

Performance analysis of congestion-aware Q-routing algorithm for network on chip

Smriti Srivastava, Minal Moharir, Shivaneetha Gunisetty

Department of Computer Science and Engineering, R. V. College of Engineering, Bengaluru, India

Article Info

Article history:

Received Jun 2, 2023

Revised Oct 18, 2023

Accepted Oct 21, 2023

Keywords:

Benchmark

Congestion-aware Q-routing

Network-on-chip

Q-learning

Q-routing

ABSTRACT

A network on chip's performance is greatly impacted by network congestion due to the substantial increase in latency and energy utilized. Designing routing strategies that keep the network informed of the status of traffic is made easier by machine learning techniques. In this work, a reinforcement-based congestion-aware Q-routing (CAQR) technique has been presented. The proposed algorithm performed better in comparison to the conventional XY routing method tested against the SPEC CPU2006 benchmark suite in the gem5 NoC simulator tool. The suite used has 4 benchmarks, namely, namd, lbm, leslie3d and bzip2 which can be used for the cores in the network in any combination. The tests were run with 16 cores on a 4x4 network with the maximum instruction count supported by the system (here 5,000). The proposed Q-routing algorithm showed an average of 19% reduction for benchmark simulation as compared to the Dimension-ordered (X-Y) routing for readings of average packet latency which is a crucial factor in determining a network's efficiency. The analysis also shows an average reduction of 24%, 10%, 23% and 47% in terms of average packet network latency, average flit latency, average flit network latency and average energy consumption across various benchmarks.

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



Corresponding Author:

Smriti Srivastava

Department of Computer Science and Engineering, R. V. College of Engineering

Bengaluru, India

Email: smritis@rvce.edu.in

1. INTRODUCTION

As technology continues to advance, communication networks require optimal performance to enable faster data transmission. Communication on the chip is an emerging technology where different modules integrate on a single chip, known as a system-on-chip (SoC) [1]. The network on chip (NoC) [2] technology has become a cost-friendly approach for data transfer in SoCs [3]. Overcoming the latter's challenges of reliability and scalability along with providing modularity [4], the NoC is simply a systematic architecture designed especially for communication among subsystems integrated onto a chip via a network. The interconnection network features a topology with many nodes positioned in a specific pattern, where every node consists of a functional component—the processing element (PE) and a router. These routing components are connected via bidirectional links, enabling efficient communication within a network by determining the appropriate path to transmit packet data [1], [5].

In an NoC architecture, the transmission of data packets to distant nodes raises concerns regarding latency, mainly because of the massive hop counts in an overloaded network. To mitigate this issue, a wireless-NoC (WiNoC) architecture has been introduced, which helps to minimize the hopping distance by incorporating wireless capabilities in specific nodes and utilizing them for data transmission [6]. But the congestion issues remained in the busy nodes while dealing with dense traffic scenarios. To minimize the

congestion problem, the network must dynamically learn the traffic information. The technological advancement helps to provide an optimal solution using machine learning (ML), as it enables the network to analyze the current state and improve decision-making capabilities. Research on the integration of ML capabilities with network communication has been ongoing for several years. The ultimate goal is to transmit data packets into least congested route, ensuring fast delivery to their destination and ML approach is a valuable one for achieving this objective. In this work, a “congestion-aware Q-routing” (CAQR) strategy is developed which employs reinforcement learning (RL) concepts [7]. A value-based RL strategy utilized Q-table for managing Q-values. The Q-routing is a routing algorithm which is adaptive and congestion-aware that uses Q-learning to estimate Q-values to observe the outcomes of specific actions. The algorithm is intended to monitor both local and global congestion status, and utilizes Q-values from a generated Q-table to direct a node towards optimal route for forwarding data packets. The section 3.2 provides a detailed discussion of this approach. The algorithm has been developed using Gem5, a NoC simulator [8]. The proposed framework strives to enhance network abilities by minimizing the average packet latency (APL) as well as minimizing energy utilization.

This research is an extension to a previously published paper [9] where the Q-routing proposed in the study was analyzed with the synthetic traffic and this paper extends the work by analyzing the same with benchmark traffic using the SPEC CPU2006 suite. The section 2 presents the existing work. Section 3 discusses a thorough technique that comprises the ideas of RL and Q-learning and how they relate to the suggested Q-routing. Section 4 presents the outcomes of the experiment and an analysis of same. A brief discussion of possible future enhancements is presented in section 5 along with concluding the work presented so far.

