Integrated Mapping and Synthesis Techniques for Network-on-Chip Topologies with Express Channels

SANDEEP D'SOUZA, Indian Institute of Technology Kharagpur

SOUMYA J, Birla Institute of Technology and Science Pilani, Hyderabad Campus

SANTANU CHATTOPADHYAY, Indian Institute of Technology Kharagpur

The addition of express channels to a traditional mesh network-on-chip (NoC) has emerged as a viable solution to solve the problem of high latency. In this article, we address the problem of integrated mapping and synthesis for express channel-based mesh NoC topologies. An integer linear programming-based formulation has been presented for the mapping problem followed by a constructive heuristic for simultaneous application mapping and synthesis for an express channel-based NoC. The static and dynamic simulation results indicate that the obtained mappings lead to significant reduction in both average packet delay and network energy consumption. The obtained synthesized topologies were also found to be much more power efficient compared to conventional express channel topologies.

Categories and Subject Descriptors: B.4.3 [Input/Output and Data Communications]: Interconnections (Subsystems)—*Topology*; B.8.2 [Performance and Reliability]: Performance Analysis and Design Aids

General Terms: Network-on-Chip, Application Mapping, Express Channel Synthesis

Additional Key Words and Phrases: Constructive heuristic, integer linear programming

ACM Reference Format:

Sandeep D'souza, Soumya J, and Santanu Chattopadhyay. 2015. Integrated mapping and synthesis techniques for network-on-chip topologies with express channels. *ACM Trans. Archit. Code Optim.* 12, 4, Article 40 (November 2015), 26 pages.

DOI: <http://dx.doi.org/10.1145/2831233>

1. INTRODUCTION

With the growth in complexity of VLSI systems and increasing functionality, the number of cores on a single chip continues to grow. In today's systems, it is common to find multiple intellectual property (IP) cores integrated onto a single chip. These cores may perform different functionalities in such a system-on-chip (SoC) environment. Traditional SoCs use a bus to interface the different cores. With the increase in the number of cores, bus-based SoCs suffer degradation in performance. Thus, in present and upcoming VLSI systems, where highly parallelized systems are the norm and fabrication technology is at its limits, optimizing on-chip communication has become critical to ensure performance increments in line with Moore's law. The network-on-chip (NoC)

Authors' addresses: S. D'souza (corresponding author), Department of Electronics and Electrical Communication Engineering, Indian Institute of Technology Kharagpur, West Bengal, India, 721302; email: smd-souza@ece.iitkgp.ernet.in; Soumya J., Department of Electrical and Electronics Engineering, Birla Institute of Technology and Science Pilani, Hyderabad Campus, Hyderabad, Telangana, India, 500078; email: soumyaj@hyderabad.bits-pilani.ac.in; S. Chattopadhyay, Department of Electronics and Electrical Communication Engineering, Indian Institute of Technology Kharagpur, West Bengal, India, 721302; email: santanu@ece.iitkgp.ernet.in.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© 2015 ACM 1544-3566/2015/11-ART40 \$15.00

DOI: <http://dx.doi.org/10.1145/2831233>

Fig. 1. (a) Mesh network. (b) CMesh network.

paradigm [Dally and Towles 2001; Atienza et al. 2008; Marculescu and Bogdan 2009] has evolved as a viable approach to solve the communication problem. In an NoC, multiple cores communicate through a fabric of routers interconnected according to a given topology. One or more IP cores may be attached to a single router. The nature of the network allows it to scale up to possibly hundreds of cores.

To interconnect a large number of cores, multiple network topologies have been proposed for NoCs. Due to their simplicity, low design complexity, and easy mapping onto 2D silicon substrates, the ring topology [Pham et al. 2006] and 2D mesh topology have become the most popular. Up to a few cores, the ring network is simple and efficient to use. For example, the Intel Sandy Bridge Architecture [Intel Sandy Bridge 2011] uses a ring interconnection network to interface the CPU cores with the GPU. For a core count of the order of a few tens, the mesh architecture provides better scalability. It provides a regular and symmetric structure that is easy to design and replicate. Most industrial designs with a core count up to hundred, such as the 64-core TILE processor from Tilera [Wentzlaff et al. 2007] and the 80-core Tera-scale chip from Intel [Vangal et al. 2008] use the mesh topology. However, these topologies face scalability issues when the number of cores is of the order of a few hundred [Grot et al. 2009]. As the number of cores grow, the intercore communication latency increases tremendously.

For large and parallel multiprocessor systems-on-chip (MPSoCs), the mesh topology faces many issues and is inherently slower as messages are forwarded hop by hop. To alleviate the problem of a large network diameter, multiple variants of the mesh topology have been proposed, such as concentrated mesh (CMesh) [Balfour and Dally 2006], butterfly fat tree [Pande et al. 2003], and mesh-of-tree [Leighton 1992]. In Balfour and Dally [2006], increasing core concentration at a single router has been proposed as a means to reduce the network diameter. CMesh [Balfour and Dally 2006] achieves this reduction by co-locating multiple terminals at each network interface with a crossbar interconnect. Mesh and CMesh are illustrated in Figure 1.

The addition of express channels [Grot et al. 2009; Kim et al. 2007; Kumar et al. 2007] to mesh-based NoCs has been proposed in recent literature. By means of express channels, nonadjacent routers are directly connected, thus reducing network latency due to multiple hops. Significant among them are the topologies of flattened butterfly (FB) [Kim et al. 2007] and multidrop express channels (MECS) [Grot et al. 2009]. The FB topology involves flattening a conventional butterfly network for use on a chip with the aim of reducing the hop count through the use of high-radix routers. Each router has a dedicated link to all routers sharing the same row or column as the source router. However, this causes large interconnection complexity, with a channel count that is quadratic with respect to the number of nodes [Grot et al. 2009]. To overcome this

Fig. 2. (a) FB network. (b) MECS network.

problem, the MECS topology has been proposed in Grot et al. [2009]. MECS uses a point-to-multipoint communication fabric that provides a high degree of connectivity in a bandwidth-efficient manner [Grot et al. 2009]. A single channel is used to connect multiple routers by means of a multidrop system. The FB and MECS network topologies are illustrated in Figure 2.

For a given application, an SoC consists of several IP cores, with different inter-communication bandwidth requirements between them. A key point for optimizing performance is the reduction in communication latency and power consumption, with an increase in throughput. This necessitates proper mapping of IP cores to the routers of the interconnection fabric. This mapping of cores to the NoC must be performed according to the communication needs of the said application. This is referred to as the application mapping problem. The application mapping problem is NP-hard [Pop and Kumar 2004] and is a well-researched domain. The input to the application mapping problem is the *core graph* [Faruque et al. 2008]. The core graph contains the communication requirements between the different IP cores, which make up the application. The application mapping problem consists of mapping the core graph to the *network topology graph* [Faruque et al. 2008]. For conventional architectures, a good number of algorithms have been proposed in the literature. However, for express channel-based NoCs, there is a lack of application mapping techniques. In Zhu et al. [2014], a turn reduction algorithm for mapping (TRAM) has been proposed for express channel-based NoCs. However, the TRAM approach fails to provide good solutions in all situations, especially in the case of larger designs with a higher core count. To overcome this shortcoming, in the present work we have proposed a constructive heuristic-based approach to solve the problem of application mapping in an express channel-based NoC.

The issue of high router radix also makes an express channel network with all-to-all links (within the same row or column as the source router) unfeasible in real designs. This is especially true for a system with a large core count, where area and power constraints make high router radix and a high interconnection density unfeasible. In such scenarios, system designers may need to restrict router radix and add express channels judiciously only when and where required. In this work, we also present an integrated mapping and synthesis technique to synthesize semicustom express channel-based NoCs. The salient features of this work are as follows:

- A constructive heuristic has been proposed to efficiently map cores onto an express channel-based mesh NoC with concentration one (i.e., one IP core per router).
- A Kernighan-Lin (KL) partitioning-guided constructive heuristic has been proposed to efficiently map cores onto an express channel-based CMesh NoC with core

concentration (i.e., multiple IP cores are mapped to a single router). In this work, this concentration has been taken as four.

- An integrated mapping and synthesis technique for express channel-based NoCs has been proposed. The proposed approach gives the system designer the ability to restrict the router radix and maps the IP cores onto the simultaneously synthesized topology.

The article is organized as follows. Section 2 presents the related work. Section 3 presents a mathematical formulation of the problem as well as delay models for express channel-based NoCs. Our proposed approach (for both the mapping and the synthesis problems), as well as an integer linear programming (ILP)-based mapping formulation, has been elaborated in Section 4. Section 5 enumerates the experimental results. Section 6 concludes the article.

2. RELATED WORK

The design of systems that meet different requirements of an application is a complex task. From the point of view of an NoC, custom topologies may be developed to meet the requirement of a given application. This is a complex process involving floorplan generation, custom network routing components, and application-specific routing algorithms. Several works covering synthesis of complex application-specific NoCs have been presented in academia [Benini 2006; Srinivasan et al. 2006].

