On Runtime Communication- and Thermal-aware Application Mapping in 3D NoC

Bing Li
South China University of Technology, China
l.b07@mail.scut.edu.cn

Xiaohang Wang
South China University of Technology, China
xiaohangwang@scut.edu.cn

Amit Kumar Singh
Department of Electronics and Computer Science,
University of Southampton, UK
a.k.singh@soton.ac.uk

Terrence Mak
Department of Electronics and Computer Science,
University of Southampton, UK
Shenzhen Institute of Advanced Technology and
Guangzhou Institute of Advanced Technology, CAS, China

ABSTRACT

Many-core systems connected by 3D Network-on-Chips (NoC) are emerging as a promising computation engine for systems like cloud computing servers, big data systems, *etc.* Mapping applications at runtime to 3D NoCs is the key to maintain high throughput of the overall chip under a thermal/power constraint. However, the goals of optimizing both the communication latency and chip peak temperature are contradicting due to several reasons. Firstly, exploiting the vertical TSV links can accelerate communications, while low peak temperature prefers that the tasks to be mapped closer to the heat sink, instead of using the vertical links. Secondly, mapping tasks in close proximity can reduce communication latency, but at the cost of poor heat dissipation. To address these issues, in this paper, we propose an efficient runtime mapping algorithm to reduce both communication latency and overall application running time under thermal constraint. In essence, this algorithm first selects a 3D cuboid core region of a specific shape for each incoming application by setting the region's number of occupied vertical layers and its distance to the heat sink, in order to optimize its communication performance and peak temperature. Next, the exact locations of the core regions in the chip are determined, followed by a task-to-core mapping. The experimental results have confirmed that, compared to two recently proposed runtime mapping algorithms, our proposed approach can reduce the total running time by up to 48% and communication cost by up to 44%, with a low runtime overhead.

CCS CONCEPTS

• **Computer systems organization → Multicore architectures**;

KEYWORDS

3D NoC, Application mapping, Thermal management, On-chip Resource management

ACM Reference format:
Bing Li, Xiaohang Wang, Amit Kumar Singh, and Terrence Mak. 2017. On Runtime Communication- and Thermal-aware Application Mapping in 3D NoC. In *Proceedings of NOCS '17, Seoul, Republic of Korea, Oct. 19–20, 2017,* 8 pages.
https://doi.org/10.1145/3130218.3130228

NOCS '17, Oct. 19–20, 2017, Seoul, Republic of Korea

ACM ISBN 978-1-4503-4984-0/17/10…$15.00
https://doi.org/10.1145/3130218.3130228

1 INTRODUCTION

Many-core systems have been widely used as an engine to provide sufficient computation power in cloud computing servers, big data systems, *etc.* In these systems, multiple applications with various workload characteristics arrive and leave the system at runtime. Mapping tasks to cores online is the key to improve system performance.

3-Dimensional (3D) integration can improve the system integration and reduce global wire length. Through-Silicon-Via (TSV) is one of the popular approaches among various 3D integration techniques [13, 16]. 3D networks-on-chip (NoC) adopting the TSV technique has lower network latency and power consumption, and higher bandwidth [18, 19]. However, as more dies stacked vertically, power density (W/m^2) increases, and the length of heat conduction path increases, resulting in higher propagation delay and higher leakage power [4]. The major challenge of runtime application mapping in 3D NoC is to achieve the contradicting goals of optimizing communication and temperature, which requires to address the following two aspects:

1) Whether to exploit the vertical links or not leads to different communication and temperature behaviors. Lower communication latency requires that the tasks of an application to exploit the TSV connections as much as possible. That is, the tasks with high communication volume should be mapped crossing multiple vertically adjacent layers in the same column, as TSVs provide high bandwidth and shorter distance as in Fig. 1(a). On the other hand, lower temperature requires that the tasks to be mapped to the layer close to the heat sink as in Fig. 1(b), instead of crossing multiple layers.

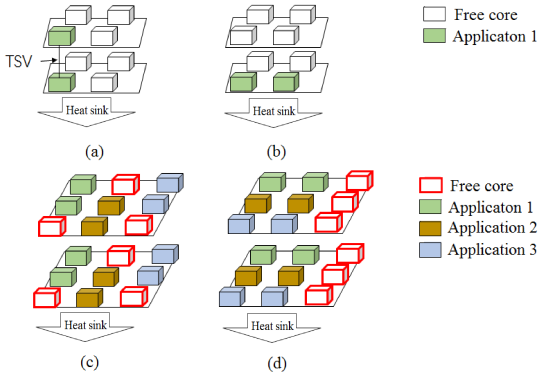

Figure 1: (a) Mapping tasks to exploit vertical wire for higher communication efficiency. (b) Mapping tasks close to heat sink for lower temperature. (c) Free cores scattered. (d) Free cores packed in close proximity.

2) For an application with some running applications, whether to map tasks in scattered or close proximity might lead to fragmentation or high temperature. A phenomenon called "fragmentation" often occurs causing free cores scattered (not forming a contiguous region). Fragmentation increases communication distance and latency for tasks of incoming application as they are mapped to noncontiguous free cores regions [17], as in Fig. 1(c). Mapping tasks in close proximity alleviates fragmentations. However, if we take thermal effect into consideration, mapping tasks in close proximity accumulates heat faster and tends to have high peak temperature as in Fig. 1(d), where the peak temperature is 2℃ higher than that in Fig. 1(c)[1].

In this paper, we proposed an algorithm to address the above challenge, in order to optimize the communication and computation performances under a thermal constraint in 3D NoC systems. The algorithm has two steps. First, a 3D cuboid core region of a specific shape is selected for each application. Second, the exact locations of the core regions in the chip are determined, followed by a task-to-core mapping.

The rest of paper is organized as follows. Related work is reviewed in Section 2. The problem is formulated in Section 3. The proposed runtime mapping algorithm is detailed in Section 4. Experimental results are evaluated in Section 5. Finally, Section 6 concludes the paper.

