Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Performance-driven assignment and mapping for reliable networks-on-chips*#

Qian-qi LE†[1,2], Guo-wu YANG[1], William N. N. HUNG[3], Xiao-yu SONG[4], Fu-you FAN[1]

([1]*School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China*)

([2]*Department of Information and Computing Science, Chengdu University of Technology, Chengdu 610059, China*)

([3]*Synopsys Inc., Mountain View, CA 94040, USA*)

([4]*Department of Electronic and Computer Engineering, Portland State University, Portland, OR 97207-0751, USA*)

†E-mail: leqqi777@163.com

Received Feb. 19, 2014; Revision accepted Sept. 2, 2014; Crosschecked Oct. 22, 2014

Abstract: Network-on-chip (NoC) communication architectures present promising solutions for scalable communication requests in large system-on-chip (SoC) designs. Intellectual property (IP) core assignment and mapping are two key steps in NoC design, significantly affecting the quality of NoC systems. Both are NP-hard problems, so it is necessary to apply intelligent algorithms. In this paper, we propose improved intelligent algorithms for NoC assignment and mapping to overcome the drawbacks of traditional intelligent algorithms. The aim of our proposed algorithms is to minimize power consumption, time, area, and load balance. This work involves multiple conflicting objectives, so we combine multiple objective optimization with intelligent algorithms. In addition, we design a fault-tolerant routing algorithm and take account of reliability using comprehensive performance indices. The proposed algorithms were implemented on embedded system synthesis benchmarks suite (E3S). Experimental results show the improved algorithms achieve good performance in NoC designs, with high reliability.

Key words: Network-on-chip (NoC), Mapping, Assignment, Reliability
doi:10.1631/jzus.C1400055 **Document code:** A **CLC number:** TP202; TN402

1 Introduction

The increasing number of components on a single chip leads to continuous saturation problems for buses on systems-on-chips (SoCs). Network-on-chip (NoC) is a new SoC paradigm for solving scalable communication requests, separating the processing units and communication infrastructures. In this way, complex communication problems between multi-processors are solved effectively (Bjerregaard and Mahadevan, 2006; Marculescu *et al.*, 2009).

An NoC application usually consists of a few subtasks. These subtasks are accomplished by a set of intellectual property (IP) cores. IP core assignment, mapping, and routing play key roles in the design and implementation of NoC platforms (Orgas *et al.*, 2005; Cheng *et al.*, 2011). The purpose of IP core assignment is to select proper IP cores for the subtasks. IP core mapping is used to arrange suitable locations for the IP cores in NoC topologies. They significantly affect the performance of NoC systems. Power consumption, delay, and area are important indices used to evaluate the quality of NoCs. Many studies have focused on this field (Tang and Kumar, 2003; Jena and Sharma, 2007; Sepulveda *et al.*, 2011). With the increasing scale of integrated circuits, more and more complex designs are integrated on-chip, where

* Project supported by the National Natural Science Foundation of China (Nos. 60973016 and 61272175), the National Basic Research Program (973) of China (No. 2010CB328004), the Youth Backbone Teacher Foundation of Chengdu University of Technology (No. JXGG201305), and the Bagui Scholarship Project, China

Electronic supplementary materials: The online version of this article (http://dx.doi.org/10.1631/jzus.C1400055) contains supplementary materials, which are available to authorized users

fault-tolerant communication and reliability are drawing increasing attention. The reliability of NoC architectures has been studied widely (Refan *et al.*, 2008; Yu and Ampadu, 2010).

In this paper, we take account of reliability using comprehensive performance indices in NoC design. We adopt different evaluation indices and models for assignment and mapping, in accordance with different application requirements. We also design a fault-tolerant routing to improve system reliability.

Both IP core assignment and mapping are highly complex problems. If there are N subtasks and M candidate IP cores, M^N assignment solutions exist. If there are M IP cores and N mapping locations, $N!/(N-M)!$ mapping designs exist. Both problems are NP-hard problems (Liu *et al.*, 2011). The size of the solution space increases quickly as the problem scales up. At present, the methods used usually obtain near optimal solutions by employing optimization algorithms. Tang and Kumar (2003) used a two-step genetic algorithm to optimize NoC layout. Hu and Marculescu (2003) used a branch and bound method. Jena and Sharma (2007) used the non-dominated sorting genetic algorithm version II (NSGA-II). Sepulveda *et al.* (2011) applied an immune algorithm (IA). da Silva *et al.* (2010) used NSGA-II and micro-genetic algorithm (micro-GA).

The algorithms mentioned above are traditional intelligent algorithms which generally have some drawbacks. For example, they easily converge on local optima. Furthermore, both IP core assignment and mapping have multiple optimization objectives. So, based on NoC application requirements and constraints, we have designed improved multi-objective optimization algorithms: PSOGA, PSOSA, and scatter search. PSOGA combines particle swarm optimization (PSO) with the genetic algorithm (GA), PSOSA combines PSO with simulated annealing (SA), and scatter search (SS) is efficient for searching for optimal solutions and is different from other intelligent algorithms (Rao and Arvind, 2005).

2 Preliminaries

2.1 Problem description

The purpose of IP core assignment is to select a set of IP cores according to every subtask type, from an IP core repository. IP core mapping is used to map the selected IP cores onto proper processing elements (PEs) in the NoC architecture based on communication traffic between subtasks. The process is shown in Fig. 1.

