IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Mathematical Modelling of Engineering Problems
Vol. 10, No. 6, December, 2023, pp. 2149-2157
Journal homepage: http://iieta.org/journals/mmep

An Energy-Efficient Routing Protocol for a Modified Fat-Tree Topology in System-on-Chip Design

Akash Yadav[1*], Mushtaq Ahmed[1], Bhavna Ambudkar[2], Deepak Kumar[1]

[1] Malaviya National Institute of Technology Jaipur, Rajasthan 302017, India
[2] Symbiosis Institute of Technology Pune, Maharashtra 412115, India

Corresponding Author Email: 2019rcp9150@mnit.ac.in

Special issue: Emerging Trends in Computational Intelligence, Networks Technologies, and Wireless Communication Systems

https://doi.org/10.18280/mmep.100627

Received: 4 September 2023
Revised: 8 November 2023
Accepted: 20 November 2023
Available online: 21 December 2023

Keywords:
Fat-Tree topology, power consumption, power efficiency, routing protocol, system-on-chip, sniper simulator

ABSTRACT

The rapid evolution of integration technology has significantly influenced System-on-Chip (SoC) design, characterized by the unprecedented integration of numerous functional modules onto ever-shrinking chips. Consequently, facilitating robust communication between these burgeoning modules has become critical to optimizing system functionality and performance. Traditional Bus architecture, however, proves inadequate in realizing this goal. To circumvent these communication impediments, mesh and torus interconnected architectures have been proposed, with cores interconnected within a single chip using diverse routing strategies for seamless operation. This study investigates a deterministic, deadlock-free routing protocol for a modified Fat-Tree topology that incorporates a core module at the intermediate level. Utilizing the Sniper simulator environment for evaluation, it is demonstrated that the proposed protocol surpasses the X-Y routing method used in both mesh and torus topologies in terms of power efficiency and throughput. The results show a 4%-8% reduction in power consumption compared to other approaches, indicating the superior efficacy of the proposed method.

1. INTRODUCTION

First proposed by Gordon Moore in 1965, Moore's Law accurately predicted an approximate biennial doubling of transistors integrated onto a single chip (Figure 1), leading to continuous advancements in chip design. The emergence of System-on-a-Chip (SoC) technology in the 1980s was facilitated by the advent of very large scale integration (VLSI) technology, which enabled the integration of thousands of transistors onto a single silicon chip.

This innovative approach has been primarily characterized by its emphasis on miniaturization, transitioning from a system involving multiple circuits to the integration of these functionalities onto a compact silicon chip. Over the years, a consistent trend toward smaller IP chips has been observed, shrinking from 180 nanometers in the early 2000s to a mere 10 nanometers currently, as technology has progressively advanced [2]. Such a reduction in size has made it possible for designers to incorporate a greater number of transistors into the SoC.

The term "SoC" encapsulates the integration of the processor, memory, bus, hardware device drivers, among other components, onto a single platform. This technology has revolutionized design methods without altering the underlying functionality of systems. A wide range of systems, most notably in the realm of embedded computing, have harnessed this technology to consolidate their components, thereby achieving greater power and energy efficiency. Communication between IPs is facilitated through the memory and buses of a SoC, with bus arbiters ensuring real-time interaction and shared resource utilization among IPs as required [3].

Cutting-edge system-on-chip (SoC) architectures are currently anchored in the implementation of symmetric and asymmetric multicore designs within a Globally Asynchronous and Locally Synchronous (GALS) framework. This architectural strategy employs a growing number of functional modules, thereby enhancing overall system functionality and performance. These modules collaborate seamlessly, efficiently executing tasks, and communicating via a shared bus to distribute workloads and meet performance benchmarks [4, 5].

However, traditional bus architectures exhibit limitations when tasked with the escalating demands of intensive parallel communication within these intricate systems. The deficiencies of bus architectures are prominently exposed in situations demanding robust concurrent data exchange [2]. Additionally, the continual incorporation of modules into SoCs amplifies the need for enhanced bandwidth, precipitating a corresponding surge in power consumption associated with bus arbitration.

Meeting the imposed requirements and constraints of the

applications operational within the system is a fundamental consideration in SoC design. The demand for greater bandwidth, necessitated by increased data transfer rates between expanding functional modules, ensures optimal performance while mitigating bottlenecks. However, a significant hurdle is the escalating energy consumption during module communication, which imposes limitations on both application performance and architectural design. This escalating power consumption is emerging as a formidable challenge in contemporary SoC design [6].

The selection of suitable routing protocols and the configuration of traffic patterns significantly influence the overall performance within interconnected architectures employing Mesh or Torus topologies. Making an informed selection in this regard can profoundly impact the system's efficiency and effectiveness.

The Fat-Tree topology, an alternative to the Mesh and Torus topologies, presents several advantages. The tree structure minimizes communication distances, particularly beneficial for local communication patterns. It ensures minimal latency, facilitating the swift traversal of data across the network, and maximizes throughput, enabling optimal data flow through the system. The Fat-Tree topology exhibits superior scalability due to its higher bandwidth, unlike Mesh and Torus topologies where scalability is constrained by fixed bandwidth [7].

A range of parameters, including area, throughput, latency, and power consumption, can measure the SoC's performance, which directly impacts the system's performance [8, 9]. The modified Fat-Tree topology proposed in this paper reduces memory requirements in intermediate nodes, thus lowering power consumption [10, 11].

In contrast to the conventional Fat-Tree topology, where cores exist only at the leaf level, the proposed modification incorporates nodes or cores at every level. A routing protocol encompassing all Fat-Tree leaf nodes and intermediate nodes is utilized in this adaptation. When compared to Mesh and Torus topologies using different benchmarks, the Fat-Tree topology demonstrates enhanced performance in terms of lower power dissipation and improved throughput.

The paper is structured as follows: Section 2 delves into various routing algorithms. The importance of power estimation in SoC design, along with different tools for estimating power consumption, are explored in Section 3. Section 4 details the experimental design and the analysis of results, and finally, conclusions are drawn in Section 5.

