International Journal of Computer Applications (0975 - 8887)
Volume 186 - No.62, January 2025

Design and Functional Verification of a 1x4 Switch for Packet-Based Data Transmission

Aravalli Sainath Chaithanya
Assistant Professor
Department of Electronics and Communication Engineering
Rajiv Gandhi University of Knowledge Technologies,
Basar - 504107, Telangana, India

Myadari Radhika
M.Tech
Department of Electronics and Communication Engineering
Rajiv Gandhi University of Knowledge Technologies,
Basar - 504107, Telangana, India

ABSTRACT

This work presents the design and functional verification of a 1x4 switch, a key component in packet-based communication protocols operating at the network layer of the TCP/IP model. The switch facilitates intelligent routing of data packets from a single input to multiple outputs, ensuring efficient and reliable communication. Addressing the growing need for robust verification in modern ASIC design—where verification consumes 60% of the design cycle and 90% of chip failures result from inadequate verification—this study develops a System Verilog-based verification environment. The design incorporates finite state machines (FSMs), FIFOs, and memory modules, with extensive simulation across diverse scenarios to validate functionality. State-of-the-art EDA tools, including Xilinx ISE 14.7 and Synopsys VCS 2021.09, are utilized for design synthesis and verification. This approach achieves comprehensive coverage, enhances reliability, and ensures reusability, making it a significant contribution to the field of network hardware design.

General Terms

Data Packet, Functional Verification, Routers, Switch, VLSI-SoC

Keywords

1x4 Switch, Finite State Machine (FSM), FIFO, Functional Verification, Packet-Based Communication and System Verilog

1. INTRODUCTION

In the evolving landscape of technological communication, efficient data transmission across networks is essential. Communication networks, built on interconnected hardware and software, enable seamless information exchange and resource sharing. Routing [1], a fundamental process, ensures data packets travel from source to destination accurately and efficiently. The widely adopted Transmission Control Protocol/Internet Protocol (TCP/IP) model, with its four-layer architecture—comprising the Internet Layer, Host-to-Network Layer, Transport Layer, and Application Layer, as shown in Figure 1—provides a robust framework for this process.
Within this framework, switches play a critical role at the Internet Layer (also referred to as the Network Layer in the OSI model). They are pivotal in forwarding data packets to their correct destinations, managing network traffic, and addressing challenges such as packet loss or congestion. By facilitating efficient routing and maintaining the integrity of communication, switches ensure smooth data flow even in complex, multi-network environments.

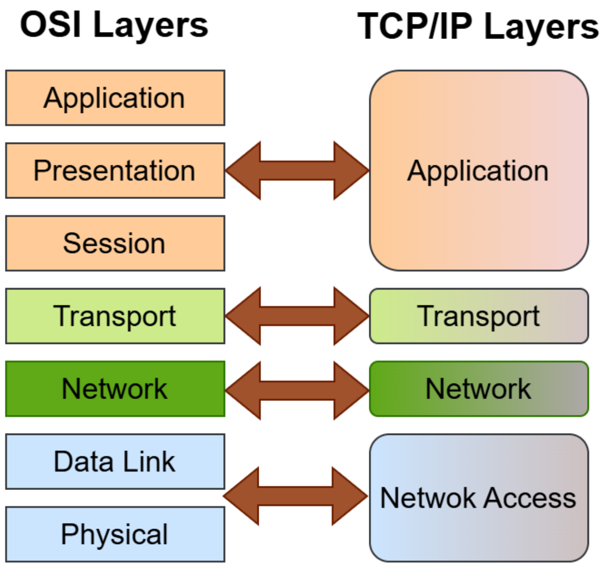

Fig. 1. TCP/IP and OSI Layers

Given the essential role that switches play in network functionality, verifying their design is crucial. Complex designs, such as Network-On-Chip (NoC) routers and switches [2], pose significant challenges due to their intricacy, power constraints, and real-time performance requirements. Functional bugs in these devices are a primary cause of ASIC [3] respins, with nearly 60% of the design cycle dedicated to verification. To mitigate delays and improve Time-To-Market (TTM), modern verification methodologies

have become indispensable. Object-oriented approaches in Hardware Verification Languages (HVLs), such as SystemVerilog [4], enable robust testing to validate designs against specifications, ensuring reliable operation under various conditions, including edge cases and error scenarios.
Verification plays a pivotal role in ensuring switches meet design requirements and perform as intended. It prevents issues such as data corruption, misrouting, or network congestion, thereby safeguarding network performance. This process involves rigorous testing and simulation of real-world scenarios to identify and address potential issues before deployment.
This paper presents the design and functional verification of a 1x4 network switch based on the TCP/IP model, highlighting the challenges, methodologies, and tools employed to ensure proper functionality. The subsequent sections explore the design, verification strategies, and solutions to overcome challenges, offering a comprehensive perspective on the critical role of verification in deploying reliable network devices.

2. LITERATURE REVIEW

This section explores key research in router design methodologies, verification approaches, and testbenches, providing an overview of existing methods for verifying designs under test (DUTs) and emphasizing the critical role of robust verification environments. Insights from prior studies are leveraged to enhance the functional verification and optimization of the 1x4 switch for efficient packet routing.
The study in [5] highlights the benefits of FIFO optimization techniques in real-time data transmission and multi-core systems. Techniques such as pipeline registers and Gray-code-based pointers address buffering challenges effectively. While asynchronous FIFOs handle multiple clock domains, their complexity is less critical for the 1x4 switch in synchronous, packet-based data transmission. This project builds upon these concepts by employing a synchronous FIFO design to ensure efficient and reliable packet routing.
The research in [6] provides valuable insights into FIFO optimization, FSM design, and system architecture for data buffering and flow management. Applying these techniques, this project implements a reduced-state FSM for improved response times and clock domain synchronization to prevent metastability. These enhancements ensure stability and robustness in the 1x4 switch.
The work in [7] on the design and verification of a 1x3 router serves as a foundational reference. The router's functionality was validated using FIFO monitoring, FSM implementation, and error handling for packet sizes of 4, 14, and 16 bytes. Lessons learned, such as resolving coding flaws and ensuring correct state management, directly influenced the design and development of the 1x4 switch.
The study in [8] focuses on a four-port NoC router with optimized FIFO-based buffers to manage area and power consumption. Using X-Y routing and Round Robin arbitration, the design achieves area efficiency but faces scalability issues and delays under high traffic. These limitations inspired further optimization in this project, focusing on latency, throughput, and network reliability for the 1x4 switch. By addressing error handling and adapting FIFO designs, the current work overcomes challenges observed.
The paper in [9] discusses a 4×4 mesh NoC architecture implemented on an Artix 7 FPGA. Its emphasis on scalability, reusability, and efficient bandwidth provides valuable insights into optimizing data flow and routing protocols. These principles are adapted in this project to enhance the 1x4 switch's handling of packet data, ensuring minimal latency and efficient communication.
This project integrates design procedures, drawing on insights from [10], and employs modern verification methodologies, i.e., SystemVerilog-based functional verification, for the 1x4 switch. Advanced techniques, such as random test vector generation and robust testbench frameworks, enhance the DUT's verification across diverse communication scenarios. The emphasis on comprehensive testing environments ensures functional compliance and reliability in packet-based protocols.

