Journal of Electrical and Computer Engineering Innovations
(JECEI)
Journal homepage: http://www.jecei.sru.ac.ir

Research paper

# An Adaptive Routing Algorithm for Wireless Network on Chips

***A. Tajary**, E. Tahanian***

*Faculty of Computer Engineering, Shahrood University of Technology, Shahrood, Iran.*

**Article Info**

**Article History:**
Received 16 December 2021
Reviewed 24 January 2022
Revised 16 February 2022
Accepted 20 April 2022

**Keywords:**
Wireless network on chip
Routing
NoC

*Corresponding Author's Email Address:
*tajary@shahroodut.ac.ir*

**Abstract**

**Background and Objectives:** Wireless Network on Chip (WNoC) is one of the promising interconnection architectures for future many-core processors. Besides the architectures and topologies of these WNoCs, designing an efficient routing algorithm that uses the provided frequency band to achieve better network latency is one of the challenges.
**Methods:** Using wireless connections reduces the number of hops for sending data in a network, which can lead to lower latency for data delivery and higher throughput in WNoCs. On the other hand, since using wireless links reduces the number of hops for data transfer; this can result in congestion around the wireless nodes. The congestion may result in more delay in data transfer which reduces the network throughput of WNoCs. Although there are some good routing algorithms that balance traffic using wired and wireless connections for synthetic traffic patterns, they cannot deal with dynamic traffic patterns that existed in real-world applications. In this paper, we propose a new routing algorithm that uses the wireless connections as much as possible, and in the case of congestion, it uses the wired connection instead.
**Results:** We investigated the proposed method using eight applications from the Parsec benchmark suite. Simulation results show that the proposed method can achieve up to 13.9% higher network throughput with a power consumption reduction up to 1.4%.
**Conclusion:** In this paper, we proposed an adaptive routing algorithm that uses wireless links to deliver data over the network on chip. We investigated the proposed method on real-work applications. Simulation results show that the proposed method can achieve higher network throughput and lower power consumption.

©2022 JECEI. All rights reserved.

## Introduction

Demand for more computation power within a chip resulted in multicore systems in which the bus technology was used to connect processing elements and cache modules on a single chip [1], [2]. Due to the poor scalability of bus base systems, a new paradigm is formed for future many-core systems: Network on Chip (NoC) [3]-[6]. In the NoC, many processing elements and cache systems are connected through an on-chip interconnection network. Each interconnection node contains a processing element and a router [7]. The nodes can connect in 2D (like mesh) or 3D structures.

Although this paradigm is promising, some issues should be considered. One of the main issues in the traditional NoCs is the large latency of packets for traveling between the distant source and destination nodes. Sending packets to a neighbor node can be done in one hop, while sending a packet to a distant node,

may take tens of hops, which results in long latency and probably makes congestion in the network [8], [9]. To alleviate this problem, several new technologies like 3D NoC [10]-[12], Photonic NoC [13]-[15] and Wireless NoC [16]-[18] has been proposed.

Wireless NoC (WNoC) is one of the main options for future NoCs [16], [17]. There are many kinds of research conducted to investigate different aspects of this technology such as the possible architectures and routing algorithms in these new architectures [18]-[22]. In the WNoCs, some nodes have a wireless antenna which makes them the wireless nodes. To reduce the number of hops, sending packets through the wireless nodes is preferred over the traditional wired nodes [9]. However, this makes congestion around the wireless nodes and leads to more latency for packets and more congestion in the network.

A very famous solution is to consider an additional cost for the wireless connections [23], [9], [24]. The number of hops is a common metric to make the selection between the wired and wireless paths. Considering the additional cost for the wireless paths makes them be less selected by the packets. Therefore, the congestion around the wireless nodes will be decreased.

Although this method is effective in some situations, it has the main drawback, especially, when the network experiences the traffic of real applications. In real-world applications, the traffic pattern is not static and changes over time. Therefore, when a constant cost is considered, it is possible to have no congestion at one time and surprisingly observe huge congestion at another time.

To reduce the noise effect of the wireless links on wired links, several methods have been proposed [34], [9]. For example, the method in [9] uses parallel-plate waveguide as a dedicated structure for transferring wireless signal. Since we consider the architectural view of the NoC, we did not consider these effects in our simulations. It is also important to note that, hierarchical NoC is one of the promising architectures for wireless NoCs. Of course, if the wired path is shorter than the wireless one, the routing algorithm forces the source of flit to send it through wire path.

In this paper, we present a method to balance the traffic for avoiding congestion in the WNoC. In this method, the congestion at wireless nodes is continually checked. Whenever the congestion is detected, it uses a wired connection. To avoid congestion, average flit latency for wireless and wired connections is compared and consequently, the routing will be done. However, we consider some level of randomness in the path selection which makes it possible to react to the changes in the traffic pattern.

For wireless communications, we used the radio-hub component in [25]. It can support one or more wireless channels. We used multiple channels for sending and receiving of the data and acknowledge signals. It is important to note that each wireless node transmits the flits by only a specific channel but can receive all the channels. According to the destination wireless node ID, it can accept or reject the received flits. So, since the frequencies of the transmitters are different, the effect of collision has been neglected.

Many kinds of research on the WNoC used synthetic traffic patterns available in NoC simulators such as Noxim [25] and booksim [26]. In this paper, we ran the experimental results on the real-world applications from the Parsec benchmark suite [27].

Simulation results showed that the proposed routing algorithm can improve the network throughput by 13.9% while reducing the power consumption by 1.4% over related works.

