Power-Aware Mapping for Network-on-Chip Architectures under Bandwidth and Latency Constraints

Xiaohang Wang[1,2], Mei Yang[2], Yingtao Jiang[2], and Peng Liu[1]

[1]Department of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, China, 310027
[2]Department of Electrical and Computer Engineering, University of Nevada, Las Vegas, Las Vegas, NV 89154
Emails: [1]{baikeina@yahoo.com.cn, liupeng@isee.zju.edu.cn}, [2]{meiyang, yingtao}@egr.unlv.edu

Abstract—**This paper investigates the bandwidth- and latency-constrained IP mapping problem that maps a given set of IP cores onto the tiles of a mesh-based Network-on-Chip (NoC) architecture to minimize the power consumption due to inter-core communications. By examining various applications' communication characteristics shown in their communication trace graphs, two distinguishable connectivity templates are realized: the graphs with tightly coupled vertices and those with distributed vertices. Different mapping heuristics are developed for these templates: tightly coupled vertices are mapped onto tiles that are close to each other while the distributed vertices are mapped following a graph partition scheme. The proposed template-based mapping algorithm achieves on average 15% power saving compared with MOCA, a fast greedy-based algorithm. Compared with a branch-and-bound algorithm, the proposed algorithm can generate results of almost the same quality but require much less CPU time.**

Keywords-Network-on-Chip (NoC); power-aware; IP mapping; Template-based Efficient Mapping (TEM).

I. INTRODUCTION

With the continuous scaling of CMOS technologies, interconnects dominate both performance and power dissipation in future System-on-Chip (SoC) designs. Multiprocessor System on Chip (MPSoC) [2] designs have emerged and shown to deliver high performance yet reasonably low power consumption. Networks-on-chip (NoC) has been considered as a viable communication infrastructure in MPSoC [9]. Regular mesh remains the dominant NoC architecture of choice due to its distinct features: structured network wiring, modularity, and standard interfaces [8].

This paper focuses on mapping the IP cores onto a regular tile-based NoC architecture and the mapping is subjected to the latency and bandwidth constraints as imposed by many real-time multimedia applications. In the literature, a number of algorithms have been proposed to solve this IP mapping problem, and they fall into four general categories:

- *Branch-and-bound algorithms* [8], which can generate very high quality results. With large queue size, however, this algorithm demands high memory depth and suffers from long CPU time.
- *Framework-based approaches*, such as the ones using simulated annealing (SA)/genetics algorithm (GA)/tabu search (TS) [1, 6, 10, 11]. These algorithms typically require considerably longer time than a greedy-based mapping algorithm.
- *Linear programming based schemes* [13]. Computation is time-consuming and there is no guarantee that high-quality solutions can be always found.
- *Greedy-based heuristics* [5, 12, 15], which require significantly low CPU time. The following greedy algorithms are based on different observations.

The algorithm MOCA [15] achieves a sound balance between the run time and the quality of solutions. The problem of this algorithm is that the graph partition algorithm may separate some tightly coupled regions (i.e. the regions made of IP cores with large degrees and strongly connected) into different sets, making the vertices in a tightly coupled region adversely mapped onto tiles physically far apart. Particularly, the power performance of MOCA deteriorates drastically when latency constraints are applied.

In this paper, we propose a Template-based Efficient Mapping (TEM) algorithm which generates high-quality mapping results with low run time. This algorithm is designed based on two distinguishable connectivity templates extracted from various applications' communication trace graphs: the graphs with tightly coupled vertices and those with distributed vertices. Correspondingly, different mapping strategies are proposed for these two templates. Simulation results show that the proposed TEM algorithm achieves better result than MOCA [15].

The rest of the paper is organized as follows. Section II formally defines the mapping problem. Section III introduces the two templates derived from various applications' communication trace graphs. The mapping algorithm based on the two templates is presented in Section IV. Section V presents and discusses the simulation results, and finally, Section VI concludes the paper.

This work is in part supported by NSF under grant no. ECCS-0702168 and NSF of China under grant no. 60873112.

