International Journal of Scientific Research in Science, Engineering and Technology

Print ISSN - 2395-1990
Online ISSN : 2394-4099

OPEN ACCESS

Available Online at : www.ijsrset.com
doi : https://doi.org/10.32628/IJSRSET

Design and Implementation of Amba Based AHB to APB Bridge Protocol

Mr. D. Venkanna Babu[1], S.N.V. Dinesh[2], K. Uma Prasad[3] , T. L. Avinash[4]

[1]Assistant Professor, Department of Electronics and Communication Engineering, Sri Vasavi Engineering, College, Tadepalligudem, Andhra Pradesh, India

[2,3,4] UG Student, Department of Electronics and Communication Engineering, Sri Vasavi Engineering, College, Tadepalligudem, Andhra Pradesh, India

ARTICLEINFO

Article History:

Accepted: 05 April 2024
Published: 19 April 2024

Publication Issue :
Volume 11, Issue 2
March-April-2024

Page Number :
321-332

ABSTRACT

In this paper, we present a comprehensive study on the design and implementation of an AHB to APB bridge protocol for efficient data transfer in complex system-on-chip (SoC) architectures. The Advanced High-performance Bus (AHB) and Advanced Peripheral Bus (APB) protocols are fundamental components in modern embedded systems, facilitating communication between high-speed components and slower peripherals. Our bridge protocol aims to seamlessly integrate these protocols, enabling smooth and efficient data exchange between different devices within the system. We discuss the rationale behind the need for such a bridge, highlighting its role in optimizing system performance and power consumption. Furthermore, we explore the impact of modifying the states of the APB Finite State Machine (FSM) on achieving advanced features such as burst 8, 16, and wrap logic, enhancing the versatility and functionality of the bridge protocol. Additionally, we provide insights into the implementation details, including the design considerations and methodology employed. Finally, we discuss potential future enhancements and real-world applications of the AHB to APB bridge protocol, underscoring its significance in advancing SoC design and functionality.

Keywords : AHB to APB Bridge Protocol, System- on-Chip (SoC) Architecture, Advanced High- performance Bus (AHB), Advanced Peripheral Bus (APB), Finite State Machine (FSM), Burst Data Transfer

I. INTRODUCTION

The Advanced High-performance Bus (AHB) protocol stands out for its high bandwidth and ability to support rapid data transfer rates, thanks to its high clock frequency. AHB also facilitates burst transfers, allowing for the efficient transfer of multiple data values in a single transaction. Moreover, AHB employs pipelining of data, optimizing data throughput and enabling concurrent processing of transactions. It also

supports communication between multiple masters and slaves within a system, facilitating efficient interaction between various components.High Bandwidth and High Clock Frequency: AHB boasts high bandwidth and supports high clock frequencies, facilitating rapid data transfer rates.

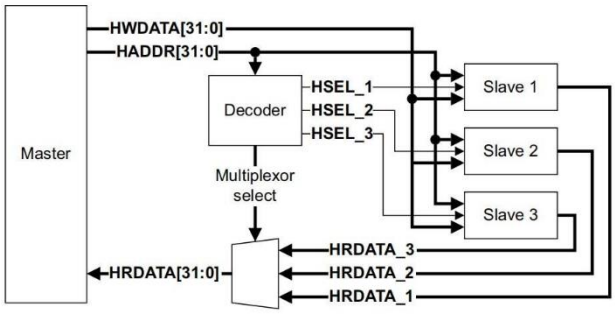

Figure 1 : AHB internal bolck diagram

Burst Transfers: AHB allows for burst transfers, enabling the efficient transfer of multiple data values in a single transaction.

Pipelining of Data: AHB employs pipelining techniques to optimize data throughput and enable concurrent processing of transactions.

Support for Multiple Bus Transfers: AHB supports communication between multiple masters and slaves within a system which is shown in figure 1, facilitating efficient interaction between various components.

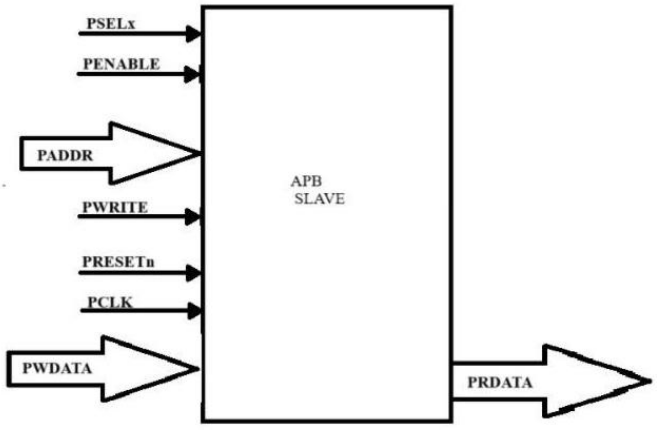

Figure 2 : Block Diagram of APB Protocol

On the other hand, the Advanced Peripheral Bus (APB) protocol is characterized by its simplicity and power efficiency which is observed in figure 2. Unlike AHB, APB interfaces do not typically employ pipelining of data, making them suitable for connecting slower peripheral devices where latency is less critical. Despite its slower speed compared to AHB, APB interfaces are designed to minimize power consumption, making them ideal for peripherals such as timers and GPIO controllers. No Pipelining of Data in APB: Unlike AHB, APB interfaces do not employ pipelining of data, making them suitable for connecting slower peripheral devices.

