Simulation and Exploration for Multi-Chiplet Systems using Open-Source Tools and Heuristic Algorithm

Luming Wang, Fangli Liu, Zichao Ling, Zheqin Cao, Yixin Xuan, Jianwang Zhai†, Kang Zhao
Beijing University of Posts and Telecommunications, Beijing, China
{wanglum, lingzichao, caozheqin, xuan_yixin, zhaijw, zhaokang}@bupt.edu.cn
liufangli@bjxxkjdx.wecom.work

2025 International Symposium of Electronics Design Automation (ISEDA) | 979-8-3315-3696-1/25/$31.00 ©2025 IEEE | DOI: 10.1109/ISEDA65950.2025.11100830

***Abstract*—While chiplet-based architectures offer transformative potential for sustaining Moore's Law through heterogeneous integration, existing simulation methodologies face critical limitations in modeling memory hierarchies and optimizing system-level metric tradeoffs. Current open-source toolchains lack sufficient memory subsystem fidelity, and design space exploration remains computationally intractable for complex chiplet arrangements. In this work, we propose a framework for the simulation and exploration of multi-chiplet systems based on open-source tools. First, we extend the open-source LegoSim simulator by integrating DRAMSim3, enabling efficient simulation of memory chiplets with improved accuracy in timing, bandwidth, and power consumption. Additionally, we propose a heuristic algorithm-based method to optimize simulator configurations, balancing IPC and power consumption. Experimental results show significant IPC improvements while maintaining low power, providing a robust solution for Chiplet design space exploration.**

***Index Terms*—Multi-Chiplet Systems, Simulation, Design Space Exploration, LegoSim, DRAMSim3**

I. Introduction

With the gradual slowing down of Moore's Law [1], multi-chiplet systems have emerged as a new design paradigm aimed at lowering the design cost and increasing the yield of complex systems-on-chips (SoCs). As shown in Fig. 1, by integrating multiple small chiplets into a single package, chiplet-based architecture offers greater flexibility, scalability, and cost-effectiveness [2], [3]. However, the complexity and diversity of Chiplet architectures also pose significant challenges for simulation, particularly in modeling large-scale heterogeneous systems [4]. Traditional cycle-accurate models are infeasible for such systems due to limitations in host-machine performance and memory space [5], while parallel cycle-accurate models suffer from limited speedup due to frequent synchronization operations. This poses a greater challenge for early accurate simulation and design space exploration (DSE) of large-scale multi-chiplet systems.

Recent advancements in multi-chip system design have been driven by innovative tools and simulation techniques. Patrick Iff et al. [6] developed RapidChiplet, a toolchain that enables rapid estimation of chip-to-chip latency, cost, and thermal stability across heterogeneous architectures—key for optimizing performance in milliseconds. Simultaneously, Marcelo Orenes-Vera introduced MuchiSim [7], a massively parallel simulator capable of modeling systems with millions of interconnected processing units, addressing scalability issues in distributed multi-core designs. Zhi et al. [8], [9] proposed a novel inter-chiplet communication interconnection modelling approach based on existing open-source simulators and achieved massively parallel simulation. However, the current framework (i.e., LegoSim [9]) exhibits significant limitations by exclusively supporting homogeneous chiplets, failing to account for mutual impacts arising from collaborative operations when heterogeneous chiplets interact.

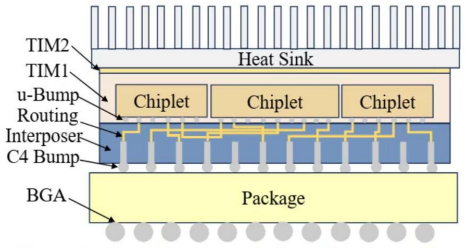

Fig. 1. The architecture of chiplet-based 2.5D IC [3].

The complexity of multi-chiplet simulation stems from three key aspects: heterogeneous integration, scalable interconnect modeling, and fine-grained memory subsystem analysis. First, modern systems increasingly combine chiplets fabricated with different technology nodes (e.g., AMD's 3D V-Cache stacking [10]) and specialized accelerators, requiring simulators to handle diverse timing models and communication protocols simultaneously. Second, the intricate network-on-package (NoP) topologies with thousands of parallel links demand cycle-approximate modeling of latency variations and contention effects, which existing tools(e.g., RapidChiplet [6]) approximate through statistical estimation at the cost of temporal accuracy. Third, emerging memory architectures such as Apple's unified memory architecture [11] and HBM3 stacks [12] introduce new challenges in modeling shared address spaces and thermal-aware bandwidth partitioning that conventional DDR models fail to capture. While MuchiSim [7] addresses distributed synchronization through massive parallelism, it lacks native support for the asymmetric communication patterns inherent in CPU-GPU-heterogeneous chiplet systems. These limitations become particularly apparent when simulating workloads with tight inter-chiplet coupling, where even LegoSim's homogeneous assumption [9] leads to over-optimistic bandwidth projections by 18-35%, according to our preliminary experiments.