II. PROBLEM FORMULATION

A. Architecture description and power model

Without loss of generality, the NoC system under consideration is composed of NxN tiles interconnected by a 2-D mesh network. The power model used in [8] is followed in this study. The average power consumption for sending one bit of data from tile t_i to tile t_j can be represented as

$$E^{t_i,t_j}_{bit} = \eta_{hops} \times E_{Sbit} + (\eta_{hops} - 1) \times E_{Lbit} \qquad (2)$$

where η_{hops} is the number of routers traversed from tile t_i to tile t_j, E_{Sbit} is the power consumed by the switch, and E_{Lbit} is the power consumed on the links between tiles t_i and t_j.

B. Problem Description

We assume that before IP mapping is performed, a given application described by a set of concurrent tasks is already bounded and scheduled onto a list of selected IP cores. As defined below, the communication patterns between any pair of IP cores of an application are described by its Communication Trace Graph, whereas the NoC architecture is modeled by its Architecture Characterization Graph.

Definition 1 A *Communication Trace Graph* (CTG) $G=(P, E)$ is an undirected graph, where a vertex/node $p_k \in P$ represents an IP core (a processor, an ASIC device or a memory unit, etc.), and an edge $e_i=(p_k, p_j)\in E$ represents the communication trace between vertices p_k and p_j. For each edge e_i,

- $\omega(e_i)$ defines the communication bandwidth request between vertices p_i and p_j given in bits per second (bps). $\omega(e_i)$ sets the minimum bandwidth that should be allocated by the network in order to meet the performance constraints.
- $\sigma(e_i)$ represents the latency constraint, which is given in number of hops instead of an absolute number in cycles [15].
- $W(e_i)$ represents the weight of edge e_i. The weight is defined in the same way as that in [15]. Among all the traces in the graph, let e_i be the trace with the highest bandwidth requirement, and e_j be the trace with the tightest (lowest) latency constraint. An integer K is defined as the minimum value required to ensure that among all the traces in the graph, $\frac{\omega(e_i)}{\sigma(e_i)^K} \le \frac{\omega(e_j)}{\sigma(e_j)^K}$. Once K is determined, each edge is assigned a weight $W(e) = \frac{\omega(e\ )}{\sigma(e\ )^K}$.

Definition 2 An *Architecture Characterization Graph* (ACG) $\breve{G}=(T, L)$ is an undirected graph, where each vertex $t_i\in T$ represents a tile and each edge $l_i\in L=(t_k, t_j)$ represents the link between tiles t_k and t_j. For each link l_i,

- $bw(l_i)$ defines the bandwidth provided between tiles t_k and t_j.
- $c(l_i)$ defines the link cost of l_i, i.e., power consumption for transmitting one bit data from t_k to t_j.

In this paper, we focus on regular NoC architectures which have $bw(l_i)=B$, $c(l_i)= C$ for each $l_i\in L$, where B and C are constants. $h_{k,\ j}$ is the set of links forming one of the shortest paths from tile t_k to tile t_j ($h_{k,\ j} \subseteq L$). $dist(h_{k,\ j})$ determines the number of elements in $h_{k,\ j}$ (i.e. it is the hop count of the shortest path between tile t_k and tile t_j).

Definition 3 A mapping algorithm M: $P \rightarrow T$ maps each vertex in CTG onto an available tile in ACG. $M(p_i)$ represents the mapped tile in ACG, where $p_i \in P$ and $M(p_i) \in T$.

Definition 4 A routing algorithm R: $E \rightarrow H$, finds one of the shortest routing path between $M(p_k)$ and $M(p_j)$ for each edge $e_i=(p_k, p_j)\in E$. The links of forming this path belongs to set $h_{M(p_k),M(p_j)}$.

The IP mapping problem is formulated as follows.

Given a CTG(P, E) representing the communication pattern of an application and an ACG(T, L) representing the target NoC architecture, where $|P| \leqslant |T|$, find a mapping M:$P \rightarrow T$ which maps all the vertices in CTG onto available tiles in ACG and generates a deadlock-free and minimal routing paths for all edges in CTG, such that the total power consumption is minimized, i.e.,

$$\text{Min } \{ \sum_{\substack{i=0 \\ e_i=(p_k,p_j)\in E}}^{|E|} \omega(e_i) \times \sum_{\substack{m=0 \\ l_m \in h_{M(p_k),M(p_j)}}}^{|h_{M(p_k),M(p_j)}|} C \} \qquad (3)$$

