Leveraging Compute-in-Memory for Efficient Generative Model Inference in TPUs

Zhantong Zhu[1,2], Hongou Li[1,2], Wenjie Ren[1], Meng Wu[1], Le Ye[1], Ru Huang[1] and Tianyu Jia[1†]
[1]School of Integrated Circuits, Peking University, Beijing, China
[2]School of EECS, Peking University, Beijing, China
[†]Corresponding Author: tianyuj@pku.edu.cn

***Abstract*—With the rapid advent of generative models, efficiently deploying these models on specialized hardware has become critical. Tensor Processing Units (TPUs) are designed to accelerate AI workloads, but their high power consumption necessitates innovations for improving efficiency. Compute-in-memory (CIM) has emerged as a promising paradigm with superior area and energy efficiency. In this work, we present a TPU architecture that integrates digital CIM to replace conventional digital systolic arrays in matrix multiply units (MXUs). We first establish a CIM-based TPU architecture model and simulator to evaluate the benefits of CIM for diverse generative model inference. Building upon the observed design insights, we further explore various CIM-based TPU architectural design choices. Up to 44.2% and 33.8% performance improvement for large language model and diffusion transformer inference, and 27.3× reduction in MXU energy consumption can be achieved with different design choices, compared to the baseline TPUv4i architecture.**

***Index Terms*—Generative model inference, Tensor processing unit (TPU), Compute-in-memory**

I. Introduction

Generative models, such as large language models (LLMs) and diffusion models (DMs), have exhibited exceptional performance in generating content across various modalities. For example, LLMs have dominated NLP tasks, powering applications like ChatGPT [1]. DMs have achieved state-of-the-art performance in image and video generation, e.g. OpenAI's DALL-E 3, Stability AI's Stable Diffusion (SD) [2], and OpenSORA [3]. The growing demand for AI generative models underscores the necessity of designing high-performance acceleration hardware for the model deployment. Currently, GPUs and Tensor Processing Units (TPUs) serve as the primary hardware platforms for AI inference and training, featuring massive parallel computing components, e.g., Tensor Cores in NVIDIA GPUs [4] and matrix multiply units (MXUs) in Google TPUs [5], [6]. However, these mainstream acceleration hardware typically consumes over 350W of TDP power, necessitating cross-stack innovations to enhance their computational efficiency.

In recent years, compute-in-memory (CIM) technique has emerged as a promising design approach, offering impressive energy efficiency and computational density. Fig. 1 illustrates the performance evolution of CIM-based designs over the past few years. Early efforts in CIM macros achieved performance levels below 500GOPS with high efficiency [7], [8]. Subsequent developments incorporated multiple CIM macros into a larger core-level design, achieving over 1TOPS performance [9]–[11]. Recently, CIM-based AI chip product have also been developed, such as a quad-core SoC from Axelera AI that delivers 52.4TOPS per core [11], demonstrating the scalability of CIM for high-performance chips. Despite these advancements, there is still a significant performance gap between CIM-based AI chips and established accelerators like GPUs or TPUs. For instance, the NVIDIA A100 GPU achieves 312TFLOPS@BF16 [4], while the Google TPUv4 delivers 275TFLOPS@BF16 [6].

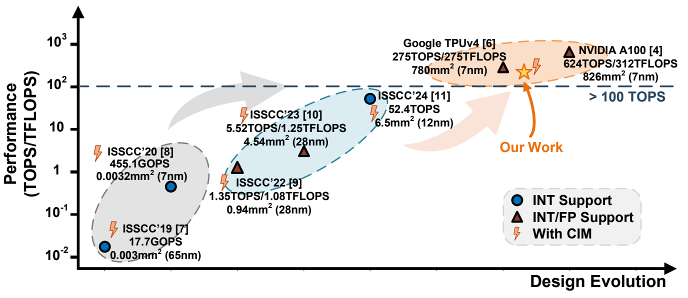


Fig. 1. Evolution of the computing performance of CIM-based designs.

It is a natural thought to utilize CIM technique to enhance the efficiency of TPUs or GPUs. In this work, we explore the design methodology and benefits for CIM-based TPUs. Two key questions are explored by this paper. *First*, what efficiency improvements can CIM bring to TPUs? *Second*, how can we redesign CIM-based TPUs to optimize performance for various generative models? To explore these questions, we begin by establishing a comprehensive architecture modeling for CIM-based TPUs, building on the TPUv4i architecture [5]. We then analyze the computational characteristics of diverse generative models, including mainstream LLMs and DMs, on our CIM-based TPU model. It is observed that significant area and energy benefits can be obtained by leveraging CIM technique for TPU.

Based on the observed design insights, we further explore the optimizations of existing TPU architecture with different design choices of CIM-based MXUs. Compared to the baseline TPUv4i, a maximum 44.2% and 33.8% performance improvement can be obtained for LLM and DM inference, respectively. Moreover, a maximum 27.3× reduction in MXU energy consumption can be obtained by leveraging CIM technique. To the best of our knowledge, this is the first architectural modeling, analysis, and design exploration of a CIM-based TPU. Our experiments and observations will provide valuable insights for designing the next generation of efficient, high-performance acceleration hardware.

The contributions of this work are summarized as follows.

