Congestion Aware Priority-based Routing using On-Chip Memory for NoC Architecture

Malathi Naddunoori [1]*M. Devanathan [2]

[1] Research scholar, Department of ECE, REVA University, Bengaluru, India, Email id: n.malathiraj@gmail.com

[2] Associate Professor, Department of ECE, REVA University, Bengaluru, India, Email id: devanathan.m@reva.edu.in

KEYWORDS

Arbiter, Congestion aware priority-based routing using on-chip memory, Level-encoded dual-rail, Network on chip, System on chip.

ABSTRACT

Field Programmable Gate Array (FPGA) involves an array of programmable logic blocks and configurable interconnects which allows users to establish digital circuits and specialized functionality. The Network on Chip (NoC) is a router-based packet-switching network among System on Chip (SoC) modules. However, the node processes incoming data, routes operations, and transmits it to the next node before storing the output in a buffer to avoid congestion. This approach, though effective in managing data flow, increases both area and power consumption. To overcome this problem, the Congestion Aware Priority-based Routing using On-Chip Memory (CAPROCM) is proposed to minimize area and power consumption by removing crossbar switches and Input/Output (I/O) buffers. Initially, the priority encoder determines the priority of the packet and encodes the priority information to an arbiter. The arbiter ensures that the packet with a high priority is processed first. Then, the XY routing process is generated to evaluate the routing path based on the packet's address. Level-Encoded Dual-Rail (LEDR) is utilized for transferring data and ensures error correction and detection. CAPROCM is evaluated with Look Up Table (LUT), Flip Flop (FF), I/O, Global Buffer (BUFG), Bonded Input/ Output Block (Bonded IOB), Slice registers, and Power. While compared to the existing methods like NoC-based hardware-software co-design model, the proposed CAPROCM achieves better power consumption of 0.324 W for Virtex-7 XC7VX690-3 FPGA device.

1. Introduction

Field Programmable Gate Array (FPGA) has significantly advancement in both heterogeneity and capacity over the past few decades. FPGA has numerous embedded hard blocks like fracturable multi-precision multipliers, on-chip memories, and high-speed transceivers for enhancing their efficiency besides their soft programmable routing and logic system [1]. Recently, the advancement of processors and System-on-Chip (SoC) and Multi-processor SoC (MPSoC) has accomplished a new qualitative level that is an integration of Network on Chip (NoC) [2] [3]. For various core systems, NoC has become a vital on-chip communication model because it enables parallel processing by generating low latency and high bandwidth [4] [5]. NoCs are constructed by utilizing regular or irregular topologies and mesh is a primary regular topology established by effectively interrelating the nearby routers [6] [7]. The switching, topology, and routing approaches are significant factors to consider while constructing a NoC architecture because they directly impact the scalability and efficiency of the interconnected network in the chip. The routing process establishes the most efficient path for messages and data to be transmitted from the communicating device to target receiver [8]. The routers route packets that are received from source cores to destination cores through links in NoC. The routing approach embedded within the router is responsible for forwarding the packets which play a primary role in packet delivery [9] [10].

NoCs architecture is established to address the need for flexibility and efficiency within a system [11]. The routing approach operates the messages to destinations by minimal, effective, and low paths which consider congestion control policies [12]. Unicast-based, tree-based, and path-based are the three routing techniques for multicasting. A broadcast or multicast packet is rendered as numerous

packets at the source node in the unicast multicasting technique [13]. A message is rendered at every branch in the tree-based multicasting technique; hence, numerous versions of message are transmitted by the tree [14] [15]. The tree-based routing technique performance is minimized in high-network load which results in high area overhead because of using virtual and physical buffers. Every source router generates a packet to be transmitted to a predetermined destination in a path-based multicasting routing technique with a destination list inserted in the meta-data of the packet header [16] [17]. However, the node processes the incoming data, routes the operations, and transmits it to next node before storing the output buffer to avoid congestion which leads to high area and power consumption. To solve this issue, the CAPROCM is proposed to minimize area and power consumption by removing crossbar switches and I/O buffers.

The main contribution of this research is as follows:

- The priority encoder analyze the priority of packets and ensures that the packet with the highest priority is processed first by the arbiter. XY routing process is generated which determines the routing path and then LEDR transfers the data effectively.
- By executing these processes, a CAPROCM is effectively designed by using on-chip memory which reduces area and power consumption.
- The main aim of CAPROCM is to evaluate with Virtex 7 – xc7vx458t FPGA device. Additionally, the proposed architecture is determined with five FPGA devices like Virtex-7 XC7VX690-3, Xilinx LX240T, Xilinx LX760, Artix-7 XC7A100T FPGA, and Spartan 3 XC3S400 FPGA devices.

This research paper is organized as follows: Section 2 determines the literature survey of the existing method. Section 3 explains a detailed description of the proposed architecture. Section 4 evaluates the simulation results. Section 5 summarizes the overall part of the conclusion.

2. Literature survey

The related work about NoC architecture was discussed in this section along with their advantages and disadvantages

Mazumdar [18] suggested a NoC-based hardware-software co-design architecture to manage the thread data flow. Initially, a hash-based thread distribution model was used to support the distribution of an enormous number of synchronized threads across available cores. Then, the lightweight model was applied to increase the router's microarchitecture and NoC topology was enabled to control by core instructions directly. By employing a hybrid architecture which was a combination of 2D mesh and rings, the NoC's scalability performance was enhanced. However, NoC-based hardware-software co-design suffered from high power consumption due to complex routing and frequent data transmission.

Singh [19] implemented a Multi-Priority-based iterative round-robin matching with Slip (MPiSLIP)-based NoC for MPSoC and Multi-Core SoC (MCSoC) systems. The MPiSLIP was utilized to compute the various priorities available for the routing process without congestion based on grant arbiters, priority encoders, accept arbiters, decision registers, and priority selector decisions. Look ahead By Route Computation (LBRC) was utilized to determine the effective path among input to output port. LBRC depended on two-hop neighbour router status generation, next router address, and adaptive route computation which enhances model performance. However, MPiSLIP suffers from high area and power consumption due to enlarged switching activity over multiple priorities.

