SMT-Based Contention-Free Task Mapping and Scheduling on 2D/3D SMART NoC with Mixed Dimension-Order Routing

DAEYEAL LEE, BILL LIN, and CHUNG-KUAN CHENG, University of California San Diego, USA

SMART NoCs achieve ultra-low latency by enabling single-cycle multiple-hop transmission via bypass channels. However, contention along bypass channels can seriously degrade the performance of SMART NoCs by breaking the bypass paths. Therefore, contention-free task mapping and scheduling are essential for optimal system performance. In this article, we propose an SMT (Satisfiability Modulo Theories)-based framework to find optimal contention-free task mappings with minimum application schedule lengths on 2D/3D SMART NoCs with mixed dimension-order routing. On top of SMT's fast reasoning capability for conditional constraints, we develop efficient search-space reduction techniques to achieve practical scalability. Experiments demonstrate that our SMT framework achieves $10\times$ higher scalability than **ILP (Integer Linear Programming)** with $931.1\times$ (ranges from $2.2\times$ to $1532.1\times$) and $1237.1\times$ (ranges from $4\times$ to $4373.8\times$) faster average runtimes for finding optimum solutions on 2D and 3D SMART NoCs and our 2D and 3D extensions of the SMT framework with mixed dimension-order routing also maintain the improved scalability with the extended and diversified routing paths, resulting in reduced application schedule lengths throughout various application benchmarks.

CCS Concepts: • **Computer systems organization** → **Multicore architectures; Interconnection architectures;**

Additional Key Words and Phrases: SMT, SMART NoC, task mapping, scheduling

ACM Reference format:

Daeyeal Lee, Bill Lin, and Chung-Kuan Cheng. 2021. SMT-Based Contention-Free Task Mapping and Scheduling on 2D/3D SMART NoC with Mixed Dimension-Order Routing. *ACM Trans. Archit. Code Optim.* 19, 1, Article 5 (December 2021), 21 pages.
<https://doi.org/10.1145/3487018>

1 INTRODUCTION

Network-on-Chip (NoC) is a widely used interconnection fabric that provides a highly scalable low-latency on-chip communication solution for **multiprocessor system-on-chips (MP-SoCs)** [6]. However, since communications between cores in a regular NoC are achieved by routing messages hop-by-hop from the source to the destination, their effectiveness at non-local communications quickly diminishes due to long on-chip latencies, which degrades performance and limits

Chung-Kuan Cheng and Bill Lin acknowledge the support from NSF CCF-2110419 and NSF IIS-1956339.

Authors' address: D. Lee, B. Lin, and C.-K. Cheng, University of California San Diego, 9500 Gilman Dr, La Jolla, CA, 92093; emails: ldaeyeal@ucsd.edu, billin@ucsd.edu, ckcheng@ucsd.edu.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2021 Association for Computing Machinery.

1544-3566/2021/12-ART5 \$15.00

<https://doi.org/10.1145/3487018>

the flexible usage of on-chip resources. Delays due to router pipelines, queuing, and serialization all contribute towards a much longer on-chip latency than an ideal point-to-point interconnect.

Recently, an NoC design called **SMART (Single-cycle Multi-hop Asynchronous Repeated Traversal) NoC** [14, 15] has been proposed that enables a flit to traverse many router hops within a single clock cycle, potentially from the source all the way to the destination. Such a multi-hop traversal is made possible by utilizing efficient router-bypass mechanisms and properly engineered wires with asynchronous repeaters. In particular, router data-paths can not only be dynamically but also statically [3] configured to enable multi-hop traversal so that flits can bypass the pipelines of intermediate routers entirely, resulting in ultra-low latency performance. Although SMART offers outstanding advantages, the performance benefits can only be fully realized if there is no contention among flows that share common links along their routing paths. When contention occurs, bypass paths must terminate early, and the corresponding flits must be stopped and buffered at intermediate routers for arbitration, degenerating in the worst-case to hop-by-hop communication. Without proper contention management, the benefits of SMART can easily vanish.

In this article, we address the aforementioned contention problem by developing an **SMT (Satisfiability Modulo Theories)-based** contention-free task mapping and scheduling framework for the embedded computing application in which an application task graph can be statically compiled to a multi-core or parallel processing platform for non-preemptive execution based on 2D and 3D SMART NoC architectures. In particular, tasks are mapped to processors and scheduled to ensure contention-free routing of all messages over a SMART NoC from their source to their destination in a single cycle. Our SMT formulation can find theoretically optimal solutions that minimize the application schedule length. In contrast to prior ILP-based formulation for 2D SMART NoCs [27], our SMT formulation is substantially more compact, thanks in part to SMT's expressive power in capturing conditional constraints that constitute a large proportion of the task mapping and scheduling problem. Combined with efficient search-space reduction techniques, our SMT formulation is considerably more scalable, enabling our framework to find optimal solutions for far larger problem instances with dramatically faster runtimes.

In addition, we extend our SMT-based formulation to consider the 3D SMART NoC case. Three-dimensional (3D) IC integration is becoming increasingly important, and correspondingly, 3D NoCs are emerging as promising solutions that can deliver lower latency, higher throughput, and reduced energy consumption in comparison with their 2D counterparts [9]. In particular, recent SMART NoC advances [1, 5, 13] have substantially reduced the wiring and area overhead of SMART NoCs to enable 3D extensions. Moreover, the emergence of **monolithic 3D (M3D)** integration has opened up new possibilities for designing SMART 3D NoC architectures with **monolithic inter-tier vias (MIVs)** for vertical interconnections [17], which have much smaller dimensions than the more popular **through silicon vias (TSVs)** [25]. Further, a 3D SMART NoC can potentially operate at higher clock frequencies due to the utilization of vertical interconnects, which results in a reduction of the effective physical link distances [13]. In the context of our task mapping and scheduling problem, a 3D SMART NoC provides greater path diversity, which makes it easier for our SMT-based formulation to find better optimal solutions with contention-free routing.

The main contributions of our work are as follows:

- We propose an SMT-based contention-free task mapping and scheduling framework for 2D/3D SMART NoCs, including support for mixed dimension-order routing.
- We devise a concise model under SMT's support for expressive modeling, resulting in the fast reasoning of conditional constraints.

Fig. 1. SMART router microarchitecture.

Fig. 2. Multi-hop Traversal over a SMART-hop path.

- We develop efficient search-space reduction techniques, e.g., adaptive boundary condition and breaking design symmetry to further improve scalability.
- We demonstrate that our framework achieves smaller formulation complexity with $16.6\times/12.1\times$ and $26.2\times/18.7\times$ reductions of variables and constraints on average and $10\times$ higher scalability with $931.1\times$ (ranges from $2.2\times$ to $1532.1\times$) and $1237.1\times$ (ranges from $4\times$ to $4373.8\times$) faster average runtimes for finding optimum solutions on 2D and 3D SMART NoCs, respectively.
- Our experiments further demonstrate that the 3D extension with mixed dimension-order routing not only maintains the improved scalability but also helps to reduce the application schedule length by exploiting the greater path diversity.

The rest of this article is organized as follows: Section 2 provides preliminary information. Section 3 presents our SMT formulation and search-space reduction techniques for improving scalability. Section 4 validates the proposed frameworks with extensive experimental results. Section 5 outlines additional related work. Section 6 concludes the article.

2 PRELIMINARIES

In this section, we first describe the basic concept of SMART and its latency model and define our problem. Then we review the expressiveness of SMT for conditional constraints.

2.1 Communication Latency in SMART NoC-based MPSoC

This section describes the basic concept and communication latency model of SMART NoC. Figure 1 depicts the microarchitecture of a 5-port SMART router for a mesh network. For

Fig. 3. Example task graph (TG) and two-dimensional/three-dimensional mesh topology graph.

simplicity, only the *Core_{in}* (C_{in}), *West_{in}* (W_{in}), and *East_{out}* (E_{out}) ports are shown in detail.¹ All other input ports are identical to W_{in} , and all other output ports are identical to E_{out} .

In a SMART NoC, asynchronous repeaters, replaced with conventional clocked link drivers at every hop allow a flit to traverse multiple hops in a single clock cycle. An alternative data-path in each router allows a flit to bypass the entire router pipeline and go directly to the next router. The bold line going from W_{in} to E_{out} in Figure 1 illustrates this bypass operation. An example of a multi-hop traversal (i.e., SMART-hop) is depicted in Figure 2. A flit travels three hops from router R20 to R23 within a single-cycle via a SMART-hop path created by appropriately controlled BW_{ena} , BM_{sel} , and XB_{sel} at intermediate routers.

To set up SMART-hops, SMART performs a two-stage switch allocation: **local switch allocation (SA-L)** and **global switch allocation (SA-G)**. In the SA-L stage, buffered flits at a start router arbitrate among themselves to gain access to the output ports. For each winning flit, the start router broadcasts a **SMART-hop Setup Request (SSR)**, which carries the information about the route. Each SSR is sent through dedicated multi-drop wires that are repeated from the start router to all intermediate routers up to HPC_{max} hops away. HPC_{max} refers to the maximum number of hops that a flit can traverse in a single cycle. Upon receiving the SSRs, the recipient routers set up the control signals (i.e., BW_{ena} , BM_{sel} , and XB_{sel}) to operate in bypass or stop mode in SA-G stage. When multiple SSRs from multiple start routers arrive at the same time (i.e., contention on the bypass path), only a winning flit determined by priority policies at the recipient router can proceed and bypass the intermediate routers. Thus, the transmission latency of the SMART-hop path can be formulated as inspired by [27]:

$$L = (t_r + t_w) \cdot (N_c + 1) + t_w \cdot (M^e - 1) + L_b, \quad (1)$$

where t_r and t_w respectively denote the number of stages in a start router and the link latency between two routers; M^e refers to the amount of data units (i.e., flits); N_c denotes the number of links having flit contention on the bypass path; L_b refers to the delay due to the contention. In this article, we employ contention-free task mapping and scheduling. Consequently, the communication latency L_{cf} without contention (i.e., $N_c, L_b = 0$) can be expressed as:

$$L_{cf} = (t_r + t_w) + t_w \cdot (M^e - 1). \quad (2)$$

2.2 Problem Definition

We represent an application as a **task graph (TG)** which is a **directed acyclic graph (DAG)**, $TG = (T, E)$, where T is a set of all tasks $t_i \in T$ in the application and E is a set of edges $e_{u,v} \in E$,

¹A SMART router consists of five input/output ports (i.e., *West*, *East*, *North*, *South*, and *Core*) for connections on a mesh topology.

