International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

A Review on Design and Verification of Programmable UART with AXI

Dr Jagadeesh Basavaiah[1], Abhishek R Bharadwaj[2], Tanish Raj K S[3], Vinutha G[4],

[1]Associate Professor, Dept. of Electronics and Communication Engineering, Vidyavardhaka College of Engineering, Mysore, Karnataka, India
[2,3,4]Dept. of Electronics and Communication Engineering, Vidyavardhaka College of Engineering, Mysore, Karnataka, India

Abstract
The evolution of System-on-Chip (SoC) architectures has necessitated the integration of robust communication protocols to ensure efficient and reliable data transfer. Among these, the Universal Asynchronous Receiver/Transmitter (UART) plays a pivotal role in serial communication, particularly when interfaced with the Advanced Extensible Interface (AXI) for high-speed data transactions. This review paper delves into the design intricacies and verification challenges of programmable UARTs within SoC environments, leveraging the AXI protocol. We explore various design methodologies that cater to the low-latency and high-throughput demands of modern SoCs, examining the implementation of UART circuits based on the AMBA bus protocol and their alignment with AXI standards. Furthermore, the paper scrutinizes the verification processes, highlighting the use of SystemVerilog and Universal Verification Methodology (UVM) to ensure the reliability and performance of UART designs. This paper endeavours to offer a consolidated perspective on the present state-of-the-art in UART design and verification by conducting a thorough examination of existing literature and case studies. Additionally, it aims to shed light on potential future advancements and directions for improvement within this domain.

Keywords*:* AMBA, AXI-4, SoC, SystemVerilog, UART , UVM.

1. INTRODUCTION

The Universal Asynchronous Receiver/Transmitter (UART) is an integral component in the field of embedded systems, serving as a cornerstone for asynchronous serial communication. It operates independently of a clock signal, transmitting and receiving data sequentially, bit by bit. This simplicity and versatility make UART an ideal choice for a wide range of applications, from simple data logging to complex communication systems.

The Advanced Microcontroller Bus Architecture (AMBA) family and the Advanced Extensible Interface (AXI) collectively form a robust framework for the comprehensive testing and integration of intellectual properties (IPs) in system-on-chip (SoC) designs. AMBA, encompassing various IPs, including those from ARM and diverse suppliers, employs metric-driven protocol compliance verification. With sophisticated bridges, AMBA achieves high-frequency operation, making it well-suited for high-bandwidth, low-latency designs, ensuring exceptional performance. Together, the AMBA family and AXI provide a solid foundation for the verification and seamless integration of IPs, offering a comprehensive solution for

addressing the demands of complex SoC environments. The AXI interface comprises five channels: read data, read address, write data, write address, and write response.

The integration of UART with AXI within a System on Chip (SoC) brings together the best of both worlds: the ease of use and configurability of UART with the high-performance capabilities of AXI. This synergy is particularly advantageous in SoCs, where space constraints and power efficiency are paramount. By leveraging AXI's high bandwidth and low latency, the integrated UART can facilitate faster and more reliable data exchange, which is essential for real-time applications.

Moreover, the programmability of UART when combined with AXI's flexibility enables the SoC to cater to various communication standards and protocols, adapting to the specific needs of the application. This adaptability is crucial in today's rapidly evolving technological landscape, where devices must communicate across diverse platforms and standards.

The design of a programmable UART with AXI integration involves careful consideration of various factors, including data format, baud rate, error detection, and synchronization mechanisms. Ensuring compatibility with the AXI protocol requires meticulous planning and execution, from the initial stages of architectural design to the final stages of implementation. Verification of such a complex system is equally challenging. It necessitates a robust verification environment, often employing SystemVerilog and Universal Verification Methodology (UVM) to create a comprehensive testbench. The verification process covers all the possible scenarios, including corner cases and error conditions, to guarantee the reliability and performance of the UART-AXI module. The objectives of smart water quality monitoring system are:

1. To design a UART protocol with AXI which has minimizes latency and provides a robust error detection and handling mechanisms.
2. To design a high performance, versatile protocol which manages clock domains to ensure synchronous and reliable communication.

