Contention-aware Application Mapping for Network-on-Chip Communication Architectures

Chen-Ling Chou and Radu Marculescu
Department of Electrical and Computer Engineering
Carnegie Mellon University, USA
{chenlinc,radum}@andrew.cmu.edu

***Abstract* - In this paper, we analyze the impact of network contention on the application mapping for tile-based Network-on-Chip (NoC) architectures. Our main theoretical contribution consists of an integer linear programming (ILP) formulation of the contention-aware application mapping problem which aims at minimizing the inter-tile network contention. To solve the scalability problem caused by ILP formulation, we propose a linear programming (LP) approach followed by an mapping heuristic. Taken together, they provide near-optimal solutions while reducing the runtime significantly. Experimental results show that, compared to other existing mapping approaches based on communication energy minimization, our contention-aware mapping technique achieves a significant decrease in packet latency (and implicitly, a throughput increase) with a negligible communication energy overhead.**

I. Introduction

Due to increasing systems complexity and design productivity gap, the design of future multiprocessor systems-on-chip (MPSoCs) faces several challenges. From this perspective, Networks-on-Chip (NoCs) are a promising interconnect solution for complex chips consisting of many heterogeneous intellectual property (IP) cores. In NoC-based MPSoCs, the global wires are replaced by a network of shared links and multiple routers exchanging data packets simultaneously. Ideally, the traffic congestion in such a network should be avoided in order to maximize the system performance. In practice, however, it is difficult to completely eliminate the network contention which significantly degrades the system performance, particularly latency and throughput.

Previous work attempts to minimize the communication energy consumption [5]-[7][10]. However, the communication energy consumption is a good indicator of latency only if there is no congestion in the network. Indeed, in the absence of congestion, packets are injected/transmitted through the network as soon as they are generated and then latency can be estimated by counting the number of hops from source to destination. However, unlike previous work, we first analyze the major factors that produce network contention, and then propose a contention-aware mapping algorithm with the goal of minimizing the end-to-end packet latency. While NoCs can be designed with several choices in mind, in this paper, we limit our considerations to 2D mesh networks and wormhole *XY* routing. However, we note that our idea is applicable to other network topologies (*e.g.*, ring, torus, 3D-grid) with deterministic routing schemes.

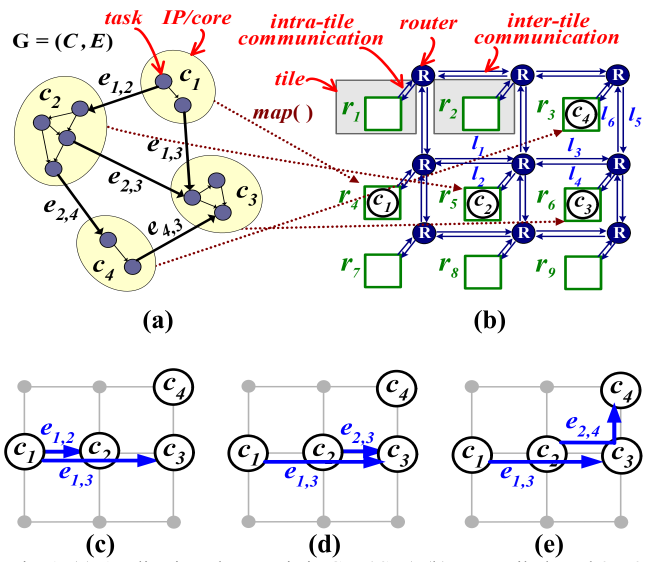

Fig. 1. (a) Application characteristic G = (*C*, *E*) (b) target tile-based 3 × 3 NoC (c) source-based contention (d) destination-based contention (e) path-based contention.

The problem of mapping a set of given IP cores to the NoC tiles is illustrated in Fig. 1. We assume that the largest application has been divided into a graph of concurrent tasks and then assigned and scheduled onto a set of available cores[1] [1]. The application graph is described by $G = (C, E)$ as shown in Fig. 1(a). Each core $c_i \in C$ represents a *cluster of tasks,* where each edge $e_{i,j} \in E$ represents the communication between the cores c_i and c_j. Note that the tasks belonging to the same core are mapped onto the same tile of the NoC. Fig. 1(b) shows the general design of a 2D tile-based NoC. Inside any tile, each core is attached to a router via the network interface which allows for exchanging packets with other cores. Each router is connected to five ports: four neighboring ports (east, south, west, and north), plus a local port connecting the core. The communication between the routers is referred to as "*inter-tile*" communication (*e.g.*, communication through links l_i where $i = 1 \sim 6$ in Fig. 1(b)), while the local communication between the core and the router is referred to as "*intra-tile*" communication. For reasons that will become clear later, our work in this

1. Here, we assume that the application partitioning process is done at core-level. By "core", we mean any IP such as general-purpose processors or digital signal processors (DSPs). In other words, the programmability and other software aspects (*e.g.*, task clustering, scheduling) for IP design are beyond the scope of this work.

paper focuses on developing a framework which is able to minimize the network contention that occurs during the *inter-tile* communication among various cores.

