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

WOOYOUNG JANG, Samsung Electronics
DAVID Z. PAN, University of Texas at Austin

In this article, we propose novel and global Architecture-Aware Analytic MAPping (A3MAP) algorithms applied to Networks-on-Chip (NoCs) not only with homogeneous Processing Elements (PEs) on a regular mesh network as done by most previous application mapping algorithms but also with heterogeneous PEs on an irregular mesh or custom network. As the main contributions, we develop a simple yet efficient interconnection matrix that can easily model any core graph and network. Then, an application mapping problem is exactly formulated to Mixed Integer Quadratic Programming (MIQP). Since MIQP is NP-hard, we propose two effective heuristics, a successive relaxation algorithm achieving short runtime, called A3MAP-SR and a genetic algorithm achieving high mapping quality, called A3MAP-GA. We also propose a partition-based application mapping approach for large-scale NoCs, which provides better trade-off between performance and runtime. Experimental results show that A3MAP algorithms reduce total hop count, compared to the previous application mapping algorithms optimized for a regular mesh network, called NMAP [Murali and Micheli 2004] and for an irregular mesh and custom network, called CMAP [Tornero et al. 2008]. Furthermore, A3MAP algorithms make packets travel shorter distance than CMAP, which is related to energy consumption.

Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed Systems

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Network-on-chip, application mapping, homogeneous/heterogeneous processing element, mixed integer quadratic programming, genetic algorithm, successive relaxation algorithm

ACM Reference Format:
Jang, W. and Pan, D. Z. 2012. A3MAP: Architecture-aware analytic mapping for networks-on-chip. ACM Trans. Des. Autom. Electron. Syst. 17, 3, Article 26 (June 2012), 22 pages.
DOI = 10.1145/2209291.2209299 http://doi.acm.org/10.1145/2209291.2209299

1. INTRODUCTION

Continuing advancements in semiconductor technology enable to integrate many cores on a single die, called System-on-Chip (SoC). As thousands of cores will be integrated to a single chip for enhanced performance and functionality by 2015 [Borkar 2007], on-chip communication techniques and application mapping algorithms become key factors in the success of the multi- or many-core chips. Recently, Networks-on-Chip (NoCs) replacing point-to-point and shared bus interconnections have been employed to solve complex on-chip communication issues. NoCs also provide great scalability and

A preliminary version of this article was presented at the *Asian and South Pacific Design Automation Conference (ASP-DAC)* 2010 [Jang and Pan 2010a]. This work is supported in part by Samsung Electronics.
Authors' addresses: W. Jang (corresponding author), System LSI Division, Samsung Electronics, South Korea; email: wooyoung.jjang@gmail.com; D. Z. Pan, Electrical and Computer Engineering Department, University of Texas at Austin, Austin, Texas.

DOI 10.1145/2209291.2209299 http://doi.acm.org/10.1145/2209291.2209299

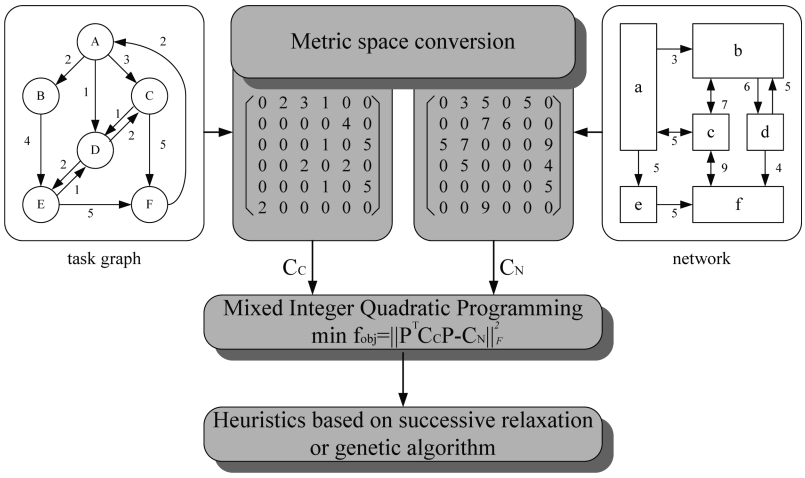


Fig. 1. Overview of A3MAP.

flexibility for the modern and future SoCs [Dally and Towles 2001; Benini and Micheli 2002; Jang 2011].

So far, most of the NoCs favor a regular mesh architecture consisting of regular rectangle tiles on which homogeneous processors are placed. The regular mesh network makes application mapping easy, increases routing efficiency, provides desirable electrical and physical properties, and reduces the complexity of resource management. Hence, most previous works have optimized application mapping on the regular mesh architecture. However, industrial SoC platforms, such as Nexperia [Dutta et al. 2001], Nomadik [STMicroelectronics], and OMAP [Texas Instruments], consist of various Processing Elements (PEs) such as a general processor, a Digital Signal Processor (DSP), a specific memory, and a peripheral. Since such physically different sized processing elements cannot be floorplanned into a regular mesh network, the resulting NoCs get an irregular mesh network or even a custom network [Chatha et al. 2008]. The irregular mesh networks are also found in a regular mesh network when any links become faulty or degraded by process and temperature variation. Therefore, application mapping and routing path allocation should deal with the abnormal links and compensate for the loss of yield and performance [Markovsky et al. 2009]. In addition, since Voltage-Frequency Island (VFI)-based NoCs have links with different bandwidth [Jang and Pan 2011a] and 3D NoCs have irregular interconnections due to manufacturing constraints [Jang et al. 2011], they are no longer a regular mesh network. On the contrary, the previous application mapping algorithms are not adaptive to various network architectures. As a result, specific mapping algorithms may be required for different network architectures. Therefore, an application mapping algorithm that can be applied to various networks is required.

In this article, we propose Architecture-Aware Analytic MAPping (A3MAP) algorithms that are analogous to analytical communication minimization in a given hard NoC. We use a metric space that exactly captures network architectures and that is simple yet efficient for an application mapping problem in various networks. In the proposed application mapping formulation, we seek to embed a core graph into the metric space of a network. Then, the quality of the application mapping is measured by the total distortion of metric embedding. Through this formulation, cores can be mapped adaptively to any different sized tiles on regular/irregular mesh and custom networks. Figure 1 shows the overview of A3MAP. Given a core graph and a network

as inputs, interconnection matrices that can model any directed/undirected and weighted/unweighted graphs are generated. Next, an application mapping problem is exactly formulated to Mixed Integer Quadratic Programming (MIQP) and then is solved by two efficient heuristics since MIQP is NP-hard [Sahni and Gonzalez 1976]. One is successive relaxation of MIQP to a sequence of Quadratic Programming (QP) providing short runtime and the other is a genetic algorithm that is an efficient random search algorithm providing high mapping quality. Importantly, our framework not only enables global, rather than local, optimization but also maps cores to various network architectures, in particular, irregular mesh and custom networks.

The rest of this article is organized as follows: Section 2 reviews the related works. Section 3 presents an NoC platform where A3MAP algorithms are performed and a novel and global A3MAP formulation to MIQP. In Section 4, a successive relaxation algorithm and a genetic algorithm are proposed as the efficient solutions of the MIQP. In Section 5, we present an application mapping approach in a large NoC. Section 6 shows experiment results in comparison with the previous state-of-art works [Murali and Micheli 2004; Tornero et al. 2008]. Section 7 is used for conclusion.

2. RELATED WORKS AND OUR CONTRIBUTIONS

In the last decade, application mapping problems have been mainly solved on a regular mesh network. In Hu and Marculescu [2003], a branch-and-bound algorithm was adopted for application mapping in a regular mesh-based NoC architecture, which minimized the total amount of power consumed in communications. Murali and Micheli [2004] presented NMAP that was a fast algorithm, where cores were mapped onto a regular mesh network under bandwidth constrains, aiming at minimizing average communication latency. Shin et al. [2004] explored 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. Hansson et al. [2005] proposed a unified algorithm, called UMARS, that couples mapping, path selection, and time-slot allocation, using a single consistent objective. The work presented in Chou et al. [2008] proposed an efficient technique for runtime application mapping onto a homogeneous NoC platform with multiple voltage levels. Chen et al. [2008] proposed a complier-based application mapping algorithm that consisted of task scheduling, processor mapping, data mapping, and packet routing to reduce energy consumption. However, since these solutions have been optimized only for a regular mesh network, they cannot be applied to various network architectures or their mapping quality severely deteriorates in irregular/custom networks.