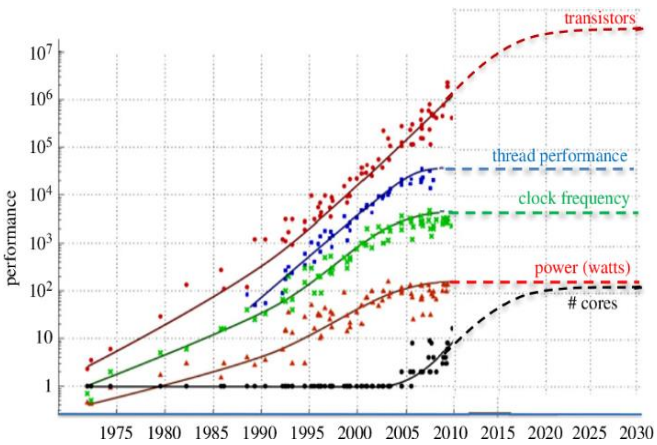

Figure 1. Moore' Law showing the growth of number of transistors in IC [1]

2. LITERATURE SURVEY

The SoC epitomizes an advanced network architecture, characterized by the integration of diverse processing modules onto a singular chip, thus forming a sophisticated interconnected parallel network. These cores are interconnected through a complex network of wiring, utilizing a variety of topologies including Mesh, Torus, Spidergon, and Fat-Tree [12, 13]. A pictorial representation of this architecture is presented in Figure 2.

A salient feature of the SoC architecture is the scalability it affords through an augmentation in the number of ports. This proliferation of potential pathways for packet routing serves to alleviate network congestion and enhance overall throughput. However, it is imperative to acknowledge that this scalability is not without its associated costs, and an increase in expenditure is inevitable.

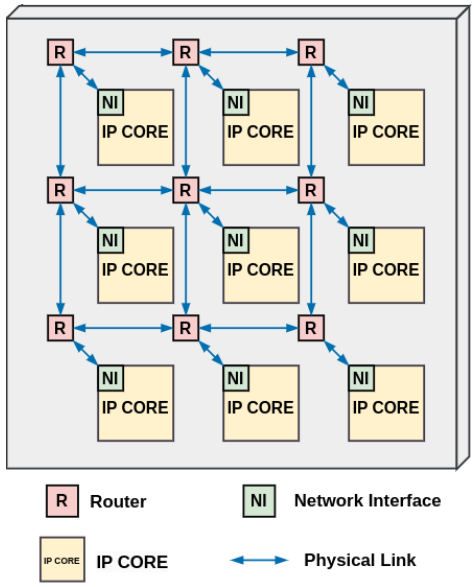

Figure 2. Typical on-chip structure where different processing nodes communicating via different associated routers

The SoC is characterized by dynamic features, facilitating versatile bandwidth configurations within the interconnections among processing cores. The specific arrangement of these cores is dictated by the selected SoC topology. The complexity of the multi-core interconnection architecture extends beyond mere core arrangement, encompassing the pivotal role of connectivity via routers. The configuration of these routers is contingent both on the chosen network topology [14, 15], as well as the applied routing algorithms [16, 17].

In view of these factors, SoC topologies are primarily classified into two distinct categories: direct and indirect topologies. The selection between these two categories profoundly influences the mode of communication and data exchange among the processing cores within the architecture. This dichotomy underscores the inherent adaptability and configurability of the designs, rendering them apt for a diverse range of applications and performance demands in contemporary computing systems.

- Direct Topology: In a direct network topology, each processing core is directly linked to a router. This layout simplifies the network architecture and encourages efficient communication. Grid-based layouts such as

2D-Mesh, 2D-Torus, and Folded Torus are commonly adopted configurations for direct topologies. The preference for these grid-like topologies is rooted in their cost-effectiveness and their potential to elevate the overall system performance. The regular, grid-like arrangement of cores and routers facilitates uncomplicated routing and streamlined data transfer, rendering them desirable choices for a variety of computing applications.

- Indirect Topology: Conversely, an indirect network topology segregates processing cores from the routers. In this configuration, routers are situated at distinct levels within the topology, separate from the processing cores positioned at the lowest level. In the upper levels of the topology, routers assume the function of transmitting packets between various levels, forming a multi-staged, tree-like structure. Prominent examples of indirect topologies include the H-tree and Fat-Tree architectures. This topology configuration allows for more intricate and versatile communication pathways, rendering it apt for scenarios where complex data routing and scalability are crucial. The hierarchical structure of the indirect topology offers advantages in terms of adaptability and resource allocation, making it a fitting choice for demanding computing environments.

2.1 2D-Mesh topology

The 2D-Mesh network topology, a grid-based configuration, is distinguished by its two-dimensional framework. This topology manifests in a grid formation, termed as k-ary, n-Mesh, where "k" denotes the count of nodes within the mesh, and "n" signifies the width or height of the network topology.

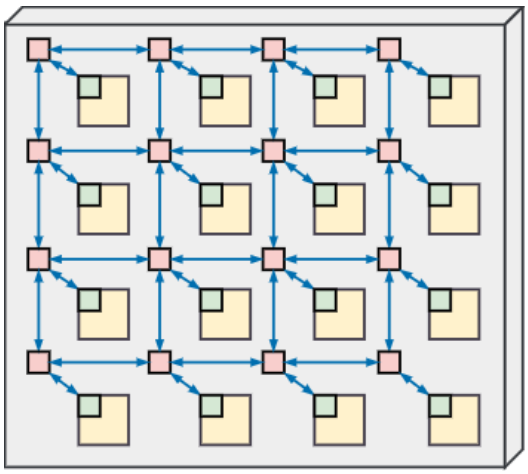

Figure 3. 2D-Mesh topology connected with each other in the GRID fashion

Figure 3 provides a visual depiction of the interconnectivity among nodes within the 2D-Mesh topology. Each node within this topology is capable of establishing connections with up to four neighbors, barring those located at the periphery of the network [18, 19].

