Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 07, No. 03, September 2021.

INTRODUCING A NEW ROUTING ALGORITHM FOR WIRELESS NETWORKS ON CHIP USING REINFORCEMENT LEARNING

Zohreh Harati[1], Esmaeel Tahanian[2], Alireza Tajary[3] and Mansoor Fateh[4]

(Received: 16-Apr.-2021, Revised: 13-Jun.-2021, Accepted: 3-Jul.-2021)

ABSTRACT

Wireless network on chip (WNoC) can be used as an alternative to bus technology in high-core chips in which the multi-hop paths between far apart cores are replaced with a wireless single-hop link. The main reason for using wireless communication is to reduce latency as well as power consumption. According to the limitation of resources, the performance of the WNoC is sensitive to the routing algorithm. While an appropriate routing algorithm reduces latency, it should avoid deadlock. In this paper, we propose a novel routing algorithm using Q-learning, which is one of the reinforcement learning methods for balancing wireless network traffic on the chip. Using such an algorithm, the nodes can make decisions based on congestion conditions in the network when transferring flits from the source node to the destination one. The simulation results show that using the proposed reinforcement learning for routing the packets considerably improves the performance of the network; more precisely, the system performance is improved by 8% compared with the previous related works.

KEYWORDS

Wireless network on chip (WNoC), Q-learning algorithm, Reinforcement learning (RL), Routing algorithm, Deadlock.

1. INTRODUCTION

Coincidence with the development of the electronic world and the increase in the ability to design circuits at the nanoscale, integration in the design of electronic devices has become the talk of the day and efforts to design and build integrated devices continue. These efforts have created a new field called the System on Chip (SoC). The SoC tries to integrate the processing, communication and interface cores. One of the problems with this integration is how to connect and communicate between these cores.

NoC [1]-[2] is a new solution for interconnection within the SoC, which was introduced in 1999. In traditional solutions, intermediate connections were made using the structure of the bus. As circuits became more compact, bus-based solutions lost their effectiveness in contrast to the need for new technologies. Crossings began to restrict and, at worst, block traffic. In networks on chip, there are networks like computer networks, in which different parts communicate with each other through this network by sending packaged data. An NoC like a computer network includes cores, routers that route traffic between cores and wires that connect devices to routers and routers to each other.

In this network, packets or flits are used to send data. In general, the package consists of at least three parts, which include the header, body and end of the package tail. Each package has only one header, one or more bodies and one end of the package tail. When sending data, the packet header is sent first and the path is reserved for data transmission. When one or more bodies are sent, by sending the end of the package tail, the reserved path is released so that it can be reserved by other nodes. In a WNoC [33]-[34], multi-hop wired paths between far apart cores are replaced by high-bandwidth single-hop long-range wireless links. Consequently, reduction of average hop count leads to better performance of the network and especially reduces latency and power consumption. The idea of the NoC using wireless links has been proposed for the first time in 2010 [3] and has so far been considered from different aspects such as routing algorithms. The routing algorithm identifies the path between the source router and the destination router. Packets have two strategies for arriving at the destination in WNoC. In the first strategy, the packets use the nearest wireless nodes to the sender and receiver for arriving at the destination. In the second strategy, the packets are routed only through wired links. Clearly, the path

Z. Harati, E. Tahanian, A. Tajary and M. Fateh are with Department of Artificial Intelligence Engineering, Shahrood University of Technology, Shahrood, Iran. Emails: [1]zohreh.harati@gmail.com, [2]e.tahanian@shahroodut.ac.ir, [3]tajary@shahroodut.ac.ir and [4]mansoor_fateh@shahroodut.ac.ir

with a minimum number of hops would be chosen by the traveling packets as the shortest path. So far, the most important routing methods that have been proposed include source and distributed routing [4], [5], [6], [7], deterministic and adaptive routing [8]-[10], minimal and non-minimal routing [11]-[14], thermal-aware routing [15]-[18] and fault-tolerant routing [19]-[22], [37].

One of the most important routing algorithms is the learning-based algorithm that can be categorized as an adaptive routing algorithm [46]. In the deterministic routing algorithm, it is decided only based on the source and destination addresses and all packets that have the same source and destination addresses pass the same route. But in the adaptive routing algorithm, in addition to the source and destination addresses, the network time traffic is also effective in determining the route. Therefore, packets that have the same source and destination addresses may use different paths with different delays for data transmission.

Reinforcement learning is a type of learning in which the correct action in each situation is determined by a standard. It is the task of the teaching agent to learn the best action in any situation by having information [38]-[40]. This is part of the specific strengths of reinforcement learning. With the help of reinforcement learning, often the complexities of the decision can be solved with the least amount of information required.

In reinforcement learning, the main purpose of learning is to perform a task or achieve a goal without which the learning agent is fed with external direct information. In reinforcement learning, when the agent performs a task that makes him closer to his goal, rewards are received and the goal is to take steps to maximize the agent's reward for the long term [23]-[24]. In reinforcement learning, rewards and punishments are used as signals to improve the final performance of the system [10], [25].

In this paper, we propose a routing algorithm based on reinforcement learning which can execute on the wireless chip and uses flit to move data in a network on a chip. In the proposed method, we use a reinforcement learning method called Q-learning, which is a value-based method. This learning has a function Q the inputs of which are states and actions. To learn this function uses a table in which rows are states and columns are actions. If we have an agent who is at first home and can perform certain movements to act, to continue the activity of the agent, he must look at the table to obtain the value of Q from the starting position and based on specific actions and choose whichever has a higher Q value, receive the reward and update the value of Q based on the Bellman relation. The function Q indicates how much the agent may be rewarded on its path.

Q-routing is a network routing method based on the Q-learning algorithm. Each node makes its own routing decisions based on information about its neighboring nodes. The node stores tables of values Q that estimate the quality of the alternative paths. These values are updated each time a node sends a packet to one of its neighbors. Thus, with packets of node paths, the values Q gradually hold more information. This routing algorithm can adapt to the network. In this way, if the packet path from the source node to the destination node is crowded, it changes the route and chooses the less crowded route. Redirection by the algorithm is based on the information in the tables of its nodes, where the information of these tables is updated based on the information obtained in each node movement. In fact, learning in this routing algorithm is the product of changing table values.

The remainder of this paper is organized as follows. In Section 2, the related work is discussed. XY routing algorithm, wireless routing algorithm and flit and Q-learning algorithm are given in Section 3. In Section 4, the proposed method for NoCs is explained. The results are reported in Section 5, while the summary and conclusion are given in the last section.

2. RELATED WORK

In the past, a lot of research has been done to improve the performance of NoC routing. Recently, learning-based algorithms have been presented that can be categorized as adaptive routing algorithms. In the deterministic routing algorithm, the paths of the packets are determined only based on the source and destination addresses and all the packets that have the same source and destination addresses pass the same route. But in the adaptive routing algorithm, in addition to the source and destination addresses, the network time traffic is also effective in determining the route. Therefore, packets that have the same source and destination addresses may use different paths with different delays for data transmission.

