Journal of AI and Data Mining
Vol 8, No 3, 2020, 409-415.
DOI: 10.22044/JADM.2020.8964.2034

A Routing-Aware Simulated Annealing-based Placement Method in Wireless Network on Chips

A. Tajary* and E. Tahanian

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

Received 28 September 2019; Revised 07 January 2020; Accepted 31 March 2020
**Corresponding author: tajary@shahroodut.ac.ir (A. Tajary).*

Abstract
Wireless network on chip (WiNoC) is one of the promising on-chip interconnection networks for on-chip system architectures. In addition to wired links, these architectures also use wireless links. Using these wireless links makes packets reach destination nodes faster and with a less power consumption. These wireless links are provided by wireless interfaces in wireless routers. The WiNoC architectures differ in the position of the wireless routers and how they interact with other routers. Thus the placement of wireless interfaces is an important step in designing the WiNoC architectures. In this paper, we propose a simulated annealing placement method, which considers the routing algorithm as a factor in designing cost function. In order to evaluate the proposed method, Noxim, which is a cycle-accurate network-on-chip simulator, is used. The simulation results show that the proposed method can reduce flit latency by up to 24.6% with about a 0.2% increase in the power consumption.

Keywords: *Simulated Annealing, Wireless Network on Chip, Placement.*

1. Introduction

With the advent of multi-core processors, the interconnection of processor components plays a key role in determining the overall performance of the system. Unlike the bus-based processor architectures, these new architectures use network-on-chip (NoC) as interconnections between cores and memory sub-systems [1, 2]. The traditional NoC architectures like mesh are not efficient and have pose new problems due to the scalability and requirement for a high bandwidth. Wireless NoC (WiNoC) is one of the promising interconnection approaches that have been proposed to overcome these problems [3-6].

In the WiNoC architectures, in addition to the wired links, the wireless links are used to connect the nodes **[**2**]**. These new links are utilized as a shortcut between distant nodes. The use of these links will make long communications into short ones, so the packets can reach their destination faster [**7**]. These wireless links are provided by the wireless nodes and their antennas. The WiNoC architectures differ in the position of the wireless nodes and how they communicate with each other and how they interact with ordinary nodes **[**8**]**. Thus locating the wireless nodes, called placement, is an important step in designing the WiNoC architectures [9, 10].

The placement of the wireless nodes in the conventional WiNoC architectures is performed using different methods like simulated annealing, genetic algorithm, and Tabu search in order to determine the position of the wireless nodes [10-12]. The objective functions of these methods are set in such a way that more traffic passes through the wireless links. The more traffic passes through the wireless links, the faster the packets arrive to their destination with less power consumption [8, 2]. For routing a packet, if the wireless path length that is counted in terms of hops is less than the wired path length, the wireless path is selected. However, evaluations have shown that using this routing method will increase competition for access to wireless links and create hotspots at wireless nodes, resulting in an increased packet latency and power consumption [2, 13, 8].

In order to overcome the congestion problem in wireless nodes, Wang *et al.* **[**13**]** have proposed a routing method that considers an additional delay

for wireless links. This extra delay makes the wireless links to be used less than before, which is in contrast to the objective of the placement methods. In other words, the placement algorithm wants to use the wireless links as much as possible but the routing algorithm tries to reduce it. Considering this routing method, the position of the wireless nodes can be optimized to utilize the wireless links.

In this paper, we propose a routing aware placement algorithm that uses a simulated annealing method to locate the position of wireless nodes. The Noxim [14] NoC simulator is used to evaluate the proposed method. The simulation results show that the proposed method can reduce flit latency to 24.6% and improve the throughput to 10.1% with only 0.2% more power consumption compared with the related methods.

The remainder of this paper is organized as what follows. In Section 2, we review the related works. In Section 3, the proposed method is described. The simulation framework and the experimental results are presented in Section 4. Finally, Section 5 concludes the paper.