A multitude of routing algorithms has been proposed for the Mesh topology to facilitate efficient data transfer. These algorithms comprise both minimal path deterministic routing options such as source routing and X-Y routing, as well as non-minimal non-deterministic routing alternatives like restricted turn-model routing and user-defined routing [20, 21]. This assortment of routing algorithms allows the Mesh topology to adapt to diverse communication needs, rendering it a flexible choice in network design. Despite its preference due to simplicity, the Mesh topology does possess a disadvantage. The primary issue lies in its large diameter, which impedes communication speed.

2.2 Torus topology

Figure 4 presents an illustrative representation of the torus network topology. This unique topology displays a characteristic feature of wraparound connectivity, where the top seamlessly links to the bottom, and the right edge smoothly transitions to the left edge. This wraparound connectivity attribute is consistently preserved at every edge node throughout the network topology. Furthermore, the torus topology adopts a grid-based structure comparable to the k-ary, n-Mesh configuration, wherein "k" represents the number of nodes within the mesh, and "n" indicates the width or height of the network topology [22].

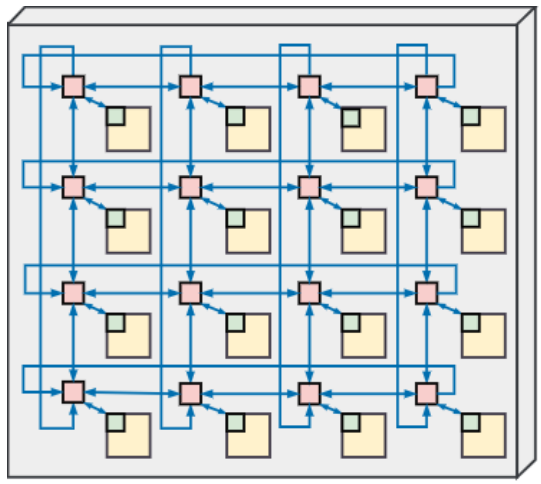

Figure 4. Torus topology connected with each other in the GRID fashion

Within this topology, each node sustains connections with its four immediate neighbors, reflecting the connectivity pattern seen in the 2D-Mesh. However, it is crucial to note that the torus topology introduces an asymmetry into the network, setting it apart from the symmetrical 2D-Mesh configuration. For instance, the route distance between the top-left and top-right nodes (as depicted in Figure 4) is greater than that between the top-left and bottom-right nodes. This results in an asymmetry within the torus, where node distances may vary based on their positions.

A variant of the torus network topology, known as the folded torus, employs two layers of nodes that are connected in a ring-like fashion. The uppermost node is linked to the bottom node to interconnect each layer of the topology. Generally, the torus topology is the preferred choice if reducing latency is a significant consideration. Conversely, when power consumption reduction is a priority, the mesh topology emerges as a more suitable option [23].

2.3 Fat-Tree topology

The Fat-Tree topology serves as a striking example of a tree-based indirect network configuration. Within tree-based topologies, the bandwidth availability scales up with each

ascending level within the tree structure. A distinguishing feature of the Fat-Tree topology is its impressive ability to increase the number of connections as one ascends toward the tree's root [24]. This inherent scalability renders the Fat-Tree topology particularly beneficial in situations involving a large number of interconnected processing cores within the system.

As illustrated in Figure 5, this topology demonstrates its capabilities by accommodating an extensive network of nodes and interconnections, thereby optimizing data flow and enhancing system performance. The architecture of the Fat-Tree topology is adept at efficiently managing the demands of large-scale processing environments, making it a favored choice for modern, data-intensive applications and computing infrastructures. Its tree-based structure, with increased bandwidth towards the higher levels, equips it to manage and facilitate the robust communication needs of complex systems.

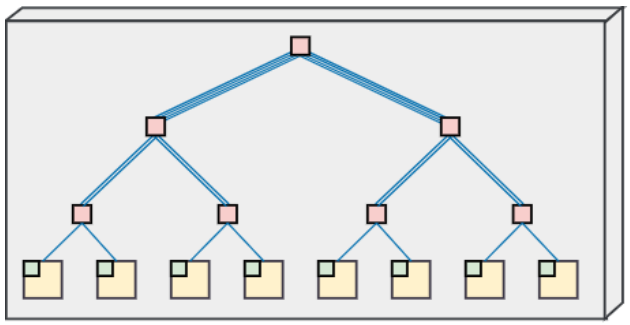

Figure 5. Typical connection of the Fat-Tree topology for tree-height 3

Securing deadlock-free routing within a SoC can be achieved through two primary strategies. The first tactic involves augmenting the number of ports or implementing virtual channels, while the second strategy entails adopting a restricted turn-model routing approach. In a fat-tree, it is possible to navigate from any leaf to any other node using either an upward or downward path. It mirrors a logical tree, and there are two types of turns: UP turns (from child to parent) and DOWN turns (from parent to child). Consequently, the application of restricted turn-based routing can effectively prevent deadlock without the necessity for virtual channels [25].

3. PROPOSED ALGORITHM

The Fat-Tree topology stands out as a prevalent choice when it comes to optimizing parallelism within SoC architectures. It excels in offering a deterministic routing mechanism, whereby only a single, predefined path exists between any two nodes within the network. One remarkable feature of the Fat-Tree topology is its capacity for scalable bandwidth, which expands proportionally with the rise in network levels.

In the context of this paper, a novel approach is introduced that uses a deterministic range-based algorithm to implement the Fat-Tree topology specifically for complete binary trees. A complete binary tree is one where each level, except the last, is completely filled, and all nodes in the last level are positioned as far left as they can be.

Problem definition: In an given Fat-Tree based on-chip interconnection communication architecture $Arch_N(C,L)$, the routers at source node C_{source} and destination node C_{dest}, find a decision function at current node $C_{current}$, for selecting an outport to forward the packet at right direction. $Arch_N(C,L)$ is a tree structure where,

- N is the number of cores in the structure
- Each node $c_i \in C$ represents a core with routers for $0 \leq i < N$, and
- $l_j \in L$ represents the tree-level of core c_i for $0 \leq j \leq \lfloor log_2(N) \rfloor$

The objective of the algorithm, detailed in Algorithm 1, is to use the destination node's level to make routing decisions. Algorithm 1 provides a step-by-step guide to implementing the Fat-Tree routing strategy. By introducing this deterministic range-based algorithm, the paper aims to enhance the efficiency and performance of the Fat-Tree topology.

