Theseus: Exploring Efficient Wafer-Scale Chip Design for Large Language Models

Jingchen Zhu[†‖], Chenhao Xue[‖], Yiqi Chen[‖], Zhao Wang[†‖], Chen Zhang[*],
Yu Shen[†], Yifan Chen[†], Zekang Cheng[§], Yu Jiang[¶], Tianqi Wang[‡],
Yibo Lin[‖], Wei Hu[†], Bin Cui[†], Runsheng Wang[‖], Yun Liang[‖], Guangyu Sun[‖††]
[†]School of Computer Science, Peking University [‖]School of Integrated Circuits, Peking University
[*]Shanghai Jiao Tong University [§]University of Science and Technology of China [¶]Tsinghua University
[‡]Huawei [††]Beijing Advanced Innovation Center for Integrated Circuits
{zjc990112, gsun} @pku.edu.cn

***Abstract*—The emergence of the large language model (LLM) poses an exponential growth of demand for computation throughput, memory capacity, and communication bandwidth. Such a demand growth has significantly surpassed the improvement of corresponding chip designs. With the advancement of fabrication and integration technologies, designers have been developing Wafer-Scale Chips (WSCs) to scale up and exploit the limits of computation density, memory capacity, and communication bandwidth at the level of a single chip. Existing solutions have demonstrated the significant advantages of WSCs over traditional designs, showing potential to effectively support LLM workloads.**

Despite the benefits, exploring the early-stage design space of WSCs for LLMs is a crucial yet challenging task due to the enormous and complicated design space, time-consuming evaluation methods, and inefficient exploration strategies. To address these challenges, we propose Theseus, an efficient WSC design space exploration framework for LLMs. We construct the design space of WSCs with various constraints considering the unique characteristics of WSCs. We propose efficient evaluation methodologies for large-scale NoC-based WSCs and introduce multi-fidelity Bayesian optimization to efficiently explore the design space. Evaluation results demonstrate the efficiency of Theseus that the searched Pareto optimal results outperform GPU cluster and existing WSC designs by up to 62.8%/73.7% in performance (with the same or lower power) and 38.6%/42.4% in power consumption (with the same or higher performance) for LLM training, while improving up to 23.2× and 15.7× for the performance and power of inference tasks. Furthermore, we conduct case studies to address the design tradeoffs in WSCs and provide insights to facilitate WSC designs for LLMs.

***Index Terms*—Wafer-scale Chip, Design Space Exploration, Large Language Model**

I. Introduction

Recent advancements in neural networks have led to a significant escalation in model size, particularly in the realm of large language models (LLMs). The evolution from Bert to GPT4 [7], [11], [38] has resulted in a parameter increase exceeding a thousand-fold. This upward trend is anticipated to persist, owing to the superior performance of larger models in tasks related to natural language understanding and content generation [38], [54]. Concurrently, it necessitates significantly enhanced computational throughput, memory capacity, and data communication bandwidth in hardware infrastructures [43], [45]. The surge in demand has markedly exceeded advancements in corresponding chip designs like GPUs [36], attributed to the field size limitation of lithography steppers. This constraint is referred to as the **reticle limit**, defined as 26 mm by 33 mm, or 858 mm² [5]. State-of-the-art chip designs, such as NVIDIA H100 [36], B200 [35], and Intel Gaudi3 [18], are approaching and even surpassing the reticle limit in scale.

Fortunately, advancements in chiplet design method [46], die-to-die interconnection [51], and 2.5D/3D stacking methods [30], have facilitated scaling beyond the reticle limit. Innovative chip designs that surpass the limit have been proposed for artificial intelligence applications [10], [19], [26], [49], etc. Among these solutions, the **Wafer-Scale Chip (WSC)** designs (e.g. Cerebras WSE2 and Tesla's Dojo) stand out as a promising approach to maximize computation density, on-chip memory capacity, and communication bandwidth. Cerebras WSE2 [26] integrates 850,000 cores, 40GB of on-chip SRAM, and 200Pb/s of on-chip fabric bandwidth on a 46,225mm² monolithic silicon substrate and achieves 7.5 PFLOPS for large-scale GEMM operation. Tesla's Dojo [10] comprises 25 D1 dies, 10TB/s on-tile bisection bandwidth, and 36TB/s off-tile aggregate bandwidth, achieving a performance of 9 PFLOPS. Compared to GPUs, these WSC designs offer over 7× the peak performance, 200× on-chip memory bandwidth, and 5× the inter-chip bandwidth, alongside significantly improved energy efficiency, enabling more effective scaling for both training and inference tasks on LLMs.

Despite the aforementioned advantages of WSCs, it is crucial to determine the optimal configurations of WSCs to strike a balance in utilizing diverse resources, achieve peak performance, and enhance energy efficiency. Our experiments have demonstrated that improper designs can significantly degrade the achievable performance of WSCs in LLM workloads, sometimes by several tens of times. Furthermore, varying application workloads and optimization objectives can result in a range of designs, each with notable differences. Hence, early-stage **Design Space Exploration (DSE)** becomes imperative in crafting efficient WSCs capable of delivering optimal performance and energy efficiency across a spectrum of application requirements.

A Design Space Exploration process typically involves three key stages: **design space construction**, **design point evaluation**, and **exploration strategy**. Although DSE has been thoroughly studied on AI accelerators [13], [31], adopting existing DSE methods to the WSC scenario introduces chal-

lenges in all these stages. Firstly, the design space of WSCs is larger and more complex, requiring consideration of numerous parameters and various constraints compared to traditional AI accelerators. Secondly, there exists several obstacles to evaluating the performance of a WSC configuration accurately and efficiently, especially with regard to the communication behaviours at scale. Lastly, conducting multi-objective optimization over the complex design space poses challenges for the explorer to efficiently characterize the target space and find optimal designs with a minimum number of sampled design points. To tackle these challenges, we propose Theseus, a DSE framework to facilitate high-efficiency WSC design for LLM workloads. The contributions of this paper can be summarized as follows:

- We construct an extensive design space that encompasses WSC architecture configurations with wide range of candidate values, and introduce parameters to explore heterogeneous WSC designs. We consider various design constraints including area, power, yield, stress, etc.
- We propose a hierarchical methodology for evaluating LLMs on WSCs through tile-level, op-level and chunk-level evaluations with considerations of communication at different levels. At the op-level, we employ a graph neural network (GNN)-based method for fast and accurate NoC estimation to support multi-fidelity optimization.
- We propose a multi-fidelity multi-objective Bayesian optimization (MFMOBO) algorithm to efficiently explore the design space of WSCs for LLMs and leverage the advantages of various evaluation methodologies.
- We conduct case studies using our proposed Theseus framework for addressing the design tradeoffs in WSCs, and provide analysis and insights to facilitate the design optimization of WSCs for LLMs.

II. Background and Related Work

A. WSC Basics

Wafer-Scale Chip (WSC) design has emerged as a promising solution to alleviate inter-reticle communication overhead with low-cost on-wafer interconnections and has demonstrated advantages in computational power and on-chip memory capacity for efficiently supporting neural network workloads. The concept of WSCs has attracted significant attention from both industry and academia. Recently, two notable commercial WSC designs have been proposed. Cerebras WSE2 [26] employs offset exposures and proprietary layers of interconnect to stitch dies together directly on monolithic wafers. This results in a uniform and continuous fabric across the reticle boundary, which provides good hardware abstraction. Tesla Dojo [10] uses a redistribution layer (RDL) to interconnect D1 chips with SerDes and integrate D1 chips with integrated fan-out system-on-wafer (InFo-SoW) packaging, with known-good-die (KGD) techniques to ensure yield requirement.

B. Related Works

Existing research on WSCs has primarily focused on two aspects: addressing hardware design challenges and optimizing software execution on WSC architectures. On the hardware side, prior studies have explored WSC design methodologies and challenges [39], [40]. On the software side, research has investigated efficient workload mapping on specific WSC engines [28], [29]. In LLM scenarios, Zhang et al. [55] evaluate the performance of LLMs on the Cerebras WSE [26], while WaferLLM [17] introduces scalable GEMM and GEMV algorithms to enhance LLM inference efficiency on the Cerebras WSE-2. However, these studies leave the architectural design space underexplored. Chiplet Cloud [42] proposes a parameterizable chiplet-based architecture designed to reduce the total-cost-of-ownership (TCO) for LLM serving. However, it entirely replaces external memory (e.g., HBM and DDR) with on-chip SRAM, significantly restricts its applicability. Furthermore, it does not account for key design considerations essential for WSC development.

C. WSC for LLM: Advantages and Challenges

The large computational power and on-chip buffer of WSCs provide evident advantages for LLM workloads, facilitating enhanced performance for compute-intensive operations and reduced off-chip memory access through efficient on-chip data reuse. Efficient inter-reticle communication is also pivotal for overall performance and power improvements. In both LLM training and inference tasks, the ever-growing model sizes necessitate the adoption of efficient parallel strategies [48] and memory optimizations [43] to meet memory capacity constraints in distributed systems. These optimizations typically increase the communication between chips and nodes, which can benefit greatly from the inter-reticle connections of WSCs. Additionally, WSCs can leverage stacking memory with high memory bandwidth to meet the demands of the decode stage during inference. Although high bandwidth may necessitate sacrificing some memory capacity, the introduced communication can still be efficiently managed by inter-reticle communication to ensure improved overall performance. However, compared to scale-out solutions, WSCs have less flexible interconnect topologies between reticles, often constrained to 2D-mesh architectures. This poses challenges for both communication performance and system evaluation at scale.

III. Motivational Analysis

Challenge 1: Design Space Construction. In addition to the basic architecture parameters in traditional accelerator chips, WSCs introduce design considerations at both the reticle and wafer levels. Due to the large scale of WSCs, design configurations such as computational capacity, memory organization and interconnection bandwidth exhibit a broader range of variability. Meanwhile, WSCs need to validate against various design constraints. To address this challenge, in Theseus, we construct an extensive design space of WSCs for LLMs, detailed in Sec. V, and propose modeling approaches for various WSC metrics including area, power, yield and stress.

