Power-Aware Run-Time Incremental Mapping for 3-D Networks-on-Chip

Xiaohang Wang[1], Maurizio Palesi[2], Mei Yang[3],
Yingtao Jiang[3], Michael C. Huang[4] , Peng Liu[1]*
[1] Zhejiang University, China
[2]Kore University, Italy
[3] University of Nevada Las Vegas, USA
[4]University of Rochester, USA
[1] {baikeina,liupeng}@zju.edu.cn, [2]maurizio.palesi@unikore.it,
[3]{mei.yang, yingtao}@egr.unlv.edu,
[4]michael.huang@rochester.edu

Abstract. 3-D Networks-on-Chip (NoCs) emerge as a powerful solution to address both the interconnection and design complexity problems facing future Systems-on-Chip (SoCs). Effective run-time application mapping on a 3-D NoC-based Multiprocessor Systems-on-Chip (MPSoC) can be quite challenging, largely due to the fact that the arrival order and task graphs of the target applications are not known a priori. This paper presents a power-aware run-time incremental mapping algorithm for 3-D NoCs that aims to minimize the communication power for each incoming application as well as reduce the impact of the mapped applications on future applications that are yet to be mapped. In this algorithm, if the vertical links are found to be shorter and provide higher communication bandwidth than horizontal links, more communications will be mapped to vertical links to reduce delay and power consumption. Extensive experiments have been conducted to evaluate the performance of the proposed algorithm and the results are compared with those obtained from the optimal mapping algorithm (branch-and-bound), a random mapping and a simple heuristic. When mapping a single application, the proposed algorithm is four orders of magnitude faster than the branch-and-bound algorithm at a small degradation of mapping quality. When mapping multiple applications incrementally, our algorithm can save 50% communication power compared to the random mapping and 20% communication power compared to the simple heuristic.

Keywords: Networks-on-chip, application mapping, 3-D IC.

1 Introduction

Advance in CMOS technology keeps driving the increase of the number of processing cores that can be integrated on a single chip. With the further increase of the number of processing cores, the integration limitations of SoCs become more significant. 3-D integration together with the NoC design paradigm provide a promising solution to overcome these limitations [1]. In the literature, several approaches exist for 3-D integration, e.g., wire bounded, microbump, through via and contactless [2]. 3-D

* Peng Liu is the corresponding author.
This work is supported in part by NSFC under the grants 60873112 and 61028004

integration can considerably reduce the global interconnection length, resulting in lower interconnection delay, power consumption and also the overall chip area [2]. Combining 3-D ICs and NoC can provide significant performance improvement and make the chip more power efficient .

The large number of processing cores integrated into 3-D NoC-based MPSoCs unquestionably offer high parallelism in computation. To better utilize these vast available computation resources, virtualization is applied to allow a single MPSoC to be shared by multiple applications which can be mapped to different networks of the chip at run time. However, the behaviors of the multiple applications vary so dramatically at run time, making it nearly impossible for these applications to be efficiently mapped offline. For these applications, run-time incremental mapping methods should be designed which could not only minimize the overall communication power but also consider future applications whose arrival orders are not known. In addition, in 3-D NoCs, the wire lengths of vertical links (in a few tens of μm) is much shorter than those of horizontal links (in a few thousand μm) [1][3]. As such, more communication could be mapped to vertical links to further reduce transmission delay and power consumption. In a simple word, a run-time mapping method for applications on 3-D NoC systems shall take multiple factors into consideration, namely the dynamic feature of applications with ever changing behaviors and the benefits of vertical links in 3-D NoCs.

Up to date, there is very little work on run-time 3-D NoC mapping reported in the open litterature. Existing approaches for offline 3-D NoC mapping [4] are not suitable for run-time mapping as they do not consider the shape of the mapped region and the impact on future applications. The 2-D NoC run-time incremental mapping algorithms [5, 6] cannot be directly applied to 3-D NoCs because the vertical links in 3-D NoCs are not an issue in those approaches.

In this paper, we propose a novel run-time incremental mapping algorithm which maps applications that can randomly enter and leave an embedded 3-D NoC system while considering the benefits of vertical links. The mapping algorithm tries to minimize overall communication power as well as minimize the impact on future applications (i.e., to reduce the fragmentation caused by small region of tiles). The proposed algorithm is composed of three steps: 1) *NoC region selection* which selects a cuboid region to reduce the impact on future applications; 2) *set matching* which allocates the application graph to the sub-regions in different layers such that the vertical links are used as much as possible; and 3) *CTG to NoC region mapping* which maps the IP cores to the tiles in different sub-regions with minimized total communication power. The higher bandwidth and lower delay properties of the vertical links in 3-D NoC are particularly exploited in set matching and region mapping steps. Experimental results have confirmed that the proposed algorithm saves up to 50% communication power over random mapping and about 20% over a simple heuristic.

The rest of this paper is organized as follows. Section 2 summarizes the related work. Section 3 provides the preliminaries and problem definition. Section 4 presents a motivating example and decomposition of the problem. Following the observations from the example, Section 5 presents the run-time mapping algorithm in detail. Section 6 presents the simulation results. Finally, Section 7 concludes the paper.

