On Task Mapping in Multi-chiplet Based Many-core Systems to Optimize Inter- and Intra-chiplet Communications

Xiaohang Wang*, *Member, IEEE,* Yifan Wang, Yingtao Jiang, Amit Kumar Singh, *Member, IEEE* and Mei Yang, *Member, IEEE*

***Abstract*—Multi-chiplet system design, by integrating multiple chiplets/dielets within a single package, has emerged as a promising paradigm in the post-Moore era. This paper introduces a novel task mapping algorithm for multi-chiplet many-core systems, addressing the unique challenges posed by intra- and inter-chiplet communications under power and thermal constraints. Traditional task mapping algorithms fail to account for the latency and bandwidth differences between these communications, leading to sub-optimal performance in multi-chiplet systems. Our proposed algorithm employs a two-step process: (1) task assignment to chiplets using binary linear programming, leveraging a totally unimodular constraint matrix, and (2) intra-chiplet mapping that minimizes communication latency while considering both thermal and power constraints. This method strategically positions tasks with extensive inter-chiplet communication near interface nodes and centralizes those with predominant intra-chiplet communication. Experimental results demonstrate that the proposed algorithm outperforms existing methods (DAR and IOA) with a 37.5% and 24.7% reduction in execution time, respectively. Communication latency is also reduced by up to 43.2% and 32.9%, compared to DAR and IOA. These findings affirm that the proposed task mapping algorithm aligns well with the characteristics of multi-chiplet based many-core systems, and thus improves optimal performance.**

***Index Terms*—Many-core systems; Chiplet; Task mapping; Performance optimization**

I. Introduction

DESIGNING and fabricating larger and high-performance many-core chips face escalating challenges in scalability, flexibility, cost, power consumption, and manufacturability [6]. Multi-chiplet system integration, as a new and promising design paradigm, offers a viable solution to address these challenges. Instead of designing a monolithic chip with a large number of cores, chiplets with specific and often smaller functionalities can be designed and manufactured independently. They are then integrated and assembled together with an interposer, RDL (redistribution layer), or substrate-based inter-chiplet interconnection.

Target multi-chiplet systems can be customized as accelerators or software managed systems, like NN-Baton [1], Simba [2], INDM [3], Gemini [4], NetFlex [5], SPRINT [A6]. These chips are deployed in various domains, including data centers, cloud computing, high-performance computing, edge computing, IoT devices, consumer electronics, automotive electronics, and telecommunications and networking, for tasks like artificial intelligence or big data processing. The state-of-the-art multi-chiplet chips consist of a dozen or more chiplets with a combined transistor count of billions and an accumulated bandwidth exceeding terabytes per second. To fully utilize the hardware resources in multi-chiplet designs, efficient task mapping techniques are essential. Task mapping involves determining which tasks should be executed on specific chiplets based on factors such as workload characteristics, communication requirements, power consumption, and thermal considerations. The task mapping results significantly impact system performance and power consumption.

While successful task mapping algorithms have been proposed for monolithic 2D or 3D many-core systems [8], where only in-chip communication flows are present and treated indistinguishably, task mapping in multi-chiplet systems is more complex. These systems involve two types of communication flows: inter-chiplet and intra-chiplet communications. Inter-chiplet communication incurs higher costs due to the need to go through the package-level communication infrastructure and additional physical layer processing. It has been reported that inter-chiplet communication latency between nodes in adjacent chiplets can be 10-30 times longer than that between adjacent nodes in the same chiplet [9], [10]. The inter-chiplet communication bandwidth is also lower due to pin limits and area constraints. Failing to consider these significant latency and bandwidth gaps in task mapping can result in sub-optimal solutions and degraded system performance. The following example illustrates this issue.

The task mapping results of applying the task mapping algorithms [8] for 2D or 3D many-core systems are illustrated

X. Wang is with the State Key Laboratory of Blockchain and Data Security in Zhejiang University, China, and Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security (e-mail: xiaohangwang@zju.edu.cn). Xiaohang Wang is the corresponding author.

Y. Wang is with the State Key Laboratory of Blockchain and Data Security in Zhejiang University, China, and Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security (e-mail: 18066527728@163.com).

Y. Jiang and M. Yang are with the Department of Electrical and Computer Engineering, University of Nevada, Las Vegas, USA. (e-mail: yingtao.jiang@unlv.edu, mei.yang@unlv.edu)

A. K. Singh is with the School of Computer Science and Electronic Engineering, University of Essex, UK. (e-mail: a.k.singh@essex.ac.uk)

This work was supported in part by the National Natural Science Foundation of China under Grants 92373205 and 62374146, in part by the National Key Research and Development Program of China No. 2023YFB4404404 and 2021YFB0300900, in part by the by Ant Group through CCF-Ant Research Fund, in part by the Key Technologies R&D Program of Jiangsu (Prospective and Key Technologies for Industry) under Grant BE2023005-2 and BE2021003, and in part by CIE-Smartchip research fund No. 2023-004. The first two authors contributed equally to this work.

in Fig. 1. The goal of CoNA [8] is to decrease both internal and external congestion. If a task is mapped to a specific processing element (PE), the tasks it communicates with are also mapped to adjacent PEs. This approach reduces congestion in the network as communicating tasks are physically closer, reducing the communication distance. Furthermore, CoNA maps communicating tasks in close neighborhoods. The benefits of the CoNA algorithm are twofold: it effectively reduces congestion in the network, and thus reduces network latency. The task graph in Fig. 1(a) is mapped to a hypothetical 2-chiplet system (Fig. 1(b)) following the algorithm in [8]. The communication latency is calculated as the product of the total inter-task communication volume and the latency between the corresponding cores. Alternatively, if we allow the tasks to be assigned to chiplets to minimize inter-chiplet communication volume and map tasks with significant inter-chiplet communication requirements to interface nodes (nodes connected to other chiplets), we obtain a task mapping (Fig. 1(c)) with a much lower communication latency. It is only 0.79 times the communication latency of the mapping shown in Fig. 1(b) The inter-chiplet core latency is calculated based on the configuration and datasheet details provided in [9], [11], [12].

The example mentioned above highlights the importance of minimizing inter-chiplet communication when mapping tasks in multi-chiplet systems. Building on this observation, we propose a two-step runtime task mapping algorithm specifically designed for multi-chiplet systems.

In the first step, tasks are allocated to each chiplet to minimize inter-chiplet communication volume. This is formulated as a binary quadratic programming problem (BQP) and further transformed into a binary linear programming problem (BLP). The constraint matrix of the BLP is proven to be totally unimodular, allowing us to solve the problem optimally using the prime-dual interior point algorithm [13].

In the second step, the tasks within each chiplet are mapped to individual cores to minimize communication latency while considering thermal and power constraints. To achieve this, tasks with high inter-chiplet communication volumes are mapped to interface nodes (nodes connected to other chiplets), while tasks with high intra-chiplet communication volumes are mapped to the centers of the chiplets.

Experimental results demonstrate that the proposed method reduces application execution time by up to 33.5% and 21.1% compared to the two best-known baseline task mapping algorithms, DAR [14] and IOA [15], respectively. Additionally, communication latency is reduced by up to 39.0% and 25.8% compared to DAR and IOA, respectively.

The remainder of the paper is structured as follows. Section III and IV provide a survey of related works and covers the necessary preliminaries. In Section IV, the runtime task mapping problem is formally formulated, along with the overall algorithm structure. The next two sections, Sections V and VI, detail the two major steps of the proposed runtime mapping algorithm, which specifically address the assignment of tasks to chiplets and intra-chiplet mapping. Section VII presents the evaluation of experimental results. Finally, Section VIII concludes the paper.

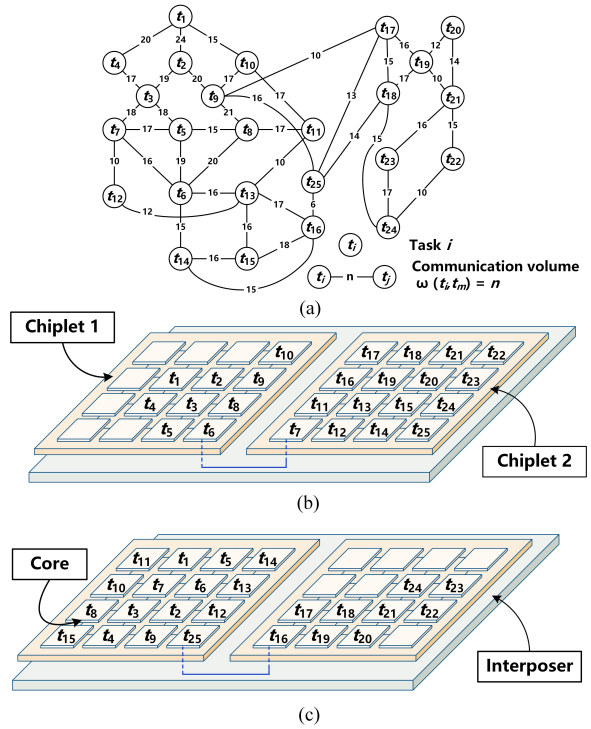


Fig. 1: Task mapping results. (a) Task graph; (b) mapping result from an existing task mapping algorithm [8]; (c) mapping result of an inter-chiplet communication minimizing task mapping algorithm.

II. Related Works

Task mapping algorithms have been proposed to optimize applications running in 2D, 3D and multi-chiplet-based many-core systems. These algorithms can be classified into three categories based on their optimization objectives:

1) Temperature-oriented optimization approaches [17]–[19]: The algorithms in [17] aim to minimize the temperature of the system by considering the thermal characteristics of the cores and their placement. The thermal-aware mapping approach in [18] reduces performance throttling and utilizes empty offset areas to expand on-chip cache capacity by spacing out chiplet hot spots away from each other in the 3D stack. However, thermal hotspot issues in the thermal-aware mapping approach configuration may give rise to elevated temperatures and consequent performance degradation. In compute-intensive workloads, the temperature peaks in the thermal-aware mapping approach configuration surpass the temperature limit, resulting in thermal throttling phenomena. This is achieved by employing techniques such as layout mirroring and offsetting. Meanwhile, the approaches in [19] focus on reducing power consumption and response time through a unified methodology integrating dynamic voltage and frequency scaling (DVFS) of processor cores and low power mode (LPM) for memory storage, coordinating a comprehensive thermal management strategy for the 3D stacked processor-memory system. However, in certain scenarios, the approaches in [19] may encounter anomalous power peaks, resulting in temperatures exceeding the thermal threshold. Additionally, the approaches in [19] exhibits significant power fluctuations during its execution process.

2) Communication-oriented optimization approaches [2], [3], [8]: The algorithms in [8] focus on reducing communication latency by considering the communication patterns between tasks and minimizing the distance between com-

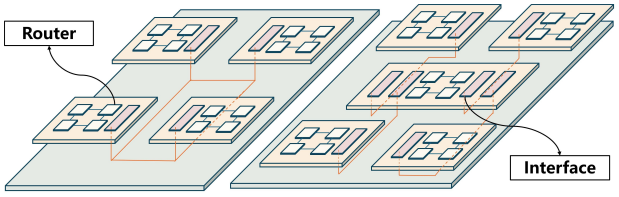


Fig. 2: (a) Tiled inter-chiplet network topology (passive interface); (b) centralized inter-chiplet network topology (active interface).

municating tasks. The approach in [3] aims to minimize traffic congestion during the switching of deep neural network (DNN) layers by emphasizing inter-chiplet communication. However, The approach was evaluated in experiments using only a subset of specific DNN models, and its applicability and performance for other types of models have not been sufficiently validated. Similarly, [2] addresses inter-chiplet communication latency by consolidating relevant data within the same region, effectively reducing the need for remote communication. However, the system's performance experiences degradation with a certain quantity of chiplets, as the inter-processing element (PE) communication latency surpass the limited parallelism.

