NoC Multi-Objects Mapping Based On Enhanced Chaos Discrete Artificial Bee Colony Algorithm

Lili Pan, Zhe Li, and Xiang Ling

Abstract—**Mapping is a key issue in NoC system-level design process. The power consumption and delay of a NoCsystem are primary determined by the mapping result. Since NoC mapping is a NP-Complete problem, which means it beyond the computation ability to obtain the global optimal solution if the size of problem is too large.Approaches in this area focuses on heuristic algorithms. In this paper, a new swarm-intelligence algorithm - Enhanced Chaos Discrete Artificial Bee Colony (ECDABC) algorithm is introduced to solve this problem,which carries out two-step optimization processes. Moreover, the enhanced chaos mechanismin can effectively avoidthe defection of premature convergence which are common in most of heuristics. By comparing with two prevalent mapping algorithms – Genetic Algorithm andParticle Swarm Optimazation algorithm,ECDABC algorithm has better performance on the optimized results in the same iterations times, and enhanced chaos mechanism has significant effect.**

Index Terms—**Networks-on-chip, mapping, multi-objective optimization, artificial bee colony algorithm.**

I. INTRODUCTION

Networks-on-Chip(NoC) is considered as a subset of System-on-Chip (SoC), which combines the state-of-the-art techniques in computer network with that in chipdesign[1]-[4]. It has been regarded as a promising technique to solve the bus-based structure issues. With the development of very-large-scale integrated circuits (VLSI), data transmission among IP cores in NoC network augments significantly. In addition to this, the delay of data transmission also needs to be shortened in order to ensure the quality of service (QoS). And thus transmissiondelay and power consumption have become two non-ignorable problems which attract designers' attention currently.

Mostpublicationsminimize the power consumption regardless of delay totally or partly [5]. Yet from what has been discussed above, we can clearly see that low intra-coresdelay is as important as low power consumption.However, someliteraturestry to solve these two problems simultaneously [6]-[7]. No doubts, NoC mapping mechanism plays a crucial role in optimizing intra-cores delay and power consumption [8].

Manuscript received April 10, 2012; revised May 14, 2012.

The authors are with the National Key Laboratory of Science and Technology on Communication, UESTC Chengdu, China (Tel: +86-02861830326, Fax: +86-02861830326, e-mail: xiangling@uestc.edu.cn, strawberrypll@yahoo.com, cnlizhe@uestc.edu, cnxiangling@uestc.edu.cn).

However, mapping is considered as a NP-complete problem [9]-[12], which is hard to get an optimal solution. Various options are available for NoC mapping according to various applications.Fig.1 illustrates the NoC mapping procedures, which directly map the tasks in application characterization graph (ACG) on resource tiles in NoC platform.

In this paper, a heuristic mapping algorithm to balance delay and power consumption in 2D mesh NoC architectureis proposed. The proposed algorithm get a better mapping result than Particle Swarm Optimazation(PSO) [12] and Genetic Algorithm (GA)[10] in the same iterative times and effectively avoid being trapped into local optimal solution by introducing chaos mechanism.

The rest of the paper is organized as follows. Section 2gives some preliminaries, includingNoC architecture and routing strategy, energy and delay models, problem formulation and related works. Section 3proposesEnhanced Chaos Discrete Artificial Bee Colony (ECDABC)algorithm. In Section 4, we present experimental results and draw conclusion in Section 5.

II. PRELIMINARIES

A. NoC Architecture and Routing Strategy

In this paper, we focus on the 2D mesh topology NoC architecture due to its simplicity and scalability. Fig 1 (b) showstypical 2D mesh architecture, from which we can see that each processing element (PE) is connected to a router and each router is connected to its neighbors.

Before we formally introduce the energy and delay model, we deem that it is necessary to explain ACG and NoC architecture graph (NAG) first. ACG is a weighted direct acyclic graph (as Fig 1 (a) shown) which is represented by symbol$G_{ACG}(V, E)$. Vertex $v_i \in V$ represents a task waiting to be mapped; edge $e_{ij} \in E$ represents data direction from task v_i to v_j ; w_{ij} represents data communication traffic between task $v_i$andv_j.For an ACG containing n nodes, we can use a $n \times n$ matrix C_c to represent. $c_{ij} \in C_c$, whose value is equal to e_{ij} in $G_{ACG}(V, E)$, is stand for data traffic between task i and task j.