2 Related Work

Research in 3-D NoCs emerges in recent years. In [1], both the analytical model of zero-load latency and the power model for 3-D NoCs are provided. Different 3-D NoC designs are shown in [7, 8, 9]. In [7], a three-layer NoC is proposed where each layer has different network topology so that the planar topology on every layer could be customized to meet different cost-performance demands. In [8], both 3-D mesh-based architectures (symmetric 3-D mesh, stacked mesh, and ciliated mesh) and 3-D tree-based architectures are analyzed in terms of network performance and energy dissipation. The analysis shows that 3-D mesh-based architectures demonstrate significant performance gains with a small area overhead compared with 2-D meshes. In [3], the 3-D Dimensionally-Decomposed (DimDe) Router is proposed. The DimDe router features is a true 3D crossbar with two vertical interconnects spanning over all layers. As such, the crossbars of all the routers with the same vertical index are fused into one unit. With the fusion of crossbar, short vertical links could be utilized such that vertical transmissions among different layers only take one hop. In addition, the crossbar is decomposed to reduce complexity. Simulation results show that the 3-D mesh NoC constructed with DimDe routers achieves better throughput, latency and energy-delay product compared with a symmetric 3-D mesh built with 7-port routers extended from 5-port routers [3].

In the literature, a number of IP mapping algorithms have been proposed for 2-D NoCs with the objective of minimizing the overall communication power. However, most of these existing mapping algorithms designed for 2-D NoCs summarized in [10] cannot be used in run-time incremental mapping for 3-D NoCs as these algorithms do not consider the short vertical links featured in 3-D NoCs. The branch-and-bound algorithm [11][12] can be extended to run-time incremental mapping for 3-D NoCs. However, the extremely long running time is not suitable for run-time mapping. In [4], a genetic algorithm based thermal-aware 3-D offline mapping is proposed. Due to the complexity, it is also not suitable for run-time 3-D NoC mapping.

In terms of run-time incremental mapping for 2-D NoCs, several schemes have been proposed. In [13], a run-time task assignment algorithm is proposed for heterogeneous processors with the task graphs limited to a small number of tasks. A dynamic task mapping scheme is proposed for NoC-based MPSoCs in [14], aiming to improve the performance by minimizing the channel load. The incremental mapping algorithms proposed for 2-D NoCs [5, 6] may be helpful for 3-D NoCs. The general idea of these run-time mapping algorithms is to find a convex tile region first, followed by mapping the incoming application to the convex region. The convexity of the tile region helps to reduce communication within the region and the impact on future applications as the remaining tiles always form a continuous shape.

Run-time mapping algorithms in traditional parallel and distributed systems [15] do not suit for the on-chip application mapping as they do not consider on-chip power consumption.

3 Problem Formulation

In this paper, our study is focused on run-time mapping for 3-D mesh-based NoCs considering the regularity and advantage of the topology [8]. Fig. 1 shows the 3-D mesh-based NoC architecture, which is composed of N layers of IP cores and routers.

Each IP core is indexed by (x, y, z) where $0 \le x \le M_x$, $0 \le y \le M_y$ and $0 \le z \le N$-1, (each layer is composed of a M_x x M_y mesh). At each layer, each IP core is on a single physical plane and connected to its router through a horizontal link. Due to its advantages, the 3-D DimDe router [3] is adopted here. As shown in Fig. 1, each router still has five horizontal ports, i.e., E, W, N, S and local. The guided flit queuing methods [3] are used to decompose the traffic into X, Y, and Z dimensions. The 5x5 crossbar in a 2-D mesh is split into two 4x2 crossbars to reduce the router complexity. Each vertical link has one input connection from its associated path set (PS) MUX and three output connections, one to the East-West crossbar, the North-South crossbar and the local port, respectively. Assume that the wire length of vertical links between adjacent layers is much shorter than the wire length of horizontal links between adjacent routers. The deterministic XYZ routing is assumed. A tile is defined to be an IP core with the corresponding part of the router on the same layer.

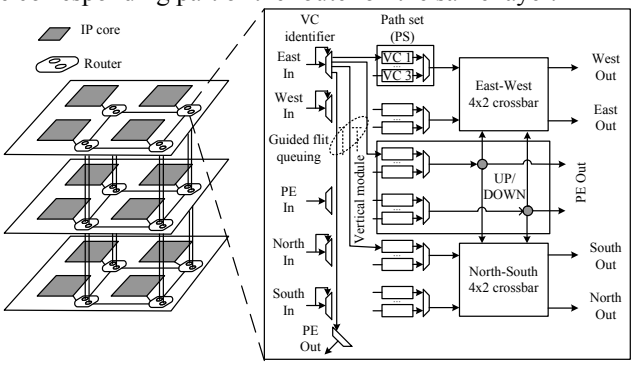


Fig. 1. The 3-D NoC architecture [3]

3.1 Communication Power Model

The communication energy needed for sending a flit from a source tile to a destination tile is composed of two parts, the energy consumed at the routers and the energy consumed on the interconnection links. The average energy consumption of sending one bit of data from tile t_i to tile t_j can be represented as

$$E^{t_i,t_j}_{bit} = E_R + E_L \tag{1}$$

where E_R and E_L are the energy consumed by the routers and links in the communication path from tile t_i to tile t_j. To be more specific,

$$E^{t_i,t_j}_{bit} = \eta E_{Rbit} + \eta_H E_{LHbit} + \eta_V E_{LVbit} \tag{2}$$

where E_{Rbit} is the energy consumed when transporting a flit at the router, η is the number of routers traversed from tile t_i to tile t_j, η_H and η_V are the number of horizontal and vertical links in the communication path, and E_{LHbit} and E_{LVbit} are the energy consumed on the horizontal and vertical links between tiles t_i and t_j. Following the wire model given in [7], $E_{Lbit} = dV^2C_{wire}/2$, where d is the length of the link, V is the supply voltage and C_{wire} is the wire capacitance.

3.2 Application and Architecture Model

Each incoming application is represented by its communication trace graph defined below.