Parane [20] presented a Performance Evaluation and design-based exploration of NoCs (P-NoC) using FPGA for chip multiprocessor architecture. P-NoC involves a fully parametrized router, topology, receptor module, and traffic generation. The ML-mesh NoC architecture was considered by different configurable parameters of NoC to evaluate the trade-off for mesh. The input First In First Out (FIFO) of the NoC router was determined by P-NoC which minimizes the logic circuit complexity. However, P-NoC struggles with high power consumption due to increased switching activity and delay because of inherent routing complexity in FPGA device.

Shahane [21] developed a small side buffer in iterative Serial in Line Protocol (iSLIP) for a single node NoC router. The iSLIP scheduler block was a router feature that has a programmable priority scheduler to generate access to the requested port for evaluating a grant to the port. The iSLIP provides a fast scheduling of most of a request which optimizes the router latency. By using a small buffer size model, the router design complexity was minimized effectively with the help of iSLIP. However, small side buffer in iSLIP results in high power consumption because of frequent buffer access and data movement.

Behera [22] introduced a 3D NoC for Embedding-Memory-Management-Unit (MMU) into NoC. The 3D NoC was considered as the unique way of improvement in NoC fabric to acquire a high performance. A FIFO buffer was utilized in NoC routers which assists in storing data packets temporarily. The distributed MMU was established to prevent certain bottlenecks which assists in the management of memory access requests in NoC. However, 3D NoC enhances power consumption due to a longer data path for memory access across various layers in the routing and data transmission process.

From the overall evaluation, the existing techniques had limitations like suffering from high power consumption and area which leads to suboptimal performance. To solve this problem, the CAPROCM is proposed by incorporating on-chip memory which minimizes area and power consumption by eliminating crossbar switches and I/O buffers. By eliminating these components, CAPROCM simplifies data routing which results in more effective processing.

3. Proposed CAPROCM architecture

In this research, the CAPROCM is proposed for NoC architecture which consumes less area and memory consumption. In the proposed router model, the crossbar switch and I/O buffers are removed to minimize power consumption. Fig. 1 shows the 4x4 mesh NoC architecture.

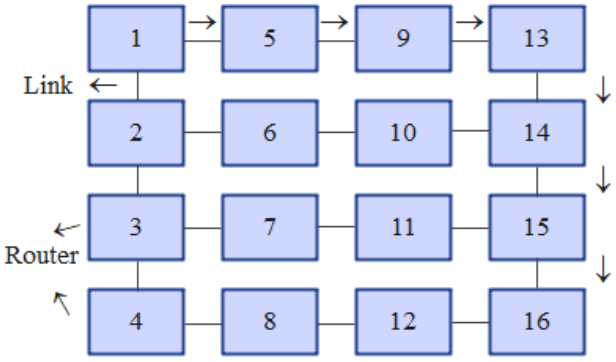


Figure. 1 4x4 mesh NoC architecture

The workflow for the proposed architecture is as follows:

- Initially, the 48 bits of data packets are fed as input to the priority encoder block. This block determines the priority of the packet depending on predefined criteria and encodes the priority information to the next stage arbiter.
- The arbiter receives a multiple packet and selects which packet to forward depending on priority information provided by the priority encoder. The arbiter ensures that the packet with a high priority is processed first.
- Then, the XY routing process is generated which has two decision-making phases to evaluate the routing path depending on the packet's address. The first decision determines the X address whether matches the target address, and the packet is transmitted to the next column (Y). If not, it moves to the next row.
- The second decision again determines the X address. If the target X address is reached, the packet progresses to the LEDR encoding stage. Otherwise, it processes routing to the next column.

- LEDR is utilized for robust data transfer and ensures error correction and detection. During this process, the packet size is minimized to 32 bits and then the output of LEDR is stored in the on-chip memory.
- By using a priority encoder and FSM-based arbiter, the long critical path problem is solved effectively by changing the port priority. Therefore, Starvation and Dead/live-lock problems are solved and enhance the area consumption.

3.1 NoC architecture

NoC architecture is a scalable and modular communication framework designed to manage the data transfer among different components of System-on-Chip (SoC). Artificial Intelligence SoC (AI SoC) establishes various instances of the same hardware type in a grid and utilizes regular topologies like mesh, ring, or torus. It assists in ensuring predictable data flow, minimizes cost, and provides effective design in scalability performance. Reconfigurable FPGAs play a significant role in evolving chip technology. In this research, priority-based routing is proposed by using on-chip memory on an FPGA device to minimize power consumption and area. Fig. 1 represents a 4x4 mesh NoC architecture with 16 Asynchronous Buffer-less NOC Routers. Every router excluding routing at the edge has 5 ports and 4 directions like West, East, South, and North and 1 local port is used to connect the processing core. Each router is associated with the communication link effectively. The overall number of links in mesh is represented using equation (1)

$$TL_{Mesh} = 2\overline{w}\sqrt{n}(\sqrt{n} - 1) \quad (1)$$

The average distance among two nodes in a mesh is represented in equation (2)

$$\overline{Dist_{Mesh}} = \frac{2}{3}\sqrt{n} \quad (2)$$

Where TL_{Mesh} represents total length in the context of mesh topology, $\overline{w}$ indicates weighted or scaled components, and n determines number of nodes. The brief discussion about the routers is explained below.

3.2 Congestion Aware Priority-based Routing using On-Chip Memory (CAPROCM)

Fig. 2 indicates the block diagram for the CAPROCM architecture. The I/O buffers and crossbar switches are removed which minimizes area and consumption of power. There is no requirement for virtual channels because it is a buffer-less model with an equal number of I/O ports. A packet that is incoming router is transmitted to the associated ports which become the router model as simple, fast, effective area and power.