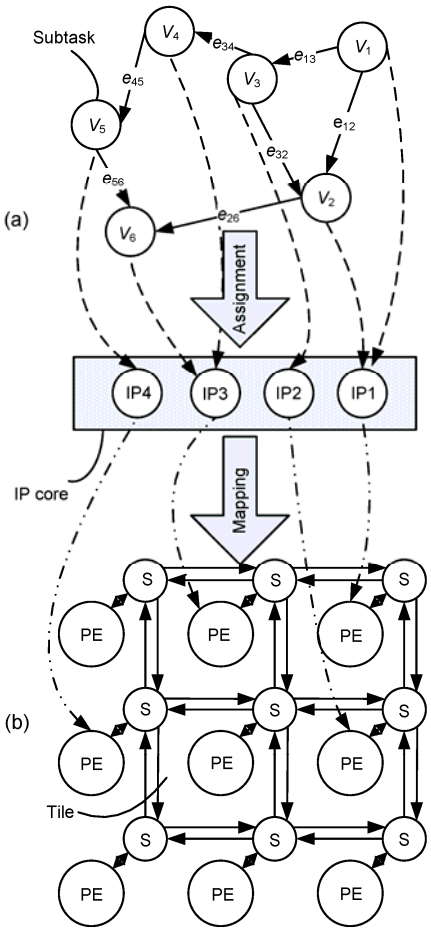

Fig. 1 Process of IP core assignment and mapping
(a) Task graph; (b) Tile-based mesh topology

We employ a task graph (TG) to describe subtasks and communication relationships for NoC applications. An example of TG is shown in Fig. 1a. It is a directed acyclic graph $G(V, E)$, where V represents the subtask set and E represents the set of connections between subtasks. A vertex $v_i \in V$ represents a subtask, and an edge $e_{ij} \in E$ represents the connection between subtasks i and j. The weight of e_{ij} represents the traffic between v_i and v_j.

Tile-based mesh topology (Fig. 1b) is used widely for its high bandwidth, simple structure, and convenience of implementation. A tile-based mesh

consists mainly of processing elements (PE) and switches (S).

2.2 Elitist non-dominated sorting

In practical applications of NoC, there are multiple optimization objectives such as delay, area, and power consumption. Some of these objectives conflict with each other, and no method can improve one objective without degrading another. In evaluation, if using single objective optimization, taking reliability as an example, the solutions with the highest reliability may need long communication times. Thus, the overall performance of the system is not high. So, we have adopted multiple objective optimization to realize good tradeoffs between multiple objectives.

Taking the mapping phase as an example, communication time, link load variance, and reliability are three optimization objectives. We want to acquire solutions with less communication time, small link load variance, and high reliability. We modify reliability to 1-reliability to change the maximization problem to a minimization problem. Suppose that there are five solutions, and that the three optimization objectives of the five solutions are denoted as: A(5, 9, 0.8), B(8, 5, 0.9), C(7, 9, 0.5), D(5, 4, 0.5), and E(5, 5, 0.4). In each solution, the digits from left to right represent, respectively, communication time, link load variance, and reliability. We omit the units to facilitate the analysis below.

For single objective optimization, supposing the objective is to minimize communication time, we just compare the communication time of the five solutions. The time values of solutions A, D, and E are the same, and are the minimum, so these three solutions are the optimal. But obviously in solution A, although the communication time is the lowest, the link load variance is too large and the reliability is too low; thus, the overall performance of the system is poor.

For multiple objective optimization, we compare all three optimization objectives of the five solutions and rank them using a non-dominated sorting method. According to the values of evaluation indices, each index value of the solutions D and E is less than or equal to the corresponding values of other solutions. Comparing solution D with E, we can observe that some index values of solution D are larger than those of E, but the others are smaller. So, solution D is ranked as 1. Among solutions A, B, and C, there is no solution whose three indices are all superior to those of the other solutions, so they are ranked as 2. From these evaluation results, we can see that the overall performance of the solutions of rank 1 is superior to that of rank 2. So, we make the solutions in rank 1 our first choice. The solutions in the same rank are provided together to deciders for reference.

Given the above, in evaluation, single objective optimization considers only one objective and cannot achieve overall performance optimization. Multiple objective optimization considers all objectives together as a whole to achieve overall performance goals. So, we need to coordinate each objective and search for Pareto optimal solutions.

Here, the solutions are sorted by an elitist non-dominated sorting method, and assigned a rank. Furthermore, we select a predetermined number of solutions into an elite archive. The elites are chosen based on their non-dominated rank. If candidate solutions belong to the same rank, the solutions are selected based on their crowding distances. The crowding distance of the ith solution is described as

$$\mathrm{crowd}_i = \sum_{m=1}^{N} \frac{\left| f_m(C_{i+1}) - f_m(C_{i-1}) \right|}{\left| f_m(C_{n-1}) - f_m(C_0) \right|}, \tag{1}$$

where C_i (i=0, 1, …, n−1) denotes the ith solution in a given non-dominated rank, $f_m(\cdot)$ is the evaluation function of the mth optimization objective, and N is the number of evaluation objectives. To avoid the impact of different dimensions of the multiple evaluation objectives, the distance between the (i−1)th and (i+1)th solutions is divided by the distance between the (n−1)th and 0th solutions.

3 Multiple objective optimization model

3.1 Evaluation models of IP core assignment

IP core assignment does not involve the layout of IP cores and communication between IP cores. So, the evaluation indices are computing power consumption, execution time, and area.

3.1.1 Computing power consumption

Computing power consumption is the power consumption when IP cores are executing the tasks:

$$\text{Power}_{\text{computing}} = \sum_{t \in \text{TG}} P_t, \tag{2}$$

where t represents subtask t in the task graph, and P_t is the power consumed when an IP core executes subtask t.

3.1.2 Execution time

The total execution time consists of critical task time and parallel task time. The critical tasks are the tasks in the longest-duration path through the task graph, and parallel tasks are the tasks which can be executed at the same time. If parallel tasks share the same IP core and some of them are the critical tasks, the parallel execution time is considered. The formula is given by

$$\text{Time}_{\text{execution}} = \sum_{t \in \text{critical-task}} \text{Time}_t + \sum_{p \notin \text{critical-task}} \text{TimePara}_p,$$

$$\text{TimePara}_p = \begin{cases} \text{Time}_p, & \text{if task } p \text{ is executed in parallel and shares the same IP core with critical tasks,} \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

where critical-task is the set of critical tasks, t is a critical task, and Time_t is the execution time of task t. p represents a task which does not belong to the critical task set and Time_p is the execution time of task p.

3.1.3 IP core area

The formula for the IP core area is given by

$$\text{Area}_{\text{computing}} = \sum_{i=1}^{N} A_i, \tag{4}$$

where A_i is the area of IP core i and N is the number of IP cores in a given NoC assignment.