Recently, heterogeneous cores have been considered for low energy consumption [Jang et al. 2010]. Smit et al. [2004] solved the problem of runtime task assignment on heterogeneous processors with task graphs restricted to the small number of vertices or the large number of vertices within degree no more than two. Carvalho et al. [2007] investigated the quality of several mapping heuristics promising for runtime use in NoC-based MultiProcessor SoCs (MPSoCs) with dynamic workloads, targeting NoC congestion minimization. Chang et al. [2008] proposed ETAHM to allocate tasks on a target multiprocessor system. It mixed task scheduling, mapping, and dynamic voltage scaling utilization in one phase and coupled an ant colony optimization algorithm. ADAM presented in Abdullah et al. [2008] was runtime application mapping in a distributed manner, targeting for adaptive NoC-based heterogeneous MPSoCs. However, the previous application mapping solutions have not considered the irregularity of NoC tiles and links which are caused by different sized heterogeneous PEs. Since the irregularities cause long detoured packets on a network, a lot of communication energy may be consumed or a quality-of-service requirement may not be guaranteed. Recently, Tornero et al. [2008] proposed a communication-aware topological mapping

technique for irregular NoCs, which matched the communication requirements of the application running on the cores with the existing network resources. Ghosh et al. [2009] proposed a technique for mapping application tasks to heterogeneous PEs on an NoC platform, operating at multiple voltage levels. Singh et al. [2009] described two runtime mapping heuristics for mapping applications onto NoC-based heterogeneous MPSoC. Le Beux et al. [2010] proposed an approach that concurrently optimizes the mapping and the partitioning of steaming applications on heterogeneous nodes. However, its mapping quality was low since it did not provide the efficient solution searching algorithm. Singh et al. [2010] described a number of communication-aware runtime mapping heuristics for the mapping of multiple applications onto an MPSoC platform in which more than one task can be supported by each PE. He et al. [2011] proposed a unified task scheduling and core mapping algorithm for various NoCs.

Such a different-sized PE is mainly considered in the topology synthesis and routing path allocation for application-specific NoC. Murali et al. [2006] presented a floorplan-aware design methodology that automated the synthesis of such application-specific NoC architectures. It considered the wiring complexity of NoCs during the topology synthesis process. Chan and Parameswaran [2008] presented NoCOUT, a methodology for generating an energy-optimized application-specific NoC topology which supports both point-to-point and packet-switched networks. Chatha et al. [2008] presented the design methodology and synthesis of an application-specific NoC architecture. It employed a three-phase synthesis approach consisting of core-to-router mapping, custom topology decision, and routing path generation. In Schafer et al. [2005], an adaptive deadlock-free routing algorithm was proposed to handle NoC layouts with embedded different sized cores. Bolotin et al. [2007] proposed hardware-efficient routing in irregular mesh NoCs and routing table size minimization based on static shortest path routing. Holsmark et al. [2008] listed the issues that a designer would encounter while designing a heterogeneous mesh topology for NoC using multiport or multiaccess point cores and presented two deadlock-free routing algorithms for irregular mesh networks.

In this article, we propose novel and global Architecture-Aware Analytic MAPping (A3MAP) algorithms. The proposed approach can be employed in most network architectures including regular/irregular mesh and custom networks. The main novelties and contributions are as follows.

—We formulate an application mapping problem to MIQP, based on a metric embedding technique. Then, we show that the formulation achieves excellent application mapping quality not only in regular networks but also in irregular/custom networks.
—We propose two effective heuristics solving the MIQP, based on a successive relaxation algorithm for short runtime and a genetic algorithm for high mapping quality. We show they provide a better trade-off between mapping quality and runtime for a small-scale network.
—We propose a partition-based application mapping approach for large-scale networks and show that it provides short runtime whereas it has little loss of mapping quality.

3. PROBLEM FORMULATION

The proposed application mapping approach is applied to NoC-based MPSoC. PEs in the MPSoC are placed on a regular mesh network, an irregular mesh network, or a custom network where a router is interconnected to a single PE and other routers. Each network link between routers or a PE and a router can have the same or different bandwidth that depends on the number of wires and operating clock frequency and wirelength that depends on the size of PEs and their physical design such as floorplanning and placement and global/detail routing. In addition, a network link can have uni- or bidirectional ways.

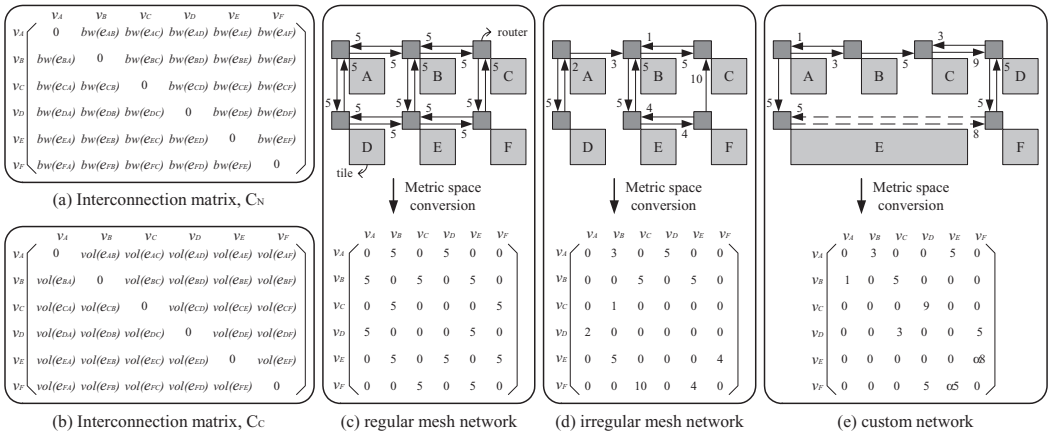


Fig. 2. Various graphs and their interconnection matrices.

One of the PEs performs the proposed A3MAP algorithms as a global manger that is commonly placed in the corner of the MPSoC. Whereas it is similar to Chou et al. [2008], our MPSoC does not need a control network which is responsible for delivering a control packet between a global manager and PE. Since control and data packets can be differentiated by different address spaces and a few control packets are not critical in packet routing performance, our MPSoC is equipped with only a data network that control packets share [Jang and Pan 2010b, 2011b]. The communication infrastructure is controlled by an Open Core Protocol (OCP) or an AMBA protocol. Packets are routed with the deterministic and minimum path and are controlled by wormhole switching. Each PE has its own computation capability and is considered as an independent subsystem.

Under such NoC architectures, we formulate an application mapping problem to MIQP using metric embedding. As inputs, we take a core graph and a network. A graph *G(V,E)* with n vertices is a directed graph, where each vertex $v_i \in V$ represents a core 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 core graph and $bw(e_{i,j})$ represents a 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 Figure 2(a). Each row in C_N represents an interconnection relation with respect to a single tile on NoC. Thus, C_N contains interconnection relations for an entire network, representing the metric space of a network. Similarly, we construct an $n \times n$ interconnection matrix C_C, corresponding to a core graph, where $c_{Ci,j} \in C_C$ is equal to $vol(e_{i,j})$ as shown in Figure 2(b).

For example, Figure 2(c), (d), and (e) show three network graphs and their metric spaces using the proposed interconnection matrix. In Figure 2(c) that is a regular mesh, all routers are interconnected by a bidirectional network link with the same bandwidth. Its interconnection matrix is symmetrically composed as shown under the network graph. In case of an irregular mesh network in Figure 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 asymmetrically composed. The irregular mesh network can be observed in a VFI-based NoC where each PE operates with its own voltage and frequency [Jang et al. 2010] and in NoC with faulty and degraded links by process and temperature variation [Markovsky et al. 2009].

In case of a custom network, there is slight difference in the composition of its interconnection matrix. In Figure 2(e), wirelength between tile E and tile F is different from other wirelengths due to tile E with a larger area. Since a packet has to cross each link within one cycle, a link between tile E and F may have more repeaters to accommodate a fast transmission time resulting in significantly higher energy consumption. The composition of an interconnection matrix for the custom network is similar to regular/irregular mesh networks except weight α is added in the matrix in order to consider efficient communication energy consumption. The hop count based on the assumption that all links consume the same communication energy is no longer suitable since links with different wirelength consume different communication energy. Let the energy consumption of each link, E_{link} computed as

$$E_{link} = E_{driver} + E_{repeaters}, \tag{1}$$

where E_{driver} and $E_{repeaters}$ are the energy consumed by the output driver of routers and repeaters on a link, respectively. If E_{link1} and E_{link2} is the energy consumption of sending one bit in a solid line and a dotted line respectively, α is the ratio of E_{link1} to E_{link2} ($= E_{link1} / E_{link2}$) where $E_{link1} < E_{link2}$. The weigh α ($0 < \alpha < 1$) reduces the available bandwidth of a long dotted link in a network such that our formulation makes the long dotted link less used. In Figure 2(e), let's suppose that dotted lines are three times longer than solid lines and a packet generated in tile *A* goes to tile *D*. The packet can choose either *A-B-C-D* or *A-E-F-D* as a routing path. Since two routing paths include the same hop counts, it takes the packet 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*. Thus, the path *A-E-F-D* may consume more communication energy than the path *A-B-C-D* since more repeaters may be inserted on the long dotted links or the router attached to tile *E* and *F* may be required to equip a stronger output driver. Therefore, it is good to assign a core with little communication to a tile with the long link or a core with a lot of communication to a tile with the short link for low dynamic energy consumption. If the energy consumption is linearly proportional to the length of wires in Figure 2(e) due to more repeaters and a stronger output driver, α is 1/3. The weight α lets a core with a lot of communication be mapped into a tile with short wires such that communication energy consumption can be further minimized. Similarly, our interconnection matrix easily accommodates other general cases.

Graph embedding [Matousek 2002] maps the vertices of graph *G(V,E)* into a chosen metric space by minimizing distortion. Thus, application mapping has a natural correspondence with graph embedding into a given two-dimensional metric space representing NoC. Thus, we seek to embed a core graph into the metric space of a network based on the interconnection matrices. The goal is that a core is mapped to each tile, satisfying the performance constraints in a core-mapped network while the number of communications generated between routers is minimal. If a network is exactly same as a core graph, graph embedding does not cause any distortion of the edges in the core graph. As a result, it always produces the best possible mapping quality on the network. However, since most core graphs are generally different from a network, some distortion is not evitable in a network. Then, the mapping quality is measured by the total distortion of embedding. By minimizing the extent by which edges in a core graph are stretched or distorted with intermediate tiles when embedded into a network, we seek to reduce the total amount of communications and obtain a better global application mapping solution in terms of energy consumption under performance constraints. Based on this concept, our concrete application mapping algorithm is formulated as follows.

