*Journal of Computer Science*

Original Research Paper

# Effect of Load and Routing Protocols on Networks on Chip (NoC): An Analysis

**[1]Shaily Jain, [1]Chander Prabha, [2]Ayman Noor, [3]Prakash Srivastava, [4]Mohammad Zubair Khan and [5]Priyadarshini Pattanaik**

*[1]Department of Computer Science and Engineering, Chitkara University Institute of Engineering and Technology, Chitkara University, Punjab, India*
*[2]Department of Computer Science, College of Computer Science and Engineering, Taibah University, Madinah, Saudi Arabia*
*[3]Department of Computer Science and Engineering, Graphic Era (Deemed to be University), Dehradun, India*
*[4]Department of Computer Science and Information, Taibah University, Madinah, Saudi Arabia*
*[5]Faculty of Computer Science and Informatics, Berlin School of Business and Innovation, Germany, France*

*Article history*
Received: 03-06-2024
Revised: 28-06-2024
Accepted: 15-07-2024

Corresponding Author:
Chander Prabha
Department of Computer Science and Engineering, Chitkara University Institute of Engineering and Technology, Chitkara University, Punjab, India
Email: prabhanice@gmail.com

**Abstract:** Communication infrastructure for multi-core Systems-on-Chip (SoCs) is provided by Network-on-Chip (NoC). Point-to-Point (P2P) and bus-based communication systems are NoCs and are two communication channels of NoC that can probably overcome the scalability and performance restrictions of NoC. Latency and throughput are two of the essential characteristic metrics measured for a routing algorithm that affect the performance of a given NoC. This study evaluated and compared static and dynamic routing algorithms Dijkstra and distance vector on the scale of increasing flit length and network traffic. In a network, considering the effects of topology, traffic, buffer, and packet size, the dynamic algorithm performs better than a static algorithm on the network's performance. Moreover, the effect of increased network traffic on throughput and average packet delay with the increase in network size in a fixed MS topology and distance vector routing protocol has been evaluated. The results show that while using the Dijkstra algorithm, the average packet delay reached 50-60 packets/cycle in comparison to the Distance Vector where it reached a maximum of 40 packets/cycle in 4 different topologies. Throughput is achieved up to 100% in both algorithms using various topologies. In only MS topology, throughput reached 100% but packet delay increased to 400 pkts/cycle with an increase in network size.

**Keywords:** Multiprocessor System on a Chip, Networks on a Chip, MS Topology, TS Topology, BFT Topology, Extended BFT Topology, Flit, Routing Algorithm, Dijkstra, Distance Vector

## Introduction

Continuous technological advancements enable the reduction of chip size. Having microscopic transistors squeezed onto silicon chips, designing and integrating a complete system consisting of a massive number of IP blocks on a single chip is becoming technically feasible. Single processors may be adequate for some general and less energy-requiring (slow processing) applications that are typical of early microcontrollers. However, many applications need multiprocessors to achieve fast and high-performance goals.

Multiprocessor Systems-on-Chip (MPSoCs) consists of numerous Processing Elements (PEs) on a Systems-on-Chip (SoC). An on-chip interconnect like AMBA or NoC is used to connect PEs on a chip. In the future, the Network-on-Chip interconnection method will be used because of the non-scalable schemes like shared buses and P2P dedicated links (Henkel *et al*., 2004). A multiprocessor is a set of processors arranged parallelly sharing a single address. A microprocessor is now the most preferred processor and its cost is low. Multiprocessors have the highest absolute performance than any uniprocessor. The application of MPSoC is mainly used in complex embedded applications. Its systems can fulfill present performance requirements that cannot be achieved by systems based on a single general-purpose processor.

Similar PEs are combined to build a homogeneous MPSoC. However, a homogeneous MPSoC supports only

a handful of applications, while a heterogeneous MPSoC can be used for various applications. A heterogeneous MPSoC is composed of particular types of PEs such as Intel (Vangal *et al*., 2007) and the proposed homogeneous MPSoCs with 80 PEs and Tilera (NVIDIA, 2009) with 100 processing elements connected through a NoC. Recently, Toshiba, IBM, and Sony proposed a heterogeneous MPSoC containing one manager processor and eight floating-point units (Kistler *et al*., 2006). By the end of 2021, MPSoC architectures would contain lakhs of PEs assembled on a single chip (Borkar, 2007). Runtime management of tasks is required for dynamic workload applications like multimedia and networking, as these tasks enter into the system at their runtime. At the same time, dynamic mapping techniques are required for mapping tasks at runtime.

The mapping problem to a regular MS-based NoC architecture has been addressed by several existing solutions (Jamali and Khademzadeh, 2009; Ezhumalai *et al*., 2009). There are two popular dynamic routing algorithms distance vector routing and link-state routing. A basic form of link-state routing (NoC-LS) showed the best results in a NoC. The reduction in data rate leads to a zero packet drop ratio (Ali *et al*., 2005). Cho and Choi (2012) presented a Multi-Path, Hybrid Shortest Path Tree (MPHSPT) algorithm, which uses multipath information to reduce the total execution time, which in turn results in a reduction in the packet loss rate. The proposed MP-HSPT algorithm computes the shortest path faster than the HPST and Dijkstra, Dynamic Dijkstra. Ebrahimi and others (Ebrahimi *et al*., 2009) proposed a hybrid path-based multicast algorithm used in MS networks for NoCs. According to their results, the DP, MP, and CP underwent higher average communication delay and high power dissipation rate than the proposed algorithm when using multicasting and high message injection rates. An analysis of the architectural design of the NOC system computation of regular topologies was proposed (Marrakchi *et al*., 2009). Zhang *et al*. (2007), proposed an on-chip interconnected analysis and evaluation of three different metrics power, and bandwidth delay, and presented a system-level floor planning-based Noc synthesis algorithm. Their solution's limitation is that it is based on a slicing floor plan in which core bends and links around cores constrain the router locations. Glass and Ni proposed deadlock and livelock-free wormhole routing algorithms for MS-connected networks (Mohapatra, 1998).

## *Architecture*

MPSoC: Fig. (1) shows an MPSoC architecture and the interconnection network connecting various processors in it. Each processor is termed a "node processor" or PE and comprises one CPU and one or two cache hierarchy levels. When the L1 cache is missed, the CPU goes to access the L2 cache and if there is a miss again in the L2 cache, it will then lead to access to the main memory (Jantsch and Tenhunen, 2003). Updation in both L1 and L2 cache is done with write-through techniques. The MPSoC uses both shared memories as well as memories connected with each node. Memory directory can globally address and access these memories directly. When there is a cache miss in the L2 cache, information about the miss is transmitted by a request packet generated by the L2 cache in the network requesting permission to access the main memory. The main memory then fulfills the request and returns a reply packet with the required data to the node (Pande *et al*., 2005).

Network on Chip (NoC): NoC is a system designed for communication between IP cores installed on an integrated circuit referred to as a chip in a system on a chip. NoCs are broadly categorized into synchronous and asynchronous clock domains. There is a substantial advantage of NoC technology over conventional bus and crossbar interconnections when networking methods are applied to on-chip communication (Hu and Marculescu, 2004).

