Journal of Artificial Intelligence and Data Mining (JAIDM), Vol. 10, No. 3, 2022, 311-320.

Shahrood University of Technology

Journal of Artificial Intelligence and Data Mining (JAIDM)
Journal homepage: http://jad.shahroodut.ac.ir

Research paper

A Simulated Annealing-based Throughput-aware Task Mapping Algorithm for Manycore Processors

Alireza Tajary* and Hossein Morshedlou

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

Article Info

Article History:
Received 26 December 2021
Revised 11 April 2022
Accepted 24 May 2022

DOI:10.22044/jadm.2022.11518.2312

Keywords:
Simulated Annealing, Manycore Processors, Task Mapping.

**Corresponding author :tajary@shahroodut.ac.ir (A. Tajary).*

Abstract

With the advent of having many processor cores on a single chip in many-core processors, the demand for exploiting these on-chip resources to boost the performance of applications has been increased. Task mapping is the problem of mapping the application tasks on these processor cores in order to achieve a lower latency and a better performance. Many research works are focused on minimizing the path between the tasks that demand a high bandwidth for communication. Although using these methods can result in a lower latency, at the same time, it is possible to create congestion in the network, which lowers the network throughput. In this paper, a throughput-aware method is proposed that uses simulated annealing for task mapping. The method is checked on several real-world applications, and simulations are conducted on a cycle-accurate network on a chip simulator. The results obtained illustrate that the proposed method can achieve a higher throughput, while maintaining the delay in the network on chip.

1. Introduction

The demand for having more processing power has resulted in having superscalar processors, multicore processors, and finally many-core processors [1] [2] [3]. In multicore and many-core processors, more than one processor core exists on a single chip [4] [5]. As the communication infrastructure of the many-core processors is the bus technology, and the bus technology does not scale well with the number of connected modules to it, the number of processor cores on multicore processors is limited. In order to have more processing cores on a single chip, many-core processors have been introduced.

The underlying technology for communication in many-core processors is the network on chip (NoC) [6]. In NoCs, multiple cores known as the processing elements (PEs) are connected through a communication topology. Many communication typologies like ring, 2D mesh, 3D mesh, and hyper-cube are proposed in NoCs. Among them, the 2D mesh is widely used as a baseline topology in many research works on NoCs. In this topology, each PE is connected to a local router, and each router is connected to four adjacent routers in the north, east, south, and west directions. Having an optimal routing algorithm and mapping of application on PEs are two of the challenges in NoCs.

A directed graph can be generated for each application. Each vertex in that graph shows a task to be run, and each edge represents the direction of padding data, and its weight shows the required bandwidth to transfer data between vertices. This directed graph is called a task graph [7] [8]. The efficient mapping of each node in a task graph on a processing element is one of the challenges in NoC [9] [10]. For example, each node in a task graph possibly sends and receives data to and from more than four nodes, while a processing element in NoC, at most, have four neighbors and there is not a dedicated link between each pair of nodes in NoC (in the 2D mesh topology).

One heuristic to tackle the mapping problem in NoC is a mapping in which the nodes with high bandwidth edges should be adjacent to each other.

This heuristic is used as the base idea for many research works in the literature [7] [9] [10]. In order to implement this idea, a cost function (known as the communication cost function) is proposed, in which the mapping with the lowest cost is considered as the best one. This means that the mapping problem can be expressed as an optimization problem. Since the search space of the cost function is large, many meta-heuristic methods like simulated annealing [11], genetic algorithm [12] [13], and PSO [14] are proposed to find the best solution.

Solving the mentioned optimization problem leads to having short paths between nodes during the program execution. This makes congested areas in the NoC around the high bandwidth demand nodes. On the other hand, based on the routing and switching algorithms, this congestion can result in having a higher latency for delivering packets. For example, for transferring packets in the wormhole switching algorithm [15], the whole path will be reserved for a packet, which means that no other packets can be delivered if it uses any hops from that path. High latency has been reported by [9] in mapping the MPEG-4 application on a 4x4 mesh NoC, resulting in 9000 clock cycles required to deliver a flit, while it should be a value around 7.

Moreover, the congestion in NoC can lead to lowering the network throughput. Since many packets are blocked by other packets, we reach a lower ratio of packet delivery in NoC. This means that the network throughput decreases.

In this paper, a throughput-ware task mapping algorithm is presented that considers the true throughput of NoC in the cost function of the optimization problem. The optimization problem is solved using the simulated annealing meta-heuristic algorithm. The simulation results show that the proposed algorithm can achieve a higher throughput compared to the related works.

The main contributions of the paper are summarized as what follows.

- We proposed a new cost function that uses the dynamic features of the task graph.
- We modified the noxim cycle accurate simulator to generate the traffic patterns based on the task graph and its corresponding mapping.
- We investigated the required simulation time in the simulator to achieve the acceptable results.
- We developed a simulated annealing algorithm to optimize the proposed cost function.
- We explain and analyse the results by the mann-whitney statistical test.

The rest of the paper is as what follows. In Section 2, we cover the background knowledge of NoC and task mapping. Section 3 covers the related works. The proposed method will be presented in Section 4. The simulation results are shown in Section 5, and Section 6 concludes the paper.

2. Background

In this section, we first introduce the NoC and routing in NoC, and then the application task graph and mapping will be discussed.

2.1. NoC and routing

NoC is accepted as the underlying technology for the communication infrastructure of the future many-core processors [4]. The NoC topology defines the arrangement of PEs and their connections in an NoC. For example, 2D mesh is a widely used topology, in which PEs are arranged in a row and column fashion. In this topology, each PE will be connected to a router and that router will be connected to its neighbor routers in the four main directions. Figure 1 shows a 4x4 NoC with 2D mesh topology, which contains 16 routers.

