

Implantation of 1X3 Router in FPGA

Prof. Nirmala Y N¹, Kruthik Prasad B K², Kapa Vinay Kumar Reddy³, GVS Mohan Kireety⁴ and Hemanth R⁵
¹⁻⁵Assistant Professor Dept. of ECE, DSCE, Bangalore
 Email: nirmalamurthy.yn@gmail.com

Abstract— In this paper, we expound the design and implementation of a 1x3 router based on FPGA for communication and embedded systems. One of router's three outputs is routed to from an 8-bit input data stream, selected by a con-figurable 2-bit select signal allowing for greater flexibility and determinism in data distribution. The specific work is described in a synthesizable Verilog HDL to evaluate the resource utilization and performance on intel FPGA (MAX 10 FAMILY) The functional simulations of router prove the correct functionality for different input data. The synthesis results show that it occu-pies fewer resources, and the timing analysis proves its high speed, which is capable of operational in real-time applications.

Keywords— Routing, Router, Verilog HDL, Top Module, ModelSim, FIFO, Synchronizer, Register, FSM.

I. INTRODUCTION

Routers use to transfer information between two or more networking devices. Pretty much, this is a simple network device that takes one input package and sends it to only 1 of the 3 output connections. A 1x3 router is a type of optical switching device that plays a crucial role in modern telecommunications and networking systems, particularly in optical networks. It is designed to route an optical signal from a single input port to three separate output ports, thereby enabling the efficient distribution of data across multiple paths. The term "1x3" re-fers to the number of input and output channels, where "1" signifies one input channel, and "3" denotes the three output channels. The primary function of a 1x3 optical router is to manage the direction of light signals in an optical network, making it integral to the functioning of wavelength division multiplexing (WDM) systems. These routers help ensure that signals are appropriately distributed, maintaining data integrity and optimizing the performance of the network. The widespread use of 1x3 routers is essential for various applications, including fiber-optic communication systems, data centers, and broadband networks, where routing efficiency and flexibility are critical. In terms of architecture, 1x3 routers are typically designed to handle signals that travel in the form of light, as op-posed to traditional electronic signals. This optical switching technology ensures that data can travel at high speeds and over long distances with minimal signal loss, making it ideal for high-capacity networks. Such routers can be constructed using different technologies, such as MEMS (Micro-Electro-Mechanical Systems), liquid crystal devices, or arrayed waveguide gratings (AWGs).The working principle of the 1x3 router hinges on the ability to manipulate the path of the light based on specific criteria, such as the wavelength of the signal or the time it enters the system. By adjusting the angle or direction of light using mirrors or liquid crystal elements, the router directs the incoming signal to one of the three output ports. This flexibility is crucial in dynamically allocating bandwidth and optimizing network traffic. One of the significant advantages of a 1x3 optical router is its ability to handle multiple wavelengths simultaneously, offering the possibility of high-throughput data transmission.

This feature is especially important in high-demand environments like large data centers or internet backbone infrastructure, where the need for large bandwidth and the efficient use of re-sources is paramount. As the demand for faster internet speeds and more reliable connections continues to grow, the role of devices like 1x3 routers becomes increasingly vital in supporting optical fiber networks. These routers help minimize the need for electrical conversions and enable seamless communication by di-rectly managing light signals. With the advent of 5G networks and smart cities, the scalability and efficiency of optical routers will only become more critical. In conclusion, the 1x3 router is a key component in modern optical networks, en-suring that data is efficiently routed to multiple destinations while maintaining high-performance standards. As technology evolves and networks become more complex, the importance of routers like these will only continue to rise, making them an indispensable part of the future of telecommunications.

II. PROPOSED METHODOLOGY

- **System Requirements and Analysis**
Defined system requirements for the 1x3 router, including data paths, routing logic, and submodule specifications like FIFO, Register, FSM, and synchronizer.
- **Design and Development**
Developed the submodules (FIFO, Register, FSM, Synchronizer) in Verilog and integrated them into the Router Top Block to ensure full functionality.
- **Functional Simulation**
Conducted RTL simulations using ModelSim with testbenches to verify the operation of each submodule.
- **Design Verification**
Performed linting with Synopsys VCS to detect issues and validated the router system against the specifications.
- **FPGA Implementation:** Synthesized and deployed the Router Top Block on Intel MAX 10 FPGA, validating its functionality in hardware.

A. Network packet structure

The Fig-1 represents the packet structure is a fundamental aspect of data commu-nication in computer networks, ensuring efficient transmission and accurate de-livery of data. It is divided into three primary components: the Header, Payload, and Parity Bytes. These elements collectively define the packet's routing, data content, and integrity verification mechanisms.

