A3MAP: Architecture-Aware Analytic Mapping for Networks-on-Chip

Wooyoung Jang and David Z. Pan
Department of Electrical and Computer Engineering
University of Texas at Austin
wyjang@cerc.utexas.edu, dpan@ece.utexas.edu

Abstract - In this paper, we propose a novel and global A3MAP (Architecture-Aware Analytic Mapping) algorithm applied to NoC (Networks-on-Chip) based MPSoC (Multi-Processor System-on-Chip) not only with homogeneous cores on regular mesh architecture as done by most previous mapping algorithms but also with heterogeneous cores on irregular mesh or custom architecture. As a main contribution, we develop a simple yet efficient interconnection matrix that models any task graph and network. Then, task mapping problem is exactly formulated to an MIQP (Mixed Integer Quadratic Programming). Since MIQP is NP-hard [15], we propose two effective heuristics, a successive relaxation algorithm and a genetic algorithm. Experimental results show that A3MAP by the successive relaxation algorithm reduces an amount of traffic up to 5.7%, 16.1% and 7.3% on average in regular mesh, irregular mesh and custom network, respectively, compared to the previous state-of-the-art work [1]. A3MAP by the genetic algorithm reduces more traffic up to 8.8%, 29.4% and 16.1 % on average than [1] in regular mesh, irregular mesh and custom network, respectively even if its runtime is longer.

I. Introduction

So far, most of the NoC (Networks-on-Chip) based MPSoCs (Multi-Processor System-on-Chip) targeting general-purposed computing favor a regular mesh network [1-9]. The regular mesh network lets task-to-tile mapping easier, increases routing efficiency, provides desirable electrical and physical properties and reduces the complexity of resource management. However, real MPSoCs consist of various processors together with a DSP (Digital Signal Processor) and a memory as shown in Philips Nexperia platform [10], STMicroelectronics Nomadik [11] and Texas Instruments OMAP [12]. Since the physically different-sized processor, DSP and memory cannot be floorplanned to a regular mesh network, a resulting NoC-based MPSoC gets an irregular mesh network or even a custom network. The irregular mesh network can be also found in a regular mesh network when some links become faulty and degraded by process variation and temperature variation. After the fault and degradation of link are detected, task mapping and routing path allocation [13] should deal with the abnormal links and compensate for the loss of yield and performance. The previous task mapping algorithms make it inefficient to perform tasks in the irregular/custom network since they are not adaptive to the variation of network.

In this paper, we propose an A3MAP (Architecture-Aware Analytic Mapping) algorithm that is analogous to analytical traffic minimization in a given NoC-based MPSoC. We use a metric space that exactly captures the interconnection of network and that is simple yet efficient for a task mapping problem in various networks. In our task mapping formulation, we seek to embed a task graph into the metric space of network. Then, the quality of task mapping is measured by the total distortion of metric embedding. Through this formulation, our A3MAP can map a task adaptively to any different sized tile both on a regular/irregular mesh and on a custom network. Fig. 1 shows the methodology of our A3MAP. Given a task graph and a network as inputs, an interconnection matrix that can model any task graph and network along interconnection is generated. Then, task mapping problem is exactly formulated to an MIQP (Mixed Integer Quadratic Programming) and is solved by two effective heuristics since the MIQP is NP-hard [15]. One is successive relaxation of MIQP to a sequence of QP (Quadratic Programming) as a fast algorithm and the other is a genetic algorithm that is efficient random search algorithm to find better mapping solution. Importantly, our framework not only enables global, rather than local, optimization but also maps a task to both a regular mesh and an irregular mesh/custom network.

To the best of our knowledge, this is the first work that addresses task mapping in an irregular mesh network and a custom network. The rest of this paper is organized as follows: Section 2 reviews related works. Section 3 presents our novel A3MAP formulation. In Section 4, our successive relaxation algorithm and genetic algorithm are proposed to solve A3MAP formulated to a MIQP. Section 5 shows experiment results in comparison with the previous state-of-art work [1]. Section 6 is used for conclusion.