satisfying

$$\forall p_i \in P,\ M(p_i) \in T, \qquad (4)$$

$$\forall p_i, p_j \in P \text{ and } p_i \ne p_j,\ M(p_i) \ne M(p_j), \qquad (5)$$

$$\forall\ l_i,\ B \ge \sum_{e_i=(p_k,p_j)} \omega(e_i) \times f(l_i, h_{M(p_k),M(p_j)}) \qquad (6)$$

$$\forall\ e_i=(p_k, p_j),\ \sigma(e_i) \ge dist(h_{M(p_k),M(p_j)}) \qquad (7)$$

where $f(l_i, h_{M(p_k),M(p_j)}) = \begin{cases} 1\ if\ l_i \in h_{M(p_k),M(p_j)} \\ 0\ if\ l_i \notin h_{M(p_k),M(p_j)} \end{cases}$

Similar to the definition adopted in [8], conditions given by (4) and (5) ensure that each IP should be mapped exactly to one tile and no tile can host more than one IP. Eqn. (6) specifies the bandwidth constraint for every link set by the communication bandwidth requirement, and Eqn. (7) ensures that the latency constraint (in terms of the number of hops) between two communicating IPs is satisfied on the mapped tiles.

III. DERIVATION OF MAPPING TEMPLATES

As alluded in Section 1, MOCA [15] does not perform well when the latency constraints are considered. This problem is illustrated in the mapping of the MPEG4 decoder, with its CTG shown in Figure 1.

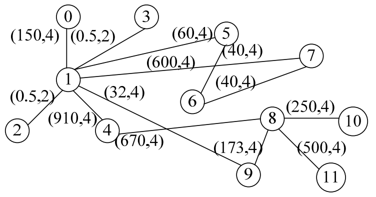

Figure 1 The CTG of MPEG4 decoder [15]. The BW request (Mbps) and latency constraint (number of hops) of each edge is labeled on the edge.

A communication path with higher bandwidth request should be mapped to links with fewer hop counts to reduce power consumption as Eqn. (3) indicates. When the latency constraint is applied, the result of MOCA deteriorates. For example, in Figure 2(a), the hop counts of edges (1,7), (8,9), (8,10) are greater than 1, which results in higher power consumption.

As a matter of fact, as shown in Figure 2(b), there is a better mapping by reducing the number of hops for these edges. In Figure 1, vertices 1 and 8 have many edges requesting high bandwidths or tight latency. In Figure 2(b), these two vertices are mapped onto the two tiles with the largest degree or maximal number of neighbor tiles (i.e., 4). In this solution, edges with higher weights are mapped with fewer hop counts than those edges with smaller weights.

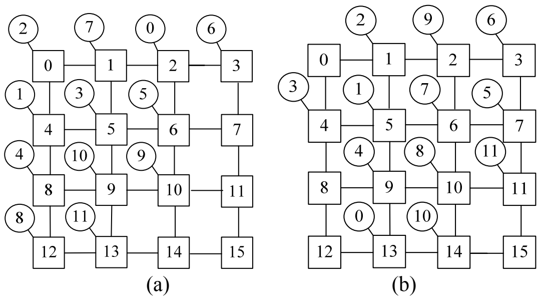

Figure 2 (a) The mapping result of MOCA on the CTG of MPEG4dec w/ latency constraints (b) A better mapping solution on the same CTG w/ latency constraint. The square boxes are tiles and the circles are IP cores.

From the above example, one can see that different mapping strategies shall be adopted for CTGs with different features. Here, we present two distinct templates derived from various CTGs. Given a CTG(P, E), and a sorted list of edges in decreasing order of edge weight denoted as $\hat{E}$:

Definition 5 A vertex $p_i \in P$ is a *hot node* if

a) p_i has a degree greater than or equal to 4, and

b) of the first 50% edges in the edge list $\hat{E}$, there are at least one edges that are connected to p_i.

Template 1: An application's CTG falls into Template 1 (tightly coupled) if there is at least one hot node in the CTG.

Template 2: An application falls into Template 2 (distributed) if there is no hot node in the application's CTG.