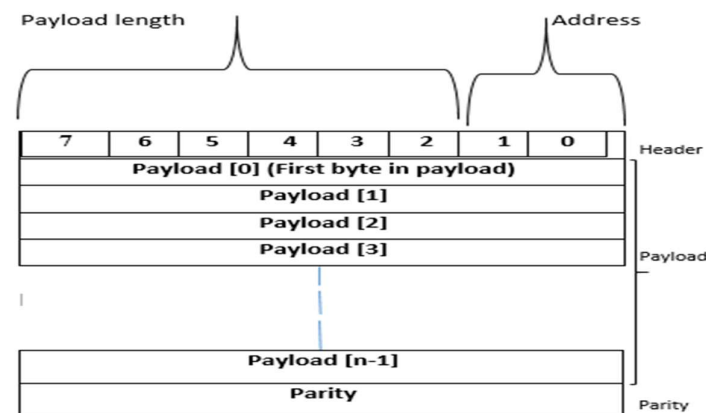


Fig-1: Network Packet Structure

- **Header:** The Header serves as the control section of the packet, contain-ing crucial information for packet processing and routing. It generally consists of two key fields: Destination Address (DA) and Length.
- **Destination Address (DA):** This field is typically 2 bits long and speci-fies the destination port where the packet should be forwarded. The router uses this address to identify the correct output port based on the match between the DA and the assigned port addresses. Each port in the network has a unique 2-bit address. If the DA matches one of the port addresses, the packet is directed to that port for further processing. An address value of "3" is considered invalid and does not correspond to any output port.

- **Length:** The Length field is 6 bits in size and indicates the number of data bytes contained in the packet's payload. This value helps the router identify the data size to process. The length can range from a minimum of 1 byte to a maximum of 63 bytes. For example, a length value of 1 indicates that the data field contains only 1 byte, while a value of 63 specifies the maximum allowable size of 63 bytes in the payload.
- **Payload:** The Payload section of the packet carries the actual data being transmitted across the network. This field's content is determined by the communication requirements and can vary in size, typically specified by the Length field in the header. The payload represents the core data that the sender wants to transfer to the receiver, excluding any control or routing information. In modern networks, the payload may carry application data, such as web page content, files, or other types of user-specific data.
- **Parity:** The Parity field is used for error detection and security. It ensures that the packet has not been corrupted during transmission. The parity bit is calculated using a bitwise parity check over both the header and payload sections. If any bit within the header or payload is altered during transit, the parity check will fail, indicating a transmission error. This helps in identifying and correcting errors in the packet before it is processed further, ensuring data integrity and reliable communication across the network.

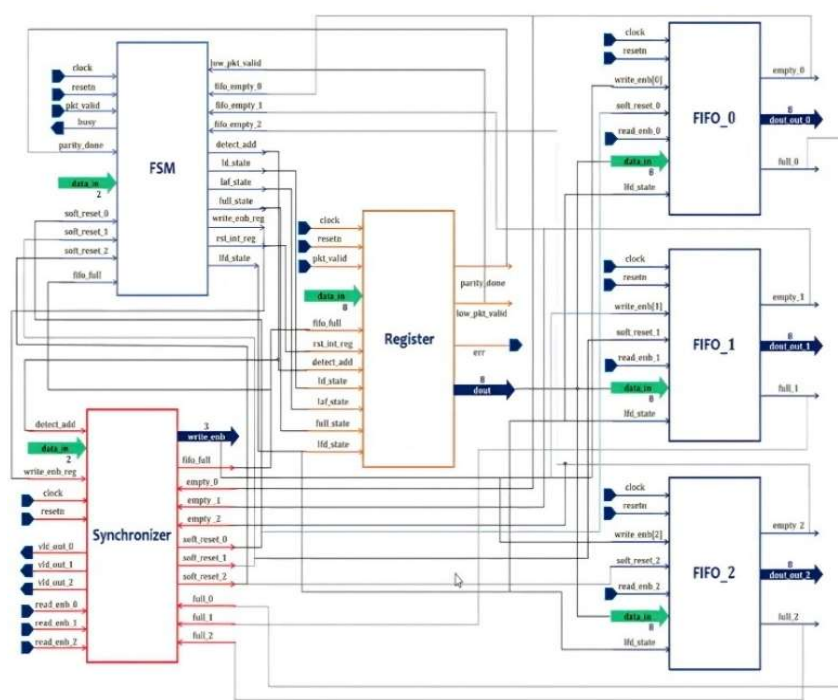


Fig-2 Block diagram of Router

The block diagram illustrates the structure and interaction of key components within a 1x3 router system. These components include a Finite State Machine (FSM), Synchronizer, Registers, and FIFO buffers (FIFO_0, FIFO_1, FIFO_2), all of which work together to facilitate the correct processing and routing of data packets.

B. Data Flow Process

The router receives incoming data packets, which are first processed by the FSM to determine the correct output port based on the packet's destination address. The FSM then directs the packet to the appropriate FIFO buffer. The synchronizer ensures that the signals between the different clock domains are consistent. Once the packet is stored in the FIFO, the router forwards it when the corresponding output port is ready to transmit. This method ensures that data is routed in an efficient, timely manner with minimal risk of data loss or overflow.

FSM:

The FSM acts as the control unit for the router, coordinating the overall flow of data. It determines the current state of the system and issues commands to other modules based on the input conditions. Specifically, it manages the flow of data by controlling signals such as `write_en`, which enables writing to the FIFO buffers, and monitors status flags like `fifo_full` to avoid overflows.

