Manuscript received Jul. 26, 2007; revised Oct. 27, 2007

Link-load Balance Aware Mapping and Routing for NoC

Zhou Wenbiao[1,2], Yan Zhang[1]
Harbin Institute of Technology Shenzhen Graduate School[1]
SoC Research Center
Shenzhen, Guangzhou
China
{zhouwenbiao,ianzh}@hit.edu.cn

Zhigang Mao[2]
Harbin Institute of Technology[2]
Microelectronics Center
Harbin, Heilongjiang
China
mao@hit.edu.cn

Abstract: The paper presents a novel mapping and routing technique for the mesh based NoC design problem with an objective of minimizing the energy consumption and normalized worst link-load. The proposed algorithm PLBMR is a particle swarm optimization (PSO) based two phases process, one is mapping core onto NoC to minimize the NoC communication energy consumption, and another is the allocation of routing path for keeping the link-load balance. and the paper presents the detail implementation of the PLBMR algorithm. Experimental results show that the proposed technique can reduce the normalized worst link-load by 20% on average while guarantee a low energy consumption.

Key–Words: NoC, Mapping, Low-power, Link-balance, Routing

1 Introduction

New technologies allow many millions transistors integrated onto a single chip and thus the implementation of complex SoC that need special communication resources to handle very tight design requirement. System architecture design is shifting towards a more communication-centric methodology [1]. Growing SoC complexity makes communication subsystem design as important as computation subsystem design. The communication infrastructure must efficiently accommodate the communication needs of the integrated computation and storage elements. In addition, some deep-submicron effects have generated: the increasing disparity between transistor and wire speeds, power delivery and dissipation, signal integrity , etc. Meanwhile, with the chip size's growing, the global clock closure is also a problem. The Network-on-Chip [2][3][4], which plays a pivotal role in future SoCs, is one of a solution for these challenges. It integrates some heterogeneous resource (for example, CPU, ASIC, DSP, etc) in a homogeneous on chip network environment.

The mapping and routing are two key steps for the NoC based system's design and implementation [6]. The low energy consumption and low latency are two main objects for NoC based system chip[5]. The whole design process of NoCs includes several processes, such as application decomposing, task assignment, topology selection, mapping IP cores onto title, and routing strategy. NoC design for a specific application offers an opportunity for optimizing the mapping of cores to different titles that act as the placeholder in the architecture, and the style of routing greatly affects the system performance and power consumption. For example, a specific application composed by a set of existing cores must be mapped onto the on-chip physical network structure, as shown in Fig.1, an application characteristic graph (APCG) that consists of 6 IP cores is mapped onto a 2x3 two-dimensional mesh NoC. Different mapping algorithms maybe obtain a different mapping position. Meanwhile, after a core has been mapped the coordinate title, the next step is to determine how to route packets. In Fig.1, the core (a) is mapped onto title-1 and core (e) is mapped onto title-6, then the shortest routing path between core (a) and core (e) has three choices:$S1 \rightarrow S2 \rightarrow S3 \rightarrow S6$,$S1 \rightarrow S2 \rightarrow S5 \rightarrow S6$, $S1 \rightarrow S4 \rightarrow S5 \rightarrow S6$.The selection of routing path will greatly affect the link-load balance of system, and other performance parameter, such as the bandwidth, delay, jitter, and resource utilization. This will even cause NoC into a congested mode, and the energy consumption of the NoC will not vary linearly with the traffic flowing through the network in a congested mode, the average latency of packet will not be a constant. All this will do harm to the system performance, and maintaining the link-load balance can avoid the congested mod to the best (not completely). These lead us to find an optimized mapping position and select a custom routing path to satisfy the link-load-balance requirement of system while minimizing the NoC communication power consumption.

The remainder of the paper is organized as fol-

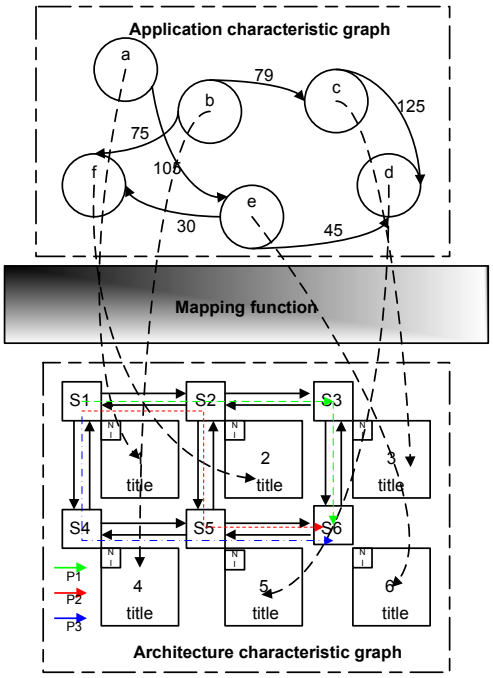


Fig. 1 A NOC Mapping and Routing Instance

lows. Section 2 discusses previous work. Section 3 presents the formation of mapping and routing problem. Section 4 proposed our new technique for the mapping and routing problem. Section 5 is the experimental results and finally section 6 concludes the paper.