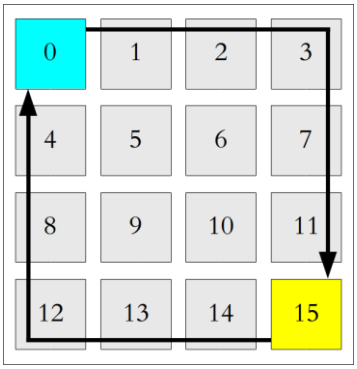

Figure 1. 4x4 2D mesh NoC with 16 routers, with XY routing.

Transferring data in NoC is a step-by-step process [6]. In the wormhole switching algorithm [16], each packet will be split into some flits; the first flit acts as the head of a worm, which reserves a path for the packet from sender to receiver. Other flits will follow the head pack to reach the destination. The last flit, known as tail, releases the path, which allows other packets to reserve it. The routing algorithm defines the movement strategy for the head flit, which will be implemented in the routers of NoC.

XY [17] is a routing algorithm that splits the routing into two steps: 1) going through the x-direction to reach the column of the destination, and 2) going through the y-direction to reach the

row of the destination. Figure b1 shows the paths to send data from node #15 to node #0 and vice versa. It is important to note that in this figure, the two paths do not use the same routers.

2.1. Task graph and mapping problem

Since there are many cores in the NoC architectures, they are suitable for running parallel applications. Parallel applications contain tasks that run parallel and cooperate with each other. Tasks exchange data between themselves to perform their processing requirements [7]. The task graph *G(V, E)* is a directed graph in which the vertex $v_i \in V$ is a task in the application and the edge $e_{ij} \in E$ is directed edge from v_i to v_j with the weight w_{ij} that represents the required bandwidth for delivering data from v_i to v_j.

The task graphs of several applications exist in the literature. For example, the task graph of Video Object Plane decoder (VOPD) application is shown in Figure 2. As shown in this figure, this application has 16 main tasks shown as nodes in this graph. Each task does a special processing on its inputs. For example, the task #1 is the variable length decoder, the task #2 is run length decoder, and the task #3 is inverse scan. The edges of the graph show the required bandwidth for transferring data between tasks in MB/S.

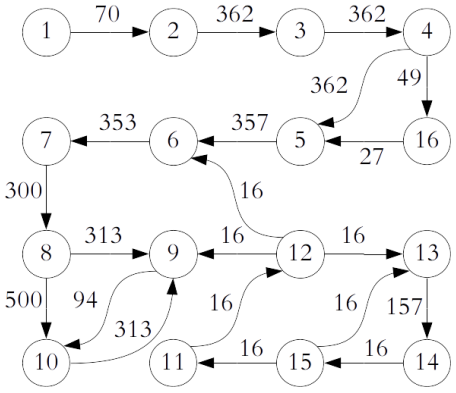

Figure 2. Task graph of VOPD.

Moreover, there is an NoC topology graph *N(P, L)* that represents the processing elements as vertices and the connection links between them as edges. The weight of each edge in L represents its capacity of bandwidth. A mapping from a task graph to an NoC is a one-to-one mapping of task graph vertices to NoC PEs [7]. Therefore, a mapping is a function that can be describe as:

$$f : V \to P \tag{1}$$

$$f = \{(v, p) \mid v \in V \wedge p \in P\}$$

There are two constraints for this function: 1) each element of P can be mapped by at most one element of V, and 2) each link in the NoC has a limited capacity; therefore, the next constraint of mapping is that the maximum requested bandwidth of each edge in L should be less than or equal to its capacity.

It is commonly accepted that having nodes near each other results in a lower latency and a better throughput [7] [9]. In order to find the best mapping, an optimization problem has been formulated. For each mapping, a cost can be computed as the sum of multiplying each edge weight (from task graph) to the length of the path from source and destination routers (in NoC). This cost function is shown in Equation 2 [7]. In this equation, w is the weight of the edge from the task graph, and dist(s,d) is the Cartesian distance between the source and the destination of the corresponding edge in NoC (as shown in Equation 3). Since the space of the solutions is very large, several evolutionary algorithms have been proposed to find the best solution that minimizes the cost function [11] [18] [14].

$$cost = \sum_{i=1}^{|E|} weight(e_i) \times dist(src(e_i), dst(e_i)) \tag{2}$$

$$dist(s,d) = |col(s) - col(d)| + |row(s) - row(d)| \tag{3}$$

3. Related Works

Application mapping is a challenging job in the NoC architecture design. The researchers have proposed many algorithms to overcome its complexity. These methods differ from the exact methods to the evolutionary and meta-heuristic methods.

Integer linear programming (ILP) is used in [19] for application task mapping on 2D mesh NoC. As the search space for some applications is very large, finding the optimal mapping by ILP requires a vast processing power and execution time. The communication cost (Equation 2) is used as the objective function to be reduced in [19]. In order to reduce the runtime overhead of ILP, the researchers in [20] have used clustering-based relaxation for the formulation of ILP. They broke the objective function of [19] into two parts. The first one is the objective function toward the X axis, and the next one is the objective function toward the Y axis. They showed that their method could reach optimal or near optimal solutions in a manageable time based on the objective function. In [21], the authors have proposed a performance-aware mapping algorithm on NoCs with express channels. Express channels are dedicated channels between distant nodes in NoC. They proposed a heuristic based mapping algorithm to minimize

the communication cost. The simulation results showed that this method could achieve a better energy consumption and performance compared to the ILP method. The method in [22] converts the application graph to an abstract graph, and maps this abstract graph on NoC. They used a heuristic method for the mapping, and used the communication cost as the objective function and tried to reduce the packet latency by placing tasks close to each other.