IV. ALGORITHM DESCRIPTION

The overall structure of the TEM algorithm is shown below. Before the template-based mapping takes place, the edges have to be sorted in non-increasing order in terms of the edge weight. After all the vertices are mapped, a routing allocation routine is called to find routing paths for all pairs of communicating vertices.

```
TEM ($G(P,E)$, $\check{G}(T,L)$)
Input: (1) $G$: CTG of an application.
       (2) $\check{G}$: ACG of an NoC architecture
Output: none
Function: Map the application on the NoC architecture and allocate
          routing paths
Procedure body:
{
    Sort_Edge($E$);   // $E \subset G$. Let the sorted list be $\hat{E} = e_{\pi(1)}, e_{\pi(2)}, \dots, e_{\pi(|E|)}$, i.e.,
                      // $W(e_{\pi(1)}) \geq W(e_{\pi(2)}) \geq \dots W(e_{\pi(|E|)})$.
    // Find the hot nodes if any. Check which template the application belongs to
    for each vertex $p_i$ { // Definition 5
        if (vertex $p_i$ has degree $\geq \alpha$ and $p_i$ has $\gamma$ edges in the first $\beta$ edges in $E$){
            mark $p_i$ as a hot node an place into set $\hat{H}$
        }
    }
    // If $G$ has hot nodes, it belongs to Template 1, otherwise Template 2
    // Template specific algorithm
    case 1: Template 1 { // Tightly coupled
        TEM_Template1($G$, $\check{G}$, $\hat{H}$);
        break;
    }
    case 2: Template 2 { // Distributed
        TEM_Template2($G$, $\check{G}$);
        break;
    }
    Route_Alloc($G$, $\check{G}$); // Find routing paths
}
```

A. Mapping Algorithm for Template 1

An application of template 1 (CTG has at least one hot node) is mapped based on the following observations.

a) Hot nodes should be given a higher mapping priority; that is, they shall be mapped before any other nodes are mapped. All the hot nodes in a CTG will be first mapped along with their α most significant neighbor vertices. A hot node is better mapped onto a tile in an NoC that has the maximum number of neighbor tiles.

b) Once all the hot nodes are mapped, the mapping sequence of remaining unmapped non-hot nodes will be performed based on the decreasing order of weight of edges connecting them.

As such, TEM_Template1 procedure consists of two major steps: (1) map hot nodes in CTG, (2) map other vertices.

```
TEM_Template1 ($G(P,E)$, $\check{G}(T,L)$, $\hat{H}$)
Input: (1) $G$: CTG of an application
       (2) $\check{G}$: ACG of an NoC architecture
       (3) $\hat{H}$: The set of hot nodes
Output: none
Function: Map the Template 1 application onto the NoC architecture
Procedure body:
{
    // Step 1: find and map the hot nodes with their 4 most significant
    // neighbors
    for each edge $e_{\pi(i)}=(p_k, p_j)$ {
        increase the counter of $p_k$, $p_j$ if they are hot nodes (belong to $\hat{H}$)
        if one of the vertex's counter equals $\alpha${ // suppose $p_k$'s counter is $\alpha$
            Map_Hot_Node($p_k$, $\hat{E}$, $T$);
        }
    }

    // Step 2: map the remaining unmapped vertices
    update the hop counts of edges whose terminal vertices are already
    mapped;
    // re-scan the edge list
    for each edge $e_{\pi(i)}=(p_k, p_j)${
        Map_Edge($e_{\pi(i)}$, $\hat{E}$, $T$);    // $T$ is the available tile set
    }
}
```

Before we discuss each step in detail, the following definitions are introduced.

Definition 6 Vertex p_n is a *significant* neighbor of vertex p_i if there exists an edge $(p_i, p_n) \in E$ and edge (p_i, p_n) is within the first 50% edges in the edge list $\hat{E}$.

Definition 7 Since in a mesh structure, the maximal degree of a tile is four, here the *four most significant* neighbors $\{p_{n1},..., p_{n4}\}$ of a hot node p_i are the four neighbors of p_i that have the highest bandwidth/tightest latency requirements among all the neighbors of p_i .

Definition 8 A *center tile* in a 2-D topology like mesh is a tile that has maximal number (i.e. 4) of unmapped neighbor tiles.

Definition 9 Two vertices p_k , p_j are "*close*" if (1) edge (p_k, p_j) is among the first 50% edges in the edge list or (2) a neighbor vertex of p_k (or p_j) is connected with p_j (or p_k), and this edge is among the first 50% edges in the edge list.