2. Related works

In the traditional NoCs, the long-distance packets must pass through multiple routers to reach their destination. This will reduce the NoC performance and will increase its power consumption. Using wireless links will make long paths shorter. Therefore, the packets reach their destination with a less latency. In addition, the use of wireless links allows for the creation of hierarchical and complex architectures. For example, in [7], WCube, which is a recursive wireless interconnection structure, is presented to address the scalability problems in NoC. It is a hierarchical structure containing mesh-based wired connections at the bottom layer and wireless connections at a high level and can support 1000 nodes. In [13], a WiNoC design for multicore platforms is proposed. It is a mesh-based NoC partitioned into four sub-nets. Each sub-net has a wireless router that connects to wireless routers of other sub-nets. In these two structures, the positions of the wireless routers are deterministic and specified by their design.

In [9], a scalable small-world WiNoC architecture for multicore systems has been presented. In small-world networks, the average path length is very short, making them interesting for an effective communication with low resources [15]. In this work, the simulated annealing algorithm was used to optimize the placement of wireless links. The objective of the simulated annealing method was to minimize the average hop number between each two NoC nodes.

A hierarchical small-world NoC with on-chip millimeter (mm)-wave wireless channels has been proposed in [10]. It has a novel energy-efficient transceiver design, an efficient data routing, and an optimum placement of the wireless hubs. In this architecture, a standard canonical evolutionary algorithm with tournament selection is employed for placement of wireless interfaces. The performance evaluation was performed with an NoC simulator, which reports throughput, latency, and power consumption of the architecture.

In [16] and [17], the placement of wireless interfaces has been modeled as a Facility Location problem [18], which is an NP-hard problem. In this problem, there are some customers and some potential centers for placing the facilities. A solution to this problem tries to minimize the costs of opening the facilities and to reduce the average distance of customers to facilities. A cost function based on this idea is defined and the optimization of this cost function is done using simulated annealing and genetic algorithm.

In [11], a novel hierarchical architecture for Wireless NoC has been proposed. Sub-nets are the first level of the hierarchy and the connection of the sub-nets makes the second level of the hierarchy. The placement of the wireless interfaces in each sub-net is modeled as a n-Queen problem such that each wireless interface will be a queen and the positions of the queens will be the positions of the wireless interfaces. In this paper, two methods are proposed to solve the n-queen problem: simulated annealing and genetic algorithm. The problem is solved for a 16-node NoC with 4 wireless interfaces.

3. Proposed method

In this paper, we consider the small world WiNoC as the architecture of network on chip. This small world WiNoC can be considered as the final architecture or as the sub-net of a hierarchical architecture. In this architecture, the nodes are arranged in a 2D mesh network. Each node is connected to a processing element and a router. The packets are produced and consumed in the processing elements. Routers use the source and destination data to route each packet to its destination. Some routers are equipped with a wireless interface. Routers with the wireless interface can send and receive packets wirelessly.

Using wireless links will shorten the path that a packet must pass to reach its destination. Since many packets use wireless links instead of the conventional wired links, the locations of the wireless interfaces are crucial in the performance

and power consumption of the network. Therefore, placement algorithms, which determine the location of the wireless nodes, try to use the maximum capacity of wireless links. On the other hand, sending all packets to the wireless interfaces creates hotspot situations in those nodes. To avoid this situation, Wang *et al*. **[**13**]**, have proposed a routing algorithm that incurs an additional cost for the use of the wireless links, and therefore, does not send all packets to the wireless interfaces. As it can be seen, the purpose of the placement algorithm contrasts with the purpose of the routing algorithm. Thus we present a routing-aware placement method that takes into account the routing effect at the time of placement and improves the performance and power consumption of WiNoC.

It is important to note that the proposed method is static and workload-independent. It uses simulated annealing to find the positions of the wireless routers in the network on chip. The simulated annealing method is widely used to solve problems in different subjects [19, 20]. It is a search-based optimization technique that tries to find the global optimum of a target function called the cost function. It holds just one solution and repeatedly tries to find a new solution from the previous one. The pseudo-code of the simulated annealing is shown in algorithm 1.