3.2 Evaluation models of IP core mapping

3.2.1 Reliability

When a switch fault occurs, the IP core linked with the faulty switch cannot communicate with other IP cores, which affects the entire system communication. Thus, the system reliability decreases. We propose a fault-tolerant mechanism which can recover the communication and balance the load. We first build several redundant paths by replacing the faulty switch with its neighboring switches. To ensure system performance, we evaluate the performance of redundant paths and select the best one. In this way, communication is restored with good performance. In our algorithm, we divide switch faults into end- and mid-switch faults according to the position of the faulty switch. On this basis, we design a rerouting algorithm which can automatically switch between *X*-*Y* and *Y*-*X* routings, aiming to avoid deadlock and obtain the shortest path. The rerouting decision tree is shown in Fig. 2.

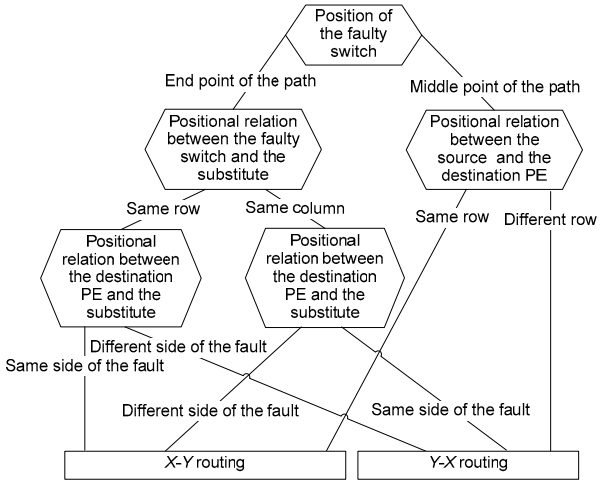

Fig. 2 Decision tree of the rerouting algorithm

The reliability model in this paper is traditional and is based on probabilistic methods. According to the reliability of the system components, the system overall reliability is computed using probabilistic methods. The reliability of an NoC system is equal to the product of the reliabilities of all the communication paths. The reliability of a path between a pair of IP cores is computed by the product of the probabilities of all the switches working properly in the path:

$$R_{\text{aPath}} = \prod_{i=1}^{N} R_i, \tag{5}$$

where R_i denotes the probability of the ith switch working properly, and N is the number of switches in this path.

On this basis, redundant structures are introduced. We consider both the shortest and redundant

paths. If the switches in the shortest path fail to work properly, we replace the shortest path with a redundant path which can bypass the faulty switch between this pair of IP cores. The reliability of the paths between the pairs of communication IP cores is

$$R_{\text{paths}} = \prod_{i=1}^{N} R_i + (1 - R_j) \prod_{i=1, i \neq j}^{M} R_i, \tag{6}$$

where the first term represents the shortest path with all switches working properly, and the second term represents a redundant path bypassing the jth faulty switch. N and M, respectively, represent the numbers of switches in the shortest path and redundant path. This model is similar to the one proposed by Refan *et al*. (2008).

For example, there are two communication paths between IP cores 1 and 5, denoted as (1, 3, 5) and (1, 2, 4, 5). At first we select the shortest path (1, 3, 5). If switch 3 breaks down, we will use the redundant path (1, 2, 4, 5). So, the reliability of the path between IP cores 1 and 5 is equal to $R_1R_3R_5+R_1R_2R_4R_5(1-R_3)$.

The reliability model in this paper is based on the above description. Therefore, the computation of reliability is related to the shortest and redundant communication paths, and obviously the communication path is related to the IP core mapping locations and routing methods. If there are multiple redundant paths, we select the one with the best performance which has lower communication power consumption and communication time. Thus, system reliability is improved, and at the same time, system performance is taken into account.

The reliabilities of NoCs involve many issues. We focus on the optimal design reliability, which is the premise and basis for ensuring system reliability. The problem we expect to solve is improving the connectedness probability of the surviving NoCs in the presence of failures. This problem is important in an NoC with nodes having fault probability. We expect to improve system reliability using optimal topologies and redundant paths to a certain extent. The topologies are the bases of reliability and the redundant paths are important means for achieving it.

3.2.2 Communication load balance

Balancing link load helps to relieve congestion points and queuing delay. It is evaluated by

$$\text{Load} = \frac{1}{L} \sum_{i=1}^{L} \left(\text{load}_i - \frac{1}{L} \sum_{i=1}^{L} \text{load}_i \right)^2, \tag{7}$$

where load_i is the ith link load, and L is the number of communication links in the mapping layout. A smaller variance means a more balanced load.

3.2.3 Communication power consumption

We sum up the power consumption of each communication IP core. The power consumption of a pair of IP cores can be computed by

$$\text{Power}_{\text{comm}}^{i,j} = P_{\text{link}} \text{hop}_{i,j} + (\text{hop}_{i,j} + 1) P_{\text{switch}}, \tag{8}$$

where P_{link} and P_{switch} are the power consumed through a link and a switch, respectively. These parameters were provided by Muralimanohar *et al.* (2007) and Das *et al.* (2009). $\text{hop}_{i,j}$ is the number of links between IP cores i and j.

3.2.4 Communication time

Because several parallel tasks may share the same link, additional time should be considered for the congestion caused by parallel subtasks. So, we build the communication time model as

$$\text{Time}_{\text{comm}} = T_{\text{para}} + \sum_{i,j \in \text{Critpath}} \left[T_{\text{link}} \text{hop}_{i,j} + (\text{hop}_{i,j} + 1) T_{\text{switch}} \right], \tag{9}$$

where i and j denote the IP cores which execute the critical subtasks. T_{link} is the time spent on a link, as proposed by Muralimanohar *et al.* (2007) and Das *et al.* (2009). T_{switch} is the queuing time in switch buffers, computed using the method proposed by Saxena *et al.* (2003). T_{para} is the additional time for parallel subtasks when they share the same link. Critpath is the set of critical paths.

4 Intelligent algorithm design

Based on the prescribed NoC design and platform constraints, we propose multi-objective intelligent optimization algorithms and apply them to NoC design.

4.1 Solution representation

4.1.1 Representation of the solution to IP core assignment

An IP core assignment solution is expressed as an array of genes. Each gene represents an IP core assigned to a subtask. For example, a solution is described as $C=(g_1, g_2, \ldots, g_t)$, where gene g_i represents the IP core assigned to the ith subtask, and t is the total number of subtasks in a task graph. Some subtasks must be completed by some specific IP cores. So, the generation of the initial solutions is partially random. Each gene is selected randomly from the candidate IP cores which can execute this type of task.

