INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT (IJSREM)
VOLUME: 08 ISSUE: 12 | DEC - 2024 SJIF RATING: 8.448 ISSN: 2582-3930

# Realization of Advanced Peripheral Bus(APB) Protocol using Verilog

Dr. B Ravi Kumar
Associate Professor
Institute of Aeronautical Engineering
Hyderabad, Telangana, India
b.ravikumar@iare.ac.in

Katte Srisha(21951A0419)
Electronics and Communication Engineering
Institute of Aeronautical Engineering
Hyderabad, Telangana, India
srishakatte@gmail.com

Boppana Akhila(21951A0410)
Electronics and Communication Engineering
Institute of Aeronautical Engineering
Hyderabad, Telangana, India
akhilaboppana2515@gmail.com

***Abstract*—This project details the Verilog implementation of the Advanced Peripheral Bus (APB) protocol, focusing on a configuration with one master and multiple slaves. As part of the Advanced Microcontroller Bus Architecture (AMBA), the APB protocol plays a crucial role in facilitating communication between a master, such as a microcontroller, and various peripheral devices (slaves) within System-on-Chip (SoC) designs. The design, created using Verilog—a prominent hardware description language—follows industry standards with an emphasis on modularity and scalability. Key aspects of the APB protocol are incorporated, such as its single-master, multi-slave setup, data transfer processes, addressing schemes, and control signals. The project prioritizes design efficiency and performance optimization while maintaining full adherence to APB specifications. Special attention is given to handling bus contention, managing timing constraints, and resolving arbitration to ensure smooth communication in a variety of SoC applications.**

***Keywords—Advanced Peripheral Bus (APB), Verilog, System-On-Chip (SoC), AMBA, Single-master, Multiple-slaves, Data transfer, Addressing schemes, Control signals, Arbitration conflicts.***

## I. INTRODUCTION

In modern System-on-Chip (SoC) designs, efficient communication between the central processing unit and peripheral devices is essential for overall system performance. [1]The Advanced Peripheral Bus (APB) protocol serves this purpose by providing a simplified, low-power interface for peripheral communication. This paper explores the realization of the APB protocol using Verilog, a hardware description language widely utilized in digital circuit design. The focus is on implementing a single-master, multiple-slave configuration to ensure smooth data transfer and control signaling. This design emphasizes both efficiency and scalability, making it suitable for various SoC environments while adhering to established industry standards.

[2]The Advanced Microcontroller Bus Architecture (AMBA) is a widely utilized standard for communication in System-on-Chip (SoC) designs,developed by ARM to harmonize the interaction between different components on a chip. AMBA consists of various bus protocols, each tailored for different performance requirements. High-performance protocols like the Advanced High-Performance Bus (AHB) and the Advanced eXtensible Interface (AXI) are designed for handling large data transfers. In contrast, the Advanced Peripheral Bus (APB) is optimized for linking low-bandwidth peripherals that don't demand high-speed communication. Its design prioritizes simplicity and power efficiency, making it well-suited for peripherals such as UARTs, timers, and GPIOs in SoCs.

[6]The APB protocol offers a simple yet effective solution for communication between a master device and multiple peripheral slaves. Unlike more complex buses, APB does not support advanced features like burst transfers or pipelining, which keeps the design lean and power-efficient. It operates using basic control signals, such as address and data lines, to manage read and write operations between the master and slave devices. [5]In this paper, the APB protocol is implemented using Verilog, focusing on handling key challenges such as bus contention and timing management. The result is a robust and scalable design that adheres to the APB specifications, ensuring reliable performance in various SoC applications.

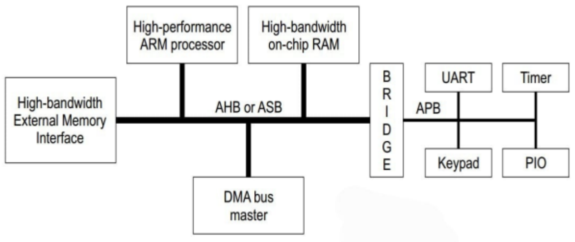

High-performance ARM processor
High-bandwidth on-chip RAM
High-bandwidth External Memory Interface
AHB or ASB
DMA bus master
B R I D G E
APB
UART
Timer
Keypad
PIO

Fig 1: AMBA Architecture

## II. EXISTING SYSTEM

In existing systems utilizing the Advanced Peripheral Bus (APB) protocol, the single master-single slave configuration is a common setup within System-on-Chip (SoC) designs. [9]This architecture features a single master device that controls communication by issuing commands and addressing a single slave device, which responds to these commands. This configuration is particularly effective for low-power peripherals such as timers, GPIOs, and simple communication modules, providing a straightforward and efficient method for data transfer with minimal power consumption.

