AIG-CIM: A Scalable Chiplet Module with Tri-Gear Heterogeneous Compute-in-Memory for Diffusion Acceleration

Yiqi Jing[1], Meng Wu[1], Jiaqi Zhou[1], Yiyang Sun[1], Yufei Ma[1,2], Ru Huang[1], Le Ye[1,3*], Tianyu Jia[1*]
[1] School of Integrated Circuits, Peking University, Beijing, China, [2] Institute for Artificial Intelligence, Peking University, Beijing, China, [3] Advanced Institute of Information Technology of Peking University, Hangzhou, China
*Corresponding Author: yele@pku.edu.cn, tianyuj@pku.edu.cn

ABSTRACT

The emergence of Diffusion models has gained significant attention in the field of Artificial Intelligence Generated Content. While Diffusion demonstrates impressive image generation capability, it faces hardware deployment challenges due to its unique model architecture and computation requirement. In this paper, we present a hardware accelerator design, i.e. AIG-CIM, which incorporates tri-gear heterogeneous digital compute-in-memory to address the flexible data reuse demands in Diffusion models. Our framework offers a collaborative design methodology for large generative models from the computational circuit-level to the multi-chip-module system-level. We implemented and evaluated the AIG-CIM accelerator using TSMC 22nm technology. For several Diffusion inferences, scalable AIG-CIM chiplets achieve 21.3× latency reduction, up to 231.2× throughput improvement and three orders of magnitude energy efficiency improvement compared to RTX 3090 GPU.

KEYWORDS

Diffusion Model, Compute-in-Memory (CIM), Heterogeneous CIM

1 INTRODUCTION

Deep generative models, such as variational autoencoders (VAEs), generative adversarial networks (GANs), and the recent emerging Diffusion models [1], have consistently maintained a heightened degree of prominence within the realm of artificial intelligence generated content (AIGC). The Diffusion models have swiftly ascended to be the major model architecture and exhibit state-of-the-art (SOTA) performance. Several renowned Diffusion-based models have been released with impressive text-conditional image generation capabilities, such as DALL-E 2 and DALL-E 3 devised by OpenAI [1, 2], Stable Diffusion developed by Stability AI [3], and Midjourney's diffusion models [4].

Diffusion models are a family of probabilistic generative models that progressively destruct data by injecting noise, then learn to reverse this noise for the target sample, i.e. image generation. Fig. 1 illustrates a typical model architecture and its corresponding operations in the AIGC applications. There are three major components in the architecture, namely VAE, latent Diffusion and conditioning encoder (CE). In the pixel space, the VAE contains an encoder E and a decoder D. The encoder E performs image compression by converting information x from pixel space to latent space. In the latent space, the denoising U-Net recovers the latent representation of the picture from random Gaussian noise with the guidance of condition embedding. The decoder D finally converts the generated data from latent space back to pixel space x'.

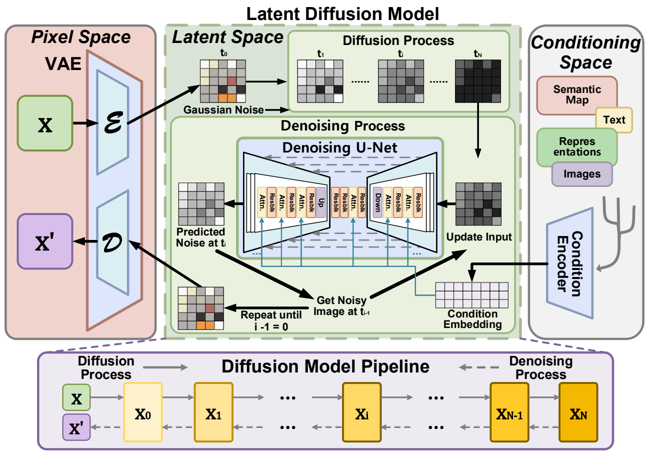

Figure 1: Model architecture of Diffusion model.

The Diffusion in latent space principally involves two interconnected processes: A predefined forward process, i.e. diffusion process, employs a sequence of forward transition kernels to progressively transform the data distribution into a simpler prior distribution, such as a Gaussian. A corresponding reciprocal process, i.e. denoising process, deploys a trained denoising U-Net to progressively undo the forward process via ordinary or stochastic differential equations (ODE/SDE) to reconstruct data. For high-quality image generation, diffusion models necessitate multiple iterative denoising steps, which form a diffusion pipeline.