MPSoCs have shared memories to exchange data between processors and the interconnected network is used to transport data from one processor to the other. Data flows are first divided into packets and then transmitted to their destinations. Network traffic on a chip comes from the following sources (Benini and De Micheli, 2002):

- Memory transactions: The network traffic increases due to a cache miss, necessitating the data to be fetched from the shared memories
- Cache coherence operations: In MPSoC, data is replicated in the cache of every node. If there is any updation in any cache or main memory, its cache copies also need to be updated with the new values or reset their valid bits. This synchronization operation creates overhead in traffic on the network
- Packet overheads: Data are transmitted in the form of packets thus creating an additional overhead of traffic, which in turn depends upon the packet size
- Contentions between packets: Routes of the packets need to be changed due to contention between packets, leading to regular network traffic diversion

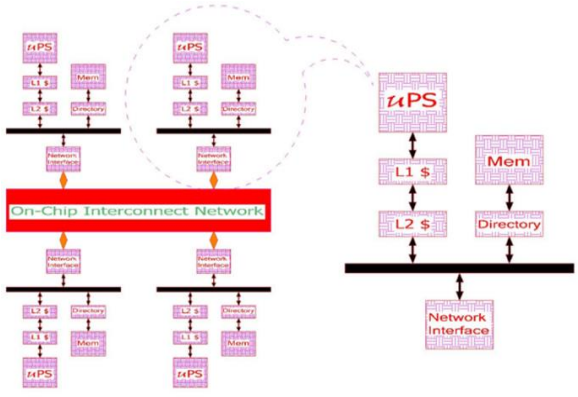


**Fig. 1:** MPSoC architecture

Multiprocessors on a chip can communicate with each other in two ways (Benini and Bertozzi, 2005).

Way 1; Signal address: In this scheme, all processors share a single memory address space in the system without any restriction on accessibility. Communication between different processors is done through these shared variables in memory. It is divided into:

- Uniform memory access multiprocessors (or symmetric multiprocessors) take the same time to access main memory. It means the access time is independent of the location of the word and processor
- Non-uniform memory access multiprocessors allow some memory accesses faster than others as the access depends on the processor and the word's location. Non-uniform memory access machines show high performance, as they are scalable

Way 2; message passing: In this scheme, communication between multiple processors occurs by explicitly sending and receiving messages. There are two basic constructed organizations possible in a network (Dally and Towles, 2001). Figure (2) shows processors connected by a single bus. The number of processors can vary from 2-32. Figure (3) shows processors connected by a network without a direct link between memory and processor.

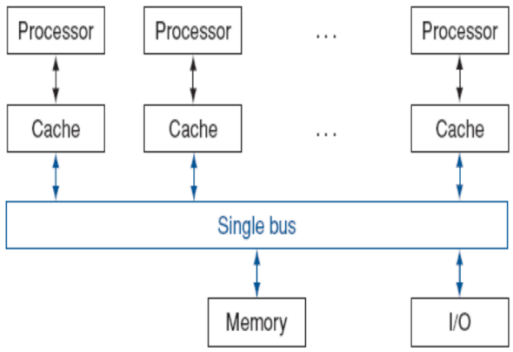


**Fig. 2:** Single bus architecture

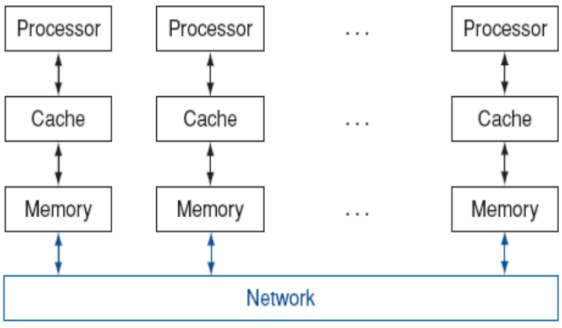


**Fig. 3:** Network architecture

## *Topologies in NOC*

There are various topologies in networks-on-chip. Four different topologies taken into consideration are namely MS, TS, BT, and EBT (Kumar *et al*., 2002; Ezhumalai *et al*., 2011). In the MS network, shown in Fig. (4), switches are connected to the resources and the number of switches directly depends on the number of resources in the network. Each switch is linked to its four neighboring switches except the corner switches present on the layout's edge. This architecture has several benefits viz. smaller switch size, higher capacity physical channels, ease of scalability, and better routing.

Torus: The following topology, shown in Fig. (5) is the TS layout. The TS and MS topology are almost similar, except that TS has a double bandwidth than the MS, as the wires are also wrapped around from the top node to the bottom node and from the right-most node to the left-most node. This architectural layout given for a particular communication packet is used for a distant transmission. Due to the presence of the extra wiring, this architecture has low contention and high bandwidth. On the contrary, due to the deployment of virtual channel switching techniques, it has a buffer of relatively large size.

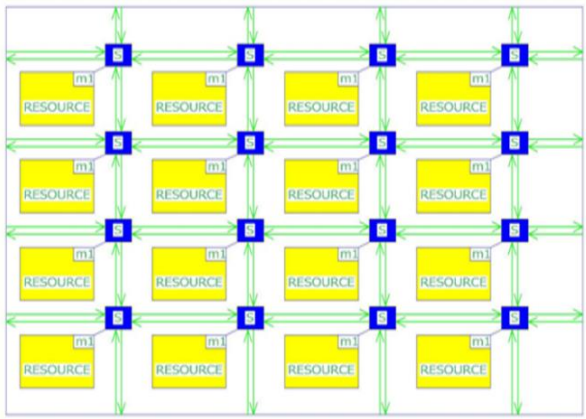


**Fig. 4:** A 4×4 MS topology

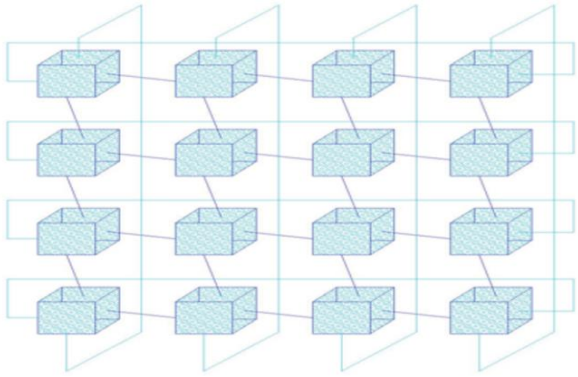

**Fig. 5:** A 4×4 TS topology

Butterfly Fat Tree: A BFT topology looks like a tree structure, as shown in Fig. (6). Each node has two coordinates (position, level) where the position is the node's location in the order from right to left and the level is the same as the level defined in a tree. Leaf nodes are numbered, starting with zero in the vertical order numbering. Each switch in the tree is associated with two parent ports and 34 child ports. In general, if n is the number of IP blocks in the architecture, then the levels in the tree would be log4 n. Moreover, there will be N/2i+1 switches at the $i^{th}$ level of the tree.