2. METHODOLOGY

2.1 AXI

Figure 1: AXI Channels

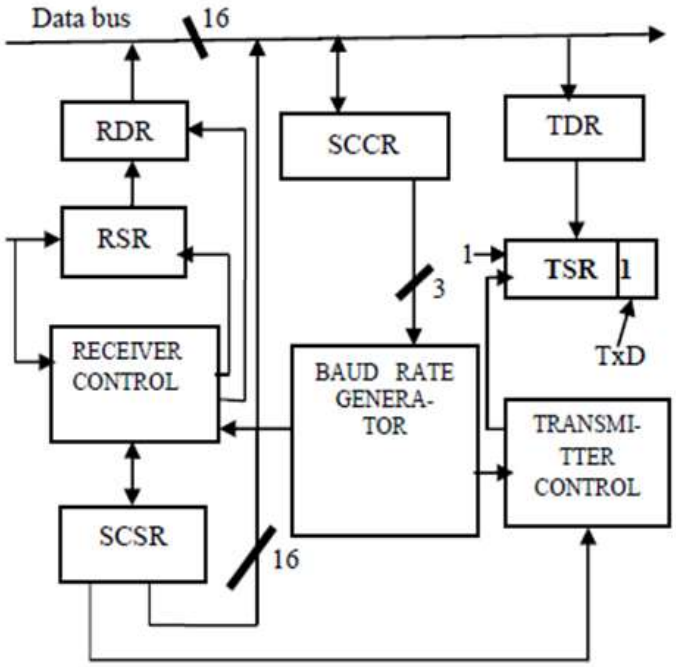
The primary communication channels in AXI4-lite include Read Address, Read Data, Write Address, Write Data, and Write Response. The interactions between the master and slave through each channel are outlined as follows:

- **Write Address Channel**: This communication channel enables the transfer of data, including the address targeted for the write operation to the slave. Upon receiving the address, the system proceeds with executing a series of write operations in bursts, followed by verification to ensure accuracy.
- **Write Data Channel**: After successfully completing the write address phase, the signal containing the data is transmitted to the previously accessed address within the slave device.
- **Write Response Channel**: After the successful completion of both the write address and write data operations, the slave device sends a response or acknowledgment back to the master.
- **Read Address Channel**: Through this channel, the master communicates the address to the slave, specifying the location from which data is to be read.
- **Read Data Channel**: Data is retrieved from the designated address and then transmitted to the master, simultaneously functioning as a flag indicating the completion of the read response.

2.2 AXI

As depicted in Figure 2, the UART system comprises three distinct kernel modules: the Transmitter, Receiver, and the baud rate generator. Each module is tasked with specific responsibilities, and a malfunction in any module can have a direct impact on the overall output of the UART. Additionally, the system incorporates six distinct registers, each serving a specialized purpose.

Figure 2: UART Block diagram



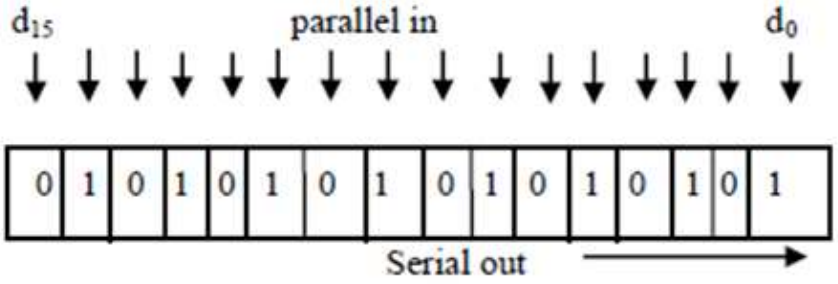
UART Registers

- **RSR (Receiver Shift Register)**: The RSR (Receiver Shift Register) is situated at the receiving end of UART. Its primary function is to receive bits sequentially from the RxD (Receive Data) pin and shift them to the right, accomplishing one bit per clock cycle. Figure 2 demonstrates the utilization of a serial-in-parallel-out shift register in the receiver, with d15 serving as the parallel output and d0 as the serial input.
- **RDR (Receiver Data Register):** Data received from the RSR is transferred to the RDR, and

subsequently, the data stored in the RDR is placed onto the data bus.

