On Design Space Exploration of Cache System in Multi-Chiplet Systems

***Abstract*— While multi-chiplet based many-core systems have emerged as a viable solution for heterogeneous integration and addressing manufacturing and technological challenges in the post-Moore's Law era, their design and optimization remain highly complex and challenging. Among the various subsystems, the cache hierarchy has significant implications for overall system performance, yet its vast design space presents substantial optimization challenges. This complexity arises from factors such as the large number of chiplets in the system, the number of cores per chiplet, memory hierarchy variations, cache size variability, caching strategies, and inter-chiplet interconnection networks. Existing design space exploration methods, such as NN-Baton and IntLP, fail to optimize the cache subsystem performance or thoroughly explore the design space. To address these limitations, we propose a novel design space exploration method for cache subsystem optimization. Our approach models cache miss rates and network latency as functions of cache hierarchy and inter-/intra-chiplet interconnection network parameters. We then define an optimization problem to minimize the concurrent average memory access time (C-AMAT) under cost and power consumption constraints. This problem is addressed using a bilevel optimization algorithm, whichiteratively solves two independent subproblems: (1) cache subsystem optimization, and (2) inter-chiplet interconnection network optimization. Experimental results show that our method reduces the application execution time by 39.7% and 39.2%, on average, compared to architectures similar to AMD Zen 4 and Intel Sapphire Rapids, respectively, and by 25.91% over IntLP. These results underscore the potential of the proposed method for optimizing cache subsystems in future multi-chiplet based many-core systems.**

***Index Terms*—Many-core Systems, Multi-chiplet System, Design Space Exploration.**

I. Introduction

Multi-chiplet based many-core systems have been a new design paradigm to overcome the area wall to provide high computation power for various applications. In multi-chiplet systems, multiple chiplets or dielets can be integrated on interposers or substrates, and chiplets communicate by die-to-die (D2D) interfaces. A key to multi-chiplet based many-core systems is the cache hierarchy design. The difference in the cache hierarchy between a multi-chiplet based many-core system and conventional single chip (SoC) many-core systems are two folds: (1) The chiplets are "clusters" where the inter-chiplet communication is much more inefficient than that of the intra-chiplet communication. For example, the transmission latency between two adjacent chiplets is an order of magnitude longer than those of two adjacent tiles inside one chiplet. Due to the much higher latency and much lower bandwidth in inter-chiplet communication, the cache access traffic should be localized as much as possible to avoid high inter-chiplet communication latency [1]. (2) The problem of designing an optimal cache hierarchy has numerous parameters and extremely high complexity, especially in a multi-chiplet system.

Fig. 1 shows inter-chiplet traffic volumes for radiosity and radix applications on AMD Zen 4 [2] and Intel Sapphire Rapids (SPR) [3] systems. Zen 4 and SPR have different cache settings, e.g., whether the LLC in each chiplet is shared or not, and whether there is a centralized IO die for the memory controller or not. According to Fig. 1, for radiosity and radix, the inter-chiplet traffic volume of SPR is 1.042x and 0.949x of Zen 4 respectively, and correspondingly the performance of SPR is 0.946x and 1.048x of Zen 4 respectively. From this example, two observations can be made: (1) Higher inter-chiplet traffic volume leads to lower performance. (2) The configuration of cache and memory subsystem impacts the inter-chiplet traffic volume and thus performance. Therefore, it needs to explore the cache subsystem design space.

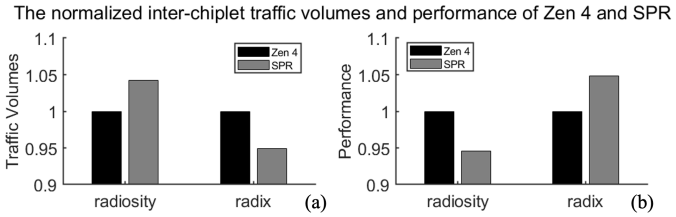

Fig. 1. Comparison of (a) inter-chiplet traffic volumes and (b) performances of AMD Zen 4 and Inter Sapphire Rapids.

However, the design space of cache hierarchy is extremely huge, leading to an optimization problem with high complexity. For example, the design variables include chiplet number, the number of cores in each chiplet, memory hierarchy in each chiplet, cache size, different inter-chiplet interconnection networks, etc. Existing design space exploration methods like IntLP [4], NN-Baton [5], and Monad [6] cannot solve the above challenge since they cannot explore such a huge space.

Therefore, in this paper, we propose an efficient optimization algorithm for design space exploration of cache subsystems in multi-chiplet systems. In essence, (1) An analytical cache performance model based on C-AMAT [7] is proposed for multi-chiplet systems. (2) An optimization problem is formulated to select the parameters to optimize cache performance under the cost and power budgets. (3) A bilevel algorithm to efficiently solve the problem is proposed, where the optimization problem is decomposed into two subproblems: (a) cache subsystem optimization, and (b) inter-chiplet interconnection network optimization, and those two subproblems are optimized iteratively.

The remainder of the paper is organized as follows. Section 2 reviews the related work. Section 3 analyzes and models multi-chiplet systems, and defines the cache performance optimization problem. Section 4 presents a bilevel optimization algorithm to solve the optimization problem. Section 5 provides experimental results. Finally, Section 6 concludes the paper.