The rest of this paper is organized as follows. In Section 2, the related works are discussed. In Section 3, the proposed method is presented. The experimental results are reported in Section 4, and the conclusions are given in section 5.

## Related Works

The XY routing algorithm is one of the traditional deadlock-free deterministic routing algorithms in NoCs [7]. The wireless-XY is an extension of the XY routing algorithm for the WNoCs. Several variants for this algorithm are proposed. The main idea is to use the wireless equipment in the WNoC as much as possible. Since the excessive use of wireless links causes congestion in the wireless nodes, researchers suggest considering a cost for the wireless connections. The value of this cost is based on the packet injection rate of the WNoC. For low packet injection rates, lower values of the cost are needed. But, as the packet injection rate increases, higher values for the cost are required. However, the researchers have considered only a predetermined value for the cost in the previous studies. It is important to note that the wireless-XY algorithm is also deterministic and deadlock-free. The deadlock-freedom is achieved by utilizing virtual channels.

In [28] a Q-tabled-based adaptive routing algorithm is proposed. In this method, each node contains its Q-table, which actions are the selection of the output port and the agent is the packet to be transmitted. The number of rows of the table is equal to the number of nodes in the network. When the current node wants to send a packet to another node, it checks the action values from the corresponding row in the table and selects the lowest value. Then the packet will go through that output port. The values of the table are updated based on the local observation of the packet latency and

the Q values of the neighbor nodes. Although the paper gives a very interesting idea, there are some issues with the algorithm. Since this routing algorithm is adaptive and the path of the packets changes over time, it is possible to encounter live lock or deadlock. To alleviate the deadlock problem, the authors used packet-based routing and did not use the well-known wormhole switching. Another problem of this method is the use of information from the adjacent nodes which needs dedicated buses for communication and therefore, leads to more power and area overhead.

In [29], Q-learning is used to propose a congestion-aware routing algorithm for NoCs. In this reference, the network is divided into some clusters and each cluster has a Q-table. Selecting an output channel is based on the density values extracted from the Q-table. Although this method uses wormhole switching, it does not consider wireless network-on-chip.

In [31] authors proposed a wireless NoC architecture that uses on-chip antennas for wireless communication between the long-distance cores. They synthesized and implemented the architecture in Altera Quartus || tool. They proposed a custom routing algorithm for the proposed architecture that uses the vertical and horizontal distance of the source and the destination node as a parameter for selecting the routing path. They used synthetic traffic in evaluation of the proposed routing algorithm.

In [32] four different routing algorithms are compared based on their performance in the wireless NoCs. The evaluations are based on the synthetic traffic patterns. They compared latency, throughput, and wireless utilization of the routing algorithms. The simulation results showed that the XY algorithm achieved the best latency and throughput. It is important to note that they did not consider extra cost on using wireless connections.

In [33] authors proposed an arbitration mechanism for crossbar switches in wireless NoCs. The arbitration mechanism tries to eliminate the port contention in wireless routers. For this new architecture, a routing algorithm is proposed that considers λ as an extra cost for wireless links. They also used synthetic traffic pattern for the evaluation of their proposed method.

## Proposed Method

The architecture of a small world NoC with 64 nodes is shown in Fig. 1. The gray squares indicate the position of the wireless nodes. This WNoC uses the mesh topology and has 8 rows and 8 columns. The data in traditional NoCs are transferred based on a router to router mechanism. For example, suppose that the green node wants to send data to the red node in Fig. 1. In each step, the data will be delivered to an adjacent router until it reaches its destination. In the XY routing algorithm, this takes four columns to the right and seven rows to the bottom which takes 11 hops. The wireless technology can be used to reduce the number of hops for delivery of the data for distant node. In this example, using wireless connections reduces the number of hops to three. If a node decides that its packet should pass through the wireless links, it sends the packet to the closest wireless router. The wireless router sends the packet to the closest wireless router to the destination, and that wireless router sends the packet to the destination node.

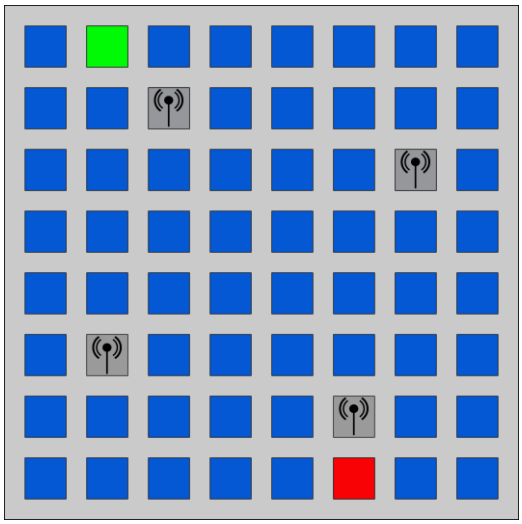

Fig. 1: The schematic view of a 64 node WNoC.