Challenge 2: Design Point Evaluation. With the expansive scale of computational resources in WSCs, the data transfer between cores and reticles emerges as a crucial concern of system performance. Due to the complexity of transmissions introduced by multi-level parallelisms when mapping LLMs onto WSCs, as well as the scheduling and mapping of task Directed Acyclic Graphs (DAGs) onto local spatial architectures, we must carefully consider traffic flow congestions

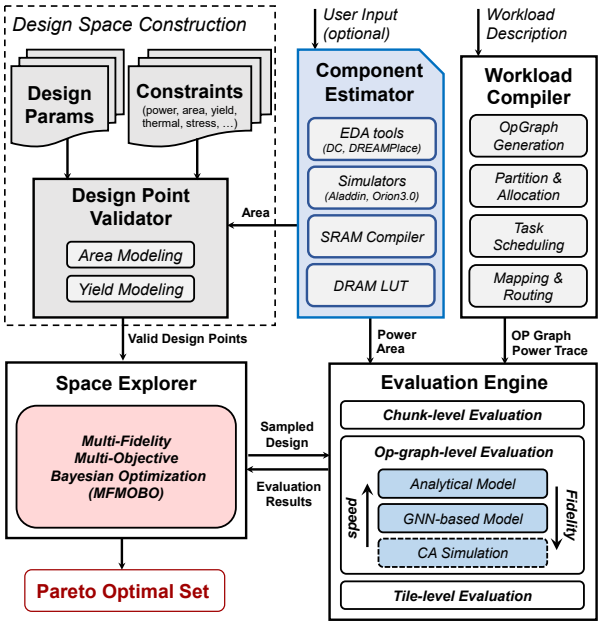


Fig. 1. Theseus Framework Overview

in NoC estimation. Traditional DSE methods rely on cycle-accurate (CA) simulators [20] for NoC evaluation, which can be time-consuming at the scale of WSCs. Existing NoC analytical models, in contrast, can provide rapid NoC estimation results. However, these models usually assume predetermined traffic patterns [37] (e.g. tornado or uniform), which may not capture the application-specific details. Recently, several machine learning-based methods have been proposed to efficiently evaluate application-specific NoC performance [25]. However, these methods face limitations in dealing with variable package sizes and the heterogeneity in NoC bandwidth, thereby impeding their direct application to WSC scenarios. To address this challenge, in Theseus, we propose a hierarchical evaluation method to reduce the estimation scale of NoC, and introduce a GNN-based method for fast and accurate NoC performance estimation.

Challenge 3: Exploration Strategy. WSC designs are designed with stringent power constraints determined by the design of heat dissipation and power delivery network [40]. This necessitates a delicate balance between optimizing both computation performance and power consumption. The multi-objective optimization considering both performance and power of WSCs is non-trivial, especially within our design space with enormous design parameters, and irregular shapes due to various design constraints. Meanwhile, enhancing the convergence and efficiency of this optimization process to minimize the number of iterations requires careful algorithm design. To address this challenge, in Theseus, we introduce MFMOBO to leverage the information from less accurate yet faster evaluation methodologies.

IV. Theseus Overview

To tackle the challenges mentioned above, we propose Theseus, an efficient design space exploration framework that jointly optimizes performance and power for WSC to search for Pareto-optimal designs. Fig. 1 presents an overview of Theseus. The DSE process begins with design space construction, which involves combining the design parameters of WSC and employing the **Design Point Validator** to discard design points that can not satisfy the constraints. Next, the **Space Explorer** employs our proposed Multi-Fidelity Multi-Objective Bayesian Optimization (MFMOBO) to iteratively sample design points from the constructed design space. To evaluate the sampled design points, we employ a **Component Estimator** to calculate the area and power of WSC basic modules, and implement a **Workload Compiler** to compile LLM workloads onto WSCs. The **Evaluation Engine** hierarchically evaluates the selected WSC design in tile-level, op-graph-level and chunk-level evaluation. The evaluation engine provides performance and power results in different fidelity according to the needs of the explorer. Finally, the explorer selects design points for the next iteration based on the feedback from the evaluation engine. This iterative optimization process continues until reaching the pre-set iterations number, and the Theseus framework outputs the searched Pareto optimal set. In the following sections, we detail the designs of individual components within Theseus.

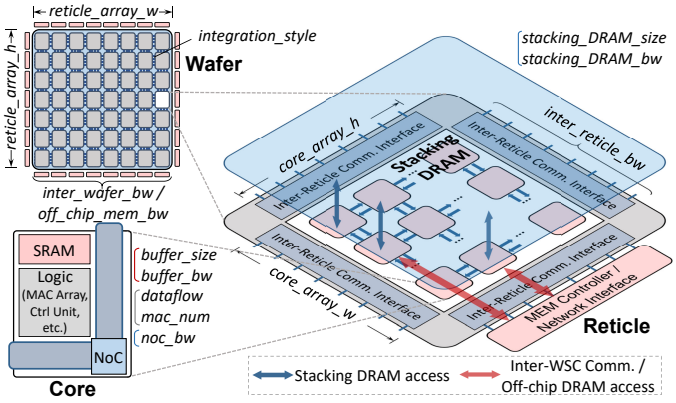


Fig. 2. Typical WSC Design and Architecture Parameters

V. WSC Design Space

A. WSC Architecture Parameters

To comprehensively explore the architecture design of WSCs, we consider crucial parameters in the three hierarchies of core, reticle and wafer. Fig. 2 labels these architecture parameters. At the **Core** level, on-chip SRAM capacity impacts the core's data reuse ability and the granularity of communication between cores. SRAM bandwidth determines the utilization of processing units and impacts intra-core dataflow optimization. Dataflow describes the pattern of data transmission and reuse across MAC units and the memory hierarchy, shaping the design of fixed datapaths and the operational efficiency of various operators [13], [46]. MAC number sets the upper limit of a single core's tensor computation capacity. Lastly, NoC bandwidth is pivotal in determining the effectiveness of inter-core communication. At the **Reticle** level, cores are connected with NoC to form a 2D-mesh array, which impacts the shape and peak performance of a reticle. Inter-reticle communication bandwidth are provided by the communication interfaces around the reticle to support data transmission across the reticle boundaries. We also consider stacking DRAM for efficient memory access based on through-silicon-via (TSV), with a certain capacity and bandwidth per

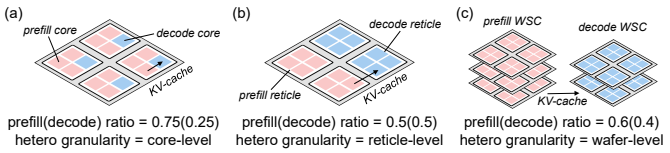


Fig. 3. Examples of Heterogeneous Modeling

unit area. At the **Wafer** level, reticles are further integrated into an array of certain height and width. The integration technology determines the area overhead and power consumption for inter-reticle communication. The memory controllers and network interfaces around the wafer provide off-chip DRAM access bandwidth and communication bandwidth between WSCs, which allows larger memory capacity and further scaling out of WSC systems.

B. Heterogeneous Modeling

To investigate the impact of heterogeneous WSC design on LLM inference performance, we introduce two parameters for characterization: prefill(decode) ratio and heterogeneous granularity. The prefill(decode) ratio represents the proportion of computational resources allocated to the prefill and decode stages. Heterogeneous granularity indicates the level of heterogeneity in the architecture hierarchy. Fig. 3 presents several examples of heterogeneous design at different levels. Core-level heterogeneity is achieved through software scheduling, where different stacking memory bandwidths are allocated to prefill and decode stage cores within the same reticle. Reticle-level and wafer-level heterogeneities are achieved by adjusting stacking memory bandwidth. In reticle-level heterogeneity, heterogeneous reticles with different memory bandwidths are integrated into the same WSC and simultaneously execute both stages, while wafer-level heterogeneity requires computations for the prefill stage and decode stage to be performed on different WSCs.

C. Defective Core Modeling

Traditionally, a core's yield is influenced by its area and the process technology employed. The relationship is encapsulated by the well-known Murphy Model [33], as formulated in Equation 1. In this model, the parameter A represents the core area (in cm^2), while D denotes the defect density (in defects per cm^2), serving as an empirical parameter associated with the process technology. The function $f(D)$ denotes the probability density function (PDF) of D.

$$Yield_{Murphy} = \int_0^\infty e^{-AD} f(D) dD \tag{1}$$

In WSC design, additional factors such as screw holes and Through-Silicon Vias (TSVs) may further impact a core's yield. To maintain the wafer's flatness and stability during manufacturing, it is often secured to the underlying PCB using screws. Screw holes are strategically placed at the intersections of reticles. The stress exerted by these screws can lead to a certain degree of yield degradation in the areas surrounding the holes. Similarly, TSVs, which involve drilling holes in the center of the reticle and filling them with conductive material for electrical interconnections between silicon layers, can also result in yield loss for nearby cores. The impact of screw holes and TSVs on yield is depicted in Fig. 4.

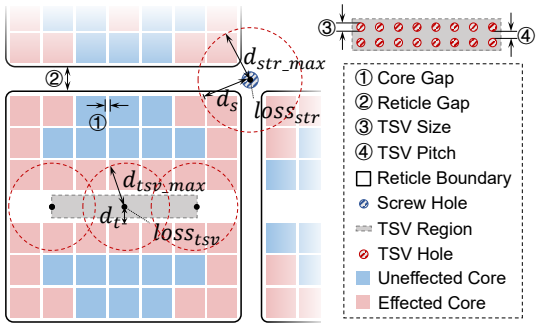


Fig. 4. Impact of Screw Holes and TSVs on Core Yield

As depicted in Fig. 4, the symbol d_{str_max} denotes the maximum distance over which a stress hole affects yield. The impact of a stress hole on yield is modeled in Equation 2:

$$Yield_{str} = 1 - loss_{str} \cdot \left(1 - \frac{d_s}{d_{str\_max}}\right)^n \tag{2}$$

where d_s signifies the distance from the stress hole to the nearest vertex of the core, $loss_{str}$ indicates the rate of yield loss attributed to the center of the stress hole, and n is a nonlinear exponent that governs the spatial decay rate of stress-induced yield degradation.

The model describing the impact of TSVs on yield closely mirrors that of screw holes. TSVs must be created on the wafer's surface due to their functional requirements and inherent characteristics. Consequently, the area designated for TSVs cannot overlap with the area allocated for a core. The additional overhead is linked to the total number of TSVs, which in turn is dictated by the bandwidth requirements of stacked DRAM. The yield of cores situated within a distance less than d_{TSV_max} from a TSV is affected, and this can be quantified similarly to Equation 2.

