Journal of VLSI Circuits and Systems, ISSN: 2582-1458 Vol. 6, No. 2, 2024 (pp.130-140)

RESEARCH ARTICLE

WWW.VLSIJOURNAL.COM

Performance Evaluation of Mesh, Spidergon, and Mesh of Spidergon (MoS) Topologies in Network-on-Chip (NoC) Architectures

Sravani S[1], Deepak Ch[1]

[1]*School of Electronics Engineering, VIT-AP University, Amaravati, Andhra Pradesh, India.*

KEYWORDS:
Latency
Mesh-of-Spidergon topology
Throughput
Network-on-Chip

ARTICLE HISTORY:

Received	: 15.10.2024
Revised	: 20.11.2024
Accepted	: 26.12.2024

https://doi.org/10.31838/jvcs/06.02.15

Abstract

Network-on-chip (NoC) is essential for efficient data transmission in system-on-chip (SoC) architectures, particularly as modern integrated circuits (ICs) become increasingly complex. The choice of topology significantly impacts the performance, throughput, and latency of NoC designs. Traditional topologies, such as Mesh and Spidergon, offer unique benefits and limitations. This study introduces a new topology, Mesh of Spidergon (MoS), which integrates the local connectivity advantages of Mesh with the global routing efficiency of Spidergon. We evaluate the network diameter and average distance of the Mesh, Spidergon, and MoS topologies. The assessment focuses on throughput and latency using Transmission Control Protocol (TCP) applications, specifically analyzing File Transfer Protocol (FTP) and Constant Bit Rate (CBR) traffic patterns under varying link failure rates of 5%, 10%, and 15%. Results show that the MoS topology achieves a throughput of 26.96 Mbps, surpassing Mesh's 12.68 Mbps and Spidergon's 23.51 Mbps using FTP protocol. For the CBR protocol, MoS reaches 34 Mbps, compared to 12.68 Mbps for Mesh and 25.23 Mbps for Spidergon, while maintaining comparable latency to Spidergon at 2.37 ms, whereas Mesh exhibits higher latency with 3.78 ms. Furthermore, MoS demonstrates enhanced resilience under link failures. Overall, the MoS topology significantly improves NoC performance, merging the strengths of existing topologies and offering a viable solution for future SoC designs.

Author e-mail:deepakchenu@gmail.com, sravani.sravanam@gmail.com

How to cite this article: Sravani S, Deepak Ch. Performance Evaluation of Mesh, Spidergon, and Mesh of Spidergon (MoS) Topologies in Network-on-Chip (NoC) Architectures, Journal of VLSI Circuits and System Vol. 6, No. 2, 2024 (pp. 130-140).

Introduction

In the world of modern microelectronics, rapid technological advancements have led to the emergence of intricate complexities and escalating demands in integrated circuit designs. As traditional monolithic approaches encounter growing obstacles related to throughput, latency, and performance, a transforming paradigm shift has materialized in the form of Network-on-Chip (NoC) architectures. NoC signifies a revolutionary strategy for interconnecting the diverse components residing within a chip, presenting an agile and efficient solution for providing evolving requirements of complex systems-on-chip (SoCs). NoC is meticulously engineered to furnish a scalable and proficient framework, thereby facilitating the seamless exchange of data and communication among various processing elements or intellectual property (IP) cores, all amalgamated onto a solitary semiconductor chip. One conspicuous facet of NoC's significance manifests in its profound influence on the delineation of on-chip topologies. These topologies meticulously direct the intricate interconnections among various chip components. The integration of Transmission Control Protocol (TCP)[1] connections within this on-chip communication framework warrants special consideration. TCP, serving as a ubiquitous transport layer protocol in the realm of computer networking, is distinguished by its commitment to reliability and its connection-oriented nature. It steadfastly ensures the orderly and error-free transmission of data between two endpoints ensconced within a network. When integrating the TCP connections within on-chip topologies, specific crucial aspects must be considered to ensure proper functionality and performance. It is a transport protocol extensively used in computer networking to guarantee reliable, ordered, and error-free data delivery between applications running on hosts connected to the Internet. It is a connection-oriented protocol, which establishes a connection between two hosts before exchanging data. It segments the data and transmits it across the network using a sliding window algorithm for flow control to prevent data overload at the receiver's end. The protocol also uses sequence numbers to ensure data is received in the correct order and check-sums to guarantee data is transmitted without errors. It offers a congestion control mechanism that monitors network conditions and adjusts the transmission rate to avoid network congestion. This mechanism ensures efficient utilization of network resources and prevents packet loss and network performance degradation caused by congestion. Applications that require reliable data transfer, such as email, file transfer, and web browsing, widely use TCP. It is also utilized as the underlying transport protocol for other protocols like FTP and CBR. FTP, which stands for File Transfer Protocol,[1] is a widely used network protocol for transferring files between source and destination in a network. It sends packets from the source to the sink in a specified path. It uses two TCP connections, one for control and another for data transfer. The control connection is responsible for sending commands and receiving responses, while the data connection is used to transfer files. Constant Bit Rate (CBR)[1] is a traffic type used in computer networks to produce a continuous flow of data at a constant rate. This traffic pattern is commonly employed in applications that require steady data transmissions, such as video or audio streaming, to provide smooth playback without interruptions. CBR traffic is generated by transmitting packets at a constant rate, regardless of the network's state or available bandwidth. This creates a deterministic traffic pattern, which means that the size and arrival time of each packet can be anticipated in advance. It is frequently utilized in network simulations to assess the performance of various network protocols and architectures under stable traffic conditions. The existing literature predominantly focuses on evaluating the performance of individual NoC topologies like Mesh, Torus, and Fat Tree, among others. While these studies provide valuable insights into throughput and latency, they often overlook energy fault tolerance. Additionally, there is a lack of exploration into hybrid topologies that can combine the strengths of different designs to achieve better overall performance. This paper addresses these gaps by proposing a novel hybrid topology called Mesh of Spidergon (MoS). The MoS topology integrates the Mesh topology for local connections and the Spidergon topology for global connections, aiming to enhance both performance and throughput. By leveraging the strengths of both topologies, MoS is designed to reduce the number of hops required for data transmission, thereby improving throughput and reducing latency. This paper is organized as follows. Section 2 gives the related work of various authors. Existing topologies, proposed topologies and metrics of topologies are discussed in Section 3. Section 4 describes the design setup of the Network simulator (NS-2.35), Python programming is used in the modelling and simulation of three topologies using the Dijkistras algorithm. Section 5 describes the three topologies results by using TCP Protocols with FTP and CBR traffic patterns. Finally, Section 6 draws the conclusion.

