AuxiliarySRAM: Exploring Elastic On-Chip Memory in 2.5D Chiplet Systems Design

Zichao Ling
Beijing University of Posts and Telecommunications
Beijing, China
lingzichao@bupt.edu.cn

Lin Li
Beijing University of Posts and Telecommunications
Beijing, China
lilin20030420@163.com

Yi Huang
Beijing University of Posts and Telecommunications
Beijing, China
2021212483@bupt.cn

Yixin Xuan
Beijing University of Posts and Telecommunications
Beijing, China
xuan_yixin@bupt.edu.cn

Jianwang Zhai*
Beijing University of Posts and Telecommunications
Beijing, China
zhaijw@bupt.edu.cn

Kang Zhao
Beijing University of Posts and Telecommunications
Beijing, China
zhaokang@bupt.edu.cn

Abstract

The "Memory Wall" dilemma remains a critical challenge in modern computing systems. While latency-sensitive applications increasingly rely on costly on-chip SRAM to meet performance requirements, SRAM scaling faces bottleneck. Currently, Chiplet-based techniques present a promising solution to this challenge by enabling optimized trade-offs between latency, capacity, and cost.

This paper introduces AuxiliarySRAM, a design methodology that decouples SRAM resources into on-die and extended chiplets, enabling elastic capacity-latency scaling. Key contributions include: (1) a lightweight network-on-chip (NoC) with simplified crossbars, dual local ports, and address prediction to reduce average latency by 49.29% and boost bandwidth by 79.35%; (2) a evaluation framework integrated with Bayesian optimization (BO) to resolve Pareto-optimal on/off-die capacity ratios, accelerated by pruning strategies (1.93× speedup); and (3) system-level evaluation provides Pareto frontier-based design guidelines and demonstrates its cost-saving advantages.

CCS Concepts

• **Hardware** → *Modeling and parameter extraction.*

Keywords

Memory Architecture, Chiplet System, Lightweight Network on Chip, Design Space Exploration

ACM Reference Format:

Zichao Ling, Lin Li, Yi Huang, Yixin Xuan, Jianwang Zhai, and Kang Zhao. 2025. AuxiliarySRAM: Exploring Elastic On-Chip Memory in 2.5D Chiplet Systems Design . In *Great Lakes Symposium on VLSI 2025 (GLSVLSI '25), June 30-July 2, 2025, New Orleans, LA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3716368.3735238

*Corresponding author.

GLSVLSI '25, New Orleans, LA, USA

ACM ISBN 979-8-4007-1496-2/2025/06
https://doi.org/10.1145/3716368.3735238

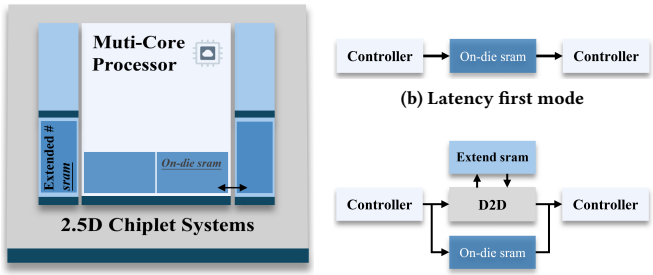

Figure 1: The concept of AuxiliarySRAM Chiplet architecture.

1 Introduction

Termed the "Memory Wall" phenomenon [1] exacerbated by semiconductor process scaling limitations, underscores a fundamental imbalance: Peak hardware floating-point operations per second (FLOPS) have outpaced dynamic random-access memory (DRAM) bandwidth by orders of magnitude (about 100x in the past 20 years) [2]. As a result, memory constraints dominate performance limitations in the post-Moore era. This predicament predominantly arises from two categories of applications: bandwidth-bound and latency-sensitive. Currently, high bandwidth memory (HBM) [3] provides an effective solution for bandwidth-bound applications, e.g., graphics computing and AI acceleration. However, latency-sensitive applications (e.g., autonomous driving decision-making and high-frequency trading systems) still rely on high-speed yet costly caches consisting of on-chip static random-access memory (SRAM).

Despite the reliance of applications on high-speed on-chip memory continues to escalate—even occupying over 50% of die area (e.g., Cortex-A76 [4])—on-chip SRAM arrays cannot be arbitrarily expanded. This fundamental limitation stems from two critical constraints: a) manufacturing costs escalate exponentially with SRAM footprint expansion, particularly at advanced process nodes below 7nm [5]; and b) capacity scaling paradoxically increases access latency due to elongated addressing paths and signal propagation delays. Consequently, a latency-capacity-cost trilemma emerges, compelling designers to make trade-offs.

Chiplet, a revolutionary heterogeneous integration technology, employs silicon interposers and through-silicon via (TSV) to enable modular system assembly, presenting new possibilities for solving

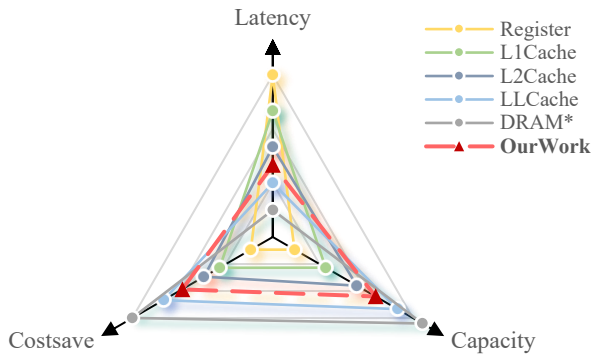

Figure 2: Position of AuxiliarySRAM in traditional Memory Architecture. *HBM is classified into DRAM.

this dilemma [6]. This paradigm significantly alleviates the substantial design, verification, and manufacturing overhead associated with monolithic system-on-chip (SoC) implementations [7]. Engineering practice (e.g., 3D V-Cache® [8]) demonstrate that fully disaggregation of low-level caches achieves unprecedented capacity scaling. Nevertheless, this approach incurs fundamental design hesitancy due to the complex process and inherent latency penalties introduced by comprehensive inter-Chiplet interconnect integration.