II. Related Work

Concurrent average memory access time (C-AMAT) [7] refines the traditional average memory access time (AMAT) model by accounting for memory concurrency and categorizes memory cycles to better model memory access performance. Layered performance matching (LPM) enables hierarchical

TABLE I
NOMENCLATURE

Symbol	Definition
v_{ijk}	The i-th cache in level j in chiplet k.
$e_{ijk,mno}$	The link connecting v_{ijk} and v_{mno}.
$w(v_{ijk})$	The weights of node v_{ijk}.
$n_{p,q}$	The p-th node in chiplet q.
$c(n_{p,q}, n_{r,s})$	The link between adjacent nodes $n_{p,q}$ and $n_{r,s}$.
$\omega(c(n_{p,q}, n_{r,s}))$	The transmission latency between $n_{p,q}$ and $n_{r,s}$.
$\lvert R_i \rvert$	The number of cores in chiplet i.
n_α	The chiplet number.
h_i	The total number of memory hierarchy in chiplet i.
m_j^i	The cache size for level j in chiplet i.

memory performance analysis, transforming overall optimization into a set of local optimizations for each cache level. In [8], researchers developed models to enhance cache miss rate estimation and cache partitioning for multi-core systems, addressing data-sharing, coherence, and runtime monitoring for efficient cache use. In [5], [6], [9], several methodologies, including NN-Baton, SASMM, and Monad, were proposed to optimize chiplet-based system design, focusing on functional partitioning, resource allocation, and performance optimization. IntLP [4] proposed an integer linear programming model for the design of multi-chiplet systems, which determines the optimal partitioning of functional components and chiplet selection. Existing methods struggle to optimize multiple cache parameters or explore sufficiently large design spaces, and most focus on narrow objectives. This paper proposes a comprehensive optimization to overcome these challenges.

III. Performance Modeling and Problem Definition

A. Systems Modeling

As shown in Fig. 2, the multi-chiplet based many-core system is modeled by two graphs: memory hierarchy graph and inter-chiplet interconnection network topology graph, characterizing its cache and interconnection subsystems.

Definition 1. Memory hierarchy graph $G_M(V, E)$. $G_M(V, E)$ is an undirected graph, where each node $v_{ijk} \in V$ is either a core, a cache unit, or a DRAM unit. v_{ijk} is the i-th cache in level j in chiplet k. The edge $e_{ijk,mno} \in E$ indicates the link connecting v_{ijk} and v_{mno} (between two levels of cache units, or the reachability of core). If v_{ijk} is connected to v_{mno}, then $e_{ijk,mno} = 1$, indicating v_{ijk} is related by v_{mno} (for example, inclusiveness), and 0 otherwise. As shown in Fig. 2 (a), each core with an L1 cache, an L2 cache, and the two cores sharing an L3 cache are shown as the memory hierarchy graph. There is an edge between v_{131} and v_{121}, indicating the inclusiveness of v_{131} over v_{121}. Otherwise, there is no connected edge between v_{131} and v_{321}, indicating that v_{131} is not inclusive over v_{321}.

Define h_k as the total number of cache hierarchy in chiplet k. The weights $w(v_{ijk})$ of node $v_{ijk} \in V$ are as follows. The node v_{ijk} with $j = 0$ is a core, where $cpi(v_{ijk})$ is the ideal CPI (assuming no cache miss), which is the average number of clock cycles per instruction. The node v_{ijk} with $0 < j < h_k$ is a cache unit, where $cs(v_{ijk})$ is cache size, $cl(v_{ijk})$ is cache line size, $as(v_{ijk})$ is associativity, $rs(v_{ijk})$ is replacement strategy, and $ac(v_{ijk})$ is the cache access rate.

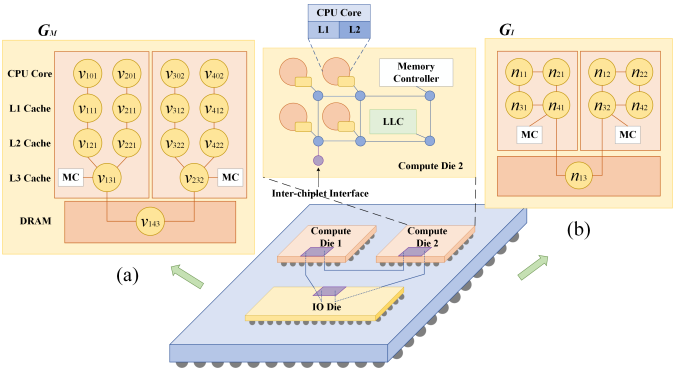

Fig. 2. The multi-chiplet based many-core system graph, (a) the $G_M(V, E)$ graph example and (b) the $G_I(N, C)$ graph example.

The node v_{ijk} with $j = h_k$ is a DRAM unit, where $bw(v_{ijk})$ is DRAM bandwidth.

$$w(v_{ijk}) = \begin{cases} \langle cpi(v_{ijk}) \rangle, & j = 0 \\ \langle cs(v_{ijk}), cl(v_{ijk}), as(v_{ijk}), & \\ rs(v_{ijk}), ac(v_{ijk}) \rangle, & 0 < j < h_k \\ \langle bw(v_{ijk}) \rangle, & j = h_k \end{cases} \tag{1}$$

Definition 2. Inter-chiplet interconnection network topology graph $G_I(N, C)$. $G_I(N, C)$ is also defined as an undirected graph where each node $n_{p,q} \in N$ is a node in chiplet q on the inter-chiplet network, which is composed of a core, an L1/L2 bank, a network interface, and a router, or is a D2D interface only. C is the set of network links, $c(n_{p,q}, n_{r,s}) \in C$ is the link between adjacent nodes $n_{p,q}$ and $n_{r,s}$. The chiplets in G_I are denoted as $R_1, \ldots, R_{n_\alpha}$, where each $R_i \subset N$, $i = 1, \ldots, n_\alpha$. n_α is the total number of chiplets. The number of cores in chiplet i is denoted by $|R_i|$.

Define $\omega(c(n_{p,q}, n_{r,s}))$ as the transmission latency between $n_{p,q}$ and $n_{r,s}$. If $q = s$, nodes $n_{p,q}$ and $n_{r,s}$ are in the same chiplet; otherwise, they are in different chiplets. Fig. 2 (b) shows an example of a three-chiplet system connected by an inter-chiplet network and its corresponding interconnection graph.