Related Work

Tetala Neel Kamala et al. conducted an evaluation of different topologies[2] in NoCand found that the Mesh and Torus topologies outperformed the Binary Tree and Butterfly Fat Tree topologies in terms of node throughput analysis. Better throughput and performance consistency. Increased complexity and area utilization compared to Mesh. In their proposal, B Joshi et al., suggested that Hypercube and Torus topologies[1] with 16 cores outperformed Fat Tree, Mesh, Mesh of Tree (MOT), King Mesh, M Mesh, Ring, and Star topologies. Additionally, the paper demonstrated that a 64-core MOT topology produced superior throughput results compared to the other topologies evaluated. According to the proposal by A Q Ansari et al., the Torus topology[3] exhibits superior throughput performance compared to the Mesh, Cmesh, and Fat Tree topologies. JenitaPriya et al. proposed that Ring topology[4] occupies less area utilization than Mesh and Torus topologies. CharnarurPanem et al. proposed that on the performance of Mesh topology[5] at various sizes, FTP achieved superior throughput results than CBR when packet size was varied. However, in terms of delay, CBR traffic outperformed FTP traffic. As the queue size increased, FTP demonstrated better throughput results than CBR traffic, but both traffic types experienced higher delays. Panem et al. proposed that using the XY routing algorithm[6] for 2D and 3D Mesh topologies yields better throughput and latency results. The FTP traffic pattern obtains better throughput than the CBR traffic pattern. According to KusumKardam et al., research, the Torus topology[7] yielded superior throughput results compared to the Mesh topology. Additionally, when using the TCP protocol for the FTP traffic pattern, the results were found to be more favorable than when using the User Datagram Protocol (UDP) for the CBR traffic pattern. K. Balamurugan et al., summarize NoC types and associated tools.[8] The Author suggests the use of NS2 and NS3 for NoC routing and performance evaluation, including speed and power consumption analysis.

TOPOLOGIES

The topology[9] significantly influences the performance, efficiency, and scalability of a system in a Network on Chip (NoC). To ensure optimal system design, it is essential to carefully consider factors like communication latency, bandwidth, fault tolerance, and throughput when designing the NoC topology.[10] Therefore, a well-designed NoC topology plays a critical role in determining the overall system's effectiveness. The design methodology for a topology is shown in Figure 1. In Network-on-Chip (NoC) design, Mesh topology[11] is a widely used network architecture as shown in Figure 2. It involves connecting processing nodes in a grid-like pattern, where each node is linked to its neighboringnodes directly. Depending on the complexity of the network, the Mesh topology can be either two-dimensional (2D) or three-dimensional (3D). In a 2D Mesh topology, each node is connected to its four adjacent nodes, whereas in a 3D Mesh topology, each node is connected to its six adjacent nodes. The direct connection between nodes results in low-latency communication and high bandwidth. It provides flexibility in network design, as nodes can be added or removed with ease without affecting the overall network performance. Spidergon topology[12] is utilized in the design of Network-on-Chip (NoC) as shown in Figure 3.

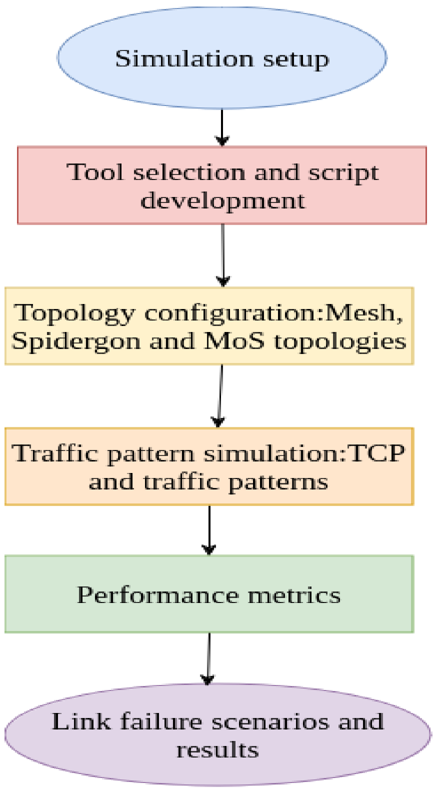
Simulation setup
Tool selection and script development
Topology configuration:Mesh, Spidergon and MoS topologies
Traffic pattern simulation:TCP and traffic patterns
Performance metrics
Link failure scenarios and results

Fig. 1: Designing methodology of a topology.

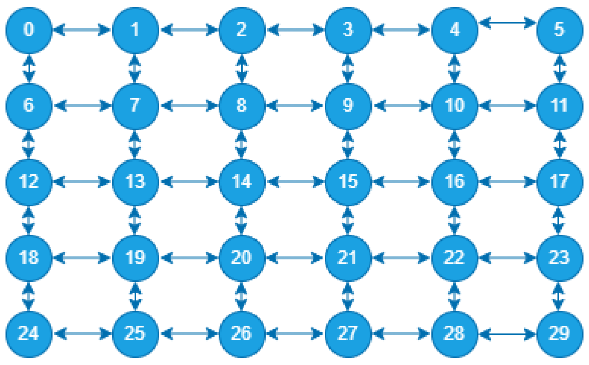

Fig. 2: A 30 node Mesh topology.

It is based on a Mesh network that connects processing nodes using a combination of vertical and horizontal links. The nodes in the Spidergon topology are organized in a grid-like fashion, similar to a Mesh topology. The key difference between Spidergon and traditional Mesh topologies lies in adding diagonal links. These links create a spider-like pattern, giving the topology its name. Using diagonal links improves the connectivity of the network and reduces communication latency between nodes. It also provides benefits such as increased bandwidth, reduced latency, and improved fault tolerance.