We clarify the network contention due to the inter-tile data communication into three types: source-based, destination-based, and path-based contention. The *source-based* contention (see Fig. 1(c)) occurs when two traffic flows originating from the same source contend for the same links. The *destination-based* contention (see Fig. 1(d)) occurs when two traffic flows which have the same destination contend for the same links. Finally, Fig. 1(e) shows the *path-based* contention when two data flows which neither come from the same source, nor go towards the same destination contend for the same links somewhere in the network.

In this paper, we first evaluate the impact of these three types of contention on average packet latency; then present an integer linear programming-based (ILP-based) contention-aware mapping technique for minimizing the network contention and communication energy consumption. We show that by mitigating a major source of contention, the end-to-end average packet latency can be significantly decreased.

The paper is organized as follows: Section II reviews the related work. The notations of application characteristics and a motivational example are described in Section III. Section IV presents the contention-aware mapping problem formulation while Section V gives the solutions via an ILP approach and LP approximation. Experimental results are shown in Section VI, while Section VII summarizes our main contribution and outlines some future work.

II. Related Work and Novel Contribution

Bender proposes a mixed integer linear programming (MILP) model for task mapping problem on heterogeneous multiprocessor systems [2], where all specified execution deadlines are met. In terms of mapping IP cores onto a mesh-based NoC platform, Hu *et al.* propose a branch and bound algorithm that minimizes the communication energy consumption with the constraints of performance handled via bandwidth reservation [5]. Murali *et al.* focus on minimizing the communication delay by exploiting the possibility of splitting traffic among multiple paths [3]. Lei *et al.* present a two-step genetic algorithm for task graph mapping with the objective of minimizing the overall execution time [9]. Ascia *et al.* apply the evolutionary computing techniques to obtain the Pareto mappings that optimize performance and power consumption [4]. Rhee *et al.* provide a MILP formulation for mapping cores onto NoC while considering the choice of core placements, switches for each core, and network interfaces for communication flows [6]. Hansson *et al.* present a scheme which incorporates mapping, routing and slot allocation such that the network size that needed to meet the application constraints is minimized [15].

Compared to previous work, our focus in this paper is on the *network contention problem*; this highly affects the latency, throughput, and communication energy consumption. We show that, by mitigating the network contention, the packet latency can be significantly reduced; this means that the network can support more traffic which directly translates into significant throughput improvements. Furthermore, our mapping solution can be used to improve the efficiency of congestion-control techniques for best-effort communication [20][21]. Last but not least, this idea can be applied to other NoC synthesis problems, such as topology selection, or path selection [8][10] and some mapping/scheduling heuristics on parallel systems [18][19] to achieve further packet latency reduction and throughput improvements.

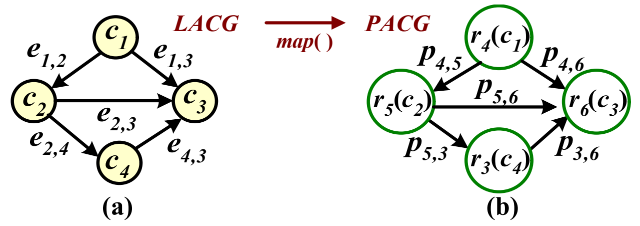

Fig. 2. Logical and physical application communication graph.

III. Preliminaries

A. Application Characteristics

To better explain the application characteristics, we need to first introduce the following definitions:

Definition 1: A *Logical Application Communication Graph* ($LACG$) = (C, E) is a weighted directed graph (see Fig. 2(a)). Each vertex $c_i \in C$ represents a core which will be allocated to one specific tile (*i.e.*, processing resource) later. Each directed edge $e_{i,j} = (c_i, c_j) \in E$ represents the communication from core c_i to c_j. The weight $w(e_{i,j})$ or w_{c_i, c_j} stands for the communication volume (bits) from core c_i to c_j within each period, while $bw(e_{i,j})$ or bw_{c_i, c_j} stands for the required bandwidth for the communication from c_i to c_j.