Farahnakian et al. used reinforcement learning to balance traffic in the network. In this work, the Q - learning technique is used and the state is the node, the agent is the packet and the action is the output port selection. In this way, there is a table for each node that shows the different states of movement of the packet from that node in the network and the values of this table are equivalent to the delay that will be spent to reach the packet from the source node to destination node. When a packet reaches a node, it checks the table, because the goal is to select the least crowded path. If the delay value is assumed to be positive, it selects the lowest value and sends the packet in that direction. When the packet is sent through this node and the receiving node receives it, the receiving node sends its information about the status of its ports to the sender node through the learning packet and the sending node uses this information to update its table. This routing is a one-way routing, because when a packet is sent from node x to node y, only the sender node table, node x, is updated. Considering the packet instead of the flit is the main drawback of this work. In addition, this method was presented only for wired NoCs. Figure 1 shows an example of one-way routing.

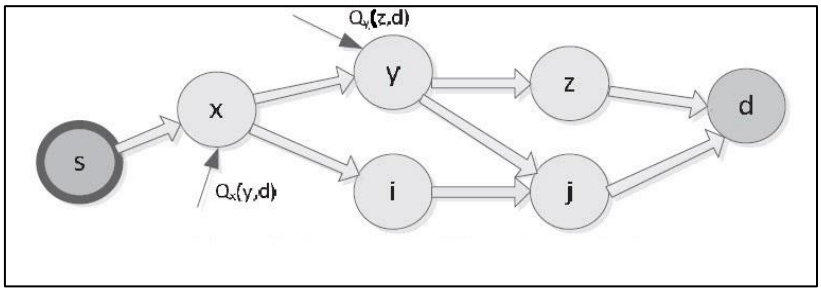

Figure 1. An example of one-way routing [26].

In reference [27] to solve the traffic problem in networks on chip, two-way reinforcement learning has been used. In this method, after selecting the path, the sender node also sends information about the congestion status of its ports when sending the packet. This information is used to update the receiver node learning table. When the packet is sent with this added information, the receiving node receives this information and extracts the required information from the received packet and sends its related information to the sending node with the learning packet. With this information that the receiver and the sender node receive, they update their tables. This is why it is said two-way. The proposed method in this reference finds the optimal path faster and improves the performance of the network. Again, this reference has considered only wired NoC as well as packet instead of the flit. Figure 2 shows an example of two-way routing.

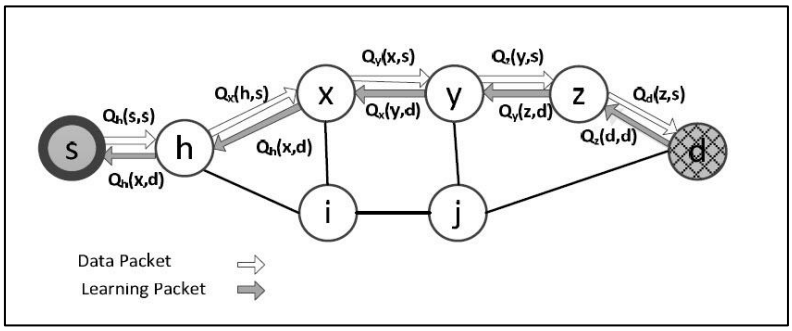

Figure 2. An example of two-way routing [27].

In reference [28], the performance of the network on the chip is optimized using reinforcement learning. Farahnakian et al. in [29] proposed an adaptive routing algorithm that distributes traffic using a learning method across the network. In reference [30], a congestion-aware routing algorithm based on Q-learning is presented, which divides the network into several clusters and each cluster maintains a table. This table stores local and general congestion information about alternative routes to send packets to the destination cluster. Each cluster can select a low-density output channel based on table information. Also, to further update the tables, both learning packages and data participate in the publication of congestion information. In this reference, flit is used for data transmission, but wireless links are not used.

It is worth noting that in all mentioned routing algorithms based on reinforcement learning, the data was transferred in a wired network and a packet was used to transfer data. In this paper, we apply reinforcement learning to improve the routing in a wireless network on chip when the flits are used to transfer data.

3. BACKGROUND

This section briefly reviews XY routing algorithm, wireless routing algorithm and Q-learning algorithm.

3.1 XY Routing Algorithm

By using the XY routing algorithm which is a kind of distributed deterministic routing algorithms, the flits first follow the X-path, then they move in the Y direction to reach their destination.

3.2 Wireless Routing Algorithm

Using wireless connections, the algorithm sends the packet faster from the source node to the destination one [41]-[43], [45]. In this kind of algorithms, when a packet reaches a node, two cases are checked:

- Send packet from the source to the destination by XY method.
- Send a packet from the source to the destination by sending the packet to the nearest wireless router to the source, receiving it at the nearest wireless router to the destination and then sending the packet to the destination node.

The selection of each of the above-mentioned cases is done by calculating the Cartesian distance between the source and the destination, once without considering the wireless node and once using the wireless node. If the first distance is less than the second distance, the XY routing algorithm method will be used; otherwise, the wireless XY routing algorithm is selected.

If there is no wireless node, the Cartesian distance (CD) is equal to:

$$CD = |\text{The distance between the source column and the destination column}| + |\text{distance between source row and destination row}| \quad (1)$$

and if there is a wireless node, the Cartesian distance is equal to [41]:

$$CD = |\text{Cartesian distance between the destination and the nearest wireless node to the destination}| + |\text{Cartesian distance between the source and the nearest wireless node to the source}| + \text{the cost of using wireless node.} \quad (2)$$

3.3 Q-learning Algorithm

Q-learning is a reinforcement learning technique in which by learning a state-action function, the agent follows a specific policy for performing different movements in different situations [44]. In the Q-learning algorithm, each state-action pair is assigned a value of Q (s, a), which is the sum of the rewards received. When the agent starts from states, operates a and follows the existing policy, until it converges to the optimal value, Equation (3) (known as Bellman's relation) is used to update.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t(s_t, a_t) \times [R(s_t) + \gamma \max Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \quad (3)$$

The core of the algorithm consists of a simple iterative update. In this way, the previous values are modified based on the new information. The Q-learning algorithm uses tables to store data. Each table maintains four fields named Current-State Space, Next-State Space, Action Space and Q-value. Current-State Space refers to everything that the agent currently perceives, while Next-State Space is determined by the Current-State and the actions selected by the agents. The Action Space indicates the actions that the agent can perform. Each Q-value is accessible by Q(s, a), representing the expected reinforcement of taking action in states. Algorithm 1 shows this procedure.

Algorithm 1. Q-learning algorithm.