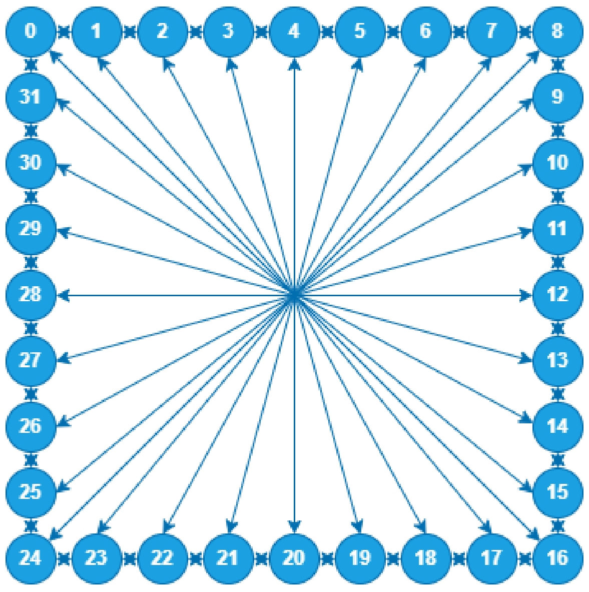

Fig. 3: A 32 node Spidergon topology.

Mesh and Spidergon are two topologies commonly used in network-on-chip (NoC) design. The Mesh topology is a well-known topology in which nodes are arranged in a regular two-dimensional grid, and each node is directly connected to its four adjacent nodes as shown in Figure 4. In contrast, the Spidergon topology is a newer approach that provides certain advantages over the Mesh topology. Hence Mesh is integrated with Spidergon topology. This integration entails using the Mesh topology for local connections and the Spidergon topology for global connections. In this hybrid topology, nodes are placed in a two-dimensional Mesh but are also linked to other nodes in the Spidergon hierarchy, allowing for a more efficient routing system and reducing the number of hops necessary to reach the destination node. This integration of Mesh and Spidergon topologies can increase the network's performance, scalability, and energy efficiency.

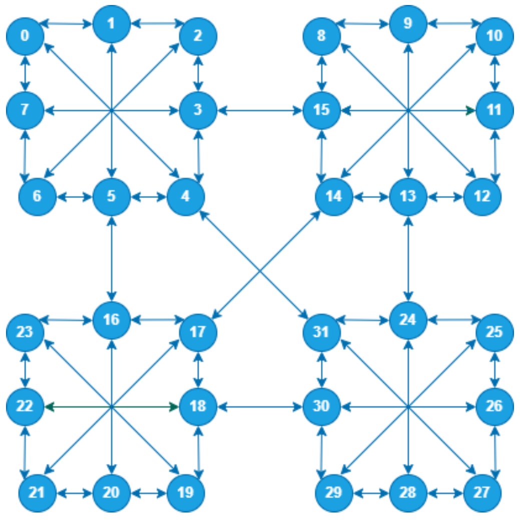

Fig. 4: A 32 node MoS topology.

In the realm of Network-on-Chip (NoC) topologies, there are key performance metrics that play a pivotal role in assessing network efficiency and scalability. Let's discuss these metrics and delve into the values attributed to the provided topologies:

1.1 Network Diameter (N.D)

- Network diameter serves as a crucial element for the utmost number of hops or links between any two nodes (or routers) within the network.
- It serves as an indicator of the worst-case latency experienced during communication between any pair of nodes in the network.
- A smaller network diameter is highly desirable, as it signifies reduced latency for communication. For the respective topologies:
- Network Diameter of Mesh is 9 mm, Spidergon is 8 mm and MoS is 5 mm.

The network diameter is calculated using equation 1.

$$D[neighbor] = \min(D[neighbor], D[current] + W(current, neighbor)) \quad (1)$$

Where D is Distance W is weight.

3.2 Average Network Diameter (A.N.D)

- Average network diameter calculates the mean distance (in hops or links) between all pairs of nodes situated within the network.
- A lower average network diameter generally signals superior communication efficiency. For the provided topologies:
- Average Network Diameter of Mesh is 3.6 mm, Spidergon is 4.61 mm and MoS is 5 mm.

Examining the average network diameter, the Mesh topology emerges as having the smallest value, implying that it is potentially more efficient, on average, for communication between pairs of nodes. In summation, the selection of a NoC topology hinges upon the precise requirements of our application:

- If minimizing latency for all-to-all communication is of paramount importance, MoS presents itself as the favored choice due to its petite network diameter.
- Should the emphasis be on average-case communication efficiency, the Mesh topology may be the preferred option, given its lower average network diameter.

Table 1 shows the network diameter, number of routers, and number of links calculated for three topologies.

In this paper, Mesh, Spidergon, and MoS topologies are evaluated under throughput and latency and also with link failures the topologies were evaluated. The importance of NS-2.35 in the design of various topologies and metrics plays a vital role in assessing the performance of various topologies. So the metrics can be evolved by using Dijkistra's algorithm[13] in Python. These are described in the design setup as follows.

Table 1: Network diameter calculation in three topologies

Topologies	Network Diameter	Average Network Distance	Routers	Link
Mesh	9	3.6	30	49
Spidergon	8	4.61	32	48
MoS	5	5	32	54

Design Setup

NS-2.35[14] is a well-known network simulator that is open-source and mainly used for simulating wireless and wired networks, as well as Network-on-Chip (NoC). This simulator is based on discrete event simulation and operates on a packet-by-packet basis, enabling network designers and researchers to evaluate network algorithms and protocols in a simulated environment. The proposed topology MoSwas designed in the NS-2.35 simulator. NS-2.35 is a flexible simulator used to model the performance and behavior of various NoC topologies novel topologies. The design of these topologies is accomplished using TCL scripts, which allow for the specification of network nodes, links, routing protocols (such as TCP), and traffic patterns. Once the simulation is executed, AWK scripts are employed to analyze the results, extracting key performance metrics such as throughput and latency as shown in Figure 1. Finally, the evaluated results are saved for further analysis, providing valuable insights into the NoCs performance evaluation.

Python with NetworkX is a versatile and powerful set of tools for working with networking graphs. Different scenarios were considered in three different topologies and comparative analysis was done in them.In Python import networkx[15] as nx statement serves to bring the NetworkX library into the codebase and assign it as nx. This shorthand enables to utilize the functions and classes from NetworkX more succinctly.