Fig. 4. Example of task mapping and scheduling.

representing precedence relations between tasks t_u and t_v . In our problem, the target application domain is embedded computing in which an application task graph can be statically compiled to a multi-core or parallel processing platform for non-preemptive execution, and tasks correspond to the blocks of significant computations, like signal processing tasks, not at the level of individual instructions. Each task and edge are respectively associated with the task execution time τ_i and the amount of data transferred between each pair of tasks. For example, Figure 3(a) shows an example task graph with five tasks and five edges including the task execution time and the amount of data.

The target architecture is SMART NoC-based homogeneous MPSoCs [14, 15] with 2D/3D mesh topology graph (MTG), $MTG = (P, L)$, where each node $p_i \in P$ represents a process element (PE) with a router and L is a set of edges representing bi-directional communication paths between adjacent PEs. Figure 3(b) and Figure 3(c), respectively, show an example 3×3 2D mesh topology with 9 PEs and $3 \times 3 \times 2$ 3D mesh topology with 18 PEs. Within this topology, each PE can accommodate more than one task. The location of assigned PEs for each task $t_i \in T$ is determined by a pair of coordinates (x^{t_i}, y^{t_i}) and $(x^{t_i}, y^{t_i}, z^{t_i})$ for 2D and 3D. If two consequential tasks are mapped to the same PE, data transmission can be skipped without spending transmission latency.

The scheduling of tasks is performed by determining the release/completion time of task execution and data transmission in such a way to achieve the minimum application schedule length. Note that we assume static task execution time and a fixed amount of data for each transmission. Also, we assume that tasks are non-preemptive and no deadline requirements are enforced in our application. Therefore, for a pair of tasks u, v having precedence relation, the produced data from task u is transmitted to the subsequent task v after the completion of the task u .

We assume XY-routing in 2D topology and XYZ-routing in 3D topology as a default routing path. In this work, to achieve better latency through the extended path diversification, we explore the impact of mixed dimension-order routing on the application schedule length (l_{app}). For 2D and 3D topology, we employ all possible routing paths (i.e., XY/YX routing paths for 2D, also known as O1TURN [23], and XYZ/YXZ, ZXY/ZYX, and XZY/YZX routing paths for 3D). Then, in our formulation, we decide one of the dimension-order routing paths that can guarantee flit contention-free routing by detecting and avoiding temporal and spatial overlaps of SMART-hop paths. When we inject the traffic to the network in the scheduled time slot, we only activate the SSR signals along the path we already have decided.

Figure 4 illustrates the impact of the flit contention on application schedule lengths. For the same example TG in Figure 3(a), the mapping and scheduling solution in Figure 4(a) provides a larger l_{app} than that of Figure 4(b) due to the flit contention on its routing paths (i.e., $t_0 \rightarrow t_1/t_0 \rightarrow t_2$ and $t_2 \rightarrow t_3/t_2 \rightarrow t_4$). Based on the above definitions, our task mapping and scheduling problem can be defined as:

Given a TG and MTG, find a contention-free mapping and scheduling function from TG to MTG so that the overall end-to-end latency of the designed NoC application (i.e., application schedule length l_{app}) is minimized.

2.3 Expressiveness of SMT for Conditional Constraints

SMT provides the feature of **OMT (Optimization Modulo Theories)** [2] to obtain an optimal solution, on top of the efficient problem-solving ability of SAT. Besides, SMT provides much more expressive modeling language (e.g., “if-then-else” for the “Either-Or” constraint, built-in Boolean cardinality functions such as “at-most k” and “at-least k”, etc.) than is possible with SAT or ILP formulas. For example, to formulate a simple conditional constraint, “if $a \leq b$ then $c = 1$ else $c = 0$ ”, SMT only requires a single-statement as:

$$(\text{assert } (\text{ite } (<= a b) (= c 1) (= c 0))), \quad (3)$$

while ILP needs six constraints with one auxiliary variable as:

$$\begin{aligned} a + (z - 1)M &\leq b; & a + zM &\geq b + \epsilon \\ 1 - M(1 - z) &\leq c; & c &\leq 1 + M(1 - z) \\ & & -zM &\leq c; & c &\leq zM, \end{aligned} \quad (4)$$

where M is a large constant chosen as an upper bound on $a - b$, ϵ is a small number stating the tolerance for when a is considered to exceed b , and z is an auxiliary Boolean (0,1) variable which indicates if $a = b$. Therefore, as the conditional constraints become more complicated, more auxiliary variables and the corresponding constraints are necessary for the ILP formulation. Since our problem contains several complex conditional constraints as well as Boolean decision variables that have a significant impact on exploring the feasible solutions, SMT can solve our problem more efficiently than ILP under (i) the powerful expressiveness (i.e., the lower formulation complexity) and (ii) the faster reasoning ability of SAT solver.

3 SMT FORMULATION FOR JOINT TASK MAPPING, SCHEDULING, AND SMART ROUTING

In this section, we describe basic SMT formulation and scalability improvement constraints for 2D/3D SMART NoC. We formulate task mapping and scheduling of 2D/3D SMART NoC as a constraint satisfaction problem (CSP) with variables and constraints. Thus, the release time of task execution and a data transmission (i.e., scheduling), as well as the task assignment (i.e., task mapping), are both determined by our constraints. The formulations that can be adopted for both 2D and 3D mesh topologies by simple adjustments of the conditions related to the corresponding coordinate variables (i.e., x , y , and z) are expressed based on the 3D mesh topology. For the constraints which have to be carefully revised according to the dimension of topology and routing schemes, we provide separate expressions and algorithms. The notations are shown in Table 1.

3.1 Basic Formulation

3.1.1 Objective. As described in Section 2.2, our goal is to find a contention-free mapping and scheduling solution so that the overall end-to-end latency of the designed NoC application (i.e.,

Table 1. Notations for the Proposed SMT Formulation

Term	Description
T, E, P	Set of Tasks, Edges, and PEs (processing elements)
t	t th task
e	e th edge
p	p th PE
x^t/y^t	x/y -coordinates of a processor on which a task t is mapped (for 2D)
$x^t/y^t/z^t$	$x/y/z$ -coordinates of a processor on which a task t is mapped (for 3D)
$e_{u,v}$	A directed edge from task u to v , $\forall u, v \in T$
τ^t	The execution time of task t
M^e	The amount of data transferred on edge e
s^t/f^t	The release/completion time of task t
$s_{u,v}^e/f_{u,v}^e$	The release/completion time of data transmission on $e_{u,v}$
α_{e_i,e_j}	0-1 indicator if edges e_i, e_j have a transmission time overlap
β_{e_i,e_j}	0-1 indicator if edges e_i, e_j have shared routing paths
r^e	The type of dimension-order routing on edge e (for mixed routing, 0-1 for 2D/0-5 for 3D)

application schedule length l_{app}) is minimized. Thus, our objective function minimizes the maximum completion time of tasks that have no out-going edges.

$$\begin{aligned}
 & \text{Min}\{l_{app}\} \\
 & l_{app} = \text{Max}\{f^t | t \in T_{final}\} \\
 & T_{final}: \text{a set of tasks } t \text{ without out-going edges, } \forall t \in T.
 \end{aligned} \tag{5}$$

3.1.2 Boundary Condition for Processor Coordinates. Given an $m \times n \times l$ PE tiles, the $x/y/z$ -coordinates of a task t are bounded by $m/n/l$, respectively. Each task can only be mapped to one PE on (x^t, y^t, z^t) , while each PE can accommodate multiple tasks without limitation on the number of assigned tasks.

$$0 \leq x^t \leq m-1; 0 \leq y^t \leq n-1; 0 \leq z^t \leq l-1, \forall t \in T. \tag{6}$$

3.1.3 Scheduling Constraints of Tasks. Constraint (7) represents the quantitative timing relation of tasks. The source task u between two consequential tasks u, v have to be released prior to the destination task v

$$\begin{aligned}
 & f^u = s^u + \tau^u, \forall u \in T \\
 & s^u < s^v, \forall u, v \in T, \forall e_{u,v} \in E.
 \end{aligned} \tag{7}$$

3.1.4 Non-Overlap of Tasks. No pairs of tasks $u, v \in T$, mapped on the same PE, can overlap:

$$(s^u \geq f^v) \vee (f^u \leq s^v), \text{ if } (x^u = x^v) \wedge (y^u = y^v) \wedge (z^u = z^v), \forall u, v \in T. \tag{8}$$

3.1.5 Scheduling Constraints of Data Transmissions. Constraint (9) represents the quantitative timing relation between tasks and data transmissions. We assume that the data transmission can be released and completed in any time slot between the completion of the precedent task and the release of the succeeding task. If the source and destination tasks are mapped on the same PE, the data can be directly transmitted without spending additional latency. Otherwise, the minimum data transmission latency is required as expressed in Constraint (10).

$$s_{u,v}^e \geq f^u; \quad s^v \geq f_{u,v}^e, \quad \forall u, v \in T, \forall e_{u,v} \in E \quad (9)$$

$$\begin{cases} f_{u,v}^e \geq s_{u,v}^e, & \text{if } (x^u = x^v) \wedge (y^u = y^v) \wedge (z^u = z^v) \\ f_{u,v}^e \geq s_{u,v}^e + t_r + t_w \cdot M^e, & \text{otherwise} \end{cases}, \quad \forall u, v \in T, \forall e_{u,v} \in E \quad (10)$$