Definition 2: A *Physical Application Communication Graph* ($PACG$) = (R, P) is a directed graph (see Fig. 2(b)), where each vertex $r = r(c_i) \in R$ represents a tile (resource) which gets assigned a cluster of tasks, c_i, and each directed edge $p_{i,j}$ represents the routing path from tile r_i to tile r_j. We denote $L(p_{i,j})$ or $L(r_i r_j)$ the set of links of the *inter-tile* communication that make up the path $p_{i,j}$ from r_i to r_j where $|L(p_{i,j})|$ is the size of that set, *i.e.*, the number of links for making up $p_{i,j}$.

Definition 3: A mapping function *map*( ) maps the cores in the $LACG$ to the tiles in the NoC; under a given routing mechanism, this results in the $PACG$.

Fig. 2 plots the $LACG$ and the $PACG$ for the application in Fig. 1, respectively. As seen, the $PACG$ shown in Fig. 2(b) shows the mapping result (see Fig. 1(b)) of the $LACG$ under the deterministic XY routing: cores c_1, c_2, c_3, and c_4 are mapped onto tiles r_4, r_5, r_6, and r_3, respectively, and $L(p_{4,5}) = \{l_1\}$, $L(p_{4,6}) = \{l_1, l_3\}$, $L(p_{5,3}) = \{l_3, l_6\}$, $L(p_{3,6}) = \{l_5\}$, and $L(p_{5,6}) = \{l_3\}$. Note that source-based contention occurs in this case since $L(p_{4,5}) \cap L(p_{4,6}) = \{l_1\} \neq \varnothing$, while the destination-based contention occurs since $L(p_{4,6}) \cap L(p_{5,6}) = \{l_3\} \neq \varnothing$. And the path-based contention occurs since $L(p_{4,6}) \cap L(p_{5,3}) = \{l_3\} \neq \varnothing$.

B. Motivational Example

To illustrate the impact of the source-based, destination-based, and path-based network contention on the packet latency, we consider the following experiment, *i.e.*, several mapping configurations (see Fig. 3) in a 4 × 4 mesh NoC: without/with only source-based contention (cases 1 *vs.* 2), without/with only destination-based contention (cases 3 *vs.* 4), and without/with only path-based contention (cases 5 *vs.* 6). We apply the *XY* routing and wormhole switching for data transmission with 5 flits per packet. The communication rate (or the packet injection rate from the source core) of transmissions in each configuration is set to be the same. For fixed injection rates in each configuration, we run 100 different experiments and calculate the corresponding average packet latency and throughput; the latency is calculated from the time when packets are generated from sources to the time when the packets reach the destination. The results are plotted in Fig. 3 with the *x*-axis showing the total injection rate of all transmissions in that configuration and the *y*-axis showing the average packet latency at the corresponding injection rate.

As seen in Fig. 3(a), for source-based contention (*i.e.*, cases 1 and 2), the throughput is the same. This makes sense since every generated packet needs to pass through the link from the source core to its router; therefore, the system performance is basically limited by the injection rate of the source core.

For destination-based contention (*i.e.*, cases 3 and 4 in Fig. 3(b)), the system throughput has about 2% improvement. We observe that the bottleneck of these two configurations (*i.e.*, cases 3 and 4) is actually due to the *link* between the router and its corresponding destination core for which all packets contend. Obviously, such *intra-tile* contention can be mitigated via careful hardware/software codesign (*i.e.*, clustering process), but can not be solved via mapping.

As seen in Fig. 3(c), there is a dramatic throughput difference when comparing cases 5 and 6 (without/with path-based contention, respectively) as 118% throughput improvement is observed (*i.e.*, the throughput improves from 0.16 to 0.35 without path-based contention in the network). Moreover, we observe that the *frequency* of occurrence of the path-based contention is much higher compared to the source-based and destination-based contention as the system size scales up. By doing several experiments involving many runs, we observed that the ratio of path-based to source-based contention and the ratio of path-based to destination-based contention increase linearly with the network size (*i.e.*, for 4 × 4, 6 × 6, 8 × 8, and 10 × 10, the ratios are 1.2, 2.5, 4.0, 5.6, respectively). Therefore, in the remaining of this paper, we focus on minimizing the *path-based* contention since this has the most significant impact on the packet latency and can be mitigated through the mapping process.