- We establish an architecture model and simulator to evaluate design benefits of CIM-based TPUs.
- We analyze the inference characteristics and CIM benefits for generative models on CIM-based TPUs.
- We further explore architecture optimizations for CIM-based TPUs, with substantial improvement obtained.

II. Background

A. *Generative AI Models*

Among generative models, two prominent model architectures are LLMs and DMs, as illustrated in Fig. 2. The inference of LLMs is primarily driven by a stack of Transformer layers. Before these, a token embedding step processes the input prompt sequence. A prediction head generates the next output token after the Transformer layers. LLMs inference consists of two distinct stages: *Prefilling* (Summarization) and *Decoding* (Generation). During Prefilling stage, the model takes a prompt sequence as input and generates a Key-Value Cache (KV Cache) for each Transformer layer. In the Decoding stage, the model iteratively takes one token as input, generates the next token, and updates the KV Cache by incorporating the results from the current iteration. Notably, these two stages exhibit distinct characteristics, i.e., Prefilling is primarily compute-bound and Decoding is memory-bound [12]. Since the capabilities of LLMs are closely tied to model size [13], the unprecedented scaling of model size exacerbates the challenges of designing efficient hardware for LLM deployment.

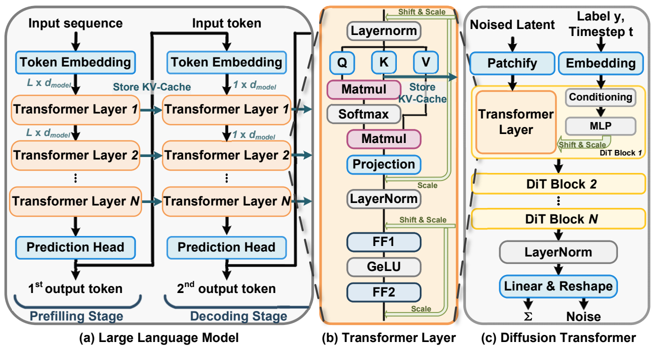
Diffusion Transformer (DiT) [14] is a new class of DMs based on the Transformer architecture instead of the U-Net architecture adopted in earlier DMs [15]. The DiT architecture has demonstrated strong performance and has been widely adopted in recent DMs, such as SD v3 [2]. As illustrated in Fig. 2 (c), the core component of DiT inference process is the sequence of DiT blocks. Each block includes a Transformer layer, augmented with conditioning, shift and scale operations.

It is clear that the core architecture of LLMs and DiTs is the Transformer architecture. We conducted inference evaluations to analyze the runtime breakdown of these generative models on NVIDIA A100 GPUs [1]. Analysis targets included Llama2-13B [16] with the Alpaca dataset [17] and DiT-XL/2 with an image resolution of 512×512, as shown in Fig. 2 (d). It is observed that the Transformer layers dominate the inference time, accounting for 98.35% in Llama2-13B and 99.31% in DiT-XL/2. By contrast, other components contribute minimally to the overall inference time. For Llama2-13B, the token embedding and prediction head account for only 0.70% and 0.95%, respectively. Similarly, in DiT-XL/2, the pre-processing (patchify and embedding) and post-processing (LayerNorm, linear & reshape) layers only account for 0.35% and 0.34% of the total inference time.

B. *Compute-in-Memory for Efficient Computing*

Compute-in-memory is a efficient computing paradigm to alleviate the efficiency challenge by fusing MAC operations into memory. Early CIM designs utilize analog-domain computation [18], performing MAC operations in voltage, charge or time domain to achieve extremely high energy efficiency. However, analog CIMs face limitations for large-scale implementation due to notable non-idealities such as process variations and low bit precision [19]. Therefore, we focus on digital SRAM-based CIM using digital-domain computation [11], which offers greater robustness and flexibility for scalable, high-performance hardware design.

[1]DiT inference was run on a single A100-PCIe-40GB GPU, while Llama2-13B inference utilized two GPUs.



(d) Inference Latency Breakdown of Generative Models

Generative Model	Layer Name	Latency (ms)	Breakdown
Llama2-13B [16]	Token Embedding	0.41	0.70%
	Transformer Layers	**57.91**	**98.35%**
	Prediction Head	0.56	0.95%
DiT-XL/2 [14]	Pre-Process	1.18	0.35%
	DiT Blocks	**338.10**	**99.31%**
	Post-Process	1.15	0.34%

Fig. 2. Generative model architecture and runtime breakdown.

A typical digital CIM macro is organized into multiple banks, each corresponding to one output channel. Within each bank, the bitcell array is further divided into sub-arrays, with each sub-array handling one input channel [7], [8]. For example, [8] presents a 7nm CIM macro with an average energy efficiency of 351TOPS/W@INT4. Beyond INT operations, recent digital CIM designs have also incorporated floating-point support for both training and inference while preserving high energy and area efficiency [9], [10], [20]. For instance, [20] showcases an SRAM-based CIM macro using a cell array with multi-precision floating-point computing units, achieving SOTA energy efficiency of 31.6TFLOPS/W and area efficiency of 2.05TFLOPS/mm^2. CIM macros have already been adopted into AI processor designs [9], [10], [21]. For instance, [9] demonstrates a digital CIM processor capable of INT8/INT16/BF16/FP32 operations through a unified FP/INT pipeline, achieving 36.5TOPS/W@INT8 and 29.2TFLOPS/W@BF16 system efficiency.

