Design and Verification of AHB Bus Protocol using Vivado Tool

Arjun[1], Chetan[1], Guruprasada L[1], Shreyas Gowda B S[1]
[2]Dr. Ravi J

[1]Students, Global Academy of Technology,

[2]Professor, Global Academy of Technology, Bengaluru 560098

Abstract

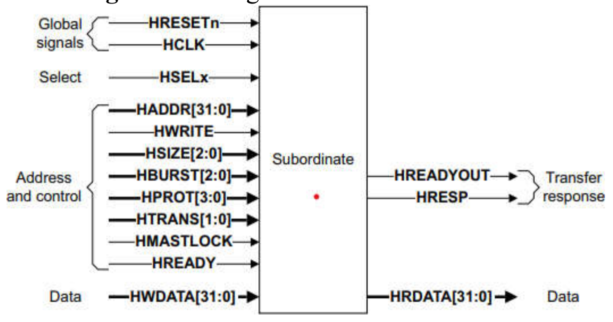
The AMBA AHB protocol is an on-chip communication standard that defines the rules for transferring data between master and slave devices. It's widely used in System -On -Chip (SoC) designs and microcontrollers. A M B A includes different bus architectures, such as Advanced System Bus (ASB), Advanced Peripheral Bus (APB), and Advanced High-Performance Bus (AHB). A H B stands out as the high-performance and high-bandwidth option, making it the preferred choice for systems with high-frequency clocks. This makes it popular for system designers. Verification is a crucial part of VLSI design to ensure that everything works as expected. This project focuses on designing the AHB protocol with a single master and multiple slaves using System Verilog. It also includes verification using Hardware Verification Languages (HVL) like System Verilog. The design and verification are carried out using the Vivado tool, with waveform simulations to check the results.

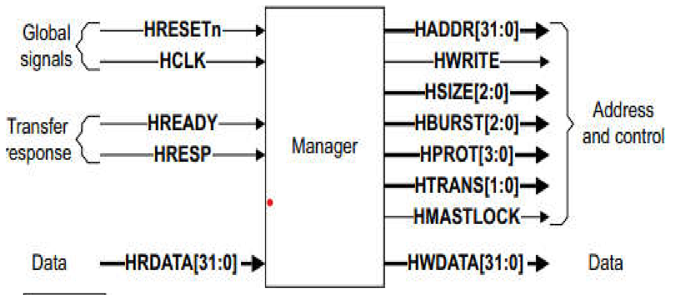
Keywords: Wrap, AXI, INCR, AMBA, System Verilog

I. INTRODUCTION

The three buses that make up AMBA. Embedded microcontrollers can be utilized with the Advanced System Buses (ASBs), which are the most efficient buses. An advanced peripheral bus (APB) link provides low-voltage bandwidth. The goal of the most recent bus generation, the AHB bus, is to manage needs while adhering to the specifications of a functional design style. a typical system bus with high of t bandwidth throughput and support for various bus managers. they make he functionalities needed for sophisticated, standard clock systems. Among these is AHB Lite, a basic kind with numerous slaves and a single master. Thus, complicated procedures like mediation confirmation or retries are not necessary. The most crucial component of the VLSI area is verification. It is imperative to verify the functioning of all SoC designs to ascertain if they align with the provided information or not. The semiconductor sector benefits from verification as it boosts production. However, verification processes need to be quick and effective, hence, UVM is used to achieve effective authentication. The authentication process requires adherence to the standard method. In comparison to other verification techniques, it is more rapid, reusable, efficient, and portable.

Figure 2: Manager Interface





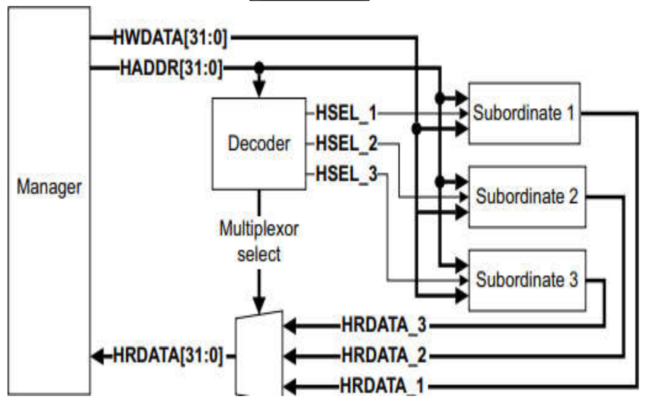


Figure 3:Subordinate Interface

Operation:

1 Requirement Analysis: Begin by thoroughly understanding the AHB bus protocol specification, including its various interfaces and their respective functionalities. Identify specific design requirements, such as data width, address space, burst types, and transaction types, to tailor the protocol implementation.

