

TSIM4ICS: Trace-Driven SystemC-TLM Simulation Framework for I/O Die-Based Multi-Chiplet Systems

Youngechul Yoon

Dept. of Computer Science and Engineering
Seoul National University
Seoul, South Korea
yyc704@snu.ac.kr

Soonhoi Ha

Dept. of Computer Science and Engineering
Seoul National University
Seoul, South Korea
sha@snu.ac.kr

Abstract—The growing demand for high-performance, energy-efficient, and heterogeneous computing has spurred research on chiplet architectures, which enable modular, scalable, and cost-effective system designs. While most prior studies focus on distributed Network-on-Package(NoP)-based chiplet architectures, this paper addresses multi-chiplet systems that employ an I/O die as a communication hub. The I/O die integrates and standardizes inter-chiplet and external interfaces, thereby enhancing modularity, scalability, multi-vendor integration, and performance-power optimization. We present TSIM4ICS, a trace-driven SystemC-TLM simulator for multi-chiplet systems that estimates end-to-end application performance. Each chiplet model generates communication traces to the I/O die, capturing die-to-die(D2D) links and DDR-interface latency/bandwidth to reveal the impact of remote accesses. Using a multi-NPU chiplet model running partitioned CNN workloads, our simulator allows exploration of the design space across the number of NPUs, chiplets, and workload partitioning, supporting HW/SW co-design. TSIM4ICS is publicly available online¹ to promote reproducible and practical evaluation of chiplet systems.

Index Terms—Trace-driven, I/O die, SystemC-TLM, Chiplet

I. INTRODUCTION

To sustain Moore’s Law, the industry has shifted from monolithic system-on-chip architectures to chiplet-based architectures, which provide a cost-efficient path to achieving higher performance. However, chiplet systems inevitably incur greater die-to-die (D2D) communication overhead compared to intra-die communication, and this overhead must be carefully managed to realize their performance potential. As a result, there is a growing need for simulators that can accurately capture chiplet system behavior.

Chiplet technology is still in its early stages, and several architectural approaches have been explored. Broadly, these can be divided into two categories: one treats chiplets as nodes connected in a Network-on-Chip(NoC)-like topology [1]–[5], while the other introduces an I/O die that serves as a communication hub [6], [7]. Although most simulation frameworks to date focus on NoC-like interconnects, support for I/O die-based architectures remains limited.

Moreover, because architectural design is inherently exploratory, fast simulation is essential. Existing works [1], [2], [5], [8] have primarily relied on gem5 [9], BookSim2 [10], or other cycle-accurate simulators [1], [11]. While accurate,

these are often too slow to run full applications and are thus restricted to synthetic traffic or small benchmarks. On the other hand, analytical models can run real applications but fail to capture the details of chiplet communication, reducing the reliability. What is lacking is a simulation framework that balances architectural fidelity with sufficient speed to enable meaningful design-space exploration.

In this paper, we address this gap by proposing a novel simulation framework that:

- supports multi-chiplet systems with an I/O die, enabling evaluation of hub-based communication structures;
- employs a trace-driven approach to accelerate exploration while maintaining structural fidelity; and
- provides chiplet-level heterogeneity and flexibility, making it well-suited for design-space exploration of emerging chiplet systems.

In our simulation framework, TSIM4ICS(Trace-driven SIMulator for I/O die-based multi-Chiplet Systems), each chiplet model generates communication traces that are fed into the I/O die, capturing D2D link behavior and DDR-interface latency/bandwidth to expose the performance impact of remote accesses. The overall communication architecture is simulated on a SystemC [12]-TLM [13] backplane, driven by the input traces from each chiplet.

As a case study, we employ a multi-NPU chiplet model running partitioned CNN workloads. This setup enables exploration of the design space across the number of NPUs per chiplet, the number of chiplets, and workload partitioning strategies, thereby supporting HW/SW co-design. Through this capability, our simulator provides a flexible and efficient platform for evaluating emerging multi-chiplet architectures.

II. BACKGROUND

A. Chiplet Network-on-Package Architectures

1) *Distributed Network-on-Package(NoP)*: As illustrated in Fig. 1(a), distributed NoP refers to an architecture where processor chiplets are interconnected in a NoC-like topology, with interfaces directly established between adjacent chiplets. This approach offers strong scalability and is well-suited for workloads with intensive inter-chiplet communication. However, since no centralized communication hub exists, each communication path requires its own dedicated interface, which

