IEICE Electronics Express, Vol.6, No.24, 1737–1744

Crinkle: A heuristic mapping algorithm for network on chip

Samira Saeidi[1a)], **Ahmad Khademzadeh**[2b)], **and Fatemeh Vardi**[1c)]

[1] *CE Department, Islamic Azad University, Science & Research Branch, Tehran, Iran*

[2] *Iran Telecom Research Center, Tehran, Iran*

a) *samirasa25@gmail.com*

b) *zadeh@itrc.ac.ir*

c) *f.vardi.a@gmail.com*

Abstract: In this paper, a heuristic mapping algorithm which maps tasks, using priority lists and the crinkle moving pattern is proposed. To evaluate this algorithm, a set of real (i.e. Video Object Plan Decoder) and random applications have been used and the results have been compared. By reducing the number of hops between IP cores, the energy consumption and the completion time of the application (time which all tasks in the task graph execute wholly) have been optimized. Compared to other mapping algorithms, the algorithm execution time (due to its low complexity) is considerably lower.

Keywords: network on chip, task graph, mapping algorithm, energy consumption, communication cost

Classification: Integrated circuits

References

[1] U. Y. Ogras and J. Hu, "Key Research Problems in NoC Design: A Holistic Perspective," *in Proc. CODES+ISSS'05*, New Jersey, USA, pp. 69–74, 19–21, Sept. 2005.

[2] J. Hu and R. Marculescu, "Energy-Aware Mapping for Tile-based NoC Architectures Under Performance Constraints," *in Proc. ASP-DAC'03*, pp. 233–239, Jan. 2003.

[3] S. Saeidi, A. Khademzadeh, and A. Mehran, "SMAP: An Intelligent Mapping Tool for Network on Chip," *ISSCS 2007*, PID360581, Iasi, Romania, July 2007.

[4] T. Lei and S. Kumar, "A Two-step Genetic Algorithm for Mapping Task Graphs to Network on Chip Architecture," *in Proc. DSD'03*, pp. 180–187, Sept. 2003.

[5] D. Shin and J. Kim, "Power Aware Communication Optimization for Network on Chips with Voltage Scalable Links," *in Proc. CODES + ISSS'04*, pp. 170–175, 8–10, Sept. 2004.

[6] A. Mehran, S. Saeidi, A. Khademzadeh, and K. Badie, "Spiral: Spiral: A Heuristic Mapping Algorithm for Network on Chip," *IEICE Electron. Express*, vol. 4, no. 15, pp. 478-484, 2007.

[7] S. Murali and G. Demicheli, "Bandwidth-constrained mapping of cores onto NOC architectures," *DATE*, pp. 896–901, 2004.

DOI: 10.1587/elex.6.1737
Received October 28, 2009
Accepted November 17, 2009
Published December 25, 2009

[8] T. Shen, C. H. Chao, Y. K. Lien, and A. Y. Wu, "A New Binomial Mapping and Optimization Algorithm for Reduced-Complexity Mesh-Based On-Chip Network," *Proc. First Int. Symp. Networks-on-Chip (NOCS'07)*, June 2007.

1 Introduction

Developments in CMOS technology have lead to integration of IP cores which are connected together in a single chip. The significant issue, in designing such chips, is how to put these IP cores together. Many challenges in this field led to the appearance of a paradigm called Network on Chip (NoC). There are several platforms with different network topologies for NoC, but the most simple and feasible example could be the 2-D mesh [2].

Communication infrastructure, communication pattern and application mapping are three important topics for NoC design optimization [1]. This paper has focused on application mapping and presented a heuristic algorithm which places IPs with a high connection degree close the center of the mesh whilst placing IPs with high communication data size adjacent to each other. The efficiency of the algorithm is directly related to these double objectives.