Extended butterfly fat tree: The EBT architecture (EBFT) shown in Fig. (7) is derived from the BT topology shown in Fig. (6). The switch size of the architecture is constant. In this network, the leaves have IPs, and switches are situated at the internal nodes. Each switch is connected to further two-parent ports and four child ports. If *x* is the number of IPs, then the number of levels will be log4×.

## *Research Gaps and Problem Formulation*

Table 1 presents the research gaps in previous work done by researchers. Network throughput is the average rate of messages delivered successfully using a communication channel in a network. This data may be transmitted over a physical link or a logical link or pass through a particular network node. The unit of measurement used to measure throughput is bits per second (bit/s or bps). Average packet delay is a concept in packet-switching technology. The Delay in the packet delivery is due to a store-and-forward delay in each router, which further causes the queuing Delay of that packet across the network. Throughput and average packet delay are two crucial factors, which influence network performance. The choice of a good routing algorithm is another influential factor.

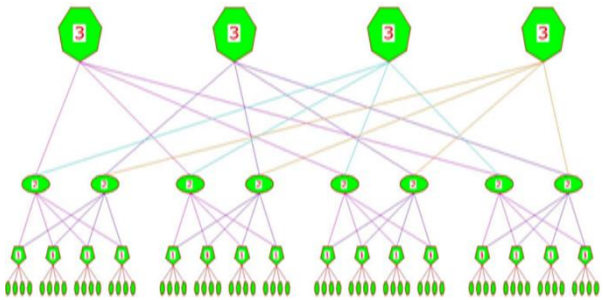

**Fig. 6:** A 4×4 BFT topology

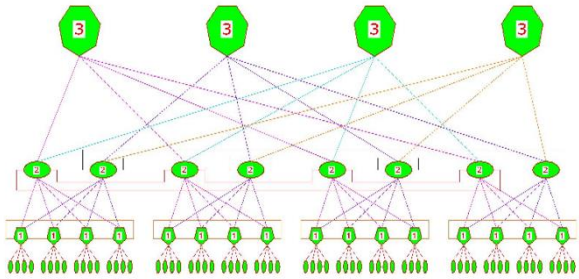

**Fig. 7:** A 4×4 EBFT topology

**Table 1:** Research gaps in previous work

| Reference | Main findings | Methodology used |
|---|---|---|
| Saliu *et al.* (2021) | The XY routing algorithm performed better when the packet injection rate (PIR) was low. The DyAD routing and Age-aware algorithms performed better when the PIR was a high | Varying load at different Packet Injection Rates (PIR) under random traffic patterns using a 4×4 mesh topology was in the Noxim simulator. They used FIFO input buffer channel with a depth of 5 flits, a flit size of 32 bits and a packet size of 3 flits |
| Singh *et al.* (2013) | The XY and OE routing algorithms were compared in terms of average latency, average throughput, and total network power under varying traffic loads on a 3×3 2D mesh topology. The performance of the two routing algorithms were compared | The study using a 3×3 2D mesh topology and routing algorithms evaluated were XY and OE. Simulations were performed using the NIRGAM NoC simulator version 2.1. Simulations where conducted under constant bit rate (CBR) traffic conditions. The performance metrics evaluated were average latency, average throughput, and total network power |
| Bhaskar (2022) | The analysis of the effects of network parameters on power consumption of Network-on-Chip will help in designing new routing and allocation algorithms for efficient NoC operation. The study discusses the trade-offs between power, throughput, and delay metrics for an efficient NoC. The key parameters analyzed include network size, routing, traffic patterns, flit size, and buffer size | Systematically varied network size, routing, and traffic patterns, flit size, and buffer size using Mesh topology measured the power consumption, throughput, and delay |
| Haghi *et al.* (2016) | The study evaluated the effects of the routing algorithm, buffer size, virtual channel, and subnet on the time latency and throughput of on-chip interconnect architectures. | The methodology used in the study involves: Evaluating the effects of four parameters (routing Algorithm. Delay under fully A, XY, and West F; |

**Table 1:** Count.

| | | |
|---|---|---|
| | The goal was to determine the critical points and trade-offs in how these parameters affect overall system performance | buffer size, virtual channel, and subnet) on time latency and throughput in both wire and wireless Network-on-Chip (NoC) approaches. Dividing the study into two parts: 1) Evaluating the effects of routing algorithm and buffer size 2) Evaluating the effects of virtual channels and subnets when switching from wire to wireless NoC using a hybrid topologies. They have used the Booksim and Noxim simulators which are based on System C, to perform the evaluations |
| Nagalaxmi *et al*. (2023) | The proposed deadlock-free shortest routing algorithm has high throughput, low area and power utilization, and lower latency compared to existing algorithms like XY. The proposed the algorithm also has a simpler implementation with lower hardware overhead compared to other algorithms | The key aspects of the methodology are: It abandons the traditional dimensional order routing (e.g. XY routing) where packets always go in the X direction first, which can cause blocking. Instead, it uses a distributed deterministic routing mechanism that treats odd and even columns differently, to reduce network congestion and delay |

Network load, flit length, and network size also contribute to network performance. Hence, the proposed methodology determines the effect of all these parameters on throughput and packet delay on networks on a chip. Figure (8) describes the methodology used.

Routing algorithm: Routing is the process of path selection to send data packets in a given network. Routing is performed for various types of networks like the telephone network, an example of circuit switching. A logical address is assigned to the packet forwarded from its source node to its destination node with the required address traveling through several intermediate nodes like routers in packet switching networks. The routing process usually directs packets to various network destinations based on the route present in routing tables. Thus, creating and maintaining routing tables in the router's memory is crucial for efficient routing. Most routing algorithms use a single dedicated network path at a time, i.e., all packets follow the same path. In TCP/IP, routing can be divided into two types: Static routing and dynamic routing. In static routing, the routing table is prepared and maintained manually using the route command. Since the passage of time, the network expands and the number of gateways increases, it would become challenging to maintain routing tables manually. Hence, this limitation restricts the static routing for a single network communicating only to one or two other networks. While in dynamic routing, daemons automatically update the routing table. Routing daemons continuously receive information through broadcasting from other routing daemons present in the network and continuously updating the routing table.

Dijkstra's static routing algorithm: Static routing is a means to path selection configuration of routers in computer networks. Routers do not communicate about the existing topology of the network in this routing technique. Routes are added to the routing table manually and systems in this network route data through a data network depicted by static or fixed paths. The system administrator updates these static routes into the router. One disadvantage of this type of routing is that when there is an amendment in the network or any network failure occurs between two statically defined nodes, then the packets will not be rerouted. Dijkstra algorithm is an example of static routing (Dijkstra, 1959). This algorithm finds the shortest path with the lowest cost from one vertex to every other vertex in the graph for a defined source node.