A simpler approach involves mapping an application onto regular topologies to minimize a given cost function. The cost function can be in terms of several metrics, such as communication latency, network throughput, network power, and area. Several algorithms have been proposed in the literature to solve the application mapping problem. A comprehensive survey can be found in Sahu and Chattopadhyay [2013]. Several mixed ILP-based formulations have been proposed in Faruque et al. [2008], Ozturk et al. [2007], and Ghosh et al. [2009]. ILP-based approaches provide the optimal mapping solution for a given application and network. However, these approaches require a large amount of computational resources and do not scale easily to large core counts. Some methods propose constraint relaxation in the ILP formulation to provide accurate results with a reduction in computation time. In Tosun [2011], a clustering-based relaxation for ILP formulations has been proposed. However, to enable scaling to larger designs with lower computation costs, heuristics such as NMAP [Murali and Micheli 2004a] and metasearch techniques have been proposed in academia. NMAP is a communication-aware three-stage iterative mapping technique that has been proposed in Murali and Micheli [2004a]. It consists of an initial mapping phase, followed by minimum path computation and a final phase where pair-wise swapping of vertices is undertaken to refine the solution. SUNMAP [Murali and Micheli 2004b] is another such tool that performs RTL-level NoC topology exploration by minimizing area and power consumption. In Hu and Marculescu [2003], a branch-and-bound algorithm to minimize communication energy of mapping has been proposed. A two-stage KL partitioning-based mapping heuristic for mesh and mesh-of-tree topologies has been proposed by Sahu et al. [2014]. A discrete particle swarm optimization (DPSO)-based approach has been proposed in Pham et al. [2006]. Energy- and buffer-aware application mapping (EBAM) [Celic and Bazlamacci 2014] handles the application mapping issue as a joint optimization problem for minimizing the energy consumption and buffer utilization simultaneously by employing a genetic algorithm. In Moein-Darbari et al. [2009], the CGMAP method has been proposed, which employs a chaos-genetic-based algorithm that obtains close results compared to other metaheuristic algorithms.

The presence of express channels adds interconnections between nonadjacent routers. Hence, the preceding methods cannot be applied directly. For a mesh-based

Fig. 3. Two paths with latencies from node 1.

NoC, the latency of communication between two tiles depends purely on the Manhattan distance between the nodes. In an express channel-based mesh network, every router has a direct connection to the other routers in the same row or column. Therefore, in the absence of a turn, a packet can directly travel from the source to the destination. Thus, the express channels introduce an extra dependence on the number of turns in the routing path. Zhu et al. [2014] have proposed a TRAM onto express channel-based NoCs (MECS). The TRAM approach [Zhu et al. 2014] uses the KL heuristic to place heavily communicating cores in the same row or column of the network topology, thus reducing the number of turns required for communication. Assigning cores to routers within the row is an assignment problem. TRAM employs the Hungarian method [Kuhn 2005] to optimally perform this assignment. A packet delay model and mathematical formulation of the application mapping problem for express channel-based NoCs has also been proposed in Zhu et al. [2014]. We elaborate on this model further in Section 3.

For an express channel-based mesh network with large network diameter, the link latency to the routers farther away in the same row or column may sometimes be more than the total latency to the nodes in adjacent rows or columns. The TRAM approach causes suboptimal mappings for such networks with a large diameter. The initial row-based core clustering in TRAM provides a poor starting solution. This problem has been enumerated using an example 8×8 network with 64 routers (Figure 3). In the example, let the nodes be numbered row-wise from left to right, with the top left node numbered as 1. The number within the node is used to indicate the packet delay incurred by a packet traveling from the first node to that particular node. It can be seen that the latency from node 1 to node 7 is 12. The latency from node 1 to node 10 is 11. This is lower than the latency from node 1 to node 7, even though nodes 1 and 10 do not share the same row or column. However, the TRAM [Zhu et al. 2014] algorithm will prefer node 7 over node 10 for mapping a core with a greater degree of communication with the core mapped to node 1. This causes a poor mapping solution. In addition, for core graphs with a lower degree of connectivity (as is the case for most real applications), placing several cores in a single row may not make sense, especially when all of them do not communicate. Hence, in such situations, a net increase in the number of turns may possibly occur using TRAM.

In this article, a constructive heuristic has been proposed to solve the mapping problem. The proposed single-stage heuristic effectively maps cores with a high degree of communication close to each other and generates a mapping solution in a constructive fashion. The method scales efficiently to larger designs providing efficient mapping solutions with low computation time.

Another approach toward system design can be the synthesis of a semicustom topology [Ogras and Marculescu 2006]. Semicustom topologies can be designed by taking a general-purpose topology and adding additional express channels as per the requirements of the designer. For a given mapping onto a mesh-based topology, an iterative link addition approach has been proposed in Ogras and Marculescu [2006]. Between each pair of cores, long distance links are added, and the benefit of link addition is calculated. Based on the benefit, the required links are added. It has been found that the addition of such links improves both the static and dynamic properties of the network [Ogras and Marculescu 2006].

Based on the constructive heuristic proposed in this article, we also present an integrated mapping and synthesis approach. The approach intelligently adds dedicated express links based on the router radix limitations specified by the system designer.

3. STATEMENT OF THE PROBLEM

3.1. Definitions

Definition 1. The *core graph* of an application is a directed graph $G(C, E)$. Each node $c_i \in C$ of the graph represents an IP core. The directed edges $e_{ij} \in E$ represent the communication between the cores c_i and c_j . Each edge e_{ij} has a weight denoted by $comm_{ij}$, which represents the bandwidth requirement of the communication from core c_i to c_j .

Definition 2. The *topology delay graph* is an undirected graph $T(R, L)$. Each node $r_i \in R$ of the graph represents a router of the network topology. The directed edges $l_{ij} \in L$ represent the link between routers r_i and r_j . Each edge l_{ij} has a weight denoted by d_{ij} , which represents the link latency of the communication from r_i to r_j .

Definition 3. A *mapping solution* is defined as the mapping of each core $c_i \in C$ to a router $r_j \in R$. Each router may have one or more nodes. A mapping can be one-to-one where a single core $c_i \in C$ is mapped to a router $r_j \in R$, or a many-to-one mapping where a cluster of cores is mapped to the nodes of a single router. The size of this cluster is same for each router and is referred to as the *concentration factor* of the network. For an application graph with n cores, the set of mapping solutions are the permutations $map(i) = j$ for $i, j \in \{0, 1, 2, 3, \dots, n-1\}$.

Definition 4. The *average packet delay (APD)* is the average latency faced by the data packets while traversing the network. For a given *core graph* $G(C, E)$ and the *topology delay graph* $T(R, L)$, the APD depends on the *mapping solution*. For a given mapping solution, the APD is calculated by

$$APD = \frac{\sum_{i=1}^n \sum_{j=1}^n comm_{ij} * d_{map(i)map(j)}}{\sum_{i=1}^n \sum_{j=1}^n comm_{ij}}, \quad (1)$$

where $comm_{ij}$ represents the bandwidth requirement of the communication from c_i to c_j and $d_{map(i)map(j)}$ represents the network communication delay between the routers to which cores c_i and c_j are mapped.

3.2. Topology Delay Models

The delay offered to a packet by the network depends on the mapping of the source and destination cores, as well as the interconnections of the topology. The delay experienced by a packet is a combination of *router delay*, *link delay*, and delay due to *router contention*. The unit-length link delay T_L is the number of clock cycles between adjacent tiles. The delay offered by a link can be assumed to be directly proportional to its length. Router delay T_R is the number of clock cycles a packet takes to go through a router and depends on the number of router pipeline stages. The router contention latency T_C is proportional to the traffic in a network. For each router traversed, the packet experiences router delay as well as contention delay. For a packet traveling from source core c_i to destination core c_j , the delay can be given by

$$d_{map(i)map(j)} = a * (T_R + T_C) + length * T_L, \quad (2)$$

where a is the number of routers encountered on the path taken by the packet and $length$ is the link distance traversed by the packet. Both a and $length$ depend on the topology as well as the path chosen by the routing algorithm.

In an express channel-based network, the routing algorithm must favor the use of express channels as much as possible. Normal hop-by-hop forwarding is used only in the case of contention. Hence, dimension-order routing is typically used for express channel-based networks. Techniques such as adaptive routing [Palesi et al. 2009; Qian et al. 2012] may generate a large number of turns [Zhu et al. 2014], causing most packets to go through in a hop-by-hop fashion, thus causing greater packet latency. Dimension order routing routes the packet first to the correct position in a higher dimension before attempting to route in the next dimension.

For, a mesh-based NoC employing dimension order routing, the number of routers traversed by a packet is equal to one more than the *Manhattan distance* M_{ij} between the source and destination router. All routers from source to destination contribute to the delay experienced by the packet. This is due to the hop-by-hop packet forwarding of a mesh NoC. The delay experienced by a packet traveling from router r_i to r_j , can be given by Zhu et al. [2014].

$$d_{ij} = (M_{ij} + 1)(T_R + T_C) + M_{ij}T_L \quad (3)$$

For an express channel-based mesh NoC employing dimension order routing, the number of routers traversed by a packet from source to destination is either of two cases:

- Two routers (source and destination) when the source and destination are either in the same row or column (i.e., there exists a direct express channel). The source router will directly send the packet to the destination router, except in the case of router contention.
- Three routers (source, destination, and turning point) when both source and destination are not in the same row or column (i.e., a turn needs to be made).