1) Map hot nodes. Procedure Map_Hot_Node maps each hot node and its four most significant neighbor vertices. Each hot node p_i is associated with a counter ctr_i that counts the number of times that vertex p_i appears as a terminal vertex of the edges in the sorted edge list.

- Case 1: the current hot node p_i is not mapped yet. Check whether this hot node is *close* (Definition 9) to any of those mapped hot nodes. If yes, this hot node is mapped to a tile with minimum hops to an already mapped hot node. Otherwise, a center tile is selected and it is allocated to this hot node. Next its four most significant neighbors are mapped to the neighboring tiles if they have not been mapped.
- Case 2: the current hot node has already been mapped because it belongs to the first α neighbors of a previously mapped hot node. In this case, only the four most significant neighbor vertices need to be mapped. The procedure Improve_Edge can be called for optimization.

2) Map other nodes. In this step, all the remaining unmapped vertices are mapped. Procedure Map_Edge maps the remaining unmapped vertex/vertices.

There are three cases to consider.

- Case 1: Neither of the two terminal vertices p_k, p_j of edge $e_{\pi(i)} = (p_k, p_j)$ is mapped. Search the edges from $e_{\pi(i)+1}$ to $e_{\pi(|E|)}$. If one of the neighbors of the two terminal vertices is found and is mapped, then the vertex with a mapped neighbor is mapped onto a tile that has the minimum hop count to its mapped neighbor. On the other hand, if none of the two terminal vertices' neighbors is mapped, an available tile is selected and immediately allocated to one of the vertices, after which the other vertex is mapped onto a tile with a minimum hop count to the just mapped tile.
- Case 2: One of the two terminal vertices is mapped, but the other one is not. In this case, only the unmapped vertex needs to be mapped onto a tile with minimum hop count to the mapped tile.

The Map_Hot_Node procedure is listed below

```
Map_Hot_Node($p_{hi}$, $\hat{E}$, $T$)
Input: (1) $p_{hi}$: A hot node
       (2) $\hat{E}$: The sorted edge list
       (3) $T$: The available tile set in NoC architecture
Output: none
Function: Map hot node $p_{hi}$ and its neighbors
Procedure body:
{
   // Case 1
      if ($p_{hi}$ is unmapped){ // 1) Map the hot node $p_{hi}$
      check if other mapped hot nodes are close to $p_{hi}$;   // Definition 9
      if (there is a mapped hot node $p_{hk}$ close to hot node $p_{hi}$){
          map $p_{hi}$ to a tile with minimum hop count to $p_{hk}$'s tile;
      }
      else{
          map $p_{hi}$ to $t_l$ = Find_Center_Tile();
      }
      // 2) Map the four most significant neighbors
          for each of $p_{hi}$'s four most significant neighbors {
          // suppose $p_n$ is one of such neighbors
          if ($p_n$ is unmapped){
              // Two criteria should be observed in mapping neighbors as
              // described in text
              map $p_n$ to a tile with minimum hops to $p_{hi}$'s tile;
          }
      }
   }
   // Case 2
   else if ($p_{hi}$ is mapped) {
      for each of $p_{hi}$'s four most significant neighbors {
          // suppose $p_n$ is one of such neighbors
          if ($p_n$ is not mapped) {
              map $p_n$ to a tile with minimum hop count to $p_{hi}$'s tile ;
          }
      }
   }
}
```

Figure 2(b) shows the mapping result of the CTG of MPEG4 decoder. The complexity of TEM_Template1 is $O(|\hat{H}|\cdot(|E|+|T|)+|E|\cdot(|E|+|T|))$, which can be further simplified as $O((|E|+|T|)^2)$ since $|\hat{H}|\leq|P|\leq|T|$.

B. TEM Algorithm for Template 2

For a Template 2 CTG, the graph partition algorithm proposed in [4] is followed to partition both CTG and ACG into four smaller regions so that each region in CTG can be mapped onto a region in ACG in a divide-and-conquer manner.