In [23], the researchers have presented the NMAP method that tries to reduce the routing path between the routers in NoC considering the required bandwidth between the nodes, which results in a lower communication delay. An optimized version on NMAP has been presented in [24] as ONMAP. It uses an optimized, search based near-optimal mapping heuristic for task mapping to reduce the energy consumption and packet latency in NoCs. In [25], the authors have proposed the CastNet, which considers the power consumption of NoC in addition to the communication delay in NoC. In [26], the LMAP method has been proposed, which uses a bi-partitioning method for mapping based on the bandwidth demand.

Simulated annealing has been used in [11] for the task mapping problem in NoC considering the distance of the routers in the cost function of the simulated annealing algorithm. The same idea on the cost function has been presented in [18] to find the optimal solution by the ant colony technique. The objective function in these methods is the communication cost function. A genetic algorithm is also used to find the optimal solution in NoC [12] [13]. Particle swarm optimization has been used in [14] to find the optimal solution to the task mapping problem in the NoC that considers the statics operations of the NoC. In [30], the authors have proposed a multi-level mapping algorithm for deep learning neural networks on NoCs. One of the objectives of this method is reducing the communication cost based on the static parameters of the mapping. Finally, in [8], the authors have used the cuckoo search optimization with lévy flight for application mapping in NoC. Their method works in two steps. In the first step, they place the maximum communicating tasks together. The second step optimizes the first placement on NoC. The objective in this method is also reducing the cost function.

Table 1 shows a summary of the related works. As shown in this table, the related works try to minimize the communication cost using different algorithms. It is important to note that in some applications, reducing the communication cost does not result in a better mapping. Therefore, an updated objective function is required for the mapping algorithms.

Table 1. Summary of the related studies

Type	Reference	Objectives
Exact	[19] [20]	reducing communication cost in manageable time using ILP
Heuristic	[21] [22] [23] [24] [25] [26] [30]	reducing communication cost to achieve lower latency and power consumption
Meta heuristic	[8] [11] [12] [13] [14] [18]	reducing communication cost

4. Proposed Method

The proposed method uses a custom objective function to be used in the simulated annealing algorithm. The motivation for using this custom objective function is described in the next sub-section.

4.1 Motivation and contributions

The cost function defined in Equation 2 tries to minimize the path between nodes, especially between the high communicating nodes. Finding the optimum mapping for this objective function does not yield the best configuration for NoC while running the application. In order to further illustrate this idea, we use an example. Please consider Figure 3. In this figure, (a) shows a task graph of an application that contains three tasks. This task graph should be mapped on an NoC with six PEs arranged in two rows and three columns. Figures 3.a and 3.b are two mappings for this task graph. The cost of mapping for (b) is 40, while the cost of mapping for (c) is 45. Hence, the optimization algorithm prefers the first mapping over the second one. Now consider the required bandwidth for each link in the NoC. In the first mapping, we have two links that are shared between two paths. The first link (the link between nodes 1 and 2) requires a bandwidth of 25, and is shared for sending data from node #1 to the other nodes. The second link (the link between nodes 2 and 3) is also shared, and requires a bandwidth of 15. On the other hand, in the second mapping, just one of the links is shared, and it requires the bandwidth of 15. Therefore, the second mapping shows better results in terms of the packet delay and the network throughput. Based on this discussion, we consider a new objective function based on the throughput of NoC for each mapping.

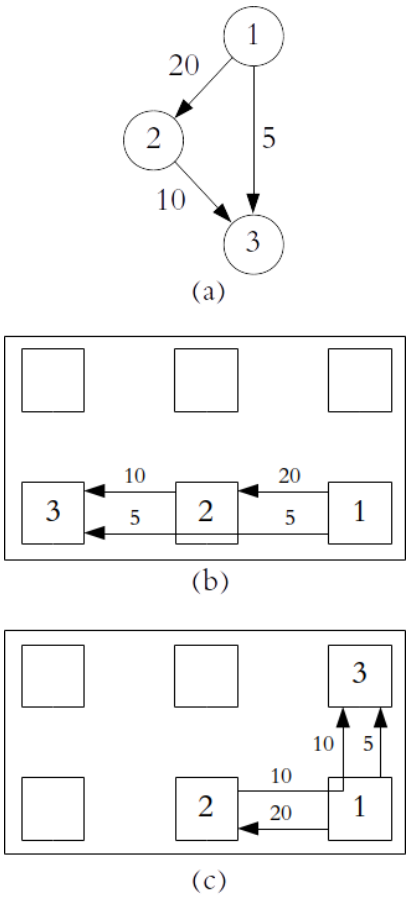

Figure 3. A task graph with two mappings: a) task graph, b) first mapping, c) second mapping.

4.2. Routing-aware cost function

The cost function of the optimization problem will be the negative of the network throughput of NoC. The reported throughput value will be extracted from the output of the NoC simulator that the task graph of the application is mapped on it and simulates execution of the application. As the delivery of data in NoC is executed by the routing algorithm, the effect of the routing algorithm on the network throughput of the NoC will be considered. It is also important to note that the more network throughput, the better the mapping. Therefore, the cost function is defined as the negative of the network throughput.

Since we want to use the simulated annealing algorithm for optimizing the cost function, we are required to compute the cost function for many task mappings on NoC. Therefore, the time for computing each cost function should be small. One main issue about extracting the network throughput is the runtime overhead for running the NoC simulator. In order to overcome this issue, we simulate NoC for a small time, extract the network throughput, and use that value to compute the cost function. The simulated annealing method optimizes this cost function.