In a network comprising N processing cores, each individual node carries specific information. Every node actively records the range data of its child nodes. Through a recursive process, it also maintains knowledge of the ranges of its grandchildren, stemming from its own children. Consequently, each node, at the very least, possesses information concerning the positions of its immediate offspring.

The first step is to find the level of the destination node. If the current node is the same as the destination node, it means the destination has been reached. In this case, set the direction to the destination and return the current core_id. If the destination falls within the range of the current node's children, proceed with the downward direction represented as DOWN. Calculate L and R, which represent the distances between the destination and the left and right children of the current node. If L is less than R, it means the left child is closer to the destination, so return the ID of the left child (core_id.childL), otherwise, return the ID of the right child (core_id.childR). If the destination is not within the range of the current node's children, set the direction to move upwards represented as UP and return the ID of the parent node (core_id.parentNode).

Algorithm 1. Routing Protocol for Fat-Tree

```
Result: core_id
level=findLevel(destination)
if destination == core_id then
    direction=destination;
    return(core_id);
else if core_id.nodeRange[level][L] ≤ destination
AND core_id.nodeRange[level][R] ≥ destination then
    direction=DOWN;
    L=destination - core_id.nodeRange[level][L]
    R=core_id.nodeRange[level][R] - destination
    if L < R then
        return(core_id.childL);
    else
        return(core_id.childR);
    end if
else
    direction=UP;
    return(core_id.parentNode);
end if
```

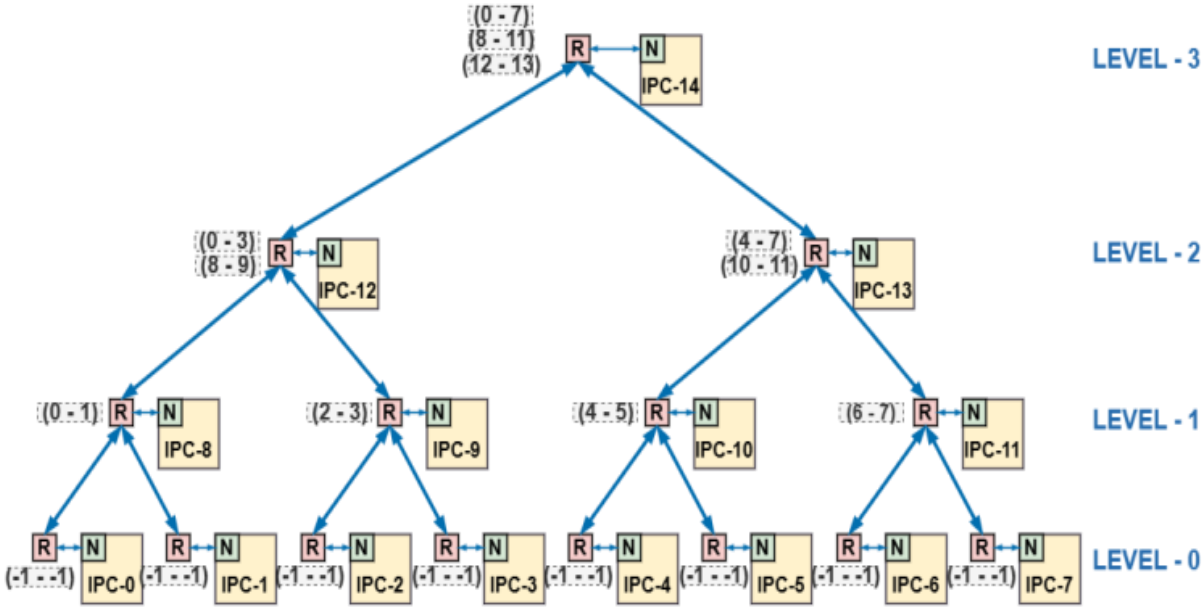

Figure 6. Hybrid Fat-Tree having connected nodes in the intermediate levels

Figure 6 provides a visual representation of the Fat-Tree topology, featuring fifteen nodes labeled IPC-0 to IPC-14. Nodes situated at level 0 are leaf nodes, bereft of any children, hence their child range is defined as {-1 to -1}. For instance, Node IPC-8 at level 1 contains children IPC-0 and IPC-1, resulting in a child range of {0 - 1}. In a similar vein, Node IPC-9's child range is {2 - 3}, and so forth. At level 2, Node IPC-12 boasts both children and grandchildren, consequently housing two sets of range values—one encompassing its immediate grandchildren, denoted as {0 - 3}, and the other detailing the range of its own children, {8 - 9}. A parallel scenario is observed with Node IPC-13. The root node, positioned at three different levels (levels 0, 1, and 2), holds comprehensive information about all its descendants. The first range spans the farthest grandchildren, {0 - 7}, followed by the grandchildren at the next level with the range value {8 - 11}, and lastly, the range value pertaining to its immediate children, which is {12 - 13}.

In the packet transmission process, the router adopts a strategy that relies on evaluating the difference between its own designated range and the intended destination's position. If the destination node is not situated within its range, the router directs the packet upwards.

This ingenious mechanism plays a pivotal role in reducing hop latency, leading to a substantial enhancement in the overall system's throughput. The Fat-Tree topology, with its inherent characteristics, manages to achieve remarkable efficiency, characterized by its computational complexity of $O(\log(N))$. This complexity ensures that the system operates with remarkable efficiency, even as the number of processing cores, denoted as N, scales up.

4. EXPERIMENTAL SETUP AND RESULT ANALYSIS

The algorithm designed for the Fat-Tree topology is subjected to simulation using the Sniper simulator. This simulator, renowned for its high-performance capabilities, operates as a parallel and cycle-accurate tool specially crafted for the next-generation simulation needs [26, 27]. Within this simulation environment, a diverse array of computing scenarios can be explored, including both homogeneous and heterogeneous multicore architectures. The interval core model, a notable feature of this platform, empowers users to construct Cycle Per Instruction (CPI) stacks. These stacks provide invaluable insights into the number of cycles lost due to various system characteristics and behaviors.