3) Approaches balancing multiple objectives, such as performance, power consumption, and reliability, etc. [14], [15], [20]–[23], [42]: The algorithms in [14], [15] take into account both temperature and communication aspects, aiming to find a balance between minimizing temperature and reducing communication latency. C. Wu *et al.* [20] proposed an approach for the collaborative optimization of reliability, communication energy, and performance (CoREP) in network-on-chip (NoC)-based reconfigurable architectures. This method identifies the optimal mapping by exhaustively exploring all potential mappings through a search tree to minimize CoREP costs. In addition, [21] considers both energy consumption and reliability, utilizing a branch-and-bound method with a partial cost ratio to systematically determine the best mapping solution. Moreover, [22], [23] discuss a run-time resource manager (RTRM) framework designed to achieve multi-objective optimization by employing hierarchical adaptive mechanisms. P. Veda [42] *et al.* proposed a method that optimizes reliability by employing fault-tolerant mapping through spare cores in the Mesh-of-Tree (MoT) network to address core failures. It does not directly apply to chiplet systems, where inter- and intra-chiplet network topologies are substantially different from MoT.

III. Preliminaries

In this section, a system model is presented that describes the structure and components of a chiplet system, along with an application model that formally defines the applications to run on it. Chiplet systems face challenges due to thermal issues, which can negatively affect power consumption and performance. To address these challenges, a thermal power capacity model is introduced.

A. System Model

The multi-chiplet-based many-core system is modeled as an undirected graph $G = (C, L)$, where C is the set of nodes and L is the set of links connecting those nodes. A node is composed either of a core, L1 cache and L2 cache bank, a router and a network interface (NI) or a D2D (die-to-die) interface, as the case of a chiplet dedicated solely to interconnection, without any processing elements. The set of chiplets is represented as $R = \{R_1, R_2, \ldots, R_g\}$. A chiplet R_j has multiple cores. Each core $c_{ij} \in C$ is defined by coordinates $(x_{ij}, y_{ij}, p_j, q_j)$, where x_{ij} and y_{ij} are the horizontal and vertical coordinates of c_{ij} in the chiplet (intra-chiplet coordinate), and p_j and q_j are the horizontal and vertical coordinates of that chiplet in the system (chiplet coordinate). A chiplet R_j communicates with other chiplets through its cross-chiplet nodes, collectively referred to as $\mathcal{H}_j$. The weight $\omega^{\mathrm{C}}(c_{ij}, c_{mn})$ of each edge $l(c_{ij}, c_{mn}) \in L$ denotes the latency when data flows from core c_{ij} to core c_{mn}. Two typical types of inter-chiplet interconnections, termed tiled and centralized, are illustrated in Fig. 2(a). The communication latency between two cores $\omega^{\mathrm{C}}(c_{ij}, c_{mn})$ is obtained by summing the latency incurred at the routers and all the links along the routing path from c_{ij} to c_{mn}. Similarly, the communication latency $\omega^{\mathrm{R}}(R_i, R_j)$ of two chiplets R_i and R_j is obtained by adding the latencies of chiplets and the interposer/level links along the routing path from R_i to R_j.

A tiled topology (Fig. 2)(a) refers to a configuration in which multiple chiplets are organized in a regular mesh-like structure, forming a tiled arrangement. In this topology, each chiplet is treated as a node, with interconnections established through a network to enable communication and data exchange between them. This structured and scalable framework facilitates efficient coordination and optimized system performance, making it well-suited for multi-chiplet systems.

A centralized topology refers to a configuration in which multiple chiplets are interconnected through a central node. In this topology, each chiplet is linked to the central node, which facilitates communication and data exchange between the chiplets. The central node plays a crucial role in coordinating and managing the overall system operation. This topology allows for centralized control and coordination of chiplets, enabling efficient data transfer and synchronization among chiplets. This topology supports efficient communication and resource sharing, making it a suitable choice for multi-chiplet systems requiring streamlined communication and coordination.

In the centralized topology, as depicted in Fig. 2(b), each chiplet consists of interconnection and computation components. Additionally, some routers are dedicated to connecting with other chiplets through the physical layer (PHY). For instance, in the centralized topology, there is a central chiplet that features a 2×2 intra-chiplet mesh network. This central chiplet serves as the hub, responsible for connecting all the other computing chiplets, as illustrated in Fig. 2(b).

The inter- and intra-chiplet network follows a hierarchical design, composed by network-on-chip (NoC, also referred to as intra-chiplet network) and network-on-package (NoP) [24], which is demonstrated in the following figure.

The NoC is connected by buffered routers, with each chiplet utilizing D2D interfaces [25] to connect to other chiplets. The NoP is connected via wires in a passive interposer or routers in an active interposer.

TABLE I: Nomenclature

Symbol	Definition
$G = (C, L)$	The undirected graph that represents a many-core system, where C is the set of cores, and L is the set of links connecting the cores
c_{ij}	the i^{th} core in the j^{th} chiplet
$loc(c_{ij})$	The coordinates $(x_{ij}, y_{ij}, p_j, q_j)$ of core c_{ij}, where x_{ij} and y_{ij} denote the coordinates of c_{ij} in the chiplet with its coordinates marked as p_j and q_j in the entire inter-chiplet network (chiplet coordinate)
$P_{\text{th}}((c_{ij})$	The power capacity of core c_{ij}
$P_{\text{c}}((c_{ij})$	The power consumption of core c_{ij}
R_j	Chiplet j
$\mathcal{H}_j$	A set of the cores connecting to the routers in R_j
R	Chiplet set, i. e., $R = \{R_1, R_2, \ldots, R_g\}$
$\|R\|$	The number of chiplets
t_i	Task i
$M(t_i) = c_{kj}$	A mapping function that maps task t_i to core c_{kj}
$r(t_i)$	The chiplet that runs task t_i
$A = (T, E)$	A directed graph represents an application where T has all the tasks of application A, and E is the set of all the communications within application A
$\|T\|$	The number of all the tasks
$\|R_m\|$	The maximum number of tasks in the chiplet m
$l(c_{ij}, c_{mn})$	The undirected edge between cores c_{ij} and c_{mn}
$\omega^{c}(c_{ij}, c_{mn})$	The latency when data flows from cores c_{ij} and c_{mn}
$\omega^{\text{R}}(R_i, R_j)$	The latency between chiplets R_i and R_j
$e(t_i, t_m)$	The directed edge indicating that task t_i precedes task t_m
$v(t_i, t_m)$	The communication latency in bits from task t_i to task t_m
$det(\mathbf{W})$	The determinant of matrix $\mathbf{W}$
Φ_i	The i^{th} thermal correlation region of the chip ($i = 1, 2, 3, 4$)
$P_a(\Phi_i)$	The power consumption of region Φ_i

To prevent deadlock in the inter- and intra-chiplet network, the approach outlined in [26] is adopted.

Table I summarizes the symbols and their definitions used throughout this paper.

B. Application Model

Each application is modeled as an undirected graph $A = (T, E)$, where T is the set of tasks in the application and E is the set of directed edges representing the communications between tasks. Each task $t_i \in T$ is mapped to a core $c_{kj} \in C$ when the application arrives at the system and there are enough free cores. The weight $v(t_i, t_m)$ of each edge $e(t_i, t_m) \in E$ is the communication latency from task t_i to task t_m. A mapping function $M(t_i) = c_{kj}$ indicates that task $t_i \in T$ is assigned to run on core $c_{kj} \in C$.

C. The Thermal Power Capacity Model

Task mapping is constrained by the thermal/power capacity of the chip. In what follows, the thermal power capacity model (TPC model) in [16] is extended to apply to multi-chiplet-based many-core systems.

The power capacity (*i.e.*, the maximum allowable power consumption) of a core c_{ij} with coordinates $loc(c_{ij})$ is influenced by the power consumption of adjacent cores due to thermal correlation. However, depending on the physical distances, the neighboring cores of c_{ij} have different levels of thermal correlation with c_{ij}, and the combined thermal correlations of all its neighboring nodes ultimately determine the power capacity of c_{ij}.

As indicated in Fig. 3, the cores are divided into four regions based on their thermal correlations with respect to core c_{ij}.

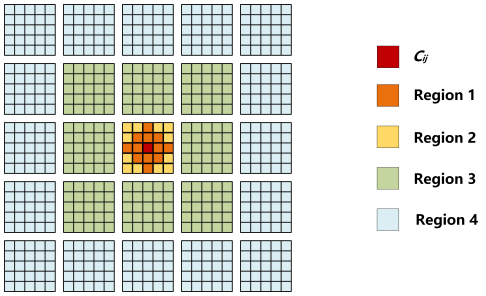


Fig. 3: Regions with different thermal correlations with respect to core c_{ij}

- The cores in region 1 are represented as Φ_1, where $\Phi_1 = \{c_{ij} \mid loc(c_{ij}) = (x_{ij} \pm \delta_1, y_{ij} \pm \delta_2, p_j, q_j)\}$, $\delta_1, \delta_2 \in \{0, 1, 2\}$ and $\delta_1 + \delta_2 \leq 2$.
- The cores in region 2 are represented as Φ_2, where $\Phi_2 = \{c_{ij} \mid loc(c_{ij}) = (x_{ij} \pm \gamma_1, y_{ij} \pm \gamma_2, p_j, q_j)\}$, $\gamma_1, \gamma_2 \leq 2$ and $\gamma_1 + \gamma_2 \leq 2$.
- The cores in region 3 are represented as Φ_3, where $\Phi_3 = \{c_{ij} \mid (p_j \pm \sigma_1, q_j \pm \sigma_2)\}$, $\sigma_1, \sigma_2 = 0$ or 1.
- The cores in region 4 are represented as Φ_4, where $\Phi_4 = \{c_{ij} \mid (p_j \pm \epsilon_1, q_j \pm \epsilon_2)\}$, $\epsilon_1, \epsilon_2 \geq 1$.

The power capacity of a core c_{ij} can be determined by the power consumption of the neighboring cores in the different thermal correlation regions of c_{ij}:

$$P_{\text{th}}(c_{ij}) = \zeta P_{\text{a}}(\Psi_1) + \iota P_{\text{a}}(\Psi_2) + \chi P_{\text{a}}(\Psi_3) + \kappa P_{\text{a}}(\Psi_4) \quad (1)$$

where ζ, ι, χ and κ are regression coefficients, $P_{\text{th}}(c_{ij})$ is the power capacity of core c_{ij}, and $P_{\text{a}}(\Psi_i)$ is the total power consumption of cores in region i.

The power consumption [16] of a core c_{ij} is determined as

$$P_{\text{c}}(c_{ij}) = \sum [g(c_{ij}, c_{mn}) \cdot (T(c_{ij}) - T(c_{mn}))] + \sum \left[g(c_{ij}, c_{ij}^{\text{amb}}) \cdot (T(c_{ij}) - T(c_{ij}^{\text{amb}})) \right] \quad (2)$$

where $g_{i,k}$ is the thermal conductance between core c_{ij} and its neighboring core c_{mn}, which may be within the same chiplet or across different chiplets. $T(c_{ij})$ and $T(c_{mn})$ are the temperatures of core c_{ij} and core c_{mn}, respectively, $g(c_{ij}, c_{ij}^{\text{amb}})$ is the thermal conductance to the ambient environment, $T(c_{ij}^{\text{amb}})$ is the ambient temperature.

The training data of this thermal power capacity model (Eqns. (1-2)) is obtained by running multiple applications in the multi-chiplet system and randomly varying the power consumption of the cores in the four regions. For each set of experiments, the maximum allowable power consumption of core c_{ij} is recorded as the output. Maximum likelihood regression is then used to obtain the values of the coefficients in Eqn. (1).

IV. Problem Formulation and Problem Mapping/Overall Algorithm Structure

A. Problem Definition

Given a chiplet system G, and applications represented as A, the goal is to map all tasks in A onto G in a way that minimizes the application execution time while meeting applicable thermal and power constraints.

$$\min \sum_{t=1}^{|A|} E(A_t) \quad (3)$$

subject to:

$$P_{\text{c}}(c_{ij}) \leq P_{\text{th}}(c_{ij}), \quad \text{for each } c_{ij} \in C \quad (4)$$

Due to the correlation between communication latency and application execution time [7] in chiplet systems, this paper utilizes communication latency as the primary objective function in the problem-solving process.

Here the communication latency is defined as the cumulative sum of the products of inter-task communication volume and communication latency, given below:

$$C_{\text{total}} = \sum_{i=1}^{|T|} \sum_{m=1}^{|T|} v(t_i, t_m) \cdot \omega^{\text{C}}(M(t_i), M(t_m)) \tag{5}$$

where C_{total} is the total communication latency of the system, $|T|$ is the number of tasks in the task graph of the application, $v(t_i, t_m)$ is the communication volume between task t_i and task t_m, $M(t_i)$ is the core running task t_i, and $\omega^{\text{C}}(M(t_i), M(t_m))$ is the communication latency between the two cores running tasks t_i and t_m.