When the source and destination routers are not in the same row or column, a turn is made necessary. The *turn function* [Zhu et al. 2014] δ_{ij} is used to determine whether packets from router r_i to r_j need to make a turn. For an express channel-based mesh network, with M rows and N columns and nodes numbered from left to right and top to bottom, the turn function can be calculated as follows:

$$\begin{aligned} \delta_{ij} &= 0 && \text{if } i - (i \bmod N) = j - (j \bmod N) \\ &= 0 && \text{if } i \bmod N = j \bmod N = 1 \\ &= 1 && \text{otherwise.} \end{aligned} \quad (4)$$

Using the turn function, the delay experienced by a packet traveling from router r_i to r_j , in an express channel-based NoC can be given by Zhu et al. [2014].

$$d_{ij} = (\delta_{ij} + 2)(T_R + T_C) + M_{ij}T_L \quad (5)$$

For an express channel-based CMesh NoC with concentration $conc_{node}$, the delay function is slightly different. Concentration implies that $conc_{node}$ number of cores are mapped to a single router. Increasing concentration leads to an increase in the router radix, thus increasing its complexity. In most designs, the concentration, $conc_{node}$ is taken as 4. The concentration in the X direction is given by $conc_x$ and in the Y direction by $conc_y$. Hence, the concentration can be given by

$$conc_{node} = conc_x * conc_y. \quad (6)$$

In this case, when dimension order routing is used, the number of routers traversed by a packet from the source node to destination node is one of the three cases:

- One router (shared router) when the source and destination nodes share the same router.
- Two routers (source and destination) when the source and destination nodes have corresponding routers in either the same row or column (i.e., there exists a direct express channel). The source router will directly send the packet to the destination router, except in the case of router contention.
- Three routers (source, destination, and turning point) when neither source and destination nodes have routers that are neither in the same row nor in the same column (i.e., a turn needs to be made).

For an express channel-based CMesh network with $M \times N$ cores, the router topology will form a mesh with $M/2$ rows and $N/2$ columns. If the nodes are numbered from left to right and top to bottom such that $i, j \in \{0, 1, 2, 3 \dots, M * N - 1\}$, the turn function can be calculated as follows:

$$\begin{aligned} \delta_{ij} &= 0 && \text{if } i - (i \bmod 2N) = j - (j \bmod 2N) \\ &= 0 && \text{if } \left\lfloor \frac{i}{2} \right\rfloor \bmod(N) = \left\lfloor \frac{j}{2} \right\rfloor \bmod(N) \\ &= 1 && \text{otherwise} \end{aligned} \quad (7)$$

When the source and destination node share a router, the packet encounters only one router delay. The *locality function*, β_{ij} is used to determine whether packets from $node_i$ to $node_j$ need to make a turn. For an express channel-based CMesh network with M rows and N columns and the nodes numbered from left to right and top to bottom such that $i, j \in \{0, 1, 2, 3 \dots, M * N - 1\}$, the locality function can be calculated as follows:

$$\begin{aligned} \beta_{ij} &= 0 && \text{if } i - (i \bmod 2N) = j - (j \bmod 2N) \\ &&& \text{AND } \left\lfloor \frac{j}{2} \right\rfloor \bmod(N) = \left\lfloor \frac{i}{2} \right\rfloor \bmod(N) \\ &= 1 && \text{otherwise} \end{aligned} \quad (8)$$

Using the turn function and the locality function, the delay experienced by a packet traveling from $node_i$ to $node_j$, in an express channel-based CMesh NoC can be given by

$$d_{ij} = (\delta_{ij} + \beta_{ij} + 1)(T_R + T_C) + \beta_{ij}M_{map(i)map(j)}T_L, \quad (9)$$

where, $M_{map(i)map(j)}$ is the Manhattan distance between the routers to which the source node $node_i$ and the destination $node_j$ are mapped.

3.3. Application Mapping Problem Formulation

For an NoC, the application mapping problem can be formulated in terms of minimizing the APD of the network. As given in Zhu et al. [2014], the problem can be formulated as follows:

Given, an application with a *core graph* $G(C, E)$ with n cores and a network topology with a *topology delay graph* $T(R, L)$.

Find the optimal solution, $map(i) = j$, where $i, j \in \{1, 2, 3, \dots, n\}$,

To minimize the APD

$$APD = \frac{\sum_{i=1}^n \sum_{j=1}^n comm_{ij} * d_{map(i)map(j)}}{\sum_{i=1}^n \sum_{j=1}^n comm_{ij}} \quad (10)$$

4. OUR APPROACH

In this section, we present an integrated mapping and synthesis technique for express channel-based mesh networks. The application mapping problem described in Section 3 has the form of a quadratic assignment problem (QAP) [Zhu et al. 2014]. In general a QAP is found to be *NP-hard* [Leighton 1992]. Such problems can be solved using ILP techniques. However ILP techniques require a large amount of computational resource and time, and as the number of cores increase, these methods do not scale. Thus, there is a need for efficient methods which generate good mapping solutions, keeping in mind the architecture of the network topology. Two such express channel-based network topologies have been considered in our work:

- The express channel-based mesh network (topology 1) with a concentration factor of one ($conc_{node} = 1$) (i.e., each router in the network fabric has one core mapped to it).
- The express channel-based CMesh network (topology 2) with a concentration factor of four ($conc_{node} = 4$) (i.e., each router in the network fabric has four cores mapped to it).

Both of the preceding topologies share similarities in their structure; however, increase in router radix in topology 2 necessitates proper clustering of cores and optimization of intercluster communication latency, as compared to intercore communication latency in topology 1. The CMesh network is preferred in the case of a large number of cores, where increased router complexity is preferred over a large network diameter. However, in many situations, design constraints do not allow the use of high-radix routers. In such a scenario, a custom topology must be synthesized that meets the prescribed design constraints. Using the formulations proposed in Section 3, we propose to provide good mapping solutions for the given topologies using the following methods:

- An ILP-based formulation for application mapping in an express channel-based NoC
- A constructive heuristic-based approach (for topology 1)
- A modified KL clustering followed by a constructive heuristic-based approach (for topology 2)
- A constructive heuristic-based approach for mapping onto a simultaneously synthesised custom express channel topology (integrated mapping and synthesis approach) based on the maximum router radix specified by the system designer.

4.1. ILP Formulation for Express Channel-Based NoCs

This section presents an ILP formulation for the problem of mapping.

4.1.1. Parameters and Variables. The parameters and variables used in the ILP formulation are noted in Table I.

Table I. ILP Parameters and Variables

Parameters and Variables	Definitions
d_{st}	The precalculated network communication delay between the routers s and t .
$comm_{ij}$	The bandwidth requirement from core i to core j .
m_i^s	Binary variable: $m_i^s = 1$ if core i is mapped to the router s ; otherwise, $m_i^s = 0$.
P_{ij}^{st}	Binary variable: $P_{ij}^{st} = 1$ if a path exists between routers s and t to which cores i and j have been mapped, respectively; otherwise, $P_{ij}^{st} = 0$.

4.1.2. Objective Function. Our objective is to minimize the APD by selecting suitable routers in the NoC for mapping. The objective function can be formulated as follows. If cores i and j are mapped to routers s and t and a path exists between them in the network, P_{ij}^{st} is equal to 1. This multiplied by d_{st} gives the communication delay between routers s and t to which cores i and j have been mapped. Communication delay multiplied by bandwidth, averaged with bandwidth of all edges, gives the APD, which has to be minimized over all edges in the core graph. E is the set of edges in the core graph, R is the set of routers, and C is the set of cores.

$$\min \left(\left[\sum_{e_{ij} \in E} comm_{ij} \left(\sum_{s,t \in R} d_{st} \times P_{ij}^{st} \right) \right] \div \sum_{e_{ij} \in E} comm_{ij} \right) \quad (11)$$

4.1.3. Constraints. The following is the set of constraints framed to solve the mapping problem:

(1) *Mapping constraints:*

—Each core has to be mapped onto only one router.

$$\forall i \in C, \sum_{s \in R} m_i^s = 1 \quad (12)$$

—Each router can have at most one core mapped onto it.

$$\forall s \in R, \sum_{c_i \in C} m_i^s \leq 1 \quad (13)$$

(2) *Constraints for core graph edges:*

—Each edge present in the core graph has to be mapped onto a path in the NoC considered.

$$\forall e_{ij} \in E, \forall s, t \in R, m_i^s + m_j^t - P_{ij}^{st} \leq 1 \quad (14)$$

and,

$$2 \times P_{ij}^{st} \leq m_i^s + m_j^t$$

This completes the formulation. The objective function along with the constraint set can be fed to any ILP solver to get mapping for minimizing the APD. We have used the CPLEX tool [CPLEX 2013] for this purpose. However, except in the case of very small NoCs, it takes a huge amount of computation time to arrive at the solution. Hence, we have proposed a constructive heuristic to obtain mappings for larger core graphs, which is explained in the next section.

4.2. Constructive Heuristic for Express Channel-Based Mesh NoCs

We now propose a constructive heuristic-based approach for the problem of application mapping in express channel-based NoCs assuming a single core mapped to each router. The basic idea behind the approach is to constructively generate a mapping solution

such that highly communicating cores are mapped close to each other. The order of cores to be mapped is decided by considering the edge communication weights $comm_{ij}$ of the core graph.

The first step involves sorting the edges of the core graph in descending order of bandwidth requirement $comm_{ij}$. Let e_{ij} be the edge with the maximum bandwidth requirement. The mapping process starts with this edge. Let c_i and c_j be the cores attached to this edge of the core graph. For each core, the total bandwidth is calculated, and the core with the higher communication bandwidth is selected as the first core to be mapped. The other core is mapped to a node one hop away from the first node. For a network with n nodes, there exist n possible start positions where this core can be mapped. However, as a mesh-based NoC is symmetric, there exist $n/4$ unique starting points. For each of these starting points, we carry out the mapping, and the solution with the lowest APD is chosen as the final mapping. At any point during execution of the constructive heuristic, a subset of the cores C' is already mapped. Let the cores belonging to C' be mapped to the set of nodes R' . Given the nature of the algorithm, R' always forms a continuous set—for instance, for each $r_i \in R'$, there exists a neighboring $r_j \in R'$. The next core to be chosen is the core having the highest communication bandwidth to any core in C' . The possible mapping positions for this core are one hop away from R' . Let the unmapped chosen core of the corresponding edge be c_k . The core c_k is added to C' and is iteratively mapped to all nodes that are one hop away from the routers in the set R' . For each of the possible positions, the local APD of the mapped cores C' is calculated. The position with the lowest local APD is chosen as the mapping, and the selected node is added to R' . The procedure is iterated over until all of the cores have been mapped. In the runtime of the algorithm, a scenario may arise where multiple positions may have the lowest local APD. For example, let the mapping of core c_k to m positions of the set $Min_Positions = \{r_1, r_2, \dots, r_m\}$ have the same local APD. In such a scenario, a mapping considering $R' \cup \{r_m\}$ for each of the m positions of the $Min_Positions$ set is separately carried out. For this tie-breaking mapping call, in case of a tie in APD during a subsequent local APD calculation, the core is mapped to the first element of its corresponding set. At the end, the position with the lowest APD is chosen, and the core is mapped to the corresponding position. The algorithm then iteratively goes on to map the other unmapped cores. The proposed algorithm is described next.