Although Diffusion models demonstrate impressive capability, several hardware deployment challenges exist. A single execution of Diffusion model involves multiple (i.e. >100) diffusion and denoising iterations, each requiring significant computations for intricate model architecture. In our analysis of an image editing model [5], 100 denoising steps ultimately consume more than 10 seconds with more than 4000J average energy consumption to render a single image on a NVIDIA RTX 3090 GPU, which is infeasible from the perspective of user satisfaction.

Based on our observation, the recurring denoising procedures utilize identical model weights. Hence, it has come to our attention that employing compute-in-memory (CIM) [6, 7] with weight stationary dataflow would be a suitable approach to accelerate Diffusion. However, merely depending on a handful of CIM macros is insufficient to overcome all obstacles. As the feature resolutions and layer dimensions within the U-Net model fluctuate, corresponding

DAC '24, June 23–27, 2024, San Francisco, CA, USA
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0601-1/24/06…$15.00
https://doi.org/10.1145/3649329.3657373

Table 1: Different generation models and their parameters.

Model	Type	Param. Size	Sample Steps	Reuse Rate	FID Score
DALLE·2	Diffusion	**3.5B**	Unkown	$\mathbf{10^4}$	**10.4**
DeepFloyd IF	Diffusion	**4.3B**	100/50/75	$\mathbf{3\times10^4}$	**6.7**
Stable Diffusion	Diffusion	**2.6B**	100	$\mathbf{4\times10^4}$	**7.6**
DM-GAN	GAN	84M	-	4×10^2	32.6
NVAE	VAE	216M	-	1×10^3	33.1

weight amounts and reuse rates of all layers are also diversified. The unbalanced layer-wise computation leads to significant underutilization of CIM arrays and diminishes the benefit of hardware parallelism. As various model configurations present in Diffusions, it is essential to devise diverse dataflow and data reuse ratio macros, whereas traditional CIMs are often custom-designed with predetermined sizes and restricted dataflow [8].

In this paper, we demonstrate a hardware accelerator design, i.e. AIG-CIM, with tri-gear heterogeneous digital CIM for flexible data reuse demand in Diffusion models. To the best of our knowledge, this is the first CIM-based Diffusion accelerator with significant performance and efficiency improvements. To deal with the large parameter numbers, we adopt tile-based CIM SoC design and multi-chip-module (MCM) chiplets to scale the performance and capacity for large models. A flexible CIM cluster with tri-gear storage and compute density ratios is presented and optimized for diverse computation intensity in different U-Net layers. The entire design methodology takes circuit, architecture, mapping and scheduling design space exploration into account to explore the design trade-offs. We implemented the AIG-CIM accelerator design using the TSMC 22nm technology and evaluated the chiplet performance based on an in-house simulator. With our techniques, chiplet module with 8×8 AIG-CIM chips achieve an average of 21.3× latency reduction, 231.2× throughput improvement and significant energy saving compared to a Nvidia RTX3090 GPU.

The contributions of this work are summarized as follows:

- The model architecture and computation bottlenecks for recent Diffusion models are analyzed in-depth. It is observed the denoising task suits weight stationary CIM well.
- Multi-gear CIM cluster is developed for the diverse workload requirements in Diffusion U-Net. A collaborative DSE framework is built to explore cross-level architecture parameters.
- Multi-chip AIG-CIM chiplet module is built and evaluated to scale the Diffusion performance. Careful workload tiling is conducted to achieve good utilization and performance.
- The AIG-CIM accelerator is implemented and evaluated using a TSMC 22nm, achieving 21.3× latency reduction, 231.2× throughput gain and significant energy saving.

2 DIFFUSION BACKGROUND AND ANALYSIS

2.1 Model Comparison and the U-Net Structure

As shown in Table 2, several Diffusion models are evaluated and compared with GAN and VAE from different aspects. The sample steps indicate the execution iterations for the denoising process. The reuse rate indicates the average reused times of weight parameters during the model execution, and the frechet inception distance (FID) score represents the average distance between the generated image features and the genuine counterparts.

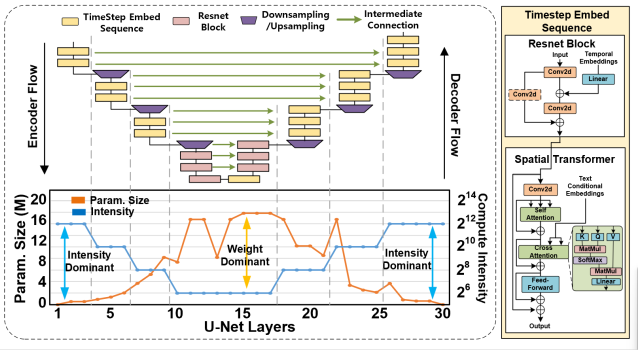

Figure 2: The analysis of U-Net denoising in Stable Diffusion.