Algorithm 1. The proposed simulated annealing method.

Parameters:
T_0: The initial temperature
α: The temperature reduction rate
T_{min}: The minimum temperature
NoC_Dims: The dimensions of NoC
WI_count: The number of wireless interfaces
δ: Additional overhead for wireless paths
WI_pos: The positions of wireless interfaces

Inputs: T_0, α, T_{min}, *NoC_Dims, WI_count,* δ
Outputs: *WI_pos*

```
1: T = T0
2: WI_pos = Initial_Placement(NoC_Dims, WI_count)
3: while (T > T_min)
4:    new_WI_pos = Perturb(WI_pos)
5:    Δcost = Cost(new_WI_pos, δ) – Cost (WI_pos, δ)
6:    if (Δcost < 0)
7:       WI_pos = new_WI_pos
8:    else
9:       r = Random_Value(0,1)
10:      if (r < e^(-Δcost/T))
11:         WI_pos = new_WI_pos
12:   T = α × T
```

Initially, using the *Initial_Placement* function, an arbitrary solution is generated. After that, using the *Perturb* function, a new solution is made from that solution. If the cost of the new solution is less than the cost of the previous one, the new solution replaces the previous one; otherwise, the new solution replaces the previous one with a particular probability. This specific probability depends on the current temperature and the difference between the costs. At first, the temperature is high but as the algorithm goes forward, the temperature decreases. The higher the temperature or the lower the cost difference, the more likely the new solution will be accepted. The probability of acceptance is calculated using the exponential function shown in line 10 of algorithm 1. After each iteration, the temperature decreases at the rate of α. The loop continues until the temperature drops to below a threshold value (T_{min}).

There are six parameters that precisely define this simulated annealing algorithm: 1) T_0: the initial temperature, 2) T_{min}: the minimum temperature, 3) α: the rate of temperature drop, 4) the *Initial_Placement* function that performs the initial placement, 5) the *Perturb* function that produces a new solution from the last solution, and 6) the *Cost* function. The first three parameters are numerical and their values are determined after multiple runs of the algorithm. The next three parameters are of function type, and the details of the operation of each one are given below.

The first function, *Initial_Placement*, produces an arbitrary solution. In our case, a solution is a data structure that specifies the locations of wireless interfaces in the target NoC. Thus this function should know the dimensions of the target NoC and the number of wireless interfaces. It randomly selects the router locations from NoC and marks them as wireless interfaces as much as the number of wireless interfaces. The second function, *Perturb*, produces a new solution from the current solution. As its name implies, it should create perturbation in the current solution. In our implementation, the perturbation is to randomly select a wireless interface and change its location.

The *Cost* function is the third and the most important function in this algorithm. This function shows how good or bad a solution is. The lower the value of this function is, the better the solution is. Moreover, the effect of the routing algorithm is considered in this function. In this function, the average number of hops for a packet to deliver from its source to its destination is computed in two cases: 1) when no wireless interfaces exist in NoC (H_B in (1)), 2) when wireless interfaces exist in NoC (H_W in (2)).

$$H_B = \frac{\sum_{s=1}^{n}\sum_{d=1}^{n}(Wired\ \ path\ \ length\ \ from\ \ s\ \ to\ \ d)}{n^2} \quad (1)$$

$$H_W = \frac{\sum_{s=1}^{n}\sum_{d=1}^{n}(Smallest\ \ path\ \ length\ \ from\ \ s\ \ to\ \ d)}{n^2} \quad (2)$$