We posit that such concerns can be eliminated through a trade-off design and algorithmic synergy, thereby proposing an innovative AuxiliarySRAM Chiplet architecture (Figure 1(a)) inspired by the auxiliary fuel tank mechanism in jet fighters. This structure disassembles the on-chip memory of a specific functional module into on-die retention and extended Chiplet. External Chiplet activation is dynamically determined by application contexts through adaptive scheduling algorithms, enabling elastic switch between latency-sensitive (Figure 1(b)) and capacity-prioritized (Figure 1(c)) modes, while achieving simultaneous energy efficiency optimization. Key design blueprints are outlined below:

$\mathcal{A}$. Design elastic capacity scaling with multi-tiered latency management to optimally fulfill workload demands while enhancing energy efficiency.
$\mathcal{B}$. Design a multi-design compatible SRAM chiplet shared library to reduce non-recurring engineering (NRE) costs and minimize the main die area.

In this work, we primarily focus on the hardware exploration and modeling-based evaluation of the proposed design paradigm. Figure 2 illustrates the position of AuxiliarySRAM in the traditional memory architecture. Firstly, we develop a multi-bank chiplet employing a lightweight network-on-chip (NoC) and establish a quantitative model. To address the design space challenge in on-die/off-die optimization, we implement Gaussian process (GP)-based Bayesian optimization (BO) to identify the Pareto frontier. Furthermore, we incorporate this methodology into the GIA [9, 10] chiplet integration framework, systematically assessing composite chiplet performance metrics in system-level contexts. Finally, we address parameter explosion and placement constraints by optimizing the selection engine and annealing process, boosting framework performance. The main contributions are as follows:

- We propose AuxiliarySRAM, an innovative partition decoupling on-chip memory into on-die part and extended chiplet, establishing a cost-effective method to overcome conventional SRAM constraints through dynamic scaling across multiple capacity-latency tiers.
- We design a lightweight on-chip network and adopted series of optimization techniques to enable the SRAM chiplet to reduce access latency and increase maximum bandwidth while maintaining scalability, attaining practical applicability.
- We expand the capabilities of the original Chiplet evaluation framework, integrated a BO method with GP to solve the Pareto frontier of capacity ratios, and accelerated the solving process through pruning.
- Experimental results demonstrate that the lightweight NoC reduces average latency by 49.29% and enhances maximum bandwidth by 79.35%, while system-level evaluation reveals that the AuxiliarySRAM scheme achieves a maximum cost reduction of 26.3% under equivalent capacity, and the pruning method attains a 2.13x speedup.

Table 1: Specification of Typical Chiplet Interconnect

Specification	ACC	AIB	BoW	SerDes	UCIe
Max Rate (Gbps/wire)	128	2	32	224	32
Latency (ns)	6	3.56	2-4	/	<2
Power (pJ/bit)	2.5	0.85	0.25-10	0.8	0.25-1.25
Transmission	Serial	Parallel	Parallel	Serial	Serial

2 Preliminaries

2.1 Chiplet Interconnction

Chiplet interconnection forms the backbone of communication in chiplet-based architectures, enabling data exchange between heterogeneous silicon dies within a single package. These interfaces called Die-to-Die (D2D), typically employ advanced packaging technologies such as silicon interposers, flip-chip, TSVs, and microbumps to establish dense vertical interconnects with sub-millimeter pitches.

As a latency-sensitive design, the performance of D2D interconnects critically determines the viability of SRAM Chiplet integration. Modern D2D standards (summarized in Table 1) adopt distinct physical and protocol-layer strategies to balance bandwidth, energy efficiency, and latency between TX/RX, substantiating the foundation for the quantitative modeling framework.

2.2 Evaluation Framework

We adopt GIA design automation framework [9, 10] to systematically evaluate and optimize Chiplet systems. It includes four stages: chiplet selection, network topology generation, chiplet placement, and interposer mapping. 1) Chiplet selection employs integer linear programming (ILP) to map application tasks onto chiplets while minimizing power, latency, and cost under bandwidth and resource constraints. 2) Network topology generation clusters high-communication chiplets using graph partitioning to reduce latency. 3) A simulated annealing (SA) based placer optimizes thermal profiles and interposer power by iteratively perturbing chiplet positions and orientations. Finally, 4) a routability-driven mapping algorithm assigns channels and configures routers to realize the target topology, prioritizing high-traffic paths.

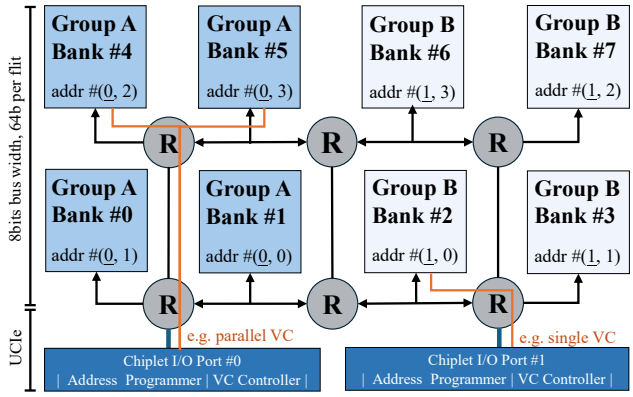

Figure 3: 2×4 SRAM Bank Mesh Chiplet with dual D2D-ports. Each memory bank is configured with an 8 KB capacity, 64-bit data width, and 1 read/write port shared by up to 2 routers.