4.1.2 Representation of the solution to IP core mapping

An IP core mapping layout is expressed as a sequence $p=(p_1, p_2, \ldots, p_d)$, where p_i (i=1, 2, …, d) represents the tile assigned to IP core i. For example, in p=(4, 6, 5, 3, 2, 8, 7, 0, 1), the first digit is 4, which denotes that the first IP core maps to tile 4. d represents the number of tiles in an NoC topology.

4.2 Hybrid PSOGA algorithm

PSO is an algorithm of swarm intelligence, which is often applied to solving complex optimization problems because of its fast convergence (Masehian and Sedighizadeh, 2010). However, the standard PSO is weak in local search. We propose two efficient solutions to remedy this problem using the evolutionary operator of GA and the local search strategy of SA.

PSOGA combines PSO with GA. We first update the standard PSO formula to meet NoC application characteristics, and then use two genetic operators, mutation, and selection, to improve PSO. Mutation helps to increase particle diversity and selection is used to select the higher fitness particles for use in the next step. The steps are as follows:

1. Generating and updating populations

A particle in PSO is a solution which represents an IP core assignment or mapping design. The positions of particles are represented in Section 4.1. The initial particles are generated partially randomly. The updating formulas of particles are given by

$$\begin{cases} V_{t+1} = \omega V_t \oplus c_1 \text{rand}_1 \text{replace}_{\text{pBest}} \oplus c_2 \text{rand}_2 \text{replace}_{\text{gBest}}, \\ P_{t+1} = P_t \oplus V_{t+1}, \end{cases} \quad (10)$$

where V_t is the velocity of a particle at time instant t, and P_t is the position. $\oplus$ represents a replacement operation. ω controls the amount of current velocity, and the relative influence of the self and global knowledge is determined by positive constants c_1 and c_2, respectively. rand_1 and rand_2 are random real numbers in the range [0, 1]. $\text{replace}_{\text{pBest}}$ is the replacement number with reference to the personal optimal solution. The process of replacement is shown in Fig. 3. $\text{replace}_{\text{gBest}}$ represents the number of replacements with reference to the global optimal solution.

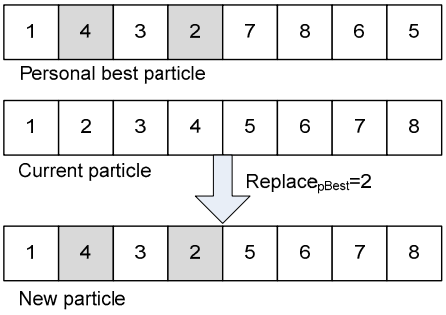

Fig. 3 Process of replacement

2. Diversifying updated populations

In the assignment phase, the mutation position is selected randomly. The mutation value is selected only from the candidate IP core set of the corresponding subtask. In the mapping phase, to ensure the validity of solutions, the elements swap with other elements of the same particle if their mutation probabilities are higher than the preset value P_m.

3. Calculating fitness

The fitness of updated particles is computed according to evaluation indices, and the new personal and global best particles are selected. If reaching a preset number of iterations, iteration terminates. Otherwise, these particles are sorted, and the high fitness particles are selected for the next iteration.

4.3 Hybrid PSOSA algorithm

The algorithm proposed in this section is based on PSO and simulated annealing (SA). SA is a search algorithm capable of escaping from local optimal solutions. The initial solutions are obtained by PSO, and then SA executes a neighborhood search. In this method, a solution with worse performance is allowed

with a probability function described in Eq. (12). This method helps escape from local optima and guides to a better solution. The pseudocode of the PSOSA algorithm is as follows:

PSOSA algorithm

```
Begin
  Generate the initial particle swam;
  Compute multi-objective evaluation indices for
    particles;
  Select initial pBest and gBest;
  Repeat
    Update particles;
    Repeat // starting local searching by SA
      Set the initial and final temperatures;
      Repeat
        Select a particle as the current particle;
        Repeat
          Generate the particle's neighbors;
          If (d1<0)+(d2<0)+…+(dN<0)≥N
            // di represents the difference of the ith evaluation
            // index, and N represents the number of evaluation
            // indices
            Accept the neighbor particle;
          Else
            If p<exp(−Δf/T)
              Accept the neighbor particle;
            End if
          End if
        Until (reach the preset number of iterations)
        Decrease the temperature;
      Until (reach the preset temperature)
    Until (all particles finish local searching)
    Update local and global best particles;
  Until (reach the termination condition)
End
```

The neighbors are particles in which the order of a few elements is different from that in the current particle. The formula for differences between the particle and its neighbor is given by

$$d_i = f_i(\text{neighbor}) - f_i(\text{current}), \tag{11}$$

where d_i (i=1, 2,…, N) represents the difference of the ith evaluation index, and $f_i(\cdot)$ represents the ith evaluation function. N is the number of valuation indices. If three or more index differences are less than 0, the neighbor is accepted. Otherwise, the neighbor is accepted with probability p_0, defined as

$$\begin{cases} p_0 = \exp(-\Delta f / T), \\ \Delta f = \max\limits_{1\le i\le N}\left\{\text{abs}\left[d_i / f_i(\text{current})\right]\right\}, \end{cases} \tag{12}$$

where T is the temperature, and abs is the function of absolute value.

4.4 SS algorithm

Scatter search (SS) is an evolutionary optimization algorithm. Compared with other evolutionary algorithms, its main characteristics are as follows: the size of the population is small; the evolution of the population is controlled partly by deterministic rules and is not totally random; a local search procedure is integral to SS (Hung and Song, 2001). The search rules of the algorithms deeply influence their search capability (Wang *et al.*, 2012b). Based on standard SS, we propose an improved multi-objective SS algorithm for IP core assignment and mapping. The main methods of our improved SS are as follows:

1. Individual generation method

Each IP core assignment solution is represented by an array of genes as described in Section 4.1.1. The initial population of assignment not only is diverse but also can execute corresponding subtasks. So, each gene is generated by selecting an IP core randomly from the candidate IP cores. Improving solutions is an important step in SS. Optimal training sequences contribute to better performance (Wang *et al.*, 2012a). A threshold, whose value is determined by the deadlines of NoC applications, is used to improve solutions. The IP cores under the threshold have opportunities to become solutions.

