

Evaluation of genetic algorithm in network-on-chip based architecture

Doraisamy Radha¹, Minal Moharir²

¹Department of Computer Science and Engineering, School of Computing, Amrita Vishwa Vidyapeetham, Bengaluru, India

²Department of Computer Science and Engineering, RV College of Engineering, Bengaluru, India

Article Info

Article history:

Received Apr 25, 2023

Revised Oct 5, 2023

Accepted Nov 16, 2023

Keywords:

Booksim

Genetic algorithm

Latency

Network-on-chip

Power

Throughput

ABSTRACT

An increase in the number of cores gives a significant bounce in performance than an improvement in any of the factors or hardware. Many core systems use network-on-chip (NoC) for efficient communications among the cores in the system. However, the problem with NoC-based communication is that it significantly consumes a large amount of power and energy because the number of routers increases with the increase in the number of cores in the system. Power consumed by such components leads to degradation of the performance. The placement of cores in the topology is non-deterministic polynomial-time hardness (NP-Hard) problem. The optimal placement of cores in NoC is essential as it minimizes latency and communication costs. Thus, the NP-Hard problem of placing cores is solved using genetic algorithm (GA) based quadtree topology. The proposed work shows the analysis of GA-based quadtree topology, which outperforms other topologies in most aspects. The performance evaluation of GA-based quadtree topology is based on latency, throughput, power, area, bisection bandwidth, and diameter. Comparing these parameters with other topologies shows the prominence of the quadtree topology. The evaluation is performed in the Booksim simulator, and the experimental results revealed that the proposed GA-based quad tree-based topology is efficient for NoC-based communications.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Doraisamy Radha

Department of Computer Science and Engineering, School of Computing, Amrita Vishwa Vidyapeetham

Bengaluru, India

Email: d_radha@blr.amrita.edu

1. INTRODUCTION

Technological development and the requirement for fast response in every field emerge in many processing cores. Intel provides an era with supercomputing speed, performance, and compatibility. Emerging technologies like big data analytics, artificial intelligence, and growth in multimedia, the internet of things (IoT) require a high-end system for processing. According to Karl Rupp, according to Moore's law, all factors go through a flat curve, whereas the number of cores only increases [1]. However, doing parallel programming for every problem in multicore/many-core is challenging for programmers. Therefore, there is a high need to utilize the cores effectively and improve performance. Topology, routers, flow control, switches, traffic pattern, routing algorithms, packet size, and buffer size are the contributors to the evaluation of the network of cores, which in turn plays a vital role in the performance of the whole architecture. Based on the arrangements of these routers, cores, and other hardware associated with the network-on-chip (NoC), there are different topologies framed like mesh, ring, and quad, each one has its advantages and disadvantages.

Mesh topology is one of the simplest and most commonly used topologies in the arrangement of cores, and a good amount of research about the performance has been done using mesh networks [2]. But in search of better performance across various factors, including less power consumption, low latency, and high performance. This has led to the discovery of other topologies such as torus, spidergon, quadtree, and Dragonfly. The various simulators allow an analysis of the performance of different topologies by varying the different aspects or components of the architecture like the number of nodes, buffer size, and routing algorithm being used. Routing algorithms play a vital role in using the cores effectively. For each topology, routing algorithms with and without adaptiveness make a notable difference in the evaluation parameters of the performance of the NoC architecture.

A few simulators used for NoC to carry out experiments to measure performance by varying the architecture configuration are Noxim, NocTweak, Nirgam, Booksim, Orion, Garnet, Dsent, Wormsim, Gem5, and a few others are meant for field programmable gate arrays (FPGA) related [3]. Booksim2 simulator is a popular NoC simulator for modeling the interconnection network for many cores. This simulator provides options to work for different topologies, routing algorithms, and different sizes. The configuration can be set accordingly, and the performance can be analyzed. Various modules are available for analyzing flit latency, packet latency, network latency, and throughput. Power and area modules are also available for evaluating power and area for the components used for the configuration settings.

Communication among the cores is also crucial to improve the performance of the multi/many-core systems. In many-core processors, NoC-based communication is considered a better solution for communication among cores when compared with bus-based communication. NoC-based chip-multiprocessors (CMP) consume considerable power for a core, private level 1 instruction, and data caches, a second level cache that could be shared or made private, a router and logic block, links, buffers, and crossbars [1]. Leakage power can also be a NoC's power consumption. The advancements place demands for high performance and low power consumption in embedded systems. Objectives have shifted from high peak performance to power efficiency [4]. Therefore, embedded systems challenge designers because of their power and performance budgets. Thus, processing power is one of the significant demands for applications in which power consumption for other components like high-resolution sensors is also high. The microgrid's performance is completely based on power consumption [5].