2 Previous Work

The design of regular NoC architecture has been proposed in [7][8]. In [7], Dally et al use the on-chip network instead of ad hoc global wiring to interconnect the entire IP module on a chip. In [8], Kumar et al implement a 2D mesh NoC architecture that consists of a mesh of IP cores. There is an associated router for each core, and each router is thereby connected to four neighbor routers. Various approaches have been reported for solving the topology selection and NoC mapping problem. In [9], Lei et al. proposed a two-step genetic algorithm that finds a mapping of cores onto NoC architecture such that the overall execution time is minimized, the first step is assigning the task onto different IP cores, and the second is mapping the IP cores onto the title of NoC.[10] introduces a linear-programming based NoC synthesis method. Krishnan Srinivasa [11] utilized the GA for the custom NoC mapping process. And a BnB algorithm is used to map cores onto a mesh-based NoC architecture and find a flexible routing path with the objective of minimizing energy and satisfying the bandwidth constraints [12]. In [13], an algorithm NMAP under different routing functions and bandwidth constraints is presents. Chan-Eun Rhee et al.[14] proposed a MILP algorithm for the many-to-many core-switch mapping for the NoC architecture with optimizing the power consumption. In [15], Krishnan Srinivasa et al introduced a technique for low energy mapping and routing in NoC architectures under bandwidth constraints and delay constraints, it provide a full-custom routing. NetChip [16] is a complete synthesis flow for a customized NOC architecture; it partitions the whole design process into topology mapping, selection, and generation and provides proper tools for their automatic execution. Also, many researchers introduced mapping and routing methods for the customized NoC[17]. However, no methodologies can involve the design of link-load balance in the course of mapping and routing selection, and in this paper, we propose a PSO based design methodology to minimize the energy consumption of NoC while guaranteeing the balance of link-load.

3 Problem Formation

The NoC mapping problem is that of mapping different cores onto different titles of NoC. More precisely stated, given a periodic application described by a set of concurrent tasks, already bounded and scheduled onto a list of selected cores, the problem is to determine how to topologically map the selected cores onto on chip network and select a routing path for different communication flows, such that certain metrics of performance are optimized. In our formulation, a periodic real-time application that already bounded and scheduled on a list of selected cores is characterized by an application characterization graph (APCG)$G(\mathbf{V}, E_C)$, it is a directed graph, where each vertex $v_i \in \mathbf{V}$represents a selected IP core, and each directed edge $e_{i,j} \in E_C$ models a communication flow between core v_i and core v_j. The edge weight $e_{i,j}$ is a 2-turples $v(e_{i,j}), w(e_{i,j})$ where $v(e_{i,j})$ denotes the information exchange volume between cores v_i and core v_j, $w(e_{i,j})$ indicates the bandwidth requirement. The communication volume, bandwidth requirement can be obtained through the set of concurrent tasks' performance and the respect deadline. In our paper, we assume that the value of $v(e_{i,j}), w(e_{i,j})$ has been computed in advance.

The underling NoC is represented by an architecture characterization graph (ARCG)$G(T, L)$, it is also a directed graph, where each vertex $t_i \in \mathbf{T}$ represents one title in the architecture, and each directed edge $l_i \in \mathbf{L}$ represents a physical link, all links are assumed to have the same length and width B. A routing path $p(t_{v_i}, t_{v_j}) \in P$ indicates that we map core v_i onto title i and core v_j onto title j, it is a set of several links

$\{l_j, l_{j+1}, \dots\}$, where P is a set of all routing path. In addition, for the same time, the link bandwidth cannot be processed by two or more communication flows.

In our paper, we define the link-load function as follows:

$$U(l_j) = \sum_{\forall e_{i,j}} w(e_{i,j}) \times f(l_j, p(t_{v_i}, t_{v_j}) \quad (1)$$

where

$$f(l_j, p(t_{v_i}, t_{v_j})) = \begin{cases} 0 & l_j \in p(t_{v_i}, t_{v_j}) \\ 1 & l_j \notin p(t_{v_i}, t_{v_j}) \end{cases}$$

Moreover, we use the normalized worst link-load γ to evaluate the link-load-balance. For every core's mapping position MAP and its source and destination routing path P set, the normalized worst link-load γ:

$$\gamma(MAP, P) = \frac{\max\limits_{\forall i \in L} U(l_i)}{2WNK_m} \quad (2)$$

Where the denominator in the (2) is the bisection bandwidth of 2D mesh NoC, W is the link bandwidth,N is the node number of NoC, and the k_m is the max radix of 2D mesh.

And we defined the energy consumption as follows: The energy $E_{bit}^{p(t_{v_i}, t_{v_j})}$ that every bit routes through the path $p(t_{v_i}, t_{v_j})$:

$$E_{bit}^{p(t_{v_i}, t_{v_j})} = (|p(t_{v_i}, t_{v_j})| + 1) \times E_R + |p(t_{v_i}, t_{v_j})| \times E_L \quad (3)$$

Where $|p(t_{v_i}, t_{v_j})|$ is the hop of path $p(t_{v_i}, t_{v_j})$, and E_R is the energy consumption of single router,E_L is the energy consumption of one link.