With two interconnection matrices, C_N for a network and C_C for a core graph, we exactly formulate an application mapping problem to Mixed Integer Quadratic

Programming (MIQP). It is similar to a Field Programmable Gate Array (FPGA) placement problem proposed in Gopalakrishnan et al. [2006]. However, a crucial difference in our work is the use of metric space that accurately captures the interconnections of a network and a core graph. The application mapping problem is equivalent to determining the assignment of a core to each tile with low energy consumption under performance constraints. This core assignment action can be mathematically presented by an $n \times n$ permutation matrix P. Column indices and row indices in P represent core identifiers and tile identifiers, respectively. For example, if $P(i, j) = 1$, then core 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 a core graph is represented by $P^T C_C P$. Finally, P minimizing the difference between the permuted interconnection matrix of a core graph $P^T C_C P$ and the interconnection matrix of a network C_N for generating little communication between routers and minimizing the distortion of C_C for a short routing path can be found. For P that is orthogonal, we formulate the application mapping problem mathematically by our objective as

$$\min f_{obj} = \left\| P^T C_C P - C_N \right\|_F^2 = \| C_C P - P C_N \|_F^2 , \tag{2}$$

where $\|X\|_F = \sqrt{\sum_i \sum_j x_{i,j}^2}$, $x_{i,j} \in X$, that is, the Frobenius norm of the matrix X and $x_{i,j} \leq 0$ to satisfy bandwidth constraints, subject to integrity and linearity constrains as follows.

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

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

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

The constraints indicate that just one element in each row and each column is 1 and other elements are 0 in the permutation matrix P.

While our formulation has a convex quadratic object function, the binary constraints on the elements of P restrict the solution space to a nonconvex 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 MIQP that is NP-hard [Sahni and Gonzalez 1976]. Algorithms we take in MIQP are successive relaxation to quickly find an application mapping solution and a genetic algorithm to achieve a high mapping quality. In the next section, we describe how they are applied in the proposed A3MAP formulation minutely.

4. A3MAP ALGORITHMS

We present an effective heuristic based on successive relaxation of MIQP to a sequence of Quadratic Programming (QP), called A3MAP-SR, to quickly find the permutation matrix P that minimizes our objective f_{obj} in Eq. (2). In addition, we apply a genetic algorithm to find a better mapping solution, called A3MAP-GA, even though it takes a longer runtime than A3MAP-SR. A genetic algorithm is an efficient random searching algorithm based on a cycle crossover and a mutation operation.

4.1. A3MAP-SR

In this section, we solve our A3MAP formulated to MIQP based on a successive relaxation algorithm [Grossmann and Kravanja 1997]. The optimal MIQP formulation can

ALGORITHM 1: A3MAP-SR

Input: **MIQP**
Output: Permutation matrix P
relax $P(i, j) \in \{0, 1\}$ to $0 \leq P(i, j) \leq 1$;
set all $P(i, j)$ to a variable;
$i_{th} = n(V_N)$;
repeat
  solve relaxed MIQP only for variables $P(i, j)$ by QP solver;
  repeat
    find $max\{P\}$ and store its location to (i_{max}, j_{max}) for $\forall P(i, j)$ that is a variable;
    if $max\{P\} \geq 1/i_{th}$ **do**
      $P(i_{max}, j_{max}) = 1$ and a non-variable;
      $P(i, j_{max}) = 0$ and a non-variable, $\forall i = 1, 2, .., n(V_N)$;
      $P(i_{max}, j) = 0$ and a non-variable, $\forall j = 1, 2, .., n(V_N)$;
      i_{th} decreases by *1*;
    end if
  until ($max\{P\} < 1/ i_{th}$)
until (all $P(i, j)$ are a non-variable)

become QP if we relax the discrete constraint of Eq. (5) to a continuous constraint as follows.

$$0 \leq P(i, j) \leq 1 \tag{6}$$

Then, the key idea behind this algorithm is to use this QP as a subroutine. QP is solved much faster and scaled much better. Then, continuous values obtained by a QP solver are guided to 0 or 1 depending upon a predefined threshold. If any continuous value is less than the threshold, it is guided to 0. Otherwise, it is guided to 1. The proposed concrete successive relaxation algorithm employed in our A3MAP formulation is shown in Algorithm 1.

After relaxing the constraint of Eq. (5) to Eq. (6) in 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. Initial i_{th} is set to the number of tiles in a network and then as an initial threshold, we use $1/i_{th}$ to guide continuous $P(i, j)$ solved by a QP solver to 1, where the threshold indicates the expected average that variable $P(i, j)$ can get. On executing the successive relaxation, i_{th} decreases by 1 whenever any $P(i, j)$ is set to 1, which means the threshold gets increased. The rest of Algorithm 1 attempts to constrain continuous values solved by a QP solver to binary values inversely. We look for the maximum $P(i, j)$ and compare it to the threshold. If it is greater than the threshold, it is set to 1 and a nonvariable. In addition, all elements on the same row or column as the maximum $P(i, j)$ are also set to 0 and a nonvariable since the sum of elements on a single row or a single column in the permutation matrix P should be 1 from the constraints in Eq. (3) and (4). This procedure repeats if the next maximum $P(i, j)$ is also greater than the updated threshold. Otherwise, we again solve the relaxed MIQP for the rest of the variables $P(i, j)$ by a QP solver and continue to guide continuous values updated to binary values. If all $P(i, j)$ are guided to 0 or 1, we get the near-optimum permutation matrix P.

For example, we assume that an application with 5 cores and NoC with 5 tiles are given. In order to allocate the cores to the tiles, we should find a 5×5 permutation matrix where only 5 elements will be 1 and 20 elements will be 0 from our A3MAP formulation. We relax the discrete constraint in Eq. (5), set all $P(i, j)$ to a variable and set an initial threshold to 1/5. Let the relaxed MIQP solved by a QP solver be as shown in Figure 3(a). Then, our A3MAP-SR algorithm looks for the maximum $P(i, j)$ among the variables. In Figure 3(a), $P(3, 2)$ is 0.5 as the maximum. Since it is greater

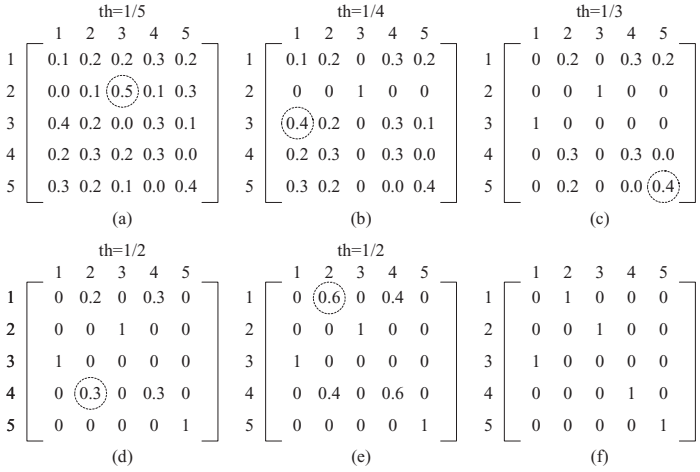


Fig. 3. Guiding continuous $P(i, j)$ to binary $P(i, j)$.

than the initial threshold 1/5, $P(3, 2)$ is guided to 1 and then $P(3, k)$ and $P(k, 2)$ where $\forall k = 1, 2, \ldots, 5$ are set to 0 as shown in Figure 3(b). The guided $P(i, j)$ to 0 or 1 is set to a nonvariable and the threshold is updated to 1/4. Since the next maximum $P(1, 3)$ and $P(5, 5)$ are greater than the threshold 1/4 and 1/3 respectively, $P(1, 3)$ and $P(5, 5)$ is guided to 1 and $P(1, k)$, $P(k, 3)$, $P(5, k)$, and $P(k, 5)$ where $\forall k = 1, 2, \ldots, 5$ are set to 0 as shown in Figure 3(c) and (d). Then, the guided $P(i, j)$ to 0 or 1 is set to a nonvariable and the threshold is updated to 1/2. Next, since the next maximum $P(2, 4)$ is less than the threshold 1/2 in Figure 3(d), we again solve the relaxed MIQP by a QP solver for the rest of variable $P(i, j)$ as shown in Figure 3(e). Then, the guiding procedure repeats until all $P(i, j)$ are guided to 0 or 1 as shown in Figure 3(f).

4.2. A3MAP-GA

The successive relaxation algorithm solves MIQP with reasonable mapping quality and runtime. For application mapping with a high mapping quality, runtime may be less important than the reduction of hop count and communication energy consumption. To reflect this demand, we develop another heuristic using a genetic algorithm. A genetic algorithm reproduces the principle of natural evolution to solve search and optimization problems. It is a promising technique for a system-level design and is especially suitable for multiple-objective optimization problems. Starting with an initial population, a genetic algorithm evolves a population using crossover and mutation operations. A genetic algorithm was previously used in Shin and Kim [2004] to explore the design space efficiently for task assignment, mapping, and routing path allocation. However, since the performance of a genetic algorithm depends on encoding, crossover, and mutation schemes, we need to select different schemes that fit well in our A3MAP formulation.