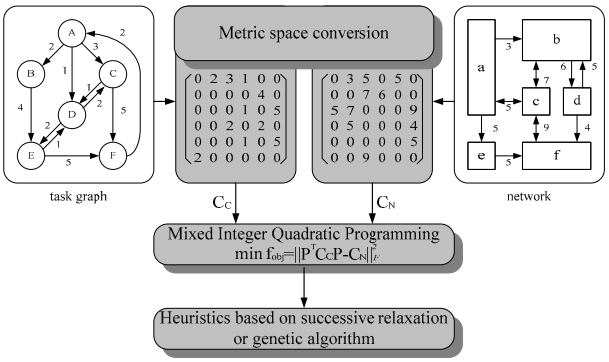


Fig. 1. Overview of A3MAP

II. Related Works and Our Contributions

In the last decade, a task mapping problem has been solved on a regular mesh network [1-5]. Murali et al. [1] present NMAP that is a fast algorithm, where the tasks are mapped onto a regular mesh network under bandwidth

978-1-4244-5766-3/10/$26.00 ©2010 IEEE

constrains, aiming at minimizing average communication delay. In [2], a branch and bound algorithm is adopted for task mapping in a regular mesh-based NoC architecture, which minimizes the total amount of power consumed in communications. Shin et al. [3] explores the design space of NoC based systems, including task assignment, tile mapping, routing path allocation, task scheduling and link speed assignment using three nested genetic algorithms. The work presented in [4] proposes an efficient technique for run-time application mapping onto a homogeneous NoC platform with multiple voltage levels. Chen et al. in [5] proposes a complier-based application mapping algorithm that consists of task scheduling, processor mapping, data mapping and packet routing to reduce energy consumption.

Recently, heterogeneous cores have been considered in an NoC-based MPSoC for low energy consumption [6-9]. Smit et al. in [6] solve the problem of run-time task assignment on heterogeneous processors. Carvalho et al. in [7] investigate the performance of several mapping heuristics promising for run-time use in NoC based MPSOCs with dynamic workloads, targeting NoC congestion minimization. Chang et al. in [8] propose ETAHM to allocate tasks on a target multiprocessor system. It mixes task scheduling, mapping and DVS (Dynamic Voltage Scaling) utilization and couples ant colony optimization algorithm. ADAM presented in [9] is run-time application mapping in a distributed manner using agents targeting for adaptive NoC based heterogeneous MPSoCs.

However, those task mapping solutions do not consider the irregularity of tiles and interconnections which generates more traffic on a network so that high energy may be consumed or QoS (Quality of Service) requirement may not be guaranteed. Our main novelty and contribution for solving these problems include followings:

- A simple yet efficient metric space is proposed to capture the interconnection of task graph or network. Then, a task mapping problem is exactly formulated to a MIQP based on a metric embedding technique.
- Successive relaxation approximation and genetic algorithm are employed to solve the MIQP. They fit well our formulation and provide a reasonable trade-off between performance and runtime.
- We show that the A3MAP can achieve good mapping performance not only in a regular network but also in an irregular network or custom network.

III. Problem Formulation

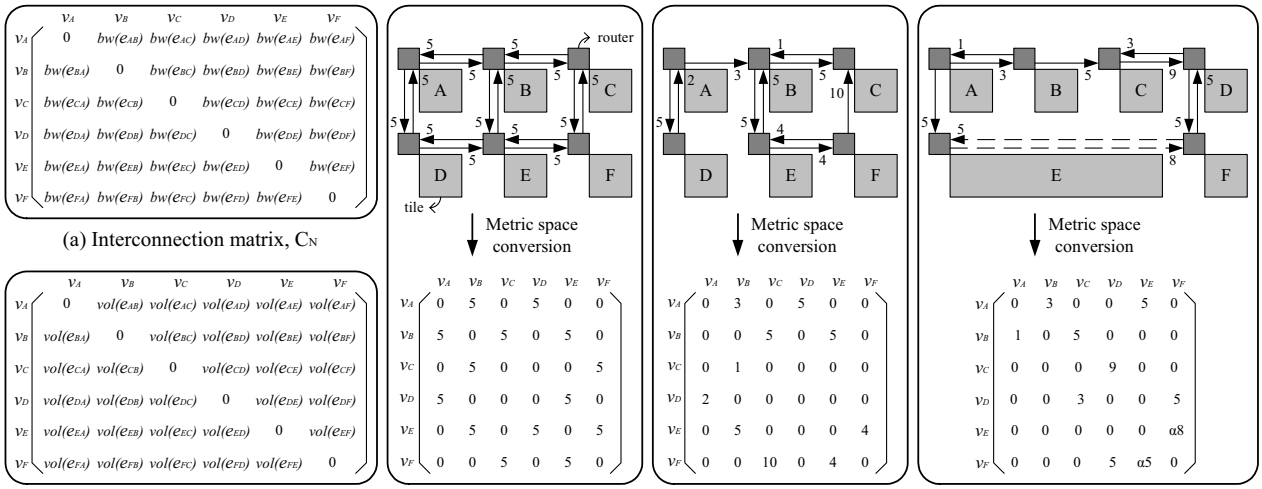
In this section, we formulate a task mapping problem to an MIQP using metric embedding. As inputs, we take a task graph and a network. We assume that each router is interconnected with one processor in the network. First, we present the metric space of graph using a proposed interconnection matrix to capture the dependencies of tasks in a task graph or tiles in a network. The graph $G(V,E)$ with n vertices is a directed graph, where each vertex $v_i \in V$ represents a task or a tile and where each directed edge $e_{i,j} \in E$ represents communication between v_i to v_j. $vol(e_{i,j})$ represents communication volume between v_i to v_j in a task graph and $bw(e_{i,j})$ represents bandwidth requirement between v_i to v_j in a network. We construct an $n \times n$ interconnection matrix, C_N corresponding to a network, where $c_{Ni,j} \in C_N$ is equal to $bw(e_{i,j})$ as shown in Fig. 2(a). Each row of C_N represents interconnection relation with respect to a single tile on an NoC. Thus, C_N contains interconnection relations for an entire network, representing the metric space of network. Similarly, we construct a $n \times n$ interconnection matrix C_C, corresponding to a task graph, where $c_{Ci,j} \in C_C$ is equal to $vol(e_{i,j})$ as shown in Fig. 2(b).