In the mapping phase, each layout of IP core mapping is represented by a sequence as described in Section 4.1.2. The initial populations are generated randomly. The method of improving mapping solutions is local search. We select the IP core with maximum degree and then place the IP cores connected with it onto its adjacent tiles.

2. Reference set and subset generation method

The reference set consists of high performance solutions and diverse solutions. The solutions are sorted and ranked. We compare the values of the same dimension between two solutions, and the number of the same values of two solutions is taken as the distance between these two solutions. We adopt 2-element subsets (Hung and Song, 2001), pairing each

solution in the reference set with another solution randomly.

3. Solution combination method

Solution combination operates on every subset. In the assignment phase, the crossover position is selected randomly, and the genes of two solutions from the crossover position to the last position are exchanged. In the mapping phase, we design a new crossover operator to avoid generating invalid solutions. The crossover process is described in Fig. 4. The mutation operator we use here is swap mutation.

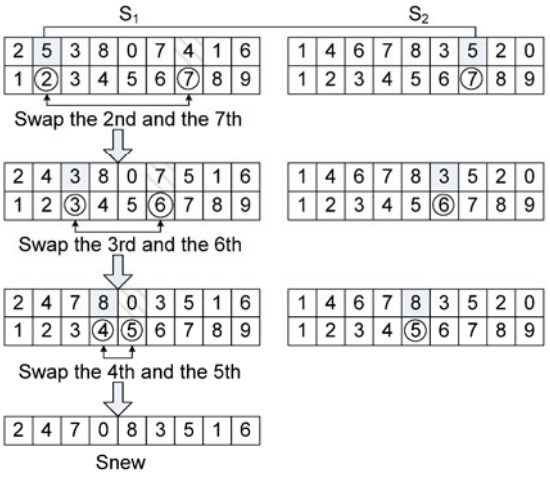

Fig. 4 Process of crossover

4. Reference set update method

Compute the evaluation indices of new solutions and the distances between new solutions and the reference set. The new solutions with high performance and long distance replace the old solutions at the lowest Pareto optimal rank.

5 Experimental results

To illustrate the performance of the proposed algorithms, we performed tests on the embedded system synthesis benchmarks suite (E3S) (http://ziyang.eecs.umich.edu/~dickrp/e3s/) and NIRGAM (http://nirgam.ecs.soton.ac.uk/home.php). E3S provides a set of task graphs of NoC applications and an IP core repository. The task graphs in E3S represent real applications. The results were compared with those from NSGA-II and micro-GA (da Silva *et al.*, 2010).

5.1 IP core assignment experimental results

We tried to obtain more solutions under the constraints of power consumption and time (da Silva *et al.*, 2010). Tables 1 and 2 show the numbers of potential solutions of IP core assignment and mapping obtained from different algorithms, respectively.

Table 1 Numbers of potential solutions of IP core assignment under different algorithms in various applications

Application	Number of potential solutions				
	NSGA-II	Micro-GA	PSOGA	PSOSA	SS
AutoTg0	2	4	12	9	10
AutoTg2	17	23	29	35	40
ConsumerTg0	9	6	10	10	12
ConsumerTg1	3	9	15	11	14
NetworkingTg2	2	6	9	7	10
OfficeTg0	8	18	22	22	23
TelecomTg1	2	2	6	6	13

Table 2 Numbers of potential solutions of IP core mapping under different algorithms in various applications

Application	Number of potential solutions				
	NSGA-II	Micro-GA	PSOGA	PSOSA	SS
AutoTg0	2	7	9	8	19
AutoTg2	11	47	48	46	54
ConsumerTg0	3	10	12	11	21
ConsumerTg1	7	18	23	18	23
NetworkingTg2	3	7	10	11	19
OfficeTg0	8	25	28	25	33
TelecomTg1	1	4	8	8	13

We obtained more solutions from our algorithms PSOGA, PSOSA, and SS than from NSGA-II and micro-GA because PSOGA and PSOSA adopt improving methods which decrease the possibility of local optimal solutions, searching for more potential solutions. SS constructs a reference set which includes high quality solutions and diverse solutions. Diverse solutions also decrease the chance of being trapped in local optima.

AutoTg2 is an NoC application with the largest number of possible solutions, and thus is the most represented case. We compared its top seven best solutions obtained from different algorithms (Fig. 5). The times obtained from our improved algorithms are better than those from NSGA-II and micro-GA. The