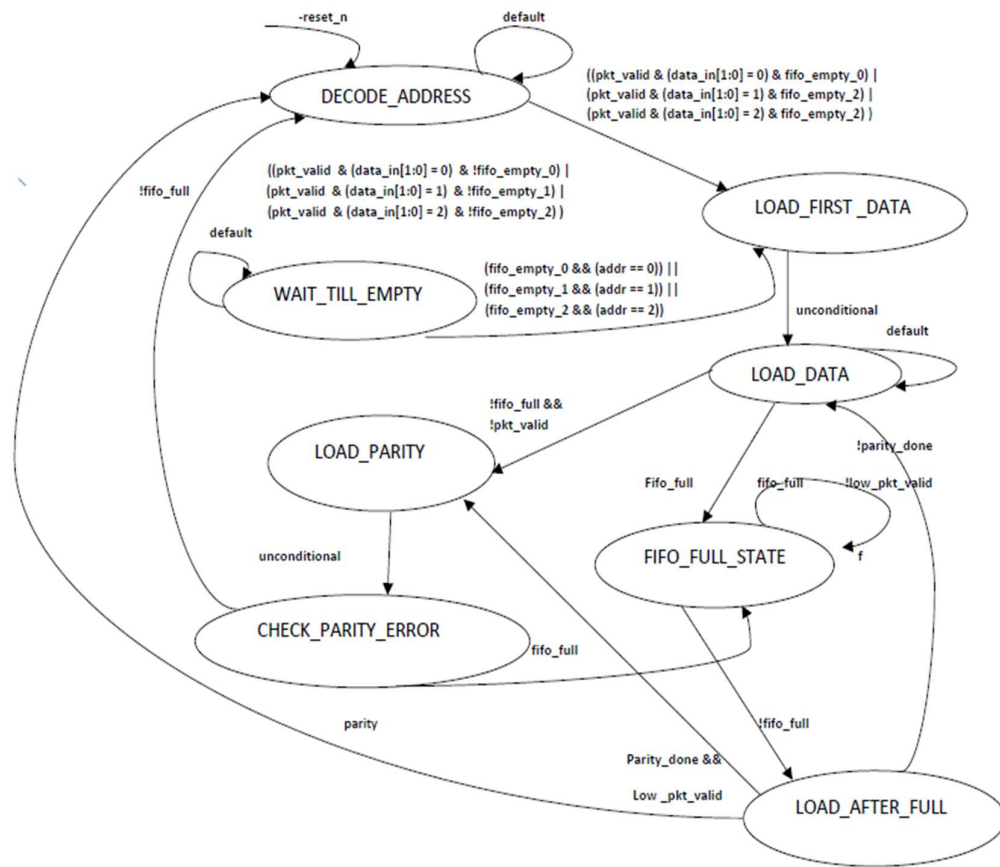


Fig 3: FSM States

The Fig-5 represents FSM, In the 1x3 router system, the Finite State Machine (FSM) is essential for managing the routing logic that controls the flow of data between the router's inputs and outputs.

The FSM operates by transitioning through a series of predefined states, each corresponding to a specific function within the router's operation. These states include decode_address, which processes incoming address data, lfd_state for managing the first stage of data flow, ld_state for loading data into buffers, laf_state for transferring data to the output, and load_parity for ensuring parity bits are included with the data. Additionally, the FSM handles fifo_full_state to monitor buffer capacity and prevent overflow, as well as check_parity_error to verify data integrity.

The FSM is responsive to various control signals and input conditions. It determines the appropriate times to perform read and write operations on the FIFO buffers based on the current state, ensuring that data is routed correctly through the system. The FSM also controls which output channel is activated, depending on the state and the address information received.

By managing these actions through specific state transitions, the FSM ensures smooth and efficient operation of the router. It prevents data conflicts, ensures that no data is lost, and maintains optimal performance even in situations with high traffic or complex routing scenarios. This careful coordination of states allows the router to function reliably and efficiently, with minimal risk of errors or delays in data transmission.

Synchronizer

The synchronizer's primary role is to handle synchronization between different clock domains, which is particularly important when dealing with components operating at different clock speeds.

By synchronizing signals such as fifo_empty and fifofull, it ensures that the data flow between the FIFO buffers and other modules is consistent and reliable.

Additionally, it controls read and write operations by using signals like written and readen.

