



Article

Optimization Strategy of Regular NoC Mapping Using Genetic-Based Hyper-Heuristic Algorithm

Changqing Xu ^{1,2,*} , Jiahao Ning ², Yi Liu ², Mintao Luo ^{3,*}, Dongdong Chen ² , Xiaoling Lin ⁴ and Yintang Yang ²¹ Guangzhou Institute of Technology, Xidian University, Guangzhou 510555, China² School of Microelectronics, Xidian University, Xi'an 710071, China³ Xi'an Microelectronics Technology, Xi'an 710054, China⁴ China Electronic Product Reliability and Environment Research, Guangzhou 510610, China

* Correspondence: cqxu@xidian.edu.cn (C.X.); lmt7674@126.com (M.L.)

Abstract: Mapping optimization of network-on-chips (NoCs) for specific applications has become one of the most important keys of the SoC top-level design. However, the topology of NoC applied is usually regular topology, such as mesh, torus, etc., which may generate a large number of isomorphic solutions during the process of NoC mapping, which may reduce the convergence speed of mapping algorithms. In this paper, we proposed a generic-based hyper-heuristic algorithm named IRC-GHH for NoC mapping. To reduce the influence of isomorphic solutions, we analyzed the symmetry of NoC topology and proposed crossover operators based on the isomorphic solution to optimize the algorithm. We studied the situation of invalid crossovers and eliminated invalid iterations by adopting an isomorphic replacement crossover (IRC) strategy. To prevent the algorithm from falling into evolutionary stagnation in the late iteration, we introduce an adaptive mechanism to increase the usage frequency of the IRC operator automatically. Compared with GHH without IRC, the GHH with IRC can achieve, on average 15.25% communication energy reduction and 7.84% communication delay reduction.



Citation: Xu, C.; Ning, J.; Liu, Y.; Luo, M.; Chen, D.; Lin, X.; Yang, Y.

Optimization Strategy of Regular NoC Mapping Using Genetic-Based Hyper-Heuristic Algorithm.

Symmetry **2022**, *14*, 1637. <https://doi.org/10.3390/sym14081637>

Academic Editor: Alexander Shelupanov

Received: 18 July 2022

Accepted: 8 August 2022

Published: 9 August 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: network-on-chip; mapping; hyper-heuristic; isomorphic replacement crossover; symmetry

1. Introduction

With the development of a multi-processor system on chip (MPSoC), network-on-chip (NoC), which is a multi-core bus architecture, has been widely concerned and applied [1–3]. However, the latency and power of NoC design have become challenges for designers. There are a large number of data transmission between different masters and slaves in an NoC-based MPSoC. Therefore, mapping IP cores with varying requirements of bandwidth in NoC will directly determine network data latency and communication power. In addition, the cache consistency problem brought by the shared storage system in which MPSoC is embedded [4] puts higher demands on the bandwidth and latency of MPSoC, which puts further pressure on the precious bandwidth on NoC. At the same time, with the increase of bandwidth pressure, NoC network node computing resources will be further exhausted, and more arbitrated computing and packet inter-mediate caching process will also lead to a further increase in network dynamic power consumption.

In order to solve the above problems at the system level, NoC mapping has been one of the hot issues in the industry. Subsequent work and proof of the NoC mapping problem proved to be NP-hard [5,6]. The industry usually uses heuristic algorithms to solve the NoC mapping problems for a particular application, such as simulated annealing (SA) [7], particle swarm optimization (PSO) [8,9] algorithm, and genetic algorithm (GA) [10,11], as well as some hybrid algorithms [12].

A genetic algorithm is a very effective algorithm, which is often used to solve NoC mapping. Because of the isomorphism of network topology, it means that the information of the network will not be changed by rotating or flipping the network in GA. Because

of the isomorphism of the network, different mapping solutions may have the same network information, which leads to the genetic algorithm isomorphism problem. Has been looked at in some previous studies. In the literature [13], the isomorphism of GA is first proposed, and the isomorphism elimination in the enhancement design of analog circuits is emphasized. Literature [14] proposed the phenomenon of isomorphism of NoC mapping solutions and proposed the use of a density direction transformation algorithm to eliminate the isomorphism of GA mapping solutions and accelerate the convergence speed of the population.

However, previous work mainly considered the possible negative effects of isomorphism solution on GA, but did not consider the introduction of isomorphism research into the hyperheuristic algorithm that is more suitable for NoC mapping and did not consider the possible positive effects of isomorphism solution during algorithm iteration.

Therefore, this paper establishes the evaluation model of NoC, such as delay and power consumption, and obtains the accurate multi-objective mapping strategy of NoC by using a genetic-based hyper-heuristic algorithm (GHH). Based on the working mechanism of crossover operators in genetic algorithm, the concept of isomorphism solution (IS) was proposed to optimize the algorithm by analyzing the generation principle of invalid crossover during the algorithm iteration. The isomorphic replacement (IRC) strategy was adopted to eliminate invalid iterations in the iteration process of GHH. At the same time, an adaptive mechanism is introduced to improve the usage frequency of IRC operator to prevent the algorithm from falling into evolutionary stagnation in the late iteration. Compared with the GHH without IRC, the GHH with IRC has a faster convergence speed and can obtain better solutions.