In the small-world WNoC, the wireless nodes are used to minimize the distance between distant nodes. As they can provide faster communication between nodes, the first insight is to send as much traffic as possible through the wireless nodes. This makes congestion around wireless nodes. For two nodes to communicate there are many options, wired connection is one of them, while there are many wireless variants. In wired connection, there are many deadlock-free routing algorithms, XY is one of them. For XY routing, the cost of using it can be computed as the following equation:

$$cost = |sx - dx| + |sy - dy| \quad (1)$$

where sx is the row number of the source node, dx is the row number of the destination node, sy is the column number of the source node and dy is the column number of the destination node. When there are some wireless nodes, another possible path would be to send the packet to a wireless node (usually the nearest one) and then the wireless node, sends the packet to another wireless node (usually the closest to the destination node), and finally, the second wireless node, delivers the packet to the destination node. In this scenario, the cost can be computed as (2):

$$cost = |sx - wnsx| + |sy - wnsy| + |wndx - dx| + |wndy - dy| + \delta \quad (2)$$

In this formula, wns is the wireless node near the source node and wnd is the wireless node near the destination node. The $\delta$ is a factor for considering a cost value for using wireless. The first idea is to have no $\delta$ value, but in this case, the congestion will happen around the wireless nodes. Many nodes prefer to send their packets through the wireless links and there will be congestion for the source wireless nodes. On the other hand, in comparison with the wireless connections, the wired network bandwidth is so limited. Therefore, the destination wireless node cannot send the packet to the destination immediately. Consequently, this causes the source wireless node to wait until the destination wireless node accepts the packet. The source wireless node keeps sending the same packet until the destination node accepts it. So having a $\delta$ value will enforce a cost on using wireless links. Now, consider two nodes that want to communicate together. There are two options for them, use wireless or just use wired connections. To find the optimum option, the costs of the two different paths are computed and the lowest value can be accepted. Having a low value for $\delta$ makes congestion around the wireless nodes and considering a high value for the cost prevent using the wireless medium as much as possible. On the other hand, the cost value considered in the previous works is constant and therefore it cannot react to the traffic type. For example, for a specific type of traffic, the cost value 2 may be proper, but if the traffic gets congested, it is not good anymore.

In this paper, we use a congestion-aware routing method that can detect congestion around a wireless node and react to it. For each node, we use a table to determine the routing algorithm. There are two routing algorithms, wired and wireless options. It should be noted that wired routing can be used for all communications, but wireless one cannot be used always. For example, if the destination node is adjacent to the source node, the wireless routing cannot be used. The routing table contains two columns and 63 rows, where the number of rows is equal to the number of the possible destinations for each node. The columns are the average latency of staying in the output buffer of the current router for each routing algorithm. As an example, the table of node 14 is shown in Fig. 2.

| Node Id | Wired Latency | Wireless Latency |
|---|---|---|
| 0 | 6.3 | 8.4 |
| 1 | 7.2 | 10.1 |
| ⋮ | ⋮ | ⋮ |
| 63 | 14.8 | 9.2 |

Fig. 2: Routing table for node 14.

As can be seen in this figure, the average latency of being in the output buffer for the wired routing is less than the latency for the wireless routing for node 0. It means that we should use wired routing for sending packets to node 0. On the other hand, consider the last row of the table which is for node 63. For this node, the wireless routing has lower latency than the wired one. So we should use wireless routing for sending packets to this node. It should be noted that we do not use this greedy idea for selecting the routing algorithm for all the packets. Instead, we use an $\epsilon$-greedy algorithm for selection. In this selection algorithm, we consider $\epsilon$ as a non-zero value and with the probability of $\epsilon$ we select the routing algorithm with the higher latency. So, with the probability of $1-\epsilon$, we use the routing algorithm with less latency, and with the probability of $\epsilon$, we select the routing algorithm with the greatest latency. Since the traffic pattern of real applications are not uniform, and they change over time, we give a chance to the non-optimum routing algorithm to refresh its latency. For example, as previously mentioned, wireless routing is better for sending packets to node 63. After a while, it may be congestion around the wireless nodes and wired routing becomes the optimum algorithm for sending packets to this node. Using$\epsilon$, we give this chance to the wired routing algorithm to refresh its latency and therefore be selected for the next packets.

Algorithm 1 describes the above-mentioned selection algorithm. In this algorithm, *r* is a random variable created from a uniform distribution between 0 and 1. If *r* is less than or equal to $\epsilon$ (line 7) then the non-optimum selection would be done, otherwise, the normal routing will be selected (line 12). The id of the destination node comes from the arrived packet, and the values of *wiredLatency(id)* and *wirelessLatency(id)* can be derived from the routing table.

Algorithm 1 Selection of the routing algorithm

```
1   Inputs:
2       r: the random value between 0 and 1
3       id: the id of the destination node
4   Output:
5       The selected routing algorithm
6   Algorithm:
7     if r <= ε then
8       if wiredLatency(id) < wirelessLatency(id) then
9         return wireless_algorithm
10      else
11        return wired_algorithm
12    else
13      if wiredLatency(id) < wirelessLatency(id) then
14        return wired_algorithm
15      else
16       return wireless_algorithm
```

To send a packet, we use wormhole scheduling which divides each packet into some flits. Delivering a packet means delivering all of its flits in order. To send a packet, each flit is sent to the output port of the source router. If the flit can be received by the next router, flit sending is done; otherwise, the flit should stay until the next router can pick it. We compute the latency of this waiting time for each flit and accordingly, we calculate the entries of the routing table. When a new latency is observed, we update the corresponding entry in the routing table according to the weighted sum of the old values and the new ones. The new value of the entry can be calculated as (3):

$$newValue = (1-\alpha) * oldValue + \alpha * observedLatency \quad (3)$$

where the value of $\alpha$ is between 0 and one. If wired routing is selected for the current packet, and then the value of the wiredLatency column will be updated, otherwise, the value of the *wirelessLatency* column will be updated. In the next section, the effect of $\epsilon$ and $\alpha$ on the proposed algorithm will be checked.