- **TDR (Transmit Data Register):** Bytes of data from the data bus are accepted by this register for transmission and then transferred to the TSR.
- **TSR (Transmit Shift Register):** Located on the transmitter side, the TSR is a shift register employed for transmitting data bit by bit, shifting each bit to the right. Figure 3 depicts the utilization of a parallel-in-serial-out shift register, where d15 serves as the parallel input and d0 as the serial output.

Figure 3: PISO



- **SCCR (Serial Communications Control Register)**: This register serves as the UART controller. The bottom 3 bits of this register are utilized for selecting the baud rate. The initial two bits, namely TIE (transmit interrupt enable) and RIE (receiver interrupt enable), function as interrupt signals and are set in SCCR whenever attention is needed by the UART receiver and transmitter.
- **SCSR (Serial Communications Status Register)**: This register provides information regarding the status or condition of the UART. It includes flags such as TDRE and RDRF, which indicate the status of various registers. The final two bits, OE (overrun error) and FE (framing error), are activated if any errors occur during transmission. An overrun error arises when the receiver is unable to process an incoming character before the subsequent one arrives. Conversely, a framing error arises when the designated "start" and "stop" bits are invalid. The "start bit" plays a crucial role in identifying the beginning of an incoming character and acts as a reference for the subsequent bits. If the data line does not maintain the expected idle state when the "stop" bit is anticipated, a framing error is triggered.

3. LITERATURE REVIEW

Yamini R and Ramya MV[1] focused on the design and verification of a full-duplex UART module using System Verilog (SV), a sequential clock signal communication system. The UART module is designed to convert parallel data to serial format for transmission and then convert it back to parallel format upon reception. The design process involves building baud rate generators, receivers, transmitters, interrupts, and FIFO modules. Verification is an important part of the process, which involves establishing a verification environment that allows test bench reuse and reduces code complexity Authors use randomization to test otherwise difficult corner conditions consider. The UART implementation is simulated using Questasim software, highlighting the effectiveness of the configuration and verification process. The paper highlights the advantages of using System Verilog over Verilog, noting that it adds more features and reduces code complexity. This research enhances the field by offering an intricate overview of the certification procedure for the UART architecture, showcasing System Verilog's capability in crafting a proficient and dependable communication protocol tailored for embedded systems. Chaithanya et al.[2] focused on interfacing the AMBA AXI4-Lite protocol with a tailored memory module This goal is accomplished by employing five distinct channels derived from the AXI4 protocol, each

playing a crucial role in overseeing efficient read and write transactions. This paper highlights the evolution of AMBA buses, with a focus on the AMBA4.0-AXI4 protocol and its variant AXI4-Lite, which is designed for simple memory peripherals. This protocol, a simplified iteration of the AXI protocol within the ARM-developed AMBA, is specifically crafted for systems requiring efficient communication with uncomplicated memory peripherals. The primary aim of this paper is to create an interface between an AXI4-Lite Master and a Customized Memory slave, facilitating efficient READ/WRITE transactions. The design strives to establish a simple and effective means for exchanging data between the master and slave peripherals. The design process is carried out using Verilog HDL, and the functionality is checked with the ISIM simulator. The design is then synthesized using the Xilinx ISE 14.4 tool. The ensuing simulation results are showcased, illustrating the successful execution of READ and WRITE operations through the integration of an AXI4-Lite Master and a customized Memory slave.