3. DESIGN METHODOLOGY

3.1 1x4 switch Design

The 1x4 switch is designed to efficiently route incoming packets to one of four possible output ports based on the destination address in the packet header. It incorporates robust FIFO management, an efficient FSM controller, and a flexible memory interface to ensure seamless data transmission across multiple ports. The switch operates synchronously with a clock signal and utilizes an active-low reset to initialize its components. This architecture, implemented in Verilog HDL [11], aims for fast data transfer by efficiently managing control signals, packet routing, and FIFO operations, ensuring high reliability and performance in packet-switched communication systems, as presented in Figure 2.

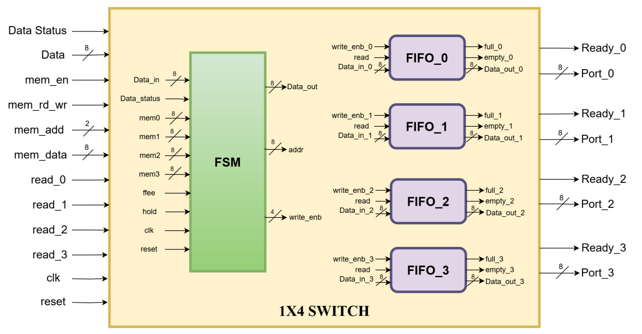

Fig. 2. 1x4 switch design architecture

3.1.1 *Clock & Reset.* :

The switch operates with a single clock signal, ensuring synchronous operation. A reset signal is used to initialize the entire system, clearing any previous data and resetting all registers, including FIFOs, to indicate the system is in a ready state. This mechanism guarantees no stale data is passed through the system.

3.1.2 *Input and Output Ports.* :

a) Input Port: Receives incoming packets and routes them based on the destination address:

—**Data:** The data signal is synchronized with the clock and carries packet payload.
—**Data_Status:** Indicates the arrival of a new packet, used to trigger data processing.

b) Output Ports: Four distinct output ports (`Port_0`, `Port_1`, `Port_2`, and `Port_3`) are available to transmit the data.

—**Read Signals (Read_0–Read_3):** Enable reading from output ports.

—**Ready Signals (Ready_0–Ready_3):** Indicate when data is ready for transmission.

—**Data Signals (Port_0–Port_3):** Transmit the payload of the packet to the appropriate output port.

3.1.3 Memory Interface. :
The memory interface is essential for routing the packets based on the destination address in the packet header. The system uses a 2-bit address space to identify one of the four output ports (0–3). The key components of the memory interface include:

—**mem_en:** Enables memory access.

—**mem_rd_wr:** Defines whether the memory is in read or write mode.

—**mem_add:** Specifies the port address for routing.

—**mem_data:** Contains the data corresponding to the routing address.

The memory interface plays a pivotal role in configuring the port addresses dynamically during operation.

3.1.4 Switch Packet Format. :
As illustrated in Figure 3, Each data packet consists of the following fields:

—**Header:** Contains essential routing information, including the destination address (DA), source address (SA), and data length.

—**Data:** The payload of the packet.

—**Frame Check Sequence (FCS):** A parity checks to verify data integrity during transmission.

The switch is designed to handle packet sizes ranging from 4 to 259 bytes, with parity checking for data integrity.

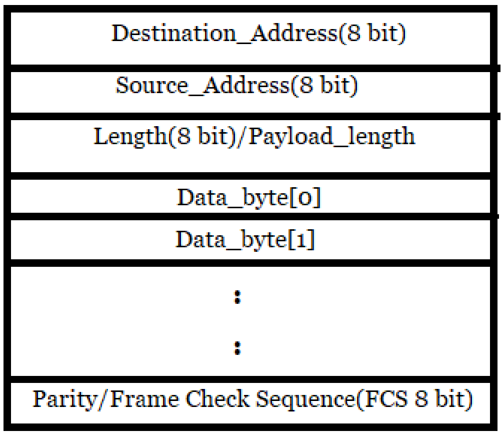
Destination_Address(8 bit)
Source_Address(8 bit)
Length(8 bit)/Payload_length
Data_byte[0]
Data_byte[1]
:
:
Parity/Frame Check Sequence(FCS 8 bit)

Fig. 3. Packet Format

3.1.5 FIFO Mechanism. :
The switch employs four 16x8 FIFOs (First-In-First-Out buffers) for data transfer between the input and output ports. The FIFO mechanism ensures that data is efficiently buffered while minimizing packet loss. Each FIFO is synchronized with the clock, with the following operations.

—**Write:** Enabled when the FIFO is not full.

—**Read:** Enabled when the FIFO is not empty.

The FIFO mechanism includes flags such as Full (indicating no space for more data) and Empty (indicating no data to read). The internal structure as shown in Figure 4 is further managed by pointer and count blocks, ensuring smooth operation:

—**Pointer Block:** Generates read/write pointers for FIFO operations.

—**Count Block:** Tracks data count and sets ”full” and ”empty” flags.

—**Read and Write Blocks:** The read block uses the read pointer to access data, while the write block uses the write pointer to insert data into the memory.