B. Miss Rate Modeling

In this paper, we model the miss rate with G_M where the parameters relate to cache miss rate of different cache levels.

Given G_M, the miss rate of node v_{ijk} $(0 < j \leq h_k - 1)$ is $M(v_{ijk})$. Define $\Pi(v_{ijk}) = \{\pi_1(v_{ijk}), \ldots, \pi_n(v_{ijk})\}$ as the set of nodes on entire reachable path from any core to v_{ijk} (excluding the cores), where $\pi_x(v_{ijk})$ is the x-th node on the reachable path and n is the total number of nodes. All upper-level nodes in cache hierarchy with connection relationships in v_{ijk} are included in $\Pi(v_{ijk})$.

According to [10], the cache miss rate $M(v_{ijk})$ of v_{ijk} is modeled as:

$$\Lambda(v_{ijk}) = \sum_{x=1_n}^{n} a_x \cdot cs\big(\pi_x(v_{ijk})\big)^{\alpha_x} + \sum_{x=1}^{n} b_x \cdot cl\big(\pi_x(v_{ijk})\big)^{\beta_x} + \sum_{x=1}^{n} c_x \cdot as\big(\pi_x(v_{ijk})\big)^{\gamma_x} \tag{2}$$

$$M(v_{ijk}) = \sum_{x=1}^{n} d_x \cdot \mathcal{S}\big(\Lambda(v_{ijk}), rs\big(\pi_x(v_{ijk})\big)\big)^{\delta_x} \tag{3}$$

$$\forall\, 0 < i \leq |R_k|,\ 0 < j \leq h_k,\ 0 < k < n_\alpha,\ \pi_x(v_{ijk}) \in \Pi(v_{ijk})$$

where $\Lambda(v_{ijk})$ is a polynomial function of cache size, cache line size, and associativity. a_x, b_x, c_x, and d_x are miss rate model cache correlation coefficients, α_x, β_x, γ_x, and

δ_x are polynomial orders, and $\mathcal{S}\big(\Lambda(v_{ijk}), rs\big(\pi_x(v_{ijk})\big)\big)$ is a mapping function that returns the miss rate, given $\Lambda(v_{ijk})$ and replacement strategy.

C. Delay Modeling

The delay model is used to estimate the transmission latency between two cores, which consists of two parts: (1) zero-load latency, which is related to the distance between the source and destination cores, (2) queuing delay, which is modeled by a queuing theory model.

1) Zero-load latency: By modeling the latency using G_I, $L(n_{p,q}, n_{r,s})$ is defined as the zero-load latency for node $n_{p,q}$ to access node $n_{r,s}$. The latency consists of two parts: the header packet latency $L_h(n_{p,q}, n_{r,s})$ and the serialization latency $L_s(n_{p,q}, n_{r,s})$ as follows.

$$L(n_{p,q}, n_{r,s}) = L_h(n_{p,q}, n_{r,s}) + L_s(n_{p,q}, n_{r,s}) \tag{4}$$

where the header packet latency $L_h(n_{p,q}, n_{r,s})$ is determined by the shortest path $l(n_{p,q}, n_{r,s})$ between node $n_{p,q}$ and node $n_{r,s}$ and the latency on the path. The serialization latency $L_s(n_{p,q}, n_{r,s})$ is related to the number of packet size (flit size). The formula for $L_s(n_{p,q}, n_{r,s})$ and $L_h(n_{p,q}, n_{r,s})$ is as follows.

$$L_s(n_{p,q}, n_{r,s}) = \mu \cdot \frac{s_i}{f} \tag{5}$$

$$L_h(n_{p,q}, n_{r,s}) = \eta \cdot l(n_{p,q}, n_{r,s}) \tag{6}$$

where μ is the transmission rate, s_i is the data packet size, f is the flit size, and η is the number of router pipeline stages.

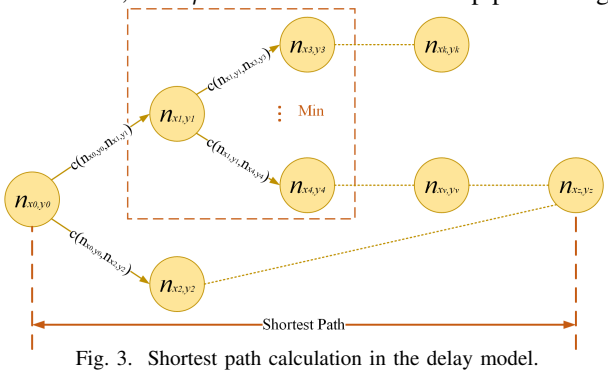

Fig. 3. Shortest path calculation in the delay model.

As shown in Fig. 3, the transmission latency of the shortest path is recursively derived using the Bellman equation [11], where the shortest path from node $n_{x_0,y_0} = n_{p,q}$ to node $n_{x_k,y_k} = n_{r,s}$ is referred to a sequence of nodes $\Gamma(n_{p,q}, n_{r,s}) = \Gamma(n_{x_0,y_0}, n_{x_k,y_k}) = \{n_{x_0,y_0} = n_{p,q}, n_{x_1,y_1}, \ldots, n_{x_z,y_z}, \ldots, n_{x_k,y_k} = n_{r,s}\}$. By comparing all possible paths between nodes, the shortest distance is ensured to be selected at each step of the recursion, as follows.

$$l^{(z)}(n_{x_0,y_0}, n_{x_z,y_z}) = \min\big\{l^{(z-1)}(n_{x_0,y_0}, n_{x_{z-1},y_{z-1}}) + c(n_{x_{z-1},y_{z-1}}, n_{x_z,y_z})\big\} \tag{7}$$
$$\forall n_{x_z,yz} \in \Gamma(n_{x_0,y_0}, n_{x_k,y_k})$$

where $l^{(z)}(n_{x_0,y_0}, n_{x_z,y_z})$ is the shortest distance from n_{x_0,y_0} to n_{x_z,y_z} in the z-th step of recursion. $l^{(z)}(n_{x_0,y_0}, n_{x_z,y_z})$ is recursively updated until the optimal state is achieved.