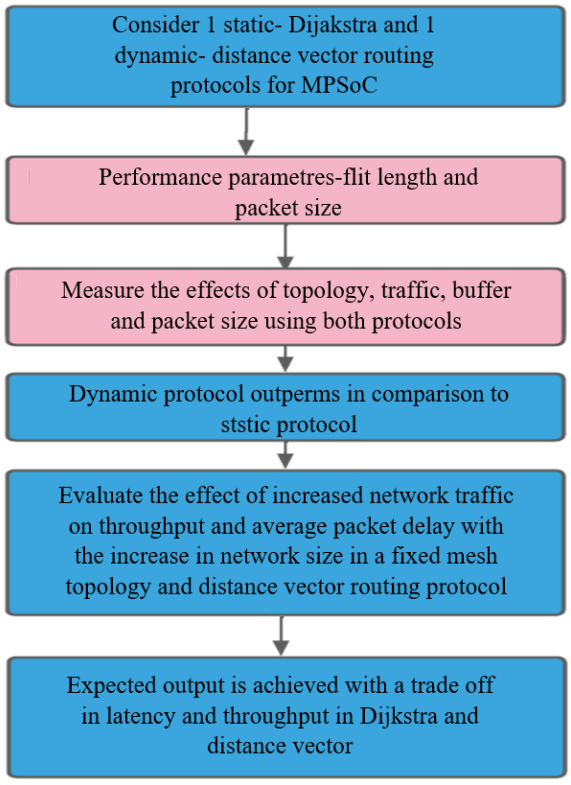


**Fig. 8:** Flowchart of proposed methodology

The shortest paths from a single vertex to a single destination vertex can also be found by stopping the algorithm when the shortest path to the destination vertex has been determined. For example, cities are the vertices of the graph, and distance is represented through the labeled edges between pairs of cities connected by a direct road. Dijkstra's algorithm (Cho and Ryeu, 2006; Newton *et al*., 2009) can be used to find the shortest route from one city to all other cities. Dijkstra's original algorithm does not use a min-priority queue and runs in O (|V|2) time where |V| is the number of vertices.

As Algorithm 1 has only one for loop and if statement nested in while loop, its complexity is calculated as O (m + n log n), where n is the length of the array d i.e., the number of nodes, and m is the size of vector S.

Distance vector dynamic routing protocol: Dynamic routing protocols carry out path determination like static routing and also update the routing table dynamically. It also determines the next-best path if the current path to a destination becomes unavailable. The capability of compensating for any dynamic updation in topology is the most crucial advantage of dynamic routing over static routing. Distance vector protocol is a dynamic routing protocol. Distance-vector routing protocols use both the Bellman-Ford algorithm and the Ford-Fulkerson algorithm for calculating the path. The basic requirement of any dynamic protocol is the timely update of topology changes, so in the distance-vector routing protocol, a router must timely notify its neighbors about changes. Link-state protocols require communicating changes in topology to every router in a network, increasing overhead and complexity. In contrast, distance-vector routing protocols have reduced computational complexity and message overhead.

## Materials and Methods

Windows-based modeling and simulation framework for NoC, Gpnocsim simulator (Hossain *et al*., 2007) is used for experimental evaluation of the problem. Gpnocsim is developed on the Java framework using object-oriented modular design concepts, especially for component-based network environments. It is open-source software and easy to use. Network configurations like topology, size, traffic flow, packet size, Routing Algorithm, etc. can easily be varied for conducting various simulations to find out their impact on various parameters like throughput and Delay. Tables 2-3 describe the simulation environment for each analysis.

Based on research gaps discussed in Table 1, the experimental setup is divided into two parts: (1) Varying 4 different topologies (BT, MS, TS, and EBFT) which are not used in any of the cited papers, varying flit length, varying flits per buffer and two different routing algorithms (Dijkstra and distance vector) which are also not used in any of the cited papers. (2) Only MS topology as many cited papers have used with IP nodes, arrival time, flit length, flits per buffer, and only DV algorithm due to its dynamic nature and better performance in setup 1.

**Algorithm 1: Dijkstra Algorithm in MPSoC**

Step 1: Create vector S which contains all nodes.
Step 2: A new vector I is created, which implements a dynamic array.
Step 3: Create temporary nodes. Distance from x using nodes in I is found. The previous node in the shortest path is set.
Step 4: Initialises set I and arrays d and s.
Step 5: Add a new element in vector S.
Step 6: Integer(i) calls the constructor of Integer class Size () returns the number of elements currently in vector S.
Step 7: Returns the element at the location specified by index i of vector S and the value of the element that was converted in Integer class object will now be converted to int by using the function intValue().
Step 8: Contains () returns true if the vector contains the element; otherwise, it returns false.
i. Re-computes d for non-I nodes and adjusts if necessary for all nodes z, not in I.
Step 9: Finally, return the shortest path.

**Table 2:** Simulation Environment 1 configuration

| Parameter | Value |
|---|---|
| Current network | 0 (BT), 1 (MS), 2 (TS) and 3(EBT) |
| No. of IP nodes | 64 (8×8) |
| Avg. inter-arrival time | 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 |
| Avg. message length | 200 bytes |
| Flit length (bits) | 10, 32 |
| Current VC count | 4 |
| No. of flits per buffer | 10, 32 |
| Traffic type | 0 (Uniform traffic) |
| Routing algorithms | Dijkstra, Distance vec. |

**Table 3:** Simulation environment 2 configuration

| Parameter | Value |
|---|---|
| Current network | 1(MS) |
| No. of IP nodes | 4×4, 8×8, 12×12, 16×16 |
| Avg. inter-arrival time | 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 |
| Avg. message length | 200 bytes |
| Flit length (bits) | 10, 32 |
| Current VC count | 4 |
| No. of flits per buffer | 10, 32 |
| Traffic type | 0 (Uniform traffic) |
| Routing algorithms | Distance vector |

**Algorithm 2: Distance Vector Dynamic Routing Algorithm in MPSoC**

Step 1: Create temporary nodes. d is taken as the distant array. s is taken as the shortest path array.

Step 2: Creates a temporary distant array at each iteration named t.

Step 3: Vector S containing all nodes is initialized. Array d is initialized. Array s is initialized. This establishes the shortest path l of length l for all nodes where z is not equal to x.

Step 4: Find the shortest path of length 2, 3, etc.

I. Copies current array d into array t.
II. d is the name of an array 0 is copied.
III. 0 is the index from which the copying will start from array d.
IV. t is the name of the array that will receive the copy of array d.
V. 0 is the index at which the copy will begin within arrays t and d.
VI. Length specifies the number of elements to be copied. Here the whole array d is copied to array t.

Step 5: Create an object of the format class by using the new Format. This creation instantiates the Format class's constructor and calls the form method by using this object, followed by a dot operator. Find the shortest path with one or more links. Finally, the path is printed. Return to the shortest path.

Step 6: The distance vector algorithm's complexity is calculated as O (mn), where m is the number of nodes and n is the number of edges in the network.