[7]The APB protocol in this setup handles basic transactions one at a time, with each data transfer cycle involving a single read or write operation. The master sends control signals and addresses to the slave, which then performs the necessary action. This simplicity ensures that the system is easy to design and operate, but it also comes with limitations. The non-pipelined nature of the communication means that the system cannot handle multiple concurrent transactions, which can be a significant drawback for applications requiring higher data throughput or more complex interactions.

Additionally, the single master-single slave configuration lacks scalability. While it is suitable for applications with minimal peripheral devices and low data requirements, it becomes less effective as the number of peripherals increases. For more advanced SoC designs, configurations with a single master and multiple slaves or multiple masters are often preferred to address higher communication demands and improve overall system flexibility and performance. The limitations of the single master-single slave setup can be mitigated by adopting these more complex configurations, which offer enhanced communication capabilities and better support for diverse applications[11].

## III. PROPOSED SYSTEM

The proposed system utilizes a single master and multiple slaves configuration within the Advanced Peripheral Bus (APB) protocol, offering an enhanced solution for managing communication in System-on-Chip (SoC) designs. In this setup, one master device coordinates the communication and controls multiple peripheral slave devices. Each slave device is connected to the master via a common bus, allowing the master to manage interactions with all connected peripherals efficiently. This architecture provides a scalable and flexible approach, accommodating a range of peripheral devices within a single communication framework.

One of the key advantages of this configuration is its ability to handle multiple peripheral devices simultaneously. The single master can issue commands to and receive responses from various slaves, which helps in managing complex systems with numerous peripherals. This setup allows for more efficient data handling and reduces the need for additional control logic, as the master can streamline communication across all connected slaves. Additionally, it simplifies the design by centralizing control, making it easier to manage and modify the system as needed.

Another benefit of the single master-multiple slaves configuration is improved system scalability. As the number of peripherals increases, the system can accommodate additional slaves without significant changes to the master's control logic. This scalability is crucial for modern SoC designs that require integration with multiple peripheral devices, such as sensors, communication modules, and memory components. By using this approach, designers can achieve a more organized and manageable system, enhancing overall performance and flexibility while maintaining efficient communication across all peripherals.

### A. Working

In the AMBA Advanced Peripheral Bus (APB) protocol, the master device in a single master and two slaves configuration operates through a series of well-defined states, which can be visualized using a state diagram. Initially, the master is in the Initialization state, where it prepares for communication by setting up the APB bus. During this phase, the master ensures that all necessary control signals are in place and the address, data, and control lines are ready for operation. This state ensures that the system is configured correctly before initiating any transactions.

Once initialization is complete, the master device transitions to the Addressing state in a communication protocol. In this state, the master places the address of the target slave device onto the bus. The bus address serves to identify the specific slave device that is to participate in the operation. Along with the address, the master sends control signals to indicate the type of operation—whether it will be a read or a write operation. These control signals are essential as they dictate how the slave device should respond.

The master remains in the Addressing state until the addressed slave acknowledges the address and control signals. This acknowledgment ensures that the slave is prepared to either send or receive data. If no acknowledgment is received, the master may attempt to resend the address or handle the error accordingly, ensuring communication integrity before proceeding.

Following the addressing phase, the master moves to the Data Transfer state. Here, depending on the operation type, the master either reads data from the selected slave or writes data to it. For read operations, the addressed slave places the requested data on the bus, which the master then reads. For write operations, the master sends data onto the bus, and the slave captures and stores this data. The master oversees this process to ensure that data transfer is executed correctly and efficiently.