NetworkX is a robust library designed for the generation, examination, and depiction of intricate networks, often represented as graphs. This practice enhances code readability and ease of use when working with network-related operations and matplotlib. pyplot is a widely used Python module for crafting diverse visualizations such as charts, plots, and graphs. With its capabilities, it allows users to create clear and insightful representations of nodes, edges, and their interconnections within a network. The nodes and edges are created as topology as shown in Figure 5. Dijkstra's algorithm stands as a widely adopted method for determining the shortest paths within a weighted graph. The procedural steps of this algorithm can be as follows: Initialization: Assign the source node distance as zero and the remaining node distances as infinity. Establish a priority queue (or min-heap) to manage the provisional distances efficiently. Explore Neighbors: Extract the node with the smallest tentative distance from the priority queue. For each neighboring node: Compute the tentative distance from the starting node to the neighbor through the current node. If this calculated distance is smaller than the presently recorded distance for the neighbor, update the distance. Termination: Conclude the algorithm when all nodes have been visited or the destination node has been reached. Outcome: The ultimate recorded distance values convey the shortest distances from the initial node to every other node in the graph. Utilizing Dijkstra's algorithm facilitates the computation of the shortest path. Subsequently, by utilizing this the metrics such as average network distance and network diameter can be derived.

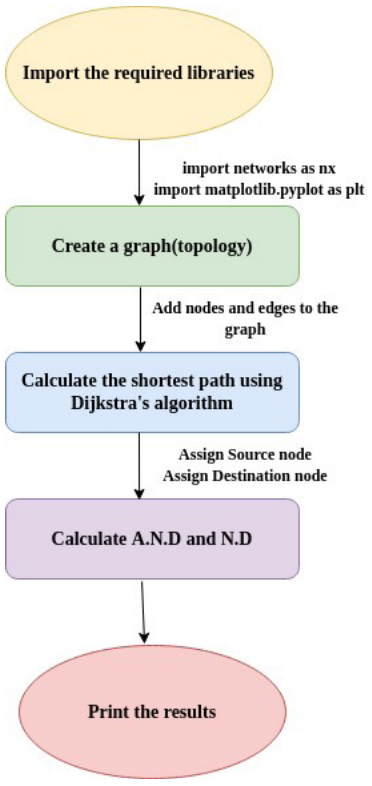
Import the required libraries
import networks as nx
import matplotlib.pyplot as plt
Create a graph(topology)
Add nodes and edges to the graph
Calculate the shortest path using Dijkstra's algorithm
Assign Source node
Assign Destination node
Calculate A.N.D and N.D
Print the results

Fig. 5: Dijkstra's algorithm in a topology's flow chart.

Results

This section introduces the concepts of throughput[2] and latency in NoC [16] topologies performance along with link failures at 5%, 10% and 15%.

5.1 Throughput

The time taken to transfer packets efficiently from source to destination is throughput. A simplified formula for computing throughput is expressed as equation 2.

$$\text{Throughput} = \frac{\text{Datapackets successfully transmitted}}{\text{Time taken for transmission}} \quad (2)$$

This equation essentially evaluates the volume of data successfully transmitted within a specified time interval. It's imperative to acknowledge that in NoC systems, throughput may fluctuate based on parameters such as packet size, and network topology. Consequently, a comprehensive analysis of NoC throughput performance necessitates the holistic consideration of these factors. As shown in figure 6, the results of throughput in TCP using the FTP traffic pattern in three topologies. The MoS topology was found to have higher throughput than the other two topologies. Spidergon topology was found next to MoS topology, and Mesh topology seemed to have a lower throughput than MoS and Spidergon topologies.

5.2 Latency

The time delay experienced by the data packets to be transferred from source to destination is known as latency.

$$\text{Latency} = (Et- St), \text{ if packets arereceived at the node 'r'} = 0, \text{ therwise} \quad (3)$$

In this equation, the latency is computed as the time elapsed between the last packet being received at the destination node and the transmission of the first packet from the source node. If packets are received at the destination node 'r', then the latency is calculated as the difference between the arrival time E_t and the start time S_t. Conversely, if no packets are received at 'r', the latency is considered as zero which was shown in equation 3. The topologies with nodes, and links are set with bandwidth. Then, the connections between traffic agents TCP and FTP traffic patterns are established. All these are written in TCL scripts. Network Distance was calculated with Dijkstra's algorithm using Python. Throughput and latency are key performance metrics for network topologies. By simulating different scenarios in NS-2.35 and analyzing the topologies using Dijkstra's algorithm can gain insights into how means network distance affects these metrics. Using tools like Python and NetworkX can streamline this analysis, providing a clear picture of network topology performance.

The research is carried out by considering the bandwidth set to 50 Mbps. The propagation time is 10 milliseconds. TCP protocol with FTP and CBR traffic patterns are observed in three topologies are shown in Table 2. Mesh topology throughput using FTP traffic scenario diminishes as packet size increases from 1000 bytes to 2048Kbytes as compared to Spidergon and MoS topologies. With improved results, Spidergon topology was observed at 1187.698 Kbytes and MoS topology at 1627.577 Kbytes. However, MoS topology outclasses Mesh and Spidergon topology as packet size increases. The MoS topology gives a higher throughput than the remaining two topologies. MoS topology gives a higher throughput of 112.57% than Mesh topology and throughput of 14.64% than Spidergon topology at 2048 Kbytes was observed using the FTP traffic scenario. Mesh topology produces comparable results using CBR traffic patterns. It generates similar results by utilizing FTP and CBR traffic patterns. Spidergon and MoStopologies generate nearly equivalent results up to 16000 bytes of packet size. However, from 32 Kbytes to 2048 Kbytes, MoS topology throughput increased quite gradually than Spidergon and Mesh topologies. MoS topology gives a higher throughput of 168.08% than Mesh topology and throughput of34.75% than Spidergon topology at 2048 Kbytes was observed using the FTP traffic scenario. Overall, three topologies utilizing both FTP and CBR patterns are demonstrated.

Especially when we observe Spidergon and MoS topologies by using FTP and CBR patterns, Spidergon and MoS topologies are observed with low throughput using CBR up to 16000 bytes. But later these topologies yield higher throughput results using CBR traffic patterns.