¹<https://github.com/cap-lab/TSIM4ICS>

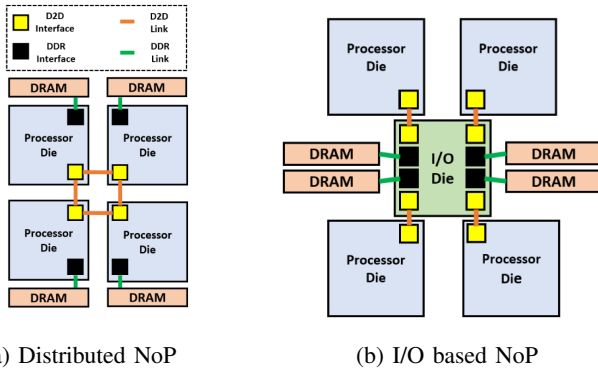


Fig. 1: Distributed NoP vs I/O based NoP

can increase the design complexity of each chiplet. Many existing simulation studies have adopted this structure [1], [2].

2) *I/O die-based NoP*: First introduced in AMD’s second-generation Rome architecture [6], the I/O die-based design connects multiple chiplets through a centralized communication hub, as shown in Fig 1(b). The I/O die not only manages communication among processor chiplets but also interfaces with memory. In later generations [7], its role has expanded to include inter-system connectivity, though that is beyond the scope of this paper.

The I/O die offers several advantages:

- It allows process decoupling from the processor chiplets, enabling the use of different, more efficient technologies tailored to specific functions.
- It simplifies the design of processor chiplets by offloading I/O complexity.
- By consolidating I/O capabilities, it supports large-capacity memory access.

This I/O die-based architecture serves as the primary focus of the simulations presented in this paper.

B. Trace-Driven Simulation Framework

The proposed trace-driven simulation framework builds on the technique introduced in the HSIM framework [14], originally developed for system-level simulation of embedded systems. HSIM consists of two main parts: component simulators and a simulation backplane. Each component simulator independently models the internal behavior of a component and generates memory access traces to the communication architecture. Since local clocks of component simulators are not synchronized with the global clock, HSIM enables their parallel execution. The backplane, implemented in SystemC-TLM, performs timing simulation of the communication architecture, while a wrapper module bridges each component simulator with the backplane by converting local time into global time and ensuring synchronization.

In our framework, each processing element is modeled by a component simulator, which can alternatively be replaced with a trace generator when its memory access trace is available in advance. As shown in Fig. 2, the simulation backplane models

both the intra-chiplet communication architecture and the I/O die-based NoP architecture across chiplets, while also capturing the memory hierarchy.

III. RELATED WORK

Among studies that simulate chiplet-based system architectures, research on distributed NoP simulation has been particularly active. A representative example is cycle accurate simulator CNSim [1], which models distributed NoP structures. The applications used in this work include synthetic traffic and PARSEC benchmarks. However, due to limited consideration of DRAM behavior, it cannot be regarded as a full-system simulation.

At the full-system simulation level, two representative works stand out. In one study [2], gem5’s Ruby memory system [9] was integrated with SystemC, enabling simulation at the Transaction-Level Modeling (TLM) abstraction. The workload used was the STREAM Triad memory benchmark, but the model did not account for an I/O die. Another study [5] used BZSim [15], a combination of BookSim2 [10] and ZSim [11] as the network simulator, and integrated DRAMSim3 [16] and CACTI [17] to build a full-system simulator for performance evaluation of chiplet architectures. This work assumed that each chiplet is paired with a dedicated HBM and chiplets are directly interconnected. Although the I/O die is mentioned, it only serves as a direct connection to DRAM, not as a true communication hub. The experiments were conducted using standard benchmarks, and due to the cycle-accurate nature of the simulator, each run required execution times on the scale of tens of hours.

There are also studies that attempt to consider I/O die-based architectures. One such example is M2M [18], which employs real workloads such as CNNs and Transformers. However it relies on an analytical performance prediction model rather than detailed simulation, and therefore cannot provide a faithful evaluation of the impact of the I/O die on performance. To the best of our knowledge, no existing work has yet provided a simulator that both evaluates the performance implications of I/O die-based architectures and achieves fast simulation speed with sufficient flexibility. This motivates the development of our new simulation framework.

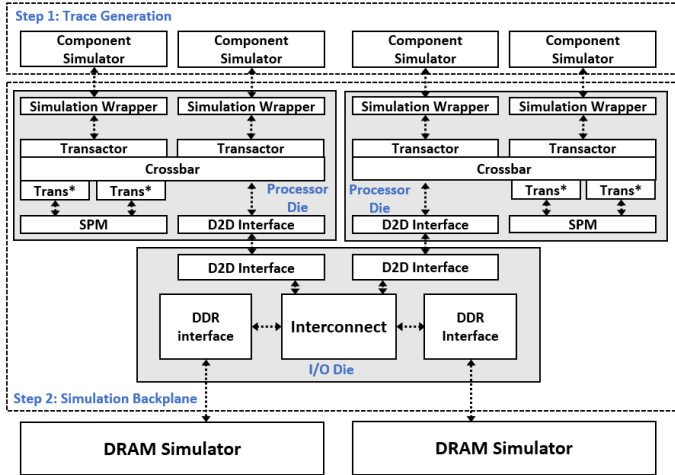
Table 1 positions our simulator: it supports full-system, cycle-approximate simulation of I/O die based NoP chiplet architectures, executes real applications, and being faster than prior simulators enables practical design space exploration (DSE).