3.1.6 Non-Overlap of Data Transmission. No pairs of data transmissions on edges $e_i, e_j \in E$ can overlap in time and space at the same time.

$$(\text{at} - \text{most1})\left(\alpha_{e_i, e_j}, \beta_{e_i, e_j}\right), \quad \forall e_i, e_j \in E. \quad (11)$$

3.1.7 Overlap of Data Transmission in Time. Constraint (12) determines the overlap of data transmissions on any pairs of edges $e_i, e_j \in E$ in time:

$$\begin{cases} \alpha_{e_i, e_j} = \text{true}, & \text{if } (s_{u_i, v_i}^{e_i} < f_{u_j, v_j}^{e_j}) \wedge (f_{u_i, v_i}^{e_i} > s_{u_j, v_j}^{e_j}) \\ \alpha_{e_i, e_j} = \text{false}, & \text{otherwise} \end{cases}, \quad \forall u, v \in T, \forall e_i, e_j \in E \quad (12)$$

3.1.8 Overlap of Data Transmission in Space. The overlap of data transmissions in space is determined by the shared routing paths between any pairs of edges $e_i, e_j \in E$. Since we assume bi-directional links, only the shared links in the same direction are detected as an overlap. Constraint (13) determines the horizontal and vertical sharing of 2D routing paths under XY-routing. Constraint (14) determines the sharing of 3D routing paths under XYZ-routing. For simplification, detailed constraints for detecting overlaps in the x and y directional links that are the same as the Constraint (13) are not expressed in Constraint (14).

$$\begin{cases} \beta_{e_i, e_j} = \text{true}, & \text{if } \left\{ (y^{u_i} = y^{u_j}) \wedge \right. \\ & \left[(x^{u_i} \leq x^{u_j}) \wedge (x^{v_i} > x^{u_j}) \wedge (x^{v_j} > x^{u_j}) \right] \vee [(x^{u_i} \leq x^{u_j}) \wedge (x^{v_i} < x^{u_i}) \wedge (x^{v_j} < x^{u_i})] \vee \\ & [(x^{u_i} \geq x^{u_j}) \wedge (x^{v_i} > x^{u_i}) \wedge (x^{v_j} > x^{u_i})] \vee [(x^{u_i} \geq x^{u_j}) \wedge (x^{v_i} < x^{u_j}) \wedge (x^{v_j} < x^{u_j})] \Big\} \\ & \vee \left\{ (x^{v_i} = x^{v_j}) \wedge \right. \\ & \left[(y^{v_i} \leq y^{v_j}) \wedge (y^{u_i} > y^{v_j}) \wedge (y^{u_j} > y^{v_j}) \right] \vee [(y^{v_i} \leq y^{v_j}) \wedge (y^{u_i} < y^{v_i}) \wedge (y^{u_j} < y^{v_i})] \vee \\ & [(y^{v_i} \geq y^{v_j}) \wedge (y^{u_i} > y^{v_i}) \wedge (y^{u_j} > y^{v_i})] \vee [(y^{v_i} \geq y^{v_j}) \wedge (y^{u_i} < y^{v_j}) \wedge (y^{u_j} < y^{v_j})] \Big\} \\ \beta_{e_i, e_j} = \text{false}, & \text{otherwise} \end{cases}, \quad \forall u, v \in T, \forall e_i, e_j \in E \quad (13)$$

$$\begin{cases} \beta_{e_i, e_j} = \text{true}, & \text{if } \left\{ (y^{u_i} = y^{u_j}) \wedge (z^{u_i} = z^{u_j}) \wedge \left(\text{Constraints for detecting shared links in } x\text{-direction} \right) \right\} \\ & \vee \left\{ (x^{v_i} = x^{v_j}) \wedge (z^{u_i} = z^{u_j}) \wedge \left(\text{Constraints for detecting shared links in } y\text{-direction} \right) \right\} \\ & \vee \left\{ (x^{v_i} = x^{v_j}) \wedge (z^{v_i} = z^{v_j}) \wedge \right. \\ & \left[(z^{v_i} \leq z^{v_j}) \wedge (z^{u_i} > z^{v_j}) \wedge (z^{u_j} > z^{v_j}) \right] \vee [(z^{v_i} \leq z^{v_j}) \wedge (z^{u_i} < z^{v_i}) \wedge (z^{u_j} < z^{v_i})] \vee \\ & [(z^{v_i} \geq z^{v_j}) \wedge (z^{u_i} > z^{v_i}) \wedge (z^{u_j} > z^{v_i})] \vee [(z^{v_i} \geq z^{v_j}) \wedge (z^{u_i} < z^{v_j}) \wedge (z^{u_j} < z^{v_j})] \Big\} \\ \beta_{e_i, e_j} = \text{false}, & \text{otherwise} \end{cases}, \quad \forall u, v \in T, \forall e_i, e_j \in E \quad (14)$$

Fig. 5. Example of adaptive boundary condition.

3.1.9 Non-Overlap of Data Transmission on the Same PE. No pairs of outgoing data transmissions from the same PE with different source tasks on edges $e_i, e_j \in E$ can overlap.

$$\begin{aligned}
 & (s_{u_i, v_i}^{e_i} \geq f_{u_j, v_j}^{e_j}) \vee (f_{u_i, v_i}^{e_i} \leq s_{u_j, v_j}^{e_j}), \\
 & \text{if } (x^{u_i} = x^{u_j}) \wedge (y^{u_i} = y^{u_j}) \wedge (z^{u_i} = z^{u_j}) \wedge (s_{u_i, v_i}^{e_i} \neq f_{u_i, v_i}^{e_i}) \wedge (s_{u_j, v_j}^{e_j} \neq f_{u_j, v_j}^{e_j}), \\
 & u_i \neq u_j, \forall u, v \in T, \forall e_i, e_j \in E.
 \end{aligned} \tag{15}$$

3.1.10 Maximum Number of Hops (HPC_{max}). The number of hops between any source/destination pairs of tasks $u, v \in T$ have to be less than or equal to HPC_{max} . Manhattan distance is used to estimate the number of hops between two consequential tasks as expressed in Constraint (16)

$$|x^u - x^v| + |y^u - y^v| + |z^u - z^v| \leq HPC_{max}, \forall u, v \in T, \forall e_{u,v} \in E. \tag{16}$$

3.2 Scalability Improvements

3.2.1 Adaptive Boundary Condition. The adaptive boundary condition reduces the search-space by narrowing the feasible time-ranges of task release-time variables. For each task, we set the low-bound as the maximum sum of task execution times on the paths from any starting tasks (i.e., tasks with no incoming edges) to the target task. For example, in Figure 5, the low-bound of $T4$ is defined as the sum of $T1$ and $T3$'s execution times. To set the upper bound, we first respectively define feasible solution boundaries f_{min} and f_{max} as the sum of task execution times on the longest path and the sum of f_{min} and the offset. The offset is empirically determined by examining the maximum gap between f_{min} and the estimated maximum upper-bound of each test case.² Note that the offset needs to be increased if there exists a specific test case that has the larger actual upper-bound than f_{min} +offset (i.e., infeasible condition). Since our goal is minimizing the application schedule length, the offset does not affect the optimality of solutions if there exist any feasible solution with the given offset. In this work, we set the offset to 500 satisfying all the experimental cases. Once the f_{min} and the offset are determined, the upper bound of each task is defined as a subtraction of the maximum sum of task execution times on the reversed paths from any finishing tasks (i.e., tasks with no outgoing edges) to the target task from f_{max} . For example, the upper bound of $T1$ in Figure 5 is the sum of $T4$, $T3$, and $T1$'s execution times.

3.2.2 Breaking Design Symmetry. The breaking design symmetry constraint excludes redundant exploration of the symmetric solutions by restricting PE assignments of tasks to the specific region, resulting in the reduction of search-space. Figure 6 depicts examples of symmetric task mapping patterns in an $m \times n$ 2D mesh topology. The example mapping in Figure 6(a) has several

²In this work, we estimate the maximum upper-bound of each test case as the sum of all task execution time divided by the minimum number of PEs in our experiments (i.e., 16 for 4×4 mesh) assuming full PE resource utilization with direct data transmissions.

Fig. 6. Example of the symmetric task mappings on 2D $m \times n$ topology.

Fig. 7. Restriction of PE assignments for excluding symmetric mapping patterns on 2D/3D topologies.

symmetric task mappings that are equivalent to their rotated (i.e., Figure 6(b)) and flipped shapes (i.e., Figure 6(c)). In an asymmetric topology (i.e., $m \neq n$), the symmetric cases are limited to the rotation in 180° and flip in horizontal/vertical direction.