By contrast, TPUs with advanced technology nodes exhibit over 10× lower efficiency. For instance, TPUv4i delivers a peak performance of 138TFLOPS@BF16 with 175W at 7nm [5], resulting in an efficiency of 0.788TFLOPS/W@BF16. This is an order of magnitude lower than the efficiency of CIM-based designs, e.g., 15TOPS/W efficiency demonstrated by a CIM SoC [11].

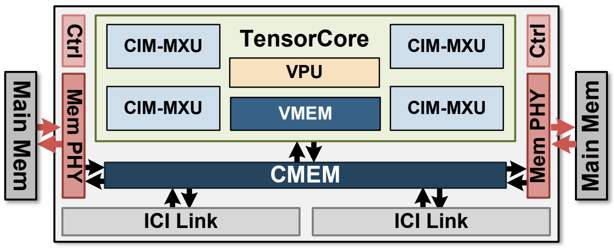


Fig. 3. Architecture modeling of CIM-based TPU.

TABLE I
ARCHITECTURE PARAMETERS FOR CIM-BASED TPU.

Key parameters	TPUv4i [5]	CIM-based TPU
Tensor Core count	1	1
MXU dimension	128×128 MACs	16×8 CIMs
CIM core dimension	N/A	128 × 256
Vector width	8 × 128	
Vector memory size	16 MB	
Common memory size	128 MB	
Main memory size	8 GB	
Main memory bandwidth	614 GB/s	
ICI link bandwidth	100 GB/s	

III. ARCHITECTURE MODELING FOR CIM-BASED TPUS

A. Architecture Overview

In this section, we introduce the architectural modeling of our CIM-based TPU, designed for generative model inference. Fig. 3 illustrates the architecture, which is based on the TPUv4i inference-only accelerator [5]. The TPUv4i performs AI computations using a TensorCore, which consists of four matrix multiply units (MXUs) for matrix multiplication, a vector processing unit (VPU) and vector memory (VMEM). The original MXUs are implemented as 128×128 systolic arrays of Multiply-Accumulate (MAC) units for parallel computation. In our model, we replace the vanilla MXU with our CIM-based MXU (CIM-MXU, details in Sec. III-B) to enhance matrix computation efficiency. The VPU remains unchanged to process other general parallel operators, such as normalization and activation functions. At the top level, the chip includes 128MB of common memory (CMEM), an on-chip interconnect (OCI), two chip-to-chip interconnect (ICI) links, and interfaces to main memory, e.g. HBMs. Both CMEM and VMEM are implemented as on-chip SRAM. We model the memory access between VMEM and CMEM using available OCI bandwidth.

To ensure accurate performance evaluation, our model leverages the prior architecture template from [22], which has been validated for accurate LLM inference, and incorporates an accurate CIM model for our CIM-based TPU. Unlike prior CIM simulator [23], our approach maintains the two-level memory hierarchy as used in TPUs. Key architectural parameters, such as CIM-MXU counts and dimensions, buffer sizes, and OCI/ICI bandwidth, are fully configurable, as outlined in Table I. With minor modifications, our architecture modeling can also be adapted to other TPU variants or GPUs.

B. Design and Modeling of CIM-MXU

CIM-MXU is designed to replace conventional MXU in TPUs to accelerate GEMM/GEMV operations. However, two challenges arise when organizing CIM macros for the high parallelism in a systolic array. First, each CIM macro has limited dimensions, making it difficult to scale up or scale out. Most CIM designs feature bespoke circuit design, meaning that modifying the data path to increase macro size (*scaling up*) is challenging. Meanwhile, the bit width of input activation and weight will increase significantly when organizing multiple CIM macros (*scaling out*) for simultaneous computation. Second, systolic matrix multiplications require frequent weight updates to maintain high hardware utilization. Furthermore, GEMM/GEMV operations in Transformers often have very low weight reuse rate, leading to frequent weight matrix updating.

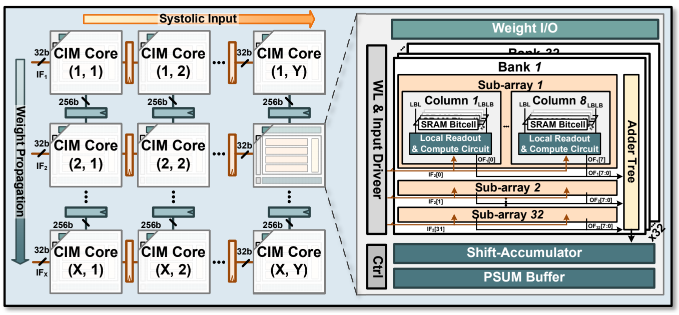


Fig. 4. Architecture and CIM design details of CIM-MXU.

In our work, we adopt a systolic data path to incorporate multiple CIM macros within the CIM-MXU, as shown in Fig. 4. At the top level, a 16×8 grid of CIM cores forms a two-dimensional systolic array, where 128 MAC operations are performed each cycle within each CIM core. The CIM computation aligns with typical weight-stationary digital CIMs [20], where the input vector broadcast to all output channels in a bit-serial manner. An output-stationary dataflow is employed in the CIM-MXU systolic array, with inputs and weights being propagated after each computation wave. In the row dimension, a 32-bit input vector is propagated systolically, moving sequentially across the CIM core columns. In the column dimension, each CIM core can perform CIM read/write operations through its dedicated weight I/O. During computation, weight matrices are propagated to the CIM core in adjacent row via interleaved SRAM read/write operations. To accommodate frequent weight updates, our CIM macro supports simultaneous computation and weight read/write operations via the weight I/O, which is similar as [24]. Such systolic data path maximizes weight and input reuse, conserving IO bandwidth and enabling the system to scale out to larger CIM core arrays.