IV. PROPOSED SIMULATION FRAMEWORK

Figure 2 illustrates the overall structure of TSIM4ICS for an example multi-chiplet system, which consists of two stages: trace generation and the simulation backplane. In the trace generation stage, a component simulator provides memory traces that are consumed by the SystemC-TLM backplane. Any simulator capable of generating the compatible trace format can be used. In our experiments, we employ an NPU simulator [19] together with its compiler. Each memory access event in the trace includes the following fields: access type (read or write),

TABLE I: Comparison of chiplet-based system simulation frameworks

Approach	Full system simulation	System level	Chiplet interconnect modeling	Speed	Workload
CNSim [1]	×	Cycle-accurate	Distributed NoP	N/A	Traffic, PARSEC
SystemC+gem5 [2]	✓	Cycle-accurate	Distributed NoP	Affordable	STREAM Triad Benchmark
BZSim+DRAMSim3 [5]	✓	Cycle-accurate	Distributed NoP	+10 hours	SPEC CPU 2017, GraphBIG, GUPS suite, XSBench
M2M [18]	✓	Analytical model	Distributed NoP I/O die based NoP	N/A	Multi-DNN models
TSIM4ICS(ours)	✓	Cycle-approximate	I/O die based NoP	Less than hour	MobileNetV2, DiscoGAN, YOLOv7-tiny, ResNet50



Note: * Transactor

Fig. 2: Overall structure of TSIM4ICS for an example multi-chiplet system

address, data, data size, and delta time. The delta time indicates the interval since the previous memory access event, measured in the local clock domain. Using this information, the wrapper module in the simulation backplane advances the component’s global clock, synchronizing it with the rest of the system.

The simulation backplane instantiates the communication architecture, bridging the processor component simulators and the DRAM simulator. The framework supports both DRAM-Sim3 [16] and Ramulator [20] as DRAM backends, while all evaluations in this paper are conducted using Ramulator.

Our simulation backplane is built on SystemC and TLM 2.0 to accelerate simulation. Traditional Transaction Level Modeling (TLM) transfers data only at the transaction level, making it faster than cycle-accurate approaches but limiting its ability to capture fine-grained communication bottlenecks. To overcome this limitation, we augment TLM with cycle awareness and develop a cycle-approximate communication model, as illustrated in Fig. 3. This model preserves the abstraction level of TLM while enabling realistic modeling of hardware communication bottlenecks.

Fig. 3 shows three communication types : Core \leftrightarrow SPM, Core \leftrightarrow DRAM, and SPM \leftrightarrow DRAM. In the example multi-chiplet system, intra-chiplet traffic is assumed to follow the AXI protocol [21]. Except for Core \leftrightarrow SPM, which remains on-die, the other paths traverse heavier external (off-die/D2D)

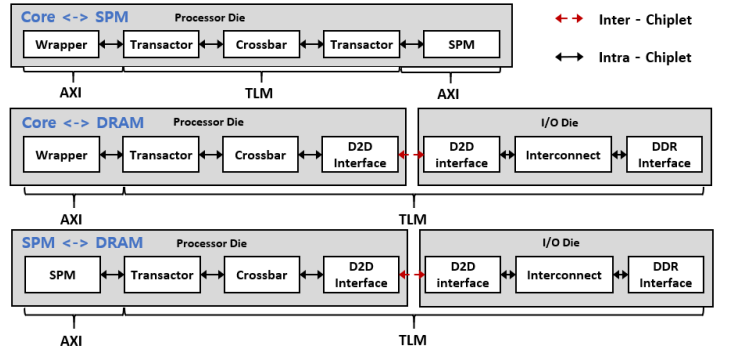


Fig. 3: Communication modeling.

communication. On the memory side, a DDR interface is provided to synchronize with the memory simulator. The following describes the role of each module in the simulation backplane.

Wrapper: Wrapper is the interface that connects the component simulator to the simulation backplane, translating memory traces into simulation packets. The wrapper runs at the same clock frequency as the component simulator and uses the provided time information to synchronize local time with the global simulation time. In addition, it segments each simulation packet to match the configured crossbar width and dispatches the segments cycle-accurately across the interconnect.