Definition 1 A *Communication Trace Graph* (CTG) $G=(P, E)$ is an undirected graph, where a vertex/node $p_k \in P$ represents a task, and an edge $e_i=(p_j, p_k) \in E$ represents the communication trace between vertices p_j and p_k.

- For edge e_i, $\omega(e_i)$ defines the communication bandwidth request between vertices p_j and p_k 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.

The 3-D NoC architecture is modeled by the architecture characteristic graph.

Definition 2 An *Architecture Characterization Graph* (ACG) $G'=(T, L)$ is an undirected graph, where each vertex $t_i \in T$ represents a tile and each edge $l_i \in L = (t_j, t_k)$ represents the link between adjacent tiles t_j and t_k. N is the number of NoC layers. For link l_i

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

In this paper, we focus on NoC architectures which have $bw(l_i)=B$, $c(l_i)=C_V$ if l_i is a vertical link, $c(l_i)=C_H$ if l_i is a horizontal link for each $l_i \in L$, where B, C_V and C_H are constants. $h_{t_j,t_k\ j}$ is the set of links forming one of the shortest paths from tile t_j to tile t_k ($h_{t_j,t_k} \subseteq L$). We also assume that the IP core with index (0, 0, 0) manages the whole mapping process.

3.3 Problem Description

The 3-D NoC run-time mapping problem is described as: Given the CTG of the incoming application and the ACG of the current 3-D NoC system, find a mapping function $M: P \rightarrow T$, with the objective of minimizing the communication power, i.e.,

$$Min\Big(\sum_{\substack{i=0 \\ e_i=(p_j,p_k)\in E}}^{|E|-1} \omega(e_i) \cdot \sum_{\substack{m=0 \\ l_k \in h_{M(p_j),M(p_k)}}}^{|h_{M(p_j),M(p_k)}|} c(l_m)\Big) \tag{3}$$

satisfying

$$\forall p_j \in P, M(p_j) \in T, \tag{4}$$

$$\forall p_j, p_k \in P \text{ and } p_j \neq p_k, M(p_j) \neq M(p_k), \tag{5}$$

$$\forall\ l_m, B \geq \sum_{e_i=(p_j,p_k)} \omega(e_i) \cdot f(l_m, h_{M(p_j),M(p_k)}), \tag{6}$$

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

Conditions given by (4) and (5) ensure that each task should be mapped exactly to one tile and no tile can host more than one task. The inequities given in (6) specify the bandwidth constraint for every link.

4 Motivating example and Problem Decomposition

4.1 Motivating Example

A motivating example of mapping a newly incoming application to a 3-D mesh-based NoC (with 3x16 tiles) is given in Fig. 2 to illustrate the composition of the problem. Fig. 2(a) gives the CTG of an example application with 16 tasks. The black tiles in Fig. 2(b) are reserved for other applications already running in the system. Before mapping, as in run-time incremental mapping for 2-D NoCs [5, 6], a region of available tiles shall be selected. Intuitively, a cuboid region is preferred to minimize the impact to future applications. Fig. 2(b) shows such a region of 18 tiles (in grey color). Next, in the mapping process, the tasks which have high communication requests should be considered first [10]. Based on the definitions in Section 3, these tasks shall be allocated to the sub-regions in different layers such that the communications with high bandwidth requests can utilize vertical links. Fig. 2(c) shows an allocation result of the tasks which have high communication requests. Last, all tasks are mapped to the tiles in the sub-regions as shown in Fig. 2(d).

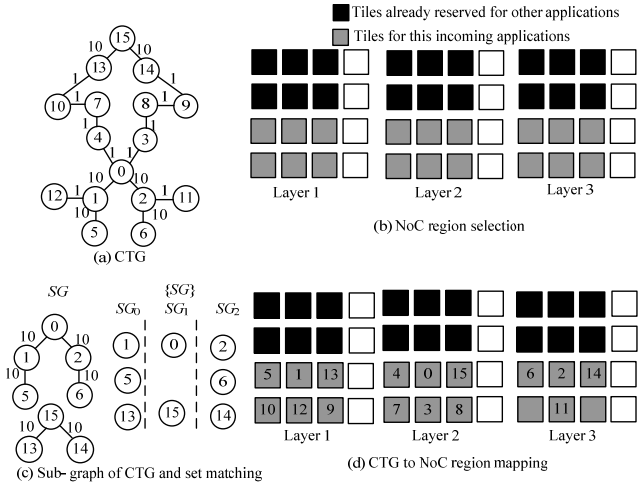


Fig. 2. Illustration of the whole problem.

4.2 A Mapping Flow by Problem Decomposition

Based on the observations from this example, the run-time mapping problem for 3-D NoCs is divided into three sub-problems as shown in Fig. 3. The three sub-problems are described below.

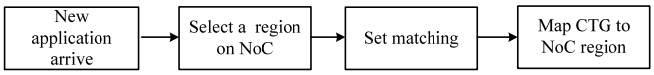


Fig. 3. The mapping process.

(4.1.1) NoC Region Selection

First, a region is selected to map each incoming application. This region is composed of N sub-regions, one on each layer. To reduce the impact on future applications, these sub-regions should be rectangle in shape and the size of each of the sub-regions shall be made as close to each other as possible.