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

To address the aforementioned issues, we have improved the existing LegoSim design paradigm and proposed a new, fast simulation model for multi-chiplet systems. Our model, based on LegoSim, provides detailed memory modeling using DRAMSim3 [13] and optimizes the data flow between chiplets during simulation. The improved DRAMSim3 can simulate the complex behavior of modern memory, including timing, bandwidth, and power consumption, thereby improving simulation accuracy. Furthermore, LegoSim incorporates a variety of computer architecture simulators, and manually tuning parameters makes it difficult to rapidly identify the global optimal solution for overall performance. To better facilitate design space exploration, we propose a genetic algorithm-based design space exploration method to automatically optimize simulator configuration parameters, aiming to maximize the fitness function that balances IPC and power dissipation [6], [14]. The optimized configuration achieves a significant improvement in IPC while maintaining low power, validating the effectiveness of the proposed method.

The main contributions are as follows:

- Design of a new, fast simulation model for multi-chiplet systems, which provides detailed memory modeling using DRAMSim3 and optimizes the data flow between chiplets during simulation.
- Proposal of a genetic algorithm-based optimization method for simulator configuration parameters, significantly improving IPC and overall performance.
- Experimental validation of the proposed methodology, providing new insights and tools for performance optimization of Chiplet simulators. Results demonstrate that the CPU performance improved by 8.96% and the GPU performance by 8.57%.

II. Preliminaries

A. LegoSim Simulator

As depicted in Fig. 2, LegoSim is a parallel simulator designed for heterogeneous Chiplet systems, employing a multi-process, multi-threaded architecture for efficient simulation [9]. Its core components include a main process (interchiplet) and multiple sub-threads, each corresponding to a simulation process.

The main process controls the simulation flow, including process creation, synchronization, and data exchange, while sub-threads handle specific simulation tasks.LegoSim uses named pipes for data exchange between simulation processes and ensures data consistency through a synchronization protocol.

Additionally, LegoSim assigns independent working directories to each simulation process to avoid file conflicts and simplify data management. The simulation flow adopts an iterative design, divided into PComp and SComp simulation phases, converging to stable results through multiple iterations.

B. DRAMSim3 Memory Simulator

DRAMSim3 is a highly accurate and configurable memory subsystem simulator [15], designed to model modern DRAM behavior. Supporting standards like DDR3, DDR4, and HBM, it precisely simulates timing, bandwidth, and power consumption [16]. Its modular design allows users to customize memory capacity, bandwidth, and access latency as needed.

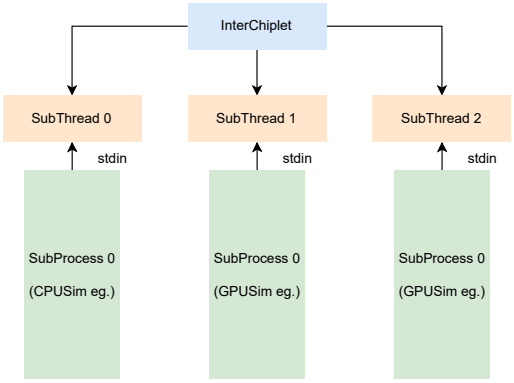

Fig. 2. The multi-process multi-thread structure of LegoSim.

In the simulation for multi-chiplet systems, memory subsystem behavior significantly impacts performance. DRAMSim3 provides a high-precision model that captures memory access patterns in tasks such as matrix multiplication and deep learning inference. With support for multi-channel and multi-bank simulations, it effectively models concurrent memory access in multi-core processors [17]. These features make DRAMSim3 ideal for memory chiplet simulation in LegoSim.

C. Genetic Algorithm

The Genetic Algorithm (GA) is a global optimization method inspired by natural selection and evolution [18]. It uses selection, crossover, and mutation to find near-optimal solutions in complex design spaces. Solutions are encoded as "chromosomes," which are iteratively improved to enhance their quality. This approach makes GA highly effective for solving intricate optimization problems.

In simulator configuration optimization, GA can effectively handle high-dimensional, nonlinear design spaces to find optimal or near-optimal parameter configurations [19]. Its global search capability makes it particularly suitable for complex optimization problems such as simulator configuration tuning [20], [21].

