ELectronics EXpress

LETTER

IEICE Electronics Express, Vol.13, No.3, 1–12

A novel two-phase heuristic for application mapping onto mesh-based Network-on-Chip

Xinyu Wang[1a)], **Haikuo Liu**[1b)], **Zhigang Yu**[2c)], **and Kele Shen**[2d)]

[1] *College of Management Science and Engineering, Dongbei University of Finance and Economics, Dalian, China*

[2] *Department of Computer Science and Technology, Tsinghua University, Beijing, China*

a) *Distribute_2008@163.com*

b) *sea_liuhaikuo@163.com*

c) *yuzg11@mails.tsinghua.edu.cn, Corresponding Author*

d) *shenkele2006@yahoo.com.cn*

Abstract: With the growing complexity of embedded VLSI products, traditional System-on-Chip (SoC) are facing severe challenges in the aspects of communicating speed and scalability. Network-on-Chip (NoC) has emerged as a viable alternative. In NoC design, application mapping is one of the most holistic researching dimensions, which maps the cores in the application to the routers in the NoC platform. Application mapping problem usually aims to reduce communication cost and power consumption of the overall system. In this paper, we focus on application mapping onto mesh network, and propose a novel two-phase heuristic algorithm. The first phase attempts to explore the potential searching spaces, while the second phase focuses on exploiting the local optima within the searching basin. To verify the effectiveness of the algorithm, this paper performs a quantitative comparisons between our proposed method and the existing mapping methods under both real application and custom generated application benchmarks.

Keywords: Network-on-Chip (NoC), application mapping problem, mesh, two-phase heuristic algorithm

Classification: Integrated circuits

References

[1] L. Benini and G. De Micheli: Computer **35** (2002) 70. DOI:10.1109/2.976921

[2] C. Killian, C. Tanougast, F. Monteiro and A. Dandache: IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **22** (2014) 242. DOI:10.1109/TVLSI.2013.2240324

[3] H. Takase, G. Zeng, L. Gauthier, H. Kawashima, N. Atsumi, T. Tatematsu, Y. Kobayashi, S. Kohara, T. Koshiro, T. Ishihara, H. Tomiyama and H. Takada: ISLPED (2011) 271. DOI:10.1109/ISLPED.2011.5993648

[4] H. Takase, G. Zeng, L. Gauthier, H. Kawashima, N. Atsumi, T. Tatematsu, Y. Kobayashi, T. Koshiro, T. Ishihara, H. Tomiyama and H. Takada: IEICE Trans. Fundamentals **97** (2014) 2477. DOI:10.1587/transfun.E97.A.2477

[5] B. Zhang, Z. Wang, H. Gu, Y. Yang and K. Wang: IEICE Electron. Express **9**

DOI: 10.1587/elex.13.20151097
Received December 21, 2015
Accepted January 6, 2016
Publicized January 25, 2016
Copyedited February 10, 2016

(2012) 1510. DOI:10.1587/elex.9.1510
[6] S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar and S. Borkar: IEEE J. Solid-State Circuits **43** (2008) 29. DOI:10.1109/JSSC.2007.910957
[7] B. Towles, J. Grossman, B. Greskamp and D. Shaw: ISCA (2014) 1.
[8] S. Tosun, O. Ozturk, E. Ozkan and M. Ozen: JSC **71** (2015) 995. DOI:10.1007/s11227-014-1348-x
[9] S. Tosun: JSA **57** (2011) 69.
[10] C. Rhee, H. Jeong and S. Ha: ICCD (2004) 438. DOI:10.1109/ICCD.2004.1347959
[11] J. Hu and R. Marculescu: IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **24** (2005) 551. DOI:10.1109/TCAD.2005.844106
[12] O. He, S. Dong, W. Jang, J. Bian and D. Pan: IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **20** (2012) 1496. DOI:10.1109/TVLSI.2011.2159280
[13] J. Soumya, S. Tiwary and S. Chattopadhyay: JSA **61** (2015) 1.
[14] M. Janidarmian, A. Khademzadeh and M. Tavanpour: IEICE Electron. Express **6** (2009) 1. DOI:10.1587/elex.6.1
[15] S. Murali and G. Micheli: DATE (2004) 896. DOI:10.1109/DATE.2004.1269002
[16] W. Shen, C. Chao, Y. Lien and A. Wu: NoCs (2007) 317. DOI:10.1109/NOCS.2007.5
[17] P. Sahu, K. Manna and S. Chattopadhyay: IJCA **115** (2015) 13. DOI:10.5120/20258-2643
[18] J. Fang, L. Yu, S. Liu, J. Lu and T. Chen: JSC **71** (2015) 4056.
[19] http://ziyang.eecs.umich.edu/~dickrp/tgff/.
[20] S. Tosun, O. Ozturk and M. Ozen: AICT (2009) 1. DOI:10.1109/ICAICT.2009.5372524
[21] J. Hu and R. Marculescu: DATE (2003) 688. DOI:10.1109/DATE.2003.1253687
[22] N. Koziris, M. Romesis, P. Tsanakas and G. Papakonstantinou: PDP (2000) 406. DOI:10.1109/EMPDP.2000.823437
[23] M. Tavanpour, A. Khademzadeh and M. Janidarmian: IEICE Electron. Express **6** (2009) 1535. DOI:10.1587/elex.6.1535
[24] F. Moein-Darbari, A. Khademzadeh and G. Gharooni-Fard: CSE (2009) 366. DOI:10.1109/CSE.2009.321

