TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol. 12, No. 11, November 2014, pp. 7869 ~ 7875
DOI: 10.11591/telkomnika.v12i11.6640

An Adaptive Genetic Algorithm for Mesh-Based NoC Application Mapping

Yizhuo Wang, Zhibiao Zhang*
School of Compute Science and Technology, Beijing Institute of Technology,
Beijing, 10081, China
*Corresponding author, e-mail: carldotz@163.com

Abstract

Application mapping is one of the key problems of Network-on-Chip (NoC) design. It maps the cores of application to the processing elements of the NoC topology. This paper presents a novel approach for NoC application mapping, which uses adaptive genetic algorithm (AGA) in the mapping. The proposed approach adaptively varies the probabilities of crossover and mutation operators in genetic algorithm, aiming to reduce the overall communication cost of NoC. Experimental results show that the proposed approach decreases the communication cost by 3% to 7% on average, compared to the existing approach using Standard Genetic Algorithm (SGA).

Keywords*: network-on-chip, application mapping, adaptive genetic algorithm, genetic algorithm*

1. Introduction

Network-on-Chip (NoC) is the design of modular and scalable communication architectures where various processing elements are connected to a router-based network using appropriate network interface [1-3]. Since application Mapping has direct effect on delay, power consumption and other performance of NoC, it is a very significant aspect of NoC design [4]. In its general form, the problem of NoC application mapping is an NP-hard problem [4].

Genetic Algorithm (GA), a stochastic search algorithm based on natural genetic operations provides a solution to the problem of NoC mapping. Sometimes the performance of Standard Genetic Algorithm (SGA) based NoC mapping cannot meet the need of practical work because the parameters of genetic operation are fixed values. Therefore, several adaptive genetic algorithms based on SGA are proposed. The Adaptive Genetic Algorithms (AGA) proposed by M. Srinivas and L. M. Patnak [5] is considered to be the most representative, and it realizes the twin goals of maintaining the diversity of population and sustaining the capacity of convergence with the use of adaptive probabilities of crossover and mutation.

In this paper, we use AGA in the NoC application mapping. First, we extend the representation for chromosomes proposed by W. Zhou, et al [6] to a general situation in which the number of application cores is less than or equal to the number of processing elements (PEs) of the NoC topology. Second, our mapping approach reduces the communication cost with an improved adaptive genetic algorithm, F-AGA, which is proposed by X. Cheng [7], and it amends the expressions for crossover probability *pc* and mutation probability *pm* to solve the problem that *pc* and *pm* are zero for the solution with the max fitness. Experimental results show that our approach decreases the communication cost by 3% to 7% on average, compared to the approach using Standard Genetic Algorithm (SGA).

The rest of this paper is organized as follows. Section 2 introduces the related work in NoC mapping and GA briefly. Section 3 presents the definitions used in this paper and describes our AGA based approach. Section 4 shows the experimental results, and Section 5 summarizes our main contribution.

2. Related Work

The NoC application mapping techniques can be classified as dynamic mapping [8-10] and static mapping, depending on the time at which the cores of application are assigned to the

Received July 10, 2014; Revised September 4, 2014; Accepted September 25, 2014

PEs of the NoC topology. Static mapping is generally recommended, as excess communication cost in dynamic mapping significantly affects overall performance of system [11]. The static mapping techniques mainly include Integer Linear Programming (ILP) [12], Branch-and-Bound (BB) [13], Particle Swarm Optimization (PSO) [14], Ant Colony optimization (ACO) [15], Simulated Annealing (SA) [16], Genetic Algorithm (GA), etc. The rest of this section will focus on the related work of GA for NoC mapping.

Lei et al. propose a two-step Genetic Algorithm (GA) for NoC mapping, which reduces the overall execution time [17]. In the first step, the tasks are assigned onto different Intellectual Properties (IPs), assuming that the edge delays are constant and equal to the average value of the edge delays. In the second step, the IPs are mapped to the PEs of NoC, taking the actual edge delay based on the network traffic model, and the total system delay is minimized [11]. Zhou et al. propose a genetic algorithm based on a delay model for NoC mapping [6]. Zhou et al. also propose a representation for the chromosomes to ensure that the offspring will meet the constraint, a one-to-one constraint between IP cores and PEs. A pareto based multi-objective evolutionary computing technique is proposed by Ascia et al. [18], which optimizes performance and power consumption of NoC. Jena et al. propose MGAP, a genetic algorithm based optimization technique, which minimizes the power consumption and maximizes the throughput by reducing the number of switches in the communication path between cores [19].