Our CIM-MXU can perform both BF16 and INT8 operations as the original MXU in TPUv4i. In FP mode, mantissa bits of the weight matrix are loaded into CIM macros, and the input activation is processed by a pre-processing unit before the mantissa bits are transferred to the CIM array. The pre-processing unit performs exponent alignment and mantissa shifting for the INT MAC in CIM macro, and a post-processing unit handles the rest shift-and-accumulation and rounding operations. In INT mode, the pre-processing unit is bypassed, allowing direct loading of input activation into the CIM array.

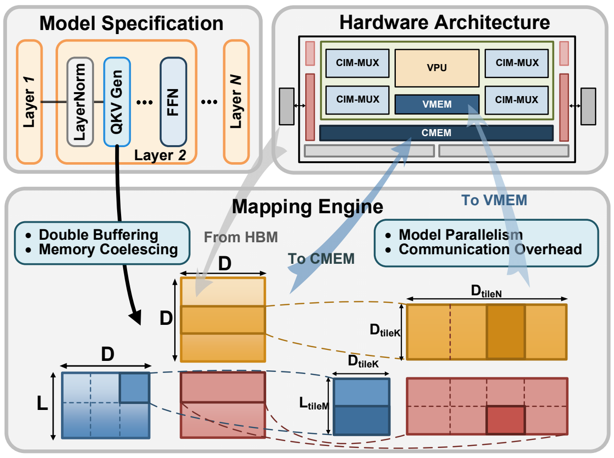


Fig. 5. Workload evaluations with a mapping engine for CIM-based TPUs.

C. *Workload Evaluations*

Mapping and scheduling. Given the computational graph and hardware configurations, a model mapping engine performs tiling and scheduling of operators onto CIM-based TPU. Fig. 5 provides a mapping example for a single layer. The model with a size of $[L, D] \times [D, D] = [L, D]$ will be partitioned into subtiles with size of $[L_{tile_M}, D_{tile_K}] \times [D_{tile_K}, D_{tile_N}]$ to fit into the on-chip CMEM. The tensors will be further partitioned to fit into the VMEM before being processed by CIM-MXUs or VPU. Since the objective space of valid mappings is very large, we prune the mapspace using heuristics similar as prior work [22], [25]. The mapping engine further explores the performance-optimal mapping to better utilize hardware resources. To overlap computation with memory access cycles, we utilize double buffering and memory coalesce technique at each level of the memory hierarchy as scheduling options.

Computation evaluation. Our CIM-based TPU supports the evaluations of a wide range of key operators in generative models, including both GEMM/GEMV and other non-linear functions. GEMM operates on three-dimensional data, in which the input matrices are tiled into smaller matrices to fit into SRAM buffers. For the baseline comparison, we use SCALE-Sim [26] to evaluate systolic arrays with a given array dimension, along with input and weight matrix sizes. We also model the computation of Softmax, LayerNorm and GeLU with similar methodology. These operators utilize vector units instead of CIM cores. We implement Softmax with algorithm [27] and approximate GeLU with *tanh*, which is the same approach used in DiTs [14]. In our performance evaluations, we exploit tensor parallelism and pipeline parallelism [28] to scale the computation capability of TPUs.

IV. Analysis of Generative Models

A. *MXU Evaluations*

To validate the CIM benefits, we first demonstrate a comparison between standalone digital MXU and a 8×16 CIM-MXU, as the parameters listed shown in Table I. We use Gemmini [29] to generate a 128×128 systolic array, which is physically implemented using Cadence Genus and Innovus to obtain post-P&R power and area. The CIM core's layout is manually drawn and the CIM-MXU is implemented using RTL for physical design. Both digital MXU and CIM-MXU are implemented using the same TSMC 22nm technology. As shown in Table II, our CIM-MXU implementation has 7.26TOPS/W and 1.31TOPS/mm^2 energy and area efficiency, which is 9.43× and 2.02× better than digital MXU while maintaining the same MACs per cycle throughput.

TABLE II
Comparison between CIM-MXU and digital MXU.

Evaluation Metrics	Digital MXU	CIM-MXU	Speedup
MACs per cycle	16384	16384	1×
Energy Efficiency	0.77TOPS/W	7.26TOPS/W	9.43×
Area Efficiency	0.648TOPS/mm^2	1.31TOPS/mm^2	2.02×

TABLE III
Configurations of evaluated generative models

Generative model	# Layers	# Heads	d_{model}
GPT3-30B	48	56	7168
DiT-XL/2	28	16	1152

B. *Model Inference Evaluations*

To further assess the effectiveness of model inference on TPUs, we select two representative generative models for evaluations, i.e., GPT-3 [30] and DiT-XL/2. We adopt the original TPUv4i architecture parameters as comparison baseline and our CIM-based TPU replaces MXUs while maintaining other hardware specifications, such as memory capacity and bandwidth. Both baseline TPU and CIM-based TPU are scaled to the same technology and frequency for fair performance and energy comparisons. The model configurations of Transformer layers in target generative models are listed in Table III. We set batch size to 8 and simulate the inference of a single Transformer layer from GPT-3, and one DiT block from DiT-XL/2 with image resolution of 512×512, using INT8 data precision. During Prefilling stage, we set the input token length to 1024. For Decoding stage, we simulate the processing of the 256th output token. Fig. 6 shows the generative model inference latency and energy consumption of MXUs for the evaluated models.