Therefore, the calculation of $l(n_{p,q}, n_{r,s})$ follows the Bellman equation as follows.

$$l(n_{p,q}, n_{r,s}) = l(n_{x_0,y_0}, n_{x_k,y_k}) = \sum_{z=1}^{k} \min\big\{\omega\big(c(n_{x_{z-1},y_{z-1}}, n_{x_z,y_z})\big)\big\} \tag{8}$$
$$\forall n_{x_z,y_z} \in \Gamma(n_{x_0,y_0}, n_{x_k,y_k})$$

where $l(n_{x_z,y_z}, n_{x_z,y_z}) = 0$, and $\omega\big(c(n_{x_{z-1},y_{z-1}}, n_{x_z,y_z})\big)$ is the transmission latency between adjacent nodes $n_{x_{z-1},y_{z-1}}$ and n_{x_z,y_z}.

2) Queuing delay: The queuing delay of two nodes $n_{p,q}$ and $n_{r,s}$ is the summation of the queuing delay of each router n_{x_z,y_z} in the path $\Gamma(n_{p,q}, n_{r,s})$, obtained by the routing algorithm. In the G/G/1 model, a single-server queue with general arrival and service distributions, the flows $f_{a_c,b_c} = \{f_{a_1,b_1}, \ldots, f_{a_g,b_g}\}$ traversing n_{x_z,y_z} are analyzed. A router n_{x_z,y_z} consists of multiple input channels $IC_i^{n_{x_z,y_z}} = \{IC_1^{n_{x_z,y_z}}, \ldots, IC_q^{n_{x_z,y_z}}\}$ competing for a single output channel $OC_j^{n_{x_z,y_z}}$. The average queuing delay $T_{i\to j}(n_{x_z,y_z})$ of router n_{x_z,y_z} for a packet from $IC_i^{n_{x_z,y_z}}$ to $OC_j^{n_{x_z,y_z}}$ is as follows [12].

$$T_{i\to j}(n_{x_z,y_z}) = \begin{cases} \dfrac{\rho_j(n_{x_z,y_z})\left(C_A^2(n_{x_z,y_z}) + C_{S_j}^2(n_{x_z,y_z})\right)}{2\left(\mu_j(n_{x_z,y_z}) - \lambda_{i\to j}(n_{x_z,y_z})\right)}, & i = 1 \\ \dfrac{\lambda_j(n_{x_z,y_z})\left(C_A^2(n_{x_z,y_z}) + C_{S_j}^2(n_{x_z,y_z})\right)}{2\left(\mu_j(n_{x_z,y_z}) - \sum_{l=1}^{i-1}\lambda_{l\to j}(n_{x_z,y_z})\right)^2}, & 2 \le i \le q \end{cases} \tag{9}$$

where $\rho_j(n_{x_z,y_z})$ is the fraction of time that the $OC_j^{n_{x_z,y_z}}$ is occupied by packets from $IC_i^{n_{x_z,y_z}}$ in router n_{x_z,y_z}, $\lambda_j(n_{x_z,y_z})$ is the average packet rate to $OC_j^{n_{x_z,y_z}}$, $\lambda_{i\to j}(n_{x_z,y_z})$ is the average packet rate from $IC_i^{x_z,y_z}$ to $OC_j^{x_z,y_z}$, $\mu_j(n_{x_z,y_z})$ is the average service rate of the $OC_j^{n_{x_z,y_z}}$. $C_A^2(n_{x_z,y_z})$ and $C_{S_j}^2(n_{x_z,y_z})$ are the correlation coefficients between the corresponding packet arrival rate and service rate obtained based on the Allen-Cunneen approximation formula [12].

The total queuing delay $W(n_{p,q}, n_{r,s})$ is the sum of queuing delays of each router $n_{x_z,y_z} \in \Gamma(n_{p,q}, n_{r,s})$ from node $n_{p,q}$ to node $n_{r,s}$ for $OC_j^{n_{x_z,y_z}}$, calculated by the following formula.

$$W(n_{p,q}, n_{r,s}) = \sum_{z=0}^{k}\sum_{i=1}^{q} T_{i\to j}(n_{x_z,y_z}) \tag{10}$$
$$\forall n_{x_z,y_z} \in \Gamma(n_{x_0,y_0}, n_{x_k,y_k})$$

In summary, $D(n_{p,q}, n_{r,s})$ is the sum of zero-load latency $L(n_{p,q}, n_{r,s})$ and queuing delay $W(n_{p,q}, n_{r,s})$, represented as follows.

$$D(n_{p,q}, n_{r,s}) = L(n_{p,q}, n_{r,s}) + W(n_{p,q}, n_{r,s}) \tag{11}$$

D. Memory Concurrency Modeling

According to [7], define average hit concurrency $C_H(v_{ijk})$ and ratio of pure miss cycles over miss cycles $\kappa(v_{ijk})$ from a memory-centric perspective. $C_H(v_{ijk})$ and $\kappa(v_{ijk})$ can be modeled by a regression as follows [7].