2 Modular Design Architecture: Adopt a modular approach to design, breaking down the protocol into distinct components like master, slave, and interconnect modules. Leverage System Verilog's features for encapsulation, parameterization, and hierarchical design to ensure scalability and reusability.

3 Interface Definition and Signal Mapping: Define the interface ports and signal connections for each module, adhering to the protocol specification. Establish clear communication paths between modules, ensuring proper data and control flow.

4 RTL Coding: Implement the protocol design in System Verilog, emphasizing coding guidelines, naming conventions, and structured coding practices to enhance readability and maintainability. Utilize data structures like arrays and structs to organize and manage protocol-specific information efficiently.

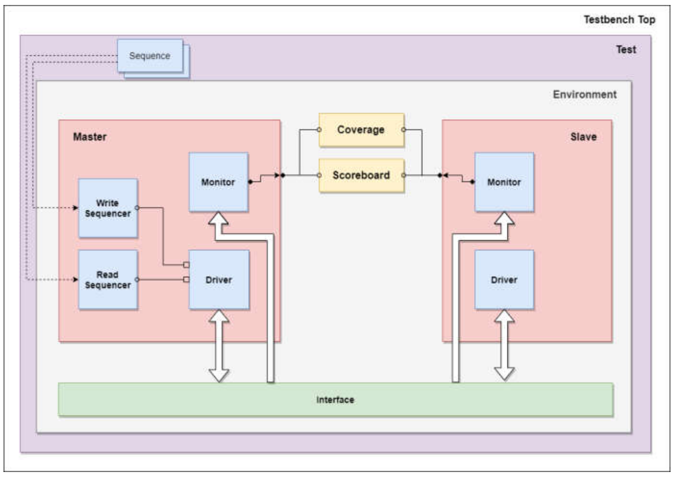
5 Functional Verification: Develop a comprehensive testbench environment, encompassing the AHB protocol design and relevant test scenarios. Employ constrained random testing to generate diverse stimuli, covering a wide range of transaction types, address ranges, and data widths. Implement assertions to verify protocol-specific properties and constraints, ensuring compliance with the standard.

6 Functional Coverage Analysis: Define a set of coverage goals to track the completeness of the verification process. Monitor the coverage metrics to identify untested scenarios and refine the testbench to achieve higher coverage.

7 Performance Verification: Evaluate the performance of the AHB protocol design by conducting simulations with varying traffic loads, transaction rates, and data transfer sizes. Analyze the results to ensure that the design meets specified throughput and latency requirements.

8 Corner Case Testing: Design specialized test cases to exercise edge conditions and corner cases, such as boundary values, exceptional scenarios, and asynchronous events. Verify that the AHB protocol implementation gracefully handles these critical situations.

9 Documentation and Reporting: Maintain comprehensive documentation, including design specifications, test plans, verification results, and any design decisions or tradeoffs madeduring the project. Generate detailed reports summarizing the design and verification process, highlighting key achievements and areas for potential improvement



Burst operation :

The AHB protocol supports a variety of data transfer types, including single transfers, undefined-length bursts, and fixed-length bursts of 4, 8, or 16 beats. These burst transfers can either be incrementing or wrapping. In an incrementing burst, the address of each subsequent transfer increases linearly, accessing consecutive memory locations. In contrast, a wrapping burst resets the address within a specific boundary once it reaches the end of the defined range, which is especially useful for circular buffer operations.
The boundary at which wrapping occurs is determined by the size of each data transfer (controlled by the **HSIZE** signal) multiplied by the number of beats in the burst (defined by **HBURST**). For example, a 4-byte word transfer in a 4-beat wrapping burst will wrap at 16-byte boundaries, such as addresses 0x30 and 0x3C. If a manager (master device) encounters a limitation, such as a 1KB address boundary, it may not be able to initiate an incrementing burst. In such cases, it can fall back to performing a single transfer or a burst with a single beat followed by an undefined-length burst. Proper alignment is important in burst transfers. Each transfer in a burst must start at an address that matches the transfer size, meaning word (4-byte) and halfword (2-byte) transfers should be aligned to 4-byte and 2-byte boundaries, respectively. Misaligned addresses can lead to incorrect behaviour or simulation errors. In fact, earlier versions of the AHB specification (Issues A and B) mandated that IDLE shared addresses must also be aligned to avoid such issues.
The AHB protocol includes several types of burst modes. The most basic is the Single Data Transfer (SDT), where only one data item is moved per transaction. More advanced bursts include INCR4, INCR8, and INCR16, where the address increments by one word after each transfer, allowing for 4, 8, or 16 consecutive data movements. In the Wrap Burst mode, the address does not follow a straight path but instead loops within a set boundary, making it ideal for operations involving circular data buffers.

