An Optimized Mapping Algorithm Based on Simulated Annealing for Regular NoC Architecture

Liulin Zhong, Jiayi Sheng, Ming'e Jing*, Zhiyi Yu, Xiaoyang Zeng, Dian Zhou

State Key Laboratory of ASIC & System, Fudan University, Shanghai, China
*mejing@fudan.edu.cn

Abstract: Network on chip (NoC) architecture is viewed as a potential solution for the interconnect demands of the emerging multi-core systems since it renders the system high performance, flexibility and low-cost. Mapping tasks onto different cores of the network is a critical phase in NoC design because it determines the energy consumption and packet latency. In order to reduce the energy consumption of applications running on multi-core architecture, we propose a new mapping strategy based on Simulated Annealing (SA). By allocating tasks that have big communication volume to adjacent places on the mesh, the proposed method overcomes the shortcoming of blind search in traditional SA. The experiment results reveal that the solutions generated by the proposed algorithm reduce average energy consumption by 56.56% in mapping 16 tasks and 66.32% in mapping 49 tasks compared with traditional Simulated Annealing (SA). [1]

Keywords: Simulated Annealing; mapping; NoC, multi-core system

1. Introduction

For NoC-based multi-core processors, the performance is largely determined by the method of task-mapping. Task-mapping intends to place different tasks onto the processing elements to improve the performance and/or energy efficiency of system. In recent years, various task mapping methods have emerged such as: Simulated Annealing [1], Genetic Algorithm [2], ILP [3], Branch and Bound [4], and BMAP [5]. Genetic algorithm demands a long time to converge while ILP takes much effort to do calculations. Besides, Branch and Bound algorithm requires intelligent pruning technique since it is a enumerate method. BMAP, which adopts binominal method, changes the hardware configuration which leads to high cost. Simulated Annealing (SA) is an iterative probabilistic algorithm which is used in a lot of fields. The feature of hill-climbing makes it especially suitable for the problems with a lot of local minima. However, the shortcomings of SA can not be ignored since its search is blind and much time is wasted on unpromising solution space. So our work focuses on generating a better algorithm based on SA which is more effective in searching a less energy-consuming placement.

[1] This work was supported in part by a grant from Science and Technology Commission of Shanghai Municipality 10706200300, NSFC 60773125, and State Key Laboratory of ASIC & System 09ZD002 and 10KF008. Shanghai Rising-Star Program 11QA1400500, National Significant Science and Technology Projects – 01 Special 2010ZX01030-001-001-03.

2. Problem formulation

The mapping problem in NoC design can be illustrated by an application with 9 tasks and a 3×3 mesh topology in Fig.1. The upper part of Fig.1 shows a communication task graph (CTG) $G_T < T, E >$, where each vertex $t_i \in T$ stands for a task and the directed edge $e_{ij} \in E$ stands for the interconnect dependence between t_i and t_j. Vol (e_{ij}) represents the communication volume from t_i to t_j. The lower part of Fig.1 shows a network topology graph (NTG) $G_C < C, R >$, where each vertex $c_i \in C$ stands for a core in the network and the directed edge $E_{bit}^{m(t_i),m(t_j)} \in R$ stands for the energy consumption when transporting one bit data from c_i to core c_j.

***Mapping Problem**:* Given a communication task graph CTG and a network graph NTG, find a one to one mapping function M: $T \rightarrow C$ from CTG to NTG with

$$\min : \{Energy = \sum_{e_{ij} \in E} Vol(e_{ij}) \times E_{bit}^{m(t_i),m(t_j)}\} \quad (1)$$

such that $\forall\ t_i \in T$, map(t_i)$\in C$, $\forall\ t_i \neq t_j$, map(t_i)$\neq$map(t_j).

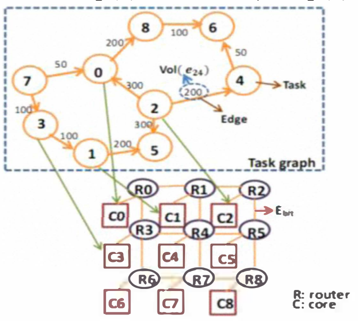


Figure 1. A mapping example

Although the proposed algorithm is concerned about application-mapping problems on homogenous mesh-based NoC architectures, it can also be adopted in other network topologies such as Fat-Tree [6], Torus [7] and Binary Tree [8]. It is suitable to the topologies whose energy consumption depends on the communication volume and the number of routers on the communication path.