2. RELATED WORK

DeepNR, an adaptive routing technique was proposed in [10] which was based on deep RL. It incorporates routing directions as actions, network information as state representations, and queueing delay as the reward function. With the Gem5 simulator, DeepNR was tested for artificial and real-time traffic scenarios. Additionally, the proposed work is tested against the benchmarks from SPEC CPU 2006. Based on the information about traffic and congestion at the NoC, Reza and Le [11] proposes a similar routing technique which employed three RL algorithms at runtime. Power-saving methods such as power gating and the concept of dynamic voltage and frequency scaling (DVFS) were used in NoCs by Zheng and Louri [12]. During runtime, an artificial neural network (ANN) based RL approach is used to predict the traffic status of NoC. The use of deep RL (DRL) in router less NoC architecture [13] has recently been reported as well as the optimization of energy usage and power consumption [14].

Farahnakian *et al.* [15], a method called “Clustered Quality” (CQ) routing which clusters the network and provides potential solution to minimize the overhead issue. The inter and intra cluster has been used by Q-routing and XY routing respectively for packet transmission. Each cluster maintain a CQ-table with a design that is exactly like a Q-table to enable this strategy. Assuming the identical traffic scenarios for every cluster, potentially resulting in unfavorable outcomes. Additionally, the energy usage and latency of the proposed framework is comparatively high for WiNoC.

Hu and Marculescu [16] proposed a method where the routing policy called DyAD is designed by combining deterministic routing methods with the adaptive routing methods. In case of network congestion, the policy works adaptively otherwise deterministically. This shows an improvement in performance as compared to a completely adaptive routing policy. Wu *et al.* [17] used a method called the contention-aware input selection (CAIS) which selects an input channel among many options that are under contention for same output channel. This method showed improved routing efficiency. It makes its selection by observing the number of requests for an input channel and the one with higher contention levels than the others is chosen thereby removing possible congestion in future.

Ebrahimi *et al.* [18] presented an agent-based NoC (ANoC) structure that estimates the congested areas by getting the global congestion information sent across the network. The network is parted into divisions called clusters and each such obtained division is a cluster agent which is responsible for communicating the status of its local congestion with the neighboring cluster and circulate the information. A method called congestion aware selection (CAS) is designed and using the global and local information, the packets are routed efficiently.

Chen *et al.* [19] present ML algorithms that make use of ANN concepts. The designed algorithms can be applied to solve various wireless networking challenges. It has an overview of the basic architectures for each type of ANN and the tutorial summarizes the specific wireless problems which can be used for future work.

Nilsson *et al.* [20] proposed a memory less switch design which decides how the packets are emitted. If there is a congestion observed the packets are deflected in a non-ideal path. A novel approach

called proximity congestion awareness (PCA) was designed so as to keep the information of the neighboring switches in the current switch. These are called the stress values which indicate the load level in that switch. Hence, the one with the least stress value among the neighboring switches is the path with least congestion. Farahnakian *et al.* [21] employed an NoC simulator based on Omnet++ and evaluated a routing method that is CAQR approach. It works by estimating the current traffic state in the network to mitigate congestion. However, in the absence of congestion, the latency is relatively high in comparison to conventional routing strategies. High efficiency was demonstrated by Majer *et al.* [22], through the dynamic selection of packet routing policies using an RL technique. By selecting an optimal routing technique based on various network states has been implemented for the corresponding network state. Reza [23], showcased the effectiveness of deep Q-learning (DQL) in enabling a single agent to maintain a comprehensive record of Q-value vectors for every router action within the network, thereby eliminating the need for individual Q-tables at each node and minimizing the associated overhead.

Deb *et al.* [24], presented adaptive routing techniques that are both cost-effective and capable of forwarding packets to distant nodes through specialized channels constructed using a technology called the transmission line (TL) technology. The inclusion of additional TLs on the chip decreases the network diameter, by reducing the APL. The objective listed for the architectures, secured bank treasury receipt (SBTR) and electronic (e-SBTR), is to minimize the number of intermediate hops and thereby reduce packet latency. The effectiveness of these techniques is evaluated by utilizing benchmark mixes from paralax of one arc second (PARSEC) and SPEC CPU 2006 and the findings reveal that the architecture e-SBTR outperformed the current express virtual channel method, as it attains less hop count and reduced packet latency.