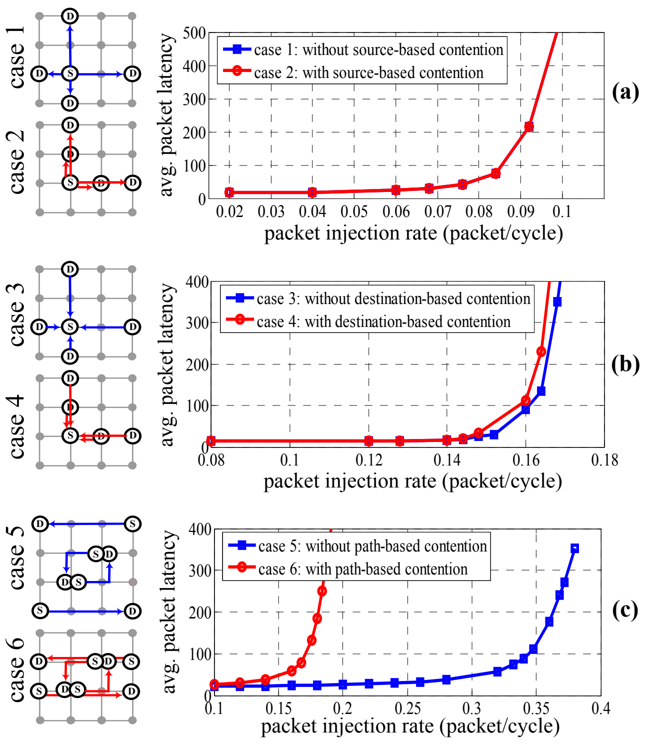

Fig. 3. The (a) source-based (b) destination-based (c) path-based contention impact on average packet latency.

IV. Contention-aware Mapping

A. Problem Formulation

Given the application characteristics and the NoC architecture, our objective is to map the IP cores onto the NoC tiles such that the sum of the weighted communication distance and path-based network contention are minimized under a given routing mechanism. Of note, minimizing the weighted communication distance directly contributes to minimizing the communication energy consumption as well. More formally:

Given the *LACG* of the application, the routing mechanism, and the NoC architecture

Find a mapping function *map*( ) from $LACG = (C, E)$ to $PACG = (R, P)$ which minimizes:

$$\min\left\{ \frac{(1-\alpha)}{\beta} \times \sum_{\forall e_{i,j} \in E} [w(e_{i,j}) \times |L(map(e_{i,j}))|] \right. \quad (1)$$

$$\left. + \frac{\alpha}{\gamma} \times |L(map(e_{i,j})) \cap L(map(e_{k,l}))| \right\} \text{ for } i \neq k \text{ and } j \neq l$$

such that:

$$\forall c_i \in C, \quad map(c_i) = r(c_i) \in R \quad (2)$$

$$\forall c_i \neq c_j \in C, \quad r(c_i) \neq r(c_j) \quad (3)$$

$$\forall \; link \; l_k, \quad \sum_{\forall (c_i, c_j) \in E} bw_{c_i, c_j} \times l_k^{map(c_i), map(c_j)} \leq B_k \quad (4)$$

$$\text{where } l_k^{map(c_i), map(c_j)} = 1 \text{ if } l_k \in L(map(c_i), map(c_j))$$

$$\text{and } B_k \text{ is the capacity for link } l_k$$

Since the communication distance and path-based contention count have different units, the normalization of these two metrics is approximated by assuming a worst-case scenario. More precisely, β is set to $(\sum_{\forall e_{i,j} \in E} w(e_{i,j})) \times (2 \times (N-1))$ for an $N \times N$ NoC platform, where the second factor, $2 \times (N-1)$, is the longest distance in the network. γ is set to the average number of path-based contentions of reasonable random mapping configurations. α is a weighting coefficient meant to balance the communication distance and the contention count. More precisely, we set α as the ratio of "the number of cores" to "the number of tiles + 1" (*i.e.*, $\alpha = |C|/(|R| + 1)$). If the number of cores is much smaller than the number of tiles (*i.e.*, α is small), in order to avoid a higher communication distance, the first term in (1) has a higher weight. Equations (2) and (3) basically mean that each core should be mapped to exactly one tile and no tile can host more than one core. Finally, equation (4) guarantees that the load of each link will not exceed its bandwidth.

V. ILP-based Approach

This section presents the ILP formulation of the contention-aware mapping problem. To solve this problem for large system sizes, we propose a LP approximation followed by an mapping heuristic as presented later in this section.

A. Parameters and Variables

The given parameters are as follows:

- $MD_{r_s r_t}$ stands for the Manhattan Distance from tile r_s to r_t.
- The NoC architecture consists of $|K|$ uni-directional segment links with IDs $\{l_1, l_2, ..., l_{|K|}\}$.
- For each link l_k, where $k = 1 \sim |K|$, $l_k^{r_s r_t}$ represents whether or not this link l_k is part of the routing path from tile r_s to tile r_t, *i.e.*, $l_k^{r_s r_t} = \begin{cases} 1, \text{ if } l_k \in L(r_s, r_t) \\ 0, \text{ otherwise} \end{cases}$.

Of note, the above parameters are known under a given NoC architecture with a fixed routing mechanism.