2 RELATED WORK

Existing runtime application mapping algorithms in NoC can be classified into 3 categories: 1) communication, 2) temperature, and 3) both communication and temperature.

Mapping algorithms in the first category focus on communication optimization, improving system throughput by reducing network latency [2, 7–9, 28]. Most of the algorithms in this category target 2D NoC systems as in [2, 7–9]. These approaches map communicating tasks to cores close to each other so that communication latency and power are reduced. For example, in [8], Mohammad *el al.* proposed a stochastic hill climbing algorithm that starts from a first node and maps the tasks to a set of nodes forming a contiguous region around it. A few of them target the 3D NoC system as in [28]. In [28], Ziaeeziabari *et al.* proposed a latency-aware task mapping algorithm. It divides communications of a given application graph into low volume and high volume communication subgraphs. Then it maps application subgraphs one by one based on their total intra communications considering where the vertical channels are located in the network. However, these approaches do not consider thermal aspects.

[1]The Experimental setup is described in Section V.A

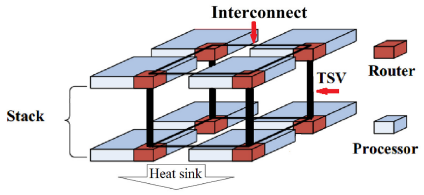

Figure 2: An example of 3D mesh NoC architecture

Mapping algorithms in the second category focus on temperature optimization [6, 11, 20, 25, 26]. In [6], Cui *et al.* proposed a B2T algorithm that maps all the tasks to the bottom layer which is close to the heat sink, followed by a second step which moves low-power tasks to the top layer. In [26], Zhu *et al.* proposed the TAPP (Temperature-Aware Partitioning and Placement) algorithm to reduce on-chip hotspots. TAPP spreads high-power cores and routers across the chip by performing a hierarchical bi-partitioning of the cores and concurrently conducting placement of the cores onto tiles, and achieves high efficiency and scalability. These algorithms ignore communication distance, which might lead to a higher network latency. In [11], Hamedani *et al.* explored the temperature constraints for thermal aware mapping of 3D networks on chip, focusing on the design of thermal management algorithm.

Mapping algorithms in the third category focus on jointly optimizing communication latency and temperature [1, 5, 14, 15, 23, 27]. In [15], Mosayyebzadeh *el al.* proposed an algorithm using fuzzy logic to adjust the impact of heat emission capability, inter-task distance inside application, and distance from hot spot. It reduces the communication delay by mapping tasks with large communication volumes to cores close to each other. It also reduces power consumption and peak temperature by mapping tasks to cores that are close to the heat sink. However, this method does not fully exploit the vertical links to optimize latency. Further, all these approaches do not consider fragmentation [17], which might lead to high communication latency after running many applications.

3 PROBLEM FORMULATION

3.1 System Model

In this work, we adopt the 3D NoC system model as that of [6]. Fig. 2 shows the architecture of a 3D mesh NoC with TSVs as the vertical links.The 3D NoC is modeled as a directed graph $G(C, L)$, where C is the set of cores and L is the set of links connecting the cores. The cores of the system model support various voltage/frequency (V/F) levels. The deterministic XYZ routing is used. The layer which lies next to the heat sink is referred to as the bottom layer and the most distant one from the heat sink is referred to as the top layer. A centralized platform resource manager is designed to monitor the arrival of application, manage resources and perform application mapping. Table 1 summarizes the definition of symbols used in this paper.

Table 1: Symbol Definition

Symbol	Definition
$G(C, L)$	A 3D mesh NoC system
G_w	X dimension (width) of the 3D NoC
G_l	Y dimension (length) of the 3D NoC
G_h	Z dimension (height) of the 3D NoC
C	The set of cores in a 3D NoC system
L	The set of links connecting the cores in a 3D NoC system
S	The set of applications
A_i	Application i
T_i	The set of tasks of A_i
E_i	The set of edges of A_i
$e(i, j)$	Edge e between two tasks
$M(t) = c$	A mapping function binding task t to core c
$TT(e)$	The transmission time of data between two communicating tasks
$V(i, j)$	The traffic volume between two communicating tasks
$D(i, j)$	The Manhattan distance between two cores
$P(c)$	The power of the core c
$P_M(c)$	The thermal power capacity of the core c
$\sigma(S)$	The overall running time of applications in the set S
CV_i	The average communication volume of each task in A_i
RP	Reference point
CI_i	The corner index for A_i
NOL_i	The number of occupied layers of an application
MD_i	The minimal distance to heat sink of an application
RT_i	The running time of an application
ERT_i	The estimated running time of an application
$\hat{\sigma}(S)$	The estimated overall running time of a set applications
$\hat{\sigma}^*(S)$	The minimum overall running time of a set applications

3.2 Application Model

Each application is modeled as a directed graph $A = (T, E)$, where T is the set of tasks of the application and E is the set of directed edges representing data communication amongst the tasks. Each task $t_i \in T$, where i = 1, 2 , ..., n, has a weight equal to its execution time. For Each edge $e \in E$, where j = 1, 2, ..., m, has a weight corresponding to data volume between the two communicating tasks, representing traffic.

A mapping function $M(t) = c$, for each $t \in T_i$, each $c \in C$ binds tasks to the cores, such that task t is mapped to core c. Each edge $e \in E_i$ has a weight of transmission time, after the two communicating tasks are mapped. The transmission time between two tasks i and j depends on the communication distance between the cores to which they are mapped, and their traffic volume. For each edge $e = (i, j)$, the transmission time can be modeled by Equation 1.

$$TT(e) = \alpha \cdot V(i, j) + \beta \cdot D(z, j) \tag{1}$$

where $V(i, j)$ is the traffic volume between the two tasks i and j, and $D(i, j)$ is the distance between two cores to which the two tasks are mapped. α and β are regression coefficients of the linear regression model, which can be computed using the maximum likelihood method in [10]. The running time of each application i is the makespan of task graph, denoted as RT_i.