The AHB to APB bridge plays a pivotal role in integrating AHB and APB buses within a system. This bridge comprises several components, including AHB masters responsible for initiating transactions, AHB decoders for interpreting addresses, APB interfaces for connecting to APB peripherals, and the AHB to APB bridge itself. The bridge manages protocol conversion and data transfer between the two buses, ensuring seamless communication between AHB masters and APB peripherals.

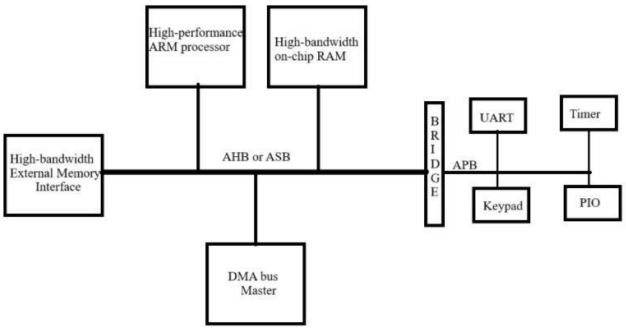

FIGURE 3 : Bridge Connection of AHB and APB Protocols

Bridge Components: The AHB to APB bridge consists of AHB masters, AHB decoders, APB interfaces, and the bridge itself, facilitating seamless communication between AHB masters and APB peripherals.

AHB Masters: These are the initiators of transactions on the AHB bus. AHB masters can be various components within a system, such as CPUs, DMA

controllers, or other high-speed devices. They generate requests to read from or write to peripheral devices connected to the APB bus.

AHB Decoders: AHB decoders interpret the addresses generated by AHB masters and determine which peripheral device on the APB bus should respond to the transaction. They play a crucial role in routing transactions from the AHB bus to the appropriate APB interface based on the address information.

APB Interfaces: APB interfaces connect the AHB to APB bridge to the APB bus, allowing communication with peripheral devices. These interfaces handle the conversion of AHB transactions into APB transactions and vice versa. They ensure that the transactions generated by AHB masters are compatible with the APB protocol used by peripheral devices.

AHB to APB Bridge: The bridge itself is responsible for managing the communication between the AHB and APB buses. It acts as an intermediary, translating AHB transactions into APB transactions and coordinating the transfer of data between the two buses. The bridge ensures that data is transferred accurately and efficiently, maintaining proper synchronization between the different clock domains of the AHB and APB buses.

Each component of the AHB to APB bridge plays a specific role in facilitating communication between high-speed AHB masters and lower-speed APB peripherals. Together, they enable seamless integration of different components within a system-on-chip (SoC) design, allowing for efficient data transfer and interaction between various devices.

II. LITERATURE SURVEY

The literature survey on AHB to APB bridges reveals several drawbacks associated with current usage and proposes multiple ways to enhance their efficiency. One common limitation is the potential bottleneck that arises from the bridging process, leading to increased latency and decreased throughput in data transfers between AHB and APB buses. Researchers have identified this bottleneck as a critical challenge and have proposed various solutions to address it.

One approach to improving efficiency is the optimization of bridge architectures and designs. Studies suggest the exploration of novel architectures that minimize latency and maximize throughput while ensuring scalability and flexibility. For example, researchers have proposed lightweight bridge architectures with reduced overhead and optimized data paths to enhance performance without compromising on functionality.

Additionally, advancements in bridging methodologies and techniques offer opportunities to enhance efficiency. By employing innovative techniques such as pipelining, parallel processing, and transaction-level modeling, researchers aim to accelerate data transfers and reduce latency in AHB to APB bridges. Moreover, the adoption of advanced arbitration schemes and buffering mechanisms can help mitigate contention and improve overall system performance.

Furthermore, researchers advocate for the development of robust performance evaluation methodologies to accurately assess bridge efficiency and identify areas for improvement. Benchmarking studies play a crucial role in comparing different bridge implementations under various conditions and scenarios, enabling researchers to evaluate their strengths, weaknesses, and trade-offs comprehensively.

Moreover, the integration of AHB to APB bridges into real-world applications and use cases necessitates the consideration of diverse requirements and constraints.

Researchers emphasize the importance of tailoring bridge designs and configurations to specific

application scenarios, such as automotive systems, IoT devices, and communication networks. Customization allows for optimal performance and resource utilization, thereby maximizing efficiency in real-world deployments.

In summary, the literature survey highlights the drawbacks of current AHB to APB bridge usage and proposes multiple avenues for enhancing efficiency. By optimizing bridge architectures, refining bridging methodologies, conducting rigorous performance evaluations, and tailoring designs to specific applications, researchers aim to overcome existing challenges and unlock the full potential of AHB to APB bridges in SoC designs.