Transactor: The transactor serves as a bridge between the cycle-accurate domain and the TLM domain. It receives simulation packets from cycle-accurate modules (e.g., Wrapper, SPM), emulates crossbar channel occupancy, and delivers the corresponding TLM simulation packets at the appropriate timing. Conversely, it also performs the reverse operation by converting TLM simulation packets back into the cycle-accurate domain.

Crossbar: Crossbar is the intra-chiplet communication hub that connects the wrapper, SPM banks, and the D2D interface. It provides one-to-one connections between the Wrapper and each SPM bank inside the chiplet.

SPM: Scratchpad Memory (SPM) is a software-controlled shared memory within the processor chiplet. It is used for communication between processing components inside a chiplet and also serves as a prefetch buffer to hide DRAM access latency. In our framework, we assume a hidden controller that handles SPM access traces generated by a multi-NPU compiler. Depending on the design, SPM can be modeled as the last-level cache in the memory hierarchy.

D2D Interface : Die-to-die (D2D) interface is the communi-

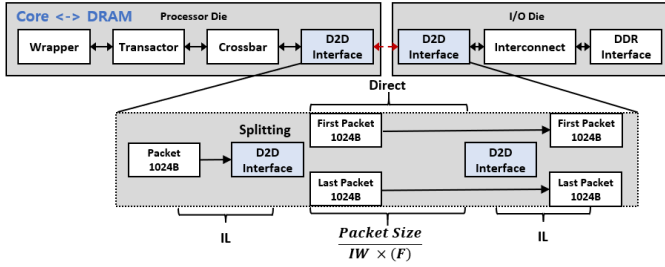


Fig. 4: Splitting simulation packet handling inter-chiplet latency.

cation link between chiplets. Because it bridges the intra-chiplet and inter-chiplet networks, it plays a critical role for performance evaluation in chiplet systems. This module exposes three tunable parameters:

- Frequency (F) : Primary parameter controlling bandwidth
- Interface Width (IW) : Amount of data transferred per cycle (bits/cycle)
- Interface Latency (IL) : Additional delay to account for protocol conversion and boundary-crossing overheads. (ns)

Naive TLM modeling, which represents packet communication as a single transaction, fails to accurately capture interconnect and channel bottlenecks. In particular, when a transaction carrying data passes through a D2D interface and then re-enters an intra-chiplet channel, the latency introduced by the D2D link requires the channel to be occupied for a longer duration than in the case of intra-chiplet transfers alone. To properly model this scenario, we issue separate transactions at both the entry and exit points of the inter-chiplet boundary.

As shown in Fig. 4, a 1024B packet carrying data undergoes splitting at the D2D interface after incurring the protocol conversion latency (IL). The first simulation packet is transferred to the opposite die and occupies the channel, after an additional IL delay, proceeds to the next module. The last packet, which releases the channel, must wait for a delay calculated as $\frac{PacketSize}{IW * F}$ plus IL, as determined by the D2D interface parameters, before moving on to the next module. These split simulation packets enable accurate modeling of channel bottlenecks. These two transactions are then merged again at the TLM domain endpoints, such as the transactor or the DDR interface. By adopting this method, we avoid excessive use of simulation packets, thereby preserving the abstraction level of TLM while still enabling cycle-level latency modeling through the I/O die.

Interconnect: A simple interconnect connects the D2D interface and the DDR interface inside the I/O die, with configurable latency added to each route.

DDR Interface: The DDR interface acts as a TLM endpoint, bridging the simulation backplane and the DRAM simulator while also managing synchronization.

V. EVALUATION

A. Overview

We conducted three sets of experiments on chiplet architectures with an I/O die. First, we compared a monolithic system

TABLE II: Total DRAM access

Workload	Total DRAM access
MobileNetV2 [22]	4.2 MB
DiscoGAN [23]	5.6 MB
YOLOv7-tiny [24]	10.7 MB
ResNet50 [25]	27.6 MB

TABLE III: Simulation parameters of modules

Module parameters	
Core	1.0 GHz
Crossbar	Channel width (512 bit)
SPM	Frequency (1.0 GHz), Request size(64B), Access delay(1.35ns), Size(8MB)
D2D interface	Frequency (1.0 GHz), IW (128 bit), IL(0 ns)
DRAM	Type(LPDDR4), Frequency (1.6 GHz), Request size(64B), Capacity(3GB)

Note: () means default setting

against an I/O die-based system to evaluate changes in DRAM access latency and overall performance. Second, we varied key I/O die parameters to analyze their impact on memory latency. Finally, we explored system configurations that achieve near-optimal performance under fixed conditions. For design space exploration (DSE), we considered two scenarios: optimizing the number of chiplets for homogeneous applications and optimizing core allocation across chiplets for multi-application workloads.