The total number of links, routers, and buffers required is dependent on the number of cores N , the number of local buffers, and the virtual channels N_{VC} to be used. It is calculated as given in (1)-(3). C-Mesh is a concentrated mesh like a Mesh with wrap-around links to reduce the diameter of the Mesh topology. For C-Mesh, the requirements are calculated as given in (4) and (5). Dragonfly is a hierarchical network with three levels: router, group, and system. At the bottom level, each router has connections to cores, local routers in the current group, and global routers in other groups. Where N_g represents the number of groups and n_{gr} represents the number of routers in group g . This can be mathematically represented as given in (6) and (7):

$$N_l = 2\sqrt{N\sqrt{N-1}} * N_{VC} \quad (1)$$

$$N_r = n \quad (2)$$

$$N_b = 5 * N_b * N_{VC} * N \quad (3)$$

$$N_l = 2\sqrt{N\sqrt{\frac{N}{2}-1}} * N_{VC} \quad (4)$$

$$N_r = \frac{n}{4} \quad (5)$$

$$N_r = n_{gr} * N_g \quad (6)$$

$$N_l = N_p \quad (7)$$

Dragonflies are constructed for a minimal number of endpoints. The parameter 'p' denotes the number of terminals per router untouched. p is, however, a key factor in the Dragonfly construction, as it determines not only the final scalability of the topology but also the required router radix [6]. Moreover, the total number of links employed in the Dragonflies greatly varies with d and b, and consequently, so does the bandwidth that is made available. If a substantial amount of bandwidth is available within the topology, e.g.,

when the Dragonfly is electrically balanced, it is interesting to populate the S routers with more terminals to exploit the available bandwidth ideally [7].

Quadtree topology is based on a spatial data structure that stores elements in a partitioned 2D space. The 2D space is recursively subdivided into four quadrants. This type of hierarchical arrangement of cores helps to place the elements of the exact location in the same partition. The closest point for the given coordinates can be made using quadtrees. The quadtree has various applications like image processing, geographical distancing, and networking. The quadtree data structure is widely used in digital image processing and computer graphics for modeling spatial segmentation of images and surfaces. An efficient indexing scheme is proposed for a linear (pointerless) quadtree data structure. Such a quadtree is stored using a uni-dimensional array of nodes. The indexing scheme has the property that the navigation between any pair of nodes can be computed in constant time.

Moreover, navigation across multiple quadtrees can also be achieved at the same cost [8]. A distributed quadtree index that adapts the quadtree is described as that enables more robust access to data in peer-to-peer (P2P) networks. The indexing usage is easy, scalable, and has good load-balancing properties [9]. A design, implementation, and performance evaluation of NoC-based multicore architecture for accelerating the median breakpoint problem in phylogeny has been designed. On the network architecture front, the superiority of a quadtree over a mesh in energy efficiency is demonstrated for this application class. The proposed work is to analyze the quadtree topology in NoC architecture for better performance in terms of throughput, power, area, and bisection bandwidth [10]. The quad has been applied over various domains to solve many problems and is presented in many of the author's works comprising greedy-quadtree-greedy routing algorithm on quadtree for guaranteed delivery, lower hops, and load balancing for geographic routing. A node represents a multi-quadtree network with one or more squares, and each square can have one or more representative nodes. This representation helps load balancing and shortest paths in routing [11]. Fault tolerance is required for the optimal functioning of NoC architecture. A mesh of 2*2 routers has been arranged with a spare router placed at the center—the designed quad-spare mesh [12]. NoC seems to replace traditional bus-based communications as it provides high reliability and scalability. Reconfigurable network on chip has been designed to provide encryption, authentication, and denial of service attacks [13].

A genetic algorithm (GA) has been used to construct the decision tree for classifying mobile users. The crossover and mutation operators are prominent in building optimal decision trees. Traditional algorithms consume more time and complex mechanisms to make a decision [14]. The designed GA-based decision tree was compared with support vector machine (SVM) on the breast cancer dataset is 78.50% [15]. The GA has been combined with classification and regression tree (CART) to boost the performance while performing classification. Random CART trees have been generated to make the initial population, and in each iteration, the performance of the CART tree has been boosted using various genetic operators such as crossover and mutation [16].

A GA has been used to construct the distance matrix for phylogenetic tree construction. As the GA uses the natural genetic operator's crossover and mutation, the tree generated is highly trustworthy as the distance matrix is generated by GA [17]. The decision trees generated using C4.5 are subjected to undergo crossover, and mutation still converges to produce an optimal decision tree [18]. The application mapping in NoC is non-deterministic polynomial-time hardness (NP-Hard) problem solved using GA. GA has been used to minimize the communication cost of NoC and improve the convergence of network partitioning [19]. The placement of cores on the device has been solved using adaptive GA to minimize the communication cost. The adaptive GA does not rely on fixed probability to converge towards the minimum global optimal solution. The balance between crossover and mutation has been controlled for the optimal placement of cores [20]–[22]. To snoop data from the adjacent wavelength channels in a shared photonic waveguide, which introduces a severe security threat [23], [24]. The GA is one of the first population stochastic algorithms [25].

2. METHOD

2.1. Genetic algorithm-based quad tree topology

The GA is a meta-heuristic algorithm inspired by the behavior of genetic chromosomes. The population-based GA could act over trees and provide optimal global solutions. Given these advantages, GA is proven to be the best solution for solving problems in which solutions are represented as trees. Regarding GA, a population of a potential solution is taken randomly for the given problem, where each solution is referred to as a chromosome. The objective of GA is to find the chromosome that best evaluates the given fitness function.