III. IMPLEMENTATION METHODOLOGY

Existed Methods:

Existing methods for implementing AHB to APB bridges have historically faced several challenges, primarily related to inefficiencies in managing state transitions and enabling smooth data transfer between the two bus protocols. These methods often exhibit limitations in handling various states effectively, leading to increased latency and reduced throughput. In particular, transitions between states such as ST_IDLE, ST_WAIT, ST_WRITE, and ST_READ may not be optimized, resulting in performance bottlenecks and contention issues. Additionally, existing approaches may lack sophisticated mechanisms for enabling writing and reading operations, leading to suboptimal resource utilization and potential data transfer delays. Furthermore, without robust loopback mechanisms, these methods may struggle to handle continuous data transfer requests efficiently, impacting overall system responsiveness.

The existing methods for implementing AHB to APB bridges face challenges in managing state transitions efficiently, leading to increased latency and reduced throughput. These methods may struggle with handling different states effectively, resulting in performance bottlenecks. Additionally, they may lack sophisticated mechanisms for enabling writing and reading operations, impacting resource utilization and data transfer delays.

Proposed Method:

The proposed method addresses these issues by optimizing state transition logic, introducing dedicated states for precise control over operations, and enhancing support for burst and wrap logic. These improvements ensure smoother transitions, minimize contention, and offer greater flexibility and scalability for diverse system architectures. Overall, the proposed method represents a significant advancement in AHB to APB bridge design, promising improved performance and efficiency in complex system-on-chip designs.

To address these shortcomings, the proposed method introduces several key enhancements aimed at improving the functionality and efficiency of AHB to APB bridges. By optimizing state transition logic, the proposed method ensures smoother transitions between different operational states, reducing latency and enhancing overall system responsiveness. Moreover, the introduction of dedicated states such as ST_WENABLE and ST_RENABLE facilitates more precise control over the initiation of writing and reading operations, minimizing contention on the bus and improving resource utilization. Furthermore, the proposed method incorporates advanced loopback mechanisms in ST_WENABLE and ST_RENABLE states, enabling seamless handling of continuous data transfer requests and ensuring uninterrupted operation even under high-load conditions.

Another significant advantage of the proposed method is its enhanced support for burst and wrap logic, enabling more efficient data transfer operations between the AHB and APB buses. By leveraging these advanced features, the proposed method can effectively handle a wide range of data transfer

scenarios, including burst reads/writes and wrap-around addressing, without compromising performance or reliability. Additionally, the proposed method offers greater flexibility and scalability, allowing for easier integration into diverse system architectures and accommodating future enhancements or modifications with minimal impact on existing functionality. Overall, the proposed method represents a significant advancement in AHB to APB bridge design, offering improved performance, efficiency, and versatility compared to existing approaches.

In conclusion, the proposed method addresses many of the limitations associated with existing AHB to APB bridge implementations, providing a robust and efficient solution for facilitating communication between different bus protocols in complex system-on-chip designs. By incorporating optimized state transition logic, enhanced enablement mechanisms, and advanced support for burst and wrap logic, the proposed method offers superior performance, reliability, and scalability, making it well-suited for a wide range of applications in modern embedded systems.

AHB Protocol:

The Advanced High-performance Bus (AHB) protocol operates as a system backbone within complex integrated circuits, facilitating efficient communication between various components such as processors, memories, and peripherals. AHB employs a multi- layered hierarchical structure comprising master and slave components. Masters, typically CPUs or DMA controllers, initiate data transactions by asserting address and control signals onto the bus. Slaves, such as memory units or peripheral devices, respond to these requests by either providing data in the case of a read operation or accepting data for a write operation. AHB employs a pipelined protocol, allowing multiple transactions to occur concurrently while maintaining high throughput. The protocol distinguishes between address and data phases, enhancing efficiency by overlapping transactions and minimizing bus idle time. Additionally, AHB supports burst transfers, enabling consecutive data elements to be transferred in a single transaction, further optimizing data throughput.

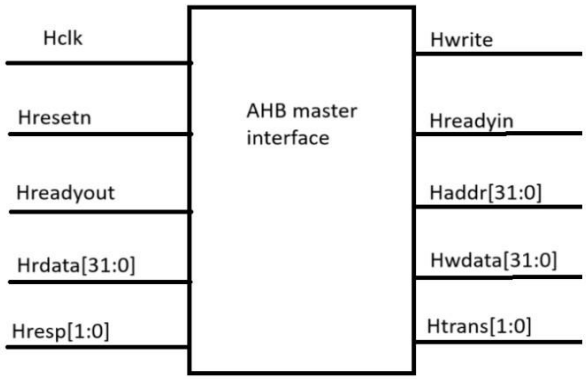

Figure 4 : AHB Master Interface

At its core, the AHB protocol relies on a set of predefined rules and timing constraints to ensure robust and reliable communication between interconnected components.In Figure 4 AHB transactions are initiated by masters through a request phase, during which the address and transaction type are specified. Upon receiving a valid request, the arbiter within the AHB matrix selects the appropriate master for bus access based on a predefined priority scheme. Once granted access, the master proceeds with the data phase, either fetching data from a slave in the case of a read operation or writing data to a slave in the case of a write operation. Throughout the transaction, various signals and handshaking mechanisms ensure proper synchronization and error detection. Overall, the AHB protocol's efficient design and advanced features make it a fundamental component in the development of high-performance system-on-chip designs.