### *Experimental Setup and Testbed*

Reconfigurable topologies allow comparison between different architectures: MS, TS, and BT, which are already implemented in the simulator. To implement routing algorithms, modification is done in the gpnocsim by creating new classes, Dijkstra and Distance_vector that implement the router interface's getDestination method. Two routing protocols: One static Dijkstra's routing algorithm and another, a dynamic, i.e. distance vector routing protocol are implemented. The experimental setup used a network of 64 nodes having four different topologies MS, TS, BFT, and EBFT. The message length was taken as 200 bytes with the current virtual channel count as 4. A uniform traffic in the network is applied rather than random traffic to decrease complexity. The effect of increased network load on different topologies with two different routing algorithms has been analyzed. To achieve this, the message inter-arrival time has been gradually increased from 1-10, which means on average a node produced 1 packet /cycle to 10 packets/cycle.

## Results and Discussion

After the experimental setup, the evaluations have been performed with an average iteration of 10 evaluations per setup. The further section discusses the results obtained after setting simulation environments 1 and 2. Discussion about the results and research gaps are mentioned at the end of this section.

### *Simulation Environment 1*

### *Analysis of the Effect of Network Load and Routing*

For simulation, the gpnocsim simulator has been used with routing algorithms implemented in the Java programming language. It works on a Windows-based machine. When the network load starts with 1 packet/cycle, throughput is 100% taking Dijkstra's routing protocol and flit length of 10 bits, as shown in Fig. (9). However, as the network load increases, throughput decreases and reaches up to 40%. MS topology shows the best performance with a 100% throughput of 5 packets/cycle, but later on, it decreases drastically due to increased network traffic. Extended BFT shows the worst performance with the lowest throughput values.

Moreover, TS delivers better throughput as compared to BFT. It shows that the impact of network traffic is significant in network performance. It will decrease further if network traffic increases and might sometimes lead to 0 also. The throughput and average Delay values indicate that with the increase in the flit length from 10-32, the throughput decreases and the average delay increases.

Figure (10) illustrates an 8×8 network simulation with four different topologies and a flit length of 32 bits using static Dijkstra's algorithm. With the increase in packet size, throughput decreases as compared to flit length 10. Again, MS topology outperforms the other three topologies. Moreover, with the increase in traffic, throughput decreases as in the earlier case.

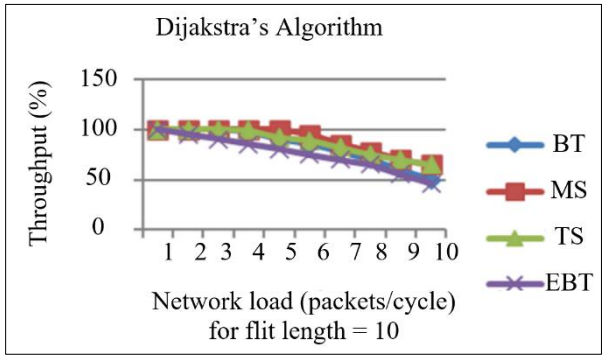


**Fig. 9:** Throughput vs network load for flit length=10 in Dijkstra's algorithm

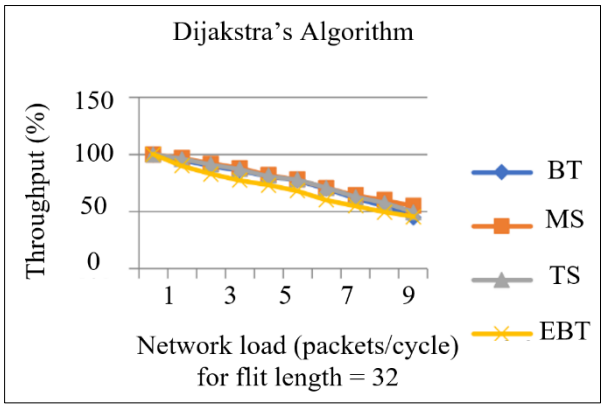


**Fig. 10:** Throughput vs network load for flit length = 32 in Dijkstra's algorithm

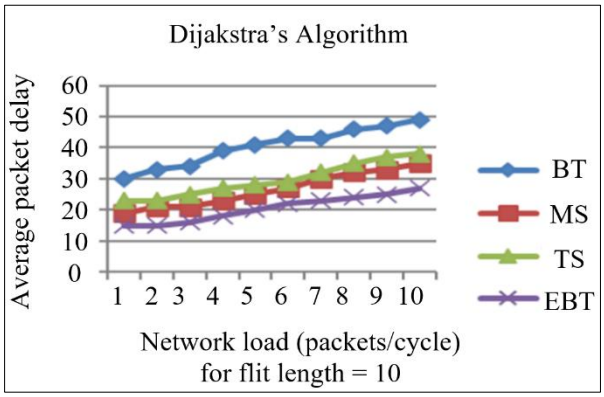


**Fig. 11:** Latency vs. network load for flit length = 10 in Dijkstra's algorithm

The graph in Fig. (11) represents the average packet delay in Dijkstra's algorithm with a flit length of 10 bits. Latency is measured in clock cycles. Its value goes from 15-50 clock cycles. Maximum Delay is observed in BFT and minimum Delay in EBFT, while MS and TS topologies show average behavior. As the network traffic increases from 1 packet/cycle to 10 packets /cycle, the average delay value also increases.

Figure (12) shows the performance evaluation using the average packet delay for flit length as 32 bits. The EBFT outperforms the other three topologies. BFT shows the worst performance with a maximum packet delay of 57 clock cycles. MS and TS topologies show almost similar behavior.

Figure (13) shows the throughput simulation results using four well-known topologies in NOC with a dynamic routing protocol distance vector. The runtime updation in routing table values shows better throughput than Dijkstra's algorithm under the same environment. The throughput decreases with the increase in network traffic. Firstly it starts with 100% throughput but later decreases to 83%. It may be due to the packet losses occurring because of network congestion.

Figures (14-16) illustrate the simulation results of throughput and average packet delay for the NoC environment as described in Table 2. With the increase in network load, throughput decreases, and average packet delay increases in both graphs. MS and TS topologies delivered the best performance for throughput while EBFT outperforms in the case of average packet delay calculation.

