

# Design and Verification of a 10G Ethernet MAC Layer using Universal Verification Methodology

*Sambhrama Managuli*

*M.Tech VLSI Design & Embedded Systems, School of ECE,, REVA University, Bangalore, India,*

[sambhramamanaguli@gmail.com](mailto:sambhramamanaguli@gmail.com)

*Dilna U*

*Assistant Professor, School of ECE, REVA University, Bangalore, India, [dilna.u@reva.edu.in](mailto:dilna.u@reva.edu.in)*

## Abstract

The growing necessity for high-speed data communication has led to the importance of reliable Ethernet MAC implementations in contemporary SoC and FPGA designs. In this paper, the design and verification of 10 Gigabit Ethernet MAC layer with AXI-Stream interface and System-Verilog-based UVM (Universal Verification Methodology) have been presented. The proposed design of the MAC includes AXI-Stream interface, AXI-to-FIFO conversion block, FIFO-based buffering, and transmit engine which works with Ethernet frames. The AXI-Stream interface represents a convenient and flexible approach to data packet transfer from user logic to the MAC, while FIFO isolates data input flow from the transmission procedure. The verification environment based on the use of UVM is created to provide stimulus generation, transaction monitoring, checking of the data sent, and calculating of functional coverage. Both the standard transmission and fault-injection modes are taken into consideration. Simulation results confirm proper functioning of packets sending and AXI-Stream handshakes in the case of normal operation. The created environment was able to cover the Ethernet functionality in the simulation process, and tests with error injections confirmed the possibility of identifying incorrect transmission cases using this verification environment. The proposed methodology allows us to create a base for verification of Ethernet MAC functionality.

**Keywords:** Ethernet MAC, IEEE 802.3, System-Verilog, Universal Verification Methodology (UVM), AXI, Functional Verification.

---

## 1. INTRODUCTION

Ethernet has emerged to be one of the most common technologies that are used in digital communications due to its flexibility, compatibility, and standard IEEE 802.3 specifications. With increased interest in high-speed data rates in communication systems, Ethernet MACs are becoming more common in FPGAs and SoCs. In these high-speed environments, testing of the design is equally important as the design itself since even the slightest mistake in handling data or protocols can lead to packet corruption or data loss. A 10-Gigabit Ethernet MAC takes care of Media Access Control related functionality between the system-side logic and the physical transmission interface. The MAC takes care of the processing of Ethernet frames and control of the associated tasks such as formatting of frames, transmission of frames, and other related protocol operations. In the case of an implementation in an FPGA or SoC, there should be an interface that allows packet data to be provided to the MAC from the surrounding system. In this project, the AXI-Stream protocol is chosen to accomplish this task since it is meant for streaming data at high speeds and provides simple valid ready based flow control. The implementation uses TDATA, TVALID, TREADY, TKEEP, and TLAST to transfer packet data and identify valid bytes and frame boundaries.

The MAC architecture that is suggested in this paper is based on the modular approach. AXI-Stream signals come to the interface and get translated to the internal representation through an AXI-to-FIFO bridge. The FIFO serves as the temporary storage of the input data and creates the isolation between the input data and the transmit logic. Then the transmit engine gets the buffered data and makes it ready for Ethernet transmission. The isolation of these processes allows improving the understandability, testing and reusability of the design. The architecture discussed in the original paper includes AXI-Stream interface, protocol translation block, FIFO and transmit engine.