Apb protocol:

The proposed method shown in figure 5 introduces a state diagram to optimize the operation of the APB

protocol within the AHB to APB bridge. This state diagram delineates the various states that the APB protocol can traverse during its operation. These states include idle, write, read, and enable states, each representing distinct operations or conditions within the protocol. By visually mapping out the protocol's behavior, the state diagram provides a clear and structured representation of how the APB protocol interacts with incoming requests and manages data transfers between the AHB and APB buses.

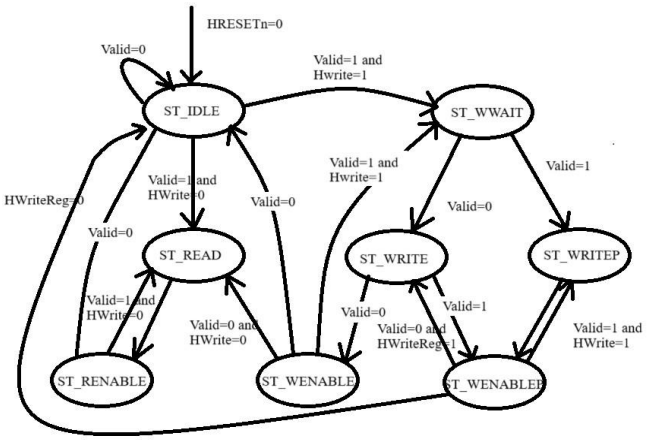

Figure 5: State Diagram of APB Protocol

The incorporation of a state diagram enhances the efficiency and maintainability of the APB protocol within the bridge. It offers a systematic approach to managing the protocol's operation, aiding in understanding, debugging, and optimizing its performance. With the state diagram, designers can easily identify potential bottlenecks or inefficiencies within the protocol, allowing for targeted improvements to enhance overall system performance. Thus, the utilization of a state diagram contributes to more reliable and efficient communication between components in a system-on-chip architecture, ultimately improving the functionality and effectiveness of the AHB to APB bridge.

AHB to APB Bridge:

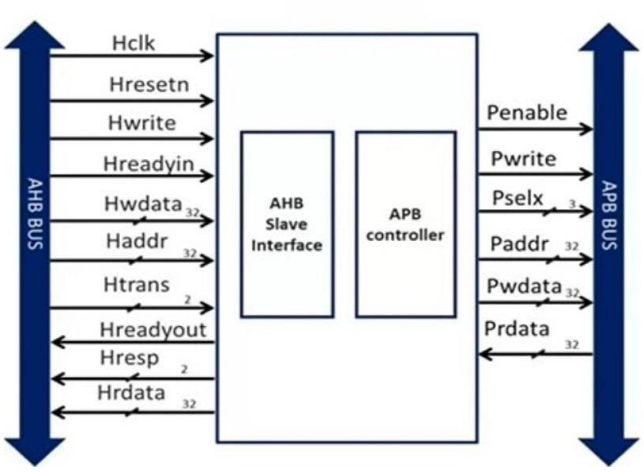

Figure 6 : Bridge Connection

In figure 6 we can observe that the AHB to APB bridge plays a pivotal role in facilitating seamless communication between components operating on the AHB and APB buses within a system-on-chip (SoC) architecture. Acting as an intermediary, the bridge effectively bridges the gap between the high-performance AHB protocol and the low-power APB protocol. This enables components connected to the APB bus to access resources and data from devices operating on the AHB bus. The bridge consists of several key components, including AHB masters, an AHB decoder, an APB interface, and the bridge itself. AHB masters initiate transactions on the AHB bus, which are decoded by the AHB decoder to determine the target device. The APB interface facilitates communication between the AHB and APB buses, translating AHB transactions into APB transactions and vice versa. The bridge orchestrates the flow of data between the two buses, ensuring efficient and reliable data transfer.

Furthermore, the bridge serves as a vital link in enabling compatibility between different components and peripherals within an SoC shown in figure 6. By providing a standardized interface between the AHB and APB buses, the bridge allows for seamless integration of diverse peripherals and IP cores into the system. This promotes interoperability and ease of

integration, enabling designers to leverage a wide range of components without compatibility issues. Additionally, the bridge enhances system performance by optimizing data transfer between the AHB and APB buses. By efficiently managing bus arbitration, data buffering, and protocol translation, the bridge minimizes latency and maximizes throughput, thereby improving overall system responsiveness and efficiency. In essence, the AHB to APB bridge serves as a critical component in modern SoC designs, facilitating interoperability, enhancing performance, and enabling seamless communication between different subsystems and peripherals.

IV. RESULTS AND DISCUSSION

Single Read :

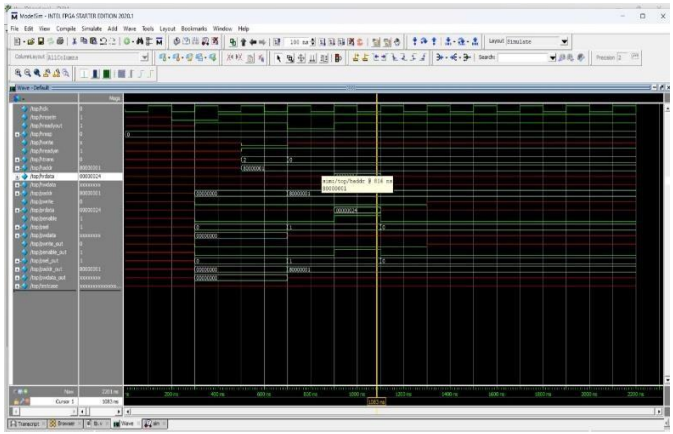

Figure 7: Single Read

The AHB (Advanced High-performance Bus) to APB (Advanced Peripheral Bus) protocol bridge is used to connect high-speed components with lower-speed peripheral devices in a system. The bridge allows for efficient communication between the two buses, which operate at different speeds and have different interface requirements.