Diffusion models exhibit significantly higher fidelity than VAE and GAN models. However, typical Diffusion-based models have billions of parameters due to their complex U-Net-based model structure. Compared to other models, the hardware requirements are orders of magnitude higher. However, the unique progressive denoising characteristic in Diffusion models enables a significant weight reuse opportunity for special hardware optimizations, e.g. leveraging weight stationary CIM technique.

Although there are extensive Diffusion algorithm developments, the U-Net structure remains the most widely used backbone for the denoising process. Due to the progressive denoising feature of Diffusion, the U-Net backbone dominates the weight and computation in AIGC models. Hence, we analyze the U-Net details from an open-source Stable Diffusion-based model [5], as shown in Fig. 2. The U-Net architecture consists of an encoder and a decoder. The encoder gradually reduces the size of the feature maps while increasing the channels. This helps in extracting high-level features and capturing contextual information from images. The decoder part mirrors the encoder by using interpolation upsampling to gradually restore the feature maps to the original image size and dimension. Additionally, the decoder utilizes skip connections that concatenate the corresponding feature maps from the encoder to preserve fine-grained details and positional information.

2.2 Model Computation Analysis

We further analyze the same Stable Diffusion-based model [5] to examine the computation breakdown and bottleneck. We ran the model on a Nvidia RTX 3090 GPU. It is observed that an inference for one image generation will take 13.9s. Within the computation, the VAE, Diffusion, and CE in Fig. 1 takes 0.08s, 13.78s, and 0.04s, respectively. Hence the Diffusion is clearly noticed as the latency bottleneck, due to the large number of U-Net denoising iterations, i.e. >100, in the latent Diffusion. Each U-Net denoising computation consumes 0.14s on this GPU.

Given the significant number of repetitive U-Net denoising processes, any acceleration for U-Net denoising will notably benefit the overall latency. Hence, we in-depth analyze the layer-wise computation conditions for a 30-layer denoising U-Net in Stable Diffusion, as shown in Fig. 2. The orange curve shows that the parameter amounts of the layers near the U-Net bottom increase significantly due to the increased dimension and dominate the entire model. However, in terms of the computational intensity, which is defined

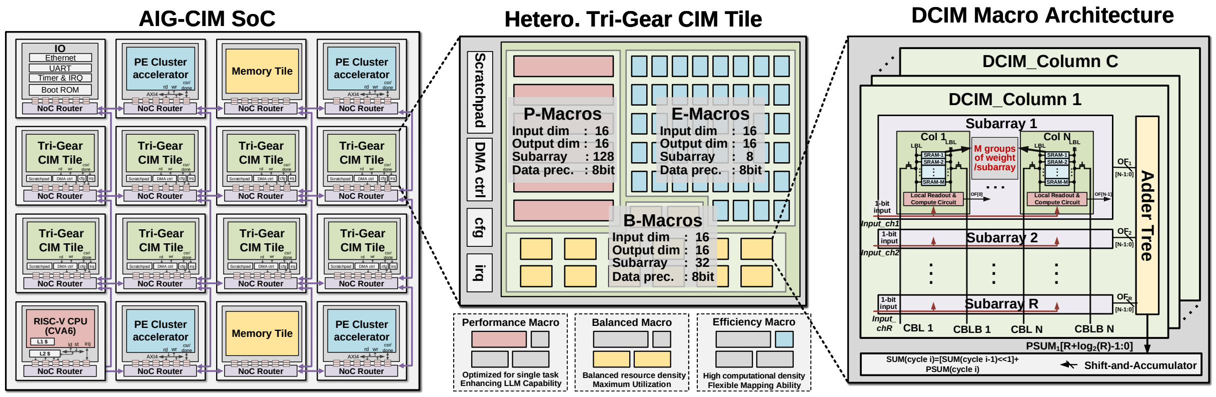

Figure 3: AIG-CIM architecture from digital CIM macro to SoC.

by the weight reuse rate [9], these heavy-weight layers are not inclined to highly reuse data. As shown by the blue curve, the computational intensity of all layers varies greatly according to the resolution and obviously, high computational intensity happens at the layers on both sides of the U-Net.