In conclusion, the yield of a core can be determined by consolidating all these elements, as outlined in Equation 3.

$$Yield_{core} = Yield_{Murphy} \times Yield_{str} \times Yield_{TSV} \tag{3}$$

D. Core Redundancy

With the error model of an individual core established, we can extend the calculation to determine the yield at both the reticle and wafer levels. Considering the substantial drop in overall yield that may occur when integrating numerous cores onto a single wafer, implementing appropriate redundancy mechanisms becomes essential to attain an acceptable yield target. A commonly employed strategy is to designate a portion of the cores as redundant, enabling them to substitute defective cores as required [26]. Assuming a reticle contains p operational cores and n redundant cores, the yield at the reticle level can be computed using Equation 4, where Y_{core} denotes the yield of individual cores within the reticle. Similarly, the wafer level yield can be further calculated using the reticle level yield.

$$Y_{PS} = \sum_{i=p}^{p+n} \binom{p+n}{i} {Y_{core}}^i (1 - Y_{core})^{p+n-i} \tag{4}$$

In setting a yield target, the selection of the percentage of redundant cores is critical, as this can lead to increased area overhead and a reduction in effective computational capability. This decision can also impact the selection of architectural

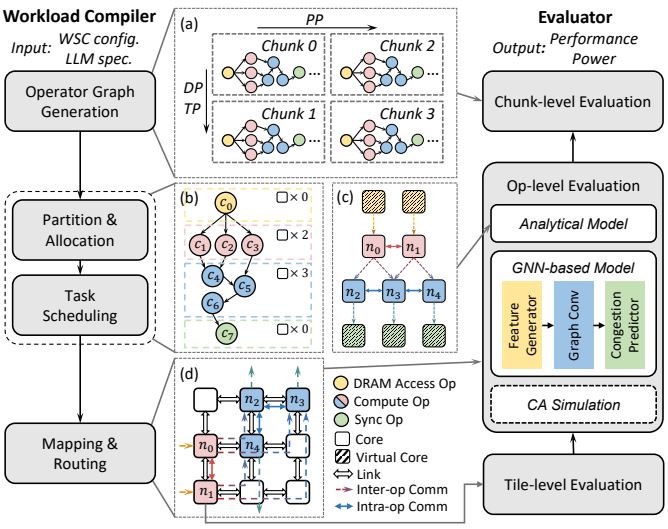


Fig. 5. Overview of the Evaluation Methodology

parameters during DSE. Furthermore, it is crucial to acknowledge that the choice of integration technique during DSE significantly influences yield. For example, Dojo's adoption of the known-good-die technique may lead to higher yields compared to Cerebras's die-stitching method. The disparity in these approaches can influence the overall yield outcomes, as well as the DSE results.

E. Design Constraints

Prior to initiating the exploration process, it is prudent to establish appropriate design constraints. This step helps in precluding invalid design solutions and enhancing the overall efficiency of the exploration. Key constraints that should be considered are discussed in this subsection.

Area Constraint: The area of the reticle and wafer cannot exceed the limits of the lithograph and silicon substrate.

Power Constraint: Considering power delivery and thermal dissipation capabilities, the power density of WSC cannot exceed the pre-determined threshold.

Yield Constraint: Considering the reserved redundant cores, WSC must meet the yield requirement to ensure manufacturing feasibility and cost efficiency.

SRAM Constraint: Some combinations of SRAM configurations are infeasible from the SRAM Compiler.

Stress Constraint: The area for stacking DRAM TSV holes should be less than the pre-determined area ratio (e.g., 1.5%) of a reticle for stress considerations.

VI. Evaluation Methodology

Fig. 5 illustrates the overview of evaluation methodology in Theseus. Specifically, we employ a Workload Compiler to determine the operating pattern and generate essential evaluation information. Based on this, we propose a hierarchical methodology to reduce the scale of evaluation, which estimates the performance and power through tile-level, op-level and chunk-level evaluation.

A. Workload Compiler

As shown in Fig. 5, Workload Compiler takes the WSC configuration and benchmark workload as inputs, and runs in the following steps:

(1)**Operator Graph Generation:** Given a parallel strategy, the LLM model is first segmented into model chunks. Workload Compiler constructs an operator graph for each chunk, as well as the corresponding data transmission between model chunks. Compute resources are evenly divided based on the number of chunks, with each chunk assigned to its designated compute resource.

(2) **Partition and Allocation:** After constructing the operator graph, Workload Compiler partitions it into disjoint subgraphs. Each subgraph is executed on the same compute resource using temporal multiplexing (intra-op parallelism), while different subgraphs are pipelined across multiple compute resources (inter-op parallelism). By adjusting the partitioning granularity, the compiler flexibly represents various intra-op and inter-op parallel strategies [8]. The optimal partitioning scheme is determined based on evaluation feedback while considering the SRAM capacity limits, and the number of compute cores allocated to each subgraph is proportional to its computational workload.

(3) **Task Scheduling:** We adopt a static scheduling approach to efficiently map operators within each subgraph onto the allocated logical compute cores. This process leverages dataflow analysis, where the multi-level nested loops of each operator are tiled along specific dimensions and parallelized across multiple cores while preserving inter-core data dependencies.

(4) **Mapping and Routing:** The scheduled logical cores are then mapped onto the physical core array at the operator level, as illustrated by arrows in Fig. 5. We leverage the state-of-the-art Gemini [9] framework to generate optimized mapping strategies, minimizing congestion hotspots under the given routing schemes. Workload Compiler then reles on routing algorithms to determine efficient data transmission paths between cores.

During the workload compiling process, we iterate through all combinations of TP, DP, PP, and micro-batch sizes that satisfy the memory capacity constraint and select the best-performance parallel strategy based on the evaluation results.

B. Tile-level Evaluation

Tile-level evaluation focuses on the evaluation of tensor operations on cores with fixed dataflow, which have been extensively studied and analyzed in prior works [23], [41]. Similarly, we perform unrolling and tiling over specific loop dimensions for tile-level evaluation, while considering the impact of SRAM capacity on data reuse. We record the interval of the tiled output for subsequent NoC estimation.

C. Op-level Evaluation

Based on the latency and communication patterns of individual cores from tile-level evaluation, op-level evaluation further estimates the performance of NoC-based core array with both inter-operator and intra-operator communications [8]. In op-level evaluation, we propose two methodologies trading off evaluation speed and accuracy for NoC estimation under complex communication scenarios, including the analytical model and GNN-based model.

Analytical Model. For fast estimation at the early iterations of DSE, we propose an analytical model for the communications on the core array within a chunk. Based on the output of

Workload Compiler, we figure out the data transmission volume on each link of the NoC and calculate the communication time between cores with the equivalent bandwidth. We label the communication delays on each edge of the logic core graph (Fig. 5(c)) and traverse the graph in topological order to select the longest path as the overall latency of the chunk. It is worth mentioning that the actual latency of a chunk needs to take into account the overhead of DRAM access and inter-chunk synchronization, which correspond to the edges connecting virtual cores in Fig. 5(c). These overheads will be considered in the chunk-level evaluation.

GNN-based Evaluation. To enhance the accuracy of NoC estimation by considering the congestion, we propose a novel GNN-based performance evaluation method. Compared to existing machine learning methodologies, our model effectively addresses the issue of variable packet sizes and the inherent heterogeneity in NoC bandwidth to support the evaluation of LLM workloads on WSCs. The overall workflow of our GNN-based evaluation method comprises three primary steps: *input graph construction*, *GNN-based prediction*, and *overall latency construction*. We define the *input graph for the GNN* as an attributed graph $G = (V, E, X_V, X_E)$, where V and E denote the sets of nodes and edges in the graph, respectively. $X_V \in \mathbb{R}^{|V|\times d_V}$ and $X_E \in \mathbb{R}^{|E|\times d_E}$ represent the feature matrices for nodes and edges, with d_V and d_E indicating the number of feature channels for nodes and edges, respectively. To accurately predict NoC traffic patterns and capture congestion information, we annotate this graph using critical information derived from the Workload Compiler and WSC architecture parameters. Specifically, the core topology graph generated during compilation (as illustrated in Figure 5 (d)) and associated annotations (including computation latency, transmitted data volume, occupied physical links, and transmission intervals) are encoded into the input graph for the GNN. Information related to computation is encoded into the node feature matrix X_V, while transmission-related information is encoded into the edge feature matrix X_E.

Fig. 5 presents the architecture of our GNN model. *Feature generator* consists of MLPs that project node features x_v and edge features x_e to initial hidden states h_v^0 and h_e^0. *Graph Convolution module* aggregates neighboring information with message passing mechanism [14] and update nodes' hidden states for T iterations. Similar to [25], message passing is conducted on both the original graph G and its reversed graph $\tilde{G}$ to model both upstream contention and downstream backpressure. We briefly summarize the message passing mechanism in Equation 5:

$$h_v^{t+1} = \alpha(h_v^t + m_v^{t+1}) \tag{5a}$$

$$mi_v^{t+1} = \sum_{u\in\mathcal{N}i(v)} \varphi(h_{u\to v}^0)h_u^t \tag{5b}$$

$$mo_v^{t+1} = \sum_{u\in\mathcal{N}o(v)} \varphi(h_{v\to u}^0)h_u^t \tag{5c}$$

$$m_v^{t+1} = \text{Concat}(mi_v^{t+1}, mo_v^{t+1}) \tag{5d}$$

where α is activation function, and φ is a linear layer that maps h_e to $\frac{|h_v|}{2} \times |h_v|$ matrix.

Congestion predictor predicts average channel waiting time y_e, which can reflect the traffic pattern of NoC. For edge $e = (u, v)$ that represents a physical link, y_e is predicted by

$$\hat{y}_e = \theta(\text{Concat}(h_u^T, h_v^T, h_e^0)), \tag{6}$$

in which θ is a MLP.

With the predicted average channel waiting time $\hat{y}$, we can reconstruct end-to-end transmission latency between cores. For a packet with k flits, the average transmission latency $t(k)$ can be calculated by:

$$t(k) = k + \sum_{\{l|\varphi(l)=1\}} \hat{y}_l, \tag{7}$$

in which $\varphi(l)$ is a Boolean indicating whether the transmission utilizes link l. We apply the GNN model on all edges between cores in the logic core graph (Fig. 5(c)). Similar to the analytical model, we can reconstruct the overall latency of a chunk by finding the critical path of the graph.

Cycle-accurate Simulation. We extend BookSim2 [20] for cycle-level simulation and dataset generation. We design a series of instructions and micro-instructions to describe the compute, memory access and communication of WSC cores, and connect the cores to Booksim routers. We add instruction support for Booksim to inject and forward packets according to the micro-instructions sent from the network interface of cores. We argue that for accelerator cores, when dealing with regular tensor operations, the latency for computation and memory access is relatively deterministic. So we simplify the estimation of computation and memory access latency inside cores by analytical models.