1 Introduction

With the mature technology of modern integrated circuit design, a growing number of cores are integrated into a single chip. Communication in SoC becomes a bottleneck since the underlying communicating architectures are inherently non-scalable [1]. NoC, a scalable and modular design approach, forms an emerging paradigm for inter-core communications within a chip [2].

Minimization of power consumption has become an important aspect in the overall system design [3, 4]. Application mapping onto a NoC platform is the first step in the generic synthesis flow. This synthesis step is non-trivial since the optimization parameters (such as power consumption, cost and performance) may vary greatly from one solution to another [5]. Given an application, it can be described as a weighted directed graph. The mapping technique aims at choosing

DOI: 10.1587/elex.13.20151097
Received December 21, 2015
Accepted January 6, 2016
Publicized January 25, 2016
Copyedited February 10, 2016

an optimal way to assign the cores in the application to the routers in the NoC platform, with the objective of minimizing power consumption or maximizing the overall performance.

Regular topologies can be reused and are easy to design. The two dimensional and well-organized mesh leads to simple layout and predictable electronic specification, which makes it the most attractive architecture for NoC design. Examples include Tile80 [6], NVIDIA Tegra K1 Mobile Processor and the router fabric in the supercomputer Anton2 [7]. In this paper, we mainly focus on application mapping onto mesh topology.

In [8], Tosun et al. model the application mapping problem with the well-accepted abstract graph models. For an application with n cores, there exist $n!$ mapping solutions. The huge solution space makes the exhaustive searching algorithms infeasible for large-scale problems. The application mapping problem is an NP-hard combinatorial optimization problem [9]. Consequently, no exact algorithm is expected to solve the problem in the polynomial time, and considerable computation time is required for even small-scale applications. This hardness is confirmed since the existing exact algorithms cannot optimally solve the problems with more than 25 cores [8]. Nevertheless, several exact optimization techniques have been developed to improve the design parameter values of the problem. The exact algorithms are mainly based on integer linear programming (ILP) [8, 10] and branch-and-bound (BB) [11] methods to obtain optimal solutions. Each exact algorithm guarantees to yield an optimal solution, while it may require intolerable computation time. Heuristic method can be a good choice since it provides satisfactory suboptimal solutions in acceptable computing time.

In this paper, we introduce a novel two-phase heuristic algorithm, which employs a different algorithmic framework. The first phase attempts to explore the potential searching spaces, while the second phase focuses on exploiting the local optima in the given searching basin. Due to this feature, the proposed algorithm is called two-phase heuristic algorithm. Since the combination of perturbation and local search is effective and efficient to solve a variety of problems, these two phases in our proposed algorithm adopt the combination. In addition to the algorithmic framework, our algorithm uses several different perturbation and local search operators. The proposed method is compared with the existing algorithms to verify its performance.

The remainder of this paper is organized as follows. The next section discusses related work. Section 3 presents the problem formulation in terms of graph model and integer programming. In section 4, we present our proposed two-phase heuristic algorithm and detail its main components. Section 5 provides the benchmarks used and the experimental settings. Besides, the experimental results are discussed in details by comparing with the existing algorithms. Finally, conclusions are provided in the last section.

2 Related work

In recent years, several studies have been conducted on mapping the cores of the application to routers of the NoC platform. The selected NoC adopts either

DOI: 10.1587/elex.13.20151097
Received December 21, 2015
Accepted January 6, 2016
Publicized January 25, 2016
Copyedited February 10, 2016

application-specific irregular topology or the well-accepted regular topology. The mapping techniques use either exact or heuristic algorithms.