The problem of task mapping is thus formulated as follows.

$$\text{Subject to:} \quad \min C_{\text{total}} \tag{6}$$

$$P_{\text{c}}(c_{ij}) \leq P_{\text{th}}(c_{ij}) \text{ for each } \text{c}_{ij} \in C \tag{7}$$

where C_{total} is the total communication latency of the system given in Eqn. (5), and the power consumption of a core, $P_{\text{c}}(c_{ij})$, as defined in Eqn. (2), must not exceed its power capacity $P_{\text{th}}(c_{ij})$.

B. Problem Mapping/Overall Algorithm Structure

The above-defined problem is a binary quadratic programming problem, which is considered to be NP-hard, with an exponential growth rate in terms of complexity. For example, for an application with only 16 tasks mapped onto a small system with 16 cores distributed across 4 chiplets, there are a total of 16^{16} possible options. To efficiently solve this problem, a two-step algorithm is proposed.

In the first step, tasks are assigned to chiplets to minimize inter-chiplet communication volume, and this sub-problem is solved optimally by an efficient algorithm [13] with polynomial time complexity.

In the second step, a heuristic approach is applied to map tasks within each chiplet. The heuristic prioritizes tasks with high inter-chiplet communication volume by mapping them to cores that connect to other chiplets, while tasks with high intra-chiplet communication volume are mapped to central cores within the chiplet.

V. Task to Chiplet Assignment

This step focuses on assigning tasks to chiplets in a way that minimizes inter-chiplet communication volume, considering the inherent cluster/community structure of a multi-chiplet system. Conventional graph partitioning or community detection algorithms [27] cannot be directly applied in this scenario because they assume that all sub-graphs or communities have the same size, implying that every chiplet in the system has the same number of cores. However, this assumption is too restrictive as most chiplet-based systems follow heterogeneous integration, and the chiplets can vary significantly from one another.

To address the size differences among multiple chiplets, this section defines the problem of task-to-chiplet assignment as a binary quadratic programming (BQP) problem. This problem is then transformed into a binary linear programming (BLP) problem to find an optimal assignment solution.

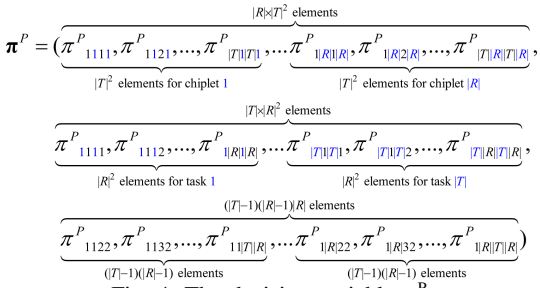


Fig. 4: The decision variable π^{P}.

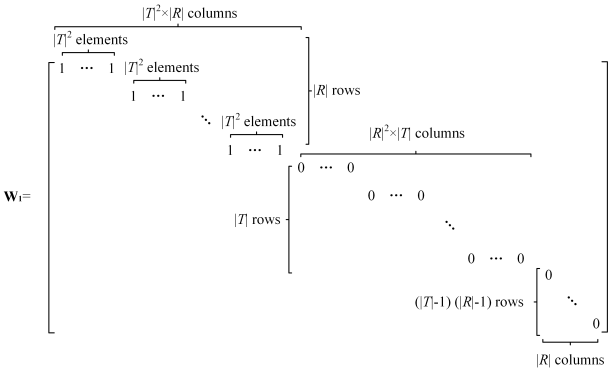


Fig. 5: The constraint matrices $\mathbf{W_1}$.

A. Problem Definition

In the task-to-chiplet assignment problem, given a chiplet system G, and applications represented as A, all the tasks in A need to be mapped to run on specific chiplets in G so that the inter-chiplet communication latency is minimized. In this case, the inter-chiplet communication latency $C_{\text{total}}^{\text{Chiplet}}$ is defined as,

$$C_{\text{total}}^{\text{Chiplet}} = \sum_{i=1}^{i\leq|T|} \sum_{j=1}^{j\leq|T|, i\neq j} \sum_{m=1}^{m\leq|R|} \sum_{n=1}^{n\leq|R|, n\neq m} (\pi_{im} \cdot \pi_{jn} \cdot v(t_i, t_j) \cdot \omega^{R}(R_m, R_n)) \tag{8}$$

where $|T|$ is the number of tasks, $v(t_i, t_j)$ is the communication volume between tasks t_i and t_j, $\omega^{\text{R}}(R_m, R_n)$ is the communication latency between chiplets R_m and R_n to which t_i and t_j are respectively mapped, and π_{im} is defined as

$$\pi_{im} = \begin{cases} 1, & \text{if } t_i \text{ is assigned to } R_m \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

The task to chiplet assignment problem thus is formulated as a BQP problem.

$$\text{Subject to:} \quad \min C_{\text{total}}^{\text{Chiplet}} \tag{10}$$

$$\begin{cases} \sum_{i=1}^{i\leq|T|} \pi_{im} \leq |R_m| \\ \sum_{m=1}^{m\leq|R|} \pi_{im} \leq 1 \\ \forall R_i, P_{\text{c}}(R_i) \leq P_{\text{th}}(R_i) \end{cases} \tag{11}$$

where $|R_m|$ is the core count of chiplet m, and $|T_m n^{\text{th}}|$ is the maximum number of tasks that can run on the core $c_m n$.

Let $\mathbf{\Pi}^{\text{P}}$, a matrix with a size of $(|T| \times |R|)^2$, represent the current mapping result, and its elements are defined in Eqn. (12).

$$\pi_{imjn}^{\text{P}} = \pi_{im} \times \pi_{jn} \tag{12}$$

Let $\mathbf{Q}$, a matrix of size $(|T| \times |R|)^2$, represent the communication latency, and its elements are defined in Eqn. (13),

$$q_{imjn} = v(t_i, t_j) \times \omega^{\text{R}}(R_m, R_n) \tag{13}$$

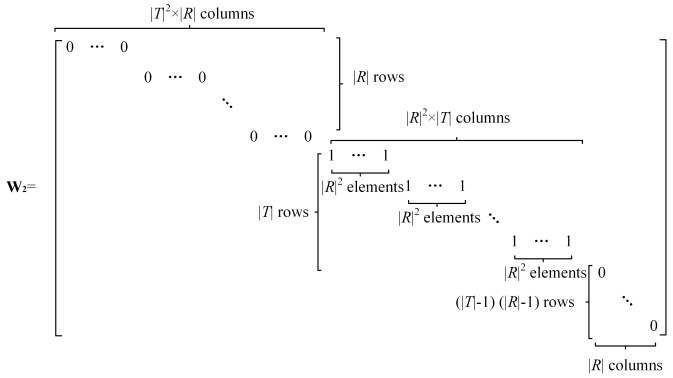


Fig. 6: The constraint matrices $\mathbf{W_2}$.

Note that $\omega^{\mathrm{R}}(R_m, R_m) = 0$. The above BQP problem can be transformed into a binary linear programming problem (BLP) as follows:

$$\min(C_{\text{total}}^{\text{Chiplet}}) = \sum_{i=1}^{|T|}\sum_{j=1}^{|T|}\sum_{m=1}^{|R|}\sum_{n=1}^{|R|} q_{imjn} \times \pi_{imjn}^{\mathrm{P}} \tag{14}$$

Subject to:

$$\begin{cases} \sum\limits_{i=1}^{i\le|T|}\sum\limits_{j=1}^{j\le|T|} \pi_{imjm}^{\mathrm{P}} \le 2|T_{mn}^{\text{th}}| \\ \sum\limits_{m=1}^{m\le|R|}\sum\limits_{m,n=1}^{n\le|R|} \pi_{imin}^{\mathrm{P}} \le 2 \\ \pi_{imjn}^{\mathrm{P}} \in \{0,1\}, i,j \le |T|, n \le |R| \\ \forall R_i, P_{\mathrm{c}}(R_i) \le P_{\text{th}}(R_i) \end{cases} \tag{15}$$

where q_{imjn} and π_{imjn}^{P} are defined in Eqns. (12)-(13) of the original paper, $|R_m|$ is the core count of chiplet R_m, and $|T_{mn}^{\text{th}}|$ is the maximum number of tasks that can run on the core c_{mn}.

Eqn. (15) ensures that the number of tasks mapped to the same chiplet R_m does not exceed the core count of the chiplet R_m, and the task t_i is assigned to only one chiplet. Note that since both π_{imjn}^{P} and π_{jnim}^{P} are counted twice in the above inequalities, which arises from the assignments of t_i to R_m and t_j to R_n, respectively, a factor of 2 has to be added into Eqn. (15). Eqn. (15) can be rewritten in matrix form,

$$\begin{cases} (\mathbf{W_1}\pi^{\mathrm{P}} \le \mathbf{2} \times (\underbrace{|R_1|, |R_2|, \ldots, |R_{|R|}|}_{|\mathbf{R}| \text{ elements}}, \underbrace{0, 0, \ldots, 0}_{|\mathbf{T}| \text{ elements}}, \underbrace{0, 0, \ldots, 0}_{(|\mathbf{T}|-\mathbf{1})(|\mathbf{R}|-\mathbf{1}) \text{ elements}})^{\mathrm{T}} \\ (\mathbf{W_2}\pi^{\mathrm{P}} \le \mathbf{2} \times (\underbrace{0, 0, \ldots, 0}_{|\mathbf{R}| \text{ elements}})^{\mathrm{T}}, \underbrace{2, 2, \ldots, 2}_{|\mathbf{T}| \text{ elements}}, \underbrace{0, 0, \ldots, 0}_{(|\mathbf{T}|-\mathbf{1})(|\mathbf{R}|-\mathbf{1}) \text{ elements}})^{\mathrm{T}} \end{cases} \tag{16}$$

where the constraint matrices $\mathbf{W_1}$ and $\mathbf{W_2}$ are defined in Figs. 5 and 6, and the decision variable π^{P} [28] is given by Fig. 4.

Eqn. (16) now can be rewritten as Eqn. (17),

$$\mathbf{W}\pi^{\mathrm{P}} \le 2 \times (\underbrace{|R_1|, |R_2|, \ldots, |R_{|R|}|}_{|R| \text{ elements}})^{\mathrm{T}}, \underbrace{(1, 1, \ldots, 1)}_{|T| \text{ elements}}, (\underbrace{0, 0, \ldots, 0}_{(|T|-1)(|R|-1) \text{ elements}})^{\mathrm{T}} \tag{17}$$

where the constraint matrix $\mathbf{W}$ is the sum of $\mathbf{W}_1$ and $\mathbf{W}_2$ as follows.

Definition 1: Let $\mathbf{A}$ be an $m \times n$ $\{0,1\}$ matrix. If $a_{i,j} = a_{i,k} = 1$ and $k > j+1$, for any $j < l < k$, $a_{i,l} = 1$, $\mathbf{A}$ possesses the "consecutive 1's property". In other words, every row of $\mathbf{A}$ is of the form $(0,\ldots,0,1,\ldots,1,0,\ldots,0)$.

Theorem 1: When matrix $\mathbf{A}$ has the "consecutive 1's property", it is a totally unimodular matrix [29].

Please refer to Section I of the supplementary material for the proof.

As so, the following linear relaxed programming (LRP) is a relaxed version of the binary linear programming (BLP) problem.