Ahmad *et al.* [25] introduced a novel framework that transmits congestion data within the data packet. The approach is executed on a field-programmable gate array (FPGA)-based mesh NoC. Compared to current congestion-aware routing (CAR) strategy, the proposed approach minimizes latency, maximizes throughput, and requires less bandwidth for exchanging the congestion data among routers. Rad *et al.* [26] a detailed summary of the congestion control (CC) strategies currently used in WiNoCs. The identified strategies are categorized into six different categories, which encompass CAR algorithms, Media access control (MAC) protocols, hardware resources-based CC, rate-based CC, task-migration using mapping and CA architectures. Objective of this study is to emphasize different traits and the limitations of CC strategies using a fresh perspective, that can help fellow researchers in developing effective schemes. Arun *et al.* [27] proposed an efficient model using 2D Mesh NoCs. Results from experiments show that this strategy reduces the total link traversals necessary to achieve multicast communication and thereby improves the average multicast transaction latency.

3. DESIGN AND IMPLEMENTATION

The concepts used in developing the proposed algorithm are described in this section. A brief introduction to Q-learning, a discussion of Q-tables and a congestion-aware Q-routing is presented. The section concludes with an introduction of SPEC CPU 2006 Benchmark.

3.1. Q-learning

A reinforcement learning agent doesn't depend on training but learns by performing set of actions and observing their results. If the result is good, it gets a better positive reward. A penalty is given otherwise. This way an optimal action, i.e., one with the maximum reward, is chosen. The Q-learning employs value-based RL strategy. The Q-table keeps track of the current state space, available actions, next state space estimated using the previous two and the Q-value. Selecting an action during the beginning of learning happens randomly—called the exploration. At the end of learning, the selection happens based on the prepared Q-table – called the exploitation. Initially, the table entries for each state s , action a are initialized to zero. An action is selected based on epsilon-greedy strategy where the value of the epsilon refers to the probability to explore or exploit. An immediate reward r is received and the new state called s' is observed. The existing Q-value denoted by $Q(s,a)$ is updated with $Q'(s,a)$ in the table based on the values of reward r and $Q(s',a)$ as shown in (1) where γ is the discount rate and α denotes the learning rate. The process continues iteratively, updating the Q-value at each step, until the learning has stopped.

$$Q'(s,a)=Q(s,a)+\alpha[r+\gamma \max_a Q(s',a)-Q(s,a)] \dots \quad (1)$$

Q-routing is designed based on the concept of Q-learning. The Q-table is built keeping a network of nodes as the environment. The Q-values helps the data packet to find the optimal neighbour. Suppose a node x needs to send a packet to destination node d via the neighbouring node y , then the Q-value $Q_x(y,d)$ needs to

be kept updated iteratively. This value here depends on three factors, viz., the queuing delay q_y when a packet spends time in node y 's queue, the transmission delay (δ) for time taken to travel from x to y and the time taken for the packet to reach ' d ' from ' y '. Thus, the q -value $Q_x(y,d)$ gets updated with $Q'_x(y,d)$ using the aforementioned factors as shown in (2) [9]. The neighbouring node with the minimum q -value will be the optimal node to send a packet through.

$$Q'_x(y,d) = Q_x(y,d) + \alpha(Q_x(y,d) + q_y + \delta - Q'_x(y,d)) \dots \tag{2}$$

3.2. Q-table

Consider an example of a 3x3 mesh network as shown in Figure. 1. The suggested approach uses a novel Q-table to assign a Q-value to each network instance. In a 2D mesh topology, there are $N \times M$ nodes with Q-tables, where ' N ' and ' M ' represents the total rows and columns respectively.

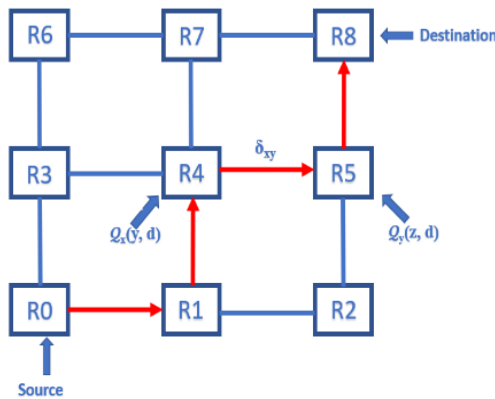


Figure 1. Q-routing algorithm

The Table 1 comprises four fields such as Q-value which correspond to each Q-table, output port, current and destination node. Assume a packet is travelling by the longest route available in the network, from $y = 0$ to the $d = 8$. Say the packet is at node 4 for instance $t_i = t$. By considering lowest Q-value at node 4, the packet chooses one of two feasible routes to travel to $d = 8$. There are only 2 ways to use to route the packet to its destination, despite there being 4 alternative paths accessible at node 4. Alternatively, node 5 or 7 may receive the packet. Both East and North ports are permitted. Next, the packet is directly transferred to the adjacent node by retrieving the minimal Q-value among two nodes in the Q-table.