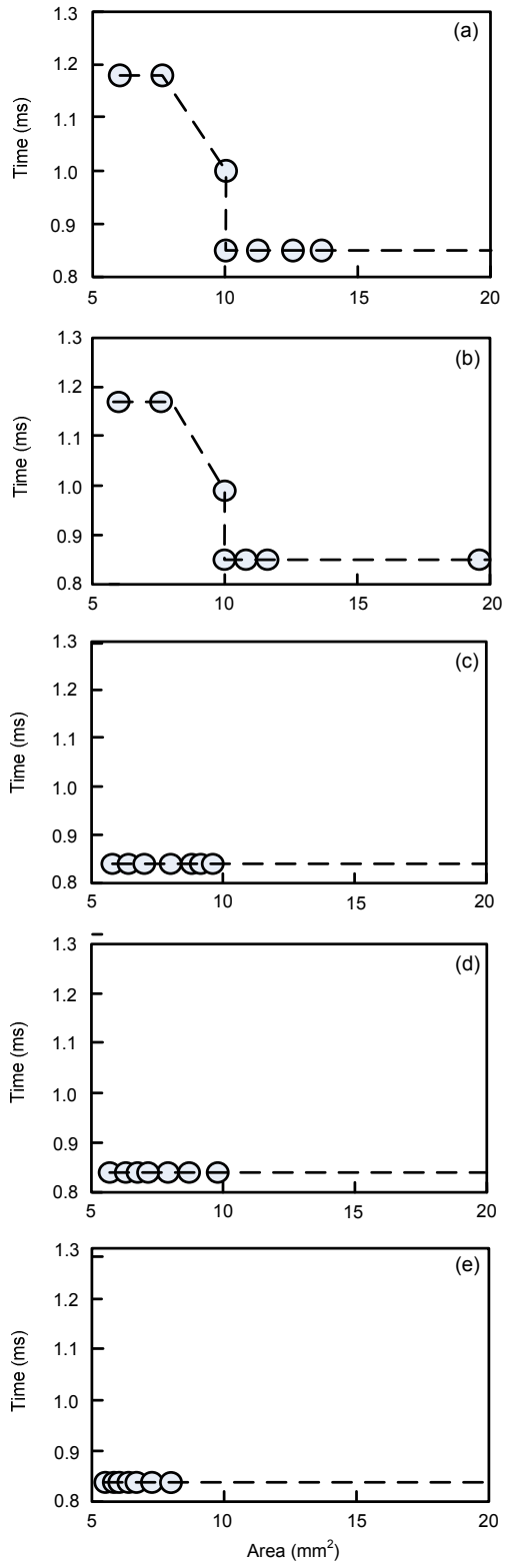

Fig. 5 Performance comparison for AutoTg2
(a) NSGA-II; (b) Micro-GA; (c) PSOGA; (d) PSOSA; (e)SS

areas obtained from NSGA-II and micro-GA are similar to those from our improved algorithms.

5.2 IP core mapping experimental results

Table 2 shows the numbers of mapping solutions obtained from different algorithms. The numbers obtained from our improved algorithms are greater than those from NSGA-II and micro-GA. The intelligent algorithms proposed in this paper are based on random search, and thus the solutions have a certain randomicity. To analyze the stable performance of the algorithms, for each evaluation index, we accumulated the best values of 10 calculations. The experimental results are shown in Fig. 6.

Fig. 6a shows that the SS algorithm obtains the minimum time values in all test cases. The values of PSOGA and PSOSA are closest to those of SS. PSOSA is superior to PSOGA in some test cases, but inferior in others. These two methods are PSO-

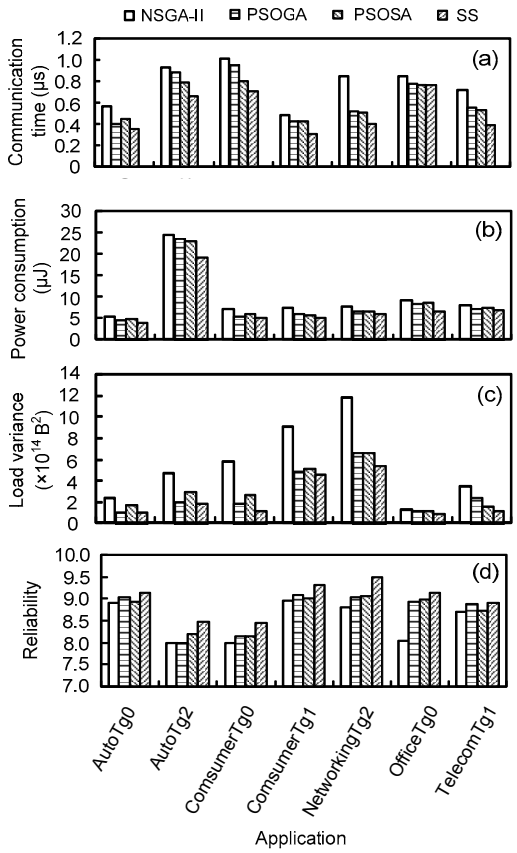

Fig. 6 Performance obtained from our improved algorithms in various applications
(a) Communication time; (b) Power consumption; (c) Load variance; (d) Reliability

based algorithms and have some similarities. The time values obtained by NSGA-II are larger than those of the other algorithms.

Fig. 6b shows that the power consumption obtained by SS is the lowest. It is a nip and tuck between PSOGA and PSOSA. The results of NSGA-II are the worst. Fig. 6c shows the variances of link load. A smaller load variance means a better balance. The load variance obtained by SS is the best and those obtained by NSGA-II are the worst.

High reliability is important to NoC mapping. We adopted redundant links to improve system reliability. The reliability of each switch R_s was set to 0.94. In Fig. 6d, the reliability obtained by SS is higher than that of the other algorithms. This is because strategies of SS can allow more reliable paths to be explored. NSGA-II gives the lowest reliability.

To evaluate further the performance of the mapping topology generated by our proposed algorithms, we performed simulations using NIRGAM, which is a cycle accurate simulator targeted at NoC research. The application AutoTg2, which has the most subtasks in E3S test cases, is the most representative case. We show its mapping and simulation results in detail.

This test application consists of nine subtasks, and the traffic between the subtasks is generated randomly (Fig. 7). To avoid generating hotspots which may degrade the NoC performance, we assume that a tile is assigned to, at most, two subtasks in a given NoC application. For example, in Fig. 7a, there are circles 6 and 8 in square 5, which represent subtasks 6 and 8 executed by the PE in tile 5. We simulated mapping results obtained from four algorithms on NIRGAM. The simulated structure was a 2D mesh topology of size 3×3. Traffic generation began after 5 clock cycles, and continued until 500 clock cycles. The interval between flits was 2 clock cycles. The traffic load of each link was set to a different value according to the communication traffic between PEs. The simulation stopped after 3000 clock cycles. NIRGAM measures NoC performance on a per-link basis. Fig. 8 shows the average latencies for different mapping results.