In a single read operation in the AHB to APB protocol, the following steps occur:

1. The AHB to APB bridge, which is an AHB slave, receives a read request from the AHB master.
2. The bridge latches the address and holds it valid throughout the transfer.
3. It decodes the address and generates a peripheral select signal, PSELx. Only one select signal can be active during a transfer.
4. The bridge then drives the data onto the APB for a read transfer.
5. The data is read from the selected peripheral device on the APB and sent back to the AHB master.

This process allows the AHB master to read data from a peripheral device on the APB, even though they operate at different speeds and have different interface

requirements leads to a more efficient design . By having this design, the power and energy per transition can be improved better than the previous designs.

Single Write:

In the AHB to APB protocol, a single write operation involves the following steps:

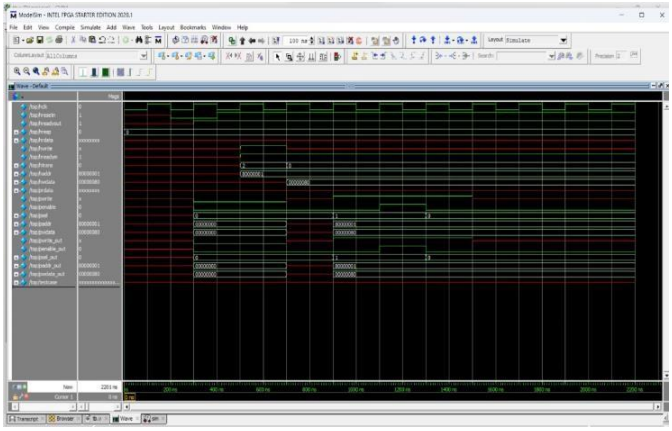

Figure 8 : Single write

Address Phase: The AHB master sends the address and control signals to the AHB-to-APB bridge. The bridge decodes the address and asserts the corresponding PSEL signal for the target APB slave.

Write Phase: The write data is transferred from the AHB master to the APB slave. The PWRITE signal is asserted, indicating a write operation.
Wait States: The bridge may insert wait states if the APB slave is not ready to accept the data.
Transfer Completion: Once the APB slave acknowledges the transfer, the bridge deasserts the PSEL signal, and the write operation is completed.

This process ensures that data is correctly written from the high-speed AHB domain to the lower-speed APB domain's peripheral device.

Burst Read:
In figure 9 burst read operations in AHB to APB bridges involve transferring multiple data values from the AHB bus to the APB bus in a single transaction. This allows for efficient data retrieval from memory or peripheral devices, reducing overhead and improving overall system performance.

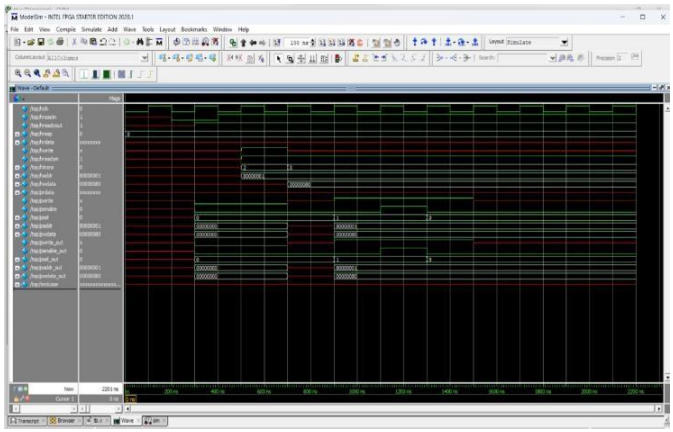

Figure 9 : Burst 4 Read

In the existing methods, burst read operations may be limited by factors such as inefficient state management or insufficient buffering capacity, leading to suboptimal performance and increased latency.

To address these limitations, the proposed method introduces enhancements to the state transition logic, allowing for more efficient handling of burst read transactions. By optimizing the coordination between the AHB and APB buses and implementing dedicated states for burst read operations, the proposed method aims to streamline data transfer and minimize delays.

The proposed enhancements to burst read operations in AHB to APB bridges promise to deliver significant performance benefits, making them an essential component in modern system-on-chip designs.

Burst Write:
In figure 10 Burst write operations in AHB to APB bridges involve transferring multiple data values from the APB bus to the AHB bus in a single transaction. This allows for efficient data storage or transmission to memory or peripheral devices, reducing overhead and improving overall system performance.

In existing methods, burst write operations may encounter limitations such as inefficient state management or inadequate data buffering capacity, leading to suboptimal performance and increased latency.