Then the energy $E_{e_{i,j}}$ that a communication edge routes though the path $p(t_{v_i}, t_{v_j})$:

$$E_{e_{i,j}} = v(e_{i,j}) \times \{(|p(t_{v_i}, t_{v_j})| + 1) \times E_R + |p(t_{v_i}, t_{v_j})| \times E_L\} \quad (4)$$

For the whole application, the energy consumption E_{E_c}:

$$E_{E_C} = \sum_{e_{i,j} \in E_C} v(e_{i,j}) \times \{(|p(t_{v_i}, t_{v_j})| + 1) \times E_R + (|p(t_{v_i}, t_{v_j})|) \times E_L\} \quad (5)$$

After given above the definition, the objective of the NoC mapping and routing is as follows:

'Finding a mapping instance MAP and a set of routing path P, such that total energy consumption of the whole NoC (not include the IP core's energy consumption) is minimized and the value of $\gamma(MAP, P)$ is minimized'

That is:

$$\min\{E(MAP)\} \quad (6)$$

$$\min\{\gamma(MAP, P)\} \quad (7)$$

Such that:

$$\forall v_i \in \mathbf{V}, \quad MAP(v_i) \in T \quad (8)$$

$$\forall v_i \neq v_i \in \mathbf{V}, \quad MAP(v_i) \neq MAP(v_i) \quad (9)$$

$$\forall\ l_k, \quad W \geq \sum b(v_{i,j}) \times f(l_k, p(t_{v_i}, t_{v_j})) \quad (10)$$

4 Particle Swarm Optimization and NoC Mapping Problem

The mapping IP cores onto NoC architecture is a NP problem [15]. Our attempt is to try to develop an algorithm that can give near optimal results within the reasonable time, or, an algorithm with the best tradeoff between result quality and computation time. Particle Swarm Optimization (PSO) has shown the potential to achieve this dual goal quite well [18][19]. We have developed a two-phase PSO based algorithm for NoC mapping problem. This section presents the process for the PSO based load balance mapping and routing in NoC architecture, therefore called PLBMR. PLBMR is a two-phase particle optimization algorithm. In the first phase, we use the PSO to map IP core onto the title on the NoC architecture to minimize the energy consumption, and the second phase is to find all routing path for every mapping pair to satisfy the link-load balance.

4.1 PLBMR I -Mapping Core onto Title

4.1.1 Particle Swarm Optimization

PSO is an algorithm proposed by Kennedy and Eberhart in 1995 [18]]. In a PSO system, multiple candidate solutions coexist and collaborate simultaneously. Each solution candidate, called a 'particle', flies in the problem search space looking for the optimal position to land. A particle adjusts its position according to its own 'experience', as well as according to the experience of neighboring particles. Tracking and memorizing the best position encountered build particle's experience. PSO system combines local search methods and with global search methods, attempting to balance exploration and exploitation. Two factors characterize a particle status on the search space: its position and velocity. Kennedy and Eberhart explore several models to manipulate these factors to accurately resemble the dynamic of the social behavior of

birds, before reaching to the following equations that amazingly achieve good performance on optimization problems [18]:

$$V_{id} = v_{id} \times (p_{id} - x_{id}) + c_2 \times Rand() \times (p_{gd} - x_{id}) \quad (11)$$

$$X_{id} = X_{id} + V_{id} \quad (12)$$

Where V_{id} called the velocity for particle i, represents the distance to the traveled by this particle from its current position, X_{id} represents the particle position, p_{id} represents its best previous position, and p_{gd} represents the best position among all particles in the population. $Rand()$ and $rand()$ are two random functions with a range $[0, 1]$. $c1$ and $c2$ are positive constant parameters called acceleration coefficients (which control the maximum step size the particle can do). The inertia weight, W is a user-specified parameter that controls, with $c1$ and $c2$, the impact of previous historical values of particle velocities on its current one. Suitable selection of the inertia weight and acceleration coefficients can provide a balance between the global and the local search. (11) is used to calculate the particle's new velocity according to its previous velocity and to the distances of its current position from both its own best historical position and its neighbors' best position. Then the particle flies toward a new position according to (12). The performance of each particle is measured according to a predefined fitness function, which is usually proportional to the cost function associated with the problem. This process is repeated until user-defined stopping criteria are satisfied. Two versions of PSO exist, *gbest* and *lbest*. The difference is in the neighborhood topology used to exchange experience among particles. In the *gbest* model, the neighborhood of the particle is the whole swarm (i.e. the best particle is determined from the entire swarm). In the *lbest* model, a swarm is divided into overlapping neighborhoods of particles. This best particle is called the neighborhood best particle. Next, we present a gbest-model PSO algorithm for the problem at-hand: NoC mapping problem.