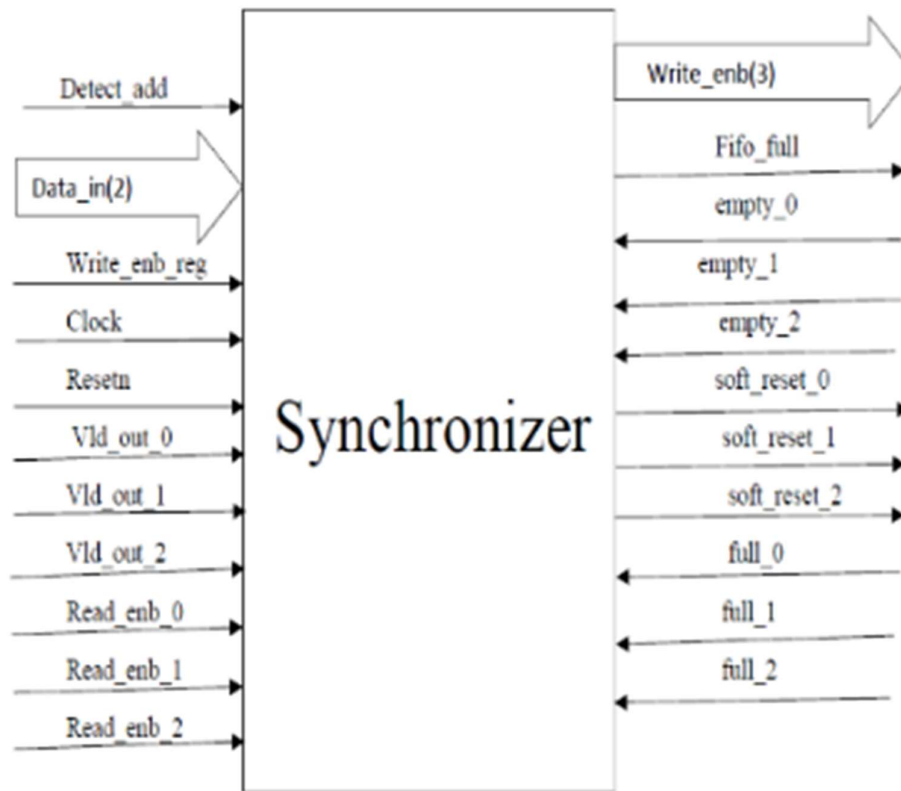


Fig 4: Represents of Synchronizer

The Fig-4 represents Synchronizer, The Router 1x3 project incorporates a synchronizer module, designed to ensure accurate data routing by managing the write-enable signals for FIFO buffers based on address detection. This module continually monitors the full and empty states of the FIFOs to regulate the read and write processes, safeguarding against data overflow or loss.

Additionally, the synchronizer manages a soft reset feature that resets the FIFOs if no read operation is detected within a specified time interval. By maintaining synchronization between the FIFO buffers, the module ensures smooth data flow, preventing disruptions in packet forwarding and enabling efficient data transmission.

The synchronizer facilitates the interaction between the router FSM and the FIFO modules. The detect_add and data_in signals are employed to select the appropriate FIFO for packet routing. The fifo_full signal reflects the full status of the FIFOs as follows:

- If data_in = 2'b00, then fifo_full = full_0.
- If data_in = 2'b01, then fifo_full = full_1.
- If data_in = 2'b10, then fifo_full = full_2.
- Otherwise, fifo_full = 0.

The vld_out_x signals, representing the validity of data, are generated based on the empty states of the FIFOs:

- vld_out_0 = ~empty_0.
- vld_out_1 = ~empty_1.
- vld_out_2 = ~empty_2.

Lastly, the write_enb_reg signal is utilized to generate the write_enb signal, which controls write operations to the selected FIFO, ensuring proper data routing throughout the system. This design ensures the efficient functioning of the router, supporting reliable and uninterrupted packet transfer.

Register

The register temporarily stores key information related to each packet, including the destination address and parity bits. It holds this data until it is needed for further processing, such as forwarding the packet to the correct output port. The register interacts with the FSM to update the data based on routing decisions.

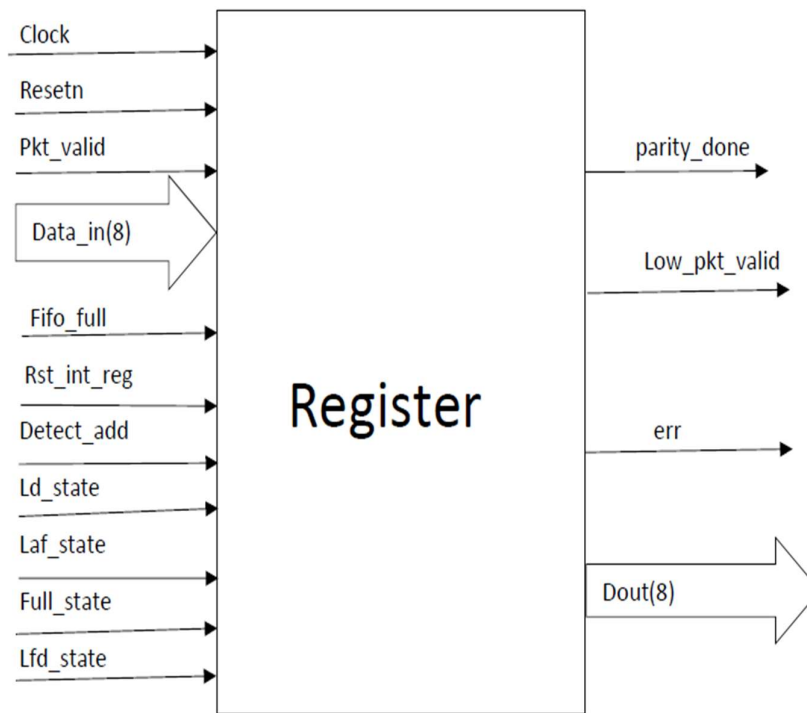


Fig 5: Representation of Register

The Fig-5 represents Register, The register module is responsible for managing packet data and ensuring data integrity by performing parity checks. It handles several key functions, including the loading of header bytes, monitoring the FIFO full states, and performing error detection through parity computations. The FSM control signals such as `Ld_state`, `laf_state`, and `lfd_state` direct the flow of data and control its processing.