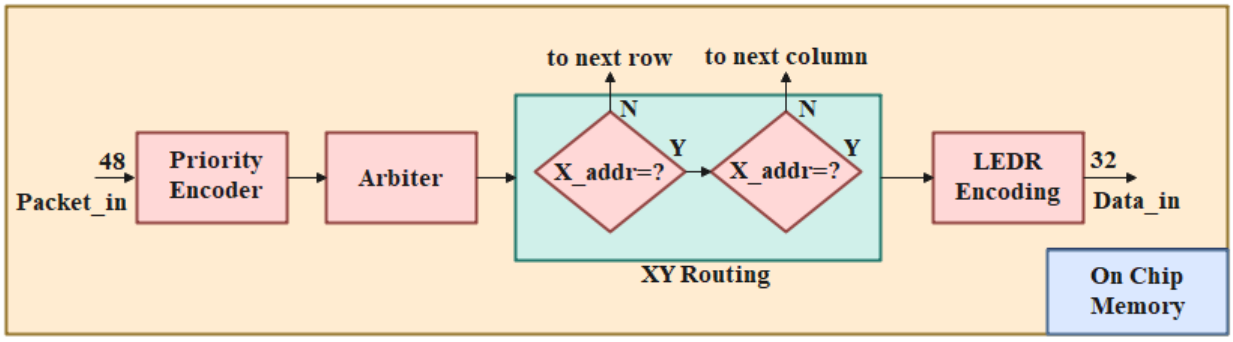


Figure. 2 Block diagram for the proposed CAPROCM

3.2.1. Priority encoder

In NoC architecture, a priority encoder is utilized to handle the order of packet transmission depending on their priority levels. It provides priority to incoming requests which ensures that high-priority packets are routed initially. This model is significant for maintaining timely communication and effectiveness within the network. When transmitting the network packets, the hardware control is to be selected by configuring hardware to generate high-priority packets which provides more flexibility in NoC. In this research, the software control is used via a priority encoder. When multiple requests are established for same, the arbiter is responsible for managing and allocating the port efficiently. A Finite State Machine (FSM)-based arbiter is designed with a priority encoder instead of a hard-wired switch model. It assigns priority in a clockwise direction which is associated with core/ processor to East, west, North, and South. The directional priority assignments assist in fast scheduling instead of traditional round-robin design. Therefore, an arbiter block is designed to manage the priority of various ports dynamically.

3.2.2. Arbiter

A priority encoder determines directional priority whereas the arbiter modifies port priority dynamically which means that the port is allocated with high priority by the priority encoder. In a router, a priority encoder and arbiter are established in that it needs an additional block to compute subsequent priority generation blocks. If input from every port is designed for various ports, and then the packet is transmitted to the associating port without any problem. Initially, the arbiter checks the port priority and begins with a port with high priority. Then for each transfer, the request is specified which is provided to the least priority in the next arbitration round. Instead of using credit management and assignment blocks, a priority block is utilized that allocates priority and certain internal registers that maintain the port track. The arbiter maintains the port track which is served according to changing the dynamic port priority. Then, the arbiter input is fed into routing control for further processing.

3.2.3. Routing controller

After the arbiter process, the XY routing model is utilized in the routing controller which is simple and appropriate for both irregular and regular topologies. An XY routing model follows the shortest path and ensures a design Dead/Live Lock-free. The initial block determines an X address, and the next block evaluates a Y address. After the packet is chosen by the arbiter, the unit of the routing controller transmits a packet to the destination by extracting a data address from the packet header and determining the destination address. The address of X is compared with the present address whether it matches a packet until the address of Y is matched. If not, then the packet is transmitted to a subsequent row without generating a Y which saves time for processing. Then, the packet is transmitted to the next row and this process is repeated until the address matches X. Once the address X is matched, the subsequent block compares a Y address with the present address. Following this, the data is processed through the LEDR procedure to ensure effective data transfer.

3.2.4. LEDR

After eliminating header data, the packet is transmitted to the LEDR block. The primary goal is to ensure asynchronous by utilizing two-stages of non-return to zero encoding. This provides enhanced performance with respect to power and area because it removes the space among two consecutive data. During this process, the packet size is minimized to 32 bits which enhances robustness of data transfer and error detection before storing the data in on-chip memory.

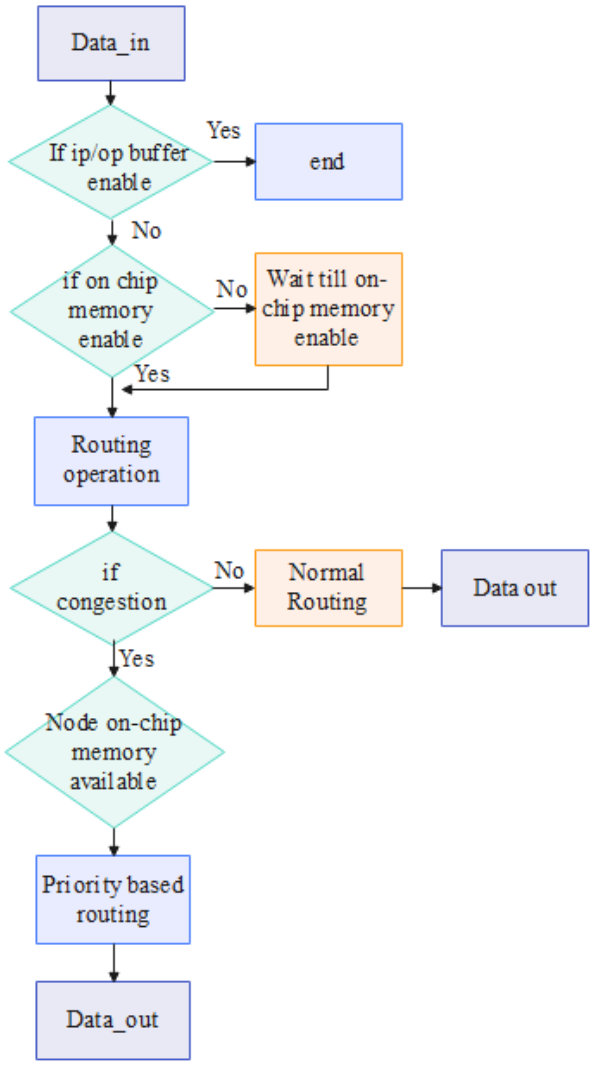


Figure. 3 Flowchart for the CAPROCM

Fig. 3 shows the flow chart for the proposed method. It illustrates a data routing process that starts with input data and verifies whether the I/O buffer is enabled. If enabled, the process ends. Otherwise, it checks next process if on-chip memory is enabled and once it is enabled, a routing operation is performed. Then, the process checks for congestion if normal routing occurs which leads to data output. If congestion is detected and on-chip memory is available, priority-based routing is performed followed by data output. This ensures data routing by using on-chip memory and priority management during congestion.