Table 1. Example of a Q-table for node 4

Current Node (y)	Destination node (d)	Output port	Q-value
4	0	South, West	...
4	1	South	...
4	2	South, East	...
4	3	West	...
4	4	Local	...
4	5	East	...
4	6	West, North	...
4	7	North	...
4	8	East, North	...

3.3. Congestion-aware Q-routing (CAQR) algorithm

Figure 2 depicts the proposed technique that was developed and used in the gem5 simulation tool. On top of the preexisting XY algorithm, the tool offers an extension for adding unique algorithms. Take packet p from source node x that is directed at destination node d . The incoming port information contained in the route information details of the flit is extracted when p is at node Y . The preceding node from which the packet originated is identified using the inbound port. Let x be the node before it. The Q-values of each node, as listed in Table 1, are placed into a static vector. A packet only has two alternative paths from node y to destination node d . The port with the lowest Q-value is ultimately chosen after the Q-values for the two

potential output ports for node Y have been extracted. The packet will be sent further down the network using this output port. Following the computation of the output port, (3) is used to determine the new Q-value of x, the preceding node.

$$Q_x(y,d)_{\text{new}} = Q_x(y,d)_{\text{old}} + 0.5*(0.7*Q_y(z,d) + q_y + \delta_{xy} - Q_x(y,d)_{\text{old}}) \dots \quad (3)$$

The Q-update approach in Figure 3 also provides a description of the procedure in Figure 2 for improving the data of the previous node. In addition, a learning rate of 0.5 is utilised for the system to attain 50% review, and a discount rate of 0.7 is applied. The RL-based model is trained for 50 steps, updating the Q-values each time. The final obtained Q-table helps to transfer every packet at the completion of training phase.

Algorithm 1 Proposed Q-routing algorithm

- 1: A packet from source node x is transferred to destination node d through node y.
 - 2: At node y, find the previous node using incoming port value available in route info.
 - 3: Minimum Q-value is obtained from its neighbouring nodes using std::minimum function.
 - (a) Get destination node and compute the possible paths for the packet
 - (b) **if** current_node == destination_node **then**
 - (c) absorb the packet
 - (d) **else if** possible output port == 1 **then**
 - (e) Q-value = Q[current_node][destination_node][output_port]
 - (f) **else**
 - (g) Find latency of both the output ports
 - (h) Q-value(1) = Q[current_node][destination_node][port_1]
 - (i) Q-value(2) = Q[current_node][destination_node][port_2]
 - (j) **if** Q-value(1) < Q-value(2) **then**
 - (k) Q-value = Q-value(1)
 - (l) **else**
 - (m) Q-value = Q-value(2)
 - (n) **end if**
 - (o) **end if**
 - 4: The packet is transmitted from node y to the neighbouring node with least latency.
 - 5: Node x receives the Q-value used by node y and updates it Q-values using Q-update algorithm.
-

Figure 2. Q-learning-based Q-routing algorithm

Algorithm 2 Q-table updation algorithm

- 1: y = get_current_node()
 - 2: dest = get_destination_node()
 - 3: p = get_previous_node()
 - 4: queuing_delay q_y = destination_queuing_delay + source_queuing_delay
 - 5: **if** Q-value(path₁) < Q-value(path₂) **then**
 - 6: δ = get_link_latency(y, next_node(path₁))
 - 7: $Q_{\text{new}}[p][\text{dest}][\text{path}_1] = Q_{\text{old}}[p][\text{dest}][\text{path}_1] + 0.5(0.7*Q[y][\text{dest}] + q_y + \delta - Q_{\text{old}}[p][\text{dest}][\text{path}_1])$
 - 8: **else if** Q-value(path₁) > Q-value(path₂) **then**
 - 9: δ = get_link_latency(y, next_node(path₂))
 - 10: $Q_{\text{new}}[p][\text{dest}][\text{path}_2] = Q_{\text{old}}[p][\text{dest}][\text{path}_2] + 0.5(0.7*Q[y][\text{dest}] + q_y + \delta - Q_{\text{old}}[p][\text{dest}][\text{path}_2])$
 - 11: **end if**
-