The main contributions of this paper are as follows:

- (1) We proposed an optimization solution of NoC-specific application mapping based on genetic-based hyper-heuristic algorithm (GHH). Compared with the traditional GA, GHH has an available algorithm to choose its own pool of operators as well as a set of automatic feedback based on the current iteration state incentives; the algorithm can according to the specific mission requirements of the application of independent choice suitable operator, a new algorithm for optimization of this will be more able to adapt to specific NoC application tasks. We construct the fitness function of the GHH, design the relevant cross-mutation operator, and design the corresponding reward function. We hope that through the hyperheuristic algorithm, the algorithm can select operators more suitable for the current stage in different stages.
- (2) We proposed an isomorphic replacement strategy for NoC mappings. We studied the characteristics of network mapping problem and puts forward the concept of isomorphic solution; we believe that based on symmetry and rotating form an isomorphic solution with the same fitness, this solution can be innovative for late to introduce new operator-isomorphic replacement operator, to improve the ineffective crossover in the late iterations, in order to make the isomorphism replacement operators (IRC) play the key role at the last stage of the optimization iterations. We also improve the reward function so that isomorphic replacement operators can automatically be massively selected from the operator pool in a suitable situation.

This paper is organized as follows. Section 2.1 covers NoC mappings and their terminologies. Section 2.2 introduces the invalid crossover and isomorphic replacement (IRC) policy in detail. Section 3 details the experimental results based on the above model and optimization method. Section 4 makes a conclusion.

2. Materials and Methods

2.1. NoC Mapping and Corresponding Terminologies

2.1.1. ARCG and APCG

Architecture characteristic graph (ARCG): NoC architecture is modeled as a directed graph $G(R, L)$. The nodes in the graph represent router $R_i \in R$, and the edge $l_{i,j} \in L$ between each routing node represents the physical link between routers. The bandwidth of the link

between router R_i and R_j , namely the weight of link $l_{i,j}$, is represented by $B(l_{i,j})$. Each routing node will also connect to a placed IP core $P_i \in P$ through the network interface NI, and we will map the application (AP) at P .

Application characteristic graph (APCG) is a directed graph $G(C, A)$ based on the communication requirements between applications. As shown in Figure 1, where each vertex $C_i \in C$ represents an intellectual property (IP) core, each edge $A_{i,j} \in A$ represents the communication between C_i and C_j , and the weight of each edge $V_{i,j}$ represents the traffic on edge $A_{i,j}$.

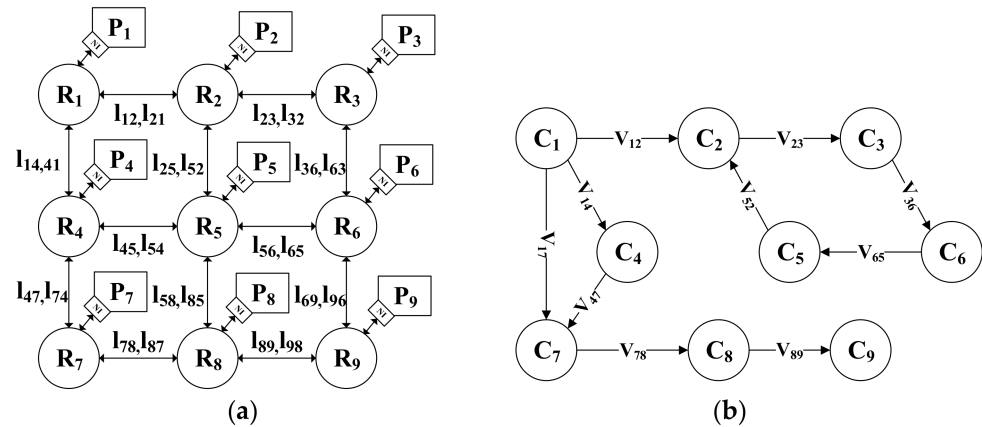


Figure 1. ARCG and APCG. (a) ARCG for 3 × 3 Mesh NoC, (b) APCG for 9 cores application.

2.1.2. Evaluation Model Construction

The energy of NoC consists of static energy and dynamic energy. Communication energy accounts for 28% of the total NoC energy and can even exceed 40% for most multimedia applications [15]. Therefore, in this paper, we focus on communication energy. The establishment of the energy assessment model E_C is shown as follows [16].

$$E_{ij} = H \times E_L + (H + 1) \times E_R \quad (1)$$

where E_{ij} is the communication energy from router R_i to R_j , E_L and E_R represent the energy consumed by transmitting one bit of data through the link and router, respectively, and H is the hop number of messages from router R_i to R_j . The total communication energy E_C is calculated from Equation (2).

$$E_C = \sum_{i=1}^{N_{core}} \sum_{j=1}^{N_{core}} V_{ij} \times E_{ij} \quad (2)$$

where V_{ij} is the number of bits to be transmitted from router R_i to router R_j . According to [17], we set the potential energy values of link, switch and read/write buffer as 0.449, 0.284, 1.056 and 2.831 pJ, respectively. Average network delay (T_{av}) of NoC is estimated based on Equation (3).

$$T_{av} = \frac{T_L + T_R}{\sum_{i=1}^{N_{core}} \sum_{j=1}^{N_{core}} \lambda_{i,j}} \quad (3)$$

where T_L is the delay of link, T_R is the delay of router, $\lambda_{i,j}$ represents the number of flits transmitted from C_i to C_j , and N_{core} represents the number of IP cores, which is also equal to the number of routers. At the system level, the time consumed by global links of NoC topology is modeled as Equation (4).

$$T_L = \sum_{i=1}^{N_{core}} \sum_{j=1}^{N_{core}} \lambda_{i,j} C_{i,j} u_t \quad (4)$$

where i and j are indexes of the target node and source node, respectively, $C_{i,j}$ is the number of links needed to transmit these packets from source to destination when implemented

in a topology, and u_t represents unit time quantity. The total delay modeling of router consumption is shown in Equation (5).

$$T_R = \sum_{i=1}^{N_{core}} \sum_{j=1}^{N_{core}} (\lambda_{i,j} C_{i,j} T_{Router}^{i,j}) \quad (5)$$

where $T_{Router}^{i,j}$ is the sum of delays of all routers when the core C_i communicates with core C_j . $T_{Router}^{i,j}$ is calculated based on the architecture model of queued router output in this paper.