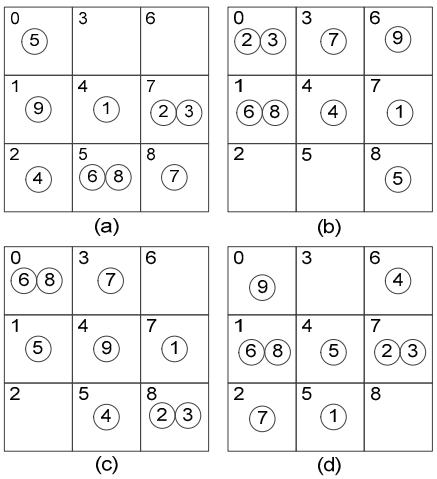

Fig. 7 Mapping topologies in AutoTg2 obtained by NSGA-II (a), PSOGA (b), PSOSA (c), and SS (d)
Each small square represents a tile in the NoC topology, and circles in the square represent subtasks assigned to this tile

The numbers 0–8 show the placements of tiles. The bar between tiles represents the average latency per flit in different direction links. For example, there are two bars between tiles 0 and 1: the left one represents the eastward link from tile 0 to 1, and the right one represents the westward link from tile 1 to 0. Fig. 8 shows that SS obtained the lowest average latency. PSOGA and PSOSA were the next lowest. NSGA-II gave the highest latency. The simulation results of other test applications are shown in Table 3 (the corresponding mapping topologies and average latencies obtained by different algorithms are referred to supplementaries Figs. S1–S12). The overall average latencies of mapping results obtained from SS are the lowest, and those of PSOGA and PSOSA come second.

Table 3 Comparison of overall average latencies of mapping results from different algorithms

Application	Average latency (clock cycle per flit)			
	NSGA-II	PSOGA	PSOSA	SS
AutoTg0	1.867	1.533	1.533	1.200
ConsumerTg0	2.057	1.719	1.667	1.500
ConsumerTg1	3.731	2.417	2.417	1.917
NetworkingTg2	1.333	1.333	1.333	1.333
OfficeTg0	1.581	1.533	1.581	1.533
TelecomTg1	1.889	1.025	1.025	1.009

6 Conclusions

We studied two important phases in NoC design: IP core assignment and mapping. Based on the

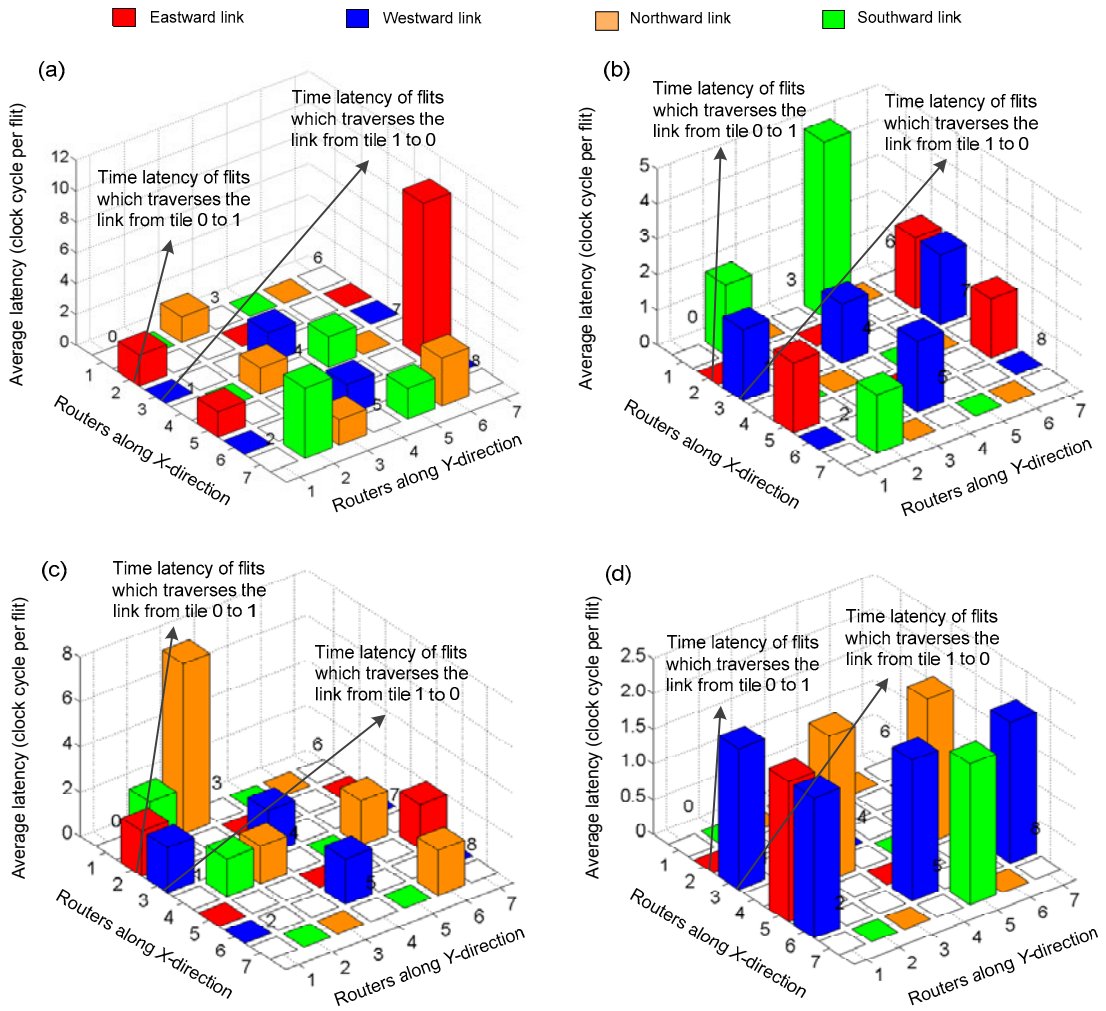

Fig. 8 Average latencies in AutoTg2 obtained by NGSA-II (a), PSOGA (b), PSOSA (c), and SS (d)

The overall average latencies (clock cycle per flit) are 2.8111, 2.3161, 2.5617, and 1.9071, respectively. References to color refer to the online version of this figure

characteristics of each phase, we designed different evaluation models and algorithms, giving a more comprehensive evaluation of NoC design. To overcome the drawbacks of traditional intelligent algorithms, we designed improved multi-objective intelligent algorithms based on the requirements and constraints of NoC design. PSOGA and PSOSA algorithms not only take the advantages of traditional intelligent algorithms, but also avoid being trapped in local optima. The search process of the improved SS algorithm is based on strategies instead of total randomness, leading to better solutions. The proposed rerouting algorithm based on a fault-tolerant mechanism can rebuild a shortest communication path for switch faults, improving the reliability of systems. The experimental results show that our proposed algorithms obtain more solutions with high reliability and performance. In the future, we will extend our intelligent algorithms to other NoC architectures based on different topologies and develop dynamic analysis models to attain more accurate evaluation.