To drive these experiments, we selected four CNN applications, MobileNetV2 [22], DiscoGAN [23], YOLOv7-tiny [24], and ResNet50 [25]. Table II summarizes their memory access characteristics. For image classification, we chose a small model (MobileNetV2) and a medium-scale model (ResNet50). In addition, we included DiscoGAN, a compact image-to-image translation model, and YOLOv7-tiny, a well-known model for real-time object detection. The parallelization method employed is pipelining, where the compiler partitions networks into multiple pipelining stages. Note that the purpose of this multi-NPU compiler is primarily to generate traces rather than to achieve optimal performance.

B. Simulation Setup

Simulation parameters for modules are summarized in Table III with their baseline configurations. The processing component model is based on the open-source NPU simulator [19], operating at a fixed frequency of 1 GHz. The number of banks in the SPM equals the number of processing components in a chiplet, allowing each component to access its assigned SPM bank without conflicts. Each SPM bank is 8 MB in size, and the interconnect between cores and SPM is implemented as a crossbar, connecting all cores to all banks with a channel width of 512 bits. The D2D interface is configured with a frequency of 1 GHz, a 128-bit width, and a base latency of 0 ns. For main memory, we employ LPDDR4 modeled in Ramulator [20], configured with a frequency of 1.6 GHz, a capacity of 3 GB, and a single channel as the baseline. We also vary the system architecture and application workload according to the parameters listed in Table IV.

None of the experiments conducted in this study required more than 40 minutes per run, demonstrating significantly

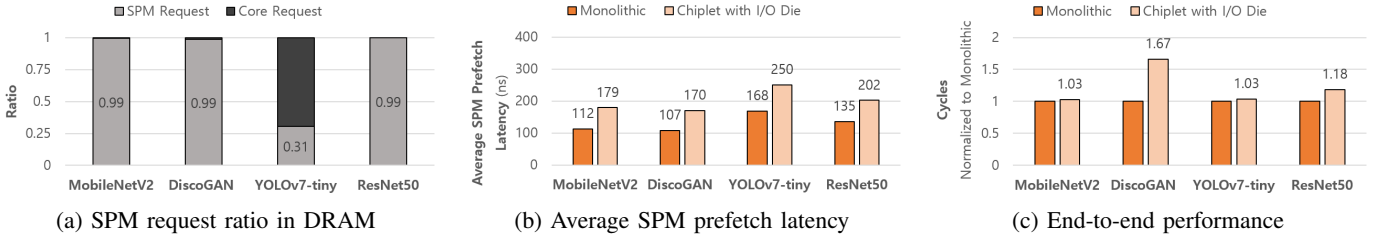


Fig. 5: Monolithic vs Chiplet-based system with an I/O die

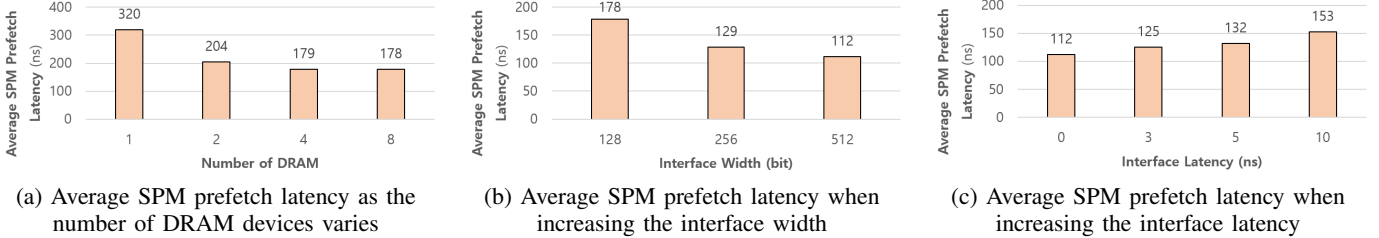


Fig. 6: Average SPM prefetch latency when changing D2D parameters for MobileNetV2

TABLE IV: Simulation parameters of system and network

System parameters	Workload parameters
Number of cores	Number of networks
Number of chiplets	Type of network
Number of DRAM	Address to save data
Number of cores in each chiplet	Number of cores for each network

faster simulation performance compared to existing simulation frameworks.

C. Monolithic vs Chiplet-based System with an I/O Die

A monolithic system is compared against a chiplet system with an I/O die using an experiment comprising four cores and a single DRAM device. In the chiplet setup, the four cores form one processor die connected to an I/O die. A single network is mapped across the four cores in the processor die.