D. Chunk-level Evaluation

Chunk-level evaluation further considers the data transfer between chunks, including TP-induced collective communication, PP-induced cross-pipeline-stage communication, and DP-induced weight update communication. Additionally, we consider the traffic related to traditional off-chip DRAM access at the chunk level. Despite the equitable partitioning of chunks in terms of workload and hardware resources, the overhead for off-chip DRAM access can be different due to the diverse locations of reticles assigned to individual chunks. Considering all inter-chunk communications and DRAM access demands, we count the amount of data transmission for each inter-reticle link and assess the communication latency based on the available bandwidth. The DRAM access latencies are then combined with op-level evaluation results to derive the overall performance of individual chunks. Meanwhile, we consider the pipeline efficiency based on micro batch size to calculate the throughput on target workloads.

E. Area and Power Estimation

Area Estimation. We utilize the SRAM compiler to generate various configurations of SRAM Macros along with their height and width. We also implement MAC array in different dataflows, NoC routers, and RISC-V core as the control unit with Chisel based on Purlin [15]. The generated RTL is then synthesized using Synopsys Design Compiler, and the netlist is further input to DREAMPlace [27] for placement to obtain the area of the core array within a reticle. In estimating the inter-reticle distance and area overhead of inter-reticle communications, we refer to Cerebras WSE2, Tesla Dojo, and Nvidia GRS [51] for relevant information.

Power Estimation. We adopt a methodology similar to Aladdin [47], where we define a series of actions, including MAC operations, NoC transmissions, inter-reticle communications, and SRAM/DRAM accesses, etc., to estimate power consumption. We utilize SRAM Compiler to obtain SRAM static power and energy consumption for read and write operations. We refer to Aladdin, Orion3.0 [21], as well as existing chiplet and WSC designs [26], [40], [50], [51] for the power estimation of computational logic, NoC components, and communication modules. During the evaluation process, we record the number of action calls to calculate dynamic power and combine it with the static power of components to estimate power consumption for WSC designs.

Component Estimator. To reduce evaluation time for each sampled design and avoid redundant estimations for identical module configurations, we develop the Component Estimator, which constructs a dataset of area and power estimates for WSC basic modules across various configurations. During the DSE process, evaluation models directly retrieve and combine area and power values from this dataset to compute the overall area and power consumption of WSCs. The area-power table can be dynamically updated as needed to ensure both efficiency and accuracy in evaluations. Additionally, the Component Estimator supports optional user-provided inputs, allowing designers to incorporate custom area and power data when available, further improving the estimation accuracy.

VII. Space Explorer Design

To efficiently explore the design space of WSCs for LLMs, we propose a multi-fidelity multi-objective Bayesian optimization (MFMOBO) algorithm. By leveraging the informative nature of low-fidelity objective functions to enhance high-fidelity optimization, MFMOBO can attain faster convergence and better results, especially in the early iterations.

Algorithm 1 Pseudo-code for the MFMOBO Algorithm

Input: $\mathbb{A}$, f_0, f_1, d_0, d_1, k, N_0 and N_1;
Init the prior of f_0: $\mathbb{D}_0 \leftarrow sample(f_0, \mathbb{A}, d_0)$
Init the prior of f_1: $\mathbb{D}_1 \leftarrow sample(f_1, \mathbb{A}, d_1)$
$(\mathbb{D}, \mathbb{M}, f, \chi) \leftarrow (\mathbb{D}_1, \mathbb{M}_1, f_1, \chi_1)$
for $i \leftarrow 0$ until $N_0 + N_1 - d_0 - d_1$ **do**
  if $i = N_1 - d_1$ **do**
    $(\mathbb{D}, f, \chi) \leftarrow (\mathbb{D}_0, f_0, \chi_0)$
  if $i = N_1 - d_1 + k$ **do**
    $\mathbb{M} \leftarrow \mathbb{M}_0$
  Update $\mathbb{M}_0$, $\mathbb{M}_1$ to fit $\mathbb{D}_0$, $\mathbb{D}_1$ respectively
  Calculate the posterior $p(y|x, \mathbb{D})$ with $\mathbb{M}$
  $x_i \leftarrow argmax_x(EHVI(\mathbb{A}, p(y|x, \mathbb{D})))$
  Evaluate $x_i : y_i \leftarrow f(x_i)$
  Update the prior: $\mathbb{D} \leftarrow \mathbb{D} \cup (x_i, y_i)$
  Calculate the Pareto set: $\chi \leftarrow$ Pareto set of $\mathbb{D}$
end for
Return the current Pareto set χ.

Algo. 1 outlines the overall procedure of our proposed MF-MOBO methodology. Initially, we sample and evaluate d_0 and d_1 points to build two prior datasets $\mathbb{D}_0$ and $\mathbb{D}_1$, corresponding to the evaluation functions of f_0 (high fidelity) and f_1 (low fidelity). We then iteratively explore the design space with N_1 trials of f_1 and N_0 trials of f_0 in total. In each iteration, we first update the surrogate models ($\mathbb{M}_0$, $\mathbb{M}_1$) to fit the datasets, and then calculate the posterior distribution of the entire space. In Theseus, we jointly optimize the performance and power of WSCs on LLM workloads to find the Pareto optimal designs. Thus we use the hypervolume as the optimization indicator and iteratively select design points with the maximum Expected Hypervolume Improvement (EHVI). Specifically, during the first N_1 trials, we evaluate the points with f_1 and predict the next promising point x_i with the surrogate model $\mathbb{M}_1$. For the following k iterations from $N_1 - d_1$ to $N_1 - d_1 + k$, we switch to f_0 as the evaluation function while still predicting the EHVI of candidate design points with $\mathbb{M}_1$. This process utilizes the information of the low-fidelity surrogate model to guide the initial search of high-fidelity optimization, and the selected design points are added to dataset $\mathbb{D}_0$ for surrogate model $\mathbb{M}_0$ to update. Finally, we switch to $\mathbb{M}_0$ for EHVI calculation during the rest of the iterations.

In Theseus, we utilize the Gaussian Process (GP) as the surrogate model. For hypervolume calculations, we define the reference point with a throughput of 0 and the power as the peak power threshold of the WSC system.

TABLE I
Candidate Values for WSC Architecture Parameters

Core		**Reticle**	
dataflow	WS, IS, OS	inter_reticle_bw	0.2-2 (× Bisection BW)
mac_num	8-4096	stacking_DRAM_bw	0.25-4 (TB/s/100mm^2)
buffer_size	32-2048 (KB)	stacking_DRAM size	8-40 (GB)
buffer_bw	32-4096 (bit/cycle)	**Wafer**	
noc_bw	32-4096 (bit/cycle)	integration_style	Die Stitching / InFO-SoW
		inter_wafer_bw	100GB/s/Network Interface
		off_chip_mem_bw	160GB/s/MEM Controller

VIII. Experiment

A. Experimental Setup

Design Space Setup. To determine the architecture parameters for our proposed WSC designs, we select candidate values as listed in Table I. Considering power delivery and thermal constraints, we set a peak power threshold of 15 kW per wafer, corresponding to a power density of $32.5W/cm^2$. This threshold is derived from prior prototype implementations [40] and is established to ensure that the junction temperature remains below 105°C in dual-heatsink WSC configurations while maintaining feasible power delivery. We consider the clock frequency of 1 GHz. For SRAM, we assume a voltage of 0.9V and use the ssg process for area and power estimation. In the NoC design, routers operate at 1V and support 8 input virtual channels (vc) and 4 buffers per vc, without physically shared buffers. We consider the area overhead for inter-reticle communication as $3900\mu m^2/Gbps$ for RDL, and $1300\mu m^2/Gbps$ for offset exposure. For stacking DRAM, TSV size and pitch are set to $5\mu m$ and $15\mu m$ [44], with $1Gbps/TSV$ DRAM bandwidth. To model the trade-off between stacking memory capacity and bandwidth, we select several existing configurations and perform linear fitting. We set the area constraint of $26mm \times 33mm$ for reticles [5] and consider $12 - inch$ wafer with $215mm \times 215mm$ available area. All the area and power data are scaled to $14nm$ according to the scaling factors in [52].

For yield modeling, we set a yield requirement of 0.9, and consider the average defect density as $D_0 = 0.1/cm^2$ [4].

In the Murphy model, the defect density is assumed to follow a symmetric triangular distribution centered at the mean defect density D_0 [33]. For stress holes, the yield loss rate and the maximum influence distance are set to 0.1 and $1mm$, respectively. We set $n = 1$ to model the linear decay of stress-induced yield degradation. Given the variability in yields across different core locations, we employ Monte Carlo sampling to estimate the yield of the reticle with redundant cores. For redundancy-based yield enhancement with minimum performance and area overhead, we refer to the design of Cerebras [24] to add extra connections in each row of the core array. These additional connections can be dynamically configured to reroute data and computation when a core experiences a fault, enabling seamless and efficient core replacement. For the consideration of the KGD technique, we directly take the reticle yield as the yield for WSCs when employing inFo-SoW integration, while further calculating the yield for die-stitching WSCs with reticle yields.

LLM Benchmarks. We select a wide range of LLMs and scale the number of attention heads, hidden dimensions, and layers according to the setups in Megatron-LM [34], GPT3 [7] and Zero-Infinity [43]. We consider a fixed sequence length of 2048 and perform activation checkpoint with a granularity of 2 layers. For inference, we consider KV cache optimization and assume a constant sequence length of 2048 for both input and output, with a batch size of 32. During the DSE process, we set the total area of the WSCs to be consistent with that of the corresponding number of GPUs.

GNN Training Setup. To generate the dataset for GNN model training and validation, we randomly select a series of WSC configurations and LLM benchmarks. We generate the execution graph with Workload Compiler and collect the communication traces by evaluating the benchmark workloads with CA simulation. By meticulously tracking the transmission of packets within the NoC, we construct the transmission feature vectors, which serve as the regression targets for each sample in our dataset. Overall, our generated dataset comprises a total of 3000 samples. The GNN model is implemented and trained using PyTorch and DGL [53]. We split the dataset into training and testing sets with proportions of 90% and 10%, respectively. For the regression task, we adopt the smooth L1 loss function. The Adam optimizer [22] is employed for parameter optimization. The learning rate is initialized at 0.001 and reduced by half whenever the loss plateaus. Hyperparameters governing GNN model size should be carefully calibrated, as excessive parameters risk overfitting on limited training data, impairing generalization to unseen transmission patterns; whereas insufficient parameters degrade prediction accuracy and hinders the efficiency of DSE process. To balance these concerns, the hidden state sizes are set to $|h_v| = |h_e| = 64$ for both nodes and edges; $T = 2$ rounds of message passing are conducted, where all rounds share the same edge feature projector φ and adopt ReLU as activation function α.