In this paper, we describe the design and implementation of the Ethernet MAC Core along with the AXI Stream interface. The primary purpose of such a design is to allow for a smooth data transfer between the user logic and the Ethernet Physical layer by strictly following the AXI handshaking protocol. The design has a modular approach, which allows each individual block to operate separately but collectively perform data transfer operation. The design of the system proposed is represented in the block diagram and comprises of four major components: the AXI-Stream interface, the Protocol Conversion Unit, FIFO Buffer, and the Transmit engine. The AXI-Stream Interface is the entry point for the data, making use of signals like TVALID, TREADY, TDATA, and TLAST for performing the handshaking and managing the data. The purpose of the Interface is to perform the data transmission only when both the receiver and the sender are ready. The verification environment created for this purpose relies on the usage of UVM sequences in order to create Ethernet transactions and a driver to apply them to the AXI-Stream interface. A monitor is used to track the activities of the device under test, while a scoreboard serves as an independent method of verifying the correctness of the behaviour. Functional coverage is applied in order to determine whether any particular scenario has been covered during the verification.

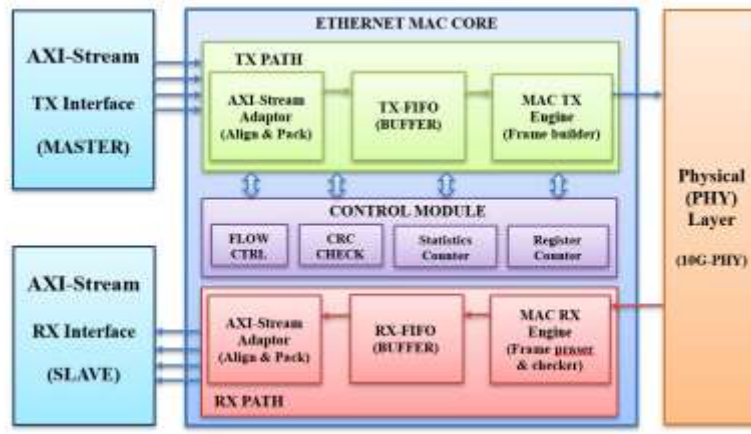


Fig 1: Ethernet MAC layer architecture.

The initial implementation specifically mentioned coverage of both normal and faulty transaction cases as well as the application of the error injection approach in order to check the robustness of the design. Thus, the primary goal of this work was to design a 10G Ethernet MAC architecture with AXI-Stream interfaces and verify its behaviour by means of a reusable UVM-based environment. This work focuses on reliable packet transmission, AXI handshaking, buffering, frame handling, functional coverage, and fault-oriented verification. The simulation results show the functional coverage have been achieved.

## 2. RELATED WORK

The verification of Ethernet MAC designs has been considered using a variety of techniques, such as directed simulation, System-Verilog verification, UVM, and various system-side interfaces. There have been previous studies that have highlighted the potential of coverage-driven verification in finding out untested functional scenarios and gaining confidence in the MAC implementation. There have been a number of research works on the verification of Ethernet MAC using System-Verilog and UVM. The use of System-Verilog and UVM for verification usually involves partitioning of the verification environment into a variety of reusable modules, such as sequence, driver, monitor, scoreboard, and coverage model.

The IEEE 802.3x pause frame is a 64-byte control frame utilized in full-duplex Ethernet communications in order to implement flow control. The pause frame has a reserved multicast destination address, MAC source address, Ethernet type, opcode indicating the pause operation, a pause timer that specifies how long the data transmission will be paused, and extra padding to ensure the minimal size of Ethernet frame. There is also a Cyclic Redundancy Check (CRC) that is utilized for error checking [7]. The flow control method used above can be helpful in decreasing packet loss during the time of congestion by pausing frame transmission for some time. Therefore, it increases bandwidth utilization, ensures reliable data delivery, and prevents buffer overflow.

There have been studies carried out that deal with Ethernet controllers' integration through interfaces like RGMII and Wishbone. The paper's bibliography contains research on UVM-based verification of an Ethernet MAC using a Wishbone interface. This can be used as a point of comparison between the approach of using AXI based interface in this work.

In this study, AXI-Stream is used as the interface on the system side. Unlike address-based transaction mechanisms, AXI-Stream supports a continuous data flow via TVALID-TREADY handshaking protocol. This is due to the nature of the interface where data transfer is inherently stream-based. The work proposed in this study utilizes AXI-Stream interface in conjunction with FIFO buffers, a transmit engine and UVM based verification. Additionally, functional coverage and error injection verification are added to the verification environment. This is used to evaluate the normal operation mode and some abnormal modes.