This section describes constructing the quadtree organization of chips using GA. Each chromosome is represented as a tree, which is shown in Figure 1. Let c_i represents the i^{th} chip. The link between the chips represents communication. Thus, the population represents a collection of chromosomes for the cores. The fitness function considered for validating the chromosomes is latency and throughput. The GA-based

quadtree has been constructed to minimize latency and maximize throughput. The population in GA has been represented with various quadtrees, and in each iteration, the population members are selected using rank-based selection. The selected parents are subjected to reproduction using a recombination operator, uniform crossover to produce offspring for the next generation. This has been repeated until convergence. At each iteration, the performance of the quadtree is evaluated in terms of fitness, which has minimum latency and maximum throughput. The working of GA-based quadtree is shown in Algorithm 1.

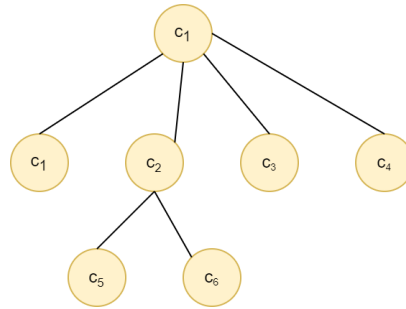


Figure 1. Chromosome in the population

Algorithm 1. GA_Based_Quad_Tree()

Input: Cores c_1, c_2, \dots, c_n

Output: Optimal Quad Tree QT

Initialize random Quad Trees as $P \leftarrow RQT_1, RQT_2, \dots, RQT_n$

While convergence

For each $RQT_i \in RQT$

 Compute fitness using Equation

End For

 Sort the fitness and assign a rank

 Select the parent random quadtrees for crossover

 Randomly select a crossover point

$off_k, off_l = crossover(RQT_i, RQT_j)$

$P \leftarrow P - \{RQT_i, RQT_j\}$

$P \leftarrow P \cup \{off_k, off_l\}$

End While

Return RQT_{Best}

2.2. Fitness function

The fitness function evaluates the individual random quadtree according to the criteria of minimal latency and minimum communication cost. The random quadtree with minimal fitness value is considered the best quadtree construction of a NoC. Fitness is represented in (8). Latency refers to the amount of time taken from starting the job until the completion of the request. Long latency incurs a huge network communication bottleneck, which is a severe threat. Thus, the first objective considered is minimizing latency. In addition, when the chips are communicated when transferring data to get the job done, it is expected to have the minimum communication cost. Thus, the second objective considered is minimizing communication costs, as shown in (10). w_1, w_2 represents the weight associated with latency and communication cost. Where $hop_{i,j}$ represents the path length between i^{th} core and j^{th} core present in the path from source to destination core. $bandwidth_{i,j}$ represents the bandwidth acquired in the link between i^{th} core and j^{th} core.

$$Minf(RQT) = w_1 * latency + w_2 * Cost_Comm \quad (8)$$

Subject to

$$w_1 + w_2 = 1 \quad (9)$$

$$Cost_Comm \leftarrow \sum_{i=1}^n hop_{i,j} * bandwidth_{i,j} \quad (10)$$

2.3. Rank based selection

The chromosomes in the population are sorted according to the fitness value and rank assigned to each chromosome. Then the probability that the chromosome is selected is done according to the rank. Using rank-based selection, the crowding phenomenon has been avoided; thus, the individuals in the population tend to converge faster with an optimal solution. In addition, the problem of stagnation and premature convergence of GA has been eliminated by using this rank-based selection. The sum of the rank has been computed using (11). The probability of selecting the individual based on the rank is given in (12). Where r_i , denotes the rank of the i^{th} chromosome. Pr_i , denotes the probability of selecting the i^{th} chromosome.

$$sum_rank \leftarrow \sum_{i=1}^n r_i \tag{11}$$

$$Pr_i \leftarrow \frac{r_i}{sum_rank} \tag{12}$$

2.4. Crossover

After selecting the individual for recombination using rank-based selection, the individuals are subjected to reproduction. Arbitrarily, a subtree has been chosen from each individual subjected for reproduction, and they are interchanged, as illustrated in Figure 2. After being subjected to reproduction, the individuals in the current generation are replaced with new offspring. This process is repeated until convergence. The diagrammatic representation of the working of Algorithm 1 is shown in Figure 3.