Algorithm 2 is the pseudocode of our genetic algorithm for MIQP. First, we generate two arbitrary permutation matrices as parent individuals. A crossover **s**cheme is widely acknowledged as critical to the success of a genetic algorithm. A crossover scheme should be capable of producing a new feasible solution (i.e., new child individual) by combining the good characteristics of parent individuals while the child individuals should be considerably different from their parent individuals. We use a cycle crossover [Oliver et al. 1987] which prevents over two cores being allocated into the same tile.

ALGORITHM 2: A3MAP-GA

Input: MIQP
Output: Permutation matrix P
generate arbitrary *parent 1*;
repeat
  generate arbitrary *parent 2*;
  (*child 1*, *child 2*) = cycle crossover (*parent 1*, *parent 2*);
  mutation of *child 1* and *2* by pair-wise swapping;
  parent 1 = one of two children with minimum f_{obj} computed by *Eq. (2)* for the next evolution;
until (no improvement during i-iterations)

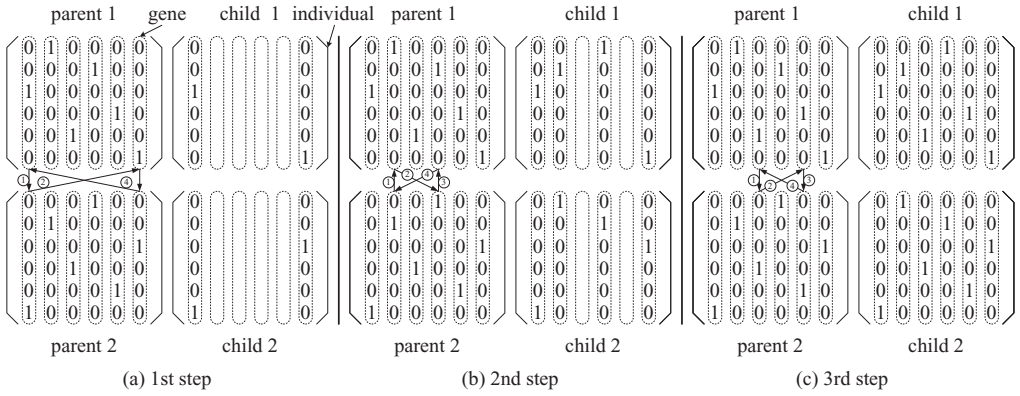


Fig. 4. Cycle crossover.

Figure 4 shows how to generate two child individuals from two independent parents based on the cycle crossover. In the first step, child 1 inherits a column from parent 1 and child 2 inherits a column from parent 2. We start to choose any inherited column in parent 1. In Figure 4(a), the first columns are arbitrarily chosen in parent 1 and 2 and then child 1 and 2 inherit the column from parent 1 and 2, respectively. Next, we look for any column in parent 1 including the same gene as the first column of parent 2. Since the sixth column of parent 1 contains the same gene, the sixth columns of parent 1 and 2 are selected. Then, child 1 and 2 inherit the sixth column from parent 1 and 2, respectively. Again, we look for any column in parent 1 including the same gene as the sixth column of parent 2. However, this procedure stops since the first column of parent 1 is again selected. In the second step, inversely, child 1 inherits a column from parent 2 and child 2 inherits a column from parent 1. The selection procedure is similar to the first step. In Figure 4(b), the second column is arbitrarily selected in parent 1 and 2 and then child 1 and 2 inherit the second column from parent 2 and 1, respectively. Next, we look for any column in parent 2 including the same gene as the second column of parent 1. Since the fourth column of parent 2 contains the same gene, the fourth columns of parent 1 and 2 are selected. Then, child 1 and 2 inherit the fourth column from parent 2 and 1, respectively. This procedure repeats until the chosen column is again chosen like the first step. If all columns of children are not filled with the column of parents after the second step, the first and the second steps repeat with the unselected columns of parents by turns. In our example, the columns of children are completely filled after the third step as shown in Figure 4(c).

Then, a mutation operation is performed for each child. In this operation, two columns randomly selected are swapped to generate a new individual. Then, the swapping is valid only when it reduces the number of traffic. The pairwise swapping operation for

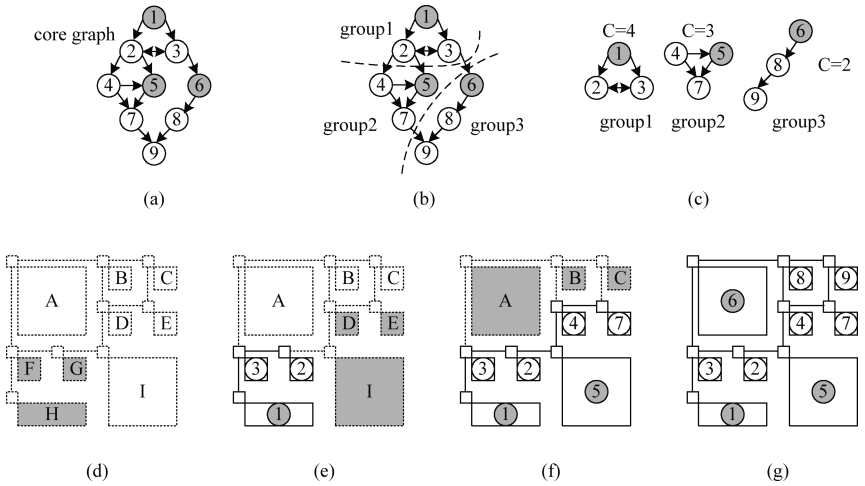


Fig. 5. Partition-based application mapping flow for large-scale NoC.

each child continues until the pair of swapped columns cannot minimize our object function, that is, Eq. (2) any more. After the mutation operation, we choose one of two children with the minimum distortion as the parent 1 for the next evolution. Those operations repeat until there is no improvement for several (i) iterations. If there is not any improvement for i-iterations, a permutation matrix providing the near-optimal performance to NoC is obtained.

Our genetic algorithm makes the superior column of parents passed down to their children and the best child again becomes any parent (parent 1) for the next evolution. In addition, since the new elements of columns (parent 2) from the outside are supplied, the possibility of local minima is relatively lower. This approach can efficiently cover wider solution spaces even if runtime is longer than A3MAP-SR.

5. A3MAP FOR LARGE-SCALE NOC

Whereas A3MAP enables global optimization, the runtime of A3MAP algorithms becomes longer and longer as the number of cores or tiles increases. Even NMAP [Murali and Micheli 2004] that is one of the fastest mapping algorithms takes a long runtime in case that the number of cores or tiles is greater than 70. Recently, since NoCs include more cores for high performance and applications are more complex, an application mapping approach with better trade-off between runtime and performance is required. Therefore, in this section, we propose a partition-based application mapping approach which can be easily extended to any large-scale NoC. In addition, we show that A3MAP algorithms are suitable for the partition-based approach.

Figure 5 is an example of the proposed partition-based application mapping approach for large-scale NoCs. In Figure 5(a), any application is scheduled to 9 cores and NoC with 9 tiles are given for a simple explanation. We assume that cores 1, 5, and 6 have two times higher computation complexity than other cores, tiles A, H, and I have four, two, and four times higher computational capacity than other tiles, respectively, and all communication volumes between the cores is just 1. We first perform k-way min-cut partitioning for the cores as shown in Figure 5(b) where k is 3. The number of the core groups partitioned *(k)* can be determined by a user. For example, if runtime constraints are much tighter than performance constraints, a number of groups are desirable. Otherwise, few groups are desirable to high performance. All groups are not required to include the same number of cores. Then, the groups are sorted in a decreasing order

by the amount of communication inside each group and then mapped in the order. As shown in Figure 5(c), group 1 is first mapped, group 2 is then mapped, and group 3 is last mapped since the communication volume of group 1, 2, and 3 are 4, 3, and 2, respectively.

Large NoC (R) including N tiles also requires being partitioned to several subnetworks (R') with a convex region. This near-convex region selection problem can be formulated as

$$\min[L_1(R') + L_1(R - R')], \tag{7}$$

where $L_1(R)$ is the total Manhattan distance between all tiles inside region R. The objective in Eq. (7) is to find a subregion R' with N' tiles of which the computational capacity must be greater than or equal to the computational complexity of cores included in the mapped subgroup. The time complexity of the near-region selection algorithm is known as $O(NlogN)$ in Chou et al. [2008]. Then, cores in each group are mapped to tiles inside the selected convex region.

In Figure 5(d), the first convex region including tile F, G, and H is selected by Eq. (7) and then cores in group 1 which have the maximum communication are mapped to tiles in the convex region by A3MAP algorithms as shown in Figure 5(e). Since the size of its interconnection matrix is 3×3, the runtime of A3MAP algorithms is hundreds of times shorter than that of A3MAP algorithms for a 9×9 interconnection matrix. Then, among the rest of the regions, the next convex region with tiles D, E, and I is selected. Based on the mapping result of group 1, cores in group 2 are mapped to tiles in the convex region as shown in Figure 5(f). Even though a 6×6 interconnection matrix is generated, its runtime is similar to that of a 3×3 interconnection matrix. This is because some variables $P(i, j)$ in the 6×6 interconnection matrix are already fixed by mapping cores in group 1. Last, based on the prior two mapping results, cores in group 3 are mapped to tiles in the last convex region as shown in Figure 5(g). Even if the interconnection matrix generated by A3MAP algorithms is 9×9, its runtime is also similar to that of the prior groups. The reason is that the number of variables $P(i, j)$ that A3MAP algorithms solve for the last group are the same as the number of variables $P(i, j)$ in the first and second groups.