$$Cost = \frac{H_W}{H_B} \quad (3)$$

The cost function is defined as the ratio of H_W to H_B, as shown in (3). When there are no wireless interfaces in NoC, the number of hops that a packet passes is equal to the distance from its source to its destination. Hence, H_B is computed as the average distance of any two routers in NoC, as shown in (1). When there are some wireless interfaces in NoC, based on the routing algorithm, a packet may or may not use wireless links. To clarify, the routing algorithm computes the number of hops that a packet should pass, with and without wireless links. Then it adds δ to the number of wireless hops. If the number of wireless hops is less than or equal to the number of wired hops, it uses the path containing the wireless links. Therefore, H_W is computed as the average number of hops that a packet must pass to reach its destination, considering the routing algorithm, as shown in (2). Since simulated annealing is an optimization method, its main purpose is to optimize the cost function. In each repetition of the algorithm, the value of the cost function changes. Since there is randomness (lines 9 and 10 of algorithm 1) in this method, the algorithm has to be run for multiple times to obtain an acceptable solution. In addition, as the best solution may be lost during execution, it is usually stored in a variable. Figure 1 shows the cost function of the solution in each repetition of the algorithm for different values for the initial temperature (T_0). As it can be seen in this figure, a low initial temperature makes the algorithm to finish rapidly. Therefore, it cannot find a good solution.

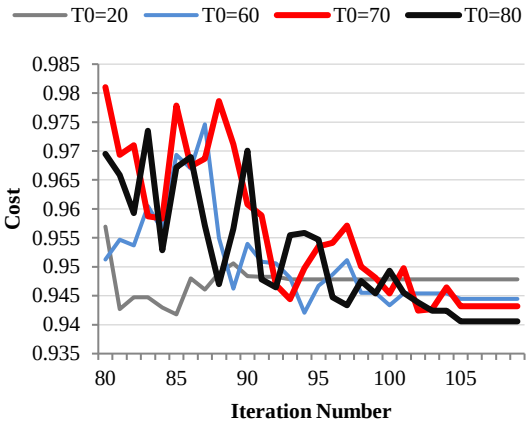

Figure 1. Changing the value of the *Cost* function during iterations of the algorithm for various values of T_0.

The effect of minimum temperature on the cost of the solution is shown in figure 2. High values for T_{min} will cause the algorithm to finish at high temperatures, which yields non-optimum solutions. The value of α should be between 0 and 1. Higher values for α make the temperature gradually get lower. As a result, the number of iterations of the algorithm increases. Figure 3 shows the number of iterations for different values of α.

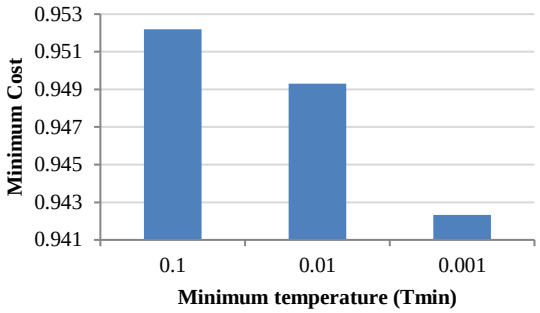

Figure 2. Effect of minimum temperature (T_{min}) on the minimum cost of solutions.

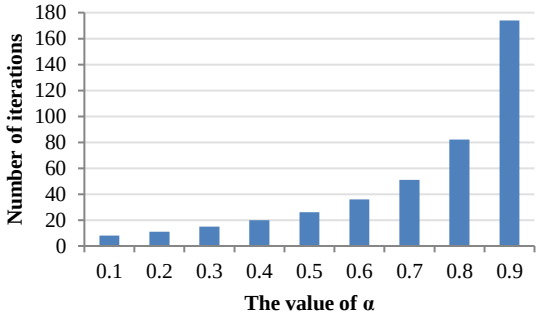

Figure 3. Effect of α on the number of iterations of the algorithm.

The effect of algorithm 1 on placement of three wireless nodes on a 16 node WiNoC is shown in figure 4.