ALGORITHM 1: Constructive Heuristic

Constructive Mapping

Input: Core Graph G , Topology Delay Graph T

Output: Optimal Mapping of G onto T

START

Sort edges $e_{ij} \in E$ of G in order of communication cost $comm_{ij}$

Select edge $e_{ij} \in E$ connecting cores c_i and c_j , with highest communication cost $comm_{ij}$

$Comm.i = \text{Total Bandwidth Requirement of } c_i$

$Comm.j = \text{Total Bandwidth Requirement of } c_j$

If $Comm.i$ greater than $Comm.j$

Core = c_i

Else

Core = c_j

End if

For all $r_k \in R$ such that r_k lies in the upper left quadrant of the mesh

Mark all cores $c_i \in C$ as unmarked

$Best_APD = \infty$

ALGORITHM 1: Continued

```

    Optimal_Mapping = phi
    Mapped_Cores = phi
    Add Core to Mapped_Cores
    Mapping[First_Core_Position] = Core
    Express_Mapping = Construct_Map (G, T, uk, Mapped_Cores, Mapping, False)
    APD = Get_APD (Express_Mapping)
    If APD less than Best_APD
        Optimal_Mapping = Express_Mapping
        Best_APD = APD
    End if
End for
Output Optimal_Mapping
END
Subroutine: Construct_Map
Input: Core Graph G, Topology Delay Graph T, First_Core_Position,
        Mapped_Cores, Mapping, Choice_Flag
Output: Best Mapping of G onto T for First_Core_Position
START
    While there exists  $c_k \in C$  such that  $c_k$  is unmapped (not a part of Mapped_Cores)
        Select edge  $e_{ij} \in E$  connecting cores  $c_i$  and  $c_j$ , with highest communication cost  $comm_{ij}$ 
        such that  $c_i$  is mapped and  $c_j$  is unmapped
        Add  $c_j$  to Mapped_Cores
        Mapping_Positions = Set of unmapped nodes at one hop from mapped nodes
        Evaluate_Local_APD(Mapping_Positions)
        Min_Positions = Positions with minimum local APD
        If Choice_Flag is True (handles tie breaking calls)
            Min_Positions = Min_Positions[1] // Discard the other positions
        End if
        Best_APD =  $\infty$ 
        For all  $r_m \in \text{Min\_Positions}$ 
            If Cardinality of Min_Positions not equal to 1
                Temp_Mapped_Cores = Mapped_Cores
                Temp_Mapping = Mapping
                Temp_Mapping[ $r_m$ ] =  $c_j$ 
                Temp_Mapping = Construct_Map (G, P,  $r_m$ , Temp_Mapped_Cores,
                    Temp_Mapping, True)
                APD = Get_APD (Temp_Mapping)
                If APD less than Best_APD
                    Best_Mapping [ $r_m$ ] =  $c_j$ 
                    Best_APD = APD
                End if
            Else
                Best_Mapping [ $r_m$ ] =  $c_j$ 
            End if
        End for
    End while
    Output Best_Mapping
END

```

The complexity of the proposed constructive heuristic is governed by the number of entries in the *Mapping_Positions* set—that is, the number of positions that have to be considered for mapping each core. Initially, when the number of already mapped cores is small, the set will have a small number of positions. The cardinality will be maximum when around half of the cores have been mapped. Subsequently, it will again start to reduce. The worst-case scenario for the algorithm occurs when each core communicates equally with any other core. It can be proved that for an n core application, the proposed heuristic has a complexity of $O(n^5 \log n)$. Details of the proof have been omitted for the sake of brevity.

4.3. KL Partitioning–Induced Constructive Heuristic for Express Channel–Based CMesh NoCs

For an express channel–based mesh network with concentration, multiple cores are mapped to the same router. This causes a single router delay between the cores sharing a router and multirouter delay among cores mapped to distinct routers. The nature of this delay makes it necessary to map a cluster of cores with a greater degree of intercommunication onto a single router. To achieve this clustering, the KL portioning algorithm [Kernighan and Lin 1970] is applied to the core graph. The KL partitioning strategy was originally developed to partition modules between hardware and software in VLSI physical design. It creates two partitions such that highly communicating and connected modules are kept in one partition. As a preconditioning to the mapping step, we propose using a modified KL bipartitioning strategy to identify the cores that can be assigned to a single partition by analyzing their bandwidth requirements. This bipartitioning is applied recursively until the number of cores in each partition is equal to the node concentration. In most practical designs, the concentration is chosen to be four. The modified KL partitioning heuristic used is elaborated next.

ALGORITHM 2: Modified KL Partitioning Heuristic

Kernighan Lin Heuristic

Input: Core Graph G , Set of Cores $C = \{c_1, c_2, \dots, c_j, \dots, c_{M*N}\}$

Output: Cluster Graph G' , Set of Core Clusters $Cl = \{cl_1, cl_2, \dots, cl_j, \dots, cl_{M*N/4}\}$

START

$C = KL_Partitioning(G, C, M*N/2)$ // Cores Reordered

For all $cl_j \in Cl$

$cl_j = \{c_{(j-1)*4+1}, c_{(j-1)*4+2}, c_{(j-1)*4+3}, c_{(j-1)*4+4}\}$

End for

Construct Cluster Communication Graph G' using G and Core Clusters

Output G' , Core Clusters

END

Subroutine: KL Partitioning

Input: Core Graph G , Set of Cores $C = \{c_1, c_2, \dots, c_j, \dots, c_{2*Partition_Size}\}$ partitioned into two non empty equal sized partitions $p1$ and $p2$ such that $p1 \cup p2 = C$ and $p1 \cap p2 = NULL$, Partition_Size

Output: Reordered Set of Cores $C = \{c_1, c_2, \dots, c_j, \dots, c_{2*Partition_Size}\}$

START

do

$current_partition = best_partition = (p1, p2)$

Unlock all Cores

While there exists $c_k \in C$ such that c_k is unlocked

$current_partition, swap_pair = Select_Swap_Pair(G, current_partition)$

Lock Cores $\in swap_pair$

ALGORITHM 2: Continued

```

    best_partition = Get_Best_Partition(best_partition, current_partition)
End while
If Comm_between_partitions(best_partition) > Comm_between_partitions(p1, p2)
    p1 = KL_Partitioning(G, p1, Partition_Size / 2)
    p2 = KL_Partitioning(G, p2, Partition_Size / 2)
    C = (p1, p2)
    Output C
Else
    (p1, p2) = best_partition
    Unlock all Cores
End if
End do
END
Subroutine: Select_Swap_Pair
Input: Core Graph G, Current_Partition partitioned into two nonempty equal-sized partitions p1 and p2 such that  $p1 \cup p2 = C$  and  $p1 \cap p2 = NULL$ 
Output: swapped_partition, swap_pair
START
    cost_log = NULL
    For each unlocked (ck ∈ p1 and cj ∈ p2)
        temporary_partition = Swap (ck and cj between p1 and p2)
        Append Comm_between_partitions(temporary_partition) to cost_log
    End for
    swapped_partition = Partition made by swap with lowest cost
    swap_pair = Swap that causes lowest cost
    Output swapped_partition, swap_pair
END

```

Following the clustering by the KL approach, the *cluster graph* $G'(Cl, E')$ is constructed. For a network with concentration $conc_{node}$, each node $cl_i \in Cl$ of the graph represents a cluster of cores $\{c_{i1}, c_{i2}, \dots, c_{i conc_{node}} \in cl_i$ belonging to a single KL partition. The directed edges $e'_{ij} \in E'$ represent the communication between the clusters cl_i and cl_j . Each edge e'_{ij} has a weight denoted by cl_comm_{ij} , which represents the bandwidth requirement of the communication from cluster c_i to c_j . The constructive heuristic reported in Section 4.2 is subsequently applied to map the *cluster graph* $G'(Cl, E')$ to the network topology. Subsequently, each core belonging to the cluster is mapped to the router to which the cluster has been mapped.

4.4. Integrated Mapping and Synthesis for Express Channel-Based Mesh NoCs

From the point of view of a system designer, the high router radix required by the mentioned express channel topologies [Kim et al. 2007; Grot et al. 2009] may make them practically infeasible. Based on the constructive heuristic described in Section 4.2, we now propose an integrated mapping and synthesis technique. The idea behind the approach is to simultaneously perform mapping and synthesis of the topology. For each router, the radix is limited by the upper bound set by the system designer. The constructive nature of the algorithm gives priority for highly communicating cores to have express channels between them. The proposed approach is similar to the constructive heuristic described in Section 4.2, with some differences that are elaborated next.