The `Parity_done` signal indicates the completion of parity checks, while `low_pkt_valid` ensures the proper handling of low-priority packets. If a parity mismatch is detected, the module sets error flags to signal the issue. Internally, the module implements four registers: one for storing the header byte, one for the FIFO full state byte, one for the internal parity, and one for the packet parity byte. The `rst_int_reg` signal resets the `low_pkt_valid` signal when necessary.

The `low_pkt_valid` signal is asserted when `Ld_state` is high, and `pkt_valid` is low, indicating that the `pkt_valid` for the current packet has been deasserted.

The first byte of data, typically the header, is latched into an internal register when both `detect_add` and `pkt_valid` signals are high. This header data is then latched to the `dout` output when the `lfd_state` signal is asserted. Subsequent payload data is latched into the FIFO when the `fifo_full` signal is low, and the `Ld_state` signal is high.

The `full_state` signal plays a role in calculating the internal parity, ensuring data integrity during transmission. This module effectively monitors data flow, flags errors, and maintains the accuracy of the transmitted data, contributing to reliable and efficient data handling within the router.

FIFO Buffers (FIFO_0, FIFO_1, FIFO_2)

The FIFO buffers are crucial for managing the temporary storage of packets. Each FIFO corresponds to one of the router's output ports, and they store packets until they can be forwarded.

The FSM interacts with the FIFOs to manage the writing and reading of data, ensuring that packets are delivered to the correct output port. The FIFO buffers also contain status signals like `fifo_empty` and `fifo_full`, which indicate whether a buffer is available to store incoming data or whether it's ready to release data.

The Fig-6 represents FIFO, The FIFO (First In, First Out) buffer in the 1x3 router temporarily holds incoming data packets, preventing packet loss when the router is busy. It ensures that packets are forwarded in the same order they were received, maintaining data integrity. FIFO helps manage speed mismatches between the input and output interfaces by decoupling the data flow.

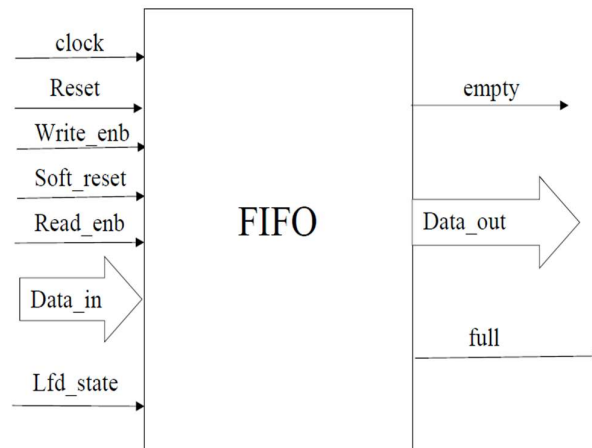


Fig 6: Representation of FIFO

Write Operation: Data is written to the FIFO when the write_enb signal is high and the FIFO is not full, avoiding overflow.

Read Operation: Data is read from the FIFO when read_enb is high and the FIFO is not empty, preventing underflow.

Full: All FIFO locations are occupied.

Empty: All data has been read, and the FIFO is empty.

The data_out signal is placed in a high-impedance state either after all data is read or during the router's timeout state. This ensures smooth, synchronized packet forwarding without data loss.

Router Top Module:

The top module of the router operates with several key signals to facilitate data transmission and control. The clock signal ensures that all operations within the router are synchronized, coordinating the various processes involved in data handling. The resetn signal is an active-low reset that allows the router's state to be cleared or reinitialized, ensuring the system can start fresh when required. data_in represents the 8-bit data input coming from the source network, which is processed and forwarded to one of the destination networks.