Deadlock freedom is the most important property of a routing algorithm. If a routing algorithm is not deadlock-free, it is not applicable. It is important to note that the proposed algorithm works just like a normal wireless routing algorithm which is deadlock-free [9]. The wireless algorithm needs a virtual channel to be deadlock-free. One virtual channel is used for regular wired routing and routing to the first wireless nodes. The other virtual channel is used for the flits that used wireless communication.

Therefore, the virtual channel of flit changes during wireless communication. Before wireless communication, its virtual channel is 0 and after that, its virtual channel is one. It should be noted that the routing from the source node to the wireless node and the routing from the wireless node to the destination node, is done using the XY routing algorithm. Therefore, the wireless node acts as a temporary destination node for the source node and a temporary source node for the destination node.

### *A. Motivation and Innovation*

As stated by the literature, using wireless links in NoC can lead to having better performance results. These performance results can be achieved by a proper routing algorithm. For example, if all nodes in the NoC try to use wireless links to deliver their data, there will be congestion around the wireless routers, which leads to high latency and lower throughput in the NoC.

To lower the wireless usage, related works use static methods like adding an extra cost to the wireless connections, for selecting between wired and wireless routes. These static methods achieve acceptable results in synthetic traffic patterns which the traffic pattern does not change over time. On the other hand, in real world applications, the network traffic pattern will change over time, so using a static method is not suitable for these applications.

In this paper, we proposed an adaptive routing algorithm in wireless NoCs that responds to the network congestion and improves the network performance. We evaluated the proposed method on real world applications from the Parsec benchmark suite.

The detail design of the routing algorithm is shown in Fig. 3. As can be seen in this figure, the routing table is the main component of the routing algorithm. Each router in the NoC maintains its own routing table. The routing table will be updated when the trail flit of the packet exits from the router.

The *ws* and *w* in Fig. 3 are the estimated wireless latency and wired latency corresponding to the packet's destination (*p.d*), respectively.

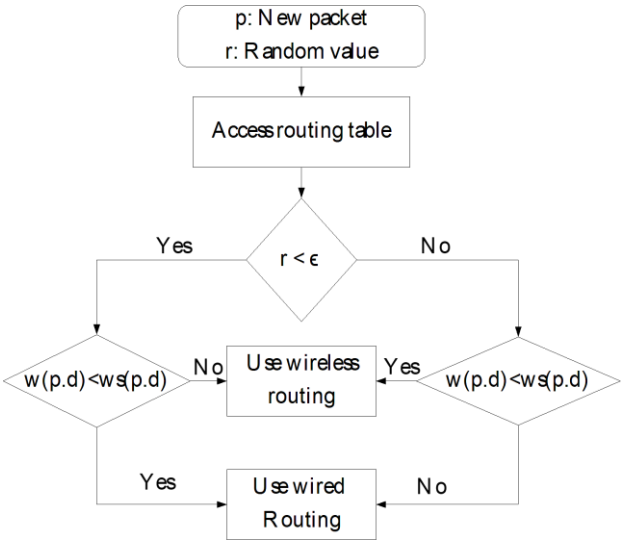


Fig. 3: the flowchart of the proposed routing algorithm.

## Results and Discussion

### *A. Real-World Benchmark Application*

We use real-world applications which are from the Parsec benchmark suite. It contains scientific benchmark programs that use multi-threaded programming and target the many-core and multicore architectures. Since the NoC architectures act as the communication infrastructure for the future many-core processors, we selected this benchmark suite for the performance evaluation of the proposed method. It is important to note that, many related works on the NoC just use the synthetic traffics generated for simple scenarios like uniform and hotspot traffics which may not occur in real-world applications.

### *B. Simulation Environment*

We use a modified version of the Noxim cycle-accurate NoC simulator for experimental results. We modified it to have the virtual channels and to be proper for implementing our routing algorithm. The traffic patterns of the applications are extracted from the Netrace [30] tool and the packets are injected into the Noxim simulator from an implemented interface. Since the Netrace tool is generating traffic for a 64 node NoC, we use a WNoC with 64 nodes to investigate the proposed algorithm.

The parameters of the NoC simulator are shown in Table 1.

Table 1: Simulation parameters

| parameter | Value |
|---|---|
| number of rows | 8 |
| number of columns | 8 |
| number of the wireless nodes | 4 |
| flit width | 32 |

### *C. Comparison Metrics*

We used three comparison metrics for comparing the routing algorithms.

The first metric is the network throughput computed as the ratio of the flits injected to the network per core in each cycle. Throughput is the main metric for comparing the routing algorithms. The second metric is the average flit latency.

It is the average value for the latency of delivering a flit. The latency of a flit is computed as the difference between the times of the injection and delivery of a flit. The third metric is the power consumption of the routing algorithm.

Routing algorithms with low power consumption are desirable.

### *D. The effect of Cost ($\delta$)*

For the first set of experiments, we investigate the effect of $\delta$ on the above-mentioned metrics.

The effect of $\delta$ on network throughput is shown in Fig. 4.

As can be seen in this figure, having greater values of $\delta$ improves the network latency.

The main reason for this effect is that in such cases, fewer packets are delivered through the wireless connection and therefore there is less congestion around wireless nodes.

It is important to note that having greater values for $\delta$ wastes the wireless equipment because very low ratio of the packets go through the wireless connections.