LLM Prefilling: During Prefilling stage, QKV generation, Projection, and FFN layers consist of GEMM operations with large matrix dimensions. Hence it is observed that these layers take up 84.9% of TPU inference latency, marking them as the primary computation bottleneck. In contrast, Attention layers, which consists of $Q \times K^T$, $S \times V^T$, and Softmax operations, contribute only 13.1% to overall latency. Since the systolic array in baseline MXUs has already been optimized for large GEMM operations, our CIM-MXU will be not bring inference latency improvement. However, the energy efficiency advantage of CIM-MXU leads to 9.21× less energy consumption than digital MXUs for Prefilling stage.

LLM Decoding: Since the input token length for LLM decoding is one, this significantly reduces the input dimensions of GEMM as GEMV operations, resulting in lower arithmetic intensity and memory bandwidth bottleneck. In baseline TPU

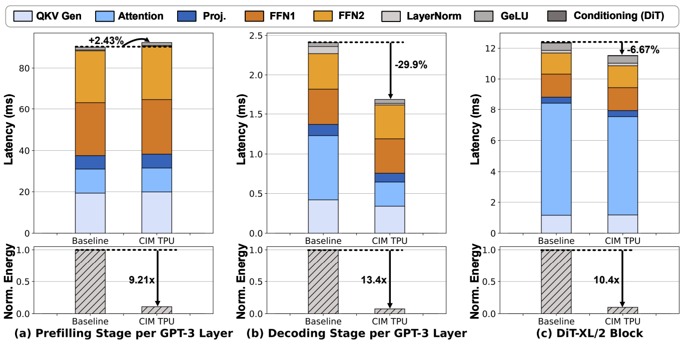


Fig. 6. Comparison between baseline and CIM-based TPU designs.

design, Attention layers account for 33.7% inference latency, primarily driven by $Q \times K^T$ and $S \times V^T$ layers. Compared to the baseline design, CIM TPU accelerates these GEMV layers by 72.7%, leading to a notable 29.9% inference latency reduction. This speedup is attributed to the CIM architecture, where the input activation vector in each CIM core is broadcast to all output channels in a bit-serial manner. Such dataflow eliminates the necessary of traversing all preceding MAC units, which is required in conventional systolic array. Benefiting from both latency and efficiency improvements, CIM-MXUs consume 13.4× less energy than digital MXU, significantly boosting the LLM decoding efficiency.

DiT Block: For a DiT block, the GEMM operations from QKV generation, Projection, and FFN consume up to 35.65% inference latency. For these GEMM computations within DiT blocks, both the digital MXU and CIM-MXU exhibits similar performance. Meanwhile, Softmax computation in Attention layers take up to 36.9% inference latency, becoming the computation bottleneck in DiT inference. It is observed that the CIM-MXU has 30.3% improvement for $Q \times K^T$ and $S \times V^T$ processing inside Attention layers due to better DiT mapping than digital MXU. Overall, the CIM-based TPU achieves a 6.67% latency and 10.4× energy reduction compared to the baseline design.

Based on the above model inference evaluations, we conclude the design observations for adopting CIM in TPUs.

CIM can significantly improve area and energy efficiency. It is clear that the CIM has notable efficiency advantages for TPUs. Our CIM-MXU contains 128 CIM cores, delivering the same peak performance as the baseline MXU with only 50% area. Meanwhile, CIM-MXU can reduce energy consumption by about one order of magnitude, significantly enhancing the energy efficiency of matrix computations.

CIM contributes differently for diverse generative models. It is observed that CIM gains the most performance benefits for GEMV-dominant layers, such as LLM Decoding. In fact, decoding consumes the most latency for LLMs, due to the large number of output tokens. Hence CIM-based TPU provides valuable performance and efficiency gains for LLM inferences. In DiT inference, the total inference time involves iteratively processing multiple DiT blocks, where the primary contributors to overall latency are Softmax and GEMM operations. Hence CIM mainly contributes to area and energy efficiency improvement rather than performance.

TABLE IV
ARCHITECTURE DESIGN CHOICES OF CIM-MXU.

Parameters	Architecture Choices		
Array dimension	8 × 8	16 × 8	16 × 16
CIM-MXU count	2	4	8

V. Architecture Exploration and Evaluation

A. *Architecture Exploration*

As our prior evaluations indicated, CIM technique provides area and energy savings compared to the baseline design. This motivates us to capitalize on these power savings and area headroom to further explore CIM-based TPU architecture for optimal design choices. We present several architecture design options in Table IV and conduct model inference analysis as before.

Fig. 7 illustrates the evaluation of GPT-3-30B and DiT-XL/2 inference latency and energy across various CIM-MXU architecture settings compared to the baseline design.