4.1.2 Core Mapping's Particle Encoding

Because NoC mapping has the constraint that different IP core cannot be mapped the same title, some randomly generated particle will violate the constraint. In order to represent a particle containing the mapping of N IP cores onto $2D$ mesh to satisfy the constraint, a convenient particle representation scheme is suggested here. Assuming the mapped $2D$ mesh NoC is represented by $1D$ array by using a left to right scan performing in a top-to-down fashion. This way an N-elements string of integers represents N different IP cores. In the proposed representation, at the i^{th} position from the left of the string, an integer between 1 and i is allowed. The string is decoded to obtain the mapping as follows. The i^{th} position denotes the mapping of IP cores i in the permutation. The decoding starts from the left-most position and proceeds serially toward the right. While decoding the i^{th} position, the first $i-1$ IP cores are already mapped, thereby providing with the i placeholders to position the i^{th} IP core. This decoding procedure is illustrated with an example having 6 IP cores $(a \sim f)$ to be mapped onto a 2×3 $2D$ mesh:

$\begin{pmatrix} 1 & 2 & 1 & 3 & 3 & 6 \end{pmatrix}$

In the above coding, the first integer is used to map a; the second integer is used to map b, and so on. All solutions will have a 1 in the first position. The coding actually starts from the second integer. The second IP cores b can be mapped onto two places:

1)Before the IP core a.

2)After the IP core a.

The first case is represented by a value '1' and the second case is represented by a value '2'. Since the second integer is '2' in the above string, the IP core 'b' appears after the IP core 'a'. Similarly, with the first two IP cores mapped as $(a \quad b)$, there are three placeholders for the IP core 'c':

1) Before a (with a value 1).

2) Between a and b (with a value 2).

3) After b (with a value 3).

Since the string has a value $'1'$ in the third place, the IP core is mapped before a, thereby constructing a partial mapping. Continuing in this fashion, the following permutation of the six alphabets is obtained:

$\begin{pmatrix} c & a & e & d & b & f \end{pmatrix}$

Next, this $1-D$ permutation is converted into 2×3 2D Mesh of alphabets representing components, as Utilizing this representing scheme for the particle, it can easily deal with the violated particle after update the particle velocity.

4.1.3 PSO for Core Mapping

In our mapping IP cores onto NoC phase, we setup a search space of M dimension for an M cores mapping problem. Each dimension has a discrete set of possible values limited to $s = \{T_i : 1 \leq i \leq N\}(M \leq N)$; such that N is the number of titles in the NoC system under consideration. For example, consider the problem to map 6 cores in Fig.1 onto a 2×3 $2D$ NoC architecture. Fig.2 shows an instance between possible mappings to a particle position coordinates in the PSO domain. Using such particle representation, the PSO population is represented as a $P \times M$ two-dimensional array consisting of N particles, each

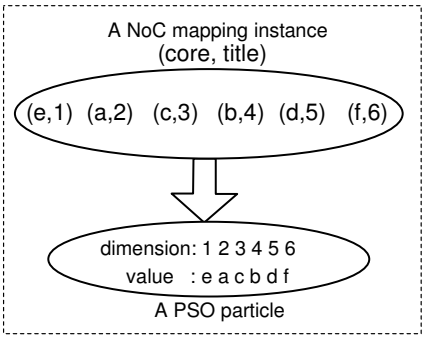


Fig. 2 Particle of NoC Mapping

represented as a vector of M cores. Thus, a particle flies in an M-dimensional search space. A core is internally represented as an integer value indicating the title number to which this core is mapped to during the course of PSO. Also, the minimum routing algorithm is used, but we still don't obtain the information which links the routing path contains, therefore, the bandwidth constraints is not considered in the PLBMR phase-1, and we only minimize the energy consumption. In the followed phase, we minimize the value of (5) while considering the link bandwidth constraints.

In our PSO algorithm instance, shown in Fig.1, we map a 6 cores APCG instance into corresponding 6-coordinate particle position. The algorithm starts by generating randomly as many potential mapping for the problem as the size of the initial population of t he PSO. It then measures particles' fitness. We use (7) as the fitness function. The algorithm keeps an updated version of two special variables through out the course of its execution:'global-best' position and 'local-best' position. It does that by conducting two ongoing comparisons: First, it compares the fitness of each particle being in its 'current' position with fitness of other particles in the population i n order to determine the global-best position each generation. Then, it compares different visited positions of a particle with its current position, in order to determine a local-best position for every particle. These two positions affect the new velocity of every particle in the population according to (11). As shown in this equation, two random parameters control amount of effect the two positions (i.e. global and local best positions) impose over the new particle velocity. The purpose of these parameters is to introduce a randomize and unbiased affect from either positions; hence, introduces a bit of exploration at some times and a bit of exploitation at another time randomly. The algorithm uses the new velocity to update the particle current position to a new position according to (12). Once all particles adjust their positions, they will constitute the new status of the PSO population. Then, the algorithm evaluates the fitness of these particles according to their new positions. Finally, the algorithm repeats this whole process of determining the global and the local best positions, updating particle position and evaluating new particle position until a user-determined criterion is satisfied. In our case, we mapped this criterion to a maximum number of generations. The proposed PSO algorithm for NoC mapping problem is summarized as follows:

Let N be the number of NoC title in the ARCG.
Let M be the number of IP cores in the APCG.
Let P be the size of the PSO population.
Let $PSO[i]$ be the position of the i^{th} particle of the PSO population represented an as an M- dimensional vector, whose entries' value belongs to the set $1, 2, \ldots, N$.
Then $PSO[i][j]$ is the processor number to which the j^{th} particle is assigned.
Let $fitness[i]$ be the cost function of the i^{th} particle according with (5).
Let $V[i]$ be the traveled distance (or velocity) of an i^{th} particle represented as an M-dimensional real-coded vector.
Let G_{best} be an index to global-best position.
Let $P_{best}[i]$ be the position of the local-best position of the i^{th} particle.
Let $P_{best\_fitness}[i]$ be the local-best fitness for the best position visited by the i^{th} particle.