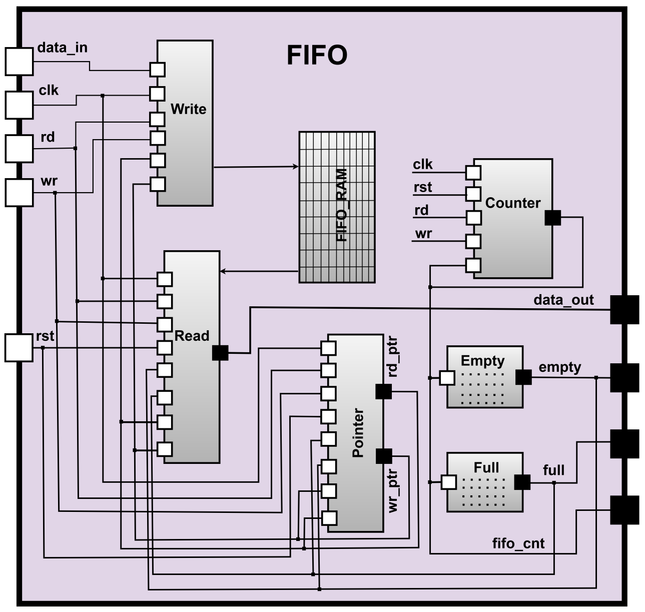

Fig. 4. FIFO Internal modules

3.1.6 FSM Module. :
The FSM controls the flow of packets through the switch, transitioning between different states based on the status of the FIFOs and the packet data. It manages critical operations such as data loading, parity checking, and control signal generation to ensure accurate data routing and integrity. The FSM includes the following states, as illustrated in Figure 5.

—**Addr_Wait:** Waits for incoming packets and extracts the destination address.

—**Data_Load:** Loads the payload into the FIFO once the destination is determined.

—**Parity_Load:** Calculates and validates the parity bit for error checking.

—**Hold_State:** Waits for available space in the FIFO to continue processing.

—**Busy_State:** Pauses operations when the FIFO is empty.

This state machine ensures the correct and efficient processing of packets, with proper synchronization and error handling.

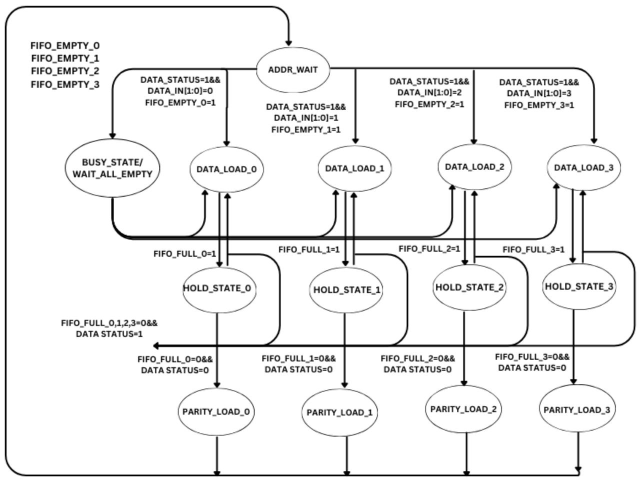
FIFO_EMPTY_0
FIFO_EMPTY_1
FIFO_EMPTY_2
FIFO_EMPTY_3
ADDR_WAIT
DATA_STATUS=1&& DATA_IN[1:0]=0 FIFO_EMPTY_0=1
DATA_STATUS=1&& DATA_IN[1:0]=1 FIFO_EMPTY_1=1
DATA_STATUS=1&& DATA_IN[1:0]=2 FIFO_EMPTY_2=1
DATA_STATUS=1&& DATA_IN[1:0]=3 FIFO_EMPTY_3=1
BUSY_STATE/ WAIT_ALL_EMPTY
DATA_LOAD_0
DATA_LOAD_1
DATA_LOAD_2
DATA_LOAD_3
FIFO_FULL_0=1
FIFO_FULL_1=1
FIFO_FULL_2=1
FIFO_FULL_3=1
HOLD_STATE_0
HOLD_STATE_1
HOLD_STATE_2
HOLD_STATE_3
FIFO_FULL_0,1,2,3=0&& DATA STATUS=1
FIFO_FULL_0=0&& DATA STATUS=0
FIFO_FULL_1=0&& DATA STATUS=0
FIFO_FULL_2=0&& DATA STATUS=0
FIFO_FULL_3=0&& DATA STATUS=0
PARITY_LOAD_0
PARITY_LOAD_1
PARITY_LOAD_2
PARITY_LOAD_3

Fig. 5. Finite State Machine

3.2 Verification Environment setup for 1x4 switch

The verification environment, depicted in Figure 6, is meticulously designed to rigorously test the functionality of the 1x4 switch under various operating conditions. This setup ensures comprehensive testing by providing stimulus, monitoring responses, and maintaining synchronization between components. Key elements include the Design Under Test (DUT), test bench components (stimulus generators, checkers, monitors), and verification strategies for functional and code coverage [12]. This framework ensures the switch operates correctly under all scenarios.

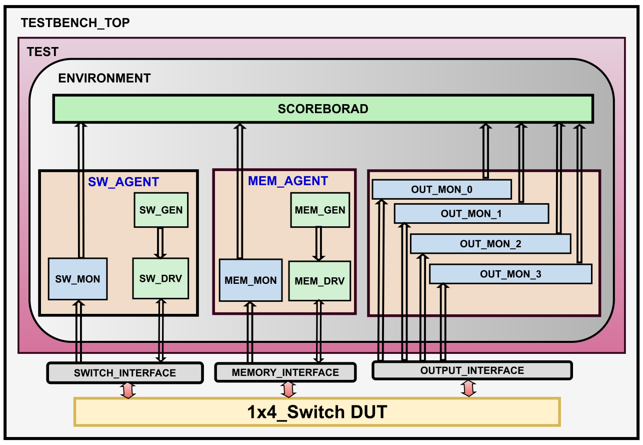

Fig. 6. Verification Environment of 1x4 Switch

3.2.1 *Design Under Test (DUT).* :

The 1x4 switch is the Design Under Test (DUT) in this verification setup. It is a digital switch designed to route a single input to one of four output ports based on control signals. The DUT's operation is validated through a series of test cases that simulate diverse operating conditions. Interfaces and Protocols: The DUT communicates using standard protocols such as AXI [13] or APB [14], with input and output signals closely monitored to verify proper functionality and compliance with protocol specifications

3.2.2 *Verification Environment Overview.* :

The verification environment is designed to thoroughly test the DUT by generating stimulus, monitoring outputs, and comparing the results with expected behaviors. Key components include:

Test Bench Architecture:

This is the core structure of the verification setup, including the interaction between the stimulus generator, driver, monitor, scoreboard, and coverage tools.

—**Stimulus Generator:** Creates test inputs that simulate various operating conditions for the DUT.

—**Driver:** Drives the generated stimulus into the DUT's input ports.

—**Monitor:** Observes and logs the DUT's outputs for comparison.

—**Scoreboard:** Compares the DUT's outputs with the expected outputs to ensure correctness.

—**Coverage Tools:** Track the portions of the DUT exercised during simulation, ensuring full test coverage.