3.3 The Thermal Power Capacity Model

We adopted the thermal power capacity model (TPC) of [21]. As in [21], we define the TPC of a core as the maximum power the core can consume, given the power consumptions of other cores. It is used at runtime to estimate the power of a core given the power consumption of its neighboring eight cores with low computing cost. The TPC of each core can be determined offline. In the rest of the paper, we use $P_M(c)$ and $P_M(x, y, z)$ to denote the power capacity of the core c at the location (x, y, z) interchangeably. The TPC of a core is bounded by the cooling capacity of the system, and the power consumption or temperature of other cores, *i.e.*, thermal correlation. The TPC of a core c can be found as,

$$P_M(x, y, z) = \theta[P(x \pm l_1, y \pm l_2, z')] = \sum_q \alpha_q \cdot P(c) \tag{2}$$

where $P(x \pm l_1, y \pm l_2, z')$ is the power consumption of the core c located at $(x \pm l_1, y \pm l_2, z')$, which is thermally correlated with core c. The function $\theta(\cdot)$ can also be found by linear regression, using the lasso method [10]. In particular, for each core at (x, y, z) we only keep the coefficients of (1) adjacent cores and (2) those with the same Z coordinate as non-zero. That is, $(x \pm l_1, y \pm l_2, z')$ with $l_1, l_2 = 0, 1$, refering to cores that are neighboring to the core (x, y, z), and $z' = 0, 1, ...G_h$ and $z' \neq z$, refering to cores that have the same Z coordinate as core c. These cores have the highest thermal correlations with the core (x, y, z). We set the coefficients of other cores to be 0, for core c.

3.4 Problem Description

The thermal- and communication-aware mapping problem is define as:

Given a set of n applications in the system.

Find a task-to-core mapping $M(A_i)$ for each application A_i in the 3D NoC system.

Such that the overall running time of these applications is minimize while the system peak temperature is below threshold.

Mathematically, the objective of our proposed algorithm is:

$$\min \quad \sigma(S) = A_{lf} - A_{fa} \tag{3}$$

$$\text{subject to} \quad P(c) \leq P_M(c), \ \forall c \in C \tag{4}$$

where $\sigma(S)$ is the overall running time of the application set S, A_{lf} and A_{fa} are the finish time of the last application and the arrival time of the first application in the set, $P(c)$ is the power of a core and $P_M(c)$ is the maximum power a core can consume which is computed from the TPC model.

4 THE PROPOSED THERMAL- AND COMMUNICATION-AWARE MAPPING ALGORITHM

4.1 Overview

The proposed algorithm has two steps:

(1) Finding a 3D cuboid core region of a specific shape for every application.
(2) Determining the exact location of each application's core region.

4.2 Finding the Shape of 3D Cuboid Core Regions

4.2.1 Characterizing the Shape of A Core Region. Two metrics, the minimal distance to heat sink (MD) and the number of occupied

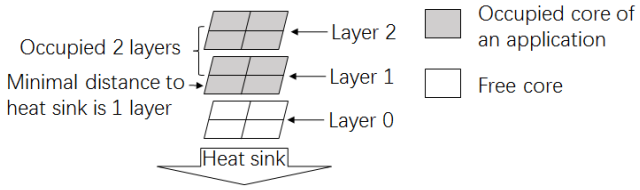

Figure 3: An example of minimal distance to heat sink (*MD*) and the number of occupied layers (*NOL*) values of a core region.

layers (*NOL*), are used to characterize the shape of a core region Layers closer to the heat sink have better cooling effect, which implies that cores in such layers can run with a higher V/F level and power consumption without violating the thermal constraint. Therefore, the distance to heat sink of a core region decides the peak temperature and computation performance. We use the minimal distance to heat sink defined below to reflect the average distance to heat sink of a core region. For example, the *MD* value of the core region for the 8-task application in Fig. 3 is 1.

Definition 4.1. The minimal distance to heat sink (*MD*) is the index of the lowest layer of A_i's core region.

On the other hand, the number of occupied layers of a core region decides the number of TSV links in a core region corresponding to how many vertical links are used in a core region to accelerate communication. For an application whose number of tasks is fixed, a "taller" core region has more TSV links than a "shorter" one. Thus, a taller core region has lower network communication latency. That is, a larger "NOL" indicates a lower communication latency. For example, the *NOL* of the 8-task application in Fig. 3 is 2.

Definition 4.2. The number of occupied layers value NOL_i of application A_i's core region equals to its number of layers.

4.2.2 Estimation of an Application's Running Time. The running time of an application depends on the execution time of each task and the communication latency. The execution time of each task can be modeled by the *MD* metric and the communication latency can be modeled by the *NOL* metric as discussed in Section 4.2.1. Consider the trade-off between model accuracy and algorithm runtime overhead, the running time of core region could be estimated by a linear regression model of *MD* and *NOL* as in Equation 5,

$$ERT_i = a_0 + a_1 \times |A_i| + a_2 \times CV_i + a_3 \times NOL_i + a_4 \times MD_i \quad (5)$$

where $|A_i|$ is the number of tasks in application A_i, CV_i is the average communication volume of each task in A_i, NOL_i is the number of occupied layers of the core region, and MD_i is the *MD* of the core region. To find the coefficients a_0, a_1, a_2, a_3, a_4, the maximum likelihood methods can be used [10].

In this step, a core region of a specific shape is selected for each application, by tuning its *MD* and *NOL* values according to its communication and computation demands. The branch-and-bound algorithm is used to find the best tree node corresponding to the best combination of shapes of core region for this set of application set.