To interpret the results, we first examine the DRAM access pattern of the I/O die-based system. Under this configuration, DRAM traffic can be classified into two categories: (i) core-originated memory accesses and (ii) SPM-originated accesses for data prefetching. As shown in Fig. 5(a), for three applications (all except YOLOv7-tiny), SPM-originated accesses constitute nearly 100% of total DRAM requests. For YOLOv7-tiny, the SPM share is 31%.

Since SPM prefetch dominates overall memory traffic, we analyze how its latency changes per application. Fig. 5(b) shows an increase of 50–60%, depending on the workload. We then assess how this increased memory access latency affects end-to-end performance. In Fig. 5(c), DiscoGAN on the chiplet system with an I/O die experiences a 67% performance drop relative to the monolithic die, whereas MobileNetV2 and YOLOv7-tiny degrade by only 3%. This indicates that, because SPM prefetch can proceed in parallel with Core \leftrightarrow SPM accesses, higher SPM prefetch latency does not uniformly translate to performance loss; its impact is workload dependent and must be evaluated by simulation.

D. Impact of I/O Die Parameters

In the second experiment, we scale the system and investigate how D2D interface parameters affect SPM prefetch latency. We use MobileNetV2 as the workload. The baseline with 4 cores in a single processor chiplet is expanded to 16 cores organized as four processor chiplets (four cores per chiplet). By instantiating multiple processor chiplets connected to a single I/O die, we observe how DRAM access latency changes when several chiplets share the hub. The number of network instances is increased to four accordingly and assigned to each chiplet.

We first vary the number of DRAM devices attached to the I/O die and analyze the resulting memory-access latency. Because the measured latency includes both network delay and DRAM internal delay, increasing the number of DRAM devices helps to separate these effects. When adding more memories, each core’s requests are distributed across the DRAM devices to avoid hotspotting. As shown in Fig. 6(a), after increasing the number of processor-chiplets, the average SPM prefetch latency decreases from 320 ns with one DRAM to 178 ns with eight DRAMs. The difference in average SPM prefetch latency between 4 DRAMs and 8 DRAMs was only 1 ns, indicating convergence.

We then fix the eight-DRAM setup so the DRAM side is largely free from bottleneck and sweep D2D interface parameters to observe latency changes. First, we scale the interface width from 128 bits, doubling up to 512 bits. As shown in Fig. 6(b), the average SPM prefetch latency drops from 178 ns to 112 ns, which is within the range observed in real hardware (AMD EPYC 7002 Series [6]: 94 ns to 114 ns with 8 DRAM channels). Next, we add interface conversion latency and measure its impact. With +3 ns, +5 ns, and +10 ns injected, Fig. 6(c) shows the SPM prefetch latency increases from the 0 ns baseline of 112 ns to 153 ns of +10 ns interface latency.

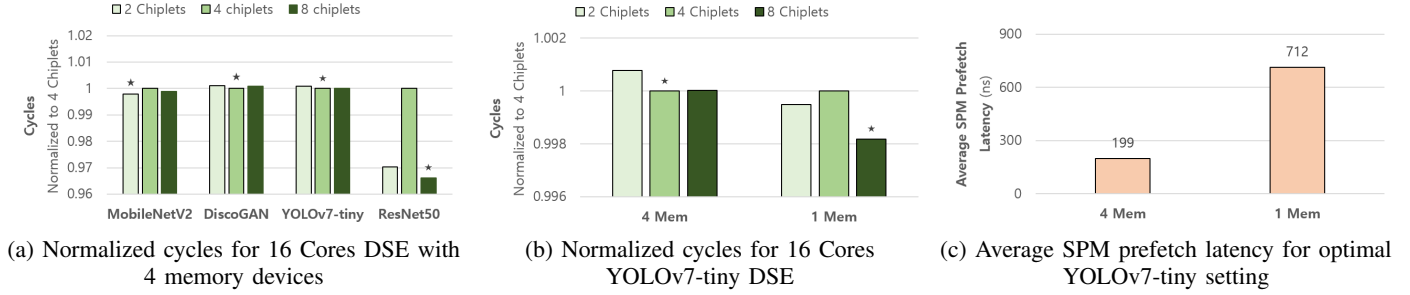


Fig. 7: Design space exploration for homogeneous networks (* : The minimum cycle count)