$$C_H(v_{ijk}) = \sum_{x=1}^{n} f_x \cdot mpki\big(\pi_x(v_{ijk})\big)^{\vartheta_x} + \sum_{x=1}^{n} l_x \cdot ac\big(\pi_x(v_{ijk})\big)^{\iota_x} + g_x \cdot \big(\ln\big(bw(v_{ihk})\big)\big)^{\xi_x} \tag{12}$$

where f_x, l_x, and g_x are coefficients, ϑ_x, ι_x, and ξ_x are polynomial orders, $\pi_x(v_{ijk})$ is the x-th node on the reachable path from the core to v_{ijk}, $mpki\big(\pi_x(v_{ijk})\big)$ is the number of memory accesses per thousand instructions of $\pi_x(v_{ijk})$, $ac\big(\pi_x(v_{ijk})\big)$ is the cache access numbers of $\pi_x(v_{ijk})$, and $bw(v_{ihk})$ is DRAM bandwidth.

$$\kappa(v_{ijk}) = \sum_{x=1}^{n} m_x \cdot mpki\big(\pi_x(v_{ijk})\big)^{\varrho_x} + o_x \cdot \left(\ln\big(M(v_{i(h_k-1)k})\big)\right)^{\zeta_x} \quad (13)$$

where m_x and o_x are correlation coefficients, ϱ_x and ζ_x are polynomial orders, and $M(v_{i(h_k-1)k})$ is the miss rate of LLC, which is defined in Eq. (3) in section 3.B.

E. Performance Modeling

Given the memory hierarchy graph G_M and inter-chiplet interconnection network topology graph G_I of the multi-chiplet systems mentioned above, with the miss rate model, delay model, and memory concurrency model, the performance model is computed as follows.

The point-to-point latency between two nodes in the inter- and intra-chiplet interconnection system is calculated using the delay model, and the average delay between adjacent cache levels j and $j-1$ is as follows.

$$\overline{D}(j-1,j) = \frac{\sum_{c=1}^{g}\left(\sum_{k=1}^{n_\alpha}\sum_{i=1}^{|R_k|} D(v_{i(j-1)k}, v_{ijk})\cdot f_{a_c,b_c}\right)}{\sum_{c=1}^{g} f_{a_c,b_c}} + \frac{\sum_{c=1}^{g}\left(\sum_{k=1}^{n_\alpha}\sum_{i=1}^{|R_k|} D(v_{ijk}, v_{i(j-1)k})\cdot f_{a_c,b_c}\right)}{\sum_{c=1}^{g} f_{a_c,b_c}} \quad (14)$$

where f_{a_c,b_c} is the flow traversing $v_{i(j-1)k}$ to v_{ijk} and in reverse, $c = 1, \ldots, g$.

Define $\mathcal{C}(v_{i(j-1)k})$ as the average concurrent memory access time of cache layer $j-1$ within a CPU cycle. $\mathcal{C}(v_{i(j-1)k})$ is closely related to the average concurrent memory access time of the next level cache $\mathcal{C}(v_{ijk})$, and the recursive relationship is as follows [7].

$$\mathcal{C}(v_{i(j-1)k}) = \frac{H(v_{i(j-1)k})}{C_H(v_{i(j-1)k})} + M(v_{i(j-1)k})\cdot \kappa(v_{i(j-1)k})\cdot\big(\mathcal{C}(v_{ijk}) + \overline{D}(j-1,j)\big) \quad (15)$$

The C-AMAT $\sigma(G_M, G_I)$ is as follows [7].

$$\sigma(G_M,G_I) = \sum_{k=1}^{n_\alpha}\sum_{i=1}^{|R_k|}\left(\sum_{j=1}^{h_k-1}\prod_{t=1}^{j} M(v_{itk})\cdot\kappa(v_{itk})\cdot\left(\frac{H(v_{ijk})}{C_H(v_{ijk})} + \overline{D}(j-1,j)\right)\right) \quad (16)$$

where $H(v_{ijk})$ is the hit time of the j-th level cache of the chiplet k, $C_H(v_{ijk})$ is the average hit concurrency of the j-th level cache of the chiplet k, $M(v_{ijk})$ is the miss rate of the j-th level cache of the chiplet k, $\kappa(v_{ijk})$ is the ratio of pure miss cycles over miss cycles, $\overline{D}(j-1,j)$ is the average latency between adjacent $(j-1)$-th level and j-th level caches, n_α is the chiplet number, h_k is the total number of memory hierarchy in chiplet k, $|R_k|$ is the number of cores in chiplet k, and t is the recursive levels in the product up to j.

F. Cost and Power Consumption Modeling

The architecture cost is the cumulative cost of each component (die and DRAM) and the chiplet integration cost c_{int}. The cost of a die $C_{die}(v_{ijk})$ is as follows.

$$C_{die}(v_{ijk}) = \frac{A_i^{die}}{Y^{die}}\cdot c_1 \quad (17)$$

$$Y^{die} = \left(1 + \frac{\mathcal{D}\cdot\sum_{i=0}^{|R_i|} A_i^{die}}{a}\right)^{-a} \quad (18)$$

where A_i^{die} is the die area, c_1 is the cost per silicon area, Y^{die} is the asymptotic yield determined by the die area A_i^{die}, defect density $\mathcal{D}$, and clustering factor a.

The cost of DRAM $C_{dram}(v_{ijk})$ is expressed as:

$$C_{dram}(v_{ijk}) = \frac{bw(v_{ijk})}{U_{bw}}\cdot c_2 \quad (19)$$

where $bw(v_{ijk})$ is DRAM bandwidth, c_2 is the monetary cost of DRAM, and U_{bw} is the bandwidth provided by each unit.