### 2.1. 10G Ethernet MAC Data Transmission:

The 10G Ethernet MAC is designed using a modular hardware design approach. The main data path is composed of the AXI-Stream interface, AXI to FIFO converter logic, FIFO buffer and the transmitter block. The respective components do their job and are clearly separated from the neighboring component through a well-defined interface. The AXI-Stream interface is responsible for input of the packet data. TDATA [63:0] is the data bus with the TVALID and TREADY signals being used to determine the availability of the transfer. TKEEP is responsible for the identification of the valid byte lanes of the present data word whereas TLAST determines the end of an Ethernet frame. The transfer happens only if both TVALID and TREADY are active. The AXI to FIFO block is responsible for translating the AXI transaction into a form suitable for storing into FIFO buffer. The validation process for the AXI-based 10G Ethernet MAC architecture will be done in a structured simulation-based methodology that will confirm the functionality, protocol compliance, and the accuracy of the data. First, the simulation environment is configured through the initialization of the environment and the forcing of all design signals to certain values. Afterward, the verification of the design through the UVM verification environment will begin with proper connections established between the virtual interfaces and the testbenches elements which are used to build the design under test.

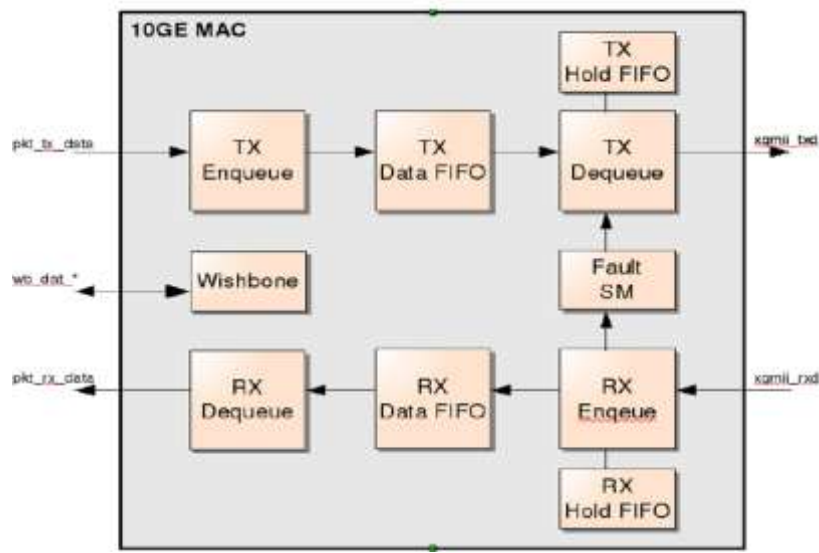


Fig 2.1 Ethernet Data Transmission flow.

Once the initialization phase completes, a synchronous reset signal is applied so that all of the blocks go to a definite idle state. Then, a stable clock that meets the timing constraints for the 10G Ethernet communication is established. This process makes sure that each of the components such as the FIFO memory, TX unit, and MAC operates starting from a predetermined point which prevents any sort of unexpected behavior due to inappropriate initialization. After reaching stability, frame transmission is carried out through AXI-Stream interface. The test bench generates frames and then applies those to DUT through usual AXI signals like tvalid, tdata, tkeep and tlast. These signals represent frame validity, frame data, byte qualification, frame end, respectively. Through the generation of different traffic conditions and packets, this phrase replicates the real network behavior and verifies the functionality of the Ethernet MAC [7].