For a more comprehensive understanding, Figure 7 offers a concise overview of the program execution environment facilitated by this simulator. This environment proves indispensable in assessing and fine-tuning algorithms within the Fat-Tree topology, paving the way for optimized performance and enhanced system efficiency.

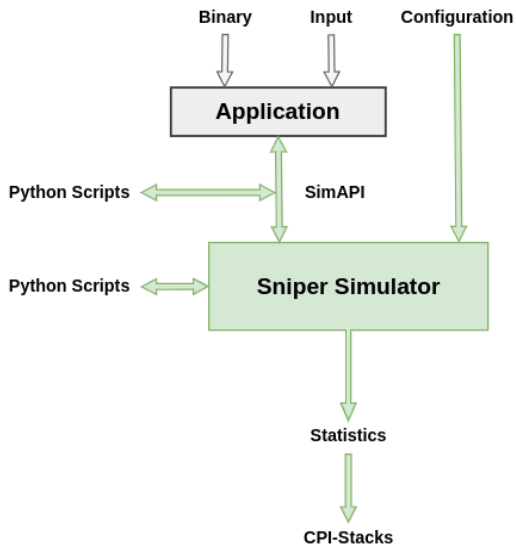

Figure 7. The program execution environment used for simulation in Sniper

In a program, every thread possesses its distinct cycle stack, comprising several cycle components. This cycle stack, often referred to as Cycle per Instruction (CPI), offers a comprehensive breakdown of the total number of cycles, delineating its constituent parts. These cycle components are graphically represented as bars in a histogram, where they are

stacked on top of each other, with the component at the base of the histogram serving as the foundation.

To provide a visual representation of this concept, Figure 8 shows the original CPI stack as visualized within the Sniper simulator, specifically for a Mesh topology employing four cores. This depiction aids in understanding how the various cycle components contribute to the overall cycle count, offering insights into the performance characteristics of the program in question.

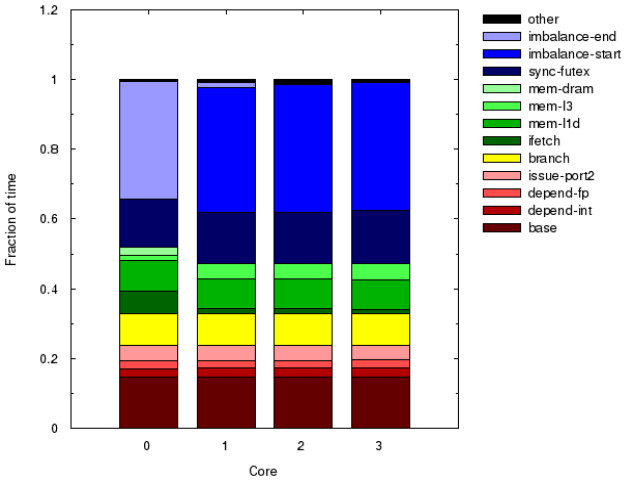

Figure 8. Visualization example of CPI stacks in Sniper for mesh topology with four cores

In our experimental setup, we employ two well-established benchmarks, namely SPLASH-2 (Barnes) and PARSEC (Bodytrack). These benchmarks hold a prominent position in the field of performance evaluation and are integrated into various multi-threaded application environments, ensuring compatibility with a range of simulation platforms [28].

- *Barnes:* The Barnes benchmark simulates the dynamics of a system of bodies in three dimensions through the utilization of the Barnes-Hut hierarchical N-body method. It serves as a valuable tool for modeling and analyzing complex interactions among objects [29].
- *Bodytrack:* The Bodytrack benchmark is centered around computer vision applications within the Intel RMS workload. This particular benchmark is dedicated to tracking the movements and positions of the human body through the processing of multiple camera images. Its relevance lies in its ability to assess and optimize the performance of systems handling real-time visual data [30].

The Fat-Tree topology is compared to 2D-Mesh and Torus topologies with an equal number of cores using *power consumption* and *throughput* as the performance parameter.

- *Throughput:* Throughput can be defined as the maximum amount of information delivered per unit of time. It can be measured as the messages per second or messages per clock cycle. In other terms, it is the maximum number of packets received out of the packed transmitted from the source over the channel. Eq. (1) shows the formula for calculating the throughput in the experiments.

$$Throughput(TP) = \frac{\prod_{i=1}^{n}\left(P_i(1-D_i)\right)}{L} \quad (1)$$

where,

- n is the number of nodes from source to destination.
- P_i represents the probability of successful transmission from node i to node $i+1$.
- D_i represents the probability of packet drop at node i.
- L represents the latency.

- *Power Consumption*: The power consumption is calculated as the total of power consumed by cores and various memories in wattage, as shown in Eq. (2).

$$PowerConsumption(PC) = \sum_{i=1}^{n}\left(Pcore_i + Pcore_{mem\,i} + Picache_i + Pdcache_i + Pdram_i + Ptransmission_i\right) \quad (2)$$

where,

- $Pcore_i$ is the power consumed by core i,
- $Pcore\_mem_i$ is the power consumed by memory of core i,
- $Picache_i$ is the power consumed by icache memory of core i,
- $Pdcache_i$ is the power consumed by dcache memory of core i,
- $Pdram_i$ is the power consumed by dram memory of core i,
- $Ptransmission_i$ is the power consumed by transmission link of core i.

In our experimental setup, we explore various configurations by altering both the frequencies and the number of symmetric cores within each topology. We systematically evaluate the performance parameters for these diverse combinations. For instance, we examine scenarios involving 8 and 16 cores, with frequency settings ranging from 2.66 GHz to 5 GHz, across Mesh, Torus, and Fat-Tree topologies. It's worth noting that we maintain a homogeneous configuration across all processing cores to ensure consistency.