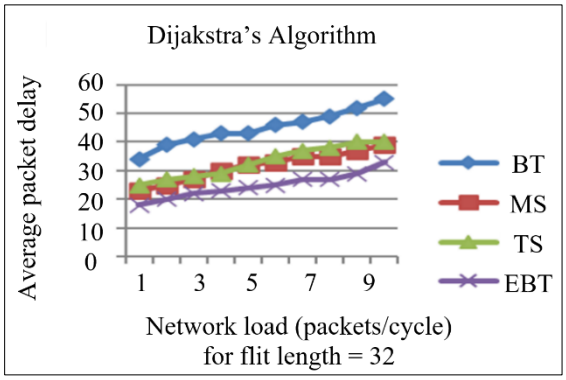


**Fig. 12:** Latency vs. network load for flit length = 32 in Dijkstra's algorithm

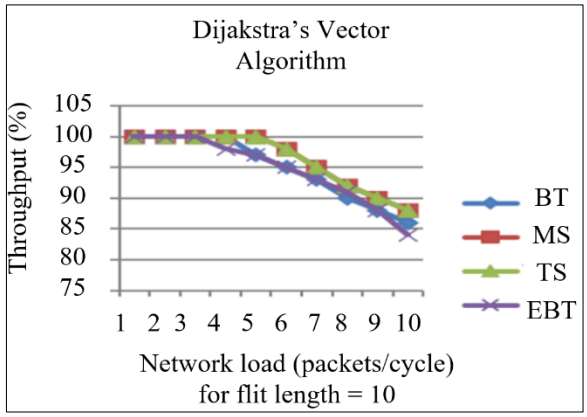


**Fig. 13:** Throughput vs. network load for flit length = 10 in distance vector algorithm

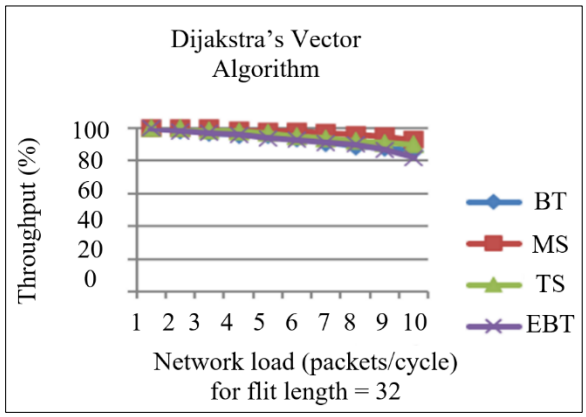


**Fig. 14:** Throughput vs. network load for flit length = 32 in distance vector algorithm

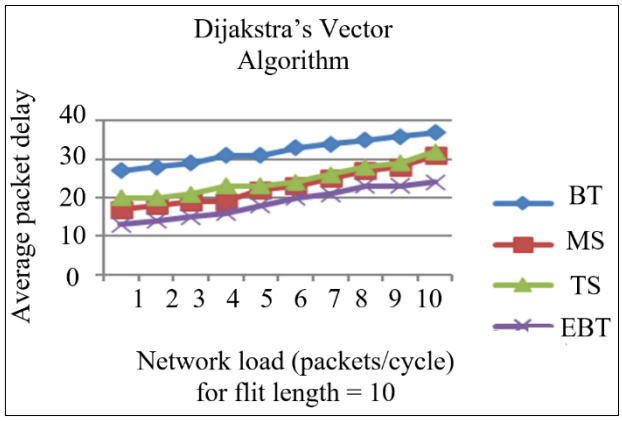


**Fig. 15:** Average packet delay vs. network load for flit length = 10 in distance vector algorithm

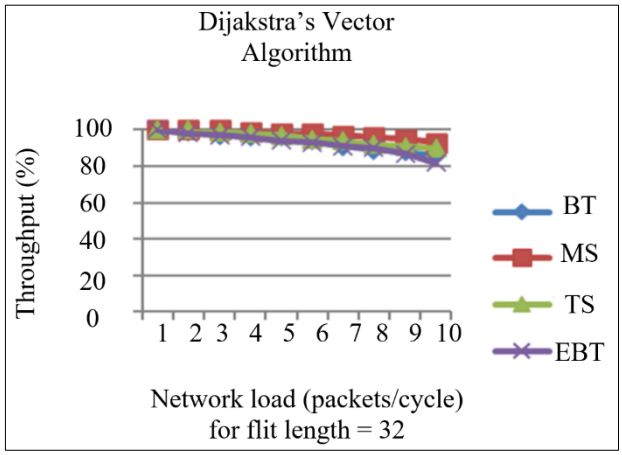


**Fig. 16:** Average packet delay vs. network load for flit length = 32 in distance vector algorithm

## *Results: Simulation Environment 2*

### *Analysis of the Effect of Network Size*

This section depicts the effect of network size on the average delay and throughput.

The simulation environment took the MS topology and distance vector routing protocol for evaluating the effect of network load and size on throughput. While keeping the packet size the same and increasing the network size from 16-256 nodes, the throughput diminished. Figures (17-18), show the effect of network size on throughput with a flit length of 32 bits and a flit length of 10 fits, respectively. Due to the attainment of the saturation point, the throughput decreased after a particular point.

Figure 19 depicts that the average packet delay increased with the increase in network size because of two reasons. The main reason is the increase in the number of contending packets with expanding network size. The supplementary reason is the increase in the average hop count of the packet.