He et al. [12] propose a unified flow combining task scheduling and core mapping (UNISM), and model it with a mixed integer linear programming (MILP) to support both mesh and costumed topologies. Soumya et al. [13] affirm that application-specific architectures are better than regular architectures in terms of power, area and performance. Based on the technique of integrated soft core and router placement, the authors propose a particle swarm optimization (PSO) algorithm to map applications onto application-specific NoC. Furthermore, a tradeoff between communication cost and overall chip area has been made to fulfill different levels of requirements.

Due to the reusability and scalability, regular topologies are usually popular and considered as the target architecture in the application mapping problem. Based on mesh, Hu et al. [11] propose an energy-aware algorithm using the branch-and-bound method. The algorithm yields solutions by walking through the tree that represents the solution space. Janidarmian et al. [14] propose a constructive heuristic algorithm (called Onyx) to assign cores to routers on a lozenge-shape path. In the algorithm, priorities are assigned to the cores based on the communication volume. The Onyx is further extended in CastNet [9] which exploits the symmetry of mesh to determine the initial routers. Other heuristics (such as NMAP [15], CHMAP [16], and GA [5, 8]) are also developed to map applications onto mesh-based NoCs.

Several previous works devote efforts to application mapping onto tree-based topologies. Sahu et al. [17] address the application mapping problem onto Butterfly-Fat-Tree network with a novel augmented PSO algorithm. Based on Mesh-of-Tree, Fang et al. [18] propose KL_GA by taking the advantage of both Kernighan-Lin algorithm and genetic algorithm. The KL_GA algorithm first generates an initial solution with the KL-based method, and then applies a GA-based algorithm to avoid premature phenomena.

3 Problem formulation

Both application and selected mesh architecture are taken as the inputs in the mapping problem.

The application can be characterized in the form of a directed graph, with vertices representing cores and directed edges representing the communicating tasks between cores. For an application with m cores, we can describe its graph with the following $m \times m$ matrix:

$$C = \begin{pmatrix} w_{0,0}, & w_{0,1}, & \cdots & w_{0,m-1} \\ w_{1,0}, & w_{1,1}, & \cdots & w_{1,m-1} \\ \vdots & \vdots & \vdots & \vdots \\ w_{m-1,0}, & w_{m-1,1}, & \cdots & w_{m-1,m-1} \end{pmatrix} \quad (1)$$

In the C matrix above, $w_{i,j}$ is the weight value associated with arc $e_{i,j}$. If arc $e_{i,j} \notin E$, $w_{i,j}$ is set to be 0.

In a $k \times k$ mesh, each router is represented by a two-element integer coordinate (x, y), $0 \leq x, y < k - 1$. Each router has a unique numeric label $(x \times k + y)$, i.e., the routers are numbered in increasing order (0 to $k \times k - 1$) from the top left corner to

DOI: 10.1587/elex.13.20151097
Received December 21, 2015
Accepted January 6, 2016
Publicized January 25, 2016
Copyedited February 10, 2016

the bottom right corner. Suppose that n is equal to $k \times k$, we can describe mesh with the following $n \times n$ matrix:

$$D = \begin{pmatrix} d_{0,0}, & d_{0,1}, & \dots & d_{0,n-1} \\ d_{1,0}, & d_{1,1}, & \dots & d_{1,n-1} \\ \vdots & \vdots & \vdots & \vdots \\ d_{n-1,0}, & d_{n-1,1}, & \dots & d_{n-1,n-1} \end{pmatrix} \tag{2}$$

In the D matrix above, each $d_{i,j}$ can be calculated as $|\lfloor i/k \rfloor - \lfloor j/k \rfloor| + |(i \bmod k) - (j \bmod k)|$. Clearly, $d_{i,j}$ represents the distance between router i and j.

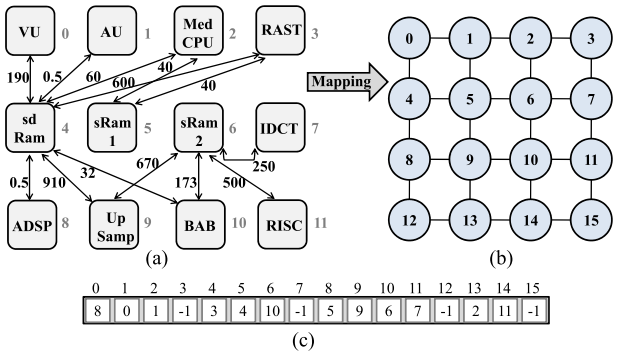

Fig. 1. An application mapping example: (a) the graph of MPEG-4 application, (b) a 4×4 mesh, (c) a mapping solution from (a) to (b)