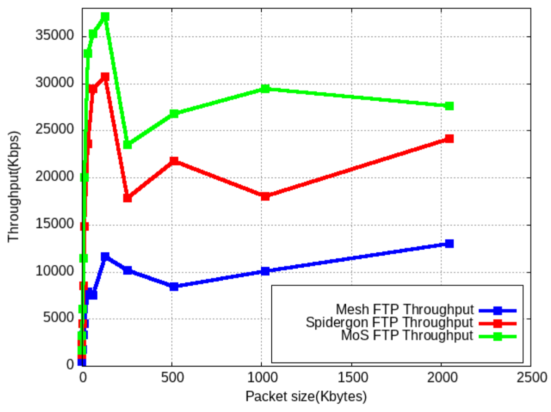

Fig. 6: Throughput in TCP using FTP traffic pattern in three topologies.

As shown in figure 6, the results of throughput in TCP using the FTP traffic pattern in three topologies. The MoS topology was found to have higher throughput than the other two topologies. Spidergon topology was found next to MoS topology, and Mesh topology seemed to have lower throughput than MoS and Spidergon topologies.

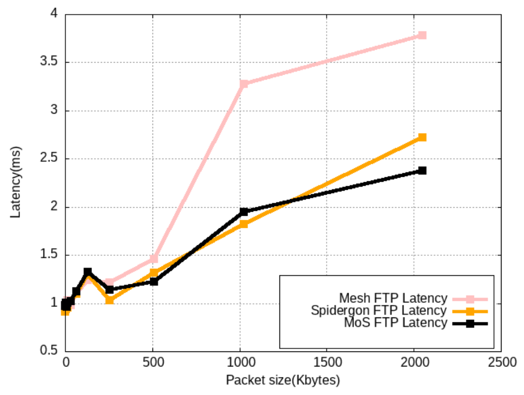

Fig. 7: latency in TCP using FTP traffic pattern in three topologies.

Figure 7 depicts the graph with the latency of all three topologies using an FTP traffic pattern. Mesh topology has been uncovered to have a higher latency than MoS and Spidergon topologies. The results of Spidergon and MoS topologies are comparable. In addition, MoS topology was reported to have lower latency than Spidergon topology.

The graph in figure 8 depicts the throughput of all three topologies using the CBR traffic pattern.[8]

Mesh topology throughput spiked once, then decreased precipitously. MoS and Spidergon topologies have comparable throughput, and MoS topology generated even better results.

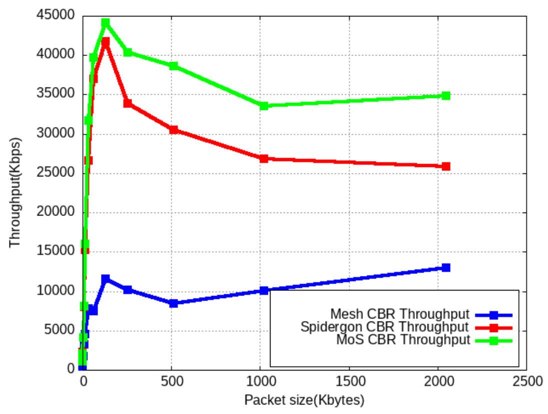

Fig. 8: Throughput in TCP using CBR traffic pattern in three topologies.

The latency of all three topologies using the CBR traffic pattern in the TCP protocol is shown in figure 9. Mesh topology outperforms MoS and Spidergon topologies in terms of latency. However, at 2048 Kbytes of packet size, all topologies generate similar results. Figure 9 shows the latency in TCP using CBR traffic pattern. The MoS topology's higher throughput can be efficient to transmit more packets per unit of time. However, the higher latency indicates that these packets are taking longer to reach their destination. As bandwidth increases, the MoS topology maintains high throughput, but latency decreases. This suggests that with more available bandwidth, the network can handle the higher data rate more efficiently, reducing delays Mesh topology offers redundancy but at the cost of higher latency and lower throughput.

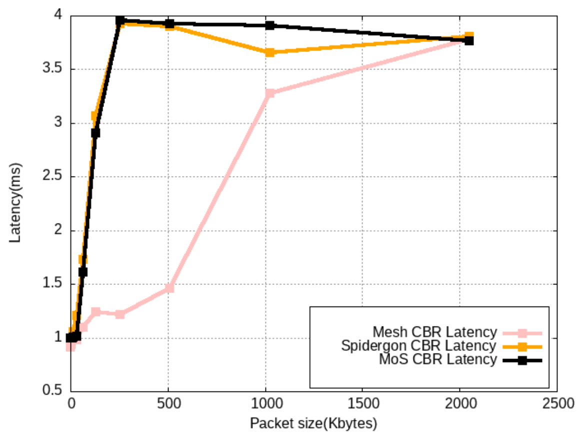

Fig. 9: Latency in TCP using CBR traffic pattern.

Table 2: Throughput using TCP protocol with FTP and CBR traffic scenario in three topologies

Packet size (P.S) (Kbytes)	Mesh FTP (Kbps)	Spidergon FTP (Kbps)	MoS FTP (Kbps)	Mesh CBR (Kbps)	Spidergon CBR (Kbps)	MoS CBR (Kbps)
1	457.023	1187.698	1627.577	457.023	1047.864	1048.033
2	886.170	2303.485	3157.397	886.170	2053.462	2054.123
4	1716.463	4462.256	6062.397	1716.463	4059.807	4062.419
8	3273.218	8509.838	11374.645	3273.218	8053.205	8063.543
16	4491.351	14832.306	19974.757	4491.351	15251.714	16004.469
32	7833.460	23605.968	33157.419	7833.460	26639.307	31645.385
64	7456.644	29425.636	35243.652	7456.644	36971.365	39748.254
128	11595.927	30693.722	37083.725	11595.927	41797.346	44101.528
256	10143.357	17852.719	23450.079	10143.357	33904.563	40381.478
512	8432.360	21734.216	26791.258	8432.360	30496.227	38635.123
1024	10006.977	18012.558	29454.562	10006.977	26876.366	33529.541
2048	12987.680	24081.222	27608.379	12987.680	25838.689	34817.735

Table 3: Latency using TCP with FTP and CBR traffic scenario in three topologies

Packet size (P.S) (Kbytes)	Mesh FTP (ms)	Spidergon FTP (ms)	MoS FTP (ms)	Mesh CBR (ms)	Spidergon CBR (ms)	MoS CBR (ms)
1	0.910937	0.910938	0.971456	0.910937	0.992800	0.992640
2	0.921177	0.921178	0.982176	0.921177	0.993600	0.993280
4	0.941657	0.941658	1.002323	0.941657	0.995200	0.994560