In the protocol, error responses are essential for maintaining the integrity and reliability of data transfers. When an error occurs during a transaction, the AHB protocol defines several mechanisms for handling and communicating these errors.
Here are some common error responses in the

AHB protocol: •

OKAY: This response indicates that the transfer was successful without any errors. The master receives this response when the slave successfully processes the transaction.

• **ERROR**: The ERROR response indicates that an error occurred during the transaction. This could be due to various reasons such as bus contention, data corruption, or a violation of the protocol's timing requirements. When a slave detects an error, it asserts the ERROR response to signal to the master that the transaction was unsuccessful.

• **RETIRE**: The RETIRE response is typically used in split transactions where a master sends a request but does not wait for the response immediately. When the slave finishes processing the request, it sends a RETIRE response to indicate that the request has been completed. This allows the master to continue with other transactions without waiting for the response.

• **SPLIT**: In a split transaction, if a slave cannot immediately process a request, it may respond with a SPLIT response to indicate that the transaction has been split into multiple phases. The master can then continue with other transactions while waiting for the completion of the split transaction.

• **RETRY**: The RETRY response indicates that the slave is temporarily unable to process the transaction and requests the master to retry the transaction later. This could happen if the slave is busy or if there is contention on the bus.

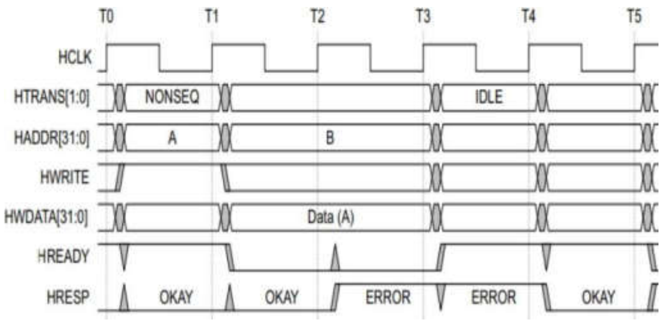


Fig. 4: Data transfer with error response

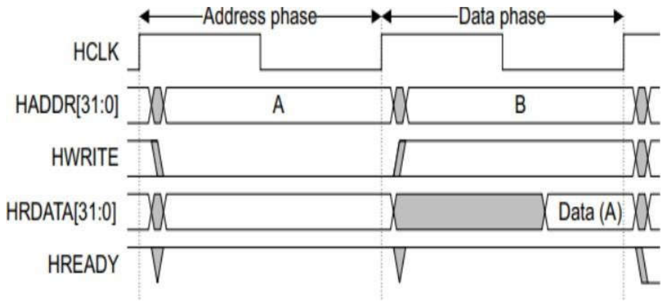


Fig 5: Read transfer

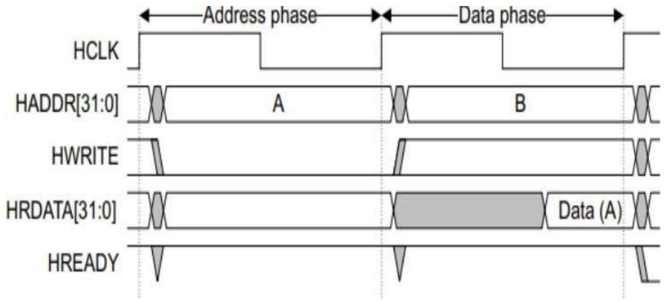
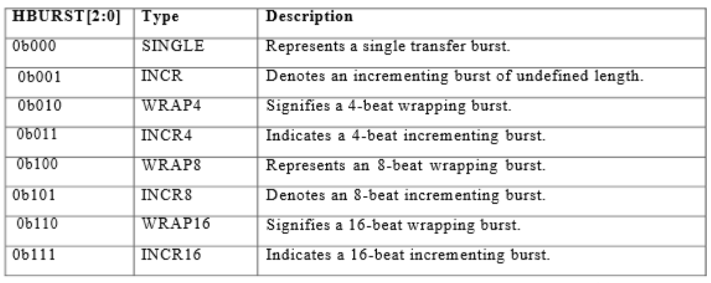