2.1.3. Genetic-Based Hyper-Heuristic Algorithm (GHH)

In this paper, we proposed a genetic-based hyper-heuristic algorithm (GHH) for NoC mapping. GHH is divided into two levels, as shown in Figure 2. At the bottom level, the operator pool is formed by the different crossover and mutation operators of the genetic algorithm, and the optimization problem evaluation function is modeled to complete the problem representation, and the initial solution of the algorithm is determined. The algorithm selects operators in the operator pool to form different genetic algorithms (GA) at a top level. There is domain shielding between the top level and bottom level, so GA is generated by observing the evaluation function of different operators under the current problem and adjusting the weights used by operators according to the set reward function. In this paper, the initialization solutions of GHH are generated by competitive bidding. The operator pool and reward mechanism are the key to GHH, which will be introduced in detail in the following chapters.

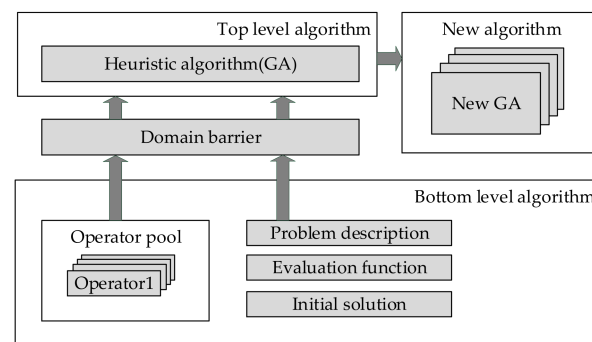


Figure 2. Hyper genetic algorithm.

In this paper, we focus on the optimization of NoC on the energy and delay. We introduce the weight $\alpha \in [0, 1]$ to characterize the varying importance of energy and delay in mapping APCG to different systems. The total cost function of the mapping mode is shown in Equation (6). Where $N(E_c)$ and $N(T_{av})$ are normalized energy and delay, respectively. In this paper, α was set as 1 to optimize the communication energy of NoC, α was set as 0 to optimize the communication delay of NoC, and α was set as 0.5 to optimize the communication energy and delay of NoC at the same time.

$$F_{co}(E_c, T_{av}) = \alpha N(E_c) + (1 - \alpha) N(T_{av}) \quad (6)$$

2.2. Optimization Strategy

In this chapter, we will focus on the common problem of invalid crossover and evolutionary stagnation in the late iteration of the traditional GA algorithm, and how we use the isomorphism of mapping solutions to construct a new operator based on a genetic-based hyper-heuristic algorithm to solve this problem.

2.2.1. Invalid Crossover and Isomorphic Genes

In GHH, the population's genetic diversity is high at the early stage of algorithm iteration. With the iteration of the algorithm increasing, the fitness function and roulette

selection rules will make those genes with high fitness dominant and eliminate the survival of the gene with low fitness. However, the rapid increase of the proportion of a dominant gene will lead to the decline of the genetic diversity of the total population, which may make GHH fall into the local optimal solution.

In the late iterative process of GHH, there are a large number of the same genes, which makes invalid crossover occur frequently. As shown in Figure 3, the parent genes P1 and P2 are exactly the same. Under the action of the crossover operator, the crossover of some fragments occurs, but the offspring O produced is no different from the parent. Therefore, such a crossover will not contribute to the evolution of the population, and such crossover is called “invalid crossover”.

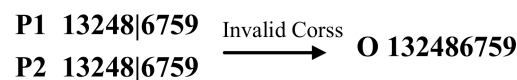


Figure 3. Invalid crossover due to the same parent genes.

Due to the convergence of GHH, a large number of the same genes in the late iteration algorithm is inevitable. We introduce the concept of “isomorphism gene” to increase the gene diversity without changing the current iteration of population fitness distribution as far as possible.

In the iterative process of GHH, there are some special genes that have the same fitness for the studied evaluation function and show symmetry in the mapping position. We call such genes “isomorphic genes”. Figure 4 shows four cases of 9 core APCG mapping in 3×3 2D Mesh, in which isomorphic genes (b–d) are obtained from the basic gene (a) through mirror symmetry and center symmetry, in addition, more isomorphic genes will be generated by centering rotation of (a–d). Under the dimension routing algorithm, the mapping results of these genes have the same fitness of power consumption and delay.