To exclude symmetric mapping patterns, we first sort task elements by the descending order of the number of incoming/outgoing edges of each task so that the following symmetry breaking constraint can reduce as many search-spaces of related tasks as possible. Then, we set the PE assignment boundary condition for the first task element to be assigned to the specific region of the topology as illustrated in Figure 7. For the remaining task elements, we recursively set the conditional constraints to keep excluding symmetric cases even if previously constrained tasks are mapped to the PEs on a line or plane that can cut the entire topology in half. Algorithm 1 describes conditional constraints excluding symmetric task mapping cases for 2D/3D mesh topology. Given a sorted queue T_s of tasks in T as descending number of in/out-degree, we first set the boundary condition for the PE assignment of the first task element (i.e., the task with the largest in/out-degree) T_1^s to the 4th octant (Lines 4–5). Then, we recursively set conditional constraints for the next task elements according to the intermediate PE assignments of previous task elements. If previous task elements are assigned on a diagonal line or plane in the topology with symmetric XY-plane (i.e., $m = n$), the location of the next task elements is restricted to the half area divided by the diagonal line or plane (Lines 7–9). When previous task elements are on one of horizontal line

ALGORITHM 1: Exclusion of symmetric task mapping cases for 2D/3D mesh topology**Input:** a task graph $TG = (T, E)$, # of tasks N_t , a SMART NoC with $m \times n / m \times n \times l$ tiles for 2D / 3D**Data:** a sorted queue of tasks in T as descending number of in/out-degree: T_s , a k_{th} task in T_s : T_k^s /* $(x, y) / (x, y, z)$ coordinate of the upper-left corner of 2D / 3D NoC tiles are $(0, 0) / (0, 0, 0)$ */

```

1:  $m_h \leftarrow \left\lceil \frac{m}{2} \right\rceil - 1, n_h \leftarrow \left\lceil \frac{n}{2} \right\rceil - 1, l_h \leftarrow \left\lceil \frac{l}{2} \right\rceil - 1$  (for 3D only)
2: for  $k = 1$  to  $N_t - 1$  do
3:   if  $k = 1$  then
4:      $(x^{T_1^s} \leq \left\lceil \frac{m_h}{n_h} \cdot y^{T_1^s} \right\rceil) \wedge (y^{T_1^s} \leq n_h);$  ▷ Set Constraint for 2D only
5:      $(x^{T_1^s} \leq \left\lceil \frac{m_h}{n_h} \cdot y^{T_1^s} \right\rceil) \wedge (y^{T_1^s} \leq n_h) \wedge (z^{T_1^s} \leq l_h);$  ▷ Set Constraint for 3D only
6:   else
7:     if  $(m = n) \wedge (\bigwedge_{i=1}^{k-1} (x^{T_i^s} = y^{T_i^s}) = \text{true})$  then
8:        $x^{T_k^s} \leq y^{T_k^s};$  ▷ Set Conditional Constraint
9:     end if
10:    if  $(\bigwedge_{i=1}^{k-1} (x^{T_i^s} = m_h) = \text{true}) \wedge (m = \text{odd})$  then
11:       $x^{T_k^s} \leq m_h;$  ▷ Set Conditional Constraint
12:    end if
13:    if  $(\bigwedge_{i=1}^{k-1} (y^{T_i^s} = n_h) = \text{true}) \wedge (n = \text{odd})$  then
14:       $y^{T_k^s} \leq n_h;$  ▷ Set Conditional Constraint
15:    end if
16:    if  $(\bigwedge_{i=1}^{k-1} (z^{T_i^s} = l_h) = \text{true}) \wedge (l = \text{odd})$  then
17:       $z^{T_k^s} \leq l_h;$  ▷ Set Conditional Constraint for 3D only
18:    end if
19:  end if
20: end for

```

or plane (i.e., x , y , and z directions) dividing the entire topology in half, the location of the next task elements is restricted to the half area divided by the horizontal line or plane (Lines 10–18).

3.3 Mixed 2D/3D Dimension-order Routing

3.3.1 Overlap of Data Transmission in Space for 2D Mixed Routing. For 2D NoC, we allow the mixed-use of XY/YX routing paths. Routing type indicator r^e is determined for each edge so that the application schedule length is minimized. r^e is set to 1 if the determined routing type is XY-routing. Otherwise (i.e., YX-routing), r^e is set to 0. Constraint (17) determines the horizontal and vertical sharing of routing paths under mixed XY/YX-routing.

$$\left\{ \begin{array}{l} \beta_{e_i, e_j} = \text{true}, \quad \text{if } \left\{ \left[[(r^{u_i} = 0) \wedge (r^{u_j} = 0) \wedge (y^{v_i} = y^{v_j})] \vee [(r^{u_i} = 0) \wedge (r^{u_j} = 1) \wedge (y^{v_i} = y^{v_j})] \vee \right. \right. \\ \quad \left. \left[(r^{u_i} = 1) \wedge (r^{u_j} = 0) \wedge (y^{v_i} = y^{v_j}) \right] \vee [(r^{u_i} = 1) \wedge (r^{u_j} = 1) \wedge (y^{v_i} = y^{v_j})] \right\} \wedge \\ \quad \left(\text{Constraints for detecting shared links in } x\text{-direction in Constraint (13)} \right) \\ \vee \left\{ \left[[(r^{u_i} = 0) \wedge (r^{u_j} = 0) \wedge (x^{u_i} = x^{u_j})] \vee [(r^{u_i} = 0) \wedge (r^{u_j} = 1) \wedge (x^{u_i} = x^{u_j})] \vee \right. \right. \\ \quad \left. \left[(r^{u_i} = 1) \wedge (r^{u_j} = 0) \wedge (x^{u_i} = x^{u_j}) \right] \vee [(r^{u_i} = 1) \wedge (r^{u_j} = 1) \wedge (x^{u_i} = x^{u_j})] \right\} \wedge \\ \quad \left(\text{Constraints for detecting shared links in } y\text{-direction in Constraint (13)} \right) \\ \beta_{e_i, e_j} = \text{false}, \quad \text{otherwise} \end{array} \right. , \forall u, v \in T, \forall e_i, e_j \in E \quad (17)$$

3.3.2 Overlap of Data Transmission in Space for 3D Mixed Routing. For 3D NoC, we allow the mixed-use of XYZ/YXZ, ZXY/ZYX, and XZY/YZX routing paths. The routing type indicator r^e is, respectively, set as 0 to 5 for XYZ, YXZ, ZXY, ZYX, XZY, and YZX routing paths. Constraint (18)

determines the sharing of links under mixed routing paths. Detailed constraints in the y and z directional links that can be defined similar to the x -directional constraint are not described in the Constraint (18).

$$\left\{ \begin{array}{l} \beta_{e_i, e_j} = \text{true}, \\ \text{if } \left\{ \left[(r^{u_i} = 0) \wedge \{ ((r^{u_j} = 0) \wedge (y^{u_i} = y^{u_j}) \wedge (z^{u_i} = z^{u_j})) \vee ((r^{u_j} = 1) \wedge (y^{u_i} = y^{v_j}) \wedge (z^{u_i} = z^{u_j})) \vee \right. \right. \\ \quad ((r^{u_j} = 2) \wedge (y^{u_i} = y^{u_j}) \wedge (z^{u_i} = z^{v_j})) \vee ((r^{u_j} = 3) \wedge (y^{u_i} = y^{u_j}) \wedge (z^{u_i} = z^{u_j})) \vee \\ \quad \left. \left. ((r^{u_j} = 4) \wedge (y^{u_i} = y^{v_j}) \wedge (z^{u_i} = z^{v_j})) \vee ((r^{u_j} = 5) \wedge (y^{v_i} = y^{v_j}) \wedge (z^{u_i} = z^{u_j})) \right\} \vee \right. \\ \quad [(r^{u_i} = 1) \wedge \{ ((r^{u_j} = 0) \wedge (y^{v_i} = y^{u_j}) \wedge (z^{u_i} = z^{u_j})) \vee ((r^{u_j} = 1) \wedge (y^{v_i} = y^{v_j}) \wedge (z^{u_i} = z^{u_j})) \vee \\ \quad ((r^{u_j} = 2) \wedge (y^{v_i} = y^{u_j}) \wedge (z^{u_i} = z^{v_j})) \vee ((r^{u_j} = 3) \wedge (y^{v_i} = y^{v_j}) \wedge (z^{u_i} = z^{v_j})) \vee \\ \quad \left. \left. ((r^{u_j} = 4) \wedge (y^{v_i} = y^{u_j}) \wedge (z^{u_i} = z^{u_j})) \vee ((r^{u_j} = 5) \wedge (y^{v_i} = y^{v_j}) \wedge (z^{u_i} = z^{v_j})) \right\} \vee \right. \\ \quad [(r^{u_i} = 2) \wedge \{ ((r^{u_j} = 0) \wedge (y^{u_i} = y^{u_j}) \wedge (z^{v_i} = z^{u_j})) \vee ((r^{u_j} = 1) \wedge (y^{u_i} = y^{v_j}) \wedge (z^{v_i} = z^{u_j})) \vee \\ \quad ((r^{u_j} = 2) \wedge (y^{u_i} = y^{u_j}) \wedge (z^{v_i} = z^{v_j})) \vee ((r^{u_j} = 3) \wedge (y^{u_i} = y^{v_j}) \wedge (z^{v_i} = z^{v_j})) \vee \\ \quad \left. \left. ((r^{u_j} = 4) \wedge (y^{u_i} = y^{u_j}) \wedge (z^{v_i} = z^{u_j})) \vee ((r^{u_j} = 5) \wedge (y^{u_i} = y^{v_j}) \wedge (z^{v_i} = z^{v_j})) \right\} \vee \right. \\ \quad [(r^{u_i} = 3) \wedge \{ ((r^{u_j} = 0) \wedge (y^{v_i} = y^{u_j}) \wedge (z^{v_i} = z^{u_j})) \vee ((r^{u_j} = 1) \wedge (y^{v_i} = y^{v_j}) \wedge (z^{v_i} = z^{u_j})) \vee \\ \quad ((r^{u_j} = 2) \wedge (y^{v_i} = y^{u_j}) \wedge (z^{v_i} = z^{v_j})) \vee ((r^{u_j} = 3) \wedge (y^{v_i} = y^{v_j}) \wedge (z^{v_i} = z^{v_j})) \vee \\ \quad \left. \left. ((r^{u_j} = 4) \wedge (y^{v_i} = y^{u_j}) \wedge (z^{v_i} = z^{u_j})) \vee ((r^{u_j} = 5) \wedge (y^{v_i} = y^{v_j}) \wedge (z^{v_i} = z^{v_j})) \right\} \vee \right. \\ \quad [(r^{u_i} = 4) \wedge \{ ((r^{u_j} = 0) \wedge (y^{u_i} = y^{u_j}) \wedge (z^{u_i} = z^{u_j})) \vee ((r^{u_j} = 1) \wedge (y^{u_i} = y^{v_j}) \wedge (z^{u_i} = z^{u_j})) \vee \\ \quad ((r^{u_j} = 2) \wedge (y^{u_i} = y^{u_j}) \wedge (z^{u_i} = z^{v_j})) \vee ((r^{u_j} = 3) \wedge (y^{u_i} = y^{v_j}) \wedge (z^{u_i} = z^{v_j})) \vee \\ \quad \left. \left. ((r^{u_j} = 4) \wedge (y^{u_i} = y^{u_j}) \wedge (z^{u_i} = z^{u_j})) \vee ((r^{u_j} = 5) \wedge (y^{u_i} = y^{v_j}) \wedge (z^{u_i} = z^{v_j})) \right\} \vee \right. \\ \quad [(r^{u_i} = 5) \wedge \{ ((r^{u_j} = 0) \wedge (y^{v_i} = y^{u_j}) \wedge (z^{v_i} = z^{u_j})) \vee ((r^{u_j} = 1) \wedge (y^{v_i} = y^{v_j}) \wedge (z^{v_i} = z^{u_j})) \vee \\ \quad ((r^{u_j} = 2) \wedge (y^{v_i} = y^{u_j}) \wedge (z^{v_i} = z^{v_j})) \vee ((r^{u_j} = 3) \wedge (y^{v_i} = y^{v_j}) \wedge (z^{v_i} = z^{v_j})) \vee \\ \quad \left. \left. ((r^{u_j} = 4) \wedge (y^{v_i} = y^{u_j}) \wedge (z^{v_i} = z^{u_j})) \vee ((r^{u_j} = 5) \wedge (y^{v_i} = y^{v_j}) \wedge (z^{v_i} = z^{v_j})) \right\} \right\} \wedge \\ \quad \left(\text{Detecting shared links in } x\text{-direction} \right) \} \vee \left\{ \text{Detecting shared links in } y\text{-direction in mixed routing} \right\} \\ \vee \left\{ \text{Detecting shared links in } z\text{-direction in mixed routing} \right\} \\ \beta_{e_i, e_j} = \text{false}, \text{ otherwise} \end{array} \right. \quad , \forall u, v \in T, \forall e_i, e_j \in E \quad (18)$$