Let Π denote the set of the permutation functions π: $\{0, 1, \dots, m-1\} \rightarrow \{0, 1, \dots, n-1\}$, then the application mapping problem can mathematically be formulated as follows:

$$\min_{\pi \in \Pi} f(\pi) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} D(i, j) \cdot C(\pi_i, \pi_j) \tag{3}$$

where $\pi \in \Pi$ is a mapping solution, and π_i is the ith element of π, which denotes the core chosen for router i. The problem objective is then to find a permutation $\pi^* \in \Pi$ that minimizes the sum of the products of the communication and distance matrices, i.e., $f(\pi^*) \leq f(\pi)$, for $\forall\, \pi \in \Pi$. Obviously, each feasible solution π can be described by a 1-dimensional array with n digits. The ith element (π_i) in the array denotes that the core π_i is mapped to router i. If $\pi_i = -1$, no core will be mapped to router i.

Fig. 1(a) shows the graph of the MPEG-4 benchmark. In this application, we have 12 vertices and 13 edges, where the weights of the edges represent the amount of data transferred between two cores. A 4×4 mesh is given in Fig. 1(b). A solution for mapping MPEG-4 onto a 4×4 mesh NoC architecture is shown in 1(c). As shown in the figure, the core ADSP is mapped to router 0. Besides, no core is connected to routers 3, 7, 12, and 15.

The problem can be solved by integer programming. The objective function and constraints are stated as follows:

$$min \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} \sum_{k=0}^{n-1} \sum_{l=0}^{n-1} x_{i,k} \cdot x_{j,l} \cdot C(i, j) \cdot D(k, l) \tag{4}$$

Subject to:

$$\sum_{i=0}^{m-1} x_{i,j} \leq 1, \quad for\ \forall j = 0, 1, \dots, n-1 \tag{5}$$

DOI: 10.1587/elex.13.20151097
Received December 21, 2015
Accepted January 6, 2016
Publicized January 25, 2016
Copyedited February 10, 2016

$$\sum_{j=0}^{n-1} x_{i,j} = 1, \quad for\ \forall i = 0, 1, \ldots, m-1 \tag{6}$$

$$x_{i,j} = 0\ or\ 1, \quad for\ \forall i = 0, 1, \ldots, m-1, j = 0, 1, \ldots, n-1 \tag{7}$$

In the integer programming based formulations above, $x_{i,j}$ is decision variable, with value of 1 meaning that core i is connected with router j. Expression (4) defines the objective function to be minimized. Equation (5) ensures that no concentration occurs at any router, while equation (6) guarantees each core has one and only one router to be connected. Equation (7) declares that $x_{i,j}$ is 0-1 variable. The problem can also be modeled with an ILP formulation, details about that can be found in [8].

4 The proposed two-phase heuristic algorithm

The proposed heuristic algorithm aims to assign cores in the application to routers in the mesh-based NoC in a one-to-one manner. The two-phase heuristics starts from an initial solution π^0, and performs the two-phase optimizing operators to improve the solution through iterations. The two phases employ different perturbation operators and local search techniques. In the first phase, we use cross-based and exchange-based operators to perturb the solution. In the second phase, three different perturbation operators (including directed, least-recently-used (LRU), and random) are introduced to disturb the solution stuck in local optima. As for the local search technique, we use the 2-opt-based steepest descent procedure to further optimizing the disturbed solutions. The perturbing operations provide the capacity to sample different areas of the search space, trying to find the most promising areas. Once such areas are detected, exploitation takes place. Local search technique tries to exploit the best points inside the detected promising areas. This algorithm stops iterating once the termination condition is satisfied. The main procedure of the proposed algorithm is given in Alg. 1. The main components are described in details in the following subsections.

Algorithm 1 The main procedure of the proposed two-phase heuristic.

Input: C matrix for the application, D matrix for the mesh, n problem scale
Output: the best solution π^*

1. begin
2. $\pi^0 = \text{Initialize}(C, D, n)$
3. $\pi^0 = \text{ReLocate}(\pi^0)$
4. $\pi^0 = \text{LS}(\pi^0)$
5. $\pi^* = \pi^0$, $H = H_{min}$
6. while stopping criterion has not been fulfilled do
7. /* the first phase */
8. $\pi^0 = \text{Perturb}(\pi^0, H)$
9. $\pi^0 = \text{LS}(\pi^0)$
10. if $f(\pi^*) < f(\pi^0)$ then $f(\pi^*) = f(\pi^0)$, $H = H_{min}$ endif
11. /* the second phase */
12. $\pi^0 = \text{IPLS}(\pi^0, \pi^*, H)$
13. if π^* not improved for λ iterations then $H = \min(H + \delta, n)$ endif
14. endwhile
15. end