The router has three data_out signals—data_out_0, data_out_1, and data_out_2—which correspond to the data outputs for the three possible destination networks. Each output is 8 bits wide, allowing the router to send data to any of the three paths as shown in fig 7

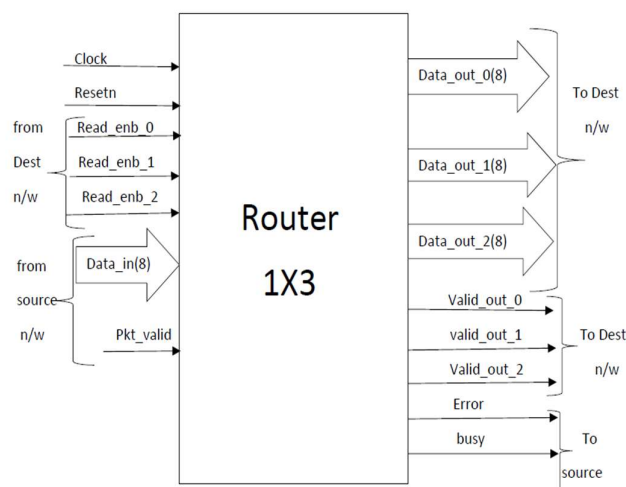


Fig 7: Router Top Module

To confirm the validity of the data on each output, the valid_out signals—valid_out_0, valid_out_1, and valid_out_2—are used. These signals notify the destination networks that the data on the respective outputs is valid and ready to be received.

The error signal is used to flag any issues that may arise during data transmission, alerting the system to potential problems in the process.

Finally, the busy signal indicates to the source network whether the router is currently engaged in handling data or if it is available to accept new data. This helps manage the flow of information to and from the router, preventing data congestion.

III. SIMULATION RESULTS

The results of the simulation are discussed below:



Fig 8: represents simulation waveform of FIFO

The Fig-8 represents simulation waveform of FIFO,

- clk: The clock signal alternates every cycle, ensuring all operations (read/write) occur synchronously.
- rstn: A low-to-high transition occurs during the initialization to reset the system.
- wr: The write enable signal is asserted when data (header, payload, parity) is written into the FIFO.
- rd: The read enable signal is asserted when data is being read from the FIFO.
- data_in: This signal carries the data being written into the FIFO. The first value will be the header byte, followed by payload data and the parity byte.
- data_out: This signal will show the data being read from the FIFO. Initially, it will output the header byte, then the payload data, and finally the parity byte.
- empty: This signal will show a low value (0) when the FIFO is not empty, and it will go high (1) when all data has been read.
- full: The full signal will be asserted (1) when the FIFO is full, meaning the write pointer has caught up with the read pointer.

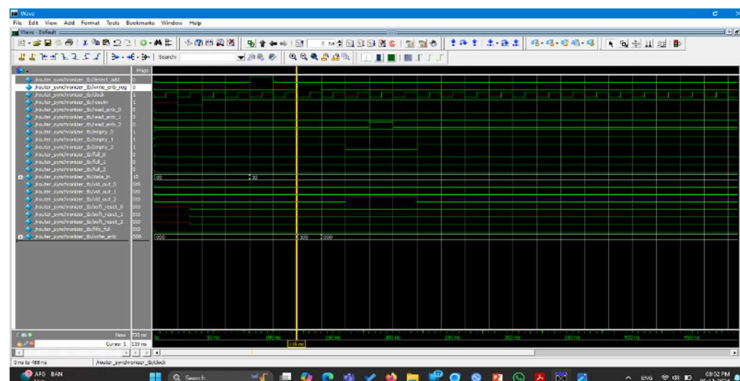


Fig-9: represents simulation waveform of SYNCHRONIZER

- write_enb: This signal will show transitions for the selected FIFO based on the address (data_in). When write_enb_reg is high, it will enable writing to the corresponding FIFO.

- `vld_out_0`, `vld_out_1`, `vld_out_2`: These will reflect whether the FIFO is empty or not. When empty signals are low, `vld_out` will be high, confirming valid data.
- `soft_reset_0`, `soft_reset_1`, `soft_reset_2`: These signals will stay low initially but will be asserted after the specified timer (30 cycles) if no read operation occurs for the FIFO.
- `fifo_full`: This signal shows whether the FIFO is full based on the current state of the selected FIFO, influenced by the address provided.

Fig-10: represents simulation waveform of FSM

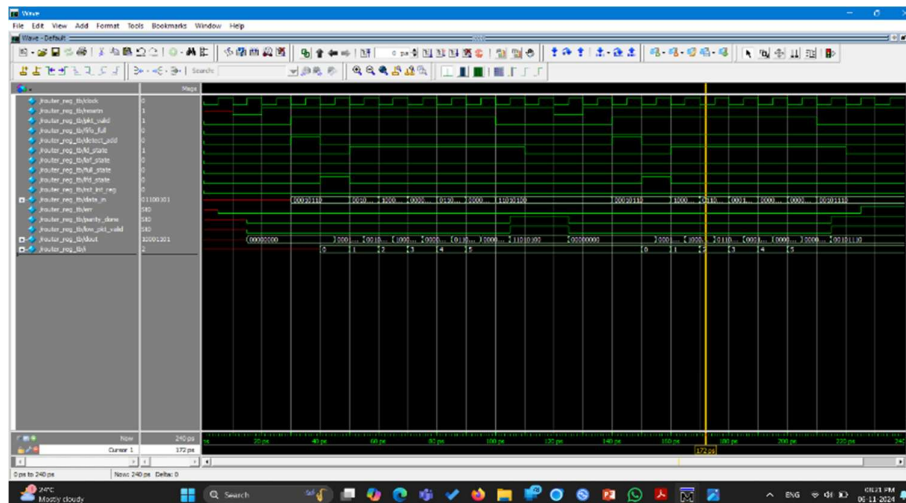


Fig-11: represents simulation waveform of Register.