STEP1*(Initialization): for each particle i in the population:*
For each core j:
Initialize $PSO[i][j]$ randomly from the set $1, \ldots, N$.
Initialize $V[i]$ randomly.
Evaluate fitness[i] according with (5).
Initialize G_{best} with the index of the particle with the best fitness (lowest cost) among the population.
Initialize $P_{best[i]}$ with a copy of $PSO[i]$.
STEP2*: Repeat until a number of generation, equal to twice the total number of IP cores, is passed:*
Find G_{best} such that $fitness[G_{best}] \geq fitness[i]$.
For each particle i: $P_{best}[i] = PSO[i]$, iff $fitness[i] > P_{best\_fitness}[i]$.
For each particle: update $V[i]$ and $PSO[i]$ according with (11) and (12).
Evaluate $fitness[i]$.

4.2 PLBMR II-Routing Path Selection

In the first phase of PLBMR, a core mapping position with minimizing the NoC power consumption is obtained. In the second phase of PLBMR, we aim to allocate a proper routing path for every communication flows, the purpose is to make a link-load balance of NoC with satisfying the link bandwidth constraints. Because the minimum-path routing algorithm is used, the IP core's mapping position determines the unique routing path length. As shown in Fig.1, IP core 'a' is

mapped onto title '1', and IP core 'e' is mapped onto title '6', then the hop number of communication flow between 'a' and 'e' is fixed. We randomly allocate any path among the three paths, and the number of hop is 3. In our NoC, the routing information is represented by a binary number. Each bit of the binary number represents routing direction, i.e. bits '1' and '0' correspond to move along the X-direction and Y-direction respectively. The hop number between source tile and destination title determines the bit number of routing information, i.e. the three routing path $P1$, $P2$, $P3$ between 'a' and 'e' is respectively $\{0\ 0\ 1\}$, $\{0\ 1\ 0\}$, $\{1\ 0\ 0\}$.

4.2.1 Routing Path's Particle Encoding

The particle of routing path in the PLBMR is represented by one-dimensional arrays of binary numbers. Each binary array represents one combination of all communication routing path set P in the APCG. For example, Fig.3 shows three particles of communication flow in the APCG which is shown in Fig.1. The particle's representing way is shown in Fig.3. For example, the communication flow $\{(a\rightarrow e),(b\rightarrow c),(b\rightarrow f),(c\rightarrow d),(e\rightarrow d),(e\rightarrow f)\}$ routing path in particle '1' is respectively $\{0\ 0\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 0\}$.

By representing the particle with this method, we can effectively generate the initial population size of PSO algorithm. However, the number of '1' and '0' is restricted for every communication flow. When the X-hop and the Y-hop of a communication flow are 'n' and 'm' respectively, the randomly generated binary numbers must assure 'n'-number of $1s$ and 'm'-number of $0s$. Thus, this method generates the combination of all communication flow's binary number, which forms a valid particle.

Data flow	a→e	b→c	b→f	c→d	e→d	e→f
Particle 1	0 0 1	1 0 0	0 1	0 1	0	1 0
Particle 2	0 1 0	1 0 0	0 1	0 1	0	1 0
Particle 3	0 0 1	0 0 1	1 0	0 1	0	1 0

Fig. 3 PSO Particle of Routing Path

4.2.2 PSO for Routing Path Selection

After having the particle, the PSO based routing path allocating process is almost the same as the phase '1'. Because of the space limitation, we don't list the algorithm code. In the course of the position updating, the new particle's values will not be a binary number, it violates the routing path representing principle. Therefore, in the algorithm, we must round these numbers to the '0' or '1'; the greater value is rounded to the binary value '1' according with a communication flow's Y-distance. The effects of converting from real number to binary number should be determined further. This adjustment is done for all particles, and the cycle of finding the Gbest,Lbest and updating the particle positions continues until satisfactory results or termination criteria are met such as population converge onto one proper routing-path allocating solution.

5 Experimental Results

We first compute the model parameters that used to evaluate the energy consumption. Assuming $0.18um$ technology and the link length between two switches is $2mm$, the capacitance of a wire is $0.56Ff/um$, the voltage swing is $3.3v$, then the energy consumption of the link is $6.115pJ$, and the bit energy value of 5×5 switches is $0.56pJ$. Given the bit-energy values and the mapping optimization results, the energy consumed in the NoC architecture can be obtained. At the same time, a minimum-path routing is used in the $2D$ mesh NoC and we assume the maximum link bandwidth of NoC is $1000Mb/s$.