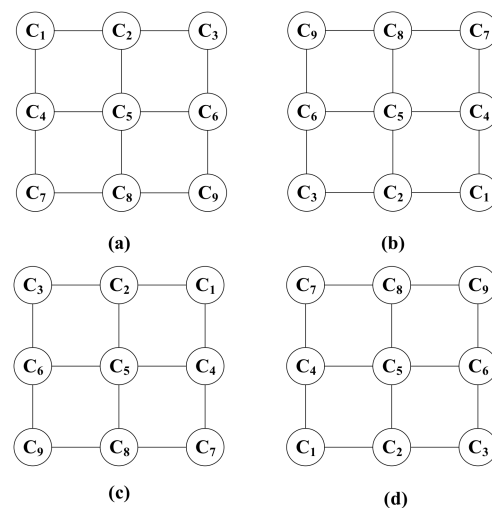


Figure 4. NoC mapping with isomorphic genes. (a) Original genetic, (b) isomorphic genes by 180 degree rotation, (c) isomorphic genes by horizontal flip, (d) isomorphic genes by 90 degree rotation.

Using isomorphic genes to replace the offspring of the invalid crossover to increase the population diversity with the fitness of the population unchanged. As shown in Figure 5, invalid crossover occurs when the gene of P1 and P2 are the same. At this time, we replace the offspring of invalid crossover with any isomorphic solution, which is called isomorphism replacement crossover (IRC). The participation of isomorphic genes in subsequent iterations will result in more evolutionary possibilities, thus giving the population a greater chance to find a global optimal solution. At the same time, IRC reduces the waste of computing resources caused by “invalid crossover”.

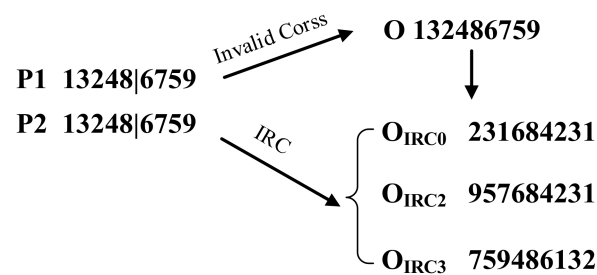


Figure 5. Isomorphism replacement crossover (IRC).

2.2.2. Isomorphism Replacement Crossover (IRC) Operator

In order to improve the original GHH, we introduce a new crossover operator, the isomorphism replacement crossover (IRC) operator in the operator pool of GHH, whose pseudo-code is as follows (Algorithm 1):

Algorithm 1. Isomorphism Replacement Crossover (IRC) Operator.

G_p : paternal genes, G_{off} : offspring genes, $TC\{\}$: typical crossover strategy, $IRC\{\}$: isomorphic replacement crossover strategy, $Size_{pop}$: size of the population, P_{cross} : crossover probability, S : similarity of genes, S_{th} : similarity threshold, SM: selection mode of isomorphic genes

Input: G_p , $TC\{\}$, $IRC\{\}$, $Size_{pop}$, P_{cross}

Output: G_{off} :

```

01: For i = 1 to  $Size_{pop}$ 
02:   If (rand() <  $P_{cross}$ )
03:     Calculate the Similarity of the paternal genes  $S$ 
04:     If ( $S < S_{th}$ )
05:        $G_{off} = TC\{G_p\}$ 
06:     Else
07:       SM = rand();
08:        $G_{off} = IRC\{G_p, SM\}$ ;
09:     Endif
10:   Endif
11: EndFor

```

Firstly, we need to calculate the similarity of the paternal genes to be crossed. We will count the consistency of each position of the two paternal genes and return the results to a consistency index similarity. If the similarity is greater than or equal to the similarity threshold S_{th} , the crossover is considered an invalid crossover. In this paper, the threshold S_{th} is set to the length of the genes. For invalid crossovers, the algorithm will generate the isomorphic gene based on the parental genes and randomly select one of them to replace the offspring generated by the invalid crossover. IRC will be added to the operator pool of GHH to improve the performance of GHH. Although “isomorphic replacement” increases population diversity without changing the current fitness distribution, it may slow down the convergence speed due to the increased population diversity. To reduce the negative impact of convergence speed caused by IRC, we will introduce some adaptive mechanisms to dynamically adjust the evolution rate of the population.

2.2.3. Optimization of Reward Mechanism

The “reward” mechanism in GHH is responsible for selecting the appropriate operator during each iteration of optimization. Referred to [15], the “reward” mechanism scores operators based on two different metrics. F_1 records the previous performance of each operator, as shown in Equation (7). F_i is the value of the objective function $F_{co}(E_c, T_{av})$ at the n -th iteration of individual i . $F_i(n-1)$ is the value of the objective function $F_{co}(E_c, T_{av})$ at the $(n-1)$ -th iteration of individual i . F_2 corresponds to the elapsed time since each

operator was last selected, as shown in Equation (8). t'_{op} represents the last time when the operator was selected, and t_{op} represents the current time when the operator was selected.

$$F_1 = \frac{\max(F_i(n)) - \max(F_i(n-1))}{\max(F_i(n))} \quad (7)$$

$$F_2 = \frac{1}{t_{op} - t'_{op}} \quad (8)$$

According to the combination of F_1 and F_2 , we propose the “reward” function $F_{Re}(n)$, as shown in Equation (9). Where, N is the total number of iterations, and n is the number of iterations completed. In this paper, β is set to 100. We use a nonlinear increasing function $e^{1/(N+1-n)}$ to increase the weight of the “reward” as the number of iterations increases. According to the “reward” mechanism, all genetic operators are assigned to “1” as the initial weight and change the weight of the operator immediately after selection.

$$F_{Re} = \begin{cases} F_{Re}(n-1) + \beta e^{1/(N+1-n)} \times F_1 \times F_2, & F_1 > 0 \\ F_{Re}(n-1), & F_1 \leq 0 \end{cases} \quad (9)$$

In the GHH, we introduce several crossover operators and mutation operators to form the operator pool and utilize the reward function to make the algorithm judge the merits of the operator by itself and make the dominant operator have a greater chance to participate in the subsequent iterations through roulette selection. However, in the GHH, a certain crossover operator may be heavily used because of the reward mechanism, resulting in no chance for IRC crossover to be used at the late iteration. The common crossover operators almost do not need to work anymore when the dominant genes are stable in the late iterations. Therefore, we design a mechanism to make the algorithm use IRC crossover operators to generate more isomorphic solutions in the late iteration, which may make the algorithm jump out of the local optimal solution to get a better solution.