The traditional mesh topology is taken as the starting point. As the mapping proceeds, express channels are added based on the communication requirements of the application. As mentioned in Section 4.2, at any point during execution of the constructive heuristic, a subset of the cores C' is already mapped. The cores belonging to C' are mapped to the set of routers R' . Given the nature of the algorithm, R' always forms a continuous set—that is, for each $r_i \in R'$, there exists a neighboring $r_j \in R'$. For the next core to be mapped, the possible mapping positions are one hop away from R' . For each of these positions, the local APD calculation depends on the following factors:

- The express channels that have already been synthesized until that point of execution
- Whether the router radix at each router permits the addition of express channels between the position being considered and another core (with which the core being mapped communicates)
- If an express channel cannot be added or does not exist, then hop-by-hop forwarding must be considered.

The position with the lowest local APD is chosen as the position of mapping, and the selected node is added to R' . Subsequently, the *Make_Connections* subroutine is invoked, and the express channels are added based on the router radix specified and the communication requirement between the mapped core and the previously mapped cores. The *Make_Connections* subroutine updates the *Router_Interconnect_Matrix* and the *Router_Radix_Array*, which respectively keep track of the network interconnections and the individual router radices. The rest of the procedure is similar in all respects to the heuristic proposed in Section 4.2. For greater clarity, the entire procedure is elaborated next.

ALGORITHM 3: Integrated Mapping and Synthesis

Constructive Mapping and Synthesis

Input: Core Graph G , Topology Delay Graph T , Router Radix Limit

Output: Optimal Mapping of G onto T

START

Sort edges $e_{ij} \in E$ of G in order of communication cost $comm_{ij}$

Select edge $e_{ij} \in E$ connecting cores c_i and c_j , with highest communication cost $comm_{ij}$

$Comm.i$ = Total Bandwidth Requirement of c_i

$Comm.j$ = Total Bandwidth Requirement of c_j

If $Comm.i$ greater than $Comm.j$

Core = c_i

Else

Core = c_j

End if

For all $r_k \in R$ such that r_k lies in the upper left quadrant of the mesh

Mark all cores $c_i \in C$ as unmarked

$Best_APD = \infty$

Optimal Mapping = ϕ

Mapped_Cores = ϕ

Add Core to Mapped_Cores

Mapping[First_Core_Position] = Core

Router_Interconnect_Matrix = Initialize to Regular Mesh Configuration // No Express Channels

Router_Radix_Array = Initialize Radix as in a Regular Mesh

Express_Mapping = Construct_MapSynth($G, T, u_k, \text{Mapped_Cores}, \text{Mapping}, \text{False}, \text{Router_Interconnect_Matrix}, \text{Router_Radix_Array}, \text{Router_Radix_Limit}$)

ALGORITHM 3: Continued

```

    APD = Get_APD (Express_Mapping, Router_Interconnect_Matrix,
                  Router_Radix_Array)
    If APD less than Best_APD
        Optimal_Mapping = Express_Mapping
        Best_APD = APD
        Best_Router_Interconnect_Matrix = Router_Interconnect_Matrix
    End if

```

```

End for

```

```

Synthesize topology as per the Best_Router_Interconnect_Matrix

```

```

Output Optimal_Mapping

```

```

END

```

Subroutine: Construct_MapSynth

```

Input: Core Graph G, Topology Delay Graph T, First_Core_Position,
       Mapped_Cores, Mapping, Choice_Flag, Router_Interconnect_Matrix,
       Router_Radix_Array, Router_Radix_Limit

```

```

Output: Best_Mapping of G onto T for First_Core_Position

```

```

START

```

```

While there exists  $c_k \in C$  such that  $c_k$  is unmapped (not a part of Mapped_Cores)
    Select edge  $e_{ij} \in E$  connecting cores  $c_i$  and  $c_j$ , with highest communication cost  $comm_{ij}$ 
    such that  $c_i$  is mapped and  $c_j$  is unmapped
    Add  $c_j$  to Mapped_Cores
    Mapping_Positions = Set of unmapped nodes at one hop from mapped nodes
    Evaluate_Local_APD (Mapping_Positions, Router_Interconnect_Matrix,
                      Router_Radix_Array, Router_Radix_Limit)
    Min_Positions = Positions with minimum local APD
    If Choice_Flag is true (handles tie breaking calls)
        Min_Positions = Min_Positions[1] // Discard the other positions
    End if
    Best_APD =  $\infty$ 
    For all  $r_m \in Min\_Positions$ 
        If cardinality of Min_Positions not equal to 1
            Temp_Mapped_Cores = Mapped_Cores
            Temp_Mapping = Mapping
            Temp_Mapping[ $r_m$ ] =  $c_j$ 
            Temp_Router_Interconnect_Matrix = Router_Interconnect_Matrix
            Temp_Router_Radix_Array = Router_Radix_Array
            Temp_Mapping = Construct_Map (G, P,  $r_m$ , Temp_Mapped_Cores,
                                         Temp_Mapping, True, Temp_Router_Interconnect_Matrix,
                                         Temp_Router_Radix_Array, Router_Radix_Limit)
            APD = Get_APD (Temp_Mapping, Temp_Router_Interconnect_Matrix,
                          Temp_Router_Radix_Array)
            If APD less than Best_APD
                Best_Mapping[ $r_m$ ] =  $c_j$ 
                Best_APD = APD
            End if
        Else
            Best_Mapping[ $r_m$ ] =  $c_j$ 
        End if
    End for

```

ALGORITHM 3: Continued

Make_Connections ($G, T, \text{Best_Mapping}, r_m, c_j, \text{Router_Interconnect_Matrix},$
Router_Radix_Limit, Router_Radix_Array)

End for

End while

Output *Best_Mapping*

END

Subroutine: Make_Connections

Input: **Core Graph G , Topology Delay Graph T , Mapping, Core_Position,**
Core, Router_Interconnect_Matrix, Router_Radix_Limit
Router_Radix_Array

Output: **Best_Mapping of G onto T for First_Core_Position**

START

For all $r_p \in \text{Mapped_Positions}$

If $\text{comm}_{\text{Mapping}[r_p]\text{Core}}$ not equal to zero **AND** no express channel exists

If $\forall$ routers $r_k \in \text{Routing_Path}$, $\text{Router_Radix_Array}[r_k] < \text{Router_Radix_Limit}$

Update *Router_Interconnect_Matrix*

End if

End if

End for

END

If the upper bound for the router radix is set equal to the maximum possible router radix (for a given express channel network, assuming all express links in the same row and column), we would obtain a mapping that is identical to the one using the constructive heuristic proposed in Section 4.2. However, in this case, the synthesized topology would contain only the express channels that are essential to the application (assuming dimension order routing). Hence, we obtain a topology with an equivalent theoretical delay as the topology that contains all of the express channels.

5. EXPERIMENTAL RESULTS

In this section, we first present the simulation methodologies and setup used for the experimentation. Subsequently, we present the simulation results for both the application mapping and the integrated mapping and synthesis for express channel networks. The simulations have been performed on several benchmark applications available in the literature [Leighton 1992]. To demonstrate the efficiency of the proposed heuristic on larger NoCs, synthetic benchmarks with core sizes of 64 and 128 have been generated using the TGFF tool [Dick et al. 1998]. These synthetic application core graphs have been named as G1 through G6 in the tabulated results. We organize the results as follows. First we present the results of the mapping onto the mesh-based express channel topology. This has been followed with the results of the mapping onto the CMesh-based express channel topology. Last, we present the result of the integrated mapping and synthesis technique proposed in this work.

5.1. Simulation Methodologies

To demonstrate the robustness of the proposed approaches, both static and dynamic simulations have been undertaken.

5.1.1. Static Performance Analysis. In the static simulations, the APD of the obtained mapping solution is computed using the relevant delay models presented in Section 3. The mappings are obtained using the proposed methods, and they are compared on

the basis of the computed APD. The APD calculation is performed by considering a three-stage pipelined router. Hence, the router delay T_R has been considered as three cycles and the unit link delay T_L as one cycle. We assume that the links of the network have sufficient bandwidth to accommodate the communication requirement of the application. Therefore, we can safely assume that the network does not suffer a high degree of contention. Thus, the contention delay T_C has been considered as one cycle. The runtime of the mapping methods has also been compared on a personal computer with an Intel Core-i5 processor.

5.1.2. Dynamic Performance Analysis. The dynamic simulations are performed using a modified version of the cycle-accurate open source BookSim interconnection network simulator [Jiang et al. 2013]. The BookSim simulator has been modified to perform simulation of a defined NoC topology with an application mapped onto it. The modified simulator generates custom traffic patterns based on the communication graph between the application cores and their mapped positions onto the given topology. Through these simulations, the APD, average packets injected (API or throughput), and total network power are obtained. The APD and API calculation are obtained using a three-stage pipelined router model. The network power estimation is performed using the Orion 2.0 [Kahng et al. 2009] NoC power model for a 32nm technology library [International Technology Roadmap for Semiconductors 2007].

5.2. Comparisons for Mapping onto Mesh-Based Express Channel NoCs

For the express channel-based mesh NoC, we compare our proposed mapping technique to the TRAM approach [Zhu et al. 2014] and the ILP formulation described in Section 4.1. To the best of our knowledge, TRAM is the only method that takes into account the nature of express channels. In Zhu et al. [2014], it has been shown that TRAM performs better than the Monte Carlo and simulated annealing methods. Hence, TRAM [Zhu et al. 2014] has been implemented and has been chosen as the benchmark for comparison with our proposed approach. For small core graphs (up to 16 cores), the results obtained using the proposed method has been compared to the mappings obtained by the ILP formulation. The ILP problem has been solved by using CPLEX solver [CPLEX 2013]. This comparison is used to demonstrate the quality of the solution obtained by the proposed constructive heuristic. For larger benchmarks, the ILP formulation failed to run. The static simulation comparisons of APD, runtime, and the percentage of traffic making turns for various benchmark applications can be found in Table II, and the dynamic simulation comparisons for APD, API, and network power can be found in Table III.