Even though most of the application mapping algorithms can be applied to the proposed partition-based approach, the A3MAP algorithms such as A3MAP-SR and A3MAP-GA are more suitable than NMAP and A3MAP solved by a full search algorithm, called A3MAP-FS. This is because the runtime of A3MAP-FS and the mapping quality of NMAP may be not satisfied in the partition-based approach. The partition-based approach gets some inevitable mapping quality loss since cores are allocated to limited tiles inside the selected convex region. In order to minimize the performance degradation in the partition-based approach, each group and convex region should include as many cores or tiles as possible, that is, the number of groups and convex region partitioned should be as little as possible. In case of applying A3MAP-FS in the partition-based approach, the number of groups cannot be reduced since it takes A3MAP-FS over 2 seconds to map just 10 cores. As a result, A3MAP-FS applied in the partition-based approach does not show the efficient trade-off between runtime and performance. In case of applying a fast application mapping algorithm like NMAP in the partition-based approach, its mapping quality cannot be satisfactory even though the number of groups partitioned is few. For example, NMAP itself achieves on average 7%, 8%, and 11% lower application mapping quality than A3MAP-FS when 10, 11, and 12 cores are mapped, respectively. As a result, even if NMAP performs on a large convex region with a number of cores, its mapping quality is significantly low in the partition-based approach. Therefore, the proposed A3MAP algorithms which provide

Table I. Hop Count and Runtime Compared to A3MAP-FS

# of core or tile	Hop count increase (%) normalized to A3MAP-FS			Runtime (times) normalized by A3MAP-FS		
	A3MAP-SR	A3MAP-GA	NMAP	A3MAP-SR	A3MAP-GA	NMAP
9	1.3	1.1	2.0	155	72	202
10	1.7	1.2	7.1	432	373	470
11	2.0	1.5	8.1	3819	1555	5287
12	2.6	1.8	10.9	47K	14K	67K
13	3.8	2.2	15.3	564K	153K	875K

better mapping quality than NMAP and shorter runtime than A3MAP-FS are suitable for the partition-based approach.

6. EXPERIMENTAL RESULTS

We implement the A3MAP-SR algorithm in CPLEX11.2 [AIMMS 2012] and the A3MAP-GA algorithm in C++. All experiments were performed on a Linux machine with Intel 2.4 GHz CoreDuo and 8GB RAM. We repeat each application mapping for ten times and compute their average to obtain reliable statistics.

6.1. Regular Mesh Network

We carry out experiments by applying A3MAP algorithms on an MPEG-4 Video Object Plane Decoder (VOPD) [Tol and Jaspers 2002], E3S benchmark suites [Dick 2012], and synthetic benchmarks. The first application including 16 cores is mapped onto a 4×4 regular mesh network. The second benchmark consists of three applications, that is, consumer, Auto-Industry (AI), and telecomm containing 12, 24, and 30 tasks respectively, which are scheduled to 9, 16, and 25 by Hu and Marculescu [2003] and then mapped to a 3×3, 4×4, and 5×5 regular mesh network, respectively. In addition, we use Task Graph For Free (TGFF) [Dick et al. 1998] to generate several sets of synthetic applications. The number of tasks and the volume of communication are randomly selected according to specific distributions.

Since the number of cores may be generally different from the number of tiles, the preprocessing is required. If the number of cores is less than the number of tiles, additional cores without any communication and computation are added in the core graph. If the number of cores is greater than the number of tiles, we perform $n(V_N)$-way min-cut or balanced core partitioning, where $n(V_N)$ is the number of tiles and the computational complexity of the grouped cores must be less than the computational capacity of PE. The min-cut partitioning reduces communication energy consumption between tiles that are assigned cores whereas the balanced partitioning for the computational complexity of cores improves the system performance by encouraging parallel computing. Then, we perform the proposed A3MAP-SR and A3MAP-GA algorithms. Finally, we allocate the routing path of packets by a Dijkstra's shortest path algorithm to compute total hop count on given networks.

Table I shows how exact and fast solution A3MAP algorithms can find in synthetic benchmarks with 9 to 13 cores, compared to the full searching approach, called A3MAP-FS that provides the best solution in terms of application mapping quality. A3MAP-SR and A3MAP-GA provide near-best solutions since their mapping qualities are just 3.8% and 2.2% lower on average than A3MAP-FS, respectively, when 13 cores are mapped in a regular mesh network. However, their runtimes are about 564 and 153 thousand times shorter than A3MAP-FS, respectively. On the contrary, the mapping quality of NMAP [Murali and Micheli 2004] which is one of the most famous core mapping algorithms is on average 15.3% lower than A3MAP-FS even if its runtime is the shortest.

Table II shows the application mapping results performed with industrial benchmarks. A3MAP-SR greatly reduces on average total hop count up to 7.4% in a regular

Table II. Hop Count Comparison for Industrial Benchmarks in Regular Mesh Networks

Application	NMAP	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
Average			7.425		11.75

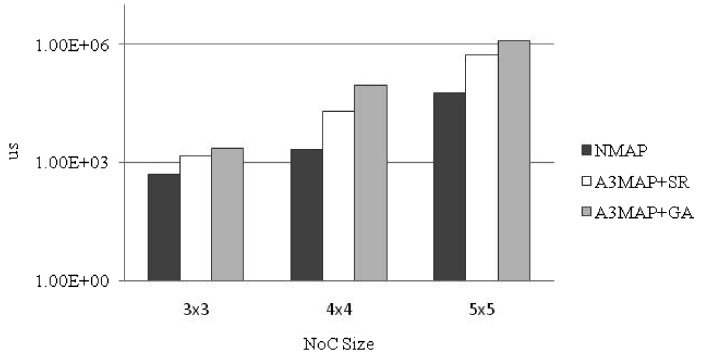


Fig. 6. Runtime comparison for industrial benchmarks in regular mesh networks.

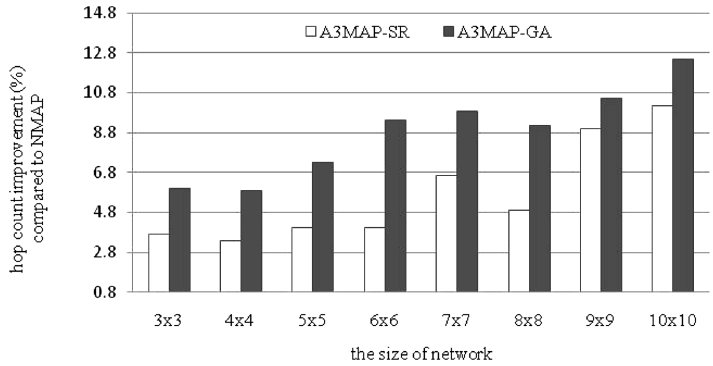


Fig. 7. Hop count improvement of A3MAP algorithms compared to NMAP for synthetic benchmarks in regular mesh networks.

mesh network, compared to NMAP. Moreover, A3MAP-GA achieves on average 3.8% and 11.8% less hop count than A3MAP-SR and NMAP, respectively. On the contrary, the runtimes of A3MAP-SR and A3MAP-GA are longer than NMAP as shown in Figure 6.

Figure 7 shows the hop count improvement of A3MAP algorithms compared to NMAP on various regular mesh networks. We generate ten synthetic task graphs per network by TGFF and compute their average improvement. As shown in Figure 7, A3MAP-SR and A3MAP-GA reduce on average total hop count up to 5.7% and 8.8%, respectively, compared to NMAP. In addition, A3MAP algorithms provide much higher mapping quality than NMAP as the size of networks increases. Finally, even if A3MAP algorithms are optimized for all kinds of network, A3MAP algorithms achieve higher application mapping quality than NMAP optimized for only regular mesh networks.

6.2. Irregular Mesh Network

In this section, our A3MAP algorithms prove more merits on irregular mesh networks. We perform NMAP on an irregular mesh network even if NMAP is optimized for a

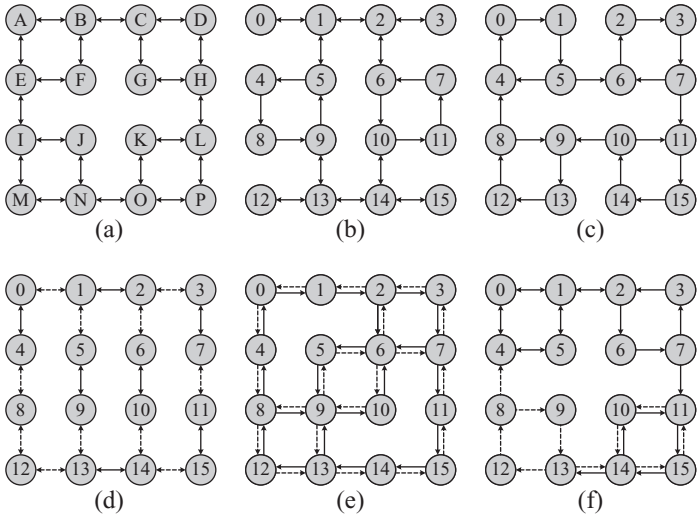


Fig. 8. Various irregular mesh networks.