The frames are transmitted further through the TX engine where the AXI handshaking protocol and the process of preparation of the received data stream for its further transmission is managed. The synchronization between the signals TVALID and TREADY is performed, proper data alignment is achieved and control information related to these frames is passed along the processing stage. In such way the data stream in the AXI format is transformed into the suitable format for further transmission processing. The processed data is transferred next to the FIFO buffer which serves as temporary storage between different functional modules. The FIFO buffer which serves as temporary storage between different functional modules. The FIFO buffer acts as an intermediary to keep the incoming data for some time. It helps compensate for the difference in the rates of data input and data output by the transmitter engine. Additionally, it helps maintain the order of the stored data. The transmitter engine gets the data from the FIFO buffer and is responsible for the transmission process. It handles frame handling and creation of necessary information on the transmitting side. Moreover, it provides the possibility of efficient communication between different modules working at different speeds.

## 2.2. AXI-Stream Protocol in Ethernet MAC Core:

AXI-Stream is chosen as the main interface used for data transfer between the system-side logic and the Ethernet MAC. In contrast to the memory mapped AXI interface, AXI Stream does not need the address of every transfer. AXI stream is thus more appropriate for continuous packets data. According to the proposed design, TDATA [63:0] is used for transferring 64 bits of the packet data on each transfer. TVALID means that the source has valid data to transfer, whereas TREADY means that the destination is ready for transfer. The transfer occurs if both signals are high during the active clock edge. TKEEP [7:0] is used for identifying valid bytes in the 64-bit word, which is very helpful in the last beat of the packet. TLAST is the indication of the end of an Ethernet frame. It helps not to advance transactions without receiver acceptance. Thus, the AXI stream in the proposed MAC provides the interface between packet generator and the buffer. TVALID-TREADY handshake allows for smooth flow control, thus preventing data corruption and data loss during the process. The information contained in the payload is conveyed using TDATA, and TKEEP signals which byte lanes are active in the data bus, while TLAST indicates that the frame has ended. This interface makes it possible to transfer Ethernet packets in an effective manner into the MAC layer and facilitates interaction with the TX processing bit and FIFO buffers [4]. Using AXI-Stream will enhance the throughput capability, design flexibility, and scalability of the system which makes it appropriate for high-speed networking systems such as 10 Gigabit Ethernet.



Fig 2.2: AXI Protocol with Ethernet MAC Core.

In addition, using AXI-Stream protocol increases the modularity and scalability of the system due to the decoupling of data producers from consumers through flow control. It is possible for the TX engine, FIFO buffer, and MAC core to function independently and concurrently yet achieve uninterrupted data transfer. In this light, the use of AXI-Stream protocol in this project is highly efficient.

### 2.3. Ethernet Data Frame Format:

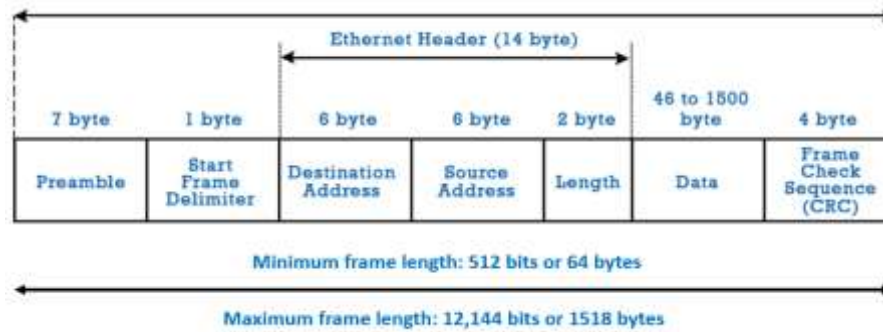


Fig 2.3: Ethernet Data Frame Format.

MAC functions with Ethernet frames based on the IEEE 802.3 protocol. The traditional Ethernet frame includes preamble, Start Frame Delimiter (SFD), destination MAC address, source MAC address, Length/Type field, payload, and Frame Check Sequence (FCS). Standard Ethernet frame size usually varies from 64 to 1518 bytes without including the preamble and SFD. VLAN-tagged frames may have 1522 bytes of length. Preamble and SFD help to achieve synchronization and show where the frame starts, while MAC addresses show who the sender and receiver of this communication are. Payload consists of the information to be transmitted and FCS is responsible for detecting errors.