Packet size (P.S) (Kbytes)	Mesh FTP (ms)	Spidergon FTP (ms)	MoS FTP (ms)	Mesh CBR (ms)	Spidergon CBR (ms)	MoS CBR (ms)
8	0.982617	0.982618	0.961324	0.982617	0.998400	0.997120
16	1.028608	0.951674	1.008601	1.028608	1.051706	1.002240
32	0.981676	1.020691	1.020422	0.981676	1.202746	1.012480
64	1.099347	1.096883	1.119321	1.099347	1.732160	1.611148
128	1.236710	1.301530	1.325856	1.236710	3.063360	2.903308
256	1.211654	1.032627	1.135539	1.211654	3.926938	3.956499
512	1.457395	1.319328	1.223193	1.457395	3.89535	3.922963
1024	3.274675	1.819264	1.946950	3.274675	3.657792	3.909312
2048	3.784608	2.721523	2.373830	3.784608	3. 804614	3.764601

Spidergon improves on these slightly with a lower network diameter but has longer average paths. MoS topology provides the best performance in terms of both throughput and latency due to its efficient structure and high interconnectivity, minimizing both the longest and average path lengths. It was reported in Table 3 with latency using TCP protocol with FTP and CBR traffic scenarios in three topologies: Mesh, Spidergon, and MoS. The bandwidth was set to 50 Mbps. As shown in Table 2, throughput increased as packet size increased, as well as latency increased as packet size spiked in Table 3. It was noticed that in Table 3 the latency of Mesh topology initially was less. However, with an increase in packet size, the latency increases in all three topologies using the FTP traffic pattern. Mesh topology had the highest latency of 3.784608 ms.Spidergon topology was observed with a lower latency of 2.721523 ms at 2048 Kbytes of packet size than MoS and Mesh topologies by using FTP traffic pattern in TCP protocol. At 2048 Kbytes of packet size, MoS topology had lower latency than Mesh topology, with 2.721523 ms. The latency of Mesh topology was higher than MoS topology with 59% and Spidergon topology with 14% than MoS topology at 2048 Kbytes. The latency of the three topologies was observed to be comparable up to 16 Kbytes of packet size using the CBR traffic pattern. However, for Spidergon and MoS topologies, latency increases. After 512 Kbytes of packet size is applied, the latency of the Mesh topology also increases. However, at 2048 Kbytes, the MoS topology's latency drops to 3.764601ms. The latency of Mesh topology was higher than MoS topology with 0.53% and Spidergon topology with 1.06% than MoS topology at 2048 Kbytes. Overall, the latency is higher for all three CBR traffic pattern topologies. Table 4 depicts the throughput of three topologies with 5%, 10%, and 15% random link failures using the FTP traffic pattern. Even when the packet size is increased, the results remain comparable in Mesh topology. It was observed that the throughput increases constantly. However, at 15% link failure, the throughput suddenly dropped after 512 Kbytes in the FTP scenario. The results of FTP traffic patterns are constant from 1024 Kbytes of packet size. However, in the first simulation analysis, the links in the Spidergon topology failed at random up to 5%, 10% and 15% random link failures are performed and simulated. Even with random link failures, the throughput of the Spidergon topology was changed and reduced by using the FTP traffic pattern. The throughput of MoS topology with link failures 5%, 10%, and 15% are observed with the FTP traffic scenario and the results are shown in Table 4. It was noticed that the throughput constantly increased in all three link failure cases and the same results were obtained. Overall MoS topology yields better output than the remaining two topologies even with all link failures.

Table 5 shows the throughput with link failures in a TCP by CBR traffic scenario. In CBR traffic patterns, increasing packet size caused a spike in throughput for Mesh topology. However, by using CBR traffic patterns, the throughput remained constant despite all random link failures in the Spidergon topology. CBR throughput was consistent at 2048 Kbytes of packet size despite a 15% link failure. For all cases in CBR, the throughput observed was at 2048 Kbytes in the Spidergon topology. At 2048 Kbytes the throughput drops in the CBR scenario, the throughput suddenly drops to zero in MoS topology. And, even with link failure rates, the throughput of the MoS topology is higher than that of the other two topologies.

Table 6 demonstrates the latency of three topologies using TCP protocol by FTP traffic scenarios with link failures. The latency was the same with all link failures in Mesh topology. The latency results of Spidergon topology with FTP traffic patterns with link failures of 5%, 10%, and 15% are shown. The latency of Spidergon topology varies after 64 Kbytes packet size. The latency result of an MoS topology with TCP protocol using FTP traffic patterns is demonstrated. As seen in the FTP results up to 64 Kbytes of packet size, the result with 2.286243 ms is the same. The latency was then reduced to 2.311870 ms at 128 Kilobyte packet size. The same results were obtained in all link failures 5%, 10% and 15%. When compared to the FTP traffic pattern Spidergon topology was observed with lower latency results than the other two topologies.

Table 7 demonstrates the latency of three topologies using TCP protocol by CBR traffic pattern with all link failures. Mesh topology was observed with higher latency results and it is constant in all link failure cases. It was observed that CBR scenario the latency is 4.397218 ms across all link failures and packet sizes. Spidergon topology was observed to have lower latency results than mesh topology. The results are observed uniformly and drop to zero at 2048 kbytes. The latency result of the MoS topology was lower when compared with the remaining two topologies. Overall, MoS topology was observed to have lower latency using CBR traffic patterns in all link failures.