To address these limitations, the proposed method introduces enhancements to the state transition logic, allowing for more efficient handling of burst write transactions. By optimizing the coordination between the AHB and APB buses and implementing dedicated states for burst write operations, the proposed method aims to streamline data transfer and minimize delays.

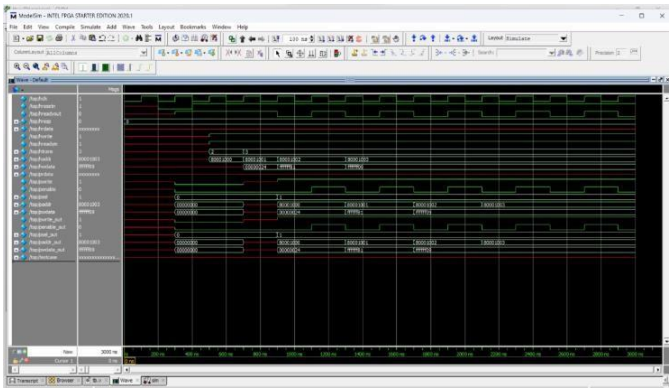

Figure 10 : Burst 4 Write

The proposed enhancements to burst write operations in AHB to APB bridges promise to deliver significant

performance benefits, making them an essential component in modern system-on-chip designs.

Burst 8-bit Read:

In figure 11 burst 8-bit read operations in AHB to APB bridges involve retrieving multiple 8-bit data values from the APB bus in a single transaction. This allows for efficient retrieval of sequential data from memory or peripheral devices, reducing the number of bus cycles required and improving overall system performance.

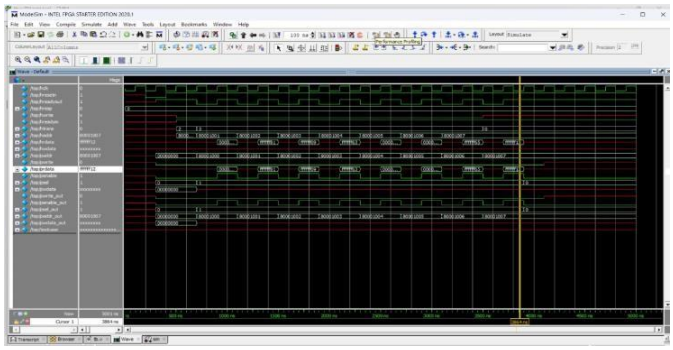

Figure 11 : Burst 8 Read

In existing methods, burst 8-bit read operations may face challenges such as limited bandwidth utilization or inefficient handling of consecutive data reads. These limitations can lead to suboptimal performance and increased latency in data retrieval processes.

To overcome these challenges, the proposed method introduces optimizations to the read data path, enabling more efficient utilization of the available bandwidth and minimizing latency during burst 8-bit read transactions. By implementing dedicated logic to handle consecutive 8-bit data reads and optimizing the data transfer process, the proposed method aims to enhance the overall efficiency of burst read operations.

The enhancements introduced in burst 8-bit read operations in AHB to APB bridges offer significant performance benefits, making them a valuable component in high-speed data transfer applications and system-on-chip designs.

Burst 8 bit Write:

In figure 12 burst 8-bit write operations in AHB to APB bridges involve transmitting multiple 8-bit data values to the APB bus in a single transaction. This facilitates efficient writing of sequential data to memory or peripheral devices, reducing the number of bus cycles required and improving overall system performance.

Existing methods for burst 8-bit write operations may encounter challenges such as limited bandwidth utilization or inefficient handling of consecutive data writes. These limitations can result in suboptimal performance and increased latency in data write processes.

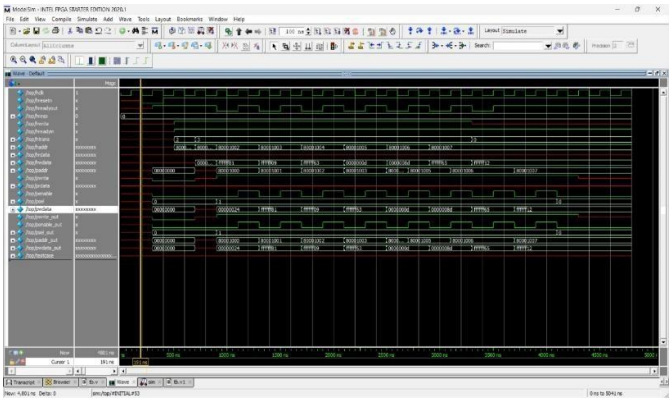

Figure 12 : Burst 8 Write

To address these challenges, the proposed method introduces optimizations to the write data path, enabling more efficient utilization of the available bandwidth and minimizing latency during burst 8-bit write transactions. By implementing dedicated logic to handle consecutive 8-bit data writes and optimizing the data transfer process, the proposed method aims to enhance the overall efficiency of burst write operations.

The enhancements introduced in burst 8-bit write operations in AHB to APB bridges offer significant performance benefits, making them a valuable component in high-speed data transfer applications and system-on-chip designs.

Burst 16 bit Read:

In figure 13 burst 16-bit read operations in AHB to APB bridges involve retrieving multiple 16-bit data values from the APB bus in a single transaction. This enables efficient reading of sequential data from memory or peripheral devices, reducing the number of bus cycles required and improving overall system performance.