2 Related work

The mapping problem for tile-based architecture was addressed in [1]. In [2], a heuristic energy aware mapping and scheduling approach which tends to run much faster than the Genetic algorithm (GA) was addressed. Authors in [4, 5] proposed a two-step and three-step GA for mapping the application onto a mesh. In [7] NMAP, a fast heuristic algorithm under bandwidth constraints was presented. Authors in [8] presented an efficient binomial IP mapping algorithm, namely BMAP. The Spiral algorithm presented in [6] uses a task priority list for task graphs and a platform priority list in the spiral fashion.

3 Problem formulation

Application mapping determines how the application is mapped onto the NoC architecture. Basically, an Application Characterization Graph (also known as task graph (TG)) and an Architecture Characterization Graph can be formulated as follows:

Definition 1: An Application Characterization Graph=G (V, E) is a weighted directed graph, where each vertex v_i represents one selected IP and each directed arc $e_{i,j}$ characterizes the communication from v_i to v_j [2]. Each $e_{i,j}$ has the $b(e_{i,j})$ property that represents the communication data size. This weight of each arc corresponds to the number of bits transmitted between tasks. This type of TG has also been addressed in [2, 4, 5] and [6].

Definition 2: An Architecture Characterization Graph=G′(T, L) is a directed graph, where each vertex t_i represents a tile in the architecture, and

DOI: 10.1587/elex.6.1737
Received October 28, 2009
Accepted November 17, 2009
Published December 25, 2009

each directed arc $l_{i,j}$ represents the physical link from t_i to t_j. Each $l_{i,j}$ has the $E_{bit}^{ti,tj}$ property which represents the average energy consumption for sending one bit of data from t_i to t_j. A model for the energy consumption has been proposed in [1] and formulated by Eq. (1), where E_{Lbit} is the bit energy consumed on the links between nodes, E_{Sbit} is the bit energy consumed on the switches between the nodes, and n_{hops} is the number of routers the bit traverses from the source to the destination tile.

$$E_{bit}^{t_i,t_j} = n_{hops} \times E_{Sbit} + (n_{hops} - 1) \times E_{Lbit} \tag{1}$$

In Eq. (1) all communication links have the same characteristic but in this study each link and switch can have differentE_{Sbit}, E_{Lbit}. Based on these assumptions, the energy consumption is given by this expression:

$$E_{bit}^{t_i,t_j} = \sum_{k=1}^{n_{hops}} E_{Sbit_k} + \sum_{k=1}^{n_{hops-1}} E_{Lbit_k} \tag{2}$$

Hence, the total energy consumption of transferring data is:

$$E_{total}^{t_i,t_j} = Datasize_{t_i,t_j} \times E_{bit}^{t_i,t_j} \tag{3}$$

4 Crinkle mapping algorithm

This algorithm is based on the Priority Lists approach. The algorithm basis is to produce up to three Task Priority Lists (TPL) named respectively TPL1, TPL2 and RTPL, and one Platform Priority List (PPL) by following a crinkle motion pattern as shown in Fig. 1 (d). It starts from the corner of the 2-D mesh platform and ends on another corner in a zigzag manner. Fig. 2 (a) shows the general block diagram of this algorithm as a flowchart.

The TPL1 can have some or even all tasks of the TG whilst the TPL2 consists of some tasks of the TG that are not in the TPL1 and finally, the Remaining Task Priority List (RTPL) includes the tasks which are neither in the TPL1 nor in the TPL2.

At first, the algorithm determines the task which has the highest connection degree in the TG as the First Priority (FP). Assuming that v_i is the FP, then the communication data size between v_i and its neighbors is examined until a neighbor of v_i with the highest communication data size is discovered as the next priority namely v_j by Eq. (4). If several neighbors have the same maximum communication data size with v_i, the neighbor with the highest sum of its communication data size is chosen.