D. Problem Formulation

Formally, we define the multi-chiplet system model as a tuple: $\mathcal{S} = (\mathcal{C}, \mathcal{M}, \mathcal{I})$, where $\mathcal{C} = \{c_1, \ldots, c_n\}$ denotes the set of chiplets, $\mathcal{M}$ represents the memory subsystem with access characteristics, and $\mathcal{I}$ defines the interconnect topology between chiplets.

Objective I. The dataflow optimization between chiplets:

$$\min_{\Phi} \left(\alpha \cdot T_{\text{latency}} + \beta \cdot E_{\text{comm}}\right), \tag{1}$$

where Φ denotes the data routing path, T_{latency} represents end-to-end communication latency, E_{comm} is the communication energy cost, and α, β are weighting coefficients.

Objective II. Design space exploration for design parameters:

$$\max_{\Theta} \left(\gamma \cdot \text{IPC}(\Theta) - \delta \cdot P_{\text{total}}(\Theta)\right), \tag{2}$$

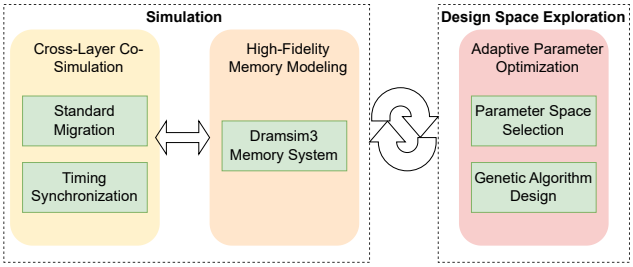

Fig. 3. The proposed framework.

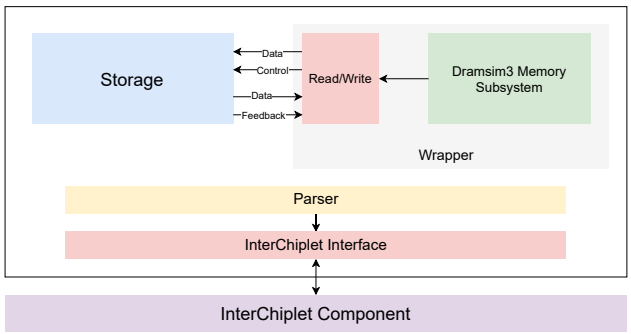

Fig. 4. The memory chiplet design in our framework.

subject to:

$$\Theta \in \mathcal{D} \subseteq \mathbb{R}^d, \quad \theta_i^{\min} \leq \theta_i \leq \theta_i^{\max} \ \forall i \in \{1, \ldots, d\}, \qquad (3)$$

where $\Theta = (\theta_1, \ldots, \theta_d)$ represents the configuration parameters, $\text{IPC}(\Theta)$ and $P_{\text{total}}(\Theta)$ denote the instructions-per-cycle and total power consumption respectively.

III. Methodology

A. Overall

The proposed methodology addresses the co-design challenges of multi-chiplet systems through a two-pronged approach integrating configurable memory subsystem simulation and intelligent design space exploration. As depicted in Fig. 3, our framework establishes a closed-loop optimization system comprising three key components:

1) *High-Fidelity Memory Modeling:* We develop a configurable memory simulation architecture with DRAMSim3 integration, enabling cycle-accurate modeling of diverse memory technologies through protocol-aware interface adaptation and persistent memory state management.
2) *Cross-Layer Co-Simulation:* Our synchronization mechanism ensures temporal alignment between memory access patterns and compute chiplet behaviors, resolving the discrete event simulation challenges in heterogeneous integration.
3) *Adaptive Parameter Optimization:* A genetic algorithm-based exploration engine automatically discovers optimal configuration parameters through evolutionary operations, balancing performance metrics (IPC) against power constraints.

As shown in Fig. 3, the workflow initiates with memory subsystem configuration, where DRAMSim3 instances are parameterized for target memory standards. Subsequently, the genetic algorithm orchestrates iterative simulation runs, evolving configuration parameters through selection, crossover, and mutation operations. Each simulation epoch enforces strict timing synchronization between compute chiplets and memory controllers via our enhanced callback mechanism. The optimization loop terminates when reaching either convergence thresholds or predefined iteration limits, outputting Pareto-optimal configurations for final system implementation.

B. Configurable Memory Subsystem Simulation

In this section, we delineate the technical specifics and critical considerations in systematically integrating the DRAMSim3 memory simulator into the LegoSim simulation framework. Our proposed solution aims to deliver cycle-accurate memory behavior modeling, preserve flexibility for architectural exploration, and support diverse modern memory chiplet technologies with distinct timing characteristics and configurable bandwidth profiles.