Figure 3. Algorithm to update Q-table

3.4. SPEC CPU2006 benchmark

The SPEC CPU2006 [28] benchmark suite used to evaluate the proposed algorithm is developed and maintained by the “standard performance evaluation corporation” (SPEC) which is a widely accepted and established industry standard benchmark used to assess the performance of machine at the processor level. The benchmark suite is structured to present practical workloads that are representative of actual-world applications, including scientific simulations, multimedia processing, and encryption algorithms as opposed to synthetic benchmarks. The suite provides for two categories of benchmarks which aim to measure the performance of compute intensive integer (CINT2006) and floating-point processing (CFP2006). Among the four used in the research, bzip2 belongs to the former category and the rest, that is, leslie3d, namd and lbm, belong to CFP2006. Bzip2 is a compression benchmark where the input sets are compressed and then decompressed at three different compression levels. The end result is then compared to the actual data after every decompression step. Leslie3d is a Computational fluid dynamics model benchmark that queries an array of turbulence phenomena such as acoustics, combustion, mixing and fluid mechanics. Namd benchmark is based on molecular dynamics that computes inter-atomic interactions to simulate a biomolecular system. Lastly, lbm is fluid dynamics benchmark that compute 3D velocity vectors for cells of the incompressible fluid aiming to simulate it in 3D. The utilization of the SPEC CPU2006 benchmark in network on chip (NoC) research serves the purpose of assessing the efficiency of processors and memory subsystems in systems based on NoC, which is a communication architecture that utilizes interconnected processing elements (PEs) to facilitate data and information exchange.

4. EXPERIMENT RESULTS AND ANALYSIS

We used gem5 simulation software's system call emulation mode to conduct the proposed work. Agarwal [5] is a cycle-accurate network model that simulates router architecture and is used to evaluate NoC. In this work, ALPHA ISA is used in conjunction with the Ruby memory design on an 4x4 2-d Mesh Topology within the Garnet network to analyze the characteristics of the proposed Q-routing method in comparison to the conventional XY routing strategy available in Gem5 using the SPEC CPU2006 benchmark suite. The link latency among the nodes has been varied randomly from the default value 1 so as to create a congestion scenario.

The tests were conducted on a 4x4 network with 16 cores, utilizing the maximum supported instruction count of the system (5,000 in this case). The algorithm was evaluated against the CPU2006 benchmark suite and Figure 4(a) shows the comparison between XY and Q-routing in terms of APL for different benchmarks. When employing the namd workload on all cores, the reduction in APL was determined to be approximately 19%. The leslie3d workload exhibited the highest APL reduction, with a value of 27% while bzip2 shows the least reduction with a value of 15%. When utilizing the lbm workload on all cores, the CAQR technique exhibited an approximate 26.5% decrease in the APL as compared to the XY routing algorithm. Additionally, when running simulations with an equal combination of all four benchmarks, the APL reduction was determined to be 9%. The average improvement across the readings here is about 19%. Figure 4(b) also shows the comparisons in terms of APNL where the highest reduction is found to be when using leslie3d with a value of 31% and lowest with the bzip2 with a value of 17.5%. The average is found to be about 24%.

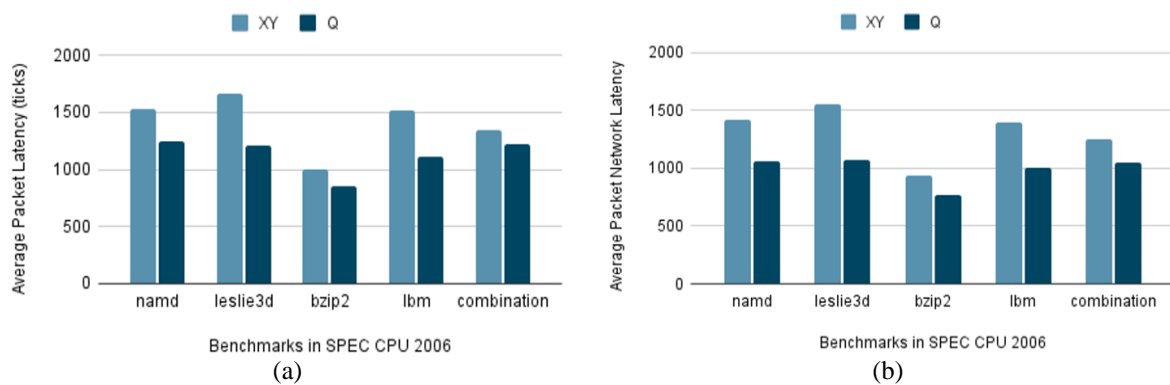


Figure 4. Comparison of XY and Q-routing in (a) terms of average packet latency (APL) and (b) average packet network latency (APNL) using SPEC CPU 2006 benchmark suite

Figure 5(a) shows the comparison between XY and Q-routing in terms of average flit latency. The highest reduction was obtained using leslie3d with a 31% improvement while with bzip2 there was no reduction observed. The average improvement across all the benchmarks is about 10%. The comparisons in terms of average flit network latency are shown in the Figure 5(b) shows highest reduction of 31% when using leslie3d and lowest of 16% when using bzip2. On an average the improvement is about 23% across all the combinations.