The major drawback of the genetic algorithms is the slow rate of convergence. It often requires the GA to evolve a large number of generations to converge to a solution. The best solution at the end is taken to be the solution of the mapping problem. To accelerate the rate of convergence, the mutation rate can be increased. However, it mostly converges to local best solutions, rather than finding the global best [11]. AGA has better capability of locating the global optimum than SGA. However, we have not found any literature that has used AGA in NoC mapping.

3. Technique

In this section, we start with the definitions used in this paper. Then we describe the adaptive genetic algorithm for mesh-based NoC mapping. The representation of population (chromosomes) and the expressions of probability are very important aspects of AGA. We introduce them separately.

Definition 1: Application Characteristic Graph ACG (V, E) is a directed graph, where each vertex $c_i \in V$ represents an IP core and each edge $e_{i,j} \in E$ represents the communication bandwidth between a source core (c_i) and a destination core (c_j). Each $e_{i,j}$ is tagged with v_{ij} which represents the communication volume from c_i to c_j.

Definition 2: NoC Architecture Graph NAG (R, C) is a directed graph, where each vertex $r_i \in R$, represents a PE. Each directed edge $c_{i,j} \in C$ represents a physical unidirectional channel which connects an output port of r_i to an input port of r_j.

Definition 3: Virtual IP Core (VIC) is an IP core that does not exist in ACG. The communication volume from a VIC to an IP core is always zero, as is the communication from an IP core to a VIC.

Definition 4: Extended Application Characteristic Graph EACG (C, E) is a directed graph, where each vertex $c_i \in V$ represents an IP core or a VIC and each edge $e_{i,j} \in E$ represents the communication bandwidth between a source core (c_i) and a destination core (c_j). Each $e_{i,j}$ is tagged with v_{ij} which represents the communication volume from c_i to c_j.

3.1. Population Representation

NoC mapping has a one-to-one constraint between IP cores and PEs in NoC application mapping, which means that different IP cores cannot be mapped to the same PE and different PEs cannot map the same IP core. Thus, some offspring will violate the constraint if they are generated by parents randomly. In order to overcome this problem, Zhou et al. [6] has proposed a convenient population representation scheme. Using this scheme, the offspring will satisfy the constraint as long as the parents do. However, the situation in which the number of IP cores is less than the number of PEs has not been considered in this scheme.

An extended population representation scheme is suggested here. This scheme is based on the scheme in [6], and is able to represent the situation in which the number of IP cores is less than the number of PEs. The mapped 2D-mesh NoC is represented by one-

dimensional array using a left to right scan performed in a top-down fashion. The chromosome is an integer array, and the value at the i^{th} position denotes the mapping of the i^{th} IP core or VIC. For the value of the i^{th} position an integer between 1 and i is allowed.

Next, we illustrate the decoding procedure of the chromosome structure with the help of an example. Figure 1(a) shows the ACG of an application with 7 IP cores, and Figure 1(b) shows the NAG of a 3x3 2D-mesh NoC.

(a) ACG (b) NAG

Figure 1. ACG with 7 IP Cores and 3x3 2D-mesh NAG

The first step is to construct EACG from ACG. In this example the number of PEs minus the number of IP cores is 2. Thus, two VIC, 'h' and 'I', should be added to the EACG. Figure 2 shows the EACG.