References

Bjerregaard, T., Mahadevan, S., 2006. A survey of research and practices of network-on-chip. *ACM Comput. Surv.*, **38**(1):1.1-1.51. [doi:10.1145/1132952.1132953]

Cheng, A.L., Pan, Y., Yan, X.L., *et al.*, 2011. A general communication performance evaluation model based on routing path decomposition. *J. Zhejiang Univ.-Sci. C (Comput. & Electron.)*, **12**(7):561-573. [doi:10.1631/jzus.C1000281]

da Silva, M.V.C., Nedjah, N., Mourelle, L.M., 2010. Power-aware multi-objective evolutionary optimisation for application mapping on network-on-chip platforms. *Int. J. Electron.*, **97**(10):1163-1179. [doi:10.1080/00207217.2010.512105]

Das, R., Eachempati, S., Mishra, A.K., *et al*., 2009. Design and evaluation of a hierarchical on-chip interconnect for next-generation CMPs. Proc. IEEE 15th Int. Symp. on High Performance Computer Architecture, p.175-186. [doi:10.1109/HPCA.2009.4798252]

Hu, J., Marculescu, R., 2003. Energy-aware mapping for tile-based NoC architectures under performance constraints. Proc. Asia and South Pacific Design Automation Conf., p.233-239. [doi:10.1109/ASPDAC.2003.1195022]

Hung, W.N.N., Song, X., 2001. BDD variable ordering by scatter search. Proc. Int. Conf. on Computer Design, p.368-373. [doi:10.1109/ICCD.2001.955053]

Jena, R.K., Sharma, G.K., 2007. A multi-objective evolutionary algorithm based optimization model for network-on-chip synthesis. Proc. 4th Int. Conf. on Information Technology, p.977-982. [doi:10.1109/ITNG.2007.10]

Liu, W., Gu, Z., Xu, J., *et al*., 2011. Satisfiability modulo graph theory for task mapping and scheduling on multiprocessor systems. *IEEE Trans. Parall. Distr. Syst.*, **22**(8):1382-1389. [doi:10.1109/TPDS.2010.204]

Marculescu, R., Ogras, U.Y., Peh, L.S., *et al*., 2009. Outstanding research problems in NoC design: system, microarchitecture, and circuit perspectives. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.*, **28**(1):3-21. [doi:10.1109/TCAD.2008.2010691]

Masehian, E., Sedighizadeh, D., 2010. Multi-objective robot motion planning using a particle swarm optimization model. *J. Zhejiang Univ.-Sci. C (Comput. & Electron.)*, **11**(8):607-619. [doi:10.1631/jzus.C0910525]

Muralimanohar, N., Balasubramonian, R., Jouppi, N., 2007. Optimizing NUCA organizations and wiring alternatives for large caches with CACTI 6.0. Proc. 40th Annual IEEE/ACM Int. Symp. on Microarchitecture, p.3-14. [doi:10.1109/MICRO.2007.33]

Orgas, U.Y., Hu, J., Marculescu, R., 2005. Key research problems in NoC design: a holistic perspective. Proc. 3rd IEEE/ACM/IFIP Int. Conf. on Hardware/Software Codesign and System Synthesis, p.69-74. [doi:10.1145/1084834.1084856]

Rao, A.R.M., Arvind, N., 2005. A scatter search algorithm for stacking sequence optimisation of laminate composites. *Compos. Struct.*, **70**(4):383-402. [doi:10.1016/j.compstruct.2004.09.031]

Refan, F., Alemzadeh, H., Safari, S., *et al*., 2008. Reliability in application specific mesh-based NoC architectures. Proc. 14th IEEE Int. On-line Testing Symp., p.207-212. [doi:10.1109/IOLTS.2008.53]

Saxena, P.C., Gupta, S., Rai, J., 2003. A delay optimal coterie on the *k*-dimensional folded Petersen graph. *J. Parall. Distr. Comput.*, **63**(11):1026-1035. [doi:10.1016/S0743-7315(03)00116-3]

Sepulveda, M.J., Strum, M., Chau, W.J., 2011. A multi-objective adaptive immune algorithm for NoC mapping. Proc. 17th IFIP Int. Conf. on Very Large Scale Integration, p.193-196. [doi:10.1109/VLSISOC.2009.6041354]

Tang, L., Kumar, S., 2003. A two-step genetic algorithm for mapping task graphs to a network on chip architecture. Euromicro Symp. on Digital System Design, p.180-187. [doi:10.1109/DSD.2003.1231923]

Wang, J., Jiao, Y., Song, X., *et al*., 2012a. Optimal training sequences for indoor wireless optical communications. *J. Opt.*, **14**(1):015401.1-015401.5. [doi:10.1088/2040-8978/14/1/015401]

Wang, J., Xie, X., Jiao, Y., *et al*., 2012b. Optimal odd-periodic complementary sequences for diffuse wireless optical communications. *Opt. Eng.*, **51**(9):095002.1-095002.6. [doi:10.1117/1.OE.51.9.095002]

Yu, Q., Ampadu, P., 2010. A flexible parallel simulator for networks-on-chip with error control. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.*, **29**(1):103-116. [doi:10.1109/TCAD.2009.2034353]

List of electronic supplementary materials

Fig. S1 Mapping topologies in AutoTg0 obtained using different algorithms

Fig. S2 Average latencies in AutoTg0 obtained using different algorithms

Fig. S3 Mapping topologies in ConsumerTg0 obtained using different algorithms

Fig. S4 Average latencies in ConsumerTg0 obtained using different algorithms

Fig. S5 Mapping topologies in ConsumerTg1 obtained using different algorithms

Fig. S6 Average latencies in ConsumerTg1 obtained using different algorithms

Fig. S7 Mapping topologies in NetworkingTg2 obtained using different algorithms

Fig. S8 Average latencies in NetworkingTg2 obtained usingy different algorithms

Fig. S9 Mapping topologies in OfficeTg0 obtained using different algorithms

Fig. S10 Average latencies in OfficeTg0 obtained using different algorithms

Fig. S11 Mapping topologies in TelecomTg1 obtained using different algorithms

Fig. S12 Average latencies in TelecomTg1 obtained using different algorithms