Such unbalanced computational intensity and parameter capacity in the denoising U-Net presents significant computation and storage challenges for conventional accelerator design. In response to the above problems, we present our accelerator solution, AIG-CIM. For the highly repetitive denoising processing, we leverage CIM to avoid the latency and energy overhead caused by frequent weight movement. For the highly varying computational intensity, we develop heterogeneous multi-gear CIM to improve hardware utilization and throughput effectively. For the high computation performance requirement, we scale AIG-CIM via chiplet integration to enhance capacity and computational support for large models.

3 DIFFUSION ACCELERATOR: AIG-CIM

3.1 Overall Architecture

In this work, we present a hardware accelerator leveraging digital CIM technique for Diffusion models, as shown in Fig. 3. AIG-CIM is an accelerator implemented as a tile-based SoC, in which the multi-gear CIM tiles support Diffusion acceleration. An open-source agile SoC prototyping platform ESP [10] is leveraged to assist the SoC integration. To support a high degree of scalability, the tiles are connected by a multiplane network-on-chip (NoC), which utilizes a packet-switched 2D-mesh topology with look-ahead routing.

The tri-gear CIM tile in AIG-CIM consists of three heterogeneous groups of CIM macros, namely performance macro (P-Macro) balanced macro (B-Macro) and efficiency macro (E-Macro), each with the same number of macros. The storage and computation ratios of these three macros are set as 128, 32, and 8, respectively, by setting different subarray dimensions. The heterogeneous CIM combination is selected to extend the optimal operation range for diversified data reuse rates. The other operators, such as non-static-weight matrix-vector-multiplication (MVM) in self-attention and cross-attention operators are processed by other dedicated digital processing element (PE) tiles.

Each digital CIM macro comprises a CIM array based on 6T SRAM, a wordline decoder and drivers, write-and-read circuits, and a controller. One array consists of C columns, where each column consists of R rows of subarrays, i.e. MxN 6T SRAM bitcells, and an adder tree for conducting vector-vector-multiplication (VVM) operations. Each subarray stores a total of M weights with N bit precision. The macro can store M weight matrices and perform K-bit input and N-bit weight MVM operations. The heterogeneous CIMs are obtained by using different subarray dimension M.

3.2 Heterogeneous Digital CIM Design

Fig. 3 right illustrates the tri-gear heterogeneous CIM cluster. The P-macros are designed with a dimension of 1028μm×220μm at 22nm and optimized for layers near the bottom of the denoising U-Net with large parameter amounts by enhanced storage capacity, i.e. larger subarray dimensions. The E-Macros are designed with a dimension of 174μm×220μm but relatively higher computational resource density for the layers with a high weight reuse rate on both sides of the U-Net. The B-Macros balance the computation and storage resource ratio to maximize the utilization. As shown in Fig. 4, the hardware characteristics are represented using rectangles, where the width represents storage density, i.e. storage capacity per unit macro area, and the height represents computational density, i.e. throughput per unit macro area. For example, P-Macro is designed for higher storage density, while E-Macro is designed towards higher computational density. For computation tasks within the model, we also use rectangles to represent their mapping and computation characteristics.

For the spatial utilization of hardware mapping, the large differences in weight amount and computational intensity among U-Net layers present challenges for homogeneous CIM mapping. As shown in Fig. 4, the mapping of $L1$ and $L7$ on both sides of the U-Net leads to wastage of storage resources and substantial computational time, i.e. quantity in the vertical direction. Similarly, the mapping of $L4$ on the bottom of the U-Net wastes computational resources and needs a large amount of hardware to meet the weight storage requirement. Heterogeneous CIM design provides effective improvements in hardware utilization by scheduling hardware based on the computational and storage requirements of each layer, e.g. E-Macros for $L1$ and $L7$ and P-Macros for $L4$.

The benefits of using heterogeneous CIM are also reflected in the temporal utilization aspect. When using homogeneous CIMs, the varying computation times of each layer and their data dependency

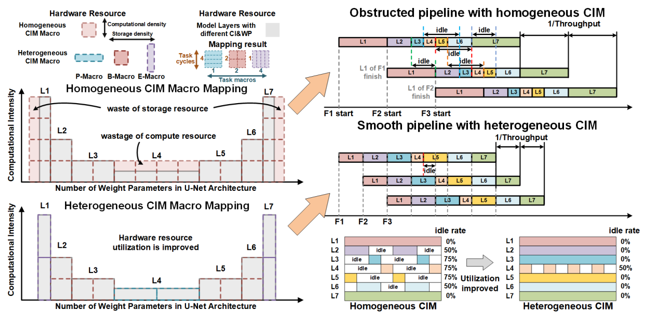

Figure 4: Heterogeneous CIM benefit in the spatial and temporal utilization.

cause long idle periods, leading to decreased temporal utilization. As shown in Fig. 4, $L3$, $L4$ and $L5$ are computed only once every four cycles, resulting in 75% wastage in the temporal dimension, bounding the overall throughput by the most extended latency layer. However, by using heterogeneous CIMs, the computational intensity of each layer is considered during mapping to balance the latency among layers and improve the temporal utilization. As shown in the example, the average temporal utilization and throughput can be doubled due to the use of heterogeneous CIMs.