4.2.3 Search tree definition. Assume n applications are to be mapped, a tree node is defined as a $2n$-vector. A tree node is defined by $N_m = \langle NOL_0, MD_0, ..., NOL_n, MD_n \rangle$, corresponding to the *NOL* and *MD* metrics of each application's core region.The tree is grown by branching new nodes from the the root down to the leaves level by level, corresponding to setting the MD_i and NOL_i values for each application's core region. Each non-leaf tree node N_j is associated with $\hat{\sigma}(S)_j^{min}$, indicating the minimal estimated running time of N_j. The weight $\hat{\sigma}(S)$ of each leaf node can be computed as $\hat{\sigma}(S) = \max_{A_i \in S}\{ERT_i\}$ for all applications, that is, the time when the last application in set S finishes execution.

The basic operations for the search tree include branch and cut.

Branching of the search tree. If a tree node is a leaf node, all the applications are mapped. We compare the $\hat{\sigma}(S)$ of this tree node with $\hat{\sigma}^*(S)$. If this tree node is not a leaf node, a new tree node should be created. The MD_i and NOL_i of new application A_i are grounded by the following max and min *NOL*s and *MD*s.

$$NOL_i^{min} = \left\lceil \frac{|A_i|}{G_w \times G_l} \right\rceil \quad (6)$$

$$NOL_i^{max} = \min\{G_h, |A_i|\} \quad (7)$$

$$MD_i^{min} = G_h - NOL_i^{max} \quad (8)$$

$$MD_i^{max} = G_h - NOL_i^{min} \quad (9)$$

where $|A_i|$ is number of tasks of A_i and G_w, G_l and G_h are the width, length and height of the 3D NoC system, respectively.

NOL_i^{min} is the minimum number of occupied layers of a core region for an application such that the application's tasks can be accommodated in the core region. NOL_i^{max} is the upper limit of the core region's number of occupied layers, which is bounded by the maximum number of vertical layers in the 3D NoC. MD_i^{min} and MD_i^{max} are the maximum and minimum *MD* values of a core region. Once NOL_i^{min} and NOL_i^{max} are determined, they can be computed in a manner such that the core region can fit inside the 3D NoC. In total, for each non-leaf tree node, a maximum of $NOL_i^{min} \times NOL_i^{max} \times MD_i^{min} \times MD_i^{max}$ tree nodes in the next level can be created.

Cutting. To reduce the search space and speed up the tree search process, some tree branches should be cut. For each of the newly created tree node, we check whether they should be discarded or not according to the following two cut rules.

Rule 1) Cut infeasible nodes. An infeasible tree node refers to a setting of *MD* and *NOL* of the core regions that they do not fit into the 3D NoC, *e.g.,* the number of cores of a region in a layer is larger than the total available cores in that layer. The number of available cores in each layer is maintained and updated at runtime to test the feasibility. Once some applications are mapped to run, the number of available cores in each layer is subtracted by the amount equal to the core count occupied by the applications in that layer. For the vertical direction, it is feasible if the sum of *MD* and *NOL* is less than or equal to G_h.

Rule 2) Cut by node dominance. Once a new tree node N_j is created, its $\hat{\sigma}(S)_j^{min}$ is compared with $\hat{\sigma}^*(S)$. If its $\hat{\sigma}(S)_j^{min}$ is larger than $\hat{\sigma}^*(S)$, which means the minimum overall application running time of N_j is longer than the best overall running time found so far among all the tree nodes, the new tree node is discarded.

Algorithm 1 shows how the search works. The tree nodes are stored in a working queue. A dummy root node is pushed to the queue at the start. In each iteration, new tree nodes are created by

Algorithm 1 Finding the Shape of Core Region for Each Application

Input: S, the set of unmapped applications
Output: MD_i and NOL_i values of each application A_i
WQ: A working queue, initialized empty;
NBN: The newly branched node
BN: The best node during search;
$\hat{\sigma}^*(S)$: A value keeps the minimum overall running time over all the tree nodes searched so far;
while *WQ is not empty* **do**
  pop the top node N_q out of WQ;
  if *N_q is not a leaf node* **then**
    branch new tree nodes;
    for *each newly branched nodes NBN* **do**
      if *no cutting rules are met for NBN and WQ is not full* **then**
        store NBN in WQ;
    end
  else
    if $\hat{\sigma}(S)_q < \hat{\sigma}^*(S)$ **then**
      $\hat{\sigma}^*(S) = \hat{\sigma}(S)_q$;
      $BN = N_q$;
end

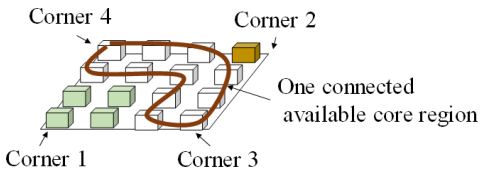

Figure 4: Four corners in one layer of a 3D NoC system

assigning different MD_i and NOL_i values to a new application's core region. In total $NOL_i^{min} \times NOL_i^{max} \times MD_i^{min} \times MD_i^{max}$ new tree nodes are created. For each of these tree nodes, if they do not meet the above two cutting rules, they are pushed to the queue. The length of the queue can be tuned to trade off the running speed and result optimality of the search algorithm. In this search process, applications with more traffic are possibly assigned with a region with larger NOL, while the computation intensive applications are possibly assigned with a region with smaller MD.

4.3 Finding the Locations for the 3D Cuboid Core Regions

In this step, the exact location of each core region in the chip is found. The goals in this step are to 1) keep the applications scattered to reduce peak temperature, and 2) reduce fragmentation. To achieve these goals, the core regions are placed at one of the four corners of the chip in a round-robin manner. Algorithm 2 shows how the location finding step works.