This process continues to check the neighbors of all tasks in the TG based on Eq. (4). In this process, if the algorithm reaches a task which has no neighbor with a high communication data size, the algorithm will be ended and the TPL1 is created.

$$NextPriority(v_i) = v_j \quad \forall v_j \in V \quad with \quad Max(b(e_{i,j})) \tag{4}$$

For some TGs, the aforementioned algorithm cannot examine all tasks in one step. When this happens, a second list, the TPL2, is built by applying

DOI: 10.1587/elex.6.1737
Received October 28, 2009
Accepted November 17, 2009
Published December 25, 2009

the same algorithm for remaining tasks starting from the same FP which has been used for building the TPL1.

If all the TG's tasks are in TPL1 and TPL2, the algorithm will be finished, otherwise the algorithm will create another list, namely the RTPL which first examines the neighboring tasks with the FP which are not in the TPL1 and the TPL2 based on the highest communication data size and then examines the remaining tasks.

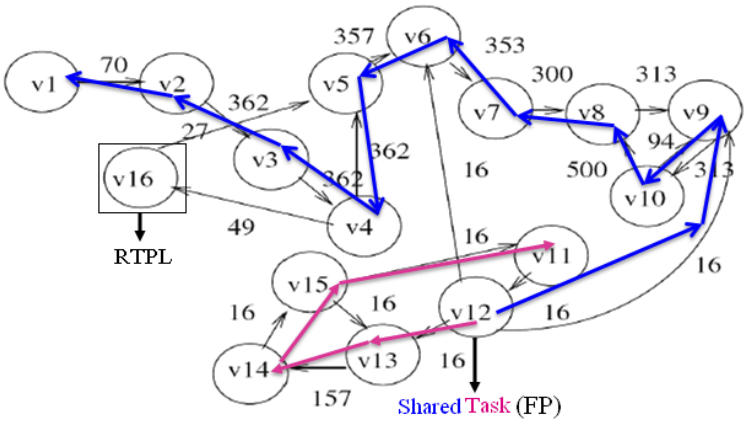
If all the TG's tasks are in the TPL1, the algorithm then reverses the TPL1 and the reversed TPL1 will be mapped onto the platform based on the crinkle pattern. The goal of reversing TPL1 is to put the FP close to the center in order to attain one of the heuristic's objectives. Otherwise the algorithm generates TPL2. After that, if all the TG's tasks are in the reversed TPL1 and TPL2, the algorithm merges the reversed TPL1 and TPL2. This merged list will be mapped onto the platform then based on the crinkle pattern. Otherwise, the algorithm generates the RTPL which the algorithm should determine its position.

There are three basic positions for the RTPL. The location of the RTPL can be at the start of the reversed TPL1, at the end of the TPL2 or even in the middle of them. Choosing the RTPL placement has a considerable impact on minimizing the Communication Cost (CommCost) after mapping.

Fig. 1 (a) shows this approach in the VOPD application clearly. The blue and pink path show TPL1 and TPL2 respectively, the RTPL has one element namely v_{16}.

Fig. 1 (b, c, d) show positions of the RTPL in the VOPD application. In this case, the RTPL has just one element but in other TGs, it can have more than one. In any case, the placement policy is the same. After the RTPL placement, the CommCost of all the different mapping which have been obtained by the Crinkle mapping algorithm are calculated based on Eq. (5) [7, 8]. The best choice is the mapping with the lowest CommCost. In this case, the algorithm selects Fig. 1 (c) for the VOPD mapping.

$$CommCost = \sum_{\forall e_{i,j}} b(e_{i,j}) \times dist(map(v_i), map(v_j)) | \forall e_{i,j} \in E, \exists b(e_{i,j}) \neq 0 \quad (5)$$



(a)

DOI: 10.1587/elex.6.1737
Received October 28, 2009
Accepted November 17, 2009
Published December 25, 2009