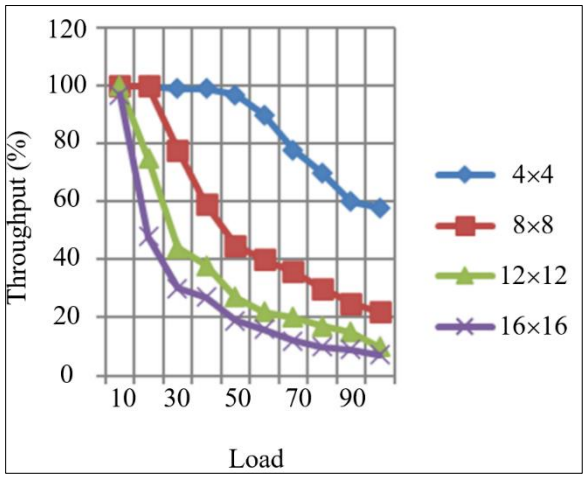


**Fig. 17:** Throughput with an increase in network size (packet size = 32 flits)

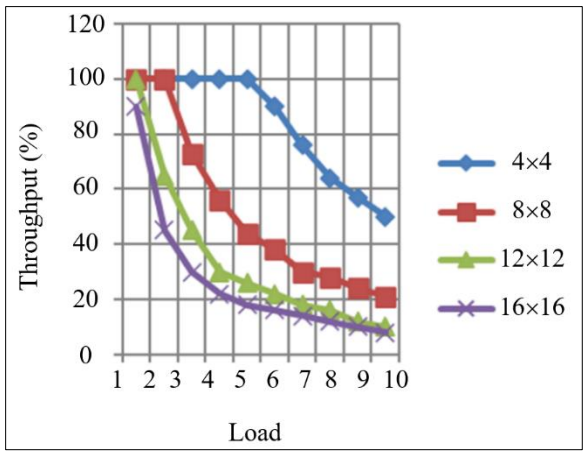


**Fig. 18:** Throughput with an increase in network size (packet size = 10 flits)

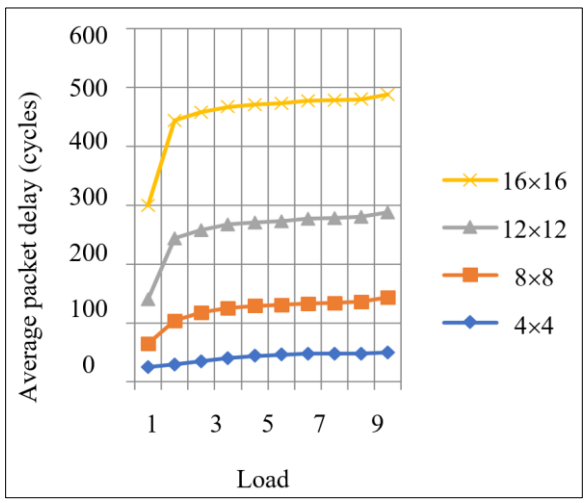


**Fig. 19:** Average packet delay with an increase in network size (packet size = 10 flits)

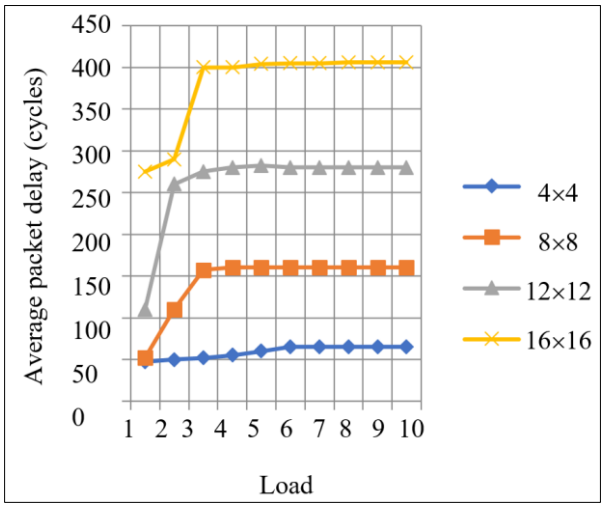


**Fig. 20:** Average packet delay with an increase in network size (packet size = 32 flits)

Figure 20 depicts the average packet delay initially increased with the increase in network load but became almost constant after a limit. With the expansion in network size, it grows exponentially. It shows that network performance is affected by varying network and packet sizes.

The results discussed for simulation environments 1 and 2, clearly indicate that there is an effect of routing protocol, flit length, network topology, and flit buffer on the performance of the network. This study evaluates network performance in terms of average packet delay and throughout, which are major factors of the same. Wireless networks suffer from interference, noise, and bandwidth variation (Khan, 2017). Dynamic routing algorithm performance is better than static routing algorithms due to regular updation in routing tables based on changing topologies (Misra and Sharma, 2013). Throughput decreases and packet delay increases with the increase in traffic load. Due to the increase in traffic in the network, packet loss increases, and hence throughput decreases, resulting in network congestion, finally leading to increased packet delay. MS topology shows the best results with the highest throughput and EBFT with the lowest throughput among the four network topologies as in MS alternative roots in case of high traffic and congestion could be easily found. EBFT shows the best results with the lowest latency and BFT with the highest latency. Due to the highly dynamic nature of the network (Taneja *et al*., 2022), static protocols give poor throughput and utilize more time, and the delay rate is high (Kaur, 2017; Gupta *et al*., 2023). Research papers cited in Table 1 have used only MS topology and not used these algorithms to evaluate the performance of a NoC. Hence, this study filled the research gap till now while using different simulation environments.

## Conclusion

This study performs evaluations in two different setups: One with 4 topologies and 2 algorithms and another with only MS with DV algorithm. Network traffic was varying in each setup. The results proved that MS topology outperforms. When comparing the two routing algorithms, throughput decreases with an increase in network traffic from 100-50% in Dijkstra while it drops to only 85% when using the DV algorithm. Similarly, the average packet delay reached a maximum of 55 packets/cycle in the Dijkstra algorithm in comparison to the distance vector (40 packets/cycle) with an increase in network traffic. Hence, MS topology with Distance Vector algorithm comes out best. So, in the second experimental setup, MS with DV has been chosen. Then the effect of an increase in network size has been evaluated with these scenarios. The throughput of the maximum network size 16×16 dropped to 10% and the average packet delay reached 400 cycles. This proves that with the increase in load and network size performance of the system starts diminishing.

### *Future Work*

The results also prove that the average packet delay and throughput have a remarkable impact due to the varying packet length and network size. This current work finds out the two bases of selecting the routing algorithm. Furthermore, metrics like network power (Singh *et al*., 2013), and area covered (Nagalaxmi *et al*., 2023), can be evaluated with modifications in the network to achieve better results.

In Saliu *et al*. (2021) the main finding of the paper is with different algorithms like XY, Dy AD, and Age aware with varying packet injection rates, and in Bhaskar (2022) delay with full A and West F. So, in the future MS topology, can be implemented with mentioned algorithms with varying PIR, varying load and network size. Different simulators like; Booksim and Noxim (Haghi *et al*., 2016) can help in different setups and evaluations.

## Acknowledgment

Thank you to the publisher for their support in the publication of this research article. We are grateful for the resources and platform provided by the publisher, which have enabled us to share our findings with a wider audience. We appreciate the efforts of the editorial team in reviewing and editing our work, and we are thankful for the opportunity to contribute to the field of research through this publication.

## Funding Information

The authors should acknowledge the funders of this manuscript and provide all necessary funding information.

## Author's Contributions

All authors equally contributed to this study.

## Ethics

This manuscript is an original work. The corresponding author certifies that all co-authors have reviewed and approved the final version of the manuscript. No ethical concerns are associated with this submission.

## References

Ali, M., Welzl, M., & Hellebrand, S. (2005). A dynamic routing mechanism for network on chip. *2005 NORCHIP*, 70–73. https://doi.org/10.1109/norchp.2005.1596991

Benini, L., & Bertozzi, D. (2005). Network-on-chip architectures and design methods. *IEE Proceedings-Computers and Digital Techniques*, *152*(2), 261–272. https://doi.org/10.1049/ip-cdt:20045100

Benini, L., & De Micheli, G. (2002). Networks on chips: a new SoC paradigm. *Computer*, *35*(1), 70–78. https://doi.org/10.1109/2.976921

Bhaskar, A. V. (2022). A Detailed Power Analysis of Network-on-Chip. *2022 IEEE Delhi Section Conference (DELCON)*, 1–7. https://doi.org/10.1109/delcon54057.2022.9752850

Borkar, S. (2007). Thousand core chips: a technology perspective. *Proceedings of the 44th Annual Design Automation Conference*, 746–749. https://doi.org/10.1145/1278480.1278667

Cho, G., & Ryeu, J. (2006). An Efficient Method to Find a Shortest Path for a Car-Like Robot. In D.-S. Huang, L. Kang, Irwin, & G. William (Eds.), *Intelligent Control and Automation: International Conference on Intelligent Computing, ICIC 2006 Kunming, China, August 16-19, 2006* (pp. 1000–1011). Springer Berlin Heidelberg. https://doi.org/10.1007/11816492_131

Cho, T., & Choi, S. (2012). A Multi-path Hybrid Routing Algorithm in Network Routing. *International Journal of Hybrid Information Technology*, *5*(3), 41–46.

Dally, W. J., & Towles, B. (2001). Route packets, not wires: on-chip interconnection networks. *Proceedings of the 38th Design Automation Conference (IEEE Cat. No.01CH37232)*, 684–689. https://doi.org/10.1109/dac.2001.935594

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, *1*(1), 269–271. https://doi.org/10.1007/BF01386390

Ebrahimi, M., Daneshtalab, M., Neishaburi, M. H., Mohammadi, S., Afzali-Kusha, A., Plosila, J., & Tenhunen, H. (2009). An efficent dynamic multicast routing protocol for distributing traffic in NOCs. *2009 Design, Automation & Test in Europe Conference & Exhibition*, 1064–1069. https://doi.org/10.1109/date.2009.5090822

Ezhumalai, P., Aravind, S., & Sridharan, D. (2009). A survey of architectural design and analysis of network on chip systems. *Proceedings of the International Conference on Signals, Systems and Communication*, 21–23.

