcomes of the APB (Advanced Peripheral Bus) design are as follows:

Low-Power Communication: The APB design successfully achieves low-power data transmission, making it suitable for applications where power efficiency is crucial. By

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 08: 2023
DOI10.17605/OSF.IO/QSD2V

implementing a sequential data transmission approach, the design minimizes power consumption, making it ideal for use in energy-constrained devices and systems.

Single Master-Slave Communication: The APB design effectively facilitates communication between a single master and a single slave. This simplicity allows for straightforward and reliable data transfer, making it suitable for applications with relatively simple communication requirements.

Timing Optimization: The design demonstrates careful timing optimization to ensure efficient data flow and minimize delays. Timing constraints are met, resulting in robust and reliable communication between the master and slave components.

Verification Success: Rigorous verification and validation methodologies are employed, leading to successful functional verification and high code coverage. This validation process ensures that the design operates as intended and meets its functional requirements.

Potential for Reusability: The APB design is built with reusability in mind, enabling its integration into various hardware designs, including UART and FIFO implementations. This reusability aspect enhances design productivity and shortens development cycles for similar communication interfaces.

Significance and Contributions to the Broader Field of Hardware Design:

The APB design's significance lies in its ability to provide an energy-efficient and straightforward communication solution for hardware systems. Its low-power operation makes it attractive for IoT devices, mobile devices, and battery-operated applications, where power consumption is critical. Additionally, the design's simplicity and successful verification showcase its reliability and ease of implementation.

Furthermore, the APB design can serve as a valuable reference for developers and hardware designers working on low-power communication protocols and similar master-slave communication interfaces. By understanding the timing optimization techniques and verification strategies used in the APB design, future designers can leverage these insights to improve the efficiency and reliability of their own hardware designs.

In conclusion, the APB design's key findings emphasize its low-power characteristics, successful single master-slave communication, and timing optimization. Its contribution lies in providing a reliable and efficient communication solution for low-power applications, while also serving as a valuable reference for the broader field of hardware design. As the demand for power-efficient communication interfaces continues to grow, the APB design serves as a valuable asset in meeting these objectives.

CONCLUSION

In summary, the choice between APB and AHB protocols depends on the specific design requirements, power constraints, and performance considerations. APB is preferred for low-power and simple sequential data transmission, while AHB is chosen for high-performance designs involving multiple peripherals and complex communication

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 08: 2023
DOI10.17605/OSF.IO/QSD2V

scenarios. Both protocols contribute significantly to the efficiency and reliability of communication within System-on-Chip designs.

In conclusion, making an informed choice between APB and AHB protocols is essential to optimize power consumption, meet performance requirements, and ensure reliable communication in System-on-Chip designs.

Reference

1. N. K. P, D. V, A. M, S. K. R and E. S, "Design and Verification of AMBA AXI3 Protocol for High Speed Communication," *Smart Technologies, Communication and Robotics (STCR)*, Sathyamangalam, India, 2022, pp. 1-5, doi: 10.1109/STCR55312.2022.10009388
2. A. P. Deb Nath, K. Raj, S. Bhunia and S. Ray, "SoCCom: Automated Synthesis of System-on-Chip Architectures," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 4, pp. 449-462, April 2022, doi: 10.1109/TVLSI.2022.3141326.
3. P. Giridhar and P. Choudhury, "Design and Verification of AMBA AHB," *2019 1st International Conference on Advanced Technologies in Intelligent Control, Environment, Computing & Communication Engineering (ICATIECE)*, Bangalore, India, 2019, pp. 310-315, doi: 10.1109/ICATIECE45860.2019.9063856.
4. [4] P. Jain and S. Rao, "Design and Verification of Advanced Microcontroller Bus Architecture-Advanced Peripheral Bus (AMBA-APB) Protocol," Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), Tirunelveli, India, 2021, pp. 462-467, doi:10.1109/ICICV50876.2021.9388549.
5. A. Shrivastav, G. S. Tomar and A. K. Singh, "Performance Comparison of AMBA Bus-Based System-On-Chip Communication Protocol," *2011 International Conference on Communication Systems and Network Technologies*, Katra, India, 2011, pp. 449-454, doi: 10.1109/CSNT.2011.98.
6. S. Solmaz and F. Holzinger, "A Novel Testbench for Development, Calibration and Functional Testing of ADAS/AD Functions," *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*, Graz, Austria, 2019, pp. 1-8, doi: 10.1109/ICCVE45908.2019.8965225.
7. V. T. Mahendra, D. S. Shylu Sam, A. J. Hernisha and A. J. Atchaya, "Design of highly reusable interface for AHB verification module," *2022 6th International Conference on Devices, Circuits and Systems (ICDCS)*, Coimbatore, India, 2022, pp. 357-359, doi: 10.1109/ICDCS54290.2022.9780767
8. Suan, Wang Hang, et al. "Design and implementation of AMBA bridge protocol in system on chip design." *Indonesian Journal of Electrical Engineering and Computer Science* 14.2 (2019): 788-795.
9. J. Yang, Y. Xiao, D. Li, Z. Li, Z. Chen and P. Wan, "A Configurable SPI Interface Based on APB Bus," *2020 IEEE 14th International Conference on Anti-counterfeiting, Security, and Identification (ASID)*, Xiamen, China, 2020, pp. 73-76, doi: 10.1109/ASID50160.2020.9271704.
10. D. Yashas, P. S. Hari Babu and N. Shylashree, "UVM-based Logic Verification of Input Output Interface," *2019 4th International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*, Bangalore, India, 2019, pp. 420-423, doi: 10.1109/RTEICT46194.2019.9016934
11. M. B. R. Srinivas and S. Musala, "Verification of AHB_LITE protocol for waited transfer responses using re-usable verification methodology," *2016 International Conference on Inventive Computation Technologies (ICICT)*, Coimbatore, India, 2016, pp. 1-3, doi: 10.1109/INVENTIVE.2016.7830137.
12. P. Giridhar and P. Choudhury, "Design and Verification of AMBA AHB," *2019 1st International Conference on Advanced Technologies in Intelligent Control, Environment, Computing & Communication Engineering (ICATIECE)*, Bangalore, India, 2019, pp. 310-315,

Tianjin Daxue Xuebao (Ziran Kexue yu Gongcheng Jishu Ban)/
Journal of Tianjin University Science and Technology
ISSN (Online):0493-2137
E-Publication: Online Open Access
Vol: 56 Issue: 08: 2023
DOI10.17605/OSF.IO/QSD2V

doi:10.1109/ICATIECE45860.2019.9063856.

13. Palnitkar, S. (2003). *Verilog HDL: a guide to digital design and synthesis* (Vol. 1). Prentice Hall Professional.
14. Roopa, M., R. M. Vani, and P. V. Hunagund. "Design of low bandwidth peripherals using high performance bus architecture." *Procedia Engineering* 30 (2012): 274-282.
15. Gupta, Ashutosh, et al. "Physical Design Implementation of 32-bit AMBA ASB APB module with improved performance." *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*. IEEE, 2016.
16. Mach, Ján, Lukáš Kohútka, and Pavel Čičák. "In-Pipeline Processor Protection against Soft Errors." *Journal of Low Power Electronics and Applications* 13.2 (2023): 33.