Definition 10 A *Block Trace Graph* (BTG) is denoted as $G'=(B, BE)$, where each vertex $b_i \in B$ is a partitioned block and $b_i \subset P$ (P is the vertex set of CTG(P, E)), and an edge $be_i=(b_k, b_j) \in BE$ exists if $p_{kl} \in b_k$ and $p_{jm} \in b_j$, $(p_{kl}, p_{jm}) \in E$. For an edge $be_i=(b_k, b_j) \in BE$,

- $W(be_i)$ represents the weight of be_i. The calculation is the same as the weight of edges in CTG.
- An CTG edge $(p_i, p_j) \in E$ belongs to a block b_k if $p_i \in b_k$ or $p_j \in b_k$. E_i represents the set of edges belonging to b_i.

The vertices in each block are classified into two types and they are mapped differently:

1) *Internal vertices*. An internal vertex p_{intern} has all its neighbors in the same block of p_{intern}.
2) *External vertices*. An external vertex p_{extern} has at least one of its neighbors not in the same block of p_{extern}.

Definition 11 A *Virtual ACG* (VACG) is denoted as $\breve{G}'=(R, CH)$, where a vertex $r_i \in R$ represents a partitioned region and $r_i \subset T$ (T is the tile set of ACG(T, L)). Each edge $ch_i=(r_k,$

$r_j) \in CH$ represents that there exists direct links between tiles in r_k and r_j. For each edge $ch_i \in CH$,

The TEM_Template 2 algorithm works as follows.

```
TEM_Template2($G(P,E)$, $\breve{G}(T,L)$)
Input: (1) $G$: The CTG of an application.
       (2) $\breve{G}$: The ACG of an NoC architecture
Output: none
Function: Map the Template 2 application onto the NoC architecture
Procedure body:
{
    // Step 1. Obtain the BTG and VACG by partitioning the CTG into blocks
    // and ACG into default regions, respectively. Map the blocks in BTG to
    // the regions in VACG.
    obtain BTG ($G'$) using the graph partition algorithm to partition CTG into
    4 blocks;
    obtain VACG ($\breve{G}'$) with default four square regions;
    map the four blocks in BTG to four default regions of VACG using
    Map_Edge;
    Partition_VACG($G'$, $\breve{G}'$);
    // Step 2. Inside each block, map the vertices to the tiles of the
    // corresponding region.
    for each block $b_i$ in BTG mapped to region $r_x$ in VACG {
        // 1) Map the external vertices.
        // Let $PE_i$ be set of external vertices in $b_i$;
        sort external vertices in $PE_i$ in the non-increasing order of the total
        weight of the external vertices. Let $p_{\Phi(1)}, p_{\Phi(2)}, \ldots, p_{\Phi(|PEi|)}$ be the
        sorted list;
        for each external vertex $p_{\Phi(i)}${
            Map_External_Vertex($p_{\Phi(i)}$, $G'$, $\breve{G}'$);
        }
        // 2) Map the internal vertices
        sort the edges in $b_i$ into $\hat{E}_i$ using Sort_Edge;
        for each edge $e_{\psi(i)} = (p_k, p_j) \in \hat{E}_i$ {
            Map_Edge ($e_{\psi(i)}$, $b_i$, $r_x$); // described in Section 4.1
        }
    }
}
```

1) Partition. In this step, the BTG is formed using a graph partition algorithm [4] to partition the CTG into four blocks. The VACG is formed by partitioning the ACG into four default regions (e.g., square regions in Figure 4. The blocks in the BTG are mapped to the default regions in the VACG. Based on the mapping result of BTG onto VACG, the actual regions in the VACG are partitioned in a top-down manner. The size of each partitioned region needs to be set equal to the size of the corresponding block in the BTG.

2) Map inside blocks. In this step, the vertices in each block are mapped to the tiles within its region. The external vertices are first mapped to the border of each NoC region, after which the internal vertices are mapped with larger weight edge first criterion.

The overall complexity of TEM_Template2 can be approximated by $O((|E|+|T|)^2)$ as $|blk_i|<|P|\leq|T|$, $|E_i|<|E|$, $|T_i|<|T|$.

V. PERFORMANCE EVALUATION

To evaluate the performance of the TEM algorithm, both TEM_Template 1 and TEM_Template2 are implemented and simulated. The brand-and-bound (BNB) [8] and MOCA [15] algorithms are also implemented and the mapping results of all three algorithms are compared. To isolate the effects of different routing algorithms, the XY routing and odd-even routing algorithms are selected. Multimedia benchmarks from [7] and [15]. The Noxim simulator [14] is modified and used in our simulations to obtain the total communication power after the mapping and the routing path allocation steps.