In order to prove the efficiency of PLBMR, we compare the performance with that of other proposed algorithm, for example, GA based mapping algorithm and BaB algorithm. For each algorithm, the results are averaged over 10 trials. In addition, we used statistical significance measure to insure the reliability of our results compared to those obtained from GA and BnB.

We used in experiments the following values for parameters that control each algorithm:

(1) PLBMR

- (a) The size of the population equals to twice the number of IP cores.
- (b) The inertia weight W is set to 0.9.
- (c) $C1 = C2 = 2.0$.

(2) GA

- (a) Crossover probability $= 0.9$.
- (b) Mutation probability $= \frac{\text{IP NUMBER}}{2}$;
- (c) Roulette wheel selection.
- (d) The size of the population equals to twice the number of IP cores.

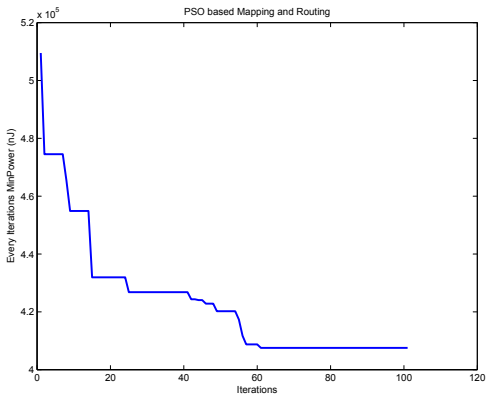
(3) BnB

- (a) compute the value of UB and LB according with [12].

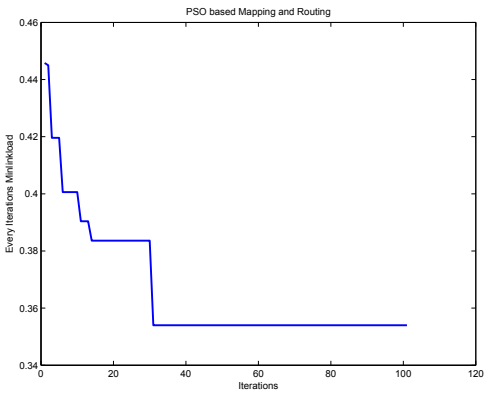
5.1 Experiments with Random APCG

We started by estimating the efficiency of the PLBMR optimization algorithm. For our experimental, we generated random application characteristic graph. Fig.4 and Fig.5 respectively shows the PLBMR algorithm optimization process for the 5×5 and 6×6 NoC.

Fig.4(a) and Fig.4(b) is a power and link-balance optimization process for 5×5 NoC. For the core-mapping phase, it gets the optimized value only after 50 iterations particle updating, and for the path-allocation phase, it gets the optimized value only after 13 iterations particle updating. For the energy consumption, the result of PSO optimization reduce 20% than the initial randomized mapping, and for the normalized worst link-load, the result reduce 13% than the initial randomized path allocation. Fig.5(a) and Fig.5(b) shows the energy consumption and link-balance optimization process for 6×6 NoC.



(a) Phase One

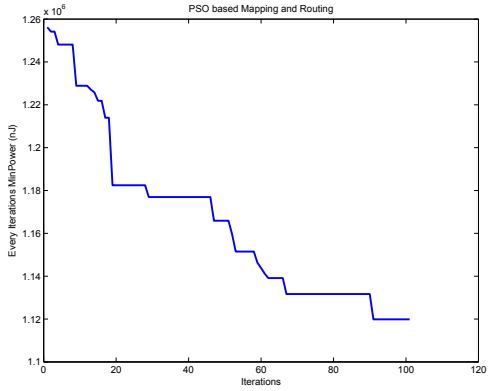


(b) Phase Two

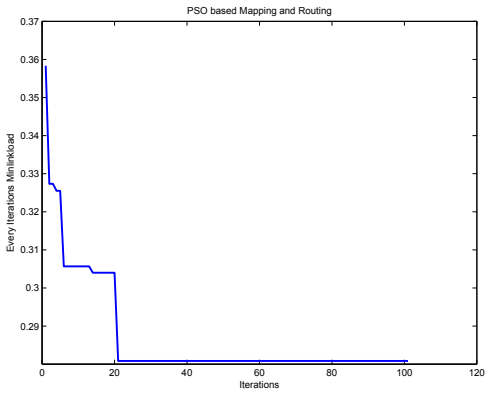
Fig. 4 PLBMR Optimization for 5×5 NoC

5.2 Comparative Result with GA and BnB

In this experiment, we firstly generated random APCG for different size (4, 9, 16, 25, 36) according with the TGFF [20],such that



(a) Phase One



(b) Phase Two

Fig. 5 PLBMR Optimization for 6×6 NoC

(1) A certain ratio $(CR = 0.5)$ of directed edges connect other different vertices among all possible edges;

(2) the required bandwidth of an edge is uniformly distributed over the range $[0, 500Mbytes/s]$;

(3) the traffic volume of an edge is uniformly distributed over the range $[0, 1Gbits]$.

(4) the NoC topology is a square 2D mesh.