Table 1. Values for possible test combinations used in the experiments by the simulator

S. No.	Name	Specifications
1	Topology	Fat-Tree, 2D-Mesh, Torus
2	Frequency	2.66 GHz, 3 GHz, 4 GHz, 5 GHz
3	No. of Cores	8, 16
4	Cache Memory	icache=32Kb, associativity=4 dcache=32Kb, associativity=4 L2 cache=512 Kb, associativity=8
5	Global Freq	1 GHz
6	Benchmarks	SPLASH2, PARSEC

A summary of the values utilized in these experiments is presented in Table 1. To enhance the accuracy and reliability of our findings, we compute the averages from five independent runs for each potential combination. Our assessment of throughput follows the equation (Eq. (1)), which calculates it as the ratio of the total number of packets successfully received at the destination to the total time elapsed in milliseconds. This methodology guarantees robust and trustworthy results, which are essential for a comprehensive evaluation of system performance.

4.1 Result analysis

Table 2 provides an insightful overview of the system's throughput performance when utilizing eight cores. Notably, for lower frequency settings, we observe relatively consistent throughput values across all the employed topologies. However, as the frequency of individual processing cores escalates, an interesting trend emerges. The Fat-Tree topology notably outperforms its grid-based counterparts in terms of throughput. This suggests that the Fat-Tree configuration exhibits a favorable response to the increased processing power, harnessing higher frequencies effectively to enhance system performance.

Table 2. Throughput comparison between the topologies

Processing Cores	Topology	Throughput (# of Packets/ms)			
		f=2.66 GHz	f=3 GHz	f=4 GHz	f=5 GHz
8-Cores	2D-Mesh	44	45	51	101
	Torus	45	50	82	103
	Fat-Tree	51	56	87	108
16-Cores	2D-Mesh	30	32	53	56
	Torus	31	34	55	59
	Fat-Tree	34	38	59	63

Furthermore, in Table 2, we extend our analysis to include the throughput results for configurations involving sixteen cores. As anticipated, with the augmentation in the number of cores, the Fat-Tree topology consistently showcases superior throughput performance when compared to both the 2D-Mesh and Torus topologies. This observation aligns with the expected behavior, as the Fat-Tree topology's inherent scalability and interconnectivity prove advantageous in scenarios where a greater number of cores are engaged. Consequently, it serves as a preferable choice to meet the heightened processing demands associated with larger core counts, ensuring optimized system performance.

The data presented in Table 2 demonstrates a clear trend. The Fat-Tree topology consistently outperforms other grid-based configurations, such as the 2D-Mesh and 2D-Torus, in terms of throughput. This notable advantage can be primarily attributed to the Fat-Tree's tree-based structure, which effectively reduces packet hop latency, subsequently leading to enhanced throughput. The reduction in hop latency is a pivotal factor that propels the system's overall performance, as it facilitates faster data transfer and more efficient communication between processing cores.

It is important to observe that the difference in hop latency between grid-based and tree-based topologies is not very large for systems with fewer processing cores. This is due to the fact that grid-based topologies, even with their O(n) complexity, function rather effectively for small core counts. However, as the number of processing cores increases, tree-based topologies provide a more effective complexity of O(log(n)), yielding a performance improvement that becomes more and more obvious. As a result, the proposed topology performs best in situations when number of cores is more.

A similar configuration is employed to compute the power consumption for all three topologies under examination. Figure 9 and Figure 10 provide graphical representations of the power consumption results for eight and sixteen cores, respectively.

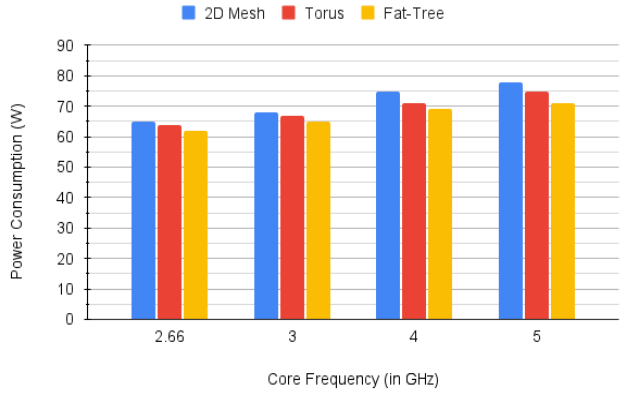

Figure 9. The comparison of power consumption for various topologies with eight processing cores

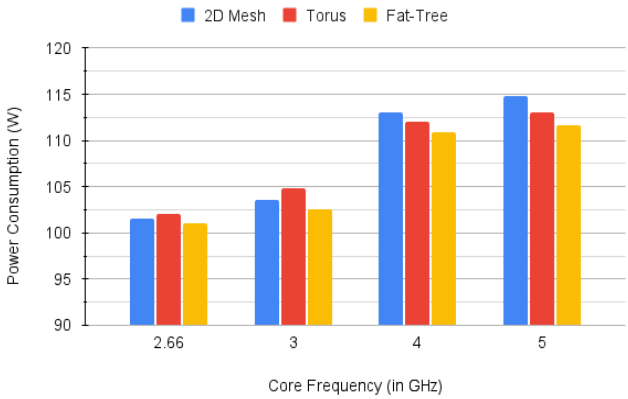

Figure 10. The comparison of power consumption for various topologies with sixteen processing cores

As anticipated, these figures unmistakably illustrate that power consumption escalates in direct correlation with the frequency increase of the processing cores, irrespective of the chosen topology. Furthermore, it's evident that power consumption exhibits a direct relationship with hop latency. As hop latency rises, so does power consumption, while conversely, lower hop latency is associated with reduced power consumption.

Notably, the Fat-Tree topology demonstrates a significant advantage in hop latency compared to the other topologies, resulting in lower power consumption. This advantageous characteristic is clearly depicted in both Figure 9 and Figure 10.

The result shows that the differences between the GRID-based and Fat-Tree topologies are not as apparent when working with fewer cores. This is in line with expectations, as the effects of topology become more prominent with a higher number of cores engaged in computation.

However, as we scale up to configurations featuring a greater number of cores, we anticipate observing a more significant disparity in IPC performance between the two topologies. The inherent advantages of the Fat-Tree topology, including its efficient routing and reduced hop latency, are expected to become increasingly apparent as the computational workload intensifies. Therefore, these IPC results provide valuable insights into how the choice of topology can impact the efficiency and performance of multi-core systems, particularly in scenarios involving a substantial

number of processing cores. On the other hand, the fat-tree topology also has same limitations. The main one is the root node that is the topology's bottleneck; if it malfunctions, whole network can collapse. In addition, network configuration and management are more difficult as the tree grows longer.