DOI: 10.1587/elex.13.20151097
Received December 21, 2015
Accepted January 6, 2016
Publicized January 25, 2016
Copyedited February 10, 2016

4.1 Solution initialization

At the beginning of the algorithm, a random solution is generated. The initial solution π^0 has a great impact on the performance of the algorithm. To reduce the performance dependency on π^0 and maintain the stability of the algorithm, we provide a relocating mechanism (ReLocate) for solution construction. For $\pi^0 = (\pi^0(0), \pi^0(1), \pi^0(2), \pi^0(3), \ldots, \pi^0(n-1))$, the ith element of π^0 will be relocated in the following steps:

Step 1: $s^0 = \pi^0$, generate i new solutions (labeled with $s^1, s^2, \ldots, s^k$) by moving $\pi^0(i)$ ahead with 1, or 2, …, or i positions in π^0;

Step 2: choose the best solution s^* from the set $\{s^0, s^1, s^2, \ldots, s^k\}$;

Step 3: replace π^0 with s^*.

In step 1, the operation of moving $\pi^0(i)$ ahead with k positions incurs k exchanges, that is, (1) exchange $\pi^0(i)$ with $\pi^0(i-1)$, then (2) exchange $\pi^0(i-1)$ with $\pi^0(i-2)$,…, and finally (k) exchange $\pi^0(i-k+1)$ with $\pi^0(i-k)$. After relocating $\pi^0(i)$, the updated solution, other than the original π^0, becomes a new starting point to relocate the next element. The initialization of the algorithm is ended by applying local search (see Line 4 of Alg. 1) on the resultant relocating solution.

4.2 The first phase

The proposed heuristic algorithm triggers a perturbing process and a local searching process in turn in the first phase. The perturbation mechanism plays a crucial role since the local search alone cannot escape from the local optima. In the perturbing process, three different perturbation operators (row-cross-based, column-cross-based and exchange-based) are adopted. For a solution π, we randomly choose a perturbation operator and perform the chosen operator on π for H times to generate a new disturbed solution. The parameter H is set to be H_{min} initially, and it will be increased by δ if the global best solution has not been updated for λ iterations (See Line 13 of Alg. 1). Whenever the global best solution is updated, H is reset to be H_{min} (See Line 10 of Alg. 1).

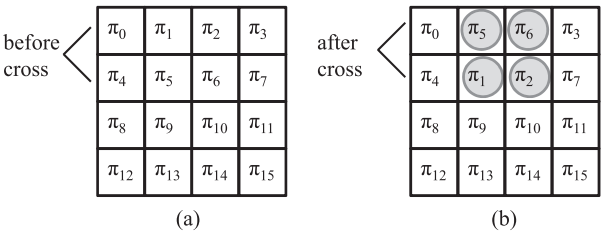

Fig. 2. Row-cross-based perturbation: (a) before cross, (b) after cross

Suppose that an application is mapped onto an $n = k \times k$ mesh, a solution $\pi = (\pi_0, \pi_1, \pi_2, \ldots, \pi_{n-1})$ can be divided into k parts according to the mesh topology. That is, the first k elements correspond to the first row, the subsequent k elements correspond to the second row, an so on. Fig. 2(a) shows an example of dividing solution π into four rows for 4×4 mesh. In the row-based-cross operator, we first randomly select two different rows, and then employ a standard uniform

DOI: 10.1587/elex.13.20151097
Received December 21, 2015
Accepted January 6, 2016
Publicized January 25, 2016
Copyedited February 10, 2016

cross on the two rows. Fig. 2(b) shows an example to perform row-based-cross operation on π with rows 1 and 2 selected. The column-based-cross operator executes in a similar way. As for the exchange-based operator, we employ a simple 2-exchange method, i.e., randomly swap two different elements in the solution.

Local search plays an important role in the performance of the heuristic algorithm, which usually iteratively improves a solution by exploring its neighborhood. The following four components are critical in the implementation of a local search operator. The first is the starting solution. In the first phase of Alg. 1, the resultant solution in the perturbing process becomes the point for local search (LS). The second one is the move used to generate neighboring solutions for a given solution. We adopt the simplest 2-opt move to generate new neighboring solutions. The third one is the acceptance criterion. First-accept (FA) and best-accept (BA) are two popular acceptance criteria. The FA criterion accepts the first neighboring solution which results in an objective increment after moving. The BA criterion checks all neighboring solutions and accepts the best one among them. In LS, the FA criterion is used. The fourth is the stopping condition of local search operator. LS will be ended when no more improvement can be achieved.