3.2.5. On-Chip memory

The output of LEDR encoding is a 32-bit data packet which is stored using on-chip memory. On-chip memory is defined as the storage of memory that is integrated directly into a processor or microchip. It provides effective and rapid access to data which minimizes delay compared to external memory. Table 1 represents the on-chip memory register map. It organizes data with each address storing a specific data value and making rapid retrieval and processing which is significant for processing data in NoC systems.

Table 1. On-chip memory register map

Address	**Data**
0	Data 0
1	Data 1
2	Data 2
. .	. .
32	Data 32

By using an on-chip memory, less area and power are consumed compared to off-chip alternatives. The data rate varies for each node based on signal processing. In nodes that don't use priority, the on-chip memory is utilized, resulting in reduced congestion. This process has become effective and performs a buffer-less FSM state when the node is ideal. The proposed architecture block can enhance efficiency and flexibility in handling complex data routing tasks by using on-chip memory. These elements make an effective resource allocation, accurate control over data path, effective arbitration of conflicting requests, and robust error correction and detection capabilities. This architecture enhances overall system performance by reducing data transfer and ensuring rapid access times which is significant for real-time applications like data streaming.

4. Simulation results

The proposed CAPROCM architecture is simulated using Xilinx ISE 14.2 software to evaluate the proposed architecture simulation results. The primary goal is to determine the proposed architecture using Virtex 7 – xc7vx458t FPGA device. Additionally, it is analyzed with five FPGA devices like Virtex-7 XC7VX690-3, Xilinx LX240T, Xilinx LX760, Artix-7 XC7A100T FPGA, and Spartan 3 XC3S400 devices.

4.1 Performance analysis

The CAPROCM architecture is evaluated using five different FPGA devices. The CAPROCM is evaluated with LUT, Flip Flop FF, I/O, LUT Random Access Memory (LUTRAM), BUFG, Bonded IOB, Slice registers, power, and Block Random Access Memory (BRAM). BUFG ensures even clock distribution across FPGA which is significant for synchronizing each module effectively. It handles various clock domains and enables a smooth process of asynchronous data streams. BUFG prevents data corruption and timing errors by reducing clock skew which is essential for managing data integrity in high-speed operations. Bonded IOBs act as an interface between internal logic and external signals of FPGA. It ensures that incoming and outgoing signals meet the required voltage for suitable data interpretation. Also, it manages different I/O protocols which makes the NoC architecture useful and capable of interfacing with various external devices and networks.

Fig. 4 determines the performance of NoC Arithmetic Logic Unit (ALU) routing. Effective NoC ALU routing is significant in maintaining high performance in complex SoC and high-speed processors. It determines the data processing efficiency within multi-score systems. NoC ALU increases the pathways for data transfer and speed which minimizes latency by ensuring minimal congestion and leads to better performance.