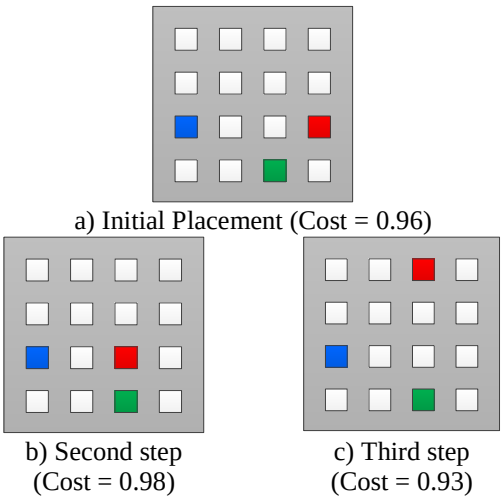

Figure 4. Three steps of algorithm 1 on a 16 node WiNoC.

In this figure, the red, blue, and green boxes are the positions of the three wireless nodes. Figure 4a shows the initial placement of these three nodes. As it can be seen, the positions are not balanced and the *cost* of this situation is 0.96. Figure 4b shows a

perturbation of this situation, where the position of the red wireless node is changed. The *cost* of this situation is 0.98, which is higher than the cost of the initial placement. In this case, the acceptance of this placement is dependent on the random value *r* (line 9 in algorithm 1). Figure 4c shows another placement, in which the position of the red wireless node is changed. The *cost* of this placement is 0.93, which is lower than the cost of the initial placement. Therefore, this placement replaces the previous one.

4. Evaluation

We used the Noxim NoC simulator to accurately evaluate the proposed method. Noxim is a cycle-accurate NoC simulator that can be programmed and configured to support different NoC architectures and algorithms **[**14**]**. Moreover, it reports the throughput, latency, and power consumption for a given configuration. In our case, we converted the output of the simulated annealing method to the inputs of Noxim, and then we performed the simulations, and finally, the outputs of the simulations were reported in the form of figures in this section. The simulation parameters of Noxim are shown in table 1.

Table 1. Simulation parameters of Noxim.

Parameter	Value
Dimensions	8 × 8
Packet Size	3 to 6 flits
Wired buffer size	4 flits
Wireless buffer size	4 flits
Number of virtual channels	2
Clock cycle	1 ns
Traffic distribution	Uniform random

4.1. Effect of δ on latency, throughput, and power consumption

The value of δ has a significant effect on the NoC performance. Low values for δ allow for more traffic to be sent through the wireless links, resulting in hotspots in the wireless interfaces, and high values for δ allow for less traffic to pass through the wireless links, which yields less wireless utilization. In order to examine the effect of δ on the NoC performance, various simulations with different values of δ were performed. In each simulation, three values for the packet injection rate (*PIR*) were considered: 0.001, 0.01, and 0.05. The average flit latency when there are 6 wireless links for different values of δ is shown in figure 5.

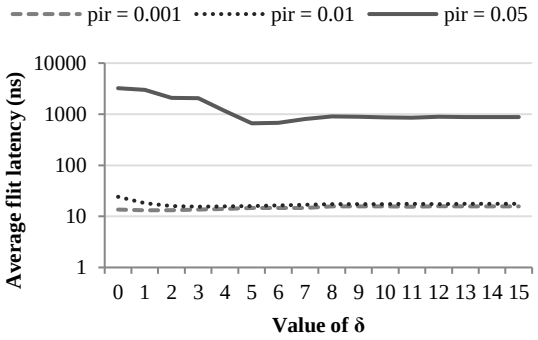

Figure 5. Effect of δ on the average flit delay for different values of *PIR* (6 wireless interfaces).

As shown in this figure, when the packet injection rate is low, the δ value does not affect the flit latency. In this case, no hotspot will be created in wireless interfaces. On the other hand, as the rate of packet injection increases, the δ value becomes an important factor in flit latency. For example, in the packet injection rate of 0.05, the flit latency when δ is five is 20% of the latency that occurs in the δ of zero. It is also important to note that the *y* axis in figure 5 is logarithmic. As this figure shows, increasing *PIR* saturates the network, and therefore, the flit latency increases dramatically.