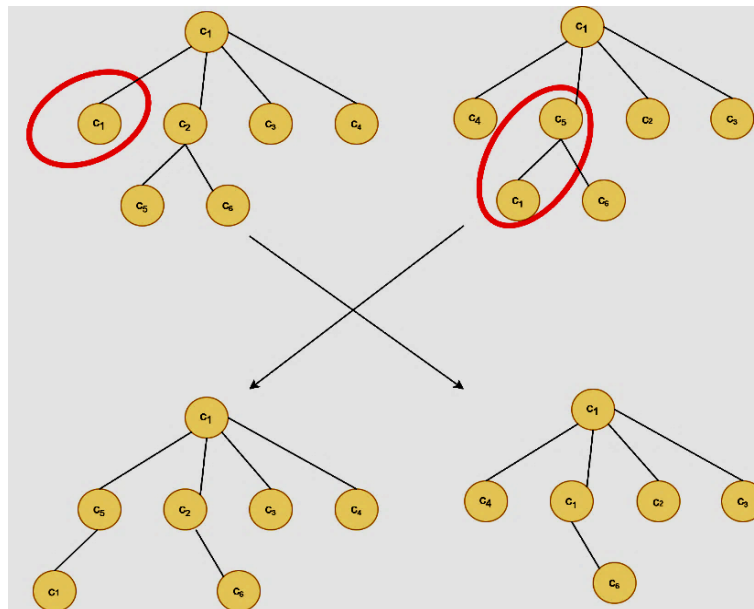


Figure 2. Crossover of individuals in the population

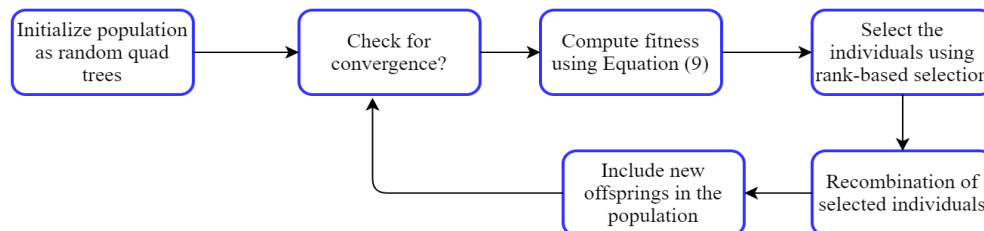


Figure 3. Working of GA-based quadtree

3. RESULTS AND DISCUSSION

The results produced by the simulations on Booksim are shown in the following figures. The results are obtained by varying the several associated parameters related to overall NoC architecture, including

traffic, number of routers, and number of virtual channels. A comparison of these results across various topologies like mesh, C-Mesh, Dragonfly, and fat trees is discussed in this section. In this section, other factors besides latency and processing power are also considered for determining a good architecture. These factors include power consumption and area. By the end of the section, the most powerful architecture could be determined by analyzing various factors.

The experiment set up for our comparison is that there are $16 \times 16 = 256$ nodes, an injection rate is 0.01, and there are 8 virtual channels per router. The traffic followed is uniform so that each core in the Network is performing an equal amount of work as that of the other and it is equally busy as like others. Figure 4 is evident from the fact that the areas of quadtree topology are much less than the other compared architectures, like Mesh, C-Mesh, and Dragonfly. Of all those, C-Mesh performs the worst in all the compared areas, like switch area, channel area, and other components area. The quadtree topology beats all its components in all aspects of the area. It is seen that it occupies only about sq. units of area for fitting all the 256 cores along with their related components like routers, channels, and other connecting components. The details make us consider putting quadtree to use when space is a concern and opening its doors for their use in portable and handheld devices.

Here Figure 4 shows the switch area of topology, Figure 5 shows the channel area of topology, Figure 6 shows the other areas of topology, and Figure 7 shows the total areas of various topologies. In addition, we could see that there is only a linear increase in the area, i.e., for 64 cores, it is 1 square unit, and for $64 \times 64 = 256$ cores, the total area is only $1 \times 4 = 4$ square units, which is impressive. Figure 8 shows the comparison of the total area with an increase in cores in 16 nodes. Figure 9 shows the comparison of the total area with an increase in cores in 256 nodes. The area factor is scalable across various sizes, and it is good with much bigger architectures. Thus, the quadtree is perfect in terms of space and is significantly better than the other topologies, and the increase in space is linear, making it eligible to be used even with bigger core sizes.

As we discussed, power consumption is also an essential factor to consider. Figure 10 shows the consumptions of channel wire power of topologies for 256 nodes, injection rate = 0.01, vcs = 8, and uniform traffic. Figure 11 shows the consumptions of switch power of topologies for 256 nodes, injection rate = 0.01, vcs = 8, and uniform traffic. Figure 12 shows the consumptions of other components' power of topologies for 256 nodes, injection rate = 0.01, vcs = 8, and uniform traffic.