LLM Inference. Our simulation encompasses both Prefilling and Decoding stages. We set the input and output sequence lengths to 1024 and 512, respectively, to reflect typical real-world scenarios, in which Decoding dominates the latency and energy consumption of MXUs. Due to the memory-bound nature of LLM Decoding, CIM-MXUs exhibit limited performance gains when MXU counts and array dimensions continue to increase. For example, although the 8 CIM-MXU configuration with 16×16 CIM cores has 2× peak performance compared to the same number of CIM-MXUs with 16×8 CIM cores, only 2.5% performance improvement is achieved for LLM inference, at the cost of a 95% energy increase. To effectively harness the efficiency advantages of CIM technique, adopting smaller-sized CIM-MXUs can yield significant energy savings with minimal performance degradation. For example, even with only two CIM-MXUs in the TPU, featuring a smaller 8×8 CIM core array has 38% latency increase while gaining 27.3× energy savings. Considering the trade-off between latency and energy, we adopt four CIM-MXUs with 8×8 array dimension as the optimized architecture for LLM inference, denoted as *Design A*.

DiT Inference. For compute-bound DiT inference, we observe that CIM-based TPUs with more or larger CIM-MXUs achieve better inference latency by leveraging higher peak performance. Specifically, CIM-based TPUs with 4 and 8 CIM-MXUs (16×16 CIM cores) achieve 25.3% and 33.8% inference latency reduction, respectively. However, this enhanced performance also comes with increased energy cost, since more matrix computation components are added to the TPU. Fortunately, the high efficiency of CIM ensures that the CIM-MXU configuration with the highest performance, i.e., 8 CIM-MXUs each with 16×16 CIM cores, still consumes 3.56× less power compared to the vanilla systolic MXU in TPUs. When MXU performance decreases, DiT inference latency increases linearly, e.g. two CIM-MXUs with 8×8 CIM cores has a 100% higher latency than the baseline design. However, while inference runtime becomes longer, the CIM-MXU power is reduced by 20× compared to the baseline due to fewer

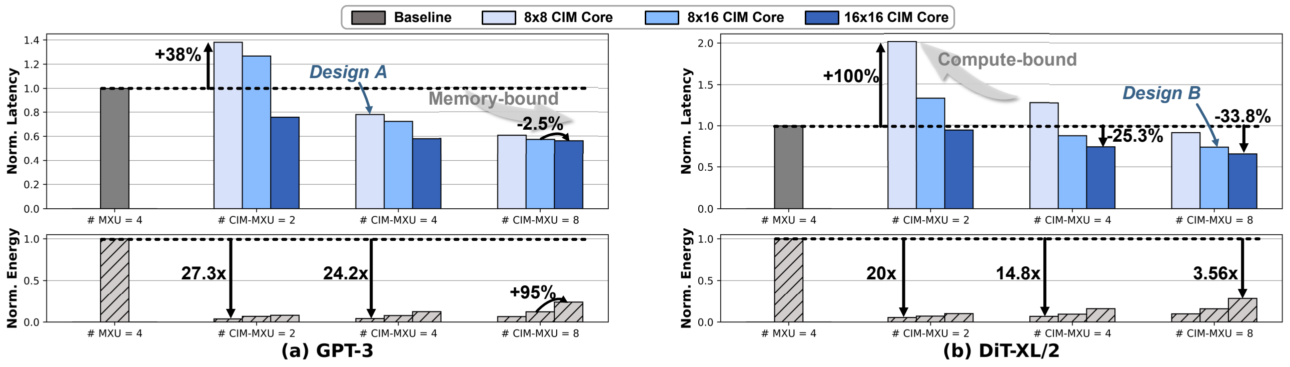


Fig. 7. Architecture exploration of different CIM-MXU configurations.

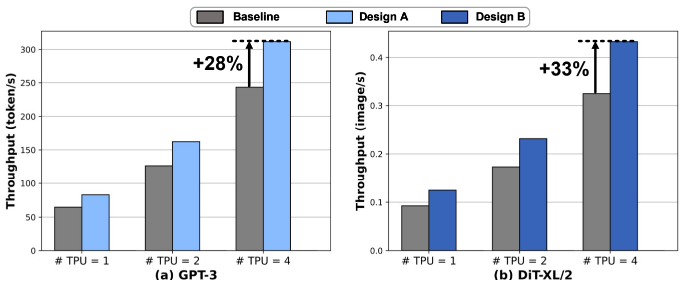


Fig. 8. Comparison between inference throughput.

CIM cores. Considering latency, energy and area trade-offs of MXUs, we derive an optimal CIM-MXU architecture for DiT inference, featuring 8 CIM-MXUs each with a 16×8 array dimension, which is denoted as *Design B*.

It is observed that none of the optimized TPU designs are ideal for all generative model inferences. Despite *Design A* having only half the peak performance of the baseline MXU, it allows for more flexible mapping strategies and a higher utilization rate, thereby improving hardware efficiency. In contrast, *Design B* is equipped with higher performance for faster DiT inference performance with lower energy cost.

B. Evaluation of Multi-Device Inference

Beyond standalone CIM-based TPU architecture explorations, we also extend our evaluation to multi-TPU inference scenarios, satisfying the need for large-scale deployment of generative models. To accommodate large batch sizes for model inferences, we scale the number of TPUs and implement up to 4-way pipeline parallelism with 4 TPUs interconnected in a ring topology to fully utilize the two ICI links on each TPU chip, as the default configuration in TPUv4i [5]. We evaluate the inference throughput of generative models for the baseline TPU and our optimized CIM TPUs, *Design A* and *Design B*.