Figure 6 shows the same results when there are 8 wireless interfaces. As shown in this figure, having more wireless interfaces results in a less flit latency. For example, consider the average flit latency when *PIR* is 0.05 and the value of δ is 5. The average flit latency when the number of wireless interfaces is 8 is 25.9% less than the average flit latency when the number of wireless interfaces is 6. This happens because having more wireless interfaces will allow for more packets to use wireless links and reach their destination faster.

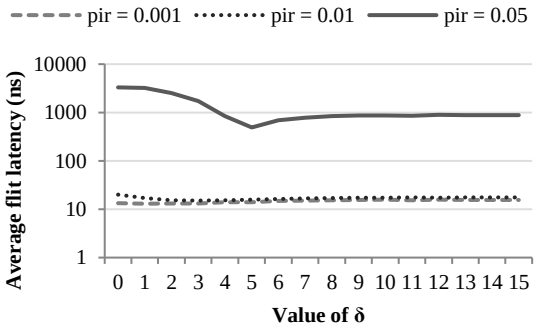

Figure 6. Effect of δ on the average flit delay for different values of *PIR* (8 wireless interfaces).

The throughput of the architecture in terms of flit per core per cycle for various values of δ is shown in figure 7. In these simulations, 6 wireless nodes are placed in NoC. As shown in this figure, when

the packet injection rate is low, the throughput is not affected by the value of δ. This means that all flits arrive at their destinations without waiting in the middle routers. However, as the packet injection rate increases, the value of δ shows its effect. For example, in the packet injection rate of 0.05, there is a peak throughput value in the δ value of 5. In fact, this figure shows that when *PIR* is 0.05 and the δ value is less than 4, using wireless interfaces, degradation of the throughput occurs. This happens because of the congestion in the wireless routers.

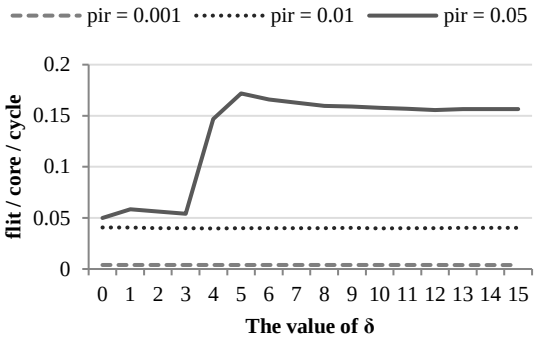

Figure 7. Effect of δ on the average throughput for different values of *pir*.

The power consumption of NoC with 6 wireless interfaces for various δ values is shown in figure 8. As shown in this figure, when *PIR* is low, the power consumption of the method does not change significantly and remains almost constant for different values of δ. However, as the packet injection rate increases, the power consumption increases, and there is a peak value for the δ value of 5. The increase in power consumption in the δ value of 5 is due to the increase in the throughput in this δ value.

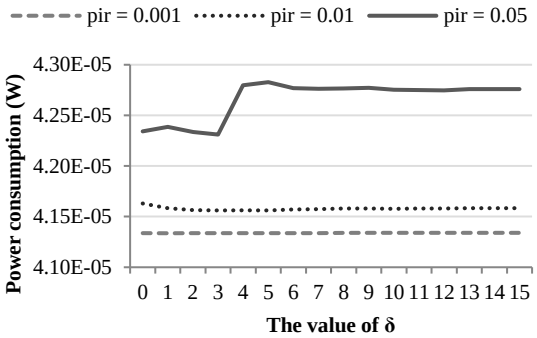

Figure 8. Effect of δ on power consumption for different values of *pir*.