Fig 6:Write transfer



HBURST[2:0]	Type	Description
0b000	SINGLE	Represents a single transfer burst.
0b001	INCR	Denotes an incrementing burst of undefined length.
0b010	WRAP4	Signifies a 4-beat wrapping burst.
0b011	INCR4	Indicates a 4-beat incrementing burst.
0b100	WRAP8	Represents an 8-beat wrapping burst.
0b101	INCR8	Denotes an 8-beat incrementing burst.
0b110	WRAP16	Signifies a 16-beat wrapping burst.
0b111	INCR16	Indicates a 16-beat incrementing burst.

Table 4: lists the possible burst types.

II. Locked Transfer

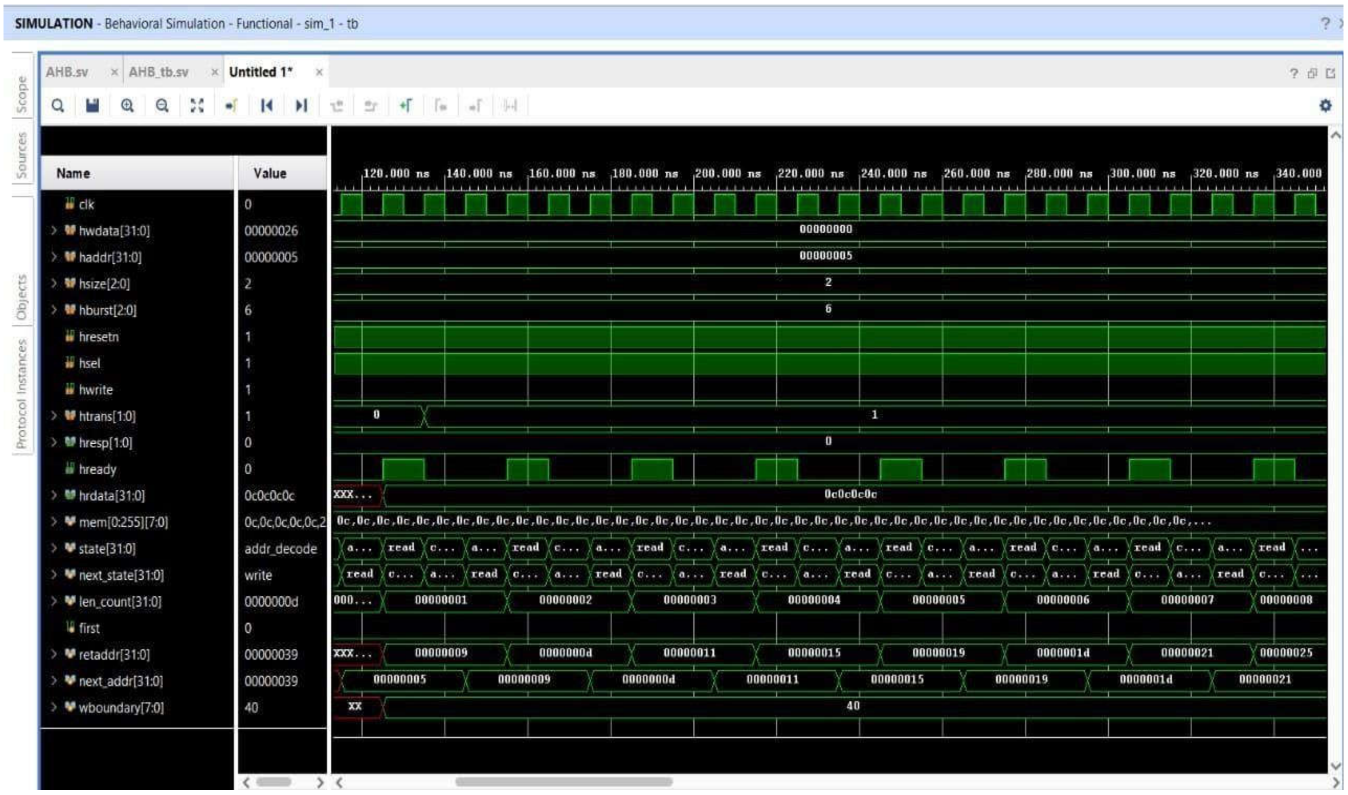
When a master needs to perform a sequence of operations on a shared resource without interruption, it can initiate a locked transfer Lock Phase.
The master initiates a locked transfer by issuing a request to acquire the lock on the shared resource. This request includes the LOCK signal asserted by the master. Response Phase,The slave responsible for managing the shared resource responds to the lock request. If the resource is available, the slave responds with an OKAY response, indicating that the lock has been acquired. If the resource is currently locked by another master, the slave responds with a RETRY or SPLIT response, indicating that the lock is not available at the moment.
Another phase of locked transfer is the Transfer Phase. Once the lock is acquired, the master proceeds with the sequence of operations it needs to perform on the shared resource. These operations typically involve reading from or writing to the resource. Unlock Phase: After completing the sequence of operations, the master releases the lock by issuing a request to unlock the resource.