3. Description of Simulated Annealing algorithm

The pseudo code of SA is shown in Fig.2, where initial temperature T0, final temperature Tstop and update factor k are three critical parameters. The annealing procedure runs from a high temperature T0 to a low temperature Tstop, during the process it gradually updates T by performing $T=T\times k$ (line 14). SA iteratively searches the solution space and evaluates these changes by the cost function as shown in formula (1). It can be seen that the algorithm not only accepts a move into a better state, but may also run into a worse state with a changing probability (line 10). As the simulation proceeds, the algorithm becomes greedier because the possibility of accepting a worse state gradually decreases (line 10). It terminates when the final temperature Tstop is reached (line 4).

Generally, as $T0$ and k increase, the results become better. This is because with more iteration times, the solution[2] space can be searched more thoroughly and a better result is likely to be found.

```
1: T=T0
2:generate a starting solution s, calculate total energy
    consumption E(s)
3: j=0   /*j stand for the number of accepted solutions*/
4: while T>Tstop do
5:   for i=1 to L do
     /*L stands for the number of perturbations under each T*/
6:     generate a new solution t in the neighborhood of s
7:       $\Delta E$ =E(t)-E(s)
8:     if $\Delta E$<0 then
9:       s=solution[j] =t;   j=j+1
10:    else if exp(-$\Delta E/T$ )>random[0,1] then
11:      s=solution[j]=t;   j=j+1
12:    end if
13:  end for
14:  $T=T\times$k
15: end while
16: find the best result in the array of solution[j].
```

Figure 2. Pseudo code of traditional simulated annealing

Although SA is a widely used algorithm in solving combinatorial optimization problems, it lacks an analysis on the features of practical problems and demands further improvement in efficiency.

4. Optimized SA with Task Combination Strategy

As can be seen from Section 3, SA is a relatively blind searching algorithm without any guidance. The initial placement and perturbations are randomly chosen. Such a disadvantage is vital, leading to much time wasted in unpromising solution space. As is known, the solution space of mapping is huge and there is no possibility to search it thoroughly in limited time. So our proposed algorithm aims at offering important information to the blind search to get better results.

In this paper, a solution of SA algorithm corresponds a mapping. We use the words placement, solution and layout interchangeably throughout the paper.

The traditional SA algorithm does not distinguish between different tasks or communication edges, so all of the nodes and edges in the task graph possess equal status in task allocation. However, in practical applications, the communication volume between different tasks is usually very divergent, especially in video applications such as VODP [9] and MPEG4 [9]. So it is not reasonable to view different nodes as equal. Thus our optimization aims at giving the node pairs with big communication volume the priority and combining them as a group during the layout phase.

Before elaborating the details of our algorithm, we first give the necessary definitions.

Definitions:

1) *Special node:* Task nodes that consume top communication volume are defined as special nodes. Two tasks which have large communication volume are viewed as relevant task to each other.

2) *Big node:* a node that regroups more than one task whose communication volume is large. A special node only belongs to one big node.

Take a communication task graph shown in Fig.3 (a) as an example, suppose t_1, t_2 and t_7 have much greater communication, then combine them as a big node and both of them are special nodes.

After big nodes are introduced, the space of solutions shrinks sharply. Take 4×4 2D mesh as an example, the original number of placements is 16!, whose order of magnitude is 10^{13}. Suppose there are 4 big nodes (each of them contains two tasks), the number of placements reduces to about 5.9×10^{10}. Thus, the shrink of space makes it easier to get a better result.

Since the traditional SA wastes a lot of time on the placements that are much worse than the optimum, an intelligent algorithm should put the special nodes as nearer as possible to reduce the entire energy consumption. The three main steps to implement the strategy is given below:

a) Find the tasks that have large communication volume.

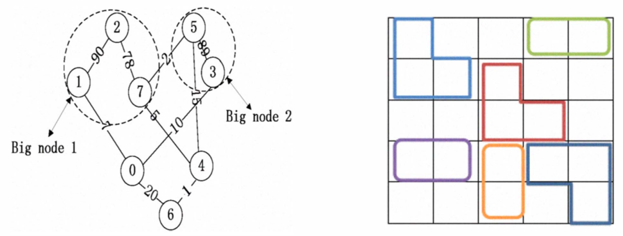
To realize a), a ranking algorithm is used to sort the communication list and pick out the edges that have heavy weights. For example, we can choose the top x% of the edges, where x can be adjusted according to the number of edges and the scale of applications.

b) Generate an initial placement. In this process, every big node is allocated to adjacent places on NoC first. While the other tasks are placed randomly like the traditional SA.

A few of possible placements of big nodes are shown in Fig.3 (b). The step is critical since it guarantees that the priority of big node has been realized in the initial layout and no time is wasted on unpromising placements. The pseudo code is shown in Fig.4.

c) Try to keep relevant nodes stay close to each other on the mesh after place exchanges.