NAG is a directed acyclic graph as an abstraction of NoC platform. NAG can be represented as $G_{NAG}(R, P)$, $r_i \in R$ represents an available resource tile and $r_{ij} \in P$ represents the path from resource tile r_i to r_j. Notice that number of resource tiles should no less than the number of tasks.

Consequently, a mapping solution can be represented as a $n \times n$ matrix P whose rows and columns stand for resource tiles and tasks respectively, i.e., $p_{ij} = 1$ means the *j*-th task is mapped to the *i*-th resource tile so that only one element

can be equal to 1 in every row and column of P .Thus mapping result can be represented as below:

$$\mathrm{Map}(P, \mathrm{C}_c) = P^T \mathrm{C}_c P \quad (1)$$

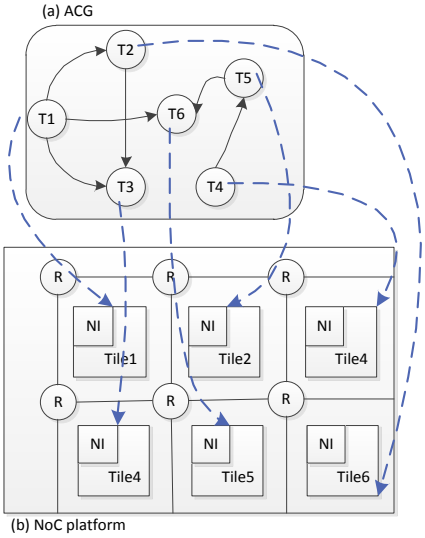

Fig.1. NoC mapping process

Task	1	2	3	4	5	6	...	N
Resource	1	2	3	4	5	6	...	N

Fig. 2. Task-resource List

Current sequence	1	2	3	4	5	6	...	N
Sequence after enhanced chaos	7	3	m	6	5	2	...	4

Fig. 3. Enhance-chaos-sequence List

B. Power Consumptionmodel

The power consumption of NoC architecture can be divided into two parts: computation power consumption and communication power consumption.

$$\text{Power} = \sum_i P_i^{core} + \sum_j P_j^{path} \quad (2)$$

where P_i^{core} represents the computation power consumption of taski in ACG and P_j^{path} represents transm $Delay = \sum_i A_i \cdot T_{ip} + \sum_i w_j\{(|x_m - x_n| + |y_m - y_n| + 1) \times d_{R_{bit}} + (|x_m - x_n| + |y_m - y_n|) \times d_{L_{bit}}\}$ itting power consumption of edgej. Actually, the computation power consumption is determined by both task scale (A_i) and IP core type (T_{ip}). Consequently, P_i^{core} can be formulated as (notice that T_{ip} is a constant):

$$P_i^{core} = A_i \cdot T_{ip} \quad (3)$$

Communication power consumption of j-th path from resource tile $m(x_m, y_m)$ to tile $n(x_n, y_n)$ via path p_{mn} can be formulated as:

$$(|x_m - x_n| + |y_m - y_n| + 1) \times P_{S_{bit}} + (|x_m - x_n| + |y_m - y_n|) \times P_{L_{bit}}\} \quad (4)$$

where $P_{bit}^{m,n}$ is communication power consumption per bit, $P_{S_{bit}}$ is switch power consumption, and $P_{L_{bit}}$ as link consumption between two adjacent resource tiles.