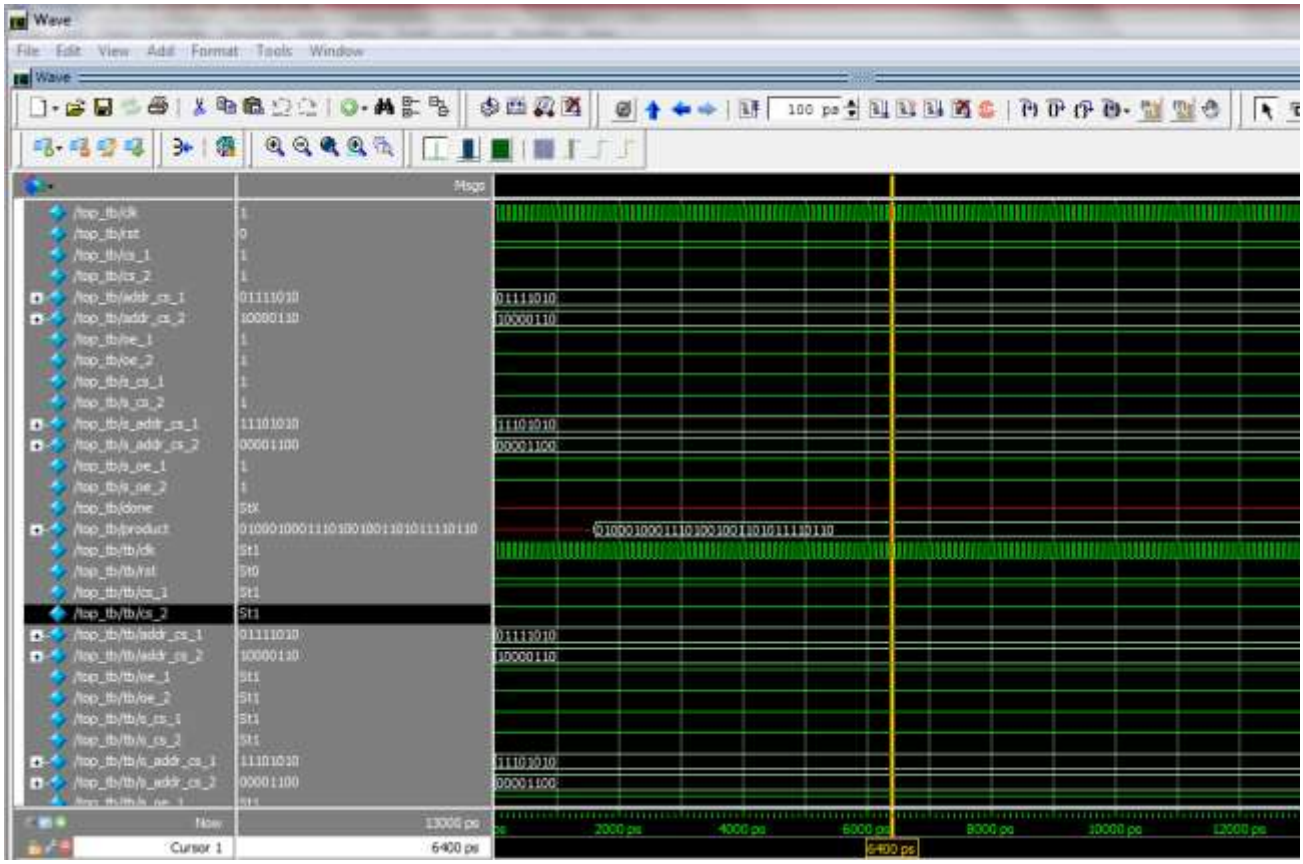


Figure. 4 Performance of NoC ALU routing

Fig. 5 demonstrates the performance of router path delay. Path delay refers to the time taken for data packets to travel from source to destination by different routers. Lower path delays result in rapid data transfer and minimized latency which is essential for managing better performance in communication networks. The router path delay reduces bottlenecks, increases routing process, and enhances hardware resources to make rapid and reliable data delivery.

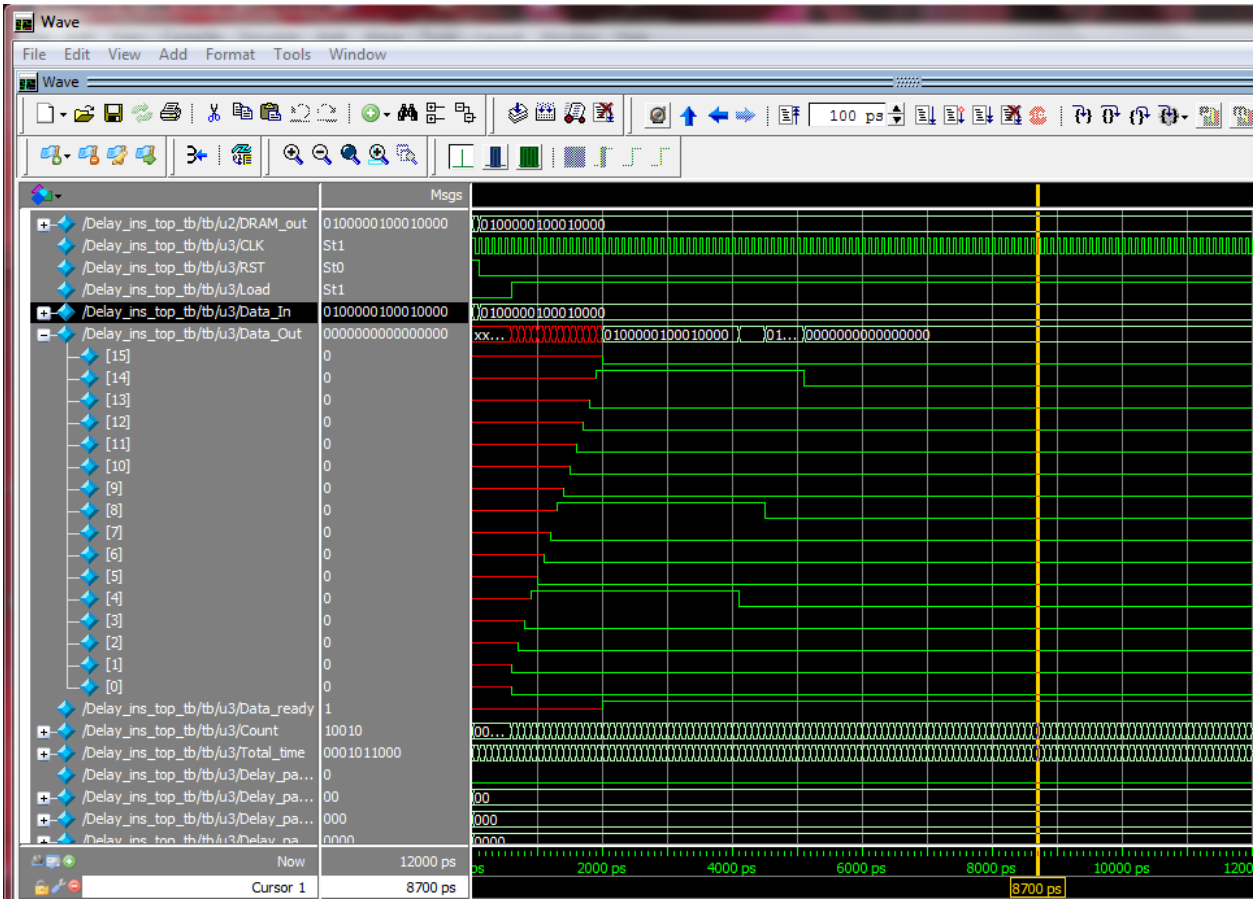


Figure. 5 Performance of router path delay

Fig. 6 shows a performance of priority tasks-based (tb) in NoC architecture which plays a vital role in maintaining resource utilization. It represents processors P1 to P8 which generates requests. The count of requests from P1 is stored in count 1 which occurs 3 times. Similarly, P2 is count 2, P3 is count 3 and up to P8 is count 8. This analysis determines how often each processor generates requests. Among counts 1 to 8, counts 5, 7, and 8 are the least priority because it has only one request. Therefore, the 5, 7, and 8 processor's on-chip memory exhibits low priority. By using that on-chip memory, other nodes are executed to remove congestion.