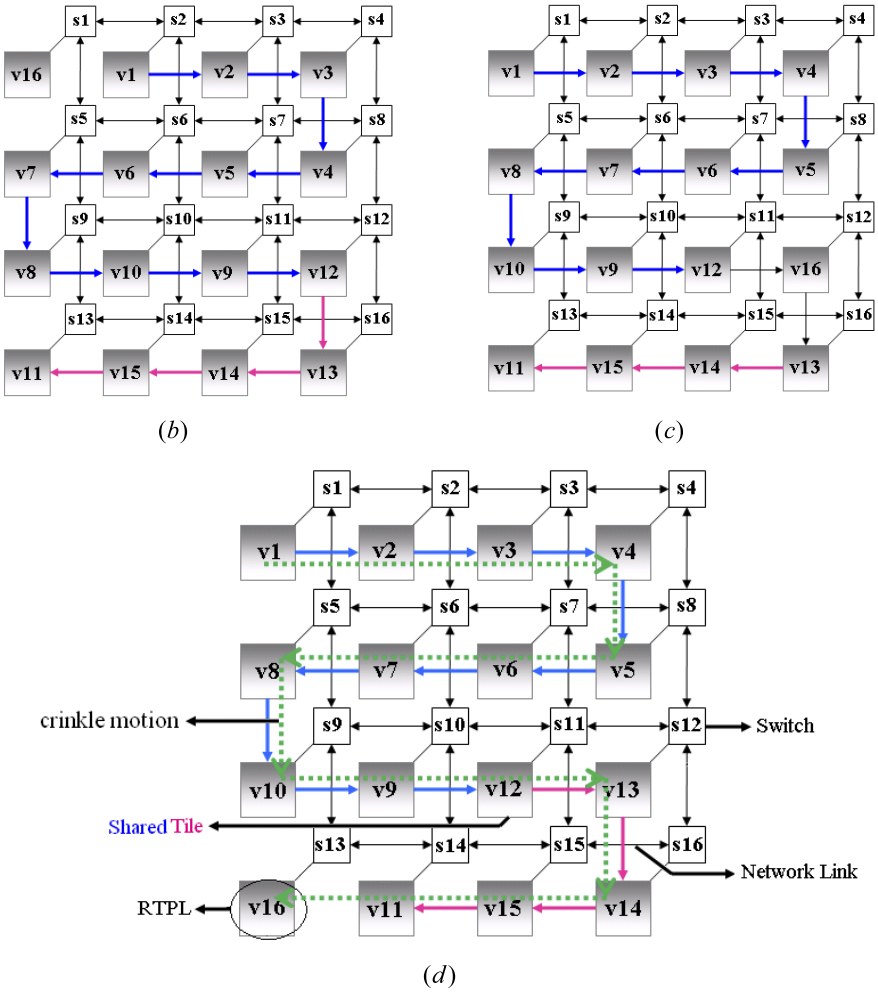


Fig. 1. (a) Special path in VOPD, (b,c,d) Three kinds of RTPL placement

Where dist (a, b) is the minimum number of hops between nodes a and b.

More details are presented as pseudo code in Fig. 2 (b).