1. **Input**: **PIR**: packet Injection Rate, **T**: simulation Time, **WN**: wireless Nodes Set, **lr**: learning rate, $m \times n$: network size, **df**: discount factor.
2. **Output**: Final Q (s_t, a_t)
3. ST ← 1,2,3, …, T; N ← 1,2, …, $m \times n$
4. Initialize Q(s,a) arbitarily
5. **For** i ∈ ST do
6. Initialize s
7. **For** j ∈ N **do**
8. Select action of agent from s using policy derived from Q(e.g, Є-greedy)
9. $\mathbf{Q}(s_t, a_t) \leftarrow \mathbf{Q}(s_{t+1}, a_{t+1}) + \boldsymbol{a}[r_{t+1} + \gamma max_a * \mathbf{Q}(s_{t+1}, a_{t+1}) - \mathbf{Q}(s_t, a_t)]$
10. s ←s';
11. end
12. end
13. Return Q(s_t, a_t)

4. THE PROPOSED METHOD

In this section, to use Q-learning in wireless NoC, we first properly modify the conventional Q-table. Then, we explain how to fill this table. Finally, we describe the proposed Q-learning algorithm for the wireless NoC.

4.1 The Customized Q-table for the Wireless NoC

The customized Q-table for node 1 in a 3*3 WNoC is illustrated in Table 1. The table contains one field for the source, one field for the destination, four fields for the neighboring nodes (indicating the nodes we must first go to when moving from the source to the destination), four fields for the direction and four fields for the delay.

Table 1. Table values in a 3 * 3 network with two nodes; namely, 1 and 6, as the wireless nodes.

Source	Neighboring node				Direction				Delay				Destination
Node1	0				3				0				Node0
Node1	1				4				0				Node1
Node1	2				1				0				Node2
Node1	0	4	6		3	2	5		0	0	0		Node3
Node1	4				2				0				Node4
Node1	2	4			1	2			0	0			Node5
Node1	0	4	6		3	2	5		0	0	0		Node 6
Node1	4		6		2		5		0		0		Node 7
Node1	2	4	6		1	2	5		0	0	0		Node 8

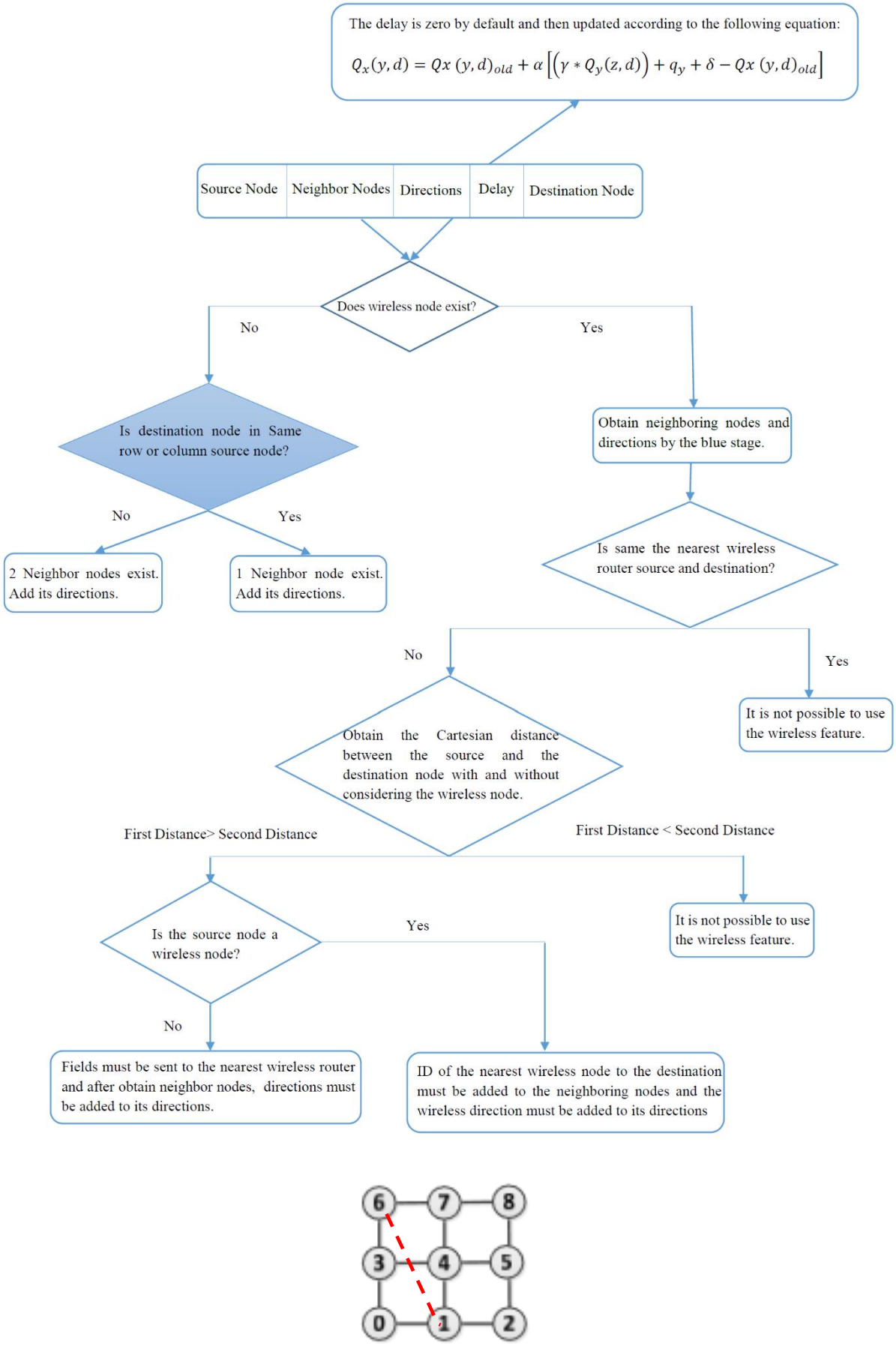

Figure 3. A typical 3*3 mesh network with a wireless link between nodes 1 and 6.

- In that table, there is one row for all destinations. For example, in a 3 * 3 mesh, we have 9 rows for each source node, which is from 0 to 8.
- Each row contains information about sending data from the source node to the destination node.
- There is a table for each node in the network.

Filling the four fields of the neighbor nodes is as follows:

- If there is no wireless node in the network and the destination node is in the same row or column of the source node, the source node has one direction to send data to the destination node and therefore there is only a neighboring node in the table. On the other hand, if the source node and the destination node are not in the same row or column, there are two directions for sending data. Consequently, the source node has two neighboring nodes.
- If there is a wireless node in the network, in addition to obtaining the neighboring nodes, the use of wireless capability should be checked.