Functional Verification Components:

—**Environment Class:** Manages integration and interaction between verification components.

—**Test Class:** Defines specific test scenarios, applies stimuli, and validates results.

—**Test Scenarios:** Includes normal operations, edge cases, and random inputs to test robustness.

3.2.3 *System Verilog Constructs for Synchronization.* :

Proper synchronization between the various components of the verification environment is crucial. This is achieved through SystemVerilog constructs [12]:

—**Semaphores:** Used to synchronize access to shared resources during the simulation.

—**Events:** Provide a mechanism for components to synchronize at specific points during the simulation.

—**Queues and Mailboxes:** Facilitate communication between different components in the test bench.

3.2.4 *Functional and Code Coverage.* :

Comprehensive verification ensures the DUT is exercised under all conditions:

—**Functional Coverage:** Ensures that all functional aspects of the DUT are tested, including all input combinations and control signal values.

—**Code Coverage:** Tracks the execution of code lines in both the DUT and the verification environment, ensuring that all logic has been tested.

3.2.5 *Assertions and Checkers.* :

Assertions verify that the DUT behaves as expected. It covers:

—**Property Assertions:** Used to verify that the DUT's behavior adheres to specified properties during simulation.

—**Sequence Assertions:** Ensure that specific sequences of events happen in the expected order, providing further verification of correct functionality.

3.2.6 Test Cases and Scenarios. :
The verification environment uses diverse test cases for comprehensive coverage:

—**Directed Tests:** Target specific functionalities or edge cases.

—**Random Tests:** Check robustness with randomly generated stimuli.

—**Edge Cases:** Assess DUT behavior under extreme or boundary conditions.

3.2.7 Testbench Control Flow. :
This section describes Simulation Workflow:

—**Setup:** Instantiate the DUT and initialize verification components.

—**Execution:** Apply test scenarios, check results, and monitor simulation.

—**Termination:** End simulation after all tests complete or upon detecting failures.

3.2.8 Reporting and Debugging. :
Effective reporting and debugging ensures DUT correctness, it constitutes:

—**Result Logs:** All results, including pass/fail status, are logged during the simulation for later analysis.

—**Waveform Analysis:** Waveforms are examined to visually inspect the signals and ensure they match expectations.

—**Debugging Techniques:** Methods such as signal tracing, adding logging to the testbench, or using waveform viewers to isolate issues in the DUT or verification environment.

4. IMPLEMENTATION DETAILS AND DISCUSSIONS

This section outlines the design and verification results for the 1x4 Switch. The design was implemented in Verilog and simulated using Xilinx ISE Design Suite 14.7. Functional verification was performed with the online EDA tool and Synopsys 2021.09, while RTL-to-Netlist synthesis was carried out using Xilinx ISE 14.7.

4.1 Synthesis and Simulation Results

The RTL schematic and timing diagrams illustrate the top-level view, internal functional modules, and operational flow of the 1x4 switch, confirming its functionality and alignment with the design specifications. The results of RTL synthesis for the FIFO, FSM, and switch components are presented in Figures 7, 8, 9, and 10.
FIFO (First in First out) is a memory management mechanism where data is queued in the order it is written and read in the same sequence, as observed in the timing diagram of Figure 11. The following processes take place:
Initialization:

—When reset is active (`reset=0`), the FIFO initializes, and the input data (`data_in[7:0]`) is ready for simulation.

Write (Push) Operation:

—Data is written to the FIFO at the location pointed to by `write_ptr` when the full signal is deasserted (`full=0`).

—The `write_ptr` increments after each write, pointing to the next available location.

Read (Pop) Operation:

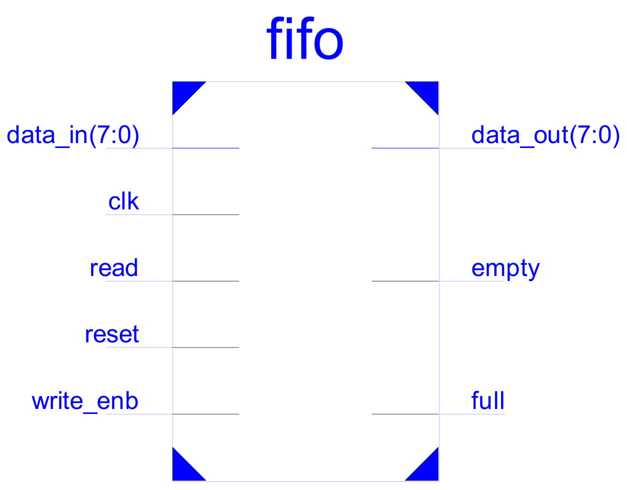

Fig. 7. FIFO RTL Schematic

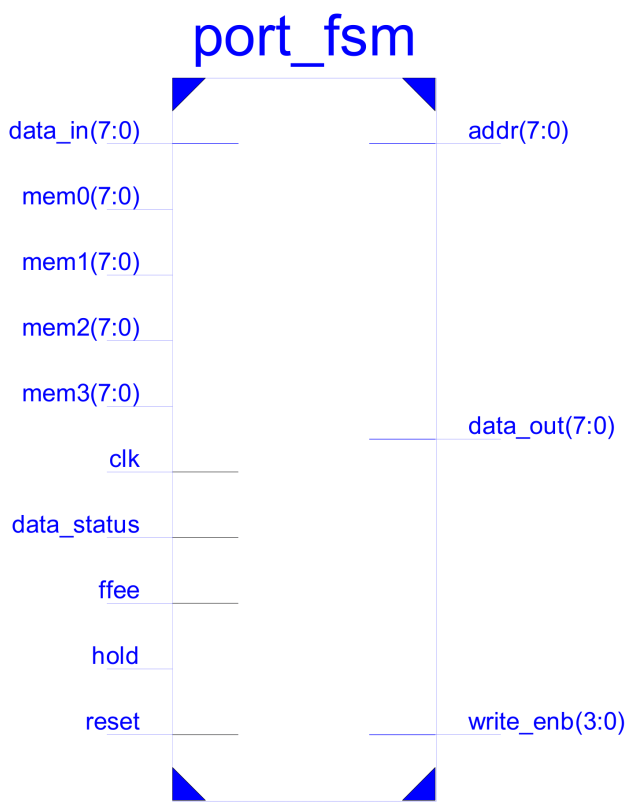

Fig. 8. FSM RTL Schematic

—Data is read from the FIFO at the location pointed to by `read_ptr` when the empty signal is deasserted (`empty=0`).