Considering the randomness of crossover and mutation, it is not unwise to begin to use the IRC operator at a specific iteration. A simple idea is to judge whether the optimal solution is no longer updated in the long term, which is called evolutionary stagnation. An “evolutionary stagnation counter” (ES_cnt) is used to count the iterations of the evolutionary stagnation. If it exceeds the evolution stagnation threshold, the ISP crossover operator is used to replace the common crossover operators. In this paper, the evolution stagnation threshold is set to 150.

3. Experimental Results and Analysis

In this section, we present the evaluation of the proposed GHH. The MWD [18], MPEG-4 [19], VOPD [19], and RAND25 which is generated by Task Graph For Free (TGFF) [20] is applied as target application to test the efficiency of our proposed algorithm. The topology of NoC is 2D Mesh and the routing algorithm is the dimensional routing algorithm. The GHH operator pool consists of four crossover operators and one mutation operator. The crossover operator consists of two conventional operators and their corresponding versions of IRC operators, which are used to study the influence of IRC operators on the algorithm iteration process. The parameters of GHH is shown in Table 1. All experiments are implemented by MATLAB R2020a.

Table 1. Parameters of GHH.

Parameters	Value
The population size	100
The number of iterations	2000
Initialization mode	Tournament
The mutation rate	0.1
Initial crossover rate	0.2
Crossover operators	Discrete recombination Single point recombination IRC-discrete recombination IRC-single point recombination
Mutation operators	Random mutation

3.1. Invalid Crossover and Optimal Crossover Rate

In order to verify the influence of the IRC operator on the iteration process of GHH, we selected VOPD as the target application and 4×4 as the size of NoC to conduct experiments at different crossover rates. The optimization factor α is 0.5. We also counted the invalid crossover rate (the rate of actual invalid crossover to the number of ideal crossovers) generated by each generation population in the iteration process before and after the introduction of IRC operator. At the same time, the algorithm convergence process under corresponding conditions was recorded. All experiments were repeated 10 times and then the average results are shown in Figures 6 and 7. Figure 6 shows the invalid crossover rate before and after the introduction of IRC operator. Figure 7 shows the mapping results before and after the introduction of IRC operator.

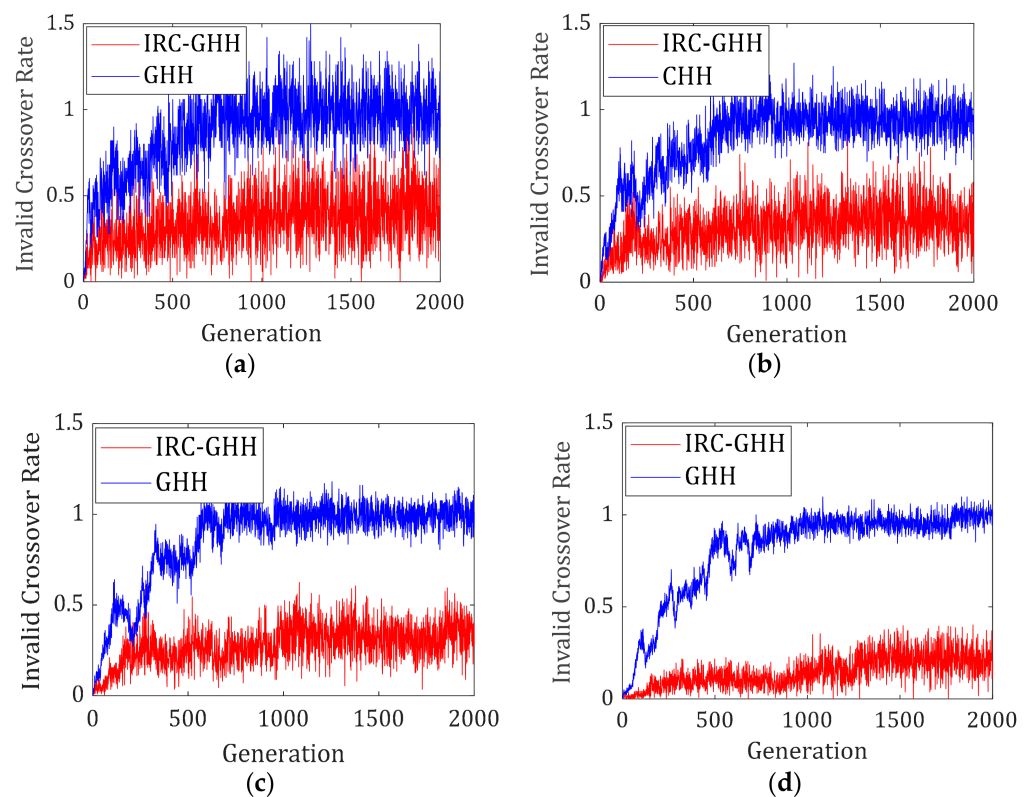


Figure 6. The invalid crossover rate before and after the introduction of the IRC operator. (a) $P_c = 0.10$, (b) $P_c = 0.20$, (c) $P_c = 0.60$, (d) $P_c = 0.80$.