G. Sreenivasulu et al. [3], the focused on the design and validation of an APB Bridge adhering to the specifications of the AMBA 4.0. This APB Bridge functions as an Intellectual Property (IP) core, proficiently converting high-performance AXI4.0-lite transactions into low-power APB 4.0 transactions. Its primary role is to enable seamless communication among various components within a System-on-Chip (SoC). The paper introduced the concept of AMBA as an on-chip bus widely used in SOC designs, extending beyond microcontroller devices to various ASIC and SOC parts. The paper explained the various buses defined in the AMBA architecture, placing particular emphasis on the APB, designed for minimal power consumption. The architecture includes elements like the Direct Memory Access (DMA) operating as a Master, alongside Static Random Access Memory (SRAM) and UART functioning as Slaves. The paper described the state diagram and the block diagram for two scenarios: transferring data from UART to SRAM and from SRAM to UART. It also presents the verification architecture, results including waveforms for each step of data transfer, coverage report, and UVM report. The study concludes by highlighting the significance of the AMBA APB Bridge as a plug-and-play IP protocol released by ARM, supporting various functionalities such as data width, remapping, and configurable timing parameters. The design and verification of the APB Bridge were executed using Virology and Universal Verification Methodology (UVM) with the VCS tool.

S. Math et al. [4] conducted a study on implementing data transactions across a System-on-Chip (SoC) bus using the AMBA AXI4 protocol. The paper outlines the project's aim, which is to carry out data transactions on the SoC bus utilizing the AXI4 protocol implemented in hardware description language (HDL), with a particular focus on read and write operations. It highlights the features of the AMBA AXI4 protocol, an advancement from AMBA AXI3, including support for burst lengths of up to 256 beats, unaligned data transfers, and the capability to interface with 16 masters and 16 slaves. The paper also delves into the architecture and functionality of the AMBA AXI4 system, detailing its various supported channels. The study utilized the Verilog Compiler Simulator (VCS) tool for simulations, with the operating frequency configured to 100MHz. It included two test cases for conducting multiple read and write operations. Findings reveal that a single read operation requires 160ns, whereas a single write operation demands 565ns.These results underscore the efficiency and effectiveness of the AXI4 protocol in managing data transactions on an SoC bus, which is critical for the performance and reliability of modern electronic systems. In the conclusion and future prospects section, the paper discusses the constraints of the AMBA AXI4 system, including limitations on burst data and beats of information transfer. It also proposes directions for future research and enhancements in the protocol.

Samanth and Nayak [5] delved into the design and System Verilog (SV)-oriented verification of the

AMBA AXI protocol, a crucial component in the realm of high-performance and high-frequency system designs. The significance of the AMBA AXI protocol lies in its ability to tackle the challenges associated with low latency and high bandwidth, essential for the seamless integration of Systems on Chip (SoC). The paper suggests a comprehensive design and verification approach for the AXI-Bus and Memory interface, a pivotal aspect for SoC integration. The design, implemented using Verilog HDL, supports both single and burst-based data transfers, providing a versatile solution for a wide range of applications. The authors discuss the design challenges and present a design that does not require complex bridges, which are typically necessary for high-frequency operations. Furthermore, the paper emphasizes the importance of a robust verification process to ensure the SOC integration's reliability. The authors utilize SV-based verification methodologies, including SV assertions, to systematically verify the SOC. This approach simplifies checking all protocol specifications during the verification stage, making it more efficient.

Kalyan and Swaroop [6] examined the verification of the AMBA-AXI protocol using the Universal Verification Methodology (UVM). The AXI protocol, integrated into the ARM's AMBA specification, is extensively utilized in high-performance SoC designs for its capacity to operate at high frequencies and its minimal latency. The focus of the paper is on the verification process, which is critical for ensuring the reliability and correctness of the AXI protocol implementation. The authors utilize UVM, a standardized methodology for verifying integrated circuits, to create a robust verification environment. This environment allows for the systematic testing of the AXI protocol's features, including its multiple channels for write and read operations. UVM offers a framework for creating reusable verification components and utilizing sophisticated verification methods, including constrained-random stimulus generation and functional coverage. The paper demonstrates how UVM can be effectively used to verify the AXI protocol, ensuring that it meets the design specifications and operates correctly under various conditions. The verification process involves checking the signaling of the five channels of the AXI protocol: write address, write data, write response, read address, and read data. The results confirm the effectiveness of the UVM-based verification approach, showcasing the correctness of the AXI design and its suitability for high-performance SoC applications.