—The `read_ptr` increments after each read, pointing to the next data to be read.

Simulation Behavior:

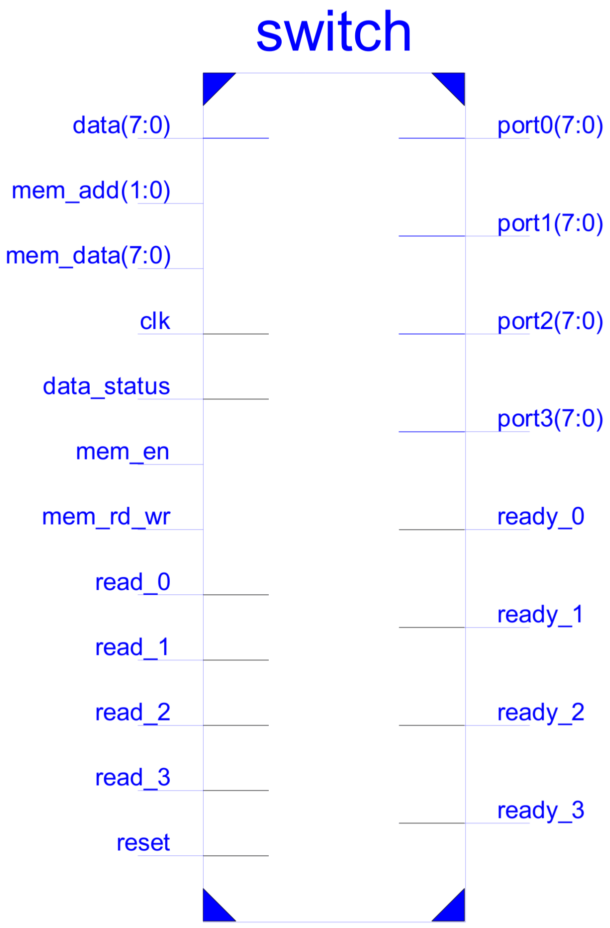

Fig. 9. 1x4 Switch Top-Level View

—During simulation, you can observe data written to a specific location (e.g., `write_ptr=1`) being read in the next clock cycle from the same location using the `read_ptr`.

—This operation continues sequentially for all 16 locations, provided that `full` remains deasserted during write operations and `empty` remains deasserted during read operations, which can be observed in the figure.

Key Constraints:

—Writing is allowed only when the FIFO is not full (`full1`).

—Reading is allowed only when the FIFO is not empty (`empty1`).

The FSM, which monitors the switch mechanism, is detailed in the preceding sections, and its RTL schematic is shown in Figure 9. The 1x4 Switch is designed to route data packets to one of four output ports, functioning similarly to a de-multiplexer. Its operation is governed by configuration parameters stored in memory, specifically the `[7:0] mem_data` signal and the `[1:0] mem_add` address field.

The switch compares the destination address (DA) in the packet header with the configured port address. If a match is found, the packet is directed to the corresponding port (0, 1, 2, or 3), as depicted in Figure 12. The switch supports dynamic payload lengths, where the number of transmitted bytes is determined by

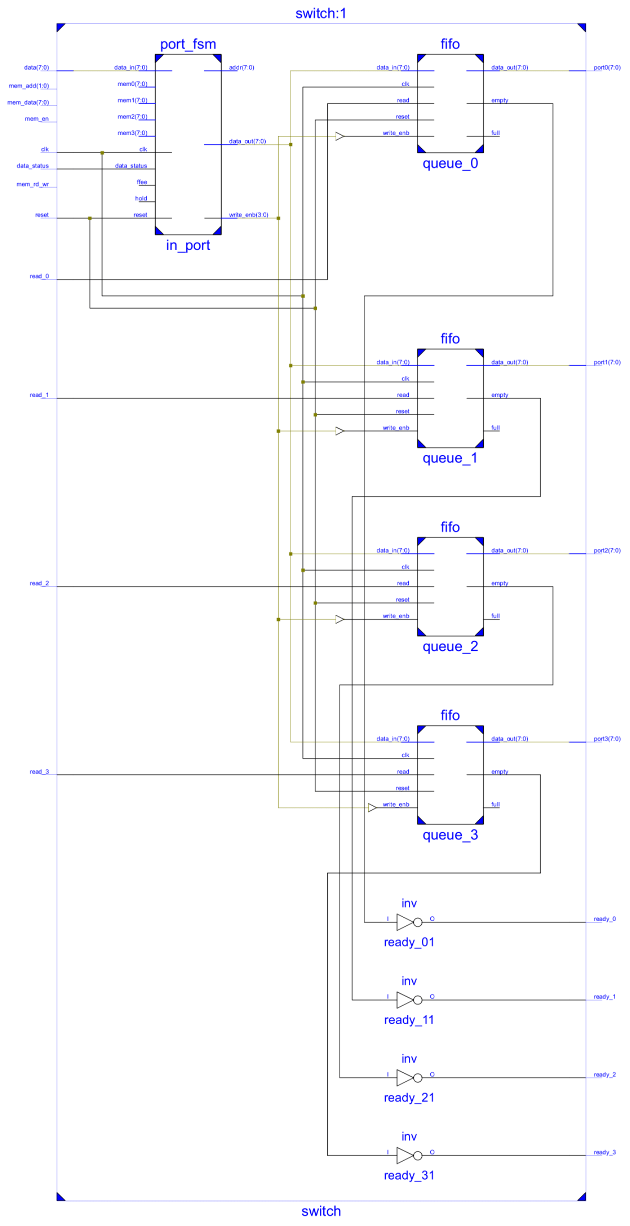

Fig. 10. 1x4 Switch Internal View

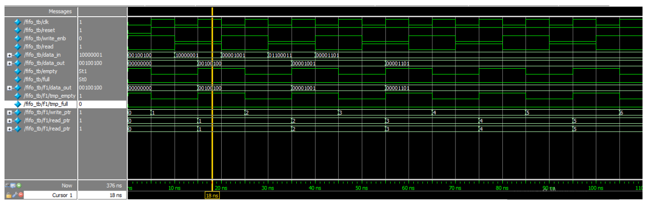

Fig. 11. Simulation results of FIFO

the `payload_length` field. Control signals `ready_0`, `ready_1`, `ready_2`, and `ready_3` are used to indicate which output port is

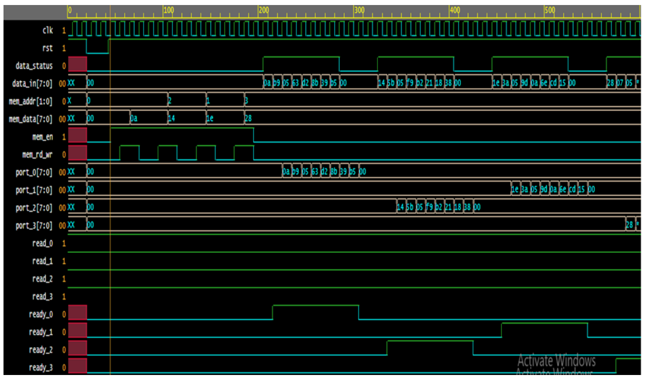

Fig. 12. Simulation results of 1x4 Switch

currently active, providing feedback about the port receiving the data. This mechanism ensures efficient routing and minimizes delays.