LPR:

$$\min C_{\text{total}}^{\text{Chiplet}} = \mathbf{q}^{\mathrm{T}}\pi^{P} \tag{18}$$

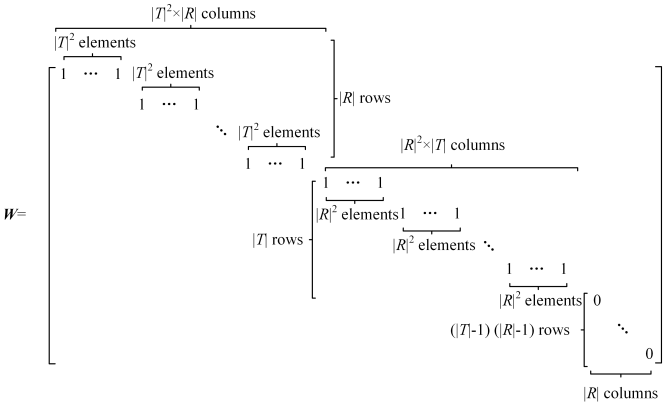


Fig. 7: The constraint matrices $\mathbf{W}$.

$$\mathbf{q} = (\overbrace{\underbrace{q_{1111}, q_{1121}, \ldots, q_{|T|1|T|1}}_{|T|^2 \text{ elements for chiplet } 1}, \ldots \underbrace{q_{1|R|1|R|}, q_{1|R|2|R|}, \ldots, q_{|T||R||T||R|}}_{|T|^2 \text{ elements for chiplet } |R|}}^{|R|\times|T|^2 \text{ elements}},$$
$$\overbrace{\underbrace{q_{1111}, q_{1112}, \ldots, q_{1|R||R|}}_{|R|^2 \text{ elements for task } 1}, \ldots \underbrace{q_{|T|1|T|1}, q_{|T|1|T|2}, \ldots, q_{|T||R||T||R|}}_{|R|^2 \text{ elements for task } |T|}}^{|T|\times|R|^2 \text{ elements}},$$
$$\overbrace{\underbrace{q_{1122}, q_{1132}, \ldots, q_{11|T||R|}}_{(|T|-1)(|R|-1) \text{ elements}}, \ldots \underbrace{q_{1|R|22}, q_{1|R|32}, \ldots, q_{1|R||T||R|}}_{(|T|-1)(|R|-1) \text{ elements}}}^{(|T|-1)(|R|-1)|R| \text{ elements}})$$

Fig. 8: The vector $\mathbf{q}$.

Subject to:

$$\begin{cases} \mathbf{W}\left((\pi^{P})^{\mathrm{T}}\right) \le \mathbf{b} \\ \pi_i^{P} \in \{0,1\}, \quad i \le |\pi^{P}| \end{cases} \tag{19}$$

where $\mathbf{b}$ and $\mathbf{q}$ are given in Eqn. (20) and Fig. 8, respectively.

$$\mathbf{b} = 2 \times \left(\underbrace{|R_1|, |R_2|, \ldots, |R_{|R|}|}_{|R| \text{ elements}}, \underbrace{2, 2, \ldots, 2}_{|T| \text{ elements}}, \underbrace{0, 0, \ldots, 0}_{(|T|-1)(|R|-1) \text{ elements}}\right)^{\mathrm{T}} \tag{20}$$

B. Task to Chiplet Assignment Algorithm

To solve the problem, we apply the prime-dual interior point algorithm in step 1. Step 2 is included to ensure no single chiplet exceeds the power consumption constraint, which requires checking the consumption of each chiplet.

The relaxed linear programming problem (LPR) defined in Eqn. (17) can be solved using the prime-dual interior point algorithm [13]. However direct application of the primal problem may involve a large number of iterations. To improve the convergence rate, the primal problem is transformed to its dual problem given in Eqns. (21-22).

Dual problem (DP):

$$\max \bar{C}_{\text{total}}^{\text{Chiplet}} = \mathbf{b}^{\mathrm{T}}\pi^{D} \tag{21}$$

Subject to:

$$\begin{cases} \mathbf{W}^{\mathrm{T}}\pi^{D} \ge \mathbf{q} \\ \pi_i^{D} \in \{0,1\} \end{cases} \tag{22}$$

where π^{D} is the solution of the dual problem.

The interior point algorithm [13] finds a path (dubbed central path) within the feasible region in which the solutions gradually converge to the optimal solution. During this process, the constraints of the feasible region [30] appear as the penalty term of the objective function. Specifically, the LPR in Eqn. (17) becomes Eqn. (23-24).

The unconstrained problem (UC) of DP:

$$\max \mathbf{b}^{\mathrm{T}}\pi^{D} + \mu \sum_{i=1}^{|\pi^{\mathrm{D}}|} \ln s_i \tag{23}$$

Subject to:

$$\begin{cases} \mathbf{W}^{\mathrm{T}}(\pi^D)^{\mathrm{T}} + \mathbf{s} = \mathbf{q} \\ \mathbf{s} \geq 0, \pi^D \text{ is unrestricted} \end{cases} \tag{24}$$

where $\mathbf{s}$ is the slack variable to catch the points in the feasible region closer to the center, s_i is the element of $\mathbf{s}$, and μ is a positive hyper-parameter that controls the strength of penalty. Note that the penalty is given as $\mu \sum_{i=1}^{|\pi^{\mathrm{D}}|} \ln s_i$.

The presence of the penalty term influences the value of the objective function, which corresponds to a specific point within the feasible region. The value of the penalty term will increase as $\mathbf{s}$ approaches 0. As a result, the interior point method will push the center point of the feasible domain towards the origin.

The algorithm finds the optimal solution of $(\pi^{\mathrm{P}}, \pi^{\mathrm{D}}, \mathbf{s})$ iteratively, and the solution after the k^{th} iteration is denoted as $(\pi^{P(k)}, \pi^{D(k)}, \mathbf{s}^k)$. Starting from some initial points in the feasible region, the iteration will continue its search in the feasible region [13] with the penalty gradually decreasing. Consequently, the constraints of the feasible region are gradually satisfied.

The initial point $((\pi^P)^0, (\pi^D)^0, \mathbf{s}^0)$ can be computed as follows.

$$\begin{cases} (\pi^P)^0 = \dfrac{1}{|\widetilde{\pi}|}\mathbf{e} \\ (\pi^D)^0 = \dfrac{1}{|\hat{\pi}|}\mathbf{e} \\ \mathbf{s}^0 = \dfrac{\mathbf{q}^{\mathrm{T}}\mathbf{W}}{|\mathbf{s}|} \end{cases} \tag{25}$$

where $\mathbf{e}$ is a unit vector, $\mathbf{q}$ is the coefficient vector of the objective function, and $\mathbf{W}$ is the constraint matrix. For the k$^{\text{th}}$ ($k > 1$) iteration, the following steps are performed.

Step 1.1: Calculating the search directions.

The search directions of $((\pi^{\mathrm{P}})^k, (\pi^{\mathrm{D}})^k, s^k)$ are denoted as $d_{(\pi^{\mathrm{P}})^k}$, $d_{(\pi^{\mathrm{D}})^k}$, and d_{s^k}, respectively, and they are determined as:

$$\begin{cases} \mathbf{d}_{(\pi^{\mathrm{P}})^k} = (\mathbf{W}\hat{\mathbf{D}}_k^2\mathbf{W}^{\mathrm{T}})(\mathbf{W}\hat{\mathbf{D}}_k^2(\mathbf{g}^k - \mathbf{p}^k) + \tau^k) \\ \mathbf{d}_{s^k} = \mathbf{g}^k - \mathbf{W}^{\mathrm{T}}\mathbf{d}_{(\pi^{\mathrm{D}})^k} \\ \mathbf{d}_{(\pi^{\mathrm{D}})^k} = \hat{\mathbf{D}}_k^2(\mathbf{p}^k - \mathbf{d}_{s^k}) \end{cases} \tag{26}$$

where $\hat{\mathbf{D}}_k$ is an intermediate matrix to update the direction of search, $\mathbf{g}^k$ is the gradient vector of the objective function in DP, $\mathbf{p}^k$ is the penalty term of the objective function, τ is a step size to control the extent of the dual variable update in each iteration, and τ^k is the value of τ in the k-th iteration.

Step 1.2: Updating the intermediate variables.

$f^k = \mu^k\mathbf{e} - \mathbf{\Pi}_k^{\mathrm{P}}\mathbf{S}_k\mathbf{e}$ is a vector of dual variables in DP, and it is treated as the multipliers of each constraint in DP. Note that $\mathbf{e}$ is a unit vector, $\mathbf{\Pi}_k^{\mathrm{P}}$ and $\mathbf{S}_k$ are diagonal matrices whose diagonal entries are $\widetilde{\pi}_i^k$ and s_i^k, respectively. $(\pi^{\mathrm{P}})_i^k$ and s_i^k are the i^{th} row and the i^{th} column elements of the matrix $\widetilde{\mathbf{\Pi}}_k$ and $\mathbf{S}_k$, respectively. The penalty term of the objective function is updated by calculating $p^k = (\mathbf{\Pi}_k^{\mathrm{P}})^{-1}f^k$. Updating the search direction is done by the calculation of the intermediate matrix $\hat{\mathbf{D}}_k = \mathbf{\Pi}_k^{\mathrm{P}}\mathbf{S}_k^{-1}$.

To ensure that the subsequent iteration solution satisfies the constraints and moves closer to the optimal solution, the step sizes for both the primal and dual problems, denoted respectively by β_{P} and β_{D}, need to be updated in each iteration [13]. That is,

$$\beta_P = \frac{1}{\max\left\{1, -d_{\pi^P{}_i}{}^k / \alpha(\pi^P)^k\right\}} \tag{27}$$

and

$$\beta_D = \frac{1}{\max\left\{1, -d_{s_i}{}^k \Big/ \alpha s_i{}^k\right\}} \tag{28}$$

where α is set to be 0.99 [13] to maximize the step size and accelerate the convergence.

Once the directions of search and the step length are determined, the solution vectors can be updated to the $(k+1)^{\text{th}}$ iteration as follows:

$$\begin{cases} (\pi^{\mathrm{P}})^{k+1} \leftarrow (\pi^{\mathrm{P}})^k + \beta_P \mathbf{d}_{\pi^{\mathrm{P}}}{}^k \\ (\pi^{\mathrm{D}})^{k+1} \leftarrow (\pi^{\mathrm{D}})^k + \beta_D \mathbf{d}_{\pi^{\mathrm{D}}}{}^k \\ \mathbf{s}^{k+1} \leftarrow (\pi^{\mathrm{P}})^k + \beta_D \mathbf{d}_s{}^k \end{cases} \tag{29}$$

The search space parameter μ^k of the central path serves as an optimality measure for the solution at the current iteration. It is used to determine if the current solution on the central path has reached optimality. The value of μ^k is calculated as $\mu^k = \frac{((\pi^{\mathrm{P}})^k)^{\mathrm{T}}\mathbf{s}^k}{|\pi^{\mathrm{P}}|}$ and is compared against a predefined criterion, often referred to as condition 1, to assess the level of optimality for the current iteration solution $(\pi^k, \mathbf{s}^k)$ on the central path (see condition 1). The central path in the algorithm refers to a continuous path of the feasible solutions of the LPR problem as the weight of the penalty term increases from the optimal solution to the infinite point (*i.e.*, the solution when the constraint is infeasible).

By adjusting the step size, the algorithm can strike a balance between the accuracy of the solution (see condition 1) and the convergence speed, which is determined by the value of $d_{(\pi^{\mathrm{D}})}{}^k$ (see conditions 2 and 3) during the iterative process. τ^k is updated in the kth iteration by calculating $\tau^k = b - W(\pi^{\mathrm{P}})^k$. g^k represents the rate of change of the dual variable vector $(\pi^{\mathrm{D}})^k$ with respect to the objective function, and it can be updated as follows: g^k can be updated by having $\mathbf{g}^k = \mathbf{q} - \mathbf{W}^{\mathrm{T}}(\pi^D)^k - \mathbf{s}^k$. g^k could be used to check for optimality (see condition 1) and unboundedness (see condition 3).

Condition 1: If $\mu^k < \epsilon_1$, $\frac{\|\tau^k\|}{(\|b\|+1)} < \epsilon_2$, and $\frac{\|g^k\|}{(\|q\|+1)} < \epsilon_3$, then the algorithm should stop, and the optimal solution is obtained [13].

Condition 2: If $q^{\mathrm{T}}(\pi^{\mathrm{D}})^k < 0$, where q^{T} is defined in Eqn. (8), LPR in Eqn. (17) is unbounded [13].

Condition 3: If $d_{(\pi^{\mathrm{D}})}{}^k < 0$, DP in Eqn. (23-24) is unbounded [13].

If none of these conditions are met, the algorithm continues to its next iteration (back to step 1) by setting $k \leftarrow k + 1$.

After the mapping, each chiplet will be checked to see if its power consumption is within the thermal power capacity.