Simulated annealing is a meta-heuristic algorithm to find the global optimum of a given function. It is widely used in research to find the best solution to the optimization problems [27]. The temperature is the main parameter in this algorithm. At the start of the algorithm, the temperature is high, and during the runtime of the algorithm, the temperature decreases based on a cooling function. The algorithm uses one solution during its runtime. In each step, a new solution will be generated from the current solution based on a perturbation function. Then the cost of the new solution will be computed. If the cost function of the new solution is better than the cost function of the current solution, the current solution will be replaced by the new solution; otherwise, replacing the current solution with the new solution is based on probability. This probability is higher when the temperature is high or the difference in the costs is low. The probability of accepting a solution is shown in Equation 4.

$$P(cost_n, cost_c, T) = \frac{e^{cost_c - cost_n}}{T} \quad (4)$$

The detailed design of the algorithm is shown in Algorithm 1. As shown in this algorithm, the algorithm begins with setting the initial temperature and temperature reduction rate (alpha) and having a random mapping as the first solution. The main loop of the algorithm iterates until the temperature is small enough. The temperature value decreases by the rate of alpha. At each temperature, we generate 20 new mappings from the previous one. The best mapping (in terms of the cost) will be selected as the candidate for the next temperature. Generating these new mappings from the last one is done through the perturb function.

The perturb function makes a new mapping from the old one. It uses three methods for its job, as shown in Figure 4. In the first method, it selects two routers in NoC and exchanges their corresponding tasks numbers (Figure 4.a). In the second method, three routers will be selected and their corresponding task numbers will be changed roundly (Figure 4.b). In the third method, four routers will be selected and their task number will be exchanged two by two, as shown in Figure 4.c.

4.3. Framework of proposed method

The main framework of the proposed method is shown in Figure 5. As shown in this figure, the first step in this framework is creating a random mapping. In a random mapping, the tasks in the

task graph will be randomly mapped to PEs in NoC. In the next step, a new mapping will be generated from the last mapping using the perturb function (as shown in Figure 4). After that, the cost of mapping should be computed (the dotted box).

Algorithm 1: Proposed simulated annealing algorithm for task mapping.

```
Inputs:
   initT: the initial temperature
   minT: the minimum temperature
   alpha: the alpha value for temperature reduction
   nipt: number of iterations per temperature
   nipm: number of perturbs for each mapping
Outputs:
   bestAnswer: the mapping with the lowest cost

Main Variables:
   bestAnswer: the mapping with the lowest cost
   bestCost: the cost of the bestAnswer
   currentAnswer: the current mapping
   currentCost: the cost of the currentAnswer
   T: the temperature
Procedure:

   currentAnswer = a random mapping
   currentCost = Cost(currentAnswer)

   best Answer = currentAnswer
   bestCost = currentCost

   T = initT

   While (T < minT) do
      for i from 1 to nipt
         mapping = Perturb(currentAnswer)
         cost = Cost(mapping)

         for j from 1 to nipm
            anotherMapping = Pertrurb(currentAnswer)
            anotherCost = Cost(anotherMapping)

            if anotherCost < cost then
               mapping = anotherMapping
               cost = anotherCost
            end if
         end for

         if cost < bestCost then
            bestCost = cost
            bestAnswer = mapping
         end if

         if cost < currentCost then
            currentAnswer = mapping
            currentCost = cost
         else
            delta = cost - currentCost
            r = a random from [0,1)

            if r < P(cost, currentCost, T)  then
               currentAnswer = mapping
               currentCost = cost
            end if
         end if
      end for
      update T
   end while
```

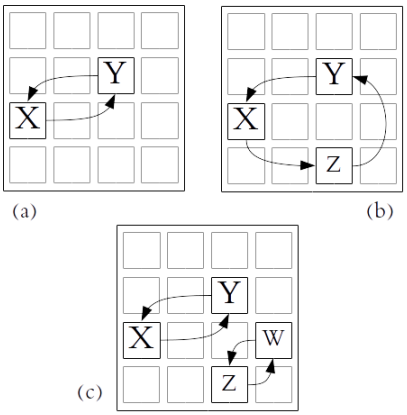

Figure 4. Three methods in perturb function: a) swapping two nodes, b) circulating three nodes, and c) swapping four nodes, two by two.

Cost of a mapping is the negative of its corresponding NoC throughput. Therefore, in order to extract the throughput of a mapping, we modified the noxim simulator to generate the traffics based on the task graph and mapping. Hence, the inputs of the noxim will be the mapping, the task graph, and its base configuration. After running the noxim simulator, n script extracts the NoC throughput from its output. Finally, the cost of the mapping will be computed as the negative of the extracted NoC throughput.

After computing the cost of a mapping, the cost will be compared to the current cost in the SA algorithm. The SA algorithm maintains a cost value for accepting new mappings. If the cost of the new mapping is less than the current cost, the mapping will be accepted, and the current cost will be updated; otherwise, the new mapping will be accepted based on the acceptance probability. In both cases, the temperature of the SA algorithm will be decreased based on the temperature reduction policy. If the temperature is less than the margin value, the algorithm will be finished and the best accepted mapping will be reported; otherwise, a new mapping will be generated by perturbing the current mapping, and this process will be repeated for the new mapping.

4.4 Complexity analysis of proposed method

The main idea of the proposed method is using another cost function for the comparing mappings. The complexity of the traditional cost function (Equation 2) can be expressed as O(|V|+|E|). The |V| factor is used since each vertex in V should be mapped on a PE in NoC and each mapping is done in a constant time. The |E| factor is used because for each edge in E, its weight should be multiplied by its Cartesian length, which is a constant time.