For example, Fig. 2(c), (d) and (e) show three network graphs and their metric spaces using the proposed interconnection matrix. In Fig. 2(c) that is a regular mesh, all routers are interconnected bidirectionally. Its interconnection matrix is composed symmetrically as shown in the below of Fig. 2(c). In case of an irregular mesh network in Fig. 2(d), interconnections between tile A and tile B or between tile C and tile F are unidirectional and tile D is not interconnected to tile E. Since the bandwidth of links is also different, its interconnection matrix is composed asymmetrically. This type of network is shown in VFI (Voltage Frequency Island) based NoC where each processor operates with own voltage and frequency and in NoC with faulty [14] and degraded links by process and temperature variation [13]. In case of a custom network, there is slightly difference in the composition of interconnection matrix. In Fig. 2(e), the wirelength between tile E and tile F is different from others due to different sized tile E. The composition of interconnection matrix for the custom network is similar with the previous two cases, yet weight α is added in the matrix in order to consider low energy consumption. Let the energy consumption of single link, E_{link}, computed as:

$$E_{link} = E_{drivers} + E_{repeaters} \quad (1)$$

where $E_{drivers}$ and $E_{repeaters}$ are the energy consumed by drivers and repeaters on a link respectively. If E_{link1} and E_{link2} are the energy consumption of dashed link and dotted link respectively, α is equal to the ratio of E_{link1} and E_{link2} ($=E_{link1}/E_{link2}$). The weigh α ($0<\alpha<1$) reduces the bandwidth of link with a long wire in the network. For example, we assume that a packet generated in tile A is transmitted to tile D, the dotted links are three times longer than the solid links and it takes the packet one cycle to pass each router. The packet can choose either A-B-C-D or A-E-F-D as a routing path. Since they pass the same number of router, all two paths take the same clock cycle to reach tile D while the total wirelengh of path A-E-F-D is longer than that of path A-B-C-D. Consequently, the path A-E-F-D consumes high energy since more repeaters on link EF are inserted or a bigger output driver is required for fast transmission time. Thus, it is good to assign a task with small traffic to a long link and a task with big traffic to a short link for low energy consumption. The weight α lets a task with heavier traffic mapped into a tile with short wires such that the energy consumption can be more minimized.