After completing the data transfer, the master transitions to the Idle state. In this state, the master is not engaged in any communication but is instead waiting for the next transaction to initiate. This idle phase allows the system to pause between transactions and ensures that the master can respond promptly when a new command or operation is required. The state diagram thus outlines the master's progression from initialization through addressing and data transfer, concluding with an idle state, ensuring organized and controlled communication with the two slave devices.

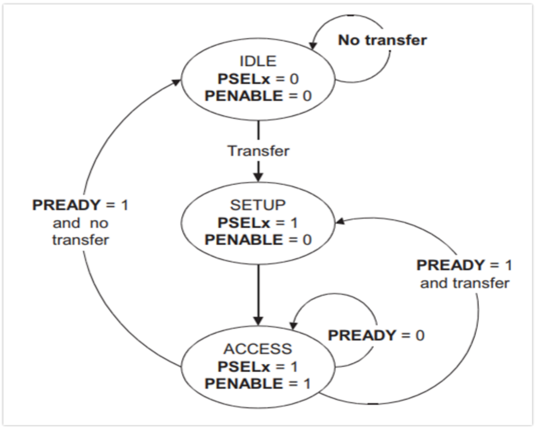


Fig 2: Operating System

Interfacing of APB master and slave:

The interfacing of an Advanced Peripheral Bus (APB) master with slave devices is central to the protocol's operation within a System-on-Chip (SoC) design. The master device controls the communication by generating the necessary control signals and addressing information. It places the address of the target slave device on the address bus, along with control signals that specify the type of operation, such as read or write. The slave devices, connected to the same APB bus, receive these signals and respond accordingly. This setup ensures that the master can efficiently manage interactions with multiple slaves by directing specific commands and data to the intended device. The interface between the master and slave involves clear signal pathways and synchronization mechanisms to ensure accurate and timely data transfers, maintaining system integrity and performance.

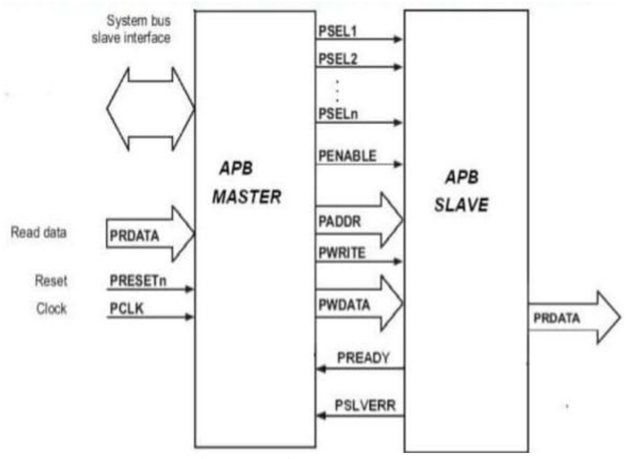


Fig 3: Interfacing of APB master and slave

APB Slave:

The functioning of an APB slave device involves responding to commands issued by the master device based on the addressing and control signals it receives. When the master places an address and control signals on the bus, the addressed slave decodes this information to determine the nature of the operation—whether it is a read from or write to its internal registers or memory. For a read operation, the slave retrieves the requested data from its storage and places it on the data bus for the master to read. Conversely, for a write operation, the slave captures the data sent by the master and stores it in the designated location. Throughout this process, the slave must handle signals accurately to ensure proper data transfer and maintain synchronization with the master's commands, ensuring reliable communication and data integrity within the SoC.

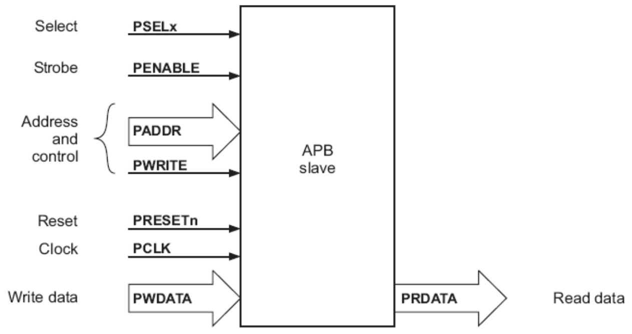


Fig 4: APB Slave Architecture

Read and Write operations:

In the AMBA APB (Advanced Peripheral Bus) protocol, communication is structured in a way that a single master controls the data flow between one or more slave devices. The protocol is often used in systems that require low power and low bandwidth, making it particularly suitable for accessing registers in peripheral devices such as GPIOs (General-Purpose Input/Output), timers, or UARTs (Universal Asynchronous Receiver-Transmitters). In a system with one master and two slaves, the master is responsible for initiating all read and write transfers, and the slaves respond according to the commands issued by the master.