The variables of interest are as follows:

- $m_{c_i}^{r_s}$ shows the mapping result and can only take values in $\{0, 1\}$. More precisely, this variable is set to 1, if the core c_i is mapped onto tile r_s.
- $p_{c_i c_j}^{r_s r_t}$ shows the communication path result and can only be $\{0, 1\}$. This variable is set to 1, if the communication path is made up from tiles r_s to r_t, where c_i and c_j are mapped onto.
- $z_{l_k\_(c_i c_j c_m c_n r_s r_t r_p r_q)}$ shows the path-based contention and can only be $\{0, 1\}$. This variable is set to 1, while the cores c_i, c_j, c_m, and c_n are mapped onto tiles r_s, r_t, r_p, and r_q and at the same time, the communication path from tile r_s to tile r_t shares the link l_k with the path from tile r_p to tile r_q.

B. Objective Function

Our objective is to minimize the weighted communication distance and the path-based network contentions as well, *i.e.*,

$$\left\{ \frac{(1-\alpha)}{\beta} \times \left[\sum_{\forall (c_i, c_j) \in E} w_{c_i, c_j} \times \left(\sum_{\forall r_s, r_t \in R} MD_{r_s r_t} \times p_{c_i c_j}^{r_s r_t} \right) \right] + \frac{\alpha}{\gamma} \times \sum_{\substack{\forall l_k \\ \forall (c_i, c_j), (c_m, c_n) \in E \\ \forall r_s, r_t, r_p, r_q \in R}} z_{l_k\_(c_i c_j c_m c_n r_s r_t r_p r_q)} \right\} \quad (5)$$

C. Constraints

The following constraints are used:

- One-to-one core-to-tile mapping: Each tile cannot accept more than one core (see (6)). Each core should be mapped onto a specific tile (see (7)). Equation (8) makes sure that variables $m_{c_i}^{r_s}$ are set to be either 0 or 1.

$$\forall r_s \in R, \sum_{\forall c_i \in C} m_{c_i}^{r_s} \le 1 \quad (6)$$

$$\forall c_i \in C, \sum_{\forall r_s \in R} m_{c_i}^{r_s} = 1 \quad (7)$$

$$\forall c_i \in C, r_s \in R, \ 0 \le m_{c_i}^{r_s} \le 1 \quad (8)$$

- Communication path: Any two communicating cores that belong to two different tiles make up a path. Therefore,

$$\forall (c_i, c_j) \in E, \quad p_{c_i c_j}^{r_s r_t} = \begin{cases} 1, \text{ if } \left(m_{c_i}^{r_s} = 1\right) \text{ and } \left(m_{c_j}^{r_t} = 1\right) \\ 0, \text{ otherwise} \end{cases} \quad (9)$$

To transform (9) into an ILP formulation, we impose the following constraints:

$$m_{c_i}^{r_s} + m_{c_j}^{r_t} - 1 \le p_{c_i c_j}^{r_s r_t} \le \frac{m_{c_i}^{r_s} + m_{c_j}^{r_t}}{2} \quad (10)$$

$$0 \le p_{c_i c_j}^{r_s r_t} \le 1 \quad (11)$$

- Bandwidth constraint on each link: For each k, all possible paths through link l_k cannot exceed its bandwidth B_k.

$$\sum_{\forall r_s, r_t \in R} \sum_{\forall (c_i, c_j) \in E} bw_{c_i, c_j} \times l_k^{r_s r_t} \times p_{c_i c_j}^{r_s r_t} \le B_k \quad (12)$$

- Path-based network contention count: This type of contention occurs when two paths with different sources or different destinations contend for the same link. Therefore,

$$\begin{gathered} (\forall l_k^{r_s r_t} \in L(r_s, r_t)) \& (\forall l_k^{r_p r_q} \in L(r_p, r_q)) \\ \forall (c_i, c_j), (c_m, c_n) \in E \\ i \ne m \ \& \ j \ne n \\ \forall r_s, r_t, r_p, r_q \in R \\ z_{l_k\_(c_i c_j c_m c_n r_s r_t r_p r_q)} = 1 \text{ if } m_{c_i}^{r_s} = m_{c_j}^{r_t} = m_{c_m}^{r_p} = m_{c_n}^{r_q} = 1 \end{gathered} \quad (13)$$

To transform (13) into an ILP formulation, we impose the following constraints:

$$p_{c_i c_j}^{r_s r_t} + p_{c_m c_n}^{r_p r_q} + l_k^{r_s r_t} + l_k^{r_p r_q} - 3 \le z_{l_k\_(c_i c_j c_m c_n r_s r_t r_p r_q)} \quad (14)$$

$$0 \le z_{l_k\_(c_i c_j c_m c_n r_s r_t r_p r_q)} \le 1 \quad (15)$$

Equations (14) and (15) determine whether or not the path-based contention occurs; if so, this variable is set to be 1.