Figure 6(a) shows the comparisons in terms of average energy utilization in mJ. The highest reduction is found to be in leslie3d of about 53% and least in the combination network with 37% reduction. The average reduction across all readings is found to be 47%. The Figure 6(b) also shows the average power consumption in mW. The readings are almost same for both XY and Q-routing for all the cases which is justified by the fact that Q-learning would find the optimal path by the reinforcement mechanism which requires traversing random paths in the beginning of the simulation. The results indicate that the congestion-aware Q-routing algorithm from this study performed better than the XY routing algorithm, particularly with regards to average packet latency, which is a significant factor in determining a network's efficiency.

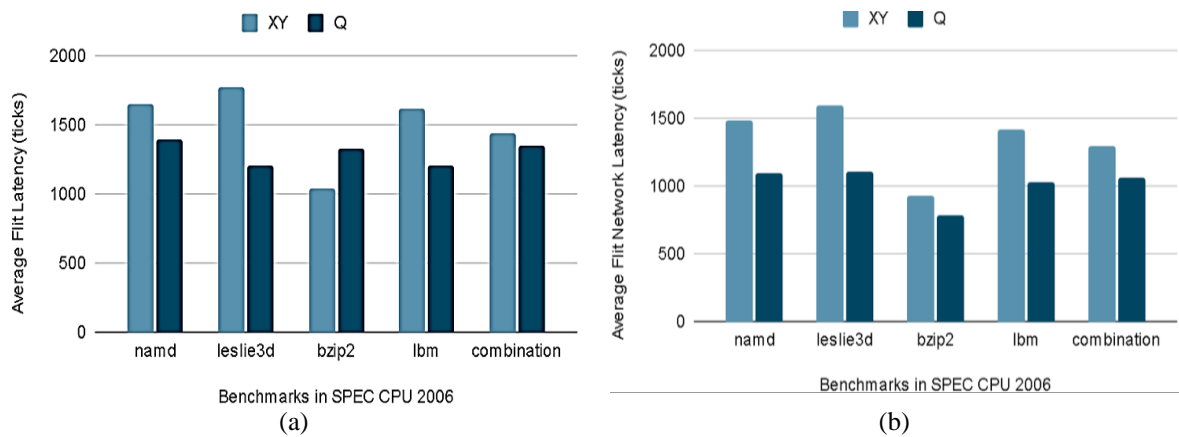


Figure 5. Comparison of XY and Q-routing in (a) terms of average flit latency (AFL) and (b) average flit network latency (AFNL) using SPEC CPU 2006 benchmark suite

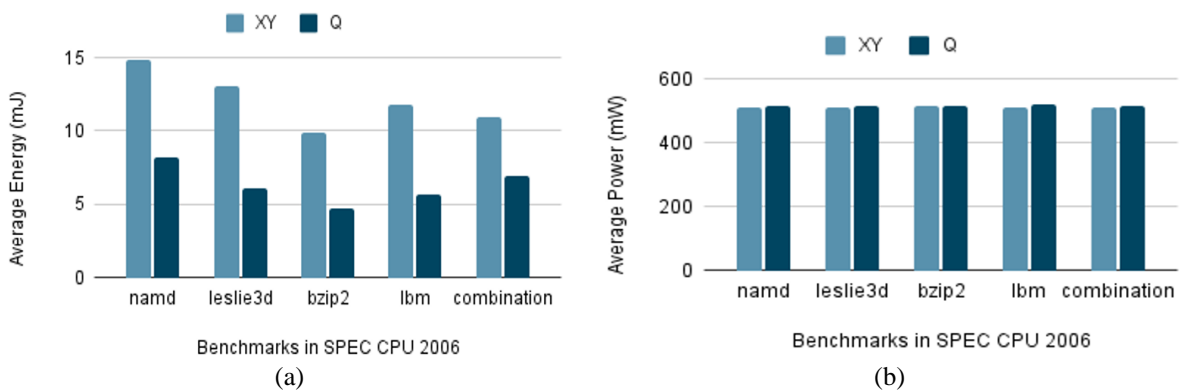


Figure 6. Comparison of XY and Q-routing in (a) terms of average energy utilization and (b) average power using SPEC CPU 2006 benchmark suite

5. CONCLUSION





The research presents a Q-routing method that is congestion-aware that lowers average packet latency and also reducing the energy utilization in an NoC. The algorithm is tested using the CPU2006 benchmark suite. In terms of average packet latency, the proposed congestion-aware Q-routing (CAQR) algorithm clearly outperformed the XY routing algorithm, which is an important component in determining a network's efficiency. This Work can further be extended to implement congestion-aware Q-routing for a

WiNoC and perform the analysis of various simulation parameters like average packet latency, power, energy and area with various traffic patterns.