Essentially, if a chiplet R_i exceeds the power constraint, a task $t_j \in R_i$ is selected which has the maximum power consumption in R_i and swapped with another $t_k \in R_p, p \neq i$ and $R_p \in R^C$, ensuring minimal E value. This process involves the following sub-steps.

Step 2.1: Identify chiplets that exceed power constraint: $R^P = \{R_i | P_{\mathrm{c}}(R_i) > P_{\mathrm{th}}(R_i)\}$

Step 2.2: Iteratively swap high-power tasks with alternatives in other chiplets to adhere to power constraints.

1) Sort tasks in chiplet $R_i \in R^P$ according to their respective power consumption in descending order.
2) For each task t_j in R_i:
   Identify a target chiplet set R^C and select a task t_k to swap with t_j such that the value of E is minimized.

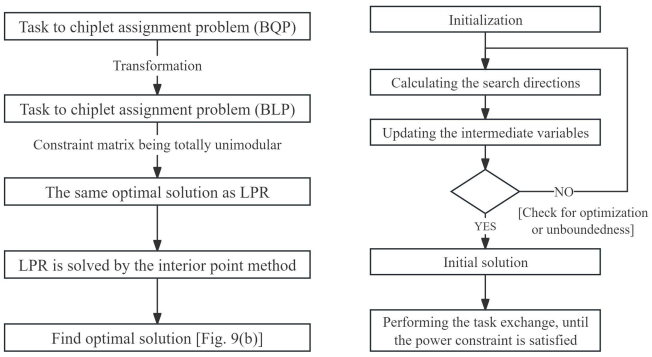


Fig. 9: (a) The process of task to chiplet assignment algorithm; (b) the process of finding optimal solution.

$$\begin{cases} R^C & = \{R_p \mid \arg\min\limits_{i,p\le|R|} \omega^R(R_i,R_p)\} \\ t_k & = \arg\min\limits_{P_{\rm c}(R_i)\le P_{\rm th}(R_i)} E = \alpha C_{\rm total}^{\rm Chiplet} + \beta \sum_i^{|R|} P_{\rm c}(R_i) \end{cases} \tag{30}$$

where α and β are the coefficients to balance the inter-chiplet communication latency and power consumption, and E is the objective function value that balances between the communication latency and power consumption.

3) Swap t_j and t_k between chiplet sets R_i and R^C. Iterate through these steps until all chiplets in R^P conform to the power constraint $P_{\rm c}(R_i) < P_{\rm th}(R_i)$.

Fig. 9 shows the process of the task to chiplet assignment algorithm and its details.

The complexity of the prime-dual interior point algorithm is $O(\|\pi^{\rm P}\|^3)$ [13].

Please refer to Section II of the supplementary material for the example of task to chiplet assignment.

VI. Intra-Chiplet Task Mapping

In the second step (Section IV-B), the goal is to map tasks to cores in a way that minimizes the overall communication latency, taking into account the communication characteristics of the tasks. There are two types of cores that are given special consideration in this step.

The first type is the cores located in the geometric center of the chiplet (referred to as center nodes). These cores have the maximum number of free neighbor cores, making them suitable for tasks that involve a high volume of communication. By mapping such tasks to the center nodes, expensive inter-chiplet communications can be avoided as much as possible.

The second type of cores is those close to the interface nodes, which are responsible for connecting to other chiplets. These cores are preferred for tasks that incur a high volume of intra-chiplet communications. By mapping these tasks to the interface nodes, the communication latency within the chiplet can be reduced.

Let z_i denote the ratio of intra-chiplet to inter-chiplet communication volumes for task t_i.

$$\begin{cases} z_i = \dfrac{\sum\limits_{\forall t_i\in R_j, t_m\in R_j}^{i\neq m} v(t_i,t_m)+1}{\sum\limits_{\forall t_i\in R_j, t_n\in R_k}^{i\neq n} v(t_i,t_n)+1} \\ j = r(t_i) = r(t_m); k = r(t_n) \end{cases} \tag{31}$$

where $r(t_i)$ is the chiplet mapped to run t_i. Note that in Eqn. (31), both the sigma terms in the numerator and denominator are added with 1. This mathematical treatment will help avert the potential division by zero problem resulting from the fact that a task to be mapped may actually experience zero inter-chiplet communication.

One can see that a high value of z_i indicates high intra-chiplet communication volume, while a lower z_i indicates that it has high inter-chiplet communication volume. In what follows, the tasks, t_i's are sorted by their respective z_i values in descending order, and the sorted tasks are kept in the task queue Z_c.

A. Intra-chiplet task mapping

Task mapping is performed iteratively. In each iteration, the head task in Z_c, *i.e.*, the one with the highest z_i value, is mapped to a core close to the geometric center of its chiplet. Then the tail task in Z_c, *i.e.*, the one with the smallest z_i value, is mapped to a core close to the interface node of its chiplet. Formally, in each iteration, two steps are performed.

Step 1: The head task t_i, which has the lowest z_i value among all the tasks in Z_c, gets removed from the queue. Assume t_i is assigned to chiplet R_j. There are two cases to consider.

1) If neither t_i nor its communicating tasks are mapped, t_i is mapped to a free core close to the geometric center of R_j. That is, a free core set is identified that satisfies,

$$\begin{cases} \Omega_1 = \{c_{mj} \mid \arg\min \omega^c(c_{mj}, c_{oj})\} \\ c_{mj} \in R_j, \text{ and } c_{mj} \text{ is free} \end{cases} \tag{32}$$

where c_{oj} is the geometric center of chiplet R_j, *i.e.*, $c_{oj}.x = \lfloor (R_j.X)/2 \rfloor$, $c_{oj}.y = \lfloor (R_j.Y)/2 \rfloor$, and $R_j.X$ and $R_j.Y$ are the X and Y dimensions of R_j, respectively. Ω_1 can be found by scanning the free cores in R_j, and those with $\min \omega^c(c_{mj}, c_{oj})$ are added into Ω, after which t_i is mapped to core c_{mj} such that

$$\begin{cases} M(t_i) = \arg\max \{D(c_{mj})\} \\ c_{mj} \in \Omega_1 \end{cases} \tag{33}$$

where $D(c_{kj})$ is the number of free neighbor cores of c_{kj}.

2) Assume there are l_i tasks, denoted by t_s, which communicate with t_i already mapped in R_j, t_i is mapped to a free core to minimize the partial intra-chiplet communication latency between t_i and its already mapped communicating tasks in $M(t_i)$. That is,

$$\begin{cases} M(t_i) = \arg\min \sum\limits_{s=1}^{s\le l_i} \sum\limits_{m=1}^{m\le|R_j|} v(t_i,t_s)\cdot\omega^c(c_{mj}, M(t_s)) \\ R_j = r(t_i) = r(t_s) \\ c_{mj} \in R_j \text{ and } c_{mj} \text{ is free}, M(t_s) \in R_j, t_s \in n(t_i) \end{cases} \tag{34}$$

where $n(t_i)$ is the set of communication tasks of t_i, and $r(t_i)$ and $r(t_s)$ are the chiplets running tasks of t_i and t_s, respectively.

Step 2: The tail task t_p, which has the lowest z_i value among all the tasks in Z_c, gets removed from the queue. Assume t_p is allocated to chiplet R_j. There are two cases to consider.

1) If neither t_p nor its intra-chiplet communicating tasks are mapped, t_p is mapped to a free core close to the interface nodes of R_j. That is, a free core set Ω_2 is identified that satisfies,

$$\begin{cases} \Omega_2 = \{c_{mj} | \arg\min \omega^c(c_{mj}, M(t_r))\} \\ R_j = r(t_p); k = r(t_r) \\ R_k \in N(t_p); t_r \in n(t_p) \\ c_{mj} \in R_j \text{ and is free}, M(t_r) \in R_k \text{ and is mapped.} \end{cases} \tag{35}$$

where $n(t_p)$ is t_p's communicating tasks set, $N(t_p)$ is the neighbor chiplets set of R_j. Ω_2 can be found by examining the free cores in R_j, and those with the minimum value are added into Ω_2. After that, t_p is mapped to core c_{mj} such that

$$\begin{cases} M(t_p) = \arg\min \sum_{r=1}^{r\le l_p} \sum_{m=1}^{m\le|R_j|} v(t_p,t_r)\cdot\omega^c(c_{mj},M(t_r)) \\ \quad R_j = r(t_p); R_k = r(t_r), j\neq k; R_k \in N(t_p); t_r \in n(t_p) \\ M(t_p) = c_{mj} \in \Omega_2, M(n_r(t_p)) \in R_k \end{cases} \tag{36}$$

2) Assume there are l_p tasks, denoted by t_u, which communicate with t_p and are already mapped in R_j, and l'_p tasks, denoted by t_v, which communicate with t_p and are already mapped in R_k. t_p is mapped to a free core to minimize the intra- and inter-chiplet communication latency. That is,

$$\begin{cases} M(t_p) = \arg\min \left(\sum_{u=1}^{u\le l_p} \sum_{m=1}^{m\le|R_j|} v(t_p,t_u)\cdot\omega^c(c_{mj},M(t_u)) \right. \\ \quad \left. + \sum_{v=1}^{v\le {l'}_p} \sum_{m=1}^{m\le|R_j|} v(t_p,t_v)\cdot\omega^c(c_{mj},M(t_v)) \right) \\ R_j = r(t_p) = r(t_u), R_k = N(t_p) = r(t_v); \\ M(t_v) \in N(t_p); t_u, t_v \in n(t_p) \\ c_{mj} = M(t_p) \in R_j \text{ and } c_{mj} \text{ is free}; \\ M(t_u) \in R_j, M(t_v) \in R_k \end{cases} \tag{37}$$

The iteration terminates once all the tasks in Z_c are mapped.

After the mapping, each core will be checked to see if its power consumption is within the thermal power capacity, which means,

$$\forall c_{ij} \in C, P_{\mathrm{c}}(c_{ij}) \le P_{\mathrm{th}}(c_{ij}) \tag{38}$$

If the power constraints are not met, the voltage and/or frequency level of the core is reduced to satisfy the thermal power capacity constraint.

The task mapping within a chiplet is mainly divided into two parts: priority determination and mapping. Assume the number of tasks to be mapped is $|T|$, the time complexity of the priority determination is $O(|T|\log|T|)$, and the total time complexity of the intra-chiplet task mapping is $O(|T|^2|C||R|)$. Putting things together, the overall time complexity of the entire algorithm is $O\left(\max(|T|^2|C|\cdot|R|, ||\pi^P||^3)\right)$.

Please refer to Section III.A of the supplementary material for the explanation of intra-chiplet task mapping.

VII. Experiment and Evaluation

A. Experimental Setup

The experiments were conducted on an event-driven C++ simulation [31] platform with CoMeT [32] integrated as the temperature simulator, McPAT [34] as the power simulator, and CoMeT as the thermal simulation [33] infrastructure for stacked chiplets. The 22nm CMOS technology node was employed in the simulation platform [34]. The simulator uses Sniper, and GPGPU-Sim processes to simulate individual chiplets, and they are connected by a network manager to simulate inter-chiplet networks and guarantee the correctness of the timing and function models by inter-simulator-process communication and synchronization. The simulation processes are synchronized for timing correctness [31].

The network topology used in this paper consists of tiled (Fig. 2(a)) and centralized multi-chiplet (Fig. 2(b)) systems. The tiled multi-chiplet system has 16 chiplets, where per-chiplet core counts are set to be 16, 64, 144, 256, 400, and 576. As shown in Fig. 2(a), the centralized multi-chiplet system has 16 chiplets, where each chiplet has 64, 144, or 256 cores.

Inside each chiplet, we assume a 0.25 mm^2 core area and a 10 mm [35] chiplet pitch. The thermal parameters of the interposer are shown in Table III. The area of the interposer is 2500 mm^2, and its thickness is 0.1 mm (by the 5th generation CoWoS-S platform [36]). The heat capacity of the interposer is 1.81 $\times$ 10^6 J/(m^3·K), and the thermal conductivity and the capacitance density are 35 W/(m^3·K) and 300 nF/mm^2 (cf. the thermal interface material used in CoWoS-S5 [36]), respectively.