Basic transfers

The system using the Advanced High-Performance Bus (AHB) protocol consists of several key signals and phases. The clock signal (HCLK) synchronizes the operations of the system, while the address phase involves transmitting address information on the HADDR [31:0] lines. In the data phase, data is either read from or written to the memory or peripheral on the HRDATA [31:0] lines. The write control signal (HWRITE) indicates whether the operation is a read or write, and the readiness signal (HREADY) shows whether the data phase can complete the current transfer. Transactions consist of an address phase followed by a data phase, with the HWRITE signal controlling the type of operation and the HREADY signal indicating the readiness of the bus to complete the transfer.

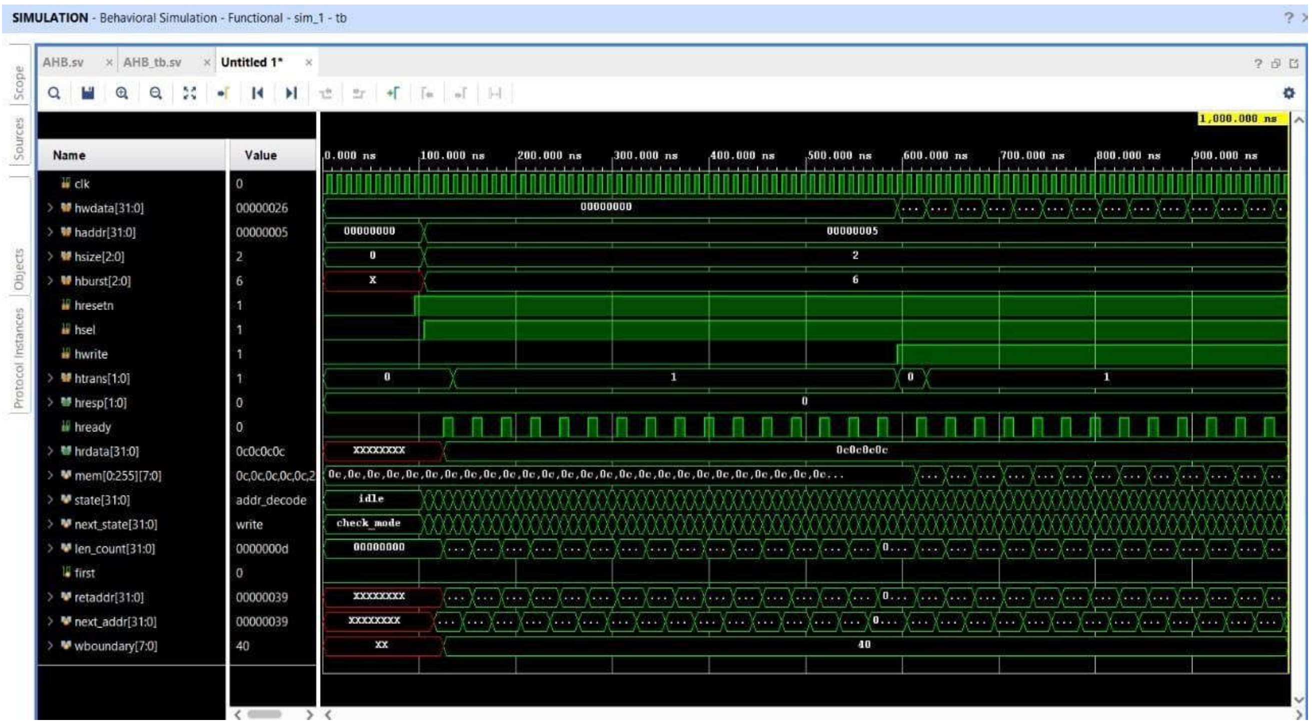
WAVEFORM

Figure 4: AHB Basic Transaction



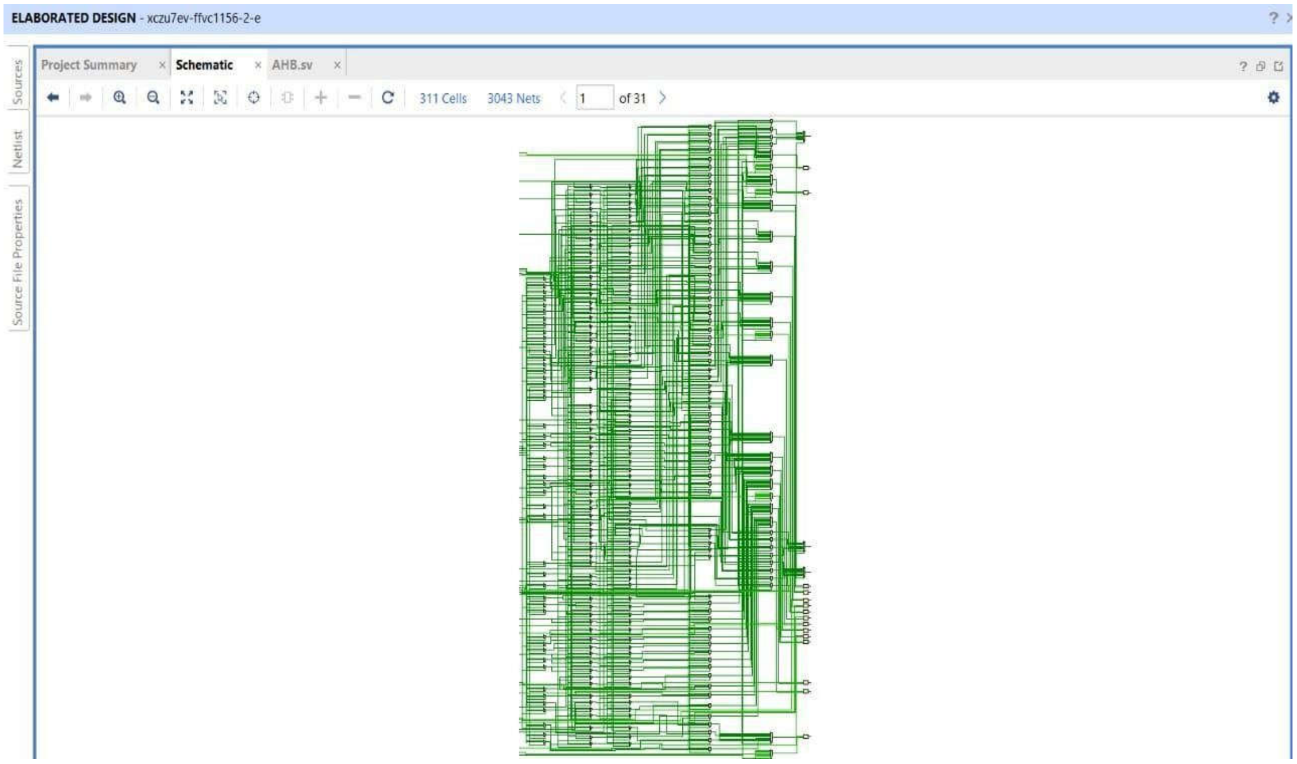
Name	Value
clk	0
hwdata[31:0]	00000026
haddr[31:0]	00000005
hsize[2:0]	2
hburst[2:0]	6
hresetn	1
hsel	1
hwrite	1
htrans[1:0]	1
hresp[1:0]	0
hready	0
hrdata[31:0]	0c0c0c0c
mem[0:255][7:0]	0c,0c,0c,0c,0c,2
state[31:0]	addr_decode
next_state[31:0]	write
len_count[31:0]	0000000d
first	0
retaddr[31:0]	00000039
next_addr[31:0]	00000039
wboundary[7:0]	40