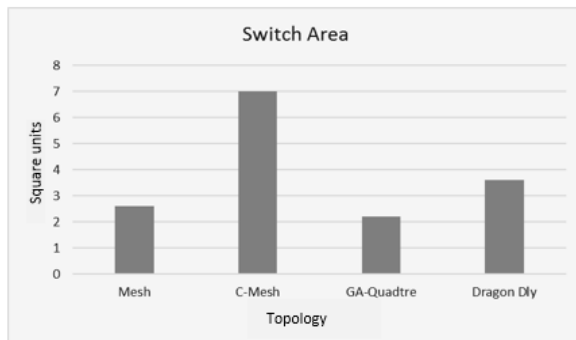


Figure 4. Switch area of various topologies

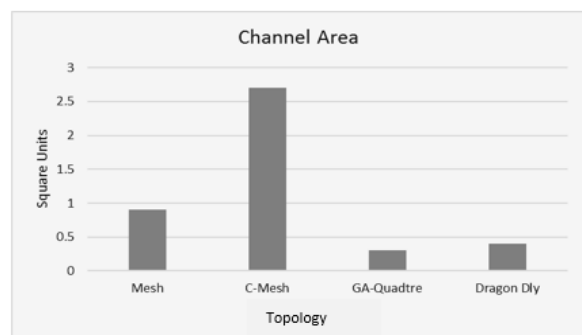


Figure 5. Channel area of various topologies

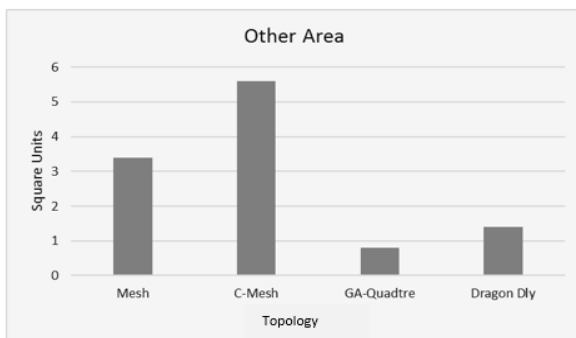


Figure 6. Other areas of various topologies

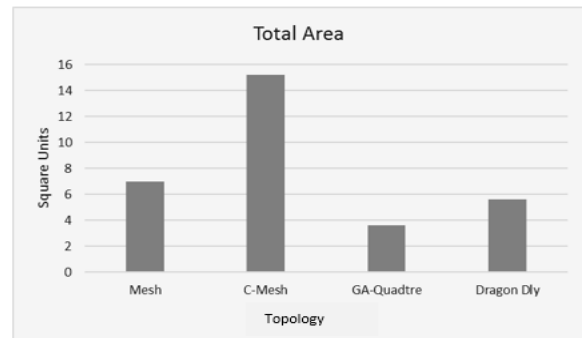


Figure 7. Total area of various topologies

Figure 13 shows the consumptions of the total power of topologies for 256 nodes, injection rate =0.01, vcs =8, and uniform traffic. As we can see in Figure 13, the quadtree topology uses less power than the other topologies. Hence, it uses only around 25 watts when compared to the worst performing C-Mesh using about 400 watts also, we could see this trend is maintained with the increase in the injection rate as there has been only steady growth so therefore it is reasonable to use the quadtree topology for efficient and better power consumption.