3 Design Methodologies

NoC architecture is chosen over conventional shared-bus or fully-connected structure to simultaneously satisfy scalability, parallelism, and large capacity requirements. In this work, we propose a lightweight, high-performance, and energy-efficient NoC-based SRAM Chiplet that employs a 2D Mesh topology (as Figure 3) to minimize area occupation and routing design complexity. The design incorporates a streamlined router unit featuring a simplified crossbar structure and accelerated routing algorithms, while implementing an address prediction mechanism to mitigate cross-boundary overhead. The architecture is further enhanced by a group-aware configurable address space allocation strategy that improves system flexibility.

3.1 Lightweight NoC Interconnection

Router Design. Routers serve as the core components of NoC, primarily consisting of a virtual channel (VC) allocator, a crossbar switch, and a buffer module. Each router incorporates 4 bidirectional directional ports and supports up to 2 local access ports, which could optimize interleaved access and cross-border latency. The buffer is implemented through a depth-configurable synchronous FIFO, which pipelines data flits received from neighboring routers or local network interface (NI) units. The routing algorithm employs deterministic XY dimension-ordered routing, with path selection strictly adhering to a horizontal-then-vertical priority scheme.

Thin Crossbar. It is observed that traffic in different directions exhibits significant differences. Thereby we propose a simplified crossbar architecture by optimizing directional bypass and dual local port mediation based on Work [11], avoiding the $O(N^2)$ spatial complexity while reducing delays from $N \times 1$ Mux.

Let $P = \{N, S, E, W, LL, LR\}$ denote the set of ports (North, South, East, West, LocalLeft, LocalRight), $M_{d:1}$ represents a d-input mux. σ_* represents the signal selector. Then we have:

- *Local ports* LL, LR integrate an $M_{4:1}$ with inputs from $\{N, S, E, W\}$, governed by a 2-bit $\sigma_L \in \{00, 01, 10, 11\}$.
- *Directional pairs* ($N \leftrightarrow S$, $E \leftrightarrow W$) are interconnected via $M_{2:1}$ modules, enabling direct *passthrough* selection through control signals $\sigma_{NS}, \sigma_{EW} \in \{0, 1\}$.
- *Orthogonal pairs* are mediated through *local port* via a two-phase protocol (e.g., $N \rightarrow LL \rightarrow E$): 1) The $M_{4:1}$ at L assigns source port selection (N) by encoding directional input $\phi(N) \rightarrow \sigma_L$, where $\phi : P \setminus \{L\} \rightarrow \{00, 01, 10, 11\}$ bijectively maps ports to control codes; 2) The destination port's $M_{2:1}$ (e.g., E) reconfigures σ_{EW} to prioritize L-sourced data, enabling non-blocking connectivity with optimized multiplexer allocation.

The control logic employs a distributed arbitration scheme, where each $M_{2:1}$ and $M_{4:1}$ operates under locally generated σ signals synchronized via a lightweight FSM. This reduces crossbar complexity from $O(|P|^2)$ to $O(|P|)$ in directional pairs, with worst-case path latency bounded by $\tau_{\text{M2:1}} + \tau_{\text{M4:1}}$ for cross-directional transfers.

Address Prediction. A lightweight self-adaptation address prediction mechanism in router nodes, consisting of three key components: 1) A *sliding window register* $W = \{A_{t-n}, A_{t-n+1}, ..., A_t\}$ storing the last n addresses in history; 2) A *gradient-based predictor* computing address increments $\Delta_i = A_i - A_{i-1}$ ($i \in [t-n+1, t]$); 3) A FSM classifying traffic patterns. If the router receives *burst transaction* request or Δ_i remains constant $\Delta_i = \Delta_{i-1}$ for k consecutive samples, the FSM triggers burst mode, pre-allocating VCs to the predicted destination node via priority mask.

Pivotal features include a neighbor-region detection module and a power coordination module. The former dynamically calculates address offsets and pre-activates adjacent bank node routing paths or downstream banks on virtual channels via FSM when thresholds are exceeded. The latter integrates a distributed power-gating architecture, where a router wake-up predictor analyzes address access periodicity to dynamically issue power mode instructions. Concurrently, a bank-local sleep manager drives a tri-state machine (active/standby/shutdown) to disconnect bank node power when idle periods exceed thresholds, while pre-charge circuits ensure wake-up latency constraints [12].

3.2 Multi-Bank Reconfiguration

Decoupling physical banks indices from logical address spaces enables flexible post-silicon reconfiguration. Each router node integrates up to $B = 2$ independent banks, where the physical address space is partitioned into $B \times C$ memory units (C denotes single-bank capacity). For *interleaved access*, an address mapping function $\mathcal{M}(A)$ decomposes the global address A into bank-selection bits A_k ($k = \log_2 B$) and bank-offset address A_C, defined as:

$$\begin{aligned} \mathcal{M}(A) &= (A \gg (n-k)) \mod B, \\ A_C &= A \mod C. \end{aligned} \tag{1}$$

where n is the global address bus width. An odd-even interleaving strategy governs dual-bank access: $\mathcal{M}(A) = 0$ triggers Bank0 access at A_C, while other cases target Bank1, enabling parallel distribution of contiguous address requests across banks. This theoretically elevates peak bandwidth to $B \times \omega$ (ω: single-bank interface bitwidth).

We introduce a *programmable address offset register* $\Delta \in \{0, 1\}^k$ to enable post-silicon address reconfiguration, dynamically optimizing storage allocation per application to minimize latency. The revised mapping function becomes:

$$\mathcal{M}'(A) = ((A \oplus \Delta) \gg (n-k)) \bmod B. \tag{2}$$

Furthermore, upon detecting bank faults or process-induced variation performance asymmetry, the system updates Δ to redistribute hot-zone address ranges, ensuring load balancing.