D. LP Approximation and Proposed Mapping Heuristic

Here, we relax the ILP problem (all variables must be integers known as the NP-complete problem [11]) to a (polynomial-time) linear programming (LP) problem and then use an heuristic to approximate the real values to integers for the critical variables. As such, we can deal with the mapping problem for larger NoC systems.

- LP approximation - Take out the integer constraints for variables $m_{c_i}^{r_s}$, $p_{c_i c_j}^{r_s r_t}$, $z_{l_k\_(c_i c_j c_m c_n r_s r_t r_p r_q)}$.
- Mapping heuristic - Based on the results of important variables (*i.e.*, $m_{c_i}^{r_s}$ and $p_{c_i c_j}^{r_s r_t}$) generated by LP approximation, the steps of the heuristic are summarized in Fig. 4.

Step 1: Start from the core c_i with the largest $m_{c_i}^{r_s}$ value in *LACG* and map the core c_i onto the tile r_s. If more than one core has the same $m_{c_i}^{r_s}$ value, then select the core c_i with the largest ($m_{c_i}^{r_s} + p_{c_i c_j}^{r_s r_t}$) value.

Step 2: Select an unmapped core c_j in *LACG* with the largest communication to the mapped cores.

Step 3: Find a tile r_t for core c_j minimizing the communication through the *MD* metric; if there is more than one tile, select the tile with largest $m_{c_j}^{r_t}$ value. If the value is the same, select the tile with largest ($m_{c_j}^{r_t} + p_{c_i c_j}^{r_s r_t}$) value.

Step 4: Repeat *Step 3* until all cores in the *LACG* get assigned to a tile.

Fig. 4. Main steps of the mapping heuristic.

Table 1: The accuracy and runtime comparisons between ILP method and LP approximation with mapping heuristic experimenting under different NoC sizes.

case	NoC size	# of cores	# of edges	mapping solution via (ILP)	CPU time (ILP)	mapping solution via (LP approximation)	CPU time (LP approximation)	accuracy
1	3×3	7	10	0.0625 *	2.4 mins	0.0625	0.24 + 0.42 secs	0%
2		8	12	0.1071 *	2.6 mins	0.1071	0.30 + 0.33 secs	0%
3		9	14	0.8714 *	3.41 mins	0.905	0.28 + 0.56 secs	3.86%
4	4×4	12	20	0.144 *	2 hours 15 mins	0.151	0.34 + 0.37 secs	4.86%
5		14	22	0.250 *	2 hours 40 mins	0.259	0.40 + 0.51 secs	3.47%
6		16	24	0.335 *	1 hour 48 mins	0.335	0.58 + 0.46 secs	0%
7	5×5	18	20	0.061	5 hours #	0.063	0.91 + 0.95 secs	3.28%
8		21	30	0.855	5 hours #	0.863	0.90 + 1.52 secs	0.93%
9		24	40	3.707	5 hours #	3.875	0.82 + 1.77 secs	4.5%

Table 1 compares the solution and runtime of the ILP approach against the LP approximation followed by the mapping heuristic[2]. The results come from a series of task graphs generated using the TGFF package [13] where the communication value between edges in each task graph varies between 5 to 20. To solve the ILP and LP problem in Section V, we utilize the *lp-solve* optimizer [12].

We consider nine cases in Table 1: the second column shows the size of the NoC, while the third and the fourth columns represent the number of cores and edges, respectively in each case. The fifth through the eighth columns in Table 1 report the mapping results (see Equation (1)) and their runtime comparing the ILP model against the LP approximation (*i.e.*, the LP-based method followed by the heuristic in Fig. 4). The last column gives the accuracy comparison between the heuristic value and the value of ILP approach obtained by *lp-solve*. Of note, for the cases marked with '*' (see "ILP solution" column), the solutions represent the optimal solution evaluated by exhaustive search. For cases marked with '#' (see "CPU time (ILP)" column), we use a timeout of 5 hours and accept the best solution found by then.

From Table 1, we can see that the LP approximation followed by the heuristic is quite efficient because it takes less than 3 seconds to give a nearly optimal value (about 3.32% away from the results of the ILP method, on average).