To implement step c), during each swap after the initial placement, we will test if the chosen node is special. If not, we will exchange it with a random node next to it. If true, we will first exchange the place of the task with a random node next to it, and then allocate its relevant tasks to the places near it (see the procedure in Fig.5). The pseudo code is presented in Fig.6.

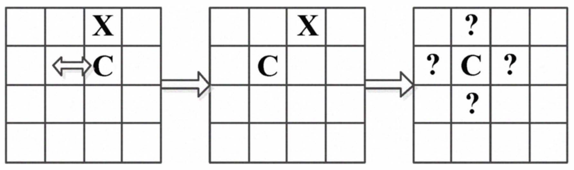


(a). big node (b). adjacent places

Figure 3. The mapping of big node

```
1: for (i=0 to $N_t$) do
   /*$N_t$ is the parameter that represents the number of tasks*/
2:   if task[i ] is not a special node   then
3:     generate a random place that has not been occupied and
       put task[i] to that place, set flag[i]=1
       /*flag[i]=1 indicates task[i] has been allocated*/
4:   else
5:     generate places to allocate task[i] and its relevant tasks.
       The number of places is equal to the size of the big node.
       Set the flags of task[i] and its relevant tasks to 1.
6:   end if
7: end for
```

Figure 4. Pseudo code for step b)



Task[c] will exchange places with a task on the left

Task[c] has been placed in a new grid and task[x] will be placed to a place near task[c]

Direction parameter d will determine which place to allocate task[x]

Figure 5. The steps of mapping a big node

```
1: randomly generate a task c and a random direction b.
   /*[b=0:east],[b=1:west],[b=2:north],[b=3:south]*/
2: if task[c] is not the special task then
3:   exchange task[c] according to b
4: else then
5:   exchange task[c] according to b
6:   generate a random direction d
6:   change the place of the relevant task[x] according to d
7:end if
```

Figure 6. Pseudo code for step c)

5. Experiment Results

Three experiments based on different benchmarks were performed. The first is to map 16 tasks onto a 4x4 mesh while the second one will map 49 tasks onto a 7x7 mesh. The last one aims at mapping the VODP task graph extracted from video application. In the first experiment, the communication list which is shown in Table 1 incorporates 16 tasks. Table 2 shows the energy consumption of traditional SA and optimized SA under different control parameters (T0 and k). The energy reduction under each pair of control parameters ranges from the minimum 40.82% to the maximum 63.49%. The average energy reduction is 56.56%. A clearer comparison using column chart is shown in Fig.7

As can be seen from Fig.7, the new algorithm can generate much better layouts with less iteration time under the condition: T0=100 and k=0.8, while the traditional SA did not find a satisfying placement with much longer iteration time under the condition: T0=10000 and k=0.87(see the first and last column in Fig.7). To get the similar results from the two algorithms, we performed another experiment, in which it takes 1.016s (T0=100000 and k=0.99) for the traditional SA to achieve the effect of the proposed algorithm. However, our algorithm only consumes 0.25s (T0=1000 and k=0.8).

We also performed the experiment of mapping 49 tasks onto a 7x7 2D-mesh. See the comparison of results in Fig.8. The energy reduction under each pair of control parameters ranges from the minimum 59.69% to the maximum 71.58%. The average energy reduction is 66.32%.

As can be seen from the above experiments, the algorithm is especially advantageous when a large number of nodes need to be allocated onto the NoC architecture, for in such cases, the solution space becomes much larger and it is increasingly impossible to find an ideal placement whose special nodes are located near each other.

Table 1. Communication list of 16 tasks

t_i	t_j	volume	t_i	t_j	volume	t_i	t_j	volume
2	0	97	10	7	205	13	2	35
2	1	1	11	1	71	13	5	9855
4	3	36	11	4	9660	14	9	450
6	0	9625	11	7	383	14	10	11
7	6	337	12	0	21	14	12	96
8	1	40	12	9	336	15	8	482
8	2	9771	12	11	9846	15	12	144
9	4	70	9	5	405			

Regarding the necessity to apply our algorithm to practical circumstances, we also performed experiment on mapping the task graph of VOPD [9] (see Fig.9), the results generated by optimized SA are better, as shown in Fig.10. The energy reduction under each pair of control parameters ranges from the minimum 33.65% to the maximum 33.70%. The average energy reduction is

33.68%.

The experiment results reveal that the optimized SA greatly outperforms the traditional SA by setting the special nodes near each other and reducing the total energy consumption of the applications running on NoC.