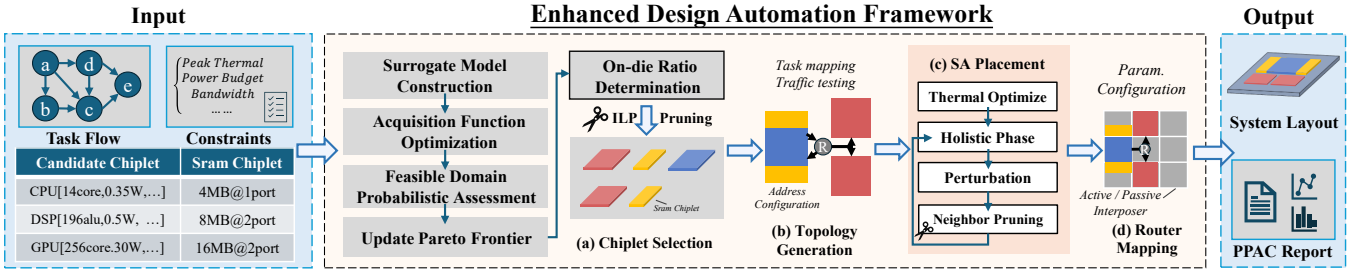

Figure 4: The proposed framework for AuxilarySRAM Chiplet systems extended from GIA.

4 Framework Augmentation

Figure 4 illustrates the framework for the automated design and evaluation of chiplet systems incorporating AuxiliarySRAM. The framework first introduces a quantitative modeling approach to systematically determine the on/off-die ratio for each design requirement. Subsequently, the optimal candidate is selected from a pool of input SRAM chiplet options, which is then subjected to GIA workflow to evaluate critical metrics, including power, performance, area, thermal, and cost efficiency. Furthermore, two novel pruning strategies are introduced to enhance the computational efficiency.

4.1 Quantitative Modeling

From the designers' perspective, once the on-chip memory specifications across the processor are frozen, it's crucial to determine whether it is suitable to implement using chiplet approaches and to decide the optimal ratio of on/off-die capacity. This decision impacts the subsequent selection from the SRAM library to achieve an excellent design. We address this issue through quantitative modeling.

Formally, we define the system parameters as a quintuple:

$$\mathcal{D} = \langle C_{on}, C_{off}, W, S_{tsv}, T_{tech} \rangle \tag{3}$$

where $C_{on} \in [0, C_{total}]$ represents the capacity of on-die SRAM. Assume that C_{total} is given, then $C_{off} = C_{total} - C_{on}$ represents the capacity of extended chiplet volume. $W \in \{32\text{bit}, 64\text{bit}, 128\text{bit}\}$ represents the width of the data bus in chiplet interconnection. $S_{tsv} \in \{\text{UCIe, SerDes, BoW, ACC, AIB}\}$ indicates the protocol standard shown in Table 1. $T_{tech} \in \{28\text{nm}, 14\text{nm}, 7\text{nm}, 5\text{nm}, 3\text{nm}\}$ indicates the technology node.

Optimization Objective. The proposed objective function is defined as a triple-objective function, aiming to perform multi-objective optimization under given constraints:

$$\min_{\mathcal{D}} \left(f_{latency}, f_{power}, f_{cost} \right). \tag{4}$$

Latency Modeling. Latency is related to modular functionality and should establish the relationship between capacity and delay based on specific application scenarios. Generally, we have:

$$f_{\text{latency}} = \underbrace{\alpha_1 H_{\text{on}}}_{\text{On-chip}} + \underbrace{\gamma_1 \frac{D_{\text{data}}}{B_{\text{link}}}}_{\text{Interconnect}} + \underbrace{\alpha_2 (1 - H_{\text{on}}) \cdot \left(\beta_1 + \beta_2 \frac{C_{\text{off}}}{D_{\text{block}}} \right)}_{\text{Chiplet Access}}, \tag{5}$$

where H_{on} is the on-die part access rate, D_{data} is task-level data transfer volume, $B_{\text{link}} = W \cdot f_{\text{clk}}$ is the bandwidth of the link, D_{block} is data block size, and $\alpha_i, \beta_i, \gamma_i$ are coefficients.

Power Modeling. The total power consumption consists of *leakage power* and *dynamic operating power*:

$$f_{\text{power}} = P_{\text{leakage}} + P_{\text{dynamic}}. \tag{6}$$

The leakage power dissipation is formulated as:

$$P_{\text{leakage}} = \underbrace{I_{\text{leak}} \cdot V_{\text{dd}} \cdot A_{\text{sram}}}_{\text{Chip leakage}} + \underbrace{I_{\text{tsv\_leak}} \cdot V_{\text{tsv}}}_{\text{TSV leakage}}, \tag{7}$$

where A_{sram} is activate SRAM area (mm^2), computed as $A_{\text{sram}} = \frac{C_{\text{total}}}{\rho_{\text{sram}}}$, and ρ_{sram} is the memory density.

Dynamic power is calculated per read/write operation:

$$P_{\text{dynamic}} = \frac{1}{T_{\text{task}}} \sum_{i \in \{\text{on,off,link}\}} E_i \cdot N_i, \tag{8}$$

where T_{task} is task duration, N_i is the number of operations, and E_i is the energy in On-die part, Off-die part or transfer link:

$$\begin{cases} E_{\text{on}} &= C_{\text{cap\_on}} \cdot V_{\text{dd}}^2 \cdot \alpha_{\text{sw}}, \\ E_{\text{off}} &= C_{\text{cap\_off}} \cdot V_{\text{dd}}^2 \cdot \alpha_{\text{sw}}, \\ E_{\text{link}} &= C_{\text{cap\_link}} \cdot V_{\text{link}}^2 \cdot L + E_{\text{tsv}} \cdot \frac{D_{\text{block}}}{W}. \end{cases} \tag{9}$$

where $C_{\text{cap\_}*}$ is capacitance, L is average interconnection length, E_{tsv} represents energy consumed by single TSV to transmit 1 bit (pJ/bit).