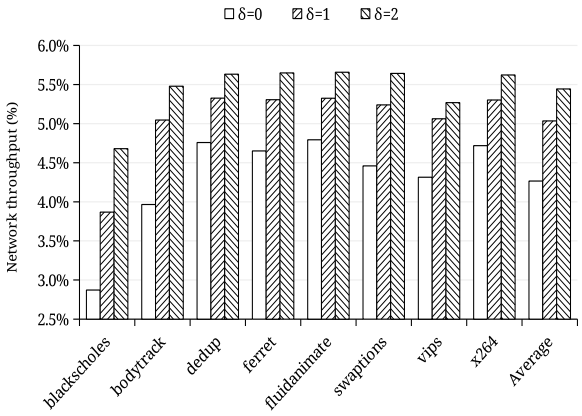


Fig. 4: The effect of $\delta$ on the throughput of the NoC.

The effect of $\delta$ on the average flit latency is shown in Fig. 5. As shown in this figure, having greater values for $\delta$ leads to lower values for average flit latency. The main reason for this effect is having less congestion around wireless nodes. Finally, Fig. 6 shows the effect of $\delta$ on the power consumption of the routing algorithm. As can be seen, the average power is almost the same for all routing algorithms which means having $\delta$ does not incur additional power consumption on the routing algorithm.

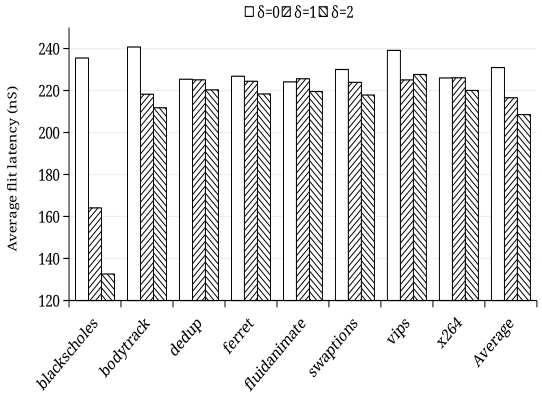


Fig. 5: The effect of $\delta$ on the average flit latency of the NoC.

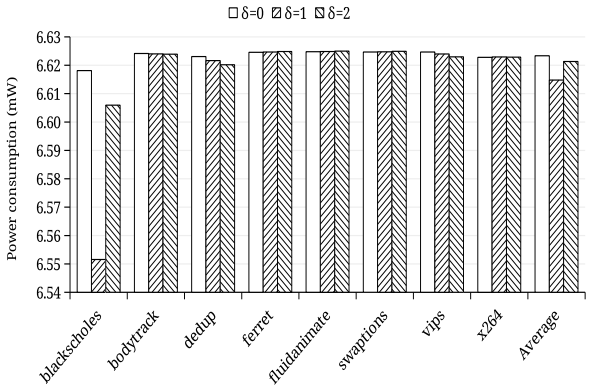


Fig. 6: The effect of $\delta$ on the power consumption of the NoC.

### E. *The effect of $\epsilon$*

To investigate the effect of the $\epsilon$, we consider five different values from 0.01 to 0.5 for it. The obtained results have been shown in Fig. 7, Fig. 8, and Fig. 9. Although the different programs have different behavior against$\epsilon$, two facts should be considered: 1) $\epsilon$ affects the network throughput of the routing algorithm, 2) the results show that $\epsilon = 0.05$ is the best value from the point of view of the network throughput.

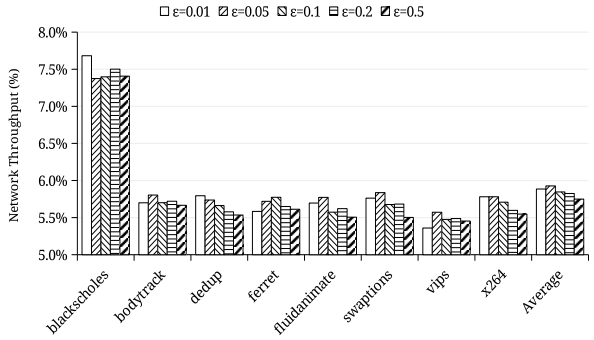


Fig. 7: The effect of $\epsilon$ on the throughput of the NoC.

The effect of $\epsilon$ on the average flit latency and the power consumption of the proposed algorithm have been illustrated in Fig. 8 and Fig. 9 respectively. As shown in these figures, on average, having a higher value for $\epsilon$leads to the greatest latency and more power consumption. Although the average flit latency for $\epsilon$=0.01 and $\epsilon$=0.05 are almost equal, the power consumption of the routing algorithm for $\epsilon$=0.05 is 2.37% higher than the power consumption of the routing algorithm with $\epsilon$=0.01.

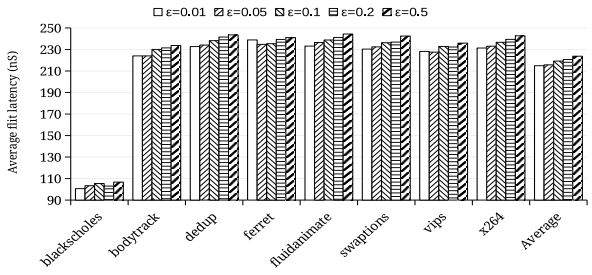


Fig. 8: The effect of $\epsilon$ on the average flit latency of the NoC.

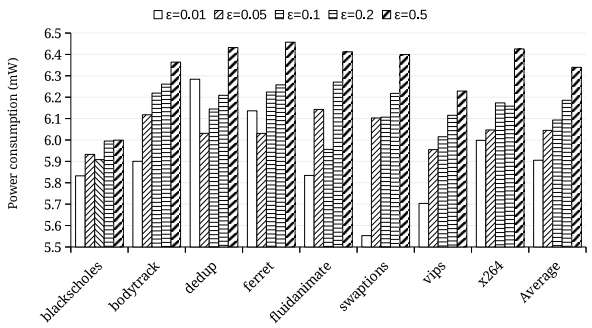


Fig. 9: The effect of $\epsilon$ on the power consumption of the NoC.