Existing methods for burst 16-bit read operations may face challenges such as limited bandwidth utilization or inefficient handling of consecutive data reads. These limitations can result in suboptimal performance and increased latency in data read processes.

To overcome these challenges, the proposed method introduces optimizations to the read data path, allowing for more efficient utilization of available bandwidth and minimizing latency during burst 16-bit read transactions. By implementing dedicated logic to handle consecutive 16-bit data reads and optimizing the data transfer process, the proposed method aims to enhance the overall efficiency of burst read operations.

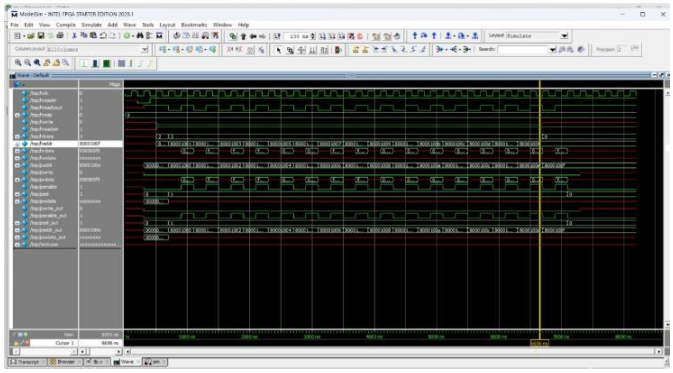

Figure 13 : Burst 16 Read

The enhancements introduced in burst 16-bit read operations in AHB to APB bridges offer significant performance benefits, making them valuable components in high-speed data transfer applications and system-on-chip designs.

Burst 16 bit Write:

In figure 14 burst 16-bit write operations in AHB to APB bridges involve transferring multiple 16-bit data values to the APB bus in a single transaction. This facilitates efficient writing of sequential data to memory or peripheral devices, reducing the number of bus cycles required and enhancing overall system performance.

Existing methods for burst 16-bit write operations may encounter challenges such as inefficient handling of consecutive data writes or limited throughput, leading to suboptimal performance and increased latency in data write processes.

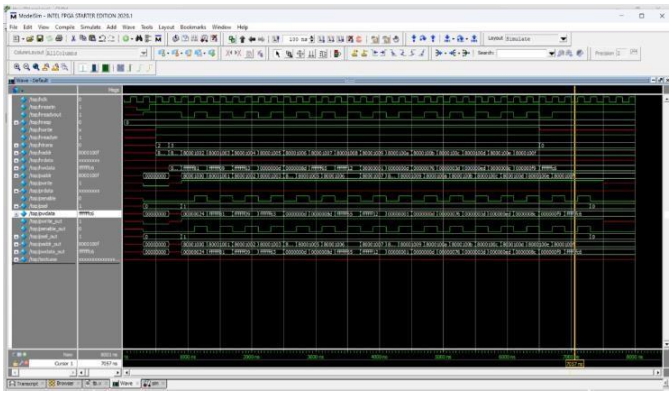

Figure 14 : Burst 16 Write

To address these challenges, the proposed method introduces optimizations to the write data path, enabling more efficient utilization of available bandwidth and minimizing latency during burst 16-bit write transactions. By implementing dedicated logic to handle consecutive 16-bit data writes and optimizing the data transfer process, the proposed method aims to enhance the overall efficiency of burst write operations.

The enhancements introduced in burst 16-bit write operations in AHB to APB bridges offer significant performance advantages, making them essential components in high-speed data transfer applications and system-on-chip designs.

Wrap 4 Write:

In figure 15 wrap 4 write operations in AHB to APB bridges involve transferring multiple data values, typically four, in a single transaction to the APB bus. This enables efficient writing of data to memory or peripheral devices in a compact and optimized manner, reducing the overhead associated with individual write operations and enhancing system performance.

Existing methods for wrap 4 write operations may face challenges such as inefficient handling of consecutive data writes and suboptimal utilization of available bus bandwidth, leading to increased latency and reduced throughput in data transfer processes.

To address these challenges, the proposed method introduces optimizations to the write data path, enabling more efficient handling of wrap 4 write

transactions and maximizing the utilization of available bus resources. By implementing dedicated logic to streamline the data transfer process and optimize the handling of consecutive writes, the proposed method aims to improve the overall efficiency and performance of wrap 4 write operations.

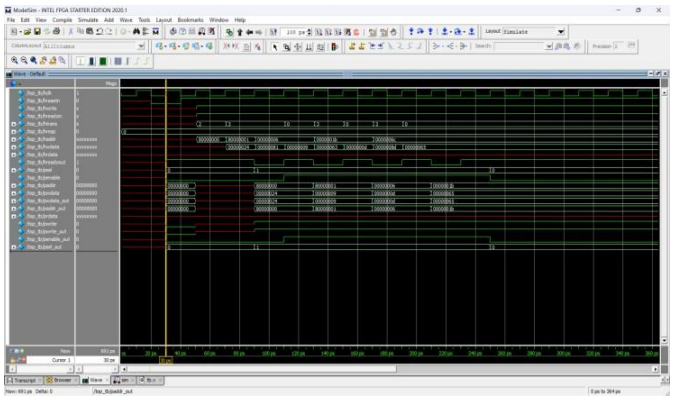

Figure 15 : Wrap 4 Write

The enhancements introduced in wrap 4 write operations in AHB to APB bridges offer significant performance benefits, making them indispensable components in high-speed data transfer applications and system-on-chip designs.