TABLE. I Comparison for Different APCG

Num	PLBMR		GA		BnB	
	Time	Energy	Time	Energy	Time	Energy
4	0.3	7.6	0.9	7.6	0.234	7.6
9	1.5	79.4	6.6	83.3	1.345	78.2
16	5.5	381.9	65.6	410.9	6.768	386.8
25	16.4	1262.9	346.6	1271.6	18.564	1269.7
36	44.9	3159.3	1370.4	3158.7	55.678	3198.9

Time(second);Energy(mJ)

Table.2 shows the result for different sizes application graph. Each result is an average of 10 test cases. Algorithm performance varies along with the scale of APCG. For the small APCG, the performance

is almost as the same, but for the large scale APCG, the PLBMR has the advantage. For example, for the APCG with 25 IP core number, PSO solution quality is better 0.7% than that of GA. At the same time, the speed is 20 times faster than GA. And the PSO solution quality is better 0.6% than that of BnB and 1.2 times faster than BnB. As the APCG scale increases, the PSO algorithm has more strength to explore the good solutions for the NoC mapping problem.

To demonstrate the effectiveness of the PLBMR technique, we applied PLBMR to the following video processing applications: *Video Object Plane Decoder* (VOPD mapped onto 3x4 cores), *MPEG4 decoder* (mapped onto 3x4 cores), and *Picture-In-Picture application* (PIP-mapped onto 2x4 cores), and *Multi-window Display application* (MWD-mapped onto 3x4 cores). The APCG of these applications are presented in [16]. Two types of comparisons are carried out based on the results obtained by running each algorithm on the four video applications. The NoC communication energy consumption generated by PLBMR, GA and BB for 4 video applications is shown in Table.3. PLBMR algorithm is on the average 12.4% better than GA and 4.9% better than BnB. For the energy consumption, At the same time, the PLBMR is 4.27 times faster GA technique, and the BnB technique is 1.91 times faster GA technique. These results indicate that the proposed PSO algorithm is a viable alternative for solving the NoC mapping problem.

TABLE. II Comparison for Video Application

Application	PLBMR		GA		BnB	
	Time	Energy	Time	Energy	Time	Energy
MPEG	2.1	40.2	8.2	45.4	3.6	42.6
NWD	1.3	9.0	7.5	10.9	3.4	9.0
PIP	1.2	3.9	2.9	4.7	1.4	3.8
VOPD	1.7	29.1	7.6	31.4	4.2	30.8
Average	1.5	20.6	6.6	23.1	3.4	21.6

Time(second);Energy(mJ)

5.3 Comparative Result for Link Balance

In order to show the impact of PLBMR algorithm for the link-load balance, we compare the PLBMR with the PSO based mapping algorithm for XY dimensional routing (we denote it as the PBMXY). We generated random application characteristic graph $g1$ to $g10$ for different parameter. Fig.6 shows the comparison results. For each application characteristic graph, the first bar is the normalized worst load for PLBMR algorithm and the second bar is for the PBMXY algorithm. For most case, the PLBMR performance is superior to the PBMXY. The normalized worst link-load reduces 18% on average by the PLBMR algorithm. The reduction rations depend on the parameter of application characteristic graph and the core mapping result. From the results, it shows the routing allocation has large effects on link-balance, and therefore it will even affect the other performance of NoC, i.e. system delay, etc.

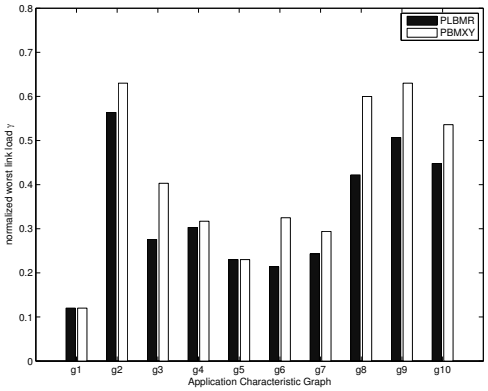


Fig. 6 Performance comparisons between PLBMR and PMXY

We also compared PLBMR algorithm with previous algorithm that related with path allocation: Hu's BnB [12] and Krishnan's MOCA [15]. For MOCA, we only consider no latency constraints. Table 3 presents the comparison of PLBMR with BnB and MOCA for g1 to g10 random APCG. As is shown in the table, the energy consumption has low discrepancies, but for the normalized worst link-load, PLBMR performed within 9% of MOCA, and 7% of BnB. Moreover, for a NoC with 36 node, our algorithm's running time is only 7 minutes with Pentium 4 $2.0G$ and $256M$ memory machine.