4.2 Functional Verification results with Different Test cases

Functionality Verification validates the switch's ability to handle various test cases, including packets with minimum and maximum sizes, data integrity checks, and reset conditions. Additionally, the switch successfully transmits data from all ports when the corresponding configuration and header addresses match. Below are simulation results of a few test cases:

4.2.1 Reset and Random test:. Figure 13 presents the simulation results of the 1x4 Switch under the random test case. This test applies constrained random stimuli along with a reset signal to the Device Under Test (DUT), aiming to verify the switch's behavior across a wide range of input conditions.
During the test, the `[7:0] mem_data` signal represents the packet's destination address (DA). The switch compares this address with the configured `[1:0] mem_add` location to determine the appropriate output port, while the `payload_length` field controls the number of bytes transmitted.
When the reset signal is asserted, all internal states of the switch are initialized to default values, as observed in the waveform. The test confirms that the 1x4 Switch handles reset conditions and routes packets correctly to all ports based on address matching.

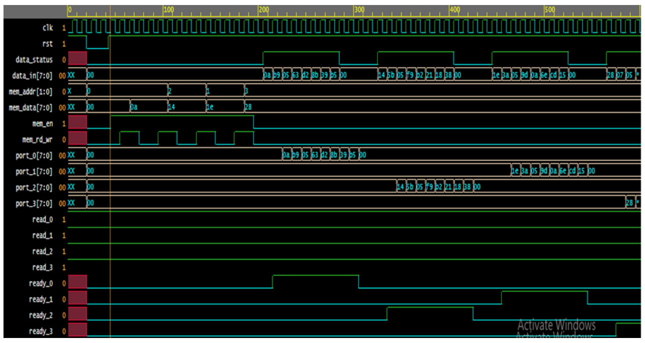

Fig. 13. Random test case

4.2.2 Packet with Minimum and maximum Size:. Figures 14 and 15 demonstrates the simulation results for the 1x4 Switch handling packets of minimum and maximum sizes. In the minimum size test case, a packet with a payload length of 0 is transmitted, containing only the DA (Destination Address), SA (Source Address), length, and parity fields, each occupying 1 byte, resulting in a 4-byte packet. The test verifies whether the switch can correctly identify and route minimal packets to the appropriate output port based on the DA field. Similarly, in the maximum size test case, a

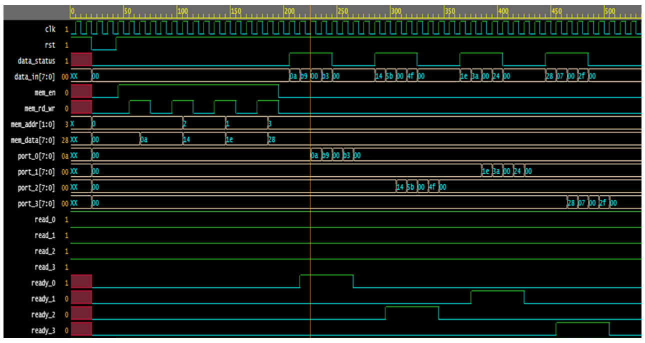

Fig. 14. Minimum Size test case

packet with a payload length of 255 is tested, containing the DA, SA, length (255), 255 data bytes, and parity, totaling 259 bytes. The switch routes the packet to the corresponding port (0, 1, 2, or 3) upon detecting a match with the configured DA, confirming the switch's ability to efficiently handle and route both minimum and maximum size packets as per the protocol.

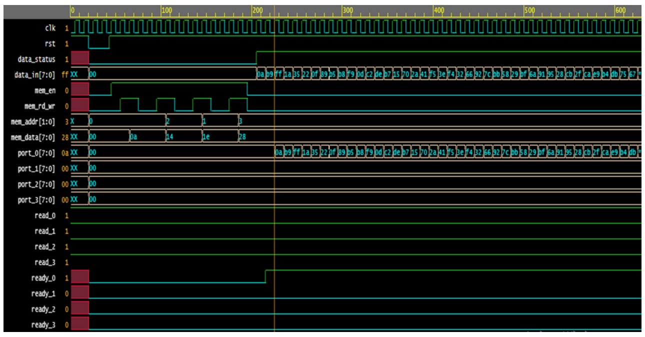

Fig. 15. Maximum Size test case

4.2.3 Packet of Data from $Port_0 to Port_3$: . Figures 16 to 19 captures the simulation results for the 1x4 Switch, testing packet routing from each port (Port0 to Port3). In each test case, the switch compares the Destination Address (DA) field of the incoming packet with the configured memory address (`mem_add`). For `mem_add` = 0, the packet is routed to Port0, for `mem_add` = 1, it goes to Port1, and similarly for `mem_add` = 2 and `mem_add` = 3, packets are routed to Port2 and Port3 respectively, based on matching DA values. These test cases verify the switch's ability to correctly route packets to the appropriate port depending on the DA field, ensuring that packets are sent to the correct destination based on the configured memory address.