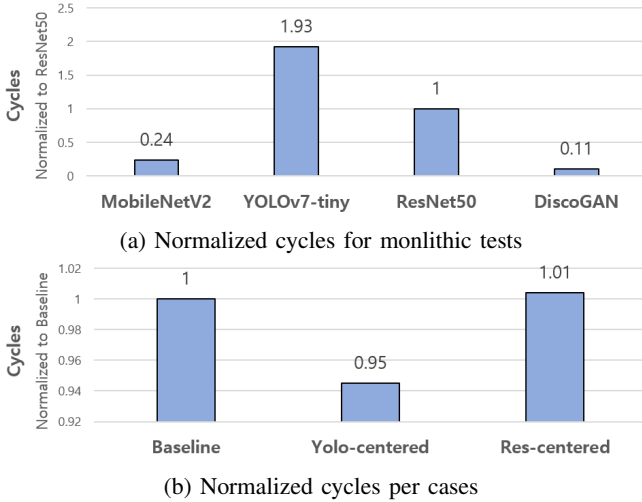


Fig. 8: Design space exploration for multi applications

E. Design Space Exploration

After confirming that the D2D interface parameters function as intended, we conducted design space exploration. First scenario explores the optimal number of chiplets under fixed conditions when running homogeneous networks. Specifically, we configure 16 processor cores, 4 DRAM devices, D2D interface latency of 3 ns, and an interface width of 512 bits, and run eight network instances. We do not consider unbalanced partitions; instead, we compare 2 chiplets (8 cores each), 4 chiplets (4 cores each), and 8 chiplets (2 cores each). For YOLOv7-tiny, the default SPM capacity was insufficient, so we doubled the SPM bank size.

As shown in Fig. 7(a), MobileNetV2 achieves its best performance with 2 chiplets, DiscoGAN and YOLOv7-tiny perform optimally with 4 chiplets, and ResNet50 with 8 chiplets. To examine how a stronger memory bottleneck affects YOLOv7-tiny whose traffic is more sensitive to Core \leftrightarrow DRAM we repeat the same DSE after reducing the DRAM count to 1. In Fig. 7(b), the optimal configuration shifts to 8 chiplets. Fig. 7(c) further shows that YOLOv7-tiny’s average SPM prefetch latency rises from 199 ns (with 4 DRAMs) to 712 ns (with 1 DRAM). These results demonstrate that the optimal chiplet count is configuration-dependent, varying with memory bottleneck and interface parameters.

To evaluate chiplet-level heterogeneity, we conducted a de-

sign space exploration with multiple applications. In this setup, we assumed eight cores running four different applications and explored the optimal chiplet combinations. As the baseline, based on the monolithic die results of Fig. 8(a), the two applications with shorter execution times (DiscoGAN and MobileNetV2) were assigned to single-core chiplets, while the two applications with longer execution times (ResNet50 and YOLOv7-tiny) were assigned to three-core chiplets. We then compared this baseline against cases where one additional core was allocated to each of the two slower applications. The case where ResNet50 used four cores is referred to as Res-centered, and the case where YOLOv7-tiny used four cores is referred to as Yolo-centered. As a result of the DSE, Fig. 8(b) shows that the YOLO-centered case achieved approximately 5% performance improvement compared to the baseline, indicating that allocating four cores to YOLOv7-tiny is the most effective strategy. Since YOLOv7-tiny represents a significantly heavier workload than the other applications, this result is consistent with expectation.

VI. CONCLUSION

In the design of multi-chiplet architectures, accurately evaluating die-to-die (D2D) overhead and mitigating its impact are critical challenges. TSIM4ICS addresses this need by modeling I/O-die-based NoP architectures at the trace-driven SystemC-TLM abstraction level. Through the integration of cycle-aware modules, it preserves TLM efficiency while achieving cycle-level fidelity, and its detailed D2D interface model closely reflects real hardware behavior.

Furthermore, combined with a multi-NPU component simulator, TSIM4ICS enables design-space exploration on real-world applications, demonstrating its effectiveness as a practical tool for research and development in chiplet-based systems. Looking ahead, the framework can be extended to incorporate power and thermal modeling, heterogeneous chiplet integration, and emerging memory technologies, thereby further enhancing its utility for next-generation system design.

ACKNOWLEDGMENTS

This work was supported by Samsung Electronics Company Ltd., under Grant IO201210-07941-01; The ICT at Seoul National University provides research facilities for this study.