If the wireless transmission is possible, two cases may occur: first, the source node is a wireless node and the other source node is not a wireless node. If the source node is wireless, the ID of the nearest wireless node to the destination must be added to the neighboring nodes. This is because the next node is the wireless node to which the fields must be sent and the wireless direction must be added to its directions. If the source is not wireless, the fields must be sent to the nearest wireless router to use the wireless capability.

If the nearest wireless source and destination are not the same, we should consider whether it is good to use wireless capability. To do this, we must obtain the Cartesian distance between the source and the destination node with and without considering the wireless node. If the second distance is less than the first one, the wireless capability will not be used; otherwise, the wireless capability will be used. Furthermore, the delay values when creating the table for each node are zero by default.

To calculate and update the delay values, Equation (4) is used, which simulates the Bellman relation [35] in the NoC for data transmission [26].

$$Q_x(y,d) = Qx\ (y,d)_{old} + \alpha\left[\left(\gamma * Q_y(z,d)\right) + q_y + \delta - Qx\ (y,d)_{old}\right] \quad (4)$$

This relation calculates the amount of new latency when sending a packet from node *x* to node *d* through the neighboring node *y*. In this regard, α is the learning rate that determines the rate of newer information. In addition, *qy* indicates how long the packet sent from node *x* waits in the input buffer of node *y*. δ indicates the length of time that a packet is sent from node *x* to node *y*. Moreover, $Q_y(z,d)$ indicates the length of time that the packet goes from node *y* to node *d* with the help of the neighboring node *z*. $Qx\ (y,d)_{old}$ indicates the amount of delay in the pre-update table and γ is the discount factor that determines the importance of future rewards and therefore affects the value of $Q_y(z,d)$.

Each of the above values is obtained as follows: The value $Qx\ (y,d)_{old}$ exists in the table. To obtain the value of $Q_y(z,d)$, a separate table must be created with node tables in each cell and this table must be publicly available. Of course, this only allows access to the next node table. Therefore, it must be checked which row of the destination equals the intended destination. When the desired row is obtained, the delay the value of which is the least and is opposite to -1 is selected as the delay from node y to destination *d*.

To calculate δ and q_y, when a packet is placed in one direction, the sc_time_stamp () function is used to find the packet's time in the desired direction and when the packet is received and a confirmation message is issued, its time will be counted. The difference between the two times indicates the amount of delay.

4.2 The Proposed Routing Algorithm

After creating the table and explaining how to update it, the performance of the proposed algorithm is examined. This algorithm considers the amount of delay and which of the delay values is less; its direction is considered as the next direction of packet movement.

In general, the procedure of the program is that for each node, a table is created and completed using the given information and for the delay fields, zero values are placed first. Then, using the proposed routing

algorithm, a packet path is selected. Because initially the amount of delay is considered zero by fefault, the direction of the first neighbor node is chosen to move the packet. After selecting the path, the delay value is calculated and updated using Equation (4) and the data will continue to move on the path of selection and this procedure continues to reach the destination.

5. EXPERIMENTAL RESULT

Noxim simulator [31] has been used to implement the learning-based routing algorithm and to perform its simulation. Noxim has a command line that can give several parameters as input to the simulator, such as buffer size, type of routing algorithm, location of wireless nodes, number of virtual channels, simulated duration and traffic type, …etc. The simulation output, including power, packet transmission delay, number of injected fields, the percentage of packets received *via* wireless, …etc., is given to the user.

In these experiments, the performance of the proposed method is compared with the XY and the WirelessXY routing algorithm. These simulations are conducted on a 4 * 4 mesh with two wireless links; namely, nodes 1 & 6 and the performance of routing algorithms is evaluated based on latency curves. It is assumed that data packets and learning packets have different lengths. The simulation time is set to 5000 ns and the buffer size is set to 4. We have also used three different traffic patterns, Transpose, Hotspot and Random, to display and compare results [36]. For the Hotspot case, 5% of the generated traffic by all cores have the same destination which can be chosen randomly. The destination of the other 95% of the generated packets by a core will be chosen randomly. Figure 4 shows the average latency as a function of the average data injection rate in random traffic.

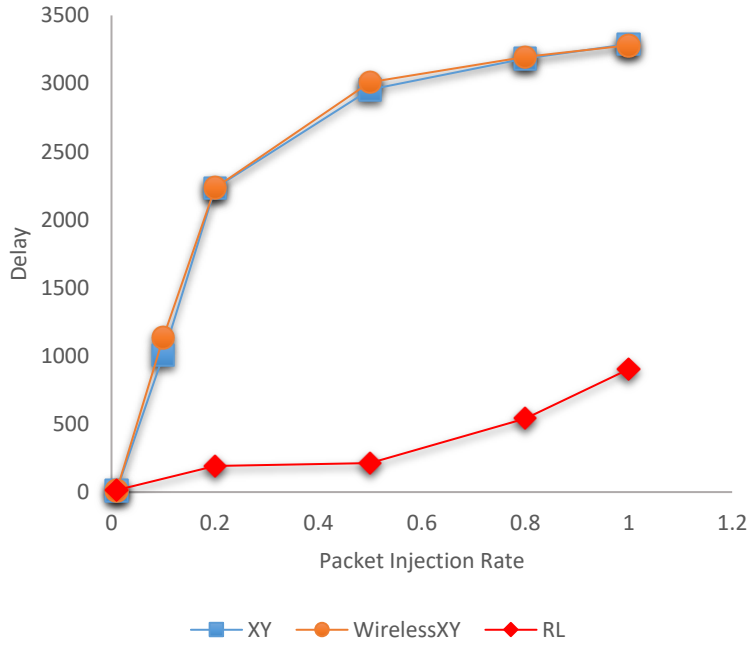

Figure 4. Comparison of delay for the proposed algorithm, XY and wirelessXY in random traffic.

The horizontal axis of the diagram shows how likely it is to be injected per clock for each core and the higher the injection rate, the higher the network load. In random traffic [32], each node with a random probability sends a packet to another node. The destination of different packets is determined using a uniform distribution randomly .As can be seen from the results, the proposed algorithm has resulted in less latency than other algorithms and in this case, the proposed algorithm has improved latency by at least up to 9%.

Figure 5 shows the average latency as a function of the average data injection rate in transpose traffic. In transpose traffic, a node (j, i) can only send packets to one node (i, j). As can be seen from the results, in this type of traffic, the proposed algorithm has led to less latency than other algorithms and in this case, the proposed algorithm has improved latency by at least 8%.