In a write operation, the master first sends the address of the target slave via the PADDR signal. The address decoder then checks the address and determines which of the two slaves should respond to this transaction. Once the appropriate slave is selected, the master sets the PWRITE signal to 1, indicating that a write operation is about to take place. Along with this, the master places the data to be written on the PWDATA signal line, which is where the slave will retrieve the information from.

At this point, the master asserts the PENABLE signal to indicate that the slave should be ready to process the transfer. The slave then captures the data from the PWDATA line and stores it in the appropriate register or memory location. Once the data is successfully written, the transfer is considered complete, and the PENABLE signal is de-asserted by the master, indicating that the transaction is over.

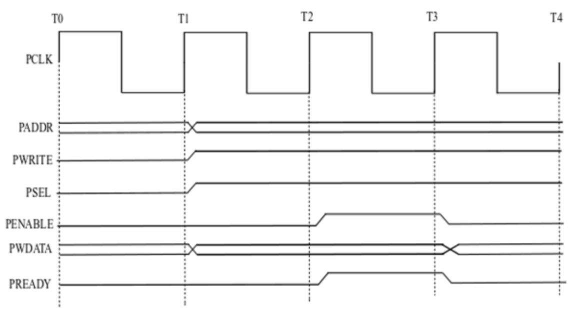


Fig 5: APB Write Cycle

The read operation follows a similar initial process. The master begins by sending the address of the slave it wants to read data from through the PADDR signal. The address decoder, once again, determines which slave will respond. Instead of setting the PWRITE signal to 1 (for write), the master now sets PWRITE to 0, which signals to the selected slave that the operation is a read.

With the PENABLE signal asserted, the selected slave places the requested data on the PRDATA line. This is where the master retrieves the information. Once the master captures the data from PRDATA, it de-asserts the PENABLE signal, marking the end of the read transaction. The master can then process the data it has received or continue with another operation.

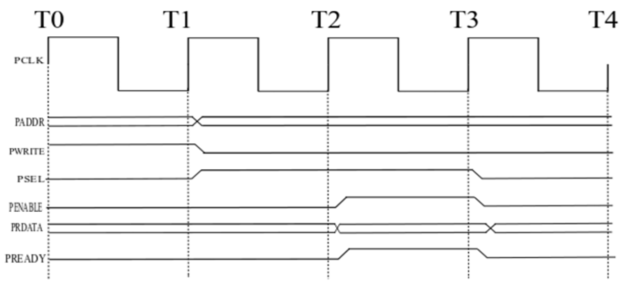


Fig 6: APB Read Cycle

Role of Address Decoder:

The address decoder plays a critical role in both read and write transfers. It examines the address provided by the master and ensures that the correct slave device is selected. For example, if the system has two slaves, the address decoder will map each slave to a specific address range. When the master sends an address, the decoder identifies which slave corresponds to that address and enables communication between the master and that particular slave.

Control Signals:

Several control signals are key to the operation of the APB protocol. The PADDR signal carries the address of the slave, while PWRITE indicates whether the operation is a read or a write. The PENABLE signal enables the selected slave to participate in the data transfer. Data flows from the master to the slave over the PWDATA line during a write operation, and from the slave to the master via the PRDATA line during a read operation.

Transaction Flow:

To summarize, the process begins with the master sending an address and determining whether it wants to perform a read or write operation. The address decoder identifies the target slave based on the address provided. Once the slave is selected, the master either writes data to the slave or reads data from it, depending on the value of the PWRITE signal. The PENABLE signal is used to manage the timing of when the slave should process the transfer. After the data exchange is completed, the master de-asserts PENABLE, marking the end of the transaction.