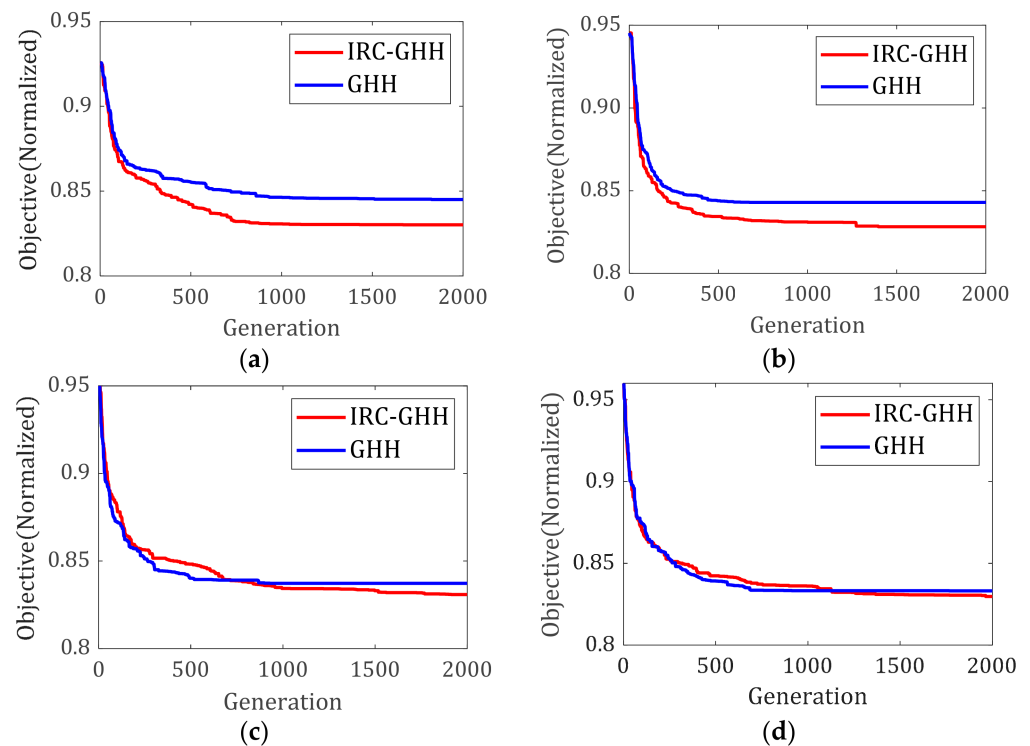


Figure 7. Mapping results before and after the introduction of the IRC operator. (a) $P_c = 0.10$, (b) $P_c = 0.20$, (c) $P_c = 0.40$, (d) $P_c = 0.80$.

It can be seen from Figure 6 that the invalid crossover rate increases with the number of iterations for the GHH without IRC operators. And the invalid crossover rate is stable at 100% when the number of iterations reaches 1000. For the GHH with IRC operators (IRC-GHH), invalid crossovers are reduced significantly, and with the increasing crossover rate, the reduction of invalid crosses is more and more obvious. Meanwhile, we also noticed from Figure 7 that there was not a huge difference between GHH and IRC-GHH in the early stage of algorithm iteration. But when the convergence speed of the GHH algorithm began to decline significantly, IRC-GHH began to show its advantages and could always obtain a better optimal solution. In addition, we can also see from the experiment results that too large or too small a crossover rate will affect the performance of IRC-GHH. In this paper, we select 0.2 as the crossover rate.

3.2. Performance of GHH

To compare the performance of IRC-GHH and GHH, we select three different weights of cost function $\alpha = 1$ (100% power), $\alpha = 0.5$ (50% power and 50% latency), and $\alpha = 0$ (100% latency) respectively. Considering the randomness of heuristic algorithms, each experiment is repeated 10 times. As shown in Figure 8, the solid line is the average results and the shadow is the fluctuation range of the algorithms. The average results of the random mapping are used as the baseline in the experiment.

As can be seen from Figure 8, IRC-GHH can always get a better solution under the different weight of cost functions, and the average communication energy and average communication delay are reduced by 7.84% ($\alpha = 0$) and 15.25% ($\alpha = 1$), respectively, compared with GHH algorithm. IRC-GHH reduces the ratio of communication energy and communication delay iteration by 9.60% ($\alpha = 0.5$) on average compared with the GHH algorithm. In addition, when the number of iterations of IRC-GHH is greater than 1000, the optimal solution of the algorithm is still in a downward trend, which indicates that the IRC operator and adaptive mechanism still work in the late iteration, so that the convergence speed of IRC-GHH is significantly faster than that of GHH in the late iteration. It is proved

that IRC-GHH can effectively jump out of local optimal solution after introducing the IRC operator and adaptive mechanism.