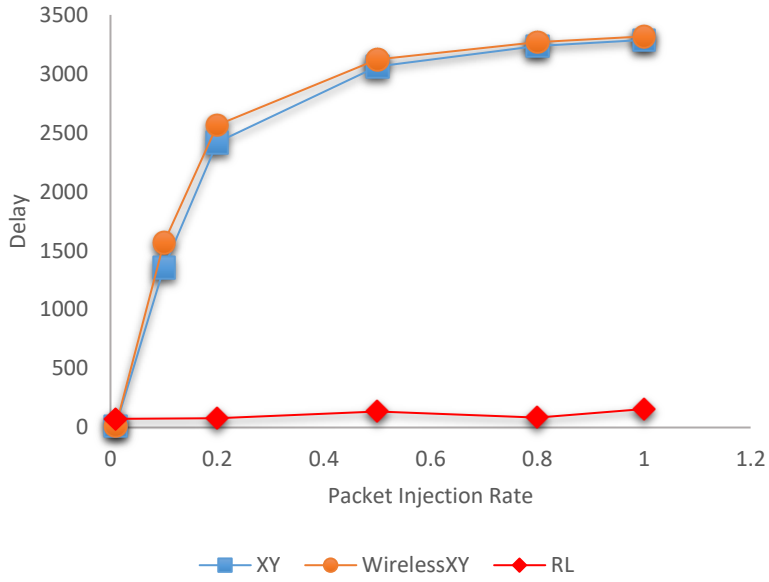

Figure 5. Comparison of delay for the proposed algorithm, XY and wirelessXY in transpose traffic.

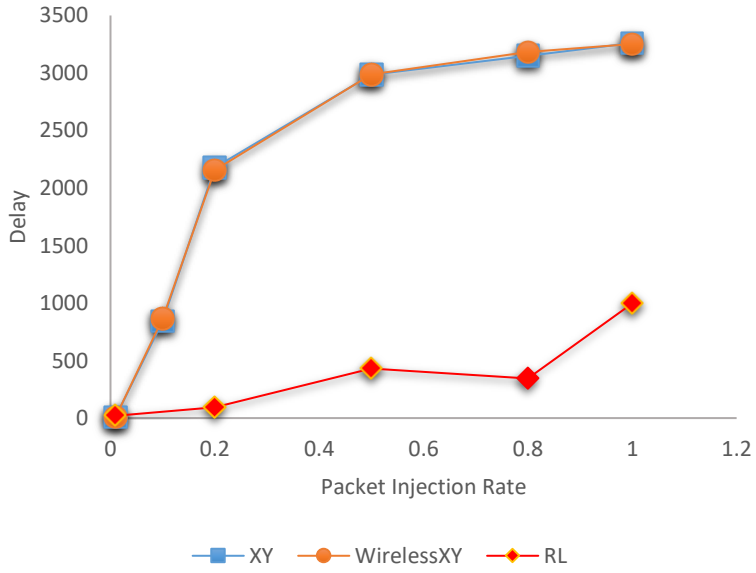

Figure 6. Comparison of delay for the proposed algorithm, XY and wirelessXY in hotspot traffic.

Figure 6 shows the average latency as a function of the average data injection rate in hotspot traffic. In hotspot traffic, there are one or more nodes selected as points that receive more traffic in addition to regular monotonous traffic. For this case, the proposed algorithm has led to less latency than other algorithms and in this case, the proposed algorithm has improved latency by at least 12%.

The results of the proposed algorithm in all cases in comparison with the WirelessXY and XY algorithms is better, since the proposed algorithm is adaptable to the congestion conditions; that is, if the packet encounters a crowded route on the way to the destination, it changes its route and does not wait for the route to be released, which causes it to be less delayed than in other algorithms.

To illustrate the movement of a packet in the path source node to the destination node, the following curves in Figure 7 are shown. These diagrams show the values of delay for the middle nodes in a 3*3 mesh where the source node is node zero and the destination node is node 5.

In this figure, the middle nodes; namely, node 1 and node 3 are examined. In addition, for each time interval, one graph is straight and in the same interval, the other graph is sloping. The sloping diagram shows that the packet is moving in this direction and the straight diagram shows that the moving packet does not exist in this path. Packet movement in both paths is performed according to the proposed

algorithm by examining the values of delay and how it works, considering that the values of delay are zero by default. The direction of the first neighbor node is selected as the direction of packet movement.

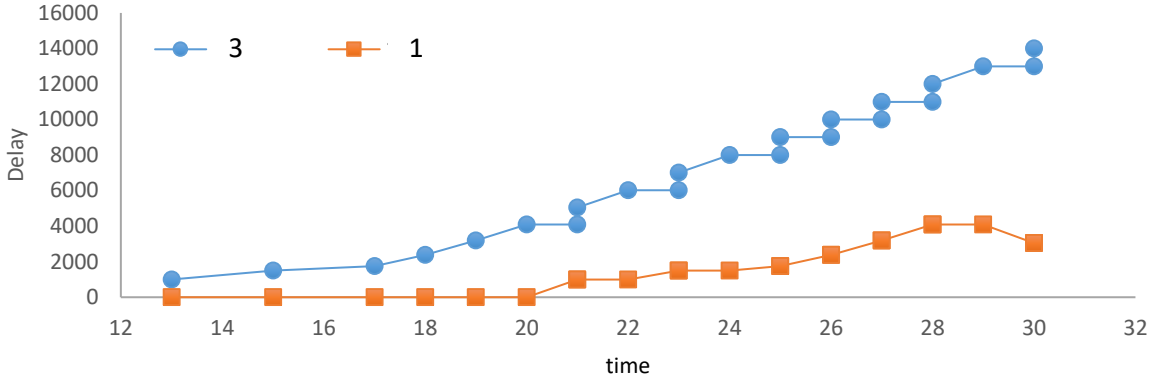

Figure 7. Graph of how the packets move at a learning rate of 0.5 for nodes 1 and 3.

Therefore, the packet starts moving in the path of node 3 for up to 20 ns. At this time, the proposed algorithm, after examining the amount of delay in the two nodes, realizes that the amount of latency in node 1 is less than in node 3, so it puts the direction of movement of the package in the path of node one. The delay is checked again in 21 ns and this time, the source node has placed the packet in the path of node 3 and the process of selecting a less crowded path continues. Finally, Figures 8 and 9 compare the throughput and power consumption of the WNoC for different routing algorithms; namely, RL-based, XY and WirelessXY.

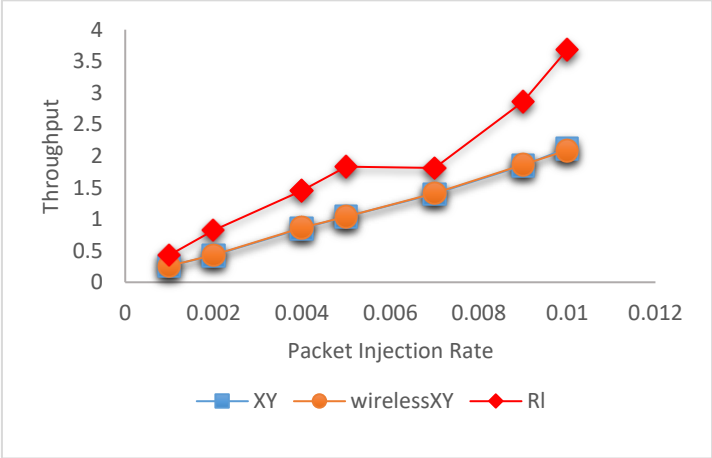

Figure 8. Comparison of throughput for the proposed algorithm, XY and WirelessXY in random traffic.