### F. *The effect of $\alpha$*

The $\alpha$ value is used to regulate the contributions of the flit latency history and the new flit latency in (3). Higher values for $\alpha$ means that we consider more weight for the new flit latency, and lower values mean that we pay more attention to the history of the flit latency. To investigate the effect of the $\alpha$ on the performance of the proposed routing algorithm, six values of $\alpha$ ranging from 0.1 to 1.0 are considered. Fig. 10 shows the effect of $\alpha$ on the network throughput of the proposed algorithm. As shown in this figure, lower values of $\alpha$ result in higher throughput. An interesting case is for $\alpha$=0.9 which has better network throughput than $\alpha$=1.0.

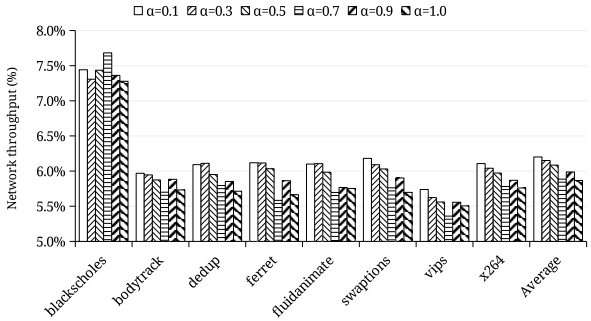


Fig. 10: The effect of $\alpha$ on the throughput of the NoC.

The effect of $\alpha$ on the average flit latency and the power consumption of the proposed method is shown in Fig. 11 and Fig. 12. As shown in Fig. 11, $\alpha$=0.5 has the lowest average flit latency and $\alpha$=1.0 has the highest average flit latency. It is important to note that the difference between these two values is less than 4.88%. As can be seen in this figure, the $\alpha$ has regular effect on the average flit latency of the proposed method. On the contrary, the power consumption of the proposed method declines with higher values of $\alpha$. The power consumption of the proposed method with $\alpha$=0.1 is the highest, while the power consumption with $\alpha$=0.9 has the lowest value. Again the difference between the highest and the lowest values is 35.8%. Having low power consumption for $\alpha$=0.9 gives the idea of a low power routing algorithm alongside a high performance routing algorithm (As in $\alpha$=0.1).

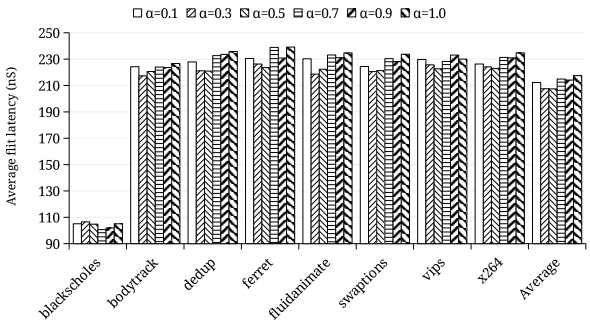


Fig. 11: The effect of $\alpha$ on the average flit latency of the NoC

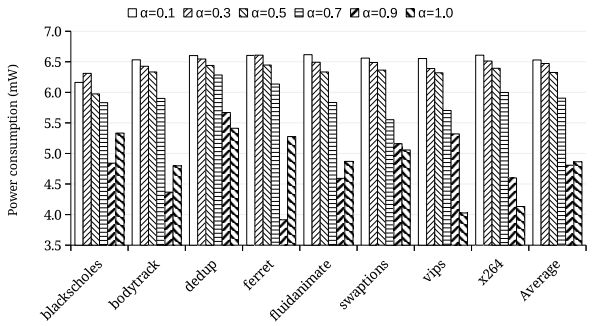


Fig. 12: The effect of $\alpha$ on the power consumption of the NoC.

### *G. Comparison to the Related Works*

To compare the proposed method with the related works, we considered two configurations: 1) a high-performance configuration with the best network throughput, and 2) a low power configuration with acceptable network throughput. We compared the proposed configurations to the routing algorithm used in [9], [31], [32], and [33]. It is important to note that the simulation results obtained by the Parsec benchmark for all the methods. The comparison of the network throughput, average flit latency, and power consumption with the related works has been shown in Table 2. As shown in this table, the first configuration has the best network throughput compared to the related works. On the other hand, the second configuration has the best power consumption. It is important to note that the method in [9] achieved the lowest flit latency.

Table 2: Comparison of the proposed method to the related works

| Method | Throughput (%) | Latency (nS) | Power (mW) |
|---|---|---|---|
| Proposed Method (α =0.1) | **6.20** | 212.32 | 6.53 |
| Proposed Method (α =0.9) | 5.98 | 214.17 | **4.81** |
| [9] | 5.44 | **208.51** | 6.62 |
| [31] | 4.53 | 229.80 | 6.61 |
| [32] | 4.27 | 230.96 | 6.62 |
| [33] | 5.20 | 242.39 | 6.54 |

## Conclusions

Having congestion around the wireless nodes is the main drawback of the conventional routing algorithms for the WNoC architectures. In this paper, we proposed a learning-based algorithm that uses a history of the flit latency to select the proper trajectory of the flits. To evaluate the proposed method, three metrics were considered: 1) the network throughput, 2) the average flit latency, and 3) the power consumption. We thoroughly investigated the parameters of the proposed routing algorithm and finally, achieved two configurations. The first one is a high-performance configuration that has the best network throughput and the second one has the lower power consumption while having an acceptable network throughput. We investigated the proposed algorithm on eight real-world scientific applications from the Parsec benchmark suite. The simulation results showed that the proposed algorithm can achieve 13.9% more throughput and consume 1.4% less power in comparison with the previous works.