Given the CTG of the incoming application, the ACG of the 3-D NoC, and the current system behavior (i.e. the used and unused tiles), this sub-problem is to find a cuboid region $SR=\{SR_0, \ldots, SR_{N-1}\}$, where SR_w represents the sub-region of unallocated tiles on layer w, with the objective of:

$$Min\ (|\ SR_w| \cdot N - |G|), \text{ for each } w=0, \ldots, N-1 \tag{7}$$

satisfying,

$$|SR_w| \cdot N \geq |G|, \tag{8}$$

$$|SR_w| = |SR_{w+1}| \text{ for } w = 0, \ldots, N-2, \tag{9}$$

Here $|G|$ gives the number of tasks in G and $|SR_w|$ gives the number of tiles in set SR_w.

(P2) Set Matching

The basic idea of set matching is to find the layers to be mapped for those tasks which have high bandwidth (BW) communication requests. When both vertical and horizontal links are available, according to the assumption that vertical links tend to be shorter than horizontal links, vertical links are thus preferred. That is, communication edges with high BW requests shall be allocated to vertical links if possible in order to reduce delay and power consumption.

Assume the edges in G are sorted in a non-increasing order in terms of their communication bandwidth request (i.e., $\omega(e_i)$). Let $SG \subset G$ be a sub-graph formed with the first α% edges in the sorted edge list [10]. The objective is to allocate the vertices in SG into N subsets to be mapped to the corresponding N sub-regions found in (P1) such that the vertical links are used as much as possible. This sub-problem is formally defined as follows.

Given the sub-graph SG of the CTG and the sub-regions $SR_0, \ldots, SR_{N-1}$ obtained from region selection step described in (P1), allocate the vertices in SG into N sets $SG_0, \ldots, SG_{N-1}$, assuming vertical links are preferred, with the objective of:

$$Max \sum_{\substack{e_i=(p_j, p_k) \\ p_j \in SG_u \\ p_k \in SG_v \\ u \neq v}} \omega(e_i), \tag{10}$$

satisfying,

$$SG_u \cap SG_v = \varnothing, \text{ for } u \neq v \tag{11}$$

$$|SG_u| \le |SR_w|, \text{ for each } u=0,\ldots, N\text{-}1 \tag{12}$$

(P3) CTG to NoC Region Mapping

With the sub-regions and sets obtained in (P1) and (P2), the application is then mapped to minimize the communication power.

Given the CTG of the incoming application, N sets of the vertices $SG_0, \ldots, SG_{N-1}$ and a set of sub- regions $SR=\{SR_0, \ldots, SR_{N-1}\}$, find a mapping function M: $P \rightarrow SR$ with the objective:

$$Min \ \Big(\sum_{\substack{i=0 \\ e_i=(p_j,p_k)\in E}}^{|E|} \omega(e_i)\cdot[MD_v(M(p_j),M(p_k))\cdot C_V + MD_H(M(p_j),M(p_k))\cdot C_H]\Big), \tag{13}$$

satisfying

$$\forall p_j \in P,\ M(p_j) \in T, \tag{14}$$

$$\forall p_j, p_k \in P \text{ and } p_j \ne p_k, M(p_j) \ne M(p_k), \tag{15}$$

$$\forall\ l_m, B \ge \sum_{e_i=(p_j,p_k)} \omega(e_i)\cdot f(l_m, h_{M(p_j),M(p_k)}), \tag{16}$$

where $f(l_m, h_{M(p_j),M(p_k)}) = \begin{cases} 1 \text{ if } l_m \in h_{M(p_j),M(p_k)} \\ 0 \text{ if } l_m \notin h_{M(p_j),M(p_k)} \end{cases}$, $MD_H()$ and $MD_V()$ denote the horizontal and vertical Manhattan distance between two tiles, respectively. Conditions given by (14) and (15) ensure that each task should be mapped exactly to one tile and no tile can host more than one task. The inequities given in (16) specify the bandwidth constraint for every link.

5 Algorithm Description

For each sub-problem listed in Section 4, an algorithm is introduced in each of following subsections. Before discussing these algorithms, the following assumption is made. All the edges of an incoming application's CTG are sorted in a decreasing order by the edge weight $\omega(e_i)$. The sorted edge list is represented as SE.

5.1 NoC Region Selection (P1)

As discussed in Section 4, a cuboid of NoC tiles shall be first found. This problem can be further divided into two small problems. First, given the incoming application's CTG G, a window with a size of size $l \cdot w$ should be found such that $N \cdot l \cdot w \ge |G|$ and $N \cdot l \cdot w$ - $|G|$ is minimized. That is, the shape and size of sub-region should be chosen to minimize fragmentation. Second, after finding the $l \cdot w$ matrix, unallocated tiles which form the $N \cdot l \cdot w$ cuboid should be found (these tile regions are called NoC regions). A window sweeping process is used as in Fig. 4(a).

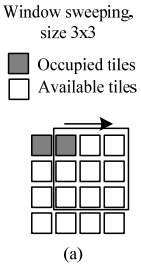
A window of size $l \cdot w$ starts from tile (0, 0, 1) and moves at the first layer. The window sweeps the whole layer until all of the tiles inside the window are found not allocated yet.

An example of applying this algorithm is shown in Fig. 2(b). The CTG shown in Fig. 2(a) has 16 tasks. Thus, a 3x2x3 cuboid is selected based on the window sweeping process (Fig. 2(b)).