4.3 The second phase

In the second phase, we introduce a perturb-based iterated local search (PILS), which alternates between a perturbation process and an iterated local search phase.

A local optima respect to a given neighborhood N is the solution π^{lb} such that $\forall\, \pi \in N(\pi^{lb}),\ f(\pi^{lb}) \leq f(\pi)$, where f is the objective function to be minimized. A basin of a local optimum π^{lb} can be defined as the solution set S that lead the local search to the given local optimum π^{lb}. Basically, our PILS process moves from one basin to another through perturbation operations, and locates to each local optima through iterated local search scheme.

Three types of perturbation operators including directed, LRU, and random are used in PILS. The directed perturbation is based on tabu search principles, which favors the swap move that minimally deteriorates the individual, under the constraint that the move has not been applied during the last γ (where γ is the tabu tenure that takes a random value from a given range) local search iterations. Specially, if a swap move leads to the global π^* updating, it is always accepted regardless of the constraint. The LRU perturbation always selects the least recently performed swap move regardless of the cost degradation by the move. The random perturbation randomly performs a swap move. The perturbation process probabilistically alternates among the three types of perturbations. Usually, a higher probability is given to the directed perturbation. Once the perturbation type is determined, the corresponding perturbations is applied for L times. L is the jump magnitude, which is initially set to be L_{min} and is gradually increased whenever the local search returned to the immediate previous local optimum. As a special case, the perturbation process will terminate whenever the global π^* is improved.

As discussed in subsection 4.2, four components are critical in the local search process. Now we introduce the four components of the iterated local search in PILS as follows. The resultant solution in the perturbing process becomes the point for iterated local search. For a given solution, we also adopt the 2-opt swap move to

DOI: 10.1587/elex.13.20151097
Received December 21, 2015
Accepted January 6, 2016
Publicized January 25, 2016
Copyedited February 10, 2016

Table I. Characteristics of the application benchmarks.

Instance ID	Application	Nodes	Edges
1	PIP	8	8
2	263enc mp3dec	12	12
3	MPEG-4	12	26
4	MWD	12	12
5	mp3enc mp3dec	13	13
6	263dec mp3dec	14	15
7	VOPD	16	21
8	AutoIndustry	24	21
9	Telecom	30	24
10	DVOPD	32	44
11	G20-1	20	19
12	G20-2	20	19
13	G30-1	30	33
14	G30-2	30	33
15	G47-1	47	46
16	G47-2	47	46
17	G62-1	62	61
18	G62-2	62	61

[1]In instances 11, 13, 15, and 17, bandwidths vary from 50 to 150 MB/s.
[2]In instances 12, 14, 16, and 18, bandwidths vary from 10 to 1500 MB/s.

generate the neighborhood. The BA acceptance criterion is used. The whole PILS will terminate when a fixed number of swap moves have been executed.

5 Evaluation and discussion

After proposing and discussing the details of the two-phase heuristic algorithm, we proceed to conduct computational analysis. To test the effectiveness of our algorithm, we conduct experiments on both real benchmarks and custom generated application benchmarks. The real application benchmarks are available in the literature. We use the ten video applications from [17]. Also, we use the TGFF tool [19] to generate the random graphs. According to [17], two groups of bandwidths are selected. Details about these applications are presented in Table I. The former 10 cases are related to real applications, and the latter 8 ones are related to custom generated applications.

5.1 Experimental setup

For the standard real application benchmarks, we collect the results from the literature. Besides, we also implement some latest algorithms such as CastNet [9], GA [8], ILP based exact algorithm for comparisons. The heuristics are coded in C++ and the ILP model is solved by CPLEX. All experiments are run on a PC Intel $i7-4650U$ 1.70 GHz dual-core processor.

Both our proposed two-phase method and GA are transformative heuristics, which solve the problem by transforming the existing solutions to arrive at better ones through multiple iterations. To have a fair comparison, the total running time for them are set to the same value in each testing instance. We divide the testing instances into three groups (small, medium, and large) with respect to the number of cores in the application. For small problems ($N \leq 20$), we set the running time as 1 minute. For medium problems ($20 \leq N \leq 30$), we set the running time as 5 minutes. For large problems ($N > 30$), we set the running time as 10 minutes.

DOI: 10.1587/elex.13.20151097
Received December 21, 2015
Accepted January 6, 2016
Publicized January 25, 2016
Copyedited February 10, 2016

Table II. Comparisons of communication cost for VOPD, MPEG-4, and PIP with ten existing methods.