Combining formula (2), (3) and (4), we may finally attain power consumption model:

$$Power = \sum_i A_i \cdot T_{ip} + \sum_j w_j\{(|x_m - x_n| + |y_m - y_n| + 1) \times P_{S_{bit}} + (|x_m - x_n| + |y_m - y_n|) \times P_{L_{bit}}\} \quad (5)$$

C. Delay Model

Define edge set $\{e_{i,k,\ldots,j}\}$ as a path from v_i to v_j in $G_{ACG}(V, E)$ if it exists. Like power consumption, the path delay can also be divided into computation delay and transmitting delay:

$$\text{Delay} = \sum_i Delay_i^{core} + \sum_j Dealy_j^{path} \quad (6)$$

where Delay_i^{core} represents the computation delay of taski in ACG and Dealy_j^{path} represents transmitting delay on edgej.Similarly,Delay_i^{core} can be formulated as

$$Delay_i^{core} = A_i \cdot T_{ip} \quad (7)$$

The transmitting delay on edge j, from resource tile $m(x_m, y_m)$ to tile $n(x_n, y_n)$ via path p_{mn}, can be represented as:

$$\{(|x_m - x_n| + |y_m - y_n| + 1) \times d_{R_{bit}} + (|x_m - x_n| + |y_m - y_n|) \times d_{L_{bit}}\} \quad (8)$$

where w_j is the weight of edge, and $d_{L_{bit}}$ and $d_{R_{bit}}$ to represent physical link delay and router's processing delay per bit accordingly. Combining formula (6), (7) and (8), we can attain delay model:

$$\text{Delay} = \sum_i A_i \cdot T_{ip} + \sum_i w_j\{(|x_m - x_n| + |y_m - y_n| + 1) \times d_{R_{bit}} + (|x_m - x_n| + |y_m - y_n|) \times d_{L_{bit}}\} \quad (9)$$

Define themaximal delay path as critical path, and thus system delay of NoC is equal to critical path delay.

$$Delay_{NoC} = \max(\text{Delay of all path}) = \text{Delay}_{CP} \quad (10)$$

D. Problem Formulation

Since our purpose is to minimize power consumption and delay simultaneously, we cannot use separated computation model to analysis and evaluate mapping algorithm, i.e., multi-objective optimization (MOP) model is used in this paper.

We convert multi-object into single object by using weighting method, which will be elaborated later. Define evaluation functionbelow:

$$\mathrm{F} = \alpha \cdot \mathrm{N}(f_{power}) + (1-\alpha) \cdot \mathrm{N}(f_{delay}) \quad (11)$$

$$\mathrm{N}(f) = \frac{f - f_{min}}{f_{max} - f_{min}} \quad (12)$$

where$\alpha \in [0,1]$ represents weighting coefficient and $\mathrm{f_{power}}$ and $\mathrm{f_{delay}}$ represent power consumption and delay mentioned in formula (5) and (9) respectively. N(f) is normalized function whose effect is to compel power consumption and delay in the same order of magnitude.In this paper,$\mathrm{f_{min}} = 0$,$\mathrm{f_{max}}$ is theexperienced maximum value of power consumption or delayobtained from plenty of experiences.

Given $\mathrm{G_{ACG}(V,E)}$ and $\mathrm{G_{NAG}(R,P)}$, for $\forall \mathrm{v_i} \in \mathrm{V}$, exists vector $\mathrm{E_i} = \{\mathrm{e_i^1}, \mathrm{e_i^2}, \cdots \mathrm{e_i^j}, \cdots \mathrm{e_i^{|R|}}\}$ and $\mathrm{T_i} = \{\mathrm{t_i^1}, \mathrm{t_i^2}, \cdots \mathrm{t_i^j}, \cdots \mathrm{t_i^{|R|}}\}$ standing for energy consumption and time consumption of *i*-th task running in each resource tile of NAG. If *i*-th task cannot run in *j*-thtile, we set $\mathrm{e_i^j} = \infty$and $\mathrm{t_i^j} = \infty$.

Our purpose is to

$$\min_{map\,(P,C_c)} (F) \quad (13)$$

under constrains:

$$\mathrm{Size(NAG)} \geq \mathrm{Size(ACG)}, \forall p_{ij} = 1, \{e_i^j \neq \infty, t_i^j \neq \infty\} \quad (14)$$

III. Enhanced Chaos Discrete Artificial Bee Colony Algorithm

A. Standard Artificial Bee Colony Algorithm (ABC)

Definition1: Food resource is a solution of problem to be optimized. Profitability of this food resource is evaluated by fitness.