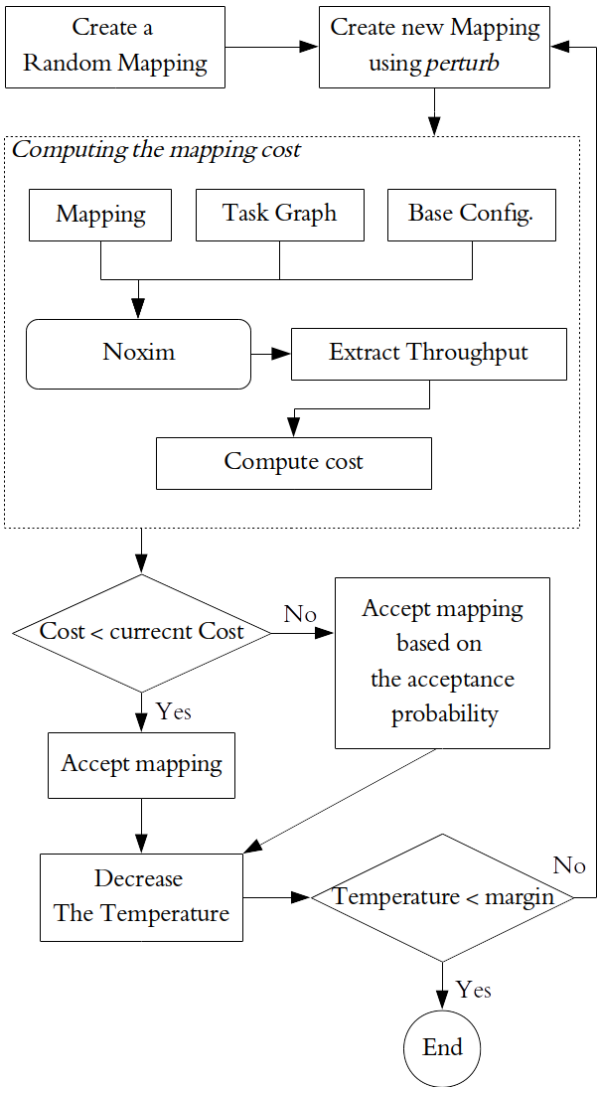

Figure 5. Framework of proposed method.

In the proposed method, the cost function will be calculated after simulation of NoC for k cycles. The number of cycles (k) is a constant. For each cycle, each node in NoC receives data from its neighbors and sends data to them based on the edges in the task graph. For each node, this delivery of data will be done in a constant time. Therefore, the complexity of the proposed method can be expressed as O(k*(|P|+|E|)). Usually, the number of nodes in NoC is almost equal to the number of the vertices in the task graph. On the other hand, k is a constant value, Hence, the complexity of the proposed method can be expressed as O(|V|+|E|). Of course, there are constant values in each complexity functions, and the constant values in the proposed method is much greater than the traditional methods but the complexity of the proposed method is equal to the complexity of the traditional methods. Therefore, for large-scale applications, the proposed method behaves likewise the related works.

5. Experimental Results

The simulated annealing algorithm is implemented in the C++ programming language, and has been compiled using the gcc compiler [28], in the Linux operating system. The simulations are run on a computer with Core i5 processor and 8 GB ram. The Noxim NoC simulator [29], which is a cycle accurate NoC simulator, is also used. Noxim uses the system library and can be compiled and run on the Linux operating system with the gcc compiler. We also used the benchmark applications from the [7] for the NoC simulation. The parameters of the environment and simulation framework are shown in Table 2.

Table 2. Environment and simulation framework

Parameter	Value
Compiler	gcc 7.5
CPU	Intel Core i5
Number of rows in NoC	Based on task graph
Number of columns in NoC	Based on task graph
NoC Flit Width	32 bit
NoC Clock Cycle	1 GHz
Warm-up time during search	100 cycles
Simulation length during search	1K cycles
Warm-up time	1000 cycles
Simulation length	500K cycles

5.1 Simulation results

For extracting the throughput while searching for the best mapping, we used 100 cycles for warm-up time and 1K cycles for simulation time. These values are replaced by 1000 and 500K for final simulations, respectively. The heuristic was to use the throughput of the network for a small number of cycles as an estimation of the final throughput of the network. In order to validate this idea, we created several configurations for different numbers of simulation cycles. Figure 6 shows the normalized throughput of the NoC with 1K cycles to the NoC with 500K cycles for various applications. In this figure, we are interested in the columns that their value is close to 1 (which is the base throughput for 1M cycles). We are also interested in the columns that their cycles are smaller. As shown in this figure, although the columns differ, on average, the difference is less than 5%.

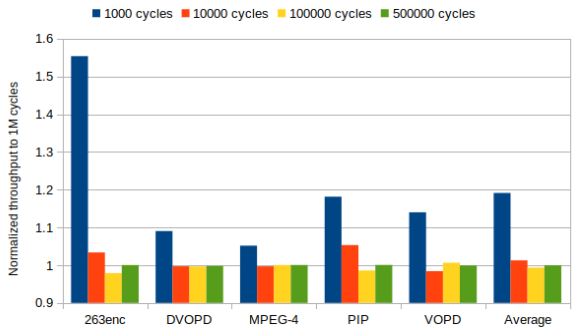

Figure 6. Normalized throughput of NoC with 1K cycles to NoC with 500K cycles.