## 3. METHODOLOGY

The proposed methodology implements a modular UVM based verification environment for verifying the 10G Ethernet MAC layer which primarily concentrates on data transfer and fundamental operations of Ethernet. The verification environment comprises a top level testbench, uvm\_env, an agent, a scoreboard, and functional coverage block in association with the design under test (DUT). The Ethernet MAC core verification is carried out based on the principle of stimulus and response.

### 3.1. Ethernet Transmission and Validation using UVM:

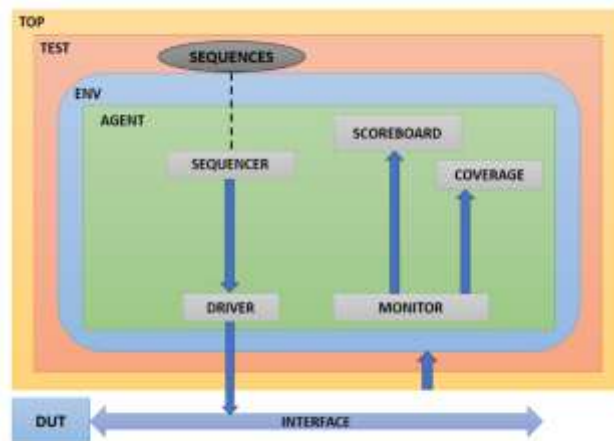


Fig 3.1 UVM based verification architecture for Ethernet.

**SEQUENCE ITEM** – the sequence item symbolizes the transactions passed from the sequence to the driver in the verification environment. It consists of frame parameters like payload data, control fields, and CRC error insertion controls (if applicable). The parameters are randomized for checking not only normal operation scenarios but also error conditions without going into details of signal implementation.

**SEQUENCE** – the sequence generates transaction-level frame objects with customizable parameters including payload data, sequence parameters, and CRC fault injection controls (if any). Constrained randomization is used for stimulating all kinds of packets for testing purposes.

**SEQUENCER** – the role of the sequencer is to act as an interface between the sequence and the driver in the UVM framework. It manages the transport of sequence items through synchronization, arbitration, and handshake processes before passing the transactions to the driver.

**DRIVER** – the driver converts transactions into signal inputs and feeds them to the DUT. It has the capability to perform intentional CRC for the verification of the DUT error detection capabilities.

**MONITOR** – the monitor captures all the output activities of the DUT. It includes processed data frames as well as TREADY/TVALID responses. The captured transactions are sent to the scoreboard and functional coverage blocks for verification purposes.

**UVM AGENT** – the UVM agent interfaces the verification environment and the DUT. It comprises the sequencer, driver and monitor in the uvm\_env.

**SCOREBOARD** – The scoreboard acts as a reference model in the verification environment. It recalculates the CRC values, monitor the sequence progress, and verify the output of the DUT. Any discrepancy leads to functional errors.

**ENV** – The uvm\_env is responsible for structuring the entire verification components, improving reusability.

**TEST** – The uvm\_test sets up the verification process and configures test scenarios, enables fault injections, and initiates the stimulation with constraints.

**DUT** – The DUT is the Ethernet Data Link Layer module which executes the CRC Generation/Verification, Sequence Monitoring, and TREADY/TVALID generation depending on Frame Validation.

**INTERFACE** – The System Verilog Interface is used for signal communication between the UVM Testbench and DUT. It is comprised of all required I/O signals so that the driver can apply the stimulus and monitor can observe the response from the DUT.

**TOP MODULE (TB\_TOP)** – The top module is used to connect the DUT and UVM testbench by instantiating the interface and connecting the interface with DUT.