The NoC region selection algorithm is shown in Fig. 4(b). The complexity of the algorithm is $xy+(|T|/|N|\cdot x\cdot y) = O(|T|/|N|)^2$.

```
NoC_region_selection(G, Q, G')
Input: (1) G: The CTG of the incoming application
       (2) Q: current unallocated tiles in NoC
       (3) G': The ACG of the NoC architecture
Output: (1) SR: a cuboid tile region allocated for the incoming application
Function: find the sub-regions of tiles for the incoming application
Procedure body:
{
  var: size_each_layer = ⌈|G|/N⌉;
  // 1. find the window size
  for (l=1; l< size_each_layer/2;l++)
        for (w=1; w< size_each_layer/2;w++)
          if (l·w > size_each_layer){
            break from the l-indexed loop;
          }
  // 2. find an N·l·w cuboid in Q for G,
  // use window sweeping at the bottom layer
  for (each tile ti in Q0) {
    // suppose Q0 is the free tile region in the bottom layer of 3-D NoC
    // suppose the coordinate of ti is (i, j, 0)
    // let R be the the rectangle tile region whose top left and bottom right ranges are (i, j, 0) and (i+l, j+w, 0)
    // let R' be the rectangle tile region whose top left and bottom right ranges are (i, j, 0) and (i+w, j+l, 0)
    // m = 0,1,…N-1
   if (all the tiles in R are available) {
         SRm = R;
         break;
   }
   else if (all the tiles in R' are available) {
         SRm = R';
         break;
   }
  }
  Q = Q -SR
}
```

(b)



(a)

Fig. 4. The NoC region selection algorithm.

5.2 Set Matching (P2)

In the second step, the sub-graph of the CTG, i.e. *SG*, which includes the tasks with high communication BW requests should be allocated into the *N* sets, one for each layer. The sub-graph *SG* is formed by the first 50% edges (denoted as *SE'*) in *SE*, Vertical links are preferred for high BW communication edges. Hence, edges with larger weight will be first allocated to vertical links if possible. However, there is only one vertical link between the two tiles (x, y, z) and $(x, y, z+1)$, $z = 0, \ldots, N-2$. Thus, if a task is allocated to tile (x, y, z) and has two neighbors in layer $z+1$, it may result in higher power consumption than mapping one of the neighbors to the same layer z. Fig. 5 shows such an example. Given the CTG shown in Fig. 5(a), if tasks 1 and 2 are mapped to layer 1 (Fig. 5(b)), the energy cost is $10\cdot(C_V+C_V+C_H)$ suppose the link energy cost are C_V and C_H for vertical and horizontal links, respectively. If task 2 is mapped to the same layer of task 0 (Fig. 5(c)), the energy cost is $10\cdot(C_V+C_H)$ which is lower than that of Fig. 5(b). Thus, an observation is made that when vertical links are preferred, a task can be allocated to the same layer of its neighbor if the neighbor already has edges allocated to adjacent layers.

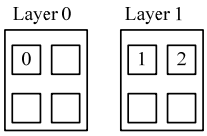
The algorithm works as follows. For each edge (p_j, p_k) in the sorted edge list *SE'*,

- If neither of the two tasks is allocated, the tasks are checked to see whether one of them has degree in *SG* greater than 1. If so, the task with a larger degree in *SG* is allocated to SG_u with ($0<u<N$-1), i.e., to a layer whose tiles have two vertical links, up and down. Otherwise, the tasks are allocated to SG_u and SG_v with $u \neq v$ and $|u\text{-}v|$ minimized.
- If one of the tasks is allocated, e.g., p_j is allocated to SG_u, the unallocated task p_k is allocated to SG_v ($u \neq v$) such that $|u\text{-}v|$ is minimized, given that p_j has no neighbor in SG_v. Otherwise, p_k is allocated to SG_u.

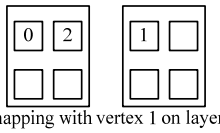
Fig. 6 shows an example of the set matching processing based on the CTG given in Fig. 2(a). The sub-graph *SG* of this CTG is given in Fig. 6(a) and the size of region on each layer is 6. The sorted edge list of the sub-graph is {(0, 1), (0, 2), (1, 5), (2, 6), (13, 15), (14, 15)}. Fig. 6(b) shows that tasks 0 and 1 are first allocated to SG_1 and SG_0 as task 0 has a degree of 2 in Fig. 6(a). When allocating edge (0, 2), task 2 is allocated to SG_0 as task 0's neighbor task 1 is already allocated to SG_1 (Fig. 6(c)). When allocating edge (1, 5), task 5 is allocated to SG_0 as task 1's neighbor task 0 is already allocated to SG_1. Similarly, task 6 is allocated to SG_2 as in Fig. 6(d). Tasks 13, 14 and 15 are allocated similarly as in Fig. 6(e) and (f). The final result of the set matching problem for the vertices in *SG* is shown in Fig. 6(f).

The set matching algorithm is shown in Fig. 5(d). The complexity of the algorithm is $O(|E||N|^2)$.

(a) sub- graph of an application



(b) a mapping with vertices 1 and 2 on layer 1



(c) a mapping with vertex 1 on layer 1 and vertex 2 on layer 0

```
Set_matching(SG, SE', SR)
Input: (1) SG: the sub-graph of the incoming application's CTG
       (2) SE': the sorted edge list of the first ζ% edge
       (3) SR: the set of sub-regions for the incoming application
Output: (1) SG_0, …, SG_{N-1}: the N sets containing the vertices in SG
Function: allocate the vertices in SG into the N sub-regions
Procedure body:
{
  for (each edge e_i =( v_j , v_k) ∈ SE' in sorted order) {
    if (neither v_j nor v_k is allocated to any sub-region) {
    // suppose v_j is the vertex with high degree in SG_u, u=0,…, N-1
      if (v_j has more than 1 neighbors in SE' and | SG_{⌈N/2⌉} |< |SR_0|)
        add v_j to SG_{⌈N/2⌉};
      else
        add v_j to SG_u such that | SG_u |< |SR_0|, for u=0,…N-1;
    }
    else if (one of the two vertices are allocated) {
    // suppose v_j is allocated to SG_u
      if (v_j has neighbors in SE' and allocated to SG_{u-1} or SG_{u+1}
          and |SG_u|<|SG|/N)
        add v_k to SG_u;
      else
        add v_k to SG_m such that | SG_m | < |SR_0| with m ≠ u,
        and |u-m| minimized and v_j has no neighbor allocated to SG_m;
    }
  }
}
```

(d) the set matching algorithm

Fig. 5. The set matching algorithm (P2).