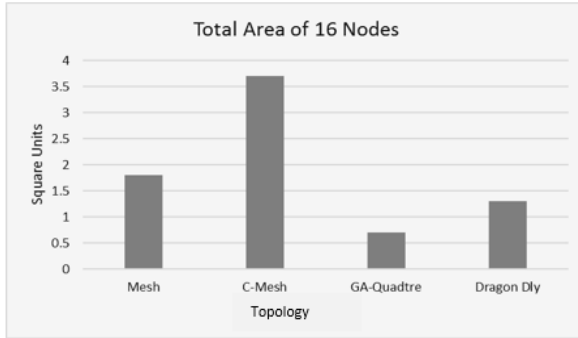


Figure 8. Comparison of total area in topologies

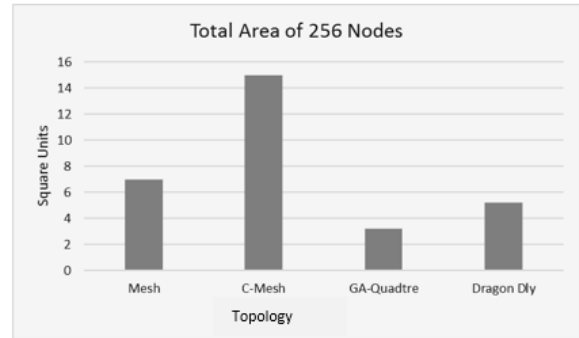


Figure 9. Comparison of total area in topologies

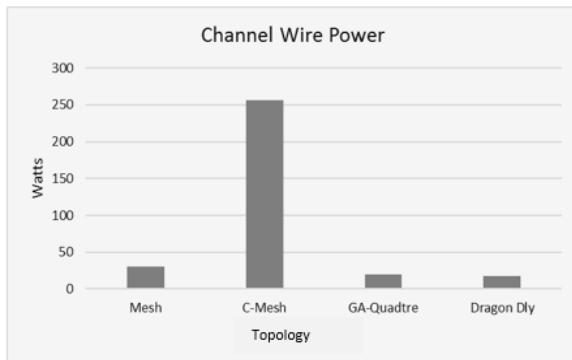


Figure 10. Channel wire power consumption of topologies

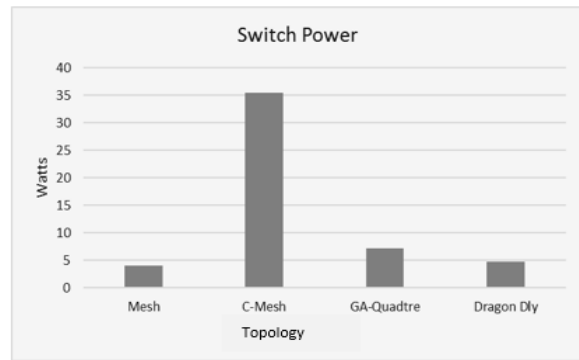


Figure 11. Switch power consumption of topologies

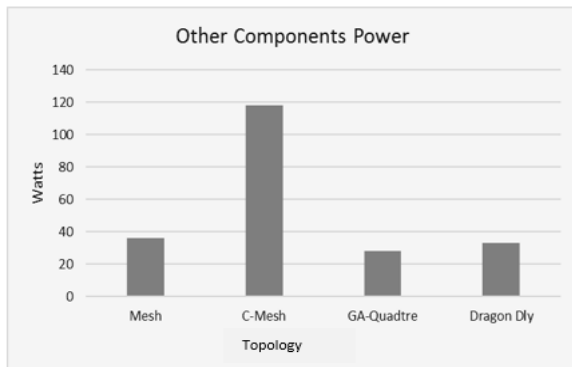


Figure 12. Other components power consumption

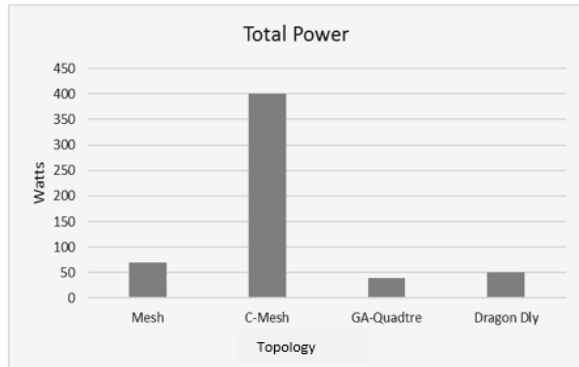


Figure 13. Total consumption of topologies

Figure 14 shows the total power consumption of various topologies with for 64 nodes with 0.01 injection rate. Figure 15 shows the total power consumption of various topologies for 64 nodes with 0.10 injection rate. The comparison of some of the latency-related factors is shown in Figures 16-18 in terms of

latency, hop average, and throughput of various topologies for 16×16 nodes=256 with injection rate =0.01, vcs=8, and uniform traffic. The quadtree performs very much better over the traditional architecture like Mesh, C-Mesh for the given experimental setup with 256 cores and the number of hops taken to travel from the source to destination routed is significantly small in the case of the quadtree. Regarding the above factors, the quadtree is one of the suitable topologies that need to be considered. The throughput is the same for all the topologies, and it remains unaffected by the topology change. In addition, we could see that the Dragonfly topology may perform slightly better in the latency and hop count over the quadtree, but it has its limitations regarding power and area.