TABLE. III Comparsion of BnB, MOCA and PLBMR

	Energy(nJ)			Normalized Worst Linkload		
	PLBMR	BnB	MOCA	PLBMR	BnB	MOCA
G1	256.7	255.2	285.7	0.564	0.587	0.634
G2	156.7	156.5	176.8	0.276	0.395	0.403
G3	189.6	178.7	198.3	0.303	0.323	0.346
G4	168.2	156.1	180.9	0.230	0.296	0.305
G5	130.8	112.9	140.2	0.214	0.325	0.387
G6	178.5	160.4	189.6	0.243	0.356	0.402
G7	567.6	530.2	599.5	0.422	0.576	0.600
G8	356.7	349.2	390.6	0.507	0.613	0.634
G9	367.5	350.1	399.7	0.448	0.568	0.597
G10	298.9	278.2	321.0	0.345	0.456	0.499

6 Conclusion

A novel PSO based heuristic algorithm called PLBMR is proposed for the design of link-load balance and low energy mesh based NoC architecture. We use particle swarm optimization algorithm to explore the large search space of NoC design effectively. The proposed algorithm optimizes the NoC energy consumption in the IP mapping phase, and guarantee the link-load balance in the routing path-allocating phase. The performance of PLBMR is evaluated in comparison with other proposed mapping and routing algorithm BnB and MOCA for a number of randomly generated applications.The experimental results indicate that the proposed PLBMR is a viable alternative for solving the mapping and routing path's allocation for NoC.

References:

[1] A. Jantsch, H. Tenhunen, Network on Chip, *Kluwer*, 2003

[2] Luca. Benini, G.D.Micheli, Networks on Chips: A New SoC Paradigm, *IEEE Computer*, 1(4), 2002, pp. 70–78.

[3] Sriram Vangal, Jason Howard, Gregory Ruhl,An 80-Tile 1.28TFLOPS Network-on-chip In 65nm CMOS, *Proc of ISSCC*, San Francisco, USA, 2007, pp. 98–100.

[4] P. Wielage, K. Goossens, Networks on silicon: Blessing or Nightmare?, *Proc of DSD'02*, Dortmund, Germany, 2002, pp. 196.

[5] Kangmin Lee and Se-Joong Lee and Hoi-Jun Yoo, Low-Power Network-on-Chip for High-Performance SoC Design, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* , 14(2), 2006, pp. 148–160.

[6] U. Y. Ogras, J. Hu, and R. Marculescu, Key Research Problems in NoC Design: A Holistic Perspective , *Proc of CODES+ISSS'05*, Jersey City, NJ, 2005, pp. 69–74.

[7] W. J. Dally ,B. Towles, Route Packets, Not Wires: On-Chip Interconnection Networks, *Proc of DAC'01*, New York, USA, 2001, pp. 684–689.

[8] S. Kumar, A. Jantsch, A Network on Chip Architecture and Design Methodology, *Proc of VLSI'02*, Pittsburgh, Germany, 4, 2002, pp. 105–112.

[9] Tang Lei, Shashi Kumar, A Two Genetic Algorithm for Mapping Task Graphs to a Network on Chip Architecture, *Proc of DSD'03*, Antalya, Turkey, 2003, pp. 180–187

[10] K Srinivasan and K. S. Chatha, and G. Konjevod, Linear programming based techniques for synthesis of network-on-chip architectures, *IEEE Transactions on VLSI Systems*, 14(4), 2006, pp. 407–420

[11] Krishnan Srinivasa, Karam and S. Chatha, ISIS: A Genetic Algorithm. based Technique for Custom On-Chip Interconnection Network Synthesis", *Proc of VLSID'05*, Kolkata, India, 2005, pp. 623–628

[12] J.Hu, R. Marculescu, Exploiting the Routing Flexibility for Energy/Performance Aware Mapping of Regular NoC architecture, *Proc of DAT'03*, Munich, Germany, 2003, pp. 1068–1093

[13] S. Murali, G. De Micheli, Bandwidth-constrained mapping of cores onto NoC architectures,*Proc of DATE'04* , Paris, France, 2004, pp. 896–901.

[14] Chan-Eun Rhee, Han-You Jeong, Soonhoi Ha, Many-to-Many Core-Switch Mapping in 2-D Mesh NoC Architectures, *Proc of ICCD'04*, San Jose, CA, USA, 2004, pp. 438–443.

[15] Krishnan Srinivasan and Karam S. Chatha, A Technique for Low Energy Mapping and Routing in Network-on-Chip Architectures, *Proc of ISLPED'05*, San Diego, California, USA, 2005, pp. 387–392.

[16] Davide Bertozzi, Antoine Jalabbert and Srinivasan Murali, NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-chip,*IEEE Transactions on Parallel and Distributed Systems*, 16(2), 2005, pp. 113–129.

[17] Martin K.-F. and Thomas Hollstein and Heiko Zimmer and Manfred Glesner, Deadlock-Free Routing and Component Placement for Irregular Mesh-based Networks-on-Chip, *Proc of ICCAD'05* , San Jose, CA, USA, pp.238-245, 2005.

[18] J. Kennedy, R.C. Eberhart, Particle swarm optimization, *Proc of ICNN'95*, 12, 1995, pp. 1942–1948.

[19] Y. Shi, R. Eberhart, Empirical Study of Particle Swarm Optimization,*Proc of CEC'99,Washington*, DC, USA, 1999, pp.1945–1950.

[20] R. P. Dick, D. L. Rhodes, W. Wolf, TGFF: Task graphs for free, *Proc of Int. Workshop Hardware/Software Codesign* , Seattle, Washington, pp.97–101, 1998.