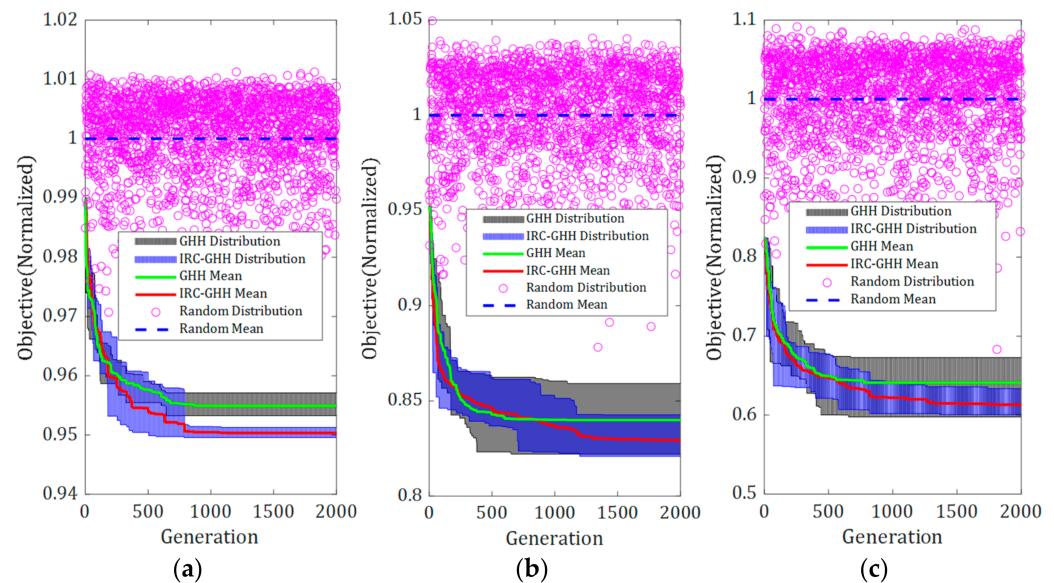


Figure 8. Performance differences compared to IRC-GHH versus GHH. (a) $\alpha = 0$, (b) $\alpha = 0.5$, (c) $\alpha = 1$.

3.3. Other Case Studies

In this section, VOPD, MWD, MPEG-4, and RAND25 were selected as APCG to verify our proposed algorithm further. The NoC mapping based on simulated annealing algorithm (SA), particle swarm optimization algorithm (PSO), GA, and IRC-GHH were carried out, respectively. A 2D Mesh was used for ARCG and the dimension routing algorithm was used for the routing algorithm. For algorithm initialization, an initial solution was selected by the tournament algorithm. The number of iterations was set to 2000. The average mapping results are shown in Table 2, which indicates that IRC-GHH outperforms other algorithms for different applications.

Table 2. Mapping results for different algorithms.

APCG	IP	Delay ($\alpha = 0$)				Power ($\alpha = 1$)				Overall Cost ($\alpha = 0.5$)			
		SA	PSO	GA	IRC-GHH	SA	PSO	GA	IRC-GHH	SA	PSO	GA	IRC-GHH
MWD	12	0.970	0.964	0.965	0.961	0.888	0.888	0.866	0.855	0.723	0.757	0.706	0.644
MPEG-4	12	0.966	0.963	0.964	0.962	0.880	0.878	0.875	0.861	0.767	0.729	0.697	0.679
VOPD	16	0.964	0.962	0.954	0.951	0.889	0.851	0.840	0.829	0.721	0.722	0.641	0.613
RAND25	25	0.994	0.993	0.984	0.983	0.982	0.976	0.949	0.948	0.950	0.953	0.903	0.894

4. Conclusions

In this paper, the symmetry of NoC topology is analyzed, and isomorphism solution and invalid crossover are studied. We proposed a generic-based hyper-heuristic (GHH) algorithm for NoC mapping. To eliminate invalid crossovers, the isomorphic replacement crossover (IRC) strategy is applied to increase population diversity, and the adaptive mechanism is introduced to prevent the algorithm from falling into evolutionary stagnation in the late iteration. We test our proposed algorithm IRC-GHH from different views and the experimental results show that IRC-GHH has a faster convergence speed and can always obtain a better solution in the same number of iterations.