## Author Contributions

The main idea of the paper is proposed by A. Tajary and E. Tahanian. The simulation and data analysis are carried out by A. Tajary. A. Tajary wrote the first draft of the manuscript and E. Tahanian corrected it.

## Acknowledgment

The authors would like to thank the editor and reviewers for their valuable comments on the manuscript.

## Conflict of Interest

The authors declare no potential conflict of interest regarding the publication of this work. In addition, the ethical issues including plagiarism, informed consent, misconduct, data fabrication and, or falsification, double publication and, or submission, and redundancy have been completely witnessed by the authors.

## Abbreviations

| | |
|---|---|
| *NoC* | Network on Chip |
| *WNoC* | Wireless Network on Chip |
| *sx* | The row number of the source node |
| *dx* | The row number of the destination node |
| *sy* | The column number of the source node |
| *dy* | The column number of the destination node |
| *wnsx* | The row number of the wireless node near the source node |
| *wnsy* | The column number of the wireless node near the source node |
| *wndx* | The row number of the wireless node near the destination node |
| *wndy* | The column number of the wireless node near the destination node |
| $\delta$ | The cost value for using wireless connection |
| $\epsilon$ | The value of epsilon in $\epsilon$-greedy algorithm |
| $\alpha$ | The update rate of the algorithm |

| | |
|---|---|
| | based on the current and previous values of the latency |
| *r* | A random value with uniform distribution between 0 and 1 |
| *ws* | The cost of using wireless links |
| *w* | The cost of using wired links |
| *p.d* | The destination of the packet |

## References


[1] H. Jang et al., "Developing a multicore platform utilizing open risc-v cores," IEEE Access, 9: 120010–120023, 2021.

[2] P. Kansakar, A. Munir, "A design space exploration methodology for parameter optimization in multicore processors," IEEE Trans. Parallel Distrib. Syst., 29(1): 2–15, 2018.

[3] Y. Liu, S. Kato, M. Edahiro, "Analysis of memory system of tiled many-core processors," IEEE Access, 7: 18964–18977, 2019.

[4] A. Vijaya Bhaskar, T. Venkatesh, "Performance analysis of network-on-chip in many-core processors," J. Parallel Distrib. Comput., 147: 196–208, 2021.

[5] F.N. Sibai, "A two-dimensional low-diameter scalable on-chip network for interconnecting thousands of cores," IEEE Trans. Parallel Distrib. Syst., 23(2): 193–201, 2012.

[6] A. Balakrishnan, A. Naeemi, "Optimal global interconnects for networks-on-chip in many-core architectures," IEEE Electron Device Lett., 31(4): 290–292, 2010.

[7] S.D. Chawade, M.A. Gaikwad, R.M. Patrikar, "Review of xy routing algorithm for network-on-chip architecture," Int. J. Comput. Appl., 43(21): 975–8887, 2012.

[8] B. Bahrami, M.A. Jabraeil Jamali, S. Saeidi, "A novel hierarchical architecture for wireless network-on-chip," J. Parallel Distrib. Comput., 120: 307–321, 2018.

[9] E. Tahanian, A. Tajary, M. Rezvani, M. Fateh, "Scalable thz network-on-chip architecture for multichip systems," J. Comput. Netw. Commun., 2020: 8823938, Dec. 2020.

[10] B. Halavar, B. Talawar, "Power and performance analysis of 3D network-on-chip architectures," Comput. Electr. Eng., 83: 106592, 2020

[11] Z. Wang, H. Gu, Y. Chen, Y. Yang, K. Wang, "3D network-on-chip design for embedded ubiquitous computing systems," J. Syst. Archit., 76: 39–46, 2017.

[12] Z. Liu, G. Song, Y. Zhou, Z. Zhang, F. Cheng, "Layout exploration for three-dimensional networks-on-chip: Chemical equilibrium simulation approach," Microelectron. J., 115: 105195, 2021.

[13] D.A. Hamdi, S. Ghoniemy, Y. Dakroury, M.A. Sobh, "A scalable software defined network orchestrator for photonic network on chips," IEEE Access, 9: 35371–35381, 2021.

[14] K. Sharma, V.K. Sehgal, "Energy-efficient and sustainable communication in optical networks on chip," Sustainable Comput: Inform. Syst., 28: 100426, 2020.

[15] A.B. Ahmed, T. Yoshinaga, A.B. Abdallah, "Scalable photonic networks-on-chip architecture based on a novel wavelength-shifting mechanism," IEEE Trans. Emerg. Top. Comput., 8(2): 533–544, 2020.

[16] D. DiTomaso, A. Kodi, D. Matolak, S. Kaya, S. Laha, W. Rayess, "A-winoc: Adaptive wireless network-on-chip architecture for chip multiprocessors," IEEE Trans. Parallel Distrib. Syst., 26(12): 3289–3302, 2015.

[17] H.K. Mondal, S.H. Gade, M.S. Shamim, S. Deb, A. Ganguly, "Interference-aware wireless network-on-chip architecture using directional antennas," IEEE Trans. Multi-Scale Comput. Syst., 3(3): 193–205, 2017.