Graph embedding [16] maps the vertices of graph $G(V,E)$ into a chosen metric space by minimizing distortion. Thus,



(a) Interconnection matrix, C_N

(b) Interconnection matrix, C_C

(c) regular mesh network

(d) irregular mesh network

(e) custom network

Fig. 2. Various graphs and their interconnection matrices

task mapping has a natural correspondence with graph embedding into a 2D metric space representing the network. Our task mapping formulation is in the form of assignment problem. We seek to embed a task graph into the metric space of network. The goal is that a task is mapped to each tile, satisfying performance in a task-mapped network while the number of traffic generated between NoC tiles is minimal. If a network has the same architecture that a task graph has, such a task mapping does not have any distortion of edges in the task graph. Consequently, it produces the best possible mapping performance on the network. Generally, since most task graphs are different from a network, some distortion of task graph is not evitable in the network. Then, the quality of task mapping is measured by the total distortion of the embedding. By minimizing the extent by which edges in the task graph are stretched or distorted with intermediate tiles when embedded into the network, we seek to reduce the total amount of communications and to obtain a better global task mapping solution in terms of energy consumption and performance.

Our formulation is similar to an FPGA (Field Programmable Gate Array) [17]. However, the crucial difference in our work is the use of metric space that accurately captures the interconnections of network. A task mapping problem is equivalent to determining the assignment of task to each tile with various objectives such as low energy consumption and high performance. This task assignment action is mathematically presented by an $n \times n$ permutation matrix P. Column indices and row indices in P represent task identifiers and processor identifiers, respectively. For example, if $P(i,j)=1$, then task j is mapped to tile i. Thus only one element in each row and each column of P can be 1; all others must be 0. The action of P on the task graph is represented by $P^T C_C P$. P is found at once to minimize the difference between the permuted interconnection matrix of task graph $P^T C_C P$ and the interconnection matrix of network C_N for generating the less number of traffic and to minimize the distortion of C_C for a short routing path. For P that is orthogonal, we formulate this problem mathematically by our objective as:

$$\min f_{obj} = \left\| P^T C_C P - C_N \right\|_F^2 = \left\| C_C P - P C_N \right\|_F^2 \quad (2)$$

where $\|X\|_F = \sqrt{\sum_i \sum_j x_{i,j}^2}$, $x_{i,j} \in X$, i.e., the Frobenius norm of the matrix X, subject to integrity and linearity constrains as follows:

$$\sum_{i=1}^{n} P(i,j) = 1, \forall j = 1,2,...,n \quad (3)$$

$$\sum_{j=1}^{n} P(i,j) = 1, \forall i = 1,2,...,n \quad (4)$$

$$P(i,j) \in \{0,1\} \quad (5)$$

While our formulation has a convex quadratic object function, the binary constraints on the elements of P restrict the solution space to a non-convex set. Thus, convex optimization techniques like gradient descent cannot be directly applied to solve this problem. Actually, this type of formulation is well known as an MIQP (Mixed Integer Quadratic Programming) problem that is NP-hard [15]. To solve our formulation, we propose two effective heuristics which can tradeoff solution quality and runtime, which will be presented in the next section.

IV. A3MAP Algorithms

A. A3MAP-SR

In this section, we solve A3MAP formulated to an MIQP by using successive relaxation [18], called A3MAP-SR. The optimal MIQP formulation becomes a QP if we relax the discrete constraint Eq. (5) to a continuous constraint as:

$$0 \le P(i,j) \le 1 \quad (6)$$

Then, the key idea behind this algorithm is using this QP as a subroutine. The QP is faster to solve and scales much better. Then, continuous values obtained by a QP solver are

guided to 0 or 1 depending upon a predefined threshold, respectively. The concrete successive relaxation algorithm is shown in Algorithm 1.

After relaxing the constraint Eq. (5) to Eq. (6) (line 1), we set all *P(i,j)* to a *variable* since any *P(i,j)* is not guided to a permanent value, 0 or 1 (line 2). In line 3, initial *inverse_threshold* is set to the number of tile in a given network. As a threshold, we use *1/inverse_threshold* to guide continuous *P(i,j)* solved by a QP solver to 1, where the threshold means the expected average that a *variable P(i,j)* can get. On executing the successive relaxation, the *inverse_threshold* decreases by 1 (line 12) whenever one *P(i,j)* is set to 1, which means the threshold gets increased. The rest of algorithm 1 attempts to constraint the continuous values solved by the QP solver to binary values inversely. In line 7, we look for the maximum *P(i,j)* and compare it to the threshold in line 8. If it is above the threshold, it is set to 1 and a *non-variable* in line 9. In addition, all elements with the same row and column address as the maximum *P(i,j)* are also set to 0 and a *non-variable* (line 10 and 11) since the summation of elements of one row and one column in the permutation matrix *P* should be 1 by the constraints, Eq. (3) and (4). This procedure repeats if the next maximum *P(i,j)* is also above the threshold (line 14). Otherwise, we again solve QP for the rest of *variable P(i,j)* and continue to guide the continuous values to binary values. If all *P(i,j)* are guided to 0 or 1, we get the final permutation matrix *P*. Finally, a pair-wise swapping of task mapped by A3MAP-SR is executed until the amount of traffic is not decreased.