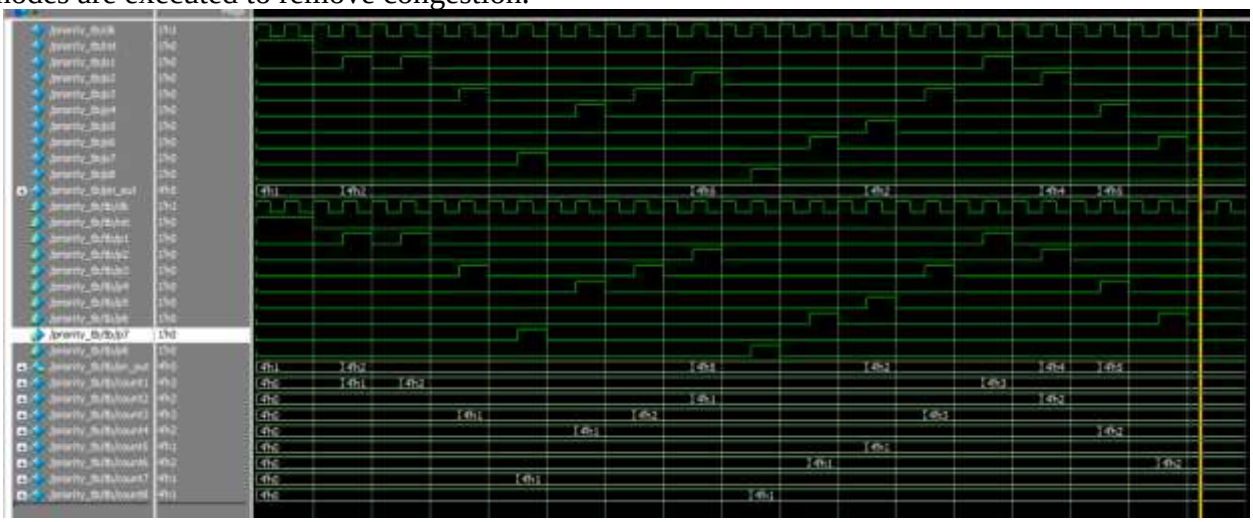
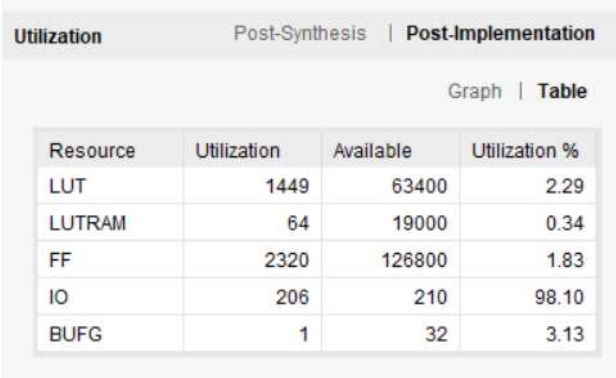


Figure. 6 Performance of Priority tb

Fig. 7 represents the performance of the area for the proposed CAPROCM using Virtex 7 – xc7vx458t FPGA device. The representation evaluates that the proposed CAPROCM obtains a resource utilization between 0.34% to 98.10% by adjusting routing paths depending on present network congestion levels and task priorities. This approach improves the use of on-chip memory resources and ensures that significant tasks are effectively processed which leads to minimized area performance.



Resource	Utilization	Available	Utilization %
LUT	1449	63400	2.29
LUTRAM	64	19000	0.34
FF	2320	126800	1.83
IO	206	210	98.10
BUFG	1	32	3.13

Figure. 7 Performance of area for CAPROCM using Virtex 7 – xc7vx458t FPGA device

Fig. 8 depicts the performance of power analysis for CAPROCM using Virtex 7 – xc7vx458t FPGA device. The result analysis shows that CAPROCM acquires less static power consumption of 0.791 W due to effectively handling data flow and reducing idle times in network elements. The routing data depends on congestion and priority, it minimizes unnecessary movement of data and maintains the network elements more effectively utilized which leads to minimal static power consumption in the system.

Figure. 8 Performance of power for CAPROCM using Virtex 7 – xc7vx458t FPGA device

Table 2 indicates the area performance for a conventional NoC using Virtex-7 XC7VX690-3 FPGA device. It demonstrates that conventional NoC obtains 0.048% to 12.820% of LUT, FF, and BRAM resources. Table 3 represents the area performance for the proposed CAPROCM using Virtex-7 XC7VX690-3 FPGA device. The CAPROCM achieves a lesser area between 0.005% to 7.122% compared to conventional NoC because of its effective utilization of on-chip resources and routing strategies. By adjusting dynamic routing paths depending on task priorities and congestion, CAPROCM reduces the hardware utilization for routing components.

Table 2. Performance of area for a conventional NoC using Virtex-7 XC7VX690-3 FPGA device

Resources	Used	Available	Utilization (%)
LUT	2478	1,952,000	0.126
FF	948	1,952,000	0.048
BRAM	9	70.2	12.820

Table 3. Performance of area for CAPROCM using Virtex-7 XC7VX690-3 FPGA device

Resources	Used	Available	Utilization (%)
LUT	1057	1,952,000	0.005
FF	751	1,952,000	0.038
BRAM	5	70.2	7.122

Table 4 evaluates the frequency performance of conventional NoC and CAPROCM for two FPGA devices. It demonstrates that Xilinx LX240T and Xilinx LX760 FPGA devices obtain a less frequency usage of 124 MHZ compared to conventional NoC due to CAPROCM reduces the requirement of frequent data and network congestion. This results in smoother transmission of data with fewer instances of operating at high frequencies to manage congestion which leads to overall fewer frequency usage.

Table 5 determines the area performance for conventional NoC using Spartan 3 XC3S400 FPGA device. It shows that conventional NoC obtains 1.785% to 62.5% of slice register, LUT, and Bonded IOB resources. Table 6 evaluates the area performance for the proposed CAPROCM using Spartan 3 XC3S400 FPGA device. When compared to conventional NoC, it clearly demonstrates the CAPROCM achieves a lesser area between 1.283% to 46.969 due to on-chip resources and routing strategies. By adjusting routing paths based on task priorities and congestion, CAPROCM minimizes hardware utilization.