Author Contributions: Methodology, C.X., J.N. and M.L.; writing—review and editing, C.X. and J.N., investigation, J.N. and D.C.; data curation, Y.L., X.L. and Y.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China Youth fund under Grant 62004146, by the China Postdoctoral Science Foundation funded project under Grant 2021M692498, the Opening Project of Science and Technology on Reliability Physics and Application Technology of Electronic Component Laboratory under Grant ZHD202006 and the Natural Science Foundation of Guangdong, China under Grant 2021A1515012293.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sodani, A.; Gramunt, R.; Corbal, J.; Kim, H.; Vinod, K.; Chinthamani, S.; Hutsell, S.; Agarwal, R.; Liu, Y. Knights Landing: Second-Generation Intel Xeon Phi Product. *IEEE Micro* **2016**, *36*, 34–46. [\[CrossRef\]](#)
2. Gangwar, A.; Xu, Z.; Agarwal, N.K.; Sreedharan, R.; Prasad, A. Traffic Driven Automated Synthesis of Network-on-Chip from Physically Aware Behavioral Specification. In Proceedings of the 2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Miami, FL, USA, 19 September 2019; pp. 122–127. [\[CrossRef\]](#)
3. Liao, W.; Deng, H.; Luo, Y.; Xiao, S.; Li, C.; Yu, Z. An Efficient and Low-Overhead Chip-to-Chip Interconnect Protocol Design for NOC. In Proceedings of the 2019 IEEE International Conference on Integrated Circuits, Technologies and Applications (ICTA), Chengdu, China, 13–15 November 2019; pp. 77–78. [\[CrossRef\]](#)
4. Sorin, D.J.; Hill, M.D.; Wood, D.A. A Primer on Memory Consistency and Cache Coherence. *Morgan Claypool* **2011**, *6*, 1–212. [\[CrossRef\]](#)
5. Tayu, S.; Ueno, S. A note on the energy-aware mapping for NoCs. In Proceedings of the 2014 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), Ishigaki, Japan, 17–20 November 2014; pp. 647–650. [\[CrossRef\]](#)
6. Zhong, L.; Sheng, J.; Jing, M.; Yu, Z.; Zeng, X.; Zhou, D. An optimized mapping algorithm based on Simulated Annealing for regular NoC architecture. In Proceedings of the 2011 9th IEEE International Conference on ASIC, Xiamen, China, 25–28 October 2011; pp. 389–392. [\[CrossRef\]](#)
7. Wu, C.; Deng, C.; Liu, L.; Han, J.; Chen, J.; Yin, S.; Wei, S. A Multi-Objective Model Oriented Mapping Approach for NoC-based Computing Systems. *IEEE Trans. Parallel Distrib. Syst.* **2017**, *28*, 662–676. [\[CrossRef\]](#)
8. Zhang, L.; Li, S.; Qu, L.; Kang, Z.; Wang, S.; Chen, J.; Wang, L. MAMAP: Congestion Relieved Memetic Algorithm based Mapping Method for Mapping Large-Scale SNNs onto NoC-based Neuromorphic Hardware. In Proceedings of the 2020 IEEE 22nd International Conference on High Performance Computing and Communications, IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS). Yanuca Island, Cuvu, Fiji, 14–16 December 2020; pp. 640–647. [\[CrossRef\]](#)
9. Bhanu, P.V.; Govindan, R.; Kattamuri, P.; Soumya, J.; Cenkeramaddi, L.R. Flexible Spare Core Placement in Torus Topology Based NoCs and Its Validation on an FPGA. *IEEE Access* **2021**, *9*, 45935–45954. [\[CrossRef\]](#)
10. Kullu, P.; Tosun, S. MARM-GA: Mapping Applications to Reconfigurable Mesh using Genetic Algorithm. In Proceedings of the 2019 22nd Euromicro Conference on Digital System Design (DSD), Kallithea, Greece, 28–30 August 2019; pp. 13–18. [\[CrossRef\]](#)
11. Rocha, H.M.G.d.; Beck, A.C.S.; Maia, S.M.D.M.; Kreutz, M.E.; Pereira, M.M. A Routing based Genetic Algorithm for Task Mapping on MPSoC. In Proceedings of the 2020 X Brazilian Symposium on Computing Systems Engineering (SBESC), Florianopolis, Brazil, 24–27 November 2020; pp. 1–8. [\[CrossRef\]](#)
12. Amin, W.; Hussain, F.; Anjum, S. iHPSA: An improved bio-inspired hybrid optimization algorithm for task mapping in Network on Chip. *Microprocess. Microsyst.* **2022**, *90*, 104493. [\[CrossRef\]](#)
13. Khalifa, Y.M.A. Isomorphism elimination to enhanced design centering of analog circuits using GA and the regionalization method. In Proceedings of the 10th IEEE International Conference on Electronics, Circuits and Systems, 2003, ICECS 2003 Proceedings of the 2003. Sharjah, United Arab Emirates, 14–17 December 2003; Volume 3, pp. 1058–1061. [\[CrossRef\]](#)
14. Weng, X.; Liu, Y.; Yang, Y. Network-on-chip heuristic mapping algorithm based on isomorphism elimination for NoC optimisation. *IET Comput. Digit. Tech.* **2020**, *14*, 272–280.
15. Xu, C.; Yi, L.; Zhu, Z.; Yang, Y.T. An efficient energy and thermal-aware mapping for regular network-on-chip. *leice Electron. Express* **2017**, *14*, 20170769. [\[CrossRef\]](#)
16. Elmiligi, H.; El-Kharashi, M.W.; Gebali, F. A Delay Model for Networks-on-Chip Output-Queueing Router. In Proceedings of the 2006 6th International Workshop on System on Chip for Real Time Applications, Cairo, Egypt, 27–29 December 2006; pp. 95–98. [\[CrossRef\]](#)

17. Xu, C.; Yi, L.; Peng, L.; Yang, Y.T. Unified multi-objective mapping for network-on-chip using genetic based hyper-heuristic algorithms. *IET Comput. Digit. Tech.* **2018**, *12*, 158–166. [[CrossRef](#)]
18. Sahu, P.K.; Chattopadhyay, S. A survey on application mapping strategies for Network-on-Chip design. *J. Syst. Archit.* **2013**, *59*, 60–76. [[CrossRef](#)]
19. Amin, W.; Hussain, F.; Anjum, S.; Khan, S.; Baloch, N.K.; Nain, Z.; Kim, S.W. Performance Evaluation of Application Mapping Approaches for Network-on-Chip Designs. *IEEE Access* **2020**, *8*, 63607–63631. [[CrossRef](#)]
20. Dick, R.P.; Rhodes, D.L.; Wolf, W. TGFF: Task graphs for free. In Proceedings of the Sixth International Workshop on Hardware/Software Codesign. (CODES/CASHE'98), Seattle, WA, USA, 18 March 1998; pp. 97–101. [[CrossRef](#)]