Algorithm 1 A3MAP-SR

Input: ***MIQP problem***

1: relax $P(i,j) \in \{0,1\}$ to $0 \le P(i,j) \le 1$;
2: set all *P(i,j)* to a *variable*;
3: *inverse_threshold* = $n(V_N)$;
4: **repeat**
5: solve relaxed MIQP by QP solver only for a *variable P(i,j)*;
6: **repeat**
7: find *max{P}* and store its location to (i_{max}, j_{max}) for $\forall$ *P(i,j)* that is a *variable*;
8: **if** *max{P}* $\ge$ *1/inverse_threshold* **do**
9: $P(i_{max}, j_{max}) = 1$ and set to a *non-variable*;
10: $P(i, j_{max}) = 0$ and set to a *non-variable*, $\forall\ i = 1,2,..,n$;
11: $P(i_{max}, j) = 0$ and set to a *non-variable*, $\forall\ j = 1,2,..,n$;
12: *inverse_threshold* decreases by *1;*
13: **end if**
14: **until (***max{P}* < *1/inverse_threshold*)
15: **until** (all *P(i,j)* are a *non-variable*)

Output: ***Permutation Matrix P***

B. A3MAP-GA

For a static task mapping where runtime is not more important than the reduction of traffic, we develop another heuristic using a genetic algorithm, called A3MAP-GA. Algorithm 2 is the pseudo-code of our genetic algorithm to solve the MIQP. First, we generate two arbitrary permutation matrices as parent individuals (line 1 and 3). A crossover scheme is widely acknowledged as critical to the success of the genetic algorithm. The crossover scheme should be capable of producing a new feasible solution (i.e., new child)

Algorithm 2 A3MAP-GA

Input: ***MIQP problem***

1: generate arbitrary *parent 1*;
2: **Repeat**
3: generate arbitrary *parent 2*;
4: (*child 1*, *child 2*) = cycle crossover (*parent 1*, *parent 2*);
5: mutation of *child 1and child 2* by pair-wise swapping;
6: *parent 1* = one of two children with minimum f_{obj} computed by Eq. (2) for the next evolution;
7: **until** (no improvement during *i*-iterations)

Output: ***Permutation Matrix P***

by combining the good characteristics of parent individuals while child individuals should be considerably different from their parent individuals. In A3MAP-GA algorithm, we use a cycle crossover as shown in Fig. 3. This crossover prevents more than two tasks being allocated into the same tile together. Fig. 3 shows how to generate two children using the cycle crossover. In the first step, Fig. 3(a), *child 1* inherits a column from *parent 1* and *child 2* inherits a column from *parent 2*. We start to choose the inherited column from one arbitrary column in *parent 1*. In Fig. 3(a), the first column is chosen in *parent 1* and then the same column as *parent 1* is also chosen in *parent 2*. Next, we look for the same content in *parent 1* as the first column of *parent 2* gets. In Fig. 3(a), the fifth column of *parent 1* contains the same content as the first column of *parent 2* contains. Then, the fifth column in *parent 2* is chosen together. Similarly, this procedure repeats until the column that is already chosen is again chosen. In the second step, Fig. 3(b), *child 1* inherits a column from *parent 2* and *child 2* inherits a column from *parent 1* inversely. The procedure is similar to the first step except the choice of column starts from any unselected column of *parent 2*. If all columns of children are not filled with the column of parent after the second step, we repeat the first and second step with the unselected columns of parents by turns. In Fig. 3, all columns of child are filled only after the second step. Then, a mutation operation is performed for each child (line 5 and 6). In this operation, two randomly selected columns are swapped to generate a new individual. Then, the swapping is kept if it reduces the number of traffic. The pair-wise swapping operation for each child continues until the pair of swapped columns cannot minimize our object function, Eq. (2). After the mutation operation, we choose the best child with the minimum cost as a parent for next evolution (line 7). If there is no improvement during *i*-iterations, we get the final permutation matrix with the optimal mapping solution.