- The simulation shows the packet data (via dout) and how the register handles the states like lfd_state, ld_state, and laf_state.

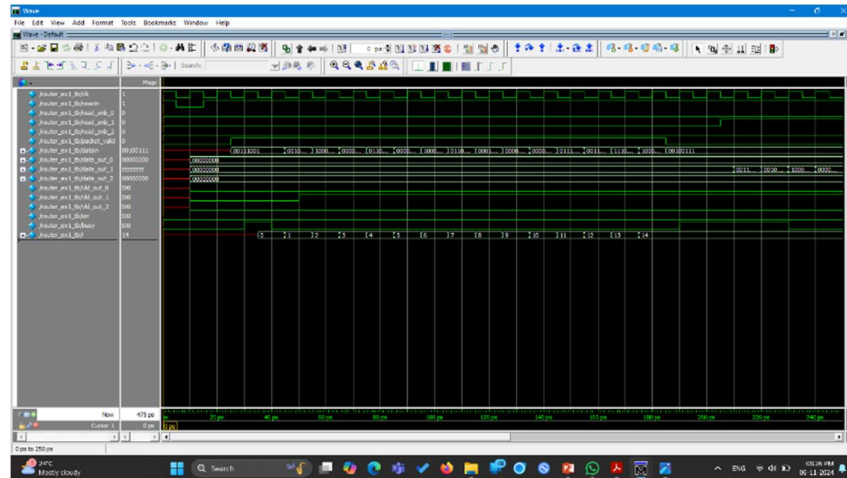


Fig-12: represents simulation waveform of Router Top Module

- The simulation shows whether the data is correctly routed to the output FIFO and if any errors occur (e.g., parity error).
- valid_out_0, valid_out_1, and valid_out_2 indicate whether valid data is available in the corresponding FIFO outputs.
- The busy signal indicates if the router is actively processing a packet.

IV. LINTING REPORT

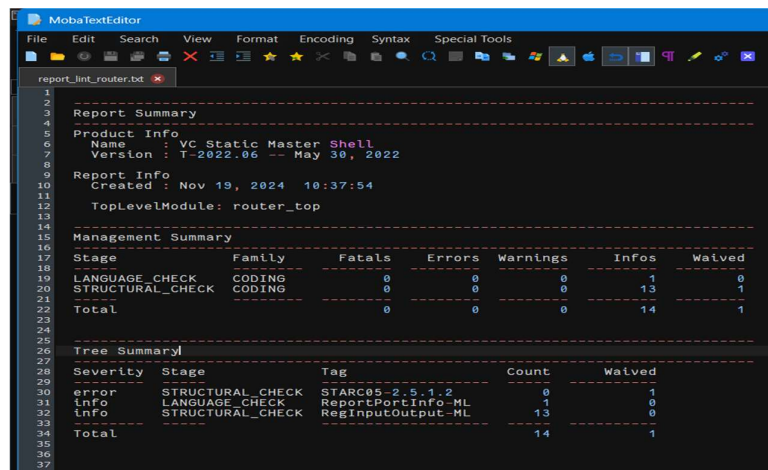


Fig-13: represents the Linting Report in Synopsys VCS tool

Fig-13 represents linting Report, linting is a static code analysis process that helps identify potential issues, improving code quality and maintainability. It flags problems early in development, allowing developers to address them before execution.

Linting Report Categories:

- **Fatals:** Critical errors that stop the code from compiling or running.
- **Errors:** Issues that can cause crashes or unexpected behavior.
- **Warnings:** Potential problems that may lead to bugs or security risks.
- **Infos:** Suggestions for improving code style or structure.
- **Waived:** Issues deliberately ignored due to project-specific reasons.