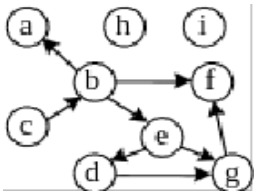

Figure 2. EACG

Then, for a chromosome structure (1, 2, 2, 4, 5, 4, 7, 1, 3), the first integer is applied to map IP core 'a'; the second integer is applied to map IP core 'b'; and so on. Clearly, the last two integers are applied to map the VIC 'h' and 'i'. The value of the first position is '1', and all solutions will have a value '1' into the first position. The second integer is applied to map 'b', and there are two situations: value '1' means 'b' is before 'a' and value '2' means 'b' is after 'a'. The second integer in the chromosome structure is '2'. So [a, b] is obtained. Similarly, the third integer is applied to map 'c', and there are three situations: value '1' means 'c' is before 'a', value '2' means 'c' is between 'a' and 'b' and value '3' means 'c' is after 'b'. The third integer in the sample chromosome structure is '2'. So [a, c, b] is obtained. Continuing in this fashion, the following permutation of the nine alphabets [h, a, i, c, b, f, d, e, g] is obtained. Cores are placed in a 3x3 2D-mesh NoC according to this permutation as shown in Figure 3 (a). Finally, VICs 'h' and 'i' should be removed from the core placement. Then the final core placement is obtained in Figure 3 (b).

(a) Core placement (b) The final core placement

Figure 3. Core Placement of a Sample Chromosome Structure

Note that this extended population representation scheme can easily deal with the above constraint condition even if the number of IP cores is less than the number of PEs.

3.2. Expressions of Probability

In the AGA proposed by M. Srinivas [5] the probabilities of crossover and mutation are adjusted by the expression (1) and (2).

$$P_c = \begin{cases} k_1 \times \left(\dfrac{f_{max} - f'}{f_{max} - f_{avg}} \right), f' > f_{avg} \\ k_2, f' \le f_{avg} \end{cases} \quad (1)$$

$$P_m = \begin{cases} k_3 \times \left(\dfrac{f_{max} - f}{f_{max} - f_{avg}} \right), f > f_{avg} \\ k_4, f \le f_{avg} \end{cases} \quad (2)$$

Where P_c is the probability of crossover; P_m is the probability of mutation; is the maximal fitness; is the average fitness; is the bigger fitness value between two parents; is the fitness value of the mutated individual; k_1, k_2, k_3 and k_4 are constants between 0 and 1.

The AGA proposed in [5] is more propitious to the probability adjustment of the later period evolution, because the probability of crossover and mutation are zero for the best solution. Here we introduce an improved adaptive genetic algorithm F-AGA [7], and the probability expressions can be described as

$$P_c = \begin{cases} P_{c_{max}} - \left(\dfrac{P_{c_{max}} - P_{c_{min}}}{f_{max} - f_{avg}} \right) \times \left(f' - f_{avg} \right), f' > f_{avg} \\ P_{c_{max}}, f' \le f_{avg} \end{cases} \quad (3)$$

$$P_m = \begin{cases} P_{m_{max}} - \left(\dfrac{P_{m_{max}} - P_{m_{min}}}{f_{max} - f_{avg}} \right) \times \left(f_{max} - f \right), f > f_{avg} \\ P_{m_{max}}, f' \le f_{avg} \end{cases} \quad (4)$$

Where P_c is the probability of crossover; P_{c_max} is the maximal crossover probability; P_{c_min} is the minimal crossover probability; P_m is the probability of mutation; P_{m_max} is the maximal mutation probability; P_{m_min} is the minimal mutation probability; is the maximal fitness; is the average fitness; is the bigger fitness value between two parents; is the fitness value of the mutated individual. The parameters are set as P_{c_max} = 0.9, P_{c_min} = 0.6, P_{m_max} = 0.2, P_{m_min} = 0.01.

3.3. The Other Aspects of the Algorithm

a) Initial population

The initial population is generated randomly. Note that the integer value at the i^{th} position of each individual in the initial population must between 1 and i.

b) Fitness function

The goal of this paper is to reduce the communication cost (CommCost) measured by

$$CommCost = \sum_{comm\_ \in Comm} H_{ij} \times comm_{ij} \quad (5)$$

Where H_{ij} is the Manhattan distance between source core 'i' and destination core 'j', and the distance between two adjacent PEs is one hop. $comm_{ij}$ is the requirement of communication bandwidth from source core 'i' to destination core 'j'.