The simulations are performed on a PC with one Intel Core2 P8600 2.4GHz processor and 2GB RAM. The run times of both TEM and MOCA range from 0.01sec~1sec for different network sizes.

Multimedia benchmarks of both Template 1 and Template 2 are tested on a mesh-based NoC. Table I lists these benchmarks. For benchmarks of both templates, we compare the degradation of TEM compared to BNB (i.e. the increase in power consumption of mapping results from the TEM compared to that from BNB) and the degradation of MOCA compared to BNB. The power consumption is normalized.

TABLE I. MULTIMEDIA BENCHMARKS AND THEIR TEMPLATES.

Benchmark	**Template**
MPEG4	1
263enc	1
263enc+MP3enc	1
263enc+MP3dec	1
263enc+263dec	1
Multimedia Systems (MMS)	1
VOPD	2
MP3enc	2
MP3enc+MP3dec	2

Table II shows the result of benchmarks of Template 1 with and without latency constraints. When the latency constraints are not considered, the result of TEM is comparable with that of MOCA. When the latency constraints are considered, the TEM algorithm outperforms MOCA for most media programs. The degradation of TEM vs. BNB is within 10% with odd-even routing. The reduction in power consumption of TEM compared to MOCA is over 15%. For MPEG4dec, the degradation of TEM is only 6% compared to that of MOCA 45%.

The Multi-Media System (MMS) benchmark [8] is also simulated on a 5x5 mesh-based NoC. Table III shows the reduction in power consumption of TEM over MOCA (normalized).

Table IV shows the result of benchmarks of Template 2 with and without latency constraints. Without latency constraints, the result of TEM is comparable with that of MOCA. On average, the degradation of TEM over BNB is within 10% for both XY routing and odd-even routing. With latency constraints, the power consumption of mapping result from TEM is slightly lower than that from MOCA for VOPD and MP3enc. For MP3enc+MP3dec, TEM achieves significant reduction in power consumption compared to MOCA. This is due to the fact that the TEM algorithm for applications of Template 2 uses a divide-and-conquer approach. Inside each partitioned block, TEM first maps external vertices to the border of the block and the remaining edges with larger weight first criterion. Thus larger weight edges are mapped first inside each block.

Evaluation of TEM on random applications is also conducted. Detailed results are presented in [16].

TABLE II. COMPARISON OF POWER CONSUMPTION (NORMALIZED) OF TEM AND MOCA OVER BNB ON BENCHMARKS OF TEMPLATE 1.

Benchmarks	**With latency constraint**				**Without latency constraint**			
	TEM		**MOCA**	**BNB**	**TEM**		**MOCA**	**BNB**
	XY	**Odd-even**			**XY**	**Odd-even**		
MPEG4	1.05	1.05	1.45	1	1.01	1.01	1.06	1
263 enc	1.07	1	1.11	1	1.4	1.03	1	1
263enc+MP3enc	1.14	1.11	1.27	1	1.01	1	1	1
263enc+MP3dec	1.06	1.01	1.28	1	1.04	1.01	1	1
263enc+263dec	1.06	1.01	1.9	1	1.06	1.02	1	1

TABLE III. COMPARISON OF POWER CONSUMPTION (NORMALIZED) OF TEM OVER MOCA ON MMS (TEMPLATE 1).

Benchmarks	**With latency constraint**			**Without latency constraint**		
	TEM		**MOCA**	**TEM**		**MOCA**
	XY	**Odd-even**		**XY**	**Odd-even**	
MMS	0.9	0.84	1	0.97	0.91	1

TABLE IV. COMPARISON OF POWER CONSUMPTION (NORMALIZED) TEM AND MOCA OVER BNB ON BENCHMARKS OF TEMPLATE 2.