### 3.2. Stimulus Generation and Error Injection Strategy:

There are two kinds of stimuli which are verified. These include good and bad transactions. In normal operation mode, valid Ethernet data is produced and transmitted via the AXI-Stream interface. Different data and different packet conditions are utilized to make sure that the MAC is not depending upon only one set of static vectors. The AXI handshake is also monitored in the process. When the fault-oriented testing takes place, then some kind of invalid situation is deliberately placed in the way of the communication. This is done in order to check whether the MAC and verification environment can differentiate between good and bad traffic

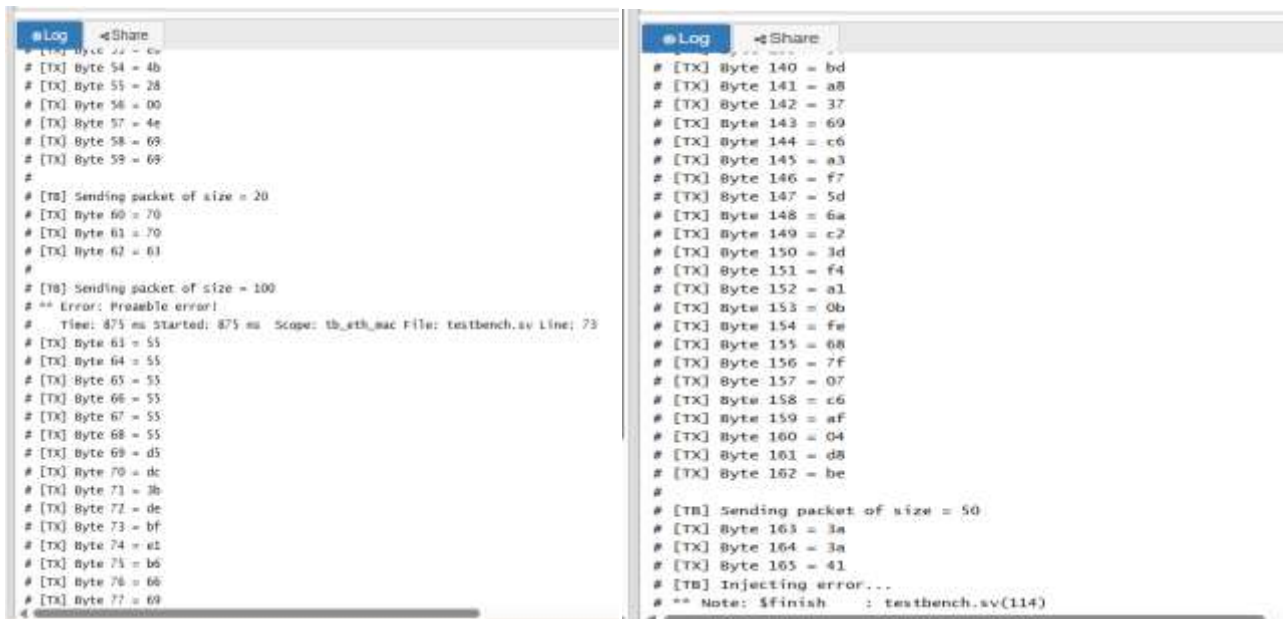


Fig 3.2: Output transmitted data in Ethernet MAC core layer with error injected.

Under normal conditions, it checks proper sequence progression and successful packet processing without CRC faults. During error injection tests, it confirms that corrupted packets are detected correctly and appropriate error responses are generated. The verification mainly focuses on CRC accuracy and sequence tracking in both normal and faulty conditions.