Ezhumalai, P., Chilambuchelvan, A., & Arun, C. (2011). Novel NoC Topology Construction for High-Performance Communications. *Journal of Computer Networks and Communications*, *2011*(1), 405697. https://doi.org/10.1155/2011/405697

Gupta, D., Rani, S., Tiwari, B., & Gadekallu, T. R. (2023). An edge communication based probabilistic caching for transient content distribution in vehicular networks. *Scientific Reports*, *13*(1), 3614. https://doi.org/10.1038/s41598-023-30315-6

Haghi, M., Thurow, K., Stoll, N., & Moradi, S. (2016). A New Methodology in Study of Effective Parameters in Network-on-Chip Interconnection's (Wire/Wireless) Performance. *International Journal of Advanced Computer Science and Applications*, *7*(10), 75–85. https://doi.org/10.14569/ijacsa.2016.071010

Henkel, J., Wolf, W., & Chakradhar, S. (2004). On-chip networks: a scalable, communication-centric embedded system design paradigm. *Proceedings 17th International Conference on VLSI Design*, 841–851. https://doi.org/10.1109/icvd.2004.1261037

Hossain, H., Ahmed, M., Al-Nayeem, A., Islam, T. Z., & Akbar, Md. M. (2007). Gpnocsim-A General Purpose Simulator for Network-On-Chip. *2007 International Conference on Information and Communication Technology*, 254–257. https://doi.org/10.1109/icict.2007.375388

Hu, J., & Marculescu, R. (2004). Application-specific buffer space allocation for networks-on-chip router design. *IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004.*, 354–361. https://doi.org/10.1109/iccad.2004.1382601

Jamali, M. A. J., & Khademzadeh, A. (2009). MinRoot and CMesh: Interconnection Architectures for Network-on-Chip Systems. *World Academy of Science, Engineering and Technology, International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering*, *3*(6), 1303–1308.

Jantsch, A., & Tenhunen, H. (2003). *Networks on Chip* (1st Ed.). Springer. https://doi.org/10.1007/b105353

Kaur, A. (2017). Vehicular Ad-hoc Network: A Survey. *International Journal of Computer Science Research (IJCSR)*, *5*(2), 35–37.

Khan, M. A. (2017). Video Transmission over Wireless Networks-A Survey. *International Journal of Electronics Engineering Research*, *9*(2), 207–216.

Kistler, M., Perrone, M., & Petrini, F. (2006). Cell Multiprocessor Communication Network: Built for Speed. *IEEE Micro*, *26*(3), 10–23. https://doi.org/10.1109/mm.2006.49

Kumar, S., Jantsch, A., Soininen, J.-P., Forsell, M., Millberg, M., Oberg, J., Tiensyrja, K., & Hemani, A. (2002). A network on chip architecture and design methodology. *Proceedings IEEE Computer Society Annual Symposium on VLSI. New Paradigms for VLSI Systems Design. ISVLSI 2002*, 117–124. https://doi.org/10.1109/isvlsi.2002.1016885

Marrakchi, Z., Mrabet, H., Farooq, U., & Mehrez, H. (2009). FPGA Interconnect Topologies Exploration. *International Journal of Reconfigurable Computing*, *2009*(1), 259837. https://doi.org/10.1155/2009/259837

Misra, R., & Sharma, P. (2013). Challenges in Mobile Ad Hoc Network for Secure Data Transmission. *International Journal of Electrical and Electronics Research*, *1*(1), 8–12. https://doi.org/10.37391/ijeer.010103

Mohapatra, P. (1998). Wormhole routing techniques for directly connected multicomputer systems. *ACM Computing Surveys*, *30*(3), 374–410. https://doi.org/10.1145/292469.292472

Nagalaxmi, T., Sreenivasa Rao, Dr. E., & Chandrasekhar, Dr. P. (2023). Design and Performance Analysis of Low Latency Routing Algorithm based NoC for MPSoC. *International Journal of Communication Networks and Information Security (IJCNIS)*, *14*(1s), 37–53. https://doi.org/10.17762/ijcnis.v14i1s.5590

Newton, P. C., Arockiam, L., & Kim, T.-H. (2009). An Efficient Hybrid Path Selection Algorithm for an Integrated Network Environment. *International Journal of Database Theory and Application*, *2*(1), 31–38.

NVIDIA. (2009). *First 100-core processor with the new tile-gx family*. Events/Press Release. https://www.nvidia.com/en-us/networking/

Pande, P. P., Grecu, C., Jones, M., Ivanov, A., & Saleh, R. (2005). Performance Evaluation and Design Trade-Offs for Network-on-Chip Interconnect Architectures. *IEEE Transactions on Computers*, *54*(8), 1025–1040. https://doi.org/10.1109/tc.2005.134

Saliu, M. S., Omuya Momoh, M., Uchenna Chinedu, P., Nwankwo, W., & Daniel, A. (2021). Comparative Performance Analysis of Selected Routing Algorithms by Load Variation of 2-Dimensional Mesh Topology Based Network-On-Chip. *ELEKTRIKA- Journal of Electrical Engineering*, *20*(3), 1–6. https://doi.org/10.11113/elektrika.v20n3.249

Singh, J. K., Swain, A. K., Kamal Reddy, T. N., & Mahapatra, K. K. (2013). Performance evalulation of different routing algorithms in Network on Chip. *2013 IEEE Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics (PrimeAsia)*, 180–185. https://doi.org/10.1109/primeasia.2013.6731201

Taneja, A., Saluja, N., Taneja, N., Alqahtani, A., Elmagzoub, M. A., Shaikh, A., & Koundal, D. (2022). Power Optimization Model for Energy Sustainability in 6G Wireless Networks. *Sustainability*, *14*(12), 7310. https://doi.org/10.3390/su14127310

Vangal, S., Howard, J., Ruhl, G., Dighe, S., Wilson, H., Tschanz, J., Finan, D., Iyer, P., Singh, A., Jacob, T., Jain, S., Venkataraman, S., Hoskote, Y., & Borkar, N. (2007). An 80-Tile 1.28TFLOPS Network-on-Chip in 65nm CMOS. *2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, 98–589. https://doi.org/10.1109/isscc.2007.373606

Zhang, L., Chen, H., Yao, B., Hamilton, K., & Cheng, C.-K. (2007). Repeated On-Chip Interconnect Analysis and Evaluation of Delay, Power, and Bandwidth Metrics under Different Design Goals. *8th International Symposium on Quality Electronic Design (ISQED'07)*, 251–256. https://doi.org/10.1109/isqed.2007.139

## Appendix

| | |
|---|---|
| Mesh | MS |
| Torus | TS |
| Butterfly Fat Tree | BT |
| Extended Butterfly Fat Tree | EBT |