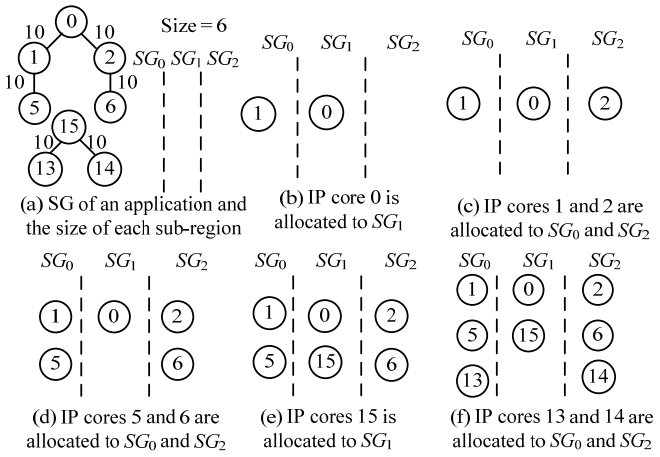


Fig. 6. An example of set matching when vertical links are preferred.

5.3 CTG to NoC Region Mapping (P3)

After finding the set of tile regions *SR* and the allocation of the vertices in *SG* to the *N* set SR_0, …, SR_{N-1}, all of the vertices in CTG should be mapped to *SR*. In this algorithm, a metric *Dist*() is used to find the weighted distance of two tiles. The *Dist*() metric is defined as: $Dist(t_a,t_b)=|x_a-x_b|C_H+|y_a-y_b|C_H+|z_a-z_b|C_V$, where (x_a,y_a,z_a) and (x_b,y_b,z_b) represent the coordinates of tiles t_a and t_b, respectively. As discussed in Section 3, *Dist*() reflects the power consumption of each communication edge after mapping the two tasks. The algorithm checks for each edge in *SE* of CTG and maps the tasks to minimize their distance. It works as follows. For each edge (p_j, p_k),

- If neither of the two tasks p_j, p_k are mapped. There are three cases to consider

  1) Both tasks are already allocated to the *N* sets SR_0, …, SR_{N-1}, e.g., p_j allocated to SG_u and p_k allocated to SG_v ($0 \le u, v \le N-1$). If p_j has high degree (>2) in *SG*, find a tile in SR_u with large number of free neighbor tiles. Otherwise, find an unused tile in SR_u for p_j. Then find a tile for p_k in SR_v to minimize the *Dist*() metric.

  2) One of the two tasks already allocated to the *N* sets, e.g., p_j allocated to SG_u. If p_j has high degree (>2) in *SG*, find a tile in SG_u with large number of free neighbor tiles. Otherwise, find an arbitrary tile in SG_u for p_j. Then find a tile for p_k in *SR* to minimize the *Dist*() metric.

  3) Neither of the two tasks allocated to the *N* sets yet. If p_j has high degree (>2) in *SG*, find a tile in *SR* with large number of free neighbor tiles. Otherwise, find an arbitrary tile in *SR* for p_j. Then find a tile for p_k in *SR* to minimize the *Dist*() metric.

- If one of the two tasks is mapped, e.g., p_j is mapped to tile *t*. If the unmapped task p_k is allocated to SG_u, find a tile in SR_u for p_k to minimize the *Dist*() metric. Otherwise, find a tile for p_k in *SR* to minimize the *Dist*() metric.

The CTG to NoC region mapping algorithm is shown in Fig. 7. The complexity of the CTG to NoC region mapping algorithm (P3) is $O(|E||T|^2)$. The total complexity of the run-time incremental mapping algorithm (combining the algorithms introduced in section 5.1 to 5.3) is $O(|E||T|^2)$.

Assume the vertical links are preferred for communications with high BW requests. Fig. 2(d) shows the mapping result given the CTG in Fig. 2(a) based on the results of NoC region selection (Fig. 2(b)) and set matching (Fig. 2 (c)).

```
CTG_to_NoC_mapping (G, SE, SR)
Input: (1) G: the CTG of the incoming application
       (2) SE: the sorted edge list of the CTG
       (3) SR: the set of sub-regions for the incoming application
       (4) SG_0, …, SG_{N-1}: the N sets containing the vertices in SG
Output: (1) MAP: mapping table for each IP core
Function: map each IP core to a tile in SR
Procedure body:
{
  for (each edge e_i =( p_j, p_k)∈ SE) {
    if (neither p_j nor p_k are mapped) {
      if (p_j ∈ SG_u and p_k ∈ SG_v) { // u,v=0,…,N-1
          MAP[p_j] = t_a s.t. t_a ∈ SR_u;
          // if p_j has more than two neighbors in SG,
          // find a t_a have more than two free neighbor tiles
          MAP[p_k] = t_b s.t. t_b ∈ SR_v and Dist(t_a, t_b) is minimized;
      }
      else if (suppose p_j ∈ SG_u) { // u =0,…,N-1
          MAP[p_j] = t_c s.t. t_c ∈ SR_u;
          // if p_j has more than two neighbors in SG,
          // t_c should also have more than two free neighbor tiles
          MAP[p_k] = t_d s.t. t_d ∈ SR and Dist(t_c, t_d) is minimized;
      }
      else {
          MAP[p_j] = t_e s.t. t_e ∈ SR;
          // if p_j has more than two neighbors in SG,
          // t_e should also has more than two free neighbor tiles
          MAP[p_k] = t_f s.t. t_d ∈ SR and Dist(t_e, t_f) is minimized;
      }
    }
    else if (only one IP core is mapped){
        // suppose p_j is mapped to tile t
        if (p_k ∈ SG_v) { // v=0,…,N-1
          MAP[p_k] = t_g s.t. t_g ∈ SR_v and Dist(t, t_g) is minimized;
        }
        else {
          MAP[p_k] = t_h s.t. Dist(t, t_h) is minimized;
        }
    }
  }
}
```