4.2. Comparison with related works

We compared the proposed method with the methods described in **[**11**]** and **[**9**]**. The outputs of these methods were extracted and converted to the inputs of Noxim. The simulations were performed on Noxim with the same configuration for all methods (except for the position of the wireless node). Table 2 shows the simulation results of these methods for 8 wireless nodes and a packet injection rate of 0.1. As shown in this table, compared to n-Queen **[**11**]**, the proposed method had a 24.6% better latency, 10.1% better throughput with 0.2% overhead in the power consumption. It is also important to note that the number of wireless interfaces is not important in our method; the method in **[**11**]** tries to solve the n-Queen problem, which forces that the number of wireless modules should be equal to the number of rows or columns of NoC. In addition, the proposed method has a 1.3% better latency and 7.2% better throughput than the method presented in **[**9**]** with less than 0.002% overhead in the power consumption.

Table 2. Comparison of the proposed method with the related works.

	Parameters		
	Latency (nS)	**Throughput**	**Power (W)**
Baseline SA [9]	497.808	0.167044	4.2864e-05
n-Queen [11]	651.679	0.162678	4.2778e-05
The proposed method	**491.339**	**0.179105**	4.2865e-05

5. Conclusions

In this paper, we proposed a routing-aware placement method for wireless nodes in network on chips. The method was implemented as a simulated annealing algorithm. The method supports having an arbitrary number of wireless nodes in an NoC with an arbitrary size. The Noxim NoC simulator was used to evaluate the proposed method. The simulation results show that the proposed method reduces the latency by up to 24.6% over the existing methods with a 0.2% overhead in power consumption.

References

[1] Pande, P., Grecu, C., Jones, M., Ivanov, A., & Saleh, R. (2005). Performance evaluation and design trade-offs for network-on-chip interconnect architectures. IEEE Transactions on Computers, vol. 54, no. 8, pp. 1025-1040.

[2] Wang, S. & Jin, T. (2014). Wireless network-on-chip: a survey. The Journal of Engineering, vol. 2014, no. 3, pp. 98-104.

[3] Choi, W. *et al*. (2018). On-Chip Communication Network for Efficient Training of Deep Convolutional Networks on Heterogeneous Manycore Systems. IEEE Transactions on Computers, vol. 67, no. 5, pp. 672-686.

[4] Abadal, S., Torrellas, J., Alarcón, E., & Cabellos-Aparicio, A. (2018). OrthoNoC: A Broadcast-Oriented Dual-Plane Wireless Network-on-Chip Architecture. IEEE Transactions on Parallel and Distributed Systems, vol. 29, no. 3, pp. 628-641.

[5] Narde, R. S., Venkataraman, J., Ganguly, A., & Puchades, I. (2019). Intra- and Inter-Chip Transmission of Millimeter-Wave Interconnects in NoC-Based Multi-Chip Systems. IEEE Access, vol. 7, pp. 112200-112215.

[6] Karkar, A. *et al*. (2018). Network-on-Chip Multicast Architectures Using Hybrid Wire and Surface-Wave Interconnects. IEEE Transactions on Emerging Topics in Computing, vol. 6, no. 3, pp. 357-369.

[7] Lee, S. *et al*. (2009). A scalable micro wireless interconnect structure for CMPs. MobiCom '09 Proceedings of the 15th annual international conference on Mobile computing and networking, Beijing, pp. 217-228.

[8] Murray, J, Wettin, P, Pande, P, & Shirazi, B (2016). Sustainable Wireless Network-on-Chip Architectures, 1st ed.: Morgan Kaufmann.

[9] Ganguly, A. *et al*. (2011). Scalable Hybrid Wireless Network-on-Chip Architectures for Multicore Systems. IEEE Transactions on Computers, vol. 60, no. 10, pp. 1485-1502.

[10] Deb, S. et al. (2012). Design of an efficient NoC architecture using millimeter-wave wireless links. Thirteenth International Symposium on Quality Electronic Design (ISQED), Santa Clara, CA, pp. 165-172.

[11] Bahrami, B., Jabraeil Jamali, M., & Saeidi, S. (2018). A novel hierarchical architecture for Wireless Network-on-Chip. Journal of Parallel and Distributed Computing, vol. 120, pp. 307-321.

[12] Mineo, A., Palesi, M., Ascia, G., & Catania, V (2016). Exploiting antenna directivity in wireless NoC architectures. Microprocessors and Microsystems, vol. 43, pp. 59-66.