B. Performance Model Verification

To ensure the reliability of our exploration results, we first validate the accuracy of our performance model. We collect publicly available performance data from both H100 clusters [6] and the Cerebras System [2]. We then adjust hardware configurations and workloads in the Theseusframework to generate corresponding performance results. For GPU clusters, we disregard the mesh-based NoC topology and instead rely on empirical utilization metrics to evaluate compute and communication performance. Table II presents the actual performance and evaluation results across different platforms and workload configurations. The results demonstrate that our evaluation maintains an error margin within 5.5% compared to publicly reported data, confirming its high accuracy. The errors may come from inaccuracies in utilization estimates, as well as overheads related to kernel launches and the software stack. Despite these factors, our performance model remains sufficiently accurate to meet the needs of early-stage DSE. Furthermore, DSE focuses more on relative relationships rather than absolute values. Therefore, in subsequent validations, we further analyze the ordinal association of the evaluation results.

TABLE II
PERFORMANCE MODEL VERIFICATION*

Platform	Task	Config	Performance	Evaluation Results	Error
H100 Cluster	Llama3-70B (Inference)	(128,128)	5837.9	5802.5	0.6%
		(128,2048)	5072	5361.2	3.7%
		(2048,128)	670.7	669.5	0.2%
		(2048,2048)	2386.7	2308.3	3.3%
Cerebras CSX System	Llama-3.3 70B (Inference)	-	2200	2080.7	5.4%
	Llama-3.3 405B (Inference)	-	969	928.4	4.2%

* Due to limited available details regarding cluster and software configurations, we make reasonable assumptions in our evaluation.

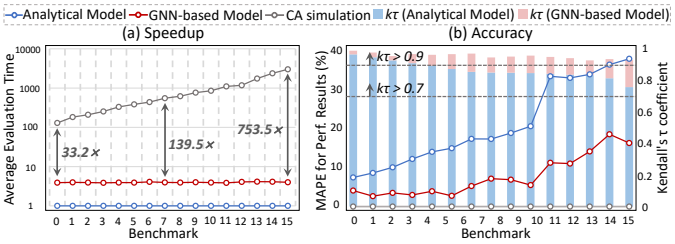


Fig. 6. Evaluation Speedup and Accuracy Comparison

Fig. 6 compares the speedup and accuracy of our evaluation models across 16 growing-scale benchmarks where CA simulation is refered to as the ground truth. As depicted in Fig. 6 (a), the evaluation time with CA simulation increases significantly with the growing scale of workloads. In contrast, both the analytical model and GNN-based evaluation model maintain stable evaluation times across different workloads. Compared to CA simulation, GNN-based evaluation methodology demonstrates a speedup ranging from 33.2× to 753.5×, with an average speedup of 220.2×. Fig. 6 (b) shows the evaluation accuracy, where CA simulation results are referred to as the ground truth. By aggregating congestion-relevant information, GNN-based evaluation outperforms the analytical model on all benchmarks, achieving an average error rate of 7.44% compared to 20.29% for the analytical model. To further validate the impact of errors from the two evaluation models on DSE results, we calculate and analyze Kendall's τ (KT) coefficient of the analytical model and GNN-based evaluation in comparison to CA simulation results. KT coefficient measures the ordinal association between the two evaluation methodologies and ground truth. In practice, a KT

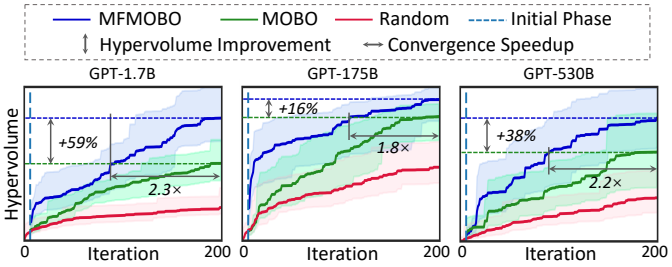


Fig. 7. Optimization Results Comparison

correlation above 0.9 is efficient for early-stage exploration of a design, while a KT correlation above 0.7 can generally benefit multi-fidelity optimization. As depicted in Fig. 6 (b), the KT coefficient of the analytical model gradually decreases from 0.9 to 0.73 with increasing model size, while for GNN-based evaluation, the KT coefficient remains consistently above 0.9. Although both the analytical model and GNN-based evaluation model deviate further from the ground truth as the workload scale increases, the KT coefficient results indicate that they can still be efficient for the DSE process.

C. Explorer Efficiency Analysis

To assess the efficiency of our proposed MFMOBO methodology, we compare MFMOBO with random search and traditional multi-objective Bayesian optimization (MOBO). We employ the GNN-based evaluation model for both random search and MOBO while applying the analytical model together with GNN in MFMOBO. Both random search and MOBO iterate 200 times, whereas MFMOBO performs 100 iterations for initialization with the low-fidelity analytical model and takes the remaining time for high-fidelity iteration. For both MOBO and MFMOBO, we select an initial set with 6 design points. Fig. 7 presents the optimization results for GPT-1.7B, GPT-175B, and GPT-530B, with similar trends observed for all benchmark LLMs. Our experiments are conducted on a server equipped with an Intel Xeon E5-2690 CPU and an NVIDIA Tesla V100 (32GB) GPU, with all experiments repeated 10 times to calculate the average hypervolume improvement over optimization iterations. The total runtime for both MFMOBO and MOBO averages about 350 minutes. During the Bayesian Optimization process, the major time overhead consists of two components: (1) BO algorithm updates, including EHVI computation and GP parameter updates. (2) Design point evaluation, using either the analytical model or GNN-based evaluation. Thanks to the efficient evaluation speedup in Theseus, design point evaluation takes only about 27 minutes, with the remaining time dedicaded to Bayesian optimization updates. This is primarily due to the $O(n^3)$ complexity of kernel parameter updates, which increases with the number of iterations. In contrast, random search completes in just 30 minutes, as it primarily rely on design point evaluations. However, despite its shorter runtime (only 10% of the time required by BO methods), random search performs significantly worse in hypervolume improvement and fails to achieve consistent improvement over iterations. Compared with vanilla MOBO, our proposed MFMOBO achieves an average 2.1× faster convergence to the same hypervolume on all LLM benchmarks, and on average 42% hypervolume improvement within the same iteration time, resulting in optimal designs that closely approximate the real Pareto frontier. By leveraging the advantages from enhanced convergence and accelerated evaluation speed, Theseus attains an overall speedup of over 400× compared with MOBO with CA simulation.

IX. Case Studies and Insights

A. Core Granularity Tradeoffs

In the design of WSCs for LLM workloads, the granularity of the core emerges as a critical factor. We define the computational power of the core (in FLOPS) as a representative measure of core granularity and explore combinations of all other parameters (i.e., SRAM capacity, NoC bandwidth, etc.) to find the optimal design in performance and energy consumption. Fig. 8 illustrates the trend of training throughput and Energy-Delay Product (EDP) concerning the computational power of cores for LLMs of different scales.

Generally, large cores tend to exhibit better area and energy efficiency than small cores, since they replace NoC-based interconnections between small cores with fixed datapaths. Besides, large cores can reduce the number of nodes in the NoC, simplifying routing complexity and alleviating NoC traffic. However, large cores face challenges related to utilization, module efficiency, and yield considerations.

Utilization. Small cores generally offer more scheduling flexibility, leading to better resource utilization. This is because within a core, the computation is operated in fixed dataflow, which relies on the parallelism of specific dimensions to fully utilize all processing units (MACs). However, in LLM workloads, the dimensions of the operators are generally large enough to effectively utilize computational resources within individual cores across various dataflows, as long as the SRAM capacity meets the requirement of data reuse. Within our candidate range of core granularity, the design of large cores does not pose utilization challenges.

Module Efficiency. Effectively designing large cores with high computational power necessitates a corresponding increase in SRAM capacity and NoC bandwidth. However, both large-capacity SRAM designs and high-bandwidth NoC routers are inefficient in area and energy consumption, which may contribute to the decrease in throughput and EDP when core computational power surpasses a certain threshold.

Yield Consideration. Larger cores exhibit lower yields and incur higher overhead due to redundant cores and extra connections. Consequently, the increase in core granularity may lead to a decline in both performance and energy efficiency.

In our experimental setup, the optimal computational power for cores falls within the range of 512G-1TFLOPS.

Takeaway 1: For LLM workloads, WSC can be designed with large cores, while considering utilization, module efficiency and yield requirements.

B. Integration Style Tradeoffs

In Fig. 8, we also compare the performance under integration styles of die stitching and Info-SoW. Compared to die stitching integration, Info-SoW introduces a larger area and power overhead for inter-reticle communication. However, since know-good-die (KGD) technology can only be applied to

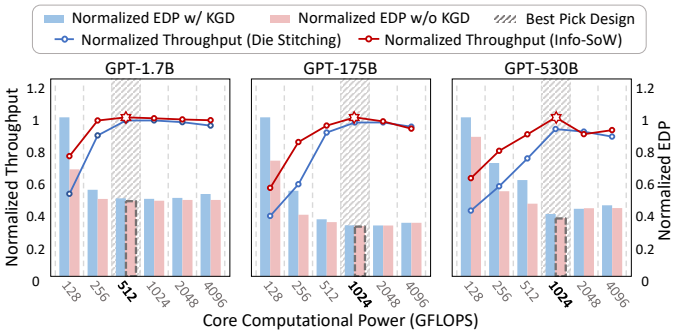


Fig. 8. Core Granularity Exploration Results

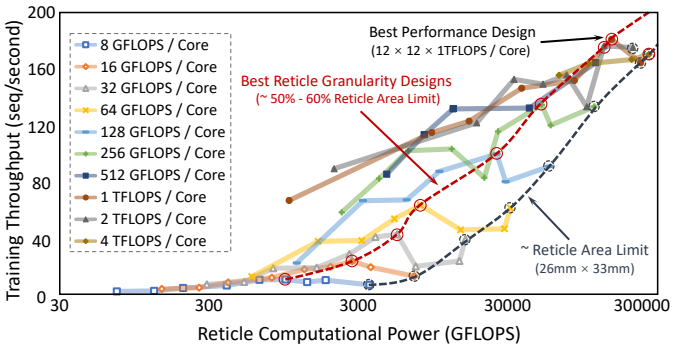


Fig. 9. Reticle Granularity Exploration Results

InFo-SoW, InFO-SoW can always provide better performance and power consumption than die stitching due to this flexibility in yield guarantee. Meanwhile, for both integration styles, WSC designs should balance between the benefits of enhanced inter-reticle bandwidth and the introduced area overhead affecting computational resources. In our experimental setup, optimal inter-reticle bandwidths generally fall between 0.5× and 1× of the bisection bandwidth of a reticle.