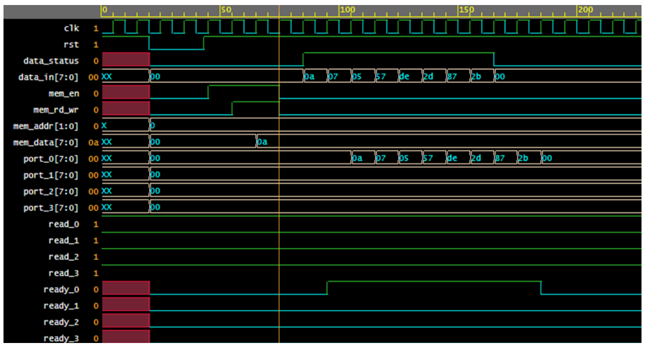

Fig. 16. Data packet from Port$_0$

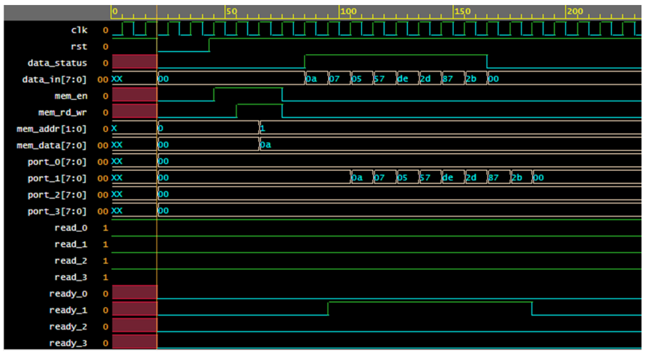

Fig. 17. Data packet from Port$_1$

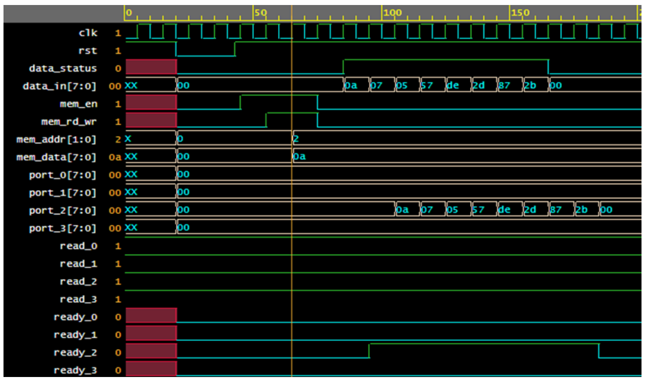

Fig. 18. Data packet from Port$_2$

4.2.4 Packet of data transmission in between reset applied:. Figure 20 illustrates the simulation result for the 1x4 Switch handling packet transmission during a reset. The output ports are configured with unique addresses, and the switch compares the *DA* (Destination Address) field of the packet with the configured port address to determine the correct output port (0, 1, 2, or 3). In this test case, a reset signal is applied during an ongoing packet transmission to evaluate the DUT's (Design Under Test) behavior under such conditions. The simulation results confirm that the packet transmission is halted, and no data is output from the ports when the reset is asserted, validating the reset functionality of the switch.

4.2.5 Packet of data sending when data status is zero:. Figure 21 represents the simulation result for the 1x4 Switch with a data status signal set to zero. The output ports are configured with unique addresses, and the switch compares the *DA* (Destination Address) field of the packet with the configured port address to route the packet to the appropriate port (0, 1, 2, or 3). In this test case, a valid signal or data status signal is not asserted during the packet transmission process, simulating a scenario where the transmission is not active. The simulation confirms that no packet data is transmitted or output from the ports under these conditions, validating the switch's behavior when the data status is zero.

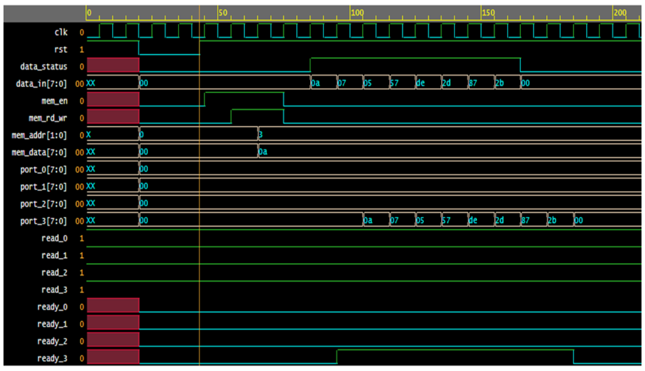

Fig. 19. Data packet from Port$_3$

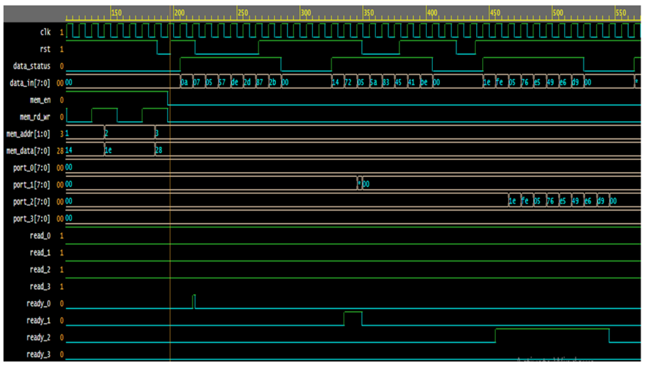

Fig. 20. Data packet transmission in between reset

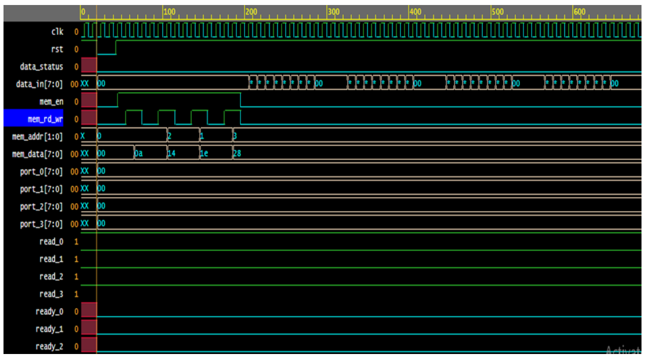

Fig. 21. test case for data status zero

4.2.6 Packet of data sending with bad FCS/parity:. Figure 22 provides the simulation result for the 1x4 Switch with a *Bad FCS/Parity* testcase. The output ports are configured with unique addresses, and the switch compares the *DA* (Destination Address) field of the packet with the configured port address to determine the correct routing (ports 0, 1, 2, or 3). In this test case, a packet with a corrupted parity or *FCS* (Frame Check Sequence) is applied to the *DUT* to test its response. The simulation reveals that the packet is transmitted through the ports despite the corrupted parity, indicating an error in handling or validating the parity/*FCS*.