Gaikwad and N.Patil[7] discussed the verification of the AMBA AXI on-chip communication protocol, addressing the challenges of verifying on-chip bus protocols in SoC designs .The main emphasis is on the AXI protocol, which is well-suited for system designs requiring high bandwidth and low latency. The paper delineates the architecture of the AXI protocol, detailing its characteristics and transaction channels, and contrasts it with other AMBA protocols like APB and AHB. The paper highlights the need for a robust verification mechanism due to the increasing complexity of SoC designs and the use of standardized signal bus architecture. They emphasize the importance of verification in ensuring the synchronization and communication between IP cores within the SoC. The document explains the intricacies of the AXI protocol, including its support for burst write and read transactions, out of order transactions, parallel write and read transactions, and overlapped transactions. The verification environment for the AXI protocol is built using System Verilog, and the results of the verification analysis are presented. The verification is conducted for three cases, focusing on incrementing burst feature, fixing AWSIZE or AWLEN, and randomizing all values. The simulations are carried out using the Mentor Graphics QuestaSim EDA tool, and the waveforms are analyzed to demonstrate the successful write and read operations for the AXI protocol. The paper is a valuable resource for VLSI engineers, providing insights into the complexities of on-chip communication protocols and the methodologies for their verification.

Tidala[8] discussed the implementation of a high-performance Network on Chip (NOC) using the AXI4

protocol interface on a Field-Programmable Gate Array (FPGA). The objective of the study is to grasp on-chip communication using the enhanced iteration of the AMBA AXI protocol. The paper delineates the challenges faced and offers an experimental examination of the complexities entailed in crafting on-chip communication utilizing the AXI4 Protocol. The selection of the AXI protocol is based on its advanced attributes, such as straightforward interface registration suitable for applications featuring multiple masters. The paper highlights the crucial role of on-chip communication in real-time applications and its significance in the embedded field. It emphasizes the need for efficient quality of service, high data transmission, low latency, and low power consumption. A comprehensive examination of the AXI4 protocol is presented, emphasizing its advantages, distinct data phases, address control, backing for unaligned data transfers, transactions with burst lengths, and extensions that encompass signaling for low power consumption. The interface and interconnect structure, as well as the revisions history of the AXI protocol, are also discussed. The use case of real-time route analysis in programmable logic to double data rate synchronous dynamic random-access memory (PL to DDR) is presented to demonstrate the practical application of the AXI interface. The document delves into the details of the AXI interface between master and slave, data transmission, addressing channels, clocking, and reset sequences. It explains how the AXI protocol facilitates burst-based data transfers and provides flexible addressing and data transfer mechanisms. The experimental and modeling results of the AXI interface on NOC using Verilog are presented, showing the performance and bandwidth observed for different burst lengths.

Mahesh and Sakthivel.S.M[9] discussed the verification of the AMBA AXI Bus protocol using SV and a coverage-driven verification methodology. It emphasizes the importance of verifying the synchronization between intellectual property cores in System on Chip (SOC) designs. The paper highlights the significance of verification IP, as it covers 70% of the time in SOC development. The AMBA AXI Bus architecture is detailed, focusing on the read and write transaction phases. For the read transaction phase, the interaction between the master and slave interfaces is elucidated, including the role of read address and read data channels. Similarly, the write transaction phase is explained, encompassing address, data, and response channels. The verification environment is outlined, comprising a Generator, Bus Functional Model (BFM), Mailbox, Monitor, AXI Interface, and AXI Master/Slave. This environment facilitates the development of test cases and allows for the verification of the Design Under Verification (DUV).The verification plan delineates the properties to be verified, such as system connectivity, transaction routing, and data integrity. The verification process employs a coverage-driven methodology to attain full effectiveness. In the results and discussion section, verification of write and read architectures is addressed, including the assessment of parameters like valid count, busy count, and bus utilization. Utilizing code coverage mode analysis ensures the functionality verification of the design, reaching an 80% coverage level.