Cost Modeling. Cost analysis primarily references the original GIA method [9, 10]. It integrated the costs associated with chip fabrication, assembly, IO overhead, and the impact of power:

$$f_{\text{cost}} = \frac{1}{Y_{\text{assembly}}} \left(\sum_{i=1}^{N_c} \text{Co}_{\text{die},i} + \text{Co}_{\text{assembly}} \right) \tag{10}$$

$$Y_{\text{assembly}} = Y_{\text{align}}^{N_c} \times Y_{\text{bond}}^{N_p} \tag{11}$$

where Co_i represents cost, Y_{align} is the yield of alignment process, and Y_{bond} is the yield of the bonding process.

Design Space. In practical design processes T_{tech} is typically predetermined. Thus based on the objective function (i.e, Equation (4)), the design space can be constructed as a 3-dim parameter space:

$$\mathcal{S} = \{ (C_{on}/C_{total}, W, S_{tsv}) \, | \mathbf{x} \in \mathcal{D}, x_5 = t_{\text{tech}} \}. \tag{12}$$

Inequality constraints could be incorporated into the formulation. Then the feasible solution domain is bounded by this hypersurface:

$$\mathcal{F} = \left\{ \mathbf{x} \in \mathcal{S} \,\middle|\, \bigwedge_{i=1}^{M} g_i(\mathbf{x}) \le 0 \right\} \quad \text{s.t.} \quad g_i(\mathbf{x}) \le 0, \quad \forall i = 1, 2, \ldots, M. \tag{13}$$

It is a multi-objective optimization challenge, where objectives are inherently conflicting. Indeed, objective functions (4) exhibit

Algorithm 1 Core Process of Bayesian Optimization

Input: Design space dimensionality D, maximum iteration count $N_{\max}$, convergence threshold ε
Output: Pareto front solution set P

1: // Stage 1: Initial Sampling
2: $(X, Y, C) \leftarrow \text{LatinHypercubeSampling}(N_{\text{init}}, D)$
3: $P \leftarrow \text{FilterParetoFront}(Y, C)$
4: **for** $t \leftarrow 1$ **to** $N_{\max}$ **do**
5: // Stage 2: Surrogate Model Construction
6: $\mathcal{GP} \leftarrow \text{GaussianProcess}(X, Y)$
7: $\rho_{\text{feas}} \leftarrow \text{CalcFeasibility}(C)$
8: // Stage 3: Acquisition Function Optimization
9: $\text{EHVI} \leftarrow \lambda x : \mathbb{E}[\text{HVI}(x) \cdot \rho_{\text{feas}}(x)]$
10: $x_{\text{next}} \leftarrow \arg\max_{x \in \mathcal{X}} \text{EHVI}(x)$
11: // Stage 4: Evaluation and Update
12: $(y_{\text{new}}, c_{\text{new}}) \leftarrow \text{Evaluate}(x_{\text{next}})$
13: $X \leftarrow X \cup \{x_{\text{next}}\},\ Y \leftarrow Y \cup \{y_{\text{new}}\},\ C \leftarrow C \cup \{c_{\text{new}}\}$
14: $P \leftarrow \text{UpdateParetoFront}(P, y_{\text{new}}, c_{\text{new}})$
15: **if** $\Delta\text{HV}(P) < \varepsilon$ **then**
16: **break**
17: **end if**
18: **end for**
19: **return** P

a nonlinear coupling relationship, with their dominance relationship showing non-monotonic characteristics when the gradients of multiple nonlinear functions change drastically. Moreover, the feasible region delineated by inequality constraint parameter spaces can form complex fractal boundaries or discrete connected domains due to the nonlinearity of the constraint function group, resulting in non-convex and discontinuous feasible regions.

Thus, the concept of Pareto optimality becomes particularly crucial. It effectively demonstrates the trade-offs between different objectives by identifying all non-dominated solutions, thereby avoiding the partiality and limitations brought about by single-objective optimization. Therefore, the optimization problem can be formulated as finding the **Pareto frontier** solution set, that is:

$$\mathcal{P} = \left\{ \mathcal{D}^* \in \mathcal{F} \,\middle|\, \nexists \mathcal{D}' \in \mathcal{F},\ f_i(\mathcal{D}') \prec f_i(\mathcal{D}^*), \quad \forall i \in \{\text{latency}, \text{power}, \text{cost}\} \right\} \tag{14}$$

where the relationship $\prec$ is defined as:

$$\mathcal{D}' \prec \mathcal{D}^* \iff \forall i,\ f_i(\mathcal{D}') \le f_i(\mathcal{D}^*) \ \wedge\ \exists j,\ f_j(\mathcal{D}') < f_j(\mathcal{D}^*). \tag{15}$$

4.2 BO-GP Solution

To solve the complex optimization problem mentioned above, as detailed in Algorithm 1, we employ Bayesian Optimization with Gaussian Processes (BO-GP), which has been widely used in chip exploration [13]. The process consists of four stages:

Initialization. The design space $\mathcal{D}$ (with variables in Equation (3)) is sampled via Latin Hypercube Sampling (LHS) [14] to generate N_{init} initial configurations (line 2). Each candidate point is evaluated using physics-based models for objectives (Equation (4)) and constraints (g_1-g_7). Only feasible solutions satisfying all constraints initialize the Pareto front $\mathcal{P}$ (line 3).