The transmission energy consumption between adjacent chiplets is 1.17 PJ/bit [1], and the transmission delay between adjacent chiplets is composed of the following three parts: 1) the processing overhead of packetization and depacketization times can be obtained from [11]; 2) the transceivers' transmission delay is adopted from [9]; and 3) the interposer wire delay and power models are adopted from [12].

Table IV, and Table V summarize the configurations of real benchmarks. There are two types of real benchmarks. The first set of benchmarks are from PARSEC and SPLASH-2 on an x86 ISA multi-chiplet system simulator [31] to extract task graphs, with the configurations shown in Table IV. For PARSEC benchmarks, the inter-thread/task communication is computed by summing the input and output data sizes for each thread/task. The metric for inter-thread/task communication is the communication volume (in Byte). In the simulation, streaming assembly (SASS) is used for the GPU multi-chiplet simulator, while x86-64 is used for the CPU multi-chiplet simulator. The task graph is extended to include dummy nodes representing shared memory units, and thread-to-memory traffic is profiled with corresponding volumes as edge weights in the task graph. This approach enables a more explicit representation of communication patterns, especially in the context of shared memory programs like PARSEC and SPLASH-2.

The second set of benchmarks is neural network application benchmarks that include Yolov3 [38], ResNet-50 [39], Transformer [40], and GCN [41], which are run on a GPU multi-chiplet system simulator [31]. The real application benchmarks (Yolov3, ResNet-50, Transformer, and GCN) were parallelized using methods described in the following references: Benchmarks [43] for ResNet50, [44] for Yolov3, [45] for Transformer, and [37] for GCN.

Table VI presents the configurations of the servers connected by a Wi-Fi router within an intranet.

To evaluate the algorithm proposed in this paper, two previously proposed task mapping algorithms, DAR [14] and IOA [15], were selected as the baseline for comparison.

DAR [14] improves the system performance in a many-core system by dynamic allocation/reallocation of dark cores.

IOA [15] uses a multi-objective particle swarm optimization algorithm, which considers the power capacity and communication latency as objectives and finds the optimal solution by collaboration and information sharing among individuals in the population.

The multi-chiplet system is modeled as a graph, where inter- and intra-chiplet links have different edge weights. Tasks are mapped directly onto this system model using the DAR/IOA methods.

TABLE II: Event-driven C++ simulation platform configuration for inter- and intra-chiplet network

Inter- and intra- chiplet network configuration			
Flit size	128 bits		
Intra-chiplet communication latency	Router: 2 cycles; link: 1 cycle		
Inter-chiplet communication latency	Router: 2 cycles		
Routing Algorithm	XY routing for intra- and inter-chiplet connection networks		
Buffer depth	4 flits per route		
Network topology		Tiled	Centralized
	Intra-chiplet size	Mesh, 4×4/8×8/12×12/16×16/20×20/24×24	Mesh, 8×8/12×12/16×16
	Inter-chiplet size	Mesh, 4×4	Shown in Fig. 2(a)

TABLE III: CoMeT configurations of chiplet interposer

Chiplet	
Core area	0.25 mm^2
Chiplet pitch	10 mm
Interposer	
Area	2500 mm^2
Thickness	0.1 mm
Heat capacity	1.81×10^6 J/($m^3 \cdot$ K)
Thermal conductivity	35 W/(m $\cdot$ K)
Capacitance density	300 nF/mm^2
Inter-chiplet transmission energy consumption	1.17 PJ/bit

TABLE IV: The x86 ISA multi-chiplet system simulator configurations

Core architecture	x86-64
Main memory size	2GB
Get/Decode/Submit Size	2004/4/4
Baseline frequency	3GHz
ROB size	64
L1 data cache (private)	16 KB, 2-way, 32B line, 2 cycles, 2 ports
L1 instruction cache (private)	32 KB, 2-way, 64B line, 2 cycles, 2 ports
L2 cache (shared)	64KB slices/core, 64 line, 6 cycles, 2 ports
Benchmarks	
Barnes, Blackholes, Canneal, Dedup, Ferret, Fluidanimate, Freqmine	

TABLE V: Single instruction multi-threaded processor core (SIMT) configurations

Number of clusters	15
Processor cores/clusters	32
Warp size	32
Shared memory/processor cores	48KB
Number of registers	32768
L1 data cache	16KB, 4-way set assoc, 64B lines
L1 instruction cache	2KB, 4-way set assoc, 64B lines
Level 2 cache	786 KB
Memory	
Bandwidth/Memory Module	8 Bytes / cycle
Number of memory controllers	6
Memory Controller Type	FR-FCFS
Intra-chiplet network topology	Mesh
Inter-chiplet network topology	Mesh
Benchmarks	
ResNet50 [43], Yolov3 [44], Transformer [45], and GCN [37]	

TABLE VI: The configurations of the 4-server network.

The configuration of server network	
CPU	Intel(R) Core (TM) i7
Memory	16G
Wi-Fi router	ASUS RT-AX58U Dual Band WIFI Router

B. Validating the Error of the Proposed Thermal Power Capacity Model

In this set of experiments, the TPC model is validated. Each core is randomly assigned a power consumption value. The power consumption of adjacent cores is gradually increased until the temperature of an adjacent core reaches the temperature threshold P_{th}, and the maximum allowed power consumption of this core P_{c} is thus recorded. The regression error of the thermal power capacity model is defined as follows:

$$e = \left| \frac{P_{\text{c}} - P_{\text{th}}}{P_{\text{th}}} \right| \times 100\% \tag{39}$$

One can see from Fig. 10 that the average and maximum errors are 7.4% and 8.5%, respectively. In this case, this model is considered reasonably accurate for the subsequent experiments with results reported in the next subsections.

C. Peak Temperature Analysis

In the following experiments, the peak temperatures of the proposed method, DAR, and IOA are compared for different intra-chiplet network sizes. One can see from Fig. 10(a) that the mapped tasks generated by all three approaches are able to generate the mapping results that meet the thermal constraints of 80^oC.

D. Task Mapping Performance Evaluation on Tiled and Centralized Multi-chiplet Systems

Multiple chiplets can be organized following a tiled or centralized topology (Section III). In this subsection, different benchmarks are mapped to tiled multi-chiplet systems using the three methods. These three methods are thus compared in terms of application execution time and communication latency.

1) Real Application Benchmarks

In this experiment, the benchmarks are selected to include real-world applications, including Dedup, Ferret, Fluidanimate, Freqmine, Barnes, Blackholes, and Canneal from the PARSEC benchmark set.

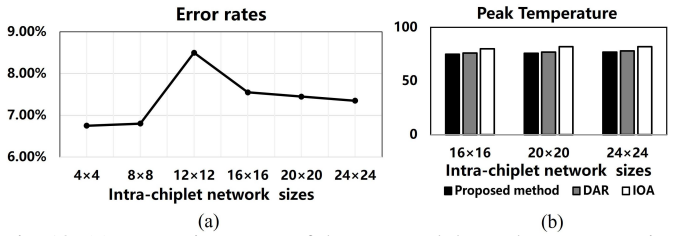
Please refer to Section III.B of the supplementary material for the experiments of real applications benchmarks with different numbers of tasks.

The preceding discussion pertains to characteristics of typical applications that are either computation-intensive or communication-intensive. In the following, the focus is on neural network (NN) applications that exhibit greater demands in terms of both computation and communication.

ResNet50 [43], Yolov3 [44], Transformer [45], and GCN [37] are neural network applications with distinct characteristics when mapped to a chiplet system. ResNet50's depth increases computational complexity and communication latency, while Yolov3's object detection pipeline demands high computation and memory resources. Transformer's attention mechanism requires efficient handling of pairwise interactions, and GCN's graph-based data processing poses challenges in communication and computation.

The performance is evaluated by the tasks graph of ResNet50 [43], Yolov3 [44], Transformer [45], and GCN [37], which is extracted by the GPU multi-chiplet system simulator [31]. Fig. 11(a) shows the application execution times of the three methods by running ResNet50 with different network sizes. The proposed method reduces the application execution time by 34.7% and 25.7% over DAR and IOA, respectively. From Fig. 11(b), it can be seen that the proposed method reduces the communication latency by 41.7% and 31.2% over DAR and IOA, respectively.

The IOA method achieves a maximum radius of mapping



(a) (b)

Fig. 10: (a) Regression errors of the proposed thermal power capacity models with respect to different intra-chiplet network sizes. (b) Peak temperatures with respect to different intra-chiplet network sizes.

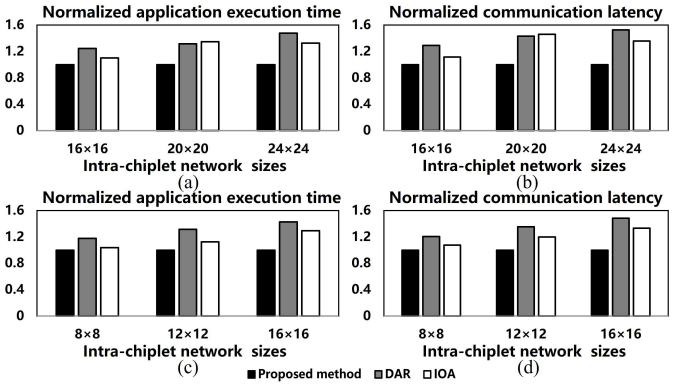


Fig. 11: (a) Application execution times and (b) communication latencies of the three methods when running ResNet50 with different intra-chiplet network sizes in the tiled multi-chiplet system. (c) Application execution times and (d) communication latencies of the three methods when running ResNet50 with different intra-chiplet network sizes in the centralized multi-chiplet system.

3 chiplets across different application scales, while the DAR achieves a maximum radius of mapping 5 chiplets. In contrast, the algorithm presented in this paper confines the maximum radius to 2 chiplets during the mapping process. Furthermore, the average distance between tasks mapped to chiplets in the IOA approach is measured at 2.41 chiplets, whereas the same metric in the DAR is 2.85 chiplets. Notably, the proposed algorithm in this study achieves an average distance of 1.23 chiplets between tasks mapped to chiplets. Once again, our algorithm outperforms the DAR and IOA methods in optimizing task mapping by reducing inter-chiplet communication latency.

In the context of mapping tasks to centralized multi-chiplet systems, the task graphs of ResNet50 [43], Yolov3 [44], Transformer [45], and GCN [37] are also applied to evaluate the performance of the proposed algorithm in mapping tasks in centralized multi-chiplet systems, and the application traces again are extracted from the GPU multi-chiplet system simulator [31]. The experimental results demonstrate that the DAR and IOA achieved a reduction of the proportion of inter-chiplet communication of 50.1% and 32.0%, respectively. Fig. 11(c) shows the application execution times of the three methods by running ResNet50 with different network sizes. It can be seen from Fig. 11(c) that the proposed method reduces the application execution time by 30.8% and 15.3% over DAR and IOA, respectively. Fig. 11(d) shows the communication latencies of the three methods by running ResNet50 with different network sizes, and it can be seen from Fig. 11(d) that the proposed method reduces the communication latency by 34.9% and 20.3% over DAR and IOA, respectively.

In the context of mapping tasks to tiled multi-chiplet systems, in addition to the case of ResNet50, consistent mapping results for all other NN applications, as depicted in Fig. 12.

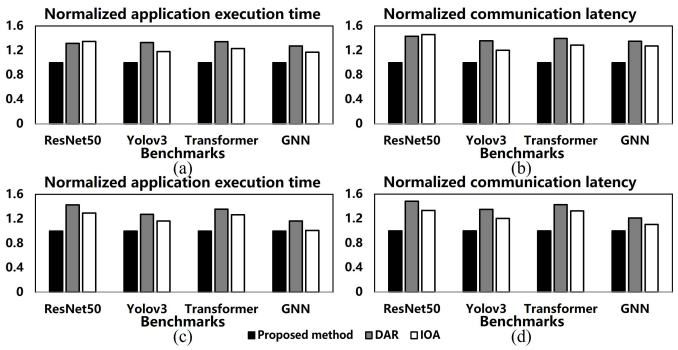


Fig. 12: (a) Application execution times and (b) communication latencies of the three methods when running different applications in the tiled multi-chiplet system. (c) Application execution times and (d) communication latencies of the three methods when running different

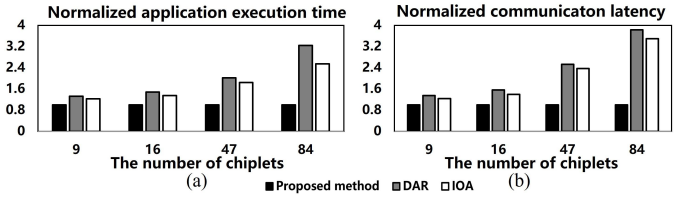


Fig. 13: (a) Application execution times and (b) communication latencies of the three methods with different numbers of chiplets.