c) Reproduction

Tournament selection is used in this paper. Tournament selection randomly samples k individuals from parent population, and then selects the best one into the mating pool. In this paper k is set to 2.

d) Crossover

Our experiments show that the uniform crossover is better than single-point crossover and multi-point crossover. So we use uniform crossover in this paper. After calculating crossover probability, a uniform crossover is done with the new crossover probability.

e) Mutation

After calculating mutation probability a mutation is done by selecting a position 'i' at the representation of chromosome with the new crossover probability, and the integer value assigned at i^{th} position is a random integer between 1 and i.

f) Elite-preservation strategy

In order to speed up the convergence rate of AGA and ensure that the offspring is always better than the parent, elite-preservation strategy is used in this paper. Elite-preservation strategy ensures the best individual always survives to be an individual of the next generation. Then, the best individual will replace the worst individual of next generation in the AGA proposed in this paper.

g) Terminal criterion

The termination criterion is set to be 500 generations.

4. Experimental Results

To evaluate the performance of the proposed AGA based NoC application mapping approach, we implemented the AGA and the SGA in C++. The crossover probability and the mutation probability of SGA are fixed as P_c = 0.9 and P_m = 0.05.

We generate 6 ACGs with TGFF [20] randomly, and generate 6 NAGs for the ACGs. The number of cores and the number of PEs of the 6 test cases are shown in Table 1.

Table 1. The 6 Mapping Cases

Cases	Number of IP cores	Number of PEs
1	60	8*8
2	46	7*7
3	20	5*4
4	19	8*8
5	44	7*7
6	19	4*5

We made 100 simulation runs for each case in Table 1. Considering the randomization effect of SGA and AGA, the algorithms were executed for 500 generations in each simulation run.

Figure 4 shows the communication costs at each generation of the genetic algorithms for the test case 1. Each point in the figure represents the average value of 100 simulation runs. From the figure, we can see that the AGA outperforms the SGA, and the SGA tends to get stuck at a local optimum at point A while the AGA does until point B.

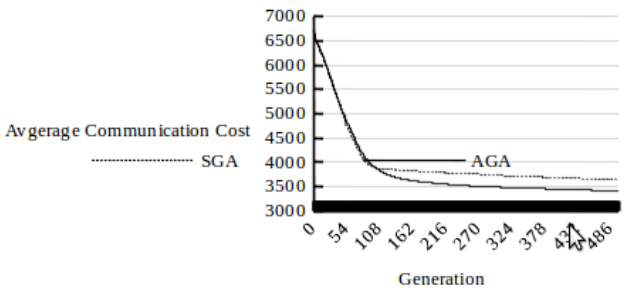

Figure 4. Variations of the Communication Costs (test case 1)

Table 2 shows the final result of the mapping cases in Table 1. As we can see from Table 2, after 500 generations, the AGA based approach decreases the communication cost by 3% to 7% on average, compared to the SGA based approach.

Table 2. Communication Costs of the Final Mappings

Cases	SGA	AGA	AGA/SGA
1	3645.63	3416.16	0.93705615
2	4025.68	3824.88	0.95012023
3	994.962	978.439	0.98339334
4	1812.01	1769.11	0.97632463
5	2828.26	2726.67	0.96408039
6	1671.73	1649.82	0.98689382

5. Conclusion

In this paper an adaptive genetic algorithm (AGA) based approach for NoC application mapping is proposed. We present a population representation scheme which can easily deal with the constraint condition in NoC application mapping even if the number of IP cores is less than the number of PEs. The proposed approach adaptively varies the probabilities of crossover and mutation operators. Our experiments show that the AGA enhances the capability of premature convergence prevention and produces lower communication cost than SGA.

The proposed approach only considers the communication cost. In future work, the approach will be extended by considering the other aspects of NoC such as the network delay.

Acknowledgement

The authors thank the anonymous reviewers for their valuable comments on the manuscript. This work was partially supported by the National Natural Science Foundation of China under grant NSFC- 61300011.