Surrogate Modeling. Independent GP models are trained for each objective function ($f_{\text{latency}}, f_{\text{power}}, f_{\text{cost}}$) to capture nonlinear relationships with design variables (line 6). Binary classification GPs predict constraint satisfaction probabilities $P(g_j \le 0)$ for each g_j, with joint feasibility probability $P_{\text{feasible}} = \prod_j P(g_j \le 0)$ computed to define the feasible region (line 7).

Acquisition Function Optimization. The expected hypervolume improvement (EHVI) [15] (line 9) quantifies improvement potential in the hypervolume dominated by $\mathcal{P}$ relative to reference point $\mathbf{f}_{\text{ref}}$. Feasibility probabilities from constraint GPs are integrated into EHVI to prioritize constraint-satisfying regions. The next candidate x_{next} is selected by maximizing EHVI using gradient-based optimization (e.g., L-BFGS-B [16]) (line 10).

Update and Convergence. New point is evaluated via high-fidelity models to obtain y_{new} and c_{new} (line 12). The dataset $\{X, Y, C\}$ is updated, and GP models are retrained. The Pareto front $\mathcal{P}$ is updated with non-dominated solutions using fast non-dominated sorting (line 14). Convergence is declared when the relative hypervolume change over three iterations drops below $\epsilon = 1\%$ (lines 15-17).

4.3 Pruning Optimization

Extended SRAM part requires binding to its corresponding host die to form a complex. However, the original GIA framework [9] treats all chiplets as independent entities, necessitating modification to the selection phase for constraints such as edge-I/O interfaces placement. Additionally, the baseline SA permits arbitrary Chiplet orientations and potentially occupies central regions. This generates redundant solutions and increases iterations without improving quality. Therefore, we conduct further optimization on GIA framework.

Pre-screening. The ILP engine is extended to optimize SRAM instance selection alongside chiplet mapping. A binary decision variable s_k is introduced to indicate whether the k-th SRAM instance is selected. The revised objective function in this sub-stage minimizes a weighted sum of memory cost, power (P), and latency (FT):

$$k_{\text{Mem}} \cdot \sum s_k \text{Cost}(s_k) + k_P \cdot P + k_{FT} \cdot FT \qquad (k_{\text{Mem}} + k_P + k_{FT} = 1). \tag{16}$$

Two critical constraints are added:

(1) Total SRAM area must not exceed the budget, enforced by $\sum s_k \cdot A_k \le A_{\text{budget}}$.
(2) Selected instances must align with TSV interface capabilities, $s_k = 1 \implies \text{Interface}_k \in S_{\text{tsv}}$.

Neighbor Pruning. A pruning strategy is integrated into the simulated annealing process to enforce peripheral placement of SRAM Chiplets. The chip area is divided into peripheral and core regions, with the peripheral zone being defined as a convex polygon along the die edges. During solution perturbation, candidate placements are rejected if any SRAM instance spans the core region or aligns its long axis toward the chip center. Invalid configurations are discarded prior to energy evaluation, reducing computational overhead. For cases where peripheral placement is temporarily infeasible, a penalty term proportional to the SRAM-to-edge distance is added to the energy function to guide optimization. This approach preserves the global search capability of SA while systematically enforcing design-specific geometric constraints.

Table 2: Comparison Across Different Mesh and Router Configuration.

Configuration	2×2@1port		2×4@1port		2×4@2ports		4×4@2ports		8×4@3ports	
	τ_{avg}	B_{max}	τ_{avg}	B_{max}	τ_{avg}	B_{max}	τ_{avg}	B_{max}	τ_{avg}	B_{max}
Baseline	**15.07**	4.71	**20.45**	4.37	**18.01**	5.47	**26.51**	5.18	**36.15**	6.48
+ Thin Crossbar	**13.43**	5.03	**18.97**	4.69	**16.45**	5.92	**25.33**	5.53	**35.12**	6.91
+ Dual Localport	**10.59**	6.48	**14.59**	6.04	**13.87**	7.97	**21.08**	7.55	**31.18**	8.60
+ Address Predictor	**9.48**	7.34	**13.32**	6.98	**12.86**	9.46	**19.76**	8.76	**28.77**	9.53
+ Grouped Address	**9.67**	7.33	**13.18**	6.99	**9.78**	10.65	**15.70**	10.68	**17.12**	11.64

5 Evaluation

We conduct experiments from two perspectives to systematically assess whether the proposed design meets the criteria outlined in the blueprints. For blueprint A, we employed chiplet-level RTL simulation and architectural simulator cross-validation to check basic functionality and characterize critical performance metrics, including access latency, bandwidth, and design reference parameters. Then for blueprint B, system-level evaluation integrates both on-die memory and off-chip chiplet library into the GIA framework [9, 10] to explore architectural scalability and quantify cost-effectiveness through comprehensive design space exploration.

5.1 Experimental Setup

We establish the baseline NoC based on an enhanced version of the open-source Verilog LisNoC [17, 18] implementation. Each SRAM bank node is synthesized using OpenRAM [19], configured as an 8KB single-port SRAM (64-bit width × 1024 wordlines). The design operates at 1.0V supply voltage, with operating frequency autonomously optimized by the OpenRAM toolchain. Circuit-level timing and power characteristics, including read/write latency and dynamic energy per access, are extracted through HSPICE simulations using the generated netlist and freePDK45.

To validate functional correctness and quantify performance improvements, we integrate the extracted circuit metrics into a cycle-accurate network traffic C++ simulator (e.g. [20]). The simulator uniformly injects 64-byte read/write request packets from a centralized master node, with destination addresses randomized across the mesh. These requests are concurrently forwarded to both the baseline RTL implementation and our modified design to ensure bit-exact equivalence, while recording key parameters (e.g., $H_{\text{on}}, T_{\text{task}}, N_i$).