4 EXPERIMENTAL RESULTS

4.1 Experimental Setup

We have implemented the proposed framework in both *ILP/SMT* formulas including (i) the basic formulations and (ii) the scalability improvement constraints for 2D topology. Then, we have extended SMT framework to 3D topology and implemented mixed dimension-order routing scheme for both 2D/3D frameworks. Our frameworks are validated on a workstation with Intel Xeon E5-2650L at 1.8 GHz and 128 GB memory. The *Gurobi* (version 9.0.2) [10] and *Z3* (version 4.8.5) [7] solvers are used to produce the optimized solutions for ILP and SMT, respectively.

We employ applications from (i) randomly generated cases by TGFF tool [8] and (ii) real benchmarks, including MWD (multi-window display), H263 encoder/MP3 decoder, H263 decoder/MP3 decoder, MP3 encoder/decoder, MMS(multi-media system) [11], Robot (Newton-Euler dynamic control calculation), Sparse (Random sparse matrix solver), and RS-32 encoder (Reed-Solomon code encoder) [20]. We consider 4×4 , 6×6 , 8×4 , 8×8 , and 16×16 2D Mesh and $4 \times 4 \times 4$ and $8 \times 8 \times 4$ 3D Mesh for 2D/3D SMART NoC architecture and assume the homogeneous PEs with the same execution efficiency. The random applications are generated with the maximum in/out-degree of 2/2 or 2/3 (suffixed with “_a”). We use $HPC_{max} = 8$ that is the best achievable for 2D configurations when energy is taken into consideration in [14]. HPC_{max} may affect the

Table 2. Formulation Complexity of ILP and SMT on 2D/3D SMART NoC

Main Variables	#Variables (ILP/SMT 2D/3D)					
Tasks	$4 \cdot T $ (2D) / $5 \cdot T $ (3D)					
Data Transmissions	$2 \cdot E $					
Overlap Flags	$ E ^2 - E $					
Constraints	ILP 2D			ILP 3D		SMT 2D/3D
	#Var (Aux)	#Constraints	#Var (Aux)	#Constraints	#Var (Aux)	#Constraints
Timing (tasks)	-	$ E + T $	-	$ E + T $	-	$ E + T $
Non-overlap (tasks)	$\frac{1}{2} \cdot (T ^2 - T)$	$5 \cdot (T ^2 - T)$	$5 \cdot (T ^2 - T)$	$\frac{13}{2} \cdot (T ^2 - T)$	-	$\frac{1}{2} \cdot (T ^2 - T)$
Timing (data trans.)	$7 \cdot E $	$16 \cdot E $	$10 \cdot E $	$19 \cdot E $	-	$4 \cdot E $
Non-overlap (data trans.)	-	$\frac{1}{2} \cdot (E ^2 - E)$	-	$\frac{1}{2} \cdot (E ^2 - E)$	-	$\frac{1}{2} \cdot (E ^2 - E)$
Overlap in time	$ E ^2 - E $	$3 \cdot (E ^2 - E)$	$ E ^2 - E $	$3 \cdot (E ^2 - E)$	-	$\frac{1}{2} \cdot (E ^2 - E)$
Overlap in space	$\frac{3}{2} \cdot (E ^2 - E)$	$\frac{9}{2} \cdot (E ^2 - E)$	$\frac{6}{2} \cdot (E ^2 - E)$	$\frac{12}{2} \cdot (E ^2 - E)$	-	$\frac{1}{2} \cdot (E ^2 - E)$
Non-overlap on same PE	$\frac{13}{2} \cdot (E ^2 - E)$	$\frac{41}{2} \cdot (E ^2 - E)$	$8 \cdot (E ^2 - E)$	$12 \cdot (E ^2 - E)$	-	$\frac{1}{2} \cdot (E ^2 - E)$
Breaking symmetry	$\frac{1}{2} \cdot T $	$6 \cdot T $	$4 \cdot T $	$6 \cdot T $	-	$ T $
Total	ILP 2D			ILP 3D		SMT
#Variables (Main + Aux)	$\frac{1}{2} \cdot T ^2 + 30 \cdot E ^2 + \frac{9}{2} \cdot T - 21 \cdot E $			$5 \cdot T ^2 + 45 \cdot E ^2 + 4 \cdot T - 30 \cdot E $		$ E ^2 + E + 4 \cdot T $
#Constraints	$5 \cdot T ^2 + \frac{109}{2} \cdot E ^2 + 2 \cdot T - \frac{75}{2} \cdot E $			$\frac{13}{2} \cdot T ^2 + \frac{157}{2} \cdot E ^2 + \frac{1}{2} \cdot T - \frac{117}{2} \cdot E $		$\frac{1}{2} \cdot T ^2 + 2 \cdot E ^2 + \frac{3}{2} \cdot T + 3 \cdot E $

$|T|$ = #Tasks, $|E|$ = #Edges, #Var (Aux) = # of auxiliary variables.

Table 3. Formulation Complexity Evaluation on 2D/3D SMART NoC

TestCase	Tasks	Edges	2D						3D					
			#Variables			#Constraints			#Variables			#Constraints		
			ILP		SMT	ILP		SMT	ILP		SMT	ILP		SMT
			Total	Aux	inc.	Total	SMT	inc.	Total	Aux	inc.	Total	SMT	inc.
tgff1	10	9	1,675	1,545	130	12.9×	2,776	256	10.8×	2,474	2,334	140	17.7×	3,962
tgff2	22	24	10,345	9,657	688	15.0×	17,358	1,485	11.7×	15,714	15,004	710	22.1×	25,578
tgff3	31	34	20,512	19,198	1,314	15.6×	34,406	2,918	11.8×	31,158	29,813	1,345	23.2×	50,672
tgff4	41	43	33,826	31,770	2,056	16.5×	56,653	4,702	12.0×	51,952	49,855	2,097	24.8×	84,655
tgff5	51	62	64,808	60,698	4,110	15.8×	109,401	9,210	11.9×	100,078	95,917	4,161	24.1×	164,093
tgff20	201	245	1,013,036	951,962	61,074	16.6×	1,712,204	141,183	12.1×	1,592,759	1,531,484	61,275	26.0×	2,624,757
tgff35	351	435	3,171,862	2,980,798	191,064	16.6×	5,363,446	441,700	12.1×	4,995,915	4,804,500	191,415	26.1×	8,237,709
tgff50	501	606	6,224,082	5,854,236	369,846	16.8×	10,519,070	862,296	12.2×	9,823,519	9,453,172	370,347	26.5×	16,202,741
tgff3_a	30	36	22,077	20,625	1,452	15.2×	37,295	3,182	11.7×	34,197	32,715	1,482	23.1×	56,169
tgff4_a	40	47	38,049	35,633	2,416	15.7×	64,204	5,399	11.9×	59,298	56,842	2,456	24.1×	97,458
tgff5_a	51	62	64,728	60,618	4,110	15.7×	109,552	9,225	11.9×	101,388	97,227	4,161	24.4×	167,174
tgff10_a	102	118	239,515	225,065	14,450	16.6×	403,236	33,474	12.0×	369,642	355,090	14,552	25.4×	604,757
MWD	12	12	2,683	2,479	204	13.2×	4,534	411	11.0×	4,112	3,896	216	19.0×	6,766
H263encMP3dec	12	12	2,667	2,463	204	13.1×	4,483	410	10.9×	3,969	3,753	216	18.4×	6,452
MP3encMP3dec	13	13	3,148	2,914	234	13.5×	5,323	477	11.2×	4,853	4,606	247	19.6×	8,000
H263decMP3dec	14	14	3,637	3,371	266	13.7×	6,147	547	11.2×	5,553	5,273	280	19.8×	9,130
MMS	40	48	38,809	36,297	2,512	15.4×	66,001	5,602	11.8×	61,450	58,898	2,552	24.1×	102,030
Robot	88	131	263,483	245,839	17,644	14.9×	449,888	38,620	11.6×	421,139	403,407	17,732	23.8×	700,969
Sparse	96	67	104,507	99,567	4,940	21.2×	164,117	13,907	11.8×	151,931	146,895	5,036	30.2×	229,848
RS-32_8_enc	262	348	1,990,333	1,867,833	122,500	16.2×	3,354,633	277,795	12.1×	3,187,636	3,064,874	122,762	26.0×	5,254,697
Average						16.6×			12.1×				26.2×	

inc. = increment ratio (ref. = SMT).

resource utilization if the number of outgoing edges from a task exceeds the number of available PEs within HPC_{max} (i.e., $4 \cdot \sum_{k=1}^{HPC_{max}}$), resulting in the diminished parallelism and performance. In our experiments, the maximum number of outgoing edges from one task is not restricted by $HPC_{max} = 8$ for all cases.