On average, the proposed method reduces the application execution time by 31.5% and 19.5% over DAR and IOA, respectively. From Fig. 12(b), it can be seen that the proposed method reduces the communication latency by 36.9% and 25.5% over DAR and IOA, respectively.

The average distance between tasks mapped to chiplets in the IOA approach is measured at 2.36 chiplets, whereas the same metric in the DAR is 2.67 chiplets. Notably, the proposed algorithm in this study achieves an average distance of 1.17 chiplets between tasks mapped to chiplets.

Fig. 12(c) shows the application execution times of the three methods by running ResNet50 [43], Yolov3 [44], Transformer [45], and GCN [37] in a multi-chiplet system with an intra-chiplet network size of 16×16. The experimental results demonstrate that the DAR and IOA achieved a reduction of the proportion of inter-chiplet communication of 46.9% and 30.8%, respectively. The proposed method reduces the application execution time by 26.6% and 14.8% over DAR and IOA, respectively. From Fig. 12(d), it can be seen that the proposed method reduces the communication latency by 33.0% and 21.3% over DAR and IOA, respectively.

Fig. 13 demonstrates that as the number of chiplet increases, the advantages of our method in terms of execution time and communication latency become more pronounced when compared to IOA and DAR. This improvement is attributed to our algorithm's recognition of the clustering structure of the multi-chiplet system, which enables it to outperform the other two methods. However, it is important to note that for single small-scale SoC chips, the benefits of our algorithm may not be evident.

With the escalation in the number of chiplets, the proportion of inter-chiplet communication latency experiences a concurrent increase. Consequently, the proposed algorithm demonstrates a reduction in ΔC_{total} (defined in Section III.A) compared to the DAR and IOA algorithms. Experimental results show that the correlation coefficient between the reduction in total communication latency ΔC_{total} and the reduction

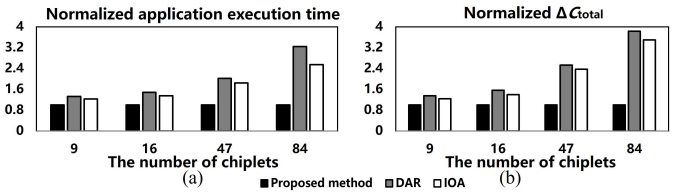


Fig. 14: (a) Application execution times and (b) ΔC_{total} of the three methods with different numbers of chiplets.

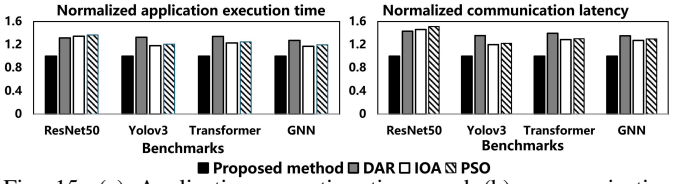


Fig. 15: (a) Application execution times and (b) communication latencies of the four methods when running different applications in the tiled multi-chiplet system.

in execution time achieved by the proposed algorithm is over 0.95. From Fig. 14, as our proposed approach can significantly reduce ΔC_{total} compared to the other two approaches, our proposed approach can also reduce the application execution time.

Fig. 15(a) shows the application execution times of the four methods—DAR, IOA, PSO [42], and the proposed approach—when running ResNet50 [43], Yolov3 [44], Transformer [45], and GCN [37] in a multi-chiplet system. PSO [42] employs a fault-tolerant mapping algorithm that utilizes spare cores within a Mesh-of-Tree (MoT) network to enhance system reliability by addressing core failures. The experimental results demonstrate that the proposed method reduces application execution time by 31.5%, 19.5%, and 21.8% compared to DAR, IOA, and PSO [42], respectively. Moreover, Fig. 15(b) highlights the reduction in communication latency by 36.9%, 25.5%, and 27.4% over DAR, IOA, and PSO, respectively. These results provide quantitative evidence of the effectiveness of the proposed method in improving both application execution time and communication latency in the multi-chiplet system.

The IOA integrates the crowding distance mechanism, which acts as a metric to evaluate communication density. This mechanism enables IOA to maintain solution diversity and effectively avoid convergence to local optima, enabling a more comprehensive exploration of the solution space. This enhanced search capability enables IOA to outperform PSO in optimizing communication latency, particularly in multi-chiplet systems, where PSO's performance is limited by its hierarchical communication model.

Fig. 16(a) shows the application execution times of the three methods by running a mix of 10 applications which is composed by Transformer, ResNet-50, and GNN with different arrival rates in the tiled multi-chiplet system. The experimental results demonstrate that the proposed method reduces the total application execution time by 61.3 % and 55.2% over DAR and IOA, respectively. From Fig. 16(b), the proposed method reduces the communication latency by 65.7% and 61.1% over DAR and IOA, respectively.

Fig. 16(c) shows the application execution times of the three methods by running a mix of 10 applications which is composed by Transformer, ResNet-50, and GNN with different arrival rates in the tiled multi-chiplet system. The experimental

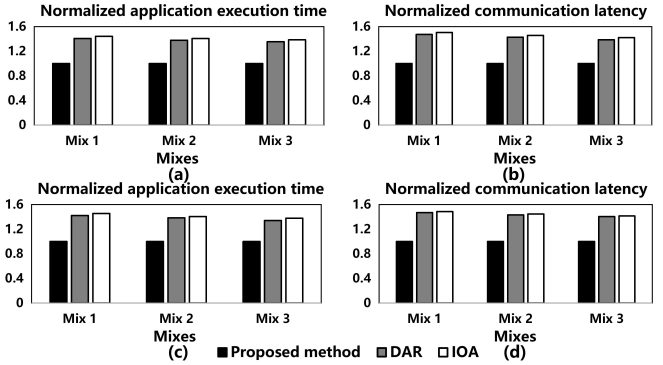


Fig. 16: (a) Application execution times and (b) communication latencies of the three methods when running different mixes with different arrival rates in a in the tiled multi-chiplet system. (c) Application execution times and (d) communication latencies of the three methods when running different mixes with different arrival rates in a in the centralized multi-chiplet system.

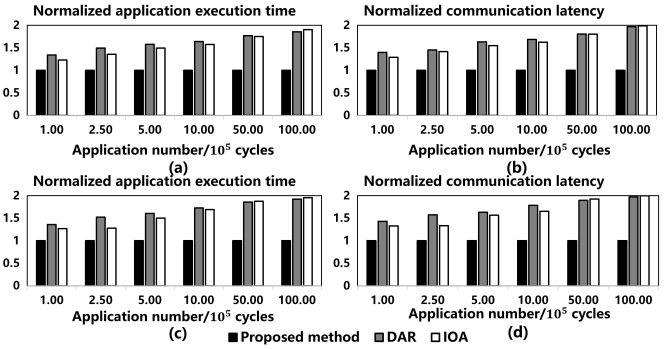


Fig. 17: (a) Application execution times and (b) communication latencies of the three methods when running different mixes in the tiled multi-chiplet system. (a) Application execution times and (b) communication latencies of the three methods when running different mixes in the centralized multi-chiplet system.

results demonstrate that the proposed method reduces the total application execution time by 66.5% and 59.4% over DAR and IOA, respectively. From Fig. 16(d), the proposed method reduces the communication latency by 71.5% and 63.3% over DAR and IOA, respectively.

Fig. 17(a) shows the total application execution times of the three methods by running different mixes of application in centralized multi-chiplet system. Mix 1 consists of a mix of 10 Transformers. Mix 2 comprises a mix involving three Transformers, four ResNet50s, and three GNNs. Mix 3 consists of a mix containing five Transformers and five GNNs. The experimental results demonstrate that the proposed method reduces the total application execution time by 37.8% and 41.0% over DAR and IOA, respectively. From Fig. 17(b), the proposed method reduces the communication latency by 43.5% and 44.9% over DAR and IOA, respectively.

Fig. 17(c) shows the total application execution times of the three methods by running different mixes of application with the arrival rate being in the centralized multi-chiplet system. Mix 1 consists of a mix of 10 Transformers. Mix 2 comprises a mix involving three Transformers, four ResNet50s, and three GNNs. Mix 3 consists of a mix containing five Transformers and five GNNs. The experimental results demonstrate that the proposed method reduces the total application execution time by 38.3% and 41.0% over DAR and IOA, respectively. From Fig. 17(d), the proposed method reduces the communication latency by 42.9% and 46.1% over DAR and IOA, respectively.

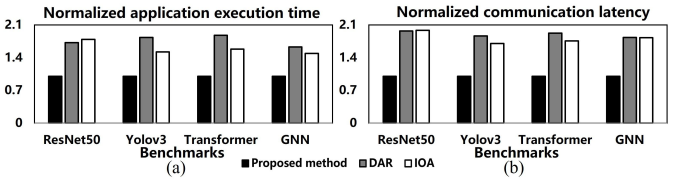


Fig. 18: (a) Application execution times and (b) communication latencies of the three methods when running different applications in the 4-server system.

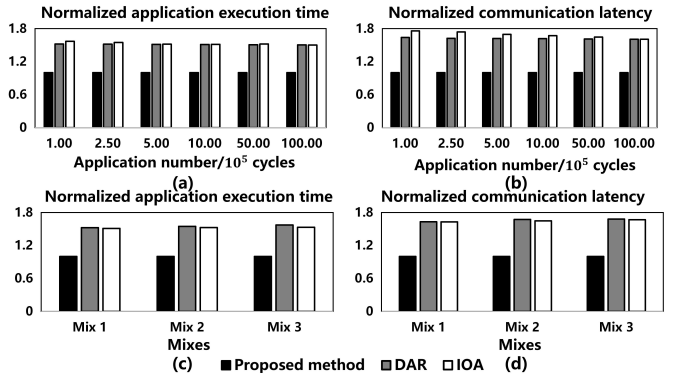


Fig. 19: (a) Total application execution times and (b) communication latencies of the three methods when running mix 1 with different arrival rates in the 4-server system. (c) Total application execution times and (d) communication latencies of the three methods when running different application mixes in the 4-server system.

Fig. 18(a) shows the application execution times of the three methods by running ResNet50 [43], Yolov3 [44], Transformer [45], and GCN [37] in the 4-server system. The experimental results demonstrate that the proposed method reduces the application execution time by 77.9% and 53.3% over DAR and IOA, respectively. From Fig. 18(b), it can be seen that the proposed method reduces the communication latency by 87.4% and 76.3% over DAR and IOA, respectively. The rationale behind this lies in our algorithm's recognition and preservation of the clustering structure within the multi-server system. Due to the significantly higher latency in inter-sever communication compared to intra-sever communication, our approach outperforms the other two methods.

Fig. 19(a) shows the application execution times of the three methods (the proposed method, DAR, and IOA) by running a mix of 10 applications consisting of Transformer, ResNet-50, and GNN with different arrival rates on the 4-server system. The experimental results show that the proposed method reduces the total application execution time by 51.2% and 52.8% compared to DAR and IOA, respectively. In Fig. 19(b), the proposed method reduces communication latency by 62.0% and 68.6% over DAR and IOA, respectively.

Fig. 19(c) shows the total application execution times of the three methods when running different mixes of applications on the 4-server system. The experimental results demonstrate that the proposed method reduces the total application execution time by 54.7% and 52.2% over DAR and IOA, respectively. As shown in Fig. 19(d), the proposed method reduces communication latency by 66.1% and 64.9% over DAR and IOA, respectively.

The proposed task mapping algorithm has worked well in both tiled and centralized chiplet systems. For tasks with high inter-task communication, the proposed algorithm tends to favor central core mapping to optimize communication latency. Conversely, for tasks with limited inter-task communication, the algorithm distributes tasks more evenly across cores to maximize resource utilization.