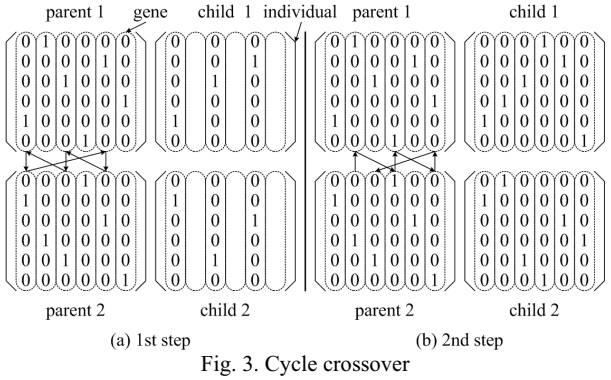


Fig. 3. Cycle crossover

Otherwise, the chosen child becomes *parent 1* and *parent 2* is also generated randomly. Then, those procedures repeat until there is no improvement for several iterations.

V. Experimental Results

We implement the A3MAP-SR algorithm by AIMMS/ CPLEX11.2 [19] and the A3MAP-GA algorithm by C++. All of the experiments were performed on a Linux machine with Intel 2.4GHz CoreDuo and 8GB RAM.

A. Regular Mesh Network

We carry out experiments by applying our task mapping algorithm on MPEG-4 VOPD (Video Object Plane Decoder) [20] and E3S benchmark suites [21]. The first application including 16 tasks is mapped onto a 4x4 regular mesh network. The second benchmark consists of consumer, AI (Auto-Industry) and telecomm application containing 12, 24 and 30 tasks respectively, which are mapped to a given 3x3, 4x4 and 5x5 regular mesh network, respectively. Since the number of task is different from the number of tile, the pre-processing is required. If the number of task is less than the number of tile, redundant vertices with no communication should be added in a task graph to make the same number of vertex as a network has. If the number of task is more than the number of tile, we perform $n(V_N)$ min-cut partitioning of task graph, where $n(V_N)$ is the number of NoC tile. Then, we perform A3MAP-SR and A3MAP-GA algorithms. Finally, we allocate the routing path of traffic by Dijkstra's shortest path algorithm to compute the total number of traffic between routers on the network. Table 1 shows that our A3MAP-SR and A3MAP-GA generates less traffic in a regular mesh network than NMAP [1] that is one of the famous NoC mapping algorithms. However, the runtime of our task mapping is a bit longer than NMAP as shown in Fig. 4.

TABLE 1. Traffic Comparison of A3MAP and NMAP [1] on Industrial Benchmarks in Regular Mesh Network

application	NMAP [1]	A3MAP -SR	Imp. (%)	A3MAP -GA	Imp. (%)
consumer	50	50	0	49	2
VOPD	4309	4265	1.0	4141	3.9
AI	187	151	19.3	147	21.4
telecomm	127	115	9.4	102	19.7

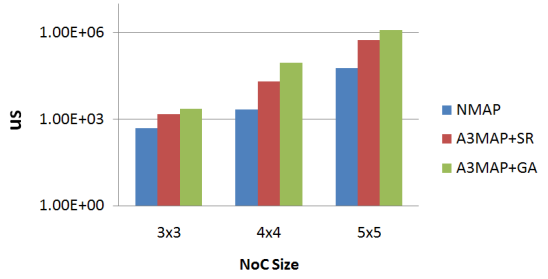


Fig. 4. Runtime comparison of A3MAP and NMAP[1]

B. Irregular Mesh Network

Our task mapping proves more merits on an irregular mesh or a custom network. We implement NMAP [1] for an irregular mesh network to compare our mapping algorithm even if it was proposed for a regular mesh network. Fig. 5 shows six irregular mesh networks on which we experiment A3MAP and NMAP. The results for MPEG-4 VOPD benchmark are present in Table 2. Our A3MAP shows better mapping improvement in an irregular mesh network than in a regular mesh. A3MAP-SR and A3MAP-GA reduce a significant amount of traffic between tiles up to 16.1% and 29.4% on average, respectively, when compared to NMAP. Especially, A3MAP shows better mapping improvement in more complex network as shown in Table 2. A3MAP avoids mapping a task containing heavy traffic to a tile containing little bandwidth and considers not only disconnection between tiles but also the direction of communication by architecture-aware manner.