The cost modeling of the multi-chiplet systems is as follows.

$$C(G_M,G_I) = \sum_{k=1}^{|R_k|} C_{die}(v_{ijk}) + C_{dram}(v_{ijk}) + c_{int} \quad (20)$$

The system power consumption is the cumulative power of each component. The power modeling of the multi-chiplet system is as follows.

$$P(G_M,G_I) = \sum_{i=1}^{n_\alpha} P_{core}(v_{ijk}) + P_{dram}(v_{ijk}) + \sum_{k=1}^{|R_k|} P_{router}(v_{ijk})\cdot r_k \quad (21)$$

where $P_{core}(v_{ijk})$ is the core power consumption, $P_{dram}(v_{ijk})$ is the DRAM power consumption, $P_{router}(v_{ijk})$ is the router power consumption, and r_k is the number of routers on each chiplet or interposer.

G. Problem Definition

With the performance, power, and cost model, under the constraints of cost and power, the optimization problem is defined to achieve the best performance of the multi-chiplet systems to minimize the C-AMAT $\sigma(G_M, G_I)$ with decision variables being G_M and G_I. The cost and power consumption of each chiplet should not exceed the respective threshold C_T and P_T. The optimization problem can be defined as follows.

$$\begin{aligned} &\min\{\sigma(G_M,G_I)\} \\ &\text{s.t. } C(G_M,G_I) \le C_T,\ P(G_M,G_I) \le P_T \end{aligned} \quad (22)$$

The problem defined above has a huge search space. To efficiently solve this problem, a bilevel optimization algorithm is proposed in the next section.

IV. The Proposed Design Space Exploration Algorithm

A. Bilevel Optimization Algorithm Overview

The bilevel optimization algorithm works as follows. The problem defined in Eqn. (2)-(21) is decomposed into two subproblems: (1) cache subsystem optimization subproblem (P1), and (2) inter-chiplet interconnection network topology optimization subproblem (P2).

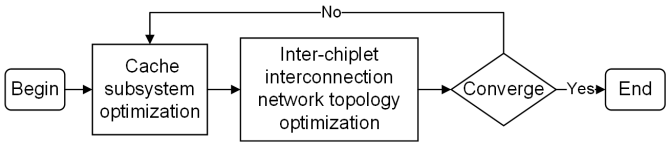

Fig. 4. Workflow of the bilevel optimization.

The workflow of bilevel optimization is shown in Fig. 4, which is performed iteratively. In each iteration, only one subproblem is solved, with the variables to the other subproblem fixed as constraints.

The optimization alternates between P1 and P2 until the target performance metric σ converges or a predefined maximum number of iterations $n_{\max}$ is reached. The convergence criteria are given by:

$$\sigma_i - \sigma_{i-1} \leq \delta \quad \text{or} \quad i \leq n_{\max} \tag{23}$$

where σ_i is the C-AMAT of the application in the i-th iteration, δ is the convergence condition, and $n_{\max}$ is the iteration limit.

B. Cache Subsystem Optimization (P1)

1) Search space definition: The objective of P1 is to select G_M parameters (cache level h_i, the numbers of cores $|R_i|$ in chiplet i, and parameters for each level of cache $w(v_{ijk})$), and it is defined as follows:

$$\begin{aligned} &\min \left\{\sigma(G_M) = \sigma\big(h_i, |R_i|, w(v_{ijk})\big)\right\} \\ &\text{s.t. } C(G_M) \leq C_T,\ P(G_M) \leq P_T \end{aligned} \tag{24}$$

This subproblem can be solved by a branch-and-bound algorithm. In this algorithm, the decision variables in this subproblem are formed as a tree node g_b^a, which is the a-th node on the b-th layer and has a total of n search quantities $\hat{x_{cb}}^a$. A search tree is formed, where a tree node g_b^a is defined as follows.

$$g_b^a = \langle \hat{x_{1b}}^a, \hat{x_{2b}}^a, \ldots, \hat{x_{nb}}^a \rangle \tag{25}$$

Each node g_b^a is associated with an objective value $\sigma(g_b^a)$, and $C(g_b^a)$ and $P(g_b^a)$ are the cost and power consumption, respectively.

2) Branching rules: Each node g_b^a corresponding to $\hat{x_{cb}}^a$ assigned will branch a maximum of p_c branches, corresponding to $\hat{x_{c+1b}}^a$ assigned to be one of $\{y_1, y_2, \ldots, y_{p_c}\}$, which is the value range of $\hat{x_{c+1b}}^a$.

3) Cut rules: The cut rules are as follows.

- Pruned by optimality: When the currently searched node g_b^a is a leaf node and the feasible solution $\sigma(g_b^a)$ obtained by the current node is still greater than the current global minimum value, it is pruned.
- Pruned by infeasibility: When the currently searched node g_b^a does not meet the constraint conditions mentioned above, i.e., $C(g_b^a) \leq C_T$ or $P(g_b^a) \leq P_T$, where $C(g_b^a)$ and $P(g_b^a)$ return cost and power given the tree node g_b^a, it is pruned.
- Pruned by LPM unmatching: LPM [7] is an indicator of the matchness of parameters of adjacent cache levels. If LPM rate is over a threshold $x\%$, it indicates the match is low and leads to poor performance.

  Based on the performance model mentioned above, we use the LPM model [7] to verify the utilization of each cache layer. $LPMR(v_{ijk})$ is defined as the matching ratio at cache level j as follows.

  $$LPMR(v_{ijk}) = \frac{f_{mem} \cdot \mathcal{C}(v_{i1k}) \cdot \prod_{j=1}^{h_k} \mu(v_{ijk})}{cpi(v_{ijk})} \tag{26}$$

  where f_{mem} is the average number of memory accesses per instruction of the core, and $\mu(v_{ijk})$ is the ratio of miss cycles over memory active cycles, both of which can be obtained through the polynomial regression models.