Benchmarks	**With latency constraint**				**Without latency constraint**			
	TEM		**MOCA**	**BNB**	**TEM**		**MOCA**	**BNB**
	XY	**Odd-even**			**XY**	**Odd-even**		
VOPD	1.02	1.02	1.1	1	1.02	1.02	1.08	1
MP3enc	1.1	1.1	1.12	1	1	1	1	1
MP3enc+MP3dec	1.14	1.11	1.41	1	1.08	1.07	1.08	1

VI. CONCLUSION

This paper presented a template-based greedy algorithm to address the IP mapping problem under the bandwidth and latency constraints. An application falls into Template 1, if there are one or more hot nodes which demand either higher communications bandwidths or tighter latency. In this case, TEM maps the hot nodes first along with their four most significant neighbors, after which the remaining IP cores are mapped in descending order based on the edge weights. On the other hand, an application is categorized as Template 2, if the communications are nearly evenly distributed among the vertices. In this case, TEM divides the NoC into regions and CTG into blocks; then it maps the IP cores inside each block in a divide-and-conquer manner. The experiments on multimedia and random benchmarks show that TEM generates high quality mapping results with low run times.

VII. REFERENCES

[1] G. Ascia, V. Catania, and M. Palesi, "Multi-objective mapping for mesh-based NoC architectures," in *Proc. 2nd IEEE/ACM/IFIP Int'l Conf Hardware/Software Codesign and System Synthesis*, 2004, pp. 182-187.

[2] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, and G. De Micheli, "NoC synthesis flow for customized domain specific multiprocessor Systems-on-Chip," *IEEE Trans. Parallel And Distributed Systems,* pp. 113-129, 2005.

[3] R. P. Dick, D. L. Rhodes, and W. Wolf, "TGFF: task graphs for free," in *Proc. 6th Int'l workshop on Hardware/software Codesign*, 1998, pp. 97-101.

[4] C. M. Fiduccia and R. M. Mattheyses, "A Linear-time heuristic for improving network partitions," in *Proc. 19th IEEE Conf. Design Automation*, 1982, pp. 175-181.

[5] A. Hansson, K. Goossens, and A. Rǎ dulescu, "A unified approach to constrained mapping and routing on network-on-chip architectures," in *Proc. 3rd IEEE/ACM/IFIP Int'l Conf Hardware/Software Codesign and System Synthesis*, 2005, pp. 75-80.

[6] H. M. Harmanani and R. Farah, "A method for efficient mapping and reliable routing for NoC architectures with minimum bandwidth and area," in *Proc. Conf. Circuits and Systems and TAISA*, 2008, pp. 29-32.

[7] J. Hu and R. Marculescu, "Energy-aware mapping for tile-based NoC architectures under performance constraints," in *Proc Conf. Asia South Pacific Design Automation*, 2003, pp. 233-239.

[8] J. Hu and R. Marculescu, "Energy-and performance-aware mapping for regular NoC architectures," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* vol. 24, pp. 551-562, 2005.

[9] A. Jantsch and H. Tenhunen, *Networks on Chip*. New York, NY: Kluwer Academic Publishers, 2003.

[10] Z. Lu, L. Xia, and A. Jantsch, "Cluster-based simulated annealing for mapping cores onto 2D mesh networks on chip," in *Proc. Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, 2008, pp. 1-6.

[11] C. A. M. Marcon, E. I. Moreno, N. L. V. Calazans, and F. G. Moraes, "Evaluation of algorithms for low energy mapping onto NoCs," in *Proc. IEEE Int'l Symp. Circuits and Systems(ISCAS)*, 2007, pp. 389-392.

[12] A. Mehran, S. Saeidi, A. Khademzadeh, and A. Afzali-Kusha, "Spiral: a heuristic mapping algorithm for network on chip," *J. IEICE Electronics Express,* vol. 4, pp. 478-484, 2007.

[13] S. Murali and G. De Micheli, "Bandwidth-constrained mapping of cores onto NoC architectures," in *Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2004, pp. 896-901.

[14] Noxim, Available at: http://sourceforge.net/projects/noxim.

[15] K. Srinivasan and K. S. Chatha, "A technique for low energy mapping and routing in network-on-chip architectures," in *Proc. 2005 Int'l Symp. Low Power Electronics and Design*, 2005, pp. 387-392.

[16] X. Wang, M. Yang, Y. Jiang and P. Liu, "A power-aware mapping approach to map IP cores onto NoCs under bandwidth and latency constraints," to appear in *ACM Trans. Architecture and Code Optimization.*