First, the applications are sorted according to their number of tasks in a descending order, *i.e.,* applications with more tasks are treated earlier. Next, the applications' core regions are placed in a round-robin manner to one of the four corners as in Fig. **??** at each iteration. We use CI_i as the corner index for application A_i. Each corner is associated with a reference point, RP, indicating its start point coordinate. Starting from RP, we first scan along the

Algorithm 2 Finding the Exact Location of Core Regions

Input: S, the set of unmapped applications
Output: $M(A_i)\ \forall A_i \in S$, the task to core mapping for each application
sort applications in S according to their number of tasks in a descending order;
set CI_i as 1;
for *the core region of each $A_i \in S$* **do**
  select RP, and scan along the X and Y directions according to the above four rules, with the constraint that it fits inside the 3D NoC;
  map the tasks of the application to the cores in that region;
  $CI_i = CI_i$ % 4 + 1;
end

X dimension, then the Y dimension, until a region is found that has $\lceil \frac{|A_i|}{NOL_i} \rceil$ free cores at layers $MD_i, ..., MD_i + NOL_i$, detailed as follows.

(1) corner 1: $RP = (0, 0, MD_i)$. Search location along $x+$ direction, followed by $y+$ direction.
(2) corner 2: $RP = (G_w, G_l, MD_i)$. Search location along $x-$ direction, followed by $y-$ direction.
(3) corner 3: $RP = (0, G_l, MD_i)$. Search location along $x-$ direction, followed by $y+$ direction.
(4) corner 4: $RP = (G_w, G_l, MD_i)$. Search location along $x+$ direction, followed by $y-$ direction.

Then, tasks of the application are mapped to each of the cores inside the free core region using existing mapping algorithms, for example, the one in [23]. This algorithm results in a contiguous free core region in the center of the chip as in Fig. **??**. Therefore, fragmentation is alleviated. Besides, since the applications are mapped such that they are separated in the four corner, the peak temperature is also reduced.

4.4 Example

There are 2 applications A_0 and A_1 in a 2×2×2 3D NoC system to be mapped. Fig. 4.4 (c) is a snapshot of a search tree for finding the shapes of core regions for A_0 and A_1. Each tree node is represented by a 4-element vector $< NOL_0, MD_0, NOL_1 \text{Th} MD_1 >$. A dummy root node is first created. For A_0, a new tree node N_0 in level 1 is created from the root with $MD_0 = 0$ and $NOL_0 = 1$. Since A_0 has 3 tasks, it have 3 cores in layer 0. Next, a new node N_1 in level 2 is created for A_1 from N_0, with $MD_1 = 0$ and $NOL_1 = 2$. Since A_1 has 2 tasks, it has 1 core in both layers 0 and 1. Since the system has 4 cores in each layer, A_0 and A_1 fit into the system. Fig. 4.4(d) shows an infeasible tree node N_2 in level 2. It requires 5 cores in layer 0 and thus exceeds the maximum number of available cores in layer 0. Thus, N_1 is kept as the result of searching.

Next, the locations of the applications' core regions are found with the MD_0, NOL_0, MD_1, NOL_1 shown in N_1. Since A_0 has one more tasks than A_1, A_0 is mapped first. Since $MD_0 = 0$ and $NOL_0 = 1$, layer 0 needs 3 free cores for A_0. Starting from corner 1, *i.e., RP* is (0, 0, 0), then scan by $X+$ direction and 3 available cores are found for A_0's core region. Now, for A_1, it needs 1 available core in both layers 0 and 1 for A_1. This time we start from corner 2, *i.e., RP* is (1,

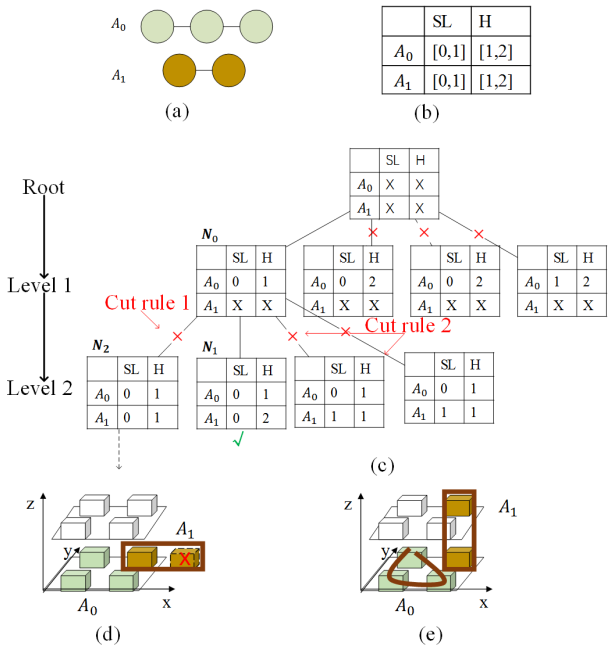

Figure 5: (a) Two applications (A_0, A_1) to be mapped. (b) The ranges of *MD* and *NOL* values of the two applications. (c) Search tree in step 1. (d) An infeasible tree node. (e) System after mapping.

1, 0) and an available core is found in each of the two layers for A_1. The locations of the the two core regions are shown in Fig. 4.4(e).

4.5 Cost Analysis

The worse case complexity of the first step of the algorithm is $O(F)$, where $F = \min\{|WQ|, G_h^{2n}\}$, and $|WQ|$ is the maximum length of the working queue WQ. The complexity of the second step includes the complexity of finding core region locations and that of mapping n applications, which are $O(n \times G_w \times G_l)$ and $O(\sum_{\forall A_i \in S} |A_i|^2 \times |E| \times |C|)$, respectively.

Overall, the complexity of the whole algorithm is $O(B)$, where $B = \max\{F, \sum_{\forall A_i \in S} |A_i|^2 \times |E| \times |C|\}$.