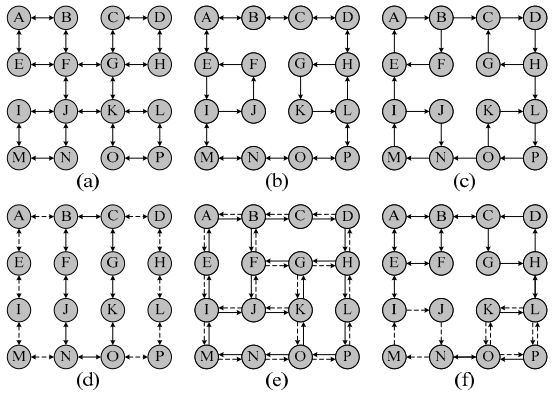


Fig. 5. Irregular mesh architectures

TABLE 2. Traffic Comparison of A3MAP and NMAP [1] in Irregular Mesh Network

Network	NMAP [1]	A3MAP -SR	Imp. (%)	A3MAP -GA	Imp. (%)
Fig. 5(a)	4869	4839	0.6	4237	13
Fig. 5(b)	5699	4619	19	4457	21.8
Fig. 5(c)	7810	7317	6.3	4619	40.9
Fig. 5(d)	4923	4301	12.6	4295	12.8
Fig. 5(e)	5706	4199	26.4	4183	26.7
Fig. 5(f)	8103	4844	40.2	4410	45.8
average	6185	5187	**16.1**	4367	**29.4**

C. Custom Network

In this section, we perform our A3MAP algorithm on custom networks with MPEG-4 VOPD benchmark. Fig. 6 is custom networks on which we experiment A3MAP and NMAP. Table 3 shows task mapping results performed on those custom networks. A3MAP-SR and A3MAP-GA reduce an amount of communication between NoC tiles up to 7.3% and 12.9% on average, respectively, when compared to NMAP. We also compute total wirelength passed by all traffics, which is more correlated to energy consumption than the number of traffic between NoC tiles. We count the

wirelength under the assumption that a short wire is 1 and a long wire is 2. In Table 3, total wirelength passed by communication traffic reduces up to 16.6% and 29.6% on average, respectively, when compared to NMAP. From these results, our weighted interconnection matrix is efficient to reduce energy consumption since the reduction of wirelength passed by communication traffic is better than the reduction of the amount of traffic. Thus, our weighted interconnection matrix is more desirable for a custom network with different-sized tiles. Similarly, our task mapping algorithm can be easily manageable for complex NoC by controlling the weighted interconnection matrix.

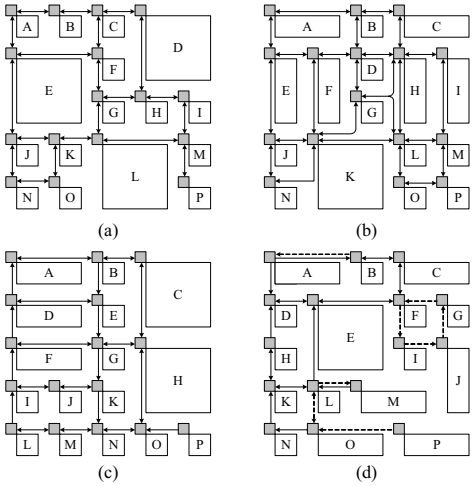


Fig. 6. Custom NoC topologies

VI. Conclusion

In this paper, we propose a novel and global A3MAP algorithm for an NoC-based MPSoC targeting general-purposing computing. Based on a metric embedding technique, we analytically formulate an NoC task mapping problem to an MIQP. Then, the MIQP problem is solved by two effective heuristics, successive relaxation as a fast algorithm and genetic algorithm to find better mapping solution. Experimental results show that A3MAP algorithm reduces a significant amount of traffic on all types of network when compared to the previous state-of-the-art work [1]. Especially, A3MAP algorithm shows more merits on an irregular mesh or a custom network. A3MAP has no limitation to map tasks to tiles on any arbitrary, faulty and degraded network. Furthermore, A3MAP is easily manageable for low energy consumption and high performance by an architecture-aware analytical manner.

References

[1] Srinivasan Murali and Giovanni De Micheli, “Bandwidth- constrained mapping of cores onto NoC architecture,” in Proc. DATE, 2004.

[2] J. Hu and R. Marculescu, “Energy-aware mapping for tile-based NoC architectures under performance constraints,” in Proc. Asian and South Pacific Design Automation Conference, 2003.