REFERENCES

- [1] H. Cai and Y. Yang, "Congestion Prediction Algorithm for Network on Chip," *TELKOMNIKA Indonesian Journal of Electrical Engineering*, vol. 11, no. 12, pp. 7392–7398, Dec. 2013, doi: 10.11591/telkomnika.v11i12.3987.
- [2] S. Kumar *et al.*, "A network on chip architecture and design methodology," in *Proceedings IEEE Computer Society Annual Symposium on VLSI. New Paradigms for VLSI Systems Design. ISVLSI 2002*, 2002, pp. 117–124, doi: 10.1109/ISVLSI.2002.1016885.
- [3] D. C. Marinescu, "Cloud Access and Cloud Interconnection Networks," in *Cloud Computing*, Amsterdam: Elsevier, 2018, pp. 153–194, doi: 10.1016/b978-0-12-812810-7.00007-8.
- [4] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of the 38th Design Automation Conference (IEEE Cat. No.01CH37232)*, 2001, pp. 684–689, doi: 10.1109/dac.2001.935594.
- [5] N. Agarwal, T. Krishna, L.-S. Peh, and N. K. Jha, "GARNET: A detailed on-chip network model inside a full-system simulator," in *2009 IEEE International Symposium on Performance Analysis of Systems and Software*, Apr. 2009, pp. 33–42, doi: 10.1109/ispas.2009.4919636.
- [6] S. Deb, A. Ganguly, P. P. Pande, B. Belzer, and D. Heo, "Wireless NoC as Interconnection Backbone for Multicore Chips: Promises and Challenges," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 2, no. 2, pp. 228–239, Jun. 2012, doi: 10.1109/jetcas.2012.2193835.
- [7] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3–4, pp. 279–292, May 1992, doi: 10.1007/BF00992698.
- [8] N. Binkert *et al.*, "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, May 2011, doi: 10.1145/2024716.2024718.
- [9] S. Srivastava, M. A. Shaikh, S. G. and M. Moharir, "Intelligent congestion control for NoC architecture in Gem5 simulator," in *2022 IEEE 15th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoc)*, Dec. 2022, pp. 353–360, doi: 10.1109/mcsoc57363.2022.00062.
- [10] R. R. R.S. *et al.*, "DeepNR: An adaptive deep reinforcement learning based NoC routing algorithm," *Microprocessors and Microsystems*, vol. 90, p. 104485, Apr. 2022, doi: 10.1016/j.micro.2022.104485.
- [11] M. F. Reza and T. T. Le, "Reinforcement Learning Enabled Routing for High-Performance Networks-on-Chip," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2021, pp. 1–5, doi: 10.1109/iscas51556.2021.9401790.
- [12] H. Zheng and A. Louri, "Agile: A Learning-Enabled Power and Performance-Efficient Network-on-Chip Design," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 1, pp. 223–236, Jan. 2022, doi: 10.1109/tetc.2020.3003496.
- [13] T.-R. Lin, D. Penney, M. Pedram, and L. Chen, "A Deep Reinforcement Learning Framework for Architectural Exploration: A Routerless NoC Case Study," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Feb. 2020, pp. 99–110, doi: 10.1109/hpca47549.2020.00018.
- [14] H. Zheng and A. Louri, "An Energy-Efficient Network-on-Chip Design using Reinforcement Learning," in *Proceedings of the 56th Annual Design Automation Conference 2019*, Jun. 2019, pp. 1–6, doi: 10.1145/3316781.3317768.
- [15] F. Farahnakian, M. Ebrahimi, M. Daneshalab, J. Plosila, and P. Liljeberg, "Optimized Q-learning model for distributing traffic in on-chip networks," in *2012 IEEE 3rd International Conference on Networked Embedded Systems for Every Application (NESEA)*, Dec. 2012, pp. 1–8, doi: 10.1109/nesea.2012.6474016.
- [16] J. Hu and R. Marculescu, "DyAD: smart routing for networks-on-chip," in *Proceedings of the 41st annual Design Automation Conference*, Jun. 2004, pp. 260–263, doi: 10.1145/996566.996638.
- [17] D. Wu, B. M. Al-Hashimi, and M. T. Schmitz, "Improving routing efficiency for network-on-chip through contention-aware input selection," in *Asia and South Pacific Conference on Design Automation, 2006.*, 2006, pp. 36–41, doi: 10.1109/aspdac.2006.1594642.
- [18] M. Ebrahimi, M. Daneshalab, P. Liljeberg, J. Plosila, and H. Tenhunen, "Agent-based on-chip network using efficient selection method," in *2011 IEEE/IFIP 19th International Conference on VLSI and System-on-Chip*, Oct. 2011, pp. 284–289, doi: 10.1109/vlsisoc.2011.6081593.
- [19] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Artificial Neural Networks-Based Machine Learning for Wireless Networks: A Tutorial," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3039–3071, 2019, doi: 10.1109/comst.2019.2926625.
- [20] E. Nilsson, M. Millberg, J. Oberg, and A. Jantsch, "Load distribution with the proximity congestion awareness in a network on chip," in *2003 Design, Automation and Test in Europe Conference and Exhibition*, 2003, pp. 1126–1127, doi: 10.1109/date.2003.1253765.
- [21] F. Farahnakian, M. Ebrahimi, M. Daneshalab, P. Liljeberg, and J. Plosila, "Q-learning based congestion-aware routing algorithm for on-chip network," in *2011 IEEE 2nd International Conference on Networked Embedded Systems for Enterprise Applications*, Dec. 2011, pp. 1126–1127, doi: 10.1109/nesea.2011.6144949.
- [22] M. Majer, C. Bobda, A. Ahmadinia, and J. Teich, "Packet Routing in Dynamically Changing Networks on Chip," 2005, doi: 10.1109/ipdps.2005.323.
- [23] M. F. Reza, "Deep Reinforcement Learning for Self-Configurable NoC," in *2020 IEEE 33rd International System-on-Chip Conference (SOCC)*, Sep. 2020, pp. 185–190, doi: 10.1109/socc49529.2020.9524761.
- [24] D. Deb, J. Jose, S. Das, and H. K. Kapoor, "Cost effective routing techniques in 2D mesh NoC using on-chip transmission lines," *Journal of Parallel and Distributed Computing*, vol. 123, pp. 118–129, Jan. 2019, doi: 10.1016/j.jpdc.2018.09.009.
- [25] K. Ahmad *et al.*, "Congestion-Aware Routing Algorithm for NoC Using Data Packets," *Wireless Communications and Mobile Computing*, vol. 2021, pp. 1–11, Aug. 2021, doi: 10.1155/2021/8588646.
- [26] F. Rad, M. Reshadi, and A. Khademzadeh, "A survey and taxonomy of congestion control mechanisms in wireless network on chip," *Journal of Systems Architecture*, vol. 108, Sep. 2020, doi: 10.1016/j.sysarc.2020.101807.
- [27] M. R. Arun, P. A. Jisha, and J. Jose, "A Novel Energy Efficient Multicasting Approach For Mesh NoCs," *Procedia Computer Science*, vol. 93, pp. 283–291, 2016, doi: 10.1016/j.procs.2016.07.212.
- [28] J. L. Henning, "SPEC CPU2006 benchmark descriptions," *ACM SIGARCH Computer Architecture News*, vol. 34, no. 4, pp. 1–17, Sep. 2006, doi: 10.1145/1186736.1186737.