4.2 ILP vs. SMT for 2D/3D SMART NoC

4.2.1 Formulation Complexity Analysis. Table 2 presents the formulation complexity of ILP/SMT frameworks for 2D/3D SMART NoCs. The number of variables and constraints is significantly related to the number of tasks $|T|$ and edges $|E|$. The main variables of our framework consist of x/y -coordinates, release/completion time of each task/data transmission, and overlap indicators in time/space. The coordinate and time variables are proportional to $|T|$ and $|E|$ while the overlap indicators are related to the number of combinations between edges (i.e., $\frac{|E| \cdot (|E|-1)}{2}$). The conditional constraints which describe the overlap between pairs of tasks and edges are

Table 4. Minimum Application Schedule Length and Simulation Runtime of ILP and SMT on 2D SMART NoC

TestCase	Tasks	Edges	Min. l_{app}		Simulation Runtime(s)											Spd.Up
			ILP	SMT	ILP					SMT						
					4 × 4	6 × 6	8 × 4	8 × 8	Avg.	4 × 4	6 × 6	8 × 4	8 × 8	Avg.		
tgff1	10	9	173	173	0.37	0.73	0.16	0.21	0.37	0.06	0.07	0.07	0.07	0.07	5.5×	
tgff2	22	24	240	240	53.21	163.94	96.11	42.84	89.03	1.46	1.77	1.27	1.66	1.54	57.8×	
tgff3	31	34	248	248	746.28	524.37	1,332.02	2,221.27	1,205.99	4.03	4.62	4.21	4.38	4.31	279.7×	
tgff4	41	43	334	334	10,400.94	3,396.40	12,181.23	10,608.73	9,146.83	9.75	10.09	10.10	10.04	9.99	915.3×	
tgff5	51	62	-	365						18.29	18.79	20.53	18.35	18.99	-	
tgff20	201	245	-	764						1,519.11	857.79	754.49	842.35	993.44	-	
tgff35	351	435	-	854	t.o.	t.o.	t.o.	t.o.	-		29,894.46	t.o.	6,209.88	18,052.17	-	
tgff50	501	606	-	903						t.o.	t.o.	t.o.	42,755.09	42,755.09	-	
tgff3_a	30	36	294	294	9,241.54	3,119.83	7,450.77	6,977.98	6,697.53	5.63	6.21	6.37	6.27	6.12	1094.1×	
tgff4_a	40	47	366	366	11,949.99	7,692.02	10,293.86	3,818.10	8,438.49	9.31	10.15	11.15	9.79	10.10	835.6×	
tgff5_a	51	62	449	449			30,221.38	37,003.00	33,612.19	20.59	21.45	24.36	21.35	21.94	1532.1×	
tgff10_a	102	118	-	383	t.o.	t.o.		t.o.	-	414.41	125.62	71.44	145.88	189.34	-	
MWD	12	12	281	281	9.34	11.09	19.93	16.69	14.26	0.84	0.77	0.85	0.79	0.81	17.6×	
H263encMP3dec	12	12	235	235	0.76	0.94	0.62	0.62	0.74	0.31	0.39	0.30	0.35	0.34	2.2×	
MP3encMP3dec	13	13	251	251	0.74	0.45	0.49	0.45	0.53	0.21	0.28	0.19	0.24	0.23	2.3×	
H263decMP3dec	14	14	219	219	4.42	3.44	0.98	3.62	3.12	0.53	0.64	0.51	0.57	0.56	5.5×	
MMS	40	48	325	325	417.33	254.15	281.47	487.79	360.19	7.44	7.83	9.09	7.48	7.96	45.2×	
Robot	88	131	-	617						146.59	151.49	138.60	138.43	143.78	-	
Sparse	96	67	-	240	t.o.	t.o.	t.o.	t.o.	-	10,366.52	16,721.70	15,150.02	38,713.89	21,934.04	-	
RS-32_28_8_enc	262	348	-	1704						2,524.37	2,608.05	2,456.07	2,600.79	2,547.32	-	
Average															931.1×	

Min. l_{app} = minimum application schedule length. Spd.Up = average runtime speed up ratio (ref. = ILP), t.o. = optimization not completed within 12 hours.

Table 5. Minimum Application Schedule Length and Simulation Runtime of ILP and SMT on 3D SMART NoC

TestCase	Tasks	Edges	Solution		Simulation Runtime (s)						
			ILP	SMT	ILP			SMT			Spd.Up
					4 × 4 × 4	8 × 8 × 4	Avg.	4 × 4 × 4	8 × 8 × 4	Avg.	
tgff1	10	9	173	173	0.16	1.50	0.83	0.10	0.11	0.11	7.9×
tgff2	22	24	240	240	150.03	76.24	113.14	1.23	1.85	1.54	73.5×
tgff3	31	34	248	248	5,874.83	2,260.49	4,067.66	4.18	4.78	4.48	908×
tgff4	41	43	334	334	12,673.58	11,948.77	12,311.18	10.49	11.22	10.86	1134.1×
tgff5	51	62	-	365	t.o.	t.o.	-	32.80	30.43	31.62	-
tgff20	201	245	-	764	t.o.	t.o.	-	1,077.30	1,060.85	1,069.08	-
tgff35	351	435	-	854	t.o.	t.o.	-	5,229.47	4,872.55	5,051.01	-
tgff50	501	606	-	903	t.o.	t.o.	-	36,580.00	20,005.61	28,292.81	-
tgff3_a	30	36	294	294	28,641.77	35,916.15	32,278.96	6.61	8.15	7.38	4373.8×
tgff4_a	40	47	366	366	6,493.93	9,944.42	8,219.18	11.01	12.26	11.64	706.4×
tgff5_a	51	62	-	449	t.o.	t.o.	-	32.20	33.41	32.81	-
tgff10_a	102	118	-	383	t.o.	t.o.	-	112.70	123.90	118.30	-
MWD	12	12	281	281	14.87	21.74	18.31	0.74	0.95	0.85	21.7×
H263encMP3dec	12	12	235	235	0.95	1.26	1.11	0.23	0.29	0.26	4.3×
MP3encMP3dec	13	13	251	251	1.49	0.96	1.23	0.22	0.24	0.23	5.3×
H263decMP3dec	14	14	219	219	1.28	4.42	2.85	0.58	0.84	0.71	4×
MMS	40	48	325	325	851.10	822.47	836.79	8.12	9.33	8.73	95.9×
Robot	88	131	-	617	t.o.	t.o.	-	167.08	163.21	165.15	-
Sparse	96	67	-	228	t.o.	t.o.	-	167.43	120.81	144.12	-
RS-32_28_8_enc	262	348	-	1703	t.o.	t.o.	-	2,510.11	3,396.16	2,953.14	-
Average											1237.1×

Min. l_{app} = minimum application schedule length. Spd.Up = average runtime speed up ratio (ref. = ILP), t.o. = optimization not completed within 12 hours.

proportional to the number of combinations of tasks and edges. ILP requires auxiliary variables and the corresponding constraints for conditional constraints. In particular, as the dimension of the structure increases from 2D to 3D, the number of auxiliary variables and constraints in ILP also shows an increment because of the more complicated conditional constraints for determining

Fig. 8. Comparison of the formulation complexity and runtime scalability (8×8 mesh) between ILP and SMT for 2D SMART NoC.

Fig. 9. Comparison of the formulation complexity and runtime scalability ($8 \times 8 \times 4$ mesh) between ILP and SMT for 3D SMART NoC.

the overlap in tasks, data transmissions, whereas the SMT keeps the same formulation complexity. The number of additional variables and constraints is represented as the multiplication of the number of corresponding SMT constraints. Note that the final estimated complexity of ILP is further reduced by around 50% across the benchmarks because we remove the duplicated literals in conditional constraints.

4.2.2 Evaluation-Formulation Complexity. Table 3 presents the comparison of formulation complexity between ILP and SMT for 2D and 3D structures. Compared to ILP, SMT has 16.6 $\times$ and 12.1 $\times$ smaller number of variables and constraints on average for 2D, respectively. For 3D, SMT respectively shows 26.2 $\times$ and 18.7 $\times$ smaller number of variables and constraints. The difference is mainly due to the auxiliary variables which occupy 92% to 95% of total variables and the corresponding constraints of ILP. Figures 8(a) and 8(b) and Figures 9(a) and 9(b) visualize the estimated and measured complexity of ILP and SMT for $|E|$ in 2D and 3D structures, respectively. Note that we assume the same $|T|$ and $|E|$ for the estimation. As $|T|$ and $|E|$ increase, the estimated number of variables and constraints in ILP has respectively saturated to 17.1 $\times$ / 11.9 $\times$ and 25.3 $\times$ / 16.9 $\times$ larger values than those of SMT for 2D and 3D.

4.2.3 Evaluation-Solutions and Runtime. Tables 4 and 5 present the comparison of the minimum application schedule length and runtime between ILP and SMT with 12 hours of the time-limit for 2D and 3D SMART NoC, respectively. We observe that the application schedule length of both ILP and SMT are the same (i.e., equal application performance) for all cases regardless of mesh sizes. Figure 10 illustrates examples of detailed task mapping and scheduling solutions provided by ILP and SMT. The detailed solution includes information on the PE assignment and a designated release time for each task. Though the detailed composition of task mapping and scheduling of these solutions can be different from each other due to the possible existence of several feasible

Fig. 10. Example task mapping and scheduling of tgff2 case in Table 4 for 4×4 mesh. (a) SMT, (b) ILP.

solutions, both ILP and SMT provided the same minimum application schedule length of 240. The runtime trend tends to increase as the size of the mesh and the number of in/out-degree increases. For the cases that the ILP can provide an optimal solution, SMT solves problems with $931.1\times$ (ranges from $2.2\times$ to $1532.1\times$) and $1237.1\times$ (ranges from $4\times$ to $4373.8\times$) faster runtime on average than ILP for 2D and 3D NoCs, respectively. Figure 8(c) and Figure 9(c), which respectively visualize the comparison of the scalability for 8×8 and $8 \times 8 \times 4$ meshes, show that SMT achieves $10\times$ higher scalability up to 500 tasks than that of ILP up to 50 tasks within 12 hours. For the largest case in ILP (i.e., tgff5_a with 8×8 mesh and tgff4_a, MMS with $8 \times 8 \times 4$ mesh), SMT provides the solution up to $1532.1\times$ faster than ILP.