[3] D. Shin, J. Kim, “Power-aware communication optimization for networks-on-chip with voltage scalable links,” in Proc. International Conf. on Hardware/Software Codesign and System Synthesis, 2004.

[4] C.-L. Chou and R. Marculescu, “Incremental run-time application mapping for homogeneous NoCs with multiple voltage levels,” in Proc. International Conference on Hardware/software Codesign and system synthesis, 2007.

[5] G. Chen, F. Li, S. W. Son and M. Kandemir, “Application mapping for chip multiprocessor,” in Proc. DAC , 2008

[6] L. T. Smit, G. J.M. Smit, J. L. Hurink, H. Broersma, D. Paulusma and P. T. Wolkotte, “Run-time assignment of tasks to multiple heterogeneous processors,” in Proc. the 4rd PROGRESS workshop on embedded system, 2004.

[7] E. Carvalho, N. Calazans and F. Moraes, “Heuristics for dynamic task mapping in NoC-based heterogeneous MPSOCs,” in Proc. The 18th IEEE International workshop on Rapid System Prototyping, 2007.

[8] P.-C. Chang, I-W. Wu, J.-J. Shann and C.-P. Chung, “ETAHM: an energy-aware task allocation algorithm for heterogeneous multiprocessor,” in Proc. DAC, 2008.

[9] M.A.A. Faruque, R. Krist and J. Henkel, “ADAM: Run-time agent-based distributed application mapping for on-chip communication,” in Proc. DAC, 2008.

[10] S. Dutta, R. Jensen and A. Rieckkmann, “Viper: A Multiprocessor SOC for Advanced Set-Top Box and Digital TV Systems,” IEEE Design and Test of Computers, vol. 18, no. 5, pp. 21-31, Sep./Oct. 2001.

[11] STMicroelectronics, “Nomadik Multimedia Processors,” http://www.st.com.

[12] Texas Instruments, “Wireless Handset Solutions: OMAP Platform,” http://www.ti.com.

[13] Y. Markovsky, Y. Patel and J. Wawrzynek, “Using adaptive routing to compensate for performance heterogeneity,” in Proc. International Symposium on Networks-on-Chip, 2009.

[14] Wooyoung Jang, Duo Ding and David Z. Pan, “A voltage-frequency island aware energy optimization framework for networks-on-chip,” in Proc. International Conference on Computer-Aided Design, 2008.

[15] S. Sahni and T. Gonzalez, “P-complete approximation problems,” Journal of ACM, 23, 1976.

[16] J. Matousek, “Lectures in Discrete Geometry,” Springer, 2002.

[17] P. Gopalakrishnan, X. Li, L. Pileggi, “Architecture-aware FPGA placement using metric embedding,” in Proc. DAC, 2006.

[18] I.E. Grossmann and Z. Kravanja, “Mixed-integer nonlinear programming: A survey of algorithms and applications,” in Large-Scale Optimization with Applications, Part II: Optimal Design and Control, A.R. Conn, L.T. Biegler, T.F. Coleman, and F.N. Santosa (Eds.), Springer: New York, Berlin, 1997.

[19] AIMMS, “Optimization software for operations research applications,” http://www.aimms.com

[20] EB.Van Der Tol, EGT. Jaspers, “Mapping of MPEG-4 decoding on flexible architecture platform,” SPIE 2002, pp. 1-13, Jan. 2002.

[21] R. P. Dick, “Embedded system synthesis benchmarks suites (E3S),” http://www.ece.northwestern.edu/~dickrp/e3s/.

TABLE 3. Traffic and Wirelength Comparison of A3MAP and NMAP [1] in Custom Network

measure	total number of traffic between tiles					total wirelength passed by the traffics				
algorithm	NMAP[1]	A3MAP-SR	Imp. (%)	A3MAP-GA	Imp. (%)	NMAP[1]	A3MAP-SR	Imp. (%)	A3MAP-GA	Imp. (%)
Fig. 6(a)	4488	4531	-1.0	4087	8.9	5879	5332	9.3	4543	22.7
Fig. 6(b)	4264	4248	0.4	4199	1.5	5505	5049	9.4	4215	23.4
Fig. 6(c)	6296	5867	6.8	5150	18.2	7835	7434	5.1	5613	28.4
Fig. 6(d)	5524	4263	22.8	4263	22.8	9196	5170	43.8	5170	43.8
average	5143	4727	**7.3**	4425	**12.9**	7104	5746	**16.6**	4885	**29.6**