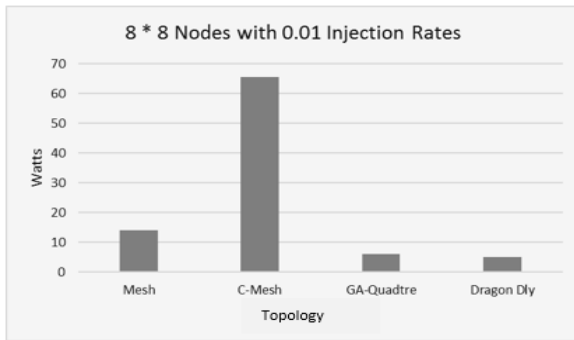


Figure 14. Total power consumption with 0.01 injection rate

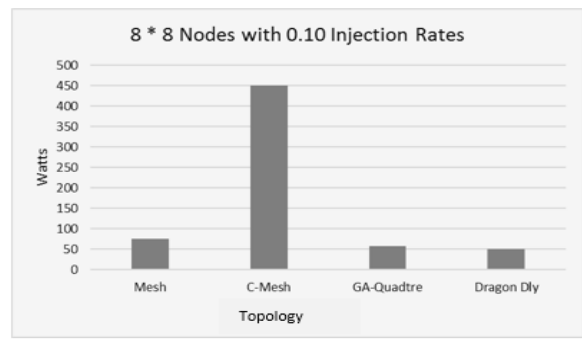


Figure 15. Total power consumption with 0.10 injection rate

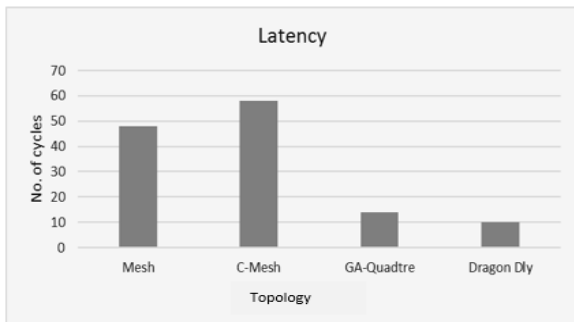


Figure 16. Latency of topologies

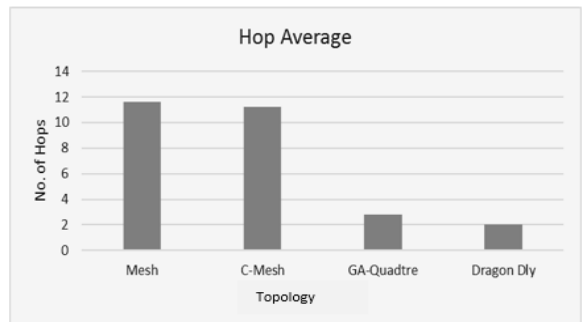


Figure 17. Hop average of topologies

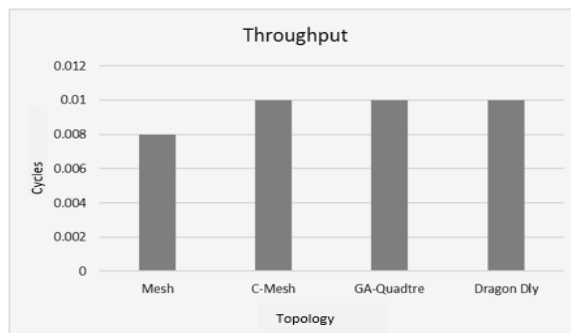


Figure 18. Throughput of topologies

4. CONCLUSION

This paper discusses topologies in the NoC architecture like Mesh, C-Mesh, quadtree, and Dragonfly. In addition, power, area, and other factors are shown as necessary, along with the latency and

computing power in determining a good NoC architecture. Moreover, it is shown that the quadtree topology performs much better in terms of area and power consumption over the other topologies without any compromise in the latency and hop count. The analysis shows that in the power and space-limited world, the quadtree topology is an essential topology to be considered today with the growth of many handheld devices like smartphones and other IoT devices that are power and space-limited in nature. The quadtree topology in their architecture is a good solution for overcoming the problem by minimizing latency and better throughput. Further studies can be done with quadtree topology for the heterogeneous arrangement of cores and their performances.

ACKNOWLEDGEMENTS

I would like to express our sincere gratitude to all those who have supported and contributed to this research project. Primarily, I extend our heartfelt thanks to our guide for his unwavering guidance, invaluable insights, and encouragement throughout the research process. No funding is raised for this research.




REFERENCES

- [1] K. Rupp and S. Selberherr, "The economic limit to Moore's law," *IEEE Transactions on Semiconductor Manufacturing*, vol. 24, no. 1, pp. 1–4, Feb. 2011, doi: 10.1109/TSM.2010.2089811.
- [2] B. B. Yusuf, T. Maqsood, F. Rehman, and S. A. Madani, "Energy aware parallel scheduling techniques for network-on-chip based systems," *IEEE Access*, vol. 9, pp. 38778–38791, 2021, doi: 10.1109/ACCESS.2021.3063901.
- [3] S. Khan, S. Anjum, U. A. Gulzari, and F. S. Torres, "Comparative analysis of network-on-chip simulation tools," *IET Computers & Digital Techniques*, vol. 12, no. 1, pp. 30–38, 2018, doi: 10.1049/iet-cdt.2017.0068.
- [4] E. Ofori-Attah, W. Bhebhe, and M. Agyeman, "Architectural techniques for improving the power consumption of NoC-based CMPs: a case study of cache and network layer," *Journal of Low Power Electronics and Applications*, vol. 7, no. 2, pp. 1–24, 2017, doi: 10.3390/jlpea7020014.
- [5] J. Kim, H. Oh, and J. K. Choi, "Learning based cost optimal energy management model for campus microgrid systems," *Applied Energy*, vol. 311, p. 118630, Apr. 2022, doi: 10.1016/j.apenergy.2022.118630.
- [6] M. Y. Teh, J. J. Wilke, K. Bergman, and S. Rumley, "Design space exploration of the dragonfly topology," in *High Performance Computing*, Cham: Springer, 2017, pp. 57–74. doi: 10.1007/978-3-319-67630-2_5.
- [7] L. Balmelli, J. Kovacevic, and M. Vetterli, "Quadrees for embedded surface visualization: constraints and efficient data structures," in *Proceedings 1999 International Conference on Image Processing (Cat. 99CH36348)*, IEEE, 1999, pp. 487–491. doi: 10.1109/ICIP.1999.822944.
- [8] E. Tanin, A. Harwood, and H. Samet, "Using a distributed quadtree index in peer-to-peer networks," *VLDB Journal*, vol. 16, no. 2, pp. 165–178, 2007, doi: 10.1007/s00778-005-0001-y.
- [9] T. Majumder, S. Sarkar, P. P. Pande, and A. Kalyanaraman, "NoC-based hardware accelerator for breakpoint phylogeny," *IEEE Transactions on Computers*, vol. 61, no. 6, pp. 857–869, 2012, doi: 10.1109/TC.2011.100.
- [10] C. Avin, Y. Dvory, and R. Giladi, "Geographical quadtree routing," in *2011 IEEE Symposium on Computers and Communications (ISCC)*, IEEE, 2011, pp. 302–308. doi: 10.1109/ISCC.2011.5983794.
- [11] Y. Ren, L. Liu, S. Yin, J. Han, Q. Wu, and S. Wei, "A fault tolerant NoC architecture using quad-spare mesh topology and dynamic reconfiguration," *Journal of Systems Architecture*, vol. 59, no. 7, pp. 482–491, 2013, doi: 10.1016/j.sysarc.2013.03.010.
- [12] S. Charles and P. Mishra, "Reconfigurable network-on-chip security architecture," *ACM Transactions on Design Automation of Electronic Systems*, vol. 25, no. 6, pp. 1–25, 2020, doi: 10.1145/3406661.
- [13] K. Kaliyan and R. Kothandaraman, "Secure decision-making approach to improve knowledge management based on online samples," *International Journal of Intelligent Engineering and Systems*, vol. 11, no. 1, pp. 50–61, 2018, doi: 10.22266/ijies2018.0228.06.
- [14] D. Liu and S. Fan, "A modified decision tree algorithm based on genetic algorithm for mobile user classification problem," *The Scientific World Journal*, vol. 2014, pp. 1–11, 2014, doi: 10.1155/2014/468324.
- [15] E. Ersoy, E. Albey, and E. Kayış, "A CART-based genetic algorithm for constructing higher accuracy decision trees," in *Proceedings of the 9th International Conference on Data Science, Technology and Applications*, SCITEPRESS - Science and Technology Publications, 2020, pp. 328–338. doi: 10.5220/0009893903280338.
- [16] M. Gupta and S. Singh, "A novel genetic algorithm based approach for optimization of distance matrix for phylogenetic tree construction," *International Journal of Computer Applications*, vol. 52, no. 9, pp. 14–18, 2012, doi: 10.5120/8229-1303.
- [17] Z. Fu, B. L. Golden, S. Lele, S. Raghavan, and E. A. Wasil, "A genetic algorithm-based approach for building accurate decision trees," *INFORMS Journal on Computing*, vol. 15, no. 1, pp. 3–22, 2003, doi: 10.1287/ijoc.15.1.3.15152.
- [18] Yin Zhen Tei, M. N. Marsono, N. Shaikh-Husin, and Yuan Wen Hau, "Network partitioning and GA heuristic crossover for NoC application mapping," in *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, IEEE, 2013, pp. 1228–1231. doi: 10.1109/ISCAS.2013.6572074.
- [19] J. Gomes Filho, M. Strum, and W. J. Chau, "Using genetic algorithms for hardware core placement and mapping in NoC-based reconfigurable systems," *International Journal of Reconfigurable Computing*, vol. 2015, pp. 1–13, 2015, doi: 10.1155/2015/902925.
- [20] S. V. R. Chittamuru, I. G. Thakkar, S. Pasricha, S. Sri Vatsavai, and V. Bhat, "Exploiting process variations to secure photonic NoC architectures from snooping attacks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 5, pp. 850–863, 2021, doi: 10.1109/TCAD.2020.3014184.
- [21] F. Jafari, A. Jantsch, and Z. Lu, "Weighted round robin configuration for worst-case delay optimization in network-on-chip," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 12, pp. 3387–3400, 2016, doi: 10.1109/TVLSI.2016.2556007.
- [22] Y.-F. Cheng, W. Shao, S.-J. Zhang, and Y.-P. Li, "An improved multi-objective genetic algorithm for large planar array thinning," *IEEE Transactions on Magnetics*, vol. 52, no. 3, pp. 1–4, Mar. 2016, doi: 10.1109/TMAG.2015.2481883.
- [23] J. Rayno, M. F. Iskander, and M. H. Kobayashi, "Hybrid genetic programming with accelerating genetic algorithm optimizer for