Table III. Hop Count Comparison for VOPD Benchmark in Irregular Mesh Networks

Application	NMAP	CMAP	A3MAP-SR	A3MAP-GA
Fig. 9(a)	4215	4911	4205	4189
Fig. 9(b)	8704	6544	5820	5345
Fig. 9(c)	6405	7194	6185	5257
Fig. 9(d)	4923	5507	4374	4199
Fig. 9(e)	4950	4259	4191	4189
Fig. 9(f)	7424	6497	4832	4081
Average	6104	5819	4935	4543
Ratio	1	0.953	0.808	0.744

regular mesh network. We also implement Tornero et al. [2008] which considers irregular networks for application mapping, called CMAP. Figure 8 shows six irregular mesh networks on which we experiment the application mapping algorithms. Figure 8(a) has only bidirectional links, Figure 8(b) has both bidirectional and unidirectional links, and Figure 8(c) has only unidirectional links. Both directions of links have the same bandwidth in Figure 8(d) whereas each direction of links has different bandwidth in Figure 8(e). In the figure, solid lines have two times higher bandwidth than dotted lines. In Figure 8(f), links have all irregularities mentioned in Figure 8(a) through (e).

Table III shows the mapping results of an MPEG-4 VOPD application on the irregular mesh networks. A3MAP algorithms achieve better application mapping improvement in an irregular mesh network than that in a regular mesh network. For example, whereas A3MAP-SR and A3MAP-GA reduce on average total hop count only by 1.0% and 3.9% in a regular mesh network, respectively, they reduce on average total hop count by 19.2% and 25.6% in irregular mesh networks, respectively, compared to NMAP. In addition, A3MAP algorithms achieve much higher mapping quality than NMAP in Figure 8(f) which is the most complex network. That is because A3MAP formulation avoids mapping cores with a lot of communication volume to tiles with little bandwidth and considers the direction of communication and various network topologies adaptively. Even if CMAP is optimized for irregular networks, its mapping quality is slightly better than that of NMAP. Since CMAP just considers the irregular wirelength of links, it is difficult to improve mapping quality on irregular mesh networks which

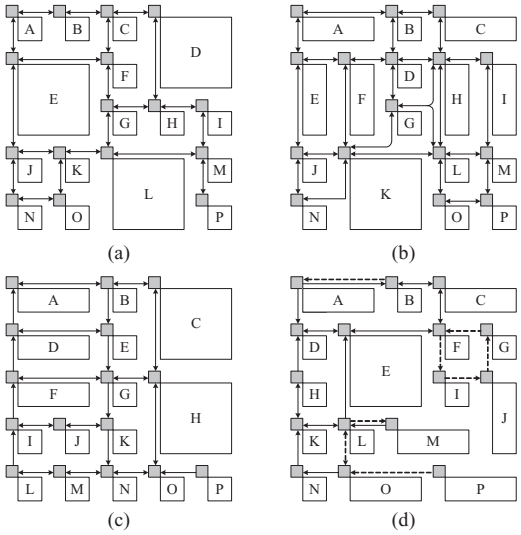


Fig. 9. Custom NoC topologies.

Table IV. Hop Count and Wirelength Comparison for VOPD Benchmark in Custom Networks

	Total hop count				Total travel distance (wirelength) by all packets			
Application	NMAP	CMAP	A3MAP-SR	A3MAP-GA	NMAP	CMAP	A3MAP-SR	A3MAP-GA
Fig. 9(a)	4488	4752	4531	4087	5879	6300	5332	4543
Fig. 9(b)	4264	4119	4248	4199	5505	4135	5049	4215
Fig. 9(c)	6296	5598	5867	5150	7835	6842	7434	5613
Fig. 9(d)	5524	5735	4263	4263	9196	9627	5170	5170
Average	5143	5051	4727	4425	7104	6726	5746	4885
Ratio	1.000	0.982	0.919	0.860	1.000	0.947	0.809	0.688

have the different direction and irregular bandwidth of links. Furthermore, the runtime of CMAP is slightly slower than NMAP. Finally, the proposed A3MAP algorithms provide better application mapping to NoC including irregular mesh networks than the state-of-the-art mapping algorithms.

6.3. Custom Network

In this section, we perform A3MAP algorithms on custom networks with an MPEG-4 VOPD benchmark and then they are also compared to NMAP and CMAP. Figure 9 shows various custom networks where A3MAP algorithms are performed. Figure 9(a) contains three PEs that have four times larger area than others. Due to the PEs, a custom network including irregular interconnections and different wirelengths is synthesized. Similarly, 16 PEs that have one of three different areas are floorplanned as shown in Figure 9(b). Figure 9(c) has both unidirectional and bidirectional links and Figure 9(d) has links with different bandwidth. In Figure 9, links have one of two different wirelengths and assume that a long link consumes two times higher communication energy than a short link since the long link requires a strong output driver or a number of repeaters. Therefore, α is set to 1/2 when the interconnection matrix of a network is composed.

Table IV shows the application mapping results on the custom networks. A3MAP-SR and A3MAP-GA reduce on average total hop count up to 8.1% and 14%, respectively, compared to NMAP. We also measure total distance traveled by all packets, which is

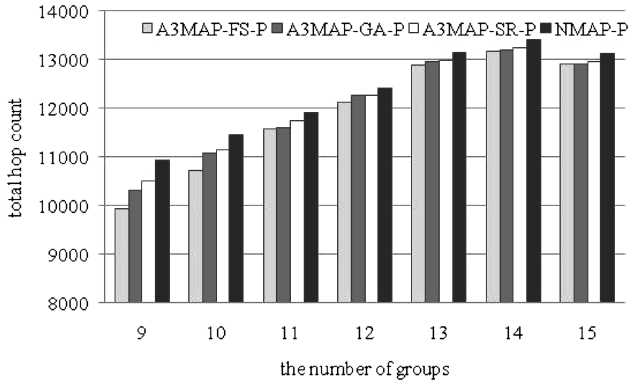


Fig. 10. Hop count comparison of partition-based application mapping algorithms.

directly related to communication energy consumption than the total hop count in custom networks including various links. In Table IV, A3MAP-SR and A3MAP-GA reduce total distance traveled by all packets on average up to 19.1% and 31.2%, respectively, compared to NMAP. CMAP also improves application mapping quality on custom networks whereas it has no benefit on irregular mesh networks. However, its hop count is 6.5% and 12.4% greater than those of A3MAP-SR and A3MAP-GA, respectively and total distance traveled by packets is 14.6% and 28.4% longer than those of A3MAP-SR and A3MAP-GA, respectively. These results prove that our weighted interconnection matrix is efficient enough for saving communication energy since the improvement of distance traveled by all packets is greater than that of hop count. Therefore, the weighted interconnection matrix is desirable for custom networks. Similarly, A3MAP can be easily manageable for more complex NoC by controlling the weighted interconnection matrix. Finally, the proposed A3MAP algorithms provide energy-efficient application mapping to NoC including various networks, compared to the state-of-the-art mapping algorithms.

6.4. Large-Scale NoC

We prove A3MAP algorithms suitable for the partition-based approach which is described in Section 5. In this experiment, core graphs with one hundred cores are generated by TGFF and mapped to a 10×10 regular mesh network. The cores are partitioned to 9 to 15 groups with the minimum cuts by hMETIS[2012] and then the groups are sorted in a decreasing order by the amount of communication inside each group. The network is also partitioned to 9 to 15 groups with a convex region. Then, A3MAP-FS which achieves the best mapping quality, A3MAP-SR, A3MAP-GA, and NMAP which is one of the fastest solutions, perform application mapping for each ordered core group on a selected convex region, which are called A3MAP-FS-P, A3MAP-SR-P, A3MAP-GA-P and NMAP-P, respectively.

Figure 10 shows the hop count comparison of the application mapping algorithms. As the number of groups increases, that is, the number of cores included in each group decreases, A3MAP-GA-P and A3MAP-SR-P achieve similar mapping quality to A3MAP-FS. In addition, total hop count of most partition-based application mapping algorithms tends to increase since cores are mapped to tiles included in a restricted convex region. On the contrary, if the number of groups decreases, the number of cores included in each group increases. As a result, since cores can be mapped to tiles included in a larger convex region, most of the application mapping algorithms improve their

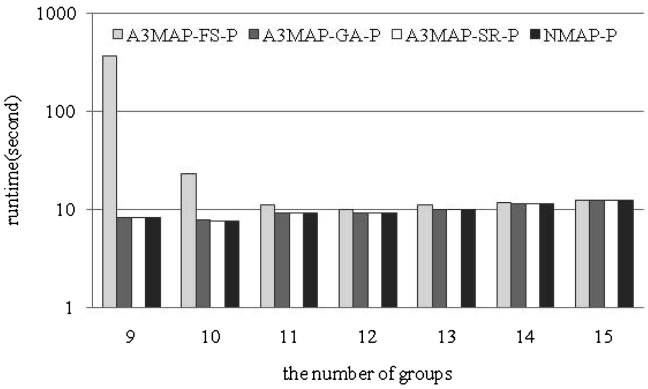


Fig. 11. Runtime comparison of partition-based application mapping algorithms.

hop count. The mapping quality of A3MAP-GA-P and A3MAP-SR-P is similar to that of A3MAP-FS whereas it is much greater than that of NMAP-P.