5.2 Chiplet-level Evaluation

Table 2 quantifies the performance enhancements achieved through progressive architectural optimizations across five mesh configurations. The baseline architecture demonstrates latency (τ_{avg}, ns) and bandwidth ($B_{max}, Gbps$) characteristics that scale predictably with network complexity, with τ_{avg} increasing from 15.07 (2×2@1port) to 36.15 (8×4@3ports) as node count grows. The thin crossbar modification reduces τ_{avg} by 10.9%-15.6% across all configurations by simplifying arbitration logic and minimizing wire congestion through streamlined crosspoint allocation. This structural optimization enables corresponding B_{max} improvements of 6.8%-10.9%, demonstrating that interconnect simplification directly enhances both latency and throughput metrics.

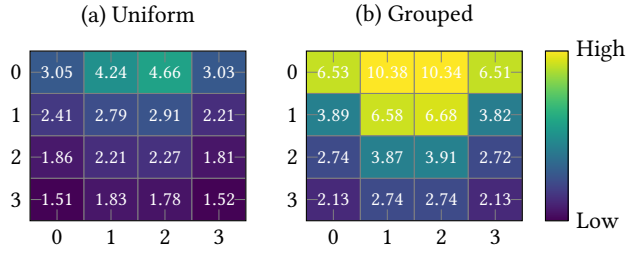

Figure 5: 4×4@2ports with vertical bisection (8 nodes/group) bandwidth (Gbps) heatmap. (1, 0) (2, 0) are IO routers.

The dual local port enhancement introduces parallel memory access paths, effectively decoupling read/write operations and reducing queuing delays. This modification yields a significant 23.5%-31.6% τ_{avg} reduction compared to baseline, with B_{max} improvements reaching 47.9% in the 2×4@2ports configuration. Address prediction further optimizes memory access patterns through speculative prefetching, achieving additional 8.9%-13.3% τ_{avg} reductions in mid-sized meshes by masking access latency. However, its effectiveness diminishes in larger 8×4@3ports configurations (9.53 vs 8.60 B_{max}), revealing limitations in prediction accuracy under high concurrency.

Grouped addressing demonstrates superior scalability through spatial partitioning of memory addresses, which reduces bank conflicts. This optimization delivers a 20.6% τ_{avg} reduction in 8×4@3ports versus the previous version, while boosting B_{max} by 22.1% through optimized request coalescing. Figure 5 clearly exhibits a multi-level characteristic, demonstrating the feasibility of elastic design.

The hierarchical combination of these techniques achieves maximum 62.6% latency reduction and 79.6% bandwidth improvement in the largest configuration, confirming that localized optimizations (crossbar streamlining, dual-port access) synergize effectively with system-level strategies (prediction, address grouping). This validates the architecture's capability to maintain performance scalability across diverse topologies while preserving design modularity.

Power Analysis. Injection rate sweep identifies three operational regimes governing power performance trade-offs (Figure 6). It can be distinguished into linear scaling phase (0-5k requests/10k cycles,R^2 = 0.998), saturation onset (7k-8k requests) and full saturation (>8k requests). The dynamic gating strategy reduces total energy by 8.05% on average exclusively in the linear scaling phase, primarily due to the random traffic generation, necessitating in-depth analysis of workload-specific behaviors.

5.3 System-level Evaluation

On/Off-die Ratio. Figure 7 reveals the distribution of on-die capacity (C_{on}) along the Pareto front in a $C_{\text{total}} = 8MB$ case. 72% of

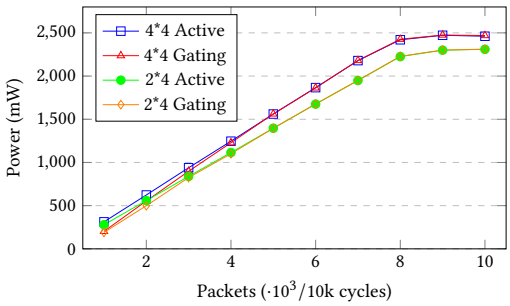

Figure 6: Injection Sweep for Always-active and Gating.

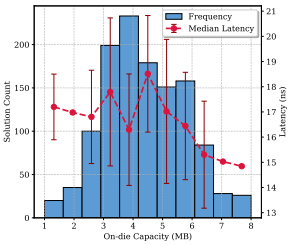

Figure 7: C_{on} Distribution.

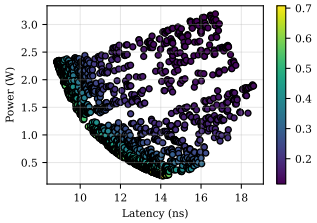

Figure 8: Pareto Frontier.

solutions concentrate within 2.6-6.8 MB, with median $C_{on} = 4.1$ MB. This clustering reflects the balance between diminishing returns in latency reduction and escalating power costs at higher C_{on}. Median-bound analysis in Figure 7 marked as red line quantifies latency reduction versus C_{on} growth. A negative correlation ($r = -0.76$) between latency and C_{on} confirms the theoretical model $f_{latency}$. Figure 8 shows a more global distribution with a clear Pareto front. Non-dominated solutions exhibit latencies from 9.2ns to 14.5ns and power from 0.21W to 2.47W, with normalized costs ranging 0.3-0.6.

Pareto-optimal analysis defines two design regimes. Optimal on-die capacity ratio κ selection criteria:

- $\kappa \approx 0.40$: Cost-constrained IoT/edge devices,
- $\kappa \approx 0.64$: Latency-critical performance systems,
- $\kappa \approx 0.53$: Balanced general computing default.