Definition2: Unemployed foragers include scouts and onlookers. Each unemployed bee has two choices: 1) seeking new food resource randomly; 2) deciding which food resource to go depending on the fitness provided by other bees.

Definition3: Employed bees share information of food to other bees at certain probability. An employed bee may turn into scouts if the food resource is empty.

There are five steps of standard ABC algorithm.

Step 1: Initialization.Number of colony is 2*N*, maximum iterationtimes isi_{max} and local searching times is *limit*. All bees are unemployed bees and each bee becomes a scout with probability of 0.5, and thus initial solution $X_1^1 \sim X_N^1$ can be got.

Step 2: Scouts become employed bees and calculate the quality of food resource based on Eqn.(15). If local searching times is less than*limit*, employed bees continue local searching, otherwise go to step 5. If the new food resource is better than the original, the original is replaced by the new.

$$fit_i = \begin{cases} 1 + abs(f_i), & f_i < 0 \\ \dfrac{1}{1+f_i}, & f_i \geq 0 \end{cases} \quad (15)$$

Step 3: Employed bees share information (*fitness*).

Step 4: Onlookers evaluate the food resource based on Eqn. (16) and chose food resource based on final fitness. And then search local food source. If the times of local searching is above limit, go to step 5.

$$\mathrm{P_i} = \frac{\mathrm{fit_i}}{\sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{fit_i}} \quad (16)$$

Step 5: If times of iteration is below$\mathrm{i_{max}}$, go to step 2, otherwise end iterations.

B. Discrete Artificial Bee Colony Algorithm (DABC)

SinceNoC mapping tries to search optimal in discrete solutions, discrete ABC algorithmis modified which executes searching and optimal processing in discrete space to ensure that one task is assigned to one resource tile.

To solve the problem above, task-resource list is introduced,see Fig.2. The digits in the first row represent tasks and digits in the second row represent resource tiles. For *i*-th task, if $e_i^j = \infty$and $t_i^j = \infty$, the *j*-th resource tile is ignored and *k*-th resource tile is selected randomly ($k \neq j$). Thus $p_i = [0_1, 0_2, \cdots 1_k, \cdots, 0_N]^T$ can be got and then *k*-th resource tile must be deleted from thesecond row in Fit.2. Keep this step until all tasks are mapped.

Local searchingis represented by Eqn. (17) and eqn. (18) (Step 2 in standard ABC):

$$j = i + \left(r \otimes (i \oplus k)\right) \quad (17)$$

$$r = rand(1, N) \quad (18)$$

where $\oplus$ represets*xor* operator and $\otimes$ represents *and* operator.

C. Enhanced Chaos Mechanism

Since above DABC algorithm is easy to fall into premature and thus cannot get a better result, chaos mechanism is introduced to avoid premature problem.

One-dimensional mechanism usually utilizes Logistic equation (19) to generate premature disturbance.

$$x^{n+1} = \mu \cdot x^n \cdot (1 - x^n), \mu \in [0,4] x \in [0,1] \quad (19)$$

whereμ is Logistic parameterwhose value is usuallybe set to 4.A good random sequence should be uniformly distributed so that it can help resolve the premature during the local searching. However, the random sequence generated by Logistic equation cannot satisfy the uniform distribution (will be detailed in section 4). So enhanced Logistic chaos mechanism based on division of interval is proposed in this paper (Eqn. (20~21)):

$$x_n^{'} = \begin{cases} x_n^p x_n & \in (0, r) \\ x_n x_n & \in [r, 1-r] \\ x_n^{1/p} x_n & \in (1-r, 1) \end{cases} \quad (20)$$

$$r \in rand(0,0.3) \quad p \in rand(0,1) \quad (21)$$

We use enhance-chaos-sequence listto explicit enhanced chaos mechanism, see Fig.3.