E. Execution time Overhead of the Proposed method

By running the proposed method 100 times in a system with different parameters, the average application execution time of the real applications from PARSEC is found to be around 3 billion cycles. The average time of the priority determination is approximately 1.6% of the mapping time. The average execution time of the proposed method is about 76 million cycles, accounting for only 2.53% of the application execution time. In a simple term, the execution time overhead of the proposed method is quite small, accounting for only 2.53% of the application execution time.

VIII. Conclusion

In this paper, a task mapping algorithm was proposed for the multi-chiplet many-core system to minimize communication latency under the power capacity. Essentially, the algorithm assigns tasks to chiplets to minimize inter-chiplet communication, followed by mapping tasks with high inter-chiplet communication to cores near the interface node and tasks with high intra-chiplet communication latency to the center of each chiplet. Experimental results demonstrate the effectiveness of the proposed method, showing significant reductions in application execution time compared to two existing methods, DAR and IOA. The proposed algorithm achieves a remarkable decrease in application execution time, by as much as 37.5% and 24.7%, compared to DAR and IOA, respectively, confirming the suitability of the proposed mapping algorithm for future multi-chiplet many-core systems. It's important to note that these outcomes remain consistent regardless of topology and size, underscoring the independence of these factors.

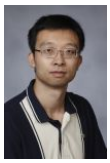
References

[1] Z. Tan, H. Cai, R. Dong, and K. Ma, "NN-Baton: DNN Workload Orchestration and Chiplet Granularity Exploration for Multichip Accelerators," in ACM/IEEE ISCA, pp. 1013-1026, 2021.
[2] Y. S. Shao *et al.*, "Simba: Scaling Deep-Learning Inference with Chiplet-Based Architecture," Commun. ACM, vol. 64, no. 6, pp. 107–116, Jun. 2021.
[3] J. Zhang *et al.*, "INDM: Chiplet-Based Interconnect Network and Dataflow Mapping for DNN Accelerators," in IEEE TCAD, 2023.
[4] J. Cai *et al.*, "Gemini: Mapping and Architecture Co-exploration for Large-scale DNN Chiplet Accelerators," IEEE Int'l Symp. on HPCA, pp. 156-171, 2024.
[5] T. Chou *et al.*, "NetFlex: A 22nm Multi-Chiplet Perception Accelerator in High-Density Fan-Out Wafer-Level Packaging," IEEE Symp. on VLSI, pp. 208-209, 2022.
[6] A. Fuchs and D. Wentzlaff, "The Accelerator Wall: Limits of Chip Specialization," in HPCA, pp. 1-14, 2019.
[7] Y. Li, A. Louri, A.Karanth. "Scaling deep-learning inference with chiplet-based architecture and photonic interconnects". In ACM/IEEE DAC pp. 931-936, 2021.
[8] M. Fattah, M. Ramirez, M. Daneshtalab, P. Liljeberg, and J. Plosila, "CoNA: Dynamic Application Mapping for Congestion Reduction in Many-core Systems," in Int'l Conf. Computer Design, pp. 364-370, 2012.
[9] D. Das Sharma, G. Pasdast, Z. Qian and K. Aygun, "Universal Chiplet Interconnect Express (UCIe): An Open Industry Standard for Innovations With Chiplets at Package Level," in IEEE Trans. Compon. Packaging Manuf. Technol., vol. 12, no. 9, pp. 1423-1431, 2022.
[10] J. Kim *et al.*, "Architecture, Chip, and Package Codesign Flow for Interposer-Based 2.5-D Chiplet Integration Enabling Heterogeneous IP Reuse," in IEEE TVLSI, vol. 28, no. 11, pp. 2424-2437, 2020.
[11] Traber A, Zaruba F, Stucki S, *et al*. PULPino: A Small Single-core RISC-V SoC, in RISCV Workshop, 2016.
[12] M. A. Kabir and Y. Peng, "Chiplet-Package Co-Design For 2.5D Systems Using Standard ASIC CAD Tools," in ASP-DAC, pp. 351-356, 2020.

[13] R. H. M. K. J. L. R. Weismantel, Nonlinear Integer Programming (50 Years of Integer Programming 1958-2008). Springer Berlin Heidelberg, 2009.
[14] X. Huang, X. Wang, Y. Jiang, A. K. Singh, and M. Yang, "Dynamic Allocation/Reallocation of Dark Cores in Many-Core Systems for Improved System Performance," IEEE Access, vol. 8, pp. 165693-165707, 2020.
[15] M. Darbandi, A. R. Ramtin, and O. K. Sharafi, "Tasks Mapping in the Network on a chip using an improved optimization algorithm," in IJPCC, vol. 16, no. 2, pp. 165-182, 2020.
[16] C. Wang, Q. Xu, C. Nie, H. Cao et al., "An efficient thermal model of chiplet heterogeneous integration system for steady-state temperature prediction", in Microelectronics Reliability, vol. 146, pp. 115006, 2023.
[17] K. Manna, P. Mukherjee, S. Chattopadhyay, and I. Sengupta, "Thermal-Aware Application Mapping Strategy for Network-on-Chip Based System Design," IEEE TC, vol. 67, no. 4, pp. 528-542, 2018.
[18] G. Kothari and K. Ghose, "Thermally Aware multi-core chiplet stacking," in IEEE/ACM ICCAD, pp. 1-9, 2023.
[19] Y. Shen, L. Schreuders, A. Pathania, and A. D. Pimentel, "Thermal Management for 3D-Stacked Systems via Unified Core-Memory Power Regulation," in ACM TECS, vol. 22, no. 5s, pp. 1-26, 2023.
[20] C. Wu *et al.*, "An Efficient Application Mapping Approach for the Co-Optimization of Reliability, Energy, and Performance in Reconfigurable NoC Architectures," in IEEE TCAD, vol. 34, no. 8, pp. 1264-1277, 2015.
[21] L. Liu *et al.*, "A Flexible Energy- and Reliability-Aware Application Mapping for NoC-Based Reconfigurable Architectures," in IEEE VLSI, vol. 23, no. 11, pp. 2566-2580, 2015.
[22] A. L. d. M. Martins, A. H. L. da Silva, A. M. Rahmani, N. Dutt, and F. G. Moraes, "Hierarchical adaptive multi-objective resource management for many-core systems," in J. Syst. Archit., vol. 97, pp. 416-427, 2019.
[23] A. L. M. Martins, M. Ruaro and F. G. Moraes, "Hierarchical energy monitoring for many-core systems," in IEEE ICECS, pp. 657-660, 2015.
[24] K. -C. Chen, P. Gratz, J. Kim, V. Narayanan, and U. Ogras, "Panel: The Future of NoCs: Challenges and Opportunities," in IEEE/ACM NoCArc, pp. 1-10, 2022.
[25] C. Chen, J. Yin, Y. Peng, M. Palesi, W. Cao, L. Huang, A. K. Singh, H. Zhi, X. Wang, "Design Challenges of Intrachiplet and Interchiplet Interconnection," in IEEE Des Test, vol. 39, no. 6, pp. 99-109, 2022.
[26] J. Yin *et al.*, "Modular Routing Design for Chiplet-Based Systems," in ISCA, pp. 726-738, 2018.
[27] A. Fabijanska, "Normalized Cuts and Watersheds for Image segmentation," in IET Conf. Image Processing, pp. 1-6, 2012.
[28] S. Lang, Introduction to Linear Algebra. Springer Science and Business Media, 2012.
[29] D. Fulkerson and O. Gross, "Incidence Matrices and interval graphs," Pacific journal of mathematics, vol. 15, no. 3, pp. 835-855, 1965.
[30] S. P. Boyd and L. Vandenberghe, "Convex Optimization," in IEEE TAC, vol. 51, pp. 1859-1859, 2004.
[31] H. Zhi, X. Xu, W. Han *et al.*, "A Methodology for Simulating Multi-chiplet Systems Using Open-source Simulators," in NANOCOM, pp. 1-6, 2021.
[32] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, "Temperature-aware Microarchitecture: Modeling and implementation," in ACM TACO, vol. 1, no. 1, pp. 94–125, 2004.
[33] L. Siddhu, *et al.*, "CoMeT: An integrated interval thermal simulation toolchain for 2D, 2.5D, and 3D processor-memory systems," in ACM TACO, vol. 19, no. 3, pp. 1-25, 2022.
[34] J. Zhai, C. Bai, B. Zhu, Y. Cai, Q. Zhou, and B. Yu. "McPAT-Calib: A Microarchitecture Power Modeling framework for modern CPUs," in IEEE/ACM ICCAD, pp. 1-9, 2021.
[35] T. Burd *et al.*, ""Zeppelin": An SoC for Multichip Architectures," in IEEE JSSC, vol. 54, no. 1, pp. 133-143, 2019.
[36] P. K. Huang *et al.*, "Wafer Level System Integration of the Fifth Generation CoWoS®-S with High Performance Si Interposer at 2500 mm^2," in IEEE ECTC, pp. 101-104, 2021.
[37] B. Jiang, Z. Zhang, D. Lin, J. Tang, and B. Luo, "Semi-Supervised Learning With Graph Learning-Convolutional Networks," in CVPR, pp. 11305-11312, 2019.
[38] Y. Zhang, B. Du, L. Zhang, and J. Wu. "Parallel DNN Inference Framework Leveraging a Compact RISC-V ISA-based Multi-core System," in KDD, pp. 627-635, 2020.
[39] L. Liao, K. Li, K. Li, C. Yang, Q. Tian *et al.*, "UHCL-Darknet: An OpenCL-based Deep Neural Network Framework for Heterogeneous Multi-/Many-core Clusters," in ICPP, pp. 1-10, 2018.
[40] J. Jiang, J. Du, D. Huang, D. Li, J. Zheng, Y. Lu. "Characterizing and Optimizing Transformer Inference on ARM Many-core Processor," in CPP, pp. 1–11, 2023.
[41] Geng, T., Li, A., Shi, R., Wu, C., Wang, T., Li, Y., Herbordt, M. C *et al.*, "AWB-GCN: A Graph Convolutional Network Accelerator with Runtime Workload Rebalancing," in MICRO, pp. 922-936, 2020.
[42] P. Veda Bhanu and Soumya J, "Fault-Tolerant Application Mapping on Mesh-of-Tree based Network-on-Chip," in J. Syst. Archit., vol. 116, 2021, pp. 102026.
[43] L.Yan, and S. Yu. "H3D-Transformer: A Heterogeneous 3D (H3D) Computing Platform for Transformer Model Acceleration on Edge Devices." ACM TDAES, vol.29.3: 1-19, 2024..
[44] L. Liao, K. Li, K. Li, C. Yang, Q. Tian et al., "UHCL-Darknet: An OpenCL-based Deep Neural Network Framework for Heterogeneous Multi-/Many-core Clusters," in ICPP, pp. 1-10, 2018.
[45] Geng, T., Li, A., Shi, R., Wu, C., Wang, T., Li, Y., Herbordt, M. C et al., "AWB-GCN: A Graph Convolutional Network Accelerator with Runtime Workload Rebalancing," in MICRO, pp. 922-936, 2020.

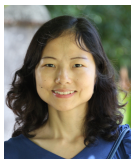
Xiaohang Wang received the B. Eng. and Ph. D. degree in communication and electronic engineering from Zhejiang University, in 2006 and 2011, respectively. He is currently a Professor at Zhejiang University. He was the recipient of PDP 2015 and VLSI-SoC 2014 Best Paper Awards.

Yifan Wang received the bachelor's degree in software engineering from South China University of Technology, Guangzhou, China, in 2023. His research interests include task mapping in multi-chiplet based many-core systems.



Yingtao Jiang received the Ph. D. degree in computer science from the University of Texas at Dallas, Richardson, TX, USA, in 2001. He is currently an Associate Dean of the College of Engineering, University of Nevada. His research interests include algorithms, computer architectures, VLSI, networking, nanotechnologies.

Amit Kumar Singh is an Associate Professor at University of Essex, UK. He received the B. Tech. degree from IIT, Dhanbad, India, in 2006, and the Ph. D. degree from Nanyang Technological University (NTU), Singapore, in 2013. His current research interests are design and optimisation of multi-core based computing systems with focus on performance, energy, and temperature.



Mei Yang received her Ph. D. in Computer Science from the University of Texas at Dallas in Aug. 2003. Her research interests include computer architectures and interconnection networks.