System Cost. Table 3 shows the cost reductions of design cases used by GIA [10] compared to monolithic solutions. At low volume (500K), NRE-dominated systems (CPUs-1 and CPUs-3) achieve 26.3% and 24.5% cost reductions via ≥ 70% intra-chiplet reuse. In contrast, the GPUs-1 requires decoupled I/O controllers (45% off-chip proportion) due to high bandwidth demands, yielding a lower cost optimization (20.9%) than CPU-centric designs. Nevertheless, it achieves 19.7% improvement over monolithic solutions, underscoring the general applicability of Chiplet-based architectures for heterogeneous systems. At high volume (10M), RE-dominated systems prioritize packaging: CPUs-1 reduces RE costs by 7.8% through 70% intra-chiplet density and bond yield Y_{bond} in Equation (11), while CPUs-4's low complexity limits savings (5.2%). CPUs-2 exhibits 22.1% vs. 6.9% optimization across volumes, demonstrating process-node tradeoffs.

Speedup. Our optimized pruning strategy is evaluated against the original SA for GIA tasks [10]. Results show a 1.93× acceleration in average convergence time with maximum temperature deviation <1.2°C. This improvement stems from 68.7% fewer invalid ILP evaluations and 41.2% enhanced local search efficiency via neighborhood pruning without quality loss.

Table 3: Total On-die Ratio, System Cost (normalized) Reductions compared to Monolithic with Manufacturing Quantities of 500K and 10M and Solve Time(sec) Reduction.

Design	On-die Ratio	Cost Save		Solve Time	
		500K	10M	GIA [10] (s)	Ours (s)
CPUs-1	40%	24.5%	7.8%	481.65	243.26
CPUs-2	35%	22.1%	6.9%	337.27	240.90
CPUs-3	25%	26.3%	8.5%	67.62	32.21
CPUs-4	40%	18.7%	5.2%	52.76	25.73
GPUs-1	55%	20.9%	6.3%	755.87	354.86
Avg.	-	22.5%	5.94%	(Speedup) 1.93×	

6 Conclusion

This paper proposes AuxilarySRAM, a lightweight on-chip memory architecture featuring a latency-optimized lightweight NoC-based Chiplet implementation. We develop analytical quantification models and integrate them into the enhanced GIA framework for system-level evaluation, showing cost efficiency improvement and promising potential for latency-intensive computing applications.

Acknowledgment

This work is supported by the National Key R&D Program of China (2022YFB2901100), the National Natural Science Foundation of China (No. 62404021), and the Beijing Natural Science Foundation (No. 4244107, QY24216, QY24204).

References

[1] Wm A Wulf et al. 1995. Hitting the memory wall: Implications of the obvious. *ACM SIGARCH computer architecture news* 23, 1 (1995), 20–24.
[2] David A Patterson. 2004. Latency lags bandwith. *CACM* 47, 10 (2004), 71–75.
[3] Hongshin Jun et al. 2017. Hbm (high bandwidth memory) dram technology and architecture. In *Proc. IMW*. 1–4.
[4] Miao Liu et al. 2025. Processing-Near-Memory with Chip Level 3D-IC. In *Proc. ASPDAC*. 302–307.
[5] Bon Woong Ku et al. 2016. How much cost reduction justifies the adoption of monolithic 3D ICs at 7nm node?. In *Proc. ICCAD*. 1–7.
[6] Shixin Chen et al. 2025. The Survey of 2.5 D Integrated Architecture: An EDA perspective. In *Proc. ASPDAC*. 285–293.
[7] Dylan Stow et al. 2019. Investigation of cost-optimal network-on-chip for passive and active interposer systems. In *Proc. SLIP*. 1–8.
[8] John Wuu et al. 2022. 3D V-Cache: the Implementation of a Hybrid-Bonded 64MB Stacked Cache for a 7nm x86-64 CPU. In *Proc. ISSCC*, Vol. 65. 428–429.
[9] Fuping Li et al. 2022. GIA: A reusable general interposer architecture for agile chiplet integration. In *Proc. ICCAD*. 1–9.
[10] Fuping Li et al. 2024. Chipletizer: Repartitioning socs for cost-effective chiplet integration. In *Proc. ASPDAC*. 58–64.
[11] Boris Grot et al. 2011. Kilo-NOC: a heterogeneous network-on-chip architecture for scalability and service guarantees. *ACM SIGARCH computer architecture news* 39, 3 (2011), 401–412.
[12] R. Chadha and J. Bhasker. 2012. *An ASIC Low Power Primer: Analysis, Techniques and Specification*. Springer New York.
[13] Chen Bai et al. 2021. BOOM-Explorer: RISC-V BOOM Microarchitecture Design Space Exploration Framework. In *Proc. ICCAD*. 1–9.
[14] Jon C Helton et al. 2003. Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems. *Elsevier RESS* 81, 1 (2003), 23–69.
[15] Kaifeng Yang et al. 2019. Multi-Objective Bayesian Global Optimization using expected hypervolume improvement gradient. *Elsevier SWEVO* 44 (2019), 945–956.
[16] Ciyou Zhu et al. 1997. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM TOMS* 23, 4 (1997), 550–560.
[17] Kazem Cheshmi et al. 2013. Quota setting router architecture for quality of service in GALS NoC. In *Proc. RSP*. 44–50.
[18] Rakotojaona Nambinina et al. 2022. Extension of the lisnoc (network-on-chip) with an axi-based network interface. In *Proc. ICCMC*. 682–686.
[19] Matthew R Guthaus et al. 2016. OpenRAM: An open-source memory compiler. In *Proc. ICCAD*. 1–6.
[20] Luming Wang et al. 2025. Simulation and Exploration for Multi-Chiplet Systems using Open-Source Tools and Heuristic Algorithm. In *Proc. ISEDA*.