If no better solution can be achieved in consecutive times

ofoptimizing, *m* tasks are chosen randomly from first row in task-resource list, and each task is numbered by Eqn. (22) and then is put into first row of enhance-chaos-sequence list. Thus a newseries can be obtained by using enhanced chaos mechanism (Eqn. (19~21)). After calculated by Eqn. (23), the new series are put into the second row of enhance-chaos sequence list.

$$i' = \frac{\mathrm{i}-1}{N-1} \tag{22}$$

$$i' = [i' \cdot (N-1)+1] \tag{23}$$

A. The Proposed ECDABC algorithm

Base on above analysis, the pseudo code of proposed ECDABC algorithm is given below. The pseudo will not be further explained due to the length limitation.

1. Initialization:
number of colony = 2*N*, iteration times = *maxCycle*, local searching times = *limit*, initial solution = $X_1^1 \cdots X_N^1$ and fitness of solution = $\mathrm{fit}_i \cdots \mathrm{fit}_N$
2. Calculating solution:
while (iteration times <*maxcycle*)
{ fori in N
{ if (local searching times of X_i<*limit*)
calculate new neighboring solution by Eqn. (24~25)
else
using chaos mechanism (Eqn. (26~28)). If the solution is better than before, using new solution instead of old one, otherwise give up.
calculate probability of optimizing based on Eqn. (23)
}
if one solution is given up to be optimized, generate a random solution to replace the original one.
}
3. Finish:
obtain mapping result.

IV. Numerical Results

A. Simulation Scenario

In order to test and verify the performance of the proposed ECDABC algorithm in real application, a multi-media system (MMS) is chosen as a test case in the simulation. There are about 40 tasks of the system. A8 × 8 2D mesh NoC is utilized in this simulation. *N* is set to 20, *limit* is 40 and *maxCycle* is 500.α inEqn. (11) is set to 0.5 which means thatthe impact of power consumption is as important as delay.

While in GA optimization, *N* is set to 50, probability of crossover is 0.75 and probability of mutation is 0.1. And in PSO optimization, *N* is also set to 50.

B. Performance of Enhanced Chaos Mechanism

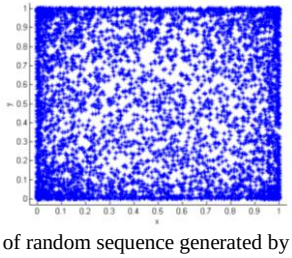

Fig. 5. Distribution of random sequence generated by Logistic mechanism

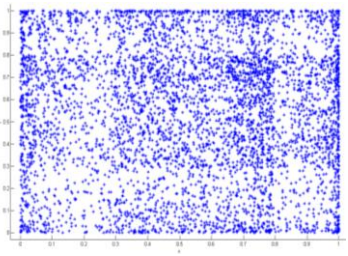

Fig. 6. Distribution of random sequence generated by enhanced Logistic mechanism

Notice that $y = \frac{1}{\sqrt{2}}(2x-1)$, which is get form Ulam-von Neumann mapping formula. From Fig. 5 we can clearly see that the distribution of random sequence generated by Logistic mechanism concentrates on the corners while that generated by our enhanced Logistic mechanism is uniformly distributed (Fig. 6), which further means that enhanced chaos mechanism has a better performance in avoiding the premature of local searching.

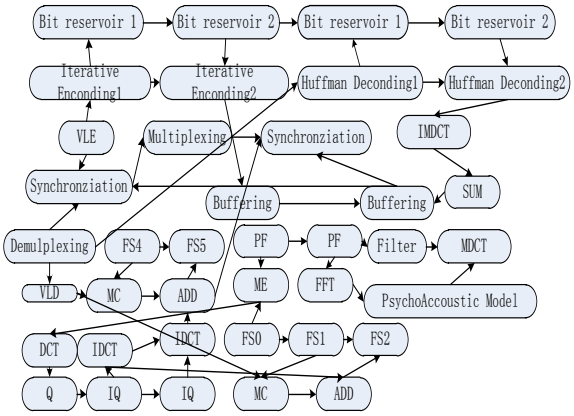

Fig. 7. Application characterization graph of MMS.