From the results of Table II, it can be noted that for the mesh-based express channel NoC, compared to TRAM [Zhu et al. 2014], the proposed heuristic on average results in a 8.69% reduction in APD cycles with a maximum reduction in the APD being around 27.67%. The average reduction in the percentage of packets that make turns is about 1.08%. The maximum reduction in percentage of turns is about 8.01%. This establishes the superiority of the proposed heuristic over TRAM [Zhu et al. 2014]. For the case of the application graph G4, we observe a decrease in APD with an increase in the percentage of traffic that takes a turn. This validates the notion stated in the example of Figure 3 that the mapping of communicating cores in different rows or columns may increase the percentage of traffic making turns but reduce the APD. The dynamic simulation results in Table III further support the superiority of the mappings obtained using the proposed approach.

5.3. Comparisons for Mapping onto CMesh-Based Express Channel NoCs

For the CMesh-based express channel network, to ensure fair comparison, the proposed approach has been compared to the KL partitioning-guided TRAM [Zhu et al. 2014].

Table II. Static Simulation Results for Mesh with Express Channels

Application ($n, M \times N$)	ILP		TRAM [Zhu et al. 2014]			Constructive Heuristic		
	<i>APD</i> (cycles)	<i>Runtime</i> (seconds)	<i>APD</i> (cycles)	<i>Runtime</i> (seconds)	<i>Turns</i> (%)	<i>APD</i> (cycles)	<i>Runtime</i> (seconds)	<i>Turns</i> (%)
MPEG4 (12,4×4)	9.09	1.10	9.45	0.010	1.74	9.12	0.009	0.93
VOPD (16,4×4)	*9.19	2686.87	9.41	0.010	1.74	9.13	0.024	0.42
DVOPD(32,4×8)	**	**	10.23	0.017	0.73	9.14	0.203	0.54
MWD (12,4×4)	*9.09	378.15	9.40	0.010	0	9.11	0.021	0
263_enc(12,4×4)	9.00	28.34	9.23	0.010	0.01	9.00	0.022	0
mp3_enc(16,4×4)	9.03	2237.80	9.07	0.010	0	9.03	0.014	0
263_dec(16,4×4)	9.00	2088.90	9.38	0.009	0.64	9.00	0.017	0
G1 (64,8×8)	**	**	10.66	0.337	6.43	9.97	3.99	5.13
G2 (64,8×8)	**	**	10.36	0.293	0	9.04	5.98	0
G3 (64,8×8)	**	**	10.47	0.296	0	9.08	5.53	0
G4(128,8×16)	**	**	12.38	8.201	6.82	10.7	114.74	10.73
G5(128,8×16)	**	**	13.67	4.996	33.04	12.17	100.52	24.94
G6(128,8×16)	**	**	12.83	1.840	6.57	9.28	120.18	0.945

*Out of memory.

**Did not run.

Table III. Dynamic Simulation Results for Mesh with Express Channels

Application ($n, M \times N$)	TRAM [Zhu et al. 2014]			Constructive Heuristic		
	<i>APD</i> (cycles)	<i>API</i> (per cycle)	<i>Power</i> (mW)	<i>APD</i> (cycles)	<i>API</i> (per cycle)	<i>Power</i> (mW)
MPEG4 (12,4×4)	12.50	0.0026	2.89	12.11	0.0027	2.96
VOPD (16,4×4)	12.37	0.0047	4.97	12.12	0.0047	4.96
DVOPD(32,4×8)	13.14	0.0052	23.25	12.13	0.0052	22.87
MWD (12,4×4)	12.44	0.0056	3.22	12.20	0.0057	3.21
263_enc(12,4×4)	12.34	0.0038	4.88	12.04	0.0038	2.97
mp3_enc(16,4×4)	12.09	0.0036	4.84	12.05	0.0036	4.82
263_dec(16,4×4)	12.44	0.0048	5.02	12.05	0.0048	4.94
G1 (64,8×8)	13.37	0.0056	85.75	12.70	0.0056	85.19
G2 (64,8×8)	13.45	0.0071	87.19	12.08	0.0072	85.79
G3 (64,8×8)	13.50	0.0079	87.88	12.11	0.0079	86.33
G4(128,8×16)	15.43	0.0083	484.81	13.71	0.0083	480.53
G5(128,8×16)	16.39	0.0081	487.41	14.70	0.0092	484.34
G6(128,8×16)	15.39	0.0066	480.68	12.27	0.0067	474.98

For the CMesh NoCs, as applications with lower core count have a trivial mapping, we have compared the methods using 64- and 128-core synthetic benchmarks. The static simulation comparisons of APD, runtime, and the percentage of traffic making turns for various benchmark applications can be found in Table IV, and the dynamic simulation comparisons for APD, API, and network power can be found in Table V.

From Table IV for the CMesh-based express channel NoC, it can be noted that compared to TRAM [Zhu et al. 2014], the proposed heuristic on average results in a 1.87% reduction in APD cycles, with the maximum reduction being about 3.90%. The average reduction in the percentage of packets that make turns is about 1.08%. The maximum reduction in percentage of turns is about 2.91%.

The dynamic simulation results amply demonstrate the superiority of the mapping solutions obtained using the proposed approach compared to the existing TRAM approach [Zhu et al. 2014]. The proposed constructive heuristic leads to a mapping solution that leads to a significant reduction in both APD and network energy consumption.

Table IV. Static Simulation Results for CMesh with Express Channels

Application ($n, M \times N$)	KL+TRAM [Zhu et al. 2014]			KL+Constructive Heuristic		
	<i>APD</i> (cycles)	<i>Runtime</i> (seconds)	<i>Turns</i> (%)	<i>APD</i> (cycles)	<i>Runtime</i> (seconds)	<i>Turns</i> (%)
G1 (64,4×4)	6.62	0.291	7.64	6.53	0.308	5.27
G2 (64,4×4)	5.47	0.229	0	5.43	0.274	0
G3 (64,4×4)	5.55	0.263	1.91	5.44	0.274	0
G4(128,4×8)	7.39	6.461	6.56	7.20	7.044	3.65
G5(128,4×8)	8.54	3.419	8.54	8.34	5.307	8.34
G6(128,4×8)	5.89	5.061	1.38	5.66	5.589	0

Table V. Dynamic Simulation Results for CMesh with Express Channels

Application ($n, M \times N$)	KL+TRAM [Zhu et al. 2014]			KL+Constructive Heuristic		
	<i>APD</i> (cycles)	<i>API</i> (per cycle)	<i>Power</i> (mW)	<i>APD</i> (cycles)	<i>API</i> (per cycle)	<i>Power</i> (mW)
G1 (64,4×4)	16.01	0.0358	11.16	15.77	0.0359	11.04
G2 (64,4×4)	15.94	0.0410	12.02	16.25	0.0410	11.91
G3 (64,4×4)	16.06	0.0448	12.88	15.47	0.0448	12.25
G4(128,4×8)	18.26	0.0404	42.75	17.69	0.0404	41.35
G5(128,4×8)	17.75	0.0259	34.59	17.39	0.0259	34.30
G6(128,4×8)	18.35	0.0510	46.55	18.84	0.0510	48.30

5.4. Comparisons for Integrated Mapping and Synthesis for Express Channel Networks

In this section, we present the results of the proposed integrated mapping and synthesis technique. To the best of our knowledge, this work is the first of its kind. Hence, we cannot compare it directly to TRAM [Zhu et al. 2014]. Therefore, we have done an indirect comparison with the TRAM approach. Since TRAM is not constructive in nature, no provision can be made for simultaneous synthesis of a custom topology. Thus, we generate a mapping solution using TRAM and remove the express channels that would not be used, assuming perfect dimension order routing between the cores. We take this to be the custom topology synthesized using TRAM. To ensure fair comparison, we generate a solution using our approach by setting the router radix limit to the maximum possible radix for a given topology. For example, in a 4×4 express channel mesh network, each router would have a radix of 7 (one port to the local core and six ports to the other routers in the same row and column). We then compare this synthesized topology with the one generated using TRAM.

In Table VI, we compare the two synthesized topologies on the basis of the maximum router radix required, the average router radix required, and the number of synthesized express links required. We also compare these topologies with the mesh-based express channel NoC with all-to-all express links (within the same row and column). This comparison has been done to illustrate the benefit of using a semicustom topology compared to general-purpose topologies [Kim et al. 2007]. For the semicustom topologies, the static simulation comparisons of APD, runtime, and the percentage of traffic making turns for various benchmark applications can be found in Table VII, and the dynamic simulation comparisons for APD, API, and network power can be found in Table VIII.