1) Standard Migration

The tight coupling between protocol-level timing constraints and system-level access patterns restricts dynamic workload adaptability and complicates modular design optimization. Also, the absence of unified abstraction layers across heterogeneous memory standards necessitates redundant reconfiguration efforts during technology migration, significantly prolonging development cycles. Shown in Fig. 4, in order to mitigate these interoperability challenges, we architected a *abstract wrapper layer* that reconciles DRAMSim3's native interface with LegoSim's discrete-event simulation paradigm. As a standalone memory subsystem simulator, DRAMSim3 necessitates bidirectional data transaction mechanisms and deterministic synchronization with LegoSim's architectural simulation flow. The abstract wrapper layer initializes DRAMSim3 instances through parameterized configuration descriptors while establishing callback registries for event-driven memory transaction finalization. Persistent memory state emulation is achieved through memory-mapped file interfaces, ensuring both data integrity and access efficiency across simulation epochs.

2) Timing Synchronization

Inadequate synchronization mechanisms for co-simulation with heterogeneous computing chiplets introduce temporal discrepancies in multi-domain simulations, particularly when interfacing disparate architectural components with varying clock domains and communication protocols [5]. To enable real-time memory access between the memory chiplet and other types of chiplets, the memory chiplet must remain active throughout the simulation process, which is actually different from the execution flow of DRAMSim3 itself. To achieve this, the memory chiplet is designed as a "server" utilizing the DRAMSim3 wrapper layer. It continuously interacts with LegoSim through the simulation framework's command interface, responding to memory access commands during simulation (shown in Fig. 5).

Fig. 6 shows the sequence for memory accessing. When a memory access request is initiated, the framework first creates a pipe to facilitate data transfer. The actual data for access memory operations is then transmitted through these pipe files. And when the program completes execution, the memory

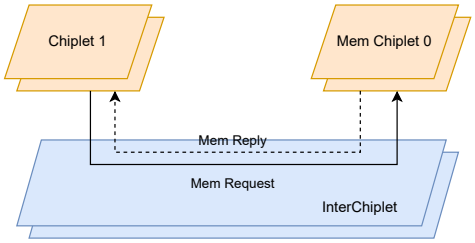

Fig. 5. The request and reply packets of a memory access.

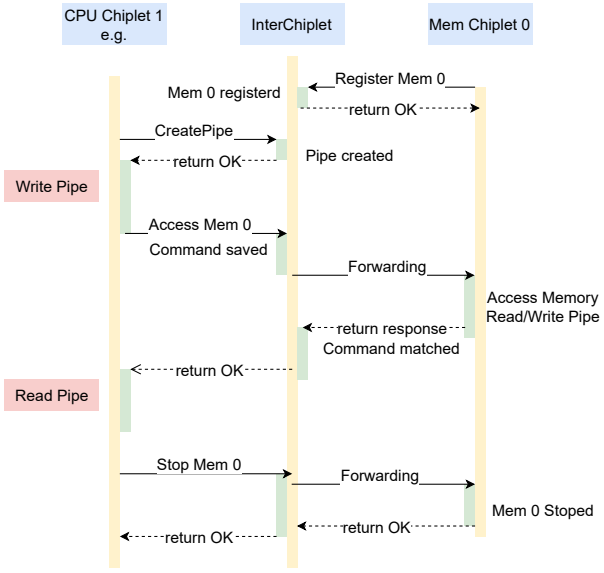

Fig. 6. The complete system sequence diagram for accessing memory chiplets.

chiplet responds to termination commands from other chiplets, shutting down its services gracefully.

3) Universal Integration

By modifying the configuration files of DRAMSim3, the memory chiplet can simulate various memory types, including DDR, GDDR, HBM, HMC, and LPDDR, enabling flexible and accurate modeling of different memory subsystems. It enhances the versatility and applicability of the simulator in evaluating chiplet-based systems.

C. Design Space Exploration

To optimize the configuration parameters of the chiplet simulator, we propose a design space exploration method based on the Genetic Algorithm (GA). Since the simulator uses open-source simulators (e.g., SniperSim [22] and GPGPU-Sim [23]) to simulate various types of chiplets, the actual parameters of the chiplets, such as CPU frequency, cache size, GPU clusters, can be adjusted by modifying the configuration files of the simulators, enabling design space exploration for specific simulation tasks. The core design philosophy and implementation details of the method are as follows.