5 EXPERIMENTAL RESULT

5.1 Experimental Setup

Experiments are performed on an event-driven C++ NoC simulator, with DSENT integrated as the power model. The simulator models the packet latency of the communications in a cycle accurate manner. The horizontal and vertical link bandwidth considered for simulation are 10 and 200. For example, to finish 200 units communication volume, horizontal link needs 20 cycles and vertical link needs 1 cycle. For processor-to-memory latency, we assumed the code and data are stored in the L1 cache and this latency is ignored considering it will be very small. The experiments are carried out on various 3D NoC systems. In the experiments, we used two kinds of benchmarks: benchmarks whose task graphs are randomly generated and real benchmark. The task graphs of the real applications are generated from the traces of SPLASH-2 [24] and PARSEC [3], which are collected by executing these applications in a 8×8 NoC-based cycle accurate many-core simulator in [22]. The configuration of random benchmarks, real benchmarks, and the 3D NoC system are listed in Table 2. The temperature threshold is 60 ℃. It is acceptable to take any reasonable temperature threshold with a re-train thermal model. HotSpot [12] is used as the temperature simulator. We used different layer thickness to simulate the different heat transmit capacity, the Hotspot parameters are showed in Table 2.

Table 2: Configurations of the Simulation

Network Parameters	
Flit size	128 bits
Latency Router	2 cycles, link 1 cycle
Buffer depth	4 flits
Routing algorithm	XYZ routing
Baseline topology	8 × 8 × 4
Random Benchmark Parameters	
Number of tasks	[15, 45]
Communication volume	[10, 200] (Kbits)
Degree of tasks	[1, 15]
Task number distribution	Bimodal, uniform
Configuration of the Extracting Trace Many-core Simulator	
Core Architecture	64 bit Alpha 21264
Baseline Frequency	3GHz
Fetch/Decode/Commit size	4/4/4
ROB size	64
L1 D cache (private)	16KB, 2-way, 32B line 2 cycles, 2 ports, dual tags
L1 I cache (private)	32KB, 2-way 64B line, 2 cycles
L2 cache (shared) MESI protocol	64KB slice/core, 64B line 6 cycles, 2 ports
Main memory size	2GB
Task Graphs of Real Applications	
Barnes, Canneal, Raytrace, Dedup, Ferret, Freqmine Streamcluster, Fluidanimate, Swaptions, Blackscholes	
Hotspot Parameters	
Die size [mm]	0.5 × 0.5
Specific heat capacity [J/(m^3 × K)]	1.75e6
Resistivity [(m-K)/W]	0.01
Layer 0 thickness [mm]	0.10
Layer 1 thickness [mm]	0.12
Layer 2 thickness [mm]	0.14
Layer 3 thickness [mm]	0.16

We compare our approach with the following two runtime mapping approaches, the Bottom-2-Top (B2T) method [6] and the fuzzy logic (FL) method [15]. The B2T scheme first maps all the tasks to the bottom layer in a 3D NoC to make the power consumption of cores in each vertical stack (cores with the same Z coordinate) identical as possible, and then moves low power tasks to the top layer to reduce the execution time [6]. FL defines three variables, *i.e.* heat transfer, distance from source core, and distance from hot spot and used rules to set the priorities of them. "Distance from

source core" represents the distance inter tasks within the application. "Heat transfer" is the heat transfer capability of specific core. "Distance from hot spot" is the distance between the hottest core and specific core. During the experiment, we set the rules as considering "distance from source core" first, then "heat transfer", and finally "distance from hot spot". That is, FL maps tasks within application to cores of shorter distance first. If shorter distance cores are not available, then it uses the cores near the heat sink. If shorter distance cores and near heat sink cores are not available, it uses cores far from the hot spot.

5.2 Validating the Application Running Time Estimation Model

The error of each application's running time estimation model is defined as follow,

$$\epsilon = \frac{|RT - ERT|}{RT} \times 100\% \quad (10)$$

where RT and ERT are the running times obtained from the simulator and the estimation model in Equation 5 for each application, respectively. The error of this estimation model is 4.82% on average, for the applications used in the experiments. Thus, the estimation is fairly accurate.

5.3 Performance Comparison

5.3.1 *Experiments with Random Benchmarks.*

Experiments with Different Sizes of 3D NoC. To evaluate the proposed algorithm, we changed the size of 3D NoC to evaluate the overall running time and communication latency of the three algorithms. Fig. 6 shows that our proposed algorithm outperforms the other two when the network size increases. On average, the overall running times of B2T and FL are 1.39× and 1.42× over our proposed approach, respectively. Fig. 6 also shows that the communication latencies of B2T and FL are 1.55× and 1.24× over our proposed approach, respectively. The reason is that our algorithm can optimize the communication and computation performances for each application by selecting the appropriate MD and NOL values of its core region, according to its communication and computation demands. It can also reduce fragmentation and peak temperature by the core region location finding step. For every application, the B2T method makes maximum use of the layer that is close to the heat sink and thus it might lead to a scenario that all the cores close to the heat sink are occupied and the upcoming application has to be mapped to other layers. Further, B2T maps applications in close proximity, resulting in higher accumulated heat.

Experiments with Different Number of Tasks. In this set of experiments, we changed the average number of tasks of the applications. Fig. 7 shows that, when the average number of tasks is large, *e.g.* 128, the overall running time of B2T and FL are 1.29× and 1.39× over our proposed approach, respectively. Fig. 6 also shows that the communication latencies of B2T and FL are 1.42× and 1.38× over our proposed approach, respectively.

Experiments with Different Communication Volumes. In this set of experiments, we compared the performances with 4 different communication volumes: 50, 100, 150, 200 (Kbits). Fig. 8 shows that, when the communication volume is large, *e.g.* 200, the overall running time of B2T and FL are 1.38× and 1.25× over our proposed approach, respectively. Fig. 6 also shows that the communication latencies of B2T and FL are 1.47× and 1.13× over our proposed approach, respectively.