5 Experimental results

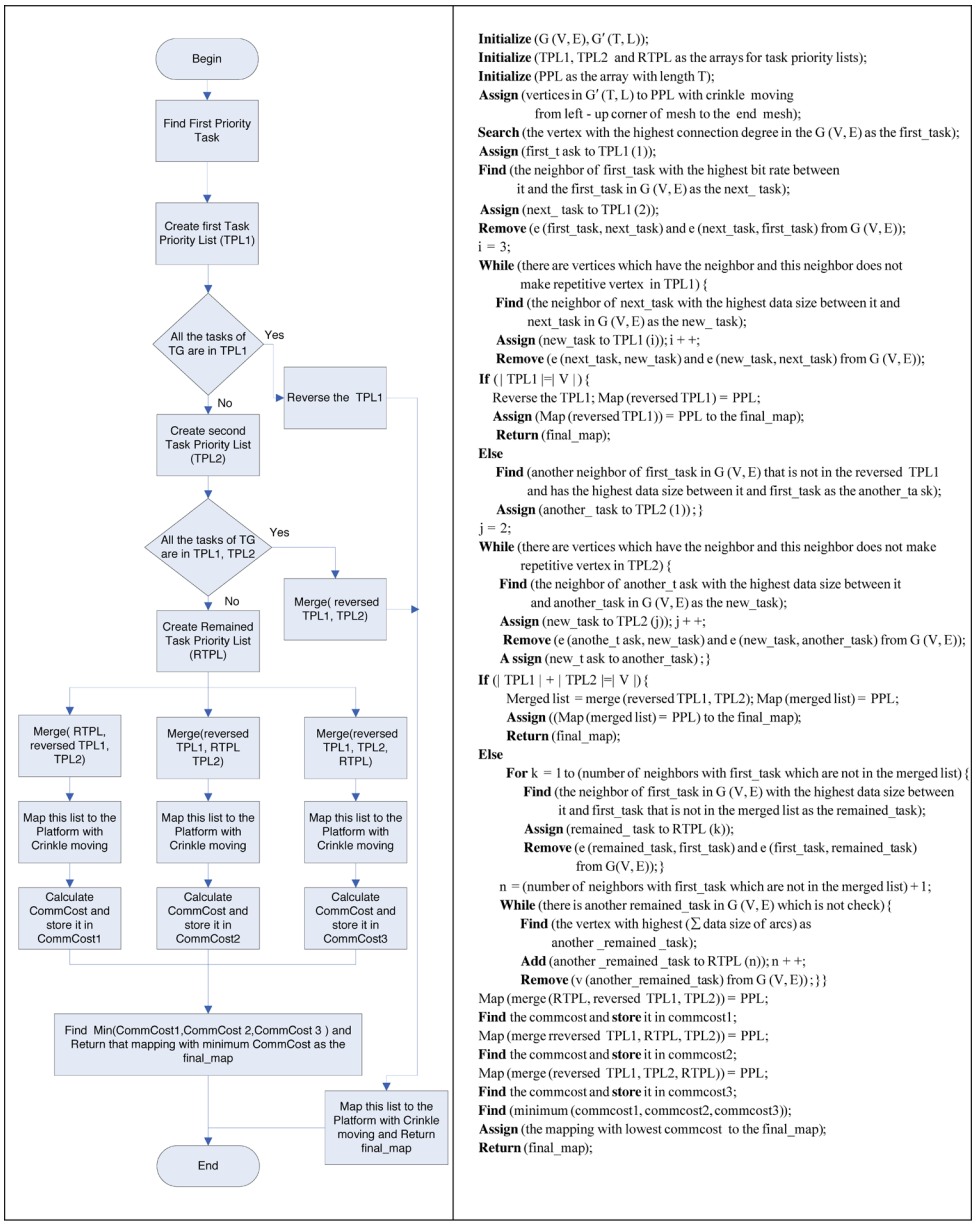
In this work we applied the X-Y routing algorithm and the list scheduling algorithm [2] to evaluate the performance of the Crinkle mapping algorithm.

A tool called SMAP for simulation has been used which measures the energy consumption, application completion time, and algorithm execution time [3]. It is also capable of creating random TGs onto the mesh platform with different sizes. To evaluate the efficiency of the proposed algorithm, its results have been compared with the GA, Random and Spiral mapping algorithms [3, 6] with the same routing and scheduling characteristics.

In this part instead of real applications, five random synthetic applications have been created for each mesh platform with the size of 3×3 to 6×6.

The results in Fig. 3 (a) and Fig. 3 (b) show that respectively the Crinkle mapping algorithm has less energy consumption and less completion time

DOI: 10.1587/elex.6.1737
Received October 28, 2009
Accepted November 17, 2009
Published December 25, 2009



(*a*) (*b*)

Fig. 2. (a) Algorithm flowchart, (b) Pseudo code of Crinkle mapping algorithm.