Table 4. Performance of frequency for conventional NoC and CAPROCM

FPGA devices	Frequency (MHZ)	
	Conventional NoC	Proposed CAPROCM
Xilinx LX240T	107	124
Xilinx LX760	107	124

Table 5. Performance of area for conventional NoC using Spartan 3 XC3S400 FPGA device

Resources	Used	Available	Utilization (%)
Slice Register	128	7168	1.785
LUT	587	7168	8.189
Bonded IoB	165	264	62.5

Table 6. Performance of area for CAPROCM using Spartan 3 XC3S400 FPGA device

Resources	Used	Available	Utilization (%)
Slice Register	92	7168	1.283
LUT	347	7168	4.840
Bonded IoB	124	264	46.969

Table 7 analysis the performance of area for conventional NoC using Artix-7 XC7A100T FPGA device. It represents that conventional NoC acquires an 0.018% to 19.396% of no. of slice LUT, I/O, and FF. Table 8 determines the performance of area for proposed CAPROCM using Artix-7 XC7A100T FPGA device. The proposed CAPROCM achieves a less performance from 0.011% to 12.5% of no. of slice LUT, I/O, and FF due to CAPROCM employs on-chip resources and routing

strategies. CAPROCM reduces utilization performance by adjusting routing paths depending on congestion and task priorities.

Table 7. Performance of area for conventional NoC using Artix-7 XC7A100T FPGA device

Resources	**Used**	**Available**	**Utilization (%)**
No. of slice LUT	35	101,440	0.034
I/O	45	232	19.396
FF	38	202,880	0.018

Table 8. Performance of area for CAPROCM using Artix-7 XC7A100T FPGA device

Resources	**Used**	**Available**	**Utilization (%)**
No. of slice LUT	15	101,440	0.014
I/O	29	232	12.5
FF	24	202,880	0.011

Table 9 depicts the performance of power analysis for conventional NoC and CAPROCM. The CAPROCM achieves a lower static and dynamic power consumption of 0.153 W and 0.041 W compared to conventional NoC architecture due to removing crossbar switches and I/O buffers which helps to reduce power consumption.

Table 9. Performance of power for conventional NoC and CAPROCM

FPGA device	**Conventional NoC (W)**		**Proposed CAPROCM**	
	Static Power	**Dynamic Power**	**Static Power**	**Dynamic Power**
Virtex-7 XC7VX690-3	0.258	0.169	0.153	0.041

4.2 Comparative analysis

Table 10 indicates the comparative analysis of existing models for the Virtex-7 XC7VX690-3 FPGA device. The CAPROCM architecture obtains a less resource utilization of 1057 LUT, 751 FF, 5 BRAM, and static and dynamic power of 0.153 W and 0.041 W compared to [18]. Table 11 determines the comparative analysis of existing models of two Xilinx FPGA devices. When compared to [19], the CAPROCM attains a lower frequency of 74 MHZ for Xilinx LX240T and Xilinx LX760 FPGA devices. Table 12 depicts a comparative analysis of existing methods using Spartan 3 XC3S400 FPGA device. The CAPROCM achieves 92 slice registers, 347 LUT, and 124 Bonded IOB compared to [21]. Table 13 illustrates the comparative analysis of existing methods using Artix-7 XC7A100T FPGA device. When compared to existing methods like [20], the proposed CAPROCM achieves a LUT usage of 15 slices. Due to adjusting routing paths based on task priorities and present network congestion levels, the network obtains more effective data transmission and results in less congestion. The proposed architecture enhances the use of on-chip memory resources and ensures that significant tasks are effectively processed. Therefore, this results in better performance based on power consumption and area. By decreasing an unnecessary memory access and using effective data pathways, the proposed approach enables it appropriate for NoC architecture. The proposed CAPROCM obtains higher frequency compared to [19]. Hence, the delay performance is reduced which performs a higher speed for NoC.

Table 10. Comparative analysis of existing methods using Virtex-7 XC7VX690-3 FPGA device

Methods	**LUT**	**FF**	**BRAM**	**Power (W)**	
				Static	Dynamic
NoC-based hardware-software co-design architecture [18]	1358	968	8	0.324	0.075
Proposed CAPROCM	1057	751	5	0.153	0.041

Table 11. Comparative analysis of existing methods using two Xilinx FPGA devices

FPGA devices	**Frequency (MHZ)**	
	MPiSLIP [19]	**Proposed CAPROCM**
Xilinx LX240T	98	124
Xilinx LX760	98	124

Table 12. Comparative analysis of existing methods using Spartan 3 XC3S400 FPGA device

Methods	**Slice Register**	**LUT**	**Bonded IoB**
Small side buffer in iSLIP [21]	124	436	245
Proposed CAPROCM	92	347	124

Table 13. Comparative analysis of existing methods using Artix-7 XC7A100T FPGA device

Methods	**No. of. Slice LUTs**
P-NoC [20]	23
Proposed CAPROCM	15

4.3 Discussion

The advantages of the proposed CAPROCM and the disadvantages of existing models are briefly discussed in this section. The disadvantages for existing models like NoC-based hardware-software co-design [16] suffered from high power consumption because of complex routing and switching mechanisms. MPiSLIP [17] suffers from high power consumption and area because of the frequent arbitration and various priority levels which results in complex control logic. P-NoC [18] struggles with high power consumption due to increased traffic congestion and longer communication paths. A small side buffer in iSLIP [19] limits its ability to manage traffic patterns effectively which leads to potential congestion and increases overall performance in the network. The proposed CAPROCM architecture overcomes these existing model's limitations. Employing on-chip memory for routing decisions depending on congestion levels and task priorities, CAPROCM architecture increases flow of data and minimizes packet delays. This architecture enhances the overall system performance and ensures that ideal for complex multi-core systems where effective resource utilization is significant. Also, the proposed CAPROCM's effective use of on-chip memory results in less power consumption.

5. Conclusion

The CAPROCM is proposed to minimize area and power consumption by removing crossbar switches and I/O buffers. CAPROCM reduces the requirement for external buffers and provides less delay and power consumption by using on-chip memory which results in more effective and reliable network performance. By prioritizing the path of data and using on-chip memory, the proposed CAPROCM avoids the requirements of high crossbar switches and I/O buffers. The CAPROCM improves the deployment of on-chip memory resources and ensures that performance is processed effectively. By dynamically adjusting routing paths based on task priorities and congestion,

CAPROCM reduces hardware utilization for routing components. By performing this process, the proposed CAPROCM obtains a better performance. Therefore, when compared to existing technique like NoC-based hardware-software co-design architecture, the CAPROCM consumes less power of 0.324 W for the Virtex-7 XC7VX690-3 FPGA device. In future, an advanced architecture will be used in NoC to enhance the model performance.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

The paper conceptualization, methodology, software, validation, formal analysis, investigation, resources, data curation, writing—original draft preparation, writing—review and editing, visualization, have been done by 1st author. The supervision and project administration, have been done by 2nd author.