Methods	VOPD		MPEG-4		PIP	
	Cost	RP	Cost	RP	Cost	RP
ILP [20]	4119	1.000	3567	1.000	640	1.000
PBB [11, 21]	4317	1.048	3763	1.055	640	1.000
PMAP [22]	7054	1.713	6128	1.718	832	1.300
BMAP [16]	4351	1.056	6280	1.761	(-)	(-)
NMAP [15]	4265	1.035	3672	1.029	640	1.000
CHMAP [23]	4249	1.032	3977	1.115	(-)	(-)
CastNet [9]	4135	1.004	3852	1.080	(-)	(-)
Onyx [14]	4249	1.032	3612	1.013	(-)	(-)
CGMAP [24]	4300	1.044	3600	1.009	(-)	(-)
GA [9]	4290	1.042	3567	1.000	(-)	(-)
Our Method	4119	1.000	3567	1.000	640	1.000

[1]Cost is measured by Hops × BW in MB/s.
[2]RP refers to relative percentage between corresponding method and ILP.
[3]The (-) refers to no results reported for the application in the literature.

CastNet is a constructive heuristic with no iterative improvement applied to the final solution. Hence, CastNet usually works much faster than the other methods.

The ILP-based method is an exact algorithm, which requires the most time consumption. We set a time limit of 8 hours on CPLEX optimizer. Apparently, the ILP-based method will always obtain the optimal solution if the optimizer terminates within the given time limit. However, if the optimizer is still searching for a better result after 8 hours, we take the obtained best solution within the time limit. Therefore, it is possible that results of the ILP-based method reported in this section are suboptimal.

5.2 Results on three benchmarks-VOPD, MPEG-4 and PIP

Most of the existing methods have reported results on the three benchmarks-VOPD, MPEG-4 and PIP. We collect the mapping results from the literature and present the cost normalized to ILP method. As shown in Table II, our proposed method produces the same solutions as ILP which yields the optimal solution. Among those methods, ILP belongs to exact mapping techniques, and methods including PMAP, BMAP, CHMAP, Onyx and CastNet belong to constructive heuristics without iterative improvement, and methods including CGMAP, GA belong to constructive heuristics with iterative improvement. Specially, CastNet is an extended method of Onyx by using the symmetry of the mesh topologies [8]. CastNet [8] and GA [9] are relatively new methods and they perform relatively better than the rest heuristics. Besides, the three methods (ILP, CastNet and GA) represent different types of methods. In the following subsection, we implement the three methods and compare our method with them under the 18 testing instances.

5.3 Comparisons with ILP, CastNet and GA under real application benchmarks

Table III reports results comparisons among four methods, including ILP, CastNet, GA and our proposed method. The ILP method yield optimal solutions for the former nine benchmarks, while it could not give the optimal solution for DVOPD within the given time limit. For the former nine benchmarks, our method always

DOI: 10.1587/elex.13.20151097
Received December 21, 2015
Accepted January 6, 2016
Publicized January 25, 2016
Copyedited February 10, 2016

Table III. Comparisons of ILP, CastNet, GA and our method for real application benchmarks.

Instances	ILP [20]		GA [8]		CastNet [9]		Our Method	
	Cost	RP	Cost	RP	Cost	RP	Cost	RP
1	640	1.000	640	1.000	640	1.000	640	1.000
2	230.407	1.000	230.407	1.000	230.407	1.000	230.407	1.000
3	3567	1.000	3567	1.000	3852	1.080	3567	1.000
4	1120	1.000	1120	1.000	1344	1.200	1120	1.000
5	17.021	1.000	17.056	1.002	17.046	1.001	17.021	1.000
6	19.823	1.000	19.906	1.004	19.823	1.000	19.823	1.000
7	4119	1.000	4290	1.042	4135	1.004	4119	1.000
8	131	1.000	135	1.031	131	1.000	131	1.000
9	97	1.000	97	1.000	97	1.000	97	1.000
10	10041	1.000	10006	0.997	9618	0.958	9522	0.948

Table IV. Comparisons of ILP, CastNet, GA and our method for custom generated benchmarks.

Instances	ILP [20]	CastNet [9]	GA [8]	Our method
11	2134*	2350	2313	2134*
12	12990*	12990*	13198	12990*
13	4704	5111	4972	4137*
14	29831	29755	28052	25082*
15	7944	5593	6325	4931*
16	48744	36199	40367	32979*
17	9771	7155	11019	6735*
18	70505	47406	75903	46227*

[1]The value with superscript ∗ is the best one among the four methods.

yield the optimal solution as ILP, while CastNet could produce optimal solutions in only five cases.