Finally, we present some simulation results for the integrated mapping and synthesis technique on several benchmarks. Based on the benchmark, the APD and number of express channels have been compared for different router radices in Table IX. The smallest router radix considered is 5 (assuming one port to the local core and four ports to the adjacent routers). The largest router radix considered depends on the

Table VI. Link and Router Radix for Semicustom Synthesized Express Channel NoCs

Application ($n, M \times N$)	Mesh-Based Express Channel NoC [Kim et al. 2007]		Custom NoC Using Modified TRAM [Zhu et al. 2014]			Custom NoC Using Integrated Mapping and Synthesis (Radix Limit = Max Radix)		
	Max. Router Radix	Express Channels (#)	Max. Router Radix	Average Router Radix	Express Channels (#)	Max. Router Radix	Average Router Radix	Express Channels (#)
MPEG4 (12,3×4)	6	26	6	4.43	7	6	4.66	10
VOPD (16,4×4)	7	48	6	4.63	10	6	4.50	8
DVOPD(32,4×8)	11	216	9	5.91	53	7	4.84	19
MWD (12,3×4)	6	26	7	4.63	10	5	4.00	2
263_enc(12,3×4)	6	26	6	4.38	6	6	4.16	4
263_dec(16,4×4)	7	48	6	4.63	10	6	4.13	2
G1 (64,8×8)	15	672	10	6.30	115	11	6.00	96
G2 (64,8×8)	15	672	8	5.56	68	6	4.59	6
G3 (64,8×8)	15	672	8	5.75	80	6	4.68	12
G4(128,8×16)	23	2,352	10	6.66	261	10	6.04	182
G5(128,8×16)	23	2,352	14	7.37	351	14	7.11	318
G6(128,8×16)	23	2,352	8	6.09	188	7	4.98	46

Table VII. Static Simulation Results for Semicustom Express Channel NoCs

Application ($n, M \times N$)	Custom NoC Using Modified TRAM [Zhu et al. 2014]			Custom NoC Using Integrated Mapping and Synthesis (Radix Limit = Max Radix)		
	APD (cycles)	Runtime (seconds)	Turns (%)	APD (cycles)	Runtime (seconds)	Turns (%)
MPEG4 (12,4×4)	9.45	0.011	1.74	9.12	0.009	0.930
VOPD (16,4×4)	9.41	0.010	1.74	9.13	0.024	0.420
DVOPD(32,4×8)	10.23	0.019	0.73	9.14	0.205	0.540
MWD (12,4×4)	9.40	0.010	0	9.00	0.021	0
263_enc(12,4×4)	9.23	0.010	0.01	9.00	0.022	0
263_dec(16,4×4)	9.38	0.009	0.64	9.00	0.017	0
G1 (64,8×8)	10.66	0.337	6.43	9.97	3.990	5.130
G2 (64,8×8)	10.36	0.293	0	9.04	5.980	0
G3 (64,8×8)	10.47	0.296	0	9.08	5.570	0
G4(128,8×16)	12.38	8.312	6.82	10.70	114.740	10.730
G5(128,8×16)	13.67	4.996	33.04	12.17	100.520	24.940
G6(128,8×16)	12.83	1.960	6.57	9.28	120.560	0.945

benchmark being used. The results of the dynamic simulations for the different custom NoCs synthesized can be found in Table X.

From Table VI, for the semicustom synthesized NoCs, it can be noted that compared to the modified TRAM [Zhu et al. 2014], the proposed integrated mapping and synthesis technique on average results in a 45.21% reduction in the number of express channels required, with the maximum reduction being about 91.18%. The average reduction in the average router radix is about 9.75%. The maximum reduction in the average router radix is about 18.61%.

From Table VII, it can be observed that the APD and the percentage of packets that make turns are identical to the results reported in Table II. This implies that for a given application and mapping technique, the semicustom synthesized NoCs have

Table VIII. Dynamic Simulation Results for Semicustom Express Channel NoCs

Application ($n, M \times N$)	Custom NoC Using Modified TRAM [Zhu et al. 2014]			Custom NoC Using Integrated Mapping and Synthesis (Radix Limit = Max Radix)		
	<i>APD</i> (cycles)	<i>API</i> (per cycle)	<i>Power</i> (mW)	<i>APD</i> (cycles)	<i>API</i> (per cycle)	<i>Power</i> (mW)
MPEG4 (12,4×4)	12.50	0.0026	2.00	12.13	0.0027	2.23
VOPD (16,4×4)	12.37	0.0047	2.96	12.12	0.0047	2.76
DVOPD(32,4×8)	13.39	0.0052	8.92	12.13	0.0052	6.12
MWD (12,4×4)	12.44	0.0056	2.41	12.20	0.0057	2.03
263_enc(12,4×4)	12.34	0.0038	2.60	12.04	0.0038	1.90
263_dec(16,4×4)	12.44	0.0048	2.98	12.05	0.0048	2.48
G1 (64,8×8)	14.12	0.0056	20.88	12.70	0.0056	18.95
G2 (64,8×8)	13.45	0.0071	17.30	12.08	0.0072	12.03
G3 (64,8×8)	13.50	0.0079	18.71	12.11	0.0079	12.73
G4(128,8×16)	17.87	0.0083	54.93	13.71	0.0083	41.76
G5(128,8×16)	19.00	0.0081	67.85	14.70	0.0092	62.43
G6(128,8×16)	16.77	0.0066	44.66	12.27	0.0067	26.63

Table IX. Static Simulation Results for Semicustom Synthesized Express Channel NoCs Using the Integrated Mapping and Synthesis Technique

Application ($n, M \times N$)	Configuration 1			Configuration 2			Configuration 3		
	<i>APD</i> (cycles)	<i>Max Router Radix</i>	<i>Express Channels</i> (#)	<i>APD</i> (cycles)	<i>Max Router Radix</i>	<i>Express Channels</i> (#)	<i>APD</i> (cycles)	<i>Max Router Radix</i>	<i>Express Channels</i> (#)
MPEG4(12,3×4)	9.160	5	5	—	—	—	—	—	—
VOPD(16,4×4)	9.148	5	8	—	—	—	—	—	—
DVOPD(32,4×8)	9.148	6	18	9.285	5	12	—	—	—
MWD(12,3×4)	—	—	—	—	—	—	—	—	—
263_enc(12,3×4)	9.002	5	3	—	—	—	—	—	—
263_dec(16,4×4)	9.023	5	3	—	—	—	—	—	—
G1 (64,8×8)	10.138	8	104	11.034	6	61	12.050	5	26
G2 (64,8×8)	9.125	5	4	—	—	—	—	—	—
G3 (64,8×8)	9.240	5	6	—	—	—	—	—	—
G4(128,8×16)	10.801	8	179	12.360	6	121	14.411	5	36
G5(128,8×16)	12.269	12	296	13.691	8	257	16.221	6	138
G6(128,8×16)	9.361	6	49	9.930	5	14	—	—	—

an identical mapping and theoretical APD as that of the mesh-based express channel NoC if we assume perfect dimension order routing. When runtimes are compared, we observe a slight increase in runtime compared to those reported in Table II. The results of Table VII also indicate that the integrated mapping and synthesis technique gives a lower theoretical APD, despite requiring lower radix routers and fewer express channels compared to modified TRAM.

The dynamic simulation results from Table VIII sufficiently demonstrate the superiority of the integrated mapping and synthesis approach compared to the modified TRAM approach [Zhu et al. 2014]. We observe that the reduction of router radix and number of express channels (as shown in Table VII) lead to a significant reduction in network power consumption. Compared to the NoCs synthesized using modified TRAM, on average we obtain a substantial power reduction of 19.18% for the NoCs synthesized using the integrated approach. The maximum reduction in power consumption

Table X. Dynamic Simulation Results for Semicustom Synthesized Express Channel NoCs Using the Integrated Mapping and Synthesis Technique

Application ($n, M \times N$)	Configuration 1			Configuration 2			Configuration 3		
	APD (cycles)	API (per cycle)	Power (mW)	APD (cycles)	API (per cycle)	Power (mW)	APD (cycles)	API (per cycle)	Power (mW)
MPEG4(12,3×4)	12.11	0.0034	1.85	—	—	—	—	—	—
VOPD(16,4×4)	12.12	0.0047	2.83	—	—	—	—	—	—
DVOPD(32,4×8)	12.10	0.0052	6.11	12.12	0.0052	5.74	—	—	—
MWD(12,3×4)	—	—	—	—	—	—	—	—	—
263_enc(12,3×4)	12.04	0.0051	1.89	—	—	—	—	—	—
263_dec(16,4×4)	12.06	0.0049	2.54	—	—	—	—	—	—
G1 (64,8×8)	13.04	0.0056	20.09	12.72	0.0056	16.31	13.76	0.0056	13.33
G2 (64,8×8)	12.07	0.0072	11.99	—	—	—	—	—	—
G3 (64,8×8)	12.13	0.0079	12.57	—	—	—	—	—	—
G4(128,8×16)	14.16	0.0083	41.64	14.35	0.0083	37.39	16.18	0.0083	31.11
G5(128,8×16)	14.90	0.0093	60.45	15.18	0.0093	56.19	16.08	0.0093	44.43
G6(128,8×16)	12.33	0.0067	27.20	12.62	0.0067	24.76	—	—	—

Fig. 4. A 16-core VOPD application NoC with all-to-all express channels (within the same row or column as the source router) (a), a synthesized NoC with router radix limit = 6 (b), and a synthesized NoC with router radix limit = 5 (c). Black blocks indicate the mapped cores.

was found to be 40.37%. From Table VIII, we can observe that the power reduction is especially significant for larger NoCs.

In Section 5.2, the dynamic simulation results for the mesh-based express channel NoCs with all-to-all links (within the same row or column as the source router) for several benchmarks was described in Table III. Comparing the results from Tables III and VIII, we observe that the semicustom express channel NoCs synthesized using the integrated mapping and synthesis technique show a large reduction in power consumption with a negligible increase in APD. This substantiates the results of Table VII, which indicated no theoretical increase in APD. Comparing the results, we find that on average a 65.55% reduction in network power consumption can be observed. The maximum reduction in power is noted to be 94.39%. This power reduction can be attributed to significant reduction in the router radix and number of express channels in the synthesized semicustom express channel NoCs.

The static and dynamic results of Tables IX and X clearly indicate the slight degradation in network performance when the router radix is reduced. However, the degradation in performance is minimal and can be observed that reducing router radix achieves significant reduction in network power.