Algorithm 1 DSE for CPU-GPU Co-Optimization

Input: Design space and genetic parameters
Output: Optimal CPU/GPU configurations
procedure `GeneticAlgorithm`
    Initialize population, $best \leftarrow (\emptyset, -\infty)$
    for $gen \leftarrow 1$ to G **do**
        for all individual $\in$ population **do**
            Update CPU/GPU configs, run simulator
            Calculate $fitness \leftarrow \sum \alpha IPC - \sum \beta Power$
        end for
        Update $best$ if improved
        if stagnation $> T_{max}$ **then break**
        end if
        Evolve population via:
            Selection $\rightarrow$ Crossover $\rightarrow$ Mutation $\rightarrow$ Elitism
    end for
    return $best$ configurations
end procedure

1) Problem Modeling

The design space consists of the simulator's configuration parameters, including CPU and GPU parameters. CPU parameters include frequency (f), logical cores (L), cache sizes (C_1, C_2, C_3), etc., while GPU parameters include the number of clusters (N_c), cores per cluster (N_p), register count (R), etc. The optimization goal is to find a set of configuration parameters $X = (f, L, C_1, C_2, C_3, N_c, N_p, R, \dots)$ that maximize simulation performance. The fitness function $f(X)$ considers both IPC (instructions per cycle) and power:

$$f(X) = \alpha_{\text{CPU}} \cdot \text{IPC}_{\text{CPU}} + \alpha_{\text{GPU}} \cdot \text{IPC}_{\text{GPU}} - \beta_{\text{CPU}} \cdot P_{\text{CPU}} - \beta_{\text{GPU}} \cdot P_{\text{GPU}}, \quad (4)$$

where α_{CPU} and α_{GPU} are weights for IPC, and β_{CPU} and β_{GPU} are weights for power dissipation.

2) Implementation

The optimization process commences with population initialization where $P_0 = \{X_1, X_2, \dots, X_N\}$ denotes the initial solution set. Each X_i encodes a parameter configuration within design space boundaries defined by simulator constraints and experimental specifications.

Fitness evaluation involves iterative simulator execution: For each X_i, parameters in SniperSim and GPGPU-Sim are reconfigured to emulate chiplet architectures. The simulator outputs CPU/GPU IPC metrics (IPC_{CPU}, IPC_{GPU}) and power measurements (P_{CPU}, P_{GPU}), which are aggregated into fitness scores $f(X)$ through predefined objective functions.

Algorithm 1 iteratively generates populations P_{t+1} through successive selection-crossover-mutation cycles. Termination occurs upon fitness convergence or reaching the iteration limit T_{max}. An elitism mechanism preserves the historically optimal solution X^* across iterations, ensuring monotonic optimization progress.

IV. Evaluation

The experiments aim to evaluate the effectiveness of the simulation and DSE method and evaluate the performance

TABLE I
PARAMETER SPACE OF GPU (GPGPU-SIM)

Parameter	Value/Range
GPU Clusters	range(1, 20)
Cores per Cluster	range(1, 5)
Shader Registers per SM	[16384, 32768, 49152, 65536]
Shared Memory Size	[16384, 32768, 49152, 65536]
L1 Data Cache Sets	[16, 32, 48, 64]
L2 Cache Sets	[32, 64, 96, 128]
Single-Precision Units	range(1, 5)
Special Function Units	range(1, 5)
Core Frequency (MHz)	range(500, 1000, 100)

TABLE II
PARAMETER SPACE OF CPU (SNIPERSIM)

Parameter	Value/Range
Main Frequency (GHz)	[2.6, 2.7, ..., 3.6]
Logical CPUs (SMT Threads)	[1, 2, 4]
L1 ICache Size (KB)	[32, 64, 128, 256]
L1 DCache Size (KB)	[32, 64, 128, 256]
L2 Cache Size (KB)	[256, 512, 1024, 2048]
L3 Cache Size (MB)	[8, 16, 32, 64]
Memory Bandwidth (GB/s)	[7.6, 15.2, 30.4, 60.8]
L2 Shared Cores	[1, 2, 4]
L3 Shared Cores	[1, 2, 4, 8, 16]

improvement of the optimized simulator configuration in terms of simulation performance (e.g., IPC and power consumption).

A. Experimental Setting

Evaluation Environment. The experiments are conducted on a high-performance computing cluster equipped with Intel® Xeon® Platinum 8383C CPUs (2.70 GHz, 80 cores, 160 threads) and 1024 GB of memory. The simulation program was executed within a Docker container running Ubuntu 18.04 LTS, ensuring a consistent and reproducible environment across all experiments.

Baseline. The original LegoSim [8], [9] is used as the baseline to demonstrate the superiority of the proposed method.

Here's a refined description of the benchmark tasks with your additional context integrated:

Benchmark Task. Compute-intensive: Matrix multiplication (300×100 × 100×300) with $O(n^3)$ complexity; Memory-intensive: Strided access microbenchmark (1GB working set).