B. Schematic Diagram

The schematic diagram illustrates a system where a master communicates with two slave devices. Each slave is equipped with an 8-bit address bus (PADDR[7:0]), an 8-bit data bus (PWDATA[7:0]), and control signals for clock (PCLK), enable (PENABLE), reset (PRESETn), and write (PWRITE). The master employs two selection signals, PSEL1 and PSEL2, to determine which slave it interacts with.

Each slave in the system includes a register (RTL_REG) and a multiplexer (RTL_MUX). When the master sends data, it directs it to the selected slave based on the PSEL signals. The addressed slave then stores the incoming data in its register. Subsequently, the slave can read the data from its register and return it to the master. The schematic outlines the connections and functions of these signals, showing how they manage and control the data transfer between the master and the slaves.

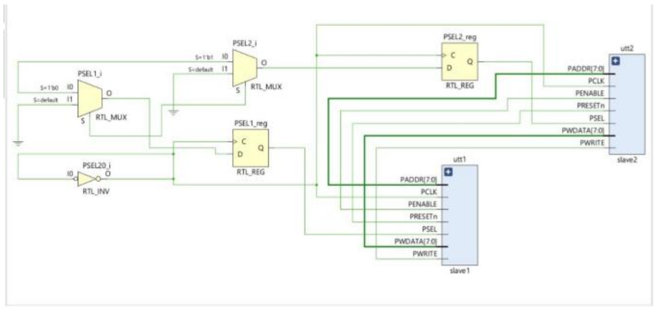


Fig 7: Schematic Diagram

## IV. RESULTS

In the AMBA APB protocol, communication between a single master and two slaves is straightforward and efficient for both read and write operations. When the master initiates a write operation, it sends the address of the target slave and places the data on the PWDATA line. The address decoder selects the appropriate slave, and the PWRITE signal is set to indicate that a write is occurring. The PENABLE signal activates the transaction, prompting the selected slave to capture and store the data. The transaction is completed when the master de-asserts PENABLE, signaling the end of the process.

In a read operation, the process is similar, but instead of sending data, the master retrieves it from the slave. The master sends the slave's address and asserts the PENABLE signal after setting the PWRITE signal to indicate a read. The selected slave then places the requested data on the PRDATA line, where the master can read it. Once the data is captured, the master de-asserts PENABLE, completing the read transaction. This ensures a smooth flow of information from the slave to the master.

Overall, the APB protocol provides an efficient way for the master to communicate with multiple slaves using simple control signals. Both read and write transactions are completed within a few clock cycles, ensuring low-latency data transfers.