In order to see the effect of alpha (temperature reduction rate) on the result of the simulated annealing algorithm, we used several values for it and extracted the value of the cost function during the execution of the algorithm for the VOPD benchmark program. Table 3 shows the value of the cost function (negative of the throughput) for different values of alpha. As shown in this table, the best cost corresponds to the alpha of value 0.99. Higher values for alpha cause the temperature to decrease slowly that yields having more iteration of the main loop.

Table 3. Value of cost function for different values of alpha

Alpha	Number of iterations	Best cost
0.1	6	-7.38264
0.3	10	-7.43125
0.5	17	-7.45417
0.8	52	-7.47292
0.9	110	-7.53681
0.95	225	-7.53889
0.98	570	-7.53681
0.99	1146	-7.52847

In order to show the effectiveness of the proposed method, we compared the throughput of the proposed method with the throughput of the methods with a distance-based cost functions. Figure 7 shows the network throughput for various methods introduced in the related works for the VOPD application. In this figure, the proposed method has two configurations: 1) the output of the proposed method with 10K simulation cycles for extracting the mapping, and 2) the output of the proposed method with 1K simulation cycles for extracting the mapping. As shown in this figure, the proposed method achieves a better throughput compared to the related works.

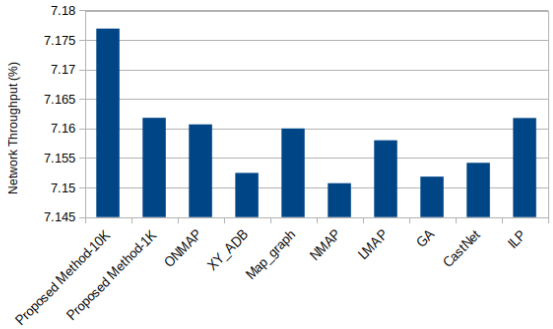

Figure 7. The throughput of proposed method compared to distance-based methods.

Figure 8 shows the normalized throughput of the proposed method compared to the simulated annealing method with the objective of reducing the communication cost for four other benchmarks namely: MPEG-4, 263enc, 263dec, and mp3enc. For extracting these results, the simulator was run for 200K cycles. As shown in this figure, on average, the proposed method can achieve 8% more throughput, compared to the simulated annealing method.

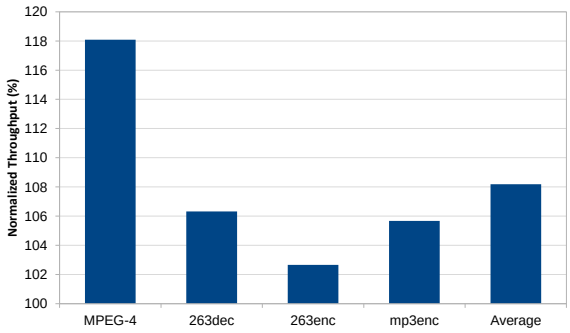

Figure 8. Normalized throughput of proposed method compared to simulated annealing method for different benchmark programs.

It is also important to note that the distance cost of the proposed method is higher than the related works. The distance cost of the proposed method and related works are shown in Figure 9. As shown in this figure, the distance cost of the proposed method is at least 4% more than the distance cost of the related works.

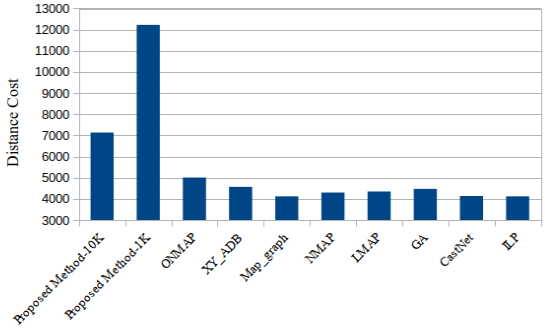

Figure 9. Distance cost of proposed method compared to related works.

As the final result, we compared the network throughput, power consumption, and average delay of flits with the related works in Table 4 for VOPD benchmark program. As shown in this table, the proposed method achieved a better network throughput with the same power consumption and flit delivery delay.

Table 4. Final result as a table comparing throughput, power, and delay of proposed method with distance-based methods for VOPD.

Method	Distance cost	Network throughput	Flit delivery delay
Proposed method-10K	7134	7.17692	7.94533
Proposed method-1K	12224	7.16182	6.76437
ONMAP	5007	7.1607	7.04287
XY_ADB	4568	7.15248	8.54206
Map_graph	4119	7.16002	6.54371
NMAP	4297	7.15074	7.84658
LMAP	4346	7.15801	9.0228
GA	4471	7.15185	6.54009
CastNet	4135	7.1542	5.79931
ILP	4119	7.16179	8.87753

5.2 Statistical testing of results

Statistical testing is used for verification of the results compared to related works. Several statistical testing methods exist, and each one is suitable for a specific task. For example, the KSPA test is used for determining the accuracy of two sets of forecasting [31]. In this paper, we used the Mann-Whitney U test, which is a non-parametric test to compare two groups of data without assuming that values have a normal distribution. In this test, the null hypothesis is that the median of the two groups are identical. On the other hand, the alternative hypothesis is that the median of the two groups are not identical; hence, the two groups differ. To compare the two groups, this test extracts a U value from the data, and for each significance level (for example 0.05), it generates a $U_{critical}$ value. If U value is greater than $U_{critical}$, then the result is not significant, and the null hypothesis will be accepted; otherwise, the null hypothesis will be rejected (the alternative hypothesis will be accepted), which indicates that the groups differ.

For comparing the results of the proposed method with the results of the related works, we executed each algorithm ten times and extracted the throughput value for each execution. Then group of ten values will be created for each method. After that, we compared the group of the proposed method to each group of the related works using the Mann-Whitney U test. The results of the tests for the significance level of 0.05 are shown in Table 5. The method column in this table shows the algorithm that the proposed method is compared to it. The last column shows whether the two groups are different. Therefore, the results of Table 4 will be verified.