Fig. 7. The CTG to NoC region mapping algorithm.

6 Performance Evaluation

The proposed algorithms have been evaluated through extensive experiments. Two sets of experiments are performed. First, single random applications are mapped to a 3-D NoC architecture. In this set of experiments, our algorithm is compared against the branch and bound (BNB) algorithm in terms of the solution quality. BNB tries to exhaustively enumerate all possible mapping results where mapping results with higher power consumption are discarded by estimating upper and lower bounds. Second, multiple applications (including random applications generated by TGFF [16] and applications from E3S [17]) are incrementally mapped to a 3-D NoC architecture. In this set of experiments, our algorithm is compared against a random mapping algorithm and a simple heuristic algorithm, namely large communication first (LCF).

The power model is adopted from the DimDe router [3]. The network is assumed to be a 6x6x3 3-D mesh. The flit size is set to be 128-bit. To study the power consumption trend, vertical and horizontal links with different lengths are chosen. The wire parameters are adopted from [1], e.g., the unit wire capacitances (C_{wire}) for vertical and horizontal links are 600fF/mm and 332fF/mm, respectively. Table I lists the wire length combinations evaluated where vertical links are preferred over horizontal links. The Noxim simulator [18] is used as the NoC simulator.

TABLE 1. Different combinations of vertical and horizontal link length

Combinations	Length of vertical links between adjacent layers	Length of horizontal links between adjacent routers
I	60um	1mm
II	90um	1mm
III	120um	1mm
IV	60um	0.5mm
V	90um	0.5mm
VI	120um	0.5mm

6.1 Mapping Single Application to 3-D NoC

First, we show the single application mapping results of our algorithm compared to that of BNB. Two types of applications are evaluated: random applications and real applications. For random applications, TGFF is used to generate the CTGs. The numbers of tasks range from 12 to 20. The communication BW requests vary from 1 to 10. The tile regions selected for our algorithm and BNB are the same. The running time of BNB is more than one hour while the running time of our algorithm is within 1 sec. From Fig. 8, we can see that the increase in power consumption of our algorithm over BNB is within 11%.

Fig. 8 shows that, the results from our algorithm get closer to that of BNB when the energy cost ratio of horizontal links over vertical links (C_H:C_V) are smaller. The ratio of C_H:C_V is the largest for combination I and smallest for combination VI. The reason is due to the fact that, as the ratio of C_H:C_V becomes smaller, the difference in vertical and horizontal communications is reduced. The power consumption of horizontal links in the mapping result produced by our algorithm has less impact on the overall communication power consumption.

6.2 Mapping Multiple Applications Incrementally to 3-D NoC

In this section, our algorithm is compared against a random mapping (RM) and large communication first (LCF) with multiple applications. Due to the complexity of BNB, it is not used for runtime mapping. In RM and LCF, the NoC region selection step is the same as in our algorithm. RM maps the tasks to the tiles in the selected region randomly. LCF sorts the edges in the non-increasing order of BW request. Then the terminal vertices of the communication edge with higher bandwidth request are mapped to links with smaller $c(l)$ values. Both random and real applications are evaluated.

First, random applications are simulated with their CTGs generated from TGFF [16]. The numbers of tasks in these CTGs vary from 12 to 20. The communication BW requests vary from 1 to 10. For each combination in Table I, five experiments are

performed for each number of tasks. In each experiment, 10 applications are mapped to a 6x6x3 network. The running time of each application is varied from 1 to 10 time units. Fig. 9 shows the normalized power saving of our algorithm over RM and LCF. Fig. 9(a) shows that our algorithm can save up to over 50% power compared to RM and up to 20% over LCF.

Real application CTGs, including auto-industry, consumer, networking, and office automation, are generated from E3S [17]. Again, for each combination in Table I, five experiments are conducted. Each experiment is composed of 10 applications which are mapped to a 6x6x3 network incrementally. Fig. 9(b) shows that our algorithm can save up to about 30~40% power compared to RM and up to 8~17% compared to LCF. The power saving in Fig. 9(b) is smaller than that in Fig. 9(a) since the CTG structures of the real applications from E3S are highly regular, e.g., mostly forming a chain-type structure. For such chain-type CTGs, RM and LCF will result in better results compared to non-regular structures.