The Fat-Tree's efficient routing and reduced hop latency contribute to its ability to operate with lower power consumption, making it an attractive choice in scenarios where power efficiency is a critical concern, such as in energy-efficient computing environments or battery-powered devices.

We also conducted an analysis of the Instruction-per-Cycle (IPC) results for both the SPLASH2-Barnes benchmark and the PARSEC-Bodytrack benchmark. Figure 11 and Figure 12 visually represent the IPC outcomes for the Barnes and Bodytrack benchmarks, respectively.

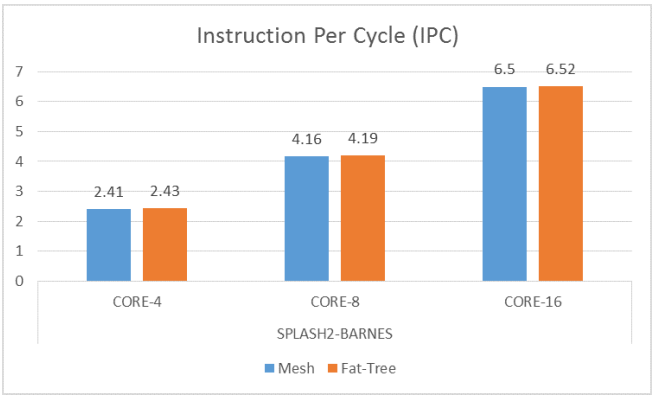

Figure 11. Comparison of Instruction-per-Cycle (IPC) for SPLASH-2 (Barnes) benchmark application in Mesh topology and Fat-Tree topology

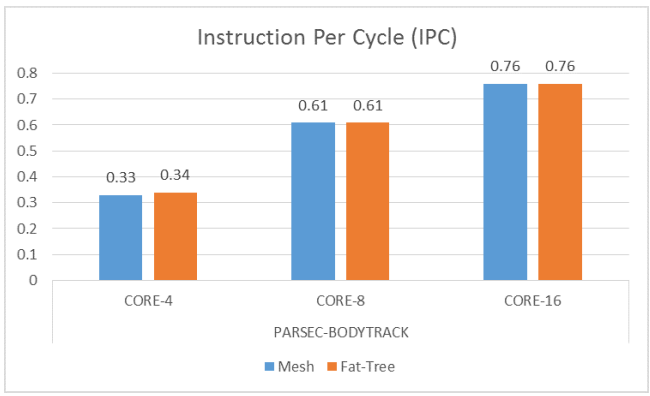

Figure 12. Comparison of Instruction-per-Cycle (IPC) for PARSEC (Bodytrack) benchmark application in Mesh topology and Fat-Tree topology

5. CONCLUSION AND FUTURE WORK

In this paper, a routing protocol for a modified Fat-Tree topology for system-on-chip was proposed. In modified Fat-Tree, cores are also present at the intermediate level of the tree. We compared the proposed approach with GRID-based topologies like 2D-Mesh and 2D-Torus based on throughput and power consumption. The analysis showed that the proposed approach offers higher throughput than the GRID-based topologies. Tree-based topology's reduction of packet hop latency is the primary factor contributing to an increase in throughput. As a result of the reduced hop latency, throughput is increased, and power consumption is decreased. In the future, an asymmetric core architecture with varying frequencies, a pipeline, and variance of cache memories with additional resources like Booth or Wallace multipliers shall be tested for the existing and new benchmarks. The performance shall also be measured for the channel bandwidth with the best efforts, guaranteed throughput, and some degree of fault tolerance. Also, other routing strategies may be compared for this modified structure of the Fat-Tree.

REFERENCES

[1] Shalf, J. (2020). The future of computing beyond Moore's Law. Philosophical transactions. Series A, Mathematical, Physical, and Engineering Sciences, 378(2166): 20190061. https://doi.org/10.1098/rsta.2019.0061

[2] Martin, G. (2003). The history of the SOC revolution. In Winning the SOC Revolution. Springer, Boston, MA. https://doi.org/10.1007/978-1-4615-0369-9_1

[3] Risset, T. (2011). SoC (System on Chip). In Encyclopedia of Parallel Computing. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-09766-4_5

[4] Ray, K., Kalita, A., Biswas, A., Hussain, M.A. (2016). A multipath network-on-chip topology. In 2016 International Conference on Information Communication and Embedded Systems (ICICES), Chennai, India, pp. 1-7. https://doi.org/10.1109/ICICES.2016.7518839

[5] Manivannan, M., Pericàs, M., Papaefstathiou, V., Stenström, P. (2017). Runtime-assisted global cache management for task-based parallel programs. IEEE Computer Architecture Letters, 16(2): 145-148. https://doi.org/10.1109/LCA.2016.2606593

[6] Ibarra-Delgado, S., Sandoval-Arechiga, R., Gómez-Rodríguez, J.R., Ortíz-López, M., Brox, M. (2020). A bandwidth control arbitration for SoC interconnections performing applications with task dependencies. Micromachines (Basel), 11(12): 1063. https://doi.org/10.3390/mi11121063

[7] Sllame, A., Hasan, A. (2014). A comparative study between fat tree and mesh network-on-chip interconnection architectures. In the 14th Annual Middle Eastern Simulation and Modelling Conference (MESM'14).

[8] Bhaskar, A.V. (2022). A new method of power analysis of Network-on-Chip using analytical modelling. In 2022 Seventh International Conference on Parallel, Distributed and Grid Computing (PDGC), Himachal Pradesh, India, pp. 222-227. https://doi.org/10.1109/PDGC56933.2022.10053136

[9] Kalyan, T.V., Mutyam, M. (2008). Word-interleaved cache: An energy efficient data cache architecture. In Proceeding of the 13th International Symposium on Low Power Electronics and Design (ISLPED'08), Bangalore, India, pp. 265-270. https://doi.org/10.1145/1393921.1393991

[10] Nain, Z., Ali, R., Anjum, S., Afzal, M.K., Kim, S.W. (2020). A network adaptive fault-tolerant routing algorithm for demanding latency and throughput applications of network-on-a-chip designs. Electronics, 9(7): 1076.