Takeaway 2: Info-SoW with KGD can outperform offset exposure due to less yield overhead, despite the larger area overhead for inter-reticle communication.

C. Reticle Granularity Tradeoffs

Similar to core granularity, we refer to the peak computational power of a reticle as the measure of reticle granularity. In exploring the design of reticle granularity, we generate different configurations of cores, as well as feasible core array sizes under the reticle area constraint to calculate the computational power of the reticle. Fig. 9 shows the optimal training throughput under given reticle granularities for GPT3 on WSC systems. As shown in Fig. 9, the computational power of a reticle varies from tens of GOPS to hundreds of TOPS. Overall, larger reticle granularity tends to have better performance, since inter-reticle communication introduces higher overhead in latency, area, and energy consumption compared to NoC. Meanwhile, larger reticle granularity can lead to reduced TP size, which reduces the amount of data transmission in collective communications. To further analyze the results, we cluster the designs with the same core granularity, where different reticle granularity reflects the core array size. We label the optimal reticle design under each core granularity, as well as the largest scale design within the reticle area constraint. It can be observed that the best performance designs are mostly not associated with the largest array size that approaches the area limit of reticles. As the number of cores increases, the overhead of redundant cores and extra connections also rises which can impact the achievable computational power under the same area.

In our experiment setup, the performance optimal reticle granularity for GPT3 is 144TFLOPS, with a core array size of 12 × 12 and a core granularity of 1TFLOPS. Notably, the optimal reticle granularity design typically occupies 50%-60% of the reticle area limit across different core granularities.

Takeaway 3: WSC reticle scale should trade between the redundancy ratio of cores and the overhead of inter-reticle connections, instead of going for the reticle limit.

D. Speedup Analysis for LLM Inference

Fig. 10 depicts the inference speedup of WSCs over the H100 baseline [3] with the same area. Fig. 10(a) shows the results on GPT-1.7B when all necessary data (weights, inputs, and KV-cache) are stored in the SRAM of WSCs. The x-axis illustrates the available on-chip SRAM bandwidth per WSC while meeting SRAM capacity requirements. WSC achieves an average speedup of 5.5× with multi-query attention (MQA) optimization, and 16.9× without MQA. Notably, the LLM decoding stage is memory-bound on GPUs, particularly under small batch sizes, which leads to significant under-utilization of compute resources. When the SRAM capacity of the WSC is sufficient, a large SRAM bandwidth ensures optimal utilization of computational resources and benefits the load of KV-cache. However, increasing either capacity or bandwidth results in an increase in the SRAM area, which may influence the area of computational resources. This explains the variation of speedup and further motivates our exploration.

Fig. 10(b) shows the inference speedup and latency breakdown of GPT-175B with stacking DRAM. Within the stress constraint, stacking DRAM bandwidths are varied from 0.25 to 4 TB/s/100mm^2. As for comparison, the HBM bandwidth of H100 is approximately 0.2 TB/s/100mm^2. The increased stacking DRAM bandwidth benefits both the decoding stage and KV-cache access, while achieving up to 6.8× speedup over GPU baseline with MQA and 9.8× without MQA. However, as bandwidth increases, the area of the TSV region grows, diminishing the effective computational power of the WSC and affecting the yield. This can lead to an increase in the latency of the prefilling stage, which may become the performance bottleneck. Meanwhile, the reduction in stacking memory capacity can result in increased inter-reticle communication, necessitating careful design of communication bandwidth to ensure the overall performance improvement.

Takeaway 4: The high bandwidth of both on-chip SRAM and stacking DRAM can efficiently speedup the inference for LLMs with different parameter scales, while the resulting capacity reduction can be alleviated through the efficient inter-reticle communication of WSCs.

E. Heterogenous Improvement

Table III presents the speedup of GPT-175B inference with various levels of heterogeneity. To address hardware heterogeneity, we separately optimize the performance of the prefill and decode stages during the exploration process, while

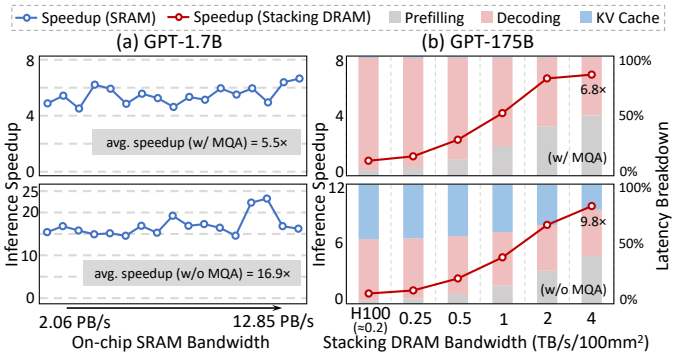


Fig. 10. LLM Inference Speedup Comparison

TABLE III
LLM INFERENCE SPEEDUP WITH HETEROGENEOUS

Stacking DRAM BW ($TB/s/100mm^2$)	Speedup w/o MQA			Speedup w/ MQA		
	Core	Reticle	Wafer	Core	Reticle	Wafer
0.25	1.44×	**1.47**×	1.09×	1.40×	**1.46**×	1.08×
0.5	**2.89**×	2.84×	2.10×	**2.84**×	2.75×	2.04×
1	**5.89**×	5.33×	3.95×	**5.84**×	4.93×	3.65×
2	8.91×	**9.51**×	7.04×	7.66×	**8.17**×	6.05×
4	12.04×	**12.84**×*	9.51×	8.72×	**9.31**×*	6.89×

* The searched best-performing configuration for the decode stage is 512GFLOPS with 32KB SRAM per core, 9×9 cores per reticle, with 0.6TB/s inter-reticle bandwidth, and 10×7 reticles per WSC.

considering the inter-stage KV-cache transfer overhead. By adjusting the resource allocation between the two stages, we can achieve the optimal overall throughput. As shown in Table III, with the same stacking memory bandwidth for the decode stage, the heterogeneous design can yield higher inference speedup compared to the exploration results in Fig. 10. This is primarily attributed to the augmentation of hardware computational power and resource utilization during the prefill stage. Due to the impact of increased stacking memory bandwidth on the effective area and yield of reticles, the computational power of reticles optimized for the prefill stage can exceed that of reticles optimized for the decode stage by over $1.6\times$ under the same area. We also compare the performance improvements resulting from different levels of heterogeneity in Table III, and highlighted the optimal heterogeneous granularity for each configuration in **bold**. Due to limited inter-wafer bandwidth, KV-cache transfer can become the bottleneck when applying wafer-level heterogeneity. For core-level heterogeneity, the flexible scheduling and resource allocation of compute-intensive and memory-intensive operators on the same reticle enables higher resource utilization. However, it also introduces an increased volume of both intra-reticle and inter-reticle transmissions, as well as overhead in compilation and control. Reticle-level heterogeneity strikes a favorable balance between these factors, although it may pose challenges to existing integration technologies.

Takeaway 5: Reticle-level heterogeneity can provide the best tradeoff for LLM inference between hardware utilization of both stages and introduced overhead.

F. DRAM Approach Analysis

In Fig. 11, we visualize the overall design space for GPT-175B training. Each point represents a sampled design configuration during our iterative exploration process, where red points correspond to designs with traditional off-chip DRAM approach and blue points correspond to those utilizing stacking DRAM. From the comparison of the Pareto frontiers, it can be observed that WSCs can benefit from stacking DRAM in both performance and power efficiency. Traditional off-chip DRAM necessitates access through the memory interfaces around the wafer, raising the transfer pressure of inter-reticle communications. Long-range DRAM-access-induced data transfer from the WSC edge can become the performance bottleneck, introducing additional power consumption, with implications on the available inter-reticle bandwidth.

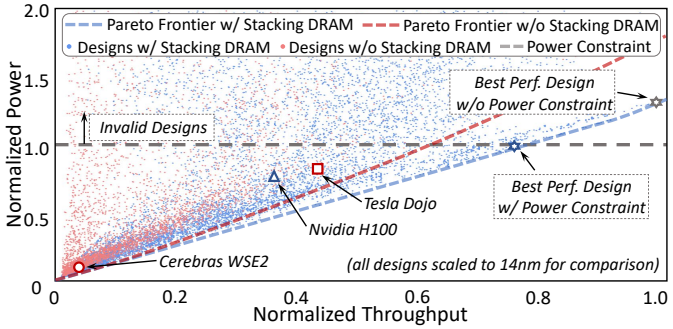


Fig. 11. Design Space for GPT-175B Training. The searched best-performing configuration is 1TFLOPS with 128KB SRAM per core, 12×12 cores per reticle, with 1.5TB/s inter-reticle bandwidth, and 9×6 reticles per WSC.

Takeaway 6: Stacking memory allows for better scaling than off-chip memory due to its higher bandwidth, improved power efficiency, and reduced communication overhead.

G. Design Space Exploration Results

We search for Pareto Optimal WSC configurations for both training and inference across various LLM benchmarks. We calculate the area of these configurations with our area model, and find that most of these configurations occupy around 50% - 70% of the $215mm \times 215mm$ wafer limit. Specifically, the heterogeneous reticles in Table III account for 55.2% (with 16.3% for decode stage reticles), while the design in Fig. 11 occupies 69.2%.

Takeaway 7: The scaling up of a single WSC does not always need to reach the wafer limit for LLM workloads.