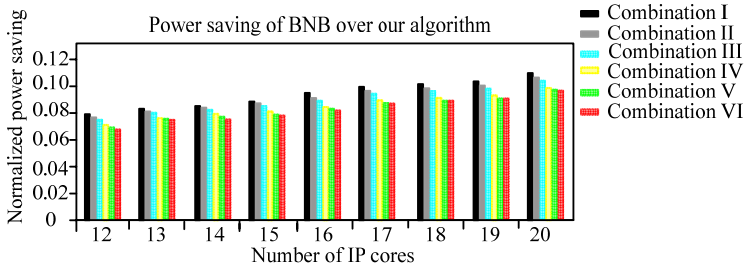


Fig. 8. Normalized power saving of BNB over our algorithm (normalized over the power consumtiopn of the results of BNB)

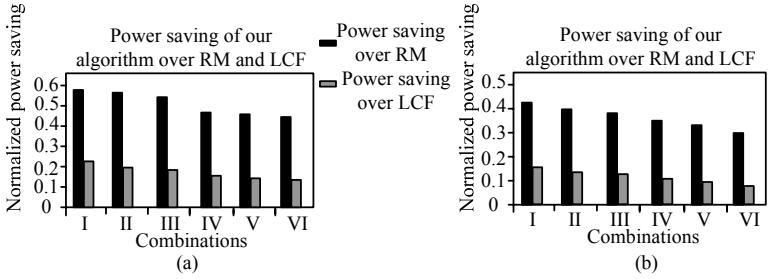


Fig. 9. Normalized power consumption of RM and LCF over our algorithm on TGFF (a) and E3S (b). Power consumption are normalized over the power consumtiopn of the results of our algorithm.

7 Conclusion

In this paper, we have proposed a power-aware run-time incremental mapping algorithm for 3-D NoCs with the objective of minimizing the communication power for each incoming application. In addition, the proposed algorithm attempts to reduce the fragmentation impact on future applications. The proposed algorithm is composed of three steps: 1) *NoC region selection* which selects a cuboid region to reduce the impact to future applications, 2) *set matching* which allocates the application graph to the sub-regions in different layers such that the vertical links are used as much as

possible, 3) *CTG to NoC region mapping* which maps the tasks to the tiles in different sub-regions with minimized total communication power. Our experiment results have confirmed that the proposed algorithm is over four orders of magnitude run time efficient than BNB when mapping a single application. It has been shown that our algorithm can also achieve 50% power saving compared to random mapping and 20% compared to a simple heuristic when mapping multiple applications incrementally.

References

1. V. F. Pavlidis and E. G. Friedman: 3-D topologies for networks-on-chip. IEEE Trans. Very Large Scale Integration Systems, vol. 15, no. 10, pp. 1081-1090, (2007).
2. W. R. Davis, J. Wilson, S. Mick, J. Xu, H. Hua, C. Mineo, A. M. Sule, M. Steer, and P. D. Franzon: Demystifying 3D ICs: the pros and cons of going vertical. IEEE Design and Test of Computers, vol. 22, no. 6, pp. 498-511, (2005).
3. J. Kim, C. Nicopoulos, D. Park, R. Das, Y. Xie, V. Narayanan, M. S. Yousif, and C. R. Das: A novel dimensionally-decomposed router for on-chip communication in 3D architectures. In: Int'l Symp. Computer Architecture, vol. 35, pp. 138-149, (2007).
4. C. Addo-Quaye: Thermal-aware mapping and placement for 3-D NoC designs. In: IEEE Int'l SoC Conf., pp. 25-28, (2005).
5. C. L. Chou and R. Marculescu: Run-time task allocation considering user behavior in embedded multiprocessor networks-on-chip. IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 29, no. 1, pp. 78 - 91, (2010).
6. C. L. Chou, U. Y. Ogras, and R. Marculescu: Energy-and performance-aware incremental mapping for networks on chip with multiple voltage levels. IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 27, no. 10, pp. 1866-1879, (2008).
7. H. Matsutani, M. Koibuchi, and H. Amano: Tightly-coupled multi-layer topologies for 3-D NoCs, In: Int'l Conf. Parallel Processing, pp. 75-85, (2007).
8. B. S. Feero and P. P. Pande: Networks-on-Chip in a three-dimensional environment: a performance evaluation. IEEE Trans. Computers, vol. 58, no. 1, pp. 32-45, (2009).
9. D. Park, S. Eachempati, R. Das, A. K. Mishra, Y. Xie, N. Vijaykrishnan, and C. R. Das: MIRA: a multi-layered on-chip interconnect router architecture. In: Int'l Symp. Computer Architecture, pp. 251-261, (2008).
10. 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. ACM Trans. Architecture and Code Optimization, vol. 7, no. 1, pp. 1-31, (2009).
11. A. H. Land and A. G. Doig: An automatic method for solving discrete programming problems. Econometrica, vol. 28, pp. 497-520, (1960).
12. 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, no. 4, pp. 551-562, (2005).
13. L. T. Smit, G. J. M. Smit, J. L. Hurink, H. Broersma, D. Paulusma, and P. T. Wolkotte: Run-time assignment of tasks to multiple heterogeneous processors. In: Progress Embedded System Symp., pp. 185-192, (2004).
14. E. Carvalho, N. Calazans, and F. Moraes: Heuristics for dynamic task mapping in NoC-based heterogeneous MPSoCs. In: IEEE/IFIP Workshop Rapid System Prototyping, pp. 34-40, (2007).
15. V. Lo, K. J. Windisch, W. Liu, and B. Nitzberg: Noncontiguous processor allocation algorithms for mesh-connected multicomputers. IEEE Trans. Parallel and Distributed Systems, vol. 8, no. 7, pp. 712-726, (1997).
16. TGFF: task graphs for free. [Online]. Available: http://ziyang.eecs.umich.edu/~dickrp/tgff/.
17. R. Dick: Embedded system synthesis benchmarks suite(E3S). [Online]. Available: http://ziyang.eecs.umich.edu/~dickrp/e3s/.(2002)
18. Noxim. [Online]. Available: http://noxim.sourceforge.net/.