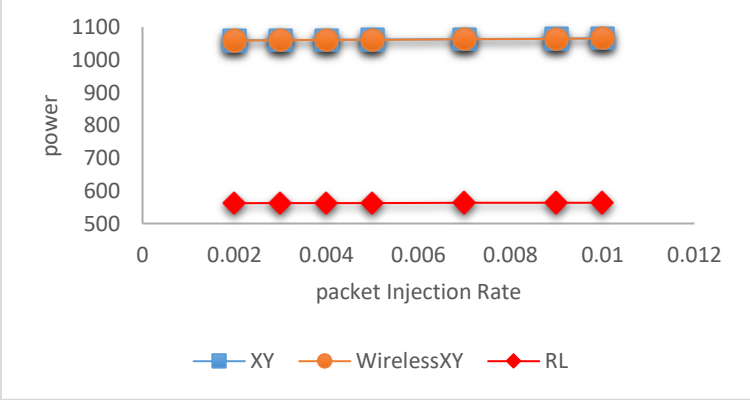

Figure 9. Comparison of power consumption for the proposed algorithm, XY and WirelessXY in random traffic.

5.1 Comparison with Previous Works

Figures 10 and 11 compare the proposed algorithm with reference [30]. The horizontal axis shows the injection rate of the package and the vertical axis shows the delay or time it takes the package to reach the destination. In these figures, a network with dimensions of 8*8 for two different traffics; namely, random traffic and hotspot is considered.

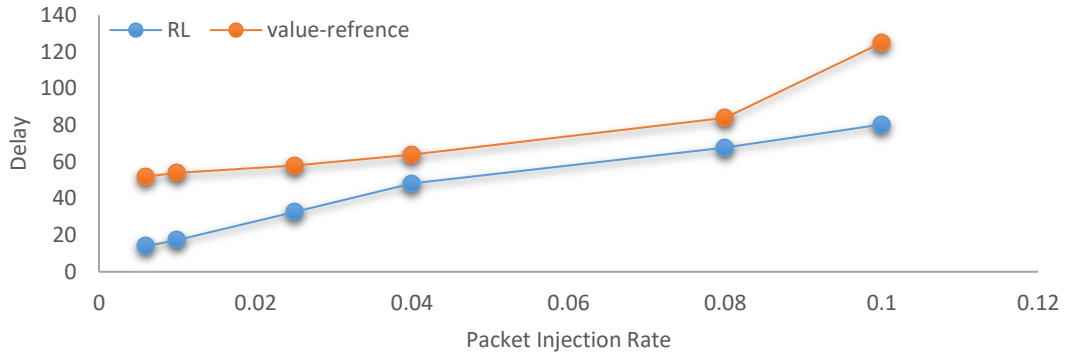

Figure 10. Comparison of delay for the proposed algorithm and the algorithm proposed in [30] for random traffic in an 8*8 network.

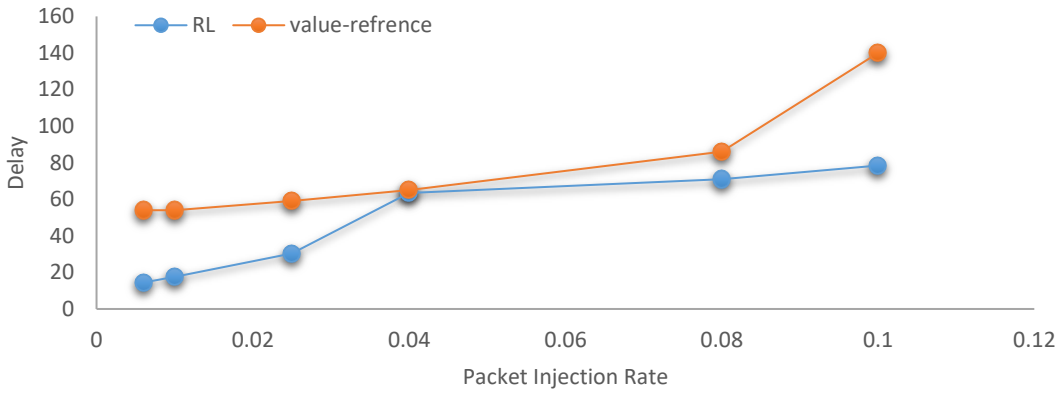

Figure 11. Comparison of delay for the proposed algorithm and the algorithm proposed in [30] for hotspot traffic in an 8*8 network.

As shown in Figures 10 and 11 for an 8*8 NoC, the delay evaluated using the proposed algorithm is always less for both traffic types.

The better results in the proposed algorithm are due to the use of wireless links. These links lead to a tendency for nodes to send their data (for ease of data transmission) *via* these links and this could significantly reduce congestion on the network and allow less delay to the destination.

6. CONCLUSION

In this paper, a routing algorithm based on reinforcement learning in a wireless chip network is proposed, which can make decisions based on congestion conditions in the network when transferring flits from the source node to the destination one. This algorithm uses the Q-learning method. In this method, there is a table for each node that shows the different states of packet movement of that node in the network and the values of this table are equivalent to the delay which is spent to get the packet from the source node to the destination node. This table is first filled with the initial value of zero and then updated by moving from one node to another node. When a packet comes to a node, the table of the node is considered. If the delay value is assumed to be positive, it will select the delay's lowest value and send the packet in that direction. In fact, learning in this algorithm is done by updating and changing the table

values of each node. To evaluate the algorithm, this method was compared with the XY and the WirelessXY algorithms in different types of traffic. The experimental results show that the reinforcement learning-based routing algorithm can improve the delay at least by around 8% for all traffic types.

The reinforcement learning algorithm has some disadvantages and limitations as compared to other learning algorithms, as in large networks on the chip where each node contains many data, the reinforcement learning algorithm cannot analyze these networks. Therefore, as future work, we intend to use deep reinforcement learning rather than reinforcement learning to develop our approach to develop more complex environments.

REFERENCES

[1] S. Kundu and S. Chattopadhyay, Network-on-Chip: The Next Generation of System-on-Chip Integration, Taylor & Francis, 2014.

[2] R. Venugopalan, S. Kumar Goel and Y.-H. Lee, "Network-on-Chip System and a Method of Generating the Same," U.S. Patent Application 16/879,567, Filed September 3, 2020.

[3] A. Ganguly, K. Chang, S. Deb, P. Pratim Pande, B. Belzer and C. Teuscher, "Scalable Hybrid Wireless Network-on-Chip Architectures for Multicore Systems," IEEE Transactions on Computers, vol. 60, no. 10, pp. 1485-1502, 2010.