However, since the runtime of A3MAP-FS rapidly gets long in the larger convex region as shown in Figure 11, it shows an inefficient trade-off between mapping quality and runtime. The application mapping quality of A3MAP-FS-P can be obtained by A3MAP-GA-P or A3MAP-SR-P if A3MAP-GA-P or A3MAP-SR-P performs in a network that is partitioned to fewer groups. In addition, their runtime is much faster than that of A3MAP-FS-P. For example, the mapping quality of A3MAP-FS-P on a network partitioned to 10 groups is worse than the mapping quality of A3MAP-SR-P and A3MAP-GA-P on a network partitioned to 9 groups in Figure 10. In addition, the runtime of A3MAP-SR-P and A3MAP-GA-P on a network partitioned to 9 groups is much faster than that of A3MAP-FS-P on a network partitioned to 10 groups. On the contrary, even though NMAP-P shows slightly faster runtime than A3MAP-SR-P and A3MAP-GA-P in Figure 11, its mapping quality is much worse in a network with few groups in Figure 10. Therefore, A3MAP algorithms are more suitable for the partition-based approach in large-scale NoC.

Next, we check how many hop counts the partition-based A3MAP algorithms increase in regular networks, irregular networks, and custom networks. We use synthetic benchmarks with 25, 36, 49, 64, 81, and 100 cores. We make each partitioned group not include more than 16 cores such that 25, 36, 49, 64, 81, and 100 cores are partitioned to 2, 3, 4, 4, 6, and 7 core groups, respectively. The groups are not required to include the same number of cores when the cores are partitioned with the minimum cuts. We perform the partition-based A3MAP algorithms ten times with different core graphs and networks.

Figure 12 shows the hop count of A3MAP-SR-P normalized by A3MAP-SR in regular networks, irregular networks, and custom networks, called A3MAP-SR-P-R, A3MAP-SR-P-I, and A3MAP-SR-P-C, respectively. The hop count performed by A3MAP-SR-P slightly increases compared to A3MAP-SR due to the partitioning process. In addition, the hop count increases in most networks as the number of PEs increases. Consequently, A3MAP-SR-P increases on average total hop count by 1.5%, 2.0%, and 2.6% in regular mesh, irregular mesh, and custom networks, respectively.

Similarly, Figure 13 shows the hop count of A3MAP-GA-P normalized by A3MAP-GA in regular mesh, irregular mesh, and custom networks, called A3MAP-GA-P-R, A3MAP-GA-P-I, and A3MAP-GA-P-C, respectively. A3MAP-GA-P increases on average total hop count by 1.7%, 2.5%, and 3.2% in regular mesh, irregular mesh, and custom networks, respectively.

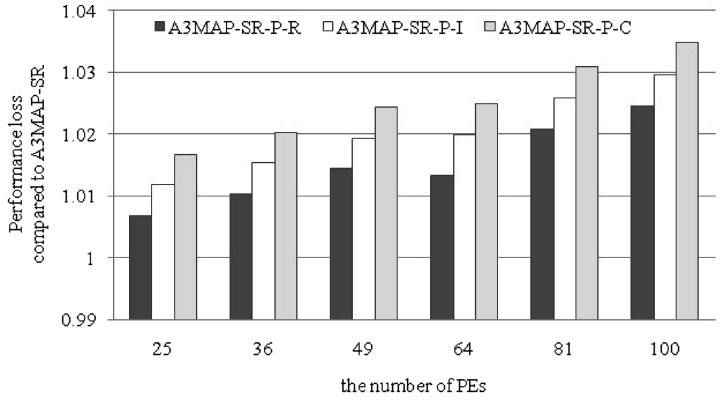


Fig. 12. Hop count of A3MAP-SR-P normalized by A3MAP-SR on regular mesh, irregular mesh, and custom networks.

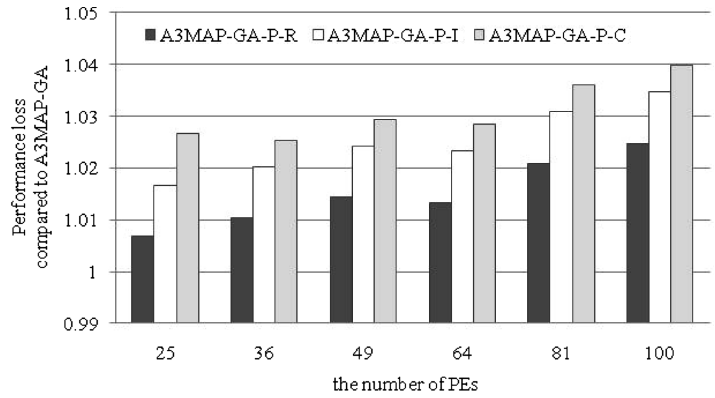
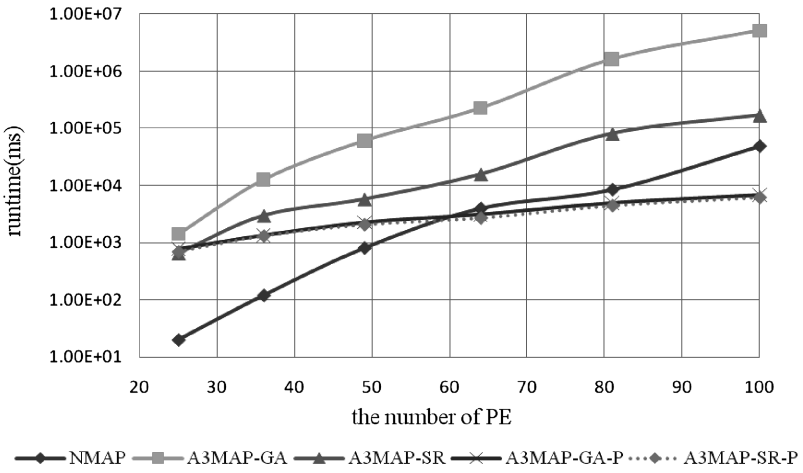


Fig. 13. Hop count of A3MAP-GA-P normalized by A3MAP-GA on regular mesh, irregular mesh, and custom networks.

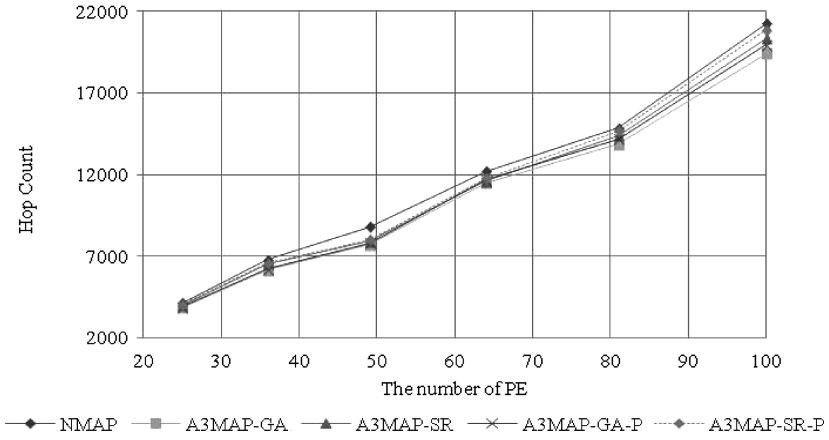
Figure 14(a) shows the runtime of A3MAP-GA, A3MAP-SR, A3MAP-GA-P, A3MAP-SR-P, and NMAP. A3MAP-SR-P and A3MAP-GA-P show that the increase of their runtime is less than others as the number of PEs increases. Consequently, when the number of PEs is more than 60, they are the fastest even if their runtimes in 25 PEs are similar to A3MAP-SR. Figure 14(b) shows their hop counts, where A3MAP-SR-P and A3MAP-GA-P achieve less hop count than NMAP. Even A3MAP-GA-P achieves better mapping quality than A3MAP-SR as the number of PEs increases. This is because applications with many tasks have more chance of being partitioned with fewer cuts. However, in case that a number of cuts increase, A3MAP-SR achieves better mapping quality than A3MAP-GA-P. Finally, the A3MAP algorithms are more suitable for the partition-based approach in large-scale NoCs since they provide an efficient trade-off between runtime and mapping quality.

7. CONCLUSION

In this article, we propose novel and global Architecture-Aware Application MAPping (A3MAP) algorithms for NoC. Based on a metric embedding technique, we analytically formulate an application mapping problem to MIQP. Then, the MIQP is solved by two effective heuristics, that is, a successive relaxation algorithm providing short runtime and a genetic algorithm providing high mapping quality. In addition, we propose the



(a) runtime



(b) hop count

Fig. 14. Overall comparison.

partition-based approach for large-scale NoCs, where A3MAP algorithms provide an efficient trade-off between runtime and mapping quality. Experimental results show that our A3MAP algorithms greatly reduce hop count on various networks, compared to the previous state-of-the-art works. Especially, A3MAP algorithms show more merits on irregular mesh and custom networks. All networks can be easily converted to the simple but efficient interconnection matrix such that our A3MAP algorithms have no limitation to map cores to tiles on any arbitrary, faulty, and degraded network. Furthermore, A3MAP algorithms are easily manageable for low communication energy consumption and high performance by an architecture-aware analytical manner.