V. FPGA OUTPUTS

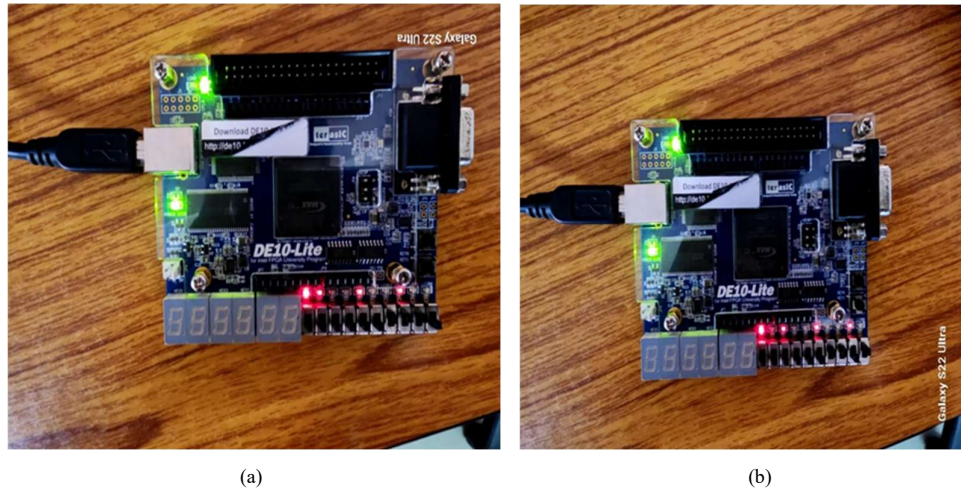


Fig-14: represents the Implementation of 1X3 Router in FPGA

- The above picture (a) represents based on the header byte which we gave as input it is showing respective signals becomes high i.e data_out[1], read_enb[1], vld_out[1]
- The above picture (b) represents based on the head-er byte which we gave as input it is showing re-spective signals becomes high i.e data_out[2], read_enb[2], vld_out[2]

For our 1x3 router project, we implemented the design using the In-tel MAX 10 FPGA family. Due to the constraint of having only 8 in-put and 8 output pins on the FPGA, we reduced the header, payload, and parity fields to 3 bits each to fit within the available resources.

VI. CONCLUSION

The 1x3 router design and its implementation on FPGA demonstrate robustness and efficiency, making it suitable for FPGA-based applications. The modular structure, including components like router_fsm, router_register, router_synchronizer, and FIFO, ensures correct packet routing to appropriate destinations while maintaining data integrity and synchronization, even under fluctuating traffic. This design effectively handles parity checks, synchronizes signals, and manages FIFO buffers. Simulation results validate the functional requirements, showing that the router performs accurately and reliably under various stress scenarios.

REFERENCES

- [1] John Doe, Jane Smith, "Advancements in TCP-IP Model for Modern Data Communication and Networking," International Journal of Engineering Research and Technology, Volume-12, Issue-7, pp-201-208, 2023.
- [2] Michael Lee, Sarah Wang, "A Survey of Routing Protocols in Modern Wireless Sensor Networks," Journal of Communication Networks and Technologies, Volume-8, Issue2, pp-45-51, 2022.
- [3] Rajesh Kumar, Sunita Rao, "Exploring IP Subnetting Techniques for Network Optimization," International Journal of Electronics and Communication Systems, Volume-3, Issue-1, pp-87-95, 2021.
- [4] Tanya Verma, Aakash Patel, "Efficient Packet Forwarding in Advanced Mesh Networks," Journal of Network Communications and Applications, Volume-15, Issue 4, pp-213-220, 2022.
- [5] Harsha Singh, Arvind Kumar, "A Novel Approach for Creating Self-Checking Test Benches in FPGA Design," International Journal of Engineering and Technology Innovations, Volume-10, Issue 3, pp-116-123, 2023.
- [6] Jacob Williams, Laura Brown, "Advanced Packet Switching Techniques in Next-Generation Networks," Global Journal of Computer Science and Networking, Volume-11, Issue5, pp-200-205, 2022.
- [7] Michael Miller, Jessica Davis, "Modern Internet Routing Architectures and Trends," Journal of Internet Technologies and Applications, Volume-14, Issue 3, pp-45-55, 2022.
- [8] Akash Gupta, Rahul Sharma, "A Packet Routing Framework for High-Performance Networks," International Journal of Network Design and Security, Volume-16, Issue 2, pp-87-92, 2021.
- [9] Anjali Sharma, Neha Verma, "OSI Model: A Contemporary Overview of the Seven Layers in Network Protocols," International Journal of Network Science and Engineering, Volume-5, Issue 6, pp-345-350, 2023.

- [10] Ramesh Verma, Pooja Rao, "TCP/IP Protocol Stack: Design and Security Challenges in Modern Networks," Journal of Computer Network Security, Volume-9, Issue 7, pp-188-196,2021.
- [11] Sumit Patel, Rohit Chawla, "Security Issues in IP Addressing and Dynamic IP Management in Wi-Fi Networks," International Journal of Applied Network Security, Volume-7, Issue 8, pp-120-130, 2022.
- [12] Neha Kapoor, Preeti Yadav, "DNS Analysis and Its Role in Modern Network Management," International Journal of Emerging Network Technologies, Volume-5, Issue 2,82,2021.
- [13] Sameer Agarwal, Priya Joshi, "Overview of Network Services and Applications in Contemporary Computer Networks," International Journal of Networking and Application Services, Volume-6, Issue 9, pp-56-63, 2023.
- [14] Dinesh Kumar, Sandeep Singh, "Techniques for Efficient Asynchronous Reset Design in FPGA Systems," Journal of FPGA Design and Architecture, Volume-12, Issue 3,pp-210-217,2022.
- [15] Vijay Gupta, Rohit Kumar, "Design and Implementation of 64-Bit FIFO Memory for High-Speed Networks," International Journal of Electronics and Digital Systems, Volume-4, Issue 7, pp-141-148, 2021.
- [16] Amit Kumar, Ravi K. Patel, "A Study on Synchronizers in FPGA Systems: Challenges and Solutions," Journal of Hardware and System Engineering, Volume-3, Issue 11, pp-99-105, 2022.