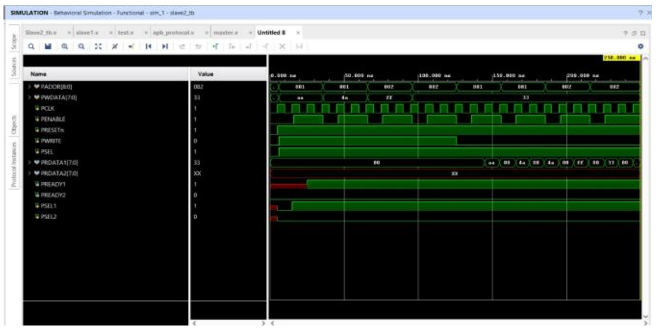


Fig 8: Read from Slave 1

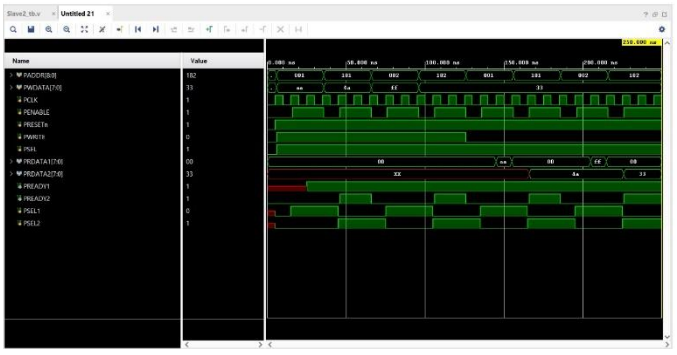


Fig 9: Read from Slave 2

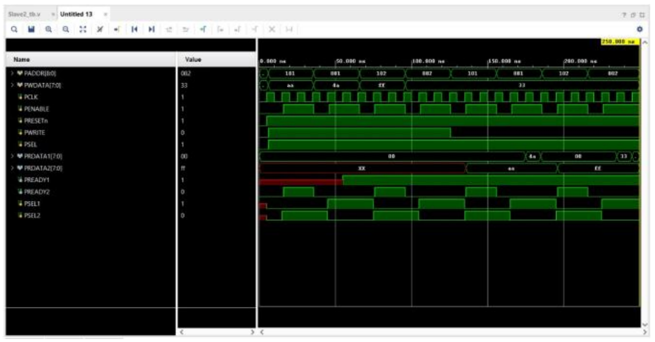


Fig 10: Write operation

PARAMETERS:-

In Vivado software, when designing communication protocols or interfaces, common parameters associated with addressing and control signals. These parameters are set in the IP configuration or HDL code, allowing you to manage how data and control signals are handled within a given system.

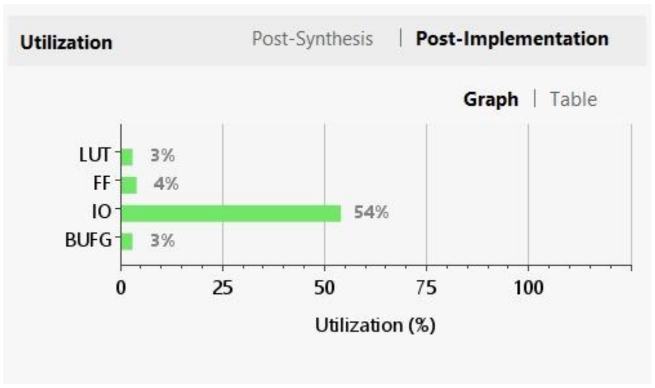
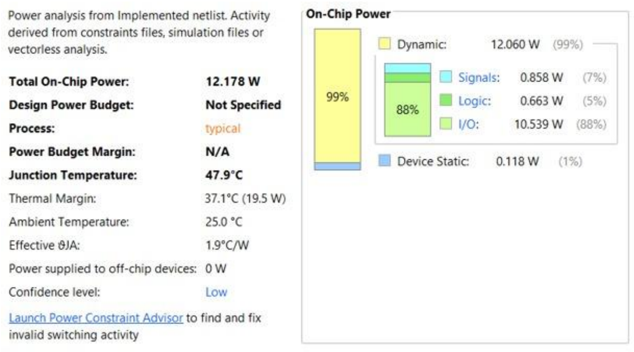


Fig 11: Parameters



Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

| | |
|---|---|
| **Total On-Chip Power:** | **12.178 W** |
| **Design Power Budget:** | **Not Specified** |
| **Process:** | typical |
| **Power Budget Margin:** | **N/A** |
| **Junction Temperature:** | **47.9°C** |
| Thermal Margin: | 37.1°C (19.5 W) |
| Ambient Temperature: | 25.0 °C |
| Effective ϑJA: | 1.9°C/W |
| Power supplied to off-chip devices: | 0 W |
| Confidence level: | Low |

Launch Power Constraint Advisor to find and fix invalid switching activity

**On-Chip Power**

| | | | |
|---|---|---|---|
| 99% | Dynamic: | 12.060 W | (99%) |
| 88% | Signals: | 0.858 W | (7%) |
| | Logic: | 0.663 W | (5%) |
| | I/O: | 10.539 W | (88%) |
| | Device Static: | 0.118 W | (1%) |

Fig 12: Power Analysis

## V. CONCLUSION

The AMBA APB protocol offers a straightforward and organized way for a single master to manage data transfers between multiple slaves. The read and write operations follow a systematic approach where the master initiates communication by addressing specific slaves, which then either provide data or accept and store it. The use of control signals such as PWRITE and PENABLE ensures proper coordination between the master and the slaves, resulting in smooth transactions and effective communication. The simplicity of the protocol's structure allows for predictable and easy-to-implement communication between the master and peripherals.

As system-on-chip (SoC) designs continue to evolve, the role of the AMBA APB protocol can be extended to support more complex peripheral configurations. Future developments may involve integrating more sophisticated addressing schemes or enhanced control mechanisms to support a larger number of slaves. Additionally, the APB protocol could be adapted to work alongside other protocols in hybrid bus systems, ensuring seamless communication across different components in more complex and scalable SoCs. Further refinements in the protocol could also involve improved error-handling mechanisms and more dynamic control for varying system needs.