Table 4: Throughput of three topologies using FTP traffic scenarios with link failure

P.S (Kbytes)	Mesh 5% (Kbps)	Spidergon 5% (Kbps)	MoS 5% (Kbps)	Mesh 10% (Kbps)	Spidergon 10% (Kbps)	MoS 10% (Kbps)	Mesh 15% (Kbps)	Spidergon 15% (Kbps)	MoS 15% (Kbps)
1	184.612	507.298	**551.724**	186.083	354.312	**552.172**	186.924	233.178	**553.068**
2	359.765	988.338	**1076.602**	361.237	687.340	**1077.050**	362.077	418.194	**1077.946**
4	710.073	1950.420	**2126.360**	711.544	1353.397	**2126.808**	712.385	788.225	**2127.703**
8	1410.688	3874.583	**4225.874**	1412.159	2090.499	**4226.322**	1413.000	788.225	**4227.218**
16	2025.267	6535.848	**7302.362**	1689.603	4162.675	**7302.810**	2027.579	1821.353	**7303.706**
32	3369.607	10795.797	**12114.169**	3371.078	2090.499	**12114.617**	3371.919	1819.133	**12115.513**
64	3591.842	14462.068	**14795.669**	3144.640	7117.005	**14796.117**	3594.259	4306.187	**14797.013**
128	6281.923	16786.334	**18171.651**	6283.395	8536.149	**18172.094**	6284.235	5725.830	**18172.980**
256	5384.015	12808.668	**9861.111**	5385.487	5692.532	**9861.558**	5386.432	3686.612	**9862.454**
512	3589.881	11865.040	**12547.929**	3591.352	9974.748	**12548.377**	3592.193	3461.807	**12549.273**
1024	2.452	10609.567	**10755.784**	3.923	7449.236	**10756.231**	4.764	40.762	**10757.127**
2048	2.452	7015.794	**6802.253**	3.923	7963.074	**6802.678**	4.764	40.762	**6803.496**

Table 5: Throughput of three topologies using CBR traffic scenarios with link failures

P.S (Kbytes)	Mesh 5% (Kbps)	Spidergon 5% (Kbps)	MoS 5% (Kbps)	Mesh 10% (Kbps)	Spidergon 10% (Kbps)	MoS 10% (Kbps)	Mesh 15% (Kbps)	Spidergon 15% (Kbps)	MoS 15% (Kbps)
1	95.987	339.094	**505.77**	96.752	339.094	**505.774**	97.188	339.094	**505.774**
2	186.95	658.891	**982.55**	187.718	658.891	**982.553**	188.155	658.891	**982.553**
4	368.88	1281.70	**1910.06**	369.651	1281.708	**1910.068**	370.088	1281.708	**1910.068**
8	732.75	2245.58	**3667.33**	733.518	2245.581	**3667.33**	733.954	2245.581	**3667.33**
16	1051.93	4192.39	**6270.89**	877.609	4192.396	**6270.898**	1053.13	4192.396	**6270.898**
32	1750.12	6512.54	**10137.6**	1750.88	6512.548	**10137.64**	1751.32	6512.548	**10137.64**
64	1865.54	9181.98	**13117.6**	1633.28	9181.989	**13117.60**	1866.79	9181.989	**13117.60**
128	3262.64	9631.64	**14107.5**	3263.40	9631.641	**14107.59**	3263.84	9631.641	**14107.59**
256	2796.31	8958.74	**13255.7**	2797.07	8958.748	**13255.74**	2797.56	8958.748	**13255.74**
512	3589.88	8037.92	**13694.5**	3591.35	8037.923	**13694.5**	3592.19	8037.923	**13694.5**
1024	2.452	8912.17	**11140.2**	3.923	8912.173	**11140.21**	4.764	8912.173	**11140.21**
2048	2.452	0	0	3.923	0	0	4.764	0	0

Table 6: Latency of three topologies in a TCP protocol by FTP traffic scenarios link failures

P.S (Kbytes)	Mesh 5% (ms)	Spidergon 5% (ms)	MoS 5% (ms)	Mesh 10% (ms)	Spidergon 10% (ms)	MoS 10% (ms)	Mesh 15% (ms)	Spidergon 15% (ms)	MoS 15% (ms)
1	2.283708	2.16979	**2.86243**	2.28370	2.16979	**2.86243**	2.28370	2.16979	**2.86243**
2	2.283708	2.16979	**2.86243**	2.28370	2.16979	**2.86243**	2.28370	2.16979	**2.86243**
4	2.283708	2.16979	**2.86243**	2.28370	2.16979	**2.86243**	2.28370	2.16979	**2.86243**
8	2.283708	2.16979	**2.86243**	2.28370	2.16979	**2.86243**	2.28370	2.16979	**2.86243**
16	2.283708	2.16979	**2.86243**	2.28370	2.16979	**2.86243**	2.28370	2.16979	**2.86243**
32	2.283708	2.16979	**2.86243**	2.28370	2.16979	**2.86243**	2.28370	2.16979	**2.86243**
64	2.283708	2.16979	**2.86243**	2.28370	2.16979	**2.86243**	2.28370	2.16979	**2.86243**
128	2.283708	2.31970	**2.31187**	2.28370	2.16979	**2.31187**	2.28370	2.16212	**2.31187**
256	2.283708	2.24001	**2.86243**	2.28370	2.16979	**2.86243**	2.28370	2.46346	**2.86243**
512	2.283708	2.41797	**2.86243**	2.28370	2.46574	**2.86243**	2.28370	2.39203	**2.86243**
1024	2.283708	2.31792	**2.86243**	2.28370	2.20181	**2.86243**	2.28370	2.16197	**2.86243**
2048	2.283708	2.16197	**2.41062**	2.28370	2.16197	**2.41062**	2.28370	2.16197	**2.41062**

Table 7: Latency of three topologies in a TCP protocol by CBR traffic scenarios link failures

P.S (Kbytes)	Mesh 5% (ms)	Spidergon 5% (ms)	MoS 5% (ms)	Mesh 10% (ms)	Spidergon 10% (ms)	MoS 10% (ms)	Mesh 15% (ms)	Spidergon 15% (ms)	MoS 15% (ms)
1	4.397	0.908	**0.888**	4.397	0.908	**0.888**	4.397	0.908	**0.888**
2	4.397	0.916	**0.897**	4.397	0.916	**0.897**	4.397	0.916	**0.897**
4	4.397	0.933	**0.913**	4.397	0.933	**0.913**	4.397	0.933	**0.913**
8	4.397	0.859	**0.947**	4.397	0.859	**0.947**	4.397	0.859	**0.947**
16	4.397	0.918	**0.920**	4.397	0.918	**0.920**	4.397	0.918	**0.920**
32	4.397	0.905	**0.935**	4.397	0.905	**0.935**	4.39	0.905	**0.935**
64	4.397	0.948	**0.898**	4.397	0.948	**0.898**	4.397	0.857	**0.898**
128	4.397	0.850	**0.871**	4.397	0.850	**0.871**	4.397	0.850	**0.871**
256	4.397	0.914	**0.772**	4.397	0.914	**0.772**	4.397	0.914	**0.772**
512	4.397	0.509	**0.897**	4.397	0.509	**0.897**	4.397	0.509	**0.897**
1024	4.397	0.919	**0.735**	4.397	0.919	**0.735**	4.397	0.919	**0.735**
2048	4.797	0	**0**	4.797	0	**0**	4.797	0	**0**