By utilizing multi-gear CIM hardware, we can ensure both overall pipeline requirements for maximum throughput and optimal spatial-temporal utilization by matching the storage and computation density to model characteristics.

3.3 DSE Framework of Multi-Gear DCIMs

We established a design space exploration (DSE) framework for AIG-CIM to select optimal multi-gear CIM combinations, as shown in Fig. 5. Based on software model and definition of hardware architecture, the framework generates and evaluates architectural design points with different parameter settings. The inputs of our DSE framework include architectural definitions across three levels. The circuit-level CIM parameters include the number of input and output channels as well as data precision. The tile-level parameters include the gears of heterogeneous CIM macros, the storage and computing ratios for different types of macros, the number ratio of different types of macros, and the size of the CIM tile. The parameters at the chiplet-level consist of the number of different type tiles and the task mapping schemes.

Our DSE framework consists of four engines: the tile generator, the partition and mapping engine, the chiplet mapping and scheduling engine, and the circuit-level estimator. The DSE flow of AIG-CIM design parameters is conducted as follows. First, the simulator receives the architectural inputs and generates a corresponding CIM tile with multi-gear CIM macros by the tile hardware generator. Following that, the partition and mapping engine is leveraged to partition the model into subtasks and map them onto the underlying hardware resources based on the generated CIM tile. The output is the specific gear macros for each subtask in the model, determined through an optimal hardware selection strategy. We perform an initial workload scheduling, followed by generating data transfer traces based on data dependencies for NoC simulation.

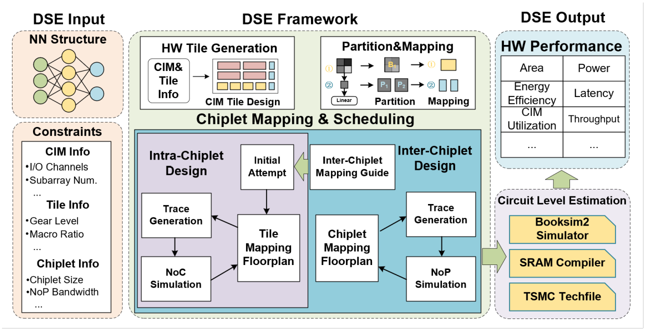

Figure 5: DSE flow for AIG-CIM architecture exploration.

We customized a cycle-accurate NoC simulator Booksim 2 [11] as an in-house evaluator for the NoC performance. The generated trace file is simulated using Booksim2 to obtain latency and power information. Based on the information, we use a simulated annealing approach to iteratively optimize the workload partition. For the scheduling at chiplet-level, we adjust the mapping scheme based on additional features introduced by chiplet-level characteristics, such as constraints on hardware spatial utilization. The simulated annealing is performed using an objective function that incorporates power and latency from NoC and NoP. After completing the above DSE, hardware performance is evaluated at the circuit level.

4 SCALE UP AIG-CIM BY CHIPLETS

To accommodate the performance requirement for Diffusion applications, we scale single AIG-CIM as chiplet to the multi-chip module. To evaluate the performance and efficiency, we adopt the commonly used 2.5-D integration as the chiplet integration technology. Here we refer to Simba [12] and build a Simba-like chiplet architecture model for analysis and evaluation. Various microarchitecture parameters, such as NoP bandwidth and performance, are expanded accordingly. In addition, we developed our workload tiling strategy for AIG-CIMs to achieve good utilization and performance.

Due to the adoption of heterogeneous multi-gear CIM, we consider the hardware mapping space as a set of gear units, each with the same number of chiplets and tiles, as well as topology. To maximize spatial and temporal utilization in parallel, the layers with different computational and storage requirements are mapped to the corresponding gear macro hardware. Fig. 6 illustrates the mapping and scheduling of deploying a U-Net denoising to a 4×4 AIG-CIM chiplets. Due to the use of heterogeneous CIM tiles, each physical tile contains subtasks with different computational intensities from different layers. For instance, the chiplet in the lower-left corner simultaneously contains tasks 1c, 2d and 4b with different computational intensities, which are respectively mapped to the E-Macros, B-Macros, and P-Macros within its internal tiles.