[11] Zhou, W., Ouyang, Y.M., Li, J.H., Xu, D.Y. (2023). A

transparent virtual channel power rating method for on-chip network routers. Integration, 88: 286-297. https://doi.org/10.1016/j.vlsi.2022.10.004

[12] Albughdar, M., Mahmood, A. (2015). Maximally adaptive, deadlock-free routing in spidergon-donut network for large multicore NOCs. In 2015 14th International Symposium on Parallel and Distributed Computing, Limassol, Cyprus, pp. 210-214. https://doi.org/10.1109/ISPDC.2015.31

[13] Heirman, W., Carlson, T.E., Che, S., Skadron, K., Eeckhout, L. (2011). Using cycle stacks to understand scaling bottlenecks in multi-threaded workloads. In 2011 IEEE International Symposium on Workload Characterization (IISWC). Austin, TX, USA, pp. 38-49. https://doi.org/10.1109/IISWC.2011.6114195

[14] Reddy, T.N.K., Swain, A.K., Singh, J.K., Mahapatra, K.K. (2014). Performance assessment of different network-on-chip topologies. In 2014 2nd International Conference on Devices, Circuits and Systems (ICDCS), Coimbatore, India, pp. 1-5. https://doi.org/10.1109/ICDCSyst.2014.6926188

[15] Alimi, I.A., Patel, R.K., Aboderin, O., Abdalla, A.M., Gbadamosi, R.A., Muga, N.J., Pinto, A.N., Teixeira, A.L. (2021). Network-on-chip topologies: Potentials, technical challenges, recent advances and research direction. In IntechOpen, Rijeka, Chapter 3. https://doi.org/10.5772/intechopen.97262

[16] Monakhova, E.A., Monakhov, O.G., Romanov, A.Y. (2023). Routing algorithms in optimal degree four circulant networks based on relative addressing: Comparative analysis for networks-on-chip. IEEE Transactions on Network Science and Engineering 10(1): 413-425. https://doi.org/10.1109/TNSE.2022.3211985

[17] Wachter, E., Erichsen, A., Amory, A., Moraes, F. (2013). Topology-agnostic fault-tolerant NoC routing method. In 2013 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, pp. 1595-1600. https://doi.org/10.7873/DATE.2013.324

[18] Zhang, X. (2023). Power: Multi-capability adaptive routing for network-an-chips. In 2023 3rd International Conference on Neural Networks, Information and Communication Engineering (NNICE), Guangzhou, China, pp. 751-754. https://doi.org/10.1109/NNICE58320.2023.10105691

[19] Gardea, J., Jin, Y., Badawy, A.-H., Cook, J. (2017). Performance evaluation of mesh-based 3D NoCs. In Proceedings of the 10th International Workshop on Network on Chip Architectures (NoCArc'17). Association for Computing Machinery, New York, NY, USA, pp. 1-6. https://doi.org/10.1145/3139540.3139545

[20] Rao, G.V.V., Kavitha, A., Arthy, P.S. (2022). Review and analysis on network on chip. In 2022 International Conference on Computer, Power and Communications (ICCPC), Chennai, India, pp. 166-170. https://doi.org/10.1109/ICCPC55978.2022.10072109

[21] Fusella, E., Cilardo, A. (2018). Understanding turn models for adaptive routing: The modular approach. In 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, pp. 1477-1480. https://doi.org/10.23919/DATE.2018.8342245

[22] Wu, K., Ye, Y.Y. (2022). Q-learning based bi-objective deadlock-free routing optimization for optical NoCs. In 2022 15th IEEE/ACM International Workshop on Network on Chip Architectures (NoCArc). IEEE Computer Society, Los Alamitos, CA, USA, pp. 1-6. https://doi.org/10.1109/NoCArc57472.2022.9911373

[23] Mirza-Aghatabar, M., Koohi, S., Hessabi, S., Pedram, M. (2007). An empirical investigation of mesh and torus NoC topologies under different routing algorithms and traffic models. In 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007), Lubeck, Germany, pp. 19-26. https://doi.org/10.1109/DSD.2007.4341445

[24] Lin, B.-C., Lea, C.-T. (2022). Designing nonblocking networks with a general topology. IEEE Access, 10: 8399-8408. https://doi.org/10.1109/ACCESS.2021.3139732

[25] Bogdanski, B., Reinemo, S.-A., Sem-Jacobsen, F.O., Gran, E.G. (2012). sFtree: A fully connected and deadlock-free switch-to-switch routing algorithm for fat-trees. ACM Transactions on Architecture and Code Optimization, 8(4): 1-20. https://doi.org/10.1145/2086696.2086734

[26] Carlson, T.E., Heirman, W., Eeckhout, L. (2011). Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation. In SC '11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, Seattle, Washington, pp. 1-12. https://doi.org/10.1145/2063384.2063454

[27] Chis, R., Vintan, L. (2014). Multi-objective hardware-software co-optimization for the SNIPER multi-core simulator. In 2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, Cluj, Romania, pp. 3-9. https://doi.org/10.1109/ICCP.2014.6936772

[28] Barrow-Williams, N., Fensch, C., Moore, S. (2009). A communication characterisation of Splash-2 and Parsec. In 2009 IEEE International Symposium on Workload Characterization (IISWC), TX, USA, pp. 86-97. https://doi.org/10.1109/IISWC.2009.5306792

[29] Singh, J.P. (1993). Parallel hierarchical N-body methods and their implications for multiprocessors. PhD Thesis, Stanford University.

[30] Huỳnh, A., Helm, C., Iwasaki, S., Endo, W., Namsraijav, B., Taura, K. (2019). TP-PARSEC: A task parallel PARSEC benchmark suite. Journal of Information Processing, 27: 211-220. https://doi.org/10.2197/ipsjjip.27.211