To demonstrate the effectiveness of Theseus framework, we compare the performance and power of searched Pareto optimal WSCs with existing designs including H100 [3], Cerebras WSE2 [26] and Tesla Dojo [10]. All comparisons are made under the same area, with both area and power values for existing designs scaled to 14nm. For H100, we ignore yield requirements and the area overhead for NVLink Serdes. Table IV presents several key design points from Fig. 11, along with the system performance, total power consumption, and performance per watt for both the best-performing WSC design and existing architectures. The results indicate that our optimized WSC design achieves the highest energy efficiency. This trend is further reflected in Fig. 11, where our Pareto frontier exhibits a lower slope compared to existing designs, indicating a more favorable energy-performance trade-off. Additionally, Fig. 11 shows that without power density constraints, even higher-performance WSC designs can be explored. This further underscores the critical importance of power management in WSC design.

TABLE IV
DSE RESULTS COMPARISON

Design	Cerebras WSE2	Tesla Dojo	NVIDIA H100	Best Perf. WSC Design
Core-level	Core	Node	SM	Core
Compute(TFLOPS)	0.008	1	7.5	1
SRAM size(KB)	48	1250	256	128
NoC BW(bit/cycle)	32	1024	-	256
Reticle-level	Reticle	D1 chip	GPU	Reticle
Core Array Size	66×154	18×20	132	12×12
Inter-Reticle BW(TB/s)	1.75	2	0.9	1.5
Wafer-level	WSE	Dojo	GPU	WSC
Reticle Array Size	12×7	5×5	1×1	9×6
Performance(token/sec)	13793.21	207420.95	172216.68	362987.52
Power(kW)	35.47	448.72	414.81	545.05
Performance/Power	388.87	462.25	415.17	**665.97**

Overall, for the training tasks across our LLM benchmarks, Theseus can find Pareto optimal WSC designs that outperform the H100 cluster by an average of 62.8% in performance (with the same or lower power) and 38.6% in power (with the same or higher performance), thanks to the high bandwidth and low power inter-reticle communication, efficient stacking DRAM and effective compute utilization. Meanwhile, the results obtained by Theseus outperform Cerebras WSE2/Tesla Dojo by up to 73.7%/46.5% in performance and 42.4%/31.7% in power, respectively. In comparison with existing WSC designs, our searched Pareto optimal designs benefit from the proper selection of core granularity, inter-reticle bandwidth, and the effectiveness of stacking DRAM. For LLM inference tasks, WSC designs improve up to 23.2×/15.7× for performance and power with sufficient SRAM capacity, and up to 12.9×/11.2× with stacking DRAM.

H. Cost Analysis

Compared to traditional chips like GPUs, the manufacturing cost of WSCs poses a significant challenge. To address this, we conduct a cost analysis of WSC fabrication, taking into account key factors such as cost per wafer, yield, DRAM stacking cost, and integration cost. Based on open-source data [1], [12], [32] and insights from our industry collaborator, the cost of a 14nm 12-inch wafer is approximately $4000, while DRAM stacking and InFo-SoW integration each add an additional $5000 per wafer. Given that the H100 GPU's HBM occupies an area equivalent to a reticle, we estimate that the cost of a WSC is approximately 1.5× that of a GPU with the same area. Our DSE results indicate that WSCs achieve a 2.1× performance improvement in training tasks and a 12.8× improvement in inference tasks. Consequently, in terms of performance per cost, WSCs outperform GPUs by a factor of 1.4× for training and 8.5× for inference. Furthermore, as discussed in our previous analysis, WSCs also demonstrate better performance per watt compared to GPUs, further reinforcing their advantages in cost efficiency.

I. Sensitivity Analysis

During the DSE process, we make several yield modeling assumptions, including average defect density, yield loss rate, linear decay of stress impact, and redundancy-based yield enhancement. In this section, we conduct a sensitivity analysis on these assumptions. Since yield primarily affects trade-offs at the core granularity level, we systematically vary these parameters and re-examine the optimal performance trends across different core granularities. All experiments are performed using GPT-175B as the workload.

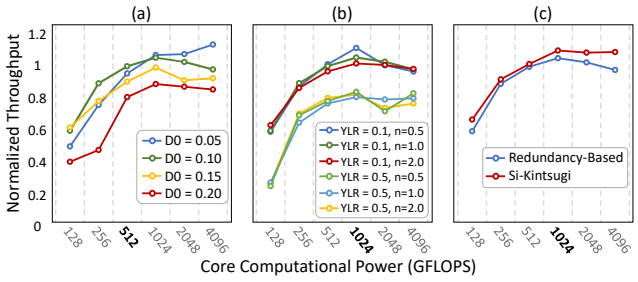


Fig. 12. Sensitivity Analysis Exploration Results

Fig. 12 presents the exploration results. In Fig. 12(a), we vary the average defect density D_0 from 0.05 to 0.2 to simulate different process technologies and maturity levels. In Fig. 12(b), we adjust the yield loss rate (YLR) and the nonlinear exponent n to explore more realistic stress impacts. The results indicate that modifying these yield-related assumptions does not alter the overall trend of the design space exploration. However, when yield is more severely affected, the increased demand for redundant resources leads to performance degradation.

In Fig. 12(c), we replace the redundancy-based yield enhancement approach with Si-Kintsugi [16], a software-centric repairing method. Under this approach, all non-defective cores are utilized for computation without redundant resources, thereby do not introduce additional area overhead. Despite this change, core-level yield still impacts the peak computational capacity of the chip, leading to similar DSE trends as redundancy-based methodologies. The key difference, however, lies in the increased complexity of on-chip communication, as defective cores must be bypassed during routing without the aid of redundant bypass links. Additionally, this method introduces higher compilation and runtime overhead, which may further affect overall system efficiency.

X. CONCLUSION

In this paper, we propose Theseus, a DSE framework designed to facilitate high-efficiency WSC design for LLM workloads. We construct a comprehensive design space for WSCs considering various design constraints. We propose hierarchical evaluation methodologies for efficient evaluation of design points and design a MFMOBO methodology to efficiently explore the WSC design space. Experimental results demonstrate that Theseus significantly enhances the efficiency of the DSE process, and the searched Pareto optimal WSC configurations can outperform existing designs. Furthermore, we analyze the exploration results and provide insights to design efficient WSCs for LLMs.

REFERENCES

[1] "Alleged prices of tsmc silicon wafers appear." [Online]. Available: https://www.techpowerup.com/272267/alleged-prices-of-tsmc-silicon-wafers-appearsa.
[2] "Cerebras inference." [Online]. Available: https://cerebras.ai/inference
[3] "Dgx h100: Ai for enterprise." [Online]. Available: https://www.nvidia.com/en-us/data-center/dgx-h100/
[4] "International roadmap for devices and systems." [Online]. Available: https://irds.ieee.org/editions/2022/irds\%E2\%84\%A2-2022-yield-enhancement
[5] "Mask / reticle." [Online]. Available: https://en.wikichip.org/wiki/mask

[6] "Tensorrt-llm performance measurements." [Online]. Available: https://nvidia.github.io/TensorRT-LLM/performance/perf-overview.html

[7] T. Brown *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[8] J. Cai *et al.*, "Inter-layer scheduling space definition and exploration for tiled accelerators," in *ISCA 2023*, 2023, pp. 1–17.

[9] ——, "Gemini: Mapping and architecture co-exploration for large-scale dnn chiplet accelerators," in *HPCA 2024*. IEEE, 2024, pp. 156–171.

[10] B. Chang, R. Kurian, D. Williams, and E. Quinnell, "Dojo: Supercompute system scaling for ml training," in *HCS 2022*. IEEE Computer Society, 2022, pp. 1–45.

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT 2019*, 2019, pp. 4171–4186.

[12] Y. Feng and K. Ma, "Chiplet actuary: A quantitative cost model and multi-chiplet architecture exploration," in *DAC 2022*, 2022, pp. 121–126.

[13] M. Gao *et al.*, "Tangram: Optimized coarse-grained dataflow for scalable nn accelerators," in *ASPLOS 2019*, 2019, pp. 807–820.

[14] J. Gilmer *et al.*, "Neural message passing for quantum chemistry," in *ICML*. PMLR, 2017, pp. 1263–1272.

[15] Y. Guo, X. Wei, J. Zhang, and G. Luo, "Purlin: A Versatile Toolkit for the Generation and Simulation of On-Chip Network," *ICCD 2022*, October 2022.

[16] E. Hanson *et al.*, "Si-kintsugi: Towards recovering golden-like performance of defective many-core spatial architectures for ai," in *MICRO 2023*, 2023, pp. 972–985.

[17] C. He *et al.*, "Waferllm: A wafer-scale llm inference system," *arXiv preprint arXiv:2502.04563*, 2025.

[18] Intel, "Intel gaudi 3 ai accelerator." [Online]. Available: https://www.intel.com/content/www/us/en/products/details/processors/ai-accelerators/gaudi3.html

[19] H. Jiang, "Intel's ponte vecchio gpu: Architecture, systems & software," in *HCS 2022*. IEEE Computer Society, 2022, pp. 1–29.

[20] N. Jiang *et al.*, "A detailed and flexible cycle-accurate network-on-chip simulator," in *ISPASS 2013*. IEEE, 2013, pp. 86–96.

[21] A. B. Kahng, B. Lin, and S. Nath, "Orion3. 0: A comprehensive noc router estimation tool," *IEEE Embedded Systems Letters*, vol. 7, no. 2, pp. 41–45, 2015.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[23] H. Kwon *et al.*, "Maestro: A data-centric approach to understand reuse, performance, and hardware cost of dnn mappings," *IEEE micro*, vol. 40, no. 3, pp. 20–29, 2020.

[24] G. Lauterbach, "The path to successful wafer-scale integration: The cerebras story," *IEEE Micro*, vol. 41, no. 6, pp. 52–57, 2021.

[25] F. Li *et al.*, "Noception: a fast ppa prediction framework for network-on-chips using graph neural network," in *DATE 2022*. IEEE, 2022, pp. 1035–1040.

[26] S. Lie, "Cerebras architecture deep dive: First look inside the hw/sw co-design for deep learning: Cerebras systems," in *HCS 2022*. IEEE Computer Society, 2022, pp. 1–34.

[27] Y. Lin, D. Z. Pan, H. Ren, and B. Khailany, "Dreamplace 2.0: Open-source gpu-accelerated global and detailed placement for large-scale vlsi designs," in *CSTIC 2020*. IEEE, 2020, pp. 1–4.

[28] Y. Lin *et al.*, "Mapping large scale finite element computing on to wafer-scale engines," in *ASP-DAC 2022*. IEEE, 2022, pp. 147–153.

[29] J. Liu *et al.*, "Partition and place finite element model on wafer-scale engine," in *DAC 2022*, 2022, pp. 631–636.