[4] J. Flich, S. Rodrigo and J. Duato, "An Efficient Implementation of Distributed Routing Algorithms for NoCs," Proc. of the 2nd ACM/IEEE Int. Symposium on Networks-on-Chip (NOCs 2008), pp. 87-96, 2008.

[5] Z. Mogharrabi-Rad and E. Yaghoubi, "ADFT: An Adaptive, Distributed, Fault-tolerant Routing Algorithm for 3D Mesh-based Networks-on-Chip," International Journal of Internet Technology and Secured Transactions, vol. 10, no. 4, pp. 481-490, 2020.

[6] R. Bishnoi, "Hybrid Fault Tolerant Routing Algorithm in NoC," Perspectives in Science, vol. 8, pp. 586-588, 2016.

[7] S. Mubeen and S. Kumar, "Designing Efficient Source Routing for Mesh Topology Network on Chip Platforms," Proc. of the 13th IEEE Euromicro Conference on Digital System Design: Architectures, Methods and Tools, pp. 181-188, 2010.

[8] Z.-S. Chen, Y. Zhang, Z. Peng and J.-H. Jiang, "A Deterministic-path Routing Algorithm for Tolerating Many Faults on Wafer-level NoC," In IEEE Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1337-1342, Florence, Italy, 2019.

[9] T. A. Eltaras, W. Fornaciari and D. Zoni. "Partial Packet Forwarding to Improve Performance in Fully Adaptive Routing for Cache-coherent NoCs," Proc. of the 27th IEEE Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), pp. 33-40, Pavia, Italy, 2019.

[10] M. M. Rahaman, P. Ghosal and T. Subhra Das, "Latency, Throughput and Power Aware Adaptive NoC Routing on Orthogonal Convex Faulty Region," Journal of Circuits, Systems and Computers, vol. 28, no. 04, DOI: 10.1142/S0218126619500555, 2019.

[11] M. Sun, Q. Liu, B. Yan and X. Wang, "Minimally Buffered Router and Deflection Routing Algorithm for 3D Mesh NoC," Proc. of Recent Developments in Intell. Computing, Communication and Devices, Part of the Advances in Intelligent Systems and Computing Book Series, vol. 752, pp. 515-522, 2019.

[12] M. Schoeberl, L. Pezzarossa and J. Sparsø, "A Minimal Network Interface for a Simple Network-on-Chip," Proc. of the International Conference on Architecture of Computing Systems (ARCS 2019), Part of the Lecture Notes in Computer Science Book Series, vol. 11479, pp. 295-307, 2019.

[13] Song, Yang and Bill Lin. "Uniform Minimal First: Latency Reduction in Throughput-optimal Oblivious Routing for Mesh-based Networks-on-Chip." IEEE Embedded Systems Letters, vol. 11, no. 3, pp. 81-84, 2019.

[14] L. Wang, X. Wang, H.-F. Leung and T. Mak, "A Non-minimal Routing Algorithm for Aging Mitigation in 2D-mesh NoCs," IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, vol. 38, no. 7, pp. 1373-1377, 2018.

[15] K.-C. Chen, "Game-based Thermal Delay-aware Adaptive Routing (GTDAR) for Temperature-aware 3D Network-on-Chip Systems," IEEE Transactions on Parallel and Distributed Systems, vol. 29, no. 9, pp. 2018-2032, 2018.

[16] W. Zhang and Y. Ye, "A Table-free Approximate Q-learning Based Thermal-aware Adaptive Routing for Optical NoCs," IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, vol. 40, no. 1, pp. 199-203, 2020.

[17] Y. Ye, W. Zhang and W. Liu, "Thermal-aware Design and Simulation Approach for Optical NoCs," IEEE Trans. on Computer-aided Design of Integrated Circuits and Sys., vol. 39, no. 10, pp. 2384 – 2395, 2019.

[18] N. Shahabinejad and H. Beitollahi, "Q-thermal: A Q-learning Based Thermal-aware Routing Algorithm for 3D Network On-Chips," IEEE Transactions on Components, Packaging and Manufacturing Technology, vol. 10, no. 9, pp. 1482 – 1490, 2020.

[19] T. H. Vu, O. M. Ikechukwu and A. Ben Abdallah, "Fault-tolerant Spike Routing Algorithm and Architecture for Three Dimensional NoC-based Neuromorphic Systems," IEEE Access, vol. 7, pp. 90436-90452, DOI: 10.1109/ACCESS.2019.2925085, 2019.

[20] Y.-Y. Chen, E.-J. Chang, H.-K. Hsin, K.-C. Chen and A.-Y. Wu, "Path Diversity-aware Fault-tolerant Routing Algorithm for Network-on-Chip Systems," IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 3, pp. 838-849, 2016.

[21] J. Zhou, H. Li, T. Wang and X. Li, "LOFT: A Low-overhead Fault-tolerant Routing Scheme for 3D NoCs," Integration, vol. 52, pp. 41-50, 2016.

[22] Y. Kurokaw and M. Fukushi, "Passage of Faulty Nodes: A Novel Approach for Fault-tolerant Routing on NoCs," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. 102, no. 12, pp. 1702-1710, 2019.

[23] R. S. Sutton, "Introduction: The Challenge of Reinforcement Learning," Proc. of Reinforcement Learning, Part of the Springer International Series in Engineering and Computer Science Book Series, vol. 173, pp. 1-3, Springer, Boston, MA, USA, 1992.

[24] S. Mahadevan, "Average Reward Reinforcement Learning: Foundations, Algorithms and Empirical Results," Machine Learning, vol. 22, no. 1-3, pp. 159-195, 1996.

[25] C. J. Watkins and P. Dayan, "Q-learning," Machine learning, vol. 8, no. 3-4, pp. 279– 292, 1992.

[26] F. Farahnakian, M. Ebrahimi, M. Daneshtalab, P. Liljeberg and J. Plosila, "Q-learning Based Congestion-aware Routing Algorithm for on-chip Network," Proc. of the 2nd IEEE International Conference on Networked Embedded Systems for Enterprise Applications, pp. 1-7, Perth, WA, Australia, 2011.

[27] F. Farahnakian, M. Ebrahimi, M. Daneshtalab, J. Plosila and P. Liljeberg, "Adaptive Reinforcement Learning Method for Networks-on-Chip," Proc. of the IEEE International Conference on Embedded Computer Systems (SAMOS), pp. 236-243, Samos, Greece, 2012.

[28] S.-C. Kao, C.-H. Huck Yang, P.-Y. Chen, X. Ma and T. Krishna, "Reinforcement Learning Based Interconnection Routing for Adaptive Traffic Optimization," Proceedings of the 13th IEEE/ACM International Symposium on Networks-on-Chip, pp. 1-2, DOI: 10.1145/3313231.3352369, 2019.