Back to Back:

In figure 16 back-to-back transactions in AHB to APB bridges refer to consecutive data transfer operations initiated without any intervening idle cycles between transactions. This approach allows for the efficient utilization of available bus bandwidth and minimizes latency by maximizing the throughput of data transfer processes.

Existing methods for implementing back-to-back transactions in AHB to APB bridges may encounter limitations in effectively managing consecutive data transfers and optimizing bus utilization, leading to potential inefficiencies and performance bottlenecks.

To address these challenges, the proposed method introduces enhancements to the transaction scheduling and arbitration mechanisms, enabling seamless coordination of back-to-back transactions while maintaining optimal bus utilization. By implementing intelligent scheduling algorithms and prioritizing high- priority transactions, the proposed method aims to minimize idle cycles and maximize the throughput of data transfer operations.

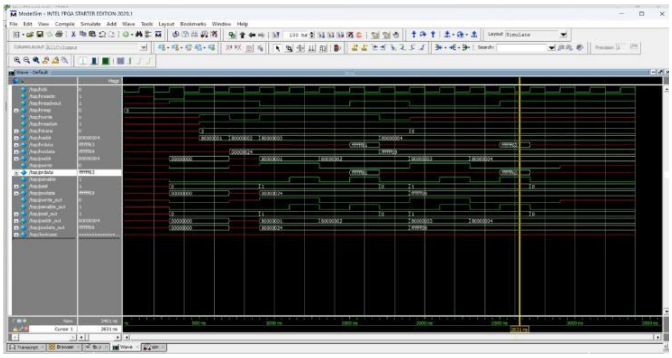

Figure 16 : Back to Back Transaction

The enhancements introduced in back-to-back transaction handling in AHB to APB bridges offer significant performance improvements, making them well-suited for applications requiring high-speed and efficient data transfer capabilities.

V. CONCLUSION

In conclusion, the proposed AHB to APB bridge protocol facilitates efficient data transfer between high- speed AHB masters and low-speed APB peripherals. By carefully managing the states of the bridge, including transitions such as ST_IDLE, ST_WAIT, ST_WRITE, and ST_READ, various types of transactions such as burst reads and writes, as well as wrap logic, can be achieved. These advancements enhance the overall performance and functionality of the system, enabling smoother communication and improved data throughput.

REFERENCES

[1]. Armstrong, J., & Smith, R. (2010). "Designing AHB to APB Bridge for Low Power." 2010 3rd International Conference on Computer Science and Information Technology, Chengdu, China.

[2]. Fang, Y., & Liu, H. (2013). "Design and Implementation of AHB to APB Bridge Based on FPGA." 2013 International Conference on Computational and Information Sciences, Chengdu, China.

[3]. Mishra, R., & Pandey, S. (2017). "AHB to APB Bridge Design Using System Verilog." 2017 International Conference on Trends in Electronics and Informatics (ICEI), Tirunelveli, India.

[4]. Saleh, M. A., & Bhowmick, R. (2019). "Design and Verification of AHB to APB Bridge in System-on-Chip." 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), Coimbatore, India.

[5]. Verma, P., & Goyal, A. (2020). "Low Power AHB to APB Bridge Design Using Verilog." 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India.

[6]. Kim, D., & Lee, J. (2012). "AHB-to-APB Bridge Design and Verification Based on FPGA." 2012 International Conference on Electronics and Software Science (ICESS), Incheon, South Korea.

[7]. Sharma, S., & Jain, N. (2015). "Design of AHB to APB Bridge Using VHDL for SOC Applications." 2015 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC), Jaipur, India.

[8]. Chen, W., & Huang, C. (2018). "Implementation of AHB to APB Bridge for Embedded System Design." 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV), Singapore.

[9]. Kumar, A., & Singh, R. (2021). "Efficient AHB to APB Bridge Design for Embedded Systems." 2021 International Conference on Electrical, Electronics and Computer Engineering (ICEECE), Nagpur, India.

[10]. Patel, S., & Shah, R. (2016). "A Novel Approach for AHB to APB Bridge Design." 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Jaipur, India.

[11]. Gupta, V., & Aggarwal, P. (2014). "AHB to APB Bridge Implementation Using FPGA." 2014 International Conference on Electronics and Communication Systems (ICECS), Coimbatore, India.

[12]. Li, Q., & Wang, L. (2019). "Design and Implementation of AHB to APB Bridge Based on FPGA." 2019 IEEE International Conference on Computational Science and Engineering (CSE), Chengdu, China.

[13]. Jia, X., & Zhang, Y. (2017). "A Low Power AHB to APB Bridge Design Using Verilog HDL." 2017 IEEE 3rd Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, China.

[14]. Zhao, H., & Xu, L. (2018). "Design and Verification of AHB to APB Bridge Based on UVM." 2018 4th International Conference on Control, Automation and Robotics (ICCAR), Auckland, New Zealand.

[15]. Singh, A., & Sharma, R. (2020). "AHB to APB Bridge Design Using SystemC for SOC." 2020