[13] Wang, C., Hu, W., & Bagherzadeh, N. (2011). A Wireless Network-on-Chip Design for Multicore Platforms. 19th International Euromicro Conference on Parallel, Distributed and Network-Based Processing, Ayia Napa, pp. 409-416.

[14] Catania, V., Mineo, A., Monteleone, S., Palesi, M., & Patti, D. (2017). Improving energy efficiency in wireless network-on-chip architectures. ACM Journal on Emerging Technologies in Computing Systems, vol. 14, no. 1, pp. 1-24.

[15] Teuscher, C. (2007). Nature-inspired interconnects for self-assembled large-scale network-on-chip designs. Chaos: An Interdisciplinary Journal of Nonlinear Science, vol. 17, no. 2.

[16] Bahrami, B., Jabraeil Jamali, M., & Saeidi, S. (2016). Proposing an optimal structure for the architecture of wireless networks on chip. Telecommunication Systems, vol. 62, no. 1, pp. 199-214.

[17] Bahrami, B., Jabraeil Jamali, M., & Saeidi, S. (2017). A Demand-Based Structure for the Architecture of Wireless Networks on Chip. Wireless Personal Communications, vol. 96, no. 1, pp. 455-473.

[18] Korte, B. & Vygen, J. (2006). Combinatorial Optimization: Theory and Algorithms, 3rd ed.: Springer-Verlag Berlin Heidelberg.

[19] Keshavarz, H.R. & Saniee Abadeh, M. (2018). MHSubLex: Using Metaheuristic Methods for Subjectivity Classification of Microblogs. Journal of AI and Data Mining, vol. 6, no. 2, pp. 341-353.

[20] Kirkpatrick, S., Gelatt Jr., C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. Science, vol. 220, no. 4598.

نشریه هوش مصنوعی و داده کاوی

یک روش جایابی مبتنی بر تبرید شبیه‌سازی شده آگاه از مسیریابی در شبکه‌های بر روی تراشه بی‌سیم

علیرضا تجری* و اسماعیل طحانیان

دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شاهرود، شاهرود، ایران.

ارسال ۲۰۱۹/۰۹/۲۸؛ بازنگری ۲۰۲۰/۰۱/۰۷؛ پذیرش ۲۰۲۰/۰۳/۳۱

چکیده:

شبکه‌های بر روی تراشه بی‌سیم، یکی از شبکه‌های امیدوار کننده برای معماری‌های نوین سیستم بر روی تراشه است. این معماری‌ها، علاوه بر لینک‌های سیمی، از لینک‌های بی سیم نیز استفاده می‌کنند. استفاده از این لینک‌های بی سیم باعث می شود که بسته‌ها سریع‌تر و با مصرف توان کمتری به گره های مقصد برسند. معماری‌های متفاوتی برای شبکه‌های روی تراشه بی‌سیم ارائه شده است که در موقعیت مسیریاب‌های بی‌سیم و نحوه تعامل آن‌ها با سایر مسیریاب‌ها تفاوت دارند. تعیین موقعیت مسیریاب‌های بی‌سیم (جایابی) گامی مهم در طراحی این معماری‌ها است. در این مقاله، یک روش جایابی مبتنی بر تبرید شبیه سازی شده ارائه شده است که الگوریتم مسیریابی را در طراحی تابع هزینه در نظر می‌گیرد. برای ارزیابی روش پیشنهادی، از ناکسیم که یک نرم‌افزار شبیه‌ساز شبکه بر روی تراشه است، استفاده شده است. نتایج شبیه‌سازی نشان می‌دهد که روش پیشنهادی می‌تواند تاخیر فلیت را تا ۲۴٫۶٪ با حدود ۰٫۲٪ سربار در مصرف توان کاهش دهد.

کلمات کلیدی: تبرید شبیه‌سازی‌شده، شبکه‌های بر روی تراشه بی‌سیم، جایابی.