To prevent bandwidth bottleneck for a specific tile, we consider not only power and latency during the iterative scheduling, but also the impact of critical NoC traffic. Based on the power, latency and maximum latency information generated by Booksim 2 NoC simulator, the communication overhead in current tiling is calculated. Then the neighboring solution is generated by randomly swapping the positions of two subtasks in different tiles. During the iteration

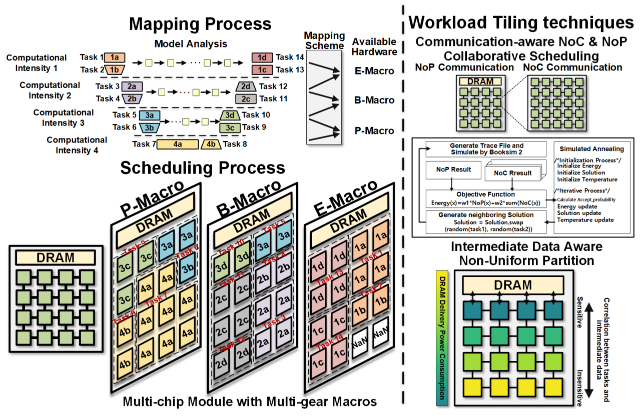
Mapping Process
Model Analysis
Mapping Scheme
Available Hardware
E-Macro
B-Macro
P-Macro
Scheduling Process
DRAM
P-Macro
B-Macro
E-Macro
Multi-chip Module with Multi-gear Macros
Workload Tiling techniques
Communication-aware NoC & NoP Collaborative Scheduling
NoP Communication
NoC Communication
Simulated Annealing
Objective Function
Intermediate Data Aware Non-Uniform Partition
DRAM Delivery Power Consumption
Sensitive
Insensitive
Correlation between tasks and intermediate data

Figure 6: Mapping and scheduling for AIG-CIM chiplets.

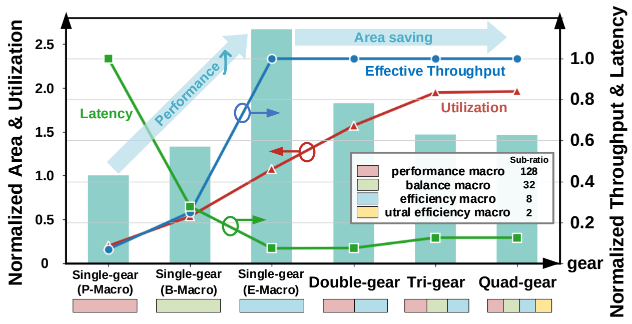

Figure 7: Selection of multi-gear CIMs for Diffusion.

process, the new neighboring solutions are continuously generated and the corresponding communication-aware objective function, i.e. the cost of NoC and NoP, is computed. The scheduling solution is continuously optimized based on the objective function score until the annealing temperature not decreased.

It is worth mentioning that U-Nets use intermediate residual connections between corresponding blocks to maintain the detailed feature information. The varying amounts of intermediate data generation also guide our task scheduling to optimize the data movements between AIG-CIM chiplets and DRAM. Scheduling layers that generate or receive more intermediate results closer to DRAM can relax data transfer pressure. Hence, a non-uniform partition was generated based on the intermediate data volume and the distance to the DRAM.

5 EXPLORATIONS AND EVALUATIONS

5.1 Explorations for AIG-CIM

In this work, we select Stable Diffusion model as our acceleration target. To design a well-optimized hardware accelerator, we first perform design space exploration of AIG-CIM based on our in-house DSE framework and simulators. We evaluate the performance of different CIM accelerations for several mainstream Diffusion models, including Pix2Pix [5] and Stable-Diffusion v2 [3]. The design points are evaluated based on the cost and comprehensive hardware metric, which is defined as EQoS by multiple energy and quality of service (QoS), i.e. delay times reciprocal of throughput.

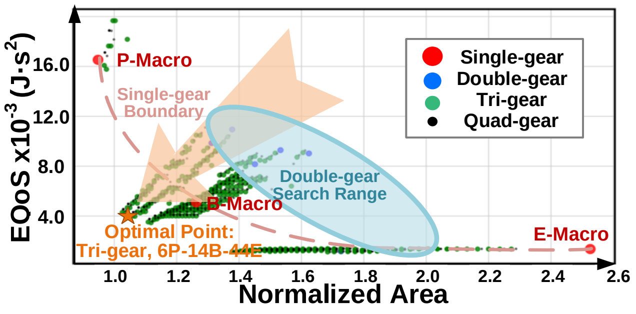

Figure 8: Design space exploration for multi-gear macros.