Algorithm 1: Branch and Bound Algorithm

Input : The cache level h_i, the numbers of cores $|R_i|$ in chiplet i, and the parameters for each level of cache $w(v_{ijk})$.
Output: The cache access time of application σ^*.
Initialization:
- Initialize WQ as a working queue;
- Set g_b^a as the current node, and g_{b+1}^a as the newly branched node;
- Initialize $G^* = \emptyset$ as the best node;
- Set $\sigma^* = +\infty$ (optimal value found so far);
- Push root node into WQ;

```
while WQ is not empty do
    Pop the top node g_b^a out of WQ;
    if g_b^a is not a leaf node then
        Branch new tree nodes;
        foreach newly branched node g_{b+1}^a do
            if no cutting rules are met for g_{b+1}^a and
              WQ is not full then
                Push g_{b+1}^a into WQ;
    else
        if σ(g_b^a) ≤ σ* then
            Update σ* = σ(g_b^a);
            Set G* = g_b^a;
```

  Define a threshold $x\%$ as the optimization target for the memory system, starting with matching at the L1 cache. Using the LPM model efficiently eliminates suboptimal branches. It ensures only configurations meeting the performance threshold are explored, thus reducing search overhead.
- Pruned by Length of WQ When the length of WQ is greater than W, it is pruned.

4) Workflow: Algorithm 1 outlines how the search process works. The algorithm works as follows.

- Initialization: Initialize σ^* to $+\infty$, which records the best objective function so far. Initialize g^* to an empty state ($\emptyset$), which will record the best node found so far in the search tree. The root node is enqueued into WQ.
- Search process: In each iteration of the search process, check if the work queue WQ is not empty. If WQ is not empty, g_b^a is dequeued from WQ. If g_b^a is not a leaf node, branch new tree nodes according to the branch rules. New nodes are enqueued into WQ if they don't meet the cut rules. If g_b^a is a leaf node, check if it is the best solution. If so, $\sigma^* = \sigma(g_b^a)$.
- Termination: If at any point the working queue WQ becomes empty, the search process terminates.

The time complexity of the BNB algorithm depends on the size of the search tree, determined by the branching factor p_c and the depth h_d. In the worst case, it explores $\mathrm{O}(p_c^{h_d})$ nodes, with each node involving $\mathrm{O}(1)$ operations for branching, pruning, and comparison. Pruning strategies (optimality, infeasibility, LPM, WQ length) significantly reduce the effective number of nodes, making the practical complexity $\mathrm{O}(\max\{|WQ|, p_c^{h_d}\})$.

C. Inter-chiplet Interconnection Network Topology Optimization (P2)

The objective of P2 is to select an inter-chiplet interconnection topology for G_I, and flit size f_i, and it is defined as follows.

$$\min\{\sigma(G_I)\} \quad \text{s.t. } C(G_I) \leq C_T,\ P(G_I) \leq P_T \tag{27}$$

To solve this problem, a few predefined topologies $t_i = \langle t_1, t_2, \ldots, t_j \rangle$, including Mesh, Ring, Torus, Crossbar, Clos, and Fat tree, and the flit size $f_i = \{32, 64, \ldots, 1024\}$, will be selected. An exhaustive search is performed to find the optimal topology and flit size.

V. Experimental Results

A. Experiment Setup

Experiments were performed on the sniper simulator with McPAT as the power simulator.

TABLE II
THE PARALLEL MULTI-CHIPLET X86-64 SIMULATOR CONFIGURATION FOR DESIGN SPACE EXPLORATION

Configuration	
Core architecture	x86-64
Main memory size	2GB
Get/Decode/Submit Size	4/4/4
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
Level 2 cache (shared)	64KB slices /core, 64 line, 6 cycles, 2 ports
Flit size	32, 64, 128, 256, 512, 1024 bits
Intra-chiplet communication latency	Router: 2 cycles; link: 1 cycle
Inter-chiplet communication latency	Router: 2 cycles
Buffer depth	4 flits
Benchmarks	
barnes, cholesky, fft, lu.cont, radiosity, radix, raytrace, streamcluster, swaptions, water.nsq, CNN, GCN, ResNet50, TPC_H	

Table 2 lists the configurations of the multi-chiplet systems simulator. The benchmarks are from PARSEC, SPLASH-2, neural network, and database applications. The area and power model of routers are from DSENT, the area and power model of SRAM are from CACTI, and the area and power model of processor cores are from McPAT. The transmission energy consumption between adjacent chiplet is 1.17 PJ/bit [5], and the transmission latency between adjacent chiplets is composed of the following three parts: 1) packetization and depacketization overheads in the D2D interface, adopted from [13], 2) transceiver transmission latency, adopted from [14], and 3) interposer wire delay and power, adopted from [15].

As shown in Fig. 5, the selection of the $x\%$ threshold for the LPM model was obtained through experiments. Fig. 5 shows that $x = 65\%$ strikes a good balance of average application performance and algorithm search time, and thus is used in the experiments.

To evaluate the algorithm proposed in this paper, IntLP [4] is chosen for comparison. IntLP uses an integer linear programming model to determine the optimal partitioning of functional components and the selection of chiplets. In addition, Zen 4 and SPR are simulated in sniper to consider for comparison.

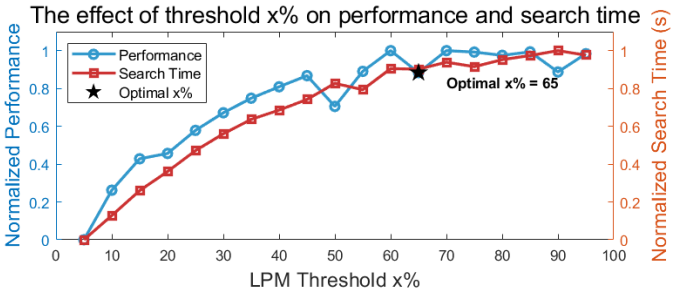

Fig. 5. The effect of threshold $x\%$ on the average application performance and the algorithm search time on the above benchmarks.