References

[1] A. Boutros, E. Nurvitadhi, and V. Betz, "Architecture and application co-design for beyond-FPGA reconfigurable acceleration devices", *IEEE Access*, Vol. 10, pp. 95067-95082, 2022.
[2] E.V. Lezhnev, V.V. Zunin, A.A. Amerikanov, and A.Y. Romanov, "Electronic Computer-Aided Design for Low-Level Modeling of Networks-on-Chip", *IEEE Access*, Vol. 12, pp. 48750 – 48763, 2024.
[3] P. Agarwal, T.K. Garg, and A. Kumar, "Analysis of 3D NoC router chip on different FPGA for minimum hardware and fast switching", *National Academy Science Letters*, Vol. 47, No. 1, pp.35-39, 2024.
[4] Z. Jiang, X. Dai, R. Wei, I. Gray, Z. Gu, Q. Zhao, and S. Zhao, "NPRC-I/O: A NoC-based Real-Time I/O System with Reduced Contention and Enhanced Predictability", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 12, No. 12, pp. 4629 – 4642, 2023.
[5] P. Bhamidipati, and A. Karanth, "HREN: A hybrid reliable and energy-efficient network-on-chip architecture", *IEEE Transactions on Emerging Topics in Computing*, Vol. 10, No. 2, pp. 537-548, 2022.
[6] S. Jagadheesh, P.V. Bhanu, J. Soumya, and L.R. Cenkeramaddi, "Reinforcement learning based fault-tolerant routing algorithm for mesh based noc and its fpga implementation", *IEEE Access*, Vol. 10, pp. 44724-44737, 2022.
[7] S. Kashi, A. Patooghy, D. Rahmati, and M. Fazeli, "A multi-application approach for synthesizing custom network-on-chips", *The Journal of Supercomputing*, Vol. 78, No. 13, pp. 15358-15380, 2022.
[8] P. Agarwal, T.K. Garg, and A. Kumar, "Electronics Hardware Chip Design for Router–Router Communication", *Proceedings of the National Academy of Sciences, India Section A: Physical Sciences*, Vol. 93, No. 4, pp. 703-710, 2023.
[9] M.T. Balakrishnan, T.G. Venkatesh, and A.V. Bhaskar, "Design and implementation of congestion aware router for network-on-chip", *Integration*, Vol. 88, pp. 43-57, 2023.
[10] N.A. Kumar, G. Shyni, G. Peter, A.A. Stonier, and V. Ganji, "Architecture of Network-on-Chip (NoC) for Secure Data Routing Using 4-H Function of Improved TACIT Security Algorithm", *Wireless Communications and Mobile Computing*, Vol. 2022, No. 1, p. 4737569, 2022.
[11] B.N.K. Reddy, M.Z.U. Rahman, and A. Lay-Ekuakille, "Enhancing Reliability and Energy Efficiency in Many-Core Processors Through Fault-Tolerant Network-On-Chip", *IEEE Transactions on Network and Service Management*, 2024.

[12] Q. Ijaz, H.L. Kidane, E.B. Bourennane, and G. Ochoa-Ruiz, "Dynamically scalable noc architecture for implementing run-time reconfigurable applications", *Micromachines*, Vol. 14, No. 10, p. 1913, 2023.
[13] F. Yazdanpanah, "A two-level network-on-chip architecture with multicast support", *Journal of Parallel and Distributed Computing*, Vol. 172, pp. 114-130, 2023.
[14] J. Luo, W. Wu, Q. Xing, M. Xue, F. Yu, and Z. Ma, "A low-latency Fair-Arbiter Architecture for Network-on-chip switches", *Applied Sciences*, Vol. 12, No. 23, p. 12458, 2022.
[15] A.S. Kumar, and B.N.K. Reddy, "An Efficient Real-Time Embedded Application Mapping for NoC Based Multiprocessor System on Chip", *Wireless Personal Communications*, Vol. 128, No. 4, pp. 2937-2952, 2023.
[16] S.G. Jayshree, and D. Pati, "Design and area performance energy consumption comparison of secured network-on-chip with PTP and bus interconnections", *Journal of The Institution of Engineers (India): Series B*, Vol. 103, No. 5, pp. 1479-1491, 2022.
[17] M.V. Sudhakar, P.R. Reddy, U. Penchalaiah, and P.R. Reddy, "A deep learning based latency aware predictive routing model for network-on-chip architectures", *International Journal of Communication Systems*, Vol. 36, No. 17, p. e5602, 2023.
[18] S. Mazumdar, A. Scionti, S. Zuckerman, and A. Portero, "NoC-based hardware software co-design framework for dataflow thread management", *The Journal of Supercomputing*, Vol. 79, No. 16, pp. 17983-18020, 2023.
[19] S. Singh, J.V.R. Ravindra, and B.R. Naik, "Design and implementation of network-on-chip router using multi-priority based iterative round-robin matching with slip", *Transactions on Emerging Telecommunications Technologies*, p. e4514, 2022.
[20] K. Parane, B.M.P. Prasad, and B. Talawar, "P-NOC: performance evaluation and design space exploration of nocs for chip multiprocessor architecture using FPGA", *Wireless Personal Communications*, Vol. 114, No. 4, pp. 3295-3319, 2020.
[21] P. Shahane, "Design and Implementation of Single Node NoC Router using Small Side Buffer in Input Block and iSLIP Scheduler", *International Journal*, Vol. 9, No. 4, 2020.
[22] D. Behera, S.N. Mishra, P.K. Sahoo, and H.A. Shah, "An enhanced approach towards improving the performance of embedding memory management units into Network-on-Chip", *e-Prime-Advances in Electrical Engineering, Electronics and Energy*, Vol. 6, p. 100332, 2023.