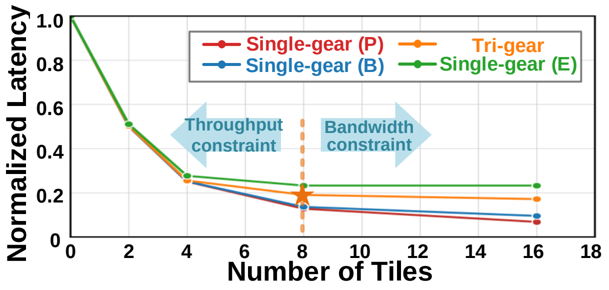

Figure 9: Normalized latency with different macros.

In order to obtain balanced NoC traffic distance and performance, we fix the tile dimensions and fill in the desired CIM macros. As our DSE results shown in Fig. 7, single-gear homogeneous CIM poses challenges in effectively balancing performance and area cost. In the case of single gear design, adopting P-Macro and E-Macro leads to insufficient QoS and excessive area respectively. Conversely, adopting B-Macro achieves a more effective trade-off between area and performance. For multi-gear designs, better flexibility and throughput can be obtained than single-gear designs, e.g. the more gears, the lower area cost due to increasing utilization. For CIM design beyond three gears, the benefits have already reached saturation. Therefore, the tri-gear CIM design is selected.

Fig. 8 demonstrates partial design points of exploring the quantity of multi-gear macros in one tile. The design is explored to be closer to the origin, i.e. better EQoS and lower area cost. It is observed that the trade-off boundary of performance and area cost of homogeneous CIM can be broken through by heterogeneous multi-gear design. Beyond three gears, the distance to the origin is saturated. The final selected design point is a Tri-gear CIM tile with 6 P-Macros, 14 B-Macros and 44 E-Macros. Considering the length of the layout wordline and bitline in the physical implementation, we chose macros with 16 input and 32 output channels.

The accelerated latency with different CIM macros and tile numbers are also evaluated. As shown in Fig. 9, the throughput of the CIM macro is highly related to the macro's computation density. When using high throughput macros such as E-Macro, the latency shows a linear decrease within a certain range of tiles due to the improved compute throughput. The performance starts to flatten out beyond eight tiles due to the NoC bandwidth saturation.

Table 2: The simulated results of E/B/P-macros at 22nm.

Bitcell Type	6T (logic rule)
Bitcell Area	0.445µm×0.804µm
Supply Voltage	0.6-0.8V
Frequency	500MHz
Input,Weight,Output Prec.	1b per cycle(bit-serial extension) 8b 12b
Input Channel	16
Output Channel	32
Throughput	64GOPS
Energy efficiency	14-27TOPS/W

	E-Macro	B-Macro	P-Macro
Macro Size(Kb)	32	128	512
Macro Area(mm^2)	0.038	0.076	0.226
SRAM Density(Mb/mm^2)	**0.822**	**1.645**	**2.212**
Throughput Density(TOPS/mm^2)	**1.684**	**0.842**	**0.283**

5.2 Implementation and Benefits

After the design exploration, we select optimal architectural design parameters and implement them using TSMC 22nm technology. The circuit design and layout of the CIM macros are first implemented and evaluated using Cadence Virtuoso, as results summarized in Table 2. The E/B/P-Macros are implemented with the same number of input channels, output channels and weight bit width, but different storage-computation ratios, i.e. 8, 32 and 128 respectively. The simulated macro efficiency with post-layout extraction is 14-27 TOPS/W with 12.5-50% input toggle rate and weight sparsity at 0.6-0.8V supply voltage.

AIG-CIM SoC was integrated using the open-source ESP tool [10], in which the tri-gear CIM tile includes E/B/P-Macros. AIG-CIM is finally completed using Cadence Innovous, as the layout view shown in Fig. 10. With TSMC 22nm, a single AIG-CIM consumes 103.7 mm^2 area, and 1.51W power at 500MHz. At the chiplet-level, we evaluated the performance by simulated per-chip results and an in-house BookSim NoC simulator. The energy consumption of inter-chip interconnection is also considered based on scaling prior 2.5D accelerator design [12], which is the same approach as [13].