Figure 5: FIXED Burst-type Transaction



Name	Value
clk	0
hwdata[31:0]	00000026
haddr[31:0]	00000005
hsize[2:0]	2
hburst[2:0]	6
hresetn	1
hsel	1
hwrite	1
htrans[1:0]	1
hresp[1:0]	0
hready	0
hrdata[31:0]	0c0c0c0c
mem[0:255][7:0]	0c,0c,0c,0c,0c,2
state[31:0]	addr_decode
next_state[31:0]	write
len_count[31:0]	0000000d
first	0
retaddr[31:0]	00000039
next_addr[31:0]	00000039
wboundary[7:0]	40

Figure 6: Schematic View



RESULTS

The verification environment for the AHB protocol is built using System Verilog modules like Interface, Driver, Generator, Transaction, Scoreboard, Assertions, and Monitor. In this setup, test cases are created in the Generator by specifying a test-case number. The Driver sends these test-case packets to the Design Under Verification (DUV). The DUV then processes different burst-type transactions (such as single, incrementing, and wrapping bursts) according to the AHB protocol.

The Monitor observes the outputs from the DUV and sends them to the Scoreboard. The Scoreboard compares these outputs with the reference (Golden Model) outputs. If the DUV output matches the Golden Model, the design is considered verified. All these steps are simulated using software like Questa, which generates waveforms to visualize the verification results.

In simple terms:

Test cases are generated and sent to the design. The design's outputs are checked against expected outputs. If they match, the design works correctly. This process is repeated for different types of data bursts using the AHB protocol, and the results are visualized using simulation tools. displayed in Figures 4 to 8

I. CONCLUSION

Proofing of AHB Bus Protocol for single grasp-single Slave for constant, INCR, and WRAP has been accomplished by means of designing the Verification IP using the usage of system

II. REFERENCES

[1] Harsha Garua, Keshav Sharma, Chusen Duari, "verification of AMBA AHB bus protocol implementing incr and wrap burst using system Verilog", International Journal of Research in Engineering and Technology, Volume-2 Issue 6, 2019.

[2] G. Kanaka Maha Lakshmi, M. Manasa Lakshmi, "AMBA-HB Protocol Verification by using System Verilog", International Research Journal of Engineering and Technology Volume-3 Issue-8, 2016.

[3] K. Lakshmi, M.M. Dasu, "Verification of the AHB20cp Bridge using System Verilog and effective bus Utilization calculation for AHB 3.0 Protocol", International Journal for Research & Development in Technology, Volume-6, Issue-1, 2016.

[4] Rashmi Samanth, Subramanya G. Nayak, "Design and SV Based Verification of AMBA AHB Protocol for SOC Integration", International Journal of Recent Technology and Engineering, Volume-8, Issue-2, 2019.

[5] Dr. Priyanka Choudhury, Perrumalla Giridhar "Design and Verification of AMBA AHB", IEEE International Conference on the Advanced Technology in Intelligent Control, Environment, Computing & Communication Engineering 2019.

[6] P.Harishankar, Mr.Chosen Duari Mr.Ajay Sharma,"Design and Synthesis of Efficient FSM for Master and Slave Interface in AMBA AHB", International Journal of Engineering Development and Research, Volume 2, Issue 3, 2014.

[7] Shivakumar B.R Deeksha L, "Efficient Design and Implementation of AMBA AHB Bus Protocol using Verilog", IEEE International Conference on Intelligent Sustainable System, 2019.

[8] Mr. M. Naresh Kumar, K.Manikanta Sai Kishore "Design and Implementation of Efficient FSM for AHB Master and Arbiter", International Journal and magazines of Engineering Technology, Management and Research. 2015.

[9] Shraddha divekar, Archana Tiwari "Multichannel AMBA AHB with Multiple Arbitration Technique", International Conference on Signal Processing and Communication, Apr 3-5, 2014.