VI. Experimental Results

A. Experiments using Synthetic Applications

We first evaluate the contention impact on application mapping for a 4×4 NoC platform under two different scenarios: energy-aware mapping [5] and our contention-aware mapping. To do a fair comparison, both scenarios are formulated in ILP format with the timeout set to 2 hours (for the energy-aware mapping in ILP format, we set α to 0 and take off the constrains in Equations (13)-(15)). Several sets of synthetic applications are generated using the TGFF package [13]. The number of cores used in this experiment ranges from 12 to 16, while the number of edges are set from 15 to 50 (organized in 5 categories as shown in Table 2). For each category, we generate 10 random task graphs and the corresponding results (*i.e.*, communication energy consumption, system throughput) are measured by a C++ simulator using the bit energy model in [17] and are compared to the results of energy-aware mapping approach.

Table 2: Communication energy and throughput comparison between energy-aware in [5] and contention-aware mapping.

# of edges	15	20	30	40	50
comm. energy overhead	1%	2%	7%	11%	8%
throughput improvement	18.5%	18.2%	24.1%	21.8%	13.5%

As it can be seen in Table 2, under the contention-aware mapping, the communication energy consumption is up to 11%[3] larger compared to the energy-aware mapping solution; however, the system throughput can be improved by 19.22%, on average. That is to say, the contention-aware mapping effectively reduces the path-based contention with negligible energy loss while the reduction of the path-based contention results in great system throughput improvements.

B. Experiments using Real Applications

To evaluate the potential of our contention-aware idea for real-time examples, we apply it to several benchmarks, such as examples with high degree of parallelism (*Parallel-1* and *Parallel-2*) [14], *LU Decomposition* [14], and *MPEG4 decoder* [7]. In Table 3, the first three benchmarks are mapped onto a 3×3 NoC platform, while the last is onto a 4×4 NoC platform. The first through the fifth columns in Table 3 show, respectively, the name of the benchmark, the number of cores and edges in the *LACG*, the communication energy loss and the throughput savings of our contention-aware solution compared to the energy-aware solution [5].

As seen in Table 3, our contention-aware solution can achieve 17.4% throughput savings, on average, with the communication energy loss within 9% compared to energy-aware solution in [5].

2. The mapping heuristic is implemented using the C programming language on an Intel Pentium 4 CPU (2.6GHz with 768 MB memory).

3. We note that this is only the communication energy part. If the communication energy consumption is around 20% of the *total* energy consumption (as shown in [16]), we have only 2.2% energy loss.

Table 3: Communication energy overhead and throughput improvement of our contention-aware solution compared to the energy-aware solution [5].

benchmarks	cores	edges	comm. energy overhead	throughput improvement
Parallel-1	9	13	0%	16.9%
Parallel-2	9	15	8.8%	20.4%
LU Decomposition	9	11	6.5%	14.1%
MPEG4 Decoder	12	26	3.6%	18.2%

Fig. 5 plots the *LACG* of the *Parallel-1* benchmark (see Fig. 5(a)), the mapping results under two scenarios: energy-aware mapping [5] using ILP approach and our contention-aware mapping approach (see Fig. 5(b) and (c), respectively), and the average packet latency comparison for different injection rates in these two scenarios (see Fig. 5(d)). The existing path-based contentions are highlighted in the mapping results. As seen the energy-aware mapping result in Fig. 5(b), there are two pairs of path-based contention in the network, while no path-based contention occurs when using the contention-aware approach. We observe that for such path-based contention, the average latency goes up dramatically after the packet injection rate exceeds a critical point (*i.e.* the network gets into the congestion mode, see Fig. 5(d)). Also, when contention-aware constraints are taken into consideration during the mapping process, the throughput for *Parallel-1* moves from 0.2173 (packet/cycle) to 0.254 which represents about 16.9% throughput improvement.

VII. Conclusion and Future Work

In this paper, we have addressed the issue of core-tile mapping for NoC-based platforms while considering the network contention minimization. We have reported our results obtained from many experiments involving both synthetic and real benchmarks. The results show significant reduction of packet latency (and implicitly, throughput improvements) by reducing the network contention.

Although in this paper we focus on 2-D mesh NoCs with *XY* routing, our idea can be further adapted to other architectures implementing under different network topologies with deterministic routing schemes. Moreover, the idea of minimizing the network contention is not limited to core-tile mapping as presented. Instead, it can be applied to other NoC synthesis problems and the mapping/scheduling heuristics on parallel systems to achieve further system throughput improvements.