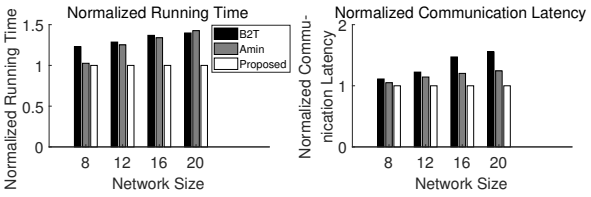

Figure 6: Comparison with different sizes of 3D NoC

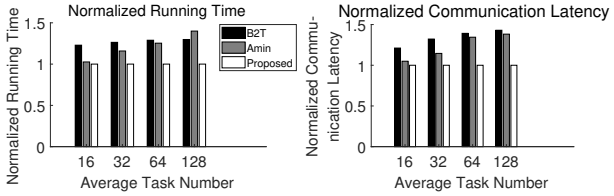

Figure 7: Comparison with different number of tasks of applications

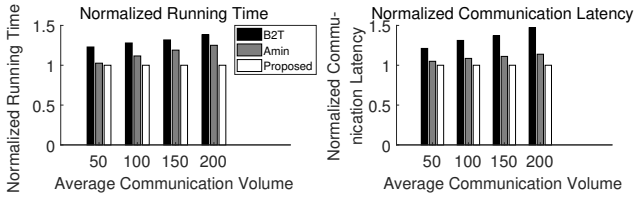

Figure 8: Comparison with different communication volumes

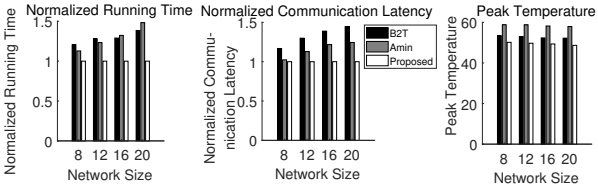

Figure 9: Comparison with real benchmarks

5.3.2 Experiments with Real Benchmarks. To evaluate the proposed framework, we compared the performances on real benchmarks with 4 different network sizes. Fig. 9 shows that, when the network size is large, *e.g.* 20 × 20 × 4, the overall running time of B2T and FL are 1.38× and 1.48× over our proposed approach, respectively. Fig. 9 also shows that the communication latencies of B2T and FL are 1.44× and 1.24× over our proposed approach, respectively. Fig. 9 also compares the peak temperature of different algorithms, showing that our algorithm reduces the peak temperature for B2T and FL by 3°C and 7℃. All of the three mapping algorithms are below the 60℃ thermal threshold.

We also captured the normalized communication cost and running time of each real benchmark application , as shown in Fig. 10. Numbers 0 to 9 represent barnes, blackscholes, canneal, dedup, ferret, fluidanimate, freqmin, raytrace, streamcluster and swaptions respectively.

5.3.3 Runtime Overhead of the Proposed Algorithm. The average runtime overheads of B2T, FL and our algorithm are in the order of 3.5M, 5M and 4M cycles for one real benchmark, which are averaged

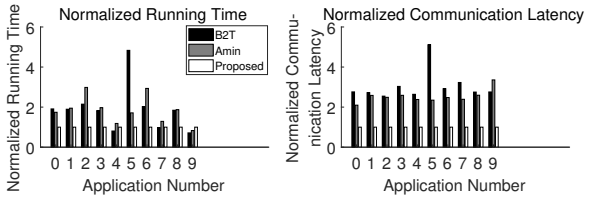

Figure 10: Comparison for each real benchmark

by running each of the algorithms for fifty times with different real benchmark applications. The execution times of the benchmark applications are much longer than 4M cycles. Thus the overhead of the proposed algorithm is acceptable.

6 CONCLUSION

In this paper, we proposed a runtime communication- and thermal-aware mapping algorithm to optimize performance under the thermal constraint in 3D NoCs. Experimental results show that our proposed approach can reduce up to 48% overall running time compared to existing mapping algorithms.

7 ACKNOWLEDGEMENT

This research program is supported by the Natural Science Foundation of China No. 61376024 and 61306024, Natural Science Foundation of Guangdong Province 2015A030313743, Special Program for Applied Research on Super Computation of the NSFC-Guangdong Joint Fund (the second phase), and the Science and Technology Research Grant of Guangdong Province No. 2016A010101011 and 2017A050501003.

REFERENCES

[1] Charles Addo-Quaye. 2005. Thermal-aware mapping and placement for 3D NoC designs. In *Proc. IEEE Int'l SoC Conf.* 25–28.

[2] Iraklis Anagnostopoulos, Vasileios Tsoutsouras, Alexandros Bartzas, and Dimitrios Soudris. 2013. Distributed run-time resource management for malleable applications on many-core platforms. In *Proc. Design Automation Conf.* 168:1–6.

[3] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. 2008. The PARSEC benchmark suite: Characterization and architectural implications. In *Proc. the 17th international conf. Parallel architectures and compilation techniques.* ACM, 72–81.

[4] Chih-Hao Chao, Kai-Yuan Jheng, Hao-Yu Wang, Jia-Cheng Wu, and An-Yeu Wu. 2010. Traffic-and thermal-aware run-time thermal management scheme for 3D NoC systems. In *Proc. ACM/IEEE Int'l Symp. Networks-on-Chip.* 223–230.

[5] Marco Cox, Amit Kumar Singh, Akash Kumar, and Henk Corporaal. 2013. Thermal-aware mapping of streaming applications on 3D multi-processor systems. In *Proc. Symp. Embedded Systems for Real-time Multimedia.* 11–20.

[6] Yingnan Cui, Wei Zhang, Vivek Chaturvedi, Weichen Liu, and Bingsheng He. 2016. Thermal-aware task scheduling for 3D network-on-chip: a bottom to top scheme. *J. Circuits, Systems and Computers* 25, 1 (2016), 224–227.