References

[1] L Benini, G De Micheli. Networks on Chips: a new SoC paradigm. *IEEE Computer.* 2002; 35(1): 70–78.
[2] WJ Dally, B Towles. *Route packets, not wires: on-chip interconnection networks*. Proceedings of the 38th Design Automation Conference. Las Vegas, USA. 2001: 684–689.
[3] S Kumar, A Jantsch, JP Soininen, M Forsell, M Millberg, J Oberg, K Tiensyrja, A Hemani. *A network on chip architecture and design methodology*. Proceedings of ISVLSI. Pittsburgh, USA. 2002: 117–124.
[4] R Pop, S Kumar. A survey of techniques for mapping and scheduling applications to network on chip systems. School of Engineering, Jönköping University. Report Number: 04:4. 2004.
[5] M Srinivas, LM Patnak. Adaptive Probabilities of Crossover and Mutation in Genetic Algorithm. *IEEE Trans. on Systems, Man and Cybernetics.* 1994; 24(4): 656 - 666.
[6] W Zhou, Y Zhang, Z Mao. *An application specific NoC mapping for optimized delay.* IEEE International Conference on Design and Test of Integrated Systems in Nanoscale. La Marsa, Tunisia. 2006: 184–188.
[7] Cheng Xian-yi, Yang Chang-yu. *Robot path planning based on adaptive isolation niche genetic algorithm*. Proceedings of 2nd International Symposium on Intelligent Information Technology Application. Shanghai, China. 2008: 151-154.
[8] G Chen, F Li, M Kandemir. *Compiler-directed application mapping for NoC based chip multiprocessors*. Proceedings of LCTES. San Diego, California, USA. 2007: 155–157.
[9] CL Chou, UY Ogras, R Marculescu. Energy- and performance-aware incremental mapping for NoCs with multiple voltage levels. *IEEE Transactions on Computer-Aided design of Integrated Circuits and Systems*. 2008; 27(10): 1866–1879.
[10] E Carvalho, F Moraes. *Congestion-aware task mapping in heterogeneous MPSoCs*. International Symposium on SoC. Tampere, Finland. 2008: 1–4.
[11] PK Sahu, S Chattopadhyay. A survey on application mapping strategies for network-on-chip design. *J. Syst. Archit..* 2013; 59(1): 60–76.

[12] C Ostler, KS Chatha. *An ILP formulation for system-level application mapping on network processor architecture.* Proceedings of Design, Automation and Test in Europe (DATE). Nice, France. 2007: 1–6.
[13] J Hu, R Marculescu. *Energy-aware mapping for tile-based NoC architectures under performance constraints.* Asia and South Pacific Design Automation Conference (ASP-DAC). Kitakyushu, Japan. 2003: 233–239.
[14] I Kennedy, RC Eberhart. *Particle swarm optimization*. Proceedings of IEEE International Conference on Neural Networks. New Jersey, USA. 1995: 1942–1948.
[15] A Colorni, M Dorigo, V Maniezzo. Distributed optimization by ant colonies. actes de la première conférence européenne sur la vie artificielle. Paris, France. 1991: 134–142.
[16] HM Harmanani, R Farah. *A method for efficient mapping and reliable routing for NoC architectures with minimum bandwidth and area.* IEEE International Workshop on Circuits and systems and TAISA Conference (NEWCAS-TAISA). Los Alamitos, CA. 2008: 29–32.
[17] T Lei, S Kumar. *A two-step genetic algorithm for mapping task graphs to a network on chip architecture*. Proceedings of the Euromicro Symposium on Digital System Design (DSD). Belek-Antalya, Turkey. 2003: 180–187.
[18] G Ascia, V Catania, M Palesi. *Multi-objective mapping for mesh-based NoC architectures*. ACM International Conference on Hardware/Software Codesign and System Synthesis. Stockholm, Sweden. 2004: 182–187.
[19] RK Jena, GK Sharma. *A multi-objective evolutionary algorithm based optimization model for Network-on-Chip synthesis.* IEEE International Conference on Information Technology (ITNG). Las Vegas, Nevada, USA. 2007: 977–982.
[20] RP Dick, DL Rhodes, W Wolf. *TGFF: task graphs for free.* Proceedings of the 6th International Workshop on Hardware/Software Co-Design. New York, USA. 1998: 97-101.