In Figure 4, the synthesized NoCs using the integrated mapping and synthesis technique for the 16-core VOPD application have been illustrated for different router

Fig. 5. Change in APD and network power with router radix for the 128-core G5 application.

radices. The graph of change in APD and network power with respect to router radix for the 128-core G5 application (generated using the TGFF tool [Dick et al. 1998]) is illustrated in Figure 5.

6. CONCLUSION AND FUTURE WORK

Express channel-based NoCs have emerged as a viable alternative to solve the inherent deficiencies of a mesh-based NoC. In this article, an efficient constructive heuristic has been proposed for both the application mapping problem and the semicustom NoC synthesis problem for express channel-based networks. The proposed method outperforms the previously proposed TRAM approach [Zhu et al. 2014]. The constructive heuristic effectively maps highly communicating cores close to each other, thus providing a mapping solution with significantly lower APD and minimizing the number of turns. The integrated mapping and synthesis technique provides for simultaneous mapping and link synthesis based on the router radix restrictions set by the designer. The constructive nature of the approach favors link addition between highly communicating cores when the radix is restricted. This minimizes degradation in performance due to system design constraints. Significant power savings have also been obtained using the proposed approach. This work can be extended to other topologies such as 3D NoCs and topologies using application-specific routing techniques. The addition of network congestion models in the cost function can also be done to make the proposed technique congestion aware.

REFERENCES

- D. Atienza, F. Angiolini, S. Murali, A. Pullini, L. Benini, and D. G. Micheli. 2008. Network-on-chip design and synthesis outlook. *Integration—The VLSI Journal* 41, 2.
- J. Balfour and W. J. Dally. 2006. Design tradeoffs for tiled CMP on-chip networks. In *Proceedings of the 20th Annual International Conference on Supercomputing*. 187–198.
- L. Benini. 2006. Application specific NoC design. In *Proceedings of the Design, Automation, and Test in Europe Conference and Exhibition (DATE'06)*. 1–5.
- C. Celic and C. Bazlamacci. 2014. Evaluation of energy and buffer aware application mapping for networks on chip. *Microprocessors and Microsystems* 38, 4.

- CPLEX. 2013. IBM ILOG CPLEX. Retrieved October 14, 2015, from <http://www.ibm.com/software/in/integration/optimization/cplex>.
- W. J. Dally and B. Towles. 2001. Route packets, not wires: On-chip interconnection networks. In *Proceedings of the Design, Automation, and Test in Europe Conference and Exhibition (DATE'01)*. 684–689.
- R. P. Dick, D. L. Rhodes, and W. Wolf. 1998. TGFF: Task graphs for free. In *Proceedings of the 6th International Workshop on Hardware/Software Codesign*. 97–101.
- A. Faruque, M. Abdullah, R. Krist, and J. Henkel. 2008. ADAM: Run-time agent-based distributed application mapping for on-chip communication. In *Proceedings of the Design, Automation, and Test in Europe Conference and Exhibition (DATE'08)*. 760–765.
- P. Ghosh, A. Sen, and A. Hall. 2009. Energy efficient application mapping to NoC processing elements operating at multiple voltage levels. In *Proceedings of the IEEE International Symposium on Network-on-Chip (NoCS'09)*. 80–85.
- B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu. 2009. Express cube topologies for on-chip interconnects. In *Proceedings of the International Symposium on High Performance Computer Architecture*. 163–174.
- J. Hu and R. Marculescu. 2003. Energy-aware mapping for tile-based NoC architectures under performance constraints. In *Proceedings of the 2003 Asia and South Pacific Design Automation Conference (ASP-DAC'03)*. 233–239.
- Intel Sandy Bridge Architecture Document. 2011. <http://www.intel.com/content/dam/doc/manual/64-ia-32-architectures-optimization-manual.pdf>.
- International Technology Roadmap for Semiconductors. 2007. <http://www.itrs.net/ITRS%201999-2014%20Mtgs,%20Presentations%20&%20Links/2007ITRS/Home2007.htm>.
- N. Jiang, J. Balfour, D. Becker, B. Towles, W. J. Dally, G. Michelogiannakis, and J. Kim. 2013. A detailed and flexible cycle-accurate network-on-chip simulator. In *Proceedings of the 2013 IEEE International Symposium on Performance Analysis of Systems and Software*. 86–96.
- A. B. Kahng, B. Li, L. Peh, and K. Samadi. 2009. Orion 2.0: A fast and accurate NoC power and area model for early stage design space exploration. In *Proceedings of the Design, Automation, and Test in Europe Conference and Exhibition (DATE'09)*. 423–428.
- B. Kernighan and S. Lin. 1970. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal* 49, 2, 291–307.
- J. Kim, J. Balfour, and W. J. Dally. 2007. Flattened butterfly topology for on-chip networks. *Computer Architecture Letters* 6, 2, 37–40.
- H. Kuhn. 2005. The Hungarian method for the assignment problem. *Naval Research Logistics* 2, 1–2, 83–97.
- A. Kumar, L. S. Peh, P. Kundu, and N. K. Jha. 2007. Express virtual channels: Towards the ideal interconnection fabric. In *Proceedings of the IEEE International Symposium on Computer Architecture*. 150–161.
- F. T. Leighton. 1992. *Introduction to Parallel Algorithms and Architectures: Arrays–Trees–Hypercubes*. Morgan Kaufmann, San Mateo, CA.
- R. Marculescu and P. Bogdan. 2009. The chip is the network: Toward a science of network-on-chip design. *Foundations and Trends in Electronic Design Automation* 2, 4, 371–461.
- F. Moein-Darbari, A. Khademzade, and G. Gharooni-Fard. 2009. CGMAP: A new approach to network-on-chip mapping problem. *IEICE Electron Express* 6, 1, 27–34.
- S. Murali and G. D. Micheli. 2004a. Bandwidth constrained mapping of cores onto NoC architectures. In *Proceedings of the Design, Automation, and Test in Europe Conference and Exhibition (DATE'04)*. 896–901.
- S. Murali and G. D. Micheli. 2004b. SUNMAP: A tool for automatic topology selection and generation for NoCs. In *Proceedings of the Design, Automation, and Test in Europe Conference and Exhibition (DATE'04)*. 914–919.
- U. Y. Ogras and R. Marculescu. 2006. It's a small world after all: NoC performance optimization via long range link insertion. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 14, 7, 693–705.
- O. Ozturk, M. Kandemir, and S. W. Son. 2007. An ILP based approach to reducing energy consumption in NoC based CMPs. In *Proceedings of the IEEE International Symposium on Low Power Electronics and Design (ISLPED'07)*. 411–414.
- M. Palesi, R. Holmsmark, S. Kumar, and V. Catania. 2009. Application specific routing algorithms for networks on chip. *IEEE Transactions on Parallel and Distributed Systems* 20, 3, 316–330.
- P. P. Pande, C. Grecu, A. Ivanov, and R. Saleh. 2003. High-throughput switch-based interconnect for future SoCs. In *Proceedings of the IEEE International Workshop on System-on-Chip for Real Time Applications*. 304–310.
- D. Pham, S. Asano, M. Bolliger, M. N. Day, H. P. Hofstee, C. Johns, J. Kahle, A. Kameyama, Y. Masubuchi, M. Riley, D. Shippy, D. Stasiak, M. Suzuoki, M. Wang, J. Warnock, S. Weitzel, D. Wendel, T. Yamazaki,

- and K. Yazawa. 2006. Overview of the architecture, circuit design, and physical implementation of a first-generation cell processor. *IEEE Journal of Solid-State Circuits* 41, 1, 179–196.
- R. Pop and S. Kumar. 2004. *A Survey of Techniques for Mapping and Scheduling Applications to Network on Chip Systems*. Research Report 04:4. School of Engineering, Jönköping University, Jönköping, Sweden.
- Z. Qian, P. Bogdan, G. Wei, C. Y. Tsui, and R. Marculescu. 2012. A traffic aware adaptive routing algorithm on a highly reconfigurable network-on-chip architecture. In *Proceedings of the 11th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*.
- P. K. Sahu and S. Chattopadhyay. 2013. A survey on application mapping strategies for network-on-chip design. *Journal of Systems Architecture* 59, 1, 60–76.
- P. Sahu, T. Shah, K. Manna, and S. Chattopadhyay. 2014. Application mapping onto mesh based network-on-chip using discrete particle swarm optimization. *IEEE Transactions on Very Large Scale Integrated (VLSI) Systems* 22, 2.
- K. Srinivasan, K. S. Chatha, and G. Konjevod. 2006. Linear-programming-based techniques for synthesis of network-on-chip architectures. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 14, 4, 407–420.
- S. Tosun. 2011. Cluster-based application mapping for network-on-chip. *Advances in Engineering Software* 42, 10.
- S. R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, and S. Borkar. 2008. An 80-tile sub-100-W TeraFLOPS processor in 65-nm CMOS. *IEEE Journal of Solid-State Circuits* 43, 1, 29–41.
- D. Wentzlaff, P. Griffin, H. Hoffman, L. Bao, B. Edwards, C. Ramey, M. Mattina, C. Miao, J. F. Brown, and A. Agarwal. 2007. On-chip interconnection architecture of the tile processor. *IEEE Micro* 27, 5, 15–21.
- D. Zhu, L. Chen, S. Yue, and M. Pedram. 2014. Application mapping for express channel-based networks-on-chip. In *Proceedings of the Design, Automation, and Test in Europe Conference and Exhibition (DATE'14)*. 1–6.

Received April 2015; revised September 2015; accepted September 2015