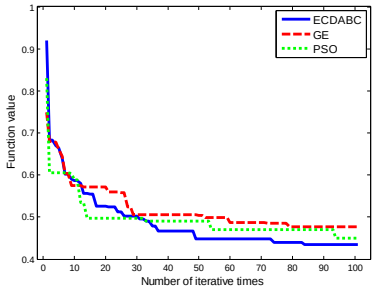

Fig. 8. Performance of ECDABC algorithm vs. PSO and GA

C. Performance of the Proposed ECDABC Algorithm

Fig. 7 shows the abstract graph of MMS, and Fig. 8 shows the performance of proposed ECDABC algorithm *vs.* PSO and GA. In this figure, we can see that proposed ECDABC algorithmcan converge to lower object value than GA and converge to lowest object value when iteration times beyond 30. Thus, our proposed algorithm has better performance in mapping and effectiveness than other two algorithms.

V. CONCLUSION

We propose a discrete artificial bee colony with enhanced chaos mechanism to solve mapping problems of 2D mesh NoC architecture.The proposed algorithm can avoid the local defection of prematurity of local searching by utilizing the randomnessand ergodicity of enhanced chaos mechanism.The experimental results show that our proposed ECDABC algorithm has better performance than PSO and GA, which means our proposed algorithm can balance the power consumption and transmission delay and thus get a feasible mapping solution.

ACKNOWLEDGEMENTS

This work is supported bythe Chinese National Major Project(No. 2011ZX03003-003-04).

REFERENCES

[1] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of Network-on-chip," ACM Comput. Surv38, 1, Article 1, Jun. 2006.

[2] L. Benini and G. De Micheli, "Network on Chips: A New SoC Paradigm," *IEEE Computer*, pp. 70-78, Jan. 2002.

[3] D. Wingard, "Micronetwork Based Integration for SoCs," in *Proc. of the 38th Design Automation Conf*, pp.673-677, Jun. 2001.

[4] J. Henkel et al, "On-chip Networks: a Scalable, Communication-centric Embedded System Design Paradigm," in *Proc. of VLSI Design*, pp. 845-851, 2004.

[5] Z. li and X. ling, "NoC Mapping Based on Choas Artificial Bee Colony Optimization," in *ICCP 2011 International Conf*, pp. 518-521, Oct. 2011.

[6] M. J. Sepulveda, M. Strum, and W. J. Chau, "A Multi-Objective Adaptive Immue Algorithm for NoC Mapping," *IEEE 17th IFIP International Conference*, pp.193-196, Oct. 2009.

[7] M. Johanna Sepulveda, Marius Strum, Wang J. Chau and Guy Gogniat, "A Multi-Objective Approach for Multi-ApplicationNoC Mapping," *IEEE LASCAS*, pp.1-4, Feb. 2011.

[8] S. Murali, M. Coenen, A. Radulescu, K. Goosens, and D. de Micheli, "Mapping and Configuration Methods for Multi-Use-Case NoCs," In *Proc. Asia and South Pacific DAC conference*, 2006.

[9] R. Marculescu, U. Y. Ogras, L. S. Peh, N. E. Jerger, and Y. Hoskote, "Outstanding Research Problems in No Design: System, Micro architecture, and Circuit Perspectives," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 28, no.1, pp.3-21, Jan. 2009.

[10] T. Lei and S. Kumar, "A Two-step Genetic Algorithm for Mapping Task Graphs to a Network on Chip Architecture," In *Proc. Digital System Design*, pp.180-187, Sept. 2003.

[11] L. Wang and X. Ling, "Energy and Latency Aware NoC Mapping Based on Chaos Discrete Particle Swarm Optimization," *IEEE CMC*, pp.263-268, Apr. 2010.

[12] P. K. Sahu, P. Venkatesh, S. Gollapalli, and S. Chattopadhyay, "Appication Mapping onto Mesh Structured Network-on-Chip Using Particle Swarm Optimization," in *VLSI 2011 IEEE Computer Society Annual Symposium*, pp.335-336, 2011.