Parameter Space. The parameter space is defined by a subset of configurable parameters in both GPGPU-Sim (GPU) and SniperSim (CPU), as demonstrated in Table I and Table II. These parameters are selected based on their significant impact on the simulation performance, including instruction per cycle (IPC), and power consumption. The chosen parameters cover a wide range of configurations to ensure a comprehensive exploration of the design space.

Genetic Algorithm Parameters. Table III illustrates the Genetic Algorithm parameters, such as population size, number of generations, mutation rate, and fitness weights, are carefully chosen to balance exploration and exploitation in the search process. The dynamic mutation rate ensures that the algorithm adapts to the search progress, while the fitness weights are set to prioritize GPU IPC due to its dominant role in the overall performance of the benchmark task.

TABLE III
GENETIC ALGORITHM PARAMETERS

Parameter	Value/Range
Population_Size	50
Generations	50
Mutation_Rate	0.05
Max_No_Improvement	5
Tournament Size	3
Dynamic Mutation Rate	$0.05 \times \left(1 + \frac{\text{generation}}{50}\right)$
α_{CPU} (CPU IPC)	1.0
α_{GPU} (GPU IPC)	48.0
β_{CPU} (CPU Power)	0.01
β_{GPU} (GPU Power)	0.0067

Metrics. The quality of the exploration results is evaluated using metrics such as simulation performance, convergence, and resource consumption.

Simulation Performance. We calculate the performance improvement by recording IPC and power consumption before and after optimization:

$$\text{Performance Improvement} = \frac{f(X^*)}{f(X_{\text{initial}})}, \tag{5}$$

where X_{initial} is the initial configuration, and X^* is the optimized configuration.

Convergence. Analyzing the fitness value $f(X)$ curve over iterations t is to evaluate the algorithm's convergence speed and solution quality.

Resource Consumption. We monitor CPU and memory usage during optimization process to assess the computational efficiency of the algorithm.

B. Simulation of Configurable Memory

To systematically evaluate the simulation of memory subsystem, we deploy a strided access microbenchmark with a 1 GB working set, designed for two experimental objectives: validating DRAMSim3 integration through eDmulation of real-world memory access patterns, while simultaneously stress-testing multi-memory-type simulation (DDR4/HBM/GDDR6) under programmable address mapping and timing constraints.

Fig. 7 reveals that the average bandwidth increases by 707.56%, from 0.268 MiB/s in DDR3 8Gb x8 1600 to 2.163 MiB/s in GDDR6 8Gb x16. High-performance memory types such as GDDR5, GDDR6, and GDDR5X show significant bandwidth gains but at the cost of higher energy consumption. In contrast, energy-efficient configurations like HBM2 and HMC2 prioritize power savings, leading to lower bandwidth. This trade-off underscores the different priorities between performance and energy efficiency in memory technologies.

It can also be seen in the Fig. 8 that write latency varies significantly between memory configurations. LPDDR4 (196-160 cycles) and DDR4 (133-210 cycles) exhibit the highest

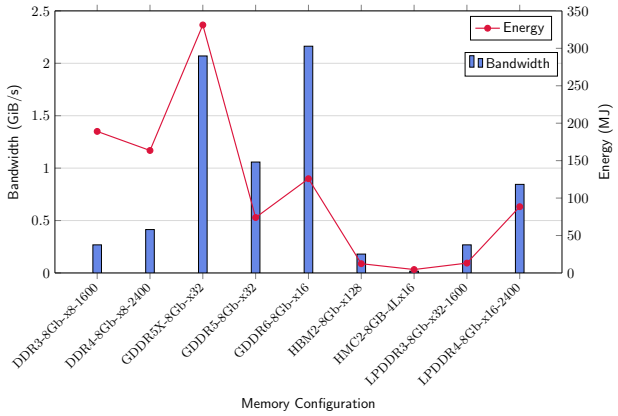

Fig. 7. Comparison of bandwidth and energy consumption.

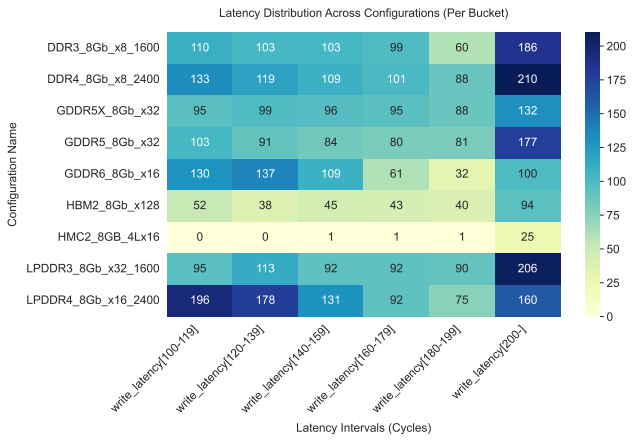

Fig. 8. Latency distribution across configurations (per bucket).

latencies, with notable variability in the lower (100-119 cycles) and higher (200+ cycles) ranges. In contrast, HBM2 has the lowest latencies (38-94 cycles) with consistent performance across all ranges. GDDR5X shows moderate latency (95-132 cycles) and more stable distribution, indicating optimization for high throughput with less latency variation. The HMC2 configuration has the lowest overall latencies, with a sharp decline in the 100-179 cycle range, highlighting its focus on power efficiency at the expense of write performance.