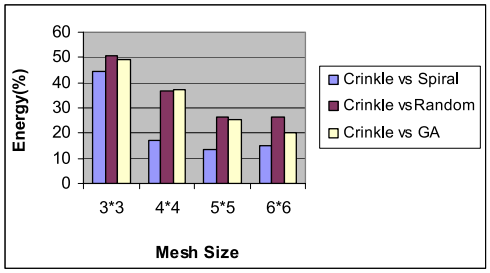
compared to the GA, Spiral and Random algorithms.

Fig. 3 (c) shows that when the mesh size increases, the Crinkle algorithm execution time does not increase rapidly, so this algorithm is more appropriate for the large mesh size and dynamic mapping.

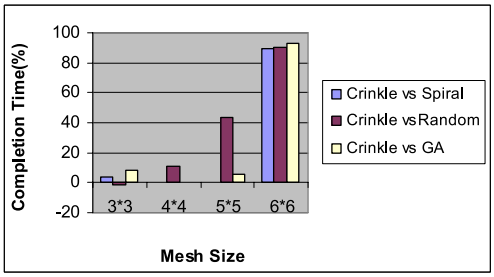
To evaluate the potential of applying the Crinkle mapping algorithm, it is compared with the NMAP [7], which has the lowest hop count among other mapping algorithms such as the PMAP, GMAP and PBB [8], with the same bandwidth constraints. Fig. 3 (d) shows that the Crinkle algorithm has less CommCost than the NMAP.

By reducing the hop count between related tasks the Crinkle algorithm can decrease the CommCost and improve performance parameters such as

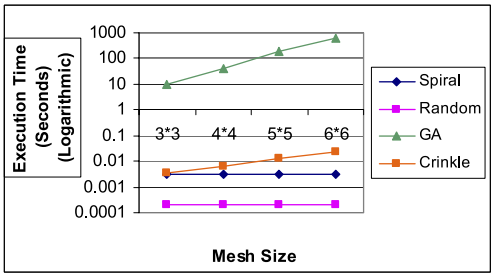
DOI: 10.1587/elex.6.1737
Received October 28, 2009
Accepted November 17, 2009
Published December 25, 2009



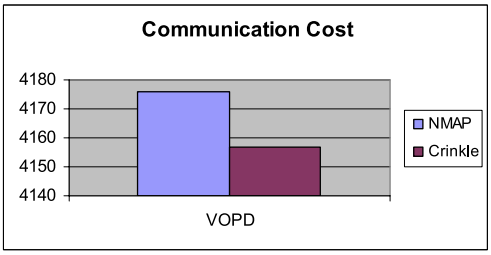
(*a*)



(*b*)



(*c*)



(*d*)

Fig. 3. (a) Percentage of reduction for the energy consumption of the Crinkle, (b) Percentage of reduction for the completion time of the Crinkle, (c) Comparison of execution times of the algorithms and (d) Reduction of communication cost.

the energy consumption and the completion time. Also, compared to the spiral algorithm, this algorithm gives more priority to communication data size rather than connection degree and this is another reason for the better performance of this algorithm.

6 Conclusion and future work

In this paper, a rule-based algorithm for the 2-D mesh called Crinkle was

DOI: 10.1587/elex.6.1737
Received October 28, 2009
Accepted November 17, 2009
Published December 25, 2009

proposed. In the Crinkle algorithm, several performance metrics such as the energy consumption, execution time and completion time have been optimized. The Crinkle algorithm has been compared to different mapping algorithms such as the Genetic, Random and Spiral. Simulation results show that by reducing the hop count between related tasks on the platform, the energy consumption, completion time and CommCost decreases and consequently the algorithm is more efficient than the other mentioned algorithms. Also the Crinkle algorithm has less CommCost than the NMAP.

The idea of implementing the Crinkle on other NoC architectures, and using different routing algorithms (not only x-y) are interesting subjects for the future work. Also, a low execution time means this algorithm can be used for dynamic mapping.

DOI: 10.1587/elex.6.1737
Received October 28, 2009
Accepted November 17, 2009
Published December 25, 2009