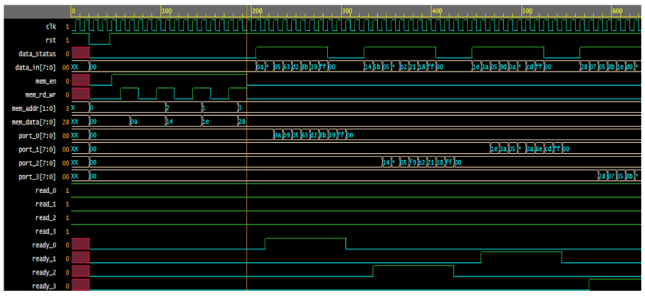

Fig. 22. test case for Bad FCS

4.2.7 Packet of data sending to cover all ports:. Figure 23 outlines the simulation result for the 1x4 Switch with the *Covering All Ports* testcase. The output ports are configured with unique addresses, and the switch evaluates the *DA* (Destination Address) field of each packet against these addresses to determine the correct routing (ports 0, 1, 2, or 3). In this test case, multiple packets are transmitted, each with a unique *DA* corresponding to a specific port. The test verifies whether the switch can correctly match the *DA* field of each packet and route it to its designated port. The simulation confirms that packets are successfully transmitted to all ports as expected, validating the switch's functionality for full port coverage.

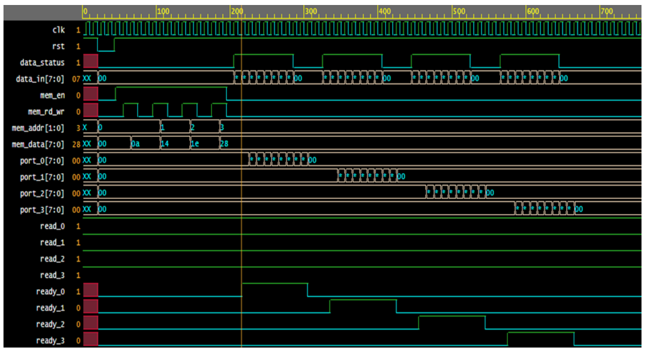

Fig. 23. test case covering all ports

5. CONCLUSION & FUTURE SCOPE

This work primarily focuses on the design and functional verification of a 1x4 switch, a packet-based communication protocol situated at the Network layer of the TCP/IP model. The switch is designed in Verilog HDL, featuring critical submodules such as FIFOs and a Finite State Machine (FSM), and is verified using SystemVerilog (SV) to ensure its functionality aligns with protocol specifications. The design was tested with various scenarios, including configuring output port addresses through the memory interface, matching the Destination Address (DA) field of packets to port addresses, and routing packets to the appropriate ports as specified. The functionality of the design was validated using sophisticated EDA tools and an online EDA playground. The synthesis and simulation processes demonstrated compliance with protocol requirements, and test cases such as minimum and maximum payload lengths, random payloads, and erroneous packets were successfully executed. However, functional coverage parameters remain an area for future exploration. Future work could extend this design by incorporating additional features and peripherals, such as routers, to cater to broader application requirements. Advanced verification strategies like UVM which offers a structured methodology for building robust and reusable verification environment, including functional coverage and constrained random testing, can be implemented to achieve comprehensive verification of the enhanced design. This would enable the design to meet stringent performance standards and expand its applicability in complex networked environments.

6. ACKNOWLEDGMENTS

Our thanks to the experts who have contributed towards development of the template.

7. REFERENCES

[1] Balchunas, "Hubs vs. Switches vs. Routers v1.33," 2014. [Online]. Available: `https://www.routeralley.com/guides/hubs_switches_routers.pdf`. [Accessed: Dec. 11, 2024].

[2] M. A. Javed Sethi, F. A. Hussin, and N. H. Hamid, "Review of Network on Chip Architectures," *Recent Advances in Electrical & Electronic Engineering*, vol. 10, pp. 4-29, 2017. doi: 10.2174/2352096510666170425102503.

[3] K. Golshan, *ASIC Design Implementation Process: A Complete Framework*, 1st ed., Springer Cham, 2024. doi: 10.1007/978-3-031-58653-8.

[4] C. Spear, *SystemVerilog for Verification: A Guide to Learning the Testbench Language Features*, Springer Science+Business Media, 2006. ISBN-13: 978-0387270364, e-ISBN-13: 978-0387270388. Available: Springer Link.

[5] Y. Gao, "Research and analysis of FIFO related working principles," *Proc. 5th Int. Conf. Computing and Data Science*, 2023. doi: 10.54254/2755-2721/14/20230770.

[6] E. Lavanya, K. Sharath Chandra, K. Naveen, and K. Rupesh, "Router 1x3-RTL router design and verification," *Int. Res. J. Modernization Eng. Technol. Sci.*, vol. 4, no. 5, May 2022. [Online]. Available: `https://www.irjmets.com`.

[7] P. Musale, P. Thakre, and Y. Raut, "Design and verification of 1*3 router," *Int. J. Modern Trends Sci. Technol.*, vol. 9, no. 6, pp. 184-194, 2023. doi: 10.46501/IJMTST0906027.

[8] Dr. Uma B. V and R. Pakala, "Design and implementation of four-port router for network on chip," *Int. J. Eng. Res. Technol.*, vol. 8, no. 6, pp. 520-524, June 2019. [Online]. Available: Publisher Link.

[9] J. Jose and M. T. V, "Design and verification of an efficient packet-based switching network-on-chip," *Int. J. Sci. Res. Eng. Manag.*, vol. 6, no. 7, pp. 1, Jul. 2022. doi: 10.55041/IJSREM15285.

[10] A. S. Chaithanya, D. Sindhuja, D. Bhavana, and P. Vennela, ”Design and interfacing of I2C master with register and LCD slaves,” *Int. J. Eng. Adv. Technol. (IJEAT)*, vol. 9, no. 4, pp. 2355–2360, Apr. 2020. doi: 10.35940/ijeat.D7901.049420.

[11] S. Palnitkar, *Verilog HDL: A Guide to Digital Design and Synthesis*, 2nd ed., Prentice Hall PTR, 2003. ISBN: 0-13-044911-3. Available: Pearson Link.

[12] ”SystemVerilog Simple Testbench,” ChipVerify. [Online]. Available: `https://www.chipverify.com/systemverilog/systemverilog-simple-testbench`.

[13] A. Sainath Chaithanya, S. Sulthana, B. Yamuna, and C. Haritha, ”Design of AMBA AXI4-Lite for effective read/write transactions with a customized memory,” *Int. J. Emerg. Technol.*, vol. 11, Issue.1, pp. 396–402, 2020. Available: Publisher Link.

[14] P. Jain and S. Rao, ”Design and Verification of Advanced Microcontroller Bus Architecture-Advanced Peripheral Bus (AMBA-APB) Protocol,” *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, Tirunelveli, India, 2021, pp. 462-467. doi: 10.1109/ICICV50876.2021.9388549.