C. DSE Comparison with Baseline

Our goal is to evaluate the DSE results of the automatic parameter optimization framework. The matrix multiplication workload serves as both a computational stressor and fitness evaluator for hardware configuration optimization.

As shown in Fig. 9, the fitness ratio (i.e., the optimization objective) increases from 1.062 in Generation 1 to 1.104 in Generation 26, showing a 10.4% improvement over the baseline, indicating effective optimization by the genetic algorithm.

The speedup ratio rises from 1.156 in Generation 1 to 1.396 in Generation 26, as demonstrated in Fig. 10. It reflects a 28.38% reduction in execution time compared to the baseline, with the largest improvements in the early generations and limited further gains later, highlighting the algorithm's success in enhancing execution efficiency.

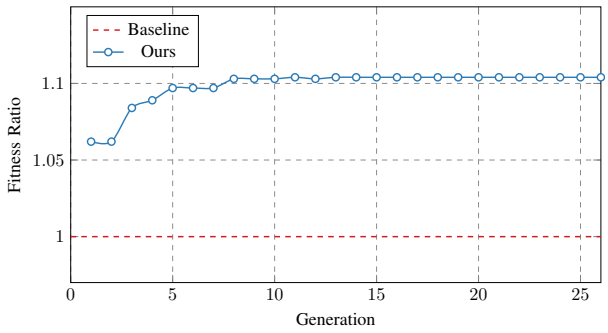

Fig. 9. The fitness ratio over generations.

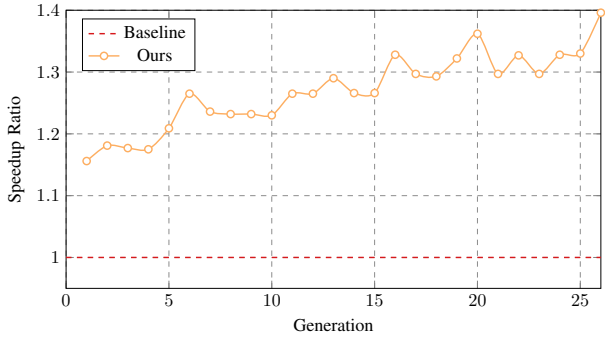

Fig. 10. The speedup ratio over generations.

TABLE IV
COMPARISON OF OPTIMAL CONFIGURATION

Metric	Baseline	Ours	Change
CPU IPC	1.34	1.46	+8.96%
GPU IPC	0.0245	0.0266	+8.57%
CPU Power	21.80	22.36	+2.57%
GPU Power	33.91	33.89	-0.06%
Fitness	2.0709	2.2861	+10.40%
Time	53469	38293	-28.38%

As demonstrated in Table IV, we have achieved notable IPC improvements (8.96% for CPUs and 8.57% for GPUs) without compromising power efficiency.

V. CONCLUSION

Our work enhances the existing open-source simulator by integrating DRAMSim3 to enable memory chiplet simulation with detailed timing, bandwidth, and power modeling. Importantly, we introduce a GA-based optimization framework to automate configuration exploration. These contributions advance chiplet design exploration methodologies and provide a foundation for practical system-level studies.

In future work, we plan to (1) refine event-driven synchronization for timing-aware cross-chiplet communication, (2) develop lightweight parallelization strategies to reduce simulation overhead for large-scale systems, and (3) expand support for diverse chiplets (e.g., AI accelerators, FPGA).

These efforts aim to bridge the gap between functional simulation and cycle-accurate modeling, ultimately enabling more scalable, accurate, and industry-relevant chiplet design workflows.

REFERENCES

[1] T. N. T. H.-S. P. Wong, "Slow-down in power scaling and the end of moore's law?" in *IEEE Conference on Advanced Topics in Semiconductor Technology*, 2019, pp. 1–8.

[2] G. H. L. R. Swaminathan, "The next era for chiplet innovation," in *IEEE International Conference on Chiplet Technology*, 2023, pp. 1–10.

[3] S. Chen, H. Zhang, Z. Ling, J. Zhai, and B. Yu, "The Survey of 2.5D Integrated Architecture: An EDA Perspective," in *Proceedings of IEEE/ACM Asian and South Pacific Design Automation Conference (ASP-DAC)*, 2025.