[30] G. H. Loh, Y. Xie, and B. Black, "Processor design in 3d die-stacking technologies," *Ieee Micro*, vol. 27, no. 3, pp. 31–48, 2007.

[31] L. Lu *et al.*, "Tenet: A framework for modeling tensor dataflow based on relation-centric notation," in *ISCA 2021*. IEEE, 2021, pp. 720–733.

[32] A. P. Lujan, "Cost and yield analysis of die-to-wafer hybrid bonding," in *ICEP 2022*. IEEE, 2022, pp. 129–130.

[33] B. T. Murphy, "Cost-size optima of monolithic integrated circuits," *Proceedings of the IEEE*, vol. 52, no. 12, pp. 1537–1545, 1964.

[34] D. Narayanan *et al.*, "Efficient large-scale language model training on gpu clusters using megatron-lm," in *SC'21*, 2021, pp. 1–15.

[35] NVIDIA, "Nvidia dgx b200." [Online]. Available: https://www.nvidia.com/en-us/data-center/dgx-b200/

[36] ——, "Nvidia h100 tensor core gpu." [Online]. Available: https://www.nvidia.com/en-us/data-center/h100/

[37] U. Y. Ogras, P. Bogdan, and R. Marculescu, "An analytical approach for network-on-chip performance analysis," *TCAD*, vol. 29, no. 12, pp. 2001–2013, 2010.

[38] OpenAI, "Gpt-4 technical report," 2023.

[39] S. Pal *et al.*, "Designing a 2048-chiplet, 14336-core waferscale processor," in *DAC 2021*. IEEE, 2021, pp. 1183–1188.

[40] ——, "Architecting waferscale processors-a gpu case study," in *HPCA 2019*. IEEE, 2019, pp. 250–263.

[41] A. Parashar *et al.*, "Timeloop: A systematic approach to dnn accelerator evaluation," in *ISPASS 2019*. IEEE, 2019, pp. 304–315.

[42] H. Peng *et al.*, "Chiplet cloud: Building ai supercomputers for serving large generative language models," *arXiv preprint arXiv:2307.02666*, 2023.

[43] S. Rajbhandari *et al.*, "Zero-infinity: Breaking the gpu memory wall for extreme scale deep learning," in *SC'21*, 2021, pp. 1–14.

[44] S. K. Samal *et al.*, "Monolithic 3d ic vs. tsv-based 3d ic in 14nm finfet technology," in *S3S 2016*. IEEE, 2016, pp. 1–2.

[45] A. Shah *et al.*, "{TACCL}: Guiding collective algorithm synthesis using communication sketches," in *NSDI 23*, 2023, pp. 593–612.

[46] Y. S. Shao *et al.*, "Simba: Scaling deep-learning inference with multi-chip-module-based architecture," in *MICRO 2019*, 2019, pp. 14–27.

[47] Y. S. Shao, B. Reagen, G.-Y. Wei, and D. Brooks, "Aladdin: A pre-rtl, power-performance accelerator simulator enabling large design space exploration of customized architectures," in *ISCA 2014*. IEEE, 2014, pp. 97–108.

[48] M. Shoeybi *et al.*, "Megatron-lm: Training multi-billion parameter language models using model parallelism," *arXiv preprint arXiv:1909.08053*, 2019.

[49] A. Smith and N. James, "Amd instinct™ mi200 series accelerator and node architectures." in *HCS*, 2022, pp. 1–23.

[50] Z. Tan, H. Cai, R. Dong, and K. Ma, "Nn-baton: Dnn workload orchestration and chiplet granularity exploration for multichip accelerators," in *ISCA 2021*. IEEE, 2021, pp. 1013–1026.

[51] W. J. Turner *et al.*, "Ground-referenced signaling for intra-chip and short-reach chip-to-chip interconnects," in *CICC 2018*. IEEE, 2018, pp. 1–8.

[52] O. Villa *et al.*, "Scaling the power wall: a path to exascale," in *SC'14*. IEEE, 2014, pp. 830–841.

[53] M. Wang *et al.*, "Deep graph library: A graph-centric, highly-performant package for graph neural networks," *arXiv preprint arXiv:1909.01315*, 2019.

[54] S. Zhang *et al.*, "Opt: Open pre-trained transformer language models," *arXiv preprint arXiv:2205.01068*, 2022.

[55] Z. Zhang, D. Parikh, Y. Zhang, and V. Prasanna, "Benchmarking the performance of large language models on the cerebras wafer scale engine," *arXiv preprint arXiv:2409.00287*, 2024.

Jingchen Zhu received the B.S. degree in computer science from Peking University, Beijing, China, in 2020, where he is now advancing as a Ph.D. candidate in the School of Computer Science. His current research interests include spatial architectures, reliability-aware design methodologies and design space exploration.

Chenhao Xue received the B.S. degree in computer science from Peking University, Beijing, China, in 2023. He is now a Ph.D. candidate in the School of Integrated Circuits, Peking University. His current research interests include system-technology co-optimization (STCO), design automation, and domain-specific accelerators.

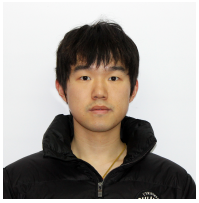
Yiqi Chen received the B.S. degree in computer science from Peking University, Beijing, China, in 2023, where he is now advancing as a Ph.D. candidate in the School of Integrated Circuits. His current research interests include Software-Hardware Codesign, CXL technology and memory system.

Zhao Wang received a B.S. degree in Computer Science from Peking University, Beijing, China, in 2021. He is currently a Ph.D. candidate in the School of Computer Science. His research focuses on on-chip networks, memory subsystems, and cache coherence protocols.

Chen Zhang (Member, IEEE) is a Tenure-Track Assistant Professor with Shanghai Jiao Tong University. Previously, he received the Ph.D. degree in EECS from Peking University in 2017 and then served as a senior researcher at Microsoft Research and a GPGPU architect at Alibaba. His research interests include computer architectures and heterogeneous computing for cloud & edge AI systems. He received FPGA 2015 Best Paper Nomination, TCAD 2019 Donald O. Pederson Best Paper Award, etc.

Yu Shen received the Ph.D. degree in computer science from Peking University, Beijing, China, in 2024. He is now a researcher in Tencent Inc. His current research interests include black-box optimization, automated machine learning, LLM inference optimization.

Yifan Chen received his B.S. degree in Computer Science from Peking University, Beijing, China, in 2023. He is currently a second-year Ph.D. student at Peking University, under the guidance of Prof. Yibo Lin. His research focuses on machine learning and optimization problems in Electronic Design Automation (EDA).



Zekang Cheng received the B.S. degree in Applied Physics from University of Science and Technology of China, Hefei, China, in 2020, where he is now advancing as a Ph.D. candidate in the School of Physics. His current research interests include chiplet architectures, low-power design methodologies, and design space exploration.

Yu Jiang received the M.E. degree of Electronic and Information Engineering from Tsinghua University, Beijing, China, in 2023. He is now working in Nvidia as an ASIC engineer in Shanghai, China. His current job focuses on system level verification.

Tianqi Wang received the Ph.D. degree in computer science from the University of Science and Technology of China (USTC), Hefei, China, in 2020. He is currently a Research Scientist at Huawei Technologies Co., Ltd., Shenzhen, China, where he is engaged in computet architecture research.His research interests include distributed systems, SoC design for distributed systems, programming models, and runtime design.

Yibo Lin received the B.S. degree in Microelectronics from Shanghai Jiaotong University in 2013. He obtained his Ph.D. degree in Electrical and Computer Engineering from the University of Texas at Austin in 2018 advised by Prof. David Z. Pan. He currently is an assistant professor in the School of Integrated Circuits at Peking University. His research interests include physical design, machine learning applications, and heterogeneous computing in VLSI CAD.

Wei Hu is a tenured associate professsor and independent PI leading the GLab at Wangxuan Institute of Computer Technology, Peking University. She obtained the B.S. degree in Electrical Engineering from University of Science and Technology of China in 2010, and the PhD degree from The Hong Kong University of Science and Technology in 2015. Her research interests include Graph Signal Processing, Graph-based Machine Learning and their applications in the processing, analysis and synthesis of geometric data and beyond.

Bin Cui is a Boya Distiguished Professor and Vice Dean in School of CS at Peking University. He obtained his B.Sc. from Xi'an Jiaotong University (Pilot Class) in 1996, and Ph.D. from National University of Singapore in 2004 respectively. From 2004 to 2006, he worked as a Research Fellow in Singapore-MIT Alliance. He is a fellow of IEEE/CCF and Cheung Kong distinguished Professor (2016). His research interests include database system, AI system and application, big data management and analytics.

Runsheng Wang is currently a Full Professor at the School of Integrated Circuits, Peking University. He also serves as the Associate Dean of the School of EECS. He received the B.S. and Ph.D. (highest honors) degrees from Peking University, Beijing, China, in 2005 and 2010, respectively. From November 2008 to August 2009, he was a Visiting Scholar with Purdue University, West Lafayette, IN, USA. His research interests include nanoscale CMOS devices and characterization, technology and circuit interaction, and new-paradigm computing.

Yun Liang is an Endowed Boya Distinguished Professor in the School of Integrated Circuit and School of EECS at Peking University. His research interest is at the hardware-software interface with work spanning electronic design automation (EDA), hardware and software co-design, and computer architecture. His recent publications investigate new algorithms, programming models, design automation tools and methodologies, and hardware for high-performance and energy-efficient computer systems.

Guangyu Sun is currently an Associate Professor in the School of Integrated Circuits at Peking University. He received his B.S. and M.S. degrees from Tsinghua University, Beijing, in 2003 and 2006, respectively, and his Ph.D. degree from the Pennsylvania State University in 2011. His research interests include design and automation for computer architecture, cross-layer co-optimization, emerging memory technologies, etc. He has published 150+ journals and refereed conference papers on ISCA, MICRO, HPCA, DAC, IEEE TCAD, etc. His work has been recognized with the DAC Under-40 Innovators Award, CCF-IEEE CS Young Computer Scientists Award, Microsoft Research Asia Collaborative Research Award, CCF-Intel Young Faculty Researcher Program, and four best paper awards. He was the general co-chair of NVMSA2021 and the TPC co-chair of NVMSA2020, RTCSA2019, APPT2017, and NAS2012. He has served as a program committee member and a track chair for over 20 conferences in these areas, including DAC, ICCAD, MICRO, HPCA, etc. He is an associate editor of ACM JETC.