4.3 2D vs. 3D SMART NoC

In Section 4.2, we have demonstrated the superior scalability of SMT over ILP for 2D and 3D SMART NoC structures. Therefore, we use only the SMT framework for implementing a mixed dimension-order routing in the following experiments because the scalability of ILP is expected to be decreased for the mixed routing which involves more complicated conditional constraints as described in Section 3.3. $8 \times 8/16 \times 16$ and $4 \times 4 \times 4/8 \times 8 \times 4$ mesh topologies are used for 2D and 3D structures, respectively.

Table 6 presents the comparison of minimum application schedule length and simulation runtime of 2D and 3D SMART NoCs. Unlike the hop-by-hop transmission-based regular NoCs, l_{app} of SMART NoC is not affected by the reduced average number of hops by the extension to the 3D routing structure. Therefore, most of cases in Table 6 have the same l_{app} for 2D XY-only and 3D XYZ-only routing NoCs except for the random sparse matrix solver (i.e., “Sparse”) case. Figure 11 depicts a task mapping solution of “Sparse” case in $4 \times 4 \times 4$ 3D mesh. The arrow lines in the

Table 6. Minimum Application Schedule Length and Simulation Runtime on 2D/3D SMART NoC

TestCase	#Tasks	#Edges	Minimum Application Schedule Length (l_{app})								Simulation Runtime (s)							
			2D (XY Only)				2D (mixed)				3D (XYZ Only)				3D (mixed)			
			8 × 8	16 × 16	8 × 8	16 × 16	8 × 8	16 × 16	8 × 8	16 × 16	8 × 8	16 × 16	8 × 8	16 × 16	8 × 8	16 × 16	8 × 8	16 × 16
tgff5	51	62	365	365	365	365	365	365	365	365	18.35	37.06	30.22	31.06	32.80	30.43	59.97	59.60
tgff10	101	124	664	664	664	664	664	664	664	664	206.32	261.02	212.92	246.56	225.21	208.07	444.76	523.50
tgff20	201	245	764	764	764	764	764	764	764	764	842.35	1,302.41	1,260.40	1,294.33	1,077.28	1,060.85	1,978.02	2,140.43
tgff30	301	369	770	770	770	770	770	770	770	770	7,339.49	7,106.80	6,910.83	7,057.29	6,212.25	6,819.45	7,625.03	7,345.88
tgff40	401	495	818	818	818	818	818	818	818	818	15,961.35	14,131.35	30,817.41	22,887.27	21,563.24	16,004.14	33,160.45	22,708.54
tgff50	501	606	903	903	-	903	903	903	-	903	42,755.09	24,422.54	t.o.	34,112.05	36,579.98	20,005.61	t.o.	23,687.59
tgff5_a	51	62	449	449	440	440	449	449	440	440	21.35	43.95	28.09	23.02	32.19	33.41	49.52	61.47
tgff10_a	102	118	383	383	383	383	383	383	383	383	145.88	159.12	129.38	137.02	112.74	123.90	195.40	224.38
tgff20_a	203	244	492	492	492	492	492	492	492	492	1,025.82	1,655.52	1,260.40	1,319.78	1,206.36	1,193.70	2,301.11	2,275.93
tgff30_a	301	359	528	528	-	528	528	528	528	528	35,734.00	9,799.09	t.o.	5,982.25	9,876.86	5,628.88	13,287.17	7,407.61
MWD	12	12	281	281	281	281	281	281	281	281	0.79	1.93	1.25	2.12	0.74	0.95	3.20	3.42
H26encMP3dec	12	12	235	235	235	235	235	235	235	235	0.35	0.55	0.70	0.58	0.23	0.29	0.83	0.96
MP3encMP3dec	13	13	251	251	251	251	251	251	251	251	0.24	0.58	0.33	0.33	0.22	0.24	0.46	0.90
H26decMP3dec	14	14	219	219	219	219	219	219	219	219	0.57	0.99	1.02	0.79	0.58	0.84	0.68	1.42
MMS	40	48	325	325	325	325	325	325	325	325	7.48	17.36	15.55	14.61	8.12	9.33	29.39	37.07
Robot	88	131	617	617	615	615	617	617	615	615	138.43	273.21	133.66	193.48	167.08	163.21	215.62	248.07
Sparse	96	67	240	-	-	-	228	228	222	222	38,713.89	t.o.	t.o.	t.o.	167.43	120.81	1,338.58	2,047.77
RS-32_28_8_enc	262	348	1704	1704	1704	1704	1704	1704	1704	1704	2,600.79	3,734.78	2,672.55	4,862.26	2,510.11	3,396.16	2,957.08	4,464.72
Average			556.0	574.6	573.9	573.9	555.3	555.3	533.9	554.4	8,084.03	3,702.84	6,332.97	4,597.93	4,431.86	3,044.46	3,743.96	4,068.85

t.o. = optimization not completed within 12 hours.

Fig. 11. Example of the extension of routing paths from 2D to 3D routing (Sparse case in Table 4, l_{app} : 240@2D → 228@3D).

black and red colors illustrate incoming data transmissions from six PEs (i.e., P3, P17, P20, P22, P28, and P45) to P17 at a certain clock cycle in the detailed scheduling solution. The black and red arrow lines respectively indicate the transmissions on the same and different XY-planes. From this solution, we observe that the extended incoming paths up to six (i.e., four from the same and two from the different XY planes) on each PE by the extension of 2D to 3D routing paths enables further reduction of the minimum l_{app} from 240 to 228.

The simulation runtime of 3D XYZ-only routing shows a reduced trend compared to 2D XY-only routing as depicted in Figure 12(a) and Figure 13(a). We observe that the “tgff30_a” and “Sparse” cases require extraordinary simulation runtime to find an optimal solution in 8×8 2D topology while the runtime of their 3D counterparts follows the normal trend. Note that the TGs of these two cases have many more parallel task routing paths compared to the other randomly generated or real application TGs as shown in Figure 14. This high-level task parallelism can cause a larger fluctuation of runtime due to the existence of multiple combinations of symmetric paths with the same application schedule length as well as the nature of the exact method to solve NP-hard problems [21].

4.4 Mixed Dimension-Order Routing

In this section, we explore the impact of the mixed dimension-order routing in both 2D/3D topologies presented in Table 6. Three cases (i.e., “tgff5_a”, “Robot”, and “Sparse”) show the reduction in the minimum application schedule length due to the diversified routing paths of mixed dimension-order routing. Figure 15 illustrates the example of path diversification in the “Sparse”

Fig. 12. Runtime scalability (2D@ 8×8 mesh, 3D@ $4 \times 4 \times 4$ mesh). (a) 2D (XY-only) vs. 3D (XYZ-only), (b) 3D (XYZ-only) vs. 3D (mixed).

Fig. 13. Runtime scalability (2D@ 16×16 mesh, 3D@ $8 \times 8 \times 4$ mesh). (a) 2D (XY-only) vs. 3D (XYZ-only), (b) 3D (XYZ-only) vs. 3D (mixed).

Fig. 14. Task graph of (a) tgff30, (b) tgff30_a, (c) Robot, and (d) Sparse cases in Table 4 [8, 24].

case with $4 \times 4 \times 4$ 3D mesh. The arrow lines in the black and red colors illustrate incoming data transmissions from 5 PEs (i.e., P9, P20, P25, P33, and P55) to P21 at a certain clock cycle in the detailed scheduling solution. The red arrow lines indicate two data transmissions with different routing paths between the same source/destination PEs. From this solution, we observe that the diversified paths enable further reduction of the minimum l_{app} from 228 to 222.

The overall runtime scalabilities of the 2D/3D mixed routing show decreased trends compared to the 2D/3D single routing as depicted in Figures 12(b) and 12(c) and Figures 13(b) and 13(c). The additional variables for the routing path type indicator, the more complicated conditional constraints for detecting the link overlap in space, and the increased search-space due to the diversified routing paths have contributed to the increment in the simulation runtime. However, despite the slightly

Fig. 15. Example of path diversification from 3D XYZ-only to mixed routing (Sparse case in Table 4, $l_{app} : 228@XYZ \rightarrow 222@mixed$).

diminished scalability, the results demonstrate that our framework still successfully finds optimal task mapping and scheduling solutions up to 500 tasks within 12 hours.

5 RELATED WORK

The problem of task mapping and scheduling has been extensively studied in the literature [4, 12, 18, 19, 22, 28, 29]. Traditional approaches to task mapping [4, 12, 22, 28, 29] do not consider SMART NoCs where contention-free routing is required to fully realized the benefits of single-cycle multi-hop traversal. An optimal algorithm based on an integer linear programming (ILP) formulation was proposed in [27] for the 2D SMART NoC case. Although their ILP formulation can find optimal solutions, the runtimes are prohibitive for large problem instances. This is in part due to ILP's inability to express conditional constraints directly. On the other hand, our SMT-based formulation enables us to leverage SMT's ability to express conditional constraints succinctly, which enables us to derive a much more compact formulation and harness the logical reasoning power of SMT solvers. SMT's ability to capture conditional constraints also enables us to easily consider the 3D SMART NoC case and mixed dimension-order routing in our formulations, which were not considered in [27]. A polynomial-time heuristic algorithm was also proposed in [27] for 2D SMART NoCs. Although their heuristic algorithm often achieves good results, exploration of optimal solutions still plays an essential role in calibrating and evaluating heuristic approaches for more advanced and complicated system configurations. Recently, SMT-based scheduling optimization frameworks for 2D NoCs have been proposed [16, 26] to overcome the limited expressiveness of ILP for the conditional constraints that constitute a large proportion of the task mapping and scheduling problem, but these works did not consider mixed dimension-order routing nor the 3D SMART NoC case.