We evaluate the AIG-CIM chiplet module for Diffusion models and compare it with NVIDIA RTX 3090 GPU in terms of latency, throughput, and power consumption. Since majority Diffusions use the same U-Net backbone, we select the representative Stable Diffusion v2 and Pix2Pix models for evaluations. As shown in Fig. 11, AIG-CIM achieves an average 21.3× latency reduction in Stable Diffusion v2, i.e. Diffusion inference for one image generation only consumes 1.14s. In addition, 231.2× throughput improvement and 70% power reduction are obtained by leveraging the AIG-CIM chiplet module. All the above improvements lead to an energy efficiency improvement by over three orders of magnitude.

6 CONCULSION

The Diffusion model is a promising method for AIGC. This paper presents a chiplet-based accelerator design, i.e. AIG-CIM, which incorporates tri-gear heterogeneous digital compute-in-memory to address the flexible data reuse demands in Diffusion models. Our framework offers a collaborative design methodology for large generative models from the computational circuit level to the chiplet system level. We implemented and evaluated the AIG-CIM accelerator using TSMC 22nm technology. The result demonstrates 21.3× latency reduction, up to 231.2× throughput improvement and three orders of magnitude energy efficiency improvement compared to the RTX 3090 GPU.

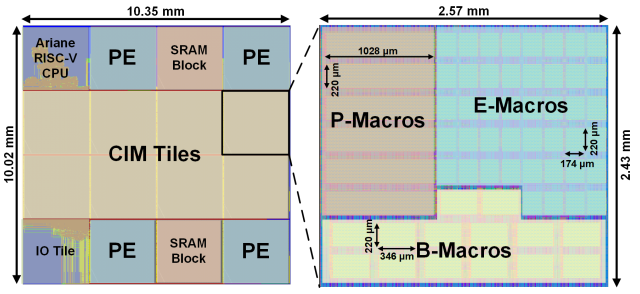

Figure 10: AIG-CIM SoC implementation and the tile layout.

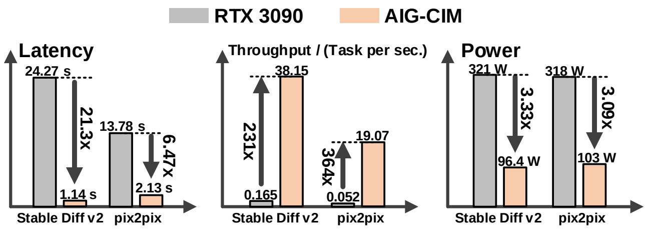

Figure 11: Comparison results with RTX 3090 GPU.

ACKNOWLEDGEMENT

This work was supported in part by NSFC Grant No. U23A6007, 92164301, 62225401.

REFERENCES

[1] A. Ramesh *et al.*, “Hierarchical text-conditional image generation with clip latents,” *arXiv preprint arXiv:2204.06125*, 2022.
[2] Z. Shi *et al.*, “Improving image generation with better captions,” *arXiv preprint arXiv:2006.11807*, 2020.
[3] R. Rombach *et al.*, “High-resolution image synthesis with latent diffusion models,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
[4] L. Aničin *et al.*, “Bias analysis in stable diffusion and midjourney models,” in *International Conference on Intelligent Systems and Machine Learning*, 2022.
[5] T. Brooks *et al.*, “Instructpix2pix: Learning to follow image editing instructions,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
[6] Y.-D. Chih *et al.*, “16.4 an 89tops/w and 16.3tops/mm2 all-digital sram-based full-precision compute-in memory macro in 22nm for machine-learning edge applications,” in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2021.
[7] 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, 2023.
[8] G. Krishnan *et al.*, “Siam: Chiplet-based scalable in-memory acceleration with mesh for deep neural networks,” *ACM Trans. Embed. Comput. Syst.*, vol. 20, 2021.
[9] Y.-H. Chen *et al.*, “Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks,” *IEEE Journal of Solid-State Circuits*, 2017.
[10] P. Mantovani *et al.*, “Agile soc development with open ESP.,” in *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, Nov. 2020.
[11] N. Jiang *et al.*, “A detailed and flexible cycle-accurate network-on-chip simulator,” in *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 86–96, 2013.
[12] Y. S. Shao *et al.*, “Simba: Scaling deep-learning inference with multi-chip-module-based architecture,” in *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, p. 14–27, 2019.
[13] Z. Tan *et al.*, “NN-baton: Dnn workload orchestration and chiplet granularity exploration for multichip accelerators,” in *ACM/IEEE International Symposium on Computer Architecture (ISCA)*, pp. 1013–1026, 2021.