5.4 Comparisons with ILP, CastNet and GA under custom generated application benchmarks

For a better comparisons among different methods, we also conduct experiments under the custom generated application benchmarks. Table IV reports the results under the eight custom generated application benchmarks. When the problem scale is small (instances G20-1 and G20-2), the ILP method could yield optimal solutions. When the problem scale grows bigger, the ILP method could not find the optimal result within the given time limit. As shown in Table IV, our method always yield the best solutions among the four methods, which only takes several minutes to determine the solutions. One may opt to these results in Table IV, claiming that the ILP-based method always obtains better solutions than any other methods. That is true if we can obtain the optimal result within the given time limits. However, as we stated in subsection 5.1, we take the obtained best solution if the ILP optimizing tool is still searching for a better result after eight hours. That means, the solutions returned by ILP method may not be suboptimum ones.

5.5 Optimal solutions from our proposed method

As discussed in subsection 5.2, our proposed two-phase heuristic algorithm could always yield the best solutions for the ten real application benchmarks. For illustrative purposes, we give the mapping solutions produced by our method

DOI: 10.1587/elex.13.20151097
Received December 21, 2015
Accepted January 6, 2016
Publicized January 25, 2016
Copyedited February 10, 2016

Table V. Optimal solutions provided by our method.

Application	Cores	Mesh	Mapping Solutions
PIP	8	3 × 3	8 7 6 4 5 1 −1 3 2
263enc mp3dec	12	4 × 4	7 9 8 −1 3 4 1 6 11 12 2 −1 10 −1 5 −1
MPEG-4	12	4 × 4	6 −1 4 9 3 10 5 1 8 7 11 2 −1 12 −1 −1
MWD	12	4 × 4	12 11 9 10 4 5 8 7 −1 1 2 6 −1 −1 3 −1
mp3enc mp3dec	13	4 × 4	11 12 13 10 −1 2 1 9 4 5 3 −1 8 6 7 −1
263dec mp3dec	14	4 × 4	12 −1 6 7 13 −1 4 5 14 11 10 3 9 8 1 2
VOPD	16	4 × 4	10 9 11 15 8 7 12 13 1 6 5 14 2 3 4 16
AutoIndustry	24	5 × 5	24 19 14 13 10 23 18 15 11 9 22 17 16 12 8 21 3 4 5 6 20 2 1 −1 7
Telecom	30	6 × 6	−1 14 16 21 20 1 11 12 15 22 4 2 −1 13 −1 24 3 25 −1 10 9 23 −1 26 6 8 7 30 29 28 5 −1 19 18 17 27
DVOPD	32	6 × 6	−1 22 21 20 19 18 25 23 26 29 30 17 24 −1 32 27 28 16 10 9 11 14 12 31 8 7 6 5 13 1 −1 −1 15 4 3 2

[1]The solution structure is in accordance with that described in section 3.

under different benchmarking cases. As shown in Table V, the first column indicates the benchmark, while the second and third columns report the number of cores in the benchmark and the selected mesh platform. The fourth column of the table shows the solution. As discussed in section 3, the number of cores in the application is less or equal to the number of routers in the NoC platform. Thus, it may happen that some router has no core to be mapped to. In this situation, the corresponding element is filled with −1.

6 Conclusion

Application mapping is one of the most important dimensions in NoC design paradigm. The problem is to determine a minimal cost assignment of m cores in the application to the n routers in the NoC platform, which is proven to be NP-hard. This paper presents a simple and effective two-phase heuristic algorithm to solve the problem. Both of the two phases adopt the combination of perturbation and local search operations to improve the efficiency of the algorithm. To verify the effectiveness of the algorithm, a quantitative comparisons have been discussed between the proposed algorithm and the existing mapping methods under both real application and custom generated application benchmarks. The results show that the proposed method is able to attain the optimal solutions for the ten real application instances. Compared to CastNet and GA, the proposed method could yield the better solutions, thus it competes very favorably with the current mapping approaches.

Acknowledgments

This work is supported by the National Nature Science Foundation of China (No. 61402086), Scientific Research Foundation of Liaoning Provincial Education Department (No. L2015165), and DUFE Excellent Talents Project (No. DUFE2015R06). Here, we also would like to give our special thanks to professor Suleyman Tosun for his suggestions on the experiments.

DOI: 10.1587/elex.13.20151097
Received December 21, 2015
Accepted January 6, 2016
Publicized January 25, 2016
Copyedited February 10, 2016