Table 5. Mann-whitney U test results for comparing proposed method with related works.

Method	U value	$U_{critical}$ (at 0.05)	is significant?
ONMAP	1	27	Yes
XY_ADB	1	27	Yes
Map_graph	2	27	Yes
NMAP	2	27	Yes
LMAP	2	27	Yes
GA	0	27	Yes
CastNet	0	27	Yes
ILP	1	27	Yes

6. Conclusions and Future Works

Task mapping is the first stage for running applications in the many-core processors. The main idea is to map the tasks with nodes that require more communications to be adjacent to each other. In some situations, this idea leads to having more congestion around the corresponding routers in the NoC, which results in a lower network throughput. In order to overcome this issue, we proposed a meta-heuristic-based mapping algorithm that considered the routing algorithm of the NoC and tried to maximize the throughput of the network (by minimizing the cost function). The simulation results showed that the proposed method could achieve more throughputs while maintaining the same power consumption and flit delivery delay. One of the drawbacks of the proposed method is having a very slow cost function. As a future work, we try to propose a faster cost function that does not need running the NoC simulator.

References

[1] P. Kansakar and A. Munir, "A design space exploration methodology for parameter optimization in multicore processors," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 29, No. 1, pp. 2–15, 2018.

[2] A. Balakrishnan and A. Naeemi, "Optimal global interconnects for networks-on-chip in many-core architectures," *IEEE Electron Device Letters*, Vol. 31, No. 4, pp. 290–292, 2010.

[3] F. N. Sibai, "A two-dimensional low-diameter scalable on-chip network for interconnecting thousands of cores," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 23, No. 2, pp. 193–201, 2012.

[4] Y. Liu, S. Kato, and M. Edahiro, "Analysis of memory system of tiled many-core processors," *IEEE Access*, Vol. 7, pp. 18964–18977, 2019.

[5] H. Jang *et al*., "Developing a multicore platform utilizing open risc-v cores," *IEEE Access*, Vol. 9, pp. 120010–120023, 2021, doi: 10.1109/ACCESS.2021.3108475.

[6] A. Vijaya Bhaskar and T. Venkatesh, "Performance analysis of network-on-chip in many-core processors," *Journal of Parallel and Distributed Computing*, Vol. 147, pp. 196–208, 2021.

[7] P. K. Sahu and S. Chattopadhyay, "A survey on application mapping strategies for network-on-chip design," *Journal of Systems Architecture*, Vol. 59, No. 1, pp. 60–76, 2013.

[8] M. J. Mohiz, N. K. Baloch, F. Hussain, S. Saleem, Y. B. Zikria, and H. Yu, "Application mapping using cuckoo search optimization with lévy flight for noc-based system," *IEEE Access*, Vol. 9, pp. 141778–141789, 2021, doi: 10.1109/ACCESS.2021.3120079.

[9] P. Mazaheri Kalahroudi, E. Yaghoubi, and B. Barekatain, "IAM: An improved mapping on a 2-d network on chip to reduce communication cost and energy consumption," *Photonic Network Communications*, Vol. 41, No. 1, pp. 78–92, Feb. 2021, doi: 10.1007/s11107-020-00911-x.

[10] W. Amin *et al*., "Performance evaluation of application mapping approaches for network-on-chip designs," *IEEE Access*, Vol. 8, pp. 63607–63631, 2020, doi: 10.1109/ACCESS.2020.2982675.

[11] C. Marcon, A. Borin, A. Susin, L. Carro, and F. Wagner, "Time and energy efficient mapping of embedded applications onto nocs," in *Proceedings of the asp-dac 2005. asia and south pacific design automation conference*, 2005, pp. 33–38. doi: 10.1109/ASPDAC.2005.1466125.

[12] T. Lei and S. Kumar, "A two-step genetic algorithm for mapping task graphs to a network on chip architecture," in *Euromicro symposium on digital system design*, 2003, pp. 180–187. doi: 10.1109/DSD.2003.1231923.

[13] W. Zhou, Y. Zhang, and Z. Mao, "An application specific noc mapping for optimized delay," in *International conference on design and test of integrated systems in nanoscale technology*, 2006, pp. 184–188. doi: 10.1109/DTIS.2006.1708657.

[14] P. K. Sahu, P. Venkatesh, S. Gollapalli, and S. Chattopadhyay, "Application mapping onto mesh structured network-on-chip using particle swarm optimization," in *2011 IEEE computer society annual symposium on VLSI*, 2011, pp. 335–336. doi: 10.1109/ISVLSI.2011.21.

[15] I. Lang, N. Kapre, and R. Pellizzoni, "Worst-case latency analysis for the versal noc network packet switch," in *Proceedings of the 15th ieee/acm international symposium on networks-on-chip*, 2021, pp. 55–60.

[16] E. Stergiou, "A study of multistage interconnection networks operating with wormhole routing and equipped with multi-lane storage," *International Journal of Parallel, Emergent and Distributed Systems*, Vol. 36, No. 3, pp. 221–239, 2021.

[17] S. D. Chawade, M. A. Gaikwad, and R. M. Patrikar, "Review of xy routing algorithm for network-on-chip architecture," *International Journal of Computer Applications*, Vol. 43, No. 21, pp. 975–8887, 2012.

[18] C. Chen and S. Cotofana, "Link bandwidth aware backtracking based dynamic task mapping in noc based mpsocs," in *Proceedings of the 2014 international workshop on network on chip architectures*, 2014, pp. 5–10. doi: 10.1145/2685342.2685343.