[29] F. Farahnakian, M. Ebrahimi, M. Daneshtalab, J. Plosila and P. Liljeberg, "Optimized Q-learning Model for Distributing Traffic in on-chip Networks," Proc. of the 3rd IEEE International Conference on Networked Embedded Systems for Every Application (NESEA), pp. 1-8, Liverpool, UK, 2012.

[30] F. Farahnakian, M. Ebrahimi, M. Daneshtalab, P. Liljeberg and J. Plosila, "Bi-LCQ: A Low-weight Clustering-based Q-learning Approach for NoCs," Microprocessors and Microsystems, vol. 38, no. 1, pp. 64-75, 2014.

[31] S. Sakamoto, R. Obukata, T. Oda, L. Barolli, M. Ikeda and A. Barolli, "Performance Analysis of Two Wireless Mesh Network Architectures by WMN-SA and WMN-TS Simulation Systems," Journal of High Speed Networks, vol. 23, no. 4, pp. 311-322, 2017.

[32] R. Mohammadi and H. Boroumand Noghabi, "SAT: Simulated Annealing and Tabu Search Based Routing Algorithm for Wireless Sensor Networks," International Journal of Computer Networks and Communications Security, vol. 4, no. 10, Paper ID: 286, 2016.

[33] A. Norollah, D. Derafshi, H. Beitollahi and A. Patooghy, "PAT-Noxim: A Precise Power & Thermal Cycle-accurate NoC Simulator," Proc. of the 31st IEEE International System-on-Chip Conference (SOCC), pp. 163-168, Arlington, VA, USA, 2018.

[34] V. A. M. Catania, S. Monteleone, M. Palesi and D. Patti, "Noxim: An Open, Extensible and Cycle-accurate Network on Chip Simulator," Proc. of the 26th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP), pp. 162-163, Toronto, Canada, 2015.

[35] S. Khan, S. Anjum, U. Ali Gulzari and F. Sill Torres, "Comparative Analysis of Network-on-Chip Simulation Tools," IET Computers & Digital Techniques, vol. 12, no. 1, pp. 30-38, 2017.

[36] I. L. P. Pires, M. A. Z. Alves and L. C. P. Albini, "Trace-driven and Processing Time Extensions for Noxim Simulator," Design Automation for Embedded Systems, vol. 23, no. 1-2, pp. 41-55, 2019.

[37] M.-H. Tang and L. I. N. Jing, "A Quantitative Study on NoC Traffic Scenarios," DEStech Transactions on Computer Science and Engineering, DOI: 10.12783/dtcse/iece2018/26648, 2018.

[38] S. Jog, Z. Liu, A. Franques, V. Fernando, H. Hassanieh, S. Abadal and J. Torrellas, "Millimeter Wave Wireless Network on Chip Using Deep Reinforcement Learning," ACM Conf. on Special Interest Group on Data Communication (SIGCOMM'20 posters/demos), p. 1-3, DOI 10.1145/3405837.3411396, 2020.

[39] Z. Li and Y. Li, "Use Deep Reinforcement Learning for NoC Design," [Online], Available: https://aml-2020.aminer.cn/proposal/ZhiyaoLi_YiweiLi.pdf, 2020.

[40] T.-R. Lin, D. Penney, M. Pedram and L. Chen, "A Deep Reinforcement Learning Framework for Architectural Exploration: A Routerless NoC Case Study," Proc. of the IEEE International Symposium on High Performance Computer Architecture (HPCA), pp. 99-110, San Diego, CA, USA, 2020.

[41] S. Deb and H. Kumar Mondal, "Wireless Network-on-Chip: A New ERA in Multi-core Chip Design," Proc. of the 25th IEEE Int. Symposium on Rapid System Prototyping, pp. 59-64, New Delhi, India, 2014.

[42] A. Ganguly, K. Chang, S. Deb, P. P. Pande, B. Belzer and C. Teuscher, "Scalable Hybrid Wireless Network-on-Chip Architectures for Multicore Systems," IEEE Transactions on Computers, vol. 60, no. 10, pp. 1485–1502, 2010.

[43] E. Tahanian, M. Rezvani and M. Fateh, "A Novel Wireless Network-on-Chip Architecture for Multicore Systems," Proc. of the 26th IEEE International Computer Conference, Computer Society of Iran (CSICC), pp. 1-8, Tehran, Iran, 2021.

[44] C. JCH Watkins and P. Dayan, "Q-learning," Machine Learning, vol. 8, no. 3-4, pp. 279-292, 1992.

[45] Md S. Shamim, N. Mansoor, R. Singh Narde, V. Kothandapani, A. Ganguly and J. Venkataraman, "A Wireless Interconnection Framework for Seamless Inter and Intra-chip Communication in Multichip Systems," IEEE Transactions on Computers, vol. 66, no. 3, pp. 389-402, 2016.

[46] K. Gola and B. Gupta "An Energy-efficient Quality of Service (QoS) Parameter-based Void Avoidance Routing Technique for Underwater Sensor Networks," Jordanian Journal of Computers and Information Technology (JJCIT), vol. 05, no. 03, pp 244-261, December 2019.

ملخص البحث:

يمكن استخدام الشّبكات اللّاسلكية على الدارات المتكاملة بديلاً عن تقنية القضبان في الدارات المتكاملة عالية المحاور، وفيها تتمّ الاستعاضة عن المسارات متعددة القفزات بين المحاور البعيدة بعضها عن بعض برابط لاسلكي ذي قفزة واحدة. إنّ السبب الرئيسي لاستخدام الاتصال اللاسلكي هو التقليل من التأخير واستهلاك الطاقة. ونظراً لتحديد الموارد، فإنّ أداء الشّبكات اللاسلكية على الدارات المتكاملة حسّاس لخوارزمية التّسيير. ففي الوقت الذي تعمل فيه خوارزمية التّسيير الملائمة على خفض التّأخير، فإنّ عليها تجنب الانسداد.

في هذه الورقة، نقترح خوارزمية تسيير مبتكراة باستخدام "تعلُّم كيو"، وهو إحدى طرق تعلم التعزيز لموازنة المرور في الشبكات اللاسلكية على الدارات المتكاملة. فباستخدام هذه الخوارزمية، يمكن للعُقد أن تتخذ القرارات بناءً على ظروف الازدحام في الشبكة عند نقل البيانات من عُقدة المصدر الى عُقدة الهدف. وقد بينت نتائج المحاكاة أن استخدام تعلّم التعزيز بواسطة الخوارزمية المقترحة لتسيير الحُزم يحسّن أداء الشّبكة بشكلٍ ملحوظ؛ إذ تمّ تحسين أداء النّظام بنسبة لا تقل عن 8% مقارنة بأداء الأنظمة المشابهة في أعمال سابقة.