Chen et.al [10] discussed the complexity of SoC design due to the integration of various functions and Intellectual Properties (IPs) within a chip. The primary obstacle lies in validating on-chip communication properties, where traditional simulation-based methods fall short for chip-level dynamic verification and hardware debugging. To tackle this challenge, the paper proposes a rule-based synthesizable AMBA AXI protocol checker comprising 44 rules to ensure the precision of on-chip communication properties. The verification approach entails utilizing Synopsys VIP for validating the AXI protocol checker. Experimental findings reveal that the chip cost of the AXI protocol checker amounts to 70.7K gate counts, with a critical path of 4.13 ns using TSMC 0.18um CMOS 1P6M Technology. The paper also compares the proposed AXI protocol checker with existing solutions, highlighting its synthesizability, debugging

support, and the number of rules. The proposed AXI protocol checker architecture comprises a protocol checker, configuration registers, and an error reference table (ERT). The protocol checker is rule-based and consists of sub-modules to check different aspects of the AXI protocol. Configuration registers allow setting parameters and enabling/disabling the checker. The ERT summarizes total errors and provides a GUI software analyzer to display error information in detail. The verification approach involves RTL trigger-based generation of bus signals and employing Synopsys VIP to validate the AXI protocol checker. Synthesis outcomes reveal a maximum frequency of 242 MHz and a total area of 70.7K gate counts for the AXI protocol checker. In conclusion, the proposed AMBA AXI protocol checker with a rule-based verification mechanism and synthesizability is positioned to enhance verification ability and debugging efficiency for on-chip bus communication properties. It provides a detailed insight into the architecture and verification strategy of the AXI protocol checker, highlighting its potential for improving verification ability and reducing debugging time in SoC designs.

Krishna [11] explored the design and creation of an AMBA-AXI4Bus Based System on Chip VIP Protocol using Universal Verification Methodology (UVM). The AXI protocol, integrated into the AMBA framework, is specifically designed for high-performance, high-frequency system designs and can seamlessly handle memory controllers with increased initial access latency. Its versatility in interconnect architectures and its ability to maintain compatibility with existing AHB and APB interfaces are emphasized. The paper delineates the features and architecture of the AXI protocol, which include separate address and control/data phases, support for unaligned data transfers, burst-based transactions, and the capability to manage multiple outstanding addresses. The AXI4 slave interface design is explicated, with implementation details using Verilog RTL coding. Additionally, the paper proposes a design environment for constructing the verification setup of the AXI bus through UVM. Various verification elements, including functional coverage, code coverage, scoreboarding, and assertions, are integrated into the verification environment. Simulation results and UVM-based verification are presented, demonstrating the successful validation of the AXI protocol, encompassing write and read transactions for diverse burst types and sizes, multiple outstanding addresses transactions, and out-of-order transaction completion. Test cases are outlined to verify the working of all five channels, and the paper concludes with the successful development and verification of the VIP for the AMBA AXI-4 protocol. In summary, the document provides detailed insights into the design and development of the AMBA-AXI4Bus Based System on Chip VIP Protocol using UVM, highlighting its features, architecture, design, simulation results, UVM-based verification, and successful test cases.

Table 1: Comparison of reviewed methodologies

Sl no.	Title	Methodology	Applications
1	Design and Verification of UART using System Verilog	System Verilog	Embedded and IoT applications
2	Design of AMBA AXI4-Lite for Effective Read/Write Transactions with a Customized Memory	Verilog HDL	Video streaming
3	Design and Verification of APB Bridge Based on AMBA 4.0	Verilog	Creating multi-processor designs involving numerous controllers and peripherals.
4	Data Transactions on System-on-Chip Bus Using AXI4 Protocol	Verilog HDL	Utilized to decrease pre-charge time employing DSAS.
5	AMBA-AXI Protocol Verification	Verification	Used for complex designs