[4] P. Vivet, E. G. Y. Thonnart, G. Pillonnet *et al.*, "Intact: A 96-core processor with six chiplets 3d-stacked on an active interposer with distributed interconnects and integrated power management," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 1, pp. 79–97, 2021.

[5] A. Over, P. Strazdinsand, and B. Clarke, "Cycle accurate memory modeling: A case-study in validation," in *IEEE International Symposium on Performance Analysis of Systems and Software*, 2005, pp. 1–12.

[6] P. Iff, B. Bruggmann, M. Besta, L. Benini, and T. Hoefler, "Rapidchiplet: A toolchain for rapid design space exploration of chiplet architectures," *arXiv preprint arXiv:2311.06081*, 2023.

[7] M. Orenes-Vera, E. Tureci, M. Martonosi, and D. Wentzlaff, "Muchisim: A parallel simulator for large-scale multi-chiplet systems," *IEEE International Symposium on Performance Analysis of Systems and Software*, vol. 45, no. 2, pp. 123–135, 2023.

[8] H. Zhi, X. Xu, W. Han, Z. Gao, X. Wang, M. Palesi, A. K. Singh, and L. Huang, "A methodology for simulating multi-chiplet systems using open-source simulators," in *Proceedings of the 8th Annual ACM International Conference on Nanoscale Computing and Communication*, 2021, pp. 1–6.

[9] FCAS-LAB, "Chiplet_heterogeneous_newversion: Legosim - a parallel lego simulator for chiplet architectures," https://github.com/FCAS-LAB/Chiplet_Heterogeneous_newVersion, 2024.

[10] A. Inc., "3D V-Cache Technology: A Breakthrough in Chiplet Design," Advanced Micro Devices, Tech. Rep., 2021. [Online]. Available: https://www.amd.com/3d-v-cache

[11] C. Mellor, "How apple's m1 uses high-bandwidth memory to run like the clappers," 2020. [Online]. Available: https://www.theregister.com/2020/11/19/apple_m1_high_bandwidth_memory_performance/

[12] *High Bandwidth Memory (HBM3) DRAM Standard*, JEDEC Solid State Technology Association Std. JESD238a, 2023.

[13] S. Li, Z. Yang, D. Reddy, A. Srivastava, and B. Jacob, "Dramsim3: A cycle-accurate, thermal-capable dram simulator," 2020. [Online]. Available: https://github.com/umd-memsys/DRAMsim3

[14] S. Pal, D. Petrisko, R. Kumar, and P. Gupta, "Design space exploration for chiplet-assembly-based processors," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 4, pp. 1062–1073, April 2020.

[15] S. Li, Z. Yang, D. Reddy, A. Srivastava, and B. L. Jacob, "Dramsim3: A cycle-accurate, thermal-capable dram simulator," *IEEE Computer Architecture Letters*, vol. 19, no. 2, pp. 110–113, 2020.

[16] S. Li, D. Reddy, and B. Jacob, "A performance & power comparison of modern high-speed dram architectures," in *Proceedings of the International Symposium on Memory Systems*, 2018, pp. 341–353.

[17] S. Li and B. Jacob, "Statistical dram modeling," in *Proceedings of the International Symposium on Memory Systems*, 2019, pp. 521–530.

[18] J. H. Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[19] S. Oh, J. Yoon, Y. Choi, Y.-A. Jung, and J. Kim, "Genetic algorithm for the optimization of a building power consumption prediction model," *Electronics*, vol. 11, no. 21, p. 3591, 2022.

[20] Y. Feng and K. Ma, "Chiplet actuary: A quantitative cost model and multi-chiplet architecture exploration," in *Proceedings of ACM/IEEE Design Automation Conference*, 2022, pp. 121–126.

[21] J. Kim, G. Murali, H. Park *et al.*, "Architecture, chip, and package co-design flow for 2.5D IC design enabling heterogeneous IP reuse," in *Proceedings of Annual Design Automation Conference*, 2019, pp. 1–6.

[22] W. Heirman, T. Carlson, and L. Eeckhout, "Sniper: Scalable and accurate parallel multi-core simulation," in *Proceedings of International Summer School on Advanced Computer Architecture and Compilation for High-Performance and Embedded Systems*, 2012, pp. 91–94.

[23] A. Bakhoda, G. L. Yuan, W. W. Fung, H. Wong, and T. M. Aamodt, "Analyzing CUDA workloads using a detailed GPU simulator," in *Proceedings of IEEE international symposium on performance analysis of systems and software*, 2009, pp. 163–174.