- The primary contributions of this study are as follows:Innovative Topology Design: The integration of Mesh and Spidergon topologies into a unified Mesh of Spidergon (MoS) architecture represents a novel approach in NoC design. This hybrid topology leverages the local efficiency of Mesh and the global routing advantages of Spidergon, creating a more balanced and efficient network structure.
- Performance Evaluation: The study provides a comprehensive evaluation of the MoS topology against traditional Mesh and Spidergon topologies. Metrics such as network diameter, average network distance, throughput, and latency are analyzed using TCP applications with File Transmission Protocol (FTP) and Constant Bit Rate (CBR) traffic patterns.
- Resilience Analysis: The performance of the MoS topology is assessed under varying link failure scenarios at rates of 5%, 10%, and 15%. This analysis highlights the robustness of the MoS topology in maintaining superior throughput and comparable latency compared to its counterparts under fault conditions.

Conclusion

Network-on-Chip (NoC) is a communication architecture employed in the design of System-on-Chip (SoC). Topologies play a pivotal role in the design of NoC. The Mesh, Spidergon, and MoS topologies of NoC with network diameter were analyzed using Python programming in the paper. The throughput and latency of these topologies were evaluated under TCP applications with FTP and CBR traffic patterns. The three topologies were also evaluated for throughput and latency, with link failures at 5%, 10%, and 15%. Ultimately, the MoS topology was found to yield better results in terms of throughput. The MoS and Spidergon topologies exhibit comparable latency, and the Mesh topology is observed to have higher latency. The MoS topology can be used at the NoC architectural level, and the results can be used to assess NoC Quality of Service.

References

[1] Joshi,B.Thakur, M.K.(2018). Performance evaluation of various on-chip topologies, *Journal of Theoretical and Applied Information Technology*, 96 (15), 4883-4893.

[2] T. N. Kamal Reddy, A. K. Swain, J. K. Singh, and K. K. Mahapatra (2014). Performance assessment of different Network-on-Chip topologies, *2nd International Conference on Devices, Circuits and Systems (ICDCS)*, Coimbatore, India, 1-5.

[3] A. Q. Ansari, M. R. Ansari, and M. A. Khan (2015). Performance evaluation of various parameters of Network-on-Chip (NoC) for different topologies, *2015 Annual IEEE India Conference*, Gujarat, India,1-4.

[4] JenitaPriyaRajamanickamManokaran, Mohammed A.S. Khalid (2017). Experimental evaluation and comparison of two recent Network-on-Chip routers for FPGAs, *Microprocessors and Microsystems*, 51, 134-141.

[5] Charanarur, P., Rane, U. V., Gad, V. R., Kovendan, A. K. P., Sridharan, D., Gad, R. S., Naik, G. M (2017). Performance analysis of 16x16, 32x32, 64x64 2-D mesh topologies for network on chip application of MIMO, 1757-1764.

[6] Panem, Charanarur, Udaysingh V. Rane, and Rajendra S. Gad (2017). Network on chip performance analysis over 2-D and 3D mesh topologies for CBR and FTP applications, *National Symposium VLSI and Embedded System*, Goa, India.

[7] KusumKardam, Akanksha Singh (2016). A Review on Comparison on Network on Chip(NOC) Using Simulation Tool NS2, *International Journal of Computer Science and Information Technologies*, 7 (3), 1486-1489.

[8] K Balamurugan, S Umamaheswaran, TadeleMamo, S Nagarajan and Lakshmana Rao Namamula (2022). Roadmap for machine learning based network-on-chip (M/L NoC) technology and its analysis for researchers, *Journal of Physics Communications*, IOPscience, 6, 1-15.

[9] Chen, Jie and Li,Cheng and Gillard, Paul (2011). Network-on-chip (NoC) topologies and performance: a review, *Proceedings of the 2011 Newfoundland Electrical and Computer Engineering Conference (NECEC)*, 1-6.

[10] A. Alimi, Romil K. Patel, OluyomiAboderin, Abdelgader M. Abdalla, Ramoni A. Gbadamosi, Nelson J. Muga, Armando N. Pinto and António L. Teixeira (2021). Network-on-Chip Topologies: Potentials, Technical Challenges, Recent Advances and Research Direction, *Network-on-Chip - Architecture, Optimization, and Design Explorations*. IntechOpen.

[11] L. Bononi and N. Concer (2006). Simulation and analysis of network on chip architectures: ring, spidergon and 2D mesh, *Proceedings of the Design Automation & Test in Europe Conference*, Munich, Germany.

[12] Umamaheswari, S., RajapaulPerinbam, J.,Monisha, K., Jahir Ali, J. (2011). Comparing the Performance Parameters of Network on Chip with Regular and Irregular Topologies. *Trends in Network and Communications*, Vol 197. 177-186.

[13] Al-Yateem, Nabeel, et al. “A Comprehensive Analysis on Semiconductor Devices and Circuits.” *Progress in Electronics and Communication Engineering*, vol. 2, no. 1, Mar. 2025, pp. 1-15.

[14] Kavitha, M. “Embedded System Architectures for Autonomous Vehicle Navigation and Control.” *SCCTS Journal of Embedded Systems Design and Applications*, vol. 1, no. 1, 2024, pp. 25-28.

[15] Aric A. Hagberg, Daniel A. Schult and Pieter J. Swart (2008). Exploring network structure, dynamics, and function using NetworkX, Proceedings of the *7th Python in Science Conference (Scipy2008)*GäelVaroquaux, Travis Vaught, and Jarrod Millman (Eds), Pasadena, CAUSA, 11-15.

[16] Zhang, Wang and Hou, Ligang and Zuo, Lei and Peng, Zhenyu and Wu, Wuchen (2010). A Network on Chip Architecture and Performance Evaluation, *2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing*, Hubei, China, Vol 1, 370-373.