[19] S. Tosun, O. Ozturk, and M. Ozen, "An ilp formulation for application mapping onto network-on-chips," in *2009 international conference on application of information and communication technologies*, 2009, pp. 1–5. doi: 10.1109/ICAICT.2009.5372524.

[20] S. Tosun, "Cluster-based application mapping method for network-on-chip," *Adv. Eng. Softw.*, Vol. 42, No. 10, pp. 868–874, Oct. 2011, doi: 10.1016/j.advengsoft.2011.06.005.

[21] S. D'souza, J. Soumya, and S. Chattopadhyay, "A constructive heuristic for application mapping onto an express channel based network-on-chip," in *2015 19th international symposium on vlsi design and test*, 2015, pp. 1–6. doi: 10.1109/ISVDAT.2015.7208147.

[22] E. Alikhah-Asl and M. Reshadi, "XY-axis and distance based noc mapping (xy-adb)," in *8th international symposium on telecommunications (ist)*, 2016, pp. 678–683. doi: 10.1109/ISTEL.2016.7881908.

[23] S. Murali and G. De Micheli, "Bandwidth-constrained mapping of cores onto noc architectures," in *Proceedings design, automation and test in Europe conference and exhibition*, 2004, Vol. 2, pp. 896–901 Vol. 2. doi: 10.1109/DATE.2004.1269002.

[24] S. Khan, S. Anjum, U. A. Gulzari, F. Ishmanov, M. Palesi, and M. K. Afzal, "An optimized hybrid algorithm in term of energy and performance for mapping real time workloads on 2d based on-chip networks," *Applied Intelligence*, Vol. 48, No. 12, pp. 4792–4804, Dec. 2018, doi: 10.1007/s10489-018-1246-7.

[25] S. Tosun, "New heuristic algorithms for energy aware application mapping and routing on mesh-based nocs," *Journal of Systems Architecture*, Vol. 57, No. 1, pp. 69–78, 2011.

[26] P. K. Sahu, N. Shah, K. Manna, and S. Chattopadhyay, "A new application mapping algorithm for mesh based network-on-chip design," in *annual ieee india conference (indicon)*, 2010, pp. 1–4. doi: 10.1109/INDCON.2010.5712700.

[27] A. Tajary and E. Tahanian, "A routing-aware simulated annealing-based placement method in wireless network on chips," *Journal of AI and Data Mining*, Vol. 8, No. 3, pp. 409–415, 2020, doi: 10.22044/jadm.2020.8964.2034.

[28] GNU Project, "GCC, the gnu compiler collection." [Online]. Available: https://gcc.gnu.org/ (accessed Dec. 01, 2021).

[29] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti, "Improving the energy efficiency of wireless network on chip architectures through online selective buffers and receivers shutdown," in *13th ieee annual consumer communications networking conference (ccnc)*, 2016, pp. 668–673. doi: 10.1109/CCNC.2016.7444860.

[30] Z. A. Khan, U. Abbasi, and S. W. Kim, "An efficient algorithm for mapping deep learning applications on the noc architecture," *Applied Sciences*, Vol. 12, No. 6, 2022, doi: 10.3390/app12063163.

[31] G.-F. Fan, L.-Z. Zhang, M. Yu, W.-C. Hong, and S.-Q. Dong, "Applications of random forest in multivariable response surface for short-term load forecasting," *International Journal of Electrical Power & Energy Systems*, Vol. 139, p. 108073, 2022.

یک الگوریتم نگاشت وظایف آگاه از توان عملیاتی و مبتنی بر تبرید شبیه‌سازی شده برای پردازنده‌های بسا هسته‌ای

علیرضا تجری* و حسین مرشدلو

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

ارسال ۲۰۲۱/۱۲/۲۶؛ بازنگری ۲۰۲۲/۰۴/۱۱؛ پذیرش ۲۰۲۲/۰۵/۲۴

چکیده:

با قرار گرفتن تعداد زیادی هسته بر روی تراشه پردازنده های بسا هسته‌ای، نیاز به استفاده بهینه از این منابع برای بهبود کارآیی نرم‌افزارهایی که بر روی این پردازنده‌ها اجرا می‌شوند، افزایش یافته است. نگاشت وظایف، مسئله‌ای است که در آن، وظایف مربوط به اجزای یک نرم‌افزار، در زمان اجرا بر روی هسته‌های پردازنده نگاشت می‌شوند تا کارآیی اجرای نرم‌افزار بهبود یابد. تحقیقات زیادی بر روی کاهش مسیر بین وظایف نگاشت شده به منظور کاهش تاخیر و افزایش کارآیی ارائه شده‌اند. هر چند این روش‌ها، سعی در کاهش تاخیر دارند، اما با اینکار، باعث ایجاد تراکم در شبکه می‌شوند که در نتیجه آن، توان عملیاتی و کارآیی پردازنده کاهش می‌یابد. برای حل این مشکل، ما یک روش نگاشت وظایف مبتنی بر تبرید شبیه‌سازی شده و آگاه از توان عملیاتی ارائه کرده‌ایم. برای ارزیابی روش پیشنهادی، چندین نرم‌افزار واقعی بر روی یک شبیه‌ساز با دقت سیکل ساعت اجرا شده‌اند. نتایج شبیه‌سازی نشان داده‌اند که روش پیشنهادی می‌تواند بدون تاثیر بر روی تاخیر، توان عملیاتی سیستم را افزایش بدهد.

کلمات کلیدی: تبرید شبیه‌سازی شده، پردازنده‌های بسا هسته‌ای، نگاشت وظایف.