[18] N. Mansoor, P.J.S. Iruthayaraj, A. Ganguly, "Design methodology for a robust and energy-efficient millimeter-wave wireless network-on-chip," IEEE Trans. Multi-Scale Comput. Syst., 1(1): 33–45, 2015.

[19] S. Abadal, J. Torrellas, E. Alarcón, A. Cabellos-Aparicio, "OrthoNoC: A broadcast-oriented dual-plane wireless network-on-chip architecture," IEEE Trans. Parallel Distrib. Syst., 29(3): 628–641, 2018.

[20] R.S. Narde, J. Venkataraman, A. Ganguly, I. Puchades, "Intra- and inter-chip transmission of millimeter-wave interconnects in noc-based multi-chip systems," IEEE Access, 7: 112200–112215, 2019.

[21] K. Duraisamy, Y. Xue, P. Bogdan, P.P. Pande, "Multicast-aware high-performance wireless network-on-chip architectures," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., 25(3): 1126–1139, 2017.

[22] A. Tajary, E. Tahanian, "A routing-aware simulated annealing-based placement method in wireless network on chips," J. AI Data Min., 8(3): 409–415, 2020.

[23] W.H. Hu, C. Wang, N. Bagherzadeh, "Design and analysis of a mesh-based wireless network-on-chip," J. Supercomput., 71(8): 2830–2846, 2015.

[24] B. Bahrami, M.A. Jabraeil Jamali, S. Saeidi, "A hierarchical architecture based on traveling salesman problem for hybrid wireless network-on-chip," Wireless Networks, 25(5): 2187–2200, 2019.

[25] V. Catania, A. Mineo, S. Monteleone, M. Palesi, D. Patti, "Improving the energy efficiency of wireless network on chip architectures through online selective buffers and receivers shutdown," in Proc. 2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC): 668–673, 2016.

[26] N. Jiang et al., "A detailed and flexible cycle-accurate network-on-chip simulator," in Proc. 2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS): 86–96, 2013.

[27] C. Bienia, "Benchmarking modern multiprocessors," PhD thesis, Princeton University, 2011.

[28] F. Farahnakian, M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, "Q-learning based congestion-aware routing algorithm for on-chip network," in Proc. 2011 IEEE 2nd International Conference on Networked Embedded Systems for Enterprise Applications: 1–7, 2011.

[29] F. Farahnakian, M. Ebrahimi, M. Daneshtalab, J. Plosila, P. Liljeberg, "Adaptive reinforcement learning method for networks-on-chip," in Proc. 2012 International Conference on Embedded Computer Systems (SAMOS): 236–243, 2012.

[30] J. Hestness, S.W. Keckler, "Netrace: Dependency-tracking traces for efficient network-on-chip experimentation," The University of Texas at Austin, Department of Computer Science, 2011.

[31] M. Devanathan, V. Ranganathan, P. Sivakumar, "Congestion-aware wireless network-on-chip for high-speed communication," Automatika, 61(1): 92–98, 2020.

[32] A.I. Fasiku, B.O. Ojedayo, O.E. Oyinloye, "Effect of routing algorithm on wireless network-on-chip performance," in Proc. 2020 second international sustainability and resilience conference: Technology and innovation in building designs (51154): 1–5, 2020.

[33] F. Rad, M. Reshadi, A. Khademzadeh, "A novel arbitration mechanism for crossbar switch in wireless network-on-chip," Cluster Comput., 24(2): 1185–1198, 2021.

[34] M.S. Shamim, N. Mansoor, R.S. Narde, V. Kothandapani, A. Ganguly, J. Venkataraman, "A wireless interconnection framework for seamless inter and intra-chip communication in multichip systems," IEEE Trans. Comput., 66(3): 389–402, 2017.

## Biographies

**Alireza Tajary** received his B.Sc., M.Sc., and Ph.D. from Amirkabir University of Technology in 2008, 2011, and 2018 respectively. Since 2018, he is with the Faculty of Computer Engineering, Shahrood University of Technology as an Assistant Professor. His main research interests include computer architecture, network on chip, and fault tolerant computing.


- Email: tajary@shahroodut.ac.ir
- ORCID: 0000-0003-0530-9827
- Web of Science Researcher ID: AGY-1402-2022
- Scopus Author ID: 35811245700
- Homepage: https://shahroodut.ac.ir/en/as/?id=S908


**Esmaeel Tahanian** received the B.Sc. and M.Sc. degrees in communication engineering from K.N.T university of technology, Tehran, Iran in 2008 and 2010, respectively. He received the Ph.D. degree in communication engineering from Shahed university, Tehran, Iran in 2016. His main research interests include computer network especially wireless network and Network on Chip (NoC).


- Email: e.tahanian@shahroodut.ac.ir
- ORCID: 0000-0002-5418-2490
- Web of Science Researcher ID: AGY-1846-2022
- Scopus Author ID: 57221046099
- Homepage: https://shahroodut.ac.ir/en/as/?id=S887



**Copyrights**

©2022 The author(s). This is an open access article distributed under the terms of the Creative Commons Attribution (CC BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, as long as the original authors and source are cited. No permission is required from the authors or the publishers.



**How to cite this paper:**

A. Tajary, E. Tahanian, "An adaptive routing algorithm for wireless network on chips," J. Electr. Comput. Eng. Innovations, 10(2): 487-496, 2022.

**DOI:** 10.22061/JECEI.2022.8464.518

**URL:** https://jecei.sru.ac.ir/article_1702.html