Fig. 8 illustrates the inference throughput of GPT-3-30B and DiT-XL/2 when one, two, and four TPUs are utilized. Comparing LLM inference performance, we observe that *Design A* achieves an average 28% speedup over the baseline MXU configuration, while enjoying a remarkable 24.2× reduction in MXU energy. Due to the higher peak performance, *Design B* achieves 33% throughput improvement compared to the baseline. The CIM-MXU also exhibits 6.34× energy reduction compared to the baseline systolic array MXUs.

C. Related Work

Modeling for LLM inference. Prior work LLMCompass [22] is a hardware evaluation framework tailored to LLM inference, providing hardware architecture templates and accurate performance and area evaluations. LLMCompass explored cost-effective hardware designs, highlighting that current acceleration hardware can be over-provisioned with wasted computation components. However, LLMCompass only focused on LLMs without considering the other mainstream generative models, i.e. Diffusion Models. At hardware level, it only adopts full digital implementation of acceleration hardware and does not explore other efficient computation paradigms, such as compute-in-memory in our work.

CIM Simulators. Several CIM simulators have been proposed to evaluate the benefits of CIM for DNNs and Transformers. For example, [23] introduces a CIM modeling framework, focusing on energy modeling, DNN model mapping, and cross-stack design explorations. [31] presents a hierarchical CIM modeling structure that supports evaluating computing accuracy. However, none of these prior works explore the use of CIMs as the key computational block in high-performance accelerator chips, such as TPUs. To the best of our knowledge, our work is the first architectural modeling, analysis, and design exploration of a CIM-based TPU.

VI. Conclusion

In this work, we explore leveraging digital CIM technique in TPUs to enhance efficiency and performance for generative model inference. We refer the baseline TPUv4i architecture and construct a CIM-based TPU model. A CIM-MXU design is presented by organizing multiple CIM cores as systolic datapath to replace the vanilla digital MXU in TPU, achieving 9.43× energy and 2.02× area efficiency improvements. We evaluate the inference breakdown of mainstream generative models, including LLMs and DiTs, to analyze CIM design benefits. Furthermore, we explore architectural design choices for CIM-MXUs and present optimized TPU designs for LLMs and DiTs to enhance efficiency and performance. We also observed that the CIM design benefits can be scaled for multi-device TPUs.

Acknowledgement

This work was supported in part by NSFC Grant No. 92464202.

References

[1] OpenAI, "Introducing ChatGPT." https://openai.com/index/chatgpt/, 2022.

[2] P. Esser *et al.*, "Scaling Rectified Flow Transformers for High-Resolution Image Synthesis," Mar. 2024. arXiv:2403.03206.

[3] Z. Zheng, X. Peng, T. Yang, C. Shen, S. Li, H. Liu, Y. Zhou, T. Li, and Y. You, "Open-Sora: Democratizing Efficient Video Production for All," March 2024.

[4] J. Choquette, W. Gandhi, O. Giroux, N. Stam, and R. Krashinsky, "NVIDIA A100 Tensor Core GPU: Performance and Innovation," *IEEE Micro*, vol. 41, pp. 29–35, Mar. 2021. Conference Name: IEEE Micro.

[5] N. Jouppi, D. Yoon, M. Ashcraft, M. Gottscho, T. Jablin, G. Kurian, J. Laudon, S. Li, P. Ma, X. Ma, T. Norrie, N. Patil, S. Prasad, C. Young, Z. Zhou, and D. Patterson, "Ten Lessons From Three Generations Shaped Google's TPUv4i," in *2021 ACM/IEEE 48st Annual International Symposium on Computer Architecture (ISCA)*, 2021.

[6] N. Jouppi, G. Kurian, S. Li, P. Ma, R. Nagarajan, L. Nai, N. Patil, S. Subramanian, A. Swing, B. Towles, C. Young, X. Zhou, Z. Zhou, and D. Patterson, "TPU v4: An Optically Reconfigurable Supercomputer for Machine Learning with Hardware Support for Embeddings," in *2023 ACM/IEEE 50st Annual International Symposium on Computer Architecture (ISCA)*, 2023.

[7] X. Si, J.-J. Chen, Y.-N. Tu, W.-H. Huang, J.-H. Wang, Y.-C. Chiu, W.-C. Wei, S.-Y. Wu, X. Sun, R. Liu, S. Yu, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, Q. Li, and M.-F. Chang, "A Twin-8T SRAM Computation-In-Memory Macro for Multiple-Bit CNN-Based Machine Learning," in *2019 IEEE International Solid-State Circuits Conference - (ISSCC)*, pp. 396–398, Feb. 2019. ISSN: 2376-8606.

[8] Q. Dong, M. E. Sinangil, B. Erbagci, D. Sun, W.-S. Khwa, H.-J. Liao, Y. Wang, and J. Chang, "A 351TOPS/W and 372.4GOPS Compute-in-Memory SRAM Macro in 7nm FinFET CMOS for Machine-Learning Applications," in *2020 IEEE International Solid-State Circuits Conference - (ISSCC)*, pp. 242–244, Feb. 2020.