[7] Ewerson Luiz de Souza Carvalho, Ney Laert Vilar Calazans, and Fernando Gehm Moraes. 2010. Dynamic task mapping for MPSoCs. *IEEE Design & Test of Computers* 27, 5 (2010), 26–35.

[8] Mohammad Fattah, Masoud Daneshtalab, Pasi Liljeberg, and Juha Plosila. 2013. Smart hill climbing for agile dynamic mapping in many-core systems. In *Proc. Design Automation Conf.* 1–6.

[9] Mohammad Fattah, Pasi Liljeberg, Juha Plosila, and Hannu Tenhunen. 2014. Adjustable contiguity of run-time task allocation in networked many-core systems. In *Proc. Asia and South Pacific Design Automation Conf.* 349–354.

[10] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2001. *The elements of statistical learning.* Vol. 1.

[11] Parisa Khadem Hamedani, Shaahin Hessabi, Hamid Sarbazi-Azad, and Natalie Enright Jerger. 2012. Exploration of temperature constraints for thermal aware mapping of 3D networks on chip. In *20th Euromicro Int'l Conf. Parallel, Distributed and Network-Based Processing (PDP).* IEEE, 499–506.

[12] Wei Huang, Shougata Ghosh, Sivakumar Velusamy, Karthik Sankaranarayanan, Kevin Skadron, and Mircea R Stan. 2006. HotSpot: a compact thermal modeling methodology for early-stage VLSI design. *IEEE Trans. Very Large Scale Integration Systems* 14, 5 (2006), 501–513.

[13] Guruprasad Katti, Michele Stucchi, Kristin De Meyer, and Wim Dehaene. 2010. Electrical modeling and characterization of through silicon via for three-dimensional ICs. *IEEE Trans. Electron Devices* 57, 1 (2010), 256–262.

[14] Jiayin Li, Meikang Qiu, Jian-Wei Niu, Laurence T Yang, Yongxin Zhu, and Zhong Ming. 2013. Thermal-aware task scheduling in 3D chip multiprocessor with real-time constrained workloads. *ACM Trans. Embedded Computing Systems* 12, 2 (2013), 24:1âĂŞ22.

[15] Amin Mosayyebzadeh, Maziar Mehdizadeh Amiraski, and Shaahin Hessabi. 2016. Thermal and power aware task mapping on 3D network on chip. *Computers & Electrical Engineering* 51 (2016), 157–167.

[16] Makoto Motoyoshi. 2009. Through silicon via (TSV). *Proc. the IEEE* 97, 1 (2009), 43–48.

[17] Jim Ng, Xiaohang Wang, Amit Kumar Singh, and Terrence Mak. 2015. DeFrag: defragmentation for efficient runtime resource allocation in NoC-based many-core systems. In *Proc. Int'l Conf. Parallel, Distributed and Network-Based Processing.* 345–352.

[18] Dongkook Park, Soumya Eachempati, Reetuparna Das, Asit K Mishra, Yuan Xie, Narayanan Vijaykrishnan, and Chita R Das. 2008. MIRA: a multi-layered on-chip interconnect router architecture. In *ACM SIGARCH Computer Architecture News*, Vol. 36. 251–261.

[19] Vasilis F Pavlidis and Eby G Friedman. 2007. 3D topologies for networks-on-chip. *IEEE Trans. Very Large Scale Integration Systems* 15, 10 (2007), 1081–1090.

[20] Chong Sun, Li Shang, and Robert P Dick. 2007. Three-dimensional multiprocessor system-on-chip thermal optimization. In *Proc. IEEE/ACM Int'l Conf. Hardware/software Codesign and System Synthesis.* 117–122.

[21] Xiaohang Wang, Amit Kumar Singh, Bing Li, Yang Yang, Terrence Mak, and Hong Li. 2016. Bubble budgeting: Throughput optimization for dynamic workloads by exploiting dark cores in many core systems. In *Proc. IEEE/ACM Int'l Symp. Networks-on-Chip.* 1–8.

[22] Xiaohang Wang, Mei Yang, Yingtao Jiang, Peng Liu, Masoud Daneshtalab, Maurizio Palesi, and Terrence Mak. 2014. On self-tuning networks-on-chip for dynamic network flow dominance adaptation. *ACM Trans. Embedded Computing Systems* 13, 2 (2014), 1–8.

[23] Xiao-Hang Wang, Peng Liu, Mei Yang, Maurizio Palesi, Ying-Tao Jiang, and Michael C Huang. 2013. Energy efficient run-time incremental mapping for 3D networks-on-chip. *J. Computer Science and Technology* 28, 1 (2013), 54–71.

[24] Steven Cameron Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta. 1995. The SPLASH-2 programs: Characterization and methodological considerations. In *Proc. Annual International Symp. Computer Architecture.* IEEE, 24–36.

[25] Changyun Zhu, Zhenyu Gu, Li Shang, Robert P Dick, and Russ Joseph. 2008. Three-dimensional chip-multiprocessor run-time thermal management. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems* 27, 8 (2008), 1479–1492.

[26] Di Zhu, Lizhong Chen, Timothy M Pinkston, and Massoud Pedram. 2015. TAPP: temperature-aware application mapping for NoC-based many-core processors. In *Proc. Design, Automation & Test in Europe Conf. & Exhibition.* 1241–1244.

[27] Zuomin Zhu, Vivek Chaturvedi, Amit Kumar Singh, Wei Zhang, and Yingnan Cui. 2017. Two-stage thermal-aware scheduling of task graphs on 3D multi-cores exploiting application and architecture characteristics. In *Proc. Asia and South Pacific Design Automation Conf.* 324–329.

[28] Hesamedin Ziaeeziabari and Ahmad Patooghy. 2017. 3D-AMAP: a latency-aware task mapping onto 3D mesh-based NoCs with partially-filled TSVs. In *Proc. Int'l Conf. Parallel, Distributed and Network-based Processing.* 593–597.