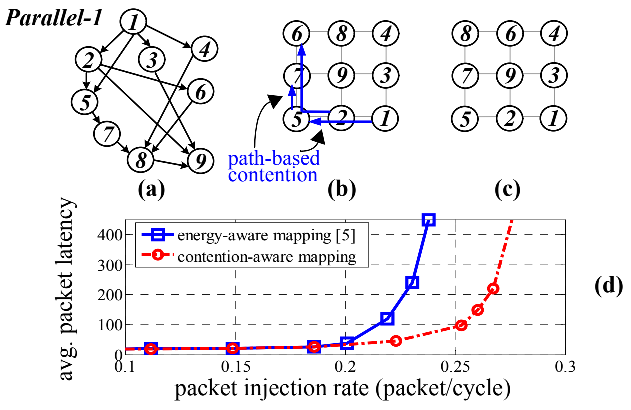

Fig. 5. (a) *Parallel-1* benchmark (b)(c) mapping results of the energy-aware approach [5] and our contention-aware method (d) average packet latency and throughput comparison under these two mapping methods.

Acknowledgments

Authors acknowledge the support of the Gigascale Systems Research Center and the Focus Center Research Program.

References

[1] J.-M. Chang, M. Pedram, "Codex-dp: co-design of communicating systems using dynamic programming," in IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, pp. 732-744, July 2000.
[2] A. Bender, "MILP based task mapping for heterogeneous multiprocessor systems," Proc. EURO-DAC '96, European, pp. 190-197, 1996.
[3] S. Murali, G. De Micheli, "Bandwidth-constrained mapping of cores onto NoC architectures," Proc. DATE, pp. 896-901, Feb. 2004.
[4] G. Ascia, V. Catania, M. Palesi, "Multi-objective mapping for mesh-based NoC architectures," Proc. CODES+ISSS, pp. 182-187, Sept. 2004.
[5] J. Hu, R. Marculescu, "Energy- and performance-aware mapping for regular NoC architectures," in IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, pp. 551-562, April 2005.
[6] C.-E. Rhee, H.-Y. Jeong, S. Ha, "Many-to-many core-switch mapping in 2-D mesh NoC architectures," Proc. ICCD, pp. 438-443, 2004.
[7] K. Srinivasan, K. S. Chatha, "A technique for low energy mapping and routing in network-on-chip architectures," Proc. International Symposium on Low Power Electronics and Design (ISLPED), pp. 387-392, 2005.
[8] K. Srinivasan, K. S. Chatha, G. Konjevod, "Linear programming based techniques for synthesis of network-on-chip architectures," Proc. ICCD, pp. 422-429, 2004.
[9] T. Lei, S. Kumar, "A two-step genetic algorithm for mapping task graphs to a network on chip architecture," Proc. Euromicro Symposium on Digital Systems Design (DSD), pp. 180-187, Sept. 2003.
[10] O. Ozturk, M. Kandemir, S. W. Son, "An ilp based approach to reducing energy consumption in nocbased CMPS," Proc. International Symposium on Low Power Electronics and Design (ISLPED), pp. 27- 29, 2007.
[11] M. Grotschel, L. Lovasz, A. Schrijver. Geometric Algorithms and Combinatorial Optimization. ISBN-10: 0387567402. Springer, Sep. 1993.
[12] http://lpsolve.sourceforge.net/5.5/
[13] Task graph for free (TGFF v3.1) D. Rhodes and R. Dick, and K. Vallerio. http://ziyang.eecs.northwestern.edu/~dickrp/tgff/
[14] B. Sethuraman, R. Vemuri, "optiMap: a tool for automated generation of NoC architectures using multi-port routers for FPGAs," Proc. DATE, pp. 947-952, 2006.
[15] A. Hansson, K. Goossens, A. Radulescu, "A unified approach to constrained mapping and routing on network-on-chip architectures," Proc. CODES+ISSS, pp. 75-80, 2005.
[16] H. G. Lee, *et al*, "On-chip communication architecture exploration: A quantitative evaluation of point-to-point, bus, and network-on-chip approaches," ACM Trans. on Design Automation of Electronic Systems (TODAES), 12(3), Aug. 2007.
[17] T. T. Ye, L. Benini, G. De Micheli, "Analysis of power consumption on switch fabrics in network routers," Proc. DAC, pp. 524-529, June 2002.
[18] O. Sinnen and L. A. Sousa, "Comparison of contention-aware list scheduling heuristics for cluster computing," Proc. Workshop Scheduling and Resource Management for Cluster Computing, pp. 382-387, 2001.
[19] O. Sinnen, L. A. Sousa, "Communication contention in task scheduling," in IEEE Trans. on Parallel and Distributed Systems, pp. 503-515, June 2005.
[20] J.W. van den Brand, C. Ciordas, K. Goossens, T. Basten, "Congestion-controlled best-effort communication for Networks-on-Chip," Proc. DATE, pp. 1-6, April 2007.
[21] U. Y. Ogras, R. Marculescu. "Analysis and optimization of prediction-based flow control in Networks-on-Chip," ACM Trans. on Design Automation of Electronic Systems (TODAES), vol. 13, no.1, Jan. 2008.