B. Performance Evaluation

In the subsequent experiments, the performance of the proposed method, Zen 4, SPR, and IntLP [4] were compared.

Fig. 6 (a) and (b) show the execution time comparison with power budgets of 100W and 160W respectively. From Fig. 6 (a), one can see that the execution time of the proposed method reduces 39.7%, 39.2%, and 25.91% over Zen 4, SPR, and IntLP, respectively. From Fig. 6 (b), one can see that at low power budget, the execution time of the proposed method reduces 40.61%, 39.49%, and 18.01% over Zen 4, SPR, and IntLP, respectively. The execution time of the four methods is normalized to that of Zen 4. The reason is that the proposed method can effectively explore the design space of cache subsystems in multi-chiplet systems, while Zen 4 and SPR are fixed designs, and IntLP can only optimize size and associativity for L1 and L2 caches, which fails to explore the design space well.

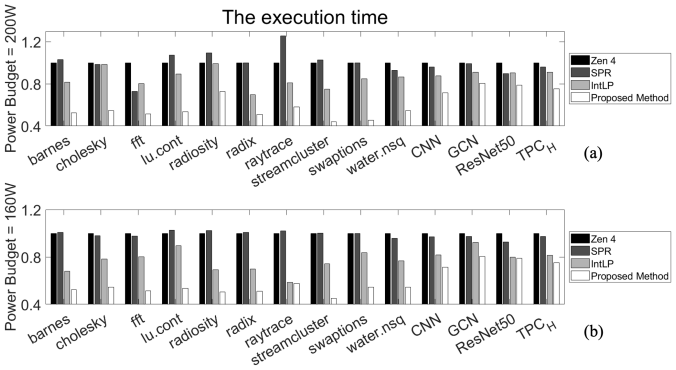

Fig. 6. The execution time of the different methods with (a) 200W and (b) 160W power budgets.

VI. Conclusion

In this paper, the problem of optimizing cache hierarchy in multi-chiplet based many-core systems was studied. Graph models were used to model the performance with respect to cache and inter-chiplet interconnection parameters. An optimization problem was formulated to minimize the cache access time of applications under cost and power constraints. A bilevel optimization algorithm was proposed to efficiently solve the above problem by iteratively solving two subproblems: (1) cache subsystem optimization and (2) inter-chiplet interconnection optimization. Experiment results confirmed superiority of the proposed method over existing ones. Thus, it is suitable for cache hierarchy optimization for future large scale multi-chiplet systems.

REFERENCES

[1] J. Kim et al., "Architecture, Chip, and Package Co-design Flow for 2.5D IC Design Enabling Heterogeneous IP Reuse," in *DAC*, 2019, pp. 1–6.

[2] R. Bhargava and K. Troester, "AMD Next-Generation 'Zen 4' Core and 4th Gen AMD EPYC Server CPUs," *IEEE Micro*, vol. 44, no. 3, 2024, pp. 8–17.

[3] N. Nassif et al., "Sapphire Rapids: The Next-Generation Intel Xeon Scalable Processor," in *ISSCC*, 2022, pp. 44–46.

[4] S. Pal, D. Petrisko, R. Kumar, and P. Gupta, "Design Space Exploration for Chiplet-Assembly-Based Processors," *IEEE Trans. VLSI*, vol. 28, no. 4, 2020, pp. 1062–1073.

[5] Z. Tan, H. Cai, R. Dong, and K. Ma, "NN-Baton: DNN Workload Orchestration and Chiplet Granularity Exploration for Multichip Accelerators," in *ISCA*, 2021, pp. 1013–1026.

[6] X. Hao, Z. Ding, J. Yin, Y. Wang, and Y. Liang, "Monad: Towards Cost-Effective Specialization for Chiplet-Based Spatial Accelerators," in *ICCAD*, 2023, pp. 1–9.

[7] J. Liu, P. Espina, and X.-H. Sun, "A Study on Modeling and Optimization of Memory Systems," *JCST*, vol. 36, no. 1, 2021, pp. 71–89.

[8] M. K. Qureshi, A. Jaleel, Y. N. Patt, S. C. S. Jr, and J. Emer, "Adaptive Insertion Policies for High Performance Caching," *SIGARCH Comput. Archit. News*, vol. 35, no. 2, 2007, pp. 381–391.

[9] A. Barai, G. Chennupati, N. Santhi, A.-H. Badawy, Y. Arafa, and S. Eidenbenz, "PPT-SASMM: Scalable Analytical Shared Memory Model: Predicting the Performance of Multicore Caches from a Single-Threaded Execution Trace," in *MEMSYS*, 2021, pp. 341–351.

[10] S. A. Przybylski, *Cache and memory hierarchy design: a performance directed approach*. Morgan Kaufmann, 1990.

[11] R. Al-Dujaily, N. Dahir, T. Mak, F. Xia, and A. Yakovlev, "Dynamic programming-based runtime thermal management (DPRTM): An online thermal control strategy for 3D-NoC systems," *ACM TODAES*, vol. 19, no. 1, 2013, pp. 1–27.

[12] A. E. Kiasari, Z. Lu, and A. Jantsch, "An Analytical Latency Model for Networks-on-Chip," *IEEE Trans. VLSI*, vol. 21, no. 1, 2013, pp. 113–123.

[13] A. Traber, F. Zaruba, and S. Stucki, "PULPino: A small single-core RISC-V SoC. In 3rd RISCV Workshop," in *RISCV Workshop*, 2016.

[14] 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," *IEEE Trans. CPMT*, vol. 12, no. 9, 2022, pp. 1423–1431.

[15] M. A. Kabir, D. Petranovic, and Y. Peng, "Coupling extraction and optimization for heterogeneous 2.5D chiplet-package co-design," in *ICCAD*, 2020, pp. 1–8.