[9] F. Tu, Y. Wang, Z. Wu, L. Liang, Y. Ding, B. Kim, L. Liu, S. Wei, Y. Xie, and S. Yin, "A 28nm 29.2TFLOPS/W BF16 and 36.5TOPS/W INT8 Reconfigurable Digital CIM Processor with Unified FP/INT Pipeline and Bitwise In-Memory Booth Multiplication for Cloud Deep Learning Acceleration," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 65, pp. 1–3, Feb. 2022.

[10] J. Yue, C. He, Z. Wang, Z. Cong, Y. He, M. Zhou, W. Sun, X. Li, C. Dou, F. Zhang, H. Yang, Y. Liu, and M. Liu, "A 28nm 16.9-300TOPS/W Computing-in-Memory Processor Supporting Floating-Point NN Inference/Training with Intensive-CIM Sparse-Digital Architecture," in *2023 IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 1–3, Feb. 2023.

[11] P. A. Hager *et al.*, "Metis AIPU: A 12nm 15TOPS/W 209.6TOPS SoC for Cost- and Energy-Efficient Inference at the Edge," in *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 67, pp. 212–214, Feb. 2024.

[12] Z. Yuan, Y. Shang, Y. Zhou, Z. Dong, Z. Zhou, C. Xue, B. Wu, Z. Li, Q. Gu, Y. J. Lee, Y. Yan, B. Chen, G. Sun, and K. Keutzer, "LLM Inference Unveiled: Survey and Roofline Model Insights," May 2024. arXiv:2402.16363.

[13] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J.-Y. Nie, and J.-R. Wen, "A Survey of Large Language Models," Nov. 2023. arXiv:2303.18223.

[14] W. Peebles and S. Xie, "Scalable Diffusion Models with Transformers," Mar. 2023. arXiv:2212.09748.

[15] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-Resolution Image Synthesis with Latent Diffusion Models," Apr. 2021.

[16] H. Touvron *et al.*, "Llama 2: Open Foundation and Fine-Tuned Chat Models," 2023.

[17] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto, "Stanford Alpaca: An Instruction-following LLaMA model." https://github.com/tatsu-lab/stanford_alpaca, 2023.

[18] Q. Liu *et al.*, "A Fully Integrated Analog ReRAM Based 78.4TOPS/W Compute-In-Memory Chip with Fully Parallel MAC Computing," in *2020 IEEE International Solid-State Circuits Conference - (ISSCC)*, pp. 500–502, Feb. 2020.

[19] S. Yu, H. Jiang, S. Huang, X. Peng, and A. Lu, "Compute-in-Memory Chips for Deep Learning: Recent Trends and Prospects," *IEEE Circuits and Systems Magazine*, vol. 21, no. 3, pp. 31–56, 2021.

[20] A. Guo *et al.*, "A 28nm 64-kb 31.6-TFLOPS/W Digital-Domain Floating-Point-Computing-Unit and Double-Bit 6T-SRAM Computing-in-Memory Macro for Floating-Point CNNs," in *2023 IEEE International Solid- State Circuits Conference (ISSCC)*, pp. 128–130, Feb. 2023.

[21] Y. Jing *et al.*, "NeRF-Learner: A 2.79mJ/Frame NeRF-SLAM Processor with Unified Inference/Training Compute-in-Memory for Large-Scale Neural Rendering," in *50th European Solid-State Electronics Research Conference (ESSERC)*, 2024.

[22] H. Zhang, A. Ning, R. B. Prabhakar, and D. Wentzlaff, "Llmcompass: Enabling efficient hardware design for large language model inference," in *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pp. 1080–1096, 2024.

[23] T. Andrulis, J. S. Emer, and V. Sze, "Cimloop: A flexible, accurate, and fast compute-in-memory modeling tool," 2024.

[24] H. Mori *et al.*, "A 4nm 6163-tops/w/b $\mathbf{4790-TOPS/mm^2/b}$ sram based digital-computing-in-memory macro supporting bit-width flexibility and simultaneous mac and weight update," in *2023 IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 132–134, 2023.

[25] A. Parashar, P. Raina, Y. S. Shao, Y.-H. Chen, V. A. Ying, A. Mukkara, R. Venkatesan, B. Khailany, S. W. Keckler, and J. Emer, "Timeloop: A Systematic Approach to DNN Accelerator Evaluation," in *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 304–315, Mar. 2019.

[26] A. Samajdar, J. M. Joseph, Y. Zhu, P. Whatmough, M. Mattina, and T. Krishna, "A Systematic Methodology for Characterizing Scalability of DNN Accelerators using SCALE-Sim," in *2020 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 58–68, Aug. 2020.

[27] M. Milakov and N. Gimelshein, "Online normalizer calculation for softmax," 2018.

[28] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism," Mar. 2020. arXiv:1909.08053 [cs].

[29] H. Genc *et al.*, "Gemmini: Enabling Systematic Deep-Learning Architecture Evaluation via Full-Stack Integration," in *IEEE/ACM 58th Design Automation Conference (DAC)*, 2021.

[30] T. B. Brown *et al.*, "Language Models are Few-Shot Learners," *CoRR*, vol. abs/2005.14165, 2020.

[31] Z. Zhu *et al.*, "Mnsim 2.0: A behavior-level modeling tool for processing-in-memory architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 11, pp. 4112–4125, 2023.