6 CONCLUSION

In this article, we develop an SMT-based task mapping and scheduling framework that guarantees contention-free data transmissions to achieve the optimal latency for 2D/3D SMART NoCs. Also, we develop link overlap detection constraints for the mixed dimension-order routing. We have reduced the formulation complexity by utilizing SMT's efficient modeling capability for the conditional constraints and also improved the scalability by introducing efficient search-space reduction techniques. We demonstrated that our SMT framework achieves 10× higher scalability than ILP, solving the problem within 12 hours up to 500 tasks for 2D and the 3D extension. Also, the 2D and 3D extensions of our SMT framework with the mixed dimension-order routing maintain the improved scalability with the diversified routing paths, resulting in the reduced latency through various application benchmarks. Lastly, we find that there are still rooms to further improve, e.g., the static task execution and data transmission time calls future research topics to accommodate the variability of real systems.

ACKNOWLEDGMENTS

The authors thank reviewers for providing valuable comments.

REFERENCES

- [1] Yashar Asgariéh and Bill Lin. 2019. Smart-Hop arbitration request propagation: Avoiding quadratic arbitration complexity and false negatives in SMART NoCs. *ACM Transactions on Design Automation of Electronic Systems* 24, 64 (2019), 64:1–64:25. <https://doi.org/10.1145/3356235>
- [2] Nikolaj Bjørner, Anh-Dung Phan, and Lars Fleckenstein. 2015. vZ-an optimizing SMT solver. In *Proceedings of TACAS*. 194–199. https://link.springer.com/chapter/10.1007/978-3-662-46681-0_14.
- [3] Chia-Hsin Owen Chen, Sunghyun Park, Tushar Krishna, Suvinay Subramanian, Anantha P. Chandrakasan, and Li-Shiuan Peh. 2013. SMART: A single-cycle reconfigurable NoC for SoC applications. In *Proceedings of DATE*. 1–6. <https://doi.org/10.7873/DATE.2013.080>
- [4] G. Chen, F. Li, S.W. Son, and M. Kandemir. 2008. Application mapping for chip multiprocessors. In *Proceedings of DAC*. 620–625. <https://doi.org/10.1145/1391469.1391628>
- [5] Xianmin Chen and Niraj K. Jha. 2016. Reducing wire and energy overheads of the SMART NoC using a setup request network. *IEEE Transactions on Very Large Scale Integration* 24, 10 (2016), 3013–3026. <https://doi.org/10.1109/TVLSI.2016.2538284>
- [6] William James Dally and Brian Patrick Towles. 2003. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers, Inc.
- [7] Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: An efficient SMT solver. In *Proceedings of TACAS*. 337–340. <https://dl.acm.org/doi/10.5555/1792734.1792766>.
- [8] R. P. Dick, D. L. Rhodes, and W. Wolf. 1998. TGFF: Task graphs for free. In *Proceedings of CODES*. 1–6. <https://doi.org/10.1109/HSC.1998.666245>
- [9] Brett Stanley Feero and Partha Pratim Pande. 2008. Networks-on-Chip in a three-dimensional environment: A performance evaluation. *Transactions on Computers* 58, 1 (2008), 32–45. <https://doi.org/10.1109/TC.2008.142>
- [10] LLC Gurobi Optimization. 2021. Gurobi Optimizer Reference Manual. Retrieved 2021 from <http://www.gurobi.com>.
- [11] Jingcao Hu and R. Marculescu. 2003. Energy-aware mapping for tile-based NoC architectures under performance constraints. In *Proceedings of ASP-DAC*. 1–6. <https://dl.acm.org/doi/10.1145/1119772.1119818>.
- [12] Jingcao Hu and R. Marculescu. 2004. Energy-aware communication and task scheduling for Network-on-Chip architectures under real-time constraints. In *Proceedings of DATE*. 234–239. <https://doi.org/10.1109/DATE.2004.1268854>
- [13] Biresh Kumar Joardar, Karthi Duraisamy, and Partha Pratim Pande. 2018. High performance collective communication-aware 3D Network-on-Chip architectures. In *Proceedings of DATE*. 1351–1356. <https://doi.org/10.23919/DATE.2018.8342223>
- [14] Tushar Krishna, Chia-Hsin Owen Chen, Woo Cheol Kwon, and Li-Shiuan Peh. 2013. Breaking the on-chip latency barrier using SMART. In *Proceedings of HPCA*. 378–389. <https://doi.org/10.1109/HPCA.2013.6522334>
- [15] Tushar Krishna, Chia-Hsin Owen Chen, Woo Cheol Kwon, and Li-Shiuan Peh. 2014. SMART: Single-cycle multihop traversals over a shared Network on Chip. *IEEE Micro* 34, 3 (2014), 43–56. <https://doi.org/10.1109/MM.2014.48>
- [16] Daeyeal Lee, Bill Lin, and Chung-Kuan Cheng. 2021. SMT-based contention-free task mapping and scheduling on SMART NoC. *Embedded Systems Letters* 1, 1 (2021), 1–4. <https://doi.org/10.1109/LES.2021.3049774>
- [17] Dongjin Lee, Das Sourav, Janardhan Rao Doppa, Partha Pratim Pande, and Krishnendu Chakrabarty. 2018. Performance and thermal tradeoffs for energy-efficient monolithic 3D Network-on-Chip. *ACM Transactions on Design Automation of Electronic Systems* 23, 5 (2018), 1–25. <https://dl.acm.org/doi/10.1145/3223046>
- [18] Edward Ashford Lee and David G. Messerschmitt. 1987. Static scheduling of synchronous data flow programs for digital signal processing. *IEEE Transactions on Computers* 100, 1 (1987), 24–35. <https://doi.org/10.1109/TC.1987.5009446>
- [19] Edward A. Lee and David G. Messerschmitt. 1987. Synchronous data flow. *Proc. IEEE* 75, 9 (1987), 1235–1245. <https://doi.org/10.1109/PROC.1987.13876>
- [20] Weichen Liu, Jiang Xu, Xiaowen Wu, Yaoyao Ye, Xuan Wang, Wei Zhang, Mahdi Nikdast, and Zhehui Wang. 2011. A NoC traffic suite based on real applications. In *Proceedings of ISVLSI*. 66–71. <https://doi.org/10.1109/ISVLSI.2011.49>
- [21] Andrea Lodi and Andrea Tramontani. 2014. Performance variability in mixed-integer programming. In *TutORials in Operations Research*. 1–12. <https://doi.org/10.1287/educ.2013.0112>
- [22] S. Murali and G. De Micheli. 2004. Bandwidth-constrained mapping of cores onto NoC architectures. In *Proceedings of DATE*. 896–901. <https://doi.org/10.1109/DATE.2004.1269002>
- [23] Daeho Seo, Akif Ali, Won-Taek Lim, and Nauman Rafique. 2005. Near-optimal worst-case throughput routing for two-dimensional mesh networks. In *Proceedings of ISCA*. 1–12. <https://doi.org/10.1109/ISCA.2005.37>
- [24] Takao Tobita and Hironori Kasahara. 2002. A standard task graph set for fair evaluation of multiprocessor scheduling algorithms. *Journal of Scheduling* 5, 5 (2002), 379–394. <http://dx.doi.org/10.1002/jos.116>

- [25] Geert Van der Plas, Paresh Limaye, Igor Loi, Abdelkarim Mercha, Herman Oprins, Cristina Torregiani, Steven Thijs, Dimitri Linten, Michele Stucchi, Guruprasad Katti, Dimitrios Velenis, Vladimir Cherman, Bart Vandeveld, Veerle Simons, Ingrid De Wolf, Riet Labie, Dan Perry, Stephane Bronckers, Nikolaos Minas, Miro Cupac, Wouter Ruythooren, Jan Van Olmen, Alain Phommahaxay, Muriel de Potter de ten Broeck, Ann Opdebeeck, Michal Rakowski, Bart De Wachter, Morin Dehan, Marc Nelis, Rahul Agarwal, Antonio Pullini, Federico Angiolini, Luca Benini, Wim Dehaene, Youssef Traval, Eric Beyne, and Paul Marchal. 2010. Design issues and considerations for low-cost 3-D TSV IC technology. *IEEE Journal of Solid-State Circuits* 46, 1 (2010), 293–07. <https://doi.org/10.1109/JSSC.2010.2074070>
- [26] Rongjie Yan, Anyu Cai, Hongyu Gao, Feifei Ma, and Jun Yan. 2019. SMT-based multi-objective optimization for scheduling of MPSoC applications. In *Proceedings of TASE*. 160–167. <https://doi.org/10.1109/TASE.2019.000-5>
- [27] Lei Yang, Weichen Liu, Peng Chen, Nan Guan, and Mengquan Li. 2017. Task mapping on SMART NoC: Contention matters, not the distance. In *Proceedings of DAC*. 1–6. <https://doi.org/10.1145/3061639.3062323>
- [28] Lei Yang, Weichen Liu, Nan Guan, and Nikil Dutt. 2019. Optimal application mapping and scheduling for Network-on-Chips with computation in STT-RAM based router. *IEEE Trans. Comput.* 68, 8 (2019), 1174–1189. <https://doi.org/10.1109/TC.2018.2864749>
- [29] Heng Yu, Yajun Ha, and Bharadwaj Veeravalli. 2010. Communication-aware application mapping and scheduling for NoC-based MPSoCs. In *Proceedings of ISCAS*. 3232–3235. <https://doi.org/10.1109/ISCAS.2010.5537920>

Received March 2021; revised August 2021; accepted September 2021