REFERENCES

AIMMS. 2012. Optimization software for operations research applications. http://www.aimms.com

BORKAR, S. 2007. Thousand core chips: A technology perspective. In *Proceedings of the IEEE/ACM Design Automation Conference (DAC'07)*.

BENINI, L. AND MICHELI, D. G. 2002. Network on chips: A new SoC paradigm. *Comput. 35*, 1, 70–78.

BOLOTIN, E., CIDON, I., GINOSAR, R., AND KOLODNY, A. 2007. Routing table minimization for irregular mesh NoCs. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE'07)*. 1–6.

CARVALHO, E., CALAZANS, N., AND MORAES, F. 2007. Heuristics for dynamic task mapping in NoC-based heterogeneous MPSOCs. In *Proceedings of the International Workshop on Rapid System Prototyping*. 34–40.

CHAN, J. AND PARAMESWARAN, S. 2008. NoCOUT: NoC topology generation with mixed packet-switched and point-to-point networks. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC'08)*.

CHANG, J. M. AND PEDRAM, M. 2000. Codex-dp: Co-Design of communicating systems using dynamic programming. *IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. 10*, 7, 732–744.

CHANG, P. C., WU, I. W., SHANN, J. J., AND CHUNG, C. P. 2008. ETAHM: An energy-aware task allocation algorithm for heterogeneous multiprocessor. In *Proceedings of the IEEE/ACM Design Automation Conference (DAC'08)*. 776–779.

CHATHA, S. K., SRINIVASAN, K., AND KONJEVOD, G. 2008. Automated techniques for synthesis of application specific network-on-chip architectures. *IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. 27*, 8.

CHEN, G., LI, F., SON, W. S., AND KANDEMIR, M. 2008. Application mapping for chip multiprocessor. In *Proceedings of the IEEE/ACM Design Automation Conference (DAC'08)*. 620–625.

CHOU, C. L., OGRAS, Y. U., AND MARCULESCU, R. 2008. Energy- and performance-aware incremental mapping for networks on chip with multiple voltage levels. *IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. 27*, 10, 1866–1879.

DALLY, J. W. AND TOWLES, B. 2001. Route packets, not wires: On-Chip interconnection networks. In *Proceedings of the IEEE/ACM Design Automation Conference (DAC'01)*. 746–749.

DICK, P. R. 2012. Embedded system synthesis benchmarks suites (E3S). http://www.ece.northwestern.edu/~dickrp/e3s/

DICK, P. R., RHODES, L. D., AND WOLF, W. 1998. TGFF: Task graphs for free. In *Proceedings of the International Workshop on Hardware/Software Codesign*. 97–101.

DUTTA, S., JENSEN, R., AND RIECKKMANN, A. 2001. Viper: A multiprocessor SoC for advanced set-top box and digital tv systems. *IEEE Des. Test Comput. 18*, 5, 21–31.

FARUQUE, A. A. M., KRIST, R., AND HENKEL, J. 2008. ADAM: Run-Time agent-based distributed application mapping for on-chip communication. In *Proceedings of the IEEE/ACM Design Automation Conference (DAC'08)*. 760–765.

GHOSH, P., SEN, A., AND HALL, A. 2009. Energy efficient application mapping to NoC processing elements operating at multiple voltage levels. In *Proceedings of the International Symposium on Networks-on-Chip*. 80–85.

GOPALAKRISHNAN, P., LI, X., AND PILEGGI, L. 2006. Architecture-Aware fpga placement using metric embedding. In *Proceedings of the IEEE/ACM Design Automation Conference (DAC'06)*. 460–465.

GROSSMANN, E. I. AND KRAVANJA, Z. 1997. *Mixed-Integer Nonlinear Programming: A Survey of Algorithms and Applications, 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.

HANSSON, A., GOOSSENS, K., AND RADULESCU, A. 2005. A unified approach to constrained mapping and routing on network-on-chip architectures. In *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis (CODES + ISSS'05)*. 75–80.

HE, O., DONG, S., JANG, W., BIAN, J., AND PAN, Z. D. 2011. UNISM: Unified scheduling and mapping for general networks on chip. *IEEE Trans. VLSI Syst. 99,* 1–14.

HMETIS. 2012. Hypergraph and circuit partitioning. http://glaros.dtc.umn.edu/gkhome/views/metis

HOLSMARK, R., PALESI, M., AND KUMAR, S. 2008. Deadlock free routing algorithms for irregular mesh topology NoC systems with rectangular regions. *J. Syst. Archit. 54*, 3–4, 384–396.

HU, J. AND MARCULESCU, R. 2003. Energy-Aware mapping for tile-based NoC architectures under performance constraints. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC'03)*. 233–239.

HU, J. AND MARCULESCU, R. 2005. Communication and task scheduling of application-specific networks-on-chip. *IEEE Proc. Comput. Digit. Tech. 152*, 5, 643–651.

JANG, W. 2011. Architecture and physical design for advanced networks-on-chip. Ph.D. dissertation, University of Texas at Austin.

JANG, W. AND PAN, Z. D. 2010a. A3MAP: Architecture-Aware analytic mapping for networks-on-chip. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC'10)*. 523–528.

JANG, W. AND PAN, Z. D. 2010b. An SDRAM-aware router for networks-on-chip. *IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. 20,* 10, 1572-1585.

JANG, W., DING, D., AND PAN, Z. D. 2010. Voltage and frequency island optimizations for many-core/NoC designs. In *Proceedings of the International Conference on Green Circuits and Systems.* 217–220.

JANG, W. AND PAN, Z. D. 2011a. A voltage-frequency island aware energy optimization framework for networks-on-chip. *IEEE J. Emerg. Select. Topics Circ. Syst. 1*, 3, 420–432.

JANG, W. AND PAN, Z. D. 2011b. Application-Aware NoC design for efficient sdram access. *IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. 30*, 10, 1521–1533.

JANG, W., HE, O., YANG, J. S., AND PAN, Z. D. 2011. In *Proceedings of the International Conference on Computer-Aided Design.* 207–212.

LE BEUX, S., BOIS, G., NICOLESCU, G., LANGEVIN, M., AND PAULIN, P. 2010. Combining mapping and partitioning exploration for NoC-based embedded systems. *J. Syst. Archit. 56*, 7, 223–232.

MARKOVSKY, Y., PATEL, Y., AND WAWRZYNEK, J. 2009. Using adaptive routing to compensate for performance heterogeneity. In *Proceedings of the International Symposium on Networks on Chip*.12–21.

MATOUSEK, J. 2002. *Lectures in Discrete Geometry*. Springer.

MURALI, S., MELONI, P., ANGIOLINI, F., ATIENZA, D., CARTA, S., BENINI, L., MICHELI, D. G., AND RAFFO, L. 2007. Designing application-aware networks on chips with floorplan information. In *Proceedings of the International Conference on Computer-Aided Design.*

MURALI, S. AND MICHELI, D. G. 2004. Bandwidth-Constrained mapping of cores onto NoC architecture. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE'04).* 896–901.

OLIVER, I., SMITH, D., AND HOLLAND, J. 1987. A study of permutation crossover operators on the traveling salesman problem. In *Proceedings of the Conference on Genetic Algorithms.* 224–230.

SAHNI, S. AND GONZALEZ, T. 1976. P-Complete approximation problems. *J. ACM 23*, 3, 555–565.

SCHAFER, F. F. M., HOLLSTEIN, T., ZIMMER, H., AND GLESNER, M. 2005. Deadlock-Free routing and component placement for irregular mesh-based network-on-chip. In *Proceedings of the International Conference on Computer-Aided Design.* 238–245.

SHIN, D. AND KIM, J. 2004. Power-Aware communication optimization for networks-on-chip with voltage scalable links. In *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis*. 170–175.

SINGH, A. K., JIGANG, W., PRAKASH, A., AND SRIKANTHAN, T. 2009. Efficient heuristics for minimizing communication overhead in NoC-based heterogeneous MPSoC platforms. In *Proceedings of the International Symposium on Rapid System Prototyping.* 55–60.

SINGH, A. K., SRIKANTHAN, T., KUMAR, A., AND JIGANG, W. 2010. Communication-Aware heuristics for runtime task mapping on NoC-based MPSoC platforms. *J. Syst. Archit. 56*, 7, 242–255.

SMIT, T. L., SMIT, J. M. G., HURINK, L. J., BROERSMA, H., PAULUSMA, D., AND WOLKOTTE, T. P. 2004. Run-Time assignment of tasks to multiple heterogeneous processors. In *Proceedings of the 4th PROGRESS Workshop on Embedded Systems.* 185–192.

STMICROELECTRONICS. 2012. Nomadik multimedia processors. http://www.st.com

TEXAS INSTRUMENTS. 2012. Wireless handset solutions: OMAP platform. http://www.ti.com

TORNERO, R., ORDUNA, M. J., PALESI, M., AND DUATO, J. 2008. A communication-aware topological mapping technique for NoCs. In *Proceedings of the 14th International Conference on Parallel and Distributed Computing.* 910–919.

VAN DER TOL, B. E. AND JASPERS, G. T. E. 2002. Mapping of the mpeg-4 decoding on flexible architecture platform. In *Proce. SPIE 4674*, 1, 1–13.

Received February 2011; revised January 2012; accepted February 2012