- 3-D metamaterial design,” *IEEE Antennas and Wireless Propagation Letters*, vol. 15, pp. 1743–1746, 2016, doi: 10.1109/LAWP.2016.2531721.
- [24] Y. Yuan and G. Wang, “Self-adaptive genetic algorithm for bucket wheel reclaimer real-parameter optimization,” *IEEE Access*, vol. 7, pp. 47762–47768, 2019, doi: 10.1109/ACCESS.2019.2910185.
- [25] Y. W. Kim, S. H. Choi, and T. H. Han, “Rapid topology generation and core mapping of optical network-on-chip for heterogeneous computing platform,” *IEEE Access*, vol. 9, pp. 110359–110370, 2021, doi: 10.1109/ACCESS.2021.3102270.

BIOGRAPHIES OF AUTHORS



Doraisamy Radha    is currently working as an assistant professor, Department of Computer Science and Engineering at Amrita School of Engineering, Amrita Vishwa Vidyapeetham. She has completed her Ph.D. in Computer Science and Engineering at Visvesvaraya Technological University, M.Tech. (CSE) at Dr. MGR University, and B.E. (CSE) at Madurai Kamaraj University. Her 22 years of teaching experience include teaching and research. Her areas of interest are data structures, cryptography, high-performance computing, multicore architecture, image processing, and web technology. She has published around 30 technical papers in national/ international journals and conferences. She can be contacted at email: d_radha@blr.amrita.edu.



Prof. Minal Moharir    is presently working as a professor, Department of Computer Science and Engineering at R V College of Engineering. Her Academic Qualification includes Ph.D. in Information and Network Security (Avinashilingam University, Coimbatore), M.Tech. (VTU) in Computer Network Engineering, and B.E. in Computer Science and Engineering (GECA). Her 20 years of experience includes teaching, training, and research. Her professional expertise includes computer networks, data communication, information security, wireless sensor networks, IoTs, high-performance computing, and GPU Computing. She can be contacted at email: minalmoharir@rvce.edu.in.