BIOGRAPHIES OF AUTHORS

Smriti Srivastava     is masters in computer science and engineering with a wide experience of 14+ years in teaching. Working currently at RV college of engineering as Assistant Professor. Area of interest include wireless networks, network on chip and machine learning. She can be contacted at email: smritis@rvce.edu.in.



Minal Moharir     is working as Professor and coordinator for cybersecurity program in Department of Computer Science and Engineering, RV college of Engineering, Bengaluru. She has completed her Ph.D. on Network security at Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, in 2014. Her research area includes Information and network security, Network performance analysis, deep learning and high-performance computing. She has completed a CARS project titled "Assessment of Privacy Protection in the Encrypted DNS Protocol (DoH/DoT) and Extension of the Analysis to High-Speed Networks" from CAIR, DRDO. She has completed six R & D projects on network performance and analysis from Citrix R&D Pvt Ltd, Bengaluru. She has received a hardware research grant from Nvidia Pvt. Ltd worth Rs.15Lakhs. She has received the best white paper award from Nokia Research for the paper titled Performance Enhancement of many-core processors using Wi-NoC. Recently she received the best researcher award from ISTE society. She has her research scholars working on topics network performance and security, high performance processing/computing of multimedia data on noc based many-core architecture and intelligent wireless NoC for many-core processor architecture. She can be contacted at email: minalmoharir@rvce.edu.in.



Shivaneetha Gunisetty     is a 2023 graduate of B.E. in computer science and Engineering at R V College of Engineering, Bengaluru, India. Areas of interest include computer networks and machine learning. She can be contacted at email: shivaneethag.cs19@rvce.edu.in.