## 4. RESULT AND DISCUSSION

The architecture for Ethernet MAC and UVM verification environment has been tested using System-Verilog based simulation tools. The simulation results demonstrate the successful compilation and testing of the design and UVM verification environment without any UVM error or any fatal errors in the current run. In normal operation, several Ethernet data words were created and transmitted via AXI-Stream interface. It can be seen from the waveforms that the design successfully accepted the packet data and transmitted it through the internal buffer. It is also possible to see from the results that the verification environment is able to create different data value rather than using only one packet. This is important as repeating the same data provides no guarantee of correct operation of a high-speed data path. Also, fault injection was used in order to check the case of invalid data transmission. In the reported test, an

invalid preamble has been introduced and it has been detected by the receiver module.



Fig 4: Waveform showing transmission between the 10G Ethernet and FIFO.

## 5. CONCLUSION

The simulation results reveal successful normal data transmissions as well as prove proper functioning of the AXI-Stream interface and MAC data processing pipeline. Abnormal operations have been also evaluated through fault injection, where the invalid-preamble experiment has revealed a protocol breach. The achieved functional coverage of the verification environment has been reported. This design serves as a basis for extending the scope of verification. The future scope of development can include increasing the coverage through constrained-random packet generation, further AXI back-pressure cases, FIFO corner cases, various Ethernet frame sizes, CRC/FCS failure cases, and extended receive path coverage. The proposed modifications will bring the project closer to covering other parts of the Ethernet MAC protocol

## REFERENCES

- [1] B. G. Nelam, P. Sharmila, D. P. S. Praveenya, N. Vedajhna, S. Nagalashmi, G. Gayathri, M. Durga, "Coverage-Driven Verification of Ethernet MAC Using System Verilog and UVM," *IEEE Conference on VLSI (INDISCON)*, 2025.
- [2] Xiaolan Zhou, Zhonghua Zhao, Ziang Zhou, Shanxin Peng, "SoC Integration and Verification of a Gigabit Etheret Controller with RGMII," *IEEE Conference on AI, IoT and Cloud Computing Technology*, 2025.
- [3] M.L.N. Charyulu, N. Mani Kanth, Vivek Kushwah, "UVM Based Verification of an Ethernet MAC using Wishbone Bus" *IEEE on Communication Systems and Network Technologies*, 2024.
- [4] Jesus Lazaro, Armando Astarloa, Aitzol Zuloaga, Jose Angel, Jaime Jimenez, "AXI Lite Reduntant On-Chip Bus Interconnect for High Reliability Systems," *IEEE Transactions on eliability*, Vol 73, No 1, 2024.
- [5] K. R. Kumari, Avinash Surabi, Surya. G., Pavan Reddy, "Design and Verification of Ethernet Protocol," *IEEE International Conference Computing Communication and Networking Technologies*, 2023.
- [6] Kundoor, Meghana Reddy, "Design and Implementation of a 10 Gigabit Ethernet XAUI test Systems," Master's Theses and Capstones. 232, 2006.
- [7] A. Tanguay, "10GE MAC Core Specification," OpenCores.Org, 2008.
- [8] Perumal B, Nagaraj P, Kuppala, Gopavaram, Tamballa, Annampelly, "Development of an Ubdersized Packet Testcase for 10GB MAC CORE Verification," *IEEE on Smart Electronics and Communication*, 2023.
- [9] Allu Jnana Prasuna, D. P. Raju, "Design and Development of Gigabit Ethernet Verification IP(VIP) Using System Verilog." *International Journal of scientific Engineering and Technology Research*, 2017.

## BIOGRAPHIES



**Sambhrama Managuli:** Received the bachelor's degree in Electronics and Communication from B.L.D.E.A'S College of Engineering and Technology, Vijayapura in 2024, and the Master's degree in VLSI Design and Embedded System from Reva University, Bangalore in 2026, respectively.



**Dilna U:** Assistant Professor, School of Electronics and Communication Engineering at REVA University, Bengaluru, India. Graduation is from Cochin University of Science and Technology, Kerala and Post-Graduation from VTU Belagavi, PhD from REVA University. Total 14 years of teaching experience in Engineering Field. My areas of Interest are VLSI Design and Embedded Systems.