## REFERENCES

[1]Akhilesh Kumar and Richa Sinha ,"DESIGN AND VERIFICATION ANALYSIS OF APB3 PROTOCOL WITH COVERAGE ",International Journal of Advances in Engineering & Technology, Nov 2011. ©IJAET ISSN: 2231-1963

[2]Padmaprabha Jain ,Satheesh Rao ,"Design and Verification of Advanced Microcontroller Bus Architecture-Advanced Peripheral Bus (AMBA-APB) Protocol",Proceedings of the Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV 2021). IEEE Xplore Part Number: CFP21ONG-ART; 978-0-7381-1183-4

[3] Muhammad Hafeez Sabparie, Emillia Noorsal, Suhana Sulaiman, and Azilah Saparon ,"Design and Simulation of Serial Peripheral Interface Core with APB Interfacing" ,JOURNAL OF ELECTRICAL AND ELECTRONIC SYSTEMS RESEARCH, VOL. 19 OCT 2021.

[4]Sandeep Jagre,Dr. Neelesh Gupta,Prof. Nishi Pandey,"Design and Implementation of AMBA APB Bridge with Low Power Consumption",IJSTE - International Journal of Science Technology & Engineering | Volume 4 | Issue 2 | August 2017 ISSN (online): 2349-784X

[5]Prashant Dwivedi, Neha-Mishra and Amit-Singh-Rajput ,"Assertion & Functional Coverage Driven Verification of AMBA Advance Peripheral Bus Protocol Using System Verilog ",2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT) | 978-1-7281-5791-7/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICAECT49130.2021.9392518.

[6] Vaishnavi R.K, Bindu.S, Sheik Chandbasha,"Design and Verification of APB Protocol by using System Verilog and Universal Verification Methodology",International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056 Volume: 06 Issue: 06 | June 2019 www.irjet.net p-ISSN: 2395-0072

[7]M. Kiran Kumar, Amrita Sajja Dr. Fazal Noorbasha," Design and FPGA Implementation of AMBA APB Bridge with Clock Skew Minimization Technique ",OSR Journal of VLSI and Signal Processing (IOSR-JVSP) Volume 7, Issue 3, Ver. I (May. - June. 2017), PP 42-45

e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197

[8]G Prathibha,G Prathibha,"APB Bridge Based on AMBA 4.0",International Journal of Engineering Research & Technology (IJERT)ISSN: 2278-0181Vol. 2 Issue 9, September - 2013.\

[9]Dr. Dileep Reddy Bolla,Dr. SatyaSrikanth Palle,Dr. Keshavamurthy," DESIGN AND IMPLEMENTATION OF ADVANCED PERIPHERAL BUS (APB) PROTOCOL IN SYSTEMVERILOG",

[10]S.Ramana, M.Pavan Kumar, N.Bhaskar, S. China Ramu, & G.R. Ramadevi. (2018). Security tool for IOT and IMAGE

compression techniques. Online International Interdisciplinary Research Journal, {BiMonthly}, 08(02), 214–223. ISSN Number: 2249-9598

[11] S. Ramana, N. Bhaskar, S. China Ramu, M. V. Ramana Murthy." A Two-Level Authentication Protocol for Secure MCommerce Transactions using AMQP Protocol", Vol 2021: Issue 06, pp:844– 861.

[12] Swamy, Sirisati Ranga, and Sridhar Mandapati. "A fuzzy energy and security aware scheduling in cloud." Int J Eng Technol 7.2 (2017): 117-124.

[13] S. Ramana, N. Bhaskar , M. V. Ramana Murthy , G. R. Rama Devi." A Two-Level Protocol For Secure Transmission Of Image Using IOT Enabled Devices", Volume 18, No. 5,pp:1040-1050 ,2021.

[14] Vuppula, Keerthi. "An advanced machine learning algorithm for fraud financial transaction detection." Journal For Innovative Development in Pharmaceutical and Technical Science (JIDPTS) 4.9 (2021).

[15] Vuppula, Keerthi. "Smart Door Unlock System Using Face Recognition and machine learning."