REFERENCES

- [1] Y. Feng, Y. Wei, D. Xiang, and K. Ma, "Evaluating chiplet-based Large-Scale interconnection networks via Cycle-Accurate Packet-Parallel simulation," in *2024 USENIX Annual Technical Conference (USENIX ATC 24)*. Santa Clara, CA: USENIX Association, Jul. 2024, pp. 731–747. [Online]. Available: <https://www.usenix.org/conference/atc24/presentation/feng-yinxiao>
- [2] F. Schätzle, C. Falquez, S. Heinen, N. Ho, A. Portero, E. Suarez, J. Van Den Boom, and S. Van Waasen, "Modeling methodology for multi-die chip design based on gem5/systemc co-simulation," in *Proceedings of the 16th Workshop on Rapid Simulation and Performance Evaluation for Design*, ser. RAPIDO '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 35–41. [Online]. Available: <https://doi.org/10.1145/3642921.3642956>
- [3] Y. Feng, D. Xiang, and K. Ma, "A scalable methodology for designing efficient interconnection network of chiplets," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2023, pp. 1059–1071.
- [4] —, "Heterogeneous die-to-die interfaces: Enabling more flexible chiplet interconnection systems," in *2023 56th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2023, pp. 930–943.
- [5] N. B. Mallya, P. Strikos, B. Goel, A. Ejaz, and I. Sourdis, "A performance analysis of chiplet-based systems," in *2025 Design, Automation Test in Europe Conference (DATE)*, 2025, pp. 1–7.
- [6] S. Naffziger, N. Beck, T. Burd, K. Lepak, G. H. Loh, M. Subramony, and S. White, "Pioneering chiplet technology and design for the amd epyc™ and ryzen™ processor families : Industrial product," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021, pp. 57–70.
- [7] "5th gen amd epyc processor architecture," Advanced Micro Devices, Inc., White Paper, March 2025, first Edition. [Online]. Available: <https://www.amd.com/en/products/processors/server/epyc/9005-series.html>
- [8] L. B. Alvarez, G. Chehaibar, S. Busch, P. Benoit, and D. Novo, "c2c-gem5: Full system simulation of cache-coherent chip-to-chip interconnects," in *2025 Design, Automation Test in Europe Conference (DATE)*, 2025, pp. 1–7.
- [9] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, p. 1–7, aug 2011. [Online]. Available: <https://doi.org/10.1145/2024716.2024718>
- [10] N. Jiang, D. U. Becker, G. Micheliogiannakis, J. Balfour, B. Towles, D. E. Shaw, J. Kim, and W. J. Dally, "A detailed and flexible cycle-accurate network-on-chip simulator," in *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2013, pp. 86–96.
- [11] D. Sanchez and C. Kozyrakis, "Zsim: fast and accurate microarchitectural simulation of thousand-core systems," *SIGARCH Comput. Archit. News*, vol. 41, no. 3, p. 475–486, Jun. 2013. [Online]. Available: <https://doi.org/10.1145/2508148.2485963>
- [12] "Ieee standard for standard systemc® language reference manual." *IEEE Std 1666-2023 (Revision of IEEE Std 1666-2011)*, pp. 1–618, 2023.
- [13] L. Cai and D. Gajski, "Transaction level modeling: an overview," in *First IEEE/ACM/IFIP International Conference on Hardware/ Software Codesign and Systems Synthesis (IEEE Cat. No.03TH8721)*, 2003, pp. 19–24.
- [14] D. Yun, S. Kim, and S. Ha, "Relaxed synchronization technique for speeding-up the parallel simulation of multiprocessor systems," in *17th Asia and South Pacific Design Automation Conference*, 2012, pp. 449–454.
- [15] P. Strikos, A. Ejaz, and I. Sourdis, "Bzsim: Fast, large-scale microarchitectural simulation with detailed interconnect modeling," in *2024 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2024, pp. 167–178.
- [16] S. Li, Z. Yang, D. Reddy, A. Srivastava, and B. Jacob, "Dramsim3: A cycle-accurate, thermal-capable dram simulator," *IEEE Computer Architecture Letters*, vol. 19, no. 2, pp. 106–109, 2020.
- [17] S. Wilton and N. Jouppi, "Cacti: an enhanced cache access and cycle time model," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 5, pp. 677–688, 1996.
- [18] J. Zhang, X. Wang, Y. Ye, D. Lyu, G. Xiong, N. Xu, Y. Lian, and G. He, "M2m: A fine-grained mapping framework to accelerate multiple dnns on a multi-chiplet architecture," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 32, no. 10, pp. 1864–1877, 2024.
- [19] D. Kang, J. Kang, H. Kwon, H. Park, and S. Ha, "A novel convolutional neural network accelerator that enables fully-pipelined execution of layers," in *2019 IEEE 37th International Conference on Computer Design (ICCD)*, 2019, pp. 698–701.
- [20] Y. Kim, W. Yang, and O. Mutlu, "Ramulator: A fast and extensible dram simulator," *IEEE Computer Architecture Letters*, vol. 15, no. 1, pp. 45–49, 2016.
- [21] Arm Ltd., *AMBA AXI and ACE Protocol Specification*, Arm Ltd., Mar. 2023, Available at: <https://developer.arm.com/documentation/ih0022/latest/>.
- [22] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," 2019. [Online]. Available: <https://arxiv.org/abs/1801.04381>
- [23] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," 2017. [Online]. Available: <https://arxiv.org/abs/1703.05192>
- [24] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
- [25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. [Online]. Available: <https://arxiv.org/abs/1512.03385>