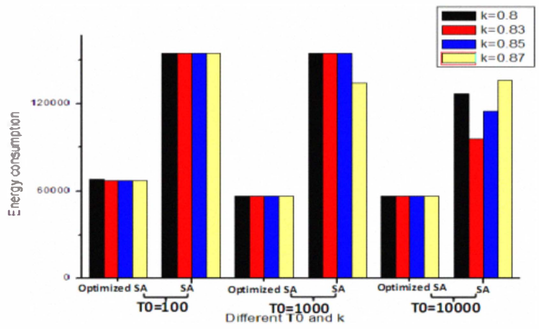


Figure 7. Comparison of the results for mapping 16 tasks

Table 2. Comparison of energy consumption for mapping 16 tasks

T0 \ k	algorithm	0.8	0.83	0.85	0.87
100	Optimized SA	68318	67304	67304	67304
	Traditional SA	154674	154674	154674	154674
	Improved	55.83%	56.49%	56.49%	56.49%
1000	Optimized SA	56466	56466	56466	56466
	Traditional SA	154674	154674	154674	134253
	Improved	63.49%	63.49%	63.49%	57.94%
10000	Optimized SA	56466	56466	56466	56466
	Traditional SA	126744	95432	114622	135931
	Improved	55.45%	40.82%	50.74%	58.46%

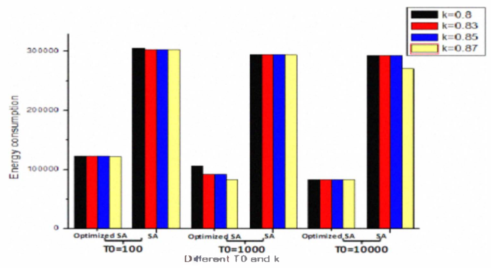


Figure 8. Comparison of the results for mapping 49 tasks

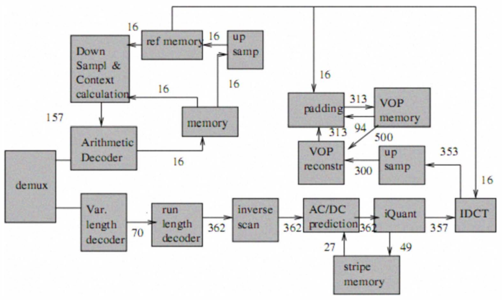


Figure 9. VODP task graph

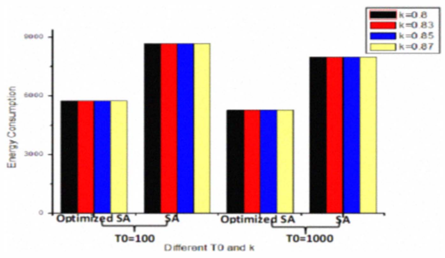


Figure 10. Comparison of results for mapping VODP task graph

6. Conclusions

This paper focused on a new mapping strategy based on SA in order to generate better layouts in task mapping problems. We analyzed the vital shortcomings of traditional SA which limit its efficiency and waste much time on the unpromising space. Inspired by the notion to propose an algorithm more efficient for mapping problems and generate more satisfying results, we come up with this new mapping strategy which overcomes the disadvantages of SA and shows outstanding efficiency in task mapping. Experiments were conducted to evaluate the performance and the results demonstrate that the proposed technique is able to generate high-quality mappings of tasks on multi-core systems.

References

[1]Wei Chen, “Task Partitioning and Mapping Algorithms for Multi-core Packet Processing Systems”, master thesis of UMASS, p.25- 26 (2009).
[2]G.Ascia, “Multi-objective mapping for mesh-based NoC architectures”, CODES-ISSS’04, (2004).
[3]Ying Yi, Wei Han and Xin Zhao, “An ILP Formulation for task Mapping and Scheduling on Multi-core Architecture”, DATE, p. 33 -38(2009).
[4]Jingcao Hu and Marculescu.R., “Energy and performance-aware mapping for regular NoC architectures”, CAD of Integrated Circuits and Systems, IEEE Transactions on 24(4), p.551 – 562(2005)
[5]Wein-Tsung Shen, “A New Binomial Mapping and Optimization Algorithm for Reduced-Complexity Mesh-Based On-Chip Network”, First International Symposium, Networks-on-Chip, p. 317 – 322 (2007)
[6]Ludovici.D, “Assessing fat-tree topologies for regular network on chip design under nanoscale technology constraints”, DATE, p.562-565(2009).
[7]Mirza-Aghatabar, “An Empirical Investigation of Mesh and Torus NoC Topologies Under Different Routing Algorithms and Traffic Models” , Digital System Design Architectures, Methods and Tools, 10th Euromicro Conference, p.19 – 26(2007).
[8]Yuan-Long Jeang, “A binary tree architecture for application specific network on chip (ASNOC) design”, Circuits and Systems, Vol.2, p. 877 – 880(2004).
[9]Suleyman Tosun, “New heuristic algorithms for energy aware application mapping and routing on mesh-based NoCs”, Journal of Systems Architecture, p.79-78 (2011).