	by Using UVM	Intellectual Property Cores (VIP)	
6	Verification of AMBA AXI on-chip Communication Property	System Verilog	High bandwidth applications
7	Design and SV-Based Verification of AMBA AXI Protocol for SOC Integration	System Verilog	Developing fully coverage driven verification IPs
8	High-Performance Network on Chip using AXI4 protocol interface on an FPGA	Verilog	High performing networking application
9	Verification IP for an AMBA-AXI Protocol using System Verilog	System Verilog	Used for complex design
10	A Synthesizable AXI Protocol Checker for SoC Integration	Verification Intellectual Property Cores (VIP)	Enhanced security and cache hint assistance.
11	Design and Development of AMBA-AXI4 Bus-Based System on Chip VIP Protocol Using UVM	Verification Intellectual Property Cores (VIP)	Used to connect different peripherals like GPIO, SPI, I2C, UART.

4. CONCLUSION

The paper navigates through the intricacies of design, emphasizing the importance of programmability and the seamless integration of the AXI protocol. The thorough verification methodologies discussed in the paper contribute to the reliability and correctness of the UART, ensuring its compatibility with diverse applications. The AXI integration not only enhances data transfer efficiency but also aligns the design with contemporary
Industry standard.

REFERENCES

1. Yamini, R., Student, E. M., & Ramya, M. (2020). Design and Verification of UART using System Verilog. International Journal of Engineering and Advanced Technology, 9(5), 1208–1211. https://doi.org/10.35940/ijeat.e1135.069520.
2. A. Sainath Chaithanya, Sameera Sulthana, B. Yamuna and Ch Haritha, “Design of AMBA AXI4-Lite for Effective Read/Write Transactions with a Customized Memory", International Journal on Emerging Technologies 11(1): 396-402(2020).
3. G.Sreenivasulu , M.Santosh Krishna , “Design and Verification of APB Bridge Based on AMBA 4.0.” (2015). International Journal of Scientific Engineering and Technology Research, 04(57), ISSN 2319-8885.
4. S. S. Math, R. B. Manjula, S. S. Manvi and P. Kaunds, "Data transactions on system-on-chip bus using AXI4 protocol," 2011 INTERNATIONAL CONFERENCE ON RECENT ADVANCEMENTS IN ELECTRICAL, ELECTRONICS AND CONTROL ENGINEERING, Sivakasi, India, 2011, pp. 423-427, doi: 10.1109/ICONRAEeCE.2011.6129797.
5. Samanth, Rashmi & Nayak, G.Subramanya. (2019). Design and SV Based Verification of AMBA AXI Protocol for SOC Integration. International Journal of Recent Technology and Engineering (IJRTE). 8. 1465-1469. 10.35940/ijrte.B2110.078219.
6. P. Naveen Kalyan and K. Jaya Swaroop, “AMBA-AXI Protocol Verification by Using UVM”, International Journal of Electronics and Communication Engineering and Technology (IJECET),

Volume 7, Issue4, July-August 2016.

7. N. Gaikwad and V. N. Patil, "Verification of AMBA AXI On-Chip Communication Protocol," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), Pune, India, 2018, pp. 1-5, doi: 10.1109/ICCUBEA.2018.8697587.
8. N. Tidala, "High Performance Network on Chip using AXI4 protocol interface on an FPGA," 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2018, pp. 1647-1651, doi: 10.1109/ICECA.2018.8474664.
9. Mahesh, G., & Sakthivel, S. M. (Year). "Verification IP for an AMBA-AXI Protocol using System Verilog." International Journal of Engineering Research and General Science Volume 3, Issue 1, January-February, 2015 , ISSN 2091-2730.
10. C. H. Chen, J. -C. Ju and I. -J. Huang, "A synthesizable AXI protocol checker for SoC integration," 2010 International SoC Design Conference, Incheon, Korea (South), 2010, pp. 103-106, doi: 10.1109/SOCDC.2010.5682961.
11. Narra Venkata Krishna, “Design and Development of AMBA-AXI4Bus Based System on Chip VIP Protocol Using UVM", International Journal of Innovative Research in Science, Engineering and Technology. 6, Issue 4, April 2017.